Data-Centric Engineering (2023), 4: e2
doi:10.1017/dce.2022.39

RESEARCH ARTICLE

Active-learning-based nonintrusive model order reduction

Qinyu Zhuang[1,2,*], Dirk Hartmann[3], Hans-J. Bungartz[1] and Juan M. Lorenzi[2]

[1]Department of Informatics, Technical University of Munich, 85748 Garching, Germany
[2]Modeling, Simulation & Optimization, Siemens AG, 81739 Munich, Germany
[3]Company Core Technology Simulation and Digital Twin, Siemens Digital Industries Software GmbH, 81739 Munich, Germany
*Corresponding author. E-mail: qinyu.zhuang@tum.de

Received: 16 April 2022; **Revised:** 08 December 2022; **Accepted:** 11 December 2022

Keywords: Active learning; joint space sampling; machine learning; model order reduction; probably approximately correct learning

Abstract

Model order reduction (MOR) can provide low-dimensional numerical models for fast simulation. Unlike intrusive methods, nonintrusive methods are attractive because they can be applied even without access to full order models (FOMs). Since nonintrusive MOR methods strongly rely on snapshots of the FOMs, constructing good snapshot sets becomes crucial. In this work, we propose a novel active-learning-based approach for use in conjunction with nonintrusive MOR methods. It is based on two crucial novelties. First, our approach uses joint space sampling to prepare a data pool of the training data. The training data are selected from the data pool using a greedy strategy supported by an error estimator based on Gaussian process regression. Second, we introduce a case-independent validation strategy based on probably approximately correct learning. While the methods proposed here can be applied to different MOR methods, we test them here with artificial neural networks and operator inference.

Impact Statement

Nonintrusive model order reduction methods draw much attention in the industry for their easy implementation. However, insufficient accuracy and stability hinder broader applications. Large amounts of training data are needed to ensure their accuracy and stability. In this article, we propose an approach to actively collect the training data. The reduced order models built in such a way have not only much better performance than those built in conventional ways but also users can know their expected accuracy and confidence before deploying them. With little required user interaction, the proposed method offers a huge potential for industrial usage to create the so-called digital twin.

1. Introduction

Model order reduction (MOR) (Willcox and Peraire, 2002; Antoulas, 2005; Volkwein, 2013), as a key component of the concept of digital twin (DT) (Hartmann et al., 2018; Rasheed et al., 2019), is increasingly attracting attention from different research fields. The traditional ways of reduced modeling, such as the Krylov subspace method (Heres, 2005), reduced basis method (Haasdonk, 2017), proper

This research article was awarded an Open Data badge for transparent practices. See the Data Availability Statement for details.

orthogonal decomposition (POD) (Lu et al., 2019), and the discrete empirical interpolation method (Chaturantabut and Sorensen, 2010) require detailed knowledge of full order models (FOMs) and are known as intrusive reduction methods. The applicability of intrusive methods is limited in industrial engineering simulation and commercial finite element method (FEM) software. Recently, the development of so-called nonintrusive reduction has attracted much attention.

Nonintrusive MOR methods use data generated by FOMs to build surrogate models that can accurately reproduce the physical behavior encoded in the data. Currently, most data-driven MOR methods are based on machine learning (ML). For example, Feed-forward Neural Network (Regazzoni et al., 2019), operator inference (OpInf) (Peherstorfer and Willcox, 2016), Long-short-term-memory (LSTM) neural network (Mohan and Gaitonde, 2018), recurrent neural network (RNN) (Kani and Elsheikh, 2017; Wang et al., 2020; Wu and Noels, 2022), deep learning (Fresca and Manzoni, 2022), sparse identification of nonlinear dynamics (SINDy) (Champion et al., 2019), and Runge–Kutta neural network (Zhuang et al., 2021) have been tested for reduced modeling.

One of the concerning points of data-driven MOR techniques is collecting enough training data, as computing the FOM to generate the training data is very time-consuming, especially when looking at large 3D simulation models used productively in the industry. Therefore, collecting the training data smartly becomes one of the focuses of the data-driven MOR techniques.

We orient our discussion by focusing on two aspects relevant to snapshot collection: data quality and data quantity. The data quality could describe many aspects, for example, the distribution of the data and the noise in the data. Unlike other applications, the data in the MOR field has fewer corruptions since it is usually obtained directly from the FOM solver, and not, for example, from potentially noisy sensors. Therefore, the main focus will be improving the training data distribution. There are some investigations (Batista et al., 2004; Gyori et al., 2022) discussing how the distribution of the data could influence the ML process itself. However, this issue is less prevalent within MOR research. To minimize the amount of training data, we employ the concept of active learning (AL). The concept of AL is widely used in the world of intrusive MOR (Bui-Thanh et al., 2008; Haasdonk et al., 2011; Lappano et al., 2016). However, the data-driven nonintrusive MOR technique faces extra challenges without intrusiveness. One challenge is finding a good stopping criteria. The stopping criteria are usually needed to validate the reduced order model (ROM) and compute the accuracy. However, currently, most validation (Przekop et al., 2012; Perez et al., 2014) is test-case-dependent, which implies that the general performance of the ROM cannot be evaluated without bias. To address this problem, a load-independent metric is proposed for a particular mechanical problem in Kuether and Allen (2016). The second difficulty is the absence of an error estimator, which can decide the data increment strategy. Such an error estimator is available in an analytical form for the intrusive MOR methods but lacking in the nonintrusive case.

In this article, we propose an alternative approach to tackle the problems mentioned above. To improve the snapshot distribution, we propose a new method for sampling the training data, and we call it joint space sampling (JSS). With this method, we collect the training data from a joint space consisting of an estimated reduced solution space and a parameter space. Besides, we propose to use a validator based on probably approximately correct (PAC) learning theory (Haussler, 1990) to decide when to stop the AL iteration. The validated ROMs are useful for deploying in real industrial use cases where complex time-dependent inputs can be expected. Furthermore, the error estimator to speed up the convergence is constructed in a data-driven way. Among different existing approaches (Paul-Dubois-Taine and Amsallem, 2015; Guo and Hesthaven, 2018; Xiao, 2019), we employ the Kriging interpolation (Krige, 1951) introduced in Guo and Hesthaven (2018) as the error estimator to decide the data increment strategy.

The whole article is structured as follows: in Section 2, MOR based on ML is introduced. In Section 3, the key components and the whole workflow of the new AL-based MOR algorithm are described in detail. Numerical tests used to validate this approach are presented in Section 4. Finally, the conclusions are presented in Section 5.

2. ML-Based MOR

The offline phase of ML-based MOR generally consists of two steps: constructing a low-dimensional (reduced) space and identifying the ROM in the reduced space by ML-based approach. In the online phase, the identified ML model will be used as a surrogate model to predict the evolution of the simulated dynamic system at a real-time speed.

2.1. Reduced space construction

Without loss of generality, we can assume the FOM is governed by the equation:

$$\dot{\boldsymbol{y}}(t) = \boldsymbol{f}(\boldsymbol{y}; \boldsymbol{\mu}), \tag{1}$$

where $\boldsymbol{y} \in \mathbb{R}^N$ is the state of the FOM and N is normally very large. $\boldsymbol{f}$ is the right-hand side (RHS) function configured by the parameter vector $\boldsymbol{\mu} = [\mu_1\ \mu_2\ \dots\ \mu_{N_m}] \in \mathbb{R}^{N_m}$.

The first step of generating a ROM is to construct the reduced space by finding a low-dimensional space defined by a projection matrix $\boldsymbol{V} \in \mathbb{R}^{N \times n}$, such that the reduced state $\boldsymbol{y}_r \in \mathbb{R}^n$ is given by projecting FOM state $\boldsymbol{y}$ onto the reduced space

$$\boldsymbol{y}_r = \boldsymbol{V}^{\boldsymbol{T}} \boldsymbol{y}. \tag{2}$$

2.1.1. Proper orthogonal decomposition

POD is a widely used method to construct the reduced space. Plenty of introductory literature for POD method is available (e.g., Lu et al., 2019). Here we will just briefly introduce it. The main ingredient of POD is a series of observations of the FOM states, which are usually called snapshots and take the form of

$$\boldsymbol{Y} = \left[\boldsymbol{y}^{(1)}\ \ \boldsymbol{y}^{(2)}\ \ \dots\ \ \boldsymbol{y}^{(N_s)}\right], \tag{3}$$

where N_s is the number of the available snapshots and $\boldsymbol{y}^{(i)}$ is the ith column in the snapshot matrix $\boldsymbol{Y}$.

Applying singular value decomposition (SVD) to $\boldsymbol{Y}$ yields

$$\boldsymbol{Y} = \boldsymbol{V}\Sigma\boldsymbol{U}^{\boldsymbol{T}}. \tag{4}$$

The left singular vectors $\boldsymbol{V} \in \mathbb{R}^{N \times k}$ can be used as the reduced space basis. This space will minimize the L^2 approximation error for the given snapshots. We can truncate $\boldsymbol{V}$ at its N_rth column. By doing this, the size of the reduced space (N_r) can be chosen freely. A large space can retain more information in the original full space, but it will make the ROM run slower during the online phase.

2.1.2. Conventional snapshot sampling

To increase the diversity of the observed FOM states in the snapshots, usually a hypercubic parameter space is predefined as

$$\mathcal{M} = [\mu_{1,\min}, \mu_{1,\max}] \times [\mu_{2,\min}, \mu_{2,\max}] \times \dots \times [\mu_{N_m,\min}, \mu_{N_m,\max}], \tag{5}$$

where $\mu_{i,\min}$ and $\mu_{i,\max}$ is the upper and the lower limit of the ith component of the system parameter vector $\boldsymbol{\mu}$. The parameter vectors $\boldsymbol{\mu}$ will be randomly selected from $\mathcal{M}$ and different initial value problems (IVPs) can be constructed based on them. In this case, FOM's states under different input parameters can be observed.

Next, we introduce two strategies for the sampling parameter values, static parameter sampling (SPS) and dynamic parameter sampling (DPS), respectively. For SPS, to construct m IVPs, m parameter vectors will be sampled from $\mathcal{M}$. For the ith IVP, we have

$$\dot{\boldsymbol{y}}(t) = \boldsymbol{f}\left(\boldsymbol{y}; \boldsymbol{\mu}^{(i)}\right), \tag{6}$$

where $\boldsymbol{\mu}^{(i)}$ is the ith parameter samples.

With DPS, to construct m IVPs where each IVP has K time steps, mK parameter vectors will be sampled from $\mathcal{M}$. Here, in this article, we use Hammersley Low Discrepancy Set (Hammersley and Handscomb, 1964) to take samples. For the ith IVP, we have

$$\dot{\boldsymbol{y}}(t) = \boldsymbol{f}\left(\boldsymbol{y}; \boldsymbol{\mu}^{(i)}(t)\right), \tag{7}$$

where

$$\boldsymbol{\mu}^{(i)}(t) = \mathrm{Interp}\left(\left\{\left(t_0, \boldsymbol{\mu}^{(i0)}\right), \left(t_1, \boldsymbol{\mu}^{(i1)}\right), \dots, \left(t_K, \boldsymbol{\mu}^{(iK)}\right)\right\}\right). \tag{8}$$

In equation (8), Interp$(\cdot)$ represent a linear interpolation of the points $t_j, \mu^{(ij)}$. Let δt be the time step size, $t_i = t_0 + i\delta t$. As we can see that DPS is K times more efficient in exploring the parameter space $\mathcal{M}$, which will make the constructed ROM perform better facing complex input parameters in its online phase.

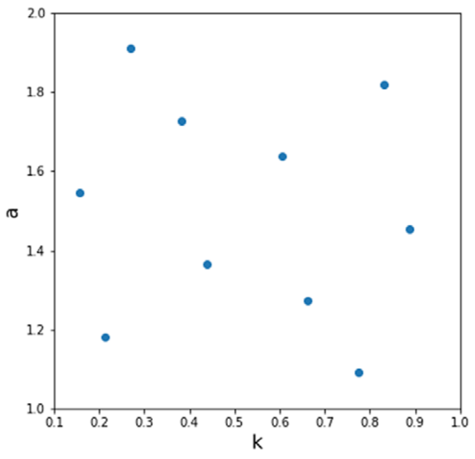
2.1.2.1. Example. Here, we use an example to show the sample distributions of the SPS and DPS methods, respectively. We consider an IVP,

$$\dot{y} = k(t)y + e^{a(t)}, \tag{9}$$

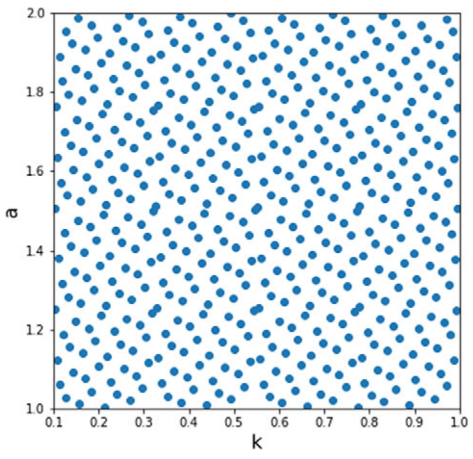
where $t \in (0,1)$. $k(t)$ and $a(t)$ are two input parameters of the system, and they are time-dependent. Their ranges are $[0.1,1]$ and $[1,2]$, respectively. We first use Hammersley Low Discrepancy Set to take samples in the parameter space $\mathcal{M}$. Then, we use the SPS and DPS methods to create the IVPs. By solving the IVPs, we obtain the solution snapshots. For each IVP, we collect 50 snapshots evenly on the time grid.

In Figures 1 and 2, we show the sample distributions in the parameter space and the solution space for the SPS and the DPS method, respectively. As observed, we can have a good sample distribution in the solution space with the SPS method. However, we have much fewer samples in the parameter space. In contrast, much more samples are taken in the parameter space with the DPS method. Nevertheless, the diversity of the sample system state is not optimal.

Through this toy example, we can see the advantage and disadvantage of the conventional sampling approaches for MOR. Later in this contribution, we propose a new sampling approach to ensure good sample distribution/diversity in both the parameter space and the solution space.

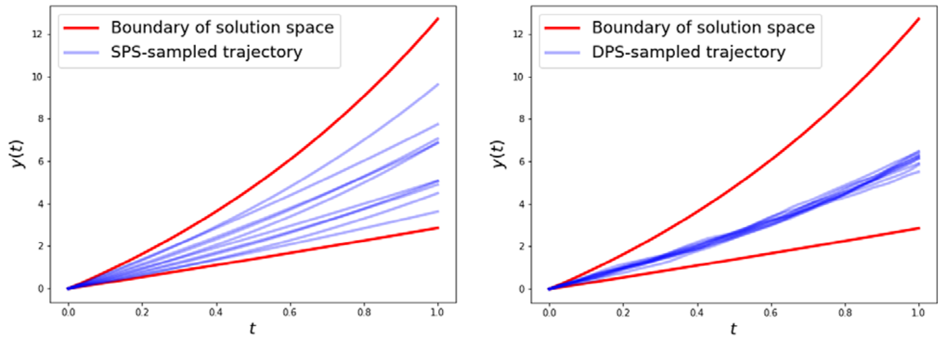


(a) SPS.



(b) DPS.

Figure 1. *The sample distribution in the parameter space $\mathcal{M}$.*



(a) SPS. (b) DPS.

Figure 2. *The sample distribution in the solution space.*

2.2. ROM identification

As introduced, the second step of ML-based MOR is model identification, which means finding the governing equation in the reduced space. The reduced governing equations are expected to have the form

$$\dot{\boldsymbol{y}}_{\boldsymbol{r}}(t) = \boldsymbol{f}_{\boldsymbol{r}}(\boldsymbol{y}_r; \boldsymbol{\mu}), \tag{10}$$

where $\boldsymbol{f}_{\boldsymbol{r}}$ is the low-dimensional representation of $\boldsymbol{f}$.

Different methods can be used to achieve this. In this article, artificial neural network (ANN) and OpInf will be briefly introduced.

2.2.1. Artificial neural network

The ROM we employ in this article is a dynamic model evolving on a discrete-time grid from t_0 to t_{end} with time step δt:

$$\frac{\boldsymbol{y}_{r,i+1} - \boldsymbol{y}_{r,i}}{\delta t} = \boldsymbol{g}\left(\boldsymbol{y}_{r,i}, \mu_i\right), \tag{11}$$

where $\boldsymbol{y}_{r,i}$ means $\boldsymbol{y}_{\boldsymbol{r}}(t = t_i)$, and $\boldsymbol{g}(\cdot)$ is the approximation to the unknown reduced RHS $\boldsymbol{f}_{\boldsymbol{r}}(\cdot)$.

To learn the mapping between states of two consecutive time steps, a multilayer perceptron (MLP) is used as $\boldsymbol{g}(\cdot)$ in equation (11). In this case, the inputs to the neural network are the current state of the system $\boldsymbol{y}_{r,i}$ and $\boldsymbol{\mu}_i$, and the predicted reduced state $\widehat{\boldsymbol{y}}_{r,i+1}$ is calculated as

$$\widehat{\boldsymbol{y}}_{r,i+1} = \boldsymbol{y}_{r,i} + \delta t \boldsymbol{g}_{\text{ee}}\left(\boldsymbol{y}_{r,i}, \boldsymbol{\mu}_i\right). \tag{12}$$

This is similar to an Explicit Euler integrator, so we denote the MLP's function as $\boldsymbol{g}_{\text{ee}}\left(\boldsymbol{y}_{r,i}, \mu_i\right)$. We call such a network explicit Euler neural network (EENN) (Pan and Duraisamy, 2018; Regazzoni et al., 2019; Zhuang et al., 2021). From equation (12), we know that an EENN is essentially a variant of residual network (He et al., 2016) which learns the increment between two system states instead of learning to map the new state directly from the old state. This leads to a potential advantage of EENNs that we can use the deeper network for approximating more complex nonlinearity without suffering too much from network degeneration.

The algorithm for training such an EENN is provided in Algorithm 1. In this contribution, all MLPs consist of one input layer, two hidden layers, and one output layer. The activation function between each two consecutive layers is rectified linear unit (ReLU) function (Glorot et al., 2011). We use Pytorch (Paszke et al., 2017) as the platform to build and train the networks. The learning rate decay and Early Stopping (Prechelt, 1998) are applied to facilitate the training.

Algorithm 1. Training of EENN

```
Require: $\mathbf{y}_{r,i}, \boldsymbol{\mu}_i, \mathbf{y}_{r,i+1}$
1: while loss > loss_tol do
2:     $\hat{\boldsymbol{y}}_{r,i+1} = \boldsymbol{y}_{r,i} + \delta t \boldsymbol{g}_{\text{ee}}(\boldsymbol{y}_{r,i}, \boldsymbol{\mu}_i)$
3:     loss = MSE$(\hat{\boldsymbol{y}}_{r,i+1}, \boldsymbol{y}_{r,i+1})$
4:     Use backpropagation to update the weights and biases of the network $g_{\text{ee}}$
5: end while
```

2.2.2. *Operator inference*

Besides the purely data-driven ROM identification method introduced in Section 2.2.1, we briefly introduce a physics-informed ROM identification method, OpInf (Peherstorfer and Willcox, 2016).

The OpInf method uses a hypothetical form for the ROM's governing equation:

$$\dot{\boldsymbol{y}}_r = \boldsymbol{F_1}\boldsymbol{y}_r + \boldsymbol{F_B}\boldsymbol{\mu} + \boldsymbol{F_C} + \boldsymbol{F_2}\boldsymbol{y}_r \otimes \boldsymbol{y}_r + \boldsymbol{F_N}\boldsymbol{\mu} \otimes \boldsymbol{y}_r, \tag{13}$$

where $\boldsymbol{F_1}, \boldsymbol{F_2}, \boldsymbol{F_B}, \boldsymbol{F_C}$ and $\boldsymbol{F_N}$ are operators to be inferred. The operation $\otimes$ is the Kronecker product:

$$\boldsymbol{y}_r \otimes \boldsymbol{y}_r = \begin{bmatrix} y_r^{(1)} \\ y_r^{(2)} \\ \vdots \\ y_r^{(N_r)} \end{bmatrix} \otimes \begin{bmatrix} y_r^{(1)} \\ y_r^{(2)} \\ \vdots \\ y_r^{(N_r)} \end{bmatrix} = \begin{bmatrix} y_r^{(1)} y_r^{(1)} \\ y_r^{(1)} y_r^{(2)} \\ \vdots \\ y_r^{(1)} y_r^{(N_r)} \\ y_r^{(2)} y_r^{(2)} \\ y_r^{(2)} y_r^{(3)} \\ \vdots \\ y_r^{(N_r)} y_r^{(N_r)} \end{bmatrix}. \tag{14}$$

As we see, OpInf assumes a quadratic dependency of RHS on the system state. OpInf can also be applied to systems that are not quadratic, as long as they can be mapped to a quadratic form through variable transformations (see Qian et al., 2020).

To infer the unknown operators, we first construct the matrices

$$\begin{aligned} \boldsymbol{Y}_{r,\text{in}} &= \left[\boldsymbol{y}_{r,0} \ \ \boldsymbol{y}_{r,1} \ \ \cdots \ \ \boldsymbol{y}_{r,K-1}\right], \\ \boldsymbol{Y}_{r,\text{out}} &= \left[\boldsymbol{y}_{r,1} \ \ \boldsymbol{y}_{r,2} \ \ \cdots \ \ \boldsymbol{y}_{r,K}\right], \\ \boldsymbol{U}_{\text{in}} &= \left[\boldsymbol{\mu}_1 \ \ \boldsymbol{\mu}_2 \ \ \ldots \ \ \boldsymbol{\mu}_K\right], \end{aligned} \tag{15}$$

where K is the number of time steps. Using the matrices and the assumed RHS, we can construct a least square problem,

$$\arg\min_{\boldsymbol{O}} \|\boldsymbol{D}\boldsymbol{O} - \boldsymbol{R}\|_2, \tag{16}$$

where

$$\begin{aligned} \boldsymbol{R} &= \frac{\boldsymbol{Y}_{r,\text{out}}^T - \boldsymbol{Y}_{r,\text{in}}^T}{\delta t}, \\ \boldsymbol{D} &= \left[\boldsymbol{Y}_{r,\text{in}}^T \ \ \boldsymbol{U}_{\text{in}}^T \ \ \boldsymbol{I} \ \ (\boldsymbol{Y}_{r,\text{in}} \otimes \boldsymbol{Y}_{r,\text{in}})^T \ \ (\boldsymbol{U}_{\text{in}} \otimes \boldsymbol{Y}_{r,\text{in}})^T\right]. \end{aligned} \tag{17}$$

By solving the least square problem (equation (16)), we get the matrix $\boldsymbol{O}$. We can unpack the matrix $\boldsymbol{O}$ as

$$\boldsymbol{O} = \left[\boldsymbol{F}_1^T \ \ \boldsymbol{F}_B^T \ \ \boldsymbol{F}_C^T \ \ \boldsymbol{F}_2^T \ \ \boldsymbol{F}_N^T\right]. \tag{18}$$

In practice, we solve the least square problem with regularization to produce a stable ROM, and for more details see McQuarrie et al. (2021).

3. AL for ML-Based MOR

As introduced in Section 2.1.2, for the conventional ML-based MOR approaches, snapshots are collected up-front. This could cause the problem that not all important information is captured in the prepared snapshots that are used to construct the ROM. We propose a solution based on the concept of the pool-based AL (Settles, 2009). The solution mainly contains the following steps: (a) we first use the JSS method to prepare a large data pool containing diverse joint samples, (b) the current ROM is validated, and the validation result is used to select the samples that are considered most valuable for the current ROM, (c) new training data are generated by the FOM and added to the current training data, and (d) the ROM is updated. The corresponding steps in the whole workflow are sketched in Figure 3.

3.1. Joint space sampling

In this section, we will introduce the new sampling method JSS. The proposed method will be used to create the data pool and the validation samples, which are necessary preparation for the AL algorithm.

3.1.1. Estimating the reduced solution space

In order to take the joint samples, we need first to estimate a reduced solution space. With the predefined parameter space $\mathcal{M}$, the boundaries of the reduced solution space will be estimated as follows:

1. Use SPS to create m IVPs for the FOM. In this article, Hammersley Set (Hammersley and Handscomb, 1964) is used to take samples in the parameter space. Each IVP is integrated from time t_0 to time t_{end} with K time steps.
2. Solving all the FOM problems will produce a snapshot matrix $\boldsymbol{Y}_{\text{estimate}} = \left[\boldsymbol{y}^{(1)} \;\; \boldsymbol{y}^{(2)} \;\; \dots \;\; \boldsymbol{y}^{(m\times K)}\right]$ containing $m \times K$ FOM state vectors.
3. Apply POD on $\boldsymbol{Y}_{\text{estimate}}$ to construct the reduced space.

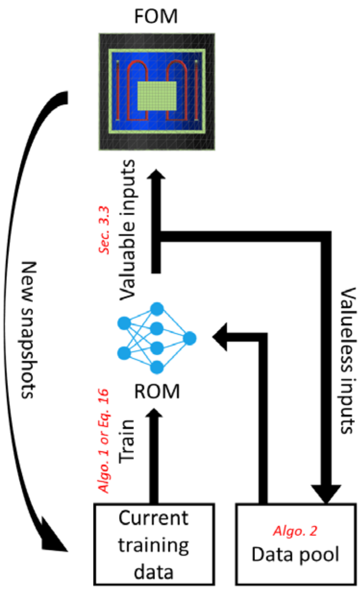


Figure 3. *The workflow of the AL-based MOR.*

4. Project $\boldsymbol{Y}_{\text{estimate}}$ onto the reduced space to get its low-dimensional representation:

$$\boldsymbol{Y}_{r,\text{estimate}} = \boldsymbol{V}^T \boldsymbol{Y}_{\text{estimate}}. \tag{19}$$

5. Find the minimum and the maximum entries for each POD component in $\boldsymbol{Y}_{\text{estimate}}$ and denote them as:

$$\left\{ y_{r,\min}^{(1)}, y_{r,\max}^{(1)}, y_{r,\min}^{(2)}, y_{r,\max}^{(2)}, \ldots, y_{r,\min}^{(N_r)}, y_{r,\max}^{(N_r)} \right\}, \tag{20}$$

where N_r is the size of the ROM.

6. The estimated space $\mathcal{Y}$ is:

$$\mathcal{Y} = \left[y_{r,\min}^{(1)}, y_{r,\max}^{(1)} \right] \times \left[y_{r,\min}^{(2)}, y_{r,\max}^{(2)} \right] \times \ldots \times \left[y_{r,\min}^{(N_r)}, y_{r,\max}^{(N_r)} \right]. \tag{21}$$

In the workflow, m can be selected as any positive integer. Since the reduced basis $\boldsymbol{V}$ computed in the step will be used as the initial reduced basis later, the greater m is, the better the initial reduced basis will be. As explained in Section 2.1.2, we can have a relatively good observation in the solution space with the SPS method. Therefore, we use the SPS-sampled state to estimate the reduced solution space. Besides, one can also use minimal intrusiveness to the full-order problem to improve the estimate. Specifically, the parameter configurations that most likely drive the system to its extreme status (boundaries of the solution space) can be included in the samples.

A joint sample $\boldsymbol{J}_{\text{sample}}$ taken from the joint space $\mathcal{J} = \mathcal{Y} \times \mathcal{M}$ has the form of $\boldsymbol{J}_{\text{sample}} = \left[\boldsymbol{y}_{r,\text{sample}} \ \ \boldsymbol{\mu}_{\text{sample}} \right]$. We can use $\boldsymbol{J}$, $\boldsymbol{V}$ and the FOM solver to perform a one-step simulation whose initial condition is $\boldsymbol{y}_{r,\text{sample}}$ and step size is δt. The simulation result will form a one-step trajectory in $\mathcal{Y}$. We can use such joint samples to prepare the data pool and the validation samples needed for the AL algorithm.

3.1.2. Loosen and trim the estimated solution space

The joint space $\mathcal{J}$ as created in Section 3.1 is a hyper-cubic space (see example in Figure 4a). We will loosen and trim $\mathcal{Y}$ to make it closer to the true reduced solution space. The first step is loosening. For this purpose, we introduce an expansion ratio β:

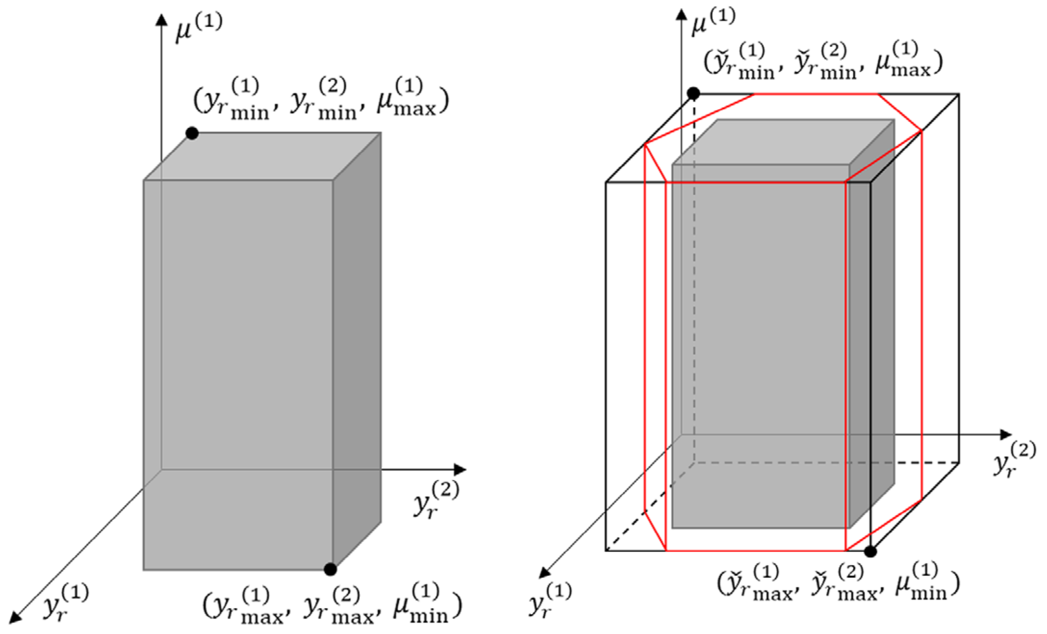
$$\begin{aligned} \Delta y_r^{(i)} &= y_{r,\max}^{(i)} - y_{r,\min}^{(i)}, \\ \breve{y}_{r,\min}^{(i)} &= y_{r,\min}^{(i)} - \beta \Delta y_r^{(i)}, \\ \breve{y}_{r,\max}^{(i)} &= y_{r,\max}^{(i)} + \beta \Delta y_r^{(i)}. \end{aligned} \tag{22}$$

We use $\breve{y}_{r,\min}^{(i)}$ and $\breve{y}_{r,\max}^{(i)}$ as the lower and upper limits for the loosened space $\breve{\mathcal{Y}}$. In our practice, $\beta \in (0,1)$ can be selected to loosen the initial estimation. To trim the space, we apply two conditions to a reduced state $\boldsymbol{y}_r$ sampled from $\breve{\mathcal{Y}}$:

$$\begin{aligned} \text{MAX}(\boldsymbol{V}\boldsymbol{y}_r) &< y_{\max}, \\ \text{MIN}(\boldsymbol{V}\boldsymbol{y}_r) &> y_{\min}, \end{aligned} \tag{23}$$

where $\text{MAX}(\cdot)$ and $\text{MIN}(\cdot)$ means the maximum and minimum entry of a vector/matrix. The value $y_{\min}$ and $y_{\max}$ can be either defined by empirical values of the physics quantity y or by $\text{MAX}(\boldsymbol{Y}_{\text{estimate}})$ and $\text{MIN}(\boldsymbol{Y}_{\text{estimate}})$. In this article, $\text{MAX}(\boldsymbol{Y}_{\text{estimate}})$ and $\text{MIN}(\boldsymbol{Y}_{\text{estimate}})$ is used. We here denote the final feasible sampling space for reduced states as $\mathcal{Y}^*$, and the final joint space as $\mathcal{J}^* = \mathcal{Y}^* \times \mathcal{M}$. A graphical example for joint space $\mathcal{J}^*$ is given in Figure 4b.

Finally, we summarize how to collect N_s samples with the proposed JSS method in Algorithm 2.


$\mu^{(1)}$
$(y_{r\,\mathrm{min}}^{(1)}, y_{r\,\mathrm{min}}^{(2)}, \mu_{\mathrm{max}}^{(1)})$
$y_r^{(2)}$
$y_r^{(1)}$
$(y_{r\,\mathrm{max}}^{(1)}, y_{r\,\mathrm{max}}^{(2)}, \mu_{\mathrm{min}}^{(1)})$
$\mu^{(1)}$
$(\breve{y}_{r\,\mathrm{min}}^{(1)}, \breve{y}_{r\,\mathrm{min}}^{(2)}, \mu_{\mathrm{max}}^{(1)})$
$y_r^{(2)}$
$y_r^{(1)}$
$(\breve{y}_{r\,\mathrm{max}}^{(1)}, \breve{y}_{r\,\mathrm{max}}^{(2)}, \mu_{\mathrm{min}}^{(1)})$

(a) An estimated joint space $\mathcal{J}$ where $N_r = 2$ and $N_u = 1$.

(b) The joint space $\mathcal{J}$ being loosened (black solid lines) and trimmed (red solid/dashed lines).

Figure 4. *The evolution of the joint space.*

Algorithm 2. Generation of a set of random joint samples

```
Require: $\mathcal{M}, N_s, m$, FOM
 1: Create a sample set $M = \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_m\}$ in $\mathcal{M}$
 2: Perform SPS with $m$ parameter samples and FOM to get $\boldsymbol{Y}_{\mathrm{estimate}}$
 3: Apply POD to $\boldsymbol{Y}_{\mathrm{estimate}}$ to get $\boldsymbol{V}$ and $\boldsymbol{X}_{\mathrm{estimate}}$
 4: Find the upper and lower limits for each POD component
 5: Create hyper-cubic solution space $\mathcal{J}$
 6: Loosen the space $\mathcal{J}$ to get $\breve{\mathcal{J}}$
 7: Determine $y_{\mathrm{max}}$ and $y_{\mathrm{min}}$
 8: $i = 0, \boldsymbol{I} = []$
 9: while $i < N_s$ do
10:     Generate a random sample $\boldsymbol{y}_{r,\mathrm{sample}}$ in $\breve{\mathcal{Y}}$
11:     if MAX$\left(\boldsymbol{V}\boldsymbol{y}_{r,\mathrm{sample}}\right) < y_{\mathrm{max}}$ and MIN$\left(\boldsymbol{V}\boldsymbol{y}_{r,\mathrm{sample}}\right) > y_{\mathrm{min}}$ then
12:         Generate a random sample $\mu_{\mathrm{sample}}$ in $\mathcal{M}$
            $\boldsymbol{J}_{\mathrm{sample}} = \begin{bmatrix} \boldsymbol{\mu}_{\mathrm{sample}} \\ \boldsymbol{y}_{r,\mathrm{sample}} \end{bmatrix}$
            $\boldsymbol{I} = \boldsymbol{I} \cup \boldsymbol{J}_{\mathrm{sample}}$
            $i{+}{+}$
13: end if
14: end while
15: return $I$
```

3.2. ROM validation

Finding adequate metrics to validate a ROM is challenging. When the training and test data sets are collected up front, the ROM could be biased toward the data set and lead to a lack of generalization for the model. To make our data-driven ROM more reliable, we must design a more generalized validating procedure.

3.2.1. Accuracy

Here, we define the relative error of the ROM in a one-step prediction as e:

$$e = \frac{\|\boldsymbol{y} - \widehat{\boldsymbol{y}}\|}{\|\boldsymbol{y}\|}. \tag{24}$$

We furthermore define ROM error as e. However, the ROM is very likely to be used for a long-term prediction and such a one-step accuracy still cannot reflect the performance of the ROM. Therefore, we will introduce the confidence of the ROM.

3.2.2. Confidence

If the ROM error is τ at confidence level η, we can construct such an inequality:

$$\Pr(e \le \tau) > \eta, \tag{25}$$

that is, the probability that the error e smaller than τ is greater than η, where η and τ is a pair of confidence and accuracy indicator that we are seeking.

We assume we have s independent tests in the validator. Then we can calculate the observed confidence $p(\tau^*)$ of a given ROM error τ^* using:

$$\begin{aligned} p(\tau^*) &= \frac{1}{s}\sum_{i=1}^{s} L(e_i \le \tau^*), \\ L(e_i \le \tau^*) &= \begin{cases} 1 & e_i \le \tau^* \\ 0 & e_i > \tau^* \end{cases}. \end{aligned} \tag{26}$$

However, we can only include a limited number of independent tests into the validator. This means, with a given τ^*, the $p(\tau^*)$ computed from the validator should also have deviation ϵ and its corresponding confidence $1-\sigma$ from the true confidence $\widehat{p}(\tau^*)$. Thus, we obtain another inequality:

$$\Pr(|p(\tau^*) - \widehat{p}(\tau^*)| < \epsilon) > 1-\sigma, \tag{27}$$

where $\widehat{p}(\tau^*)$ is the true confidence. If we predefine the deviation ϵ and the confidence $1-\sigma$ of the validator, the smallest number of the independent tests needed by the validator can be computed using Chernoff's inequality (Alippi, 2016):

$$s \ge s_{\text{PAC}} = \frac{1}{2\epsilon^2} ln\left(\frac{2}{\sigma}\right). \tag{28}$$

Let $\overline{\tau}^*$ be the smallest τ^* satisfying:

$$p(\overline{\tau}^*) = 1. \tag{29}$$

Since Inequality 27 holds for all τ^*, it also holds for $\overline{\tau}^*$:

$$\Pr(|1 - \widehat{p}(\overline{\tau}^*)| < \epsilon) > 1-\sigma. \tag{30}$$

That is the inequality:

$$|1 - \widehat{p}(\overline{\tau}^*)| < \epsilon \tag{31}$$

holds with the probability $1-\sigma$. We further get:

$$1-\epsilon < \widehat{p}(\overline{\tau}^*) < 1 \tag{32}$$

holds with probability $1-\sigma$.

Finally, the ROM error τ and its corresponding confidence η are therefore $\overline{\tau}^*$ and $1-\epsilon$, respectively. Since Inequality 32 holds with probability $1-\sigma$ and S_{PAC} is negatively correlative to σ, we know that the more test samples we have, the more reliable the computed true confidence will be.

We denote such a validator as:

$$p(\tau^*) = \text{PAC}(\tau^*; \text{ROM}). \tag{33}$$

We can predefine a variable $\overline{\tau}^*_{\text{design}}$ and compute $p\left(\overline{\tau}^*_{\text{design}}\right)$ in iterations for convergence check.

3.3. *Error estimator*

The ROM will go through all tests in the validator constructed as described in Section 3.2. We denote the ROM error snapshots as:

$$\boldsymbol{e} = \left[e^{(1)} \quad e^{(2)} \quad \dots \quad e^{(s)}\right], \tag{34}$$

where $e^{(i)}$ is e computed in the ith test. We assume there is a function:

$$e^{(i)} = \mathcal{E}\left(\boldsymbol{J}^{(i)}\right). \tag{35}$$

Such a function can be used as the error estimator to estimate the ROM error with a new input $\boldsymbol{J}^*$ which is excluded from the test inputs $\boldsymbol{I} = \left\{\boldsymbol{J}^{(1)}, \boldsymbol{J}^{(2)}, .., \boldsymbol{J}^{(s)}\right\}$. Here, we use GPR (Williams and Rasmussen, 1996) for interpolation, and the input of the interpolator is the joint input $\boldsymbol{J}$ of the ROM and the output is the estimated error. In our problem, our training dataset is the test data in the PAC validator:

$$\boldsymbol{D} = (\boldsymbol{I}, \boldsymbol{e}) = \left\{\left(\boldsymbol{J}^{(i)}, e^{(i)}\right) | i = 1, 2, \dots, s\right\}. \tag{36}$$

According to the theory of GPR, the predicted ROM error $\widehat{e}^*$ for the new joint input $\boldsymbol{J}^*$ is:

$$\widehat{e}^* = \boldsymbol{K}(\boldsymbol{J}^*, \boldsymbol{I})\boldsymbol{K}(\boldsymbol{I}, \boldsymbol{I})^{-1}\boldsymbol{e}, \tag{37}$$

where $\boldsymbol{K}$ is a covariance matrix whose entries can be computed by a preselected covariance function $\kappa(\boldsymbol{J}^*, \boldsymbol{J}_i)$. Here we use radial-basis function (RBF), which is also called squared-exponential function or Gaussian kernel and is one of the most popular choices for the covariance function:

$$\kappa\left(\boldsymbol{J}^*, \boldsymbol{J}^{(i)}\right) = \sigma^2 \exp\left(-\frac{1}{2l^2}\left\|\boldsymbol{J}^* - \boldsymbol{J}^{(i)}\right\|^2\right), \tag{38}$$

where the amplitude σ and the lengthscale l are the hyperparameters which can be tuned. RBF is appropriate when the simulated function is expected to be smooth.

Besides GPR, other interpolators or approximators, such as Spline Interpolation (Habermann and Kindermann, 2007), Radial-basis-function Interpolation (Broomhead and Lowe, 1988), and ANN can be used to construct the estimator.

3.4. *AL algorithm*

Until now, the necessary theory and components of the AL algorithm have been fully introduced. In this section, we summarize the workflow algorithmically.

First, we need to prepare the initial data pool $\boldsymbol{I}_{\text{all}}$, the PAC validator containing $\boldsymbol{I}_{\text{PAC}}$ and $\boldsymbol{O}_{\text{PAC}}$ using Algorithm 2 and the FOM. With the data pool and the PAC validator on hands, the AL algorithm can be designed as Algorithm 3. Besides the algorithm, the flow chart of the proposed workflow is given in Figure 5.

Algorithm 3. AL for the ROM

```
Require: $\boldsymbol{Y}_{\text{estimate}}$, $\boldsymbol{V}$, $\overline{\tau}^*_{\text{design}}$, $\Delta\overline{\tau}^*_{\text{tol}}$, $\boldsymbol{I}_{\text{all}}$, $\text{PAC}(\tau^*,\text{ROM})$, $\boldsymbol{I}_{\text{PAC}}$, $\boldsymbol{O}_{\text{PAC}}$
 1: Define the snapshot increment $\Delta s$
 2: Randomly select $\Delta s$ samples from $\boldsymbol{I}_{\text{all}}$ as the initial training input samples $\boldsymbol{I}_{\text{train}}$ and generate the
    training output samples $\boldsymbol{O}_{\text{train}}$ using the FOM solver
 3: Train the ROM using $\boldsymbol{I}_{\text{train}}$, $\boldsymbol{O}_{\text{train}}$ and $\boldsymbol{V}$
 4: $p\left(\overline{\tau}^*_{\text{design}}\right) = \text{PAC}\left(\overline{\tau}^*_{\text{design}};\text{ROM}\right)$ and $\overline{\tau}^*_{\text{old}} = \overline{\tau}^*$
 5: Compute the error snapshots $\boldsymbol{e}$ using $\boldsymbol{O}_{\text{PAC}}$
 6: while $p\left(\overline{\tau}^*_{\text{design}}\right) < 1$ do
 7:     Fit the GPR estimator $\mathcal{E}$ using $\boldsymbol{e}$ and $\boldsymbol{I}_{\text{PAC}}$
 8:     Evaluate $\mathcal{E}$ on $\boldsymbol{I}_{\text{all}} \setminus \boldsymbol{I}_{\text{train}}$, denote the estimation as $\boldsymbol{e}_{\text{pool}}$
 9:     Find $\Delta s$ largest $e_i$ and their corresponding input samples $\Delta\boldsymbol{I} = \left\{\boldsymbol{J}^{(1)}_{\max}, \boldsymbol{J}^{(2)}_{\max}, \dots, \boldsymbol{J}^{(\Delta s)}_{\max}\right\}$
10:     $\boldsymbol{I}_{\text{train}} = \boldsymbol{I}_{\text{train}} \cup \Delta\boldsymbol{I}$
11:     Update $\boldsymbol{O}_{\text{train}}$ using the enriched $\boldsymbol{I}_{\text{train}}$ and the FOM
12:     Update $\boldsymbol{V}$ using $\boldsymbol{Y}_{\text{estimate}} \cup \boldsymbol{O}_{\text{train}}$
13:     Train the ROM using $\boldsymbol{I}_{\text{train}}$, $\boldsymbol{O}_{\text{train}}$ and $\boldsymbol{V}$
14:     $p\left(\overline{\tau}^*_{\text{design}}\right) = \text{PAC}\left(\overline{\tau}^*_{\text{design}};\text{ROM}\right)$ and compute $\overline{\tau}^*$
15:     Update the error snapshots $\boldsymbol{e}$
16:     $\overline{\tau}^*_{\text{old}} = \overline{\tau}^*$
17:     if $|\overline{\tau}^* - \overline{\tau}^*_{\text{old}}| < \Delta\overline{\tau}^*_{\text{tol}}$ then
18:         Break
19:     end if
20: end while
```

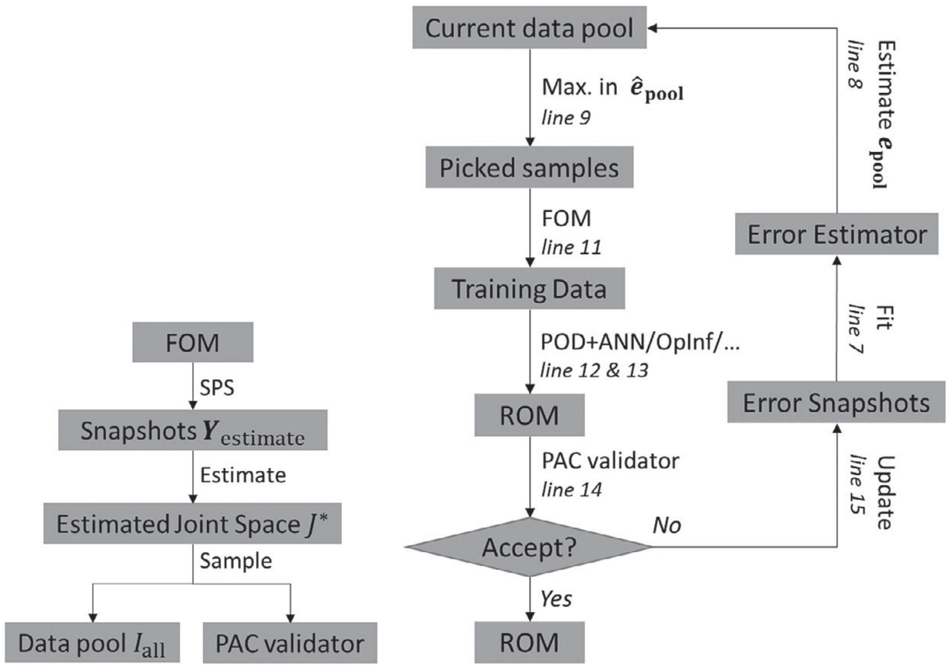


Figure 5. *The flow charts of the preparation and iteration phase of the AL-based MOR. The line numbers in (b) match those in Algorithm 3.*

Regarding Algorithm 3, we would like to clarify that the snapshots in $\boldsymbol{Y}_{\text{estimate}}$ are not used for the model identification step, only for the construction of the initial projection matrix $\boldsymbol{V}$. The neural network is therefore trained exclusively with one-step trajectories. We notice that the inclusion of $\boldsymbol{Y}_{\text{estimate}}$ does not improve the ROM's quality but considerably increases the training time for the neural network.

As seen in the algorithm, the reduced basis $\boldsymbol{V}$ is recomputed during the iterations. In principle, this should influence the estimate of the reduced solution space and require rebuilding the PAC validator using the FOM solver. However, in practice, the initial basis $\boldsymbol{V}$ produced by $\boldsymbol{Y}_{\text{estimate}}$ should already have a relatively low projection error. The refined reduced basis only slightly changes the boundary of the reduced solution space. We can still use $\mathcal{Y}$ as the space for sampling the reduced state.

In Algorithm 3, we see another convergence check $|\overline{\tau}^* - \overline{\tau}^*_{\text{old}}| < \Delta\overline{\tau}^*_{\text{tol}}$ besides $p\left(\overline{\tau}^*_{\text{design}}\right)$. Additionally, we check the improvement for the best ROM accuracy between two iterations. If the difference between them is already smaller than the predefined value, the algorithm will be considered to be in a low-efficiency status and should be stopped. Since there are many factors influencing the ROM accuracy other than the amount of training data, for example, ROM structure, ROM dimension, and the way of collecting training data, other possible modification should be considered and adopted to improve the ROM accuracy.

4. Numerical Examples

In this section, three numerical models will be used to evaluate the performance of the proposed algorithm against the conventional ML-based MOR. We integrate ANN-MOR and OpInf-MOR into the proposed method. The gained improvement is shown respectively. In the rest of the article, we use the following notation:

- ML-ROM: the ROM is constructed in the conventional workflow of ML-based MOR. The training data are collected by the DPS method, and the ROM is constructed by either POD + ANN or POD + OpInf.
- AL-ROM: the ROM is constructed by the proposed AL algorithm in Algorithm 3, where a large data pool is firstly prepared by the JSS method, then the algorithm selectively generates the training data. The ROM is constructed by either POD + ANN or POD + OpInf.

4.1. 2-D linear thermal block

The governing equation of the test system is:

$$\begin{aligned}
&\frac{\partial T(x,y,t)}{\partial t} = k_x \frac{\partial^2 T(x,y,t)}{\partial x^2} + k_y \frac{\partial^2 T(x,y,t)}{\partial y^2},\\
&T(0,y,t) = T_{\text{left}}, \quad T(10,y,t) = T_{\text{right}},\\
&T(x,-10,t) = T_{\text{down}}, \quad T(x,10,t) = T_{\text{up}},\\
&T(x,y,0) = 20,
\end{aligned} \tag{39}$$

where $x \in (0,10), y \in (-10,0), t \in (0,2]$ and the static parameters are given by $k_x = k_y = 1$. The controllable parameters $T_{\text{left}}, T_{\text{right}}, T_{\text{down}}$ and T_{up} (°C) have all the same feasible range $[20,1,000]$. Thus, for this problem $\mathcal{M} = [20,1,000] \times [20,1,000] \times [20,1,000] \times [20,1,000]$. The full-order model is discretized by the finite difference method. The square region is divided into $50 \times 50 = 2,500$ cells. We consider 100-time steps uniformly distributed.

For the AL algorithm, we first need to estimate the reduced solution space. Thus, $m = 20$ samples are taken from $\mathcal{M}$. The 20 constant-input IVPs are constructed with the sampled parameters. $\boldsymbol{Y}_{\text{estimate}}$ has totally 2,020 state vectors. The initial reduced space is constructed by 20 POD basis using $\boldsymbol{Y}_{\text{estimate}}$. The low-dimensional projection of those state vectors is then used to perform the space estimate. After estimating, loosening with $\beta = 10\%$ and trimming, we create the joint space $\mathcal{J}^* = \mathcal{Y}^* \times \mathcal{M}$ and use random sampling to get 40,000 joint samples for $\boldsymbol{I}_{\text{all}}$.

We define $\varepsilon = 3\%$ and $\sigma = 1\%$ for the validation step. Based on equation (28), we know 2,944 samples are needed by the PAC validator. The confidence level that will be investigated is $\eta_{\text{design}} = 97\%$. We pick $\overline{\tau}^*_{\text{design}} = 1\%$ and $\Delta\overline{\tau}^*_{\text{tol}} = 0.01\%$. In conclusion, we are looking for a ROM whose 97%-confident ROM error is 1%.

4.1.1. AL for ANN-ROM

In this section, we present the investigation for enhancing the ANN-MOR with our methods. The ROM has an architecture of EENN, whose MLP has [24, 80, 80, 20] neurons.

As seen in Table 1, the AL algorithm does not converge to the prescribed accuracy after 10 iterations. However, it meets the other stopping criteria $\Delta\overline{\tau}^*_{\text{tol}} \leq 0.01\%$. As we described before, when $\Delta\overline{\tau}^*_{\text{tol}} \leq 0.01\%$, we consider that continuing the iteration will not refine the ROM efficiently.

For the conventional workflow, we use DPS to create 50 IVPs, and they produce 50 solution trajectories. A total of 101 outputs from each IVP can build 100 input–output samples, therefore, we totally have 5,000 training samples. To construct the POD space, besides the sampled 5,000 snapshots, we also include $\boldsymbol{Y}_{\text{estimate}}$.

Once the ROMs are built, we use another PAC validator to compute the 97%-confident ROM error of the AL-ROM and ML-ROM. The results are given in Table 2.

It is observed that with the same amount of training samples, the two performance indicators $\overline{\tau}^*$ and $p\left(\overline{\tau}^*_{\text{design}}\right)$ of the conventional ML ROM are both much worse than the ROM trained in the AL algorithm. Based on the PAC theory, the ROM constructed with the conventional approach is not expected to have a better performance than the AL-ROM.

To further validate the ROMs, we designed two additional test cases. In Test 1, four boundary temperatures are described by four different time-dependent functions, respectively. The function curves are given in Figure 6. We pick four different elements as the observation positions. In Figure 7a,b, we present the predicted trajectories by the compared ROMs. In the trajectory figures, it is observed that both ROMs' predictions are very close to the reference. In Figure 8, we show the error fields at the final time

Table 1. *PAC scores in the AL algorithm.*

Iteration	Number of samples	$\overline{\tau}^*$ (%)	$p\left(\overline{\tau}^*_{\text{design}}\right)$ (%)
1	500	5.30	6.05
2	1,000	4.40	42.63
3	1,500	3.00	88.04
4	2,000	2.50	95.35
5	2,500	2.40	95.79
6	3,000	2.53	96.06
7	3,500	2.20	96.50
8	4,000	1.87	96.67
9	4,500	1.83	96.77
10	5,000	1.83	96.64

Note. The ROM is constructed with POD + ANN.

Table 2. *97%-confident ROM error ($\overline{\tau}^*$) by another PAC validator.*

Number of samples	AL-ROM	ML-ROM
5,000	1.83%	8.97%

Note. The ROMs are constructed with POD + ANN.

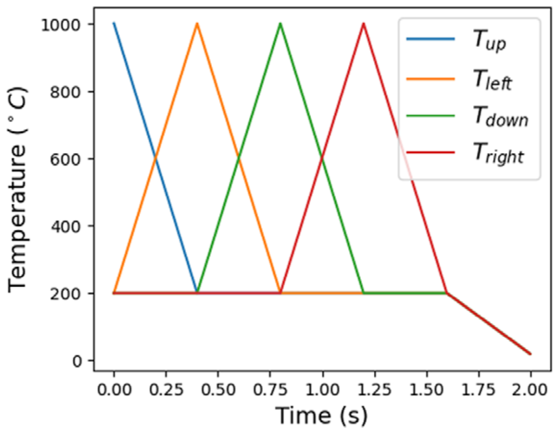
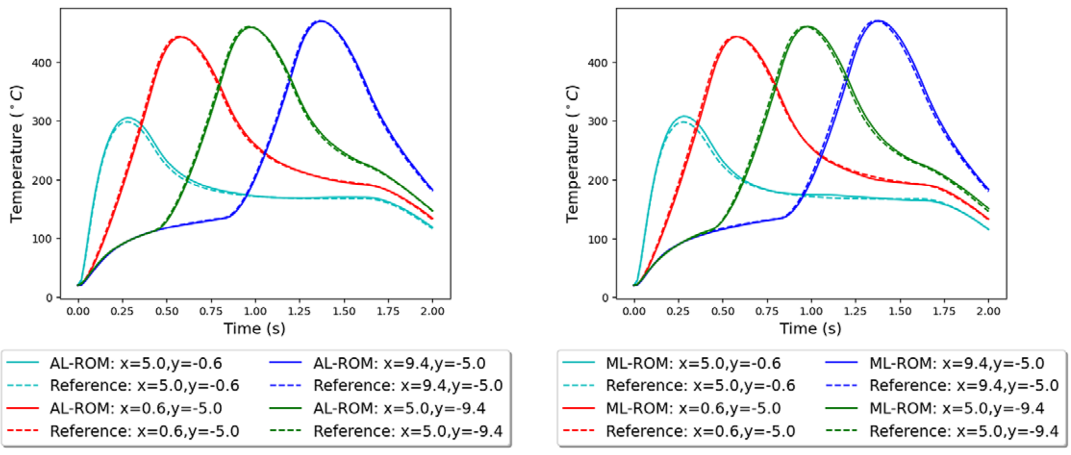


Figure 6. *Time-dependent inputs to the 2-D thermal problem for Test 1.*



(a) AL-ROM. (b) ML-ROM.

Figure 7. *Linear thermal block: the FOM and ROM solution in Test 1. The ROM is constructed with POD + ANN.*

step. To compute the error field, we first lift the reduced solution to the full space using $\boldsymbol{V}$ and calculate the absolute error between the lifted solution and the FOM solution. We can see that the AL-ROM's error is lower overall than ML-ROM's.

In Test 2, the four boundary temperatures are assigned with the same constant value $T_{\text{left}} = T_{\text{right}} = T_{\text{down}} = T_{\text{up}} = 1{,}000^{\circ}\text{C}$. The trajectories are presented in Figure 9a,b. The error fields are shown in Figure 10. In Test 2, it is observed that the trajectories predicted by the conventional ML ROM have a large difference from the reference, while the AL-ROM's prediction still matches the reference. If we check the error fields at the last time step, the AL-ROM's advantage is even more remarkable. Here a possible reason can be explained in Figure 11.

In Figure 11, we present the time evolution of the first POD coefficient of $\boldsymbol{y}_r$. We create an IVP with the maximum inputs $T_{\text{left}} = T_{\text{right}} = T_{\text{down}} = T_{\text{up}} = 1{,}000$. Similarly, another trajectory is produced with minimal input parameters $T_{\text{left}} = T_{\text{right}} = T_{\text{down}} = T_{\text{up}} = 20$. These trajectories are marked with green solid lines in Figure 11. These two trajectories are called "extreme trajectories." Since our full-order

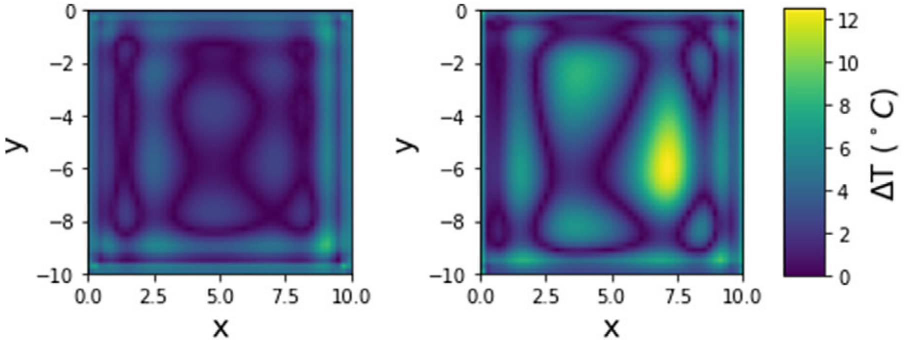
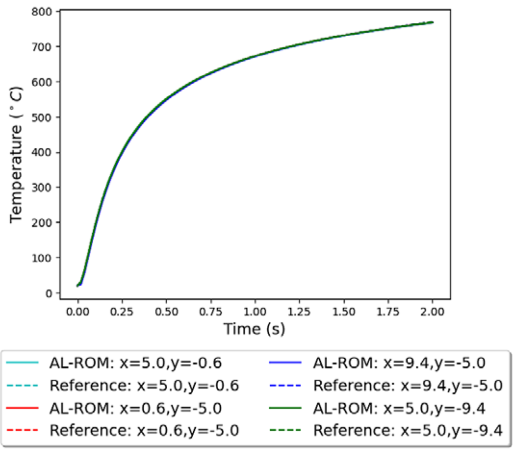
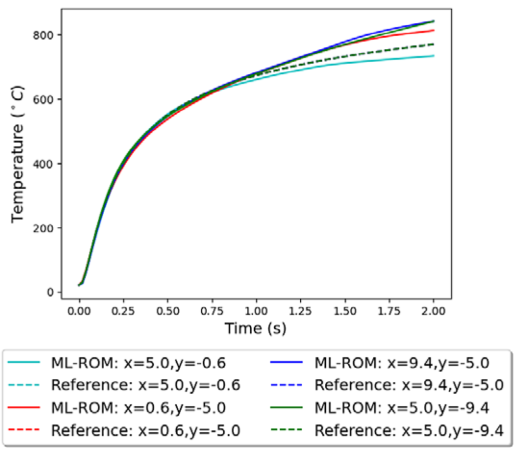


Figure 8. *Comparison between two ROMs' error (absolute) fields at t = 2s in Test 1. Left: AL. Right: ML. The ROMs are constructed with POD + ANN.*



(a) AL-ROM.



(b) ML-ROM.

Figure 9. *Linear thermal block: the FOM and ROM solution in Test 2. The ROMs are constructed with POD + ANN.*

problem is continuous, it is reasonable to assume that any trajectory produced by any parameter combination in $\mathcal{M}$ should be between these two green boundaries. Then we pick a parameter combination [510,510,510,510], which can be considered as the barycenter of $\mathcal{M}$, and the corresponding red-dashed trajectory is called "barycentric trajectory." Additionally, we plot the trajectories of the DPS-samples in light blue color. It is observed that these trajectories form a thin band whose center is approximately the barycentric trajectory. This means that assigning randomized parameter combinations to the IVP does not produce a wide distribution of the solution trajectory. A result of this narrow distribution is that the ROM will perform badly beyond the barycentric trajectory. This is reflected in Test 2. In the AL workflow, different $\boldsymbol{y}_r$ are randomly sampled in the estimated space $\mathcal{Y}^*$ and used as the initial states for the one-step snapshots. This distributes the training samples in a wider region in the reduced solution space, and the ROM trained by such a dataset should also be more reliable.

4.1.2. AL for OpInf

In this section, we perform the same tests as in Section 4.1.1, except that we use OpInf instead of ANN. The convergence of the AL algorithm is shown in Table 3. The algorithm uses 150 training samples for

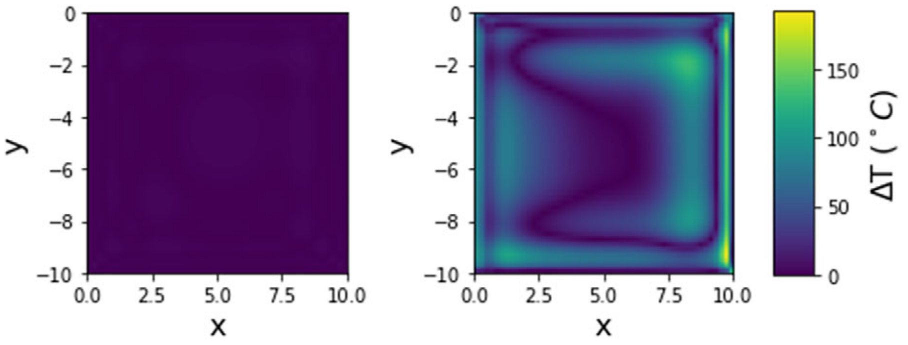


Figure 10. *Comparison between two ROMs' error (absolute) fields at $t = 2$s in Test 2. Left: AL. Right: ML. The ROMs are constructed with POD + ANN.*

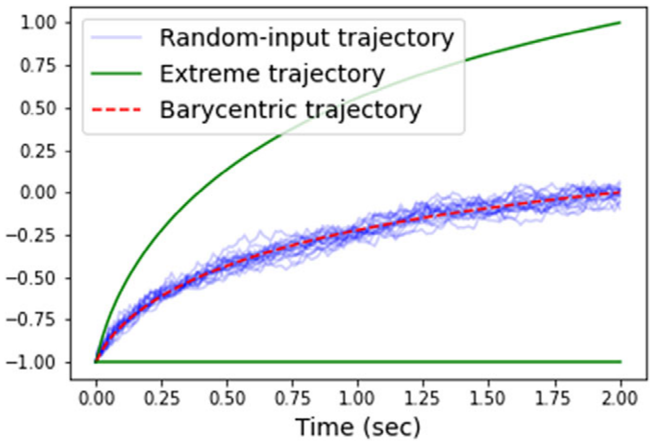


Figure 11. *The trajectories of the first POD coefficient formed by: random-input samples (blue), extreme-input samples (green), and mean-input samples (dashed red). Here the extreme inputs are* $[1{,}000, 1{,}000, 1{,}000, 1{,}000]$ *for the upper green curve and* $[20,20,20,20]$ *for the lower green curve. The mean input is* $[510,510,510,510]$.

Table 3. *PAC scores in the AL algorithm.*

Iteration	Number of samples	$\overline{\tau}^*$ (%)	$p\left(\overline{\tau}^*_{\text{design}}\right)$ (%)
1	25	24.41	0.03
2	50	17.87	0.10
3	75	9.54	4.14
4	100	5.04	37.26
5	125	1.83	94.26
6	150	1.83	96.20

Note. The ROM is constructed with POD + OpInf.

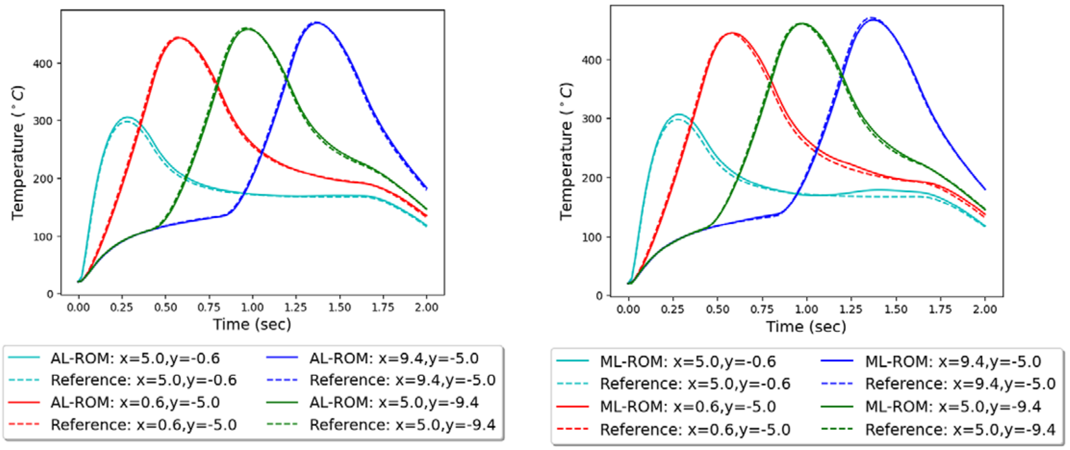
convergence, which is remarkably fewer than those needed for constructing the ANN-ROM. Besides, according to Table 4, the OpInf-ROM constructed using DPS-sampled data also has a confident ROM error that is very close to the OpInf-ROM built by the AL algorithm.

We show the same trajectory comparison for the OpInf-ROM. In Figures 12a,b and 14a,b, we can see that both ROMs have similar performance, even for Test 2 where the maximal input parameters are used in

Table 4. *97%-confident ROM error ($\bar{\tau}^*$) by another PAC validator.*

Number of samples	AL-ROM	ML-ROM
150	1.83%	2.37%

Note. The ROMs are constructed with POD + OpInf.



(a) AL-ROM. (b) ML-ROM.

Figure 12. *Linear thermal block: the FOM and ROM solution in Test 1. The ROMs are constructed with POD + OpInf.*

the online prediction. In the error fields presented in Figures 13 and 15, it is observed that the overall accuracy of the OpInf-ROM constructed by the AL algorithm is still better. But the performance of the OpInf-ROM constructed by the conventional approach is remarkably improved compared to the ANN-ROM, especially while predicting with the maximal parameters.

Through this experiment, we draw two conclusions. The first conclusion is that no matter with what kind of training data, building an OpInf-ROM with the same confident ROM error requires less training data compared to building an ANN-ROM. The second conclusion is that the OpInf-ROMs constructed in both ways (the proposed AL algorithm and the conventional approach) have similar qualities. This can be confirmed by both case-independent and case-dependent validation.

These conclusions match OpInf's feature of being physics-informed. While designing the architecture of the OpInf model, we have a hypothetical form for the governing differential equation of the FOM. By employing the hypothetical form, we reduce the flexibility of the ROM, which enables us to use fewer data to fit the ROM. Besides, the physics information restricts the extrapolated prediction, which is why even using the DPS-sampled data, the OpInf-ROM can still have good accuracy in Test 2.

4.2. Nonlinear RC ladder

The second test model is a nonlinear RC circuit model. This benchmark model is provided by The MORwiki Community (2018). It's different variants are widely used to test MOR methods, such as in Chen (1999), Rewienski (2003), Gu (2011), and Kawano and Scherpen (2019). To obtain the MATLAB code, readers should refer to Model 1 in The MORwiki Community (2018).

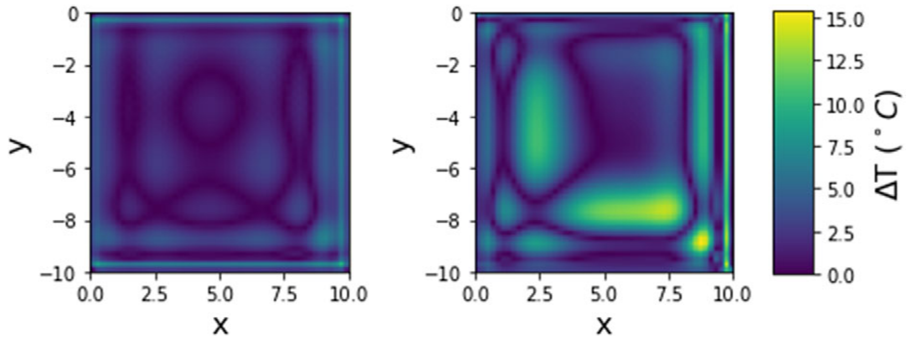


Figure 13. *Comparison between two ROMs' error (absolute) fields at $t = 2$s in Test 1. Left: AL. Right: ML. The ROMs are constructed with POD + OpInf.*

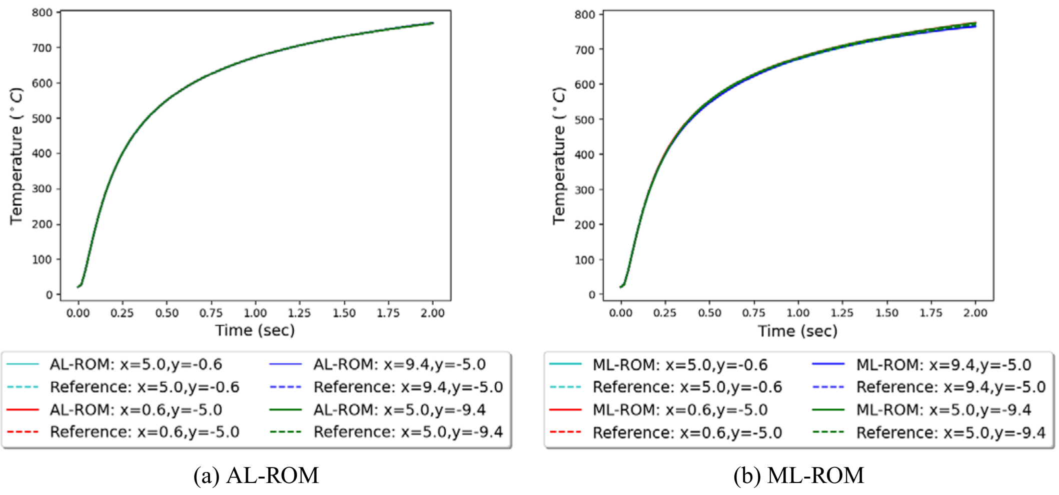


Figure 14. *The FOM and ROM solution in Test 2. The ROMs are constructed with POD + OpInf.*

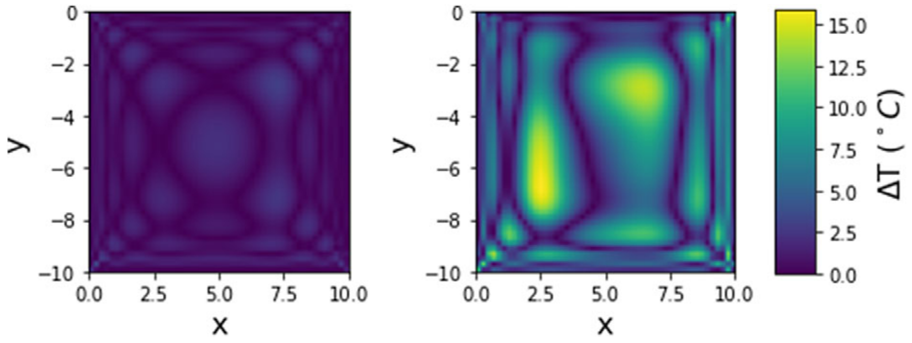


Figure 15. *Comparison between two ROMs' error (absolute) fields at $t = 2$s in Test 2. Left: AL. Right: ML. The ROMs are constructed with POD + OpInf.*

The governing equation is the same as equation (32) in Regazzoni et al. (2019):

$$\begin{cases} \dot{v}_1(t) = -2v_1(t) + v_2(t) + 2 - e^{40v_1(t)} - e^{40(v_1(t)-v_2(t))} + u(t) \\ \dot{v}_i(t) = -2v_i(t) + v_{i-1}(t) + v_{i+1}(t) + 2 + e^{40(v_{i-1}(t)-v_i(t))} - e^{40(v_i(t)-v_{i+1}(t))}, \text{ for } i = 2,3,\dots,N-1 \\ \dot{v}_N(t) = -v_N(t) + v_{N-1}(t) - 1 + e^{40(v_{N-1}(t)-v_N(t))}. \end{cases} \quad (40)$$

We assume there are $N = 128$ resistors in the circuit. Therefore, the size of the FOM is 128. The range of the input signal $u(t)$ is $\mathcal{M} = [0,1]$. To prepare the training data, the system is integrate in the time span $t \in (0,1]$ with forward Euler method, where the time step $\delta t = 2E-3$.

We create 5 IVPs with the SPS method to generate $\boldsymbol{Y}_{\text{estimate}}$ and use $\boldsymbol{Y}_{\text{estimate}}$ to create the initial POD space with $N_r = 15$. While using the JSS method to prepare the data pool I_{all}, we use $\beta = 0\%$. There are 20,000 joint samples in I_{all}.

We define $\epsilon = 3\%$ and $\sigma = 1\%$ and 2,944 additional samples are collected for the PAC validator. The investigated confidence level is $\eta_{\text{design}} = 97\%$.

4.2.1. AL for ANN

For both ANN-based ROMs, we use an MLP with [16, 64, 64, 15] neurons. The AL-ROM's convergence is presented in Table 5. The AL algorithm converges while 700 training samples are used. According to the PAC scores in Table 6, the 97%-confident ROM error of the AL-ROM is 1.00%. The 97%-confident ROM error of the ML-ROM is 3.07%, which is lower than the AL-ROM. This result tells us that the AL-ROM is assessed to be generally more accurate.

Then we test the ROMs with a specific test case. In the first test case, the input signal of the system is $\mu(t) = 0.5(\cos(2\pi t) + 1)$, which is the same as the test input used in Regazzoni et al. (2019). We monitor the variable $v_0(t)$ of the system and present the comparison between the FOM and ROM solution in Figure 16.

4.2.2. AL for OpInf

The AL-ROM's convergence is presented in Table 7. The AL algorithm converges while 50 training samples are used, which is much fewer than required by the ROM constructed with POD + ANN. In Table 8, the 97%-confident ROM error of the AL-ROM and ML-ROM is 0.50 and 2.07%, respectively. Based on the results, we expect the AL-ROM to perform generally better than the ML-ROM.

Table 5. *PAC scores in the AL algorithm.*

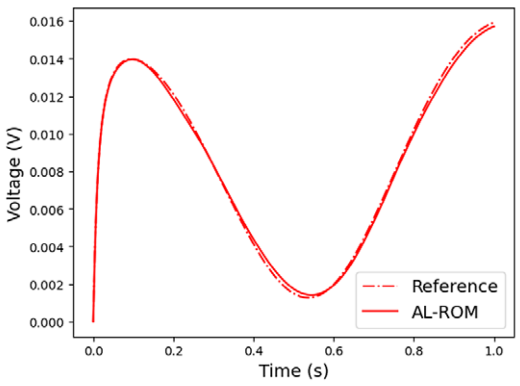
Iteration	Number of samples	$\overline{\tau}^*$ (%)	$p\left(\overline{\tau}^*_{\text{design}}\right)$ (%)
1	100	6.07	93.48
2	200	2.83	97.93
3	300	2.80	98.37
4	400	1.43	99.93
5	500	1.13	99.97
6	600	1.10	99.97
7	700	1.00	100.00

Note. The ROM is constructed with POD + ANN. $\overline{\tau}^*_{\text{design}} = 1\%$.

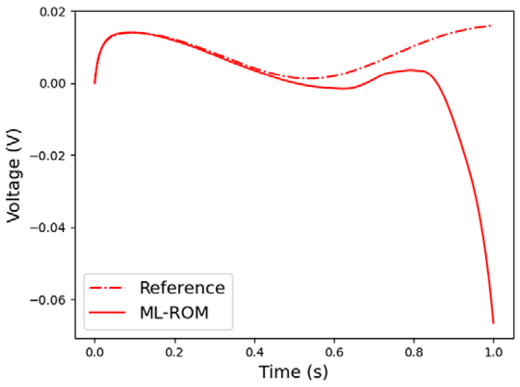
Table 6. *97%-confident ROM error ($\overline{\tau}^*$) by another PAC validator.*

Number of samples	AL-ROM	ML-ROM
700	1.00%	3.07%

Note. The ROMs are constructed with POD + ANN.



(a) AL-ROM.



(b) ML-ROM.

Figure 16. *Nonlinear RC ladder: the FOM and ROM solution in the test case, where* $\mu(t) = 0.5(\cos(2\pi t) + 1)$. *The ROMs are constructed with POD + ANN.*

Table 7. *PAC scores in the AL algorithm.*

Iteration	Number of samples	$\overline{\tau}^*$ (%)	$p\left(\overline{\tau}^*_{\text{design}}\right)$ (%)
1	10	5.40	48.17
2	20	3.33	73.61
3	30	0.87	99.46
4	40	0.60	99.97
5	50	0.50	100.00

Note. The ROM is constructed with POD + OpInf. $\overline{\tau}^*_{\text{design}} = 0.5\%$.

Table 8. *97%-confident ROM error (*$\overline{\tau}^*$*) by another PAC validator.*

Number of samples	AL-ROM	ML-ROM
50	0.50%	2.07%

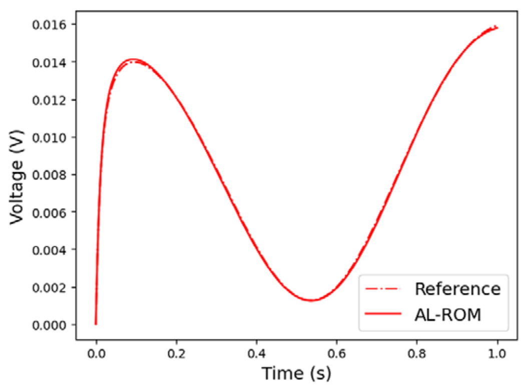
Note. The ROMs are constructed with POD + OpInf.

The same test case is used to further validate the ROMs. The corresponding results are presented in Figure 17. A similar comparison between the AL-ROM and ML-ROM is observed. In the test case, the OpInf-ROM constructed by the AL algorithm has almost the same transient response as the FOM.

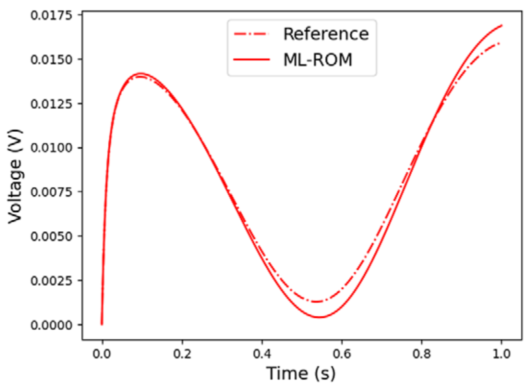
Through this numerical model, we see similar results to the linear model. To be specific, both types of the ROMs are benefited from the AL algorithm. Between them, the ANN-MOR method obtains more significant improvement as a purely data-driven method. The performance of the OpInf-ROM is also improved, but the difference is not as remarkable as the ANN-ROM.

4.3. 3-D vacuum furnace model

In Figure 18a, we present the 3-D simulation model of a vacuum radiation furnace (Hao et al., 2008). The model was created and simulated using NX Simcenter 3D. The original FEM model has 20,209 elements. The dark gray part represents the outer case of the furnace. The red parts are heaters, for which there are six in total. The blue component between the heaters and the case is the protecting shell. The cubic area at the center is the workzone where material can be placed. Since the internal space of the furnace is a vacuum,

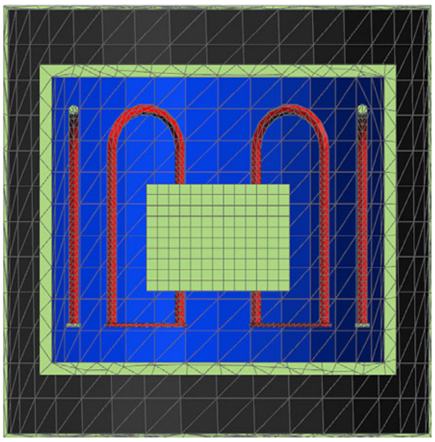


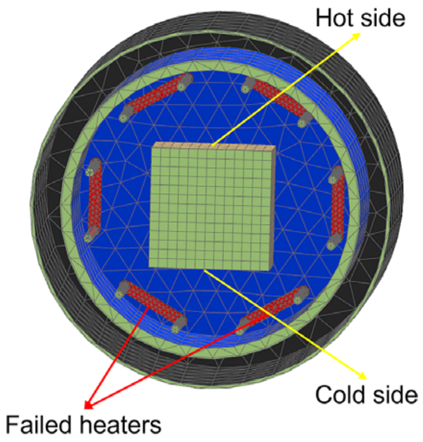
(a) AL-ROM.



(b) ML-ROM.

Figure 17. *Nonlinear RC ladder: the FOM and ROM solution in test case. The ROMs are constructed with POD + OpInf.*


(a) The cross-section of the 3-D simulation model of the vacuum radiation furnace.



(b) The failure case of the vacuum furnace model.

Figure 18. *The vacuum furnace model.*

during the industrial process, the material placed in the furnace will only be heated up by the thermal radiation from the heaters. The discrete governing equation of the system can be written as:

$$\boldsymbol{E}\frac{\partial \boldsymbol{T}}{\partial t} = \boldsymbol{K}\boldsymbol{T} + \boldsymbol{A}_{\mathrm{cv}}(\alpha)\boldsymbol{T} + \boldsymbol{R}\boldsymbol{T}^4 + \boldsymbol{B}\boldsymbol{u}, \tag{41}$$

where $\boldsymbol{E}$ is the thermal capacity matrix, $\boldsymbol{K}$ is the thermal conductivity matrix, $\boldsymbol{A}_{\mathrm{cv}}(\alpha)$ is the heat convection matrix and α is the convection coefficient, $\boldsymbol{R}$ is the radiation matrix, and $\boldsymbol{B}$ is the parameter-distribution matrix. The state (temperature) vector of this system is $\boldsymbol{T} \in \mathbb{R}^N$ where $N = 20{,}209$. We consider there are seven controllable parameters in the system: six heating power of the heaters ($\mathrm{kW/m^3}$) and the convection coefficient ($\mathrm{Wm^{-2}K^{-1}}$) of the outer surface of the case, which are stored in the parameter vector $\boldsymbol{u}$. Therefore, we have the vector $\boldsymbol{u} \in \mathbb{R}^{N_u}$ with $N_u = 7$. The parameter space is predefined to be:

$$\begin{aligned}\mathcal{M} = \left[0,10^4\right] \times \left[0,10^4\right] \times \left[0,10^4\right] \times \left[0,10^4\right] \\ \times \left[0,10^4\right] \times \left[0,10^4\right] \times [5,100].\end{aligned} \tag{42}$$

Based on physical consideration, the time grid is from 0 to 45,000 s with 450 time steps.

For the AL algorithm, $m = 20$ samples are taken from $\mathcal{M}$ to estimate the boundaries of $\mathcal{Y}$. We also include two special parameter configurations:

$$\begin{aligned} \boldsymbol{\mu}_{T_{\min}} &= [0,0,0,0,0,0,100], \\ \boldsymbol{\mu}_{T_{\min}} &= \left[10^4, 10^4, 10^4, 10^4, 10^4, 10^4, 5\right]. \end{aligned} \tag{43}$$

The initial reduced space is constructed by $N_r = 22$ POD basis. Then we create the joint space $\mathcal{J}^* = \mathcal{Y}^* \times \mathcal{M}$, where $\beta = 10\%$ is used for loosening. We use random sampling to get 40,000 joint samples for $\boldsymbol{I}_{\text{all}}$. We define $\epsilon = 3\%$ and $\sigma = 1\%$ and 2,944 additional samples are collected for the PAC validator. The investigated confidence level is $\eta_{\text{design}} = 97\%$. We pick $\overline{\tau}^*_{\text{design}} = 1\%$ and $\Delta\overline{\tau}^*_{\text{tol}} = 0.01\%$. In conclusion, we are looking for a ROM whose 97%-confident ROM error is 1%.

4.3.1. AL for ANN

The AL-ROM's convergence is presented in Table 9. We construct ROMs with our method (AL) and the conventional method (ML) with the same number of training samples (20,000). The ROM errors are 1.00% (AL) and 13.07% (ML) at the investigated confidence level (97%). Both ROMs have an MLP with [29, 80, 80, 22] neurons (Table 10).

Two test scenarios are employed to show the ROMs' performance in relatively realistic conditions. In the first test, all heaters are working with a 3-stage heating profile (Figure 19) and the convection coefficient is set to be $50\,\text{W}\,\text{m}^{-2}\text{K}^{-1}$).

In this test case, we monitor the temperature trajectories on a heater and in the workzone.

To show the error for the whole temperature field, we compute the absolute errors for all 20,209 elemental temperature individually. Four statistical measures of the absolute errors, that is, mean, standard deviation (std.), minimum and maximum, are used to reflect the error distribution at the final time point. The results are shown in Table 11.

In another test case, we use the same heating profile for four heaters and turn off two heaters (Figure 18b). This simulates a scenario where two heaters have failed during the industrial process. The existence of failed heaters will cause a nonuniform temperature field in the workzone. We call them cold side and hot side, respectively. In this test, we monitor the temperature trajectories of the cold side and the hot side using the ROMs. The corresponding comparison between ROM-predicted trajectories and FOM-predicted trajectories in two test cases is given in Figure 21 and Table 12.

Table 9. *PAC scores in the AL algorithm.*

Iteration	Number of samples	$\overline{\tau}^*$ (%)	$p\left(\overline{\tau}^*_{\text{design}}\right)$ (%)
1	4,000	2.90	78.91
2	8,000	1.60	99.52
3	12,000	1.30	99.97
4	16,000	1.10	99.97
5	20,000	1.00	100.00

Note. The ROM is constructed with POD + ANN.$\overline{\tau}^*_{\text{design}} = 1\%$.

Table 10. *97%-confident ROM error ($\overline{\tau}^*$) by another PAC validator.*

Number of samples	AL-ROM	ML-ROM
20,000	1.00%	13.07%

Note. The ROMs are constructed with POD + ANN.

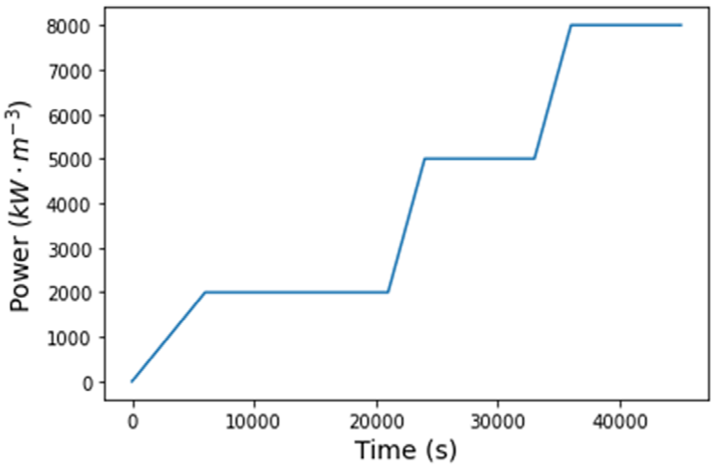


Figure 19. *The heat generation for the first test case. The strategy is applied to all six heaters.*

Table 11. *Statistical analysis for the error field ($\Delta T(t), t \in (0, 45{,}000]$) for the 3-stage heating profile.*

	AL-ROM	ML-ROM
Mean (°C)	2.29	5.72
Std. ((°C)^2)	4.30	9.29
Min. (°C)	2.14E-8	1.69E-6
Max. (°C)	96.65	104.01

Note. The ROMs are constructed with POD + ANN.

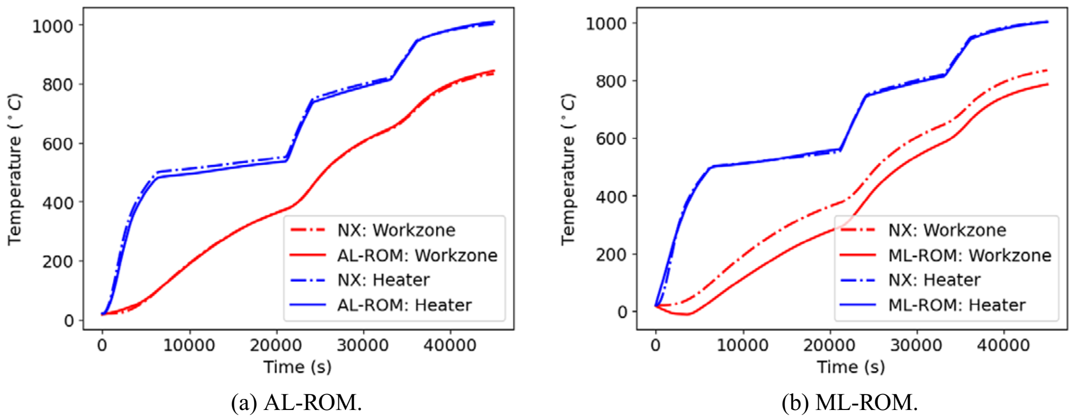
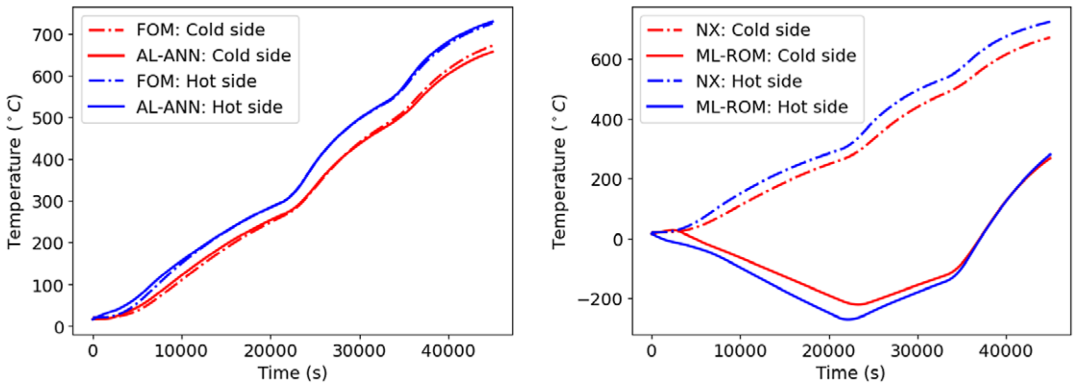


Figure 20. *Comparison between the NX solution and the ROMs' prediction for the 3-stage heating profile. The ROMs are constructed with POD + ANN.*

As observed in both trajectory comparison in Figures 20 and 21, the ROM constructed by the AL algorithm has a better agreement with the reference result. Moreover, in Tables 11 and 12, all statistical measurement of the AL-ROM's error fields are better than the ML-ROM's. These facts tell us the proposed method significantly improves the performance of the ANN-ROM in this nonlinear thermal system.



(a) AL-ROM. (b) ML-ROM.

Figure 21. *Comparison between NX solution and ROMs' prediction for the heating profile with failure. The ROMs are constructed with POD + ANN.*

Table 12. *Statistical analysis for the error field ($\Delta T(t), t \in (0, 45{,}000]$) for the heating profile with failure.*

	AL-ROM	ML-ROM
Mean (°C)	8.66	31.76
Std. ((°C)^2)	16.40	63.99
Min. (°C)	1.31E-6	8.84E-7
Max. (°C)	348.93	680.32

Note. The ROMs are constructed with POD + ANN.

4.3.2. AL for OpInf

In this section, the results of constructing the OpInf-ROM with the AL algorithm and the conventional approach are given. The confident ROM error of the OpInf-ROM successfully converges to 1.00% after five iterations. The OpInf-ROM produced by the conventional approach has a confident ROM error of 7.54% using the same amount of training data. This confident ROM error is lower than the confident ROM error of the ANN-ROM produced in the conventional way (Figures 22 and 23 and Tables 13–16).

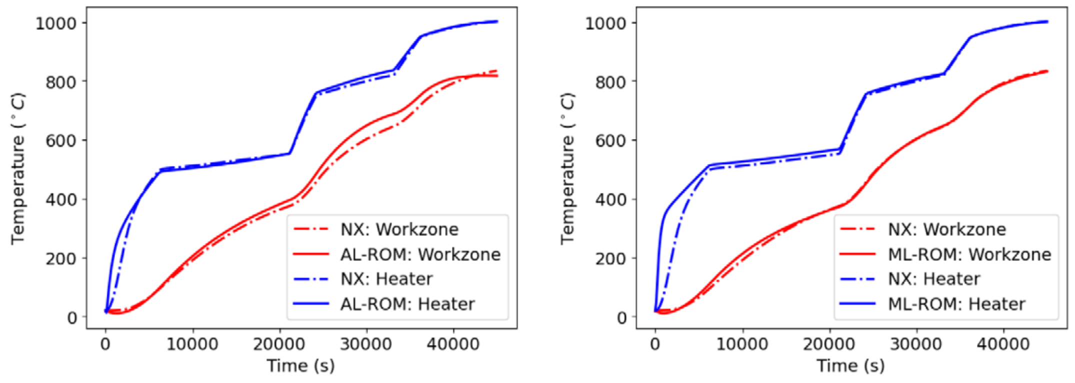
Similar to the results in Section 4.1.2, in the case-dependent validation, we can only see limited improvement by employing the AL algorithm for the OpInf method. Besides the reasons mentioned in Section 4.1.2, we also do not expect to have an extremely high temperature in the use cases of the case-dependent validation for the vacuum furnace model. This means that the temperatures of the system under the validation inputs are still considered to be located in the sampled region of the DPS method. In this case, a ROM trained with the DPS-sampled data can still predict with high accuracy.

5. Summary and Conclusions

In this work, we propose an Active-Learning-based approach to support machine-learning-based (ML-based) MOR methods. It can be applied to the reduction of any problem of the form,

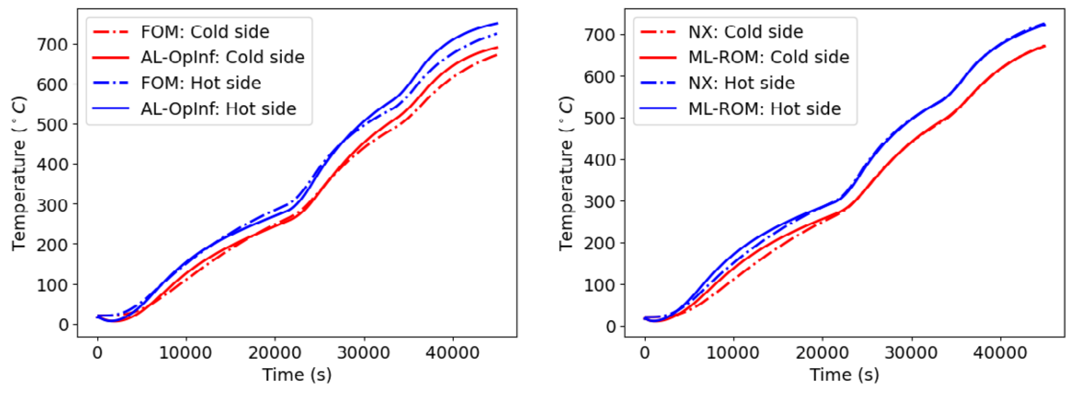
$$\dot{\boldsymbol{y}}(t) = \boldsymbol{f}(\boldsymbol{y}; \boldsymbol{\mu}), \tag{44}$$

where $\boldsymbol{y} \in \mathbb{R}^N$ is the state vector and the parameter vector $\boldsymbol{\mu} \in \mathbb{R}^{N_m}$ is time-dependent. It can also be applied to a wide variety of ML-based MOR methods including purely data-driven methods such as Deep Neural Network in Fresca and Manzoni (2022), Gaussian process regression in Guo and Hesthaven (2018)), Dynamic Mode Decomposition (Erichson et al., 2019), Residual Network in San et al. (2019), as well as



(a) AL-ROM. (b) ML-ROM.

Figure 22. *Comparison between NX solution and ROMs' prediction for the 3-stage heating profile. The ROMs are constructed with POD + OpInf.*



(a) AL-ROM. (b) ML-ROM.

Figure 23. *Comparison between NX solution and ROMs' prediction for the heating profile with failure. The ROMs are constructed with POD + OpInf.*

Table 13. *PAC scores in the AL algorithm.*

Iteration	Number of samples	$1-\overline{\tau}^*$ (%)	$p\left(\overline{\tau}^*_{\text{design}}\right)$ (%)
1	1,000	2.30	76.49
2	2,000	1.40	98.68
3	3,000	1.20	99.83
4	4,000	1.10	99.93
5	5,000	1.00	100.00

Note. The ROM is constructed with POD + OpInf. $\overline{\tau}^*_{\text{design}} = 1\%$.

physics-informed methods such as OpInf (Peherstorfer and Willcox, 2016), physics-informed neural networks (PINNs) (Kim et al., 2022) or SINDy (Champion et al., 2019). The approach cannot however be used with recurrent ML models such as those in Kani and Elsheikh (2017), Mohan and Gaitonde (2018), Maulik et al. (2020), Wang et al. (2020), and Wu and Noels (2022).

Table 14. *97%-confident ROM error ($\bar{\tau}^*$) by another PAC validator.*

Number of samples	AL-ROM	ML-ROM
5,000	1.00%	7.54%

Note. The ROMs are constructed with POD + OpInf.

Table 15. *Statistical analysis for the error field ($\Delta T(t), t \in (0, 45{,}000]$) for the 3-stage heating profile.*

	AL-ROM	ML-ROM
Mean (°C)	3.69	3.79
Std. ((°C)^2)	8.74	13.39
Min. (°C)	1.21E-6	5.15E-8
Max. (°C)	166.32	334.06

Note. The ROMs are constructed with POD + OpInf.

Table 16. *Statistical analysis for the error field ($\Delta T(t), t \in (0, 45{,}000]$) for the heating profile with failure.*

	AL-ROM	ML-ROM
Mean (°C)	5.22	5.93
Std. ((°C)^2)	12.43	21.89
Min. (°C)	1.17E-6	7.32E-7
Max. (°C)	208.38	346.93

Note. The ROMs are constructed with POD + OpInf.

One key ingredient is the generation of solution snapshots through the (active) sampling of a joint space consisting of a predefined parameter space and an estimated reduced solution space. This leads to a better coverage of these two spaces, which is otherwise difficult to achieve with simpler strategies. In addition, we present a case-independent validator constructed based on the PAC learning theory. Through numerical experiments, we find that our approach significantly improves the performance of purely data-driven MOR methods and that the PAC validator presents a good indicator of the ROM quality. However, the benefits of using these strategies are less clear when applied to the physics-informed method OpInf. We expect this to be also the case for other methods in this category, such as PINNs (Kim et al., 2022) or SINDy (Champion et al., 2019). By exploiting knowledge of the physical system, such methods are better at constructing ROMs that can generalize beyond the training data. Therefore the benefit of our improved sampling strategy is less marked.

Acknowledgments. We are grateful to Mr. Hicham Elmrini from Maya HTT for his guidance with the simulation software.

Author Contributions. Conceptualization: Q.Z., D.H., H.J.B., J.M.L.; Data curation: Q.Z.; Formal analysis: Q.Z., D.H., H.J.B., J.M.L.; Investigation: Q.Z.; Methodology: Q.Z., D.H., H.J.B., J.M.L.; Software: Q.Z.; Supervision: D.H., H.J.B., J.M.L.; Validation: Q.Z.; Visualization: Q.Z; Writing—original draft: Q.Z.; Writing—review and editing: D.H., H.J.B., J.M.L.

Competing Interests. The authors declare no competing interests exist.

Data Availability Statement. The data and the source code necessary for reproducing the results in Sections 4.1 and 4.2 will be made available under https://github.com/qinyu-zhuang/AL-MOR-paper 6 months after publication, to comply with internal policies for the publication of this information.

Ethics Statement. The research meets all ethical guidelines, including adherence to the legal requirements of the study country.

Funding Statement. This work received no specific grant from any funding agency, commercial or not-for-profit sectors.

References

Alippi C (2016) *Intelligence for Embedded Systems: A Methodological Approach*, 1st Edn. New York: Springer Publishing Company, Incorporated.

Antoulas A (2005) *Approximation of Large-Scale Dynamical Systems*. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Batista GEAPA, **Prati RC and Monard MC** (2004) A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explorations Newsletter 6*(1), 20–29.

Broomhead DS and Lowe D (1988) *Radial Basis Functions, Multi-Variable Functional Interpolation and Adaptive Networks.* Technical Report, Royal Signals and Radar Establishment, Malvern, UK.

Bui-Thanh T, **Willcox K and Ghattas O** (2008) Model reduction for large-scale systems with high-dimensional parametric input space. *SIAM Journal on Scientific Computing 30*(6), 3270–3288.

Champion K, **Lusch B**, **Kutz JN and Brunton SL** (2019) Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences 116*(45), 22445–22451.

Chaturantabut S and Sorensen DC (2010) Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal on Scientific Computing 32*(5), 2737–2764.

Chen Y (1999) *Model Order Reduction for Nonlinear Systems.* Master's thesis, Massachusetts Institute of Technology, Cambrigde, MA.

Erichson NB, **Mathelin L**, **Kutz JN and Brunton SL** (2019) Randomized dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems 18*(4), 1867–1891.

Fresca S and Manzoni A (2022) Pod-dl-rom: Enhancing deep learning-based reduced order models for nonlinear parametrized PDEs by proper orthogonal decomposition. *Computer Methods in Applied Mechanics and Engineering 388*, 114181.

Glorot X, **Bordes A and Bengio Y** (2011) Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics. JMLR Workshop and Conference Proceedings*, pp. 315–323. Fort Lauderdale, FL: PMLR.

Gu C (2011) *Model Order Reduction of Nonlinear Dynamical Systems*. Berkeley, CA: University of California.

Guo M and Hesthaven J (2018) Reduced order modeling for nonlinear structural analysis using Gaussian process regression. *Computer Methods in Applied Mechanics and Engineering 341*, 807–826.

Gyori NG, **Palombo M**, **Clark CA**, **Zhang H and Alexander DC** (2022) Training data distribution significantly impacts the estimation of tissue microstructure with machine learning. *Magnetic Resonance in Medicine 87*(2), 932–947.

Haasdonk B (2017) Reduced basis methods for parametrized pdes–a tutorial introduction for stationary and instationary problems. *Model Reduction and Approximation: Theory and Algorithms 15*, 65.

Haasdonk B, **Dihlmann M and Ohlberger M** (2011) A training set and multiple bases generation approach for parameterized model reduction based on adaptive grids in parameter space. *Mathematical and Computer Modelling of Dynamical Systems 17* (4), 423–442.

Habermann C and Kindermann F (2007) Multidimensional spline interpolation: Theory and applications. *Computational Economics 30*(2), 153–169.

Hammersley J and Handscomb D (1964) *Monte-Carlo Methods*. London: Mathuen.

Hao X, **Gu J**, **Chen N**, **Zhang W and Zuo X** (2008) 3-d numerical analysis on heating process of loads within vacuum heat treatment furnace. *Applied Thermal Engineering 28*(14–15), 1925–1931.

Hartmann D, **Herz M and Wever U** (2018) Model order reduction a key technology for digital twins. In *Reduced-Order Modeling (ROM) for Simulation and Optimization*. Cham: Springer, pp. 167–179.

Haussler D (1990) *Probably Approximately Correct Learning*. Santa Cruz, CA: University of California, Santa Cruz, Computer Research Laboratory.

He K, **Zhang X**, **Ren S and Sun J** (2016) Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Las Vegas, NV: IEEE, pp. 770–778.

Heres PJ (2005) *Robust and Efficient Krylov Subspace Methods for Model Order Reduction.* Phd Thesis in TU/e.

Kani JN and Elsheikh AH (2017) DR-RNN: A deep residual recurrent neural network for model reduction. arXiv preprint arXiv: 1709.00939.

Kawano Y and Scherpen J (2019) Empirical differential gramians for nonlinear model reduction. arXiv preprint arXiv: 1902.09836.

Kim Y, **Choi Y**, **Widemann D and Zohdi T** (2022) A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder. *Journal of Computational Physics 451*, 110841.

Krige DG (1951) A statistical approach to some basic mine valuation problems on the witwatersrand. *Journal of the Southern African Institute of Mining and Metallurgy 52*(6), 119–139.

Kuether RJ and Allen MS (2016) Validation of nonlinear reduced order models with time integration targeted at nonlinear normal modes. *Conference Proceedings of the Society for Experimental Mechanics Series 1*, 363–375.

Lappano E, **Naets F**, **Desmet W**, **Mundo D and Nijman E** (2016) A greedy sampling approach for the projection basis construction in parametric model order reduction for structural dynamics models. In *Proceedings of ISMA*. Leuven, Belgium: KU Leuven, pp. 19–21.
Lu K, **Jin Y**, **Chen Y**, **Yang Y**, **Hou L**, **Zhang Z**, **Li Z and Fu C** (2019) Review for order reduction based on proper orthogonal decomposition and outlooks of applications in mechanical systems. *Mechanical Systems and Signal Processing 123*, 264–297.
Maulik R, **Mohan A**, **Lusch B**, **Madireddy S**, **Balaprakash P and Livescu D** (2020) Time-series learning of latent-space dynamics for reduced-order model closure. *Physica D: Nonlinear Phenomena 405*, 132368.
McQuarrie SA, **Huang C and Willcox KE** (2021) Data-driven reduced-order models via regularised operator inference for a single-injector combustion process. *Journal of the Royal Society of New Zealand 51*(2), 194–211.
Mohan AT and Gaitonde DV (2018) A deep learning based approach to reduced order modeling for turbulent flow control using lstm neural networks. arXiv preprint arXiv:1804.09269.
Pan S and Duraisamy K (2018) Long-time predictive modeling of nonlinear dynamical systems using neural networks. *Complexity 2018*, 4801012.
Paszke A, **Gross S**, **Chintala S**, **Chanan G**, **Yang E**, **DeVito Z**, **Lin Z**, **Desmaison A**, **Antiga L and Lerer A** (2017) Automatic differentiation in pytorch. In *NIPS 2017 Workshop on Autodiff*. Long Beach, CA: NIPS.
Paul-Dubois-Taine A and Amsallem D (2015) An adaptive and efficient greedy procedure for the optimal training of parametric reduced-order models. *International Journal for Numerical Methods in Engineering 102*(5), 1262–1292.
Peherstorfer B and Willcox K (2016) Data-driven operator inference for nonintrusive projection-based model reduction. *Computer Methods in Applied Mechanics and Engineering 306*, 196–215.
Perez R, **Wang XQ and Mignolet MP** (2014) Nonintrusive structural dynamic reduced order modeling for large deformations: Enhancements for complex structures. *Journal of Computational and Nonlinear Dynamics 9*(3), 031008.
Prechelt L (1998) Early stopping-but when? In *Neural Networks: Tricks of the Trade*. Berlin: Springer, pp. 55–69.
Przekop A, **Guo X and Rizzi SA** (2012) Alternative modal basis selection procedures for reduced-order nonlinear random response simulation. *Journal of Sound and Vibration 331*(17), 4005–4024.
Qian E, **Kramer B**, **Peherstorfer B and Willcox K** (2020) Lift & learn: Physics-informed machine learning for large-scale nonlinear dynamical systems. *Physica D: Nonlinear Phenomena 406*, 132401.
Rasheed A, **San O and Kvamsdal T** (2019) Digital twin: Values, challenges and enablers. arXiv preprint arXiv:1910.01719.
Regazzoni F, **Dedè L and Quarteroni A** (2019) Machine learning for fast and reliable solution of time-dependent differential equations. *Journal of Computational Physics 397*, 108852.
Rewienski MJ (2003) *A Trajectory Piecewise-Linear Approach to Model Order Reduction of Nonlinear Dynamical Systems.* PhD Thesis, Massachusetts Institute of Technology.
San O, **Maulik R and Ahmed M** (2019) An artificial neural network framework for reduced order modeling of transient flows. *Communications in Nonlinear Science and Numerical Simulation 77*, 271–287.
Settles B (2009) Active learning literature survey.
The MORwiki Community (2018) Nonlinear RC ladder. MORwiki – Model Order Reduction Wiki.
Volkwein S (2013) Proper orthogonal decomposition: Theory and reduced-order modelling. *Lecture Notes, University of Konstanz 4*(4), 1–29.
Wang Q, **Ripamonti N and Hesthaven JS** (2020) Recurrent neural network closure of parametric POD-galerkin reduced-order models based on the Mori-Zwanzig formalism. *Journal of Computational Physics 410*, 109402.
Willcox K and Peraire J (2002) Balanced model reduction via the proper orthogonal decomposition. *AIAA Journal 40*(11), 2323–2330.
Williams CK and Rasmussen CE (1996) Gaussian processes for regression.
Wu L and Noels L (2022) Recurrent neural networks (RNNS) with dimensionality reduction and break down in computational mechanics; Application to multi-scale localization step. *Computer Methods in Applied Mechanics and Engineering 390*, 114476.
Xiao D (2019) Error estimation of the parametric non-intrusive reduced order model using machine learning. *Computer Methods in Applied Mechanics and Engineering 355*, 513–534.
Zhuang Q, **Lorenzi JM**, **Bungartz H-J and Hartmann D** (2021) Model order reduction based on runge-kutta neural networks. *Data-Centric Engineering 2*, e13.

Cite this article: Zhuang Q, Hartmann D, Bungartz H.-J and Lorenzi JM (2023). Active-learning-based nonintrusive model order reduction. *Data-Centric Engineering*, 4, e2. doi:10.1017/dce.2022.39