A learning-based projection method for model order reduction of transport problems[☆]

Zhichao Peng [a], Min Wang [b,*], Fengyan Li [c]

[a] *Department of Mathematics, Michigan State University, East Lansing, MI 48824, USA*
[b] *Department of Mathematics, Duke University, Durham, NC 27705, USA*
[c] *Department of Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, NY 12180, USA*

ABSTRACT

The Kolmogorov n-width of the solution manifolds of transport-dominated problems can decay slowly. As a result, it can be challenging to design efficient and accurate reduced order models (ROMs) for such problems. To address this issue, we propose a new learning-based projection method to construct nonlinear adaptive ROMs for transport problems. The construction follows the offline–online decomposition. In the offline stage, we train a neural network to construct adaptive reduced basis dependent on time and model parameters. In the online stage, we project the solution to the learned reduced manifold. Inheriting the merits from both deep learning and the projection method, the proposed method is more efficient than the conventional linear projection-based methods, and may reduce the generalization error of a solely learning-based ROM. Unlike some learning-based projection methods, the proposed method does not need to take derivatives of the neural network in the online stage.

Published by Elsevier B.V.

1. Introduction

The transport phenomena arise in many important areas of applications, such as fluid dynamics, plasma physics and electromagnetics. Efficient and accurate reduced order models (ROMs) of transport-dominated problems are highly desired. However, as pointed out by [1–4], the Kolmogorov n-width [5] of the solution manifold for a transport problem may decay slowly. If one applies conventional ROM strategies such as proper orthogonal decomposition (POD) methods [6,7] or reduced basis methods (RBM) [8] straightforwardly, linear subspaces of rather high dimensions will be needed to obtain accurate reduced order approximations. The design of a model with genuine reduced order for transport problems is hence challenging.

Overcoming this difficulty has been an active research topic in the ROM community, where two general ideas have been extensively considered: (1) to introduce online adaptivity to the reduced solution space, and (2) to identify and conduct inherent nonlinear transformations which enable the construction of efficient ROMs. Examples following the former include [9,10], where [9] adaptively enriches the reduced basis in the online stage through splitting while, [10] applies online adaptive sampling and basis reconstruction based on a local low-rank structure. Examples following the latter direction include the shift POD method [11], which utilizes a shift operator determined by the underlying wave speed. Such information is also utilized under a Lagrangian framework in designing the Lagrangian POD [12], the Lagrangian

DMD [13] and its recent extension to the case with shocks [14]. Besides, nonlinear transformations can also be obtained by the method of freezing [15–17], the transformed snapshot interpolation [18–20], the approximated Lax pairs [21,22], template fitting [23,24], the transport reversal method [25], shock fitting [26], the registration method [27–29], and the greedy Wasserstein barycenter algorithm [30].

Our proposed method is inspired by the series of work [31–34], and is also closely related to [27–29,35–37]. The key underlying idea of [31–34] is that, for a transport-dominated problem, there exists some suitable nonlinear transformation $T_{t,\mu}$, under which the transformed solution manifold displays a much faster decay in its Kolmogorov n-width and hence can be accurately approximated by a low-dimensional reduced space. Once the transformation is obtained by some means, a reduced order approximation is constructed in the form of

$$u_r(x, t; \mu) = \sum_{i=1}^{r} \alpha_i(t, \mu)\psi_i(T_{t,\mu}(x; t, \mu)) = \sum_{i=1}^{r} \alpha_i(t, \mu)\phi_i(x; t, \mu). \tag{1.1}$$

Here, x is the physical location, t is time, μ is a scalar or vector-valued model parameter. Each ψ_i is a reduced basis function in the transformed space and $\phi_i(x; t, \mu) = \psi_i(T_{t,\mu}(x; t, \mu))$ is the respective reduced basis function in the original solution manifold. The distinct feature of the adaptive reduced basis $\{\phi_i(x; t, \mu)\}_{i=1}^{r}$ is that each member depends on time t and the model parameter μ, in addition to x. The nonlinear transformation $T_{t,\mu}$ can be realized as coordinate transformations defined case by case, such as for the 2D Euler equations in [31], the 1D Burgers equation in [33] and the Navier–Stokes equations in [34]. The manifold approximations via transported subspaces (MATS) [35], on the other hand, construct the nonlinear transformation with an optimal transport tool called displacement interpolation [38,39], while the transported snapshot method [36] applies a polynomial approximation in such constructions. The Lagrangian registration based method [27,29], the ALE registration method [28] and the ALE autoencoder [37] construct $T_{t,\mu}$ by solving nonlinear optimization problems.

In this paper, we propose a new framework to design nonlinear reduced order models for transport problems by working with an adaptively constructed reduced basis $\{\phi_i(x; t, \mu)\}_{i=1}^{r}$. The framework can be decomposed into an offline stage and an online stage. In the offline stage, instead of predetermining or seeking the transformation $T_{t,\mu}$, the adaptive basis will be learned directly from data with neural networks, since the ultimate goal is to obtain the reduced order space. More specifically, $\{\alpha_i(t, \mu)\}_{i=1}^{r}$ and $\{\phi_i(x; t, \mu)\}_{i=1}^{r}$ in (1.1) will be parameterized separately as feed-forward neural networks taking t, μ or x, t, μ as inputs, and these neural networks will be trained with snapshot solutions (e.g. high-fidelity numerical solutions). In the online stage, instead of making predictions entirely with the trained neural networks as in a pure learning-based method, we only take the learned basis $\{\phi_i(x; t, \mu)\}_{i=1}^{r}$ to construct a subspace that is local to t and μ, and apply the standard projection method to find the reduced order approximation. The performance of the proposed method is demonstrated numerically through a series of linear and nonlinear scalar hyperbolic conservation laws in 1D and 2D as well as the 1D compressible Euler system. The methodology developed here can also be applied to obtain reduced order approximations for other transport problems. Unlike the aforementioned methods that explicitly identify $T_{t,\mu}$ to build ROMs, our proposed method only learns the basis and the corresponding reduced order space, with the information of the transformation $T_{t,\mu}$ implicitly encoded in the learned space. Identifying $T_{t,\mu}$ can sometimes be helpful, but the explicit form of it is not always necessary.

The proposed learning-based projection method is a combination of the deep learning method and the standard projection method. Compared with a conventional projection method (e.g. POD), or a solely learning-based method, the proposed method has several advantages in constructing ROMs. Our numerical experiments show that the proposed method is more accurate than the standard POD method for long time predictions with a similar order reduction. The reason behind is that the inherent nonlinear information of the underlying transport problems can be implicitly embedded in the adaptive basis through training while a conventional linear projection method may suffer from the slow decay of the Kolmogorov n-width. Particularly, we observe that the adaptive basis obtained in fact travels at the underlying wave speed for a simple 1D linear advection equation. Another advantage over the conventional projection method is that the neural network takes x as an input in a mesh-free manner, hence it is simple and flexible to work with different meshes in the online and offline stages, or even learn from experimental mesh-free data. Comparing with a pure learning-based approach, our method can reduce the generalization error due to overfitting with an online projection procedure utilizing the underlying PDE information.

We also want to point out that we are not the first to combine projection methods with the deep learning techniques to build ROMs for transport problems. In [40–42], auto-encoders were used to learn a low-dimensional nonlinear approximation of the solution manifold offline, and a reduced-order solution can then be obtained through an online projection . The main differences between our method and aforementioned methods are as follows. First, [40–42] are based on a different reduced-order representation. Second, in the online computations, different from our method, these methods need to take the derivative of the decoder neural network, and this could be computationally expensive. To improve their online efficiency, [42] uses a decoder with a shallow architecture and applies hyper-reduction techniques. Third, compared with these methods, our method does not require using the same spatial meshes at online and offline stages, and is thus advantageous for moving mesh methods or h-adaptive simulations with locally refined meshes. Similar ideas as in [40–42] are also utilized to develop a learning based ROM [43] under a pure data-driven framework. There, a reduced order nonlinear trial manifold is constructed by the decoder part of an auto-encoder trained offline. Instead of

using a projection-based method online, [43] evolves the coordinate in the low-dimensional latent space through an extra neural network which is trained offline to learn the reduced dynamics. This data-driven online phase is more efficient than the projection based online algorithm in [40–42] and our method, but may suffer from larger generalization errors. Another interesting work that uses deep learning to construct ROMs for transport problems is [44], which utilizes ideas similar to the MATS [35] to lighten the architectures of neural networks.

The main difference between our method and the optimal transport based method [35], optimization based registration/autoencoder [27–29,37] and the transported snapshot method [36] is that we find the adaptive reduced order space without explicitly finding the transformation, while they explicitly find the transformation. We use a neural network ansatz for the adaptive reduced basis, while these methods use different ansatz. Our method can be seen as an alternative to these methods. We would also like to make more comments about the optimal transport based methods [30,35]. Compared with deep learning based methods, these optimal transport based methods are more interpretable, and theoretical understandings have been established. However, more developments are still needed for more complicated transport problems. The greedy Wasserstein barycenter method [30] and its recent generalization to the nonconservative flows in porous media [45] are limited to 1D for now, as they use an exponential map only valid in 1D to efficiently compute the Wasserstein distance. The MATS method [35] transports reduced subspaces along characteristics and cannot directly handle the case when characteristics intersect (e.g. problems with shocks). As demonstrated in our numerical examples, our proposed method can handle 2D problems and problems with shocks. We also want to mention that [44] generalizes [35] with deep learning techniques and could handle problems with shocks, but its connection with optimal transport becomes less clear.

The paper is organized as follows. In Section 2, we briefly review the basic idea of projection-based ROM methods and explain why designing an efficient ROM of transport problems could be challenging. Then, in Section 3, we present the offline and online algorithms of the proposed learning-based projection method as well as the neural network architecture. In Section 4, the performance of the proposed method is demonstrated through a series of numerical experiments. In Section 5, we draw our conclusions and discuss some potential future research directions.

2. Background

We consider a parametric time-dependent transport problem:

$$\text{residual}(u; \mu) = 0, \ \mu \in X_{\mathcal{P}}, \tag{2.2}$$

where μ is a scalar or vector-valued model parameter and $\text{residual}(\cdot; \mu)$ represents a time-dependent parametric hyperbolic partial differential operator. The solution $u(x, t; \mu)$ depends on the location $x \in \Omega$, the time t and the parameter μ. To design a reduced order model, we assume a full order model is available, which is a high-fidelity numerical scheme solving (2.2) and can be written as

$$\text{residual}_h(u_h; \mu) = 0 \tag{2.3}$$

in its strong form. We view the solution $u_h(x, t; \mu)$ as the ground truth, and denote the solution manifold as $\{u_h(\cdot, t; \mu) : 0 \le t \le T_{\max}, \mu \in X_{\mathcal{P}}\}$.

A conventional projection-based ROM seeks a solution in the form of

$$u_r(x, t; \mu) = \sum_{i=1}^{r} \alpha_i(t, \mu)\varphi_i(x). \tag{2.4}$$

Here, r is the order of the reduced order model, $\{\varphi_i(x)\}_{i=1}^{r}$ is a reduced basis, with the expansion coefficients $\boldsymbol{\alpha}(t, \mu) = (\alpha_1(t, \mu), \ldots, \alpha_r(t, \mu))^T$. The reduced basis depends on x, and is typically obtained through the offline training. The coefficients are then determined online through projection methods. Such procedure provides efficient and accurate reduced order models for problems whose solution manifolds have fast-decaying Kolmogorov n-width [46,47].

However, it is well-known that the Kolmogorov n-width for transport-dominated problems may decay slowly [1–4]. On the other hand, the evidence in [31] indicates that under some coordinate transformations, the transformed solution manifolds of transport-like problems can have much faster decay in their Kolmogorov n-width. This can be illustrated by the following simple example. Consider the 1D advection equation

$$u_t + u_x = 0, \quad x \in [0, 2], \tag{2.5}$$

with zero boundary conditions and the following initial condition

$$u(x, 0) = u_0(x) = \begin{cases} 1, & 0.25 \le x \le 0.5, \\ 0, & \text{otherwise.} \end{cases} \tag{2.6}$$

The exact solution is $u(x, t) = u_0(x - t)$ (over the time period $[0, 1]$ of our consideration). We introduce a uniform space–time mesh with $\Delta x = \Delta t = 0.01$ and $x_i = i\Delta x$, $t_j = j\Delta t$, and use the exact solution sampled from this mesh over $t \in [0, 1]$ to define the solution manifold, whose Kolmogorov n-width with respect to the l_2 norm is measured as the nth
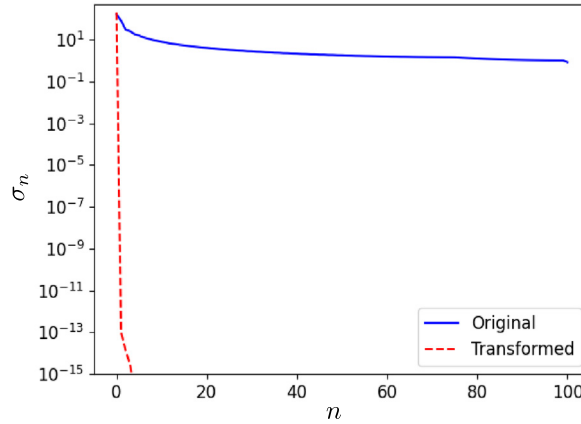
Fig. 2.1. Singular values of the original and transformed snapshot matrices.

singular value of the snapshot matrix $S = (s_{ij})$, with $s_{ij} = u(x_i, t_j) = u_0((i - j)\Delta x)$. Meanwhile, we can follow [31] and apply a t-dependent transformation $T(x; t) = x - t$, and obtain a transformed snapshot matrix $\tilde{S} = \{\tilde{s}_{ij}\}$ with

$$\tilde{s}_{ij} = u(T^{-1}(x_i; t_j), t_j) = u(x_i + t_j, t_j) = u_0(x_i).$$

In Fig. 2.1, the singular values of the original and transformed snapshot matrices are presented. We observe that the singular values of the original snapshot matrix decay fairly slowly as expected, while those of the transformed snapshot matrix decay very fast. As a result, standard ROMs such as POD will require a fairly large reduced space for accurate approximation, yet with the help of the t-dependent transformation $T(x; t)$, the dimension of the reduced space can be dramatically reduced.

The approximation power of the standard reduced order spaces (e.g. the POD space) is limited as they are linearly expanded with basis functions that are static with respect to t and μ, see (2.4). As shown above, one way to conquer this limitation is to use a reduced order space with an adaptive time-and-parameter-dependent (i.e. (t, μ)-dependent) basis $\{\phi_i(x; t, \mu)\}_{i=1}^r$, with the reduced order approximation given as:

$$u_r(x, t; \mu) = \sum_{i=1}^{r} \alpha_i(t, \mu)\phi_i(x; t, \mu). \tag{2.7}$$

In the series of paper [31,33,34], the basis used takes the form $\phi_i(x; t, \mu) = \psi_i(T_{t,\mu}(x))$, where $T_{t,\mu}$ is a suitable coordinate transformation defined problem by problem. Similar ideas have also been considered in [27–29,35–37]. In [35], $T_{t,\mu}$ is constructed with the displacement interpolation [38,39] while the polynomial approximation is used in [36]. The Lagrangian and arbitrary Lagrangian Eulerian (ALE) registration method [27–29] and the ALE autoencoder [37] find $T_{t,\mu}$ by solving optimization problems.

Following [31,33,34], we propose a learning-based projection method. To avoid designing suitable coordinate transformations $T_{t,\mu}$ *a priori* for each problem, we parameterize the basis functions $\{\phi_i(x; t, \mu)\}_{i=1}^r$ as a neural network $\mathcal{NN}^\phi(x, t, \mu; \theta^\phi)$ and determine the trainable parameters of the neural network θ^ϕ in the offline learning stage. In the online stage, we further utilize a projection method to predict solutions either at a future time or corresponding to a new value of the model parameter μ. The ROM algorithms will be detailed in Section 3.

3. Reduced order model algorithms

One natural extension to [31,33,34] is utilizing neural networks to learn a coordinate transformation $T_{t,\mu}$. However, it is nontrivial to measure how a coordinate transformation reduces the Kolmogorov n-width of the transformed solution manifold for a general problem with neural networks, not to mention to systematically find a good one. Instead, we directly learn the (t, μ)-dependent adaptive basis, hence the adaptive reduced order space, using neural networks, while the transformation $T_{t,\mu}$ is implicitly encoded.

We follow the standard offline–online decomposition of the projection-based ROM method. In Section 3.1, we present the offline stage algorithm which obtains the adaptive basis by training two neural networks with snapshot solutions. In the online stage, with the learned (t, μ)-dependent basis available, we apply a projection-based method to construct a reduced representation of the solution, and this will be detailed in Section 3.2. A discussion over the architecture of the neural network is then presented in Section 3.3.

3.1. Offline stage

In the offline training stage, the snapshot solutions are high-fidelity numerical solutions from a full order model. For the time-dependent transport problems considered in this work, we assume the full order model is given by a one-step scheme

$$\text{residual}_h(u_h^{n+1}, u_h^n; \mu) = 0 \tag{3.8}$$

in its strong form, and it can be constructed through a finite difference/volume/element method, or a discontinuous Galerkin method. Here $u_h^n = u_h(x, t_n; \mu)$ denotes the numerical solution at time t_n. The snapshot solutions can also be obtained from measurements or practical experiments.

Suppose we have the snapshot solutions, $u_h(x_i, t_j; \mu_k)$, with

$$\{x_i\}_{i=0}^{N_x} \subset \Omega, \quad \{t_j\}_{j=1}^{N_t} \subset [0, T_{\text{train}}], \quad \{\mu_k\}_{k=1}^{N_\mu} \subset \mathcal{P}_{\text{train}} \subset X_{\mathcal{P}}.$$

Note that the spatial degrees of freedom of u_h used here are nodal values. Our proposed framework can be adapted to ROMs with other types of spatial degrees of freedom, such as moments. Though the snapshot solutions are taken from $[0, T_{\text{train}}]$, the proposed method may be applied to predict solutions at a future time $t > T_{\text{train}}$.

To obtain the (t, μ)-dependent basis, we define the following rth order neural network approximation:

$$u_{\mathcal{N}\mathcal{N}}(x, t; \mu, \boldsymbol{\theta}) = \sum_{i=1}^r \alpha_i^{\mathcal{N}\mathcal{N}}(t, \mu; \theta^\alpha) \phi_i^{\mathcal{N}\mathcal{N}}(x, t, \mu; \theta^\phi)$$
$$= \boldsymbol{\alpha}^{\mathcal{N}\mathcal{N}}(t, \mu; \theta^\alpha) \cdot \boldsymbol{\phi}^{\mathcal{N}\mathcal{N}}(x, t, \mu; \theta^\phi). \tag{3.9}$$

Here, $r \ll N_x$ is the order of the ROM. $\boldsymbol{\theta} = (\theta^\alpha, \theta^\phi)$ are trainable parameters of the neural network. The r-dimensional vectors $\boldsymbol{\alpha}^{\mathcal{N}\mathcal{N}} = (\alpha_1^{\mathcal{N}\mathcal{N}}, \dots, \alpha_r^{\mathcal{N}\mathcal{N}})^T$ and $\boldsymbol{\phi}^{\mathcal{N}\mathcal{N}} = (\phi_1^{\mathcal{N}\mathcal{N}}, \dots, \phi_r^{\mathcal{N}\mathcal{N}})^T$ are the outputs of the neural networks $\mathcal{N}\mathcal{N}^\alpha(t, \mu; \theta^\alpha)$ and $\mathcal{N}\mathcal{N}^\phi(x, t, \mu; \theta^\phi)$, where $\mathcal{N}\mathcal{N}^\alpha(t, \mu; \theta^\alpha)$ and $\mathcal{N}\mathcal{N}^\phi(x, t, \mu; \theta^\phi)$ parameterize the expansion coefficients and the (t, μ)-dependent basis, respectively. The neural network is trained by minimizing the mean-square loss function:

$$\boldsymbol{\theta^*} = (\theta^{\alpha,*}, \theta^{\phi,*}) = \arg\min_{\boldsymbol{\theta}} \frac{1}{(N_x + 1)N_t N_\mu} \sum_{i=0}^{N_x} \sum_{j=1}^{N_t} \sum_{k=1}^{N_\mu} \left(u_{\mathcal{N}\mathcal{N}}(x_i, t_j; \mu_k, \boldsymbol{\theta}) - u_h(x_i, t_j; \mu_k)\right)^2. \tag{3.10}$$

The solution to the optimization problem in (3.10) is not unique in the sense that one can reorder and rescale basis functions. Keep in mind that what we truly want is not the basis itself but the associated adaptive reduced order space. In practice, the ordering and the scaling of the learned basis will be influenced by the random initialization of the neural networks. Also, due to the non-convex nature of the underlying minimization problem, the optimizer may converge to a low dimensional manifold corresponding to a local minimum. But at least for numerical tests being considered, these issues do not seem to pose any problem in the present work, either to the training of the neural networks or to the overall performance of the proposed algorithm. Nevertheless, it will be left to our future exploration to learn more structured basis functions such as an orthonormal basis.

Essentially, the neural network $u_{\mathcal{N}\mathcal{N}}(\cdot, \cdot, \boldsymbol{\theta^*})$ provides a reduced order model to approximate the true solution. However, due to the potential risk of over-fitting, the purely learning-based reduced approximation $u_{\mathcal{N}\mathcal{N}}$ is not always accurate and robust in generalization. We propose to only pass the learned adaptive basis $\{\phi_i^{\mathcal{N}\mathcal{N}}\}_{i=1}^r$ onward to the online stage.

It is worth noticing that $\mathcal{N}\mathcal{N}^\phi$ takes x as one of its inputs in a mesh free manner, and therefore it is simple and flexible to work with solution snapshots obtained on different meshes tailored for different values of the model parameter, or even solution snapshots based on experimental data. However, a conventional projection-based method may need extra steps such as interpolations [48] or full order offline re-computations [49,50] in order to work with snapshots subject to different meshes.

Remark 3.1. For now, the reduced order r is pre-selected. One way to adaptively determining r is to apply a block-by-block training strategy similar to [51]. More specifically, start with a pre-selected small r_0 and train the neural network $u_{\mathcal{N}\mathcal{N}_0} = \sum_{i=1}^{r_0} \alpha_i^{\mathcal{N}\mathcal{N}}(t, \mu; \theta^\alpha)\phi_i^{\mathcal{N}\mathcal{N}}(x, t, \mu; \theta^\phi)$. If the prediction error of a validation set is too large with the pre-selected r_0, one can train a second neural network $u_{\mathcal{N}\mathcal{N}_1} = \sum_{i=r_0+1}^{r_0+\Delta r} \alpha_i^{\mathcal{N}\mathcal{N}}(t, \mu; \theta^\alpha)\phi_i^{\mathcal{N}\mathcal{N}}(x, t, \mu; \theta^\phi)$ by minimizing the mismatch between $u_{\mathcal{N}\mathcal{N}_0} + u_{\mathcal{N}\mathcal{N}_1}$ and the full order solution u_h. Repeat this procedure until the prediction error of the validation set is small enough or reach the maximum number of basis functions allowed.

Remark 3.2. In practice, it is important to have an effective sampling strategy when the dimension of the problem or the parameter μ is high. In order to reduce the computational cost of sampling, one may apply an adaptive sampling strategy based on the greedy algorithm [8,52] or the "data exploration" in [53]. Effective sampling strategy is not the focus of this paper, and for now, we do not apply these strategies and leave them for future investigations. Similarly, the terminal training time T_{train} can also be adaptively determined, e.g. based on the validation error or an error estimator.

3.2. Online stage

In the online stage, we take the learned adaptive basis $\{\phi_i^{\mathcal{NN}}\}_{i=1}^r$ and seek a reduced-order numerical solution in their span following a projection approach. For simplicity, we denote $\phi_i^{n+1}(x; \mu) = \phi_i^{\mathcal{NN}}(x, t_{n+1}, \mu; \theta^{\phi,*})$, $i = 1, \ldots, r$. The reduced order space at t_{n+1} associated with μ is then given as

$$V_r^{n+1}(\mu) := \text{span}\{\phi_i^{n+1}(\cdot; \mu)\}_{i=1}^r.$$

We also introduce $\boldsymbol{\Phi}_\mu^{n+1} \in \mathbb{R}^{(N_x+1) \times r}$, with its (i, j)-th entry being

$$\boldsymbol{\Phi}_\mu^{n+1}(i, j) = \phi_j^{n+1}(x_i; \mu), \quad i = 0, \ldots, N_x, \quad j = 1, \ldots, r. \tag{3.11}$$

Suppose the full order model (3.8) has the following matrix–vector form:

$$\boldsymbol{F}_\mu(\mathbf{u}_\mu^{n+1}) = \mathbf{b}_\mu^n, \tag{3.12}$$

where $\mathbf{u}_\mu^{n+1} \in \mathbb{R}^{N_x+1}$ is the vector of nodal values (i.e. the spatial degrees of freedom) of the numerical solution $u_h(x, t_{n+1}; \mu)$ at $\{x_j\}_{j=0}^{N_x}$ and time t_{n+1}. $\mathbf{b}_\mu^n \in \mathbb{R}^{N_x+1}$ is a known vector determined by $\mathbf{u}_\mu^n$, and the operator $\boldsymbol{F}_\mu : \mathbb{R}^{N_x+1} \mapsto \mathbb{R}^{N_x+1}$ is derived from the full order model (3.8). We look for the reduced order approximation of the solution $u_{r,h}^{n+1} \in V_r^{n+1}(\mu)$, whose nodal values at $\{x_j\}_{j=0}^{N_x}$ (i.e. spatial degrees of freedom) are $\boldsymbol{\Phi}_\mu^{n+1} \boldsymbol{\alpha}^{n+1}$, with some $\boldsymbol{\alpha}^{n+1} = (\alpha_1^{n+1}, \ldots, \alpha_r^{n+1})^T \in \mathbb{R}^r$. Following a projection method, we require

$$(\boldsymbol{\Phi}_\mu^{n+1})^T \boldsymbol{F}_\mu(\boldsymbol{\Phi}_\mu^{n+1} \boldsymbol{\alpha}^{n+1}) = (\boldsymbol{\Phi}_\mu^{n+1})^T \mathbf{b}_\mu^n. \tag{3.13}$$

Once $\boldsymbol{\alpha}^{n+1}$ is calculated, we reach a reduced-order approximation of the solution at t_{n+1}: $u_{r,h}^{n+1} = \sum_{i=1}^r \alpha_i^{n+1} \phi_i^{n+1}$. The online algorithm is summarized in Algorithm 1.

Algorithm 1: Online projection for one time step update

1: **Input:** given the parameter μ; assume the reduced order numerical solution $u_{r,h}^n$ at time t_n is known, and it defines $\mathbf{b}_\mu^n$ in (3.12) based on the full order model (3.8).

2: **Evaluate the (t, μ)-dependent basis with the neural network $\mathcal{NN}^\phi$ trained offline:**

$$\phi_i^{n+1}(x; \mu) := \phi_i^{\mathcal{NN}}(x, t_{n+1}, \mu; \theta^{\phi,*}), \quad i = 1, \cdots, r. \tag{3.14}$$

And form the matrix $\boldsymbol{\Phi}_\mu^{n+1} \in \mathbb{R}^{(N_x+1) \times r}$ as in (3.11).

3: **Obtain $u_{r,h}^{n+1}$ through the projection:** compute $\boldsymbol{\alpha}^{n+1}$ by

$$(\boldsymbol{\Phi}_\mu^{n+1})^T \boldsymbol{F}_\mu(\boldsymbol{\Phi}_\mu^{n+1} \boldsymbol{\alpha}^{n+1}) = (\boldsymbol{\Phi}_\mu^{n+1})^T \mathbf{b}_\mu^n. \tag{3.15}$$

4: **Output**: the reduced order numerical solution at t_{n+1} for the parameter μ is

$$u_{r,h}^{n+1} = \sum_{i=1}^r \alpha_i^{n+1} \phi_i^{n+1}.$$

Compared with a standard projection-based method such as the POD method, in the online stage, we have the extra step to evaluate the (t, μ)-dependent basis with a trained neural network with known parameters $\theta^{\phi,*}$ at each time step t_{n+1}. In our numerical tests, we find that, even with this additional cost, our method still leads to computational savings compared with a full order model when the underlying mesh is well refined.

As mentioned before, the neural network $\mathcal{NN}^\phi$ takes x as one of its inputs in a mesh free manner. Although it is not deeply investigated in this work, the proposed method has the capability to be integrated with h-adaptive numerical schemes for time dependent problems in the online stage. In Section 4.3, the proposed method is combined with a moving mesh method to show its capability.

We also want to compare our proposed method with the learning-based projection methods in [40–42]. In the offline stage, taking the full order solution as the input, these methods find a low-dimensional nonlinear manifold to represent the full order solution manifold by training an auto-encoder. In the online stage, these methods seek a reduced order approximation in the form

$$u \approx u_{\text{ref}} + g(\hat{u}_r). \tag{3.16}$$

Here, u_{ref} is a predetermined reference frame, g is the decoder part of the neural-network trained offline, which is a mapping from a low-dimensional latent space to the high-dimensional solution manifold, and $\hat{u}_r$ can be seen as the coordinates in the low-dimensional latent space. In the online stage, these methods solve a minimal residual problem

$$\hat{u}_r^{n+1} = \arg\min \|\text{residual}_h(g(\hat{u}_r^{n+1}), u_h^n; \mu)\|_2. \tag{3.17}$$

To obtain $\hat{u}_r^{n+1}$, [40,41] need to take the derivatives or even the second order derivatives of the neural-network g online, which can be computationally expensive. To improve the online efficiency, [42] uses a shallow architecture for the decoder neural network g and applies the hyper-reduction techniques. Unlike these methods, we only need to compute α^{n+1} as in (3.15) and there is no need to take derivatives of neural networks at all in our method. Furthermore, since the input of our method is not full/reduced-order solutions as of the auto-encoders, our method is relatively more flexible to work with different meshes in the online and offline stages. We also want to point out that [43] proposes an alternative data-driven online phase for this auto-encoder and deep learning based framework. The data-driven online phase of [43] is more efficient than the projection-based online phase in [40–42] and our method, while the projection-based online algorithm may lead to better accuracy.

Remark 3.3. In our method, we require the projection of the residual of the full order model (3.8) onto the local reduced order space to vanish. Alternatively, in our method, one can also use a minimal residual formulation similar to [31,41,42]:

$$\boldsymbol{\alpha}^{n+1} = \arg\min_{\boldsymbol{\alpha} \in \mathbb{R}^r} \|\text{residual}_h(\sum_{i=1}^r \alpha_i \phi_i^{n+1}, u_h^n; \mu)\|_2. \tag{3.18}$$

In the matrix–vector formulation, this alternative online step becomes

$$(\boldsymbol{\Phi}_\mu^{n+1})^T (\boldsymbol{J}_\mu^{n+1})^T \boldsymbol{F}_\mu(\boldsymbol{\Phi}_\mu^{n+1} \boldsymbol{\alpha}^{n+1}) = (\boldsymbol{\Phi}_\mu^{n+1})^T (\boldsymbol{J}_\mu^{n+1})^T \mathbf{b}_\mu^n. \tag{3.19}$$

Here, $\boldsymbol{J}_\mu^{n+1}$ is the Jacobian matrix of $\boldsymbol{F}_\mu$ with respect to $\boldsymbol{u}_\mu^{n+1}$, and this can be seen as a weighted projection.

Remark 3.4. To work with the operator $(\boldsymbol{\Phi}_\mu^{n+1})^T \boldsymbol{F}_\mu(\boldsymbol{\Phi}_\mu^{n+1})$ in (3.15), an $O(N_x)$ computational cost will be needed per time step. Unlike ROM methods (e.g. POD) with static bases, the reduced basis in our method is updated dynamically and cannot be precomputed. As a result, the proposed method will not save computational time without hyper-reduction when explicit time steppers are applied. Our method is indeed more suitable when implicit time integrators are preferred, for example, in the cases of very non-uniform spatial meshes or for stiff problems.

Hyper-reduction strategies have been considered for the ROMs of transport dominant problems. To predict solutions for unseen values of physical parameter μ (yet over $[0, T_{\text{train}}]$), the autoencoder neural network based projection methods [42,54] apply the DEIM-SNS/GNAT-SNS [55] and the reduced over-collocation method [56] for the hyper-reduction, respectively. For steady state problems, the l_0 minimization based strategy in [29] has been designed. When predicting the solutions for unseen values of μ at time $t \in [0, T_{\text{train}}]$, a hyper-reduction strategy similar to the aforementioned methods may be combined with our method. However, to accurately predict solutions at future time $t > T_{\text{train}}$, an adaptive hyper-reduction strategy may be necessary, and the design of such adaptive strategy could be challenging.

Alternative to the hyper-reduction techniques, one can also enhance the computational efficiency of explicit time steppers by freezing the basis over a few time steps as in [10]. This will lead to an $O(N_x)$ computational cost for one time step followed by an $O(r)$ cost for a few time steps. The key is to design an adaptive strategy determining when to update the basis for a good balance of efficiency and accuracy.

3.3. Neural network architecture

As indicated by (3.9), we parameterize the reduced-order basis and the corresponding coefficient as independent neural networks, namely $\mathcal{NN}^\alpha(\mu, t; \theta^\alpha)$ and $\mathcal{NN}^\phi(x, t, \mu; \theta^\phi)$, respectively. More specifically, we require each neural network to output a vector of dimension r, where each component stands for one dimension in the reduced latent space. The inner product between the output vectors of the two neural networks will provide a reduced order approximation for the solution.

There is much flexibility in choosing the neural network architecture. In this paper, we take each neural network to be simply a feed-forward fully-connected neural network. A standard n-layer architecture of such neural network has a repeated compositional structure, where each layer is composed of a linear function and a nonlinear activation functions σ:

$$NN(\mathbf{x}; \boldsymbol{\theta}) = W_n \sigma \left(\cdots \sigma \left(W_2 \sigma \left(W_1 \mathbf{x} + b_1 \right) + b_2 \right) \cdots \right) + b_n.$$

Here, $\boldsymbol{\theta}$ stands for the collection of all trainable parameters, that is, $\boldsymbol{\theta} = \{W_1, b_1, \ldots, W_n, b_n\}$. Other architectures can also be considered within our proposed reduced-order model framework. An interesting choice will be the partition of unity (POU) neural network [57], which is related to h, p-adaptive methods and can potentially be beneficial for the construction of ROMs for transport problems. Another architecture that can be potentially used is the DeepONet [58]. The DeepONet can take μ and x, t in different ways, and this is natural in parameterizing a parameter-dependent function.

The entire algorithm can be summarized as the flow chart in Fig. 3.2.
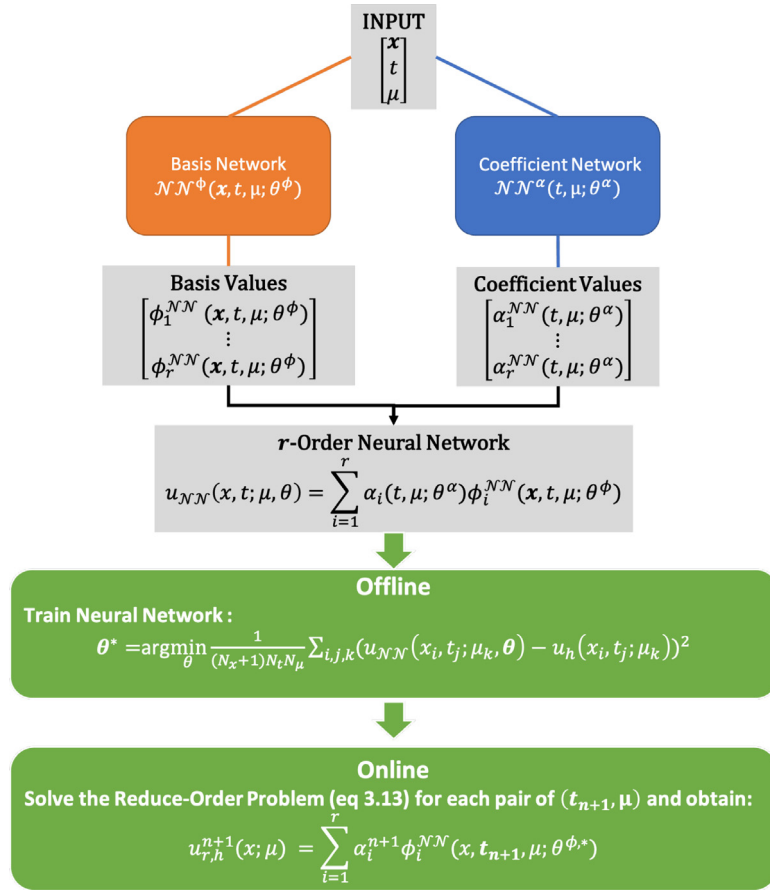
Fig. 3.2. The flow chart of the proposed algorithm.

4. Numerical experiments

In this section, we demonstrate the performance of the proposed learning-based projection (LP) method by applying it to linear and nonlinear hyperbolic conservation laws in 1D and 2D. Particularly, the 1D and 2D linear advection equation, the 1D and 2D Burgers equation, and the 1D compressible Euler system of gas dynamics are tested. We also compare our method with the standard POD method and the prediction directly provided by the neural network $u_{\mathcal{NN}}$ in (3.9). Throughout this section, the reduced order solution obtained with the learning-based projection method is referred to as "LP" and the one directly predicted by the neural network $u_{\mathcal{NN}}$ is referred to as "learning".

In our numerical experiments, we adopt the feed forward neural network architecture for both $\mathcal{NN}^{\phi}$ and $\mathcal{NN}^{\alpha}$. The activation function is taken as tanh, except for the output layers which are defined as

$$\mathbf{y}_{\text{output}} = \text{softmax}(W\mathbf{x}) + b, \tag{4.1}$$

for both the basis and coefficients. Here, W is the weight and b is the bias and $\mathbf{x}$ stands for the output of the previous layer. Among different activation functions of the output layer we have numerically experimented with, softmax outperforms other choices. As pointed out in [57], the softmax activation function introduces h-adaptivity to some extent, which may have made the neural network more suitable for the transport problems with local structures. As presented in Section 4.4, the proposed method is not sensitive to the choice of the width and the depth of the hidden layers of the neural network. Unless specified, the neural network $\mathcal{NN}^{\phi}$ has 4 hidden layers of 25 neurons per layer and an output layer with r neurons. The neural network $\mathcal{NN}^{\alpha}$ has 3 hidden layers of 25 neurons per layer and an output layer with r neurons.

Our code is implemented under the framework of Tensorflow 2.0 [59]. When an implicit time integrator such as backward Euler or Crank–Nicolson method is applied, the associated linear or linearized systems are solved by the GMRES method. If the equation is nonlinear, the Newton's method will be applied first as an outer loop iterative solver.

4.1. Preliminaries

To set the stage, we start with a 1D scalar hyperbolic conservation law

$$u_t + (f(u))_x = 0, \ x \in [x_L, x_R] \tag{4.2}$$

with an initial condition and some suitable boundary conditions.

Let $x_L = x_0 < x_1 \cdots < x_{N_x} = x_R$ be a uniform mesh in space, with $x_i = x_L + i\Delta x$ and $\Delta x = \frac{x_R - x_L}{N_x}$, and let Δt be the time step size. One of the full order models used in our experiments is the following first-order method, that involves the backward Euler method in time and a first-order conservative finite difference method in space:

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + \frac{f_{j+\frac{1}{2}}^{n+1} - f_{j-\frac{1}{2}}^{n+1}}{\Delta x} = 0. \tag{4.3}$$

Here, $u_j^n \approx u(x_j, t_n)$, $f_{j+\frac{1}{2}}^n$ is a numerical flux at $x_{j+\frac{1}{2}} = \frac{x_j + x_{j+1}}{2}$, which will be specified later. Boundary conditions are imposed through numerical fluxes. The method in (4.3) can be easily extended to non-uniform meshes, and it can also be interpreted as a finite volume method, with u_j^n as the element average.

Throughout this section, the L_2 error and the relative L_2 error for the solution associated with a test parameter μ at t_n are defined as

$$\mathcal{E}_{L_2}^n(\mu) = \sqrt{\sum_{j=0}^{N_x} (u_{\text{reduced},j}^n - u_{\text{full},j}^n)^2 \Delta x}, \tag{4.4a}$$

$$\mathcal{E}_{\text{relative}}^n(\mu) = \frac{\mathcal{E}_{L_2}^n(\mu)}{\sqrt{\sum_{j=0}^{N_x} (u_{\text{full},j}^n)^2 \Delta x}}, \tag{4.4b}$$

respectively. The average relative L_2 error for a test set $\{\mu_i\}_{i=1}^{N_\mu}$ from t_{n_0} to t_{n_m} is defined as

$$\mathcal{E}_{\text{average}} = \frac{1}{N_\mu} \sum_{i=1}^{N_\mu} \max_{n_0 \leq n \leq n_m} \left\{ \mathcal{E}_{\text{relative}}^n(\mu_i) \right\}. \tag{4.5}$$

The 2D analog of these quantities can be defined similarly.

4.2. 1D linear advection equation: learned basis functions

We first consider the 1D linear advection equation of the constant velocity 1:

$$u_t + u_x = 0, \tag{4.6}$$

with the initial condition

$$u(x, 0) = \begin{cases} 2, & 0.5 \leq x \leq 1.5, \\ 0, & \text{otherwise.} \end{cases} \tag{4.7}$$

The computational domain is $[0, 5]$, and it is partitioned with a uniform mesh with $\Delta x = 0.01$. The time step size is taken as $\Delta t = \Delta x$. We use the exact solution from $t = [0, 0.75]$ to train the neural network with the reduced order $r = 3$. The purpose of this example is to test whether the proposed method is able to capture the underlying t-dependent transformation.

In Fig. 4.3, we present the learned basis functions at three different times obtained offline. We observe that, as time evolves, all three basis functions propagate from the left to the right with a wave speed approximately 1. This qualitatively verifies that the proposed LP method is able to capture the underlying physics. In Fig. 4.4, we further qualitatively measure the difference between the learned basis functions and shifted initial basis functions with wave speed 1. As time evolves, the difference will be larger.

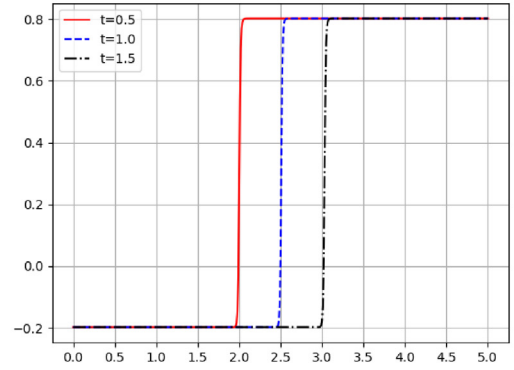### 4.3. 1D linear advection equation with a moving mesh method

The aim of this example is to show the flexibility and the effectiveness of our method to couple with a full order moving mesh method.

We consider the same linear advection equation as in Section 4.2, but with a slightly different initial condition
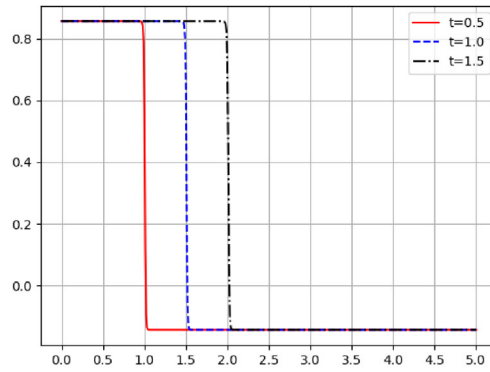
$$u(x, 0) = \begin{cases} 2, & 0.6 < x < 1.4, \\ 0, & \text{otherwise.} \end{cases} \tag{4.8}$$
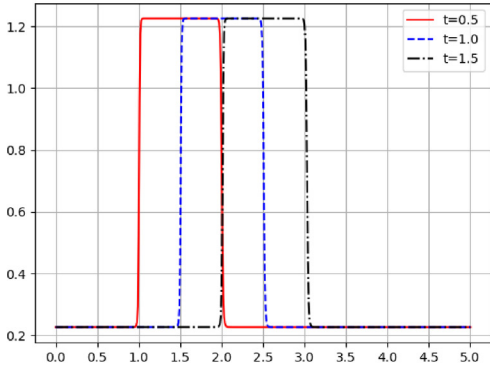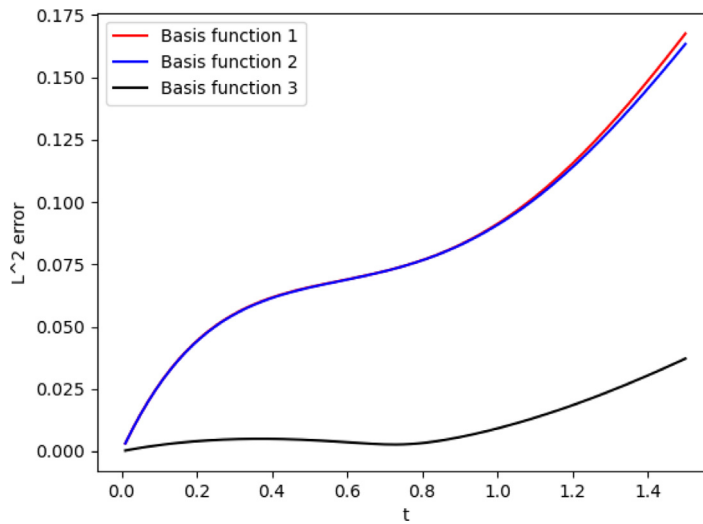
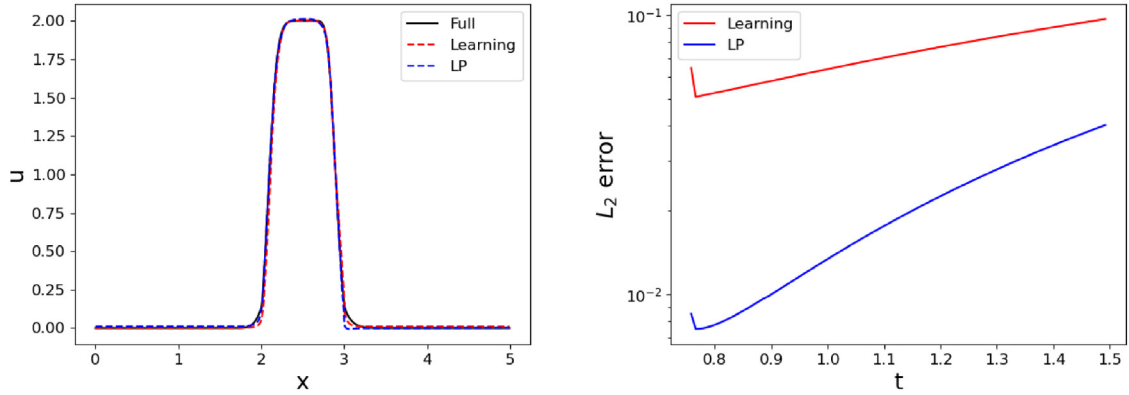(a) Basis function 1

(b) Basis function 2

(c) Basis function 3

Fig. 4.3. The learned basis functions $\phi_i^{\mathcal{NN}}$ for $i = 1, 2, 3$ at different times for the linear advection equation in Section 4.2.

Fig. 4.4. The difference between the learned basis functions and the shifted initial basis functions for the linear advection equation in Section 4.2.

(a) Predicted solutions with 15 reduced basis functions   (b) Error history with 15 reduced basis functions

Fig. 4.5. Predicted solutions at $t = 1.5$ (left) and the error history (right) with 15 reduced basis functions for the moving mesh test in Section 4.3.

We use a non-uniform moving mesh: $0 = x_0(t) < x_1(t) < \cdots < x_{N_x}(t) = 5.0$, with $N_x = 1000$. Define $h_{\text{coarse}} = \frac{1}{125}$ and $h_{\text{refine}} = \frac{1}{500}$. The grid point at time t, namely, $x_i(t)$ $(1 \le i \le N_x - 1)$, satisfies

$$x_i(t) = \begin{cases} x_{i-1}(t) + h_{\text{refine}}, & \text{if } 0.5 + t \le x_{i-1}(t) \le 1.5 + t, \\ x_{i-1}(t) + h_{\text{coarse}}, & \text{otherwise.} \end{cases} \tag{4.9}$$

This mesh moves with the underlying wave speed and is always more refined near the discontinuity of the solution. With this moving mesh, the first order upwind finite volume method with $f_{j+\frac{1}{2}}^n = u_j^n$ is applied in space, together with the backward Euler method in time and the time step size $\Delta t = h_{\text{coarse}}$. Before marching to the next time step, an extra key step of a moving mesh method is to interpolate the solution from the mesh at t^{n-1} to the mesh at t^n. In our simulation, we follow the Step 2 of the moving mesh algorithm [60] to do this interpolation. When our LP method is applied, we use the same interpolation subroutine to interpolate predicted solution from the old mesh to the new mesh. Extra cost reduction can be done in this step, and this is not explored here.

The training set is the snapshot solutions for $t \in [0, 0.75]$ (95 time snapshots), and we will predict the solution from $t = 0.75$ to $t = 1.5$. With the moving mesh, each solution snapshot corresponds to a different mesh. As a result, interpolation of basis functions from different meshes [48] or extra refinement and re-computation with a more refined common mesh [49,50] may be needed for the standard POD or RB method. Our neural networks take x, t as inputs in a mesh-free manner, and hence do not need additional treatments to work with basis functions associated with different meshes. Here, we only present results of the pure learning based method and the LP method.

In Fig. 4.5, we present the predicted solution at $t = 1.5$ and the error history with $r = 15$. Both the learning based and the LP methods match the full order solution well, and the error of the LP method is overall smaller. From Fig. 4.6(a) (resp. Figs. 4.6(c)- 4.6(d)) with $r = 5, 10, 15, 20, 25$, we further observe that the LP method leads to smaller average absolute (resp. relative) errors than the learning based method.

Finally we want to compare the computational efficiency between the LP method and the full order method. As it is mentioned, both methods share the same subroutine to interpolate a computed solution from one mesh to another. The total number of the time steps are also the same for both methods. Therefore in Fig. 4.6(b), we only show the relative average CPU time (i.e. relative to that of the full order method) to invert the linear system for one step time marching. With $N_x = 1000$, it is observed that with 5 to 25 basis functions, it takes the LP method around 36% of the time of the full order method to invert the resulting linear system.
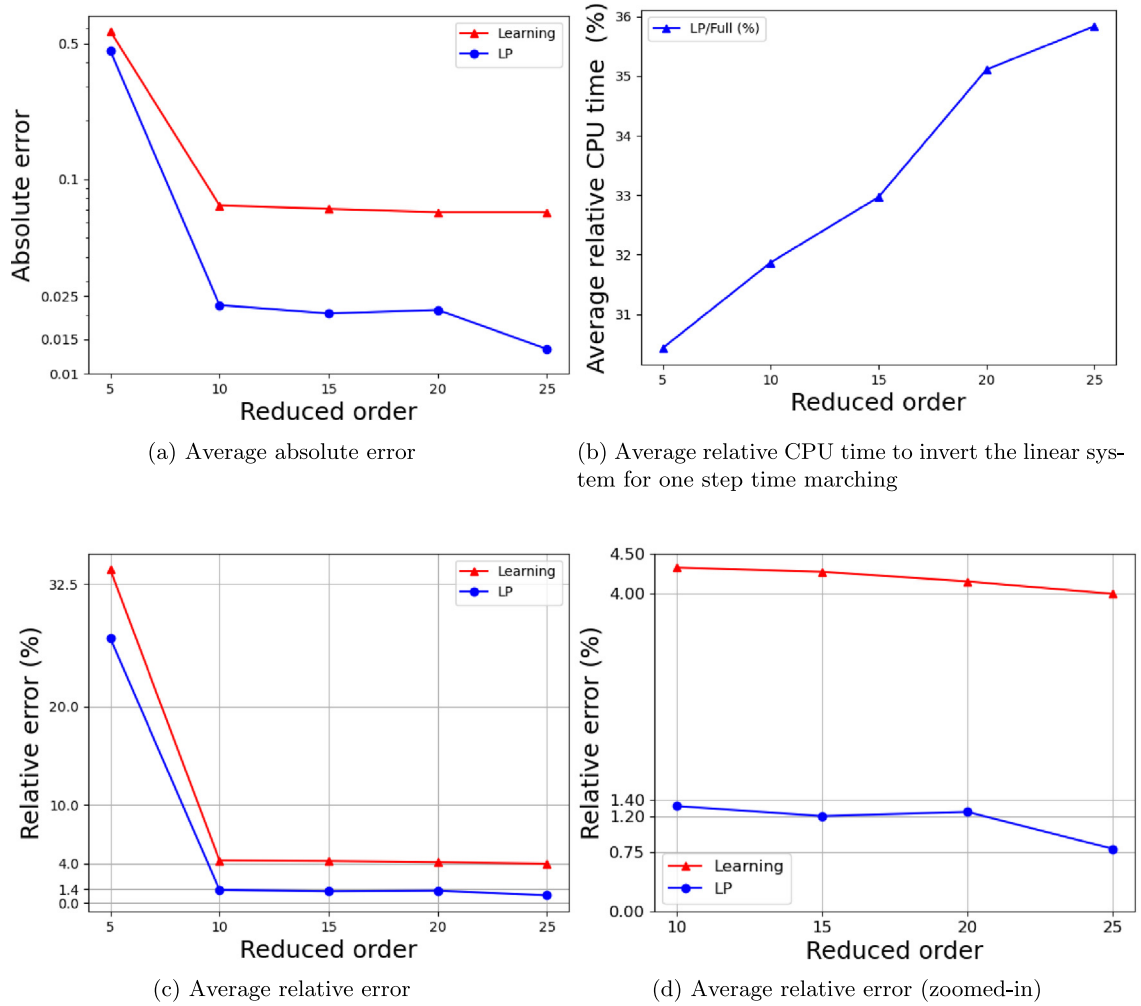
4.4. 2D linear advection equation

In this example, we consider a 2D linear advection equation:

$$u_t + b \cdot \nabla u = 0, \quad (x, y) \in [0, 1] \times [0, 1] \tag{4.10}$$

with $b = (1.0, 2.0)^T$ and the zero initial condition. The boundary condition is given at the inflow boundary $\Gamma_{\text{inflow}} = \{(x, y) \in [0, 1] \times [0, 1] : xy = 0\}$ as below,

$$u(x, y)|_{(x,y)\in\Gamma_{\text{inflow}}} = \begin{cases} 1 + \cos(8\pi x), & 0.125 < x < 0.625, \\ 0, & \text{otherwise.} \end{cases} \tag{4.11}$$

(a) Average absolute error

(b) Average relative CPU time to invert the linear system for one step time marching

(c) Average relative error

(d) Average relative error (zoomed-in)

Fig. 4.6. Average absolute/relative error and average relative CPU time to invert the linear system for one step time marching, versus the reduced order r for the moving mesh test in Section 4.3.
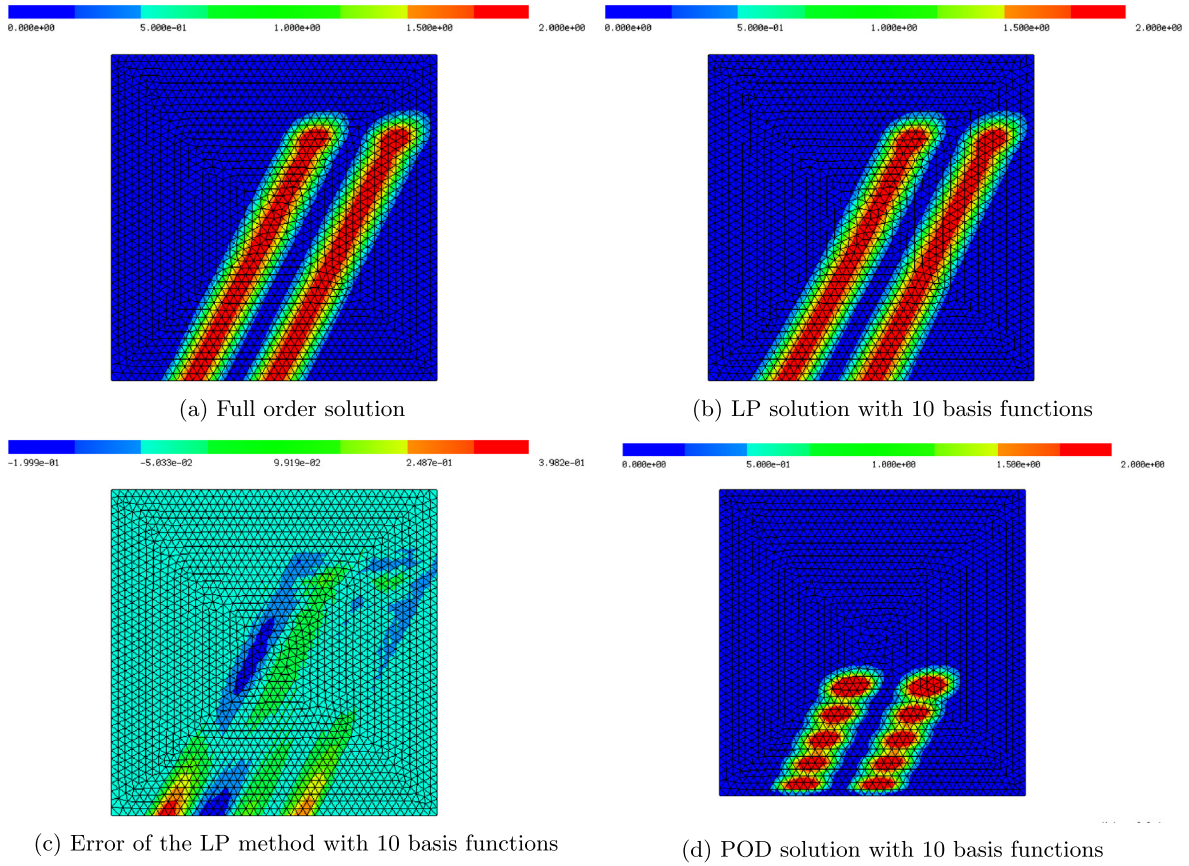
The computational domain is partitioned by a triangular mesh $\mathcal{T}_h = \{\mathcal{T}_i\}_{i=1}^{N_x}$. In space, we apply the upwind discontinuous Galerkin (DG) method: seeking $u_h \in V_h := \bigoplus_{\mathcal{T}_i \in \mathcal{T}_h} \mathcal{P}^1(\mathcal{T}_i)$ satisfying

$$\sum_{\mathcal{T}_i \in \mathcal{T}_h} \left(\int_{\mathcal{T}_i} \partial_t u_h v - (b \cdot \nabla v) u_h dx + \int_{\partial \mathcal{T}_i} \widehat{b \cdot \mathbf{n}_i u_h} v ds \right) = 0, \quad \forall v \in V_h. \tag{4.12}$$

Here, $\mathcal{P}^1(\mathcal{T}_i)$ is the linear polynomial space on $\mathcal{T}_i$, $\mathbf{n}_i$ is the outward unit normal of $\mathcal{T}_i$ along its boundary $\partial \mathcal{T}_i$, and $\widehat{b \cdot \mathbf{n}_i u_h}$ is the upwind flux that is based on the upwind mechanism and defined as $\widehat{b \cdot \mathbf{n}_i u_h} = b \cdot \mathbf{n}_i \lim_{\epsilon \to 0+} u_h(x - (b \cdot \mathbf{n}_i) \mathbf{n}_i \epsilon)$. In time, we apply the trapezoidal method that is second order accurate. The DG code is implemented with the python interface of the c++ finite element library NGSolve [61].

The mesh in our test consists of $N_x = 3710$ elements with the maximal edge length $h_{max} = 0.025$ (see Fig. 4.7(a)), and the time step size is set as $\Delta t = \frac{0.5 h_{max}}{\|b\|_2}$. The training data is the nodal values of the full order solution for $t \in [0, 0.2]$ (37 time snapshots) and we predict the solution from $t = 0.2$ to $t = 0.4$. We find it much easier to train the neural network using nodal values instead of modal coefficients as the degrees of freedom. Since NGSolve implements the DG method based on a modal basis instead of a nodal basis, this leads to extra cost to covert between the nodal values and modal coefficients of the solution.

In Fig. 4.7, we present the full order solution, the LP and POD solutions with $r = 10$ at $t = 0.4$, as well as the error in the LP solution. It is clear that the LP method matches the full order solution much better than the POD method. In Fig. 4.8, we present the absolute error at $t = 0.4$ for different methods and the relative CPU time of the LP method. We

(a) Full order solution

(b) LP solution with 10 basis functions

(c) Error of the LP method with 10 basis functions

(d) POD solution with 10 basis functions

Fig. 4.7. The full order solution, the LP solution and its error with 10 basis functions, and the POD solution with 10 basis functions, all at $t = 0.4$ for the 2D linear advection equation in Section 4.4.

observe that the LP method always leads to the smallest error, and it only needs less than 25% CPU time of the full order solve even with the extra online overhead to convert nodal values to modal coefficients.

We also use this example to illustrate that the proposed method is relatively insensitive to the width and the depth of the hidden layer of the neural network. In Fig. 4.9, we present the final value of the loss function and the prediction error of the trained neural network at $t = 0.2$ with different width and depth of the hidden layer. The two neural networks that represent the basis and the coefficients share the same width and the depth. These neural networks are also trained with the solution snapshots for $t \in [0, 2]$, and the learning rate is 10^{-3} and the number of epochs is 30. For each pair of depth and width, the value of the loss function and the prediction error are the average value of 5 different random initialization of the neural network. We observe that when the depth is not less than 2 and the width is not less than 10, the final value of the loss function and the prediction error of the trained neural network at $t = 0.2$ do not change much with respective to the depth and the width.

The last test for this example is to investigate the relation between the cost to generate the reduced basis online and the architecture of the neural networks. In Fig. 4.10, we compute the average CPU time of 100 runs to generate reduced basis for $t = 0.4$ and $r = 10$. In 4.10(a), we fix the depth of the hidden layer as 3 and vary the width of the hidden layer. In 4.10(b), we fix the width of the hidden layer as 25 and vary the depth. We observe that the computational cost of generating 10 reduced basis functions is roughly proportional to the width and the depth of the hidden layer. We also would like to point out that, as to be shown in Section 4.5, the computational time of generating the basis will be less than solving the reduced order projection equation when the mesh is refined enough.

4.5. 1D linear advection equation with (x, μ)-dependent inhomogeneous media

In this example, we consider the following 1D linear advection equation

$$u_t + (c(x; \mu)u)_x = 0 \tag{4.13}$$
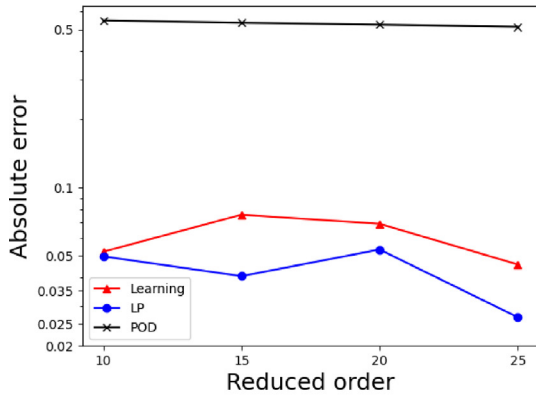
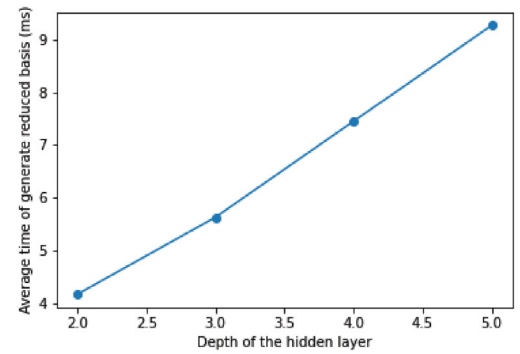(a) Absolute error at $t = 0.4$ (b) Average relative CPU time

Fig. 4.8. The absolute error at $t = 0.4$ and the average relative CPU time for the 2D linear advection equation in Section 4.4.
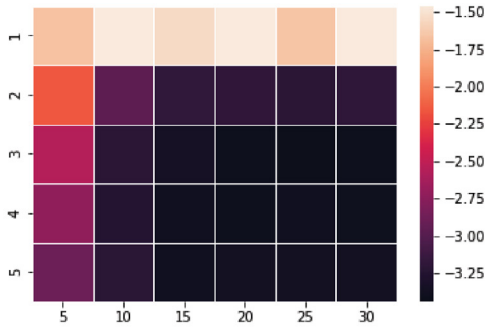
(a) Final value of the loss function of trained neural network
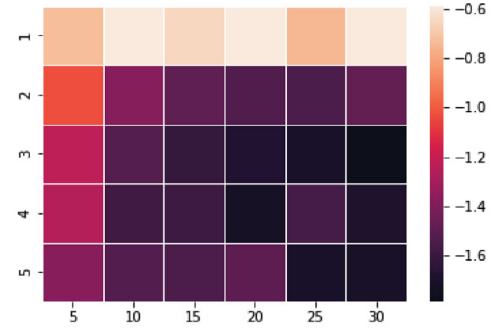
(b) The L_2 prediction error of the trained neural network at $t = 0.2$ under log-scale

Fig. 4.9. x-axis: width of the hidden layers; y-axis: depth of the hidden layers; color: the value of the loss function/error under log scale.
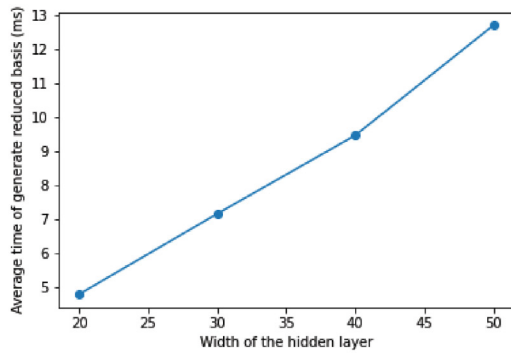
(a) Depth of the hidden layer, 3   (b) Width of the hidden layer, 25

Fig. 4.10. x-axis: width/depth of the hidden layers; y-axis: average CPU time to generate 10 reduced basis functions online.

in an inhomogeneous media where the propagation speed $c(x, \mu)$ is (x, μ)-dependent, given as

$$c(x; \mu) = 1.25 + \mu_1 \sin(\mu_2 \pi x), \quad \mu = (\mu_1, \mu_2) \in X_{\mathcal{P}} = [0, 0.5] \times [2, 4]. \tag{4.14}$$

The initial condition is

$$u(x, 0) = \begin{cases} 0.5 + 0.5 \cos(5\pi(x - 0.25)), & 0.05 \leq x \leq 0.45, \\ 0, & \text{otherwise}, \end{cases} \tag{4.15}$$

together with the zero boundary conditions. The spatial computational domain is $[0, 2.5]$, and a uniform mesh with 500 elements is used. To control the numerical dissipation of the full order model, we use the second-order Crank–Nicolson time discretization, and the numerical flux is chosen as an upwind-biased flux:

$$f_{j+\frac{1}{2}}^n = \frac{3}{4} c_j u_j^n + \frac{1}{4} c_{j+1} u_{j+1}^n \tag{4.16}$$

with $c_j = c(x_j; \mu) > 0$. The time step size is $\Delta t = 2\Delta x$.

Associated with the model parameter $\mu = (\mu_1, \mu_2)$, the training set is taken to be $\{\mu_1^{(i)} = \frac{i}{20}\}_{i=0}^{10} \times \{\mu_2^{(j)} = 2 + \frac{j}{10}\}_{j=0}^{20}$, and the testing set is constituted by 20 pairs of (μ_1, μ_2) randomly sampled from a uniform distribution on the parameter space $X_{\mathcal{P}}$. For both the training and the testing, $t \in [0, 1]$ (231×101 time snapshots) is considered.

In Fig. 4.11, we report the average relative testing error of the proposed LP method, the learning method, and the standard POD method versus the reduced order r. We observe that the LP method consistently entails the least error for varied r. When $r = 25$, the error of the LP method is smaller than 0.3%. Possibly due to the overfitting, the error of the pure learning-based method does not decay as the number of basis functions increases. The projection step in the online stage reduces the generalization error dramatically, as the PDE information is utilized through this step.

For the testing sample $(\mu_1, \mu_2) = (0.23, 3.56)$, we present in Fig. 4.12 the solutions with different reduced order $r = 15, 25$ at $t = 2$. With 15 basis functions, the LP solution matches the full order solution better than the POD method, though some non-physical oscillation exists. Additional study can be found in Appendix to understand the origin of the oscillation. With 25 basis functions, the LP solution visually coincides with the full order solution.

Our next test aims at showing the flexibility of the LP method when the online spatial meshes are different from the offline ones. Our method avoids extra efforts which are needed by conventional projection based ROMs to accomplish similar tasks [48–50]. Using the neural network trained with data generated on a uniform mesh with $N_x^{\text{off}} = 500$ spatial elements offline, we evaluate basis function values on uniform meshes with different N_x^{on} in the online computation. The relative error for different grid resolutions online with $r = 20$ is presented in the left picture of Fig. 4.13, and it is reasonably small. In the right picture of Fig. 4.13, we present the predicted solution at $\mu = (0.23, 3.56)$ and $t = 1$ with $N_x^{\text{on}} = 2500$. The full order solution with $N_x^{\text{off}} = 500$ is more dissipative than the full order solution with $N_x^{\text{on}} = 2500$, but the LP solution still matches the full order solution with $N_x^{\text{on}} = 2500$ well, while the pure learning-based method provides a more dissipative solution compared with the full order solution.

The final test we carry out is intended to examine the computational efficiency. In Fig. 4.14, for the test sample $(\mu_1, \mu_2) = (0.23, 3.56)$, we compare the CPU time of the one step online computation of the LP method with 20 basis functions and that of the full order model for different grid resolutions online. We also compare the times for basis evaluation in (3.14) and for solving the projection Eq. (3.15) during the online stage of the LP method (see Algorithm 1). For different resolutions online (N_x^{on}), the reduced-order basis is always trained with $N_x^{\text{off}} = 500$ elements in the offline stage, and the solutions predicted by our method always visually match the full order solution reasonably well. As N_x^{on} grows, the CPU time of the basis evaluation grows much slower compared with that for solving the projection equation. For small N_x^{on}, the ROM spends most of the time on the basis generation. As N_x^{on} grows, the computational time of the full order model grows much faster than that of the proposed LP method.

### 4.6. 1D Burgers equation

We consider the Burgers equation:

$$u_t + \left(\frac{1}{2}u^2\right)_x = 0, \tag{4.17}$$

and perform two sets of experiments. In the first one, we consider a Riemann problem with a parameter in the initial data, and the solutions will be predicted for a test set of parameter and for future times. In the second experiment, we start with a smooth initial profile, and shock discontinuity will form later. We will predict the solutions at future times based on the full order solution on $[0, t_0]$, over which the shock structure may or may not have been "seen" by the offline training algorithm. The first order method in (4.3) will be applied, with the numerical flux chosen as:

$$f_{j+\frac{1}{2}}^n := \frac{1}{2}\left(f(u_{j+1}^n) + f(u_j^n)\right) - \frac{\Delta x}{2\Delta t}(u_{j+1}^n - u_j^n). \tag{4.18}$$
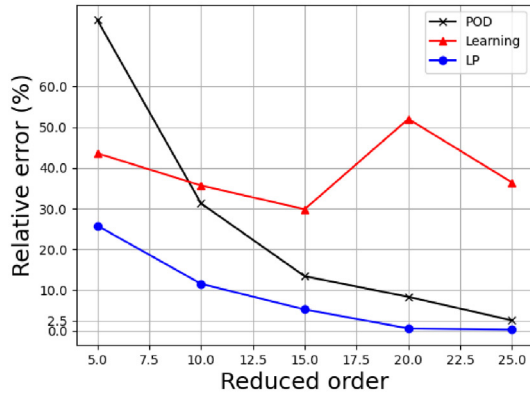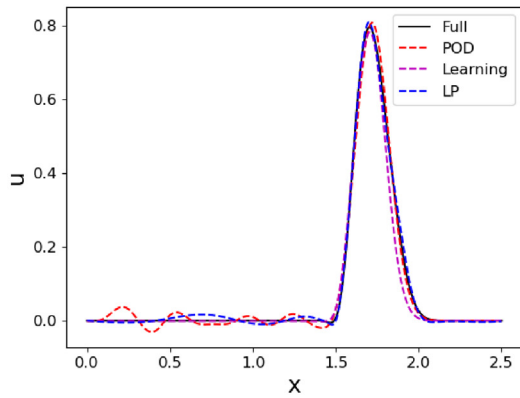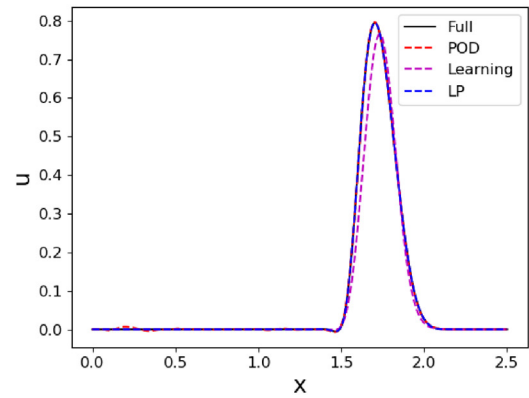
(a) Average relative error
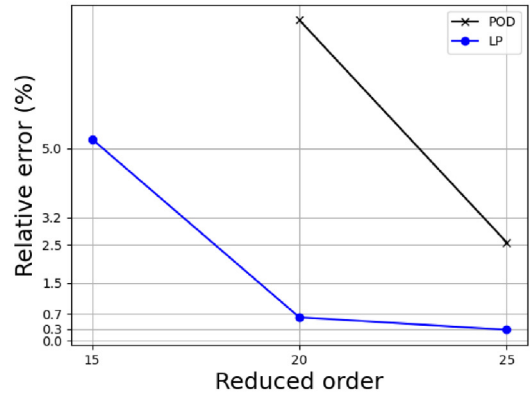
(b) Zoomed-in picture

Fig. 4.11. *x*-axis: reduced order *r*; *y*-axis: the average relative error for the test set of the linear advection equation with (x, μ)-dependent inhomogeneous media in 4.5.
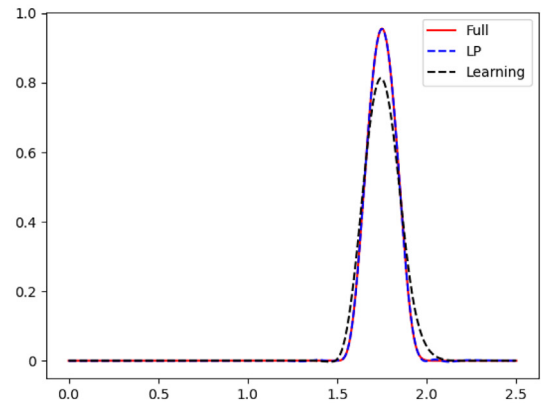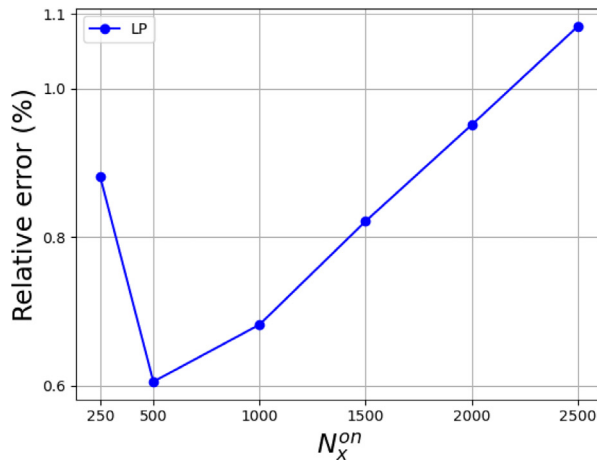

(a) 15 basis functions

(b) 25 basis functions

Fig. 4.12. Predicted solutions of the linear advection equation with (x, μ)-dependent inhomogeneous media in Section 4.5 at $(\mu_1, \mu_2) = (0.23, 3.56)$ and $t = 1$.

Fig. 4.13. Linear advection equation with (x, μ)-dependent inhomogeneous media in Section 4.5. Left: mean relative error of the test set for different online grid resolutions (N_x^{on}). Right: predicted solution at $\mu = (0.23, 3.56)$ and $t = 1$ with online mesh $N_x^{on} = 2500$. 20 basis functions are used and are trained offline with data subject to constant resolution ($N_x^{off} = 500$).
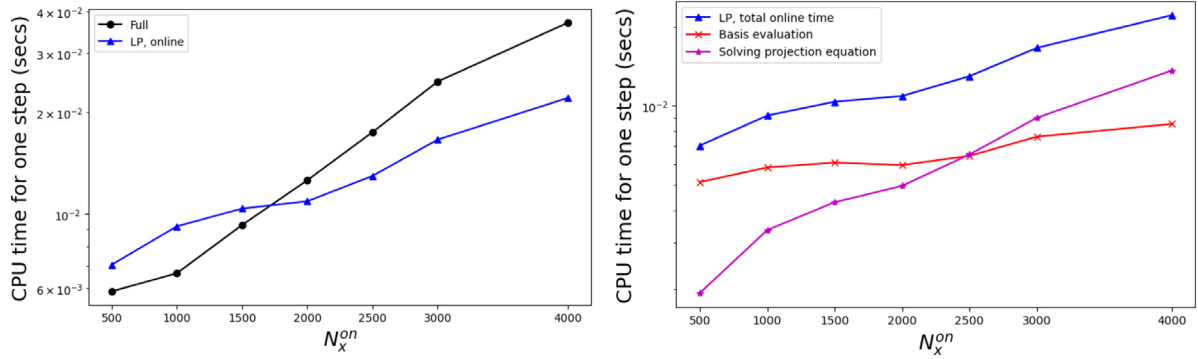
Fig. 4.14. The average CPU time for one time step as a function of different online resolution N_x^{on}. Left: full order model v.s. the online computation of the LP method. Right: online basis evaluation, the online solve of projection equation of the LP method and the total time of the online computation of the LP method.

4.6.1. Parameter-dependent Riemann problem

We consider the computational domain [0, 8] with a parameter-dependent initial condition:

$$u(x, 0) = \begin{cases} 1. + \mu, & x \in [0.5, 0.75], \\ 0., & \text{otherwise.} \end{cases} \tag{4.19}$$

Here, $\mu \in X_{\mathcal{P}} = [0, 1]$. The solution has a rarefaction fan and a moving shock. We use a uniform mesh with $N_x = 400$. The time step size is set to be $\Delta t = \Delta x$.

The training set is taken as $\{0, 0.1, 0.2, \ldots, 1\} \subset X_{\mathcal{P}}$ with $0 \le t \le 0.25$ (11 × 101 snapshots). The test set is taken as $\{0.05 \pm 0.01, 0.15 \pm 0.01, \ldots, 0.95 \pm 0.01\} \subset X_{\mathcal{P}}$, and we predict the solution from $t = 0$ to $t = 0.5$. In the offline stage of this problem, we train the neural network with 150 epochs.

In Fig. 4.15, the average relative errors for the test set are presented corresponding to different reduced order r. Overall, we observe that for both the POD method and the LP method, the error decays as r grows. However, this is not the case for the purely learning-based method. Moreover, the error of the LP method is much smaller than that of the other two methods.

In Fig. 4.16, we present the predicted solutions for $\mu = 0.76$ at $t = 0.5$ with $r = 15, 25$. We observe that the LP method performs better than the other two methods in matching the full order solution, especially in capturing the shock location. There are small oscillations near the shock location in the LP solution .

In Fig. 4.17, we present the time evolution of the L_2 error in prediction for $\mu = 0.76$ under the log-scale with $r = 15$ and 25. Among the LP method, the learning method, and the POD method, the LP method displays the slowest growth in its error over time. When $r = 25$, we observe that the error of the POD method is smaller than that of the LP method for $t \le 0.25$, then it grows dramatically fast and becomes much larger than the LP method. The reason why the POD method becomes much worse after $t = 0.25$ is that we only have the training data for $t \le 0.25$ in the offline stage. The LP method shows its advantages in providing more accurate and stable long time prediction.

In Fig. 4.18, we present the relative CPU time with different number of mesh elements in the online stage N_x^{on} and $r = 20$. The neural network is trained with $N_x^{off} = 400$. When the mesh is refined enough, the LP method does lead to smaller computational time compared with a full order solver. However, when the mesh is refined with larger N_x^{on}, the LP method produces more numerical oscillations. It becomes important on much refined meshes to control non-physical oscillations of reduced order solutions and to improve numerical stability.

### 4.6.2. Future prediction with smooth initial profile and shock formation

We consider the computational domain $[-1, 1]$ and the following initial condition

$$u(x, 0) = \frac{1}{4} + \frac{1}{2}\sin(\pi x) \tag{4.20}$$

with periodic boundary conditions. Though with a smooth initial profile, a shock will form at $t = t_\star = \frac{2}{\pi}$ and then moves as time evolves. We use a uniform mesh with $N_x = 400$ and the time step size $\Delta t = 0.25\Delta x$.

We consider two training sets, which are the full order numerical solutions for $t \in [0, t_0]$, with $t_0 = 1.1$ (when shock has formed with 881 snapshots in the training set) and $t_0 = 0.6$ (when shock has not formed yet with 481 snapshots in the training set), respectively. We predict the solution from $t = t_0$ to $t = 1.6$. The neural network is trained with 100 epochs. It is expected that the experiment with the second training data which has not "seen" any shock to be more challenging in future prediction.

In Fig. 4.19, we present the predicted reduced-order solutions at $t = 1.6$. We observe that, with both $t_0 = 1.1$ and $t_0 = 0.6$, the solutions by the LP method always match the full order solution the best. Even though the training data
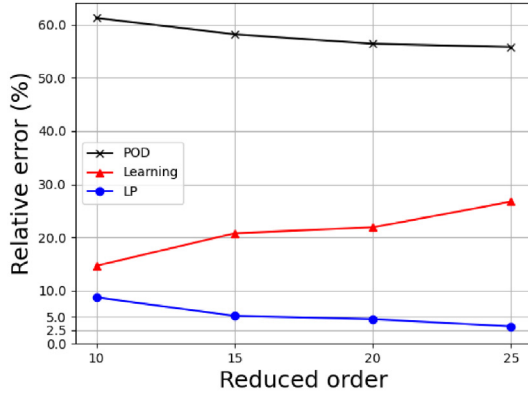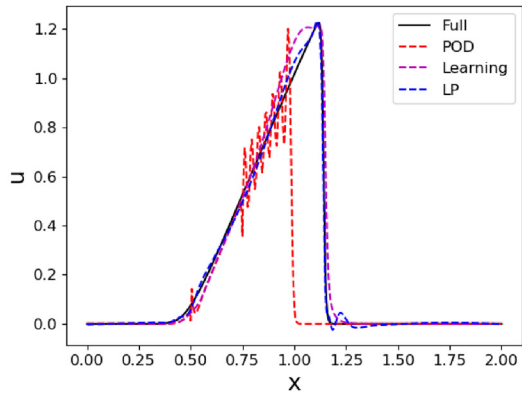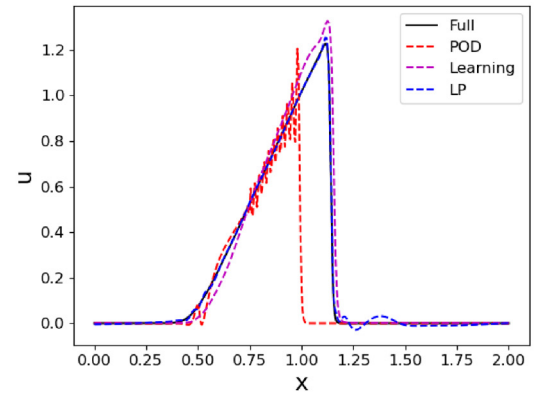
(a) Average relative error
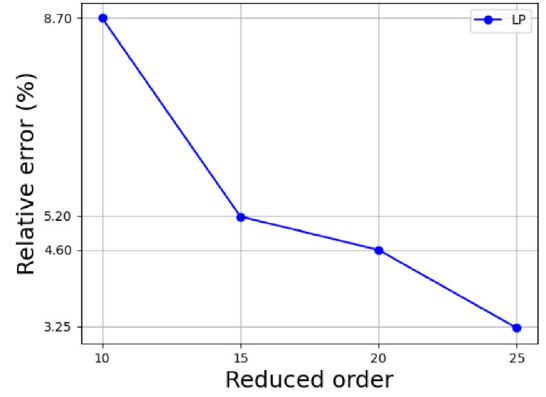
(b) Zoomed-in picture

Fig. 4.15. *x*-axis: reduced order *r*; *y*-axis: the average relative error for the test set of the Burgers equation (parameter dependent initial profile) in Section 4.6.1.
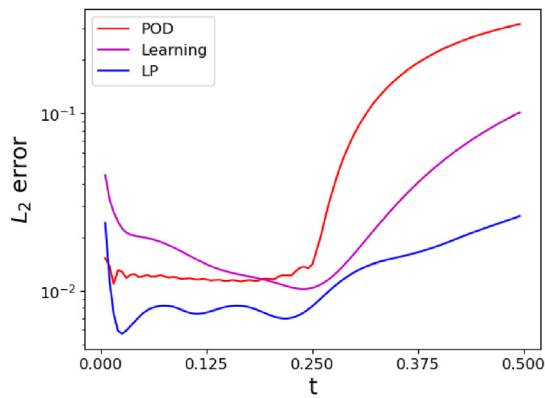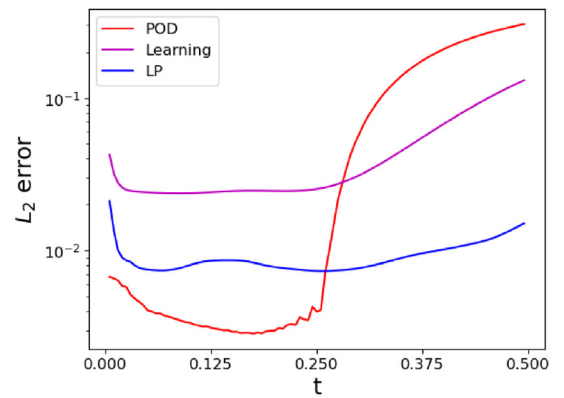

(a) 15 basis functions

(b) 25 basis functions

Fig. 4.16. Predicted solutions for the Burgers equation (parameter dependent initial profile) in Section 4.6.1 with $\mu = 0.76$ and $t = 0.5$.
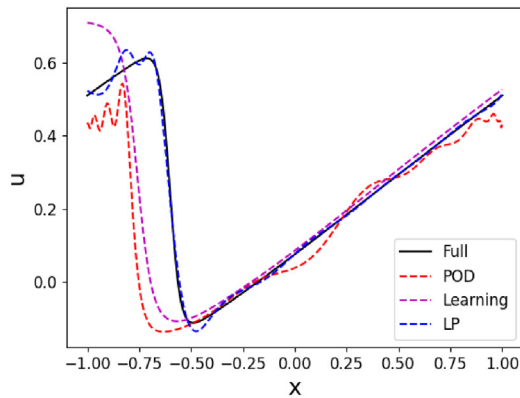

(a) 15 basis functions

(b) 25 basis functions

Fig. 4.17. The time evolution of the prediction error history for the Burgers equation (parameter dependent initial profile) in Section 4.6.1 with $\mu = 0.76$.
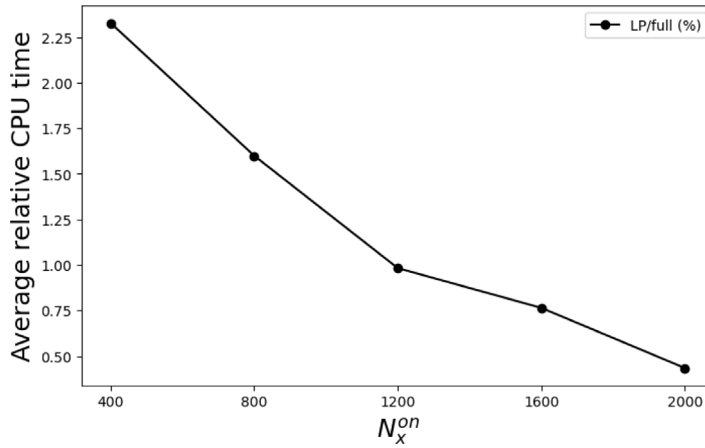
Fig. 4.18. Relative computational time with different resolutions in the online stage.
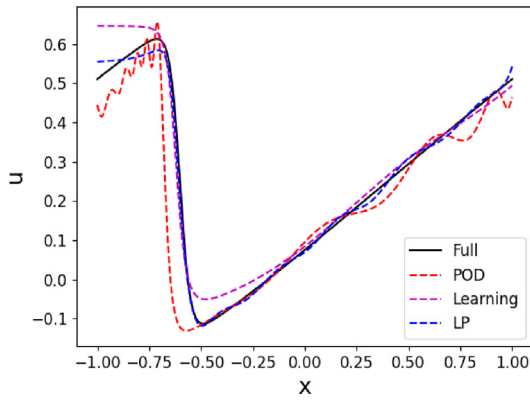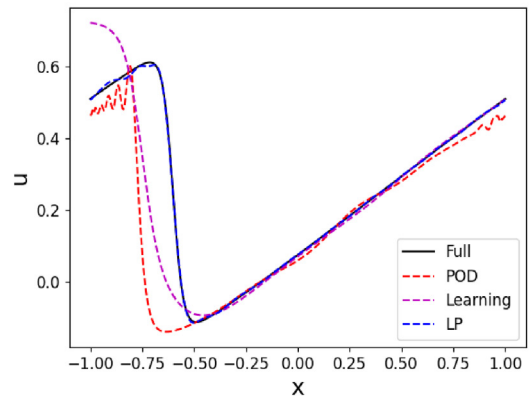

(a) 15 basis functions; training data: $t \in [0, 1.1]$
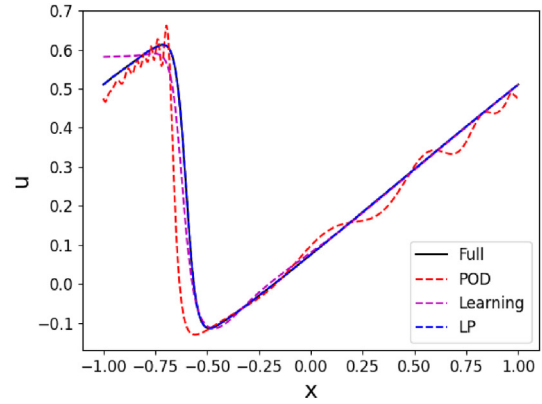
(b) 25 basis functions; training data: $t \in [0, 1.1]$

(c) 15 basis functions; training data: $t \in [0, 0.6]$

(d) 25 basis functions; training data: $t \in [0, 0.6]$

Fig. 4.19. Predicted solutions of the Burgers equation with smooth initial profile and shock formation in Section 4.6.2 at $t = 1.6$.

corresponding to $t \in [0, 0.6]$ does not include any solutions with shock features, the LP method captures the correct shock location in the predicted solutions. Besides, with the training data from $t \in [0, 0.6]$, the results of the LP method are relatively more oscillatory. The numerical oscillation is further suppressed with an increase of the reduced order r.
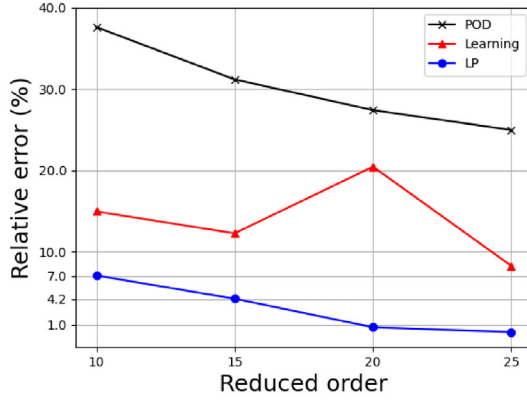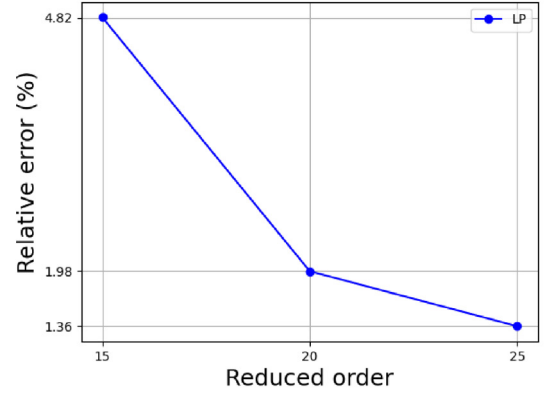
(a) Comparison between pure learning, POD and the LP method, training data: $t \in [0, 1.1]$
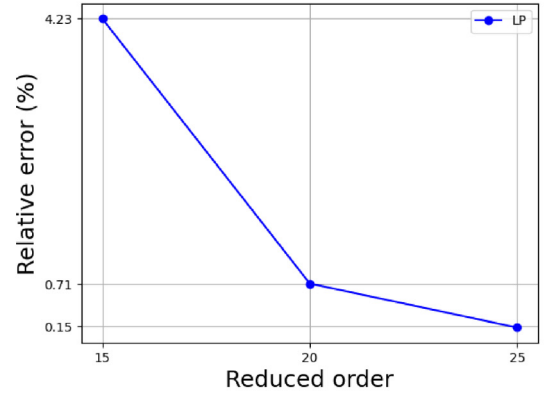
(b) Zoomed-in picture, training data: $t \in [0, 1.1]$
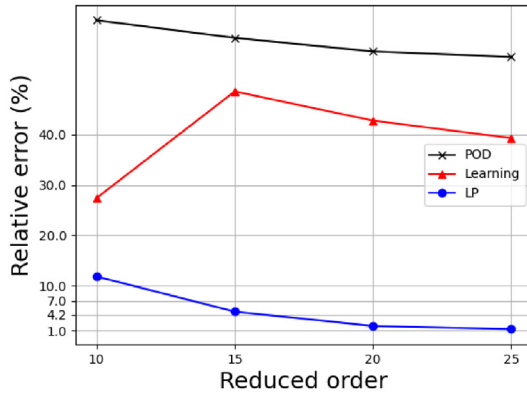
(c) Comparison between pure learning, POD and the LP method, training data: $t \in [0, 0.6]$

(d) Zoomed-in picture, training data: $t \in [0, 0.6]$

Fig. 4.20. *x*-axis: reduced order r; *y*-axis: the largest relative prediction error for the Burgers equation with smooth initial profile and shock formation in Section 4.6.2.

In Fig. 4.20, we present the largest relative prediction error from $t = t_0$ to $t = 1.6$ when the ROM is trained with the data sampled from $t = [0, t_0]$ for $t_0 = 1.1$ or $t_0 = 0.6$ versus the reduced order r. For both cases, the LP method always has the smallest relative error. When trained with $t_0 = 1.1$, the relative error of the LP method is approximately 0.71% with 20 basis functions and 0.15% with 25 basis functions. For the same number of reduced basis, the relative error of the LP method trained with $t_0 = 1.1$ is smaller than that with $t_0 = 0.6$.
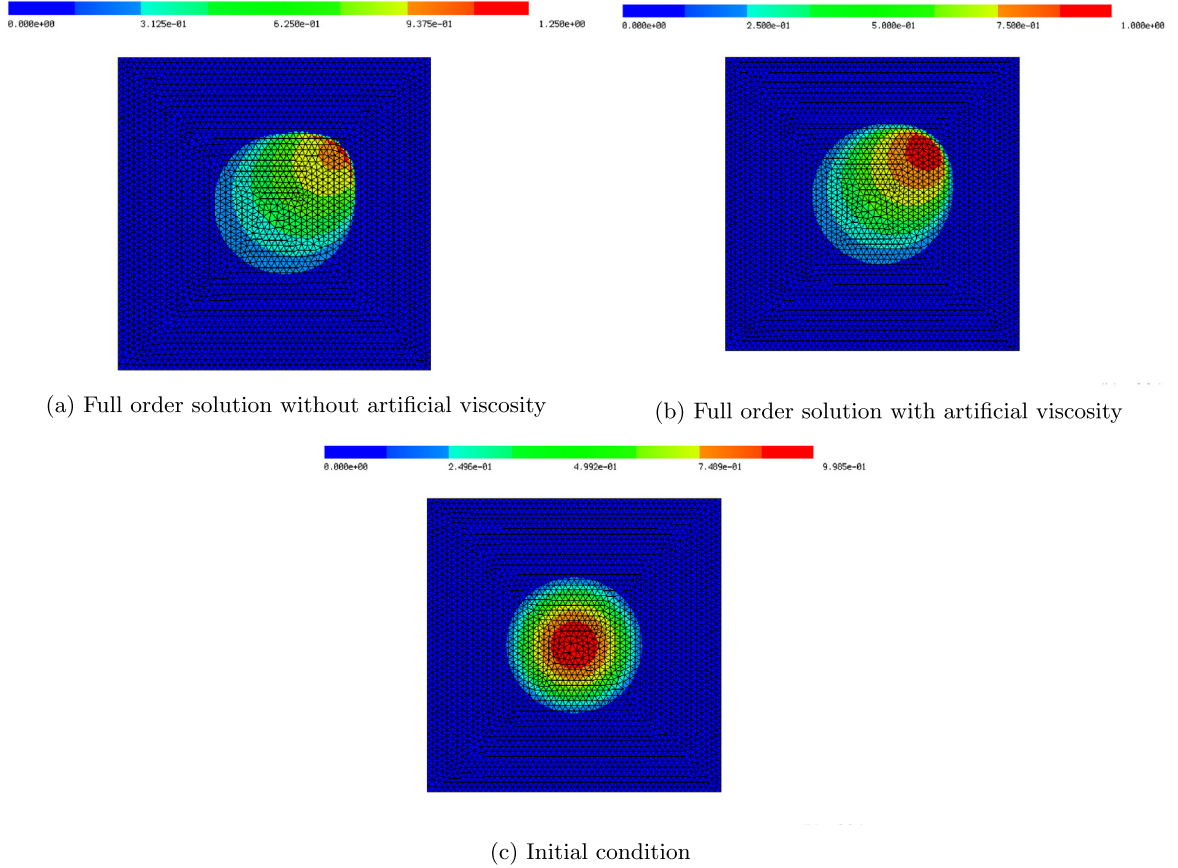
4.7. 2D Burgers equation

We consider a 2D Burgers equation:

$$u_t + \nabla \cdot \mathbf{F}(u) = 0, \quad (x, y) \in [-1, 1] \times [-1, 1], \tag{4.21}$$

with $\mathbf{F}(u) = (\frac{u^2}{2}, \frac{u^2}{2})^T$ and the following initial condition (also see 4.21(c)),

$$u(x, 0) = \begin{cases} \sin\left(\pi(0.5 - \sqrt{x^2 + y^2})\right), & \sqrt{x^2 + y^2} \le 0.5, \\ 0, & \text{otherwise.} \end{cases} \tag{4.22}$$

(a) Full order solution without artificial viscosity

(b) Full order solution with artificial viscosity

(c) Initial condition

Fig. 4.21. Initial condition, and full order solutions with and without artificial viscosity at $t = 0.42$ for the 2D Burgers equation in Section 4.7.

The computational domain is partitioned by a triangular mesh $\mathcal{T}_h = \{\mathcal{T}_i\}_{i=1}^{N_x}$, with $N_x = 5822$ and the maximal edge length $h_{\max} = 0.04$. In space, we apply a DG method: seeking $u_h \in V_h := \bigoplus_{\mathcal{T}_i \in \mathcal{T}_h} \mathcal{P}^1(\mathcal{T}_i)$ satisfying
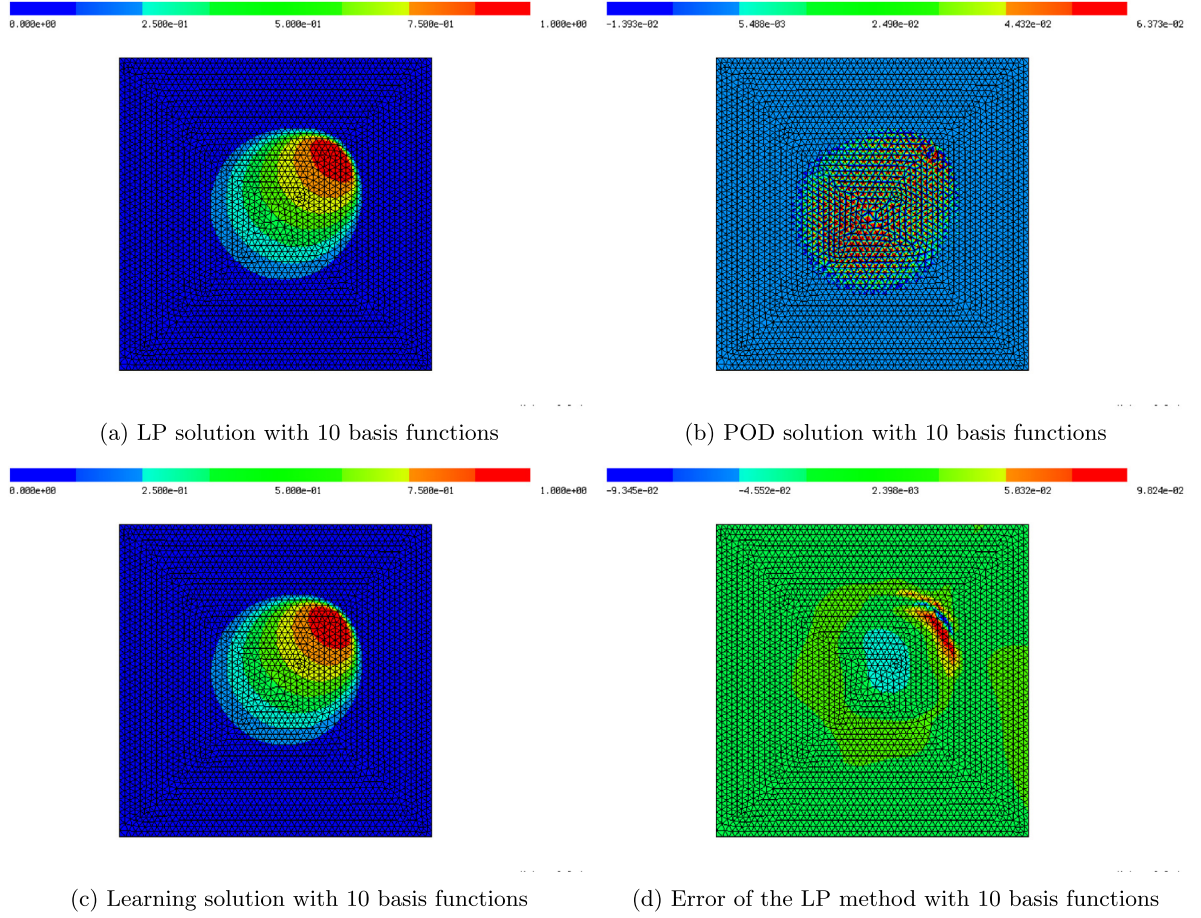
$$\sum_{\mathcal{T}_i \in \mathcal{T}_h} \left(\int_{\mathcal{T}_i} (\partial_t u_h v - \mathbf{F}(u_h) \cdot \nabla v) dx + \int_{\partial \mathcal{T}_i} \widehat{\mathbf{F} \cdot \mathbf{n}_i} v ds \right) = 0, \quad \forall v \in V_h. \tag{4.23}$$

Here we use the local Lax–Friedrichs flux

$$\widehat{\mathbf{F} \cdot \mathbf{n}_i} = \frac{\mathbf{F}(u_h^+) + \mathbf{F}(u_h^-)}{2} \cdot \mathbf{n}_i + \frac{\lambda_i}{2}(u_h^+ - u_h^-), \tag{4.24}$$

where $\mathbf{n}_i$ is the outward unit normal of $\mathcal{T}_i$ along its boundary $\partial \mathcal{T}_i$, u_h^+ is the value of u_h on $\partial \mathcal{T}_i$ from the interior of $\mathcal{T}_i$, u_h^- is the value of u_h on $\partial \mathcal{T}_i$ from the neighboring element of $\mathcal{T}_i$ sharing the same edge, and $\lambda_i = |[1, 1]^T \cdot \mathbf{n}_i| \max(|u_h^+|, |u_h^-|)$. In time, we apply the forward Euler method with the time step size $\Delta t = 0.001$. To stabilize both the full and reduced order solvers, an artificial viscosity term $\frac{1}{8} h \Delta u$ is added to the equation in an element-wise fashion (with $h|_{\mathcal{T}_i}$ as the element diameter), and is discretized by the symmetric interior penalty method [62]. The training data is the solution snapshots from $t = 0.2$ to $t = 0.35$ (151 time snapshots), and we predict the solution at $t = 0.42$. The coefficient neural network has 2 hidden layers with 25 neurons per layer, and the basis neural network has 3 layers with 25 neurons per layer. In Fig. 4.21, we present the full order solution with and without artificial viscosity. Without artificial viscosity, one can see the non-physical oscillations near the wave front in the full order solution, and the LP method produces even more oscillatory solutions.

As mentioned before, there will be no saving of computational cost with the forward Euler time integrator. The purpose of this test is to check whether the neural network is capable of capturing the underlying low-rank structure in 2D nonlinear hyperbolic problems. The reduced order solutions with 10 basis functions at $t = 0.42$ are presented in Fig. 4.22, all with the artificial viscosity strategy applied. The LP method and the pure learning-based method match the full order solution (in Fig. 4.22(a)) reasonably well, while the POD method with the same number of basis functions fails to capture the solution structure when going beyond the training set. In Fig. 4.23, the absolute errors at $t = 0.42$ versus the reduced

(a) LP solution with 10 basis functions

(b) POD solution with 10 basis functions

(c) Learning solution with 10 basis functions

(d) Error of the LP method with 10 basis functions

Fig. 4.22. Full order and reduced order solutions and error at $t = 0.42$ for the 2D Burgers equation in Section 4.7.
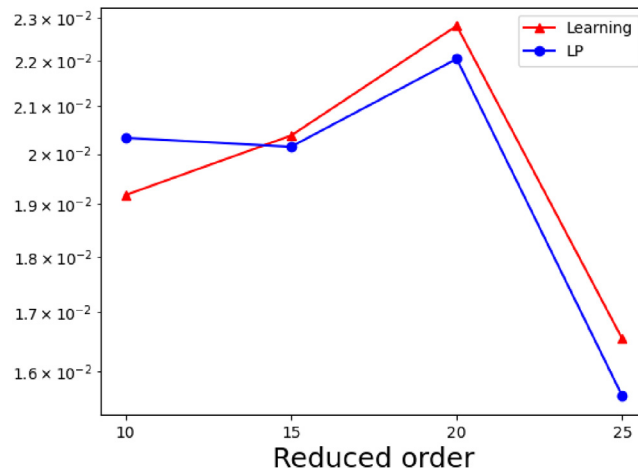
(a) Absolute error at $t = 0.42$

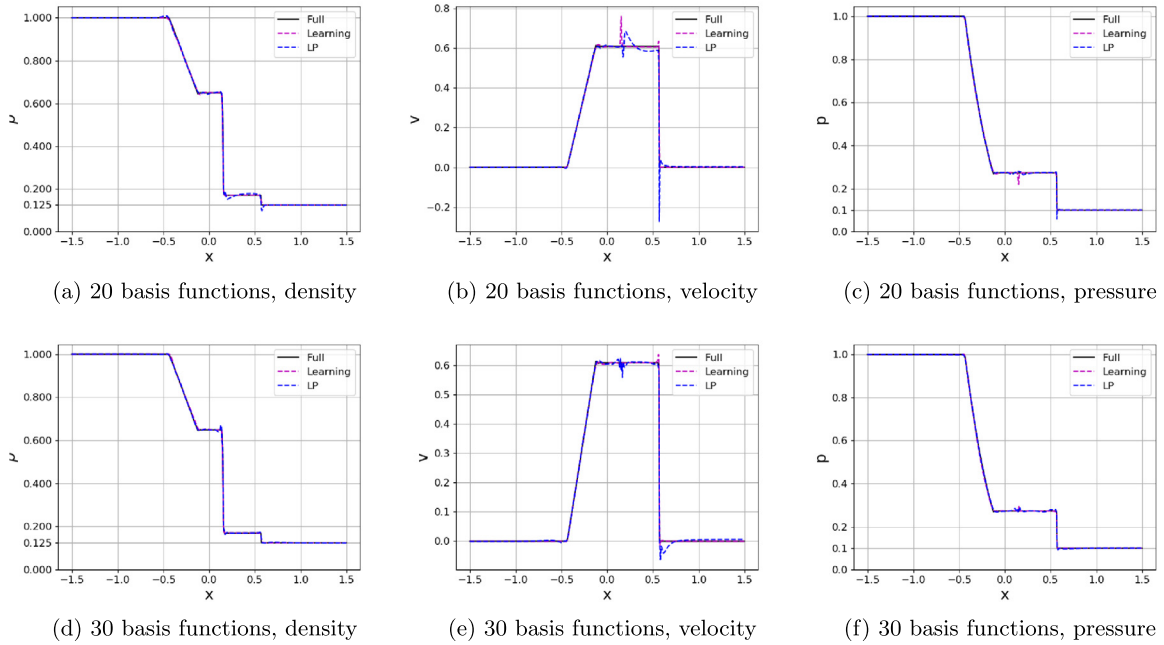**Fig. 4.23.** The absolute error at $t = 0.42$ for the 2D Bureges equation in Section 4.7.

(a) 20 basis functions, density

(b) 20 basis functions, velocity

(c) 20 basis functions, pressure

(d) 30 basis functions, density

(e) 30 basis functions, velocity

(f) 30 basis functions, pressure

Fig. 4.24. Predicted solutions for the Euler equation in Section 4.8 at $t = 0.25$.

order r are presented for the LP and the pure learning methods. The errors of the two method are close to each other, but the online stage of the pure learning method is faster, as it only needs function evaluation.

4.8. Compressible Euler system

Finally we apply the proposed LP method to the 1D compressible Euler system of gas dynamics:

$$\begin{cases} \rho_t + (\rho u)_x = 0, \\ (\rho u)_t + (\rho u^2 + p)_x = 0, \\ E_t + (u(E + p))_x = 0. \end{cases} \tag{4.25}$$

Here ρ, u, p are the density, velocity, pressure, respectively, with the total energy $E = \frac{p}{\gamma-1} + \frac{1}{2}\rho u^2$ and $\gamma = 3$ for the ideal gas. We consider the Sod's shock tube problem, with the initial conditions:

$$\rho(x, 0) = \begin{cases} 1.0, & x < 0, \\ 0.125, & x \geq 0, \end{cases} \qquad u(x, 0) = 0.0, \quad p(x, 0) = \begin{cases} 1.0, & x < 0, \\ 0.1, & x \geq 0. \end{cases} \tag{4.26}$$

For this example, the full order method is based on the fifth-order WENO finite difference method with the Lax–Friedrichs flux [63] and the explicit third order strong stability preserving Runge–Kutta method [64]. With an explicit time integrator applied, the LP method will not reduce the computational time. We want to use this example to demonstrate that the proposed method can capture the hidden low-rank structure of nonlinear hyperbolic systems.

We consider the computational domain $[-1.5, 1.5]$, and it is partitioned by a uniform mesh with $N_x = 1500$ elements. The CFL number is taken as 0.6. The training data are the solutions for $t \in [0, 0.2]$, and we use ROMs to predict solutions for $t \in [0.2, 0.25]$. In the offline stage of the LP method, we train three neural networks $\rho_{\mathcal{NN}}$, $(\rho u)_{\mathcal{NN}}$ and $E_{\mathcal{NN}}$, with each using the same architecture as we specify previously. The basis blocks of these neural networks further provide t-dependent reduced basis for ρ, ρu and E respectively. 100 epochs of training are conducted in the offline stage.

In Fig. 4.24, we present the density, velocity, and pressure of the reduced order solutions at $t = 0.25$ by the LP and the pure learning methods with $r = 20, 30$. The solution includes a rarefaction wave, a contact discontinuity, and a shock. Overall, the predicted solutions by both methods capture these wave structures well. Numerical oscillations are observed near the rightward moving contact discontinuity and the shock, and they are further suppressed when the reduced order r increases.
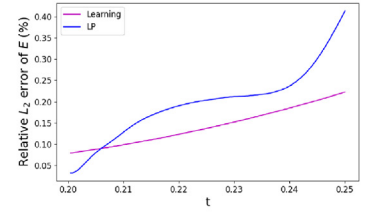
In Fig. 4.25, we plot the time evolution of the relative error by the LP and the pure learning methods in prediction from $t = 0.2$ to $t = 0.25$, and the errors are reasonably small. Compared with examples in previous sections, the error of the LP method grows faster in this example. We believe this is due to the insufficient control of numerical oscillations
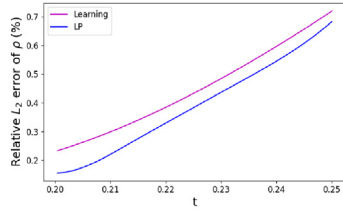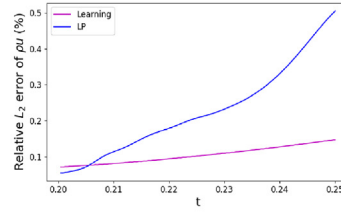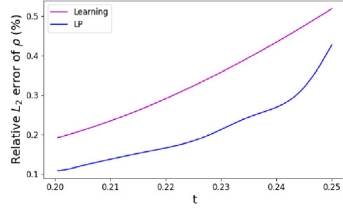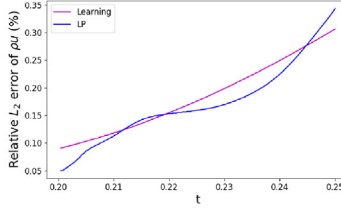
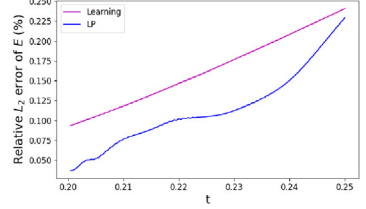(a) 20 basis functions, density   (b) 20 basis functions, velocity   (c) 20 basis functions, pressure

(d) 30 basis, density   (e) 30 basis functions, velocity   (f) 30 basis functions, pressure

Fig. 4.25. The evolution of the prediction relative error of the Euler equation in Section 4.8 from $t = 0.2$ to $t = 0.25$.

in the online projection step (3.15). We also want to mention that with 10 basis functions, the LP method suffers from negative pressure after $t = 0.2315$ due to the numerical oscillation.

Note that the results of the POD method are not reported in this example. In fact, when $r = 20, 30$, the online computation with the POD basis fails due to the presence of non-physical negative pressure. Indeed, before this phenomenon occurs, the POD results are more oscillatory than that of the other two methods. It is an important task to explore how to effectively control numerical oscillation and to preserve positivity of certain physical quantities for ROMs.

5. Conclusions

Following the offline–online decomposition, we propose a learning-based projection method for transport problems.
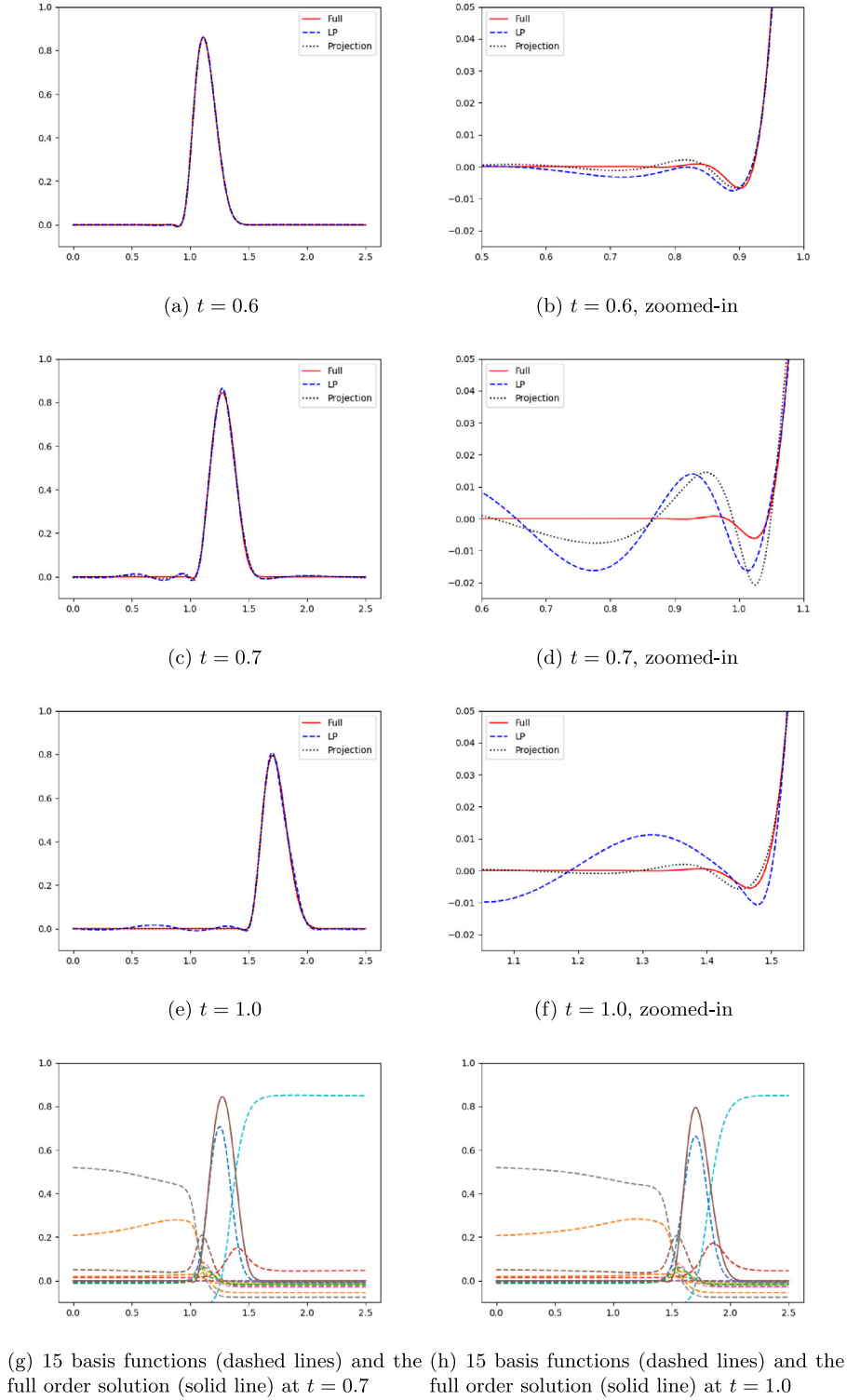
As a combination of deep learning and the projection-based method, the proposed method has the following advantages:

1. Compared with the standard POD methods, it achieves smaller long-time prediction error with the same number of basis functions.
2. Compared with a pure learning-based method using the same neural network, the online projection step may reduce the generalization error dramatically.
3. Unlike previous learning-based projection methods [40–42], there is no need to take derivatives of neural networks in the online projection step.
4. With x as an input of the neural network in a mesh free manner, it is flexible and simple for the proposed method to work with different meshes and even different schemes in the online and offline stages.

However, for problems with strong shocks, the projection based online stage may produce more numerical oscillation compared with a pure learning-based online stage.

There are still many aspects in this topic that could be further explored.

1. To design an efficient and robust hyper-reduction strategy to save the computational cost with explicit time integrators.
2. The effective and efficient control of numerical oscillations is crucial, especially when combined with hyper-reduction.
3. To apply and test the proposed method with more complicated 2D and 3D system hyperbolic conservation laws.
4. To further combine the proposed method with the adaptive moving mesh and the h, p-adaptive numerical methods which move or regenerate computational meshes for different time and/or parameter samples.
5. One drawback of the proposed method is that the learned basis is not orthonormal. To improve the robustness associated with this, one may introduce a regularization term similar to [65] in the loss function.

(a) $t = 0.6$

(b) $t = 0.6$, zoomed-in

(c) $t = 0.7$

(d) $t = 0.7$, zoomed-in

(e) $t = 1.0$

(f) $t = 1.0$, zoomed-in

(g) 15 basis functions (dashed lines) and the full order solution (solid line) at $t = 0.7$

(h) 15 basis functions (dashed lines) and the full order solution (solid line) at $t = 1.0$

Fig. A.26. The inhomogeneous media example in Section 4.5. (a)–(f): solutions at different times, where "Full" is the full order solution, "LP" is by the learning-based projection method, and "Projection" is the L^2 projection of the full order solution onto the learned reduced subspace. (g)–(h): 15 learned basis functions (dashed lines) and the full order solution (solid line) at $t = 0.7$ and $t = 1.0$.

CRediT authorship contribution statement

Zhichao Peng: Conceptualization, Methodology, Validation, Visualization, Investigation, Data curation, Writing – original draft, Writing – review & editing. **Min Wang:** Conceptualization, Writing – original draft, Writing – review & editing, Software, Validation. **Fengyan Li:** Conceptualization, Supervision, Validation, Writing – review & editing.

Acknowledgments

Appendix. More on the oscillation in Fig. 4.12 with $r = 15$

In Fig. 4.12, non-physical oscillation can be observed in the reduced order solution generated by the LP method with $r = 15$. To gain some understanding to the origin of such oscillation, in Fig. A.26(a)–A.26(f), we present the evolution of the full order solution, reduced order solution by the proposed LP method, and the L^2 projection of the full order solution onto the learned adaptive reduced space. At $t = 0.6$, all three curves match well, with slight difference observed in the zoomed-in plot. At $t = 0.7$, the oscillation in both the LP solution and the projection becomes quite visible. At $t = 1.0$, the oscillation in the LP solution remains and spreads out, while the projection of the full order solution displays little difference from the full order solution. In Fig. A.26(g)–A.26(h), we further overlay the 15 learned basis functions and the full order solution at $t = 0.7$ and $t = 1.0$, respectively. One can see that the learned basis functions faithfully capture the local nature of the full order solution, without visible oscillation in themselves. The basis functions at these two different times also capture the propagation of the solution as well as its slight reduction in amplitude as time increases. From the observations above, we attribute the numerical oscillation in the LP solution with $r = 15$ to the accumulation of the errors over time, the extrapolation of the basis functions from the training set to a test sample, and also to the online projection step. The underlying media is inhomogeneous, and this also contributes to the difference observed at $t = 0.7$ and $t = 1.0$ in the shape and amplitude of the basis functions and hence of the oscillation in the LP solution. Adaptively adding artificial viscosity in the online phase of the LP method to control these oscillations is worthy future research.

References

[1] Peter Benner, Mario Ohlberger, Albert Cohen, Karen Willcox, Model Reduction and Approximation: Theory and Algorithms, SIAM, 2017.
[2] Mario Ohlberger, Stephan Rave, Reduced basis methods: Success, limitations and future challenges, in: Proceedings of ALGORITMY, 2016, pp. 1–12.
[3] C. Greif, K. Urban, Decay of the Kolmogorov N-width for wave problems, Appl. Math. Lett. 96 (2019) 216–222.
[4] Albert Cohen, Ronald Devore, Guergana Petrova, Przemyslaw Wojtaszczyk, Optimal stable nonlinear approximation, Found. Comput. Math. (2021) 1–42.
[5] Allan Pinkus, N-Widths in Approximation Theory, Vol. 7, Springer Science & Business Media, 2012.
[6] Gal Berkooz, Philip Holmes, John L. Lumley, The proper orthogonal decomposition in the analysis of turbulent flows, Annu. Rev. Fluid Mech. 25 (1) (1993) 539–575.
[7] Martin Gubisch, Stefan Volkwein, Proper orthogonal decomposition for linear-quadratic optimal control, in: Model Reduction and Approximation: Theory and Algorithms, Vol. 5, SIAM, 2017, p. 66.
[8] Jan S. Hesthaven, Gianluigi Rozza, Benjamin Stamm, et al., Certified Reduced Basis Methods for Parametrized Partial Differential Equations, Vol. 590, Springer, 2016.
[9] K. Carlberg, Adaptive h-refinement for reduced-order models, Internat. J. Numer. Methods Engrg. 102 (5) (2015) 1192–1210.
[10] B. Peherstorfer, Model reduction for transport-dominated problems via online adaptive bases and adaptive sampling, SIAM J. Sci. Comput. 42 (5) (2020) A2803–A2836.
[11] Julius Reiss, Philipp Schulze, Jörn Sesterhenn, Volker Mehrmann, The shifted proper orthogonal decomposition: A mode decomposition for multiple transport phenomena, SIAM J. Sci. Comput. 40 (3) (2018) A1322–A1344.
[12] Rambod Mojgani, Maciej Balajewicz, Lagrangian basis method for dimensionality reduction of convection dominated nonlinear flows, 2017, arXiv:1701.04343.
[13] Hannah Lu, Daniel M. Tartakovsky, Lagrangian dynamic mode decomposition for construction of reduced-order models of advection-dominated phenomena, J. Comput. Phys. 407 (2020) 109229.
[14] Hannah Lu, Daniel M. Tartakovsky, Dynamic mode decomposition for construction of reduced-order models of hyperbolic problems with shocks, J. Mach. Learn. Model. Comput. 2 (1) (2021).
[15] C. Rowley, I. Kevrekidis, J. Marsden, K. Lust, Reduction and reconstruction for self-similar dynamical systems, Nonlinearity 16 (4) (2003) 1257.
[16] M. Ohlberger, S. Rave, Nonlinear reduced basis approximation of parameterized evolution equations via the method of freezing, C. R. Math. 351 (23–24) (2013) 901–906.
[17] W.-J. Beyn, V. Thümmler, Freezing solutions of equivariant evolution equations, SIAM J. Appl. Dyn. Syst. 3 (2) (2004) 85–116.
[18] G. Welper, h And hp-adaptive interpolation by transformed snapshots for parametric and stochastic hyperbolic PDEs, 2017, arXiv:1710.11481.
[19] G. Welper, Interpolation of functions with parameter dependent jumps by transformed snapshots, SIAM J. Sci. Comput. 39 (4) (2017) A1225–A1250.
[20] G. Welper, Transformed snapshot interpolation with high resolution transforms, SIAM J. Sci. Comput. 42 (4) (2020) A2037–A2061.

[21] J.-F. Gerbeau, D. Lombardi, Approximated Lax pairs for the reduced order integration of nonlinear evolution equations, J. Comput. Phys. 265 (2014) 246–269.

[22] J.-F. Gerbeau, D. Lombardi, E. Schenone, Reduced order model in cardiac electrophysiology with approximated lax pairs, Adv. Comput. Math. 41 (5) (2015) 1103–1130.

[23] M. Kirby, D. Armbruster, Reconstructing phase space from PDE simulations, Z. Angew. Math. Phys. ZAMP 43 (6) (1992) 999–1022.

[24] C. Rowley, J. Marsden, Reconstruction equations and the Karhunen–Loève expansion for systems with symmetry, Physica D 142 (1–2) (2000) 1–19.

[25] D. Rim, S. Moe, R. LeVeque, Transport reversal for model reduction of hyperbolic partial differential equations, SIAM/ASA Int. J. Uncertain. Quantif. 6 (1) (2018) 118–150.

[26] T. Taddei, S. Perotto, A. Quarteroni, Reduced basis techniques for nonlinear conservation laws, ESAIM Math. Model. Numer. Anal. 49 (3) (2015) 787–814.

[27] T. Taddei, A registration method for model order reduction: data compression and geometry reduction, SIAM J. Sci. Comput. 42 (2) (2020) A997–A1027.

[28] Davide Torlo, Model reduction for advection dominated hyperbolic problems in an ale framework: Offline and online phases, 2020, arXiv preprint arXiv:2003.13735.

[29] Andrea Ferrero, Tommaso Taddei, Lei Zhang, Registration-based model reduction of parameterized two-dimensional conservation laws, 2021, arXiv preprint arXiv:2105.02024.

[30] V. Ehrlacher, D. Lombardi, O. Mula, F.-X. Vialard, Nonlinear model reduction on metric spaces. Application to one-dimensional conservative PDEs in wasserstein spaces, ESAIM Math. Model. Numer. Anal. (2019).

[31] N. Cagniart, R. Crisovan, Y. Maday, R. Abgrall, Model order reduction for hyperbolic problems: a new framework, 2017.

[32] N. Cagniart, A Few Nonlinear Approaches in Model Order Reduction (Ph.D. thesis), Sorbonne Université, 2018.

[33] Nicolas Cagniart, Yvon Maday, Benjamin Stamm, Model order reduction for problems with large convection effects, in: Contributions to Partial Differential Equations and Applications, Springer, 2019, pp. 131–150.

[34] M. Nonino, F. Ballarin, G. Rozza, Y. Maday, Overcoming slowly decaying Kolmogorov n-width by transport maps: application to model order reduction of fluid dynamics and fluid–structure interaction problems, 2019, arXiv:1911.06598.

[35] D. Rim, B. Peherstorfer, K. Mandli, Manifold approximations via transported subspaces: Model reduction for transport-dominated problems, 2019, arXiv:1912.13024.

[36] N. Nair, M. Balajewicz, Transported snapshot model order reduction approach for parametric, steady-state fluid flows containing parameter-dependent shocks, Internat. J. Numer. Methods Engrg. 117 (12) (2019) 1234–1262.

[37] Rambod Mojgani, Maciej Balajewicz, Physics-aware registration based auto-encoder for convection dominated PDEs, 2020, arXiv preprint arXiv:2006.15655.

[38] R. McCann, A convexity principle for interacting gases, Adv. Math. 128 (1) (1997) 153–179.

[39] C. Villani, Optimal Transport: Old and New, Vol. 338, Springer Science & Business Media, 2008.

[40] K. Lee, K. Carlberg, Deep conservation: A latent dynamics model for exact satisfaction of physical conservation laws, 2019, arXiv:1909.09754.

[41] K. Lee, K. Carlberg, Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders, J. Comput. Phys. 404 (2020) 108973.

[42] Y. Kim, Y. Choi, D. Widemann, T. Zohdi, A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder, 2020, arXiv:2009.11990.

[43] Stefania Fresca, Andrea Manzoni, et al., A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized PDEs, J. Sci. Comput. 87 (2) (2021) 1–36.

[44] D. Rim, L. Venturi, J. Bruna, B. Peherstorfer, Depth separation for reduced deep networks in nonlinear model reduction: distilling shock waves in nonlinear hyperbolic problems, 2020, arXiv:2007.13977.

[45] Beatrice Battisti, Tobias Blickhan, Guillaume Enchery, Virginie Ehrlacher, Damiano Lombardi, Olga Mula, Wasserstein model reduction approach for parametrized flow problems in porous media, 2022.

[46] Jens M. Melenk, On n-widths for elliptic problems, J. Math. Anal. Appl. 247 (1) (2000) 272–289.

[47] Annalisa Buffa, Yvon Maday, Anthony T Patera, Christophe Prud'homme, Gabriel Turinici, A priori convergence of the greedy algorithm for the parametrized reduced basis method, ESAIM: Math. Model. Numer. Anal-Modél. Math. Anal. Numér. 46 (3) (2012) 595–603.

[48] Carmen Gräßle, Michael Hinze, POD reduced-order modeling for evolution equations utilizing arbitrary finite element discretizations, Adv. Comput. Math. 44 (6) (2018) 1941–1978.

[49] Masayuki Yano, Discontinuous Galerkin reduced basis empirical quadrature procedure for model reduction of parametrized nonlinear conservation laws, Adv. Comput. Math. 45 (5) (2019) 2287–2320.

[50] Masayuki Yano, Goal-oriented model reduction of parametrized nonlinear partial differential equations: Application to aerodynamics, Internat. J. Numer. Methods Engrg. 121 (23) (2020) 5200–5226.

[51] Wolfgang Dahmen, Min Wang, Zhu Wang, Nonlinear reduced DNN models for state estimation, 2021, arXiv preprint arXiv:2110.08951.

[52] Yvon Maday, Benjamin Stamm, Locally adaptive greedy approximations for anisotropic parameter reduced basis spaces, SIAM J. Sci. Comput. 35 (6) (2013) A2417–A2441.

[53] Jiequn Han, Chao Ma, Zheng Ma, E. Weinan, Uniformly accurate machine learning-based hydrodynamic models for kinetic equations, Proc. Natl. Acad. Sci. 116 (44) (2019) 21983–21991.

[54] Francesco Romor, Giovanni Stabile, Gianluigi Rozza, Non-linear manifold ROM with convolutional autoencoders and reduced over-collocation method, 2022, arXiv preprint arXiv:2203.00360.

[55] Youngsoo Choi, Deshawn Coombs, Robert Anderson, SNS: A solution-based nonlinear subspace method for time-dependent model order reduction, SIAM J. Sci. Comput. 42 (2) (2020) A1116–A1146.

[56] Yanlai Chen, Sigal Gottlieb, Lijie Ji, Yvon Maday, An EIM-degradation free reduced basis method via over collocation and residual hyper reduction-based error estimation, 2021, arXiv preprint arXiv:2101.05902.

[57] Kookjin Lee, Nathaniel A. Trask, Ravi G. Patel, Mamikon A. Gulian, Eric C. Cyr, Partition of unity networks: Deep hp-approximation, 2021, arXiv:2101.11256.

[58] Lu Lu, Pengzhan Jin, George Em Karniadakis, Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators, 2019, arXiv:1910.03193.

[59] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al., Tensorflow: A system for large-scale machine learning, in: 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 16, 2016, pp. 265–283.

[60] Huazhong Tang, Tao Tang, Adaptive mesh methods for one-and two-dimensional hyperbolic conservation laws, SIAM J. Numer. Anal. 41 (2) (2003) 487–515.

[61] Joachim Schöberl, C++ 11 implementation of finite elements in NGSolve, 30, 2014, Institute for Analysis and Scientific Computing, Vienna University of Technology.

[62] Mary Fanett Wheeler, An elliptic collocation-finite element method with interior penalties, SIAM J. Numer. Anal. 15 (1) (1978) 152–161.
[63] Guang-Shan Jiang, Chi-Wang Shu, Efficient implementation of weighted ENO schemes, J. Comput. Phys. 126 (1) (1996) 202–228.
[64] Chi-Wang Shu, Stanley Osher, Efficient implementation of essentially non-oscillatory shock-capturing schemes, J. Comput. Phys. 77 (2) (1988) 439–471.
[65] Min Wang, Siu Wun Cheung, Wing Tat Leung, Eric T. Chung, Yalchin Efendiev, Mary Wheeler, Reduced-order deep learning for flow dynamics. The interplay between deep learning and model reduction, J. Comput. Phys. 401 (2020) 108939.