Efficient nonlinear manifold reduced order model

Youngkyu Kim
Mechanical Engineering
University of California, Berkeley
Berkeley, CA 94720
youngkyu_kim@berkeley.edu

Youngsoo Choi
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
Livermore, CA 94550
choi15@llnl.gov

David Widemann
Computational Engineering Division
Lawrence Livermore National Laboratory
Livermore, CA 94550
widemann1@llnl.gov

Tarek Zohdi
Mechanical Engineering
University of California, Berkeley
Berkeley, CA 94720
zohdi@berkeley.edu

Abstract

Traditional linear subspace reduced order models (LS-ROMs) are able to accelerate physical simulations, in which the intrinsic solution space falls into a subspace with a small dimension, i.e., the solution space has a small Kolmogorov n-width. However, for physical phenomena not of this type, such as advection-dominated flow phenomena, a low-dimensional linear subspace poorly approximates the solution. To address cases such as these, we have developed an efficient nonlinear manifold ROM (NM-ROM), which can better approximate high-fidelity model solutions with a smaller latent space dimension than the LS-ROMs. Our method takes advantage of the existing numerical methods that are used to solve the corresponding full order models (FOMs). The efficiency is achieved by developing a hyper-reduction technique in the context of the NM-ROM. Numerical results show that neural networks can learn a more efficient latent space representation on advection-dominated data from 2D Burgers' equations with a high Reynolds number. A speed-up of up to 11.7 for 2D Burgers' equations is achieved with an appropriate treatment of the nonlinear terms through a hyper-reduction technique.

1 Introduction

Physical simulations are influencing developments in science, engineering, and technology more rapidly than ever before. However, high-fidelity, forward physical simulations are computationally expensive and, thus, make intractable many decision-making applications, such as design optimization, inverse problems, optimal controls, and uncertainty quantification. These applications require many forward simulations to explore the parameter space in the outer loop. To compensate for the computational expense issue, many surrogate models have been developed: from simply using interpolation schemes for specific quantity of interests to physics-informed surrogate models. This paper focuses on the latter because a physics-informed surrogate model is more robust in predicting physical solutions than the simple interpolation schemes.

Among many types of physics-informed surrogate models, the projection-based linear subspace reduced order models (LS-ROMs) take advantage of both the known governing equation and data with linear subspace solution representation [1]. Although LS-ROMs have been successfully applied to many forward physical problems [2–10] and partial differential equation(PDE)-constrained optimization problems [11–14], the linear subspace solution representation suffers from not being able to represent certain physical simulation solutions with a small basis dimension, such as advection-

Workshop on machine learning for engineering modeling, simulation and design @ NeurIPS 2020

dominated or sharp gradient solutions. This is because LS-ROMs work only for physical problems, in which the intrinsic solution space falls into a subspace with a small dimension, i.e., the solution space has a small Kolmogorov n-width. Although there have been many attempts to resolve these shortcomings of LS-ROMs with various methods, [15–29], all these approaches are still based on the linear subspace solution representation. We transition to a nonlinear, low-dimensional manifold to approximate the solution better than linear methods.

There are many works available in the current literature that looked into the nonlinear manifold solution represenation, using neural networks (NNs) as surrogates for physical simulations [30–48]. However, these methods do not take advantage of the existing numerical methods for high-fidelity physical simulations. Recently, a neural network-based ROM is developed in [49], where the weights and biases are determined in the training phase and the existing numerical methods are utilized in their models. The same technique is extended to preserve the conserved quantities in the physical conservation laws [50]. However, their approaches do not achieve any speed-up because the nonlinear terms that still scale with the corresponding FOM size need to be updated every time step or Newton step.

We present a fast and accurate physics-informed neural network ROM with a nonlinear manifold solution representation, i.e., the nonlinear manifold ROM (NM-ROM). We train a shallow masked autoencoder with solution data from the corresponding FOM simulations and use the decoder as the nonlinear manifold solution representation. Our NM-ROM is different from the aformentioned physics-informed neural networks in that we take advantage of the existing numerical methods of solving PDE in our approach and a considerable speed-up is achieved.

2 Full order model

A parameterized nonlinear dynamical system is considered, characterized by a system of nonlinear ordinary differential equations (ODEs), which can be considered as a resultant system from semi-discretization of Partial Differential Equations (PDEs) in space domains

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x}, t; \boldsymbol{\mu}), \qquad \boldsymbol{x}(0; \boldsymbol{\mu}) = \boldsymbol{x}_0(\boldsymbol{\mu}), \tag{1}$$

where $t \in [0, T]$ denotes time with the final time $T \in \mathbb{R}_+$, and $\boldsymbol{x}(t; \boldsymbol{\mu})$ denotes the time-dependent, parameterized state implicitly defined as the solution to problem (1) with $\boldsymbol{x} : [0, T] \times \mathcal{D} \to \mathbb{R}^{N_s}$. Further, $\boldsymbol{f} : \mathbb{R}^{N_s} \times [0, T] \times \mathcal{D} \to \mathbb{R}^{N_s}$ with $(\boldsymbol{w}, \tau; \boldsymbol{\nu}) \mapsto \boldsymbol{f}(\boldsymbol{w}, \tau; \boldsymbol{\nu})$ denotes the time derivative of $\boldsymbol{x}$, which we assume to be nonlinear in at least its first argument. The initial state is denoted by $\boldsymbol{x}_0 : \mathcal{D} \to \mathbb{R}^{N_s}$, and $\boldsymbol{\mu} \in \mathcal{D}$ denotes parameters in the domain $\mathcal{D} \subseteq \mathbb{R}^{n_\mu}$.

A uniform time discretization is assumed throughout the paper, characterized by time step $\Delta t \in \mathbb{R}_+$ and time instances $t^n = t^{n-1} + \Delta t$ for $n \in \mathbb{N}(N_t)$ with $t^0 = 0$, $N_t \in \mathbb{N}$, and $\mathbb{N}(N) := \{1, \ldots, N\}$. To avoid notational clutter, we introduce the following time discretization-related notations: $\boldsymbol{x}_n := \boldsymbol{x}(t^n; \boldsymbol{\mu})$, $\tilde{\boldsymbol{x}}_n := \tilde{\boldsymbol{x}}(t^n; \boldsymbol{\mu})$, $\hat{\boldsymbol{x}}_n := \hat{\boldsymbol{x}}(t^n; \boldsymbol{\mu})$, and $\boldsymbol{f}_n := \boldsymbol{f}(\boldsymbol{x}(t^n; \boldsymbol{\mu}), t^n; \boldsymbol{\mu})$, where $\boldsymbol{x}$, $\tilde{\boldsymbol{x}}$, $\hat{\boldsymbol{x}}$ and $\boldsymbol{f}$ are defined in. The implicit backward Euler (BE)[1] time integrator numerically solves Eq. (1), by solving the following nonlinear system of equations, i.e., $\boldsymbol{x}_n - \boldsymbol{x}_{n-1} = \Delta t \boldsymbol{f}_n$, for $\boldsymbol{x}_n$ at n-th time step. The corresponding residual function is defined as

$$\boldsymbol{r}^n_{\mathrm{BE}}(\boldsymbol{x}_n; \boldsymbol{x}_{n-1}, \boldsymbol{\mu}) := \boldsymbol{x}_n - \boldsymbol{x}_{n-1} - \Delta t \boldsymbol{f}_n. \tag{2}$$

3 Nonlinear manifold reduced order model (NM-ROM)

The NM-ROM applies solution representation using a nonlinear manifold $\mathcal{S} := \{\boldsymbol{g}(\hat{\boldsymbol{v}}) \,|\, \hat{\boldsymbol{v}} \in \mathbb{R}^{n_s}\}$, where $\boldsymbol{g} : \mathbb{R}^{n_s} \to \mathbb{R}^{N_s}$ with $n_s \ll N_s$ denotes a nonlinear function that maps a latent space of dimension n_s to the full order model space of dimension, N_s. That is, the NM-ROM approximates the solution in a trial manifold as

$$\boldsymbol{x} \approx \tilde{\boldsymbol{x}} = \boldsymbol{x}_{ref} + \boldsymbol{g}(\hat{\boldsymbol{x}}). \tag{3}$$

[1]Other time integrators can be used in our NM-ROMs.

The construction of the nonlinear function, $\boldsymbol{g}$, is explained in Section 4. By plugging Eq. (3) into Eq. (2), the residual function at nth time step becomes

$$\begin{aligned}\tilde{\mathbf{r}}_{\mathrm{BE}}^{n}(\hat{\boldsymbol{x}}_n;\hat{\boldsymbol{x}}_{n-1},\boldsymbol{\mu}) &:= \boldsymbol{r}_{\mathrm{BE}}^{n}(\boldsymbol{x}_{ref}+\boldsymbol{g}\left(\hat{\boldsymbol{x}}_n\right);\boldsymbol{x}_{ref}+\boldsymbol{g}\left(\hat{\boldsymbol{x}}_{n-1}\right),\boldsymbol{\mu}) \\ &= \boldsymbol{g}\left(\hat{\boldsymbol{x}}_n\right)-\boldsymbol{g}\left(\hat{\boldsymbol{x}}_{n-1}\right)-\Delta t\boldsymbol{f}(\boldsymbol{x}_{ref}+\boldsymbol{g}\left(\hat{\boldsymbol{x}}_n\right),t_n;\boldsymbol{\mu}),\end{aligned} \tag{4}$$

which is an over-determined system that we close with the least-squares Petrov–Galerkin (LSPG) projection. That is, we minimize the squared norm of the residual vector function at every time step:

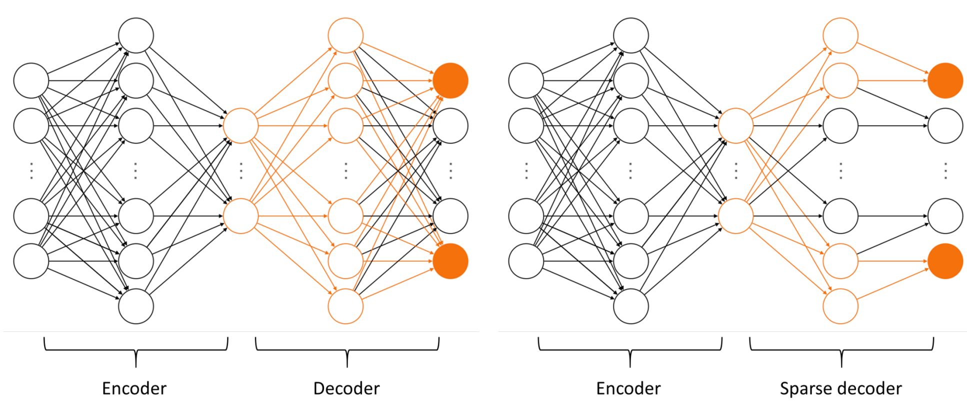
$$\hat{\boldsymbol{x}}_n = \underset{\hat{\boldsymbol{v}}\in\mathbb{R}^{n_s}}{\operatorname{argmin}} \quad \frac{1}{2}\left\|\tilde{\mathbf{r}}_{\mathrm{BE}}^{n}(\hat{\boldsymbol{v}};\hat{\boldsymbol{x}}_{n-1},\boldsymbol{\mu})\right\|_2^2. \tag{5}$$

The Gauss–Newton method with the starting point $\hat{\boldsymbol{x}}_{n-1}$ is applied to solve the minimization problem (5). However, the nonlinear residual vector, $\tilde{\mathbf{r}}_{\mathrm{BE}}^{n}$, scales with FOM size and it needs to be updated every time the argument of the function changes, which occurs either at every time step or Gauss–Newton step. More specifically, if the backward Euler time integrator is used, $\boldsymbol{g}\left(\hat{\boldsymbol{x}}_n\right)$, $\boldsymbol{f}(\boldsymbol{x}_{ref}+\boldsymbol{g}\left(\hat{\boldsymbol{x}}_n\right),t;\boldsymbol{\mu})$, and their Jacobians need to be updated whenever $\hat{\boldsymbol{x}}_n$ changes. Without any special treatment on the nonlinear residual term, no speed-up can be expected. Thus, we apply a hyper-reduction to eliminate the scale with FOM size in the nonlinear term evaluations (see Section 5). Finally, we denote this non-hyper-reduced NM-ROM as NM-LSPG.

4 Shallow masked autoencoder

The nonlinear function, $\boldsymbol{g}$, is the decoder $\boldsymbol{D}$ of an autoencoder in the form of a feedforward neural network. The autoencoder compresses FOM solutions of Eq. (1) with an encoder $\boldsymbol{E}$ and decompresses back to reconstructed FOM solution with an decoder $\boldsymbol{D}$. The autoencoder is trained to reconstruct the FOM solutions of Eq. (1) by minimizing the mean square error between original and reconstructed FOM solutions. Therefore, the dimension of the encoder input and the decoder output is N_s and the dimension of the encoder $\boldsymbol{E}$ output and the decoder $\boldsymbol{D}$ input is n_s.

We intentionally use a non-deep neural network, i.e., three-layer autoencoder, for the decoder to achieve an efficiency that is requried by the hyper-reduction (see Section 5). More specifically, the first layers of the encoder $\boldsymbol{E}$ and decoder $\boldsymbol{D}$ are fully-connected layers, where the nonlinear activation functions are applied and the last layer of the encoder $\boldsymbol{E}$ is fully-connected layer with no activation functions. The last layer of the decoder $\boldsymbol{D}$ is sparsely-connected layer with no activation functions. The sparsity is determined by a mask matrix. These network architectures are shown in Fig. 1.



(a) Without masking (b) With masking

Figure 1: Three-layer encoder/decoder architectures: (a) unmasked and (b) after the sparsity mask is applied. Nodes and edges in orange color represent active path in the subnet that stems from the sampled outputs that are marked as the orange disks. Note that the masked shallow neural network has a sparser structure than the unmasked one. A sparser structure leads to a more efficient model.

There is no way to determine hidden layer sizes *a priori*. If the number of learnable parameters is not enough, the decoder is not able to represent the nonlinear manifold well. On the other hand, too many learnable parameters may result in over-fitting, so the decoder is not able to generalize well, which means the trained decoder cannot be used for problems whose data is unseen, i.e., the predictive case. To avoid over-fitting, we first divide the training data into train and validation datasets. Then, the autoencoder is trained using the train dataset and tested for the generalization using the validation dataset. If the mean squared errors on the validation and train datasets are very different, then over-fitting has occurred. We then reduce the size of the hidden layer and re-train the model [51].

5 Hyper-reduction

The hyper-reduction techniques are developed to eliminate the FOM scale dependecy in nonlinear terms [52–54, 17, 19], which is essential to acheive an efficiency in our NM-ROM. We follow the gappy POD approach [55], in which the nonlinear residual term is approximated as

$$\tilde{\mathbf{r}} \approx \mathbf{\Phi}_r \hat{\boldsymbol{r}}, \tag{6}$$

where $\mathbf{\Phi}_r := [\boldsymbol{\phi}_{r,1}, \ldots, \boldsymbol{\phi}_{r,n_r}] \in \mathbb{R}^{N_s \times n_r}$, $n_s \leq n_r \ll N_s$, denotes the residual basis matrix and $\hat{\boldsymbol{r}} \in \mathbb{R}^{n_r}$ denotes the generalized coordinates of the nonlinear residual term. Here, $\tilde{\mathbf{r}}$ represents a residual vector function, e.g., the backward Euler residual, $\tilde{\mathbf{r}}^n_{BE}$, defined in Eq. (4). We use the singular value decomposition of the FOM solution snapshot matrix to construct $\mathbf{\Phi}_r$, which is justified in [19]. In order to find $\hat{\boldsymbol{r}}$, we apply a sampling matrix $\boldsymbol{Z}^T := [\boldsymbol{e}_{p_1}, \ldots, \boldsymbol{e}_{p_{n_z}}]^T \in \mathbb{R}^{n_z \times N_s}$, $n_s \leq n_r \leq n_z \ll N_s$ on both sides of (6). The vector, $\boldsymbol{e}_{p_i}$, is the p_ith column of the identity matrix $\boldsymbol{I}_{N_s} \in \mathbb{R}^{N_s \times N_s}$. Then the following least-squares problem is solved:

$$\hat{\boldsymbol{r}} := \underset{\hat{\boldsymbol{v}} \in \mathbb{R}^{n_r}}{\operatorname{argmin}} \quad \frac{1}{2} \left\| \boldsymbol{Z}^T (\tilde{\mathbf{r}} - \mathbf{\Phi}_r \hat{\boldsymbol{v}}) \right\|_2^2. \tag{7}$$

The solution to Eq. (7) is given as $\hat{\boldsymbol{r}} = (\boldsymbol{Z}^T \mathbf{\Phi}_r)^\dagger \boldsymbol{Z}^T \tilde{\mathbf{r}}$, where the Moore–Penrose inverse of a matrix $\boldsymbol{A} \in \mathbb{R}^{n_z \times n_r}$ with full column rank is defined as $\boldsymbol{A}^\dagger := (\boldsymbol{A}^T \boldsymbol{A})^{-1} \boldsymbol{A}^T$. Therefore, Eq. (6) becomes $\tilde{\mathbf{r}} \approx \mathcal{P} \tilde{\mathbf{r}}$, where $\mathcal{P} := \mathbf{\Phi}_r (\boldsymbol{Z}^T \mathbf{\Phi}_r)^\dagger \boldsymbol{Z}^T$ is the oblique projection matrix. We do not construct the sampling matrix $\boldsymbol{Z}$. *Instead, it maintains the sampling indices* $\{p_1, \ldots, p_{n_f}\}$ *and corresponding rows of* $\mathbf{\Phi}_r$ *and* $\tilde{\mathbf{r}}$. This enables hyper-reduced ROMs to achieve a speed-up. The sampling indices (i.e., $\boldsymbol{Z}$) can be determined by Algorithm 3 of [17] for computational fluid dynamics problems and Algorithm 5 of [56] for other problems.

The hyper-reduced residual, $\mathcal{P}\tilde{\mathbf{r}}^n_{BE}$, is used in the minimization problem in Eq. (5):

$$\hat{\boldsymbol{x}}_n = \underset{\hat{\boldsymbol{v}} \in \mathbb{R}^{n_s}}{\operatorname{argmin}} \quad \frac{1}{2} \left\| (\boldsymbol{Z}^T \mathbf{\Phi}_r)^\dagger \boldsymbol{Z}^T \tilde{\mathbf{r}}^n_{\mathrm{BE}}(\hat{\boldsymbol{v}}; \hat{\boldsymbol{x}}_{n-1}, \boldsymbol{\mu}) \right\|_2^2. \tag{8}$$

Note that the pseudo-inverse $(\boldsymbol{Z}^T \mathbf{\Phi}_r)^\dagger$ can be pre-computed. Due to the definition of $\tilde{\mathbf{r}}^n_{\mathrm{BE}}$ in Eq. (4), the sampling matrix $\boldsymbol{Z}$ needs to be applied to the following two terms: $\boldsymbol{g}(\hat{\boldsymbol{x}}_n) - \boldsymbol{g}(\hat{\boldsymbol{x}}_{n-1})$ and $\boldsymbol{f}(\boldsymbol{x}_{ref} + \boldsymbol{g}(\hat{\boldsymbol{x}}_n), t; \boldsymbol{\mu})$ at every time step. The first term, $\boldsymbol{Z}^T(\boldsymbol{g}(\hat{\boldsymbol{x}}_n) - \boldsymbol{g}(\hat{\boldsymbol{x}}_{n-1}))$, requires that only selected outputs of the decoder be computed. Furthermore, for the second term, the nonlinear residual elements that are selected by the sampling matrix need to be computed. This implies that we have to keep track of the outputs of $\boldsymbol{g}$ that are needed to compute the selected nonlinear residual elements by the sampling matrix, which is usually a larger set than the outputs that are selected solely by the sampling matrix. Therefore, we build a subnet that computes only the outputs of the decoder that are required to compute the nonlinear residual elements. Such outputs are demonstrated as the solid oragne disks and the corresponding subnet is depicted in Fig. 1(b). Finally, we denote this hyper-reduced NM-ROM as NM-LSPG-HR.

6 2D Burgers' equation

We demonstrate the performance of our NM-ROMs (i.e., NM-LSPG and NM-LSPG-HR) by comparing it with LS-ROMs (i.e., LS-LSPG and LS-LSPG-HR) that was first introduced in [56]. We solve the following parameterized 2D viscous Burgers' equation:

$$
\begin{aligned}
\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} &= \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \\
\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} &= \frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) \\
(x, y) \in \Omega &= [0,1]\times[0,1] \\
t &\in [0,2],
\end{aligned}
\tag{9}
$$

with the boundary condition

$$
u(x, y, t; \mu) = v(x, y, t; \mu) = 0 \quad \text{on} \quad \Gamma = \{(x, y) | x \in \{0, 1\}, y \in \{0, 1\}\} \tag{10}
$$

and the initial condition

$$
u(x, y, 0; \mu) = \begin{cases} \mu \sin(2\pi x) \cdot \sin(2\pi y) & \text{if } (x, y) \in [0, 0.5] \times [0, 0.5] \\ 0 & \text{otherwise} \end{cases} \tag{11}
$$

$$
v(x, y, 0; \mu) = \begin{cases} \mu \sin(2\pi x) \cdot \sin(2\pi y) & \text{if } (x, y) \in [0, 0.5] \times [0, 0.5] \\ 0 & \text{otherwise} \end{cases} \tag{12}
$$

where $\mu \in \mathcal{D} = [0.9, 1.1]$ is a parameter and $u(x, y, t; \mu)$ and $v(x, y, t; \mu)$ denote the x and y directional velocities, respectively, with $u : \Omega \times [0, 2] \times \mathcal{D} \to \mathbb{R}$ and $v : \Omega \times [0, 2] \times \mathcal{D} \to \mathbb{R}$ defined as the solutions to Eq. (9), and Re is a Reynolds number which is set $Re = 10,000$. For this case, the FOM solution snapshot shows slowly decaying singular values. We observe that a sharp gradient, i.e., a shock, appears in the FOM solution (e.g., see Fig. 3(a)). We use 60×60 uniform mesh with the backward difference scheme for the first spatial derivative terms and the central difference scheme for the second spatial derivative terms. Then, we use the backward Euler scheme with time step size $\Delta t = \frac{2}{n_t}$, where $n_t = 1,500$ is the number of time steps.

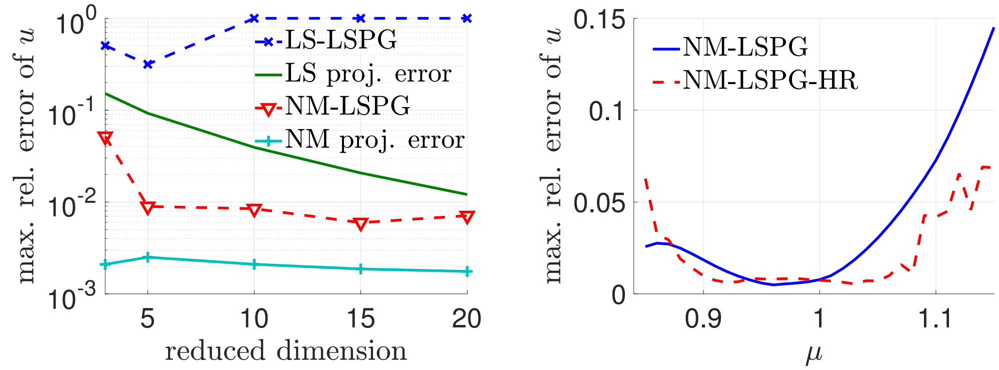
For the training process, we collect solution snapshots associated with the parameter $\mu \in \mathcal{D}_{train} = \{0.9, 0.95, 1.05, 1.1\}$, such that $n_{\text{train}} = 4$, at which the FOM is solved. Then, the number of train data points is $n_{\text{train}} \cdot (n_t + 1) = 6,004$ and 10% of the train data are used for validation purposes. We employ the Adam optimizer [57] with the SGD and the initial learning rate of 0.001, which decreases by a factor of 10 when a training loss stagnates for 10 successive epochs. We set the encoder and decoder hidden layer sizes to $6,728$ and $33,730$, respectively and vary the dimension of the latent space from 5 to 20. The weights and bias of the autoencoder are initialized via Kaiming initialization [58]. The batch size is 240 and the maximum number of epochs is $10,000$. The training process is stopped if the loss on the validation dataset stagnates for 200 epochs.

After the training is done, the NM-LSPG and LS-LSPG solve the Eq. (9) with the target parameter $\mu = 1$, which is not included in the train dataset. Fig. 2 shows the relative error versus the reduced dimension n_s for both LS-LSPG and NM-LSPG. It also shows the projection errors for LS-ROMs and NM-ROMs, which are the lower bounds that any LS-ROMs and NM-ROMs can reach, respectively. As expected the relative error for the NM-LSPG is lower than the one for the LS-LSPG. We even observe that the relative errors of NM-LSPG are even lower than the LS projected error.

We vary the number of residual basis and residual samples, with the fixed number of training parameter instances $n_{\text{train}} = 4$ and the reduced dimension $n_s = 5$, and measure the wall-clock time. The results are shown in Table 1. Although the LS-LSPG-HR can achieve better speed-up than the NM-LSPG-HR, the relative error of the LS-LSPG-HR is too large to be reasonable, e.g., the relative errors of around 37%. On the other hand, the NM-LSPG-HR achieves much better accuracy, i.e., a relative error of around 1%, and a factor 11 speed-up.

Table 1: The top 6 maximum relative errors and wall-clock times at different numbers of residual basis and samples which range from 40 to 60.

	NM-LSPG-HR						LS-LSPG-HR					
Residual basis, n_r	55	56	51	53	54	44	59	53	53	53	53	53
Residual samples, n_z	58	59	54	56	57	47	59	58	59	56	55	53
Max. rel. error (%)	0.93	0.94	0.95	0.97	0.97	0.98	34.38	37.73	37.84	37.95	37.96	37.97
Wall-clock time (sec)	12.15	12.35	12.09	12.14	12.29	12.01	5.26	5.02	4.86	5.05	4.75	7.18
Speed-up	11.58	11.39	11.63	11.58	11.44	11.71	26.76	28.02	28.95	27.83	29.61	19.58

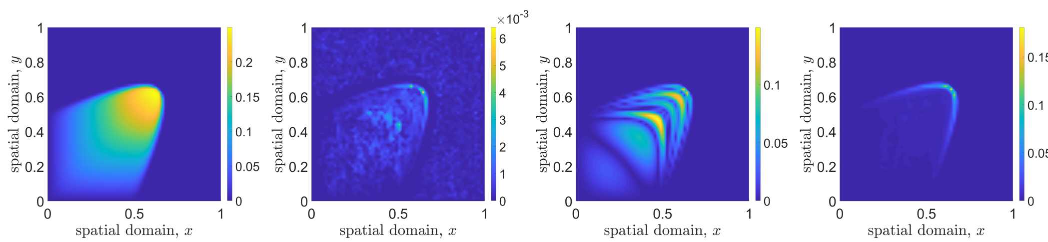


(a) Relative errors vs reduced dimensions (b) Relative errors vs μ

Figure 2: The comparison of the NM-LSPG-HR and NM-LSPG on the maximum relative errors. A maximum relative error that is 1 means the ROM failed to solve the problem.

Fig. 3 shows FOM solutions at the last time and absolute differences between FOM and other approaches, i.e., NM-LSPG-HR and LS-LSPG-HR with the reduced dimension being $n_s = 5$ and a black-box NN approach (BB-NN). The BB-NN approach is similar to the one described in [59]. The main difference is that $L1$-norms and physics constraints were not used in our loss function. This approach gave a maximum relative error of 38.6% and has a speed-up of 119. While this approach is appealing in that it does not require access to the PDE solver, the errors are too large for our application. For NM-LSPG-HR, 55 residual basis dimension and 58 residual samples are used and for LS-LSPG-HR, 59 residual basis dimension and 59 residual samples are used. Both FOM and NM-LSPG-HR show good agreement in their solutions, while the LS-LSPG-HR is not able to achieve a good accuracy. In fact, the NM-LSPG-HR is able to achieve an accuracy as good as the NM-LSPG for some combinations of the small number of residual basis and residual samples.

Finally, Fig. 2(b) shows the maximum relative error over the test range of the parameter points. Note that the NM-LSPG and NM-LSPG-HR are the most accurate within the range of the training points, i.e., $[0.9, 1.1]$. As the parameter points go beyond the training parameter domain, the accuracy of the NM-LSPG and NM-LSPG-HR start to deteriorate gradually. This implies that the NM-LSPG and NM-LSPG-HR have a trust region. Its trust region should be determined by each application.



(a) FOM (b) NM-LSPG-HR (c) LS-LSPG-HR (d) BB-NN

Figure 3: (a) Solution snapshots, u, of FOM and absolute differences of (b) NM-LSPG-HR, (c) LS-LSPG-HR, and (d) BB-NN with respect to FOM solution at $t = 2$.

7 Discussion & conclusion

In this work, we have successfully developed an accurate and efficient NM-ROM. We demonstrated that both the LS-ROM and BB-NN are not able to represent advection-dominated or sharp gradient solutions of 2D viscous Burgers' equation with a high Reynolds number. However, our new approach, NM-LSPG-HR, solves such problem accurately and efficiently. The speed-up of the NM-LSPG-HR is achieved by choosing the shallow masked decoder as the nonlinear manifold and applying the efficient hyper-reduction computation. Because the difference in the computational cost of the FOM and NM-LSPG-HR increases as a function of the number of mesh points, we expect more speed-up as the number of mesh points becomes larger.

Compared with the deep neural networks for computer vision and natural language processing applications, our neural networks are shallow with a small number of parameters. However, these networks were able to capture the variation in our 2D Burgers' simulations. A main future work for transferring this work to more complex simulations, will be to find the right balance between a shallow network that is large enough to capture the data variance and yet small enough to run faster than the FOM. Another future work will be to find an efficient way of determining the proper size of the residual basis and the number of sample points *a priori*. To find the optimal size of residual basis and the number of sample points for hyper-reduced ROMs, we relied on test results. This issue is not just for NM-LSPG-HR, but also for LS-LSPG-HR.

Broader Impact

The broader impact of this work will be to accelerate physics simulations to improve design optimization and control problems, which require thousands of simulation runs to learn an optimal design or viable control strategy. While this is not computationally feasible with high-fidelity FOMs, the development of the NM-ROMs is an important step in this direction.

Acknowledgments and Disclosure of Funding

This work was performed at Lawrence Livermore National Laboratory and was supported by the LDRD program (project 20-FS-007). Youngkyu was also supported for this work through generous funding from DTRA. Lawrence Livermore National Laboratory is operated by Lawrence Livermore National Security, LLC, for the U.S. Department of Energy, National Nuclear Security Administration under Contract DE-AC52-07NA27344 and LLNL-CONF-815209. We declare that there were no conflicting interests of any type during the production of this research.

References

[1] Peter Benner, Serkan Gugercin, and Karen Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM review*, 57(4):483–531, 2015.

[2] Mohamadreza Ghasemi and Eduardo Gildin. Localized model reduction in porous media flow. *IFAC-PapersOnLine*, 48(6):242–247, 2015.

[3] Rui Jiang and Louis J Durlofsky. Implementation and detailed assessment of a gnat reduced-order model for subsurface flow simulation. *Journal of Computational Physics*, 379:192–213, 2019.

[4] Yanfang Yang, Mohammadreza Ghasemi, Eduardo Gildin, Yalchin Efendiev, Victor Calo, et al. Fast multiscale reservoir simulations with pod-deim model reduction. *SPE Journal*, 21(06):2–141, 2016.

[5] Huanhuan Yang and Alessandro Veneziani. Efficient estimation of cardiac conductivities via pod-deim model order reduction. *Applied Numerical Mathematics*, 115:180–199, 2017.

[6] Pengfei Zhao, Cai Liu, and Xuan Feng. Pod-deim based model order reduction for the spherical shallow water equations with turkel-zwas finite difference discretization. *Journal of Applied Mathematics*, 2014, 2014.

[7] R Ştefănescu and Ionel Michael Navon. Pod/deim nonlinear model order reduction of an adi implicit shallow water equations model. *Journal of Computational Physics*, 237:95–114, 2013.

[8] M Mordhorst, Timm Strecker, D Wirtz, Thomas Heidlauf, and Oliver Röhrle. Pod-deim reduction of computational emg models. *Journal of Computational Science*, 19:86–96, 2017.

[9] Gabriel Dimitriu, Ionel M Navon, and Răzvan Ştefănescu. Application of pod-deim approach for dimension reduction of a diffusive predator-prey system with allee effect. In *International conference on large-scale scientific computing*, pages 373–381. Springer, 2013.

[10] Harbir Antil, Matthias Heinkenschloss, Ronald HW Hoppe, Christopher Linsenmann, and Achim Wixforth. Reduced order modeling based shape optimization of surface acoustic wave driven microfluidic biochips. *Mathematics and Computers in Simulation*, 82(10):1986–2003, 2012.

[11] David Amsallem, Matthew Zahr, Youngsoo Choi, and Charbel Farhat. Design optimization using hyper-reduced-order models. *Structural and Multidisciplinary Optimization*, 51(4):919–940, 2015.

[12] Youngsoo Choi, Gabriele Boncoraglio, Spenser Anderson, David Amsallem, and Charbel Farhat. Gradient-based constrained optimization using a database of linear reduced-order models. *Journal of Computational Physics*, page 109787, 2020.

[13] Youngsoo Choi, Geoffrey Oxberry, Daniel White, and Trenton Kirchdoerfer. Accelerating design optimization using reduced order models. *arXiv preprint arXiv:1909.11320*, 2019.

[14] Hongfei Fu, Hong Wang, and Zhu Wang. Pod/deim reduced-order modeling of time-fractional partial differential equations with applications in parameter identification. *Journal of Scientific Computing*, 74(1):220–243, 2018.

[15] Rémi Abgrall, David Amsallem, and Roxana Crisovan. Robust model reduction by l^1-norm minimization and approximation via dictionaries: application to nonlinear hyperbolic problems. *Advanced Modeling and Simulation in Engineering Sciences*, 3(1):1–16, 2016.

[16] Julius Reiss, Philipp Schulze, Jörn Sesterhenn, and Volker Mehrmann. The shifted proper orthogonal decomposition: A mode decomposition for multiple transport phenomena. *SIAM Journal on Scientific Computing*, 40(3):A1322–A1344, 2018.

[17] Kevin Carlberg, Charbel Farhat, Julien Cortial, and David Amsallem. The gnat method for nonlinear model reduction: effective implementation and application to computational fluid dynamics and turbulent flows. *Journal of Computational Physics*, 242:623–647, 2013.

[18] Kevin Carlberg, Youngsoo Choi, and Syuzanna Sargsyan. Conservative model reduction for finite-volume models. *Journal of Computational Physics*, 371:280–314, 2018.

[19] Youngsoo Choi, Deshawn Coombs, and Robert Anderson. Sns: a solution-based nonlinear subspace method for time-dependent model order reduction. *SIAM Journal on Scientific Computing*, 42(2):A1116–A1146, 2020.

[20] Dunhui Xiao, Fangxin Fang, Andrew G Buchan, Christopher C Pain, Ionel Michael Navon, Juan Du, and G Hu. Non-linear model reduction for the navier–stokes equations using residual deim method. *Journal of Computational Physics*, 263:1–18, 2014.

[21] PG Constantine and G Iaccarino. Reduced order models for parameterized hyperbolic conservations laws with shock reconstruction. *Center for Turbulence Research Annual Brief*, 2012.

[22] Youngsoo Choi, Peter Brown, Bill Arrighi, Robert Anderson, and Kevin Huynh. Space–time reduced order model for large-scale linear dynamical systems with application to boltzmann transport problems. *Journal of Computational Physics*, P109845, 2020.

[23] Youngsoo Choi and Kevin Carlberg. Space–time least-squares petrov–galerkin projection for nonlinear model reduction. *SIAM Journal on Scientific Computing*, 41(1):A26–A58, 2019.

[24] Tommaso Taddei and Lei Zhang. Space-time registration-based model reduction of parameterized one-dimensional hyperbolic pdes. *arXiv preprint arXiv:2004.06693*, 2020.

[25] Kevin Carlberg. Adaptive h-refinement for reduced-order models. *International Journal for Numerical Methods in Engineering*, 102(5):1192–1210, 2015.

[26] Donsub Rim, Scott Moe, and Randall J LeVeque. Transport reversal for model reduction of hyperbolic partial differential equations. *SIAM/ASA Journal on Uncertainty Quantification*, 6(1):118–150, 2018.

[27] Eric J Parish and Kevin T Carlberg. Windowed least-squares model reduction for dynamical systems. *arXiv preprint arXiv:1910.11388*, 2019.

[28] Benjamin Peherstorfer. Model reduction for transport-dominated problems via online adaptive bases and adaptive sampling. *arXiv preprint arXiv:1812.02094*, 2018.

[29] G Welper. Transformed snapshot interpolation with high resolution transforms. *SIAM Journal on Scientific Computing*, 42(4):A2037–A2061, 2020.

[30] Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5):987–1000, 1998.

[31] MWMG Dissanayake and N Phan-Thien. Neural-network-based approximations for solving partial differential equations. *communications in Numerical Methods in Engineering*, 10(3):195–201, 1994.

[32] B Ph van Milligen, V Tribaldos, and JA Jiménez. Neural network differential equation and plasma equilibrium solver. *Physical review letters*, 75(20):3594, 1995.

[33] Andrew J Meade Jr and Alvaro A Fernandez. The numerical solution of linear ordinary differential equations by feedforward neural networks. *Mathematical and Computer Modelling*, 19(12):1–25, 1994.

[34] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[35] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in neural information processing systems*, pages 6571–6583, 2018.

[36] Yuehaw Khoo, Jianfeng Lu, and Lexing Ying. Solving parametric pde problems with artificial neural networks. *arXiv preprint arXiv:1707.03351*, 2017.

[37] Zichao Long, Yiping Lu, Xianzhong Ma, and Bin Dong. Pde-net: Learning pdes from data. In *International Conference on Machine Learning*, pages 3208–3216, 2018.

[38] Christian Beck, E Weinan, and Arnulf Jentzen. Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations. *Journal of Nonlinear Science*, 29(4):1563–1619, 2019.

[39] Yinhao Zhu, Nicholas Zabaras, Phaedon-Stelios Koutsourelakis, and Paris Perdikaris. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics*, 394:56–81, 2019.

[40] Jens Berg and Kaj Nyström. A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing*, 317:28–41, 2018.

[41] Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018.

[42] Jiequn Han, Arnulf Jentzen, and E Weinan. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

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

[44] Lu Lu, Xuhui Meng, Zhiping Mao, and George E Karniadakis. Deepxde: A deep learning library for solving differential equations. *arXiv preprint arXiv:1907.04502*, 2019.

[45] Guofei Pang, Lu Lu, and George Em Karniadakis. fpinns: Fractional physics-informed neural networks. *SIAM Journal on Scientific Computing*, 41(4):A2603–A2626, 2019.

[46] Dongkun Zhang, Lu Lu, Ling Guo, and George Em Karniadakis. Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. *Journal of Computational Physics*, 397:108850, 2019.

[47] E Weinan and Bing Yu. The deep ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018.

[48] Juncai He, Lin Li, Jinchao Xu, and Chunyue Zheng. Relu deep neural networks and linear finite elements. *arXiv preprint arXiv:1807.03973*, 2018.

[49] Kookjin Lee and Kevin T Carlberg. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *Journal of Computational Physics*, 404:108973, 2020.

[50] Kookjin Lee and Kevin Carlberg. Deep conservation: A latent dynamics model for exact satisfaction of physical conservation laws. *arXiv preprint arXiv:1909.09754*, 2019.

[51] Mark A Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, 37(2):233–243, 1991.

[52] Saifon Chaturantabut and Danny C Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal on Scientific Computing*, 32(5):2737–2764, 2010.

[53] Zlatko Drmac and Serkan Gugercin. A new selection operator for the discrete empirical interpolation method—improved a priori error bound and extensions. *SIAM Journal on Scientific Computing*, 38(2):A631–A648, 2016.

[54] Zlatko Drmac and Arvind Krishna Saibaba. The discrete empirical interpolation method: Canonical structure and formulation in weighted inner product spaces. *SIAM Journal on Matrix Analysis and Applications*, 39(3):1152–1180, 2018.

[55] Richard Everson and Lawrence Sirovich. Karhunen–loeve procedure for gappy data. *JOSA A*, 12(8):1657–1664, 1995.

[56] Kevin Carlberg, Charbel Bou-Mosleh, and Charbel Farhat. Efficient non-linear model reduction via a least-squares petrov–galerkin projection and compressive tensor approximations. *International Journal for Numerical Methods in Engineering*, 86(2):155–181, 2011.

[57] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[58] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[59] Byungsoo Kim, Vinicius C. Azevedo, Nils Thuerey, Theodore Kim, Markus H. Gross, and Barbara Solenthaler. Deep fluids: A generative network for parameterized fluid simulations. *CoRR*, abs/1806.02071, 2018.