TUM School of Computation, Information and Technology
Technische Universität München

Non-intrusive Model Order Reduction based on Joint Space Sampling and Active Learning

Qinyu Zhuang

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:
Prof. Dr. Helmut Seidl

Prüfende der Dissertation:
1. Prof. Dr. Hans-Joachim Bungartz
2. Prof. Dr. Benjamin Peherstorfer

Die Dissertation wurde am 29.08.2022 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 22.11.2022 angenommen.

Acknowledgments

I would like to thank all people who have provided their selfless support during my PhD time. First of all, my sincere thanks go to my supervisor Prof. Hans-Joachim Bungartz, for his helpful advice. He gives me sufficient flexibility in choosing the research focus, and his guidance makes me more professional in the research. All of these make this thesis possible. I also thank Prof. Benjamin Peherstorfer, his comments in our regular exchange always inspire me and make the thesis more solid.

My gratitude also goes to my colleagues at Siemens, especially my advisors, Dr. Juan Manuel Lorenzi and Dr. Dirk Hartmann. I cannot remember how many conversations/discussions we have regarding my research in the past few years. Still, all of these finally become important contributions to the thesis. They always generously share their research experience and good practice with me to help me improve myself. I do not believe I can "academically grow up" so fast without them. I am thankful to Dr. Meinhard Paffrath, who gives me many valuable suggestions for the thesis and helps me translate the abstract. I also appreciate my group mates, including but not limited to Rishith Ellath Meethal, Christoph Ludwig and Diana Manvelyan. The research sometimes can be frustrating, but the time with them tells me I am not the one fighting alone. Besides, I am grateful to my RG Head, Dr. Christoph Heinrich. He builds a very comfortable environment for all PhD students in the group, which is undoubtedly beneficial for me. As an excellent company, Siemens provides me not only the funding for my PhD project but also a close vision of industry, which allows me to investigate what is really demanded in industrial use cases.

Moreover, I would like to thank the people who support me in my daily life. Although they never manage to understand what I am doing, my parents always encourage me, just like what they did in the past 28 years. My best friend Jiayun Xu, with whom I have a 16-year friendship, also helps me manage my stress and be peaceful after intensive work. Last but not least, I thank my beloved Xiao Yang. Despite the long distance and the time difference between Germany and China, she lights up my world and brings so many joys into my life.

Abstract

Model Order Reduction (MOR) aims to generate a compact and faster representation of complex Full Order Models (FOMs). This representation is known as a Reduced Order Model (ROM). Among different MOR approaches, non-intrusive MOR attracts much attention in industry for its easy implementation and solver-independent feature. Plenty of existing research focuses on improving the ROM's performance in the online phase. However, reducing costs in the offline phase is also considerably important for the application of MOR.

In this dissertation, the requirement for disk storage and memory space is reduced by tackling two critical problems in the ROM construction phase: how to improve the quality of training data and how to use as few training data as possible. A new FOM-snapshot-sampling method called Joint Space Sampling (JSS) is proposed to improve the training data quality. Compared to conventional approaches, the training data generated by JSS do not only have a good distribution in the parameter space but also have a good distribution in the reduced solution space. Given the high-quality snapshots, to minimize the number of the training data needed for constructing the ROM, the creation process is designed with a greedy framework. The ROM created in each iteration will be validated, and the validation results will be used to decide the acceptance of the ROM. Traditional case-dependent validation is replaced by novel case-independent validation to improve the generalizability of the ROM validation. The proposed case-independent validation is based on the theory of Probably Approximately Correct (PAC) Learning, and there is no need to assume potential use cases for the validation. The concept of Active Learning (AL) is employed to accelerate the convergence of the iteration. In sum, a novel MOR method based on JSS and AL is developed and named Active-Learning-based Model Order Reduction (AL-MOR). Many different Machine-Learning-based MOR methods can be integrated into the AL-MOR method.

The effectiveness of the AL-MOR method is shown with numerical experiments. Within different test cases of the numerical experiments, we verify the theories, compare different methods and demonstrate the great potential of the proposed AL-MOR method in real engineering use cases.

Kurzfassung

Modellordnungsreduktion (MOR) zielt darauf ab, eine kompaktere und schnellere Darstellung des komplexen „Full Order Models“ (FOM) zu erzeugen. Diese Darstellung wird als reduziertes Ordnungsmodell (ROM) bezeichnet. Unter den verschiedenen MOR Ansätzen hat das non-intrusive MOR besondere Bedeutung für die Industrie wegen seiner einfachen Implementierung und Unabhängigkeit vom Löser. Ein großer Teil der existierenden Forschung ist darauf ausgerichtet, das ROM in der Online-Phase zu verbessern. Aber für die Anwendung von MOR ist es auch sehr wichtig, die Kosten in der Offline-Phase zu reduzieren.

In dieser Dissertation wird der Aufwand für Platten- und Hauptspeicher reduziert durch die Behandlung zweier kritischer Probleme während der ROM - Konstruktionsphase: Verbesserung der Qualität bei gleichzeitiger Beschränkung des Umfangs der Trainingsdaten. Eine neue FOM-Snapshot Sampling-Methode, genannt „Joint Space Sampling“ (JSS), wird vorgeschlagen, um die Qualität der Trainingsdaten zu verbessern. Verglichen mit konventionellen Ansätzen haben die durch JSS generierten Trainingsdaten nicht nur eine gute Verteilung im originalen Parameterraum, sondern auch eine gute Verteilung im reduzierten Lösungsraum. Bei gegebenen Snapshots hoher Qualität wird für den Prozess der Datengenerierung ein Greedy Ansatz gewählt, um die Anzahl der Trainingsdaten zur Generierung des ROMs zu minimieren. Das in jeder Iteration erzeugte ROM wird validiert, und auf Basis der Resultate wird über die Akzeptanz des ROMs entschieden. Die traditionelle fallabhängige Validierung wird ersetzt durch eine fallunabhängige Validierung, um die Verallgemeinerbarkeit der ROM Validierung zu verbessern. Die vorgeschlagene fallunabhängige Validierung basiert auf der Theorie des „Probably Approximately Correct (PAC)“ Lernens, und dort werden mögliche Use Cases für die Validierung nicht benötigt. Das Konzept des Aktiven Lernens („Active Learning (AL)“) wird eingesetzt, um die Konvergenz der Greedy Iteration zu beschleunigen. Zusammengefasst wird eine neue MOR Methode basierend auf JSS und AL entwickelt und „Active-Learning-based Model Order Reduction“ (AL-MOR) genannt. Viele unterschiedliche MOR Methoden basierend auf maschinellem Lernen können in die AL-MOR Methode integriert werden.

Die Effektivität der AL-MOR Methode wird in numerischen Experimenten gezeigt. In verschiedenen Testfällen dieser numerischen Experimente verifizieren wir die Theorien, vergleichen die verschiedenen Methoden und demonstrieren das große Potential der vorgeschlagenen AL-MOR Methode in realen technischen Anwendungen.

Contents

List of Figures

List of Tables

Part I.

Preliminaries

1. Introduction

1.1. Motivation

With the fast development of computing hardware, more and more complex numerical models can be used for simulation in industry. Such simulation can be an alternative to physical prototyping and reduce the cost of entire project. However, most applications of numerical simulation are still concentrated in the design and engineering phases of the lifecycle. The limited application is due to the requirement for high-performance hardware for solving complex models. Moreover, the solution of such a model also needs a long time to be completed, which becomes a barrier to deploying numerical models into so-called edge devices in factories. Therefore, how to enable running a simpler model of good quality is a question worthy of consideration. By answering it, the numerical simulation can be brought into the operation phase and becomes the so-called real-time simulation [1]. This is a key part of Digital Twin (DT) [2, 3]. A DT running next to the operated machine can enable novel industrial solutions such as model-based predictive maintenance or model-based optimization.

The enabler to this change is known under the umbrella term Model Order Reduction (MOR) [4, 5, 6, 7, 8]. The lifetime of a Reduced Order Model (ROM) can be split into an offline phase and an online phase. During the offline phase, we prepare the ROM with different methods. During the online phase, the prepared ROM can run in real-time and only requires minimal computational resources from the device. Among the broad range of MOR techniques, the projection-based MOR is widely studied by previous research. Just as its name implies, this kind of MOR method focuses on seeking a low-dimensional subspace and projecting the Full Order Model (FOM) onto the subspace. The representation of the original model in the subspace is the ROM.

Different projection-based MOR techniques have different approaches to constructing the subspace. They can be roughly split into two classes: model-driven methods and data-driven methods. The former requires the knowledge of FOM, such as governing equations, involved system matrices, etc. MOR methods such as the Krylov Subspace Method [9, 10], the Balanced-truncation Method [11] can be classified into this group.

The construction of the reduced space in the data-driven methods is based on simulation data, such as input parameters and simulation results, obtained from the communication with simulators. As a representative, Proper Orthogonal Decomposition (POD) [12] is popular, and it has another well-known name Principal Component Analysis (PCA) [13] in Data Science.

A special sub-class of the data-driven MOR is non-intrusive MOR. Non-intrusive MOR methods remove the need for FOM equations or a modifiable numerical code [14]. This goal is usually achieved by data-driven techniques in system identification or Machine Learning (ML). An advantage of these MOR methods is no intrusion into numerical solvers, leading to their easier implementation in industrial applications. Examples of non-intrusive MOR methods include, such as Operator Inference (OpInf) [15] and Dynamic Mode Decomposition [16]. Besides, Artificial-Neural-Network-based (ANN-based) methods also have their places. Multilayer Perceptron (MLP) [17, 18], Recurrent Neural Network (RNN) [19, 20, 21, 22], Long-Short-Term Memory (LSTM) [23], Deep Learning [24], and Extreme Learning Machine [25] have already been investigated for this purpose.

However, one of the bottleneck problems for non-intrusive MOR is data resource. Compared to intrusive data-driven MOR, non-intrusive MOR relies more on the provided data. For intrusive methods, one can use greedy sampling methods [26, 27], which adaptively choose samples by finding the location at which the estimate of the ROM error is maximized [28]. However, these intrusive methods still require the access to the FOM solver. By contrast, in [29], the authors proposed an non-intrusive way for the Radial Basis Function (RBF)-based MOR method. In [30], the authors designed an active selection strategy to improve the quality of the snapshot matrix for inferring reduced operators. These non-intrusive methods have proven their effectiveness for specific problems/MOR methods.

Besides, in the world of ML, the concept of Active Learning (AL) [31] was proposed to facilitate the construction of ML models, where ML models are allowed to select training data actively based on the current knowledge learned by models. The family of AL has many members, which are generalized strategy for actively constructing ML models. Nevertheless, the application of corresponding methods is quite limited for MOR. In this thesis, inspired by the concept of AL, a novel method named Active-Learning-based Model Order Reduction (AL-MOR) is proposed for constructing ROMs.

1.2. Gap analysis

In reality, most engineering problems are often governed by nonlinear physics. The nonlinearity can come from physics, material properties or even external loads.

Additionally, it is difficult to get access to the internal space of commercial simulation software. Therefore, the non-intrusive MOR methods that can deal with generalized nonlinear FOMs are strongly demanded by industry. Based on this requirement, many different non-intrusive MOR methods have been developed recently. These methods usually intend to construct ROMs based on the data of FOMs. Since these data are essentially inputs and outputs of the FOMs, which are always available for software users at all levels, the principle of non-intrusiveness is preserved.

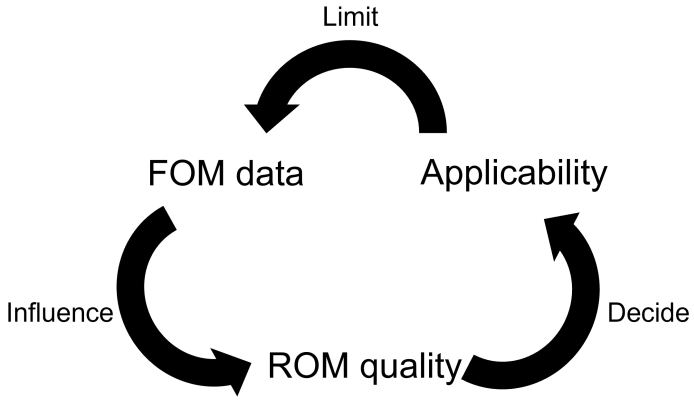


Figure 1.1.: The circle for analyzing the applicability of non-intrusive MOR in industry.

Based on our experience, we can create a circle for analyzing the applicability of non-intrusive MOR in industry. As shown in Figure 1.1, ROM quality is the main factor deciding if a ROM can be applied in industry. The information used to construct the ROM, as described before, is mainly the FOM data. The FOM data are therefore the major influence on the quality of the non-intrusive ROM. Usually, we can evaluate the FOM data in two aspects: quality and quantity, which are also concerned in other ML-related applications, e.g., [32, 33]. However, compared to other ML applications, the training data in our MOR context is obtained directly from the FOM solver. Therefore, we hardly have issues like mislabeling, data corruption, etc. The quality we care about is mainly data diversity/distribution. Besides data quality, the quantity of the FOM data directly decides how much computational resource is needed for constructing the ROM. Although the ROM's performance in the online phase is important, successful deployment of the ROM in real industrial use cases also requires that the computational resource and time budget in the offline phase should be acceptable. This can be considered as a limitation added to the step of generating FOM data. The three parts, ROM quality, FOM data, and Applicability, form a "deadlock" as Figure 1.1 and make the deployment of non-intrusive MOR in industry difficult.

There are two keys to breaking this deadlock. The first key is developing physics-informed MOR. In other research, using physics-informed ML to learn the governing equations of physics/engineering problems [34, 35, 36] also has potential to be applied to a MOR problem. By using physics-informed methods, the resulting ROMs are restricted by physics knowledge, and their model flexibility is therefore reduced. The reduction of the model flexibility eventually leads to a fact that less training data are needed for the ROMs.

The other key is improving the efficiency of snapshot collection, which will be the main focus of this thesis. By improving the efficiency of snapshot collection, we expect that with the same amount of training data, we can construct a ROM which has similar performance to an intrusive ROM and can enable accurate real-time simulation in industry.

1.3. Thesis contribution

To further improve the conventional approaches, an AL-MOR method is developed in this thesis. The proposed method can be combined with different non-intrusive MOR methods and significantly improve their performance. In this work, three existing data-driven MOR methods, the Discrete Empirical Interpolation Method (DEIM) [37], the Runge-Kutta Neural Network (RKNN) [38] and OpInf [15] will be introduced. The selected MOR methods are representative of different MOR sub-classes. The DEIM is a typical intrusive data-driven method, while the others are purely non-intrusive. It will be used as another reference in comparison apart from FOMs to show how good a ROM can be with the availability of intrusiveness. Among the other non-intrusive methods, the RKNN is based on Artificial Neural Network (ANN) and belongs to the black-box method. OpInf is a glass-box method, where limited physics knowledge will be used for ROM construction. We will enhance the non-intrusive methods, the RKNN and OpInf, with our AL-MOR method. We will compare the enhanced non-intrusive ROMs with the FOMs and the intrusive ROMs built with the DEIM. The advantages and disadvantages of each method can be investigated clearly. Through this research, the following contributions are made:

1. To get rid of determining ROM sizes manually, we improve the ROM size determination method proposed in [39]. The modified method is summarized in an algorithm which automatizes the determination process and makes ROM size determination more strategical.

2. In conventional data-driven MOR methods, while sampling training data for constructing ROMs, we only care about the diversity of sampled FOM parameters and assume the diversity of sampled FOM states is guaranteed by

the parameter diversity. However, in this thesis, we prove the independence between the parameter sample diversity and the state sample diversity using real numerical examples.

3. To increase the diversity of sampled FOM states, we propose a novel sampling strategy, named Joint Space Sampling (JSS). In this method, we create a joint space consisting of a reduced solution space for sampling model states and a parameter space for sampling model parameters. Sampling in such a joint space can produce training data with better distribution of both FOM states and FOM parameters.

4. We propose a case-independent ROM validation method based on the Probably Approximately Correct (PAC) learning theory. Using this method, we can determine the accuracy and the corresponding confidence of ROMs without designing specific use cases. The validation result is free from the bias introduced by manual generation of test cases.

5. To accelerate the construction of ROMs, we employ the concept of AL. To select samples actively, we test different sample selection methods, including the Maximum Mean Squared Error (MMSE), Query by Committee (QBC) and Passive Learning (PL) based on the Farthest Point Sampling (FPS) method. Their efficiency and accuracy are analyzed by numerical experiments.

1.4. Overview of the thesis

This thesis is organized with seven chapters. In chapter 1, some background information is given to clarify the motivation of this research. The research questions which will be addressed in this thesis are also raised. In chapter 2, we will introduce the concepts and the theories of three projection-based MOR methods. Among them, we will focus on POD, which will be the approach to constructing reduced space for all FOMs in this work. In chapter 3, we introduce advanced methods helping with reducing nonlinear FOMs. In chapter 4, the concept of AL and some popular AL methods are introduced. In chapter 5, a method determining the optimal size of reduced space is proposed. In chapter 6, we propose a novel sampling method JSS. Furthermore, we integrate the AL methods introduced in chapter 4 into the MOR context and design an AL-MOR method. Besides the selection strategies, different ways of validating ROMs are introduced. In chapter 7, we test different MOR approaches with a series of numerical experiments. The evaluation of their performances is given. Finally, in chapter 8, some discussion about the research results will be given, and we will also conclude our research.

Part II.

Theoretical Background

2. Projection-Based MOR

Model Order Reduction (MOR) aims to reduce the complexity of a numerical model. To continue the discussion, the model's complexity here is defined as the size of the numerical model. The numerical model is the discrete form of its corresponding physics problem. Typically, each physics quantity at each degree of freedom (DOF) will have a contribution to the model size. When the original system is meshed with a fine grid leading to a large number of DOFs, the resulting numerical model will also have much complexity, and solving it becomes slow and computationally intensive.

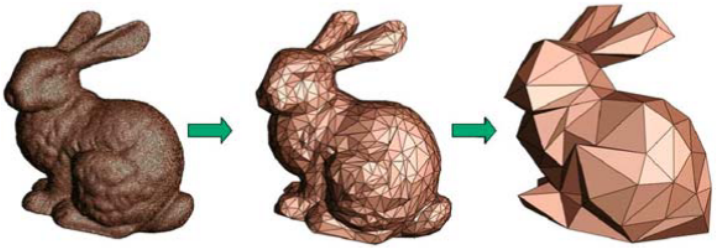


Figure 2.1.: The famous MOR rabbit.

We can take the rabbit model example (Figure 2.1) from [40] to illustrate the idea of MOR. It is a good illustration, however, here we must clarify some information. MOR does not simply discard some DOFs while keeping the others or mesh the problem with a very coarse grid. Such a simple reduction strategy will remarkably reduce the accuracy of the simulation. It is possible, however, to use MOR to find other low-dimensional representations that can actually preserve much of the accuracy of Full Order Models (FOMs). The most popular way is the so-called projection-based MOR. With this kind of MOR methods, we will first create a reduced space by defining a reduced basis. Then using the reduced basis, we can project the FOM onto the reduced space. In such a space, the important information of the FOM is retained, while the trivial information is discarded.

There are different ways to construct the reduced basis. In this chapter, we will introduce three common projection-based MOR methods: (1) Moment Matching (2) Balanced Truncation (3) Proper Orthogonal Decomposition (POD). Among them, we will focus on POD which is data-driven and more suitable to the scope of this thesis.

2.1. Moment Matching

Assuming we have a Linear Time Invariant (LTI) system:

$$\boldsymbol{E}\dot{\boldsymbol{y}} = \boldsymbol{A}\boldsymbol{y} + \boldsymbol{B}\boldsymbol{u}(t), \tag{2.1}$$

where $\boldsymbol{E}, \boldsymbol{A} \in \mathbb{R}^{N\times N}$, $\boldsymbol{B} \in \mathbb{R}^{N\times N_u}$, $\boldsymbol{y} \in \mathbb{R}^N$, and $\boldsymbol{u}(t) \in \mathbb{R}^{N_u}$.

The transfer function of this system is:

$$\boldsymbol{G}(s) = (s\boldsymbol{E} - \boldsymbol{A})^{-1}\boldsymbol{B}. \tag{2.2}$$

If we choose the expansion point as $s = 0$ and the resulting Tayler Series of Equation 2.2 is:

$$\boldsymbol{G}(s) = -\boldsymbol{A}^{-1}\boldsymbol{B} - (\boldsymbol{A}^{-1}\boldsymbol{E})\boldsymbol{A}^{-1}\boldsymbol{B}s - (\boldsymbol{A}^{-1}\boldsymbol{E})^2\boldsymbol{A}^{-1}\boldsymbol{B}s^2 - \cdots - (\boldsymbol{A}^{-1}\boldsymbol{E})^i\boldsymbol{A}^{-1}\boldsymbol{B}s^i. \tag{2.3}$$

We define:

$$\boldsymbol{M}_i = (\boldsymbol{A}^{-1}\boldsymbol{E})^i\boldsymbol{A}^{-1}\boldsymbol{B} \tag{2.4}$$

as the i^{th} moment of the LTI system.

We define a projection-based Reduced Order Model (ROM), which is the FOM's projection in a reduced space using the left reduced basis $\boldsymbol{W} \in \mathbb{R}^{N\times N_r}$ and the right reduced basis $\boldsymbol{V} \in \mathbb{R}^{N\times N_r}$:

$$\boldsymbol{W}^T\boldsymbol{E}\boldsymbol{V}\dot{\boldsymbol{x}} = \boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V}\boldsymbol{x} + \boldsymbol{W}^T\boldsymbol{B}\boldsymbol{u}(t), \tag{2.5}$$

where $\boldsymbol{W}^T\boldsymbol{E}\boldsymbol{V} = \boldsymbol{E}_r, \boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V} = \boldsymbol{A}_r \in \mathbb{R}^{N_r\times N_r}$, $\boldsymbol{W}^T\boldsymbol{B} = \boldsymbol{B}_r \in \mathbb{R}^{N_r\times N_u}$, and $\boldsymbol{x} \in \mathbb{R}^{N_r}$.

Similarly, we can define the i^{th} moment of the ROM:

$$\check{\boldsymbol{M}}_i = (\boldsymbol{A}_r^{-1}\boldsymbol{E}_r)^i\boldsymbol{A}_r^{-1}\boldsymbol{B}_r. \tag{2.6}$$

By enforcing the first N_r moments of the ROM to match the first N_r moments of the FOM, i.e.:

$$\begin{aligned}
(\boldsymbol{A}^{-1}\boldsymbol{E})^0\boldsymbol{A}^{-1}\boldsymbol{B} &= [(\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V})^{-1}\boldsymbol{W}^T\boldsymbol{E}\boldsymbol{V}]^0(\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V})^{-1}\boldsymbol{W}^T\boldsymbol{B},\\
(\boldsymbol{A}^{-1}\boldsymbol{E})^1\boldsymbol{A}^{-1}\boldsymbol{B} &= [(\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V})^{-1}\boldsymbol{W}^T\boldsymbol{E}\boldsymbol{V}]^1(\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V})^{-1}\boldsymbol{W}^T\boldsymbol{B},\\
&\vdots\\
(\boldsymbol{A}^{-1}\boldsymbol{E})^{N_r}\boldsymbol{A}^{-1}\boldsymbol{B} &= [(\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V})^{-1}\boldsymbol{W}^T\boldsymbol{E}\boldsymbol{V}]^{N_r}(\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V})^{-1}\boldsymbol{W}^T\boldsymbol{B}.
\end{aligned} \tag{2.7}$$

Equation 2.7 can be solved by Arnoldi algorithm [41, 4], and the result is:

$$\boldsymbol{V} = \begin{bmatrix}\boldsymbol{A}^{-1}\boldsymbol{B} & \boldsymbol{A}^{-1}\boldsymbol{E}\boldsymbol{A}^{-1}\boldsymbol{B} & \dots & (\boldsymbol{A}^{-1}\boldsymbol{E})^{N_r/N_u-1}\boldsymbol{A}^{-1}\boldsymbol{B}\end{bmatrix}. \tag{2.8}$$

Let

$$\boldsymbol{W}^T\boldsymbol{V} = \boldsymbol{1}, \tag{2.9}$$

and we will get the left reduced basis $\boldsymbol{W}$. The basis $\boldsymbol{V}$ and $\boldsymbol{W}$ construct a Krylov subspace. Such a method is also called one-sided Krylov subspace method.

The original Moment Matching method has good performance especially for the LTI system. Later, its variants [42, 43] can also deal with other complex systems.

2.2. Balanced Truncation

We consider a state space system:

$$\begin{cases} \boldsymbol{E}\dot{\boldsymbol{y}} &= \boldsymbol{A}\boldsymbol{y} + \boldsymbol{B}\boldsymbol{u}(t) \\ \boldsymbol{z} &= \boldsymbol{C}\boldsymbol{y} \end{cases}. \tag{2.10}$$

Besides the notation introduced in the previous section, here $\boldsymbol{C} \in \mathbb{R}^{p\times N}$ is an observation matrix and $\boldsymbol{z} \in \mathbb{R}^p$ is the observed state. As the start point of Balanced Truncation method [44, 45], we list two Lyapunov equations:

$$\begin{aligned} \boldsymbol{A}\boldsymbol{P}\boldsymbol{E}^T + \boldsymbol{E}\boldsymbol{P}\boldsymbol{A}^T &= -\boldsymbol{B}\boldsymbol{B}^T, \\ \boldsymbol{A}^T\boldsymbol{Q}\boldsymbol{E} + \boldsymbol{E}^T\boldsymbol{Q}\boldsymbol{A} &= -\boldsymbol{C}^T\boldsymbol{C}. \end{aligned} \tag{2.11}$$

The matrix $\boldsymbol{P}$ and $\boldsymbol{Q}$ solved from Equation 2.11 is the controllability and observability Gramians of Equation 2.10, respectively. The method of Balanced Truncation, aims to remove the states that are hard to control or observe. The controllability and observability Gramians of the resulting ROM are expected to be equal and both are diagonal.

Normally, for computational efficiency, we will first compute two factors $\boldsymbol{P}_C$ and $\boldsymbol{Q}_C$ such that [45]:

$$\begin{aligned} \boldsymbol{P} &\approx \boldsymbol{P}_C\boldsymbol{P}_C^T, \\ \boldsymbol{Q} &\approx \boldsymbol{Q}_C\boldsymbol{Q}_C^T. \end{aligned} \tag{2.12}$$

Then we apply Singular Value Decomposition (SVD) to $\boldsymbol{P}_C^T\boldsymbol{Q}_C$:

$$\boldsymbol{P}_C^T\boldsymbol{Q}_C = \begin{bmatrix} \boldsymbol{U}_1 & \boldsymbol{U}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{\Sigma}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\Sigma}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{\Upsilon}_1^T \\ \boldsymbol{\Upsilon}_2^T \end{bmatrix}. \tag{2.13}$$

The left reduced basis and the right reduced basis are:

$$\boldsymbol{W} = \boldsymbol{Q}_C\boldsymbol{\Upsilon}_1\boldsymbol{\Sigma}_1^{-1/2} \tag{2.14}$$

and:

$$\boldsymbol{V} = \boldsymbol{P}_C\boldsymbol{U}_1\boldsymbol{\Sigma}_1^{-1/2}, \tag{2.15}$$

where $\Sigma_1^{-1/2} = \text{diag}\left(\frac{1}{\sqrt{\sigma_1}}, \frac{1}{\sqrt{\sigma_2}}, \ldots \frac{1}{\sqrt{\sigma_{N_r}}}\right)$. Finally, the ROM is:

$$\begin{cases} \boldsymbol{W}^T\boldsymbol{E}\boldsymbol{V}\dot{\boldsymbol{y}}_r &= \boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V}\boldsymbol{y}_r + \boldsymbol{W}^T\boldsymbol{B}\boldsymbol{u}(t), \\ \boldsymbol{z} &= \boldsymbol{C}\boldsymbol{V}\boldsymbol{y}_r. \end{cases} \tag{2.16}$$

The advantages of the Balanced Truncation method are the stability of the ROM and the availability of an a priori error bound [46].

2.3. POD

POD is a universal method that can construct the reduced space for most types of full order problems without the intrusiveness to original full order problems. As it is a data-driven method, the FOM can be treated as a black box, and only its input and output data are necessary for analysis.

The workflow of POD consists of 3 steps. The first step is defining a parameter space for the FOM. Secondly, several FOM simulations will be performed with different parameter combinations in the parameter space. The solutions of these simulations are so-called snapshots. Finally, SVD will be applied to the snapshots to construct the reduced basis of the subspace.

2.3.1. Problem formulation

For POD, we can assume a more generalized FOM governing differential equation, which should have the form:

$$\frac{d\boldsymbol{y}(t)}{dt} = \boldsymbol{f}(\boldsymbol{y}(t); \boldsymbol{\mu}) \tag{2.17}$$

or for simplicity:

$$\dot{\boldsymbol{y}} = \boldsymbol{f}(\boldsymbol{y}; \boldsymbol{\mu}), \tag{2.18}$$

where $\boldsymbol{y} \in \mathbb{R}^N$ is the state vector of the FOM, or FOM state in short. N is the FOM size. $\boldsymbol{f}$ is the right hand side (RHS) of the equation, and it can be arbitrary function of $\boldsymbol{y}$ configured by system parameters $\boldsymbol{\mu} \in \mathbb{R}^{N_u}$. N_u is the number of system parameters of the FOM.

The aim of POD is to find a low-dimensional space, where we can describe physics of the original FOM with a reduced governing equation:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}_r(\boldsymbol{x}; \boldsymbol{\mu}), \tag{2.19}$$

where $\boldsymbol{x} \in \mathbb{R}^{N_r}$, and $N_r \ll N$ is called the size of the ROM.

2.3.2. Defining the parameter space

The parameter space is a multi-dimensional space defining the range of all the system parameters. Each dimension of the parameter space stands for a controllable parameter of the FOM. A simple way to define the parameter space is a hyper-cubic space whose upper/lower limit of each dimension is defined by their maximum/minimum values of the parameters:

$$\mathcal{M} = [\mu_{1,\min}, \mu_{1,\max}] \times [\mu_{2,\min}, \mu_{2,\max}] \times ... \times [\mu_{N_u,\min}, \mu_{N_u,\max}]. \tag{2.20}$$

We can use a multivariate sampling method such as Latin Hypercube Sampling [47], Hammersley Sampling, Halton Sampling [48] or Sobol Sampling [49] to sample a set (M) of parameter combinations from $\mathcal{M}$:

$$M = \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, ..., \boldsymbol{\mu}_{N_{\text{sim}}}\}, \tag{2.21}$$

and they are the samples from the parameter space which can be used to create N_{sim} full order problems.

2.3.3. Collecting snapshots

As introduced before, snapshots are nothing but a set of FOM states as solutions to full order problems. To increase the diversity of the sampled snapshots, we will excite the FOM with different system parameters collected in the predefined parameter space $\mathcal{M}$. The N_{sim} parameter configurations in M can be assigned to the FOM and N_{sim} full order Initial Value Problem (IVP)s are constructed correspondingly. In these simulations, the input parameters will be time-independent (constant):

$$\boldsymbol{\mu}^{(i)}(t) \equiv \boldsymbol{\mu}^{(i)}. \tag{2.22}$$

These simulations can produce N_{sim} FOM solutions. If we define the time span for the FOM simulation is $(t_0, t_{\text{end}}]$ and assume there are K steps in the simulation, we will have $N_s = N_{\text{sim}} \times (K+1)$ FOM state vectors in total, which are the so-called snapshots. The collection strategy using the parameters described in Equation 2.22 will be called Static Parameter Sampling (SPS) in this thesis, which is probably the most widely used snapshot collection strategy nowadays.

Snapshots have a great impact on the accuracy of ROMs. Especially the quantity and quality of snapshots. On the one hand, more snapshots are always good for constructing ROMs. On the other hand, snapshots are generated by FOMs whose solutions are complicated and slow. Therefore, the strategy of collecting snapshots must be carefully designed to reduce the workload during the offline phase of MOR. Snapshots generated by different collection strategies could have very different quality, which will further influence the reduction steps. In section 6.1 and section 6.2, two different strategies for snapshot collection will be introduced.

2.3.4. Singular Value Decomposition

If we place all the state vectors in a matrix column-wise, a snapshot matrix is obtained, which takes the form:

$$\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{y}_{10} & \boldsymbol{y}_{11} & \cdots & \boldsymbol{y}_{1K} & \cdots & \boldsymbol{y}_{N_{\text{sim}}0} & \boldsymbol{y}_{N_{\text{sim}}1} & \cdots & \boldsymbol{y}_{N_{\text{sim}}K} \end{bmatrix}, \tag{2.23}$$

where $\boldsymbol{y}_{ij}$ means $\boldsymbol{y}(t = t_j)$ in the i^{th} simulation, and $t_j = t_0 + j \cdot \delta t,\ j = 0, 1, 2, ..., K$.

The next step is finding a set of orthonormal basis vectors $\boldsymbol{V} = \begin{bmatrix} \boldsymbol{v}_1 & \boldsymbol{v}_2 & \dots & \boldsymbol{v}_{N_r} \end{bmatrix} \in \mathbb{R}^{N \times N_r}$, which should minimize the error:

$$\|\boldsymbol{Y} - \boldsymbol{V}\boldsymbol{V}^T\boldsymbol{Y}\|_2^2. \tag{2.24}$$

The interpretation to Equation 2.24 is that we first project the snapshots onto the reduced space and then do the back-projection. This error describes how much information can be retrieved after reduction. The orthonormal basis $\boldsymbol{V}$ to minimize Equation 2.24 can be computed by applying truncated SVD to $\boldsymbol{Y}$:

$$\boldsymbol{Y} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{W}^T. \tag{2.25}$$

The left singular vector $\boldsymbol{V} \in \mathbb{R}^{N \times N_r}$ is then used as the reduced basis for MOR. The smaller N_r is, the more compact the ROM is. It also corresponds to how many singular values are truncated in SVD. Due to the truncation, the information stored in the discarded singular vectors is lost. Therefore, a smaller ROM always has a larger projection error compared to a bigger ROM. For MOR, determination of the size of the ROM is essentially making a trade-off between computational efficiency and simulation accuracy. In [50], the author gives a quite complete review of the methods of determining the number of POD modes (or POD components in the language of this thesis). To quantitatively evaluate the quality of the reduction basis, Equation 2.24 can be used to compute the projection error. In section 4.2, we will introduce a way to determine the ROM size based on the projection error.

There is another interpretation for POD which also generates its name Principal Component Analysis (PCA) in the field of Data Science. Understanding this interpretation can help us understand why POD can reduce the dimension of data. The example in Figure 2.2 shows the concept of PCA.

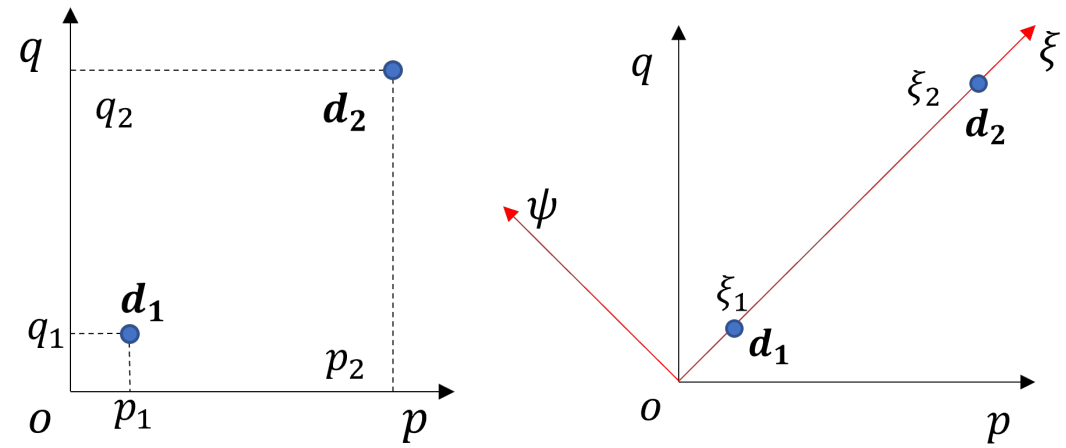
In Figure 2.2a, we use natural coordinate system to describe the positions of two data points. To clarify the position of each point without confusion, we need a p coordinate and a q coordinate. However, by re-defining coordinate system properly, as shown in Figure 2.2b, to describe the position of each point, only the axle ξ is needed and the other axle ψ can be discarded. Here, the coordinate ξ_i is the so-called principal component.

If we define the coordinates in the natural coordinate system as:

$$\boldsymbol{d}_1 = \begin{bmatrix} p_1 \\ q_1 \end{bmatrix}, \boldsymbol{d}_2 = \begin{bmatrix} p_2 \\ q_2 \end{bmatrix}. \tag{2.26}$$

Their coordinates in the new coordinate system is:

$$\boldsymbol{d}_1^* = \begin{bmatrix} \xi_1 \\ \psi_1 \end{bmatrix}, \boldsymbol{d}_2^* = \begin{bmatrix} \xi_2 \\ \psi_2 \end{bmatrix}. \tag{2.27}$$



(a) Data described by two coordinates. (b) Data described by one coordinate.

Figure 2.2.: Data dimension reduction.

We prescribe the new coordinate to be orthogonal. To use as few axis as possible, we need to maximize the squared deviation of ξ_i:

$$\text{maximize } \xi_1^2 + \xi_2^2. \tag{2.28}$$

If we define the orthonormal basis of the new coordinate system to be:

$$\boldsymbol{e}_1 = \begin{bmatrix} e_{11} \\ e_{12} \end{bmatrix}, \boldsymbol{e}_2 = \begin{bmatrix} e_{21} \\ e_{22} \end{bmatrix}, \tag{2.29}$$

such that $\xi_i = \boldsymbol{d}_i \cdot \boldsymbol{e}_1$. Then Equation 2.28 can be re-written as:

$$\text{maximize } \xi_1^2 + \xi_2^2 = \text{maximize } \boldsymbol{e}_1^T \begin{bmatrix} p_1^2 + p_2^2 & p_1 q_1 + p_2 q_2 \\ p_1 q_1 + p_2 q_2 & q_1^2 + q_2^2 \end{bmatrix} \boldsymbol{e}_1. \tag{2.30}$$

If we denote $\boldsymbol{D} = [\boldsymbol{d}_1, \boldsymbol{d}_2]$, Equation 2.30 is equivalent to:

$$\text{maximize } \boldsymbol{e}_1^T \boldsymbol{D}\boldsymbol{D}^T \boldsymbol{e}_1. \tag{2.31}$$

Then we apply Eigenvalue Decomposition to $\boldsymbol{D}\boldsymbol{D}^T$ get:

$$\boldsymbol{D}\boldsymbol{D}^T = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^T. \tag{2.32}$$

Substituting Equation 2.32 into Equation 2.31 yields:

$$\begin{gathered} \text{maximize } \boldsymbol{e}_1^T \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^T \boldsymbol{e}_1 \\ \Updownarrow \\ \text{maximize } (\boldsymbol{U}^T \boldsymbol{e}_1)^T \boldsymbol{\Sigma} (\boldsymbol{U}^T \boldsymbol{e}_1). \end{gathered} \tag{2.33}$$

Since $\boldsymbol{U}$ is an orthonormal matrix and the norm of $\boldsymbol{e}_1$ is 1, if we define $\boldsymbol{n} = \boldsymbol{U}^T\boldsymbol{e}_1$ and $\boldsymbol{n} = [n_1\ n_2]^T$, Equation 2.33 can be reformulated as:

$$\begin{aligned} &\text{maximize } [n_1\ n_2] \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix} [n_1\ n_2]^T \\ &\qquad \Updownarrow \\ &\begin{cases} \text{maximize } \sigma_1 n_1^2 + \sigma_2 n_2^2 \\ n_1^2 + n_2^2 = 1 \\ \sigma_1 > \sigma_2 \end{cases} . \end{aligned} \tag{2.34}$$

Solving Equation 2.34 gives us $n_1 = 1, n_2 = 0$ and $\boldsymbol{e_1} = \boldsymbol{U}[1\ 0]^T$. Here we see, we have selected the first column of $\boldsymbol{U}$ which corresponds to the largest eigenvalue of $\boldsymbol{D}\boldsymbol{D}^T$. Since Eigenvalue Decomposition to $\boldsymbol{D}\boldsymbol{D}^T$ is equivalent to SVD of $\boldsymbol{D}$, the reduced basis obtained in this way is the same as the one we get from the least-approximation-error theory.

2.3.5. Projection and evaluation

Following the previous sections, the reduced basis $\boldsymbol{V}$ has been derived from SVD of the snapshot matrix of the full order system. Using the reduced basis, the full order system state $\boldsymbol{y}$ can be transformed into the reduced order system state $\boldsymbol{x}$ by $\boldsymbol{x} = \boldsymbol{V}^T\boldsymbol{y}$. Now we substitute this transformation into Equation 2.18 and can get:

$$\begin{aligned} \boldsymbol{V}^T\boldsymbol{V}\dot{\boldsymbol{x}} &= \boldsymbol{V}^T\boldsymbol{f}(\boldsymbol{V}\boldsymbol{x};\boldsymbol{\mu}) \\ &\Updownarrow \\ \dot{\boldsymbol{x}} &= \boldsymbol{V}^T\boldsymbol{f}(\boldsymbol{V}\boldsymbol{x};\boldsymbol{\mu}). \end{aligned} \tag{2.35}$$

In the RHS of Equation 2.35, if the function $\boldsymbol{f}(\cdot)$ is a linear function of $\boldsymbol{y}$, then we can re-write Equation 2.35 into:

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{V}^T\boldsymbol{F}_{\text{lin}}(\boldsymbol{\mu})\boldsymbol{V}\boldsymbol{x} + \boldsymbol{V}^T\boldsymbol{f}_\mu(\boldsymbol{\mu}) \\ &= \boldsymbol{F}_{\text{lin,r}}(\boldsymbol{\mu})\boldsymbol{x} + \boldsymbol{V}^T\boldsymbol{f}_\mu(\boldsymbol{\mu}). \end{aligned} \tag{2.36}$$

In Equation 2.36, $\boldsymbol{F}_{\text{lin,r}}$ is the reduced linear operator and can be prepared during the offline phase. Since normally $N_u \ll N$, evaluating $\boldsymbol{V}^T\boldsymbol{f}_\mu(\boldsymbol{\mu})$ is also inexpensive. The solution of such a ROM is therefore fast and computationally inexpensive.

However, if we consider an arbitrary function $\boldsymbol{f}(\cdot)$, as shown in Figure 2.3, during the online phase, the evaluation of such function requires: (1) lifting the reduced state back to the full order space, i.e. $\boldsymbol{V}\boldsymbol{x}$ (2) evaluating the function in the full space, i.e $\boldsymbol{f}(\boldsymbol{V}\boldsymbol{x};\boldsymbol{\mu})$ (3) projecting the result back to the reduced space, i.e. $\boldsymbol{V}^T\boldsymbol{f}(\boldsymbol{V}\boldsymbol{x};\boldsymbol{\mu})$.

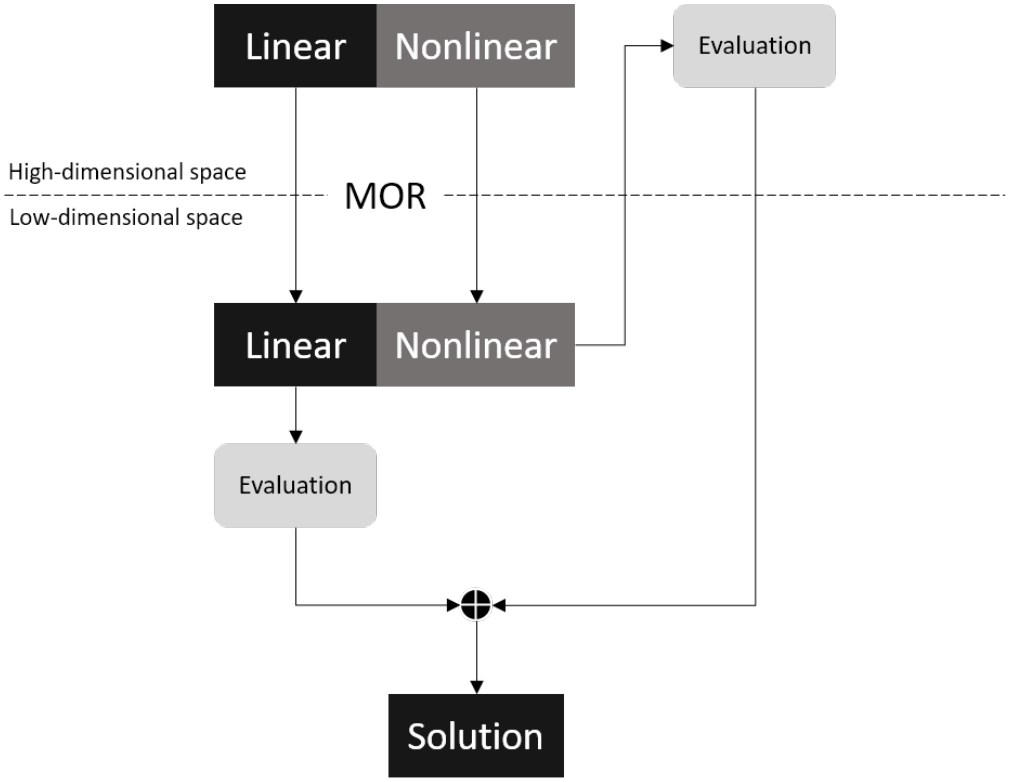


Figure 2.3.: Explanation for why a purely-POD-based ROM becomes slow treating nonlinearity.

The extra operations have high-order complexity and thus will consume additional time in the online phase, which is the main reason why POD can not be directly applied to the system with nonlinearity. To solve this problem, this thesis continues study of methods in two directions: (1) study of an intrusive method, i.e. Discrete Empirical Interpolation Method (DEIM), which has access to all system matrices and tries to find an approximation for nonlinear terms (2) study of non-intrusive MOR methods where no system matrix is available for reduction and ROM identification is therefore further needed.

2.4. Summary

In this section, three typical projection-based MOR methods are introduced. They are Moment Matching, Balanced Truncation and POD. Moment Matching and Balanced Truncation are system-theory-based methods which require detailed information of FOMs. The POD method is a data-driven method without invasion into FOMs, which matches the scope of this thesis best. Two different interpretations of the concept behind POD are introduced. One is the least-approximation-error principle, and the other is the maximum-deviation principle. FOM snapshots are the most important ingredient of POD. The conventional snapshot collection strategy SPS is also introduced. With the POD reduced basis in hands, how to project a FOM onto the reduced space is presented. In this discussion, we find that projecting a linear FOM onto the reduced space is straightforward. However, we can not directly project a FOM with nonlinearity onto the reduced space. As a result, evaluating a purely-POD-based ROM will be slow in the online phase.

3. MOR for Nonlinear Problems

In chapter 2, we have already introduced several methods which can build a reduced space where the most crucial information of Full Order Models (FOMs) is well preserved. At the end of chapter 2, a problem is brought up: once a reduced space is constructed, the linear terms of the FOM can be projected onto the reduced space using the reduced basis, and their projection will be the linear terms of the Reduced Order Model (ROM). However, simply projecting the nonlinear terms onto the reduced space will not give us desired performance boost in the online phase. The nonlinearity in the FOM therefore becomes a big challenge [51].

If we look at physics in real engineering problems, we will find that many of them have nonlinear effects. Therefore, special treatment which can enable effective reduction for nonlinearity in FOMs is required. There are different ways to achieve this goal, and two important ways are hyper-reduction [52, 53] and ROM identification. Hyper-reduction aims at significantly reducing costs of evaluating reduced nonlinear terms by computing them only at a few, selected elements or nodes of the FOM and cheaply approximating the missing information [54]. Among them, the Discrete Empirical Interpolation Method (DEIM) [37] and Missing Point Estimation [55] are well-known.

Besides hyper-reduction, ROM identification is another way to reduce nonlinear problems. Model identification or also called system identification refers to the process of going from observed data to a mathematical model [56]. Within the scope of data-driven Model Order Reduction (MOR), a more detailed definition can be given: after the reduced space is constructed by some means, e.g., Proper Orthogonal Decomposition (POD), the data obtained from the FOM can be projected onto the reduced space. Their projection becomes the reduced expression of the data. The data in the reduced expression can be considered as the output of a latent ROM. The process of finding a model which can reproduce the same output with the corresponding input is considered ROM identification.

In this chapter, the popular hyper-reduction method DEIM will be introduced. As an intrusive method, it makes use of detailed information of FOMs. We also introduce two ROM identification methods: the Artificial Neural Network (ANN) and Operator Inference (OpInf). They are non-intrusive and perfectly fit into the scope of this thesis.

3.1. Hyper-reduction: DEIM

Discrete Empirical Interpolation Method, as its name implies, it is a discrete variant of the Empirical Interpolation Method [57, 58]. DEIM can remarkably reduce the complexity of evaluating nonlinear components in FOMs. Previous research [37] shows that if the dimension of DEIM operators is selected optimally, the approximation error can be nearly negligible. A wide range of DEIM's applications can be found, such as fluid dynamics [59, 60], material mechanics [61], cardiac mechanics [62], etc. Also, many DEIM-based variants, such as [63, 64, 65], have been proposed with enhanced capability for dealing with different types of nonlinear effects.

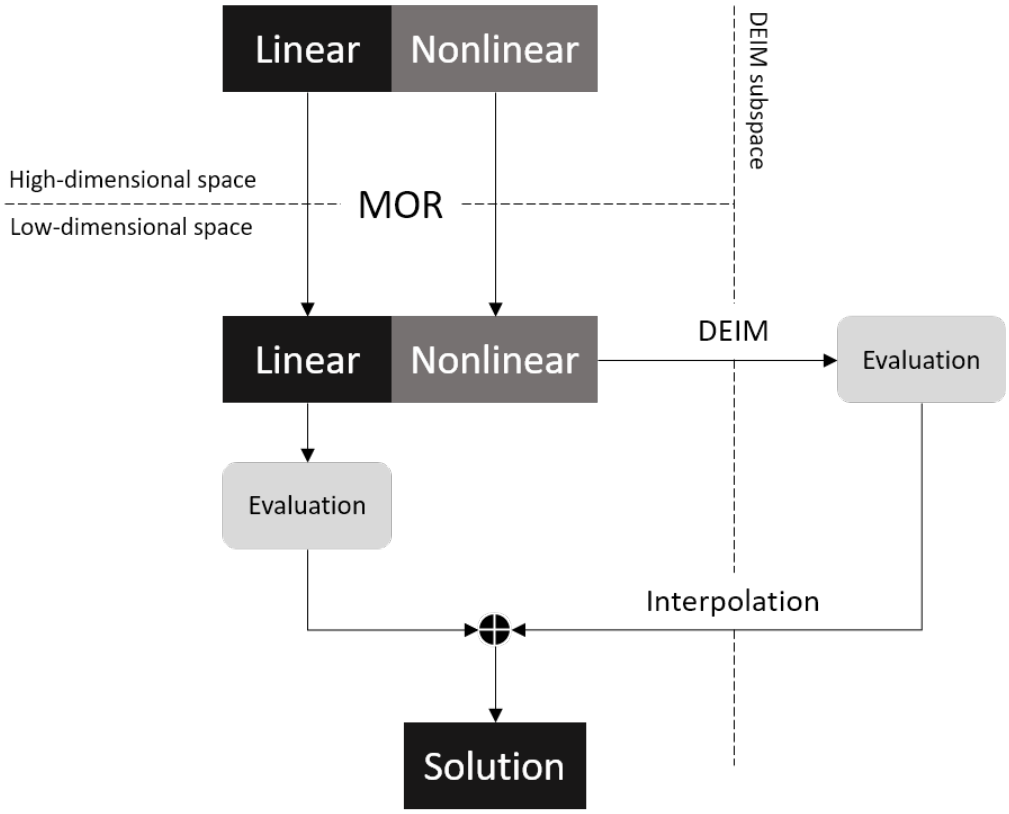


Figure 3.1.: Explanation for DEIM's acceleration dealing with nonlinearity in FOMs

With Figure 2.3, we have already explained why a purely-POD-based ROM becomes slow when the FOM has nonlinear components. To prevent the solution of ROM from evaluating complex nonlinear terms in the full space during the online phase, DEIM constructs a new subspace whose dimension is lower than the dimension of the full space. The subspace construction is completed during the offline phase. While evaluating the ROM, the nonlinear terms will be evaluated in this subspace, where

only a subset of degree of freedoms (DOFs) are selected as the points to evaluate the nonlinear terms in the governing equations. These points are called interpolation points. Then the global evaluation will be approximated by interpolating linearly from these interpolation points. This process is graphically shown in Figure 3.1.

3.1.1. Algorithm of DEIM

Recalling the governing equation Equation 2.18 defined in section 2.3:

$$\dot{\boldsymbol{y}} = \boldsymbol{f}(\boldsymbol{y}; \boldsymbol{\mu}), \tag{3.1}$$

and for simplicity, we assume that the right hand side (RHS) function $\boldsymbol{f}$ is a nonlinear function.

As the basic assumption of DEIM, we approximate the RHS of Equation 3.1 by:

$$\boldsymbol{f}(\cdot) \approx \boldsymbol{H}\boldsymbol{c}(\cdot), \tag{3.2}$$

where $\boldsymbol{H} = [\boldsymbol{h_1}, \boldsymbol{h_2}, ..., \boldsymbol{h_m}] \in \mathbb{R}^{N \times m}$ is the basis of the DEIM space and $\boldsymbol{c}(\cdot)$ is the DEIM coefficients. m DOFs in the approximation $\boldsymbol{H}\boldsymbol{c}(\cdot)$ are selected to have exactly the same value as the corresponding DOFs in $\boldsymbol{f}(\cdot)$:

$$\boldsymbol{P}^T\boldsymbol{f}(\cdot) = \boldsymbol{P}^T\boldsymbol{H}\boldsymbol{c}(\cdot), \tag{3.3}$$

where the matrix $\boldsymbol{P} \in \mathbb{R}^{N \times m}$ selects m DOFs from the both sides of Equation 3.2. We construct the matrix $\boldsymbol{P}$ as:

$$\boldsymbol{P} = [\boldsymbol{p}_{d_1}\ \boldsymbol{p}_{d_2}\ ...\ \boldsymbol{p}_{d_m}], \tag{3.4}$$

where $\boldsymbol{p}_{d_i} = [0\ 0\ ...\ 1\ ...\ 0\ 0]^T$ is a column vector whose entries are all zeros except the d_i-th entry is one, and d_i is the i-th interpolation point which needs to be determined later by the algorithm.

If Equation 3.3 is substituted into Equation 3.2, one can obtain:

$$\boldsymbol{f}(\cdot) \approx \boldsymbol{H}(\boldsymbol{P}^T\boldsymbol{H})^{-1}\boldsymbol{P}^T\boldsymbol{f}(\cdot). \tag{3.5}$$

There are two crucial ingredients needed for DEIM's approximation: the DEIM space basis $\boldsymbol{H}$ and m interpolation points $D = \{d_1, d_2, ..., d_m\}$ where we enforce the ROM to have the same evaluation as the FOM.

In Equation 2.23, the snapshots of the state vector are collected. We can further use them to compute the snapshot matrix of the nonlinear contribution

$$\boldsymbol{F} = [\boldsymbol{f}_{10}\ \boldsymbol{f}_{11}\ ...\ \boldsymbol{f}_{N_{\text{sim}}K}], \tag{3.6}$$

where $\boldsymbol{f}_{ij} = \boldsymbol{f}(\boldsymbol{y}_{ij})$. In other words, DEIM does not need additional simulation results but just substituting the snapshots of the FOM into the nonlinear function. This

substitution can provide us the snapshots of the nonlinear contribution. Applying truncated Singular Value Decomposition (SVD) to $\boldsymbol{F}$ will generate a set of space basis vectors $\boldsymbol{H} = \{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_m\}$, and they are the DEIM space basis that we are seeking for.

The determination of the other ingredient D can be completed using an iterative algorithm. The core concept of the algorithm is that in each iteration, the DOF that generates the worst approximation should be forced to match the exact solution in the next iteration. Meanwhile, the selection of DEIM space basis $\boldsymbol{H}$ will be performed as well. This procedure can be summarized by Algorithm 1.

Algorithm 1 DEIM algorithm

Initialization:
$d_1 = max(abs(\boldsymbol{h_1})), D = \{\}$
$\boldsymbol{P} = [\boldsymbol{p}_{d_1}], \boldsymbol{H} = [\boldsymbol{h_1}]$
$i = 2$
while $i \leq m$ **do**
  $\boldsymbol{c}_i = (\boldsymbol{P}^T\boldsymbol{H})^{-1}\boldsymbol{P}^T\boldsymbol{h}_i$
  $\boldsymbol{r}_i = \boldsymbol{h}_i - \boldsymbol{H}\boldsymbol{c}_i$
  $d_i = max(|\boldsymbol{r}_i|)$, $D = D \cup \{d_i\}$
  $\boldsymbol{P} = hstack(\boldsymbol{P}, \boldsymbol{p}_{d_i})$ and $\boldsymbol{H} = hstack(\boldsymbol{H}, \boldsymbol{h}_i)$
  $i++$;
end while
return $\boldsymbol{P}, \boldsymbol{H}, D$

In Algorithm 1, $max(\cdot)$ means select the maximum entry. The function $hstack(\cdot)$ stands for "stacking matrices or vectors in sequence horizontally", and in Algorithm 1, it is just appending the new column vector to the current matrix.

3.1.2. Assemble the ROM

In section 3.1.1, the necessary components for assembling the DEIM-ROM are prepared. If Equation 3.5 is substituted into Equation 3.1, the approximation to Equation 3.1 can be written as:

$$\dot{\boldsymbol{y}} = \boldsymbol{H}(\boldsymbol{P}^T\boldsymbol{H})^{-1}\boldsymbol{P}^T\boldsymbol{f}(\cdot). \tag{3.7}$$

Its projection in the reduced space is:

$$\dot{\boldsymbol{x}} = \boldsymbol{V}\boldsymbol{H}(\boldsymbol{P}^T\boldsymbol{H})^{-1}\boldsymbol{P}^T\boldsymbol{f}(\boldsymbol{V}\boldsymbol{x}; \boldsymbol{\mu}). \tag{3.8}$$

Since the matrix $\boldsymbol{P}$ is essentially a selective matrix which contains only zeros and ones, the ultimately reduced form of Equation 3.8 is:

$$\dot{\boldsymbol{x}} = \boldsymbol{V}\boldsymbol{H}(\boldsymbol{P}^T\boldsymbol{H})^{-1}\boldsymbol{f}(\boldsymbol{P}^T\boldsymbol{V}\boldsymbol{x};\boldsymbol{\mu}). \tag{3.9}$$

In Equation 3.9, the components which can be pre-computed in the offline phase are $\boldsymbol{V}^T\boldsymbol{H}(\boldsymbol{P}^T\boldsymbol{H})^{-1}$ and $\boldsymbol{P}^T\boldsymbol{V}$.

It is observed that $\boldsymbol{P}^T\boldsymbol{V} \in \mathbb{R}^{m\times N_r}$. Therefore, through DEIM hyper-reduction, the evaluation of the nonlinear function $\boldsymbol{f}(\cdot)$ is performed in a subspace whose dimension is m. Thus, a conclusion can be drawn that the efficiency of a DEIM-ROM depends on m. If $m \ll N$, the evaluation of the DEIM-ROM during the online phase will require much less computational effort. However, the accuracy of the DEIM-ROM is also determined by m. Therefore, making a trade-off between the efficiency and the accuracy is a necessary step in the construction of a DEIM-ROM. When nonlinearity is complex and strong, a bigger m should be chosen to ensure an acceptable accuracy of the resulting DEIM-ROM. Recently, there is some research focusing on different variants of DEIM, which can help DEIM overcome the difficulty of reducing complex nonlinear components. For this kind of variants, readers please refer to [63, 66, 61].

3.2. Black-box ROM identification: Artificial Neural Networks

An ANN is a simplified mathematical model of a human brain [67]. It is famous of its universal approximation [68], i.e. a properly designed ANN can approximate any nonlinear functions. This property allows using an ANN to represent an unknown ROM.

3.2.1. Multilayer Perceptron

Although there are plenty of network architectures, almost all of them have such a basic unit:

$$\boldsymbol{z}_{\text{out}} = \mathcal{A}(\boldsymbol{w}\boldsymbol{z}_{\text{in}} + \boldsymbol{b}), \tag{3.10}$$

where $\boldsymbol{z}_{\text{in}}$ is the input to the unit and $\boldsymbol{z}_{\text{out}}$ is the output from the unit. The coefficients $\boldsymbol{w}$ and $\boldsymbol{b}$ are called weights and bias respectively. The activation function $\mathcal{A}(\cdot)$ enables the unit output to have nonlinear relationship with the input. Such a unit is regarded as a "vanilla" neural net and called as single layer perceptron [69]. However, such a single layer is not able to simulate arbitrary mathematical functions. To enhance the approximation ability, usually multiple layers are concatenated as in Figure 3.2, and the output of a former layer naturally becomes the input to its next layer.

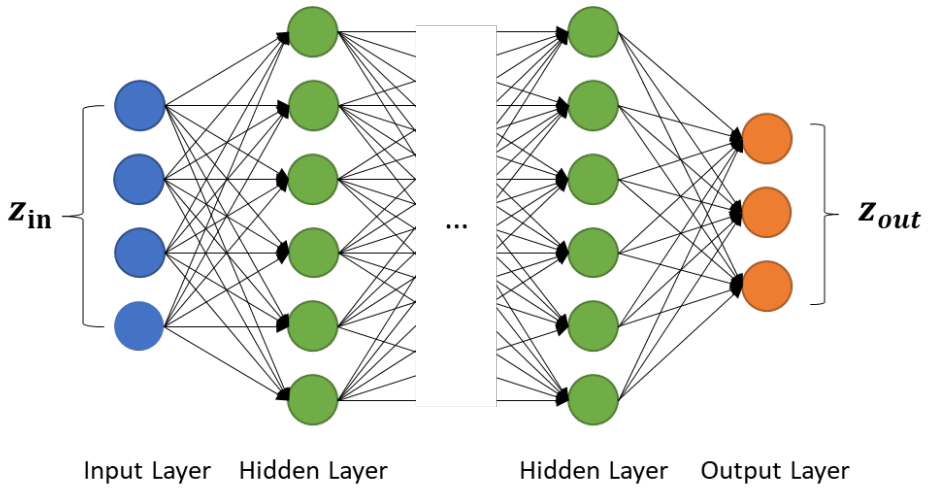


Figure 3.2.: The structure of a multilayer perceptron.

In Figure 3.2, the input layer and all hidden layers are the nonlinear layer described by Equation 3.10. Normally, the output layer has no nonlinear activation function. Therefore, its layer equation can be written as:

$$z_{\text{out}} = wz_{\text{in}} + b. \tag{3.11}$$

A Multilayer Perceptron (MLP) must have one input layer and one output layer. According to universal approximation theorem, a standard multilayer network with as few as a single hidden layer and arbitrary bounded and nonconstant activation function is a universal approximator [70]. This means, a 3-layer MLP can already be a universal approximator. However, this conclusion only cares about the depth of the network. The width, i.e. the number of neurons in each layer, also has a large influence on the approximation [71]. Therefore, a good design of the width and depth is important for a MLP.

A properly designed MLP can certainly learn the evolution of the latent ROM. Using the MLP transfer function $\mathcal{N}$, this can be written as:

$$x_{i+1} = \mathcal{N}(x_i, \mu_i). \tag{3.12}$$

However, such a simple surrogate model has many disadvantages in practice. For example, if the FOM snapshots are collected on a non-uniform time grid, such a learning approach is very likely to fail. Because such a model can only learn the mapping between two consecutive states, which does not have a consistent pattern on a non-uniform time grid. Therefore, inspired by this consideration, in [38], we propose to learn the RHS of a ROM.

3.2.2. Runge-Kutta Neural Network

Previously, we found that the direct projection of the FOM governing equation onto the reduced space has a form of:

$$\dot{\boldsymbol{x}} = \boldsymbol{V}^T \boldsymbol{f}(\boldsymbol{V}\boldsymbol{x}; \boldsymbol{\mu}), \tag{3.13}$$

and we can further summarize Equation 3.13 as:

$$\dot{\boldsymbol{x}} = \boldsymbol{N}(\boldsymbol{x}; \boldsymbol{\mu}). \tag{3.14}$$

The RHS function $\boldsymbol{N}(\cdot)$ defines the system's evolution in the reduced space. Thus, we can also use a MLP to learn the RHS of the reduced governing equation and integrate the surrogate reduced equation with a numerical integration scheme. In [38], we propose to integrate such a surrogate reduced governing equation by the Runge-Kutta (RK) method [72]. In numerical computation, the RK method is widely used to integrate the differential equations numerically. In the RK method, the update of the state vector $\boldsymbol{x}$ is:

$$\begin{aligned}
&\boldsymbol{k}_0 = 0, \\
&\boldsymbol{k}_1 = \delta t \boldsymbol{\mathcal{N}}(\boldsymbol{x}_i, \boldsymbol{\mu}_i), \\
&\boldsymbol{k}_2 = \delta t \boldsymbol{\mathcal{N}}(\boldsymbol{x}_i + c_2 \boldsymbol{k}_1, \boldsymbol{\mu}_i), \\
&\vdots \\
&\boldsymbol{k}_n = \delta t \boldsymbol{\mathcal{N}}(\boldsymbol{x}_i + c_n \boldsymbol{k}_{n-1}, \boldsymbol{\mu}_i), \\
&\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \sum_{j=1}^{n} h_j \boldsymbol{k}_j,
\end{aligned} \tag{3.15}$$

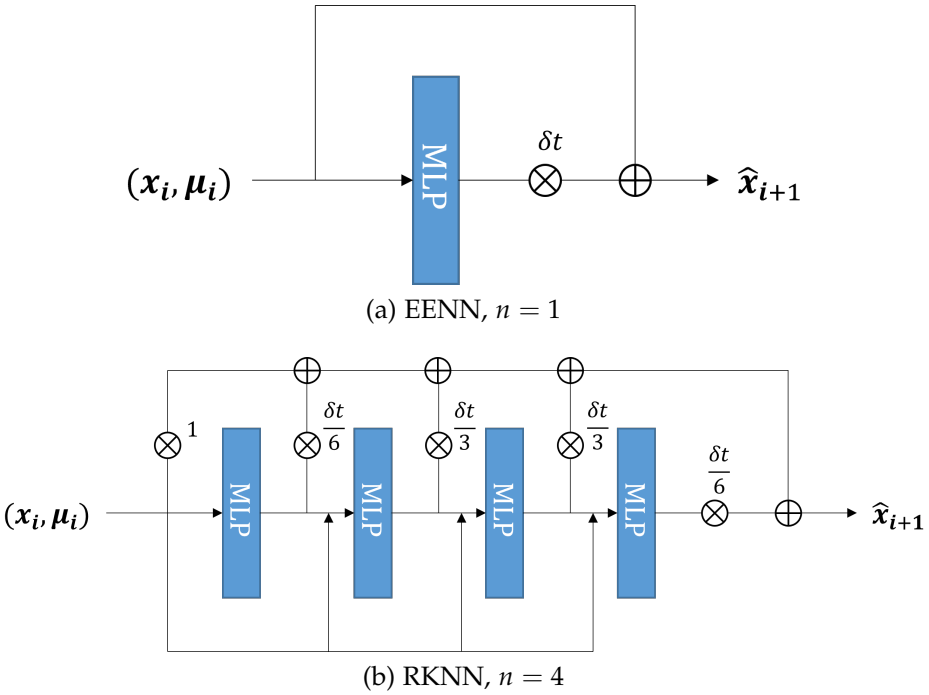
where $\boldsymbol{x}_i$ stands for the reduced state at the time t_i. The coefficients c and h are different for different RK methods.

n determines the order of the RK method. For $n = 1$, Equation 3.15 reduces to the 1^{st}-order Runge-Kutta (RK1) method, which is equivalent to Explicit Euler Integration [72]. In practice, the most popular RK method is the 4^{th}-order Runge-Kutta (RK4) method:

$$\begin{aligned}
&\boldsymbol{k}_1 = \delta t \boldsymbol{\mathcal{N}}(\boldsymbol{x}_i, \boldsymbol{\mu}_i), \\
&\boldsymbol{k}_2 = \delta t \boldsymbol{\mathcal{N}}(\boldsymbol{x}_i + \frac{1}{2}\boldsymbol{k}_1, \boldsymbol{\mu}_i), \\
&\boldsymbol{k}_3 = \delta t \boldsymbol{\mathcal{N}}(\boldsymbol{x}_i + \frac{1}{2}\boldsymbol{k}_2, \boldsymbol{\mu}_i), \\
&\boldsymbol{k}_4 = \delta t \boldsymbol{\mathcal{N}}(\boldsymbol{x}_i + \boldsymbol{k}_3, \boldsymbol{\mu}_i), \\
&\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \frac{1}{6}\boldsymbol{k}_1 + \frac{1}{3}\boldsymbol{k}_2 + \frac{1}{3}\boldsymbol{k}_3 + \frac{1}{6}\boldsymbol{k}_4,
\end{aligned} \tag{3.16}$$

which has a good balance between the integration accuracy and speed.

Two network structures can be designed based on RK1 and RK4 integration. They are called Explicit Euler Neural Network (EENN) and Runge-Kutta Neural Network (RKNN).



(a) EENN, $n = 1$

(b) RKNN, $n = 4$

Figure 3.3.: Picture of the structure of EENN and RKNN.

Looking at Figure 3.3, it is observed that the structure of EENN and RKNN both have a skip connection, i.e. a path directly connects the original input to the output. This is also called residual structure [73]. Compared with a fully-connected structure, the residual structure can prevent deep networks from network degradation, which becomes a natural advantage of using numerical-integration-guided architecture.

3.2.3. Training Artificial Neural Networks

Once the network structure is determined, the next step will be training the ANN. The ANN is initialized with some default weights $\boldsymbol{w}$ and bias $\boldsymbol{b}$. The parameters need to be updated during the training. Before training, we project the snapshot matrix $\boldsymbol{Y}$ onto the reduced space by:

$$\boldsymbol{X} = \boldsymbol{V}^T \boldsymbol{Y}, \tag{3.17}$$

where $\boldsymbol{X} \in \mathbb{R}^{N_s \times N_r}$ is the reduced snapshot matrix. Here we see, compared to the full order snapshots, the reduced snapshots only contain the trajectories of the N_r

required POD components. This feature makes the ROM trained from such a dataset more compact, and less computational resources are needed during the training. By slicing and concatenating properly, we can construct the training datasets:

$$\begin{aligned} \boldsymbol{X}_{\text{in}} &= [\boldsymbol{x}_{10} \ ... \ \boldsymbol{x}_{1K-1} \ ... \ \boldsymbol{x}_{N_{\text{sim}}0} \ ... \ \boldsymbol{x}_{N_{\text{sim}}K-1}], \\ \boldsymbol{X}_{\text{out}} &= [\boldsymbol{x}_{11} \ ... \ \boldsymbol{x}_{1K} \ ... \ \boldsymbol{x}_{N_{\text{sim}}1} \ ... \ \boldsymbol{x}_{N_{\text{sim}}K}], \\ \boldsymbol{U}_{\text{in}} &= [\boldsymbol{\mu}_{10} \ ... \ \boldsymbol{\mu}_{1K-1} \ ... \ \boldsymbol{\mu}_{N_{\text{sim}}0} \ ... \ \boldsymbol{\mu}_{N_{\text{sim}}K-1}]. \end{aligned} \tag{3.18}$$

They are collection of reduced states at time t_i, reduced states at time t_{i+1}, and system parameters at time t_i, respectively. Using them, we can perform the training of the ANN.

Algorithm 2 Training of an EENN

Require: $\boldsymbol{X}_{\text{in}}, \boldsymbol{U}_{\text{in}}, \boldsymbol{X}_{\text{out}}$

while training **do**
  $\hat{\boldsymbol{X}}_{\text{out}} = \boldsymbol{X}_{\text{in}} + \delta t \boldsymbol{\mathcal{N}}([\boldsymbol{X}_{\text{in}}, \boldsymbol{U}_{\text{in}}])$
  $\text{loss} = \text{MSE}(\hat{\boldsymbol{X}}_{\text{out}}, \boldsymbol{X}_{\text{out}})$
  Use backpropagation to update the weights $\boldsymbol{w}$ and bias $\boldsymbol{b}$ of the network
end while

Algorithm 3 Training of a RKNN

Require: $\boldsymbol{X}_{\text{in}}, \boldsymbol{U}_{\text{in}}, \boldsymbol{X}_{\text{out}}, \delta t$

while training **do**
  $\boldsymbol{k}_1 = \delta t \boldsymbol{\mathcal{N}}([\boldsymbol{X}_{\text{in}}, \boldsymbol{U}_{\text{in}}])$
  $\boldsymbol{k}_2 = \delta t \boldsymbol{\mathcal{N}}([\boldsymbol{X}_{\text{in}} + \frac{\boldsymbol{k}_1}{2}, \boldsymbol{U}_{\text{in}}])$
  $\boldsymbol{k}_3 = \delta t \boldsymbol{\mathcal{N}}([\boldsymbol{X}_{\text{in}} + \frac{\boldsymbol{k}_2}{2}, \boldsymbol{U}_{\text{in}}])$
  $\boldsymbol{k}_4 = \delta t \boldsymbol{\mathcal{N}}([\boldsymbol{X}_{\text{in}} + \boldsymbol{k}_3, \boldsymbol{U}_{\text{in}}])$
  $\hat{\boldsymbol{X}}_{\text{out}} = \boldsymbol{X}_{\text{in}} + \frac{1}{6}(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4)$
  $\text{loss} = \text{MSE}(\hat{\boldsymbol{X}}_{\text{out}}, \boldsymbol{X}_{\text{out}})$
  Use backpropagation to update the weights $\boldsymbol{w}$ and bias $\boldsymbol{b}$ of the network
end while

Algorithm 2 and Algorithm 3 describe the training process of an EENN and a RKNN, and similar methods can be used to train networks that are based on different Runge-Kutta schemes. We also use mini-batch method [74] to achieve better training result. During the training process, the aim is to reduce the difference between the predicted new states $\hat{\boldsymbol{X}}_{\text{out}}$ and the reference new states $\boldsymbol{X}_{\text{out}}$. The metric quantifies the difference is called loss function. In Algorithm 2 and Algorithm 3, we use a popular loss function named Mean Squared Error (MSE):

$$\text{MSE}(\hat{\boldsymbol{X}}_{\text{out}}, \boldsymbol{X}_{\text{out}}) = \text{Mean}(\|\hat{\boldsymbol{X}}_{\text{out}} - \boldsymbol{X}_{\text{out}}\|_2^2). \tag{3.19}$$

Besides this, there are other loss functions such as the Mean Absolute Error [75], the Smooth Mean Absolute Error [76], etc. Since predicting trajectories of ROMs is essentially a regression problem, standard MSE can satisfy our needs.

In each iteration, the loss will be computed by the selected loss function. Then backpropagation [77] is used to update the network parameters. Without loss of generality, an ANN with only input layer is used to explain the backpropagation:

1. Compute:
$$\text{loss} = \text{MSE}(\hat{\boldsymbol{X}}_{\text{out}}, \boldsymbol{X}_{\text{out}}). \tag{3.20}$$

2. Compute the derivative with respect to $\boldsymbol{w}$ using the chain rule:
$$\begin{aligned} \frac{\partial \text{loss}}{\partial \boldsymbol{w}} &= \frac{\text{MSE}(\hat{\boldsymbol{X}}_{\text{out}}, \boldsymbol{X}_{\text{out}})}{\partial \hat{\boldsymbol{X}}_{\text{out}}} \frac{\partial \boldsymbol{\mathcal{N}}}{\partial (\boldsymbol{w}\boldsymbol{X}_{\text{in}} + \boldsymbol{b})} \frac{\partial (\boldsymbol{w}\boldsymbol{X}_{\text{in}} + \boldsymbol{b})}{\partial \boldsymbol{w}}, \\ \frac{\partial \text{loss}}{\partial \boldsymbol{b}} &= \frac{\text{MSE}(\hat{\boldsymbol{X}}_{\text{out}}, \boldsymbol{X}_{\text{out}})}{\partial \hat{\boldsymbol{X}}_{\text{out}}} \frac{\partial \boldsymbol{\mathcal{N}}}{\partial (\boldsymbol{w}\boldsymbol{X}_{\text{in}} + \boldsymbol{b})} \frac{\partial (\boldsymbol{w}\boldsymbol{X}_{\text{in}} + \boldsymbol{b})}{\partial \boldsymbol{b}}. \end{aligned} \tag{3.21}$$

3. Update the parameters:
$$\begin{aligned} \boldsymbol{w}_{\text{new}} &= \alpha \frac{\partial \text{loss}}{\partial \boldsymbol{w}} + \boldsymbol{w}, \\ \boldsymbol{b}_{\text{new}} &= \alpha \frac{\partial \text{loss}}{\partial \boldsymbol{b}} + \boldsymbol{b}, \end{aligned} \tag{3.22}$$

where α is called learning rate, and it is one of the hyper-parameters that usually need to be determined before the training starts. Selecting an appropriate learning rate is challenging. If we pick a learning rate that is too small, we risk taking too long during the training process. But if we pick a learning rate that is too big, we will mostly likely start diverging away from the minimum [78]. Therefore, choosing an appropriate learning rate is very crucial for training an ANN. Another important question is when to stop the training iteration. To tackle both problems, Early Stopping [79] combined with learning rate decay [80] can be employed as shown in Algorithm 4.

The basic concept of Algorithm 4 is dynamically tuning the learning rate based on the test loss trend. First, all training data will be split into training datasets and test datasets. The MSE loss computed with the former datasets is called training loss and with the latter datasets is called test loss. The initial learning rate is assigned with a relatively big value, e.g. 10^{-3}. When the training starts, both the training loss and the test loss should drop in the early stage. Gradually, it will be observed that the test loss stops dropping but starts to increase. This indicates that the ANN is overfitting with the current learning rate. In this case, the learning rate will be scaled by a ratio $1/d$. If the test loss cannot keep decreasing with the new learning rate,

Algorithm 4 Early Stopping combined with learning rate decay

Require: $p_{\text{tol}}, q_{\text{tol}}, d, lr, \boldsymbol{X}_{\text{out}}, \boldsymbol{X}_{\text{in}}, \boldsymbol{U}_{\text{in}}$

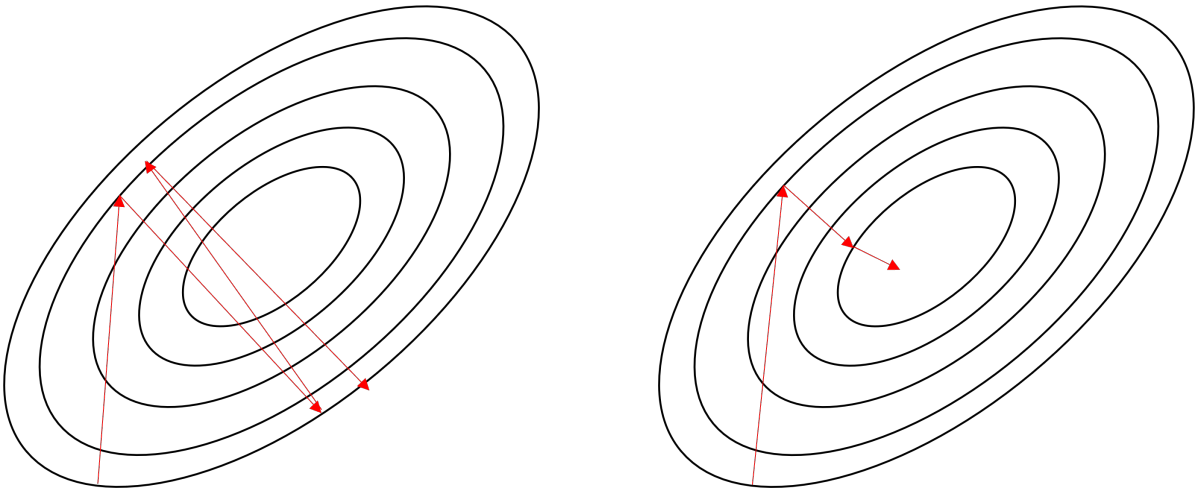
Split $\boldsymbol{X}_{\text{out}}, \boldsymbol{X}_{\text{in}}, \boldsymbol{U}_{\text{in}}$ into $\boldsymbol{X}_{\text{out,train}}, \boldsymbol{X}_{\text{in,train}}, \boldsymbol{U}_{\text{in,train}}$ and $\boldsymbol{X}_{\text{out,test}}, \boldsymbol{X}_{\text{in,test}}, \boldsymbol{U}_{\text{in,test}}$
$p = 0,\ \text{loss}_{\text{test,best}} = +\infty$
while $p \leq p_{\text{tol}}$ **do**
  $q = 0$
  while $q \leq q_{\text{tol}}$ **do**
    Train the network with the learning rate lr and the dataset $\boldsymbol{X}_{\text{out,train}}, \boldsymbol{X}_{\text{in,train}}, \boldsymbol{U}_{\text{in,train}}$
    Use the network to compute the prediction $\hat{\boldsymbol{X}}_{\text{out,test}}$ with $\boldsymbol{X}_{\text{in,test}}$ and $\boldsymbol{U}_{\text{in,test}}$
    Compute the test error $\text{loss}_{\text{test}} = \text{MSE}(\hat{\boldsymbol{X}}_{\text{out, test}}, \boldsymbol{X}_{\text{out,test}})$
    if $\text{loss}_{\text{test}} < \text{loss}_{\text{test,best}}$ **then**
      $q = 0,\ \text{loss}_{\text{test,best}} = \text{loss}_{\text{test}}$
    else
      $q++$
    end if
  end while
  $lr = lr/d, p++$
end while

the training process is stopped. The parameters p_{tol} and q_{tol} are two tolerances of the algorithm. By using this approach, the initial learning rate can be selected less carefully. In Figure 3.4, we visualize error convergence with different choices of the learning rate.

After training the ANN, the final parameters and the network architecture can be saved and reused. There are plenty of mature platforms that can facilitate the implementation of ANN, e.g. Pytorch [81], TensorFlow [82], scikit-learn [83], etc. In this thesis, Pytorch and scikit-learn are used.

3.3. Glass-box ROM identification: Operator Inference

OpInf is a data-driven and physics-informed MOR method. Unlike other black-box MOR methods, OpInf is considered as a glass-box method [84], where some knowledge about full order systems is incorporated for constructing ROMs. A hypothetical form with undetermined operators is prescribed for the ROM with the help of the known knowledge. From abundant solution data of the FOM, OpInf can infer the undetermined operators and produce the ROM in the offline phase. This process is similar to another model identification method called Sparse Identification



(a) Convergence is difficult when the learning rate is too large.

(b) Successful convergence with the adaptive learning rate.

Figure 3.4.: The convergence in the error contours with different learning rates.

of Nonlinear Dynamics (SINDy) [85].

3.3.1. Hypothetical governing equation

A basic assumption applied by OpInf is that the ROM is driven by an Ordinary Differential Equation (ODE) which takes the form of:

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{F_1}\boldsymbol{x} + \boldsymbol{F_B}\boldsymbol{\mu} + \boldsymbol{F_C} + \boldsymbol{F_2}\boldsymbol{x} \otimes \boldsymbol{x} + \boldsymbol{F_N}\boldsymbol{\mu} \otimes \boldsymbol{x}, \tag{3.23}$$

where $\boldsymbol{x}$ is the state vector of the ROM, $\boldsymbol{F_1}, \boldsymbol{F_2}, \boldsymbol{F_B}, \boldsymbol{F_C}$ and $\boldsymbol{F_N}$ are operators to be determined. The operation $\otimes$ is the Kronecker product:

$$\boldsymbol{x} \otimes \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N_r} \end{bmatrix} \otimes \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N_r} \end{bmatrix} = \begin{bmatrix} x_1 x_1 \\ x_1 x_2 \\ \vdots \\ x_1 x_{N_r} \\ x_2 x_2 \\ x_2 x_3 \\ \vdots \\ x_{N_r} x_{N_r} \end{bmatrix}. \tag{3.24}$$

As we see, a hypothesis taken by OpInf is that the RHS in the reduced space can be represented by polynomial terms up to 2^{nd} order. Because the projection from the

high-dimensional space onto the low-dimensional space is a linear projection, this implies that the RHS of the governing equation of the FOM can be represented by polynomial terms up to 2nd order.

3.3.2. Lifting the governing equation

The previous prerequisite might be considered as applying a hard restriction to the full order system, and any nonlinearity higher than 2nd order in the FOM cannot be approximated. However, if we re-write the governing equations in some certain way, the governing equations of most engineering problems can be converted in to a quadratic form, which is called lifting in this thesis or is also called Transform and Learn in [86].

Without loss of generality, we can define the governing equation of the FOM as:

$$\dot{\boldsymbol{y}} = \boldsymbol{A}\boldsymbol{y} + \sum_{i=1}^{n} \boldsymbol{f}_i(\boldsymbol{y};\boldsymbol{\mu}), \tag{3.25}$$

where $\boldsymbol{f}_i$ is an arbitrary function. As introduced in [87], we can define a new state variable $\boldsymbol{s}$ as:

$$\boldsymbol{s} = \begin{bmatrix} \boldsymbol{s}_1(\boldsymbol{y}) \\ \vdots \\ \boldsymbol{s}_n(\boldsymbol{y}) \end{bmatrix}. \tag{3.26}$$

Then we have:

$$\dot{\boldsymbol{s}} = \begin{bmatrix} \dot{\boldsymbol{s}_1}(\boldsymbol{y}) \\ \vdots \\ \dot{\boldsymbol{s}_n}(\boldsymbol{y}) \end{bmatrix} = \begin{bmatrix} \frac{\partial \boldsymbol{s}_1}{\partial \boldsymbol{y}} \frac{\partial \boldsymbol{y}}{\partial t} \\ \vdots \\ \frac{\partial \boldsymbol{s}_n}{\partial \boldsymbol{y}} \frac{\partial \boldsymbol{y}}{\partial t} \end{bmatrix}. \tag{3.27}$$

Substituting Equation 3.25 into Equation 3.27 yields:

$$\dot{\boldsymbol{s}} = \begin{bmatrix} \frac{\partial \boldsymbol{s}_1}{\partial \boldsymbol{y}} \boldsymbol{A}\boldsymbol{y} + \sum_{i=1}^{n} \frac{\partial \boldsymbol{s}_1}{\partial \boldsymbol{y}} \boldsymbol{f}_i \\ \vdots \\ \frac{\partial \boldsymbol{s}_n}{\partial \boldsymbol{y}} \boldsymbol{A}\boldsymbol{y} + \sum_{i=1}^{n} \frac{\partial \boldsymbol{s}_n}{\partial \boldsymbol{y}} \boldsymbol{f}_i \end{bmatrix}. \tag{3.28}$$

If the RHS of Equation 3.28 is in a quadratic form, then OpInf can be directly applied to it. The definition of the new state $\boldsymbol{s}$ is problem-specific and therefore needs human interactions. We will introduce an example next. For more examples of defining new state variables, readers can refer to [87].

Example The FOM equation is:

$$\dot{\boldsymbol{y}} = \boldsymbol{A}\boldsymbol{y} + sin(\boldsymbol{y}) + \boldsymbol{B}\boldsymbol{u}, \tag{3.29}$$

where $\boldsymbol{u}$ is a vector containing input parameters. If we define

$$\boldsymbol{s} = \begin{bmatrix} \boldsymbol{s}_1 \\ \boldsymbol{s}_2 \\ \boldsymbol{s}_3 \end{bmatrix} = \begin{bmatrix} \boldsymbol{y} \\ sin(\boldsymbol{y}) \\ cos(\boldsymbol{y}) \end{bmatrix}, \tag{3.30}$$

Equation 3.29 can be transformed into:

$$\begin{aligned} \dot{\boldsymbol{s}_1} &= \boldsymbol{A}\boldsymbol{s}_1 + \boldsymbol{s}_2 + \boldsymbol{B}\boldsymbol{u}, \\ \dot{\boldsymbol{s}_2} &= \boldsymbol{s}_3(\boldsymbol{A}\boldsymbol{s}_1 + \boldsymbol{s}_2 + \boldsymbol{B}\boldsymbol{u}), \\ \dot{\boldsymbol{s}_3} &= -\boldsymbol{s}_2(\boldsymbol{A}\boldsymbol{s}_1 + \boldsymbol{s}_2 + \boldsymbol{B}\boldsymbol{u}). \end{aligned} \tag{3.31}$$

The RHS of Equation 3.31 consists of polynomial terms of $\boldsymbol{s}$ and the highest term is quadratic.

We should be informed that due to different FOMs and different choices of auxiliary variables, Equation 3.23 might be different. But it is important that the highest order for the reduced state should not be higher than quadratic. This can prevent the ROM from dimensionality explosion. Taking a cubic term as an example, if we have a cubic term in the lifted FOM governing equation, this would mean that in the reduced governing equation, we expect to have such a term:

$$\boldsymbol{x} \otimes \boldsymbol{x} \otimes \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N_r} \end{bmatrix} \otimes \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N_r} \end{bmatrix} \otimes \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N_r} \end{bmatrix}. \tag{3.32}$$

The complexity of computing such a term is $O(N_r^3)$, which can become really large even for moderate values of N_r. Such computational complexity defeats the purpose of dimensionality reduction.

3.3.3. Inferring operators

In section 3.3.2, we have introduced how to transform FOM equations whose RHS cannot be described by polynomial terms up to quadratic. After transformation, the governing equation for lifted state $\boldsymbol{s}$ is in a quadratic form.

We can construct the snapshot matrix for the lifted state $\boldsymbol{s}$:

$$\begin{aligned} \boldsymbol{S} &= [\boldsymbol{s}_{10}\ \boldsymbol{s}_{11}\ ...\ \boldsymbol{s}_{N_{\text{sim}}K}] \\ &= \begin{bmatrix} \begin{bmatrix} s_1(\boldsymbol{y}_{10}) \\ s_2(\boldsymbol{y}_{10}) \\ \vdots \\ s_n(\boldsymbol{y}_{10}) \end{bmatrix} & \begin{bmatrix} s_1(\boldsymbol{y}_{11}) \\ s_2(\boldsymbol{y}_{11}) \\ \vdots \\ s_n(\boldsymbol{y}_{11}) \end{bmatrix} & \cdots & \begin{bmatrix} s_1(\boldsymbol{y}_{N_{\text{sim}}K}) \\ s_2(\boldsymbol{y}_{N_{\text{sim}}K}) \\ \vdots \\ s_n(\boldsymbol{y}_{N_{\text{sim}}K}) \end{bmatrix} \end{bmatrix}. \end{aligned} \tag{3.33}$$

To construct the reduced space for the lifted state $\boldsymbol{s}$, we need some tricks in applying POD. Since each auxiliary variable essentially represents a different physics quantity, directly applying POD to the lifted snapshot matrix will cause a problem that only the information of those auxiliary variables whose magnitudes are large is preserved. The resulting reduced space will have large projection errors for variables whose magnitudes are relatively small. To tackle this problem, we need to construct a reduced space for each auxiliary variable individually and use all the reduced spaces to create a common space for the lifted state. We first denote:

$$\begin{aligned} \boldsymbol{S}_1 &= \begin{bmatrix} s_1(\boldsymbol{y}_{10}) & s_1(\boldsymbol{y}_{11}) & \dots & s_1(\boldsymbol{y}_{N_{\text{sim}}K}) \end{bmatrix}, \\ \boldsymbol{S}_2 &= \begin{bmatrix} s_2(\boldsymbol{y}_{10}) & s_2(\boldsymbol{y}_{11}) & \dots & s_2(\boldsymbol{y}_{N_{\text{sim}}K}) \end{bmatrix}, \\ &\vdots \\ \boldsymbol{S}_n &= \begin{bmatrix} s_n(\boldsymbol{y}_{10}) & s_n(\boldsymbol{y}_{11}) & \dots & s_n(\boldsymbol{y}_{N_{\text{sim}}K}) \end{bmatrix}, \end{aligned} \tag{3.34}$$

which are snapshot matrices for the corresponding auxiliary variables.

Then we apply POD to the snapshot matrices individually:

$$\begin{aligned} \boldsymbol{V}_1 &= \text{POD}(\boldsymbol{S}_1), \\ \boldsymbol{V}_2 &= \text{POD}(\boldsymbol{S}_2), \\ &\vdots \\ \boldsymbol{V}_n &= \text{POD}(\boldsymbol{S}_n). \end{aligned} \tag{3.35}$$

The basis for the common reduced space is:

$$\boldsymbol{V} = \begin{bmatrix} \boldsymbol{V}_1 & & & \\ & \boldsymbol{V}_2 & & \\ & & \ddots & \\ & & & \boldsymbol{V}_n \end{bmatrix}, \tag{3.36}$$

and we can use $\boldsymbol{V}$ to transform between the lifted state $\boldsymbol{s}$ and its reduced state $\boldsymbol{\omega}$:

$$\boldsymbol{s} = \boldsymbol{V}\boldsymbol{\omega}. \tag{3.37}$$

Then similarly to the datasets defined in Equation 3.18, correspondingly, we construct two datasets with $\boldsymbol{\omega}$ and a dataset with $\boldsymbol{\mu}$:

$$\begin{aligned}
\boldsymbol{\Omega}_{\text{in}} &= \begin{bmatrix} \boldsymbol{\omega}_{10} & \boldsymbol{\omega}_{11} & \dots & \boldsymbol{\omega}_{1K-1} & \dots & \boldsymbol{\omega}_{N_{\text{sim}}0} & \boldsymbol{\omega}_{N_{\text{sim}}1} & \dots & \boldsymbol{\omega}_{N_{\text{sim}}K-1} \end{bmatrix}, \\
\boldsymbol{\Omega}_{\text{out}} &= \begin{bmatrix} \boldsymbol{\omega}_{11} & \boldsymbol{\omega}_{12} & \dots & \boldsymbol{\omega}_{1K} & \dots & \boldsymbol{\omega}_{N_{\text{sim}}1} & \boldsymbol{\omega}_{N_{\text{sim}}2} & \dots & \boldsymbol{\omega}_{N_{\text{sim}}K} \end{bmatrix}, \\
\boldsymbol{U}_{\text{in}} &= \begin{bmatrix} \boldsymbol{\mu}_{10} & \boldsymbol{\mu}_{11} & \dots & \boldsymbol{\mu}_{1K-1} & \dots & \boldsymbol{\mu}_{N_{\text{sim}}0} & \boldsymbol{\mu}_{N_{\text{sim}}1} & \dots & \boldsymbol{\mu}_{N_{\text{sim}}K-1} \end{bmatrix}.
\end{aligned} \tag{3.38}$$

Substituting the datasets into the assumed quadratic ROM equation (Equation 3.23) produces:

$$\frac{\boldsymbol{\Omega}_{\text{out}} - \boldsymbol{\Omega}_{\text{in}}}{\delta t} = \boldsymbol{F}_1 \boldsymbol{\Omega}_{\text{in}} + \boldsymbol{F}_2 \boldsymbol{\Omega}_{\text{in}} \otimes \boldsymbol{\Omega}_{\text{in}} + \boldsymbol{F}_B \boldsymbol{\mathcal{U}}_{\text{in}} + \boldsymbol{F}_N \boldsymbol{\mathcal{U}}_{\text{in}} \otimes \boldsymbol{\Omega}_{\text{in}} + \boldsymbol{F}_C, \tag{3.39}$$

which is essentially the explicit Euler discretization of Equation 3.23. With the prepared snapshots, the following matrices can be constructed:

$$\begin{aligned}
\boldsymbol{R} &= \frac{\boldsymbol{\Omega}_{\text{out}}^T - \boldsymbol{\Omega}_{\text{in}}^T}{\delta t}, \\
\boldsymbol{D} &= \begin{bmatrix} \boldsymbol{\Omega}_{\text{in}}^T & \boldsymbol{\mathcal{U}}_{\text{in}}^T & \boldsymbol{I} & (\boldsymbol{\Omega}_{\text{in}} \otimes \boldsymbol{\Omega}_{\text{in}})^T & (\boldsymbol{\mathcal{U}}_{\text{in}} \otimes \boldsymbol{\Omega}_{\text{in}})^T \end{bmatrix},
\end{aligned} \tag{3.40}$$

where $\boldsymbol{I}$ is the corresponding identity matrix. Substituting Equation 3.40 into Equation 3.39 produces:

$$\boldsymbol{R} = \boldsymbol{D}\boldsymbol{O}, \tag{3.41}$$

where $\boldsymbol{O} = \begin{bmatrix} \boldsymbol{F}_1^T & \boldsymbol{F}_B^T & \boldsymbol{F}_C^T & \boldsymbol{F}_2^T & \boldsymbol{F}_N^T \end{bmatrix}$. The matrix $\boldsymbol{O}$ is an unknown term to be solved. Observing Equation 3.41, the matrix $\boldsymbol{O}$ can be obtained by solving a least-square problem:

$$\arg\min_{\boldsymbol{O}} \|\boldsymbol{D}\boldsymbol{O} - \boldsymbol{R}\|_2. \tag{3.42}$$

In reality, the matrix $\boldsymbol{O}$ from the direct solution to the least-square problem is very likely to provide an instable ROM. This phenomenon is similar to the overfitting [88] in the training of ANNs. Applying linear regularization to the original problem can remarkably prevent from overfitting. Typically, there are two kinds of linear regularization can be applied to the least-square problem. They are called Tikhonov regularization (also known as L2 regularization) [89] and Lasso method (also known as L1 regularization) [90]. The difference is that the operators computed from an L1-regularized least-square problem usually have sparsity while the solution to an L2-regularized least-square problem intends to have small values for its all entries. Although many FOM system matrices are sparse, e.g. stiffness matrix, force distribution matrix, thermal conductivity matrix, etc., their projection in the reduced space is usually dense. Therefore, using L2-regularization complies this fact better.

The L2-regularized least-square problem is:

$$\arg\min_{\boldsymbol{O}} \|\boldsymbol{D}\boldsymbol{O} - \boldsymbol{R}\|_2 + \lambda \|\boldsymbol{O}\|_2, \tag{3.43}$$

and Equation 3.43 is equivalent to a new least-square problem:

$$\arg\min_{\boldsymbol{O}} \| \begin{bmatrix} \boldsymbol{D} \\ \lambda \boldsymbol{I} \end{bmatrix} \boldsymbol{O} - \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{0} \end{bmatrix} \|_2, \tag{3.44}$$

where the parameter λ decides the strength of the penalty. Thanks to the penalty added to the L2-norm of the operators, we will not have entries with extremely large magnitudes in the inferred operators. As a result, the ROM constructed by the inferred operators is less sensitive to the inputs. The larger λ is, the greater the penalty are added to the regularization. However, an over-strong regularization can lead to a ROM completely insensitive to different input parameters. In [84], it is suggested that we can use different penalties for the linear terms and the quadratic terms in $\boldsymbol{D}$. In our practice, this indeed improves the accuracy and the stability of the inferred ROM. The improved regularization can be written as:

$$\arg\min_{\boldsymbol{O}} \| \begin{bmatrix} \boldsymbol{D} \\ \boldsymbol{\lambda} \end{bmatrix} \boldsymbol{O} - \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{0} \end{bmatrix} \|_2, \tag{3.45}$$

where:

$$\boldsymbol{\lambda} = \mathrm{diag}(\underbrace{\lambda_1, \ldots, \lambda_1}_{o_1}, \underbrace{\lambda_2, \ldots, \lambda_2}_{o_2}), \tag{3.46}$$

and o_1 and o_2 describe the numbers of columns in $\boldsymbol{D}$ belonging to the linear terms and the quadratic terms, respectively. The grid searching strategy can be employed to optimally select the parameters λ_1 and λ_2.

The algorithm for searching the optimal regularization parameters is given in Algorithm 5. Nevertheless, using L2 regularization still cannot produce a guaranteed stable OpInf-ROM. For the purely projection-based MOR methods, the preservation of the stability from the FOM to the ROM can be ensured through some special treatments [91, 92]. Unfortunately, this kind of treatments usually require the intrusiveness to the FOM and is therefore impossible for the black-box/glass-box ROM identification methods.

3.4. Summary

In this section, three different ways to construct ROMs in reduced spaces are introduced. As a hyper-reduction method, DEIM needs snapshots of FOMs and detailed FOM matrices. Besides the intrusive method, we can also identify surrogate ROMs with data generated by FOMs. As a universal approximator, MLPs can also be used to identify ROMs. However, a disadvantage of the MLP is that it can only learn the mapping between two consecutive reduced states directly is explained. To tackle

Algorithm 5 Grid searching for the optimal $\boldsymbol{\lambda}$

Require: $\lambda_{1,\min}, \lambda_{1,\max}, \lambda_{2,\min}, \lambda_{2,\max}, \boldsymbol{\Omega}_{\text{out}}, \boldsymbol{\Omega}_{\text{in}}, \boldsymbol{\mathcal{U}}_{\text{in}}$

Create a uniform grid Λ between $\lambda_{1,\min}, \lambda_{1,\max}, \lambda_{2,\min}, \lambda_{2,\max}$
Split $\boldsymbol{\Omega}_{\text{out}}, \boldsymbol{\Omega}_{\text{in}}, \boldsymbol{\mathcal{U}}_{\text{in}}$ into $\boldsymbol{\Omega}_{\text{out,train}}, \boldsymbol{\Omega}_{\text{in,train}}, \boldsymbol{\mathcal{U}}_{\text{in,train}}$ and $\boldsymbol{\Omega}_{\text{out,test}}, \boldsymbol{\Omega}_{\text{in,test}}, \boldsymbol{\mathcal{U}}_{\text{in,test}}$
$\text{loss}_{\text{test,best}} = 10^6, \lambda_{\text{optimal}} = \lambda_{\min}$
for λ_1, λ_2 in Λ **do**
    Construct $\boldsymbol{\lambda} = \text{diag}(\lambda_1, \ldots, \lambda_1, \lambda_2, \ldots, \lambda_2)$
    Solve Equation 3.45 with selected $\boldsymbol{\lambda}$ and dataset $\boldsymbol{\Omega}_{\text{out,train}}, \boldsymbol{\Omega}_{\text{in,train}}, \boldsymbol{\mathcal{U}}_{\text{in,train}}$
    Construct ROM using $\boldsymbol{O}$ solved in the last step
    Use the ROM to generate prediction $\hat{\boldsymbol{\Omega}}_{\text{out,test}}$ with input $\boldsymbol{\Omega}_{\text{in,test}}$ and $\boldsymbol{\mathcal{U}}_{\text{in,test}}$
    Compute the test error $\text{loss}_{\text{test}} = \text{MSE}(\hat{\boldsymbol{\Omega}}_{\text{out,test}}, \boldsymbol{\Omega}_{\text{out,test}})$
    if $\text{loss}_{\text{test}} < \text{loss}_{\text{test,best}}$ and ROM is bounded **then**
        $\text{loss}_{\text{test,best}} = \text{loss}_{\text{test}}$
        $\boldsymbol{\lambda}_{\text{optimal}} = \boldsymbol{\lambda}$
    end if
end for
return $\boldsymbol{\lambda}_{\text{optimal}}$

this problem, an architecture called RKNN is proposed, which learns the RHS of the latent reduced equation and integrates the learned ROM as a RK4 integrator. Besides, another non-intrusive ROM identification method, OpInf, is introduced, which requires snapshots of FOMs and some basic information of RHS of FOMs. As a fully intrusive method, DEIM will be considered as another reference in this thesis apart from FOM solutions. In the following content, we will compare it to the black-box method RKNN and the glass-box method OpInf.

4. Concept of Active Learning

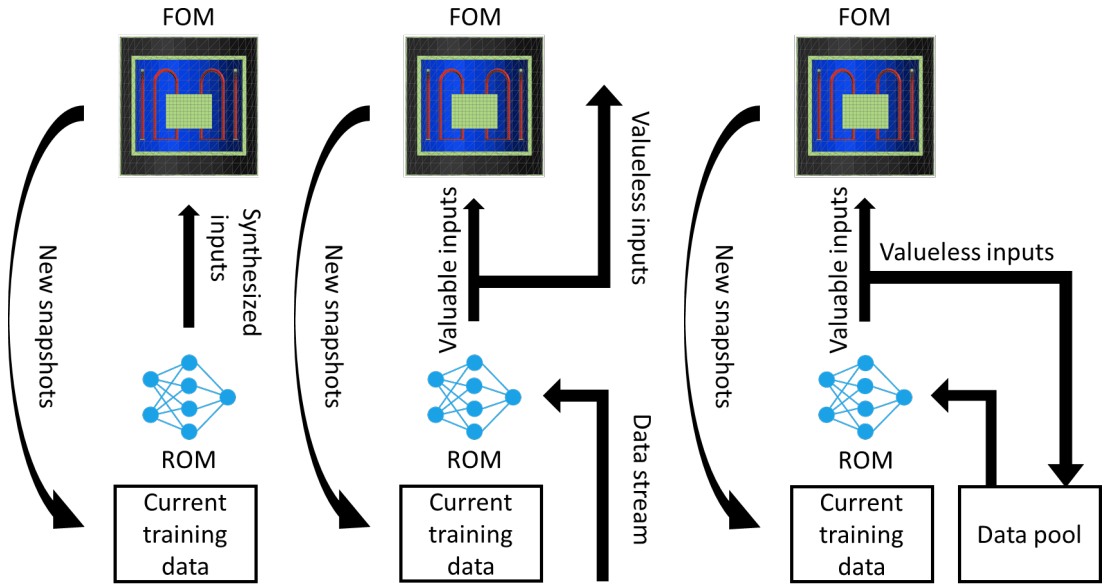
Active Learning (AL) is a family of Machine Learning (ML) methods which may query data instances (samples) to be labelled for training by an oracle (e.g., a human annotator) — can achieve higher accuracy with fewer labelled examples than passive learning [93]. In the context of Model Order Reduction (MOR), because solving a Full Order Model (FOM) is a time-consuming task, we can consider the generation of FOM snapshots as a time-consuming labelling process. The employment of AL for MOR can bring us benefits such as shorter time for Reduced Order Model (ROM) construction and a higher ROM accuracy with the same number of FOM snapshots. In [30, 94], the authors successfully employ the AL method to optimize the learned ROMs. Their work gives us strong confidence in using AL/Passive Learning (PL) to close the gap described in section 1.2.

4.1. Query strategies

Based on how an AL method queries new data, we can classify AL methods into three classes: query synthesis, stream-based selective sampling and pool-based sampling.

4.1.1. Query synthesis

The query synthesis method is proposed in [95]. The main idea of the query synthesis method is that we let the learner to synthesize new queries based on its current knowledge [96]. In our MOR context, as shown in Figure 4.1a, this would mean that we let the ROM to prepare some new inputs for the FOM solver, and the FOM solver generates the corresponding snapshots as the new training data. While preparing the new inputs, the ROM can, e.g., add some noise to the old inputs, or transform the old input by some means. The synthesis can be completely random, or based on the current status of the ROM. However, without knowing physics knowledge of the FOM, it is difficult to synthesize inputs in a meaningful way. Therefore, this method is not suitable for the research in this thesis.



(a) Query synthesis. (b) Stream-based selective sampling. (c) Pool-based sampling.

Figure 4.1.: A demonstration for different query strategies in the MOR context.

4.1.2. Stream-based selective sampling

Just like its name implies, this method designs an AL framework in a streaming fashion. As an on-the-fly interactive labelling strategy, unlabelled samples are provided in a data stream. The learner (or ROM in the language of MOR) will measure/score each sample and decide if it wants to query for labelling immediately [97]. As shown in Figure 4.1b, this will require the existence of a continuous data stream, which does not fit the scope of the thesis.

4.1.3. Pool-based sampling

Pool-based sampling is probably the most popular AL method nowadays. In this method, we prepare a data pool containing a relatively large number of unlabelled samples in such a setting. During the AL iterations, the algorithm will pick some samples from the data pool strategically and label them. The labelled samples are added to the training data. Then, the ML model is retrained/updated with the extended training data. The iterations will be carried on until the ML reaches some criteria. There are many different methods, such as [98, 99, 100], to select the unlabelled samples, but most of them are essentially based on the status of the current ML model. This process can be described as Figure 4.1c.

In the field of MOR, especially for ML-based MOR, random sampling is currently the most widely-used sampling strategy. Based on the above introduction to the methods, we know that the pool-based sampling strategy can be a good alternative for snapshot collection. However, we do not close the possibility of applying the query synthesis method and the stream-based selective sampling method to specific use cases in the MOR world.

4.2. Active selection strategies

So far, we have introduced different ways of obtaining new labelled data, and we find that the pool-based method is the most suitable one in our MOR context. As introduced in section 4.1, within the pool-based sampling method, we will choose the samples to be labelled from a data pool strategically. The selection strategy makes decisions based on sample's informativeness. Here, we will introduce two active selection strategies and a passive selection strategy.

4.2.1. Maximum Mean Square Error

In [101], this concept was proposed for the first time. The key idea of Maximum Mean Squared Error (MMSE) is to include the sample which will maximize the mean squared error of the ML model learned from the current training data:

$$\zeta^* = \underset{\zeta^*}{\operatorname{argmax}} \ \mathrm{MSE}(\mathrm{ROM}(\zeta^*), \mathrm{FOM}(\zeta^*)), \tag{4.1}$$

where ζ^* is the selected new sample. We consider such a sample is the most informative for updating the ROM. In [94], this principle is used for active data selection of the data-driven MOR. In this thesis, we will use the same principle for picking samples from the data pool.

4.2.2. Query by Committee

Besides the error based indicator, sample's informativeness can be also measured by the uncertainty of the model about that sample. A very convenient and famous way to measure the uncertainty is called Query by Committee (QBC) [102, 103, 104].

Within QBC, we will train multiple (at least 2) models and use them to predict for all the unlabelled samples, then we compute the ambiguity of the predictions. The unlabelled sample with the greatest ambiguity will be selected for labelling. In [103], the authors successfully used this method for AL for regression problems.

Uncertainty-based methods are very suitable while the ROM errors on unlabelled samples are difficult to estimate. However, in the original QBC method, we need either to train multiple ROMs to compute uncertainty or to use a ROM which has a natural uncertainty indicator, e.g., a Gaussian-Process-Regression-based ROM. In this thesis, we will introduce a workaround to resolve the limitations in section 6.4.1.3.

Besides the active selection methods introduced in section 4.2, methods such as mismatch-first farthest-traversal [105], maximizing expected model change [106] and maximizing expected error reduction [107], can be good alternative strategies. Here we encourage readers to investigate different selection strategies.

4.3. Batch query for active selection

With the methods introduced in section 4.2.1 and section 4.2.2, one can easily find the most informative sample from the data pool. However, such a serial mode (one sample at a time) is unrealistic in our MOR context. The offline phase will be slowed down as time required for model identification is considerable. Therefore, we have a strong demand of a query-in-batch mode.

In such a mode, a batch of unlabelled samples will be selected from the data pool. After being labelled, multiple new samples will be added to the training data, and the extended training data will be used to refine the ML model. A naive approach to constructing such a batch is to simply assess all the unlabelled samples, and select the most informative ones by some measure. Unfortunately this is a myopic strategy, and generally does not work well since it does not consider the redundant information content among the informative samples. In other words, the best two queries might be so highly ranked because they are virtually identical (similar), in which case labeling both is probably wasted effort [93].

To deal with such a problem, different methods have been proposed [108, 109, 110, 111, 112]. In [112], a method based on mini-batch AL considering both the informativeness and the diversity of selected samples is proposed. It is easy to combine it with different measurement of informativeness. In this thesis, we will introduce this Diverse mini-Batch Active Learning (DBAL) method and employ it.

Within the DBAL method, the informativeness of unlabelled samples will be measured first. The measurement can be error-based, model-change-based, model-uncertainty-based, etc. Then samples will be sorted in a descending order of their informativeness. Then, pre-filtering will be performed to select the top $b \cdot n$ samples from the sorted samples, where b is called pre-filter factor, and n is the desired number of new samples. Afterwards, the pre-selected samples will be classified into n clusters. The classification can be done with, for example, K Means clustering [113]. Samples in the same cluster can be considered to have strong similarity. Therefore,

Algorithm 6 Diverse mini-Batch Active Learning

Require: Number of new samples n, data pool of unlabelled samples D, pre-filter factor b

1: Compute the informativeness of all samples in D
2: Pre-filter to the top $b \cdot n$ samples based on the computed informativeness
3: Classify the $b \cdot n$ samples into n clusters
4: For each cluster, pick the unlabelled sample closest to the centroid of the cluster
5: Label the selected n samples
6: **return** The labelled samples

we will only pick one sample from each cluster. The picked samples will be the ones closest to their clusters' centroid, which can be considered the most representative samples in their clusters.

4.4. Passive selection strategy (Passive Learning)

Passive selection is also called PL. Within this method, we do not check the status of the learned ML model. Instead, we first investigate unlabelled samples' geometric characteristics in the feature space [114]. Then, we pick the samples which are considered the most informative based on the investigation. The geometric characteristics usually refer to the distance between samples.

Let us assume in the sample space we have a vacuum space. Since lack of information in this area, the ROM is very likely to be uncertain while predicting for a new sample coming from the vacuum space. Moreover, the uncertainty is expected to be maximized when the new sample is from the position which is the most distant to the current training samples. We consider such a sample is the most informative to the current ROM. This can be concluded as:

$$\zeta^* = \underset{\zeta^*}{\operatorname{argmax}} \ \operatorname{dist}(\zeta^*, \{\zeta_1, \zeta_2, \zeta_3, \dots\}), \tag{4.2}$$

where ζ_i is the i-th training sample in the current training data. There are different ways of defining the distance function $\operatorname{dist}(\cdot)$. By properly defining the distance function $\operatorname{dist}(\cdot)$, passive selection can naturally enable batch selection without having redundant information in the selected samples. We will introduce our definition later in section 6.4.2.

4.5. Summary

In this section, the concept of AL is introduced. We chose the combination of pool-based sampling and the selection strategies based on MMSE, QBC and PL as the sample selection methods for Active-Learning-based Model Order Reduction (AL-MOR). To make the selection strategies more practical in the MOR context, a mini-batch AL method called DBAL is introduced. The DBAL method takes not only the informativeness but also the diversity of unlabelled samples into consideration, which can reduce the redundant information in selected unlabelled samples. Using the AL methods introduced in this section, we can accelerate the procedure of constructing ML-based ROMs described in section 3.2 and section 3.3.

Part III.

Active-Learning-based Model Order Reduction

5. ROM Size Determination

As introduced in section 2.3.4, while applying truncated Singular Value Decomposition (SVD), we can freely choose the position of truncation. The size of the Reduced Order Model (ROM) is decided by how many singular vectors are included in the reduced basis $\boldsymbol{V}$. The ROM size directly influences two important performance indicators of the ROM: accuracy and efficiency. On the one hand, a small ROM results in a large projection error, and the physics encoded in the reduced snapshots will be distorted, which makes ROM identification also difficult. On the other hand, identifying the ROM in an unnecessarily big reduced space will require more training data, and the resulting ROM can take a long time to solve during the online phase. Therefore, an appropriate size for the ROM will be important. Naturally, a question is raised here: what is the best size for a ROM? In this chapter, an automatic way of determining the optimal ROM size will be introduced. Based on it, we propose a more reliable algorithm for choosing the ROM size.

5.1. ROM size determination based on singular value scree plot

In [115, 116, 117, 118, 39], the authors proposed and introduced some methods to determine the optimal number of reduced basis vectors, such as Eigenvalue-greater-than-one Rule [117], Scree Test [119, 39], Parallel Analysis [118], etc. In [39], an automatic decision is made by using profile likelihood on scree plot.

Scree plot is a figure reflecting the trend of singular values of the snapshot matrix calculated by SVD. In Figure 5.1, we present an example of the scree plot, where we can see a descending curve. As we know, each singular value corresponds to a Proper Orthogonal Decomposition (POD) component. The larger the singular value is, the more energy/information is contained in its corresponding POD component and the more important the corresponding POD component is. Therefore, we consider the position of a 'gap' or 'elbow' displayed in the scree plot as the best position for truncation. Based on this consideration and to automatically detect this position, the method Maximize Profile Likelihood is proposed in [39].

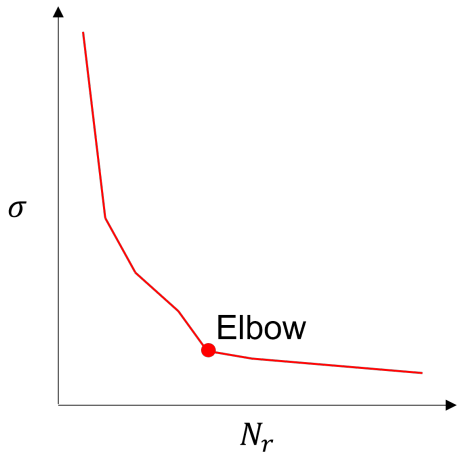


Figure 5.1.: A scree plot.

Let us assume that we extract c POD components as candidates:

$$\boldsymbol{V}_{\text{cand}} = \begin{bmatrix} \boldsymbol{v}_1 & \boldsymbol{v}_2 & \dots & \boldsymbol{v}_c \end{bmatrix}, \tag{5.1}$$

and they are sorted with the corresponding singular values ς in a descending order:

$$\Sigma_{\text{cand}} = \begin{bmatrix} \varsigma_1 & \varsigma_2 & \dots & \varsigma_c \end{bmatrix}. \tag{5.2}$$

We further assume that $\boldsymbol{V}_{\text{all}}$ is truncated at (included) the q-th column, with $1 \leq q \leq c$, which will split our singular value set Σ_{cand} into to two parts as well:

$$\begin{aligned} \Sigma_1 &= \{\varsigma_1, \varsigma_2, \dots, \varsigma_q\}, \\ \Sigma_2 &= \{\varsigma_{q+1}, \varsigma_{q+2}, \dots, \varsigma_c\}. \end{aligned} \tag{5.3}$$

If the i-th column is the right position for separation, we consider Σ_1 and Σ_2 are sampled from two different distributions. For sake of simplicity, we assume they are sampled from two different normal distributions [120], denoted as:

$$\begin{aligned} f_1(\varsigma|\mu_1, \sigma) &= \frac{1}{\sigma\sqrt{2\pi}} exp\left(-\frac{(\varsigma - \mu_1)^2}{2\sigma^2}\right), \\ f_2(\varsigma|\mu_2, \sigma) &= \frac{1}{\sigma\sqrt{2\pi}} exp\left(-\frac{(\varsigma - \mu_2)^2}{2\sigma^2}\right), \end{aligned} \tag{5.4}$$

where:

$$\begin{aligned} \mu_1 &= \text{mean}(\Sigma_1), \\ \mu_2 &= \text{mean}(\Sigma_2), \end{aligned} \tag{5.5}$$

and:

$$\sigma^2 = \frac{(q-1)\cdot \mathrm{var}(\Sigma_1) + (m-q-1)\cdot \mathrm{var}(\Sigma_2)}{m-2} \tag{5.6}$$

It might be noticed that in Equation 5.4, we use the same σ for two distributions. If we establish the distributions with the variances computed from the corresponding error sets, Equation 5.4 will have infinite values while $q = 1$ or $q = m - 1$. Then we can use Equation 5.4 to build the likelihood function:

$$l(q) = \sum_{i=1}^{q} log(f_1(\varsigma_i|\mu_1(q), \sigma)) + \sum_{j=q+1}^{m} log(f_2(\varsigma_j|\mu_2(q), \sigma)). \tag{5.7}$$

The q maximizing Equation 5.7 then is selected as the position for truncating $\boldsymbol{V}_{\mathrm{cand}}$.

5.2. Improved ROM size determination based on projection error

The ROM size selection method introduced previously uses the scree plot of singular values to decide the truncation position. A ROM constructed in such a way can be considered to be the most efficient in information preservation. However, besides the efficiency, the ROM accuracy is another performance indicator that is very important for application. Here, we propose an improved version of the selection strategy based on projection error curves. Firstly, we define the mean projection error $e_{\mathrm{proj}}^{\mathrm{mean}}$ of Model Order Reduction (MOR) as:

$$e_{\mathrm{proj}}^{\mathrm{mean}}(\boldsymbol{V}, \boldsymbol{Y}) = \mathrm{mean}\left(\frac{|\boldsymbol{V}\boldsymbol{V}^T\boldsymbol{Y} - \boldsymbol{Y}|}{|\boldsymbol{Y}|}\right), \tag{5.8}$$

and the maximum projection error $e_{\mathrm{proj}}^{\mathrm{max}}$ of MOR as:

$$e_{\mathrm{proj}}^{\mathrm{max}}(\boldsymbol{V}, \boldsymbol{Y}) = \max\left(\frac{|\boldsymbol{V}\boldsymbol{V}^T\boldsymbol{Y} - \boldsymbol{Y}|}{|\boldsymbol{Y}|}\right), \tag{5.9}$$

where $\boldsymbol{Y}$ is the snapshot matrix used to construct the reduced basis $\boldsymbol{V}$. These two errors can clearly quantify the quality of the reconstruction of the snapshots using the reduced basis.

Then we can create the curves for the mean and maximum projection error. Without loss of generality, in Figure 5.2 we use the curve of $e_{\mathrm{proj}}^{\mathrm{mean}}$ as an example.

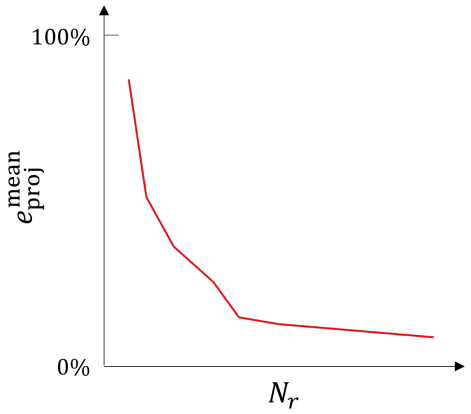


Figure 5.2.: An example for the curve of $e_{\text{proj}}^{\text{mean}}$.

Our target is the same: finding the gap or elbow existing in the error curves. We first build two sets for $e_{\text{proj}}^{\text{mean}}$ and $e_{\text{proj}}^{\text{max}}$ respectively:

$$\begin{aligned} E_{\text{proj}}^{\text{mean}} &= \left\{ e_{\text{proj},1}^{\text{mean}}, e_{\text{proj},2}^{\text{mean}}, \ldots, e_{\text{proj},c}^{\text{mean}} \right\}, \\ E_{\text{proj}}^{\text{max}} &= \left\{ e_{\text{proj},1}^{\text{max}}, e_{\text{proj},2}^{\text{max}}, \ldots, e_{\text{proj},c}^{\text{max}}, \right\}, \end{aligned} \tag{5.10}$$

where:

$$\begin{aligned} e_{\text{proj},i}^{\text{mean}} &= e_{\text{proj}}^{\text{mean}}(\boldsymbol{V}_i, \boldsymbol{Y}), \\ e_{\text{proj},i}^{\text{max}} &= e_{\text{proj}}^{\text{max}}(\boldsymbol{V}_i, \boldsymbol{Y}), \end{aligned} \tag{5.11}$$

with $\boldsymbol{V}_i = \boldsymbol{V}_{\text{all}}[:, : i]$. Here $\boldsymbol{V}_{\text{all}}[:, : i]$ means extracting the first i columns of $\boldsymbol{V}_{\text{all}}$. Since later $E_{\text{proj}}^{\text{mean}}$ and $E_{\text{proj}}^{\text{max}}$ will be treated equally, we use E to represent the both sets and e to represent the errors stored in the sets. We further assume that $\boldsymbol{V}_{\text{all}}$ is truncated at (included) the q-th column, with $1 \leq q \leq c$, this will split our error set E into to two sets:

$$\begin{aligned} E_1 &= \{e_1, e_2, \ldots, e_q\}, \\ E_2 &= \{e_{q+1}, e_{q+2}, \ldots, e_c\}. \end{aligned} \tag{5.12}$$

Similarly, we consider the two sets are sampled from two different distributions, which are denoted as:

$$\begin{aligned} f_1(e|\mu_1, \sigma) &= \frac{1}{\sigma\sqrt{2\pi}} exp\left(-\frac{(e-\mu_1)^2}{2\sigma^2} \right), \\ f_2(e|\mu_2, \sigma) &= \frac{1}{\sigma\sqrt{2\pi}} exp\left(-\frac{(e-\mu_2)^2}{2\sigma^2} \right), \end{aligned} \tag{5.13}$$

where:

$$\begin{aligned} \mu_1 &= \text{mean}(E_1), \\ \mu_2 &= \text{mean}(E_2), \end{aligned} \tag{5.14}$$

and:

$$\sigma^2 = \frac{(q-1)\cdot \text{var}(E_1) + (m-q-1)\cdot \text{var}(E_2)}{m-2}. \tag{5.15}$$

The final likelihood function is:

$$l(q) = \sum_{i=1}^{q} log(f_1(e_i|\mu_1(q),\sigma)) + \sum_{j=q+1}^{m} log(f_2(e_j|\mu_2(q),\sigma)). \tag{5.16}$$

Comparing the improved method with the original method, the most important change is the replacement of the singular value scree plot with the projection error curves. Such error curves contain both the information of the ROM efficiency as well as the information of the ROM accuracy. This two information will be the ingredients for the ROM-size-determination algorithm.

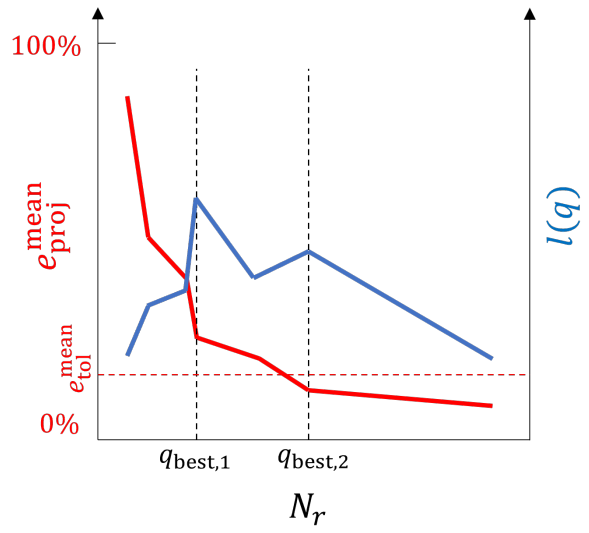


Figure 5.3.: An example for the curve of the mean projection error (red) and the curve (blue) of the corresponding profile likelihood. Two local maximums are found in the blue curve, while the ROM size $q_{\text{best},2}$ corresponding to the second local maximum has a mean projection error smaller than $e^{\text{mean}}_{\text{tol}}$ and should therefore be selected as the optimal ROM size.

Although by maximizing Equation 5.16 we can find the position of the elbow, so far we only consider for the efficiency of the ROM. To be specific, a ROM constructed

with such a size is very efficient in preserving information, but we have not thought about if the preserved information is sufficiently accurate for describing our Full Order Model (FOM). Therefore, we use the errors computed in Equation 5.12 to quantify the projection quality resulted from different truncation positions.

The algorithm selecting the optimal ROM size under given error conditions is presented in Algorithm 7. The concept behind Algorithm 7 can be explained by Figure 5.3. We again use $e^{\text{mean}}_{\text{proj},i}$ as an example. A tolerance for the projection error $e^{\text{mean}}_{\text{proj},i}$ must be defined in advance. In the algorithm, we will calculate $l(q)$ and probably we can observe one global maximum as well as some local maximums for $l(q)$. In Figure 5.3 we see two peaks in the curve of $l(q)$, and they are denoted as $q_{\text{best},1}$ and $q_{\text{best},2}$. If we select $q_{\text{best},1}$ as the ROM size, we will have the global maximum for $l(q)$, which means this is the most efficient ROM. However, a ROM with size $q_{\text{best},1}$ will have a projection error higher than our tolerance. Therefore, if we take the ROM accuracy into consideration, we should choose the local maximum where ROM size is $q_{\text{best},2}$. Algorithm 7 completely respects the principle we mentioned in section 1.2 that the ROM quality always goes first. Under this condition, we try to minimize the size of the ROM to make it more efficient while serving in the online phase.

Algorithm 7 ROM size selection

Require: $\boldsymbol{Y}, m, e_{\text{tol}}^{\text{mean}}, e_{\text{tol}}^{\text{max}}$

Apply POD to $\boldsymbol{Y}$ to get $\boldsymbol{V}_{\text{all}} = \begin{bmatrix} \boldsymbol{v}_1 & \boldsymbol{v}_2 & \dots & \boldsymbol{v}_c \end{bmatrix}$
Construct $E_{\text{proj}}^{\text{mean}}$ and $E_{\text{proj}}^{\text{max}}$
$l_{\text{mean}}^{\text{max}} = l_{\text{max}}^{\text{max}} = -\infty$, $q_{\text{mean}}^{\text{best}} = q_{\text{max}}^{\text{best}} = 1$
for $q \leq m$ **do**
    Split $E_{\text{proj}}^{\text{mean}}$ into $E_1^{\text{mean}}, E_2^{\text{mean}}$
    Compute $l^{\text{mean}}(q)$ and $e_{\text{proj},q}^{\text{mean}}$
    if $l^{\text{mean}}(q) > l_{\text{max}}^{\text{mean}}$ and $e_{\text{proj},q}^{\text{mean}} \leq e_{\text{tol}}^{\text{mean}}$ **then**
        $l_{\text{max}}^{\text{mean}} = l^{\text{mean}}(q)$
        $q_{\text{best}}^{\text{mean}} = q$
    end if
end for
for $q \leq m$ **do**
    Split $E_{\text{proj}}^{\text{max}}$ into $E_1^{\text{max}}, E_2^{\text{max}}$
    Compute $l^{\text{max}}(q)$ and $e_{\text{proj},q}^{\text{max}}$
    if $l^{\text{max}}(q) > l_{\text{max}}^{\text{max}}$ and $e_{\text{proj},q}^{\text{max}} \leq e_{\text{tol}}^{\text{max}}$ **then**
        $l_{\text{max}}^{\text{max}} = l^{\text{max}}(q)$
        $q_{\text{best}}^{\text{max}} = q$
    end if
end for
return $q_{\text{best}} = \max(q_{\text{best}}^{\text{mean}}, q_{\text{best}}^{\text{max}})$

6. Advanced Strategies for Snapshot Collection

Data sampling is a crucial step for non-intrusive Model Order Reduction (MOR). Without sufficient access to full order systems, data/output produced by Full Order Models (FOMs) is the only key with which we can identify latent dynamics. In general, the more data available, the more accurate the knowledge discovered from data is. However, in the field of MOR, collecting FOM data is computationally expensive, which must need solutions of full order problems and cost a long time in the offline phase. Besides, saving those high-fidelity solutions in devices and loading them to computer memory also have an intensive requirement for computational resources. Therefore, the decision of how much data are actually needed is the key to saving time, disk space, and computer memory, which can eventually make MOR's application more practical. In this thesis, an Active Learning (AL) algorithm for constructing a Reduced Order Model (ROM) is proposed, where the training data will be continuously extended until the ROM meets a predefined accuracy.

The algorithm is designed based on the pool-based AL method. The algorithm needs two necessary ingredients: a data pool and a strategy to select data from the data pool. In this chapter, we will propose different ways to prepare them. We start by introducing the approach of preparing the data pool, and then we propose different selection strategies using the methods introduced in section 4.2 and section 4.4.

6.1. Dynamic Parameter Sampling

The shortcoming of using the Static Parameter Sampling (SPS) method introduced in section 2.3.3 is apparent: the sample diversity in the parameter space is restricted by the number of FOM simulations. This limitation may not be observed when the parameter space is low-dimensional. However, when the FOM has a high-dimensional parameter space, the data-driven ROM requires prohibitive amounts of data to achieve an acceptable accuracy.

A sampling method called Dynamic Parameter Sampling (DPS) is proposed to tackle this problem. Similarly, DPS begins with taking samples randomly in the parameter space $\mathcal{M}$. However, the amount of samples is not matching the number of

FOM simulations N_{sim} but equal to $N_{\text{sim}} \cdot (K+1)$. These samples will form a set:

$$M_{\text{DPS}} = \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, ..., \boldsymbol{\mu}_{N_s}\}. \tag{6.1}$$

We can randomly split them into N_{sim} subsets:

$$\begin{aligned} M_{\text{DPS}}^1 =& \{\boldsymbol{\mu}_{10}, \boldsymbol{\mu}_{11}, ..., \boldsymbol{\mu}_{1K}\}, \\ M_{\text{DPS}}^2 =& \{\boldsymbol{\mu}_{20}, \boldsymbol{\mu}_{21}, ..., \boldsymbol{\mu}_{2K}\}, \\ &\vdots \\ M_{\text{DPS}}^{N_{\text{sim}}} =& \{\boldsymbol{\mu}_{N_{\text{sim}}0}, \boldsymbol{\mu}_{N_{\text{sim}}1}, ..., \boldsymbol{\mu}_{N_{\text{sim}}K}\}, \end{aligned} \tag{6.2}$$

Then, we use these subsets to create N_{sim} Initial Value Problem (IVP)s. For the i-th FOM simulation, we use the parameter samples in M_{DPS}^i to create the time-dependent parameter function $\boldsymbol{\mu}^{(i)}(t)$:

$$\boldsymbol{\mu}_i(t) = \textbf{Interp}(\bar{t}; M_{\text{DPS}}^i). \tag{6.3}$$

In Equation 6.3:

$$\bar{t} = \{t_0, t_1, t_2, \ldots, t_{\text{end}}\} \tag{6.4}$$

is the time grid for integration, and $\textbf{Interp}(\bar{t}; M_{\text{DPS}}^i)$ means interpolating based on the instances in $\bar{t}$ and M_{DPS}^i. Compared to SPS, DPS makes full use of all time steps in transient problems. The parameter sample density of DPS is $K+1$ times as high as SPS's.

However, DPS also has its own disadvantages. The most crucial one is the incomplete observation of system states. Thinking about why we use different parameter samples for taking snapshots, the reason is to explore the parameter space more efficiently. This can be translated to having a better parameter diversity in training data. Apart from this, different parameters will also make the system evolve differently. The snapshots taken during different evolution of the system can provide a relatively complete observation of system states. Training data containing adequate state samples and parameter samples are particularly important for purely data-driven MOR, i.e. Artificial Neural Network (ANN)-based approaches in this thesis. However, so far, we only pay attention to the parameter diversity in training data, and the diversity of sampled states is ignored.

Now, let us analysis the diversity of system states sampled by DPS. We can consider a FOM established by DPS as a dynamic system whose input is a stochastic process:

$$\dot{\boldsymbol{y}}(t) = \boldsymbol{f}(\boldsymbol{y}(t); \boldsymbol{\mu}_{\text{random}}(t)), \tag{6.5}$$

where $\boldsymbol{\mu}_{\text{random}}(t)$ is a random function of time. In [121], it is proven that for such a dynamic system, the mean value of system output is equal to the system output

produced by sending the mean value of the stochastic process to the system. Moreover, the variance of the output is determined by properties of the system and the stochastic process. This means for some FOMs whose output variance is small, DPS samples can be closely distributed around the mean-output trajectory.

Example We create a 1-D IVP whose governing equation is:

$$\frac{dy}{dt} = k(t)y + e^{a(t)}, \tag{6.6}$$

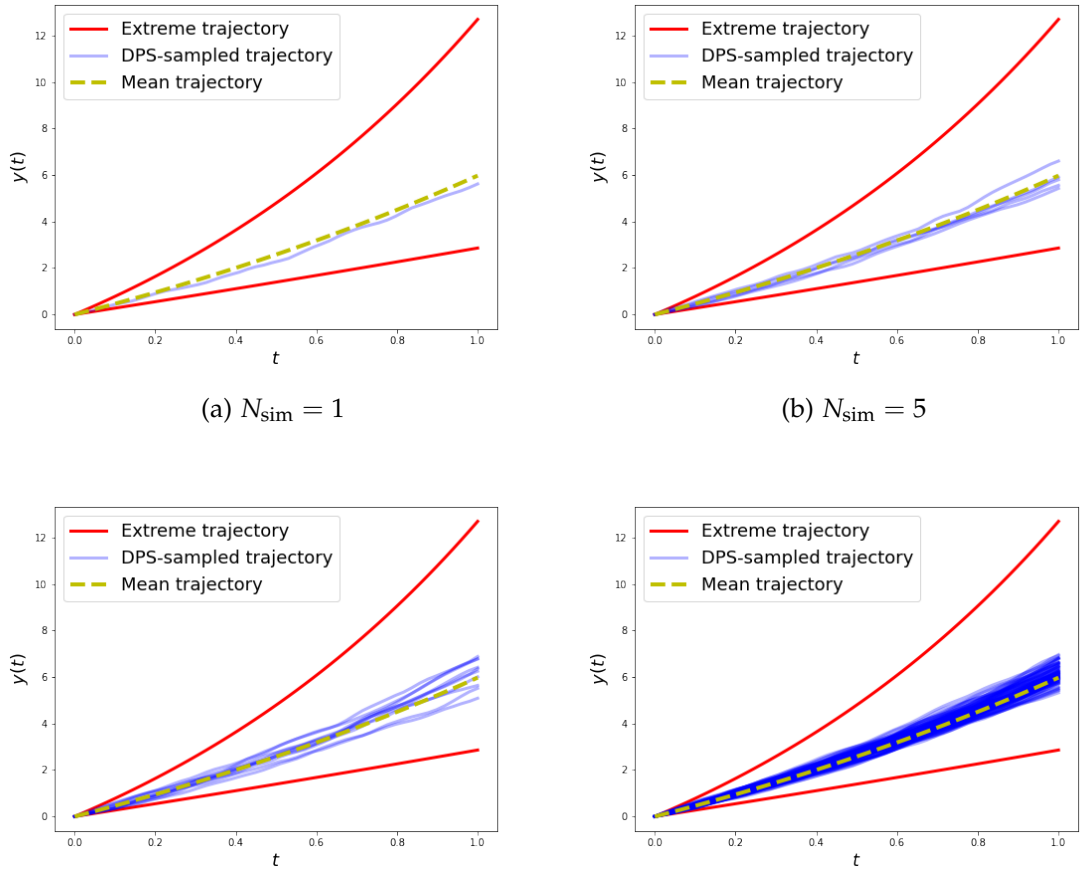
where $t \in (0, 1]$. We prescribe the parameter range as $k(t) \in [0.1, 1]$ and $a(t) \in [1, 2]$. We first integrate two IVPs configured by two parameter settings with $\delta t = 0.02$:

$$\begin{aligned} \frac{dy}{dt} &= k_{\text{min}}y + e^{a_{\text{min}}}; \\ \frac{dy}{dt} &= k_{\text{max}}y + e^{a_{\text{max}}}, \end{aligned} \tag{6.7}$$

where $k_{\text{max}} = 1, k_{\text{min}} = 0.1$ and $a_{\text{max}} = 2, a_{\text{min}} = 1$. Their solutions will form two trajectories, as the red curves shown in Figure 6.1, which are called extreme trajectories. The space between two extreme trajectories can be considered as the solution space for this parameterized system. Then we use another parameters $k_{\text{mean}} = 0.55, a_{\text{mean}} = 1.5$ to get another curve, which is yellow dashed in Figure 6.1. We call this trajectory mean trajectory. We use DPS to create $N_{\text{sim}} = 1, 5, 10, 100$ IVPs and get their corresponding solutions. The results are shown in Figure 6.1a, Figure 6.1b, Figure 6.1c and Figure 6.1d, respectively. As we see, the DPS trajectories are distributed around the mean trajectory and form a band. This exactly matches the point mentioned in [121]. Additionally, we also observe that the width of this band is growing with increasing N_{sim}. However, the speed of the growth is not constant. At the beginning, when we increase N_{sim} from 1 to 5, the width grows fast. The same trend is observed when increasing N_{sim} from 5 to 10. But when we directly increase N_{sim} from 10 to 100, we can see the growth of the band width is actually hardly visible. The additional DPS trajectories are just filling the empty space in the band where $N_{\text{sim}} = 10$. This implies that if we want to use DPS trajectories to fill the whole solution space between two extreme trajectories, we probably need to construct many different IVPs with many different parameter settings, which is very inefficient and non-practical.

6.2. Joint Space Sampling

As discussed in section 6.1, DPS-sampled states have a bad distribution, and the knowledge learned from such training data is consider lacking generalization. Com-



(a) $N_{\text{sim}} = 1$

(b) $N_{\text{sim}} = 5$

(c) $N_{\text{sim}} = 10$

(d) $N_{\text{sim}} = 100$

Figure 6.1.: An example showing the disadvantage of using DPS for sampling the snapshots. The red curves are produced by the IVPs with parameters: $k(t) \equiv 1,\ a(t) \equiv 2$ (upper) and $k(t) \equiv 0.1,\ a(t) \equiv 1$ (lower). The yellow dashed curve is produced by the IVP with parameters: $k(t) \equiv 0.55,\ a(t) \equiv 1.5$. The blue curves are sampled by DPS. The width of the band formed by the blue trajectories grows fast from $N_{\text{sim}} = 1$ to $N_{\text{sim}} = 10$ but reaches the saturation point at $N_{\text{sim}} = 10$. The growth then becomes invisible from $N_{\text{sim}} = 10$ to $N_{\text{sim}} = 100$.

pared to the DPS-sampled data, samples collected using SPS have a better distribution for the reduced state. However, since SPS uses only one parameter sample for each FOM simulation, its exploration in the parameter space is not sufficient. Unless a large number of FOM simulations are performed, the ROM can only observe little diversity for system parameters in training data. In this point of view, neither of them can be considered a good strategy for generating training data for the ROM.

A new approach for collecting snapshots is proposed in this thesis to overcome the disadvantages and named Joint Space Sampling (JSS). Just as its name implies, we generate many samples in a joint space consisting of the parameter space and an estimated reduced solution space. By taking joint samples, we will create a data pool consisting of a large number of initial states and system parameters for the full order problem.

However, unlike system parameters which already have a predefined sampling space, we have not defined such a space for the system state. So in this section, we will introduce how to create an estimated space and to sample initial states in the estimated space.

6.2.1. Reduced solution space estimation

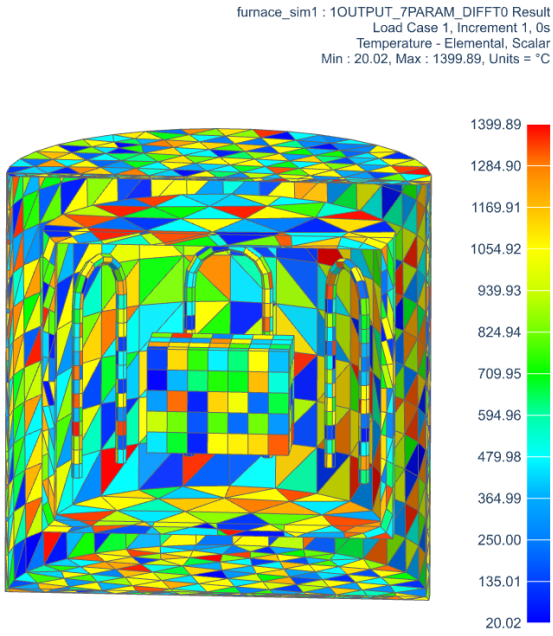
The most straightforward solution to sample an initial state is that we randomly generate values for each degree of freedom (DOF) in the FOM and use the resulting system state as the initial condition for the simulation. However, the initial system state constructed in this approach is considered as a non-physics system state, because such a sampling approach completely ignores the continuity of physics in the FOM.

We notice that during the process of Proper Orthogonal Decomposition (POD), the most correlative DOFs in the FOM are described by the same POD component in the reduced space [122]. This inspires us that sampling randomly in reduced space first and then lifting the sampled reduced state back to the full space can produce a FOM state where correlative DOFs are assigned with continuous physics quantities.

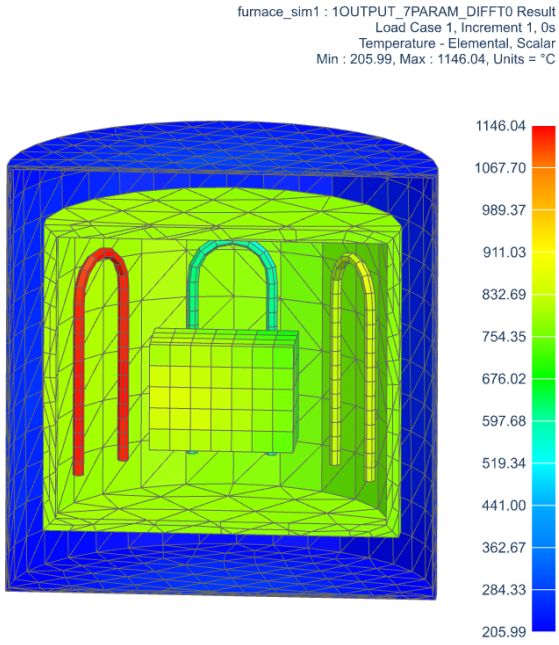
In Figure 6.2, a vacuum furnace model is given as an example. In Figure 6.2a, a FOM state is randomly sampled. In this case, the temperature field is discontinuous across the model. This kind of FOM state is non-physics and can be hardly observed in reality. In Figure 6.2b, we first construct a reduced space for the FOM. Then we generate a random reduced state in the reduced space. Finally, we lift the sampled reduced state back to the full space. The resulting FOM state has a continuous temperature field and makes sense physically.

To allow sampling reduced states randomly, we will construct a new space called reduced solution space. The first step is generating a hyper-cubic space.

1. Define the parameter space $\mathcal{M}$ for the problem based on potential use cases.



(a) A FOM state randomly sampled in the full space.



(b) A FOM state lifted from a reduced state randomly sampled in the reduced space.

Figure 6.2.: Comparison between the FOM states sampled in different ways.

2. Perform SPS with m parameter samples collected from the space $\mathcal{M}$, and the resulting snapshot matrix is denoted as $\boldsymbol{Y}_{\text{estimate}}$.

3. Apply POD to $\boldsymbol{Y}_{\text{estimate}}$, and project it onto the reduced space to get the reduced basis $\boldsymbol{V}$ and $\boldsymbol{X}_{\text{estimate}}$.

4. Find out the minimum and maximum entry in $\boldsymbol{X}_{\text{estimate}}$ for each POD component, and we denote them as:

$$\{x_{1,\min}, x_{1,\max}, x_{2,\min}, x_{2,\max}, ..., x_{N_r,\min}, x_{N_r,\max}\}. \tag{6.8}$$

5. A rough estimate for the reduced state space is:

$$\mathcal{X} = [x_{1,\min}, x_{1,\max}] \times [x_{2,\min}, x_{2,\max}] \times \cdots \times [x_{N_r,\min}, x_{N_r,\max}], \tag{6.9}$$

   and we get a joint space $\mathcal{J} = \mathcal{M} \times \mathcal{X}$

A vectoral sample taken from $\mathcal{J}$ takes the form of:

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{x} \end{bmatrix}, \tag{6.10}$$

where a reduced state and a group of system parameters are sampled simultaneously.

6.2.2. Loosening and trimming the reduced solution space

However, the reduced solution space $\mathcal{X}$ described by Equation 6.9 is just a rough estimate. We need to loosen and trim it to improve the estimation. The first step is loosening. For this purpose, we introduce an expansion ratio β and compute the distance Δx_i between the current upper and lower limit:

$$\Delta x_i = x_{i,\max} - x_{i,\min}. \tag{6.11}$$

Then we modify the original upper and lower limit by:

$$\begin{aligned} \check{x}_{i,\max} &= x_{i,\max} + \beta \Delta x_i, \\ \check{x}_{i,\min} &= x_{i,\min} - \beta \Delta x_i. \end{aligned} \tag{6.12}$$

We use $\check{x}_{i,\max}$ and $\check{x}_{i,\min}$ as the new upper and lower limit. By doing this, the upper and lower limit of each POD component is shifted on purpose. We need to loosen the reduced solution space because when we use parameter samples from $\mathcal{M}$ to collect the snapshots $\boldsymbol{Y}_{\text{estimate}}$ with SPS, we cannot ensure that the boundary (extreme) trajectories are included in $\boldsymbol{Y}_{\text{estimate}}$. In this case, we will underestimate/overestimate

the upper/lower limit of the true reduced solution space. Finally, we denote the loosened reduced solution space as $\mathcal{X}^*$.

The next step will be trimming the space. Now the space $\mathcal{X}^*$ is still defined as a hyper-cubic space. The projection of a reduced state $\boldsymbol{x}$ directly sampled from $\mathcal{X}^*$ in the full space can still be meaningless to the full order problem. Therefore, we need to add two extra inequalities to trim the hyper-cubic space $\mathcal{X}^*$:

$$\begin{cases} \mathrm{MAX}(\boldsymbol{V}\boldsymbol{x}) < y_{\max} \\ \mathrm{MIN}(\boldsymbol{V}\boldsymbol{x}) > y_{\min} \end{cases}, \tag{6.13}$$

where $\mathrm{MIN}(\cdot)$ finds the maximum entry in a matrix. The value $y_{\max}$ and $y_{\min}$ can be decided by different approaches, for example:

$$\begin{aligned} y_{\max} &= \mathrm{MAX}(\boldsymbol{Y}_{\text{estimate}}), \\ y_{\min} &= \mathrm{MIN}(\boldsymbol{Y}_{\text{estimate}}). \end{aligned} \tag{6.14}$$

Alternatively, $y_{\max}$ and $y_{\min}$ can be defined by empirical values, for example, thinking of a thermal system, then $y_{\min}$ can be the room temperature while $y_{\max}$ can be the highest temperature recorded in the system's history data. The step of trimming the reduced solution space is also very important. Without trimming step, the full states lifted from sampled reduced states can have entries which are too extreme for potential use cases.

As shown in Figure 6.3, before loosening and trimming, the joint space $\mathcal{J}$ is a hyper-cubic (or cubic) space. After loosening and trimming, it becomes a polyhedral space. The polyhedral space is produced by the loosened hyper-cubic space $\mathcal{J}^* = \mathcal{M} \times \mathcal{X}^*$ and the trimming inequalities.

6.2.3. One-step snapshot

With a joint sample $\boldsymbol{J}$ and the reduced basis $\boldsymbol{V}$ on our hands, we have an initial state and a group of system parameters needed by the FOM solver. In Algorithm 8, we present how to sample a set of joint samples. If we have enough joint samples, and they are uniformly distributed in the joint space, we only need the FOM solver to integrate for one time step δt in the time span with corresponding parameters. We denote this process as:

$$\boldsymbol{y} = \mathrm{FOM}\left(\boldsymbol{J}, \boldsymbol{V}, \delta t\right), \tag{6.15}$$

and we call a snapshot obtained in this way a one-step snapshot.

To explain why we only collect the one-step snapshot, we use the the same 1-D system as in section 6.1 as an example. We demonstrate the collected one-step snapshots in Figure 6.5. As we can see, thanks to the trimming step, all the initial states

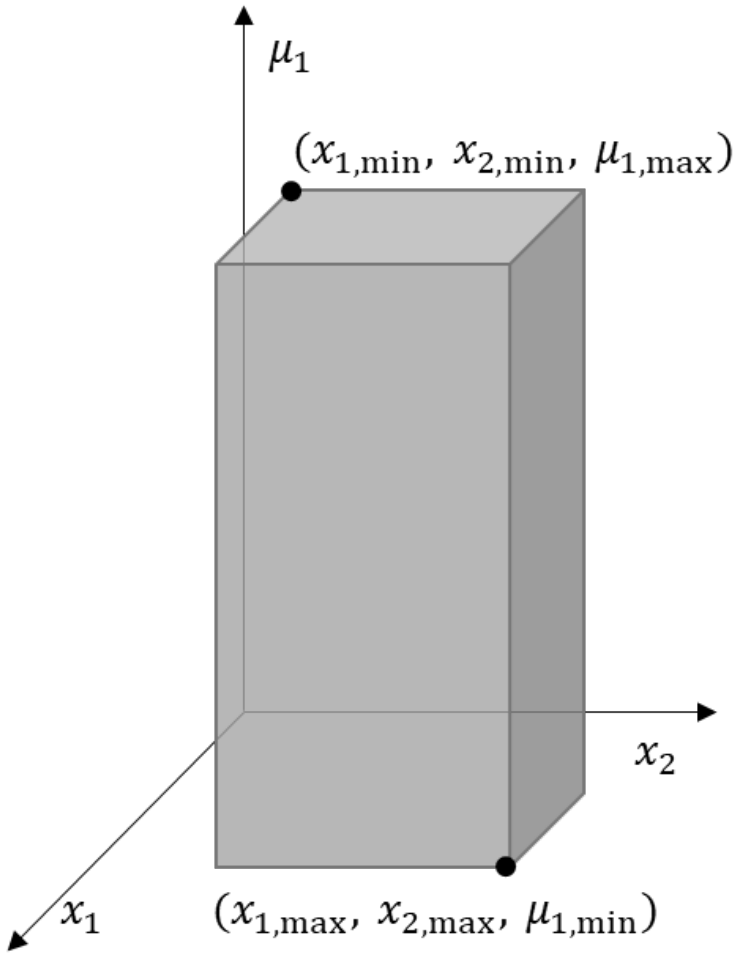


Figure 6.3.: An example for the original hyper-cubic joint space $\mathcal{J}$ where $N_r = 2$ and $N_u = 1$

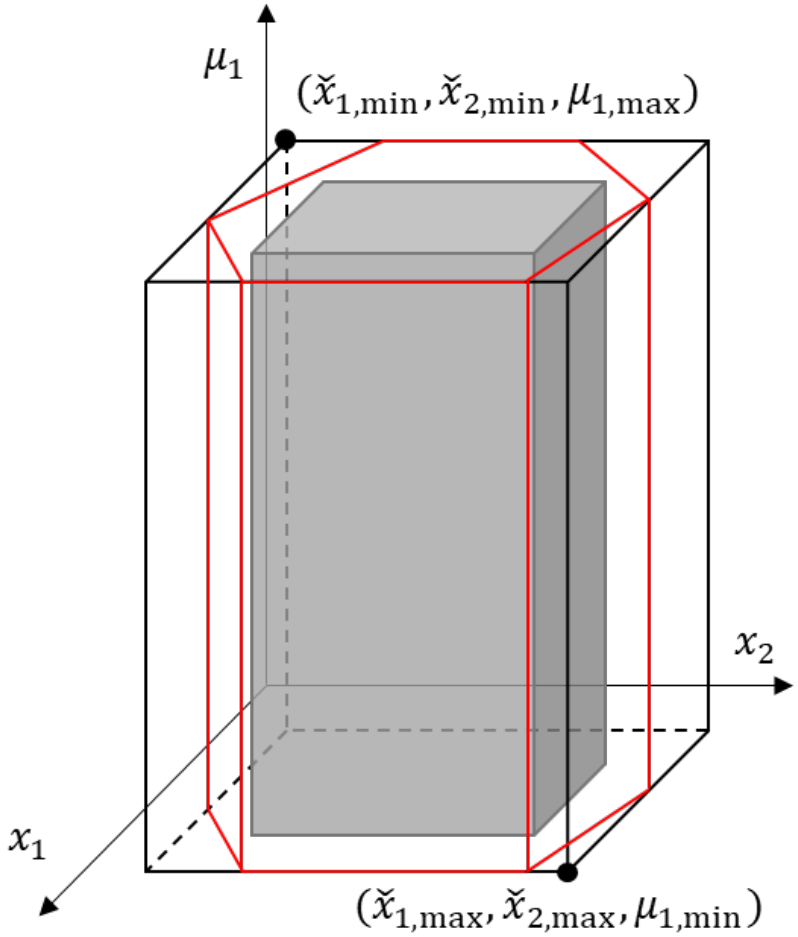


Figure 6.4.: An example for the joint space $\mathcal{J}^*$ after being loosened and trimmed where $N_r = 2$ and $N_u = 1$. The original cubic space is first loosened to the larger cubic space (black lines), then trimmed to the polyhedral space (red lines).

Algorithm 8 Generation of a set of random joint samples $\boldsymbol{J}$

Require: $\mathcal{M}, N_s, m$, FOM

Create a sample set $M = \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, ..., \boldsymbol{\mu}_m\}$ in $\mathcal{M}$
Perform SPS with m parameter samples and FOM to get $\boldsymbol{Y}_{\text{estimate}}$
Apply POD to $\boldsymbol{Y}_{\text{estimate}}$ to get $\boldsymbol{V}$ and $\boldsymbol{X}_{\text{estimate}}$
Find the upper and lower limits for each POD component, denote them as $\{x_{1,\min}, x_{1,\max}, x_{2,\min}, x_{2,\max}, ..., x_{N_r,\min}, x_{N_r,\max}\}$
Create hyper-cubic space for reduced state, denote it as $\mathcal{X}$
Loosen space $\mathcal{X}$ to get the new space $\mathcal{X}'$
Determine $y_{\max}$ and $y_{\min}$
$N_{\text{sampled}} = 0, I = \{\}$
while $N_{\text{sampled}} < N_s$ **do**
    Generate a random sample $\boldsymbol{x}_{\text{sample}}$ in $\mathcal{X}'$
    if $\text{MAX}(\boldsymbol{V}\boldsymbol{x}_{\text{sample}}) < y_{\max}$ and $\text{MIN}(\boldsymbol{V}\boldsymbol{x}_{\text{sample}}) > y_{\min}$ **then**
        Generate a random sample $\boldsymbol{\mu}_{\text{sample}}$ in $\mathcal{M}$
        $I = I \cup \left\{ \begin{bmatrix} \boldsymbol{\mu}_{\text{sample}} \\ \boldsymbol{x}_{\text{sample}} \end{bmatrix} \right\}$
        $N_{\text{sampled}} = N_{\text{sampled}} + 1$
    end if
end while
return I

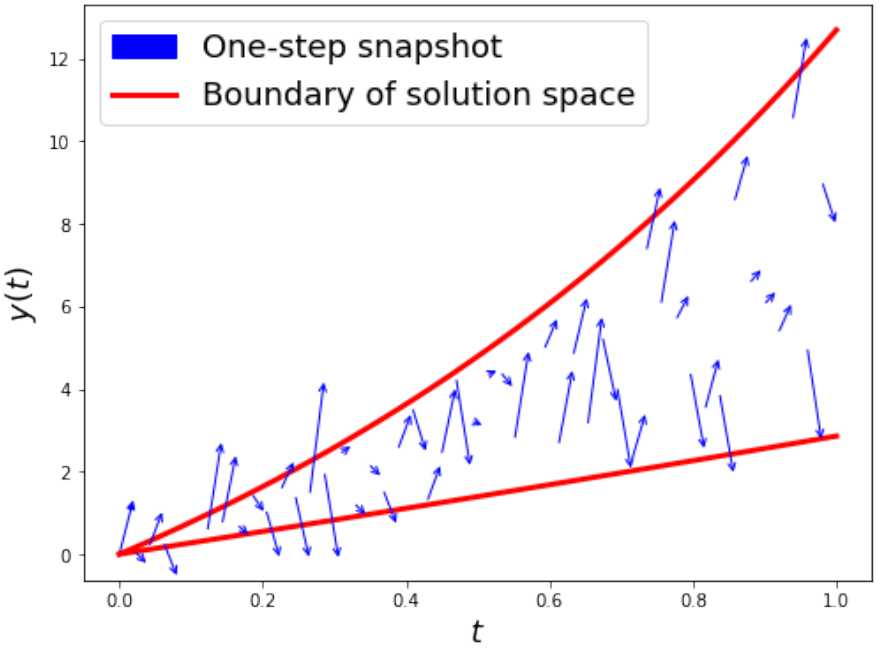


Figure 6.5.: The distribution of the one-step snapshots in the solution space. Compared to the DPS-collected snapshots, the one-step snapshots have a relatively uniform distribution in the solution space. Compared to the SPS method, the parameters sampled by the JSS method have more diversity.

(the starts of the blue arrow lines) are located inside the solution space. Nevertheless, since we assign a random parameter for each IVP, we cannot ensure that all the updated system states (the ends of the blue arrow lines) always fall into the solution space. In Figure 6.5, we indeed see some lines whose end points are outside the solution space. In this case, if we insist to collecting multi-step snapshots, the system states collected on those long trajectories can be far outside the solution space. This phenomenon will cause a problem that the solution space is expanded unnecessarily, which will influence the ROM's performance in the online phase negatively.

6.3. ROM validation

ROM validation is a step where we evaluate the validity of the ROM. Based on whether specific test cases need to be manually designed, we divide ROM validation strategies into two classes: case-dependent ROM validation and case-independent ROM validation. In this section, we will introduce how to use them to assess the quality of the ROM.

6.3.1. Case-dependent validation

Case-dependent validation is probably the most widely-used validation method for MOR. In this kind of approaches, some specific test cases will be defined. The ROM and the FOM will be used to simulate in the same test cases, and their results will be compared. As introduced in [123], we can use these types of use cases as the selection for test cases:

1. Normal events: system parameters are assigned with some foreseeable values. The designed input parameters usually have some specific meanings in industrial use cases. For example, for a radiation furnace model, we can consider using "constant-power heating-up", "increasing-power heating-up", "periodic-power heating-up", etc., as the test parameters.

2. Extreme-condition events: system parameters are assigned with extreme or unlikely combination. Thinking of the thermal system, in the test cases, we can use the highest heating power, the lowest heating power, the strongest heat convection and so on.

3. Historical events: if historical events are available, we can pick some input parameters from the record and use them as the test cases.

Here we denote the test parameters as $\boldsymbol{\mu}_{\text{valid}}(t)$ to distinguish them from $\boldsymbol{\mu}_{\text{test}}(t)$ used by ROM identification. Once we define $\boldsymbol{\mu}_{\text{valid}}(t)$ by any above means. We can perform

the ROM and the FOM simulation:

$$\begin{aligned} \boldsymbol{Y}_{\text{valid}} &= \text{FOM}(\boldsymbol{\mu}_{\text{valid}}(t)), \\ \hat{\boldsymbol{Y}}_{\text{valid}} &= \text{ROM}(\boldsymbol{\mu}_{\text{valid}}(t)), \end{aligned} \tag{6.16}$$

where:

$$\begin{aligned} \boldsymbol{Y}_{\text{valid}} &= \begin{bmatrix} \boldsymbol{y}_{0,\text{valid}} & \boldsymbol{y}_{1,\text{valid}} & \boldsymbol{y}_{2,\text{valid}} & \cdots & \boldsymbol{y}_{K,\text{valid}} \end{bmatrix}, \\ \hat{\boldsymbol{Y}}_{\text{valid}} &= \begin{bmatrix} \hat{\boldsymbol{y}}_{0,\text{valid}} & \hat{\boldsymbol{y}}_{1,\text{valid}} & \hat{\boldsymbol{y}}_{2,\text{valid}} & \cdots & \hat{\boldsymbol{y}}_{K,\text{valid}} \end{bmatrix}. \end{aligned} \tag{6.17}$$

Then we can calculate the mean absolute error or the mean relative error for the ROM solution:

$$\begin{aligned} e_{\text{abs}} &= \frac{1}{K+1} \sum_{i=0}^{K} \left\| \boldsymbol{y}_{i,\text{valid}} - \hat{\boldsymbol{y}}_{i,\text{valid}} \right\| \\ e_{\text{rel}} &= \frac{1}{K+1} \sum_{i=0}^{K} \frac{\left\| \boldsymbol{y}_{i,\text{valid}} - \hat{\boldsymbol{y}}_{i,\text{valid}} \right\|}{\left\| \boldsymbol{y}_{i,\text{valid}} \right\|} \end{aligned} \tag{6.18}$$

Usually, we can also compute the error fields at different time points for error visualization and statistical analysis:

$$\begin{aligned} \boldsymbol{e}_{\text{abs}} &= \begin{bmatrix} \boldsymbol{e}_{1,\text{abs}} & \boldsymbol{e}_{2,\text{abs}} & \cdots & \boldsymbol{e}_{K,\text{abs}} \end{bmatrix} = \left| \boldsymbol{Y}_{\text{valid}} - \hat{\boldsymbol{Y}}_{\text{valid}} \right| \\ \boldsymbol{e}_{\text{rel}} &= \begin{bmatrix} \boldsymbol{e}_{1,\text{rel}} & \boldsymbol{e}_{2,\text{rel}} & \cdots & \boldsymbol{e}_{K,\text{rel}} \end{bmatrix} = \frac{\left| \boldsymbol{Y}_{\text{valid}} - \hat{\boldsymbol{Y}}_{\text{valid}} \right|}{\left| \boldsymbol{Y}_{\text{valid}} \right|} \end{aligned} \tag{6.19}$$

Using the error field $\boldsymbol{e}$, some statistical indicators, such as Mean, Standard Deviation (Std.), Minimum (Min.), Maximum (Max.) and Median, can be computed to evaluate the ROM's performance over the entire model geometry.

6.3.2. Case-independent validation

The other model validation method is called case-independent validation. Just as its name implies, this kind of methods do not use any use-case-specific solution to test the ROM. For example, in the field of MOR, in [124], the authors use Nonlinear Normal Mode (NNM) computed from the ROM as the initial conditions and integration periods for the FOM. By investigating the corresponding FOM outputs, the quality of the ROM is evaluated. In [125], Sensitivity Analysis is applied to assess the quality of the reduced basis. In this thesis, we propose a case-independent approach based on Statistical Machine Learning (ML). The basic theory of this approach is Probably Approximately Correct (PAC) learning [126, 127, 128].

Similar ideas of using the PAC learning theory for assessing simulation models can be found in [128, 129]. There are two key performance indicators which need to be

determined in the assessment: ROM accuracy and ROM confidence. They can be determined by applying a large number of independent tests to the ROM.

First of all, we use an inequality to represent the relationship between ROM accuracy and ROM confidence:

$$Pr(e \leq \tau) > \eta, \tag{6.20}$$

where e is the prediction error of the ROM and τ is a predefined error value. The meaning of Inequality 6.20 is that the ROM's prediction error e being lower than τ has a probability of η. We should notice that the prediction error e here can be either an absolute error or a relative error, and τ should be computed in the same approach.

We can consider τ as the accuracy indicator and η as the corresponding confidence indicator. Here we clarify that although τ is the prediction error, it can certainly be used to describe the accuracy of the ROM. For simplicity, from here on, we will consider the ROM accuracy and ROM's prediction error equivalently. The accuracy and confidence is a pair of variables which are (negatively) correlated. Our target is to find a way to computing the values of these coupled variables.

Using the one-step snapshot introduced in section 6.2.3, we can construct s independent tests for the validation, and we denote the input vector of the i-th test as:

$$\boldsymbol{J}_{i,\text{in}} = \begin{bmatrix} \boldsymbol{\mu}_{i,\text{in}} \\ \boldsymbol{x}_{i,\text{in}} \end{bmatrix}. \tag{6.21}$$

We further denote the reference solution computed by the FOM as:

$$\boldsymbol{y}_{i,\text{out}} = \text{FOM}\left(\boldsymbol{J}_{i,\text{in}}, \boldsymbol{V}, \delta t\right). \tag{6.22}$$

The corresponding prediction by the ROM is:

$$\hat{\boldsymbol{y}}_{i,\text{out}} = \text{ROM}\left(\boldsymbol{J}_{i,\text{in}}, \boldsymbol{V}, \delta t\right). \tag{6.23}$$

We can compute the prediction error (relative) for test i as:

$$e_i = \frac{\|\hat{\boldsymbol{y}}_{i,\text{out}} - \boldsymbol{y}_{i,\text{out}}\|}{\|\boldsymbol{y}_{i,\text{out}}\|}. \tag{6.24}$$

Assuming there are s test cases in validation and we have defined the investigated accuracy τ, we can calculate the observed confidence $p(\tau)$ of the ROM having a ROM error smaller than τ using Equation 6.25:

$$\begin{aligned} p(\tau) &= \frac{1}{s}\sum_{i=1}^{s} L(e_i \leq \tau), \\ L(e_i, \tau) &= \begin{cases} 1 & e_i \leq \tau \\ 0 & e_i > \tau \end{cases}. \end{aligned} \tag{6.25}$$

In Equation 6.25, it is not hard to see that the lower τ is, the lower $p(\tau)$ is. This matches the description that accuracy and confidence is a pair of (negatively) correlated parameters.

However, since we can only construct limited number of tests for ROM validation, the observed confidence of the ROM must have deviation from the true confidence of the ROM. Since the true confidence of the ROM is still unknown to us, we can establish an inequality for the observed confidence and the true confidence:

$$Pr(|p(\tau) - \check{p}(\tau)| < \epsilon) > 1 - \sigma, \tag{6.26}$$

where $\check{p}(\tau)$ is the true confidence of the ROM. We can also notice that ϵ and $1 - \sigma$ is a pair of accuracy and confidence of validation itself. For holding Inequality 6.26, at least s_{PAC} test samples must be included in validation:

$$s_{\text{PAC}} = \frac{1}{2\epsilon^2} ln \left(\frac{2}{\sigma} \right). \tag{6.27}$$

Equation 6.27 is called Chernoff's bound. The proof of Equation 6.27 is given in Appendix A.1.

Assuming we find τ^* which is the smallest τ satisfying:

$$p(\tau^*) = 1, \tag{6.28}$$

which means all prepared tests can be passed with prediction errors lower than τ^*. While seeking τ^*, we can perform binary search [130] on a pre-built grid $\{\tau_1, \tau_2, ..., \tau_k\}$ of τ.

Substituting Equation 6.28 into Inequality 6.26 produces:

$$Pr(|1 - \check{p}(\tau^*)| < \epsilon) > 1 - \sigma, \tag{6.29}$$

which is equivalent to:

$$|1 - \check{p}(\tau^*)| < \epsilon \tag{6.30}$$

holding with probability of $1 - \sigma$. If we remove the absolute, we will get:

$$\begin{aligned} -\epsilon <& 1 - \check{p}(\tau^*) < \epsilon \\ & \Updownarrow \\ 1 - \epsilon <& \check{p}(\tau^*) < 1 + \epsilon \end{aligned} \tag{6.31}$$

holding with probability of $1 - \sigma$. Here we only need the left inequality $1 - \epsilon < \check{p}(\tau^*)$, because we know the true confidence cannot be larger than 1. Finally, we get the true confidence of the ROM at accuracy level τ^* is $1 - \epsilon$, and this conclusion is reliable with probability of $1 - \sigma$. Since we can only evaluate for limited number

of τ, the found τ^* must be larger than the true τ^*. In other words, we will always underestimate the accuracy of the ROM, and the coarser the grid is, the more we underestimate.

We notice that the reliability of the conclusion is influenced by the variable σ. The smaller σ is, the more reliable the conclusion is. At the same time, a smaller σ also leads to a greater s_{PAC}, which further means we need more test samples for validation.

Algorithm 9 Prepare PAC validation

Require: ϵ, σ, FOM, $\boldsymbol{V}$

1: Determine s based on $s \geq s_{\text{PAC}}$, where s_{PAC} can be computed using Equation 6.27

2: Randomly take s samples following Algorithm 8,
denoted as $I_{\text{PAC}} = \left\{ \boldsymbol{J}_{1,\text{in}}^{\text{PAC}}, \boldsymbol{J}_{2,\text{in}}^{\text{PAC}}, ..., \boldsymbol{J}_{s,\text{in}}^{\text{PAC}} \right\}$

3: Get reference solution $O_{\text{PAC}} = \left\{ \boldsymbol{y}_{1,\text{out}}^{\text{PAC}}, \boldsymbol{y}_{2,\text{out}}^{\text{PAC}}, ..., \boldsymbol{y}_{s,\text{out}}^{\text{PAC}} \right\}$,
where $\boldsymbol{y}_{i,\text{out}}^{\text{PAC}} = \text{FOM}(\boldsymbol{J}_{i,\text{in}}^{\text{PAC}}, \boldsymbol{V}, \delta t)$

4: **return** $I_{\text{PAC}}, O_{\text{PAC}}$

In Algorithm 9, we demonstrate how to prepare independent test samples for ROM validation based on PAC learning. Furthermore, we denote such a validation step as:

$$\tau^*, p(\tau^*_{\text{design}}), \boldsymbol{e}_{\text{PAC}} = \text{PAC}(\text{ROM}, \boldsymbol{V}, \tau^*_{\text{design}}). \tag{6.32}$$

In the validation step, we will have 3 important outputs. $p(\tau^*_{\text{design}})$ is the observed confidence for an expected accuracy level τ^*_{design}. τ^* is the accuracy level whose observed confidence is 1. This two outputs can reflect the deviation between the current ROM quality and the expected ROM quality. As a by-product, we can have a error snapshot matrix $\boldsymbol{e}_{\text{PAC}}$ saving the ROM errors in all tests.

6.4. Snapshot selection based on the concept of Active Learning

The key principle of AL is that a ML algorithm can achieve a higher accuracy with fewer training data if it is allowed to choose the data from which it learns [31]. In the field of ML, lots of application of AL can be found, for example, in Recommender Systems [131], Natural Language Processing [132], etc.

In conventional model-based MOR methods, similar principles are already employed, e.g., in [133, 134, 28, 26, 27]. For data-driven MOR, some research regarding

applying AL to existing MOR methods also can be found. For example, in [30], the AL concept is employed for inferring a linear system. In [94], an AL algorithm is designed for a Gaussian-Process-Regression-based MOR method. In this section, we will show how to employ the AL approaches introduced in section 4.2 in the MOR context.

A typical flowchart for Active-Learning-based Model Order Reduction (AL-MOR) can be concluded as Figure 6.6. In the flowchart, "Initialization" includes necessary

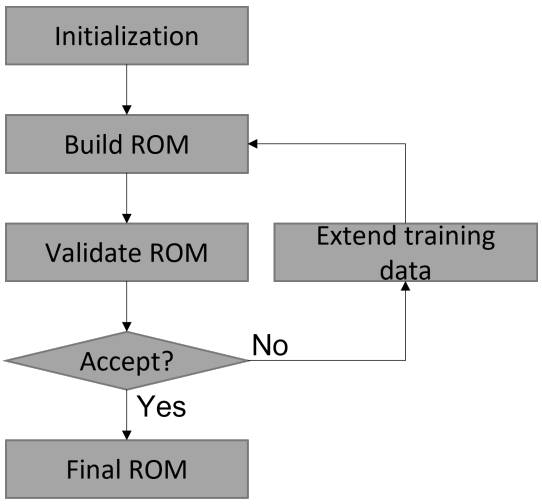


Figure 6.6.: The flowchart of AL-MOR.

steps such as constructing the FOM, pre-defining the parameter space of the FOM, collecting initial training datasets, preparing the data pool, etc. "Build ROM" simply means building the reduced space and identifying the ROM in the reduced space. As we see in Figure 6.6, so far, we have prepared all the components but only "Extend training data" is still missing. Here, in this section, we will use the active/passive selection methods introduced in chapter 4 to support the selection of samples.

6.4.1. Snapshot selection based on active selection

In this section, we will apply the active sample selection methods described in section 4.2.1 and section 4.2.2 to the procedure of constructing the ROM.

6.4.1.1. Snapshot selection based on the Maximum Mean Squared Error

Here, we will incorporate the idea introduced in [94], which is essentially active selection based on the Maximum Mean Squared Error (MMSE) introduced in section 4.2.1. We will combine it with our JSS method introduced in section 6.2.

As introduced in section 4.2.1, we want to know with what inputs, the current ROM has the largest approximation errors. Since the errors are the ROM errors with respect to the joint samples remaining in the data pool, whose reference solutions (FOM snapshots) are not computed yet, we need to estimate these errors. To keep the non-intrusiveness of the whole workflow, a data-driven error estimator is therefore demanded. Data-driven approaches are already tested for supporting error estimation for numerical simulations or MOR, such as research presented in [135, 136, 137].

Recalling the non-intrusive ROM identification methods introduced in chapter 3. We can encapsulate different data-driven ROMs into such a feedforward ML model:

$$\begin{cases} \hat{\boldsymbol{x}}_{\text{out}} = \text{ROM}(\boldsymbol{x}_{\text{in}}, \boldsymbol{\mu}_{\text{in}}, \delta t) \\ \hat{\boldsymbol{y}}_{\text{out}} = \boldsymbol{V}\hat{\boldsymbol{x}}_{\text{out}} \end{cases}, \tag{6.33}$$

or simply as:

$$\hat{\boldsymbol{y}}_{\text{out}} = \text{ROM}(\boldsymbol{J}_{\text{in}}, \boldsymbol{V}, \delta t). \tag{6.34}$$

Such a ROM is used recurrently in the online phase. After each step, the predicted reduced state will be passed to the next time point. As we can see in Equation 6.33, besides the fixed parameter δt, there are mainly two variables in the input: the reduced state and the system parameters at current time point. The predicted reduced state at the new time point is lifted to the full space using the reduced basis $\boldsymbol{V}$. Finally, the output of such a ROM is the predicted state $\hat{\boldsymbol{y}}_{\text{out}}$ of the FOM.

To assess the prediction quality, we usually measure the relative error or absolute error between the predicted full state and the reference full state. Here we use the relative error computed by Equation 6.24 as an example.

We know that:

$$\boldsymbol{y}_{\text{out}} = \text{FOM}(\boldsymbol{J}_{\text{in}}, \boldsymbol{V}, \delta t). \tag{6.35}$$

Substituting Equation 6.35 into Equation 6.24 yields:

$$e = \frac{\|\text{ROM}(\boldsymbol{J}_{\text{in}}, \boldsymbol{V}, \delta t) - \text{FOM}(\boldsymbol{J}_{\text{in}}, \boldsymbol{V}, \delta t)\|}{\|\text{FOM}(\boldsymbol{J}_{\text{in}}, \boldsymbol{V}, \delta t)\|}. \tag{6.36}$$

According to Equation 6.36, we can summarize the computation of the error with:

$$e = \Lambda(\boldsymbol{J}_{\text{in}}, \boldsymbol{V}, \delta t), \tag{6.37}$$

where Λ is the exact error function, and it is not available to us. Since usually δt is constant, and $\boldsymbol{V}$ is also constant in each iteration. Based on Equation 6.37 and simplification, we can expect to estimate the ROM error with:

$$\hat{e} = \hat{\Lambda}(\boldsymbol{J}_{\text{in}}) \tag{6.38}$$

where $\hat{e}$ is the estimation for the ROM error e. To construct the error estimator $\hat{\Lambda}$, we have different choices. Here in this thesis, we will introduce two approaches.

Error estimation based on RBF interpolation Radial Basis Function (RBF) interpolation is one of the primary tools for interpolating multidimensional scattered data [138, 139]. Here we consider to use RBF interpolation to estimate ROM errors for unlabelled samples in the data pool. To construct the error estimator, the first thing we need is training data. In our case, the training data for the error estimator is the test data used by PAC validation.

As we know, for validating the ROM with the PAC learning theory, we have constructed s independent tests. By collecting the prediction errors in PAC validation, we can construct the so-called error snapshot matrix:

$$\boldsymbol{e}_{\text{PAC}} = \begin{bmatrix} e_1 & e_2 & \dots & e_s \end{bmatrix}^T, \tag{6.39}$$

where e_i is the ROM error in the i-th test. We can further build a matrix for the joint samples in validation as:

$$\boldsymbol{I}_{\text{PAC}} = \begin{bmatrix} \boldsymbol{J}_1 & \boldsymbol{J}_2 & \dots & \boldsymbol{J}_s \end{bmatrix}, \tag{6.40}$$

whose column vector $\boldsymbol{J}_i$ is equivalent to the joint sample $\boldsymbol{J}_{i,\text{in}}^{\text{PAC}}$ in Algorithm 9, and we drop the superscript and a subscript for simplicity. Assuming we have another matrix $\boldsymbol{I}^*$ consisting of the joint samples $\boldsymbol{J}^*$ remaining in the data pool:

$$\boldsymbol{I}^* = \begin{bmatrix} \boldsymbol{J}_1^* & \boldsymbol{J}_2^* & \dots & \boldsymbol{J}_n^* \end{bmatrix}, \tag{6.41}$$

and we denote their corresponding ROM prediction errors as:

$$\boldsymbol{e}^* = \begin{bmatrix} e_1^* & e_2^* & \dots & e_n^* \end{bmatrix}^T. \tag{6.42}$$

We can use RBF interpolation to estimate the ROM error in such a way [140]:

$$\hat{\boldsymbol{e}}^* = \hat{\Lambda}(\boldsymbol{I}^*) = \boldsymbol{F}(\boldsymbol{I}^*, \boldsymbol{I}_{\text{PAC}})\boldsymbol{a} + \boldsymbol{P}(\boldsymbol{I}^*)\boldsymbol{b}, \tag{6.43}$$

where $\boldsymbol{a}$ and $\boldsymbol{b}$ are the coefficients need to be determined during the fitting process. $\boldsymbol{F}$ is a matrix whose entries can be computed by:

$$F_{ij} = \text{RBF}(\boldsymbol{J}_i^*, \boldsymbol{J}_j), \tag{6.44}$$

and $\boldsymbol{P}$ is a matrix of monomials. There are multiple choices for the RBF:

$$\begin{aligned} \text{Linear: } & -r, \\ \text{Thin Plate Spline: } & r^2 log(r), \\ \text{Gaussian: } & exp(-r^2), \end{aligned} \tag{6.45}$$

where $r = \|\boldsymbol{J}^* - \boldsymbol{J}\|$ means the distance between the remaining joint samples and the joint samples in PAC validation. To solve the coefficients $\boldsymbol{a}$ and $\boldsymbol{b}$, the following conditions are enforced:

$$\begin{cases} \boldsymbol{e}_{\mathrm{PAC}} = \boldsymbol{F}(\boldsymbol{I}_{\mathrm{PAC}}, \boldsymbol{I}_{\mathrm{PAC}})\boldsymbol{a} + \boldsymbol{P}(\boldsymbol{I}_{\mathrm{PAC}})\boldsymbol{b}, \\ \boldsymbol{P}(\boldsymbol{I}_{\mathrm{PAC}})^T\boldsymbol{a} = 0. \end{cases} \tag{6.46}$$

In this thesis, we will use Thine Plate Spline for RBFs, which is also the default choice selected by Python library SciPy [141].

Error estimation based on GPR Gaussian Process Regression (GPR) [142], also known as Kriging Interpolation [143], is a non-parametric model used for regression or interpolation. We use the same notation as defined for RBF error estimator.

Gaussian Process prescribes that $\boldsymbol{e}_{\mathrm{PAC}}$ and $\boldsymbol{e}^*$ follow a joint Gaussian distribution, which can be written as:

$$\begin{bmatrix} \boldsymbol{e}_{\mathrm{PAC}} \\ \boldsymbol{e}^* \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} \boldsymbol{K}(\boldsymbol{I}_{\mathrm{PAC}}, \boldsymbol{I}_{\mathrm{PAC}}) & \boldsymbol{K}(\boldsymbol{I}_{\mathrm{PAC}}, \boldsymbol{I}^*) \\ \boldsymbol{K}(\boldsymbol{I}^*, \boldsymbol{I}_{\mathrm{PAC}}) & \boldsymbol{K}(\boldsymbol{I}^*, \boldsymbol{I}^*) \end{bmatrix}\right), \tag{6.47}$$

and we are interested in knowing the conditional distribution:

$$p(\boldsymbol{e}^* | \boldsymbol{I}_{\mathrm{PAC}}, \boldsymbol{e}_{\mathrm{PAC}}, \boldsymbol{I}^*). \tag{6.48}$$

We can use the mean of this conditional distribution to estimate the ROM prediction errors. The mean of this distribution is [144]:

$$\boldsymbol{K}(\boldsymbol{I}^*, \boldsymbol{I}_{\mathrm{PAC}})\boldsymbol{K}(\boldsymbol{I}_{\mathrm{PAC}}, \boldsymbol{I}_{\mathrm{PAC}})^{-1}\boldsymbol{e}_{\mathrm{PAC}}. \tag{6.49}$$

To compute the covariance matrix $\boldsymbol{K}$, we introduce the so-called covariance function κ, and the entries in the covariance matrix $\boldsymbol{K}$ can be computed with:

$$K_{ij} = \kappa(\boldsymbol{J}_i^*, \boldsymbol{J}_j). \tag{6.50}$$

Here we introduce some widely used covariance functions:

$$\begin{aligned} \text{Constant: } \kappa(\boldsymbol{J}_i^*, \boldsymbol{J}_j) &= c, \\ \text{Linear: } \kappa(\boldsymbol{J}_i^*, \boldsymbol{J}_j) &= \boldsymbol{J}_i^{*T}\boldsymbol{J}_j, \\ \text{Squared exponential: } \kappa(\boldsymbol{J}_i^*, \boldsymbol{J}_j) &= \sigma^2 exp\left(-\frac{\left\|\boldsymbol{J}_i^* - \boldsymbol{J}_j\right\|^2}{2l^2}\right). \end{aligned} \tag{6.51}$$

In this paper, we employ a combination of the squared exponential function and the constant function, i.e.:

$$\kappa(\boldsymbol{J}_i^*, \boldsymbol{J}_j) = \sigma^2 exp\left(-\frac{\left\|\boldsymbol{J}_i^* - \boldsymbol{J}_j\right\|^2}{2l^2}\right) + c. \tag{6.52}$$

With the defined covariance function, we can compute the covariance matrices in Equation 6.49 and the predicted ROM errors $\hat{\boldsymbol{e}}^*$.

Using the error estimation methods introduced above, we can estimate the ROM errors for all the samples remaining in the current data pool and pick those joint samples with maximal estimated ROM errors for the next iteration. Based on this strategy, we can finalize the MMSE algorithm for MOR. In Algorithm 10, I_{all} is the

Algorithm 10 MMSE in the j-th iteration

Require: Snapshot increment Δs, current data pool $I_{\text{all}} \setminus I_j$, PAC, ROM_j

1: Compute $\boldsymbol{e}_{\text{PAC}}$ using ROM_j and PAC
2: Fit the RBF-/GPR-based error estimator using $\boldsymbol{I}_{\text{PAC}}$ and $\boldsymbol{e}_{\text{PAC}}$
3: Compute $\boldsymbol{e}^*$ using the fitted error estimator and the matrix $\boldsymbol{I}^*$ built by $I_{\text{all}} \setminus I_j$
4: Use $e_i^* \in \boldsymbol{e}^*$ as informativeness measure, and perform the DBAL method in Algorithm 6 to generate $\Delta I = \{\check{\jmath}_1, \check{\jmath}_2, \ldots, \check{\jmath}_{\Delta s}\}$
5: **return** ΔI

initial data pool prepared by Algorithm 8, I_j is the training data in the j-th iteration, and ROM_j is the ROM in the j-th iteration. The MMSE algorithm is enhanced by the Diverse mini-Batch Active Learning (DBAL) method to allow a batch mode for sample selection. The return ΔI of the algorithm is the remaining samples that are considered the most informative.

6.4.1.2. Agreement ratio

So far, we have introduced two approaches for estimating ROM errors. Later, we will test their performance with numerical experiments. Before that, we introduce a performance indicator which will be used in the experiments. We name this indicator as 'agreement ratio' and denoted as α.

Definition As we know, in the j-th iteration, we will extend the training data with the new data generated from the Δs joint samples selected from the current data pool $I_{\text{all}} \setminus I_j$. No matter with which error estimation method, we expect that the selected joint samples are the most informative for the current ROM. In other words, by predicting with them, the ROM should generate the largest prediction errors.

However, the selection strategy cannot be 100% accurate. If we compare to the samples that exactly maximizes the ROM error, we must observe some deviation. We can describe the deviation using:

$$\alpha = \frac{N(\Delta I_{\text{estimate}} \cap \Delta I_{\text{true}})}{\Delta s}, \tag{6.53}$$

where $\Delta I_{\text{estimate}}$ is the sample set selected by the estimation, and ΔI_{true} is the true selection. The function $N(\cdot)$ counts the number of entries in a set. Based on its definition, we know that the higher α is, the better agreement between the estimation and the ground truth is.

Calculus for reference agreement ration We can compute a reference α, denoted as α_{ref}, which is the expectation of the agreement rate of randomly selecting Δs samples from the data pool $I_{\text{all}} \setminus I_j$. For simplicity, we introduce $N_p = N(I_{\text{all}} \setminus I_j)$ and $N_{\text{all}} = N(I_{\text{all}})$. The reference α can be computed as:

$$\alpha_{\text{ref}} = \left(\sum_{n=0}^{\Delta s} n \cdot \frac{\binom{\Delta s}{n} \binom{N_p - \Delta s}{\Delta s - n}}{\binom{N_p}{\Delta s}} \right) / \Delta s. \tag{6.54}$$

Noticing that $N_p = N_{\text{all}} - j \cdot \Delta s$. We can further write:

$$\alpha_{\text{ref}}(j) = \left(\sum_{n=0}^{\Delta s} n \cdot \frac{\binom{\Delta s}{n} \binom{N_{\text{all}} - (j+1) \cdot \Delta s}{\Delta s - n}}{\binom{N_{\text{all}} - j \cdot \Delta s}{\Delta s}} \right) / \Delta s. \tag{6.55}$$

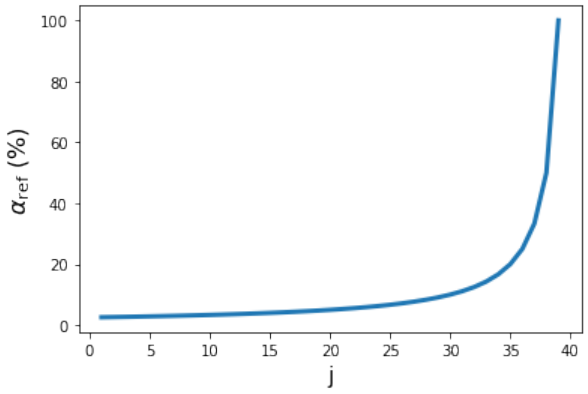


Figure 6.7.: The curve of α_{ref}. The expected agreement ratio increases with the number of performed iterations. However, unless a large amount of training snapshots are needed (more than 90% of the data pool), the expected agreement ratio of random selection is lower than 50%.

Example Let us assume that we have 20,000 joint samples in the initial data pool, and each time we pick 500 samples from the data pool. Based on Equation 6.55, we can draw the curve for the function $\alpha_{\text{ref}}(j)$, which is presented in Figure 6.7. As we can observe, the reference agreement ratio at early stages is very low, and it will increase fast in late stages. In the training process, if the agreement ratio is higher than this reference agreement ratio, we can consider snapshot selection is guided by the error estimator effectively.

6.4.1.3. Snapshot selection based on Query by Committee

As introduced in section 4.2.2, snapshot selection guided by ROM uncertainty is another potential selection strategy. The motivation to applying such an uncertainty-based strategy is our inconfidence of data-driven error estimation. The inconfidence resulted from such a consideration: if ML model A can simulate the prediction error of ML model B, then it is very likely that ML model A has a much higher accuracy and a better stability than ML model B.

Similarly, if our error estimator can predict the ROM error accurately, the error estimator will be a more stable and accurate model compared to the ROM. Then a question arises: why not directly using the model architecture of the error estimator for the ROM? If the error estimator is not stable and accurate enough, the snapshot selection based on error estimation might be ineffective. Based on this concern, an uncertainty-based strategy will be a good alternative.

According to the method introduced in section 4.2.2, we need to train multiple ROMs in each iteration. However, in reality, this is nearly infeasible. The reason is that training multiple ROMs is time consuming and will significantly drag out the offline phase.

Compared to training a ROM, fitting an error estimator, especially a RBF-based estimator, consumes much less time. In this thesis, we propose to employ error estimators constructed in different ways as committee members in the Query by Committee (QBC) method. Based on the concept of QBC, we know that the samples selected by the committee will be the most uncertain for the members in the committee. This further means that such samples are the most difficult to predict by the error estimators.

Assuming we have k error estimators, we denote them as:

$$\hat{\Lambda}_1, \hat{\Lambda}_2, ..., \hat{\Lambda}_k. \tag{6.56}$$

They are fitted with the test joint samples $\boldsymbol{I}_{\text{PAC}}$ stored in the PAC validator and the error snapshots $\boldsymbol{e}_{\text{PAC}}$ generated by the PAC validator. We use them to predict for a

joint sample $\boldsymbol{J}_i^*$ in the data pool:

$$\begin{aligned} \hat{e}_{i1}^* &= \hat{\Lambda}_1(\boldsymbol{J}_i^*), \\ \hat{e}_{i2}^* &= \hat{\Lambda}_2(\boldsymbol{J}_i^*), \\ &\vdots \\ \hat{e}_{ik}^* &= \hat{\Lambda}_k(\boldsymbol{J}_i^*). \end{aligned} \tag{6.57}$$

According to [103], we can define the ambiguity of the prediction $\hat{e}_i^*$ as:

$$a_i = a(\boldsymbol{J}_i^*) = \sum_{j=1}^{k} (\hat{e}_{ij}^* - \text{Mean}(\hat{e}_i^*))^2. \tag{6.58}$$

The large a_i is, the more difficult it is to predict the ROM error e_i^* generated by the joint sample $\boldsymbol{J}_i^*$. So what could this kind of difficulty imply? In Figure 6.8, we demonstrate two situations where this difficulty can occur. We first draw some points to represent the test inputs J in the PAC validator and their ROM errors e. Here we consider the input as a scalar instead of a multi-dimensional joint input just for demonstration.

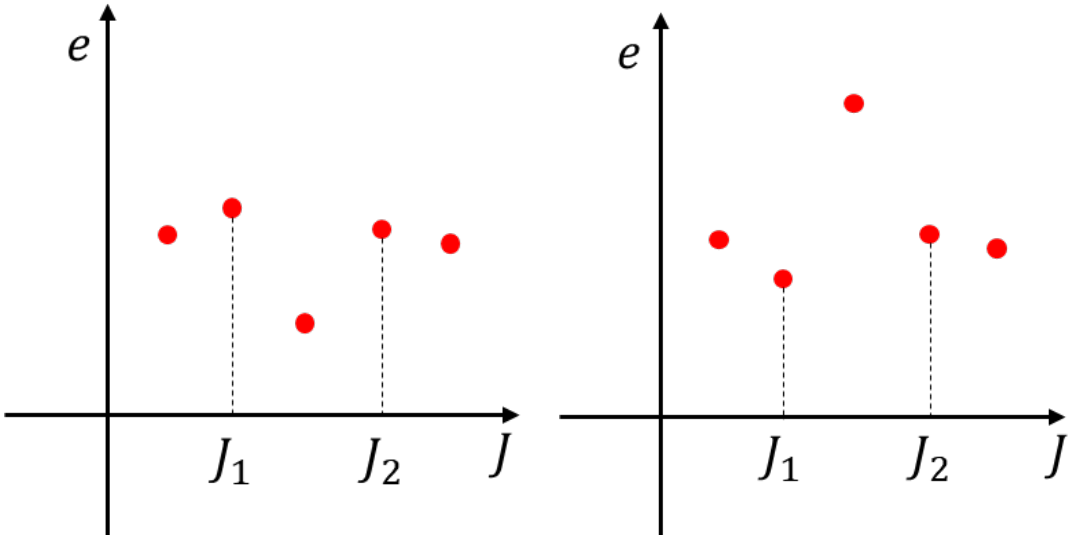
Ideally, we expect the points to distribute in a horizontal line. Because that means the ROM has a stable performance across the input space. However, in reality, we can always find that the ROM is overfitting or underfitting for some inputs. No matter for overfitting or underfitting, what can be observed in the error distribution is an outlier. The ROM error e for this outlier is much lower (overfitting) or higher (underfitting) in its local region. If we use an interpolation-based error estimator to predict for a new input, it will be the most difficult to predict for a new input from the region (J_1, J_2). The committee will also be likely to have disagreement for a new input from this region.

A new sample from such a region will be anyway informative for the model. If underfitting occurs, a new sample will definitely enrich the information in this region and help the model to improve its approximation in this region. If overfitting occurs, a new sample from this region can help the model discover the real FOM dynamics in this region more precisely.

Therefore, we can use a_i as an indicator of informativeness. The bigger a_i is, the more informative the sample $\boldsymbol{J}_i^*$ is.

To construct the committee, we can use, for example, a set of RBF estimators with different RBFs, a set of GPR estimators with different kernel functions, or a mixture of different models. In this thesis, we use three RBF estimators with a linear function, a thin-plate spline function and a Gaussian function as the RBF, respectively.

However, by solely using the QBC method, in each iteration of the AL algorithm, only one new sample can be selected. This will make the procedure of ROM refinement slow and time-consuming. A solution is to use the DBAL algorithm introduced



(a) Overfitting. (b) Underfitting.

Figure 6.8.: Two situations where it is difficult to predict the ROM error generated by an input J. Left: overfitting occurs. Right: underfitting occurs.

in section 4.3. Combined with the DBAL algorithm, and the QBC method for MOR is presented in Algorithm 11.

Algorithm 11 QBC in the j-th iteration

Require: Snapshot increment Δs, current data pool $I_{\text{all}} \setminus I_j$, PAC, ROM$_j$

1: Compute $\boldsymbol{e}_{\text{PAC}}$ using ROM$_j$ and PAC
2: Fit three different RBF error estimators using $\boldsymbol{I}_{\text{PAC}}$ and $\boldsymbol{e}_{\text{PAC}}$
3: Compute $\{a_i = a(\boldsymbol{J}_i^*) \mid \forall \boldsymbol{J}_i^* \in I_{\text{all}} \setminus I_j\}$ use the RBF error estimators
4: Use a_i as informativeness measure, and perform the DBAL method in Algorithm 6 to generate $\Delta I = \{\check{\boldsymbol{J}}_1, \check{\boldsymbol{J}}_2, \ldots, \check{\boldsymbol{J}}_{\Delta s}\}$
5: **return** ΔI

6.4.1.4. Algorithm for MOR based on active selection

So far, we have designed the active sample selection methods for MOR. Now, with all ingredients on hand, we can realize Figure 6.6 with Algorithm 12.

In Algorithm 12, the sub-routine CONV is used for determining the convergence of the algorithm. Its details are given in Algorithm 13. The coefficients α_1 and α_2 decide how much relative improvement is expected. Since snapshots are only one of the factors influencing the ROM quality, when the observed improvement is too small to

Algorithm 12 AL-MOR

Require: $\boldsymbol{V}$, I_{all}, Δs, τ^*_{design}, PAC, FOM

Randomly pick Δs samples from I_{all} as the initial snapshot set, denoted as I_1
$O_1 = \{\text{FOM}(\boldsymbol{J}_i, \boldsymbol{V}, \delta t) \mid \forall \boldsymbol{J}_i \in I_1\}$
Use I_1, O_1 and $\boldsymbol{V}$ to train ROM_1
$p_1(\tau^*_{\text{design}}), \tau^*_1, \boldsymbol{e}_{\text{PAC}} = \text{PAC}(\text{ROM}_1, \boldsymbol{V}, \tau^*_{\text{design}})$
$j = 1$
$p_0(\tau^*_{\text{design}}) = 0, \tau^*_0 = 1$
while NOT $\text{CONV}(p_j(\tau^*_{\text{design}}), \tau^*_j, p_{j-1}(\tau^*_{\text{design}}), \tau^*_{j-1})$ **do**
    Pick ΔI for generating the new training data by Algorithm 10 or Algorithm 11
    $I_{j+1} = I_j \cup \Delta I$
    $O_j = \left\{\text{FOM}(\boldsymbol{J}_i, \boldsymbol{V}, \delta t) \mid \forall \boldsymbol{J}_i \in I_j\right\}$
    Update $\boldsymbol{V}$
    Use I_j, O_j and $\boldsymbol{V}$ to train ROM_{j+1}
    $p_j(\tau^*_{\text{design}}), \tau^*_j, \boldsymbol{e}_{\text{PAC}} = \text{PAC}(\text{ROM}_{j+1}, \boldsymbol{V}, \tau^*_{\text{design}})$
    $j = j + 1$
end while
return ROM_{j+1}

continue the iteration, we should stop the iteration actively. We can validate the ROM again with case-dependent validation and combine the results with the performance indicators produced by PAC validation to decide if we should look for improvement from other sources, e.g., the ROM size or the ROM identification method, and redo the whole AL process.

Algorithm 13 CONV

Require: $p_j(\tau^*_{\text{design}}), \tau^*_j, p_{j-1}(\tau^*_{\text{design}}), \tau^*_{j-1}$

$p_{\text{tol}} = \alpha_1 p_{j-1}(\tau^*_{\text{design}})$
$\delta\tau^*_{\text{tol}} = \alpha_2 \tau^*_{j-1}$
if $p_j(\tau^*_{\text{design}}) - p_{j-1}(\tau^*_{\text{design}}) < \delta p_{\text{tol}}$ AND $\tau^*_{j-1} - \tau^*_j < \delta\tau^*_{\text{tol}}$ **then**
    return True
end if
return False

6.4.2. Snapshot selection based on passive selection

Besides the active selection strategies in section 6.4.1.1 and section 6.4.1.3, we can also use the Passive Learning (PL) approach, Farthest Point Sampling (FPS) [145], to update the snapshots for training. Methods with similar concept is already widely used in other fields of ML [146, 147, 114].

To introduce this approach, we first define the distance between two joint samples as:

$$d(\boldsymbol{J}_i, \boldsymbol{J}_j) = \left\| \boldsymbol{J}_i - \boldsymbol{J}_j \right\|. \tag{6.59}$$

Just as the method's name implies, while including a new joint sample for generating the new training data, we intend to pick the joint sample that is the most distant to the joint samples already used to generate the current training data. In this thesis, we define the distance between a joint sample $\boldsymbol{J}_i$ and a set of joint samples $I_j = \left\{ \boldsymbol{J}_{j1}, \boldsymbol{J}_{j2}, ..., \boldsymbol{J}_{jn} \right\}$ as:

$$d(\boldsymbol{J}_i, I_j) = min(\left\{ d(\boldsymbol{J}_i, \boldsymbol{J}_{j1}), d(\boldsymbol{J}_i, \boldsymbol{J}_{j2}), ..., d(\boldsymbol{J}_i, \boldsymbol{J}_{jn}) \right\}). \tag{6.60}$$

Just as presented in Figure 6.9, Equation 6.60 means the distance between the joint sample $\boldsymbol{J}_i$ and the set of joint samples I_j is defined as the shortest distance between $\boldsymbol{J}_i$ and $\boldsymbol{J}_{jk}$ in I_j.

Let us assume that in the j-th iteration we have a set of joint samples in the training data I_j. In the $(j+1)$-th iteration, we want to add a joint sample to I_j using FPS. Straightforwardly, one will consider to implement Algorithm 14 to achieve this step. However, Algorithm 14 is very inefficient in practice. If we assume there are m joint

Algorithm 14 Vanilla FPS

Require: I_j, Δs, I_{all}
1: $\Delta I = \{\}$
2: $D_j = \{\}$
3: **for** $\boldsymbol{J}$ in $I_{\text{all}} \setminus I_j$ **do**
4: Compute $d(\boldsymbol{J}, I_j)$
5: $D_j = D_j \cup d(\boldsymbol{J}, I_j)$
6: **end for**
7: Find $\boldsymbol{J}_{\text{fp}}$ corresponding to $max(D_j)$
8: $\Delta I = \left\{ \boldsymbol{J}_{\text{fp}} \right\}$
9: **return** ΔI

samples in I_{all}, to update I_j, we need to evaluate Equation 6.59 $j(m-j)$ times. To develop a faster algorithm, let us compare I_{j-1} to I_j:

$$\begin{aligned} I_{j-1} &= \left\{ \boldsymbol{J}_1, \boldsymbol{J}_2, ..., \boldsymbol{J}_{j-1} \right\}, \\ I_j &= \left\{ \boldsymbol{J}_1, \boldsymbol{J}_2, ..., \boldsymbol{J}_{j-1}, \boldsymbol{J}_j \right\}. \end{aligned} \tag{6.61}$$

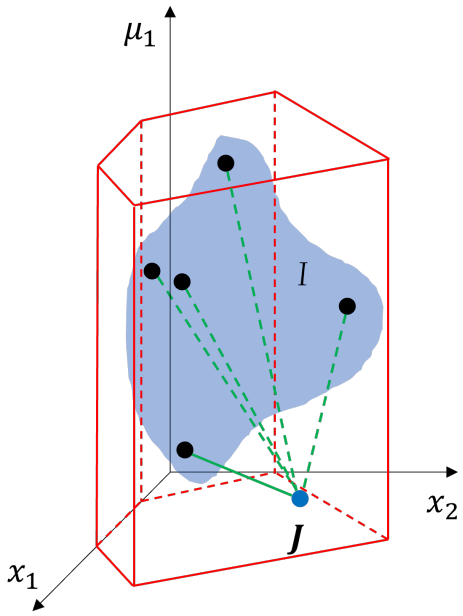


Figure 6.9.: The distance from a point to a point set is defined as the shortest distance between this point and any point in the point set. Blue point: the point. Black points in shade: the point set. Solid line: distance from the point to the point set.

As we know, to update from I_j to I_{j+1}, we need to calculate the distances between every sample in $I_{\text{all}} \setminus I_j$ to the sample set I_j. For the joint samples in $I_{\text{all}} \setminus I_j$, we have already calculated their distances to I_j while updating from I_{j-1} to I_j, except for $\boldsymbol{J}_j = I_j \setminus I_{j-1}$.

Therefore, we can improve Algorithm 14 by using a set D_{j-1} storing the point-to-set distance in the $(j-1)$-th iteration. The improved algorithm is presented in Algorithm 15. In Algorithm 15, to update I_j, we only need to evaluate Equation 6.59 $m - j$ times.

Algorithm 15 FPS

Require: I_j, I_{all}, D_{j-1}
1: $D_j = \{\}$
2: **for** $\boldsymbol{J}$ in $I_{\text{all}} \setminus I_j$ **do**
3: Compute $d_{\text{new}} = d(\boldsymbol{J}, \boldsymbol{J}_j)$
4: Denote the distance in D_{j-1} corresponding to $\boldsymbol{J}$ as d_{old}
5: $D_j = D_j \cup \{min(d_{\text{new}}, d_{\text{old}})\}$
6: **end for**
7: Find $\boldsymbol{J}_{\text{fp}}$ corresponding to $max(D_j)$
8: $\Delta I = \{\boldsymbol{J}_{\text{fp}}\}$
9: **return** ΔI

Finally, we can design the MOR algorithm based on PL as Algorithm 16. Compared to the active selection strategies, the passive selection strategy is independent to the status of the ROM. This allows us to sort all the samples in the initial data pool I_{all} up-front starting the iteration, which reduces the requirement for computer memory during the iterations. For sorting I_{all}, we can choose a random sample as initialization and sort the whole set iteratively using Algorithm 15. During the iteration, we just pick the first Δs samples from $I_{\text{all}} \setminus I_{j-1}$ as the new samples and add them to I_j. The PL method essentially selects samples based on their diversity to the current training data, we can therefore choose the size of the data batch freely without worrying about information redundancy.

6.5. Summary

In this section, another snapshot collection strategy DPS is introduced and analyzed. However, its own disadvantage is found in the analysis. In order to improve the disadvantages of SPS and DPS. A new sampling method is proposed, and it is named as JSS.

Algorithm 16 PL-MOR

Require: $\boldsymbol{V}$, I_{all}, Δs, τ^*_{design}, PAC, FOM

Sort I_{all} with Algorithm 15
Pick the first Δs samples from I_{all} as the initial snapshot set, denoted as I_1
$O_1 = \{\boldsymbol{y}_i | \boldsymbol{y}_i = \text{FOM}(\boldsymbol{J}_i, \boldsymbol{V}, \delta t),\ \text{for all } \boldsymbol{J}_i \in I_1\}$
Use I_1, O_1 and $\boldsymbol{V}$ to train ROM_1
$j = 1$
$p_0(\tau^*_{\text{design}}) = 0, \tau^*_0 = 1$
while NOT $\text{CONV}(p_j(\tau^*_{\text{design}}), \tau^*_j, p_{j-1}(\tau^*_{\text{design}}), \tau^*_{j-1})$ **do**
    $j = j + 1$
    Pick the first Δs samples from $I_{\text{all}} \setminus I_{j-1}$ as ΔI
    $I_j = I_{j-1} \cup \Delta I$
    $O_j = \left\{\boldsymbol{y}_i | \boldsymbol{y}_i = \text{FOM}(\boldsymbol{J}_i, \boldsymbol{V}, \delta t),\ \text{for all } \boldsymbol{J}_i \in I_j\right\}$
    Update $\boldsymbol{V}$
    Use I_j, O_j and $\boldsymbol{V}$ to train ROM_j
    $p_j(\tau^*_{\text{design}}), \tau^*_j, \boldsymbol{e}_{\text{PAC}} = \text{PAC}(\text{ROM}_j, \boldsymbol{V}, \tau^*_{\text{design}})$
end while
return ROM_j

Within the JSS method, we first estimate a reduced solution space where we can randomly take samples for reduced states. Lifting sampled reduced states to the full space will give us diverse full order states that compromise physics of the FOM. A joint space consisting of the estimated reduced solution space and the predefined parameter space can be created then. We can use samples taken from such a space as the input to the FOM and generate many one-step snapshots for identifying the ROM. Such snapshots are considered to have a better distribution compared to snapshots collected by the conventional methods.

For validating the trained ROM, two different methods for validation are introduced. Besides the conventional case-dependent method, the case-independent method based on PAC learning is proposed. With case-independent validation, we can know the confidence and the accuracy of the investigated ROM.

Finally, the active sample selection methods and the passive sample selection method are introduced. The active methods check the status of the current ROM to decide which unlabelled samples in the data pool should be labelled by the FOM. The method MMSE and QBC belong to this category. The passive sample selection method does not check the status of the ROM. Instead, the PL method measures samples' geometric characteristics and selects based on their distance information.

Part IV.

Experiments, discussions and conclusions

7. Numerical Experiments

In this chapter, we use numerical experiments to test different Model Order Reduction (MOR) methods. In the numerical experiments, two numerical models are used to perform these experiments. The first model is a thermal block with temperature-dependent conductivity. The conductivity has a rapid change in the high-temperature region. By observing the results of this simple but specially-designed model, we can understand why the proposed Active-Learning-based Model Order Reduction (AL-MOR) approach can construct a Reduced Order Model (ROM) better than the state-of-the-art approaches. The second model has relatively realistic settings. This numerical model is built based on a real radiation furnace. We will show the performance of different ROMs under realistic system inputs. The ROM of this Full Order Model (FOM) can be potentially used in industrial use cases such as Process Control, Virtual Sensor Monitoring, Optimal Control, etc.

7.1. Model I: thermal block with non-constant material property

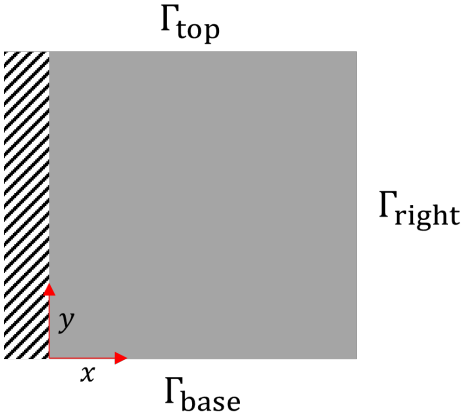


Figure 7.1.: Model I: the thermal block.

The first numerical model is heat conduction in a block area. As shown in Figure 7.1, we apply four boundary conditions to this area. On the left side of this area, we

prescribe a Neumann boundary condition that no heat flux is transferred from the area to the environment. On the top, right and base side of the area, we define three different Dirichlet boundary conditions. The temperature of each side will be fixed with a certain value. Based on the description of the problem, we can write down the governing equation for this thermal block model as:

$$\begin{aligned} &\frac{\partial T(x,y,t)}{\partial t} = k_x \frac{\partial^2 T(x,y,t)}{\partial x^2} + k_y \frac{\partial^2 T(x,y,t)}{\partial y^2}, \\ &T(x,10,t) = \Gamma_{\text{top}}, \qquad T(x,0,t) = \Gamma_{\text{base}}, \\ &\frac{\partial T(0,y,t)}{\partial t} = 0, \qquad T(10,y,t) = \Gamma_{\text{right}}, \end{aligned} \tag{7.1}$$

where $x \in (0,10), y \in (0,10), t \in (0,2]$.

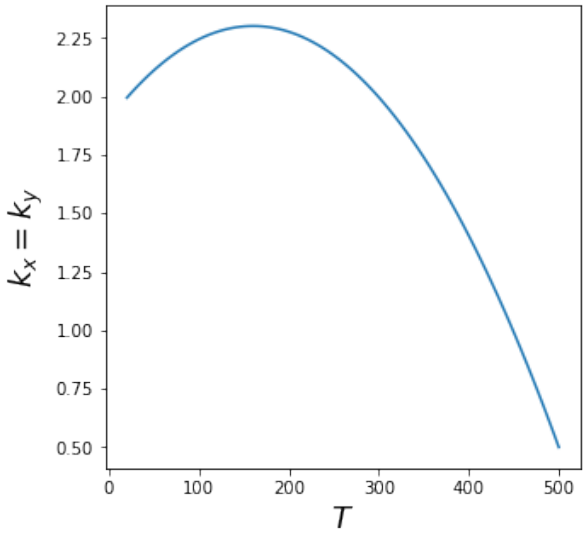


Figure 7.2.: The thermal conductivity of the material.

We further assume that the material of this block is isotropic and temperature-dependent, i.e.,:

$$k_x = k_y = 1.9 + 0.005T - 1.56 \times 10^{-5} T^2, T \in [20, 500] \tag{7.2}$$

which is graphically shown in Figure 7.2. As we see, the conductivity in the range $[20, 300]$ is relatively stable while having a fast drop in the range $[300, 500]$. This requires that the ROM must have a better performance throughout the whole temperature range.

The controllable parameters of this system are the values of the temperature boundary conditions, i.e. $\Gamma_{\text{top}}, \Gamma_{\text{base}}, \Gamma_{\text{right}}$. The range for each parameter is:

$$\Gamma_{\text{top}} \in [200, 500], \quad \Gamma_{\text{base}} \in [200, 500], \quad \Gamma_{\text{right}} \in [20, 200], \tag{7.3}$$

Then we use the Finite Difference Method [148] to discretize this block area with $50 \times 50 = 2500$ elements. The discrete form of the FOM governing equation is:

$$\boldsymbol{E}\frac{\partial \boldsymbol{T}}{\partial t} = \boldsymbol{K}(\boldsymbol{T})\boldsymbol{T} + \boldsymbol{B}(\boldsymbol{T})\boldsymbol{u}, \tag{7.4}$$

where $\boldsymbol{T}$ is the temperature state vector of the FOM, $\boldsymbol{E}$ is the thermal capacity matrix which is just an identity matrix, $\boldsymbol{K}(T)$ is the heat conduction matrix, $\boldsymbol{B}(T)$ is the equivalent heat load matrix due to the fixed temperature boundary conditions, and:

$$\boldsymbol{u} = \begin{bmatrix} \Gamma_{\text{top}} \\ \Gamma_{\text{base}} \\ \Gamma_{\text{right}} \end{bmatrix} \tag{7.5}$$

is the parameter vector storing the values of the boundary conditions.

Since we know the temperature-dependency of the material, Equation 7.4 can be re-written as:

$$\frac{\partial \boldsymbol{T}}{\partial t} = \boldsymbol{m}(\boldsymbol{T}) \odot (\boldsymbol{K}_0\boldsymbol{T} + \boldsymbol{B}_0\boldsymbol{u}), \tag{7.6}$$

where $\boldsymbol{m}(\boldsymbol{T})$ is the nonlinear term caused by the non-constant thermal conductivity. $\odot$ means component-wise product. $\boldsymbol{K}_0$ and $\boldsymbol{B}_0$ is the normalized $\boldsymbol{K}$ and $\boldsymbol{B}$.

7.1.1. Constructing the reduced space

First, we need to construct a Proper Orthogonal Decomposition (POD) space as introduced in section 2.3. According to the description above, we can construct the parameter space $\mathcal{M}$ as:

$$\mathcal{M} = [200, 500] \times [200, 500] \times [20, 200]\,. \tag{7.7}$$

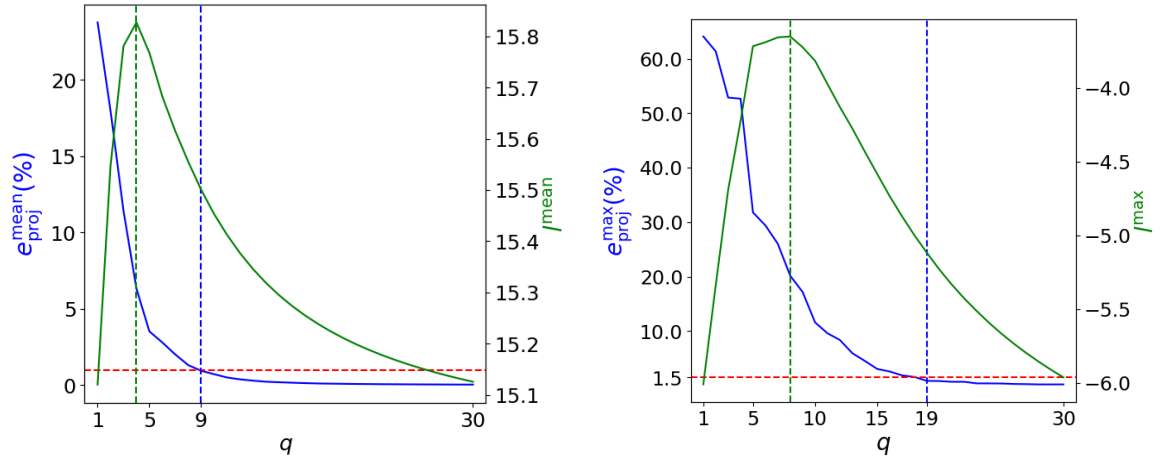
To build the snapshot matrix $\boldsymbol{\mathcal{T}}$, 30 samples are randomly taken from $\mathcal{M}$, and Static Parameter Sampling (SPS) will be used. To improve the sample distribution, here we employ Hammersley Sequence [48] to sample from this cubic parameter space. We denote the sampled parameter vectors as:

$$U = \{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_{30}\}\,. \tag{7.8}$$

We use these parameter vectors to construct 30 Initial Value Problems (IVPs). For each IVP we simulate in the time span $t \in (0,\ 2]$ with $\delta t = 2e - 2$.

To determine the size of the ROM, we gradually increase the number of reduced basis vectors in $\boldsymbol{V}$ from 1 to 30 and compute the mean and the maximum relative error of projection.

The corresponding results are shown in Figure 7.3, we use the method introduced in section 5.1 to select our ROM size. We pre-define $e_{\text{tol}}^{\text{mean}} = 1\%$ and $e_{\text{tol}}^{\text{max}} = 1.5\%$, and Algorithm 7 helps us choose a size of 19. Based on the decision, we use the first 19 singular vectors from Singular Value Decomposition (SVD) to build the reduced space for this problem.



(a) $e_{\text{proj}}^{\text{mean}}(q)$ and $l^{\text{mean}}(q)$.

(b) $e_{\text{proj}}^{\text{max}}(q)$ and $l^{\text{max}}(q)$.

Figure 7.3.: The curves for $e_{\text{proj}}(q)$ and $l(q)$. The red dashed lines in Figure 7.3a and Figure 7.3b stand for $e_{\text{tol}}^{\text{mean}} = 1\%$ and $e_{\text{tol}}^{\text{max}} = 1.5\%$. The green dashed lines mark the q maximizing $l(q)$. The blue dashed lines mark the minimum q satisfying $e_{\text{proj}}(q) \leq e_{\text{tol}}$. The optimal ROM size is selected to be $N_r = 19$ for this FOM.

7.1.2. Application of DEIM

In this section, we demonstrate how to apply the Discrete Empirical Interpolation Method (DEIM) to reduce this model.

Recalling the governing equation we conclude for the FOM:

$$\frac{\partial \boldsymbol{T}}{\partial t} = \boldsymbol{m}(\boldsymbol{T}) \odot (\boldsymbol{K}_0 \boldsymbol{T} + \boldsymbol{B}_0 \boldsymbol{u}). \tag{7.9}$$

Using the reduced basis $\boldsymbol{V}$ constructed in section 7.1.1, we can re-write Equation 7.9 as:

$$\boldsymbol{V}^T \boldsymbol{V} \frac{\partial \boldsymbol{T}_r}{\partial t} = \boldsymbol{V}^T \boldsymbol{m}(\boldsymbol{V}\boldsymbol{T}_r) \odot (\boldsymbol{K}_0 \boldsymbol{V}\boldsymbol{T}_r + \boldsymbol{B}_0 \boldsymbol{u}). \tag{7.10}$$

Then by substituting $\boldsymbol{V}^T \boldsymbol{V} \approx \boldsymbol{1}$, we get:

$$\frac{\partial \boldsymbol{T}_r}{\partial t} = \boldsymbol{V}^T \boldsymbol{m}(\boldsymbol{V}\boldsymbol{T}_r) \odot (\boldsymbol{K}_0 \boldsymbol{V}\boldsymbol{T}_r + \boldsymbol{B}_0 \boldsymbol{u}). \tag{7.11}$$

We will take the whole right hand side (RHS) as the nonlinear term and build the nonlinear snapshot matrix $\boldsymbol{F}$ as:

$$\boldsymbol{F} = \boldsymbol{m}(\boldsymbol{\mathcal{T}}) \odot (\boldsymbol{K}_0 \boldsymbol{\mathcal{T}} + \boldsymbol{B}_0 \boldsymbol{\mathcal{U}}), \tag{7.12}$$

where $\boldsymbol{\mathcal{U}}$ is a matrix consisting of the parameter vectors assigned to each time step in each IVP, i.e.:

$$\boldsymbol{\mathcal{U}} = \big[\underbrace{u_1 \quad \dots \quad u_1}_{100} \quad \underbrace{u_2 \quad \dots \quad u_2}_{100} \quad \dots \quad \underbrace{u_{30} \quad \dots \quad u_{30}}_{100}\big]. \tag{7.13}$$

Then by applying SVD to $\boldsymbol{F}$, we can get the DEIM basis:

$$\boldsymbol{H} = \begin{bmatrix} h_1 & h_2 & \dots & h_{50} \end{bmatrix}, \tag{7.14}$$

where we set the size of the DEIM space to be 50. Using Equation 3.8, we can write the RHS of Equation 7.9 as:

$$\boldsymbol{H}(\boldsymbol{P}^T\boldsymbol{H})^{-1}\boldsymbol{P}^T \left[\boldsymbol{m}(\boldsymbol{T}) \odot (\boldsymbol{K}_0 \boldsymbol{T} + \boldsymbol{B}_0 \boldsymbol{u})\right]. \tag{7.15}$$

To know how to perform the next step, we focus on the term:

$$\boldsymbol{P}^T \boldsymbol{m}(\boldsymbol{T}) \odot (\boldsymbol{K}_0 \boldsymbol{T}). \tag{7.16}$$

We use Figure 7.4 to explain the point that needs attention. After constructing $\boldsymbol{P}$,

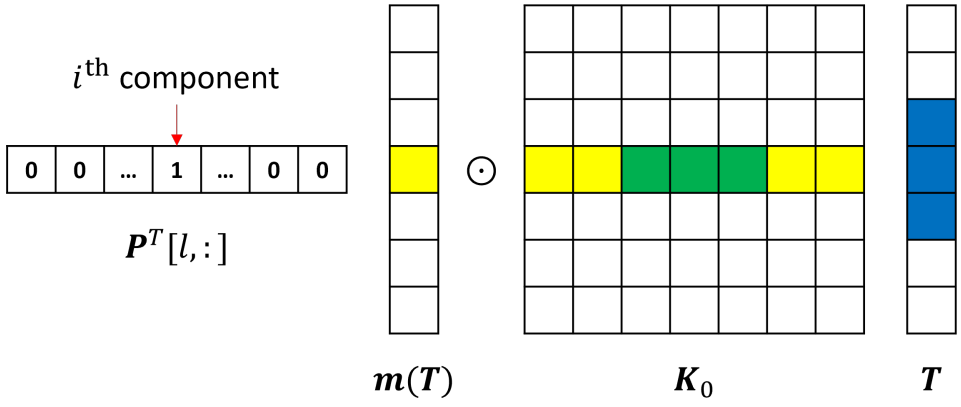


Figure 7.4.: Explanation to DEIM DOFs and their coupled DOFs. The yellow DOF in $\boldsymbol{m}(\boldsymbol{T})$ is selected by the DEIM algorithm. The green entries are non-zero entries in the corresponding row in the matrix $\boldsymbol{K}$. The blue DOFs in $\boldsymbol{T}$ are all DOFs coupled to the DOF selected by the DEIM algorithm.

we look at the l-th row of $\boldsymbol{P}^T$ whose i-th entry is 1. We know that the i-th degree

of freedom (DOF) is recognized as the DOF where we preserve the nonlinearity, i.e. where we should get the exact evaluation of the nonlinearity during the online phase. Then we know, in $\boldsymbol{m}(\boldsymbol{T})$, the i-th row needs to be kept. Correspondingly, the i-th row in the result of $\boldsymbol{K}_0\boldsymbol{T}$ is needed. To compute this contribution, we need:

$$(\boldsymbol{K}_0\boldsymbol{T})[i] = \sum_{j=1}^{N} K_{0,ij} T_j, \tag{7.17}$$

where $K_{0,ij}$ means the entry in the i-th row and the j-th column of $\boldsymbol{K}_0$. Since $\boldsymbol{K}_0$ is usually sparse, we only need to take its non-zero terms into account. If we denote the positions of the non-zero terms in the i-th row as:

$$\Phi_i = \left\{\phi_{i1}, \phi_{i2}, \ldots, \phi_{ik_i}\right\}, \tag{7.18}$$

where k_i is the number of the non-zero entries in the i-th row. Equation 7.17 is equivalent to:

$$(\boldsymbol{K}_0\boldsymbol{T})[i] = \sum_{j\in\Phi_i} K_{0,ij} T_j. \tag{7.19}$$

Recalling we have defined $D = \{d_1, d_2, \ldots, d_{50}\}$ as the interpolation points, by scanning $\boldsymbol{K}_0$, we can construct:

$$\begin{aligned} \Phi_1 &= \left\{\phi_{11}, \phi_{12}, \ldots, \phi_{1k_1}\right\}, \\ \Phi_2 &= \left\{\phi_{21}, \phi_{22}, \ldots, \phi_{2k_2}\right\}, \\ &\vdots \\ \Phi_{50} &= \left\{\phi_{501}, \phi_{502}, \ldots, \phi_{50k_{50}}\right\} \end{aligned} \tag{7.20}$$

for them. We let:

$$\Phi = \Phi_1 \cup \Phi_2 \cup \cdots \cup \Phi_{50} \tag{7.21}$$

$$= \{\phi_1, \phi_2, \ldots, \phi_k\}, \tag{7.22}$$

where k is the number of coupled DOFs. We further construct a matrix:

$$\boldsymbol{P}_{\text{couple}} = \begin{bmatrix} \boldsymbol{p}_{\text{couple},1} & \boldsymbol{p}_{\text{couple},2} & \cdots & \boldsymbol{p}_{\text{couple},k} \end{bmatrix}, \tag{7.23}$$

where:

$$\boldsymbol{p}_{\text{couple},i} = \begin{bmatrix} 0 & 0 & \ldots & 1 & \ldots & 0 & 0 \end{bmatrix}^T \tag{7.24}$$

is a vector whose components are all 0 except the ϕ_i-th component is 1, $\forall \phi_i \in \Phi$. Then we can define:

$$\boldsymbol{K}_{\text{couple}} = \boldsymbol{K}_0[:, \Phi] \tag{7.25}$$

where $\boldsymbol{K}_0[:, \Phi]$ is a sub-matrix whose columns are picked as the ϕ_i-th column of $\boldsymbol{K}_0$, $\forall \phi_i \in \Phi$.

Since there is no coupling effects existing in $\boldsymbol{B}$, we can directly define:

$$\boldsymbol{B}_{\text{DEIM}} = \boldsymbol{P}^T \boldsymbol{B}_0. \tag{7.26}$$

Therefore, Equation 7.15 can be transformed into:

$$\begin{aligned} &\boldsymbol{H}(\boldsymbol{P}^T\boldsymbol{H})^{-1}\boldsymbol{P}^T\left[\boldsymbol{m}(\boldsymbol{T}) \odot (\boldsymbol{K}_0\boldsymbol{T} + \boldsymbol{B}_0\boldsymbol{u})\right] \\ &= \boldsymbol{H}(\boldsymbol{P}^T\boldsymbol{H})^{-1}\left[\boldsymbol{m}(\boldsymbol{P}^T\boldsymbol{V}\boldsymbol{T}_r) \odot (\boldsymbol{P}^T\boldsymbol{K}_{\text{couple}}\boldsymbol{P}_{\text{couple}}^T\boldsymbol{V}\boldsymbol{T}_r + \boldsymbol{B}_{\text{DEIM}}\boldsymbol{u})\right] \\ &= \boldsymbol{H}(\boldsymbol{P}^T\boldsymbol{H})^{-1}\left[\boldsymbol{m}(\boldsymbol{P}^T\boldsymbol{V}\boldsymbol{T}_r) \odot (\boldsymbol{K}_{\text{DEIM}}\boldsymbol{T}_r + \boldsymbol{B}_{\text{DEIM}}\boldsymbol{u})\right], \end{aligned} \tag{7.27}$$

where $\boldsymbol{K}_{\text{DEIM}} = \boldsymbol{P}^T\boldsymbol{K}_{\text{couple}}\boldsymbol{P}_{\text{couple}}^T\boldsymbol{V}$. Together with Equation 7.11, we finally get:

$$\frac{\partial \boldsymbol{T}_r}{\partial t} = \boldsymbol{V}^T\boldsymbol{H}(\boldsymbol{P}^T\boldsymbol{H})^{-1}\left[\boldsymbol{m}(\boldsymbol{P}^T\boldsymbol{V}\boldsymbol{T}_r) \odot (\boldsymbol{K}_{\text{DEIM}}\boldsymbol{T}_r + \boldsymbol{B}_{\text{DEIM}}\boldsymbol{u})\right]. \tag{7.28}$$

Ahead of going to the online phase, we can pre-compute $\boldsymbol{W} = \boldsymbol{V}^T\boldsymbol{H}(\boldsymbol{P}^T\boldsymbol{H})^{-1}, \boldsymbol{V}_{\text{DEIM}} = \boldsymbol{P}^T\boldsymbol{V}, \boldsymbol{K}_{\text{DEIM}}$ and $\boldsymbol{B}_{\text{DEIM}}$. Therefore, during the online phase, we only need to solve:

$$\frac{\partial \boldsymbol{T}_r}{\partial t} = \boldsymbol{W}\left[\boldsymbol{m}(\boldsymbol{V}_{\text{DEIM}}\boldsymbol{T}_r) \odot (\boldsymbol{K}_{\text{DEIM}}\boldsymbol{T}_r + \boldsymbol{B}_{\text{DEIM}}\boldsymbol{u})\right]. \tag{7.29}$$

7.1.3. Application of the ANN

As introduced in section 3.2, we can use the Artificial Neural Network (ANN) to identify the ROM from the FOM snapshots. Here in this thesis, we will use the Runge-Kutta Neural Network (RKNN) as the fundamental architecture to construct the network. The core Multilayer Perceptron (MLP) inside the RKNN structure has 4 layers, including an input layer and an output layer. The number of neurons in each layer is $[22, 88, 88, 19]$. The input layer has 22 neurons because the size of our reduced state is 19 and the number of the system parameters is 3. The activation function $\mathcal{A}(\cdot)$ is the Rectified Linear Unit (ReLU) function [149]. The ReLU function has the mathematical expression:

$$\mathcal{A}(wz_{\text{in}} + b) = max(0, wz_{\text{in}} + b), \tag{7.30}$$

as shown in Figure 7.5.

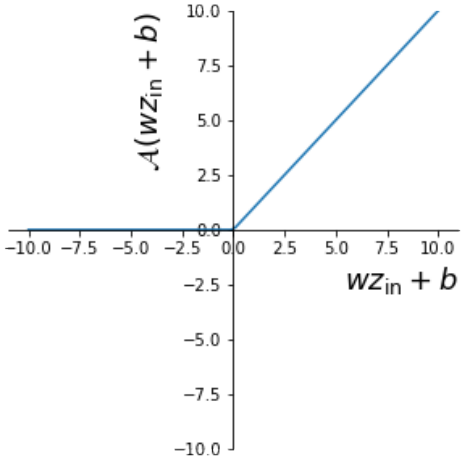


Figure 7.5.: The ReLU function.

7.1.4. Application of Operator Inference

To apply Operator Inference (OpInf), we need to first introduce some auxiliary variables to enable a quadratic description for the FOM. We define the lifted state vector as:

$$\boldsymbol{s} = \begin{bmatrix} \boldsymbol{s}_1 = \boldsymbol{T} \\ \boldsymbol{s}_2 = \boldsymbol{T}^2 \end{bmatrix}. \tag{7.31}$$

Taking all variables into consideration and using the Chain Rule, Equation 7.9 becomes:

$$\begin{aligned} \frac{\partial \boldsymbol{s}_1}{\partial t} =& (1.9 + 0.005\boldsymbol{s}_1 - 1.56 \times 10^{-5}\boldsymbol{s}_2) \odot (\boldsymbol{K}_0\boldsymbol{s}_1 + \boldsymbol{B}_0\boldsymbol{u}), \\ \frac{\partial \boldsymbol{s}_2}{\partial t} =& 2\boldsymbol{s}_1(1.9 + 0.005\boldsymbol{s}_1 - 1.56 \times 10^{-5}\boldsymbol{s}_2) \odot (\boldsymbol{K}_0\boldsymbol{s}_1 + \boldsymbol{B}_0\boldsymbol{u}) \\ =& (3.8\boldsymbol{s}_1 + 0.01\boldsymbol{s}_2 - 3.12 \times 10^{-5}\boldsymbol{s}_1\boldsymbol{s}_2) \odot (\boldsymbol{K}_0\boldsymbol{s}_1 + \boldsymbol{B}_0\boldsymbol{u}). \end{aligned} \tag{7.32}$$

Notice that the term $\boldsymbol{s}_1\boldsymbol{s}_2 \odot \boldsymbol{s}_1$ in the third line of Equation 7.32 can be considered as $\boldsymbol{s}_2 \odot \boldsymbol{s}_2$. Therefore, the RHS of Equation 7.32 contains only polynomial terms up to quadratic and can be summarized as:

$$\frac{\partial \boldsymbol{\omega}}{\partial t} = \boldsymbol{F}_1\boldsymbol{\omega} + \boldsymbol{F}_2\boldsymbol{\omega} \otimes \boldsymbol{\omega} + \boldsymbol{F}_B\boldsymbol{u} + \boldsymbol{F}_N\boldsymbol{u} \otimes \boldsymbol{\omega} + \tilde{\boldsymbol{F}}_N\boldsymbol{u} \otimes \boldsymbol{\omega} \otimes \boldsymbol{\omega}. \tag{7.33}$$

As a result, while preparing snapshots, we need to directly get the snapshots for $\boldsymbol{s}_1 = \boldsymbol{T}$ and compute the auxiliary variables $\boldsymbol{s}_2$. The size of the reduced lifted state $\boldsymbol{\omega}$ is $N_r = 19 \times 2 = 38$.

7.1.5. Study on data sampling strategies

In this test, we focus on studying the influence from data sampling methods. We prepare three data pools, and each of them has 20,000 training samples. They are created by SPS, Dynamic Parameter Sampling (DPS) and Joint Space Sampling (JSS), respectively. For the JSS method, we use the snapshot matrix $\boldsymbol{\mathcal{T}}$ collected in section 7.1.1 as $\boldsymbol{\mathcal{T}}_{\text{estimate}}$. Besides, we choose $\beta = 0.1$, $T_{\text{max}} = \text{MAX}(\boldsymbol{\mathcal{T}}_{\text{estimate}})$ and $T_{\text{min}} = \text{MIN}(\boldsymbol{\mathcal{T}}_{\text{estimate}})$ to loosen and trim the joint space.

To validate the ROM, we construct a Probably Approximately Correct (PAC) validator. The PAC validator is designed as:

$$\begin{aligned} &\epsilon = 0.03,\ \sigma = 0.01, \\ &N_s = \frac{1}{2 \times 0.03^2} ln\left(\frac{2}{0.01}\right) = 2944, \end{aligned} \tag{7.34}$$

which means the investigated ROM confidence level is $1 - 3\% = 97\%$, and its validation result is $1 - 1\% = 99\%$ reliable. For building this validator, 2,944 joint samples $\boldsymbol{J}$ are collected from the joint space.

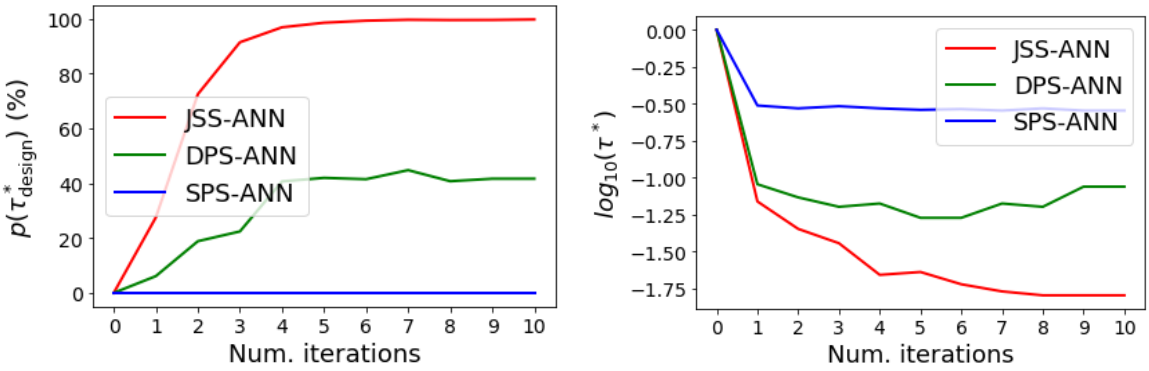
Notation for ROMs We first introduce the notation used to distinguish the ROMs constructed in different approaches:

- SPS-ANN/OpInf: a ROM identified by ANN/OpInf. Using the data randomly picked from a data pool constructed by SPS-sampled data.
- DPS-ANN/OpInf: a ROM identified by ANN/OpInf. Using the data randomly picked from a data pool constructed by DPS-samples data.
- JSS-ANN/OpInf: a ROM identified by ANN/OpInf. Using the data randomly picked from a data pool constructed by JSS-sampled data.

Results In each iteration, we *randomly* draw some samples from each data pools and add to their current training data, respectively. We train/update the ROMs with the extended training data and compute their PAC scores. The validation results are given in Figure 7.6 and Figure 7.7.

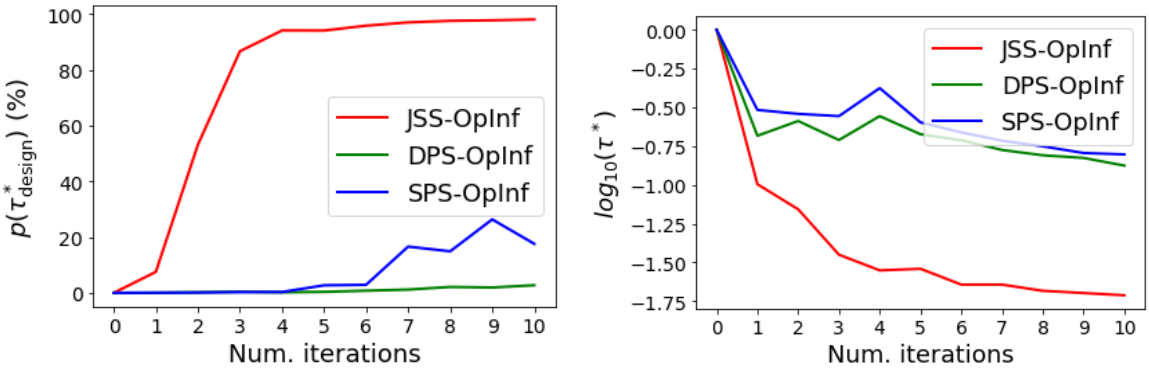
For the ANN-based approach, after refining ROM with the JSS data for 10 iterations, where the snapshot increment $\Delta s = 300$, the 97%-confident ROM error is 1.67%. With the same amount of training data, the SPS-ANN has a 97%-confident ROM error of 28.43% and the DPS-ANN has a 97%-confident ROM error of 8.70%.

For the OpInf-ROMs, after refining ROM with the JSS data for 10 iterations, where the snapshot increment $\Delta s = 25$, its 97%-confident ROM error is 1.67%. Using the



(a) $p(\tau^*_{\text{design}})$. (b) τ^*.

Figure 7.6.: The PAC scores of the ANN-ROMs change with the extension of the training data. The snapshot increment $\Delta s = 300$. The JSS-ANN has the lowest ROM error τ^* at the investigated confidence level, while the DPS-ANN takes the second place and the SPS-ANN is the worst.



(a) $p(\tau^*_{\text{design}})$. (b) τ^*.

Figure 7.7.: The PAC scores of the OpInf-ROMs change with the extension of the training data. The snapshot increment $\Delta s = 25$. The JSS-OpInf is the best ROM, which has the lowest ROM error τ^* at the investigated confidence level. The DPS-OpInf and the SPS-OpInf have similar τ^* at the investigated confidence level, but the SPS-OpInf has higher observed confidence $p(\tau^*_{\text{design}})$ for the desired accuracy. Therefore, the SPS-OpInf is considered to be better than the DPS-OpInf.

same amount of training data, the 97%-confident ROM error of the SPS-OpInf and the DPS-OpInf is 15.72% and 13.32%, respectively.

It is observed that just by using the JSS method to create the data pool, we can get a huge improvement for the ROM's confidence and accuracy with the same amount of training data. Then, in the next test, we will study how many additional benefits we can gain by employing the concept of Active Learning (AL).

7.1.6. Study on snapshot selection strategies

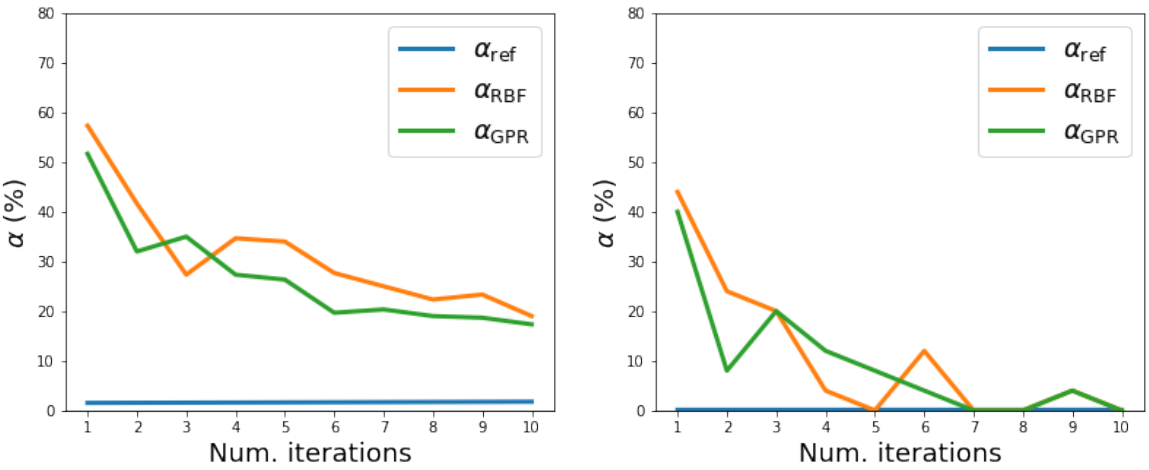
To determine the framework for the AL algorithm, we need to decide what kind of snapshot selection strategy we want to use based on their effectiveness.

Recalling in section 6.4.1, three different sample selection strategies are introduced, including the Maximum Mean Squared Error (MMSE), Query by Committee (QBC) and Passive Learning (PL). Moreover, for the MMSE method, we will investigate using the Radial Basis Function (RBF) interpolation and Gaussian Process Regression (GPR) as the data-driven error estimator, respectively. Besides using the estimated error for the MMSE method, we also use the ROM to compute its errors for all samples remaining in the data pool and pick those generating the maximal errors. This method is unrealistic in the online phase, but here it can serve as the reference for the MMSE methods.

Notation for selection methods We first introduce the notation used to distinguish different sample selection methods:

- Random: randomly picking from a data pool constructed by the samples from JSS.
- MMSE_{RBF}: the samples are actively selected from the JSS data pool, which is constructed by Algorithm 8. The selection strategy is MMSE in section 6.4.1.1. The error estimator is based on RBF interpolation.
- MMSE_{GPR}: the samples are actively selected from the JSS data pool. The selection strategy is MMSE. The error estimator is based on GPR.
- $\text{MMSE}_{\text{True}}$: the samples are actively selected from the JSS data pool. The selection strategy is MMSE. The errors are exactly computed using the ROM and the FOM.
- QBC: the samples are actively selected from the JSS data pool. The selection strategy is QBC in section 6.4.1.3.
- PL: the data are passively selected from the JSS data pool. The selection strategy is PL, i.e., Farthest Point Sampling (FPS) in Algorithm 16.

Results We first investigate the accuracy of error estimation. For both ROM identification methods, we use 10 iterations to build the ROM. The curves of the agreement ratio α in these iterations are presented in Figure 7.8.



(a) For training ANN. $\Delta s = 300$.

(b) For training OpInf. $\Delta s = 25$

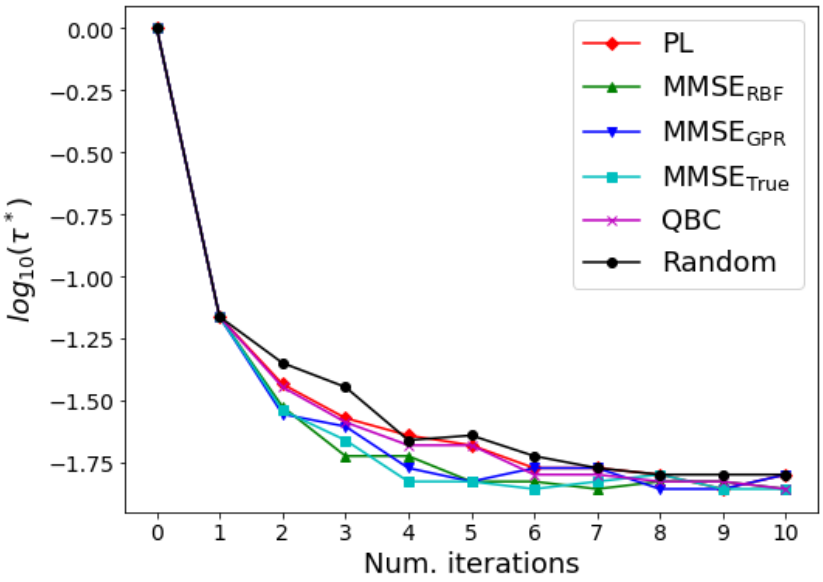
Figure 7.8.: The curves of the agreement ratio α in the first 10 iterations for the ROMs. The error estimators are fitted using the 2944 samples in PAC validation. For both ROMs, we see a clear trend: at the beginning, α is higher. But with the ROMs being refined, α drops quickly.

A very clear trend observed in Figure 7.8 is that α is descending within the iterations. As we introduced in section 6.4.1.2, α is the empirical accuracy of the error estimator. Therefore, based on the descending α, we can get a conclusion that the ROM error becomes more and more difficult to be predicted during the iterations. We also see that in most situations, α_{RBF} and α_{GPR} is higher than α_{ref}, which means the samples selected by the error estimators are more informative in terms of maximizing the Mean Squared Error (MSE) of the ROM.

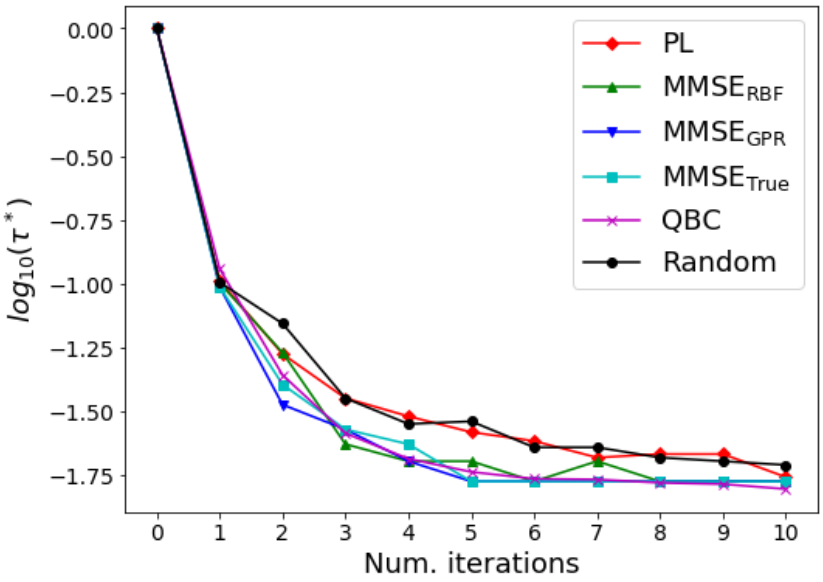
Then we generate the convergence curves of the 97%-confident ROM errors (τ^*) for each method. The curves are given in Figure 7.9.

As we can observe in Figure 7.9a, almost all non-random selection strategies can accelerate the convergence of ROM construction, except applying PL to OpInf. In Figure 7.9b, acceleration brought by PL is nearly ignoreable. Therefore, we consider PL as the least efficient among all the selection methods.

The MMSE method based on the true MSE has the most stable performance. This promising result tells us that if we can find a good approach to estimate the MSE accurately, using MMSE-based selection is very beneficial to the construction of the ROM. As for the MMSE methods based on the estimated MSE, they can generally



(a) ANN. $\Delta s = 300$.



(b) OpInf. $\Delta s = 25$.

Figure 7.9.: 97%-confident τ^* of the ROMs trained by five different methods. All methods draw samples from the same initial data pool, which is create by JSS.

speed up the convergence, but the stability of the acceleration is not as good as the $\text{MMSE}_{\text{True}}$ method.

Besides, the uncertainty-based method QBC also shows good performance. In Figure 7.9b, it is even more effective than the $\text{MMSE}_{\text{True}}$ method in accelerating the construction of the OpInf-ROM. In the following case-dependent validation, we will use the ROM constructed by the QBC method.

7.1.7. Case-dependent validation results

Besides case-independent validation, we also test the ROMs with 2 specific test cases. Four points in the thermal block are assumed to be the positions for virtual sensors (Figure 7.10). Their coordinates are: $p_1 = (1,5)$, $p_2 = (5,1)$, $p_3 = (9,5)$ and $p_4 = (5,9)$.

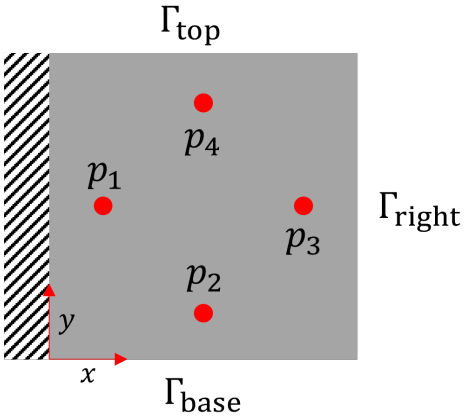


Figure 7.10.: Virtual sensor positions in the thermal block. $p_1 = (1,5)$, $p_2 = (5,1)$, $p_3 = (9,5)$ and $p_4 = (5,9)$.

Notation for ROMs As before, we first clarify the notation for different ROMs.

- DEIM: a ROM created by the hyper-reduction method DEIM.
- SPS-ANN/OpInf: a ROM identified with ANN/OpInf. The training data are picked from a data pool created with the SPS method.
- DPS-ANN/OpInf: a ROM identified with ANN/OpInf. The training data are picked from a data pool created with the DPS method.
- AL-ANN/OpInf: a ROM identified with ANN/OpInf. The training data are actively picked from a data pool created with the JSS method. The AL method is QBC.

7.1.7.1. Test Case 1

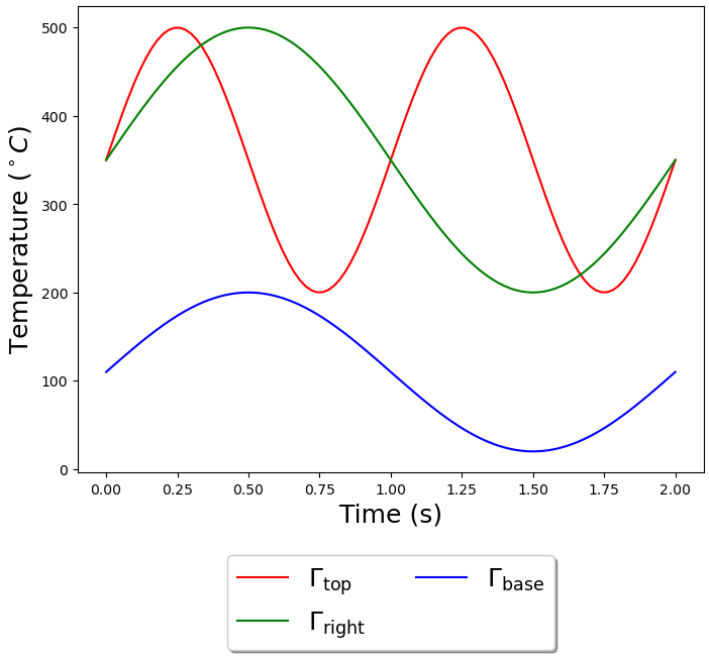


Figure 7.11.: The temperature boundary conditions used in Test Case 1.

In Test Case 1, we set the time-dependent functions of the temperature boundary conditions as Figure 7.11. The time-dependent functions are:

$$\begin{aligned} \Gamma_{\text{top}}(t) &= 350 + 150sin(2\pi t), \\ \Gamma_{\text{right}}(t) &= 350 + 150sin(\pi t), \\ \Gamma_{\text{base}}(t) &= 110 + 90sin(\pi t). \end{aligned} \tag{7.35}$$

In this test case, we use the sinus functions as the time dependent functions for the system's parameters to design a scenario where the input parameters are frequently changed during the simulation time. Through this test, we can evaluate our ROMs' exploration in the parameter space $\mathcal{M}$. The test results are presented in Table 7.1, Figure 7.12, Figure 7.13, Figure 7.15 and Figure 7.16.

For the ANN-based ROMs, the same as declared in section 6.1, the ROM built by SPS performs bad facing complex dynamic parameters. As its upgraded version, DPS snapshots can build a ROM with a better accuracy, which can catch the main trend of the FOM trajectories. This means in the training data sampled by the DPS strategy, we already have a better sample diversity for the system's parameters. But with better sample diversity in both the parameter space and the reduced solution

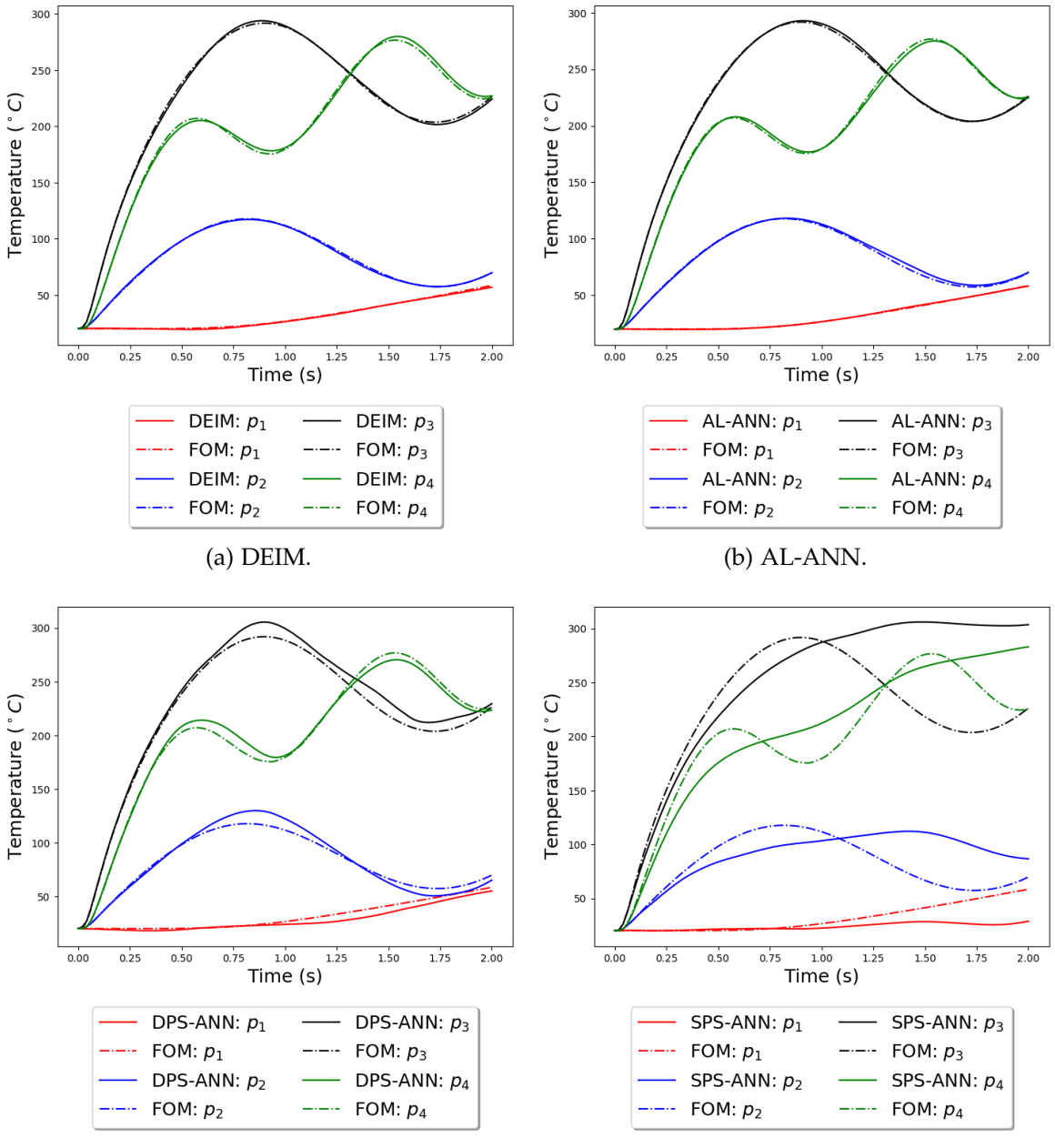
space, the AL-ANN is still leading. The prediction made by the AL-ANN is smooth and has the best agreement to the FOM solution.

But in the results of the OpInf-based ROMs (Figure 7.15 and Figure 7.16), the difference caused by using different training data is small. All the three ROMs can produce accurate prediction.

As introduced in section 3.3, an OpInf-ROM contains hypotheses to full order and reduced governing equations. The physics-informed hypotheses can restrict the flexibility of the Machine Learning (ML) model. As discussed in [150], the flexibility and the robustness of a ML model are usually two conflicting properties. A robust ML model can predict for the new samples which are not directly included in its training samples, which is also called extrapolation. For this problem, the ANN model is over-flexible and less robust. Therefore, we need the training data to be well-distributed in the parameter space and the reduced solution space. But for the OpInf-ROMs, it is appropriately flexible and more robust. This leads to the fact that the resulting models can extrapolate better in both spaces.

	DEIM	AL-ANN	DPS-ANN	SPS-ANN	AL-OpInf	DPS-OpInf	SPS-OpInf
Mean (°C)	0.97	1.09	5.18	23.31	1.22	1.02	0.91
Std. ((°C)^2)	0.86	1.28	4.74	33.75	1.58	1.55	1.08
Min. (°C)	1.13e-06	1.92e-07	2.29e-05	1.54e-06	4.80e-07	1.29e-06	1.54e-06
Max. (°C)	4.83	9.73	31.81	202.19	12.26	14.22	8.63
Median (°C)	0.72	0.63	3.90	10.67	0.60	0.47	0.50

Table 7.1.: The statistical measures for e_{abs} in Test Case 1 of the thermal block model.



(a) DEIM.

(b) AL-ANN.

(c) DPS-ANN.

(d) SPS-ANN.

Figure 7.12.: The comparison between the solution trajectories of the FOM, DEIM, and ANN-based ROMs at: $p_1 = (1,5)$, $p_2 = (5,1)$, $p_3 = (9,5)$ and $p_4 = (5,9)$ in Test Case 1 of the thermal block model. $N_r = 19$, $N_m = 50$, $N_s = 3000$. With the same amount of data, the prediction accuracy of the AL-ANN is the closest to the intrusive hyper-reduction method DEIM. The DPS-ANN can catch the main trend but its accuracy is much worse. The SPS-ANN has the worst performance.

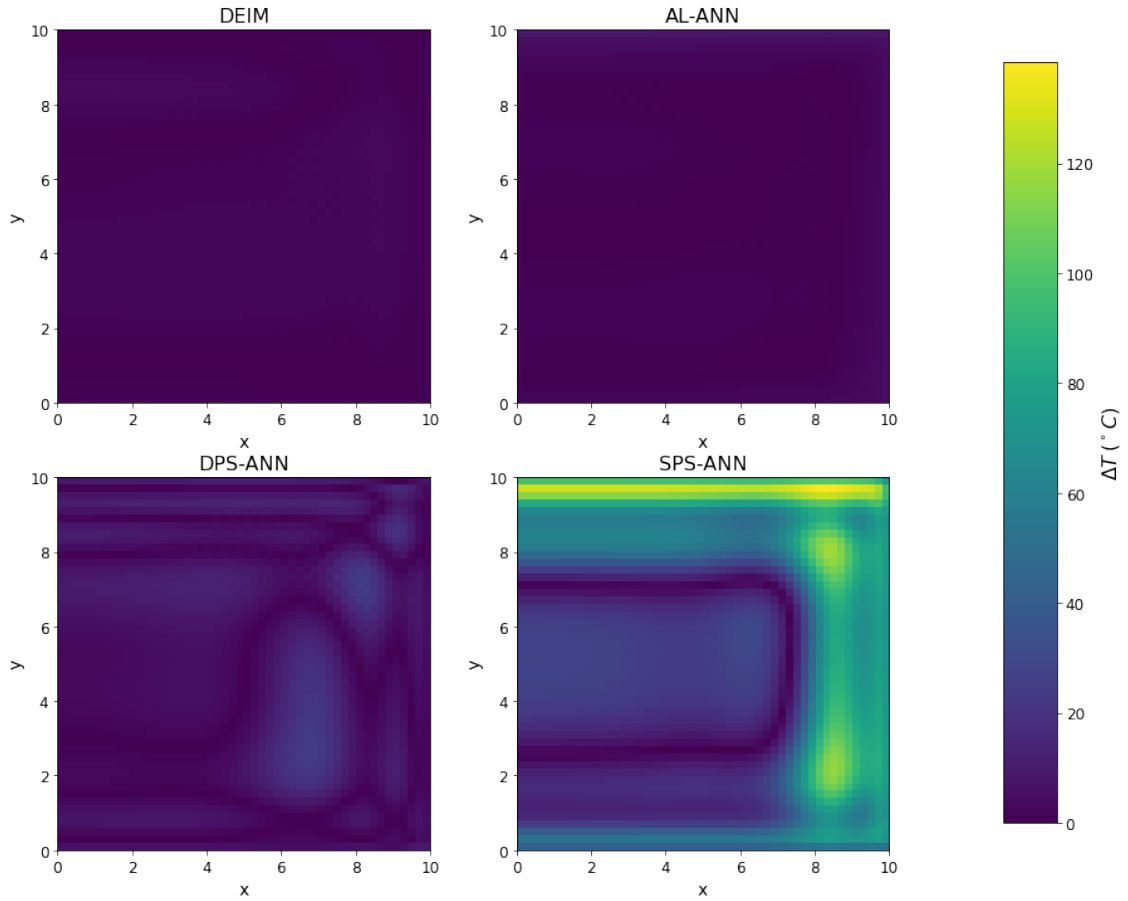


Figure 7.13.: The comparison between the error fields of (1) DEIM: upper-left (2) AL-ANN: upper-right (3) DPS-ANN: lower-left (4) SPS-ANN: lower-right, at: $t = 2$, in Test Case 1 of the thermal block model. The error field of DEIM and AL-ANN has similar error magnitude. In the error field of DPS-ANN, some bright areas are observed, where the ROM has relatively large error. The error field of SPS-ANN is much brighter in general compared to other error fields, which means SPS-ANN has the worst prediction throughout the whole geometry.

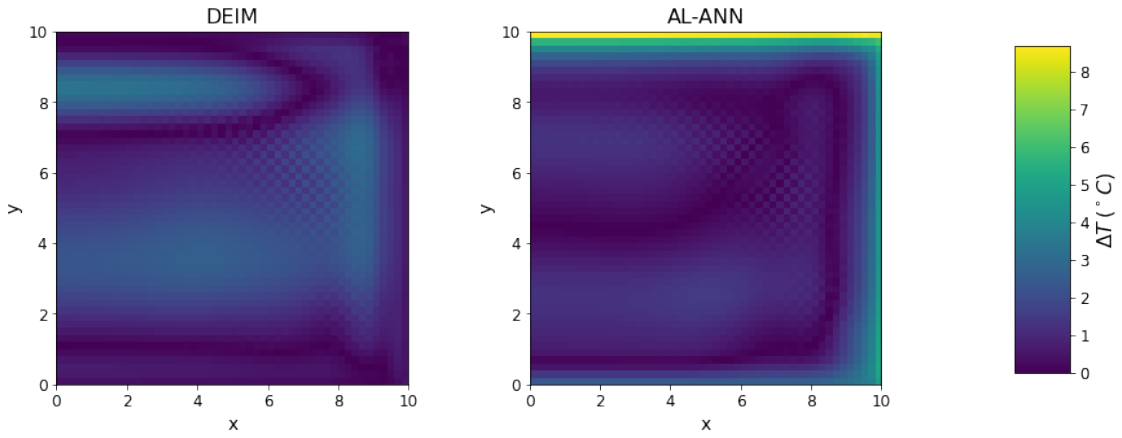
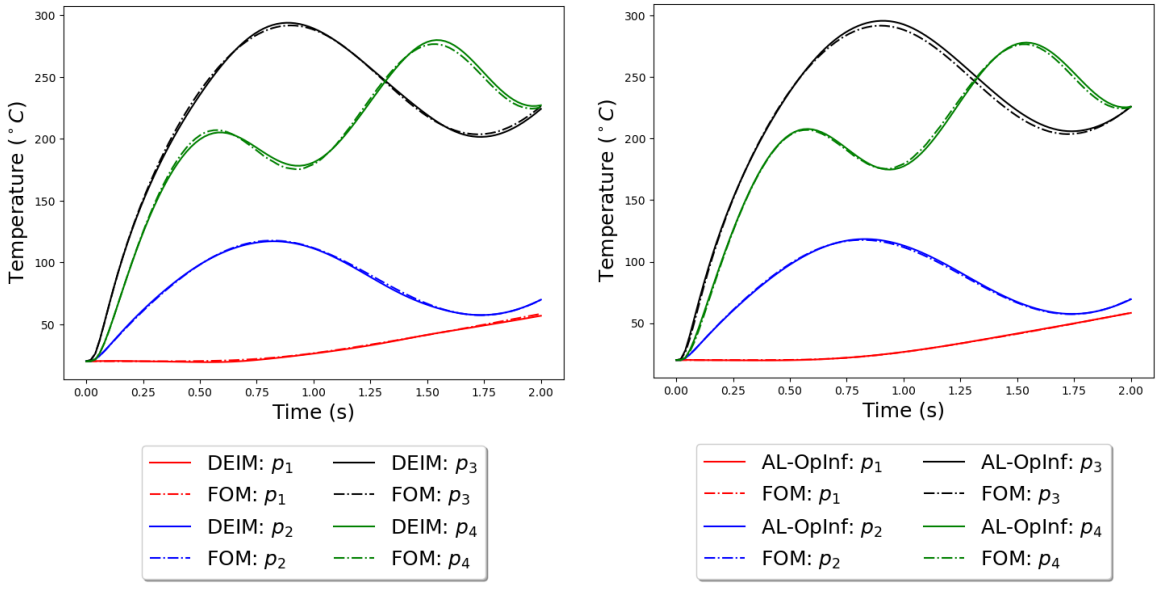
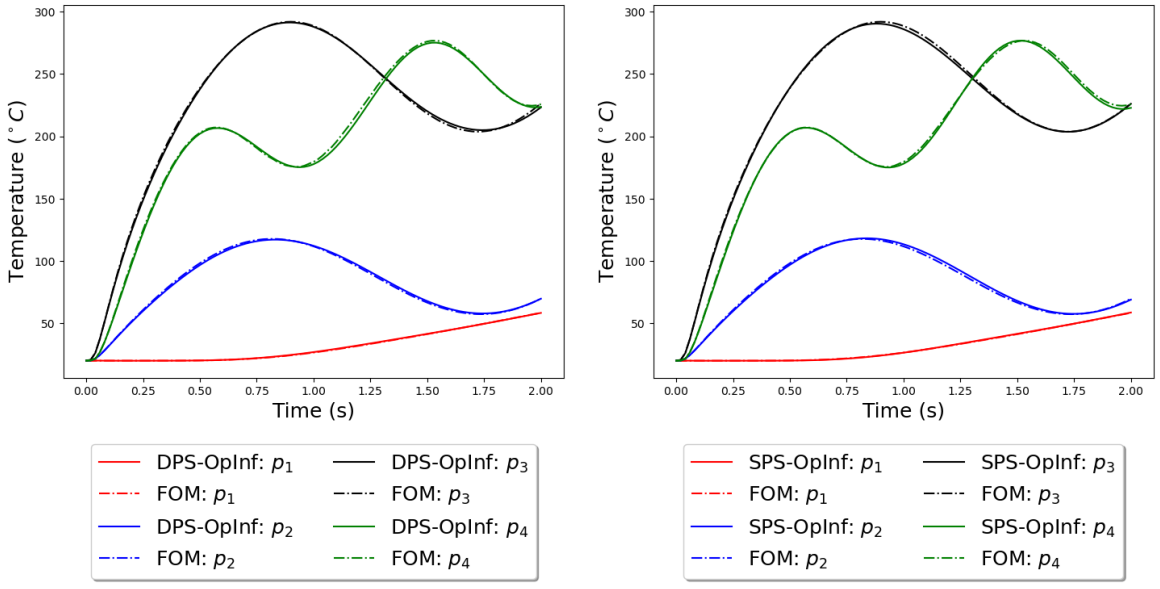


Figure 7.14.: The comparison between the error fields of (1) DEIM: left (2) AL-ANN: right, at: $t = 2$, in Test Case 1 of the thermal block model.



(a) POD-DEIM.

(b) AL-OpInf.



(c) DPS-OpInf.

(d) SPS-OpInf.

Figure 7.15.: The comparison between the solution trajectories of the FOM, DEIM, and OpInf-based ROMs at: $p_1 = (1,5)$, $p_2 = (5,1)$, $p_3 = (9,5)$ and $p_4 = (5,9)$ in Test Case 1 of the thermal block model. $N_r = 19$, $N_m = 50$, $N_s = 250$. With the same amount of data, the OpInf-based ROM's predictions at the virtual sensor positions match the FOM solutions very well. This fact applies to the OpInf-based ROMs trained in all the approaches, i.e., AL approach (Figure 7.15b), DPS approach (Figure 7.15c) and SPS approach (Figure 7.15d).

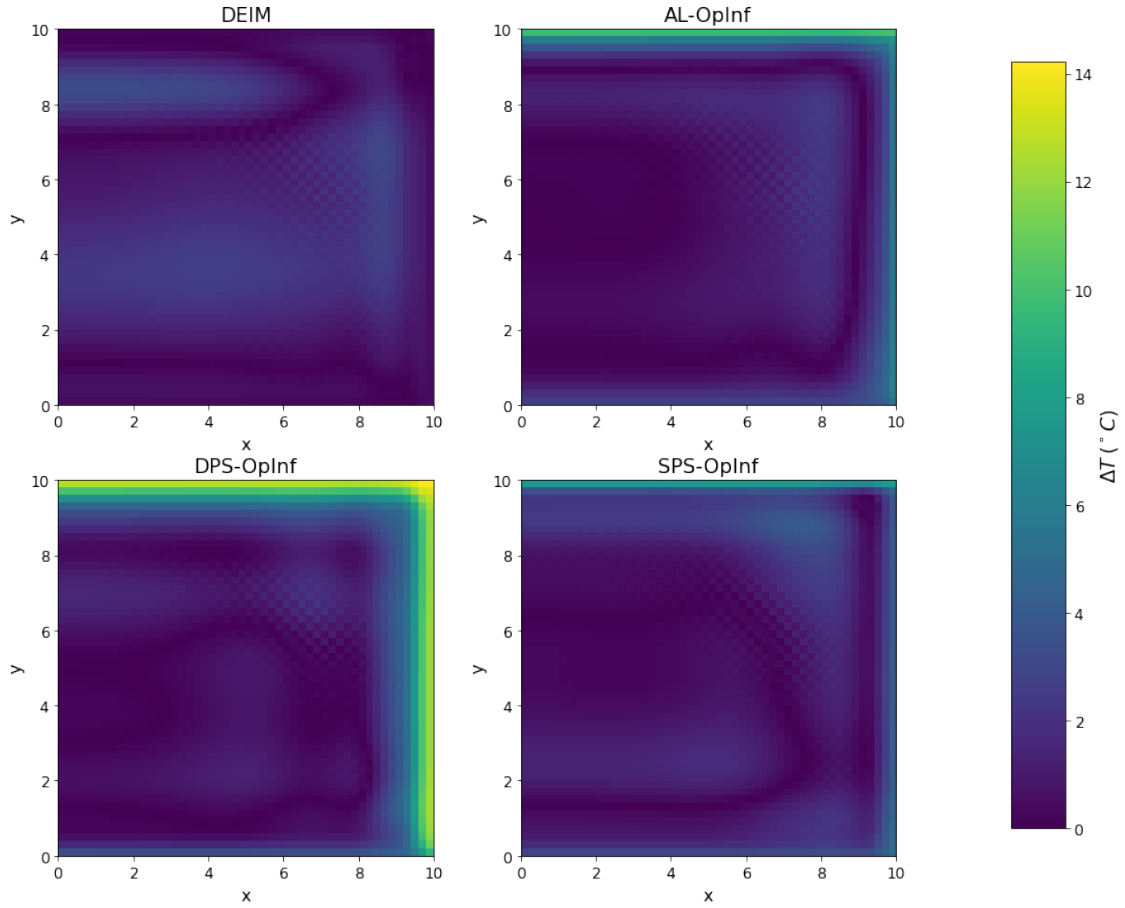


Figure 7.16.: The comparison between the error fields of (1) DEIM: upper-left (2) AL-OpInf: upper-right (3) DPS-OpInf: lower-left (4) SPS-OpInf: lower-right, at: $t = 2$, in Test Case 1 of the thermal block model. Despite that the error field of DPS-OpInf ROM has some bright areas, the prediction errors for all types of the ROMs are in a comparable level. The consistently good performance is enabled by the physics knowledge used for making the hypothetical form of the governing equation.

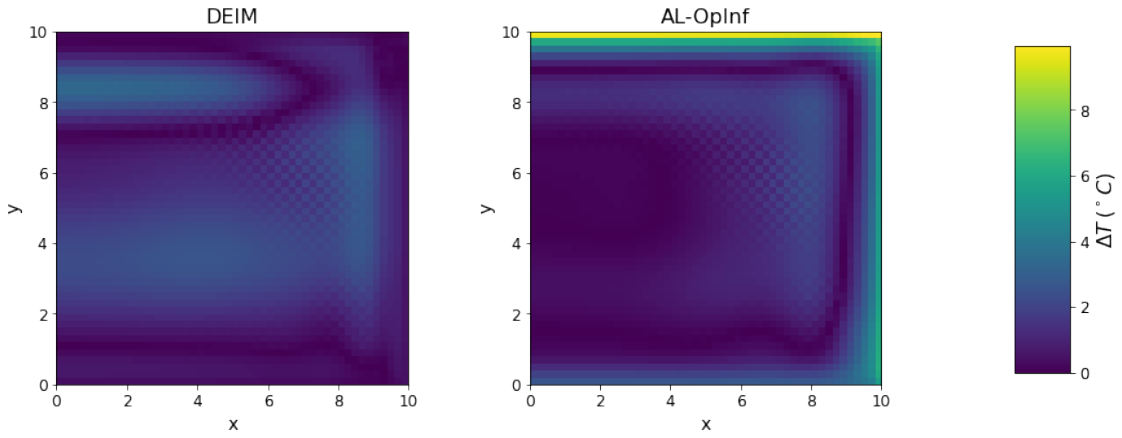


Figure 7.17.: The comparison between the error fields of (1) DEIM: left (2) AL-OpInf: right, at: $t = 2$, in Test Case 1 of the thermal block model.

7.1.7.2. Test Case 2

In this test, we choose the same points for observation as in Figure 7.10. But we will use a constant parameter combination which is:

$$\Gamma_{\text{top}}(t) \equiv 500,\ \Gamma_{\text{right}}(t) \equiv 500,\ \Gamma_{\text{base}}(t) \equiv 200. \tag{7.36}$$

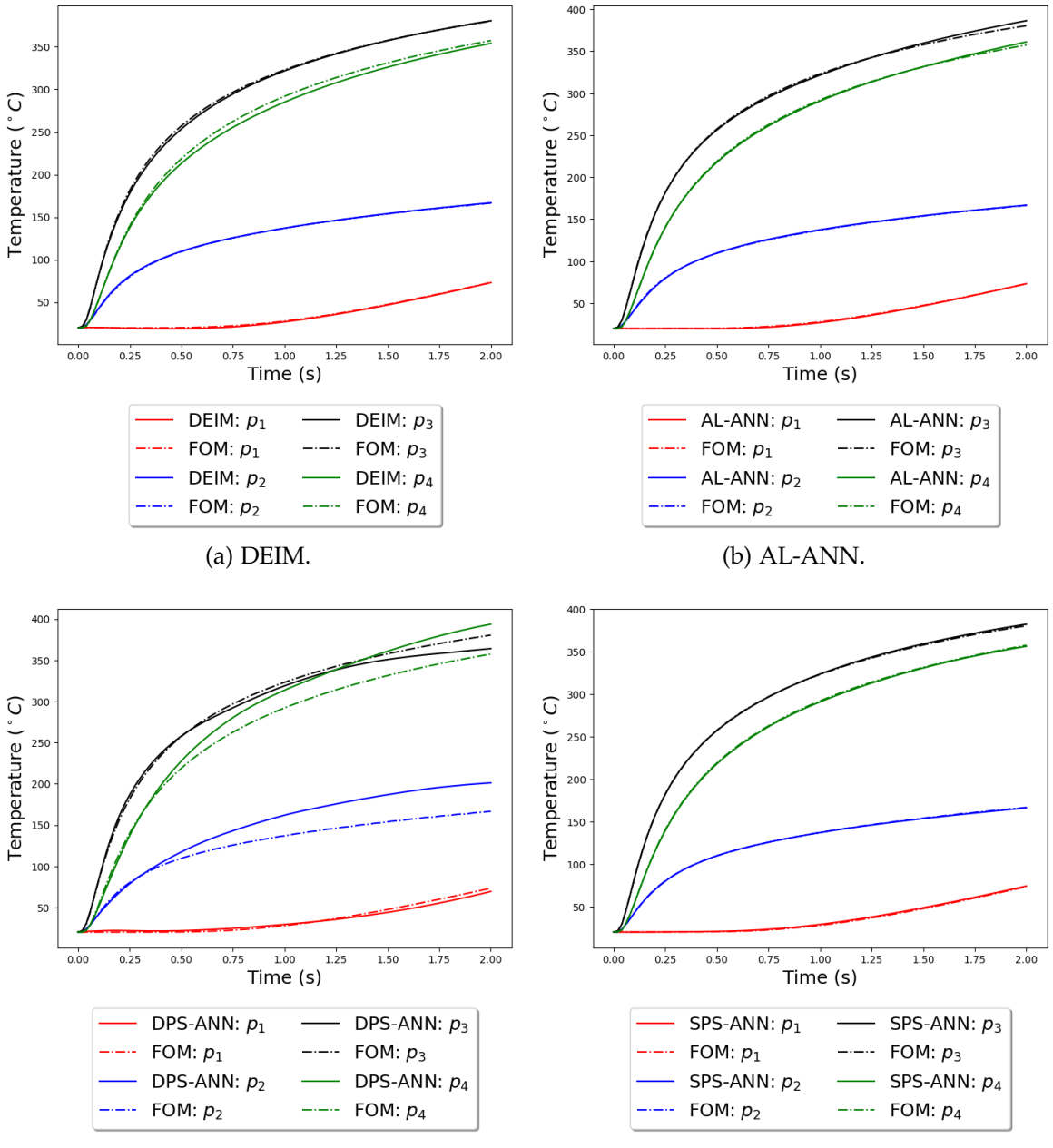
In this test case, we aim to lift the temperature field of the thermal block to the highest value. According to Figure 7.2, the material's thermal conductivity drops quickly in the temperature range $[300,\ 500]$. Testing the ROMs in this temperature range can provide us a more generalized evaluation for the ROM's performance in the whole possible state space. Through such a test case, we can evaluate the sample diversity with respect to system states. The test results are presented in Table 7.2, Figure 7.18, Figure 7.19, Figure 7.21 and Figure 7.22.

In Figure 7.18 and Figure 7.19, both the the AL-ANN and the SPS-ANN can make accurate predictions at the virtual sensor positions. If we further check Table 7.2, we will see that the AL-ANN has the best performance, the DPS-ANN performs worst, and the SPS-ANN has an acceptable performance. As a purely-data-driven ROM identification method, the ANN uses its universal approximation to simulate the input-output relation of the latent physics in the reduced space. However, the approximation is optimized only for the training data, and the quality of extrapolation is not guaranteed. Compared to the DPS-ANN, the training data for the SPS-ANN have much more samples in the high-temperature region, where the sharp drop of the thermal conductivity occurs. We believe this is the reason why the SPS-ANN is much better than the DPS-ANN in this test case.

For the OpInf-ROMs, the results are similar to Test Case 1. All the OpInf-ROMs have comparably accurate predictions in Test Case 2. This further proves that the OpInf method can take the advantage of being physics-aware and minimize the dependency on the training data.

	DEIM	AL-ANN	DPS-ANN	SPS-ANN	AL-OpInf	DPS-OpInf	SPS-OpInf
Mean (°C)	1.53	0.82	15.85	2.87	0.80	0.95	0.82
Std. ($(°C)^2$)	1.51	0.95	14.08	2.70	1.04	1.50	0.98
Min. (°C)	1.81e-05	1.23e-06	2.29e-05	1.16e-05	2.37e-07	1.77e-06	6.65e-06
Max. (°C)	7.18	8.12	104.11	14.05	16.16	14.64	9.62
Median (°C)	1.06	0.52	12.14	2.20	0.49	0.40	0.41

Table 7.2.: The statistical measures for e_{abs} in Test Case 2 of the thermal block model.



(a) DEIM.

(b) AL-ANN.

(c) DPS-ANN.

(d) SPS-ANN.

Figure 7.18.: The comparison between the solution trajectories of the FOM, DEIM, and ANN-based ROMs at: $p_1 = (1,5)$, $p_2 = (5,1)$, $p_3 = (9,5)$ and $p_4 = (5,9)$ in Test Case 2 of the thermal block model. $N_r = 19$, $N_m = 50$, $N_s = 3000$. With the same amount of data, the prediction accuracy of the AL-ANN in Figure 7.18b and the SPS-ANN in Figure 7.18d are both close to the intrusive hyper-reduction method DEIM and can be considered as accurate prediction. The DPS-ANN in Figure 7.18c has the largest deviation to the FOM solution.

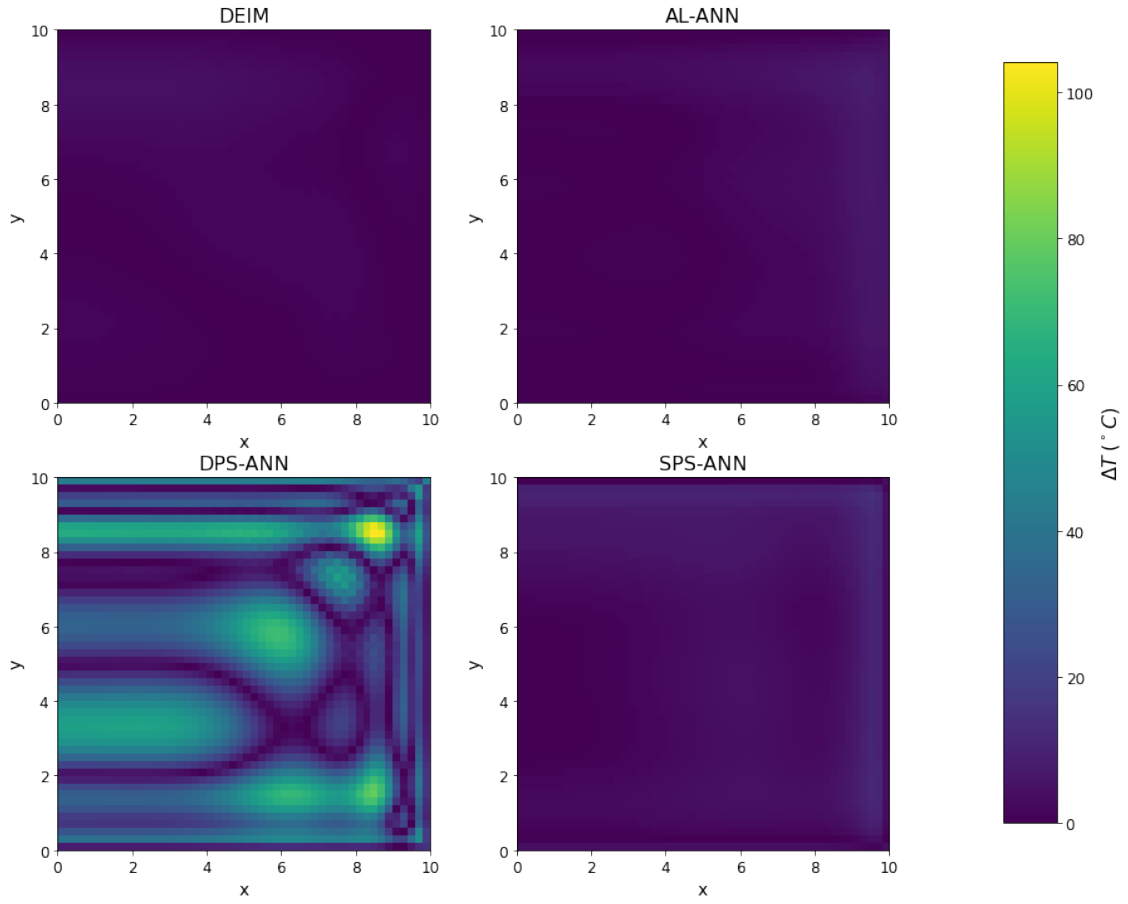


Figure 7.19.: The comparison between the error fields of (1) DEIM: upper-left (2) AL-ANN: upper-right (3) DPS-ANN: lower-left (4) SPS-ANN: lower-right, at: $t = 2$, in Test Case 2 of the thermal block model. The error field of DEIM, AL-ANN and SPS-ANN has similar error magnitudes. The error field of DPS-ANN is much brighter in general compared to the other error fields, which means DPS-ANN has the worst prediction throughout the whole geometry.

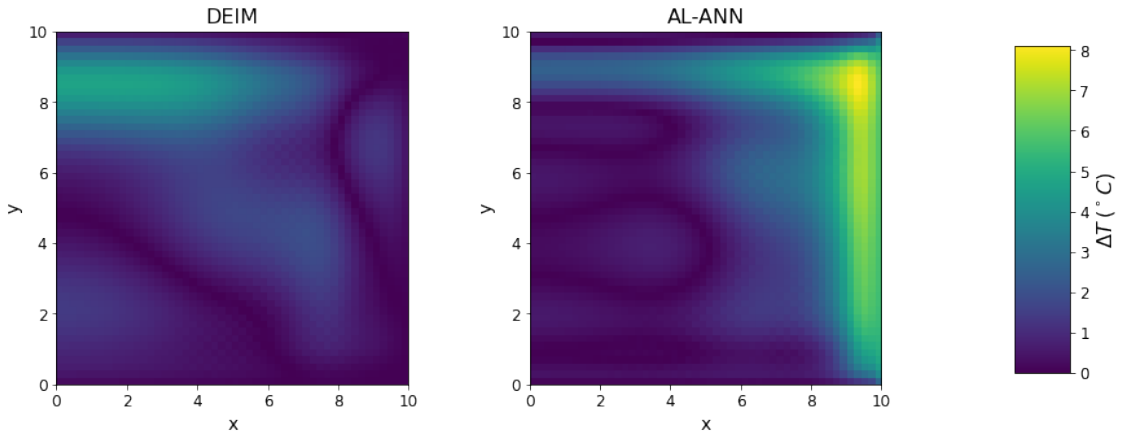
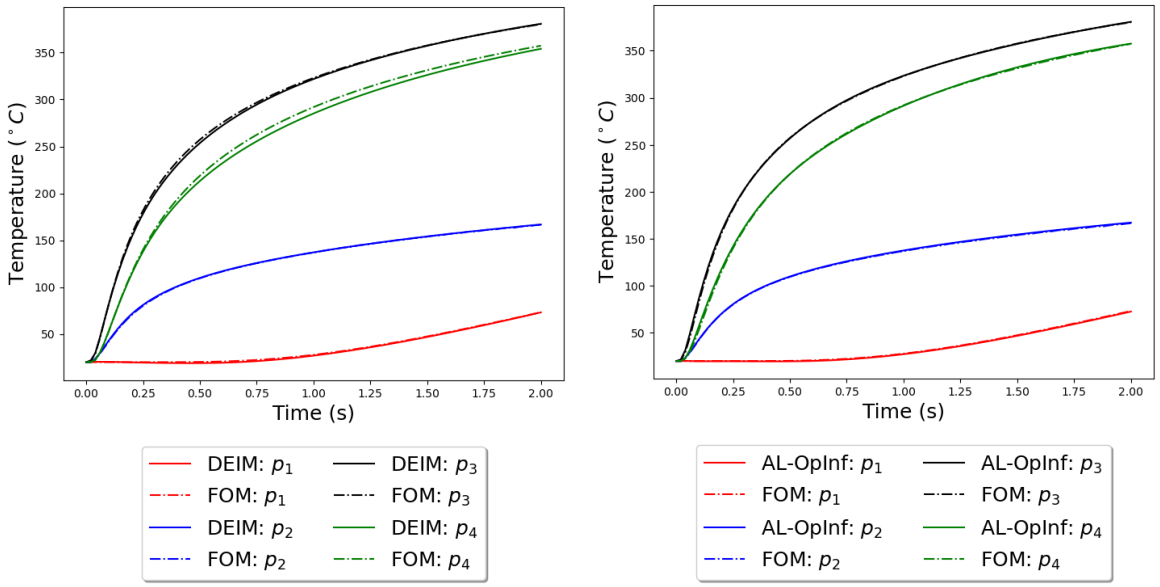
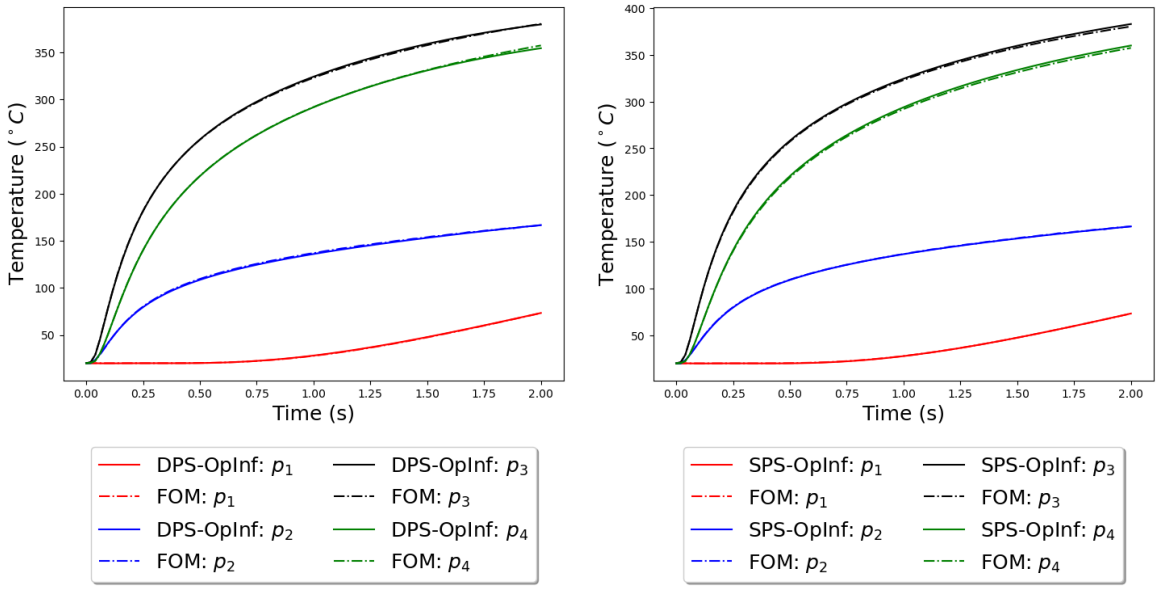


Figure 7.20.: The comparison between the error fields of (1) DEIM: left (2) AL-ANN: right, at: $t = 2$, in Test Case 2 of the thermal block model.



(a) DEIM.

(b) AL-OpInf.



(c) DPS-OpInf.

(d) SPS-OpInf.

Figure 7.21.: The comparison between the solution trajectories of the FOM, DEIM, and ANN-based ROMs at: $p_1 = (1,5)$, $p_2 = (5,1)$, $p_3 = (9,5)$ and $p_4 = (5,9)$ in Test Case 2 of the thermal block model. $N_r = 19$, $N_m = 50$, $N_s = 250$. At the investigated points, the OpInf-ROMs trained by the SPS data (Figure 7.21d), the DPS data (Figure 7.21b) and the AL data (Figure 7.21b) all have similar performances to the DEIM.

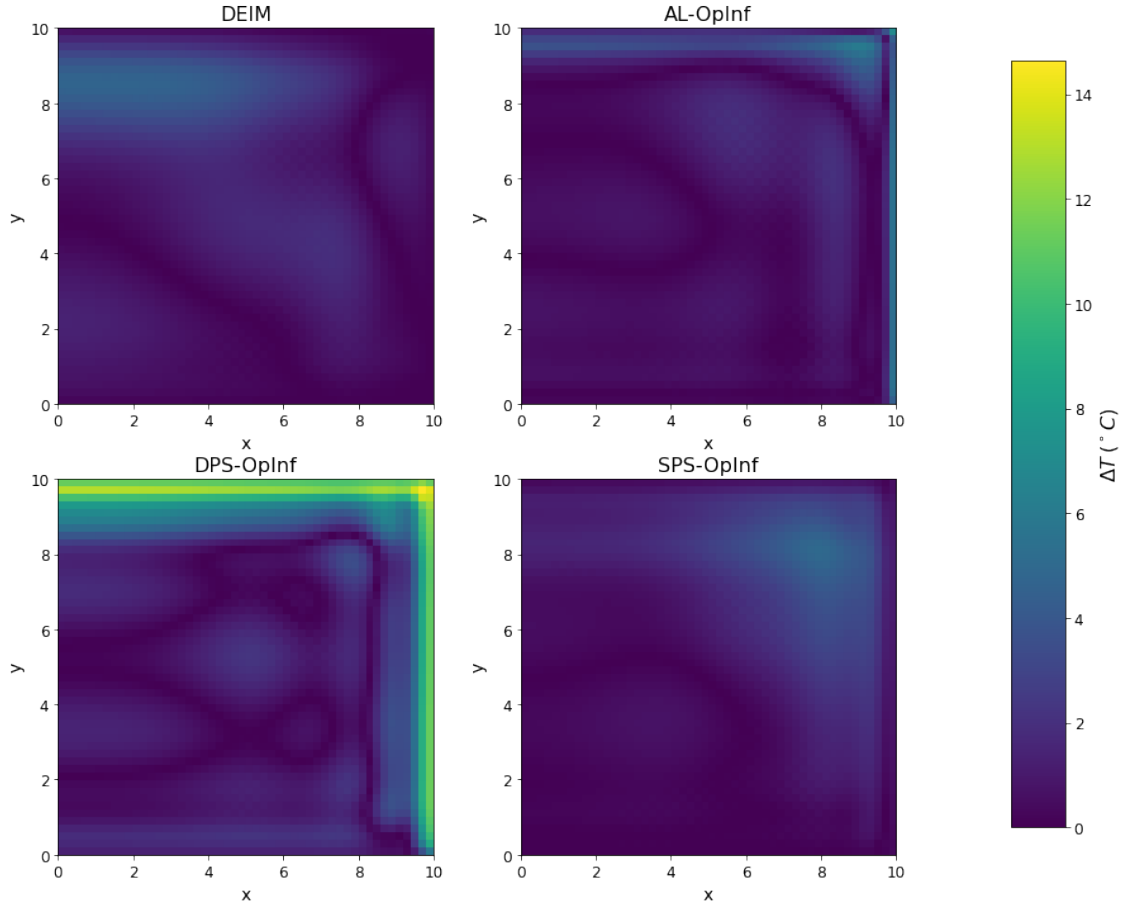


Figure 7.22.: The comparison between the error fields of (1) DEIM: upper-left (2) AL-OpInf: upper-right (3) DPS-OpInf: lower-left (4) SPS-OpInf: lower-right, at: $t = 2$, in Test Case 2 of the thermal block model. Similar to Figure 7.16, among all the error fields, the DPS-OpInf is the brightest but still comparable to the others. The bright areas in the error field of DPS-OpInf are close to the high-temperature boundary conditions, which means the DPS data lack of observation in this temperature range. Despite of this, the prediction errors for different ROMs can be still considered to have similar magnitudes.

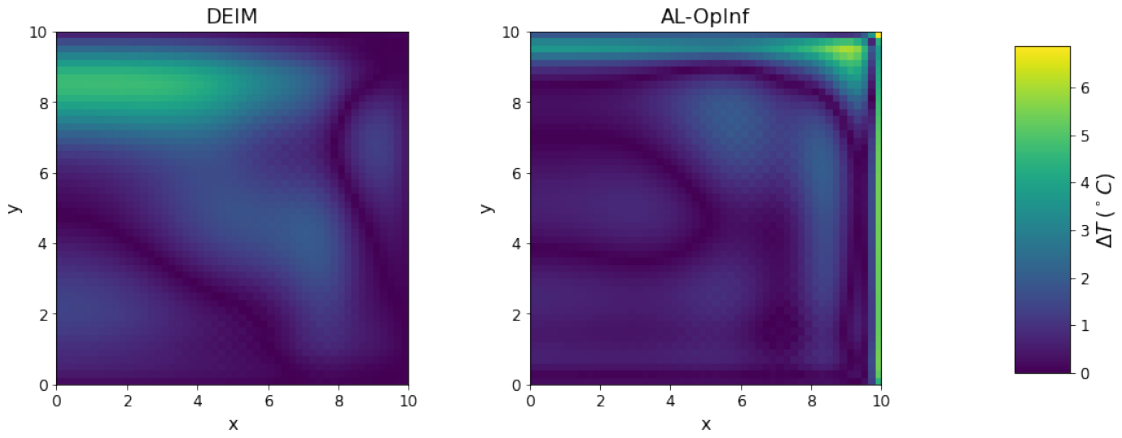


Figure 7.23.: The comparison between the error fields of (1) DEIM: left (2) AL-OpInf: right, at: $t = 2$, in Test Case 2 of the thermal block model.

7.1.8. Experiment discussions and conclusions

In this section, we use a thermal block made of temperature-dependent material to test different ROMs. Through two test cases, we can clearly see the advantages and disadvantages of the ROMs constructed in different ways.

Data sampling strategy Regarding the sampling strategies of training data, their influence on the ROM quality depends on the ROM identification method.

- For the non-physics-aware method, i.e., ANN-based MOR, since the ROM learns everything from training data, well-distributed training data can significantly improve the ROM's quality. In PAC validation, it is observed that trained with the same number of data, the JSS data construct an ANN-ROM whose 97%-confident ROM error is 1.67%. This is much lower than the ANN-ROM trained with the DPS data, which is 8.70%. The worst is the ANN-ROM trained with the SPS data, whose 97%-confident ROM error is 28.43%.
- For the physics-informed method, i.e., OpInf, since the ROM is informed with a hypothetical form of the full-order governing equation, the model's robustness and capability of extrapolation is naturally better. In PAC validation, it is observed that trained with the same number of data, the JSS data construct an OpInf-ROM whose 97%-confident ROM error is 1.67%. This is better than the OpInf-ROM trained with the DPS data (13.32%) and the OpInf-ROM trained with the SPS data (15.72%).

Through the comparison, we can conclude that the data prepared by the JSS method are more informative for constructing the ROM. Using the same number of samples, the ROMs trained with the JSS data have a much higher confident accuracy than the other ROMs.

Snapshot selection strategy Although the snapshot selection strategy is not the factor deciding the quality of the ROM, an appropriate snapshot selection strategy can accelerate the construction of the ROM. In this experiment, we test random selection, MMSE with the RBF estimator, MMSE with the GPR estimator, MMSE with the true error, the QBC method and the PL method.

For the MMSE method based on the error estimator, we first test the accuracy of the error estimator. In the results, the accuracy of both error estimators decreases with the iteration.

Then we generate the convergence curves for different approaches, We find that all the AL methods can accelerate the convergence, but the effectiveness is different. The PL method is the least effective. The QBC is the best choice. The MMSE method's

effectiveness is influenced by the accuracy of the error estimator, and the more accurate the error estimator is, the more effective the MMSE method is.

Case-dependent validation Here we conclude the ROM performance in the case-dependent validation.

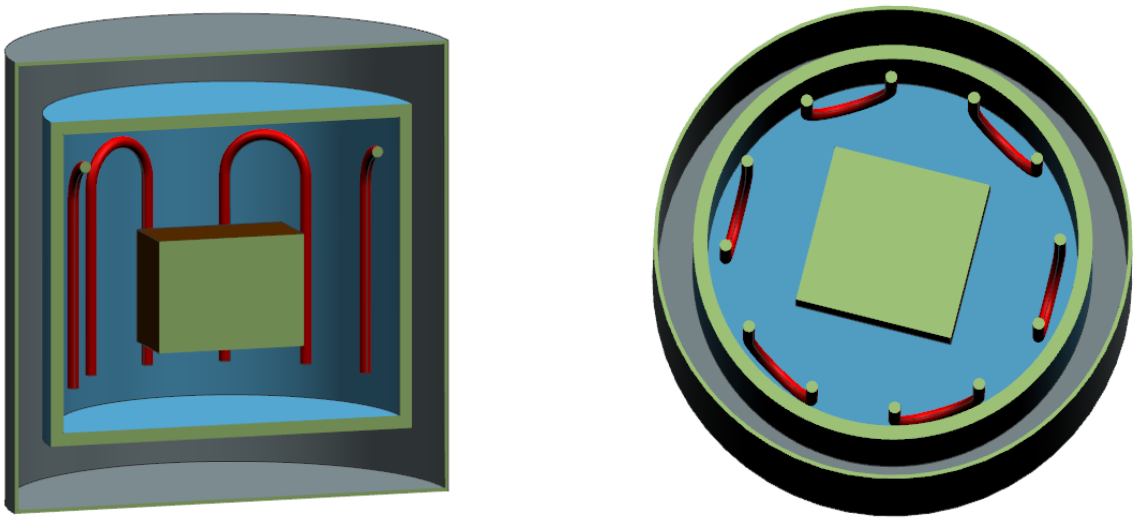
- For the ANN-based ROMs, based on Figure 7.12, Figure 7.13, Figure 7.18 and Figure 7.19, we can verify that the SPS-ANN has good accuracy in the high-temperature region (Figure 7.18d) but is inaccurate facing the complex input parameters (Figure 7.12d). On the contrary, the DPS-ANN is inaccurate in the high-temperature region (Figure 7.18c) but is relatively better facing the complex input parameters (Figure 7.12c). The AL-ANN has good prediction for the complex parameters as well as the high-temperature states, and we consider it as the best ANN-based ROM.
- For the OpInf-based ROMs, as a conclusion to Figure 7.15, Figure 7.16, Figure 7.21 and Figure 7.22, the SPS-OpInf that has a narrow sample distribution in the parameter space and the DPS-OpInf that has a narrow sample distribution in the reduced solution space can have a comparably good performance to the AL-OpInf that has a good sample distribution in both parameter space and reduced solution space. The OpInf-ROM's feature of being physics-informed significantly reduces the ROM's dependency on training data.

Conclusion Through this experiment, we can draw an initial conclusion that using the AL algorithm with a data pool constructed by JSS is a very efficient way of building a stable and accurate ROM, especially for non-physics-informed ROMs, i.e., ANN-ROMs. The core improvement is brought by the good distribution of the data sampled by the new sampling method JSS, and the AL method facilitates the construction of the ROM. In the next section, we will use a more realistic simulation model to further test different methods.

7.2. Model II: 3-D numerical model of a vacuum furnace

The second test model is a 3-D simulation model of a vacuum furnace. The model is essentially a thermal radiation problem.

In Figure 7.24, the 3D model used for the numerical experiments is presented. The geometry of the model is reproduced based on the 3D model used in [151], whose prototype is the high-pressure-gas-quenching furnace VHQ-446HF. The cubic part at the center of the model is the workzone where the processed material will be placed.



(a) Vertical cross-section.

(b) Horizontal cross-section.

Figure 7.24.: The 3D model used for simulating the vacuum furnace. Brown: workzone, red: heaters, blue: protection shield, gray: outer case.

The U-shaped tubes surrounding the workzone are heaters. During the operation phase, there will be heat generation inside the heaters, so the temperature of the heaters will increase. Through the radiation flux, the high-temperature heaters will heat up the material in the workzone. To prevent from overheating, a reflection shell is installed between the heaters and the outer case.

The material of the outer case and the reflection sheet is copper, whose thermal capacity is 385 $J/(kg \cdot K)$ and thermal conductivity is 387 $W/(m \cdot K)$. The heaters are made of Titanium. The thermal capacity of Titanium is 526 $J/(kg \cdot K)$ and its thermal conductivity is 6.7 $W/(m \cdot K)$. Without loss of generality, we define the emissivity of the reflection sheet to be 0.2 while other parts are defined to be black-body.

The governing equation of such a problem takes a generalized form:

$$\boldsymbol{E}\frac{\partial \boldsymbol{T}(t)}{\partial t} = \boldsymbol{K}\boldsymbol{T}(t) + a\boldsymbol{T}(t) + \boldsymbol{R}\boldsymbol{T}^4(t) + \boldsymbol{B}\boldsymbol{u}(t), \tag{7.37}$$

where $\boldsymbol{T}$ is the vector for the temperature field of the model. $\boldsymbol{E}$ and $\boldsymbol{K}$ is the thermal capacity matrix and the thermal conductivity matrix, respectively. $\boldsymbol{R}$ stands for the radiation heat transfer matrix, hereinafter called radiation matrix. $\boldsymbol{T}^4$ means element-wise-power-of-4 for the temperature field $\boldsymbol{T}$, which is also the nonlinear term in this problem. The matrix $\boldsymbol{B}$ is the distribution matrix of the parameter vector $\boldsymbol{u}$. By multiplying with $\boldsymbol{B}$, the parameters will be assigned to the corresponding DOFs. In this model, we design 7 controllable parameters, and they are 6 heating powers

$\{h_1,\ h_2,\ \ldots,\ h_6\}$ and the convection coefficient a. The range for each parameter is:

$$\begin{aligned} h_1,\ h_2,\ \ldots,\ h_6 &\in [0, 10^4]\ kW/m^3, \\ a &\in [5, 100]\ W/(m^2 \cdot K). \end{aligned} \tag{7.38}$$

Based on this, we can get the parameter space $\mathcal{M}$ for this problem, which is:

$$\mathcal{M} = [0, 10^4] \times [0, 10^4] \times [0, 10^4] \times [0, 10^4] \times [0, 10^4] \times [0, 10^4] \times [5, 100]. \tag{7.39}$$

The FOM is created and simulated by the commercial simulation software Simcenter 3D. We simulate the problem in the time span $t \in (0,\ 45000]$ with $\delta t = 100$.

7.2.1. Constructing the reduced space

We start with constructing a good POD space for the vacuum furnace model. 30 samples are randomly sampled from the predefined parameter space $\mathcal{M}$ using Hammersley Sequence. Based on them, 30 IVPs are constructed. In each IVP, 450 times steps are used. Through this process, we build the snapshot matrix $\boldsymbol{\mathcal{T}}$.

Using the snapshot matrix $\boldsymbol{\mathcal{T}}$, we can build the POD space of the optimal size. We use the same error tolerance as before:

$$e_{\text{tol}}^{\text{mean}} = 1\%,\ e_{\text{tol}}^{\text{max}} = 1.5\%. \tag{7.40}$$

We first assume the optimal ROM size can be found under 30. Then we produce the projection error curves for the constructed snapshot matrix $\boldsymbol{\mathcal{T}}$ in Figure 7.25.

With $q = 22$, the profile likelihood is maximized under the conditions that:

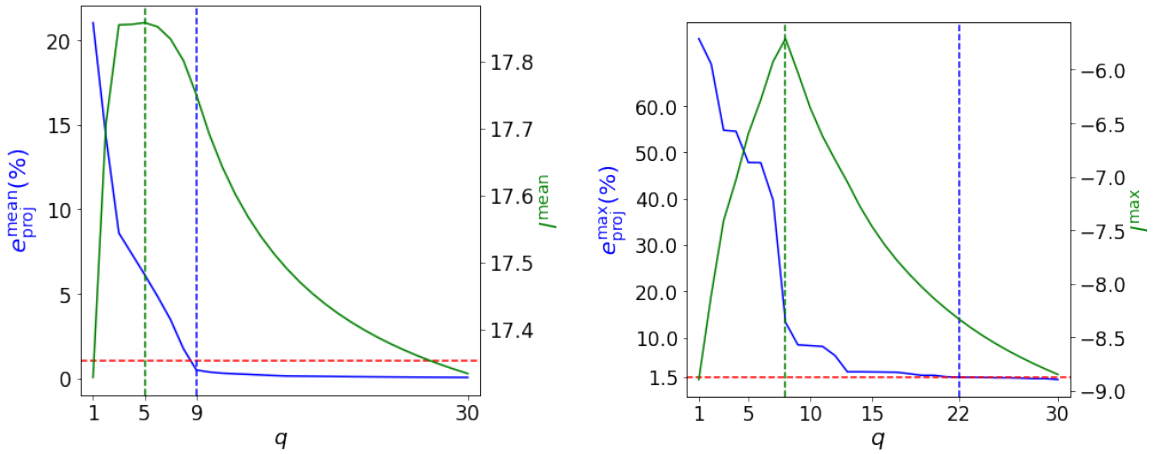
$$\begin{cases} e_{\text{proj}}^{\text{mean}}(q) \leq e_{\text{tol}}^{\text{mean}} \\ e_{\text{proj}}^{\text{max}}(q) \leq e_{\text{tol}}^{\text{max}} \end{cases}. \tag{7.41}$$

Therefore, the optimal ROM size for this FOM is determined to be 22. The first 22 singular vectors will be used to construct the reduced basis $\boldsymbol{V}$.

7.2.2. Application of DEIM

In the last section, we have already constructed the reduced basis $\boldsymbol{V}$, which establishes a bridge between a full state and a reduced state by $\boldsymbol{T} \approx \boldsymbol{V}\boldsymbol{T}_r$. Inserting this relation into Equation 7.37, we can get:

$$\frac{\partial \boldsymbol{T}_r}{\partial t} = \boldsymbol{V}^T\boldsymbol{E}^{-1}\boldsymbol{K}\boldsymbol{V}\boldsymbol{T}_r + a\boldsymbol{V}^T\boldsymbol{E}^{-1}\boldsymbol{V}\boldsymbol{T}_r + \boldsymbol{V}^T\boldsymbol{E}^{-1}\boldsymbol{R}(\boldsymbol{V}\boldsymbol{T}_r)^4 + \boldsymbol{V}^T\boldsymbol{E}^{-1}\boldsymbol{B}\boldsymbol{u}. \tag{7.42}$$



(a) $e^{\text{mean}}_{\text{proj}}(q)$ and $l^{\text{mean}}(q)$.

(b) $e^{\text{max}}_{\text{proj}}(q)$ and $l^{\text{max}}(q)$.

Figure 7.25.: The curves for $e_{\text{proj}}(q)$ and $l(q)$. The red dashed lines in Figure 7.3a and Figure 7.3b stand for $e^{\text{mean}}_{\text{tol}} = 1\%$ and $e^{\text{max}}_{\text{tol}} = 1.5\%$. The green dashed lines mark the q maximizing $l(q)$. The blue dashed lines mark the minimum q satisfying $e_{\text{proj}}(q) \leq e_{\text{tol}}$. The optimal ROM size is selected to be $N_r = 22$ for the FOM of the vacuum furnace.

The term $\boldsymbol{V}^T\boldsymbol{E}^{-1}\boldsymbol{R}(\boldsymbol{V}\boldsymbol{T}_r)^4$ in Equation 7.42 needs to be evaluated in the full space during the online phase. Therefore, we should apply DEIM to reduce this term. For this purpose, the snapshots of the nonlinear term need to be prepared, which is:

$$\boldsymbol{F} = \boldsymbol{\mathcal{T}}^4. \tag{7.43}$$

By applying the steps described in section 3.1, we can get the DEIM basis $\boldsymbol{H}$ and the selection matrix $\boldsymbol{P}$. But unlike the trouble we have in section 7.1.2, the computation for the nonlinear contribution $\boldsymbol{\mathcal{T}}^4$ is an element-wise operation, and different DOFs are not coupled in such operation. Therefore, we can directly build the DEIM approximation as:

$$\begin{aligned}
&\boldsymbol{V}^T\boldsymbol{E}^{-1}\boldsymbol{R}\boldsymbol{H}(\boldsymbol{P}^T\boldsymbol{H})^{-1}\boldsymbol{P}^T\boldsymbol{T}^4 \\
&\approx \boldsymbol{V}^T\boldsymbol{E}^{-1}\boldsymbol{R}\boldsymbol{H}(\boldsymbol{P}^T\boldsymbol{H})^{-1}\boldsymbol{P}^T(\boldsymbol{V}\boldsymbol{T}_r)^4 \\
&= \boldsymbol{V}^T\boldsymbol{E}^{-1}\boldsymbol{R}\boldsymbol{H}(\boldsymbol{P}^T\boldsymbol{H})^{-1}(\boldsymbol{P}^T\boldsymbol{V}\boldsymbol{T}_r)^4 \\
&= \boldsymbol{R}_{\text{DEIM}}(\boldsymbol{W}\boldsymbol{T}_r)^4,
\end{aligned} \tag{7.44}$$

where:

$$\begin{cases} \boldsymbol{R}_{\text{DEIM}} = \boldsymbol{V}^T\boldsymbol{E}^{-1}\boldsymbol{R}\boldsymbol{H}(\boldsymbol{P}^T\boldsymbol{H})^{-1} \\ \boldsymbol{W} = \boldsymbol{P}^T\boldsymbol{V} \end{cases}. \tag{7.45}$$

Substituting Equation 7.44 into Equation 7.42 results in:

$$\frac{\partial \boldsymbol{T}_r}{\partial t} = \boldsymbol{K_r}\boldsymbol{T}_r + a\boldsymbol{A}_r\boldsymbol{T}_r + \boldsymbol{R}_{\text{DEIM}}(\boldsymbol{W}\boldsymbol{T}_r)^4 + \boldsymbol{B_r}\boldsymbol{u}, \tag{7.46}$$

where:

$$\begin{cases} \boldsymbol{K_r} = \boldsymbol{V}^T\boldsymbol{E}^{-1}\boldsymbol{K}\boldsymbol{V} \\ \boldsymbol{A}_r = \boldsymbol{V}^T\boldsymbol{E}^{-1}\boldsymbol{V} \\ \boldsymbol{B_r} = \boldsymbol{V}^T\boldsymbol{E}^{-1}\boldsymbol{B} \end{cases}. \tag{7.47}$$

The matrices $\boldsymbol{W}$, $\boldsymbol{R}_{\text{DEIM}}$, $\boldsymbol{K_r}$, $\boldsymbol{A}_r$, $\boldsymbol{B_r}$ can be computed in the offline phase, and evaluating Equation 7.46 does not have much computational complexity during the online phase. Here we only apply DEIM to the nonlinear term such that there is no approximation error but only the projection error for the linear terms in Equation 7.37.

7.2.3. Application of the ANN

The ANN used to identify the ROM is a RKNN with 1 input layer, 2 hidden layers and 1 output layer. The numbers of neurons are [29, 116, 116, 22]. The activation function is ReLU.

7.2.4. Application of Operator Inference

For employing OpInf, we need auxiliary variables again. For this problem, we define the lifted state vector as:

$$\boldsymbol{s} = \begin{bmatrix} \boldsymbol{s}_1 = \boldsymbol{T} \\ \boldsymbol{s}_2 = \boldsymbol{T}^2 \\ \boldsymbol{s}_3 = \boldsymbol{T}^3 \end{bmatrix}. \tag{7.48}$$

Then we can construct the lifted governing equation as:

$$\begin{aligned} \frac{\partial \boldsymbol{s}_1}{\partial t} &= \boldsymbol{E}^{-1}\boldsymbol{K}\boldsymbol{s}_1 + a\boldsymbol{E}^{-1}\boldsymbol{s}_1 + \boldsymbol{E}^{-1}\boldsymbol{R}\boldsymbol{s}_1^4 + \boldsymbol{E}^{-1}\boldsymbol{B}\boldsymbol{u}, \\ \frac{\partial \boldsymbol{s}_2}{\partial t} = 2\boldsymbol{T}\frac{\partial \boldsymbol{T}}{\partial t} &= 2\boldsymbol{E}^{-1}\boldsymbol{K}\boldsymbol{T}^2 + 2a\boldsymbol{E}^{-1}\boldsymbol{T}^2 + 2\boldsymbol{E}^{-1}\boldsymbol{R}\boldsymbol{T}^5 + 2\boldsymbol{E}^{-1}\boldsymbol{B}\boldsymbol{u} \otimes \boldsymbol{T}, \\ &= 2\boldsymbol{E}^{-1}\boldsymbol{K}\boldsymbol{s}_2 + 2a\boldsymbol{E}^{-1}\boldsymbol{s}_2 + 2\boldsymbol{E}^{-1}\boldsymbol{R}\boldsymbol{s}_2 \otimes \boldsymbol{s}_3 + 2\boldsymbol{E}^{-1}\boldsymbol{B}\boldsymbol{u} \otimes \boldsymbol{s}_1, \\ \frac{\partial \boldsymbol{s}_3}{\partial t} = 3\boldsymbol{T}^2\frac{\partial \boldsymbol{T}}{\partial t} &= 3\boldsymbol{E}^{-1}\boldsymbol{K}\boldsymbol{T}^3 + 3a\boldsymbol{E}^{-1}\boldsymbol{T}^3 + 3\boldsymbol{E}^{-1}\boldsymbol{R}\boldsymbol{T}^6 + 3\boldsymbol{E}^{-1}\boldsymbol{B}\boldsymbol{u} \otimes \boldsymbol{T}^2, \\ &= 3\boldsymbol{E}^{-1}\boldsymbol{K}\boldsymbol{s}_3 + 3a\boldsymbol{E}^{-1}\boldsymbol{s}_3 + 3\boldsymbol{E}^{-1}\boldsymbol{R}\boldsymbol{s}_3 \otimes \boldsymbol{s}_3 + 3\boldsymbol{E}^{-1}\boldsymbol{B}\boldsymbol{u} \otimes \boldsymbol{s}_2, \end{aligned} \tag{7.49}$$

We can assume that the projection of Equation 7.49 in the reduced space takes such a form:

$$\frac{\partial \boldsymbol{\omega}}{\partial t} = \boldsymbol{F}_1\boldsymbol{\omega} + \boldsymbol{F}_2\boldsymbol{\omega} \otimes \boldsymbol{\omega} + \boldsymbol{F}_B\boldsymbol{u} + \boldsymbol{F}_N\boldsymbol{u} \otimes \boldsymbol{\omega} + \boldsymbol{F}_C. \tag{7.50}$$

Then we can use the method described in section 3.3 to identify the reduced system matrices $\boldsymbol{F}_1$, $\boldsymbol{F}_2$, $\boldsymbol{F}_B$, $\boldsymbol{F}_N$ and $\boldsymbol{F}_C$. Here we can see, for different full order problems, hypothetical governing equations can be different.

7.2.5. Study on data sampling strategies

In this test, we focus on testing the influence of the data sampling methods. We prepare three data pools, and each of them have 40,000 samples. They are created by SPS, DPS and JSS, respectively. A PAC validator with 2,944 test samples is built. With this validator, we can check the 97%-confident ROM error with a validation reliability of 99%. We set the designed ROM error τ^*_{design} to be 1%.

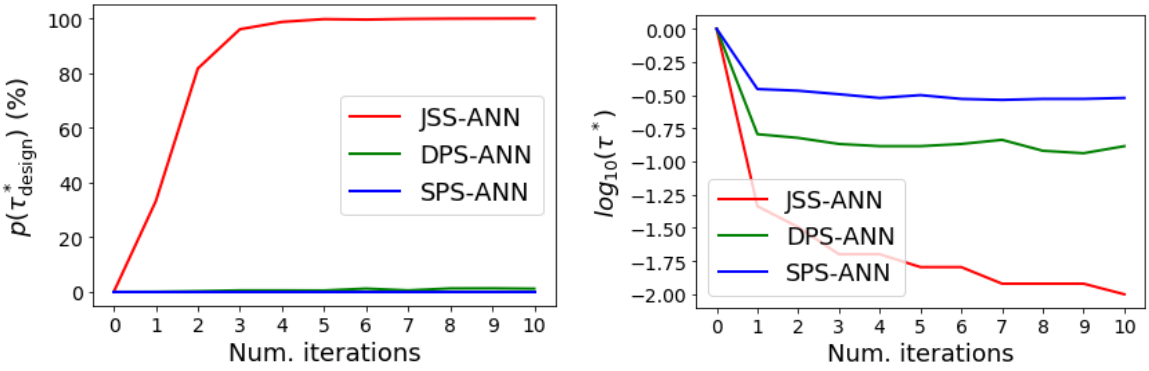
To perform the JSS method, we use $\boldsymbol{\mathcal{T}}$ collected in section 7.2.1 as $\boldsymbol{\mathcal{T}}_{\text{estimate}}$. We use $\beta = 0.1$ to loosen the joint space. To trim the joint space, we define $y_{\max} = 1500$ and $y_{\min} = 20$. Here the upper and lower limit for the values in the lifted space is defined by empirical values. 20 $^\circ C$ is the room temperature, and 1500 $^\circ C$ is lower than the melting point of the material of the heaters (Titanium, 1668 $^\circ C$). According to the application of the equipment, we know this assumption builds meaningful boundaries for the temperature range in the full space.

In each iteration, we randomly draw a certain number (2,000 for the ANN-ROM and 500 for the OpInf-ROM) of samples from each data pool and add to their corresponding training data. We train/update the ROMs with the extended training data and compute their PAC scores.

The results are given in Figure 7.26 and Figure 7.27, the results from PAC validation are presented. For the ANN-based approach, using the JSS data, the observed confidence $p(\tau^*_{\text{design}})$ of the predefined ROM error converges to 100% after 10 iterations. However, the ROMs constructed with the other two types of data still have a very low observed confidence for the designed ROM error τ^*_{design}. The SPS-ANN has a 97%-confident error of 29.15%, and the DPS-ANN has a 97%-confident error of 11.56%.

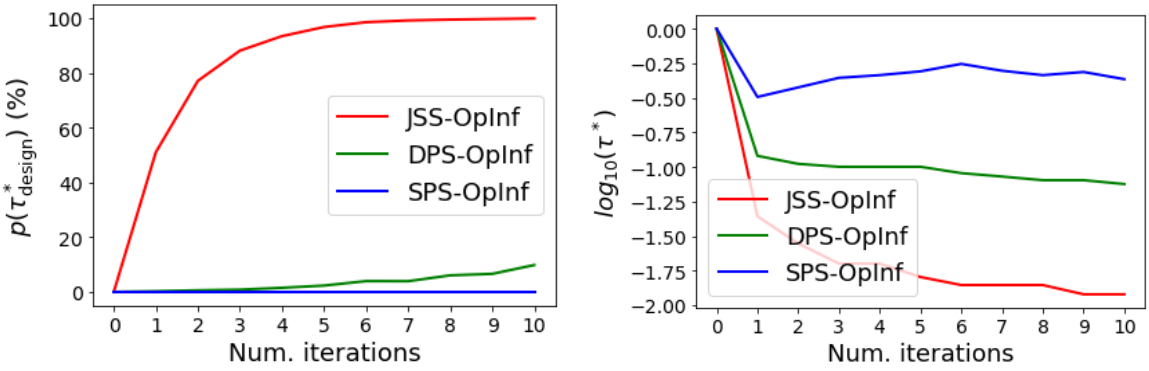
A similar conclusion can be applied to the OpInf-based approach. Using the JSS data, the algorithm converges after 10 iterations. When it converges, the 97%-confident error of the JSS-OpInf is 1.94%. However, with the same number of the SPS samples and the DPS samples, the ROMs still have τ^* that are far greater than τ^*_{design}. The SPS-OpInf has a 97%-confident error of 43.22%, and the DPS-OpInf has a 97%-confident error of 7.54%.

In sum, for this model with a higher-dimensional parameter space, the advantage of using the JSS method to collect training data is even more significant.



(a) $p(\tau^*_{\text{design}})$. (b) τ^*.

Figure 7.26.: The PAC scores of the ANN-ROMs change with the extension of the training data. The snapshot increment $\Delta s = 2000$. Both PAC scores of the JSS-ANN are the best, followed by the DPS-ANN. The SPS-ANN is the worst.

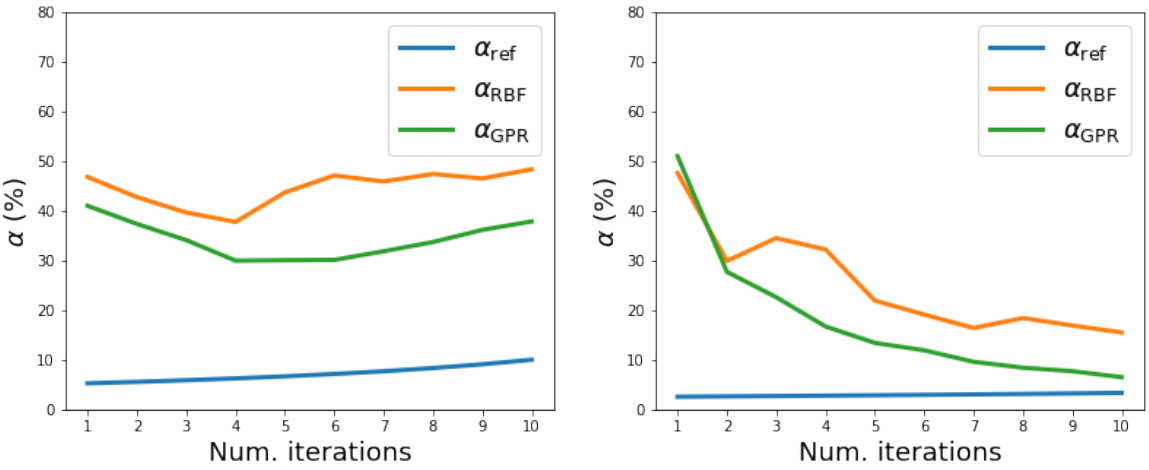


(a) $p(\tau^*_{\text{design}})$. (b) τ^*.

Figure 7.27.: The PAC scores of the OpInf-ROMs change with the extension of the training data. The snapshot increment $\Delta s = 500$. Both PAC scores of the JSS-OpInf are the best, followed by the DPS-OpInf. The SPS-OpInf is the worst.

7.2.6. Study on snapshot selection strategies

In this test, we will analyze the accuracy of the error estimators using the agreement ratio α. In Figure 7.28, we run the AL algorithm for 10 iterations. During the iterations, we compute the agreement ratio α between the estimated and the true ROM errors. While estimating for the OpInf-ROM, we see a similar trend as observed in section 7.1.6, where α_{RBF} and α_{GPR} drop within the iterations. However, we see a different trend during training the ANN-ROM, where both agreement ratios firstly decrease, then increase. While training the ANN-ROM, more training data are demanded and drawn from the data pool. Therefore, the number of the remaining samples in the data pool decreases faster within the iteration. This probably causes the increasing α in late iterations.



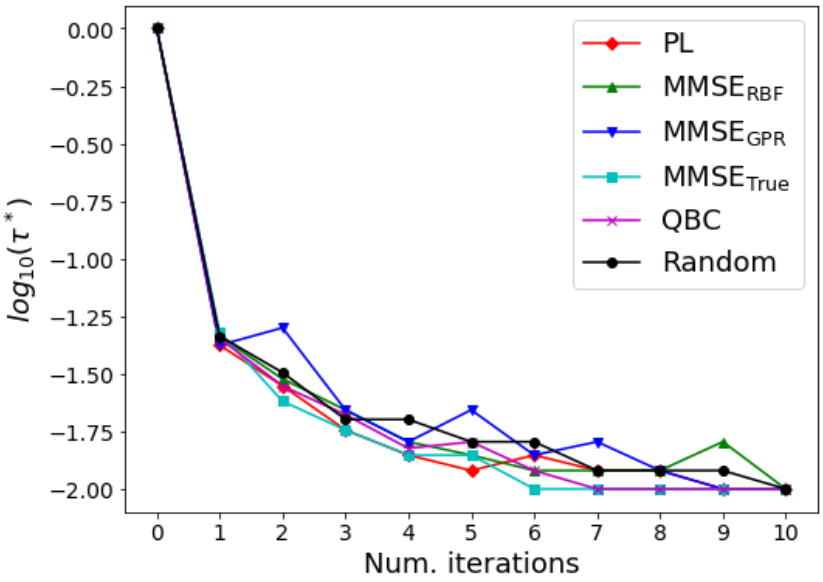
(a) For training ANN-ROM. $\Delta s = 2000$.

(b) For training OpInf-ROM. $\Delta s = 500$.

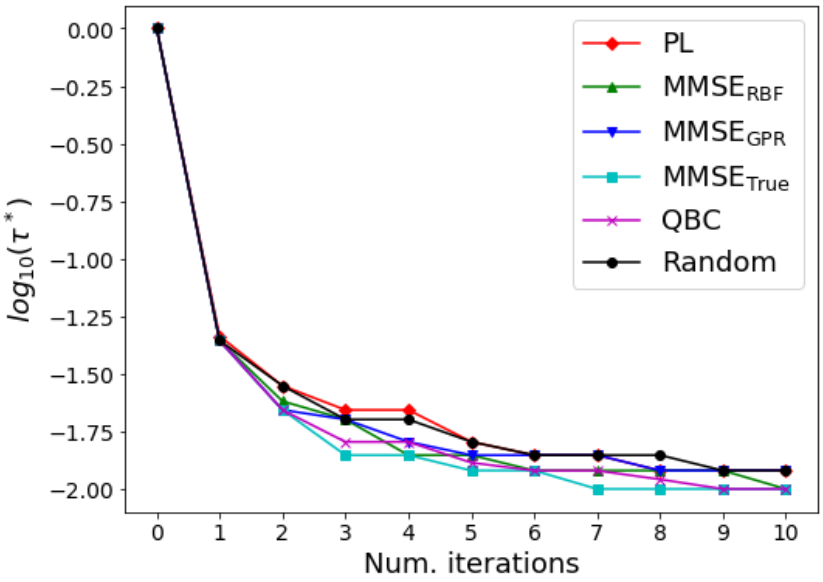
Figure 7.28.: The curves for the agreement ratio α in the first 10 iterations for the ROMs. The error estimators are fitted using the 2944 samples in the PAC validator. For the ANN-ROM, α_{RBF} and α_{GPR} firstly decrease then increase. For the OpInf-ROM, α_{RBF} and α_{GPR} generally decrease with the iteration proceeding.

Then we track the convergence of the 97%-confident ROM error τ^* during the iterations. In Figure 7.29, the convergence curves of using different methods to training ROMs are presented.

In Figure 7.29a, we see that although the MMSE method based on true MSEs successfully accelerates the convergence of both the ANN-ROM and the OpInf-ROM, the MMSE methods based on the estimated MSEs is not effective when applied to the ANN-ROM. When applying them to the ANN-ROM, the convergence rates are



(a) τ^* of the ANN-ROMs. $\Delta s = 2000$.



(b) τ^* of the OpInf-ROMs. $\Delta s = 500$.

Figure 7.29.: 97%-confident τ^* of the ROMs trained by five different methods. All methods draw samples from the same initial data pool, which is create by JSS.

similar to random selection. Between them, the convergence rate of the MMSE_{RBF} is better than the MMSE_{GPR}. Recalling in Figure 7.28a, the accuracy of the RBF error estimator is higher than the GPR error estimator, which can be the reason for different behaviors in Figure 7.29a.

In Figure 7.29, the QBC method accelerates the convergence rates of both the ANN-ROM and the OpInf-ROM. Since the QBC method does not require an accurate error estimator but uses uncertainty in error estimation, the accuracy of error estimators does not influence sample selection. In Figure 7.29a, its performance is as good as the MMSE method based on true MSEs of the ANN-ROM.

As for the PL method, it successfully speeds up the convergence of the algorithm for the ANN-ROM. However, for the OpInf-ROM, it does not show any advantages to random selection.

Based on the above observations, we will use the QBC method to build the ROMs actively.

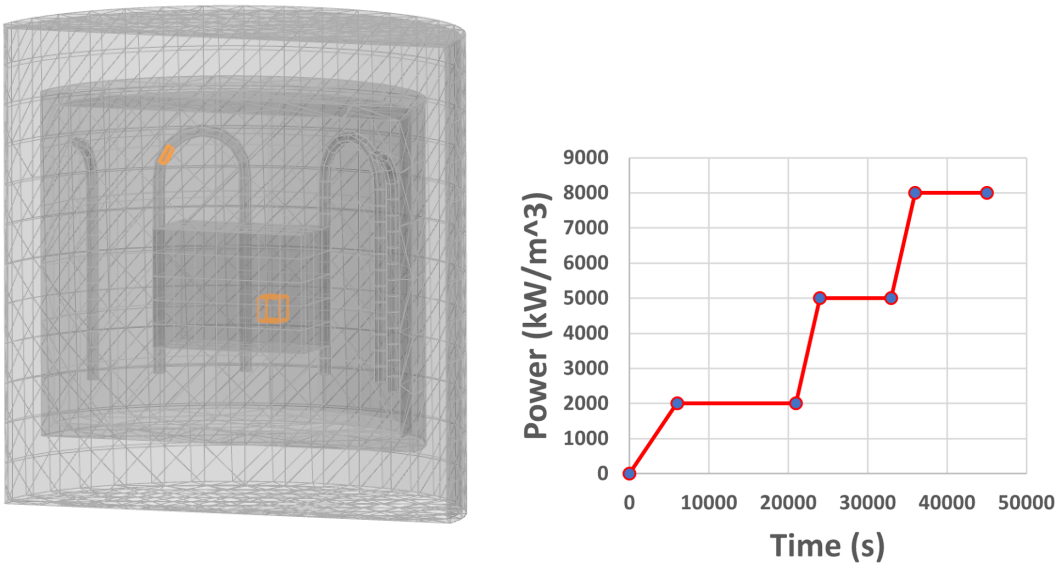
7.2.7. Case-dependent validation results

In this section, the ROMs will be tested by two different use cases. In the first use case, we use a 3-stage heating profile optimized in [151]. In the second use case, two heaters are assumed to have failures and cannot generate heat flux throughout the process. The ROMs are used to perform the real-time simulation for the described use cases. Such ROMs can be used for, e.g., process control or virtual sensor monitoring for the hot working.

7.2.7.1. 3-stage heating

In this use case, an element in the workzone and an element in the heater are chosen as the positions for temperature monitoring, and they are shown in Figure 7.30a. The time-dependent function of the volumetric power of the heaters is shown in Figure 7.30b. The convection coefficient a is defined with $50W/\left(m^2 \cdot K\right)$ throughout the process. The predicted trajectories are given in Figure 7.31 and Figure 7.32. The statistical analysis is given in Table 7.3.

According to Figure 7.31, the ANN-ROM built by our AL algorithm has a much better performance than the rest ANN-ROMs. Its prediction in the high-temperature region is even better than the DEIM-ROM. The DPS-ANN takes the second place, whose prediction at the heater is accurate. But when it predicts for the element in the workzone, it has a relatively large deviation to the FOM solution. The SPS-ANN is the worst and has relatively big errors both in the prediction of the heater temperature and workzone temperature.



(a) Two virtual sensors are placed at: (1) the workzone (2) the heater.

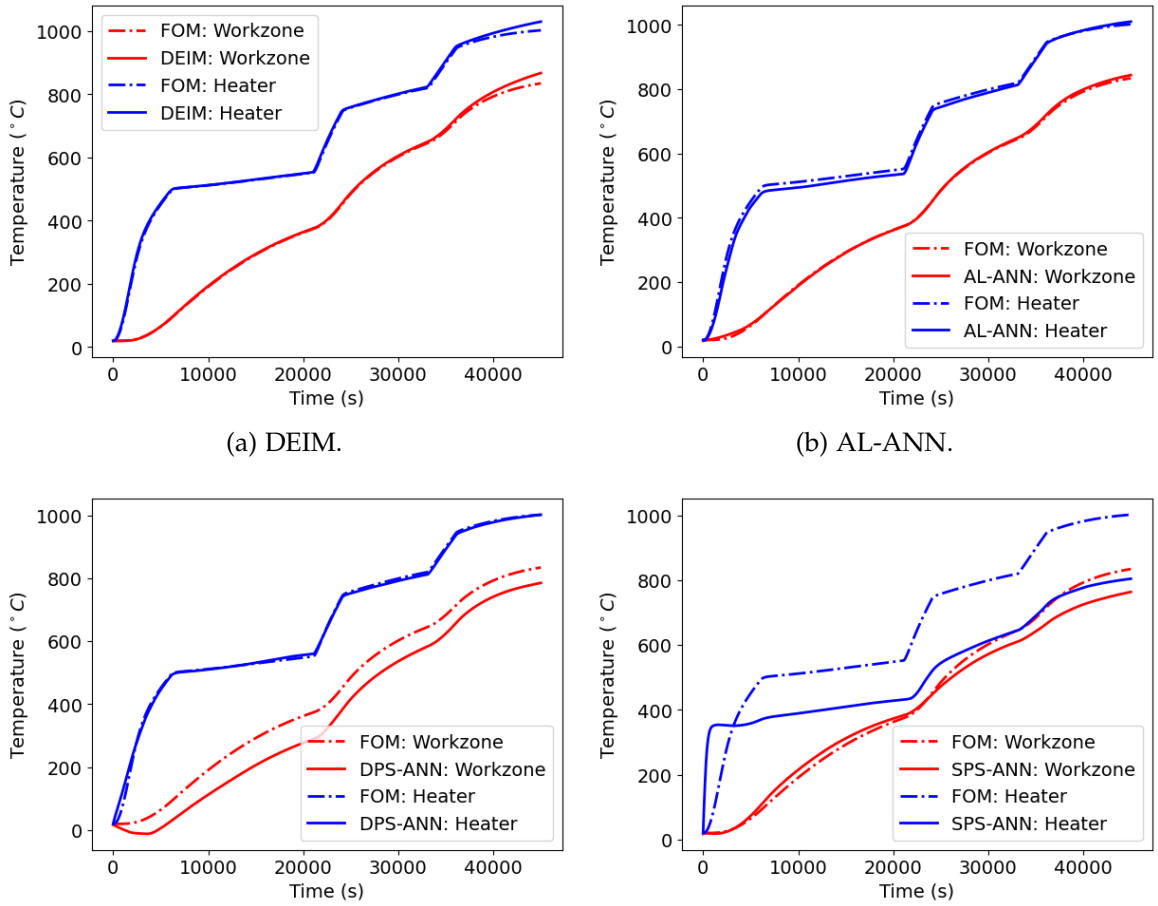
(b) The 3-stage heating profile.

Figure 7.30.: Left: the virtual sensor positions. Right: the 3-stage heating profile in the first use case.

In the test of the OpInf-ROMs, the AL-OpInf and the DPS-OpInf can predict with similar accuracy, which can be also verified by the data in Table 7.3. In Table 7.3, we see that their maximum errors are both higher than the maximum error of the DEIM-ROM. Besides, the standard deviations of the AL-OpInf and the DPS-OpInf are also higher than the DEIM-ROM's. However, we can also observe that their mean errors are both lower than the mean error of the DEIM-ROM. This means their performances in this use case are already very close to the intrusive ROM.

	DEIM	AL-ANN	DPS-ANN	SPS-ANN	AL-OpInf	DPS-OpInf	SPS-OpInf
Mean (°C)	5.35	2.29	5.72	18.42	3.69	3.79	102.40
Std. ((°C)^2)	6.47	4.30	9.29	34.09	8.74	13.39	246.05
Min. (°C)	1.33e-02	2.14e-08	1.69e-06	1.43e-06	1.21e-06	5.15e-08	3.46e-07
Max. (°C)	32.70	96.65	104.01	315.83	166.32	334.06	3953.85
Median (°C)	2.42	1.03	2.30	10.81	1.33	1.71	15.12

Table 7.3.: The statistical measures for e_{abs} in the 3-stage heating.



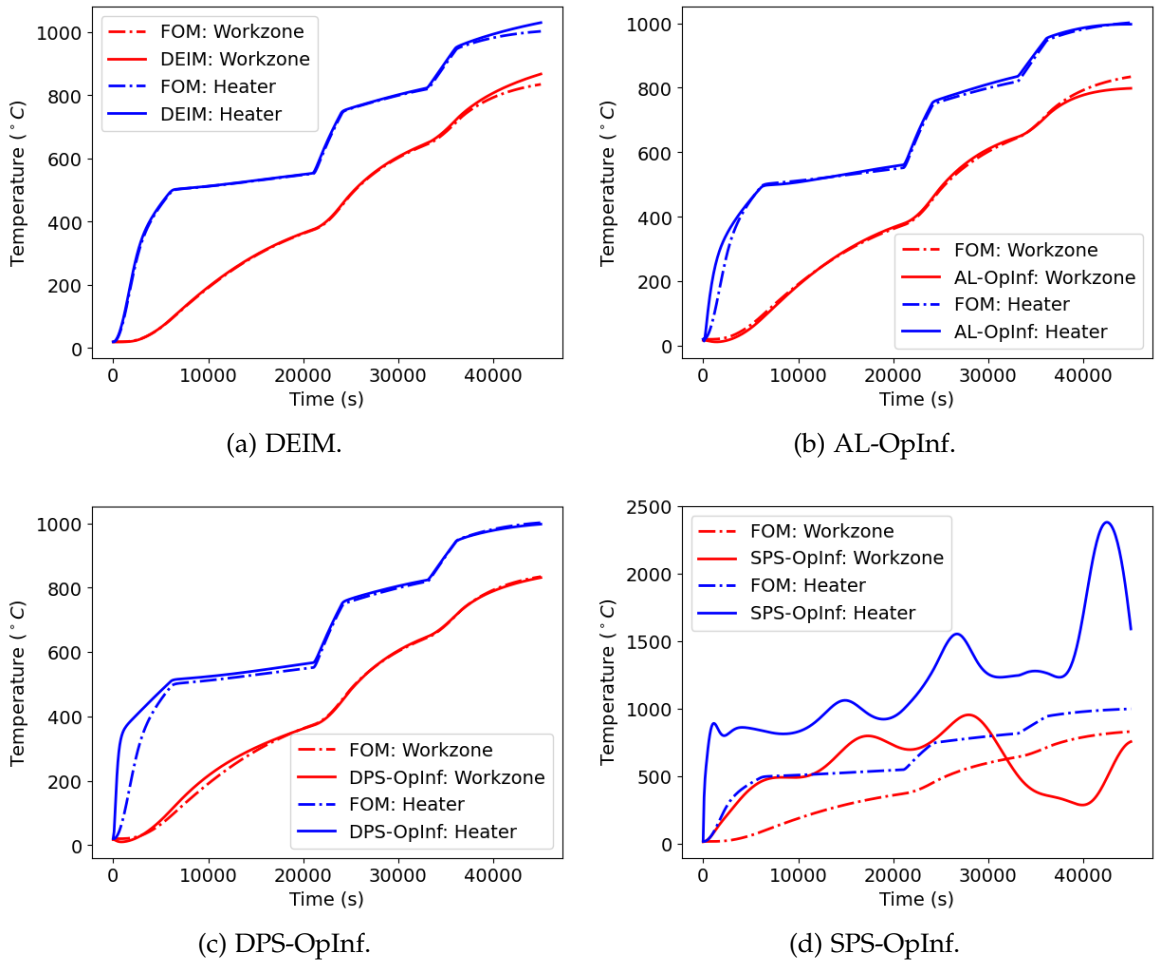
(a) DEIM.

(b) AL-ANN.

(c) DPS-ANN.

(d) SPS-ANN.

Figure 7.31.: The comparison between the solution trajectories of the FOM, DEIM, and ANN-based ROMs in the 3-stage heating. $N_r = 22$, $N_m = 100$, $N_s = 16000$. At the virtual-sensor positions, the prediction made by the AL-ANN is the best and comparable to the prediction by the DEIM. The second place is taken by the DPS-ANN while the SPS-ANN is the worst one. However, both of their accuracy are significantly worse than the AL-ANN's.



(a) DEIM.

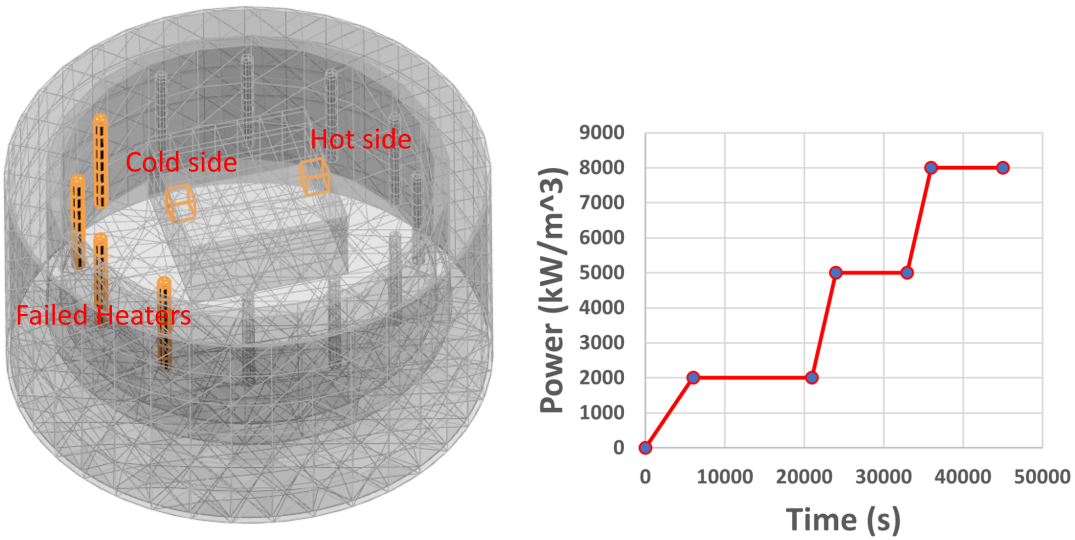
(b) AL-OpInf.

(c) DPS-OpInf.

(d) SPS-OpInf.

Figure 7.32.: The comparison between the solution trajectories of the FOM, DEIM, and OpInf-based ROMs in the 3-stage heating. $N_r = 22$, $N_m = 100$, $N_s = 5000$. At the virtual-sensor positions, the predictions made by the AL-OpInf and DPS-OpInf are comparably good and their quality is close to the prediction by the DEIM-ROM. The SPS-OpInf is the worst.

7.2.7.2. Failed heating



(a) Two virtual sensors are placed at: (1) the cold side (2) the hot side.

(b) The 3-stage heating profile for the working heaters.

Figure 7.33.: The virtual sensor positions and heating profile in the second use case.

In this use case, we assume two out of six heaters are failed during the process. Due to the failure, the temperature field in the furnace is no longer symmetric. Assuming that the final temperature at $t = 45000\ s$ is crucial for the success of the material's hot working, we will be interested in knowing the temperature field in the workzone at the final time. As shown in Figure 7.33a, we pick an element on the side close to the failed heaters and an element on the opposite. They are named as "cold side" and "hot side", respectively. During the simulation/prediction, the temperature at these two places will be monitored. For the rest working heaters, they are operated using the same 3-stage heating profile as normal. The convection coefficient is still defined with $50W/\left(m^2 \cdot K\right)$.

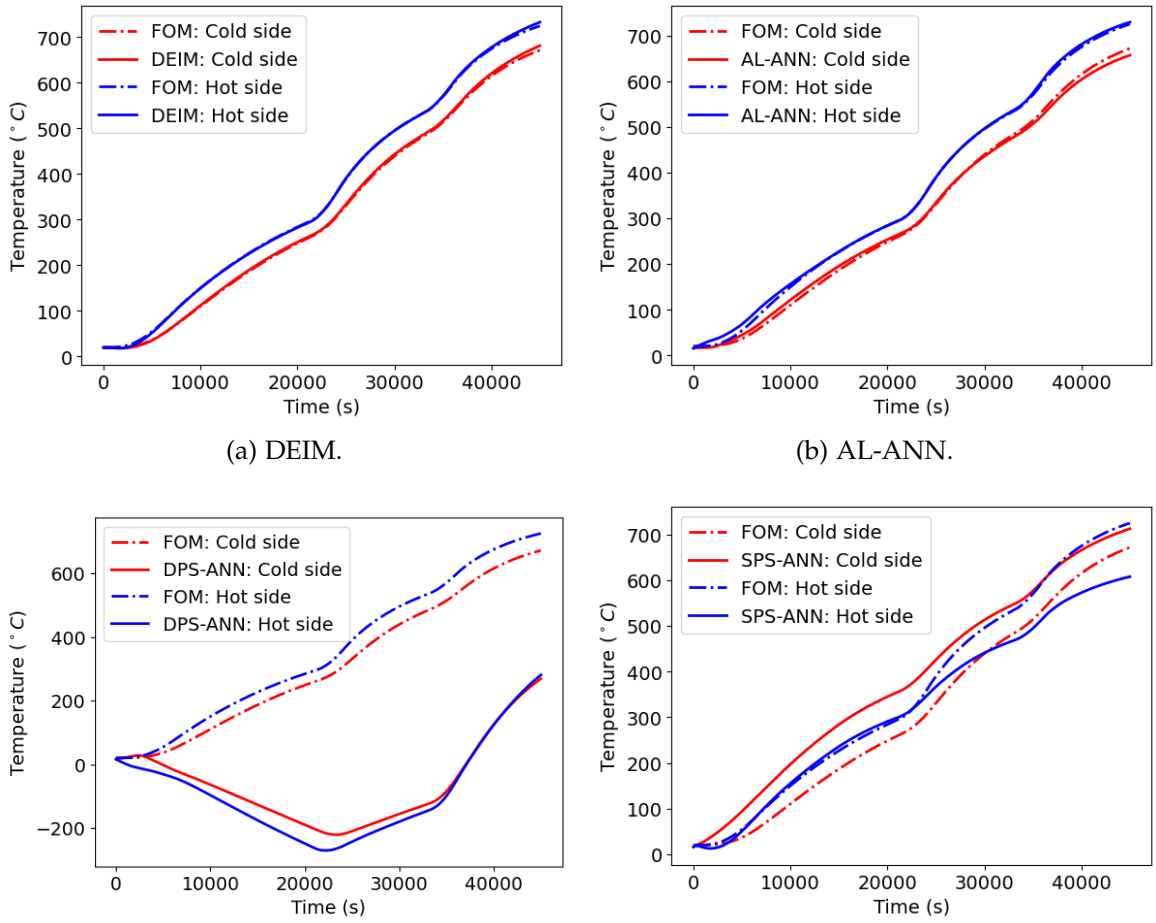
In Figure 7.34, it is observed that the AL-ANN's prediction is almost as good as the ROM constructed by the intrusive hyper-reduction method DEIM. Both of them can distinguish the temperature difference between the cold side and the hot side. The prediction made by the DPS-ANN diverges from the FOM solution. Although the SPS-ANN's prediction does not diverge, the comparison between the two sides is completely wrong.

For the OpInf-ROMs, both the AL-OpInf and the DPS-OpInf give a right comparison for the temperature on the two sides. It is also difficult to say which one is outperforming just by observation. According to the statistical analysis in Table 7.4,

their performance is also comparable. But the SPS-OpInf is much worse than them. Its performance diverges from the FOM solution. This conclusion can also be proven by the data in Table 7.4.

	DEIM	AL-ANN	DPS-ANN	SPS-ANN	AL-OpInf	DPS-OpInf	SPS-OpInf
Mean (°C)	2.48	8.66	31.76	21.32	5.22	5.93	94.49
Std. ((°C)^2)	1.99	16.40	63.99	61.63	12.43	21.89	218.69
Min. (°C)	2.74e-03	1.31e-06	8.84e-07	3.11e-06	1.17e-06	7.32e-07	4.13e-06
Max. (°C)	10.08	348.93	680.32	629.03	208.38	346.93	5426.79
Median (°C)	2.30	2.60	11.78	4.30	1.88	2.24	25.80

Table 7.4.: The statistical measures for $\boldsymbol{e}_{\text{abs}}$ in the failed heating.



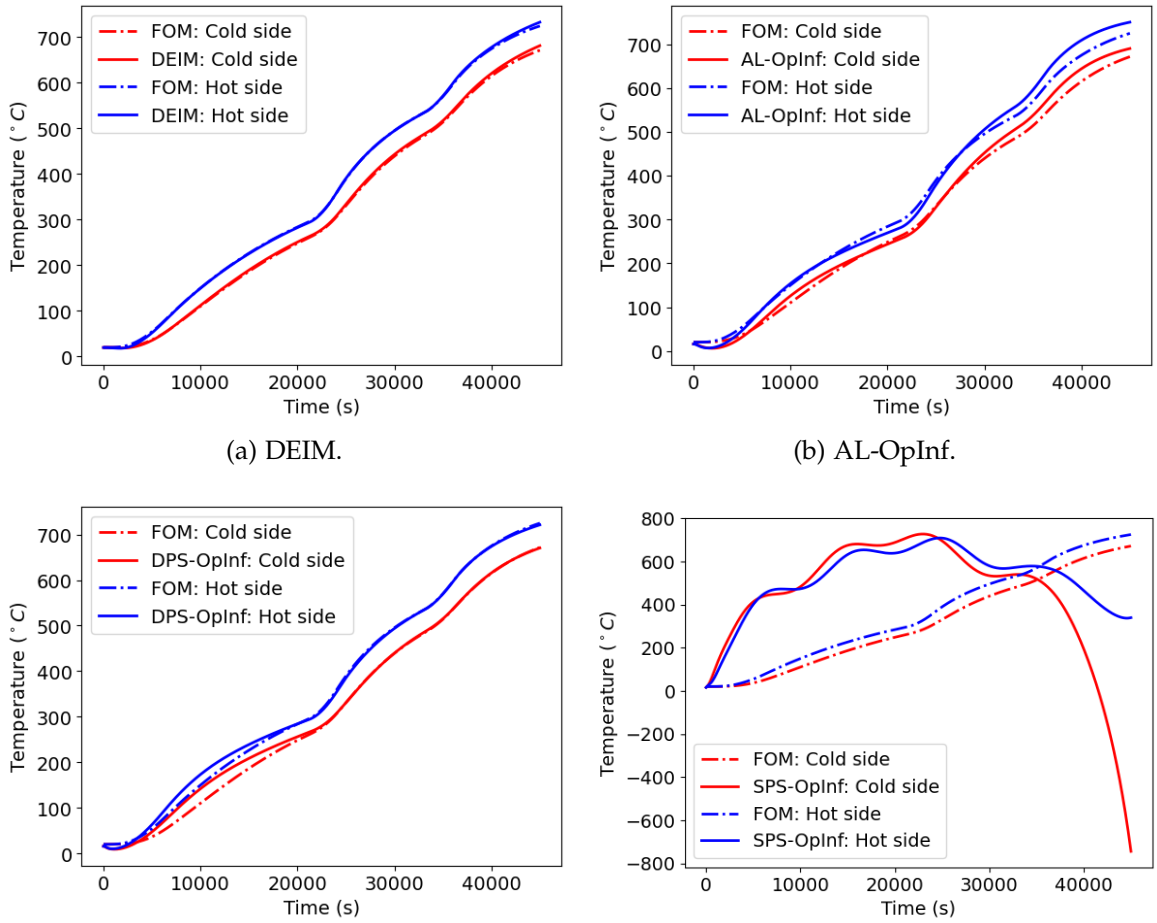
(a) DEIM.

(b) AL-ANN.

(c) DPS-ANN.

(d) SPS-ANN.

Figure 7.34.: The comparison between the solution trajectories of the FOM, DEIM and ANN-based ROMs in the failed heating. $N_r = 22$, $N_m = 100$, $N_s = 16000$. At the virtual-sensor positions, the predictions made by the AL-ANN have the highest accuracy and are comparably good to the DEIM-ROM. The DPS-ANN's prediction diverges from the FOM solution, and the SPS-ANN predicts the temperature comparison incorrectly.



(a) DEIM.

(b) AL-OpInf.

(c) DPS-OpInf.

(d) SPS-OpInf.

Figure 7.35.: The comparison between the solution trajectories of the FOM, DEIM and OpInf-ROMs in the failed heating. $N_r = 22$, $N_m = 100$, $N_s = 5000$. At the virtual-sensor positions, the AL-OpInf and DPS-OpInf can predict with the similar accuracy, which is acceptably lower than the DEIM. However, the SPS-OpInf's prediction has terrible accuracy.

7.2.8. Experiment discussions and conclusions

The experiment model (vacuum furnace) in this section has relatively realistic settings, and similar models can be frequently found in industrial use cases. Using this model, we can better assess the applicability of different methods.

Data sampling strategy A significant difference between this experiment model and the previous experiment model is that the simulated system has a high-dimensional parameter space. The experiment results also show some differences facing this new challenge.

- For the ANN-based MOR, similar to the observation made in the experiments of the thermal block model, the ROMs' performance is remarkably influenced by training data. It is observed that with the same number of data, the ANN-ROM constructed with the JSS data has the smallest 97%-confident ROM error which is 1.00%. This is much smaller than the ANN-ROM trained with the DPS data, which is 11.56%. The worst is the ANN-ROM trained with the SPS data, whose 97%-confident ROM error is 29.15%.
- The OpInf-ROM trained by the JSS data has a 97%-confident ROM error of 1.94%, which takes an advantage to the rest OpInf-ROMs. The second place is taken by the OpInf-ROM trained with the DPS data, whose 97%-confident ROM error is 7.54%. The OpInf-ROM trained with the SPS data has the largest 97%-confident ROM error, which is 43.22%.

Through the tests, we see the data sampled by the JSS method provide more diverse information to the training process. By using the JSS data, we can construct a ROM with a higher confident accuracy using fewer training samples.

Snapshot selection strategy Similar to the thermal block model, we observe that random snapshot selection is the slowest for constructing ROMs, and AL methods can generally accelerate the convergence of the algorithm. Among all the investigated methods, the QBC method makes the best balance between practicality and effectiveness. The MMSE method requires to have a good estimation for the ROM's MSE. If this requirement can be fulfilled, it is believed to be the most effective method. However, learning an accurate error estimator from a limited number of the test samples in PAC validation is a tough task. Last but not least, the PL method is effective for the ANN but not the OpInf-ROM in this experiment.

Case-dependent validation Here we conclude the ROM performance in case-dependent validation.

- For the ANN-based approaches, based on Figure 7.31, Figure 7.34, Table 7.3, and Table 7.4, a fact we can confirm is that the ANN-ROM trained by the AL algorithm always has the best performance. Solely based on Figure 7.31 and Figure 7.34, it is difficult to say which one is better between the SPS-ANN and the DPS-ANN. However, if we check the statistical analysis in Table 7.3 and Table 7.4, the indicators of the DPS-ANN are always better than the SPS-ANN's. Therefore, we can only draw a conclusion that the DPS-ANN is generally better than the SPS-ANN but sometimes performs worse at the virtual-sensor positions.
- For the OpInf-based approaches, based on Figure 7.32, Figure 7.35, we can get a first impression that the AL-OpInf and the DPS-OpInf have similar performances at the positions of the virtual sensors. In Table 7.3 and Table 7.4, we can see the AL-OpInf is generally better than the DPS-OpInf. The SPS-OpInf is worse than them. Since in this experiment, there is no temperature-dependent property, the biggest challenge comes from the high-dimensional parameter space. As a sampling method lacking of sample diversity in the parameter space, it is not out of our expectation that the SPS-OpInf performs the worst.

Conclusion With this relatively realistic numerical model, once again, we see that our AL-MOR method improves the accuracy and the confidence of different data-driven ROMs. Facing the challenge from the high-dimensional parameter space, the ROMs constructed with AL-MOR have consistently good performances in PAC validation. Furthermore, even if we compare the ROMs constructed with AL-MOR to the intrusive ROM constructed with DEIM, the differences are acceptable. Thinking of its easy implementation, a ROM constructed with AL-MOR will be a good alternative to the DEIM-ROM.

7.3. Summary

In this chapter, we systematically investigate different MOR methods through a series of numerical experiments. Two numerical models are used to perform these experiments. In the first numerical model, the system's material property is dependent on temperature and has a sharp change in the high temperature region. The second numerical model is a more realistic furnace model, which has a large number of controllable parameters. Through the experiments, we can analyze the effectiveness, applicability and compatibility of the proposed AL-MOR method.

Effectiveness The effectiveness indicates how much improvement we can gain by integrating an existing ML-based MOR method into the proposed AL-MOR algorithm. According to the experiment results, we can expect that the proposed algorithm can enhance the ML-based MOR methods that are purely dependent on data. The methods belong to this class, including but not limited to the RKNN in this thesis, Deep Neural Network in [152], GPR in [94], Dynamic Mode Decomposition [153], Residual Network in [154]. These methods learn latent dynamics from training data without any guidance from physics knowledge. Their ROMs' quality strongly relies on training data's quality. Therefore, the good data distribution provided by the JSS method can significantly improve the ROMs' performance in the whole solution and parameter space.

Another class of methods are aware of a hypothetical form of FOM equations. They use it to guide/restrict the training of ROMs. They are an important member of the so-called physics-informed-Machine-Learning-based MOR method. Apart from the OpInf method introduced in this thesis, the Physics-Informed Neural Network in [155] and Sparse Identification of Nonlinear Dynamics in [156] also can be included to this class. They reduce the flexibility of ML models with physics constraints. In return, their models have better generalization outside training data. Because of this, the influence of the distribution and the diversity of training data is minimized. As a result, we do not see a significant improvement in the OpInf method in the experiments. We cautiously extend this conclusion to other methods belonging to this class.

Applicability The applicability of the proposed method is mainly decided by the type of the full order problem. The one-step snapshot used by the JSS method makes sense only if the FOM is a time-invariant system. However, for a time-variant system:

$$\dot{\boldsymbol{y}} = \boldsymbol{f}(\boldsymbol{y}, t; \boldsymbol{\mu}), \tag{7.51}$$

such one-step snapshots are not feasible anymore. This is because the update of the system state depends not only on the current system state and input parameters but also on the time of observation. Besides the time-variant system, FOMs having discontinuity in their parameter spaces and solution spaces will also be problematic. However, luckily, they are rarely found in industrial use cases.

Compatibility The compatibility refers to how much extra effort is needed to integrate an existing ML-based MOR method into the proposed algorithm. A foreseeable incompatibility is caused by the architecture of the ML model that is used to identify the ROM. Specifically, ML models with recurrent architecture will be challenging to be integrated into the proposed algorithm. Since the one-step snapshots generated by

the JSS method do not belong to the same trajectory, recurrent ML models cannot use them as data sequence for the training process. The ROMs belonging to this category include but are not limited to Recurrent Neural Network (RNN) in [21, 20, 22], Long-Short-Term Memory (LSTM) in [23, 157], Neural ODE in [157], etc. For them, a potential solution is to increase the number of steps used in the JSS method.

8. Conclusions and Outlook

In this work, aiming to enhance conventional approaches of non-intrusive Model Order Reduction (MOR), we propose an Active-Learning-based Model Order Reduction (AL-MOR) method. The proposed method can generate high-quality training data for constructing Machine-Learning-based Reduced Order Models (ROMs). The training data generated by the new sampling method, Joint Space Sampling (JSS), have a better distribution in both the parameter space and the reduced solution space. Using the data sampled by JSS as the core, an Active Learning (AL) algorithm is employed. They together form the AL-MOR method. The AL-MOR method focuses on providing an efficient, automatic, and unbiased way to construct non-intrusive ROMs.

In the first step in the proposed method, we use Proper Orthogonal Decomposition (POD) to construct a reduced space. We start with determining the size of the reduced space. For this step, we propose a method to automatically determine the reduced dimension based on the projection error plots, as an extension of the work in [39]. By using our method, one is allowed to limit the projection error with a reduced space that is as small as possible.

Afterwards, we need to identify a ROM to approximate the latent dynamic in the reduced space. We show that the instability and the inaccuracy of conventional Machine-Learning-based ROMs is partially caused by the bad distribution of training data. The disadvantages of using two conventional training data sampling methods, i.e., Static Parameter Sampling (SPS) and Dynamic Parameter Sampling (DPS), are pointed out. To produce training data with a good distribution in both the parameter space and the reduced solution space, we propose to sample in a joint space consisting of a predefined parameter space and an estimated reduced solution space. The new sampling method is named JSS.

Next, we prepare a large number of joint samples in a data pool with the JSS method. Each of them defines an initial state and a group of system parameters for a one-step simulation of the Full Order Model (FOM). In each iteration of the AL algorithm, some joint samples will be selected from the data pool. The one-step simulations defined by them will be performed. The simulation results are then used as training data to identify the ROM. The training data will be extended with the proceeding of the algorithm, and the ROM will be refined.

We test three kinds of AL-based strategies as alternative methods to select the

joint samples to be added in each iteration. The first is the Maximum Mean Squared Error (MMSE), where we create an error estimator of the ROM Mean Squared Errors (MSEs) on the remaining candidate joint samples in the data pool. The joint samples maximizing the ROM error will be chosen to define the new simulations in the next iteration. To construct such an error estimator, we test Radial Basis Function (RBF) interpolation and Gaussian Process Regression (GPR). However, in the research, we observe that it is difficult to estimate the ROM MSE accurately. The inaccurate error estimation makes the efficiency of the active selection unstable. Another investigated AL method is the Query by Committee (QBC) method [102, 103, 104]. In our experiments, this model-uncertainty-based method effectively accelerates the convergence of the ROM construction. The last investigated AL method is the Passive Learning (PL) method [114]. Within this method, in the new iteration, we always include the joint samples that are the most distant to the samples in the current training data. The experiments found that the PL method is not very effective for Operator Inference (OpInf).

As a final conclusion for the selection strategy, before an accurate error estimator is available for the MMSE method, the QBC method is considered the best strategy for sample selection.

Last but not least, a case-independent validation method based on Probably Approximately Correct (PAC) learning is proposed. Using this method, we can evaluate a generalized accuracy and confidence of the ROM, which can better decide the acceptance of the refined ROM during the iterations.

In sum, we get a novel AL-MOR method based on JSS and AL. The method is non-intrusive and is therefore easy to be applied even without access to the original solver of the FOM. This is illustrated in Figure 8.1. According to the results from the numerical experiments, we would emphasize these advantages of the proposed method:

- By using data collected with the JSS method, a constructed ROM can predict accurately and stably. The improvement brought by the JSS method is dominant in the whole method package in this thesis.
- By using the concept of AL, we can strategically extend training data by picking new samples from a data pool created by the JSS method. As a result, the convergence rate of ROM construction is accelerated.
- By using PAC validation, we can assess a ROM without bias towards specific use cases. The PAC scores are good indicators for terminating the iteration.

The non-intrusive ROMs constructed with the AL-MOR method can be as good as the intrusive ROMs constructed with the Discrete Empirical Interpolation Method

(DEIM). Besides, the proposed method only requires little human interaction and are therefore very promising to be employed in industrial settings.

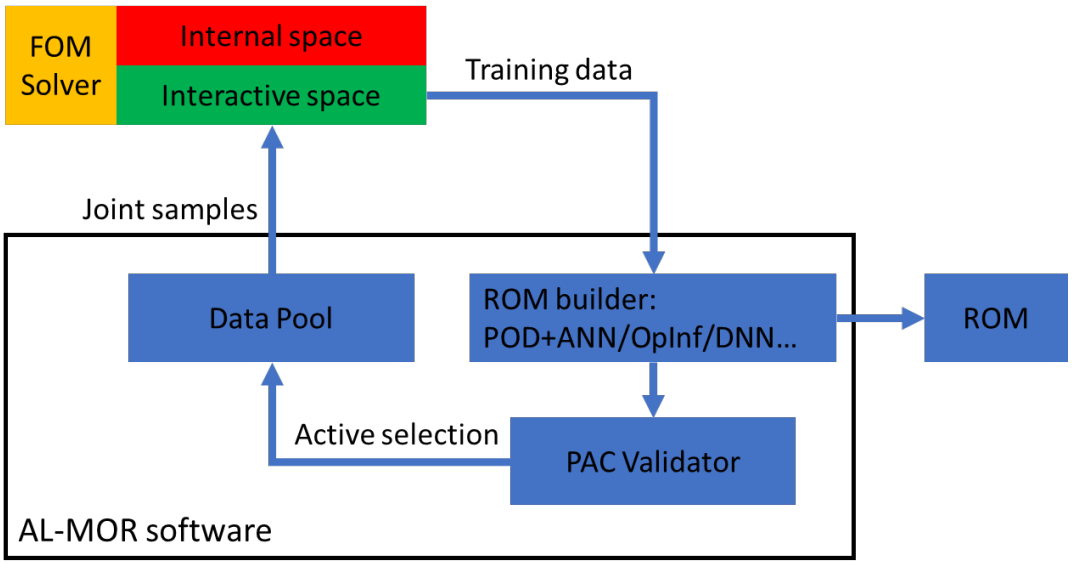


Figure 8.1.: The software architecture of the ROM creator based on the AL-MOR method.

Based on the research results, we believe the proposed AL-MOR method still has potential improvement. Some possible directions for future work are presented next:

- It would be interesting to investigate how to minimize the number of snapshots needed for estimating the reduced solution space.
- The way for estimating the reduced solution space is still elementary. Can we estimate the reduced solution space more accurately? This can give us improvement in three aspects:
  - In the sample pool, we will have fewer samples that are actually non-reachable for the FOM. The identified ROM will be, therefore, more specialized for the given full order problem.
  - By discarding the non-reachable snapshots from the training data, we can further reduce the time needed for training the Machine Learning (ML) model.
  - Since the validation samples are also collected from the same space, by using the validation samples collected from the better estimated space, the evaluation for the ROM performance will be more meaningful to users.

- Can we find a more accurate way to predict/estimate the ROM error? As we can see in the numerical experiments, the effectiveness of the MMSE method strongly relies on the accuracy of the error estimation. If an accurate error estimator is available, we can construct the ROM in a more efficient way.
- In PAC validation, currently, we can only evaluate the confidence and accuracy of the one-step-ROM prediction. It will be very useful if we can find a way to evaluate the confidence and accuracy of the recurrent prediction in the temporal space given with the desired number of prediction steps.
- If we can estimate how many training samples are needed given with a desired ROM confident accuracy, we can be more strategical while deciding the snapshot increment size for the AL algorithm.
- There are also other query strategies specializing in querying unlabelled data in batch mode [158, 159, 160]. It would be interesting to investigate their performance in the framework of AL-MOR.

A. Appendix

A.1. Proof of Chernoff's bound

Here we consider a Bernoulli random variable x. If we assume $E[x] = p$, we will know that $Var[x] = p(1-p)$. We further assume we have n observed samples, then we can define the empirical mean of x as:

$$\hat{E}_n = \frac{1}{n}\sum_{i=1}^{n} x_i. \tag{A.1}$$

Also we know that $E[\hat{E}_n] = E[x] = p$ and $Var[\hat{E}_n] = p(1-p)/n$.

The Chernoff's bounds tells us the inequality:

$$Pr(|\hat{E}_n - E[\hat{E}_n]| < \epsilon) > 1 - \sigma \tag{A.2}$$

holds for any accuracy $\epsilon \in (0,1)$ and confidence $\sigma \in (0,1)$ if that at least:

$$n \geq \frac{1}{2\epsilon^2} ln\left(\frac{2}{\sigma}\right) \tag{A.3}$$

independent samples are observed.

Since it is a binomial distribution, we know:

$$Pr(\hat{E}_n > p + \epsilon) = Pr(n\hat{E}_n > n(p+\epsilon)) = \sum_{k>n(p+\epsilon)}^{n} \binom{n}{k} p^k (1-p)^{n-k} \tag{A.4}$$

and

$$Pr(\hat{E}_n < p - \epsilon) = Pr(n\hat{E}_n < n(p-\epsilon)) = \sum_{k=0}^{k\leq n(p-\epsilon)} \binom{n}{k} p^k (1-p)^{n-k}. \tag{A.5}$$

Chernoff provided a bound [161] for them:

$$Pr(\hat{E}_n \geq p + \epsilon) \leq e^{-2n\epsilon^2} \tag{A.6}$$

and

$$Pr(\hat{E}_n \leq p - \epsilon) \leq e^{-2n\epsilon^2}. \tag{A.7}$$

We know that:

$$\begin{aligned} Pr(|\hat{E}_n - E[\hat{E}_n]| \geq \epsilon) &= Pr(|\hat{E}_n - p| \geq \epsilon) \\ &= Pr(\hat{E}_n \geq p + \epsilon) + Pr(\hat{E}_n \leq p - \epsilon). \end{aligned} \tag{A.8}$$

Therefore, we have:

$$Pr(|\hat{E}_n - E[\hat{E}_n]| \geq \epsilon) \leq 2e^{-2n\epsilon^2} \tag{A.9}$$

and further:

$$Pr(|\hat{E}_n - E[\hat{E}_n]| < \epsilon) > 1 - 2e^{-2n\epsilon^2}. \tag{A.10}$$

Inequality A.2 will hold if:

$$\begin{aligned} 1 - \sigma \leq & 1 - 2e^{-2n\epsilon^2} \\ & \Updownarrow \\ n \geq & \frac{1}{2\epsilon^2} ln\left(\frac{2}{\sigma}\right). \end{aligned} \tag{A.11}$$

This conclusion can be extended to variables with generic distribution using Hoeffding Inequality [162].

Finally, we get:

$$s_{\text{PAC}} = \frac{1}{2\epsilon^2} ln\left(\frac{2}{\sigma}\right). \tag{A.12}$$

Glossary

AL Active Learning.

AL-MOR Active-Learning-based Model Order Reduction.

ANN Artificial Neural Network.

DBAL Diverse mini-Batch Active Learning.

DEIM Discrete Empirical Interpolation Method.

DOF degree of freedom.

DPS Dynamic Parameter Sampling.

DT Digital Twin.

EENN Explicit Euler Neural Network.

FOM Full Order Model.

FPS Farthest Point Sampling.

GPR Gaussian Process Regression.

IVP Initial Value Problem.

JSS Joint Space Sampling.

LSTM Long-Short-Term Memory.

LTI Linear Time Invariant.

Max. Maximum.

Min. Minimum.

ML Machine Learning.

MLP Multilayer Perceptron.

MMSE Maximum Mean Squared Error.

MOR Model Order Reduction.

MSE Mean Squared Error.

NNM Nonlinear Normal Mode.

ODE Ordinary Differential Equation.

OpInf Operator Inference.

PAC Probably Approximately Correct.

PCA Principal Component Analysis.

PL Passive Learning.

POD Proper Orthogonal Decomposition.

QBC Query by Committee.

RBF Radial Basis Function.

RE random-extended.

ReLU Rectified Linear Unit.

RHS right hand side.

RK Runge-Kutta.

RK1 1^{st}-order Runge-Kutta.

RK4 4^{th}-order Runge-Kutta.

RKNN Runge-Kutta Neural Network.

RNN Recurrent Neural Network.

ROM Reduced Order Model.

SINDy Sparse Identification of Nonlinear Dynamics.

SPS Static Parameter Sampling.

Std. Standard Deviation.

SVD Singular Value Decomposition.

Bibliography

[1] J. Bélanger, P. Venne, and J.-N. Paquin. "The what, where and why of real-time simulation". In: *Planet Rt* 1.1 (2010), pp. 25–29.

[2] A. Rasheed, O. San, and T. Kvamsdal. "Digital Twin: Values, Challenges and Enablers". In: *arXiv preprint arXiv:1910.01719* (2019).

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

[4] A. Antoulas. *Approximation of Large-Scale Dynamical Systems*. Society for Industrial and Applied Mathematics, 2005.

[5] M. Hinze and S. Volkwein. "Proper orthogonal decomposition surrogate models for nonlinear dynamical systems: Error estimates and suboptimal control". In: *Dimension reduction of large-scale systems*. Springer, 2005, pp. 261–306.

[6] K. Willcox and J. Peraire. "Balanced model reduction via the proper orthogonal decomposition". In: *AIAA journal* 40.11 (2002), pp. 2323–2330.

[7] F. Chinesta, A. Huerta, G. Rozza, and K. Willcox. "Model order reduction". In: *Encyclopedia of Computational Mechanics* (2016).

[8] U. Baur, P. Benner, and L. Feng. "Model order reduction for linear and nonlinear systems: a system-theoretic perspective". In: *Archives of Computational Methods in Engineering* 21.4 (2014), pp. 331–358.

[9] R. W. Freund. "Model reduction methods based on Krylov subspaces". In: *Acta Numerica* 12 (2003), pp. 267–319.

[10] P. J. Heres. "Robust and efficient Krylov subspace methods for model order reduction". In: (2005).

[11] S. Gugercin and A. C. Antoulas. "A survey of model reduction by balanced truncation and some new results". In: *International Journal of Control* 77.8 (2004), pp. 748–766.

[12] R. Pinnau. "Model reduction via proper orthogonal decomposition". In: *Model order reduction: theory, research aspects and applications*. Springer, 2008, pp. 95–109.

[13] S. Wold, K. Esbensen, and P. Geladi. "Principal component analysis". In: *Chemometrics and intelligent laboratory systems* 2.1-3 (1987), pp. 37–52.

[14] Z. Bai and L. Peng. "Non-intrusive nonlinear model reduction via machine learning approximations to low-dimensional operators". In: *Advanced Modeling and Simulation in Engineering Sciences* 8.1 (2021), pp. 1–24.

[15] B. Peherstorfer and K. Willcox. "Data-driven operator inference for nonintrusive projection-based model reduction". In: *Computer Methods in Applied Mechanics and Engineering* 306 (2016), pp. 196–215.

[16] A. Alla and J. N. Kutz. "Nonlinear model order reduction via dynamic mode decomposition". In: *SIAM Journal on Scientific Computing* 39.5 (2017), B778–B796.

[17] O. M. Agudelo, J. J. Espinosa, and B. De Moor. "Acceleration of nonlinear POD models: a neural network approach". In: *2009 European Control Conference (ECC)*. IEEE. 2009, pp. 1547–1552.

[18] Q. Wang, J. S. Hesthaven, and D. Ray. "Non-intrusive reduced order modeling of unsteady flows using artificial neural networks with application to a combustion problem". In: *Journal of computational physics* 384 (2019), pp. 289–307.

[19] A. Adel and K. Salah. "Model order reduction using artificial neural networks". In: *2016 IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. IEEE. 2016, pp. 89–92.

[20] J. N. Kani and A. H. Elsheikh. "DR-RNN: A deep residual recurrent neural network for model reduction". In: *arXiv preprint arXiv:1709.00939* (2017).

[21] Q. Wang, N. Ripamonti, and J. S. Hesthaven. "Recurrent neural network closure of parametric POD-Galerkin reduced-order models based on the Mori-Zwanzig formalism". In: *Journal of Computational Physics* (2020), p. 109402.

[22] L. Wu and L. Noels. "Recurrent Neural Networks (RNNs) with dimensionality reduction and break down in computational mechanics; application to multi-scale localization step". In: *Computer Methods in Applied Mechanics and Engineering* 390 (2022), p. 114476. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2021.114476. URL: https://www.sciencedirect.com/science/article/pii/S0045782521006940.

[23] A. T. Mohan and D. V. Gaitonde. "A deep learning based approach to reduced order modeling for turbulent flow control using LSTM neural networks". In: *arXiv preprint arXiv:1804.09269* (2018).

[24] B. Foad, R. Elzohery, and D. R. Novog. "Demonstration of combined reduced order model and deep neural network for emulation of a time-dependent reactor transient". In: *Annals of Nuclear Energy* 171 (2022), p. 109017.

[25] O. San and R. Maulik. "Neural network closures for nonlinear model order reduction". In: *Advances in Computational Mathematics* 44.6 (2018), pp. 1717–1750.

[26] M. A. Grepl and A. T. Patera. "A posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations". In: *ESAIM: Mathematical Modelling and Numerical Analysis* 39.1 (2005), pp. 157–181.

[27] K. Veroy, C. Prud'Homme, D. Rovas, and A. Patera. "A posteriori error bounds for reduced-basis approximation of parametrized noncoercive and nonlinear elliptic partial differential equations". In: *16th AIAA Computational Fluid Dynamics Conference*. 2003, p. 3847.

[28] T. Bui-Thanh, K. Willcox, and O. Ghattas. "Model reduction for large-scale systems with high-dimensional parametric input space". In: *SIAM Journal on Scientific Computing* 30.6 (2008), pp. 3270–3288.

[29] W. Chen, J. S. Hesthaven, B. Junqiang, Y. Qiu, Z. Yang, and Y. Tihao. "Greedy nonintrusive reduced order model for fluid dynamics". In: *AIAA Journal* 56.12 (2018), pp. 4927–4943.

[30] W. I. T. Uy, Y. Wang, Y. Wen, and B. Peherstorfer. "Active operator inference for learning low-dimensional dynamical-system models from noisy data". In: *arXiv preprint arXiv:2107.09256* (2021).

[31] B. Settles. "Active learning literature survey". In: (2009).

[32] T. Kavzoglu. "Increasing the accuracy of neural network classification using refined training data". In: *Environmental Modelling & Software* 24.7 (2009), pp. 850–858.

[33] M. S. Hansen, S. Kozerke, K. P. Pruessmann, P. Boesiger, E. M. Pedersen, and J. Tsao. "On the influence of training data quality in k-t BLAST reconstruction". In: *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine* 52.5 (2004), pp. 1175–1183.

[34] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang. "Physics-informed machine learning". In: *Nature Reviews Physics* 3.6 (2021), pp. 422–440.

[35] J.-L. Wu, H. Xiao, and E. Paterson. "Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework". In: *Physical Review Fluids* 3.7 (2018), p. 074602.

[36] W. Chen, Q. Wang, J. S. Hesthaven, and C. Zhang. "Physics-informed machine learning for reduced-order modeling of nonlinear problems". In: *Journal of Computational Physics* 446 (2021), p. 110666.

[37] S. Chaturantabut and D. C. Sorensen. "Nonlinear model reduction via discrete empirical interpolation". In: *SIAM Journal on Scientific Computing* 32.5 (2010), pp. 2737–2764.

[38] Q. Zhuang, J. M. Lorenzi, H.-J. Bungartz, and D. Hartmann. "Model order reduction based on Runge–Kutta neural networks". In: *Data-Centric Engineering* 2 (2021).

[39] M. Zhu and A. Ghodsi. "Automatic dimensionality selection from the scree plot via the use of profile likelihood". In: *Computational Statistics & Data Analysis* 51.2 (2006), pp. 918–930.

[40] W. H. Schilders, H. A. Van der Vorst, and J. Rommes. *Model order reduction: theory, research aspects and applications*. Vol. 13. Springer, 2008.

[41] B. Salimbahrami and B. Lohmann. "Krylov subspace methods in linear model order reduction: introduction and invariance properties". In: *Sci. Rep.. Inst. of Automation*. Citeseer. 2002.

[42] T. Breiten and T. Damm. "Krylov subspace methods for model order reduction of bilinear control systems". In: *Systems & Control Letters* 59.8 (2010), pp. 443–450.

[43] A. Astolfi. "Model reduction by moment matching for linear and nonlinear systems". In: *IEEE Transactions on Automatic Control* 55.10 (2010), pp. 2321–2336.

[44] V. Mehrmann and T. Stykel. "Balanced truncation model reduction for large-scale systems in descriptor form". In: *Dimension Reduction of Large-Scale Systems*. Springer, 2005, pp. 83–115.

[45] P. Benner and A. Schneider. "Balanced truncation model order reduction for LTI systems with many inputs or outputs". In: *Proceedings of the 19th international symposium on mathematical theory of networks and systems–MTNS*. Vol. 5. 2010.

[46] M. Geuß. "A Black-box method for parametric model order reduction based on matrix interpolation with application to simulation and control". PhD thesis. Technische Universität München, 2015.

[47] J. C. Helton and F. J. Davis. "Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems". In: *Reliability Engineering & System Safety* 81.1 (2003), pp. 23–69.

[48] T.-T. Wong, W.-S. Luk, and P.-A. Heng. "Sampling with Hammersley and Halton points". In: *Journal of graphics tools* 2.2 (1997), pp. 9–24.

[49] S. Burhenne, D. Jacob, and G. P. Henze. "Sampling based on Sobol'sequences for Monte Carlo techniques applied to building simulations". In: *Proc. Int. Conf. Build. Simulat*. 2011, pp. 1816–1823.

[50] I. Jolliffe. *Principal Component Analysis, Second Edition*. Springer, 2002, pp. 111–127.

[51] J. K. White et al. "A trajectory piecewise-linear approach to model order reduction of nonlinear dynamical systems". PhD thesis. Massachusetts Institute of Technology, 2003.

[52] D. Amsallem, M. J. Zahr, and C. Farhat. "Nonlinear model order reduction based on local reduced-order bases". In: *International Journal for Numerical Methods in Engineering* 92.10 (2012), pp. 891–916.

[53] B. Kramer and K. E. Willcox. "Nonlinear model order reduction via lifting transformations and proper orthogonal decomposition". In: *AIAA Journal* 57.6 (2019), pp. 2297–2307.

[54] S. Jain and P. Tiso. "Hyper-reduction over nonlinear manifolds for large nonlinear mechanical systems". In: *Journal of Computational and Nonlinear Dynamics* 14.8 (2019), p. 081008.

[55] P. Astrid, S. Weiland, K. Willcox, and T. Backx. "Missing point estimation in models described by proper orthogonal decomposition". In: *IEEE Transactions on Automatic Control* 53.10 (2008), pp. 2237–2251.

[56] L. Ljung. "System identification". In: *Wiley encyclopedia of electrical and electronics engineering* (1999), pp. 1–19.

[57] S. Chaturantabut and D. C. Sorensen. "Application of POD and DEIM on dimension reduction of non-linear miscible viscous fingering in porous media". In: *Mathematical and Computer Modelling of Dynamical Systems* 17.4 (2011), pp. 337–353.

[58] M. Barrault, Y. Maday, N. C. Nguyen, and A. T. Patera. "An 'empirical interpolation'method: application to efficient reduced-basis discretization of partial differential equations". In: *Comptes Rendus Mathematique* 339.9 (2004), pp. 667–672.

[59] R. Ştefănescu and I. M. Navon. "POD/DEIM nonlinear model order reduction of an ADI implicit shallow water equations model". In: *Journal of Computational Physics* 237 (2013), pp. 95–114.

[60] D. Xiao, F. Fang, A. G. Buchan, C. C. Pain, I. M. Navon, J. Du, and G. Hu. "Non-linear model reduction for the Navier–Stokes equations using residual DEIM method". In: *Journal of Computational Physics* 263 (2014), pp. 1–18.

[61] F. Ghavamian, P. Tiso, and A. Simone. "POD–DEIM model order reduction for strain-softening viscoplasticity". In: *Computer Methods in Applied Mechanics and Engineering* 317 (2017), pp. 458–479.

[62] D. Bonomi, A. Manzoni, and A. Quarteroni. "A matrix DEIM technique for model reduction of nonlinear parametrized problems in cardiac mechanics". In: *Computer Methods in Applied Mechanics and Engineering* 324 (2017), pp. 300–326.

[63] B. Peherstorfer, D. Butnaru, K. Willcox, and H.-J. Bungartz. "Localized discrete empirical interpolation method". In: *SIAM Journal on Scientific Computing* 36.1 (2014), A168–A192.

[64] R. Stefanescu and A. Sandu. "A Goal-Oriented Adaptive Discrete Empirical Interpolation Method". In: *arXiv e-prints* (2019), arXiv–1901.

[65] A. Hochman, B. N. Bond, and J. K. White. "A stabilized discrete empirical interpolation method for model reduction of electrical, thermal, and micro-electromechanical systems". In: *2011 48th ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE. 2011, pp. 540–545.

[66] Y. Sato, M. Clemens, and H. Igarashi. "Adaptive subdomain model order reduction with discrete empirical interpolation method for nonlinear magneto-quasi-static problems". In: *IEEE transactions on magnetics* 52.3 (2015), pp. 1–4.

[67] B. C. Csáji et al. "Approximation with artificial neural networks". In: *Faculty of Sciences, Etvs Lornd University, Hungary* 24.48 (2001), p. 7.

[68] F. Scarselli and A. C. Tsoi. "Universal approximation using feedforward neural networks: A survey of some existing methods, and some new results". In: *Neural networks* 11.1 (1998), pp. 15–37.

[69] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. Springer, 2009.

[70] K. Hornik. "Approximation capabilities of multilayer feedforward networks". In: *Neural networks* 4.2 (1991), pp. 251–257.

[71] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang. "The expressive power of neural networks: A view from the width". In: *Advances in neural information processing systems* 30 (2017).

[72] S. A. Teukolsky, B. P. Flannery, W. Press, and W. Vetterling. "Numerical recipes in C". In: *SMR* 693.1 (1992), pp. 59–70.

[73] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[74] M. Li, T. Zhang, Y. Chen, and A. J. Smola. "Efficient mini-batch training for stochastic optimization". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2014, pp. 661–670.

[75] J. Qi, J. Du, S. M. Siniscalchi, X. Ma, and C.-H. Lee. "On mean absolute error for deep neural network based vector-to-vector regression". In: *IEEE Signal Processing Letters* 27 (2020), pp. 1485–1489.

[76] P. J. Huber. "Robust estimation of a location parameter". In: *Breakthroughs in statistics*. Springer, 1992, pp. 492–518.

[77] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[78] N. Buduma and N. Locascio. *Fundamentals of deep learning: Designing next-generation machine intelligence algorithms*. " O'Reilly Media, Inc.", 2017.

[79] L. Prechelt. "Early stopping-but when?" In: *Neural Networks: Tricks of the trade*. Springer, 1998, pp. 55–69.

[80] K. You, M. Long, J. Wang, and M. I. Jordan. *How Does Learning Rate Decay Help Modern Neural Networks?* 2020. URL: `https://openreview.net/forum?id=r1eOnh4YPB`.

[81] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. "Automatic differentiation in PyTorch". In: (2017).

[82] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: `https://www.tensorflow.org/`.

[83] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[84] S. A. McQuarrie, C. Huang, and K. E. Willcox. "Data-driven reduced-order models via regularised Operator Inference for a single-injector combustion process". In: *Journal of the Royal Society of New Zealand* 51.2 (2021), pp. 194–211.

[85] S. L. Brunton, J. L. Proctor, and J. N. Kutz. "Discovering governing equations from data by sparse identification of nonlinear dynamical systems". In: *Proceedings of the national academy of sciences* 113.15 (2016), pp. 3932–3937.

[86] C. Gu. "QLMOR: A projection-based nonlinear model order reduction approach using quadratic-linear representation of nonlinear systems". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 30.9 (2011), pp. 1307–1320.

[87] E. Qian, B. Kramer, B. Peherstorfer, and K. Willcox. "Lift & learn: Physics-informed machine learning for large-scale nonlinear dynamical systems". In: *Physica D: Nonlinear Phenomena* 406 (2020), p. 132401.

[88] I. V. Tetko, D. J. Livingstone, and A. I. Luik. "Neural network studies. 1. Comparison of overfitting and overtraining". In: *Journal of chemical information and computer sciences* 35.5 (1995), pp. 826–833.

[89] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

[90] R. Tibshirani. "Regression shrinkage and selection via the lasso". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 267–288.

[91] B. N. Bond and L. Daniel. "Guaranteed stable projection-based model reduction for indefinite and unstable linear systems". In: *2008 IEEE/ACM International Conference on Computer-Aided Design*. IEEE. 2008, pp. 728–735.

[92] R. Pulch. "Stability preservation in Galerkin-type projection-based model order reduction". In: *Numerical Algebra, Control and Optimization* 9.1 (2019), p. 23.

[93] B. Settles. "From theories to queries: Active learning in practice". In: *Active learning and experimental design workshop in conjunction with AISTATS 2010*. JMLR Workshop and Conference Proceedings. 2011, pp. 1–18.

[94] M. Guo and J. Hesthaven. "Reduced order modeling for nonlinear structural analysis using Gaussian process regression". In: *Computer Methods in Applied Mechanics and Engineering* 341 (July 2018). DOI: 10.1016/j.cma.2018.07.017.

[95] D. Angluin. "Queries and concept learning". In: *Machine learning* 2.4 (1988), pp. 319–342.

[96] I. Alabdulmohsin. "Learning via Query Synthesis". PhD thesis. 2017.

[97] C. C. Loy, T. M. Hospedales, T. Xiang, and S. Gong. "Stream-based joint exploration-exploitation active learning". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2012, pp. 1560–1567.

[98] V.-L. Nguyen, M. H. Shaker, and E. Hüllermeier. "How to measure uncertainty in uncertainty sampling for active learning". In: *Machine Learning* 111.1 (2022), pp. 89–122.

[99] D. Wu, C.-T. Lin, and J. Huang. "Active learning for regression using greedy sampling". In: *Information Sciences* 474 (2019), pp. 90–105.

[100] D. Mahapatra, B. Bozorgtabar, J.-P. Thiran, and M. Reyes. "Efficient active learning for image classification and segmentation using a sample selection and conditional generative adversarial network". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2018, pp. 580–588.

[101] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn. "Design and analysis of computer experiments". In: *Statistical science* 4.4 (1989), pp. 409–423.

[102] S. Kee, E. Del Castillo, and G. Runger. "Query-by-committee improvement with diversity and density in batch active learning". In: *Information Sciences* 454 (2018), pp. 401–418.

[103] R. Burbidge, J. J. Rowland, and R. D. King. "Active learning for regression based on query by committee". In: *International conference on intelligent data engineering and automated learning*. Springer. 2007, pp. 209–218.

[104] H. S. Seung, M. Opper, and H. Sompolinsky. "Query by committee". In: *Proceedings of the fifth annual workshop on Computational learning theory*. 1992, pp. 287–294.

[105] Z. Shuyang, T. Heittola, and T. Virtanen. "Active learning for sound event detection". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 28 (2020), pp. 2895–2905.

[106] W. Cai, Y. Zhang, and J. Zhou. "Maximizing expected model change for active learning in regression". In: *2013 IEEE 13th international conference on data mining*. IEEE. 2013, pp. 51–60.

[107] G. Zhao, E. Dougherty, B.-J. Yoon, F. Alexander, and X. Qian. "Efficient active learning for Gaussian process classification by error reduction". In: *Advances in Neural Information Processing Systems* 34 (2021).

[108] Y. Guo and D. Schuurmans. "Discriminative batch mode active learning". In: *Advances in neural information processing systems* 20 (2007).

[109] A. Kirsch, J. Van Amersfoort, and Y. Gal. "Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning". In: *Advances in neural information processing systems* 32 (2019).

[110] S. Chakraborty, V. Balasubramanian, and S. Panchanathan. "Adaptive batch mode active learning". In: *IEEE transactions on neural networks and learning systems* 26.8 (2014), pp. 1747–1760.

[111] S. C. Hoi, R. Jin, J. Zhu, and M. R. Lyu. "Batch mode active learning and its application to medical image classification". In: *Proceedings of the 23rd international conference on Machine learning*. 2006, pp. 417–424.

[112] F. Zhdanov. "Diverse mini-batch active learning". In: *arXiv preprint arXiv:1901.05954* (2019).

[113] H.-H. Bock. "Clustering methods: a history of k-means algorithms". In: *Selected contributions in data analysis and classification* (2007), pp. 161–172.

[114] H. Yu and S. Kim. "Passive sampling for regression". In: *2010 IEEE International Conference on Data Mining*. IEEE. 2010, pp. 1151–1156.

[115] G. Y. Kanyongo. "The influence of reliability on four rules for determining the number of components to retain". In: *Journal of Modern Applied Statistical Methods* 5.2 (2005), p. 7.

[116] W. F. Velicer, C. A. Eaton, and J. L. Fava. "Construct explication through factor or component analysis: A review and evaluation of alternative procedures for determining the number of factors or components". In: *Problems and solutions in human assessment* (2000), pp. 41–71.

[117] N. Cliff. "The eigenvalues-greater-than-one rule and the reliability of components." In: *Psychological bulletin* 103.2 (1988), p. 276.

[118] J. L. Horn. "A rationale and test for the number of factors in factor analysis". In: *Psychometrika* 30.2 (1965), pp. 179–185.

[119] R. B. Cattell. "The scree test for the number of factors". In: *Multivariate behavioral research* 1.2 (1966), pp. 245–276.

[120] J. K. Patel and C. B. Read. *Handbook of the normal distribution*. Vol. 150. CRC Press, 1996.

[121] "Chapter 4 Analysis of Dynamical Systems Whose Inputs are Stochastic Processes". In: *Introduction to Stochastic Control Theory*. Ed. by K. J. Åström. Vol. 70. Mathematics in Science and Engineering. Elsevier, 1970, pp. 91–114. DOI: https://doi.org/10.1016/S0076-5392(09)60282-4. URL: https://www.sciencedirect.com/science/article/pii/S0076539209602824.

[122] R. B. McCammon. "Principal component analysis and its application in large-scale correlation studies". In: *The Journal of Geology* 74.5, Part 2 (1966), pp. 721–733.

[123] R. G. Sargent. *Simulation model verification and validation*. Tech. rep. Institute of Electrical and Electronics Engineers (IEEE), 1991.

[124] R. J. Kuether and M. S. Allen. "Validation of nonlinear reduced order models with time integration targeted at nonlinear normal modes". In: *Nonlinear Dynamics, Volume 1*. Springer, 2016, pp. 363–375.

[125] A. Hay, J. T. Borggaard, and D. Pelletier. "Local improvements to reduced-order models using sensitivity analysis of the proper orthogonal decomposition". In: *Journal of Fluid Mechanics* 629 (2009), pp. 41–72.

[126] L. G. Valiant. "A theory of the learnable". In: *Communications of the ACM* 27.11 (1984), pp. 1134–1142.

[127] D. Haussler. *Probably approximately correct learning*. University of California, Santa Cruz, Computer Research Laboratory Santa ..., 1990.

[128] C. Alippi. *Intelligence for embedded systems*. Springer, 2014.

[129] M. Pedergnana, S. G. Garcia, et al. "Smart sampling and incremental function learning for very large high dimensional data". In: *Neural Networks* 78 (2016), pp. 75–87.

[130] A. Lin. "Binary search algorithm". In: *WikiJournal of Science* 2.1 (2019), pp. 1–13.

[131] N. Rubens, M. Elahi, M. Sugiyama, and D. Kaplan. "Active learning in recommender systems". In: *Recommender systems handbook*. Springer, 2015, pp. 809–846.

[132] F. Olsson. "A literature survey of active machine learning in the context of natural language processing". In: (2009).

[133] R. Milk, S. Rave, and F. Schindler. "pyMOR–generic algorithms and interfaces for model order reduction". In: *SIAM Journal on Scientific Computing* 38.5 (2016), S194–S216.

[134] B. Haasdonk. "Convergence rates of the pod–greedy method". In: *ESAIM: Mathematical modelling and numerical Analysis* 47.3 (2013), pp. 859–873.

[135] H. Bao, N. T. Dinh, J. W. Lane, and R. W. Youngblood. "A data-driven framework for error estimation and mesh-model optimization in system-level thermal-hydraulic simulation". In: *Nuclear Engineering and Design* 349 (2019), pp. 27–45.

[136] H. Bhatia, S. N. Petruzza, R. Anirudh, A. G. Gyulassy, R. M. Kirby, V. Pascucci, and P.-T. Bremer. "Data-Driven Estimation of Temporal-Sampling Errors in Unsteady Flows". In: *International Symposium on Visual Computing*. Springer. 2021, pp. 235–248.

[137] J. Reis, J. P. Moitinho de Almeida, P. Diez, and S. Zlotnik. "Error estimation and adaptivity for PGD based on complementary solutions applied to a simple 1D problem". In: *Advanced Modeling and Simulation in Engineering Sciences* 7.1 (2020), pp. 1–22.

[138] G. B. Wright. *Radial basis function interpolation: numerical and analytical developments*. University of Colorado at Boulder, 2003.

[139] F. Lindner, M. Mehl, and B. Uekermann. "Radial basis function interpolation for black-box multi-physics simulations". In: (2017).

[140] G. E. Fasshauer. *Meshfree approximation methods with MATLAB*. Vol. 6. World Scientific, 2007.

[141] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[142] C. E. Rasmussen. "Gaussian processes in machine learning". In: *Summer school on machine learning*. Springer. 2003, pp. 63–71.

[143] D. G. Krige. "A statistical approach to some basic mine valuation problems on the Witwatersrand". In: *Journal of the Southern African Institute of Mining and Metallurgy* 52.6 (1951), pp. 119–139.

[144] E. Schulz, M. Speekenbrink, and A. Krause. "A tutorial on Gaussian process regression: Modelling, exploring, and exploiting functions". In: *Journal of Mathematical Psychology* 85 (2018), pp. 1–16.

[145] Y. Eldar, M. Lindenbaum, M. Porat, and Y. Y. Zeevi. "The farthest point strategy for progressive image sampling". In: *IEEE Transactions on Image Processing* 6.9 (1997), pp. 1305–1315.

[146] C. Moenning and N. A. Dodgson. *Fast marching farthest point sampling*. Tech. rep. University of Cambridge, Computer Laboratory, 2003.

[147] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. "Pointnet++: Deep hierarchical feature learning on point sets in a metric space". In: *Advances in neural information processing systems* 30 (2017).

[148] S. Larsson and V. Thomee. *Partial differential equations with numerical methods*. Vol. 45. Springer, 2003.

[149] X. Glorot, A. Bordes, and Y. Bengio. "Deep sparse rectifier neural networks". In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2011, pp. 315–323.

[150] J. M. Hernández-Lobato. "Balancing flexibility and robustness in machine learning: semi-parametric methods and sparse linear models". In: (2010).

[151] X. Hao, J. Gu, N. Chen, W. Zhang, and Z. Xunwei. "3-D Numerical analysis on heating process of loads within vacuum heat treatment furnace". In: *Applied Thermal Engineering* 28.14-15 (2008), pp. 1925–1931.

[152] S. Fresca and A. Manzoni. "POD-DL-ROM: enhancing deep learning-based reduced order models for nonlinear parametrized PDEs by proper orthogonal decomposition". In: *Computer Methods in Applied Mechanics and Engineering* 388 (2022), p. 114181.

[153] N. B. Erichson, L. Mathelin, J. N. Kutz, and S. L. Brunton. "Randomized dynamic mode decomposition". In: *SIAM Journal on Applied Dynamical Systems* 18.4 (2019), pp. 1867–1891.

[154] O. San, R. Maulik, and M. Ahmed. "An artificial neural network framework for reduced order modeling of transient flows". In: *Communications in Nonlinear Science and Numerical Simulation* 77 (2019), pp. 271–287.

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

[156] K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton. "Data-driven discovery of coordinates and governing equations". In: *Proceedings of the National Academy of Sciences* 116.45 (2019), pp. 22445–22451.

[157] R. Maulik, A. Mohan, B. Lusch, S. Madireddy, P. Balaprakash, and D. Livescu. "Time-series learning of latent-space dynamics for reduced-order model closure". In: *Physica D: Nonlinear Phenomena* 405 (2020), p. 132368.

[158] P. Kumar and A. Gupta. "Active learning query strategies for classification, regression, and clustering: a survey". In: *Journal of Computer Science and Technology* 35.4 (2020), pp. 913–945.

[159] W. Cai, M. Zhang, and Y. Zhang. "Batch mode active learning for regression with expected model change". In: *IEEE transactions on neural networks and learning systems* 28.7 (2016), pp. 1668–1681.

[160] J. O'Neill, S. Jane Delany, and B. MacNamee. "Model-free and model-based active learning for regression". In: *Advances in computational intelligence systems*. Springer, 2017, pp. 375–386.

[161] H. Chernoff. "A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations". In: *The Annals of Mathematical Statistics* (1952), pp. 493–507.

[162] W. Hoeffding. "Probability inequalities for sums of bounded random variables". In: *The collected works of Wassily Hoeffding*. Springer, 1994, pp. 409–426.