A Posteriori Error Estimation and Adaptivity for Model Order Reduction of Large-Scale Systems

Dissertation

zur Erlangung des akademischen Grades

doctor rerum naturalium
(Dr. rer. nat.)

von **M. Sc. Sridhar Chellappa**

geb. am **02.02.1992** in Chennai, Indien

genehmigt durch die Fakultät für Mathematik

der Otto-von-Guericke-Universität Magdeburg

Gutachter: **Dr. Lihong Feng**

Prof. Dr. Peter Benner

Prof. Dr. Yanlai Chen

eingereicht am: **25.04.2022**

Verteidigung am: **13.09.2022**

PUBLICATIONS

The material presented in this thesis have either been published, accepted or submitted for publication.

The material in Chapter 3 is an extended version of the following publications:

[57]: S. Chellappa, L. Feng, V. de la Rubia, and P. Benner, Inf-Sup-Constant-Free State Error Estimator for Model Order Reduction of Parametric Systems in Electromagnetics, e-prints 2104.12802, arXiv, 2021, https://arxiv.org/abs/2104.12802. *submitted to IEEE Transactions on Microwave Theory and Techniques*

[56]: S. Chellappa, L. Feng, V. de la Rubia, and P. Benner, Adaptive Interpolatory MOR by Learning the Error Estimator in the Parameter Domain, in Model Reduction of Complex Dynamical Systems, Eds: P. Benner, and T. Breiten, and H. Faßbender, and M. Hinze, and T. Stykel, and R. Zimmermann, vol. 171 of International Series of Numerical Mathematics, Birkhäuser, Cham, 2021, pp. 97 – 117, https://doi.org/10.1007/978-3-030-72983-7_5.

Chapter 4 is based on the material published in

[53]: S. Chellappa, L. Feng, and P. Benner, Adaptive Basis Construction and Improved Error Estimation for Parametric Nonlinear Dynamical Systems, *Internat. J. Numer. Methods Engrg.*, 121 (2020), pp. 5320–5349, https://doi.org/10.1002/nme.6462.

Chapter 5 is based on the following two publications:

[55]: S. Chellappa, L. Feng, and P. Benner, An Adaptive Sampling Approach for the Reduced Basis Method, in Realization and Model Reduction of Dynamical Systems - A Festschrift in Honor of the 70th Birthday of Thanos Antoulas, Eds: C. Beattie, and P. Benner, and M. Embree, and S. Gugercin, and S. Lefteriu, Springer, Cham, 2022, pp. 137 – 155, https://doi.org/10.1007/978-3-030-95157-3_8.

[54]: S. Chellappa, L. Feng, and P. Benner, A Training Set Subsampling Strategy for the Reduced Basis Method, *J. Sci. Comput.*, 89, 63 (2021), pp. 1 – 34, https://doi.org/10.1007/s10915-021-01665-y.

Chapter 6 is an extension of the ideas introduced in Chapters 4 and 5 to coupled systems and is so far unpublished.

ABSTRACT

In this thesis, we study *a posteriori* error estimation and adaptivity with the goal of *automatic* model order reduction of large-scale systems. We propose efficient *offline adaptive techniques* that are aimed at (a) bringing down the significant offline cost often associated with generating reduced-order models and (b) minimizing the user interference in obtaining efficient reduced-order models. We consider adaptivity in two aspects: *adaptive basis enrichment* and *adaptive training set sampling*. The adaptive techniques we propose are enabled by efficient and sharp *a posteriori* error estimators. The error estimators not only guide the offline generation of reduced-order models, but also provide error certification for their online use. Starting with the class of parametric linear steady, time-harmonic, and dynamical systems, we introduce an *inf-sup-constant-free* error estimator targeted towards systems with small or vanishing *inf-sup* constant. This is especially true for many systems arising in electromagnetics. We incorporate the error estimator within a greedy algorithm to adaptively enrich the projection basis. We iteratively compute a data-driven surrogate model of the error estimator in order to enable the adaptive sampling of the training set. Following this, the adaptive techniques are then extended to the class of (parametric) nonlinear dynamical systems. We introduce an improved *a posteriori* error estimator for the output variable and employ it within a greedy algorithm to obtain a compact reduced-order model. Our improved error estimator is able to additively decompose the error contribution arising from the approximation of the state vector and the nonlinear vector via hyperreduction. Making use of this, we adaptively and simultaneously add/remove basis vectors to/from the two projection matrices. To address the curse of dimensionality often associated with parametric problems, we introduce two separate strategies to adaptively sample the training set. The first is a *bottom-up* sampling where a data-driven surrogate of the improved error estimator is utilised to iteratively add/remove parameter samples to/from a coarse training set. The second is a *top-down* approach in which we start from a fine training set and iteratively identify the most important samples to be retained. As a final contribution, the combined adaptive basis enrichment and adaptive training set sampling approach is extended to coupled systems. Throughout the thesis, we validate our theoretical results and algorithms by performing numerical experiments on several large-scale examples chosen to represent a wide range of applications.

ZUSAMMENFASSUNG

In dieser Arbeit untersuchen wir a posteriori-Fehlerabschätzung und Adaptivität mit dem Ziel der automatisierten Modellordnungsreduktion von Systemen mit großer Zustandsraumdimension. Wir stellen effiziente offline-adaptive Verfahren vor, die darauf abzielen, (a) die beträchtlichen offline-Rechenkosten zu senken, die oft mit der Erzeugung von Modellen reduzierter Ordnung verbunden sind, und (b) den Einfluss des Benutzers bei der Generierung effizienter Modelle reduzierter Ordnung zu minimieren. Wir betrachten zwei Aspekte des Konzepts der Adaptivität genauer: die adaptive Basisanreicherung und die adaptive Auswahl der Trainingsmengen. Die von uns vorgeschlagenen adaptiven Verfahren werden durch effiziente und genaue a posteriori-Fehlerschätzer ermöglicht. Die Fehlerschätzer werden nicht nur für die offline-Generierung von Modellen reduzierter Ordnung benötigt, sondern liefern auch eine Fehlerzertifizierung für deren Nutzung in der online-Phase. Ausgehend von der Klasse parametrischer linearer, stetiger, zeitharmonischer und dynamischer Systeme führen wir einen inf-sup-Konstantenfreien Fehlerschätzer ein, der auf Systeme mit einer sehr kleinen oder verschwindenden inf-sup-Konstante ausgerichtet ist. Dies gilt insbesondere für viele Systeme aus der Elektromagnetik. Wir integrieren den Fehlerschätzer in einen Greedy-Algorithmus zur adaptiven Anreicherung der Projektionsbasis. Wir berechnen iterativ ein datengetriebenes Ersatzmodell des Fehlerschätzers, um die adaptive Auswahl der Trainingsmenge zu ermöglichen. Im Anschluss daran werden die adaptiven Verfahren auf die Klasse der (parametrischen) nichtlinearen, dynamischen Systeme erweitert. Wir führen einen verbesserten a posteriori-Fehlerschätzer für die Ausgangsvariable ein und verwenden ihn in einem Greedy-Algorithmus, um ein (kompaktes) Modell reduzierter Ordnung zu erhalten. Unser verbesserter Fehlerschätzer ist in der Lage die Beiträge, die sich aus der Approximation des Zustandsvektors und des nichtlinearen Vektors ergeben, mithilfe von Hyperreduktion additiv zu zerlegen. Auf dieser Grundlage fügen wir adaptiv und simultan Basisvektoren zu den beiden Projektionsmatrizen hinzu bzw. entfernen sie. Um den "Fluch der Dimensionalität" zu umgehen, der oft mit parametrischen Problemen verbunden ist, führen wir zwei separate Strategien zur adaptiven Auswahl der Trainingsmenge ein. Bei der ersten handelt es sich um ein Bottom-up-Sampling, bei dem eine datengesteuerte Approximation des verbesserten Fehlerschätzers verwendet wird, um iterativ Parameterproben zu einer groben Trainingsmenge hinzuzufügen oder daraus zu entfernen. Der zweite ist ein Top-Down-Sampling-Ansatz, bei dem wir von einer feinen Trainingsmenge ausgehen und iterativ die wichtigsten Proben identifizieren, die beibehalten werden sollen. Als letzter Beitrag wird der kombinierte Ansatz der adaptiven Basisanreicherung und der adaptiven Trainingsmengenauswahl auf gekoppelte

Systeme erweitert. Während der gesamten Arbeit validieren wir unsere theoretischen Ergebnisse und Algorithmen durch numerische Experimente an mehreren groß angelegten Beispielen, die ein breites Spektrum von Anwendungen repräsentieren.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF SYMBOLS

$\mathbb{R}$, $\mathbb{C}$	fields of real and complex numbers
$\mathbb{C}_-$	open left complex half plane
$\mathbb{R}_+$	strictly positive real line
$\mathbb{R}_{\geq 0}$	non-negative real line: $\mathbb{R}_+ \cup \{0\}$
$\mathbb{R}^n$, $\mathbb{C}^n$	vector space of real/complex n-tuples
$\mathbb{R}^{m\times n}$, $\mathbb{C}^{m\times n}$	real/complex $m \times n$ matrices
$\lvert S\rvert$, $\lvert x\rvert$	cardinality of the set S, absolute value of some real or complex quantity x
$L^2(\Omega)$	the equivalence class of square integrable functions $f(x)$ defined over the domain Ω, such that $\int_\Omega \lvert f(x)\rvert^2 dx < \infty$
$\langle \cdot, \cdot \rangle$	inner product
$\jmath$	imaginary unit ($\jmath^2 = -1$)
$\mathrm{real}(A)$, $\mathrm{imag}(A)$	real and imaginary part of a complex quantity $A = \mathrm{real}(A) + \jmath \cdot \mathrm{imag}(A) \in \mathbb{C}^{n\times m}$
a_{ij}	the (i,j)-th entry of a matrix A
$A(i:j,:)$, $A(:,k:\ell)$	rows $i,\ldots,j$ of A, columns $k,\ldots,\ell$ of A
$A(i:j,k:\ell)$	rows $i,\ldots,j$ and columns $k,\ldots,\ell$ of A
A^T	the transpose of A
A^{-1}	the inverse of nonsingular A
$A^{-\mathsf{T}}$	the inverse of A^T
$A^\dagger$	the pseudoinverse of nonsingular A
e_i	i-th column of the $n \times n$ identity matrix I
$\sigma_{\max}(A)$	the largest singular value of A
$\sigma_{\min}(A)$	the smallest singular value of A
$\mathrm{range}(A)$	linear subspace spanned by the columns of the matrix A
$\mathrm{orth}(A)$	orthonormal basis which spans the same subspace spanned by the columns of the matrix A

$\|u\|_p$	$:= (\sum_{i=1}^{n} \|u_i\|^p)^{1/p}$ for $u \in \mathbb{C}^n$ and $1 \leq p < \infty$
$\|u\|_\infty$	$:= \max_i \|u_i\|$, the maximum norm of u
$\|A\|_p$	$:= \sup\{\|Au\|_p : \|u\|_p = 1\}$, subordinate matrix p-norm, $1 \leq p \leq \infty$
$\|A\|_{\mathrm{F}}$	$:= \sqrt{\sum_{i,j} \|a_{ij}\|^2} = \sqrt{\mathrm{tr}(A^* A)}$, the Frobenius norm of matrix $A \in \mathbb{C}^{m \times n}$
$\|u\|$, $\|A\|$	Euclidean vector or subordinate matrix norm $\|\cdot\|_2$
$A \otimes B$	the Kronecker product of A and B
$\mathbb{E}[x]$	expected value of a random scalar or vector x
$\binom{n}{k}$	combination of n quantities taken k at a time; $\binom{n}{k} = \frac{n!}{(n-k)\ k!}$
$\partial_{x_j} f := \frac{\partial}{\partial x_j} f$	partial derivative with respect to x_j of f
$\partial_{x_j x_k} f = \frac{\partial^2}{\partial x_j \partial x_k} f$	$:= \partial_{x_j} \partial_{x_k} f$, second order partial derivative with respect to x_j and x_k of f
$\partial_{x_j}^2 f := \partial_{x_j x_j} f$	second order partial derivative with respect to x_j of f
$\partial_t f := \frac{\partial}{\partial t} f$	the derivative with respect to time of f
$\nabla f := (\partial_{x_1} f, \ldots, \partial_{x_n} f)^{\mathrm{T}}$	the gradient of f
$\Delta f := \sum_{i=1}^{n} \partial_{x_i}^2 f$	the *Laplacian operator* applied to f
$\nabla^2 f := \begin{bmatrix} \partial_{x_1}^2 f & \cdots & \partial_{x_1 x_n} f \\ \vdots & \ddots & \vdots \\ \partial_{x_n x_1} f & \cdots & \partial_{x_n}^2 f \end{bmatrix}$	the *Hessian matrix* of f

LIST OF ACRONYMS

AdSS	Adaptive Snapshot Selection
BT	Balanced Truncation
DAE	differential-algebraic equation
DEIM	Discrete Empirical Interpolation Method
DMD	Dynamic Mode Decomposition
EIM	Empirical Interpolation Method
FDM	finite difference method
FEM	finite element method
FOM	full-order model
FVM	finite volume method
IMQ	inverse multiquadrics
IRKA	Iterative Rational Krylov Algorithm
LTI	linear time-invariant
MIMO	multiple-input multiple-output
MM	Moment Matching
MMM	Multi-moment Matching
MOR	Model Order Reduction
ODE	ordinary differential equation
P-LTI	parameterized LTI
PDE	partial differential equation
PMOR	parametric MOR
POD	Proper Orthogonal Decomposition
QoI	quantity of interest
RBF	Radial Basis Function
RBM	Reduced Basis Method

ROM	reduced-order model
SISO	single-input single-output
SVD	singular value decomposition
TPS	thin-plate splines

CHAPTER 1

INTRODUCTION

Contents

1.1. Motivation

Mathematical modelling and numerical simulations have become indispensable tools in many areas of science and technology. An increasing number of scientific disciplines are realising the benefits of using computer-aided simulations as the preferred design and analysis tool, in place of expensive real-world laboratory experiments. The rise of numerical simulation to prominence has thrown up several challenges. On the one hand, there is a demand for high-performance computing hardware to enable faster, more efficient simulations of complex mathematical models. An example of this scenario is in the field of machine learning and artificial intelligence where newer hardware architectures being introduced, for example, the graphics processing units (GPUs) are enabling more efficient training of large models. On the other hand, there has been a growing emphasis on developing faster, more computationally efficient software or algorithms that offer good performance at a modest computational cost. Parallel to the dominance of numerical experiments, there has also been a trend of increasingly complex mathematical models in many areas of science. To support a detailed description of physical phenomena, models are now highly nonlinear, coupled and often involve multiple physics.

In most cases, analytical solutions of the models are not available, or when available, they are simply intractable. Numerical solutions of these complex models are realized by discretizing the governing set of ordinary differential equations (ODEs) or partial differential equations (PDEs) with standard techniques such as the finite element method (FEM), finite difference method (FDM) or the finite volume method (FVM).

To achieve good approximation, usually a fine discretization is considered, resulting in a large number of degrees of freedom and hence a large number of equations. In recent years, the *order* of a discretized model, defined as the number of equations (differential or algebraic) that need to be solved during a simulation, has risen dramatically. Two areas where even the existing high-end computer infrastructure finds it difficult to cope with the rising complexity of mathematical models are *real-time* and *multi-query* applications. Real-time computing is of importance in fields such as optimal control that involve feedback control systems. The microcontrollers that actuate a system need to provide controlling signals in real-time, subject to varying operating conditions of a device. To enable this, a mathematical model of the system that the controller seeks to control, needs to be solved in real time. Given the limited computing power and storage available in microcontrollers, this task is often impossible when the models are *large-scale*. Multi-query applications involve repeated simulation of a mathematical model, usually during the design or optimization stage of modelling, or for uncertainty quantification. An engineer may wish to study the system and its response subject to variations in one or several physical/geometrical parameters. However, this task is often not easily achievable, unless specialized computing hardware is used. Owing to this limitation of hardware-based solutions in the above situations, a variety of algorithmic solutions have been considered.

One such solution to the aforementioned demand for efficient computational methods to speed up numerical simulations is *Model Order Reduction (MOR)*, an area that has been under active development over the past three decades [11, 12, 26, 32, 33, 34, 36, 167, 168, 183]. MOR is a dimension reduction technique, with the goal to generate a *surrogate model* for the large-scale, complex mathematical model (hereafter referred to as full-order model (FOM)). This surrogate model, commonly called the reduced-order model (ROM), has the same structure as the FOM but brings significantly reduced computational cost. Therefore, it enables rapid simulation both in the real-time and multi-query scenarios mentioned above. Moreover, the system response resulting from solving the ROM is nearly identical to the one produced by simulating its FOM. Actually, the key concepts underlying MOR were introduced across different fields, under different names. It is only recently, that there has been a unification of the differing ideas under the framework of MOR.

The most widely used MOR approaches include: Balanced Truncation (BT) [141], Moment Matching (MM) or Krylov-subspace methods [101, 202], Proper Orthogonal Decomposition (POD) [77, 117, 189], and Reduced Basis Method (RBM) [112, 167]. These methods can be characterized as *projection-based* MOR methods since the underlying philosophy is to identify a *low-dimensional subspace* of the solution space associated with the discretized FOM and to *project* the FOM onto this subspace. The first two techniques, BT and Krylov-subspace methods, are mainly used in *frequency-domain* applications where models are obtained by applying suitable integral transforms such as the Laplace or Fourier transform to the governing ODE. They are often the methods of choice in systems and control theory owing to the advantage of input-independence. The latter two techniques POD, RBM are usually (though not always) preferred for *time-domain* applications. In addition to projection-based MOR methods, there also exist *data-driven methods*, such as the Loewner method [14, 138] and the operator-inference method [161]. These methods do not require access to the model of the

system, instead yield an approximation purely from data, by means of interpolation or least-squares data fitting. More recently, techniques originating in the fields of artificial neural network and deep learning have also been successfully applied to generate surrogate models [89, 95, 114, 128].

While many MOR methods were initially suggested for non-parametric systems, a major research topic in the past decade has been the extension of such methods to parametric systems, which is called parametric MOR (PMOR) [35]. Among frequency-domain methods, BT was extended recently to linear parametric systems in [194], while many other Krylov-subspace methods such as the Iterative Rational Krylov Algorithm (IRKA) [102], MM have also been extended [82, 118, 153]. Within time-domain MOR methods, the POD-greedy method has emerged as the most successful PMOR technique. For a more complete list and a general overview of PMOR techniques, we refer to the survey [35] and also to the recent books [12, 32, 33].

To achieve computational efficiency, all PMOR methods follow the *offline-online* paradigm. The *offline stage* involves solving several FOMs in order to obtain solution snapshots, from which a good linear subspace is identified for projection. This stage is usually computationally demanding. Once the ROM is built up by projection, it can be solved for any new parameter instance, cheaply, at the *online stage*. It is ensured that the computational complexity at the online stage scales only with the reduced order of the ROM.

Despite many successes of applying MOR methods to a variety of real-world problems, there remain persistent issues that limit the widespread adoption of ROMs. These include: reliable and cheap error estimation for parametric nonlinear systems [35], accurate error estimation for systems having resonance behaviour [91], offline-efficient algorithms to construct the ROMs, efficient MOR for convection-dominated problems [157], MOR methods preserving passivity, stability, etc. Another major issue particular to PMOR methods is the curse of dimensionality associated with the sampling of parameters [105, 113].

This thesis aims at addressing some of these issues faced by the current MOR techniques. In particular, we will consider the problem of *a posteriori* error estimation, for both frequency- and time-domain MOR methods and propose two novel error estimators. Using these error estimators, we aim to propose efficient techniques of adaptivity: in the context of basis enrichment and training set sampling.

1.2. Error Estimation for MOR: state-of-the-art

An important consideration that arises during MOR is the quantification of the error resulting from the ROM approximation. For time-domain MOR methods, the interest is either estimating the error in approximating the state variable or the output quantity of interest (QoI). For frequency-domain MOR methods, the error in approximating the transfer function in different norms is studied [11] and some *a priori* error bounds are introduced. Not all MOR methods come with *a priori* error bounds, therefore, *a posteriori* error bounds are derived. Such error bounds (or estimates) are crucial for providing guarantees to the end-user about the reliability of MOR methods. Additionally, they also play a role in adaptively improving the quality of the ROMs and this

shall be one of the main foci of this thesis.

Frequency-domain MOR There exist a number of works covering *a posteriori* error estimation for non-parametric systems. For frequency-domain MOR methods, an hierarchical error estimator, that uses two different ROM approximations to compute the error was proposed in [101] for Krylov-subspace methods applied to non-parametric systems; residual-based error estimators were considered in [81, 116]. A bound for the transfer function approximation error in the $\mathcal{H}_2$-norm was introduced in [208]. Error bounds for parametric systems, inspired from those used in the RBM were proposed in [81]. More recently, error estimators for linear parametric systems were introduced in [83, 85]. The efficient application of error estimation to systems involving many parameters or systems whose parameters cover a wide range of values is still an open issue. For such systems, computing the error estimator for many parameters is hampered by the *curse of dimensionality*. Error estimation for electromagnetic systems is another challenging topic. Many electromagnetic systems such as antennas often exhibit large oscillatory or resonant behaviour at certain frequencies. As a result, some existing error estimators that need the so-called *inf-sup* constant turn out to be inaccurate at and around resonant frequencies (see Chapter 3 for a formal introduction).

Time-Domain MOR Similar to the frequency-domain, error bounds (or estimators) have also been proposed for time-domain MOR methods. The most prominent among them is the RBM, where an *a posteriori* error bound or estimator is an essential part of the greedy algorithm. Error bounds for both state and output quantities have been proposed for linear and nonlinear elliptic PDEs, for both coercive and non-coercive systems [133, 177, 200, 201]; these were later extended to linear parabolic time-dependent systems [100] and then to non-affine, nonlinear systems [97, 98]. All these error bounds were derived for systems arising from a weak formulation of the PDE in the function space setting. More generally, error bounds for systems arising from other discretization techniques such as the finite volume method, were considered in the Euclidean vector space in [71, 214]. While reliable error bounds exist for nonlinear and non-affine systems, they are often computationally involved and not sharp. For the RBM, the usual *rule-of-thumb* is to consider a finely sampled training set consisting of many parameter samples. This makes the evaluation of the existing error bounds inefficient, consuming a large percentage of the offline training time.

1.3. Adaptivity for MOR: state-of-the-art

Adaptivity in MOR has received considerable attention recently in several works such as [8, 61, 96, 126, 151, 158] and [9, 46, 71, 87, 105, 124, 125, 157, 160]. It is hard to give a single definition of what constitutes adaptivity, since it has been used in many contexts. Roughly, it can be described as a set of techniques that tailor a certain MOR method to the particular problem under consideration. The goals include, but are not limited to: (a) obtaining a compact ROM with an optimally small order, (b) reducing the offline training time for the ROM and improving ROM accuracy, (c) reducing the online evaluation time for the ROM and improving online accuracy. While some of these

objectives may seem opposing, adaptive MOR approaches aim for a balance amongst them. At the offline stage, many works have considered adaptivity in the context of snapshot generation [96, 126, 151], basis generation [8, 61, 158], parameter and training sampling [105, 124, 125], basis enrichment [87]. At the online stage, adaptivity has been proposed to interpolate among ROMs defined for different operating regimes and parameters [9] and also to adaptively enrich the basis [46]. There exist methods that incorporate new data available online to improve ROM quality [160].

Among offline adaptive methods, there is still considerable room for improvement with regards to adaptive basis enrichment and adaptive parameter sampling since existing methods are still rather heuristic or require complex implementation. Moreover, the issue of adaptive basis enrichment for nonlinear systems, balancing the approximation quality of the state and nonlinear terms is not yet fully addressed.

1.4. Objectives

This thesis is devoted to addressing two outstanding problems in MOR: *a posteriori* error estimation and *adaptivity*. We aim to propose sharp and cheap *a posteriori* error estimators with a broad focus on (parametric) linear and nonlinear systems. Within linear parametric systems, we take up both steady and time-dependent dynamical systems. We also pay special attention to linear time-harmonic systems exhibiting resonances, such as those occurring in electromagnetics. The nonlinear systems we consider are time-dependent. For both types of linear and nonlinear systems, we consider adaptive generation of ROMs at the offline stage. We consider adaptivity in the context of basis enrichment and parameter sampling. The adaptive strategies aim at reducing the high offline training costs often associated with generating ROMs. Leveraging *a posteriori* error estimation and adaptivity, our ultimate goal is to perform *automatic* model order reduction.

1.5. Contributions

The main contributions of this thesis include:

1. *A posteriori error estimation for frequency-domain MOR:* We have developed an *inf-sup-constant-free a posteriori* state error estimator for general linear, parametric systems. We use this error estimator in an adaptive greedy algorithm to generate accurate ROMs at a reduced offline computational cost compared to existing approaches. One specific application we target is electromagnetic systems that exhibit resonance behaviour.

2. *Adaptive parameter sampling for frequency-domain MOR:* Using existing *a posteriori* error estimators for the reduced transfer functions of linear parametric systems, we propose a new adaptive algorithm to generate the ROM. The algorithm incorporates a data-driven surrogate of the error estimator. The surrogate model keeps offline computational costs low and paves the way for efficient parameter sampling.

3. *A posteriori error estimation for time-domain MOR:* We propose an *a posteriori* output error estimator based on a modified output term, that exploits an approximate solution to a discrete-time dual problem to achieve sharp error estimation. When the dual problem is parametric, the approximate solution to the dual problem is computed simultaneously with the ROM construction for the original model. For a non-parametric dual problem we use Krylov-subspace methods to solve the dual system.

4. *Adaptive basis enrichment for time-domain MOR:* Using the time-domain output error estimator as a 'feedback', we propose an adaptive greedy algorithm to iteratively and simultaneously enrich the projection bases for approximating the state and nonlinear variables.

5. *Bottom-up adaptive parameter sampling for time-domain MOR:* We build a data-driven surrogate model for the proposed output error estimator and incorporate its construction within a greedy algorithm. We use the surrogate error model to adaptively update the parameter training set, *bottom-up*, starting from a training set with small cardinality. This approach is tailored for problems that have many parameters or whose parameters vary in wide ranges. The proposed strategy is combined with the adaptive basis enrichment scheme to achieve a higher level of adaptivity.

6. *Top-down adaptive parameter sampling for time-domain MOR:* In contrast to the bottom-up adaptive parameter sampling, we propose a *top-down* parameter sampling approach that starts with a large, finely sampled training set and identifies a small set of important samples through a sparse sampling strategy. We propose several variants of a two-stage adaptive algorithm, which utilize different sampling criteria. The approach is applicable to systems with a large number of parameters and those with vector-valued quantities of interest. This top-down parameter sampling approach can also be used in conjunction with the adaptive basis enrichment scheme.

7. *Adaptive MOR for Coupled Systems:* We extend the proposed adaptive basis enrichment scheme and the adaptive parameter sampling strategy to coupled systems.

1.6. Outline

The organization of the thesis is as follows:

To lay a common ground for the kind of systems and MOR techniques we consider, Chapter 2 reviews several basic concepts which will be used throughout this thesis. We start with the mathematical settings of the systems of interest. These include parametric steady, time-harmonic, and dynamical systems. Following this, we briefly review projection-based MOR methods. Then, some frequency- and time-domain MOR methods are reviewed, based on which the algorithms in this thesis are developed. Following this, we discuss hyperreduction in the context of MOR. The chapter ends with a discussion on Radial Basis Functions (RBFs) and RBF-based data interpolation.

Chapter 3 concerns one of the main contributions of this thesis. In this chapter, the focus is on MOR of *linear* steady/time-harmonic and dynamical systems. We first propose a new *inf-sup-constant-free* approach for *a posteriori* state error estimation, then incorporate the estimator within an adaptive greedy ROM construction procedure. We briefly analyse the computational costs of this greedy procedure. Then, two real-life models of electromagnetic circuits are used to illustrate our approach. The second part of the chapter focusses on *a posteriori* output error estimation for frequency-domain methods. We review existing approaches and propose an efficient RBF-based surrogate model that learns an *inf-sup-constant-free* output error estimator. We then integrate the surrogate error estimator within an adaptive greedy procedure to construct ROMs efficiently. The significant computational gains and accuracy of the method are demonstrated on several benchmark examples.

In Chapter 4, we introduce another major contribution of this thesis, viz., *a posteriori* error estimator for general parametric nonlinear dynamical systems. We review the existing approaches on error estimation and point out some of the drawbacks. Following this, we propose the *a posteriori* error estimator based on a modified output term and discuss its computational aspects and advantages. Then, we briefly highlight a fast approach for computing the *inf-sup* constant involved in the error estimator using RBF interpolation. In the second part of the chapter, we turn our attention to adaptivity. There, we introduce a comprehensive adaptive enrichment scheme for the POD and RBM methods, that simultaneously enriches the projection matrices for both the state and the nonlinear term of the given problem. We discuss its computational aspects and illustrate its performance through numerical examples involving both non-parametric and parametric problems.

To address the non-trivial choice of training set for the RBM, in Chapter 5 we propose two adaptive approaches. The first approach is an extension of the RBF-based adaptive sampling method proposed in Chapter 3 for frequency-domain systems. We show the efficiency of this approach through well-known benchmark examples. Our second approach is based on a *subsampling approach*, where sparse sampling strategies are used to identify the most important samples in a fine training set with large cardinality. We again illustrate the merits of this approach through examples involving parametric linear and nonlinear dynamical systems.

As a final contribution, in Chapter 6 we apply the adaptive basis enrichment and adaptive training set sampling schemes to parametric coupled systems. While the extension is straightforward, we detail some computational aspects that improve efficiency. The method is illustrated using the model of batch chromatography in process engineering.

We summarize this thesis in Chapter 7. Based on our experience, we highlight several promising directions and ideas for future research.

CHAPTER 2

MATHEMATICAL PRELIMINARIES

Contents

In this chapter, we review the mathematical fundamentals and existing MOR methods considered in this thesis. We begin by introducing the setting of the differential equations that describe the large-scale models we consider. Our focus is on systems both in the frequency- and the time-domain. In the frequency-domain, we consider linear parametric steady, time-harmonic, and time-dependent/dynamical systems, while in the time-domain we consider parametric nonlinear dynamical systems. In general, our approaches also apply to linear dynamical systems as special cases. Next, we introduce concepts of projection-based MOR and the formulations of ROMs for the different types of systems considered in the thesis. This is followed by two subsections devoted to highlighting the particular MOR techniques used in the thesis. In the frequency-domain, we offer an overview of the available methods but focus more on the Multi-moment Matching (MMM) method and its computational aspects. Among time-domain MOR methods, we discuss the POD method and the RBM and sketch the state-of-the-art

algorithms. We then discuss the notion of hyperreduction concentrating on its general formulation. We detail the algorithms of two hyperreduction methods: Empirical Interpolation Method (EIM) and Discrete Empirical Interpolation Method (DEIM). We conclude this chapter with an overview of radial basis interpolation techniques.

Before discussing the concrete mathematical details, we highlight the notations used and also the styles used for the algorithms and figures.

General Notations We denote scalar-valued quantities with lower- and upper-case letters (e.g., t, T) or lower-case Greek alphabets (e.g., α, ω). We employ bold, lower-case letters to represent vectors (e.g., $\mathbf{x} = [\mathrm{x}_1, \ldots, \mathrm{x}_n]$) where $\mathrm{x}_1, \ldots, \mathrm{x}_n$ represent the components of the vector. Bold, upper-case letters denote matrices (e.g., $\mathbf{A}$). Moreover, $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_N]$ with $\mathbf{a}_1, \ldots, \mathbf{a}_N$ being the vectors forming the columns of the matrix. Upper case letters written in the script style (e.g., $\mathscr{A}$) stand for linear or nonlinear operators. Sets and spaces are described with calligraphic letters (e.g., $\mathcal{A}$) with the exception of the L^2 space, for which we use the upper-case alphabet. The calligraphic letter $\mathcal{L}$ is reserved for the Laplace transform. We reserve upper-case Fraktur letters for the Laplace-transformed variables (e.g., $\mathfrak{F}(s)$) For real- and complex-valued Euclidean spaces of appropriate dimensions, we use blackboard letters $\mathbb{R}^n$ and $\mathbb{C}^n$, respectively.

Algorithmic Notations Throughout the following chapters, we present the pseudocodes/algorithmic sketches of various methods that are either being used or proposed in this thesis. To enable easy representation, we will often *call* an algorithm (say **Algorithm A**) from within another (**Algorithm B**), with appropriate inputs. The resulting output from the call to **Algorithm A** will be used subsequently in **Algorithm B**. Where needed, we shall clarify the particular method/algorithm being called with appropriate comments.

In all pseudocodes, we denote the variables associated with the loop, such as error, iteration indices, etc. with italicized lower case letters or words (e.g., k, j, *err_max*, *indx_start*, *indx_end*, *iter* etc.)

Figures In order to avoid the legends obstructing details of the plots, we have uniformly placed legends outside their corresponding figures, for all the figures in this thesis.

2.1. Parametrized PDEs

In this thesis, we focus on large-scale systems arising from the discretization of parametrized PDEs. The general form of the PDEs we consider is

$$\frac{\partial}{\partial t} g(\mathbf{z}, t, \boldsymbol{\mu}) = \mathscr{R}[g(\mathbf{z}, t, \boldsymbol{\mu}), \mathbf{u}(t), \boldsymbol{\mu}]. \tag{2.1}$$

The quantity $g : \Omega \times \mathbb{R}_{\geq 0} \times \mathcal{P} \to \mathbb{R}$ is the field variable - the physical quantity modelled by the PDE[1]. We consider the spatial domain $\Omega \subset \mathbb{R}^d$ with $d \in \{1, 2, 3\}$ and $\mathbf{z} \in \Omega \subset \mathbb{R}^d$ denotes the spatial variable. Let $\mathcal{V}$ be a suitable Hilbert space of functions on the domain Ω, such as $L^2(\Omega)$ and we have $g \in \mathcal{V}$. We assume that the boundary of the spatial domain $\partial\Omega = \Gamma_{\text{dir}} \bigcup \Gamma_{\text{neu}}$ is endowed with appropriate boundary values decomposed into Dirichlet and Neumann conditions, with Γ_{dir} denoting the parts of the boundary with Dirichlet conditions and Γ_{neu} is the part with Neumann conditions. The variable $t \in [0, T] \subset \mathbb{R}_{\geq 0}$ denotes time and $\boldsymbol{\mu} = [\mu_1, \ldots, \mu_{N_p}]^\mathsf{T} \in \mathcal{P} \subset \mathbb{R}^{N_p}$ is the vector of parameters characterizing the PDE and can be physical, material or geometrical parameters. It can also represent additional source or boundary terms. $\mathscr{R}[\cdot, \cdot, \boldsymbol{\mu}]$ is a parametrized linear/nonlinear differential operator with three arguments, viz., the field variable, a time-dependent input $\mathbf{u} : \mathbb{R}_{\geq 0} \to \mathbb{R}^{N_I}$ and the parameter $\boldsymbol{\mu}$. We allow the operator to be nonlinear with respect to (w.r.t) the state variable. Additionally, it can be non-affine w.r.t the parameter $\boldsymbol{\mu}$. In many practical situations, some quantity (quantities) obtained as a function of the field variable is (are) of interest. For instance, in the context of control systems, this could be the value of the field variable over some region of space where it needs to be controlled; in aeronautical systems, the lift and drag coefficients are typical output quantities. We consider output quantities $\mathbf{y} : \mathbb{R}_{\geq 0} \times \mathcal{P} \to \mathbb{R}^{N_O}$, where the output variable is a linear functional of the field variable, i.e., $\mathbf{y}(t, \boldsymbol{\mu}) = \mathbf{o}(g(\mathbf{z}, t, \boldsymbol{\mu}), \boldsymbol{\mu}) \in \mathbb{R}^{N_O}$.

In applications such as electromagnetics, linear, time-dependent PDEs (for example, the Maxwell's equations) are first represented in a time-harmonic form using the Laplace or Fourier transform of the field variable. We formally introduce the Laplace transform in Definition 2.2. The general system of time-harmonic PDEs is shown below

$$\mathscr{H}[\mathfrak{G}(\mathbf{z}, s, \boldsymbol{\mu}), \mathfrak{U}(s), \boldsymbol{\mu}] = 0. \tag{2.2}$$

Here, $\mathfrak{G} : \Omega \times \mathbb{C} \times \mathcal{P} \to \mathcal{V}$ is the field variable after an integral transform. $\mathcal{V}$ now denotes a complex Hilbert space on the domain Ω. The variable $s \in \mathbb{C}$ is the Laplace variable and the input in the complex domain is given by $\mathfrak{U} : \mathbb{C} \to \mathbb{C}^{N_I}$. For such systems the output variable is $\mathbf{y}(s, \boldsymbol{\mu}) = \mathbf{o}(\mathfrak{G}(\mathbf{z}, s, \boldsymbol{\mu}), \boldsymbol{\mu}) \in \mathbb{C}^{N_O}$ with $\mathbf{y} : \mathcal{P} \times \mathbb{C} \to \mathbb{C}^{N_O}$.

2.2. Discretized Systems

In order to perform numerical simulations, the PDE needs to be discretized in space and/or time domain. In the following section, we first consider discretization in space, which results in differential-algebraic equations (DAEs), ordinary differential equations (ODEs) for time-dependent problems, or algebraic equations for steady problems. Following that, we consider further discretization of DAEs or ODEs in the time domain.

2.2.1. Discretization in Space

In applications such as systems and control theory, (non)linear, time-dependent parametrized PDEs are first spatially discretized using numerical discretization techniques such as

[1]For PDEs with more than one field variable we have $\mathbf{g} : \Omega \times \mathbb{R}_{\geq 0} \times \mathcal{P} \to \mathbb{R}^m$ with m denoting the number of field variables; see Chapter 6 for a detailed discussion.

the FEM, FVM, etc. This results in a semi-discrete system of *nonlinear* DAEs/ODEs:

$$\begin{aligned} \mathbf{E}(\boldsymbol{\mu})\frac{d}{dt}\mathbf{x}(t,\boldsymbol{\mu}) &= \mathbf{A}(\boldsymbol{\mu})\mathbf{x}(t,\boldsymbol{\mu}) + \mathbf{f}(\mathbf{x},\boldsymbol{\mu}) + \mathbf{B}(\boldsymbol{\mu})\mathbf{u}(t), \qquad \mathbf{x}(0,\boldsymbol{\mu}) = \mathbf{x}_0(\boldsymbol{\mu}), \\ \mathbf{y}(t,\boldsymbol{\mu}) &= \mathbf{C}(\boldsymbol{\mu})\mathbf{x}(t,\boldsymbol{\mu}). \end{aligned} \tag{2.3}$$

Here, $\mathbf{x}(t,\boldsymbol{\mu}) \in \mathbb{R}^N$, with N being the number of degrees of freedom. $\mathbf{x}_0(\boldsymbol{\mu})$ is the parameter-dependent initial condition for the ODE. $\mathbf{E}(\boldsymbol{\mu}) \in \mathbb{R}^{N\times N}$ is a matrix arising from the spatial discretization employed (for example, $\mathbf{E}(\boldsymbol{\mu})$ is the mass matrix when using the FEM). Further, $\mathbf{A}(\boldsymbol{\mu}) \in \mathbb{R}^{N\times N}$ is the matrix arising from discretization of the linear part of the operator $\mathscr{R}[\cdot,\cdot,\boldsymbol{\mu}]$ in Eq. (2.1). The nonlinear term $\mathbf{f}(\mathbf{x},\boldsymbol{\mu})$ with $\mathbf{f} : \mathbb{R}^N \times \mathcal{P} \to \mathbb{R}^N$ is the discretization of the nonlinear part of the operator $\mathscr{R}[\cdot,\cdot,\boldsymbol{\mu}]$ in Eq. (2.1). $\mathbf{u}(t) \in \mathbb{R}^{N_I}$ is the time-dependent input vector, while $\mathbf{B}(\boldsymbol{\mu}) \in \mathbb{R}^{N\times N_I}$ is the input matrix. $\mathbf{y}(t,\boldsymbol{\mu}) \in \mathbb{R}^{N_O\times N_I}$ is the discretized output (matrix) and $\mathbf{C}(\boldsymbol{\mu}) \in \mathbb{R}^{N_O\times N}$ is the discrete representation of the output functional $\mathbf{o}(\cdot,\boldsymbol{\mu})$.

When Eq. (2.1) is a linear problem, the $\mathbf{f}$ in Eq. (2.3) disappears, resulting in a *linear dynamical system*:

$$\begin{aligned} \mathbf{E}(\boldsymbol{\mu})\frac{d}{dt}\mathbf{x}(t,\boldsymbol{\mu}) &= \mathbf{A}(\boldsymbol{\mu})\mathbf{x}(t,\boldsymbol{\mu}) + \mathbf{B}(\boldsymbol{\mu})\mathbf{u}(t), \qquad \mathbf{x}(0,\boldsymbol{\mu}) = \mathbf{x}_0(\boldsymbol{\mu}), \\ \mathbf{y}(t,\boldsymbol{\mu}) &= \mathbf{C}(\boldsymbol{\mu})\mathbf{x}(t,\boldsymbol{\mu}). \end{aligned} \tag{2.4}$$

We assume the system matrices $\mathbf{E}(\boldsymbol{\mu}), \mathbf{A}(\boldsymbol{\mu}), \mathbf{B}(\boldsymbol{\mu})$ and $\mathbf{C}(\boldsymbol{\mu})$ possess the following *parameter affine representation*:

$$\mathbf{E}(\boldsymbol{\mu}) = \mathbf{E}_0 + \sum_{i=1}^{Q_\mathbf{E}} \theta^i_\mathbf{E}(\boldsymbol{\mu})\mathbf{E}_i, \tag{2.5a}$$

$$\mathbf{A}(\boldsymbol{\mu}) = \mathbf{A}_0 + \sum_{i=1}^{Q_\mathbf{A}} \theta^i_\mathbf{A}(\boldsymbol{\mu})\mathbf{A}_i, \tag{2.5b}$$

$$\mathbf{B}(\boldsymbol{\mu}) = \mathbf{B}_0 + \sum_{i=1}^{Q_\mathbf{B}} \theta^i_\mathbf{B}(\boldsymbol{\mu})\mathbf{B}_i, \tag{2.5c}$$

$$\mathbf{C}(\boldsymbol{\mu}) = \mathbf{C}_0 + \sum_{i=1}^{Q_\mathbf{C}} \theta^i_\mathbf{C}(\boldsymbol{\mu})\mathbf{C}_i. \tag{2.5d}$$

where $\mathbf{E}_i \in \mathbb{R}^{N\times N}, i = 0,\ldots,Q_\mathbf{E}$; $\mathbf{A}_i \in \mathbb{R}^{N\times N}, i = 0,\ldots,Q_\mathbf{A}$; $\mathbf{B}_i \in \mathbb{R}^{N\times N}, i = 0,\ldots,Q_\mathbf{B}$ and $\mathbf{C}_i \in \mathbb{R}^{N\times N}, i = 0,\ldots,Q_\mathbf{C}$. The quantities $\theta^i_\mathbf{E}$, $i = 1,\ldots,Q_\mathbf{E}$, $\theta^i_\mathbf{A}$, $i = 1,\ldots,Q_\mathbf{A}$, $\theta^i_\mathbf{B}$, $i = 1,\ldots,Q_\mathbf{B}$ and $\theta^i_\mathbf{C}$, $i = 1,\ldots,Q_\mathbf{C}$ are scalar-valued functions of the parameters.

Remark 2.1:
The assumption of affine parameter dependence of the system matrices is not true in general. However, there exists techniques such as *hyperreduction* which can be utilized to obtain such a representation. A detailed discussion is postponed to Section 2.6. ♢

In order to obtain a faithful representation of the PDE, a very fine discretization of the spatial domain is generally preferred. Therefore, the number of degrees of freedom, N, is usually large, and can be up to $O(10^7)$ or greater. Such a large order of the system poses a major computational challenge, especially when Eq. (2.3) (or Eq. (2.4)) needs to solved repeatedly, or when the system needs to be simulated over a long time interval.

Equation (2.4) is called linear time-invariant (LTI) system in systems and control theory. In the parametric case, we denote it as parameterized LTI (P-LTI) system. When $N_I = N_O = 1$, such systems are called *single-input single-output (SISO)* systems. They are termed *multiple-input multiple-output (MIMO)* systems when $N_I, N_O > 1$. Usually, for many systems, the number of inputs and outputs is much smaller than the state-space dimension, i.e., $N_I, N_O \ll N$.

An important tool in the analysis of LTI and, more generally, P-LTI systems is their *frequency-domain* representation. The frequency domain representation of P-LTI systems is obtained by applying suitable integral transforms, such as Laplace transform or Fourier transform. Here, we use the Laplace transform.

Definition 2.2 (Laplace Transform):
Let $\mathbf{q} : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ be a locally integrable, vector-valued function. Its one-sided Laplace transform is given by $\mathcal{L} : \mathbf{q} \mapsto \mathfrak{Q}$ with

$$\mathfrak{Q}(s) := \mathcal{L}[\mathbf{q}(t)](s) := \int_0^\infty e^{-s\,t}\,\mathbf{q}(t)\,dt, \qquad s \in \mathbb{C}$$

where $s = \alpha + \jmath\omega$, $\jmath$ is the imaginary unit and $\omega := 2\pi f$ is the angular frequency with units radians/s with f being the ordinary frequency with units Hz. ◊

The strength of the frequency domain representation is that it allows transforming linear differential equations to linear algebraic equations, resulting in a simplified solution through (shifted) matrix inversion operations. For a fixed parameter $\boldsymbol{\mu}$, let the one-sided Laplace transforms of $\mathbf{x}(t), \mathbf{u}(t), \mathbf{y}(t)$ be $\mathfrak{X}(s), \mathfrak{U}(s)$ and $\mathfrak{Y}(s)$, respectively. Assuming zero initial condition $\mathbf{x}_0(\boldsymbol{\mu}) = 0$ and applying the Laplace transform to Eq. (2.4) yields

$$s\mathbf{E}(\boldsymbol{\mu})\mathfrak{X}(s,\boldsymbol{\mu}) = \mathbf{A}(\boldsymbol{\mu})\mathfrak{X}(s,\boldsymbol{\mu}) + \mathbf{B}(\boldsymbol{\mu})\mathfrak{U}(s), \tag{2.6a}$$
$$\mathfrak{Y}(s,\boldsymbol{\mu}) = \mathbf{C}(\boldsymbol{\mu})\mathfrak{X}(s,\boldsymbol{\mu}). \tag{2.6b}$$

Eliminating $\mathfrak{X}(s,\boldsymbol{\mu})$, we obtain

$$\mathfrak{Y}(s,\boldsymbol{\mu}) = \big(\mathbf{C}(\boldsymbol{\mu})(s\mathbf{E}(\boldsymbol{\mu}) - \mathbf{A}(\boldsymbol{\mu}))^{-1}\mathbf{B}(\boldsymbol{\mu})\big)\mathfrak{U}(s).$$

Hereafter, we denote by $\breve{\boldsymbol{\mu}} := [s\ \boldsymbol{\mu}]^\mathsf{T} \in \mathbb{C}^{N_p+1}$ the augmented set of parameters.

Definition 2.3 (Parametric Transfer Function):
With the shifted matrix $\mathbf{R}(\breve{\boldsymbol{\mu}}) := (s\mathbf{E}(\boldsymbol{\mu}) - \mathbf{A}(\boldsymbol{\mu}))$ being invertible $\forall\, s \in \mathbb{C}\backslash\mathcal{P}$, $\forall\, \boldsymbol{\mu}$, and $\mathcal{P}$ being the set of eigenvalues of $\mathbf{R}(\breve{\boldsymbol{\mu}})$ $\forall\, \breve{\boldsymbol{\mu}}$, the matrix-valued rational function $\mathbf{H} : \mathbb{C}^{N_p+1} \to \mathbb{C}^{N_O \times N_I}$ defined as

$$\mathbf{H}(\breve{\boldsymbol{\mu}}) := \mathbf{C}(\boldsymbol{\mu})\mathbf{R}(\breve{\boldsymbol{\mu}})^{-1}\mathbf{B}(\boldsymbol{\mu}) \tag{2.7}$$

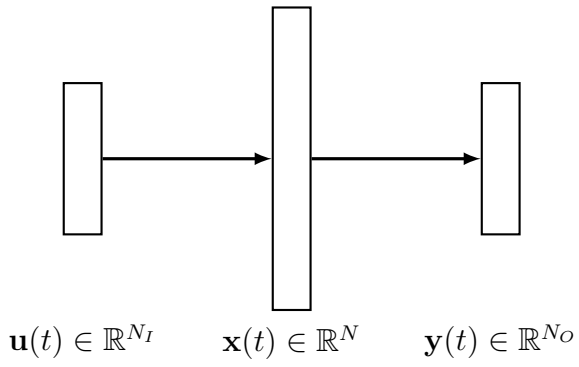


Figure 2.1.: Input-Output System.

is the parametric transfer function corresponding to the P-LTI state-space representation in Eq. (2.4). ◊

We assume that for any parameter, the spectrum of the matrix pencil $(s\mathbf{E}(\boldsymbol{\mu})-\mathbf{A}(\boldsymbol{\mu})) \equiv \mathbf{R}(\breve{\boldsymbol{\mu}})$ falls strictly within the open left-half of the complex plane denoted by $\mathbb{C}^-$. This ensures the asymptotic stability of Eq. (2.7). The transfer function encodes the dynamical behaviour of the system and its major advantage is that it is input-independent.

One way to understand transfer functions is to see them, for a fixed parameter $\breve{\boldsymbol{\mu}}$, as a map from the input to the output, i.e., $\mathbf{H} : \mathbb{R}^{N_I} \to \mathbb{R}^{N_O}$ that passes through a high-dimensional intermediate state in $\mathbb{R}^N$ (see Figure 2.1). The transfer function in Eq. (2.7) may be equivalently written as

$$\mathbf{H}(\breve{\boldsymbol{\mu}}) := \big(\mathbf{C}(\boldsymbol{\mu})\mathbf{R}(\breve{\boldsymbol{\mu}})^{-1}\big)\mathbf{R}(\breve{\boldsymbol{\mu}})\big(\mathbf{R}(\breve{\boldsymbol{\mu}})^{-1}\mathbf{B}(\boldsymbol{\mu})\big).$$

From this, we can define the following two shifted linear systems:

Definition 2.4 (s-Primal System):
The shifted linear system that maps the input to the state vector, given by

$$\mathbf{R}(\breve{\boldsymbol{\mu}})\mathbf{X}_{\mathrm{pr}}(\breve{\boldsymbol{\mu}}) = \mathbf{B}(\boldsymbol{\mu}) \tag{2.8}$$

is called the frequency-domain primal system or s-primal system with $\mathbf{X}_{\mathrm{pr}}(\breve{\boldsymbol{\mu}}) \in \mathbb{C}^{N\times N_I}$. Note that the s-primal system is nothing but the frequency-domain system in Eq. (2.6) with $\mathfrak{U}(s) = 1$. ◊

Definition 2.5 (s-Dual System):
The shifted linear system that maps the state vector to the output, given by

$$\mathbf{R}(\breve{\boldsymbol{\mu}})^{\mathsf{T}}\mathbf{X}_{\mathrm{du}}(\breve{\boldsymbol{\mu}}) = \mathbf{C}(\boldsymbol{\mu})^{\mathsf{T}} \tag{2.9}$$

is called the frequency-domain dual system or s-dual system with $\mathbf{X}_{\mathrm{du}}(\breve{\boldsymbol{\mu}}) \in \mathbb{C}^{N\times N_O}$. ◊

We will utilize the s-primal and s-dual systems in the context of *a posteriori* error estimation of input-output systems in Chapter 3.

In many applications, such as electromagnetics or circuit simulation, the preferred approach for solving linear PDEs (e.g., the Maxwell's equations) is somewhat different.

The time-dependent PDE Eq. (2.1) is first transformed to a time-harmonic PDE by using the Laplace transform. Following this, it is discretized using standard FEM techniques in a complex vector space $\mathbb{C}^N$. This process results in the following *time-harmonic* system:

$$\begin{aligned} \mathbf{A}(\breve{\boldsymbol{\mu}})\mathbf{X}(\breve{\boldsymbol{\mu}}) &= \mathbf{B}(\breve{\boldsymbol{\mu}}), \\ \mathbf{Y}(\breve{\boldsymbol{\mu}}) &= \mathbf{C}(\breve{\boldsymbol{\mu}})\mathbf{X}(\breve{\boldsymbol{\mu}}). \end{aligned} \tag{2.10}$$

We do not offer any detailed discussion here and instead refer to [140] for a complete description. In the above equation, $\mathbf{X}(\breve{\boldsymbol{\mu}}) \in \mathbb{C}^{N\times N_I}$ is the discretized solution matrix for the time-harmonic PDE (such as Eq. (2.2)) and $\mathbf{Y}(\breve{\boldsymbol{\mu}}) \in \mathbb{C}^{N_O}$ is the output quantity of interest. The matrices $\mathbf{A}(\breve{\boldsymbol{\mu}}) \in \mathbb{C}^{N\times N}, \mathbf{B}(\breve{\boldsymbol{\mu}}) \in \mathbb{C}^{N\times N_I}$ result from the spatial discretization of the operator $\mathscr{H}$ in Eq. (2.2). In Eq. (2.10) above, the first equation is similar in form to Eq. (2.8). Depending on the context, we will use the term s-primal system to denote either of these equations.

Remark 2.6:
The use of a complex vector space to discretize the Maxwell's equations is not a limitation. In fact, the discretized system can be reformulated as a real system with real-valued matrices and solution vector with dimension $2N$, by using the isomorphism between $\mathbb{C}$ and $\mathbb{R}^2$. This issue will be further detailed in Chapter 3. ◊

Remark 2.7:
For the case of $\breve{\boldsymbol{\mu}} = \boldsymbol{\mu}$, Eq. (2.10) is the discrete form of an elliptic PDE such as

$$\mathscr{R}[g(\mathbf{z}, \boldsymbol{\mu}), \boldsymbol{\mu}] = 0$$

which is a special case of Eq. (2.1), i.e., we call Eq. (2.10) with $\breve{\boldsymbol{\mu}} = \boldsymbol{\mu}$ the *steady* system. ◊

2.2.2. Discretization in Time

Time-discretization is needed for numerically simulating the semi-discretized systems Eqs. (2.3) and (2.4). To this end, we discretize the time variable t into $N_t := K + 1$ parameter-independent time instances $0 =: t^0 < t^1 < \cdots < t^k < \cdots < t^K := T$ with $t^k = k\Delta t$, $k = 0, 1, \ldots, K$.

Both fully explicit and fully implicit time discretization approaches exist in literature. Explicit techniques such as the Forward Euler scheme, Runge-Kutta 45 scheme, etc. obtain the future value of the state variable $\mathbf{x}(t)$ at a given parameter, from its current or past values. We have,

$$\mathbf{x}(t^{k+1}) = \mathbf{G}(\mathbf{x}(t^k), \mathbf{x}(t^{k-1}), \ldots, \mathbf{x}(t^{k-\mathfrak{s}})).$$

with $\mathfrak{s}$ chosen based on the particular scheme adopted. In the case of the Forward Euler scheme $\mathfrak{s} = 0$. Implicit techniques, such as the Backward Euler scheme, Crank-Nicolson method, etc., determine the state variable at a future time instant by solving an equation (possibly nonlinear) that involves both the past state values and the future state value itself

$$\mathbf{G}(\mathbf{x}(t^{k+1}), \mathbf{x}(t^k), \mathbf{x}(t^{k-1}), \ldots, \mathbf{x}(t^{k-\mathfrak{s}})) = 0.$$

Explicit schemes are straightforward to implement and carry less computational burden. But, they often impose a severe restriction on the value of Δt: extremely small values of Δt are required for accurate results. Implicit schemes do not have such a strict time step restriction, however, they require extra computation in solving a possibly nonlinear system of equations at each time step, using, for instance, the Newton iterations. Moreover, implicit schemes are highly suitable for problems exhibiting *stiffness*.

A more nuanced approach that combines the benefits of the fully explicit and fully implicit methods is the family of Implicit-Explicit (IMEX) methods [16]. Such methods make a distinction in the discretization of the linear and nonlinear terms of the ODE. An implicit scheme discretizes the linear part and an explicit method discretizes the nonlinear part. In this thesis, we use either a fully implicit or an IMEX time discretization method, depending on the system. Details regarding the particular method used will be mentioned in the numerical sections.

Discrete Full-Order Model Using the first-order IMEX method (often referred to as IMEX Euler scheme) to discretize Eq. (2.3) or its special case Eq. (2.4) in time, the resulting fully discrete system reads

$$\frac{1}{\Delta t}\big(\mathbf{E}(\boldsymbol{\mu})\mathbf{x}^{k+1} - \mathbf{E}(\boldsymbol{\mu})\mathbf{x}^{k}\big) = \mathbf{A}(\boldsymbol{\mu})\mathbf{x}^{k+1} + \mathbf{f}(\mathbf{x}^{k}, \boldsymbol{\mu}) + \mathbf{B}(\boldsymbol{\mu})\mathbf{u}(t^{k}), \qquad \mathbf{x}(t^{0}, \boldsymbol{\mu}) = \mathbf{x}_{0}.$$

The quantity $\mathbf{x}^k \in \mathbb{R}^N$ is the fully discretized numerical solution of the PDE in Eq. (2.1) at some time t^k. For the sake of clean notation, we have not explicitly shown the parameter dependency of $\mathbf{x}^k$. Rewriting the above equation by separating all terms at time t^{k+1} to the left, we get the following fully-discrete system

$$\begin{aligned} \overline{\mathbf{E}}(\boldsymbol{\mu})\mathbf{x}^{k+1} &= \overline{\mathbf{A}}(\boldsymbol{\mu})\mathbf{x}^{k} + \overline{\mathbf{f}}(\mathbf{x}^{k}, \boldsymbol{\mu}) + \overline{\mathbf{B}}(\boldsymbol{\mu})\mathbf{u}(t^{k}), \\ \mathbf{y}^{k+1} &= \mathbf{C}(\boldsymbol{\mu})\mathbf{x}^{k+1} \end{aligned} \tag{2.11}$$

where $\overline{\mathbf{E}}(\boldsymbol{\mu}) := \mathbf{E}(\boldsymbol{\mu}) - \Delta t\mathbf{A}(\boldsymbol{\mu})$, $\overline{\mathbf{A}}(\boldsymbol{\mu}) := \mathbf{E}(\boldsymbol{\mu})$, $\overline{\mathbf{B}}(\boldsymbol{\mu}) := \Delta t\mathbf{B}(\boldsymbol{\mu})$ and $\overline{\mathbf{f}} := \Delta t\mathbf{f}$.

2.3. Concepts of Projection-based MOR

As highlighted in Chapter 1, the FOMs obtained through the space and/or time discretization of parametric PDEs (Eqs. (2.3), (2.4), (2.10) and (2.11)) are usually large-scale systems, with large value of N. To enable repeated and real-time evaluations of such systems, surrogate models, or ROMs, with reduced computational complexity are desired. Over the years, a number of ways to obtain surrogate models have been suggested in the literature. In the survey [35], three prominent categories of surrogate models are mentioned. They are:

(i) data-fit ROMs,

(ii) hierarchical ROMs, and

(iii) projection-based ROMs.

Data-fit ROMs are oblivious to the underlying model and they rely purely on data to perform interpolation or regression. Popular methods in this category include: *response-surface models*, kernel methods [80] such as *radial basis function (RBF) interpolation* and *gaussian process regression* [170]. In recent years, a number of methods based on neural networks and machine learning methodologies have been proposed for MOR [114, 128]. These methods also fall under the category of data-fit ROMs. Due to their reliance on data and not on the model, these methods are often called *non-intrusive* methods. Hierarchical surrogate models involve using a hierarchy of different models with varying levels of accuracy to describe the underlying physics of the model. Prominent approaches are mesh coarsening techniques, alternative basis expansion, multigrid methods, etc. [15, 129]. Finally, projection-based MOR methods seek to identify a lower-dimensional subspace of the solution space over which the FOM solutions evolve and project the governing equations onto this subspace. This results in the *order* (number of equations) reduction of the FOM. Moreover, these methods preserve the structure of the underlying model and its associated physics. This can be beneficial when certain special properties of the system, such as, *stability*, *passivity*, etc. need to be preserved. Projection-based methods are *intrusive* techniques. Beyond these three categories, there exist other data-driven methods such as the Loewner approach [14] and the operator-inference approach [31, 161]. In this thesis, we shall restrict our focus exclusively to projection-based approaches.

Next, we introduce the main idea of projection-based MOR through the example of the linear system in Eq. (2.4). Formally, we seek a subspace $\mathcal{S}_V$ with dimension $n \ll N$. We use the term *reduced basis* (RB) to denote a basis of $\mathcal{S}_V$. Let $\{\mathbf{v}_i\}_{i=1}^n$, with $\mathbf{v}_i \in \mathbb{R}^N$, be the basis vectors constituting the RB. We define

$$\mathcal{S}_V := \operatorname{span}\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\} \subset \mathbb{R}^N.$$

We set $\mathcal{S}_V$ to be the *trial space* that approximates the solution space of the FOM. For Eq. (2.4) we seek an approximation to the solution $\mathbf{x}(t, \boldsymbol{\mu})$ in the form

$$\mathbf{x}(t, \boldsymbol{\mu}) \approx \widetilde{\mathbf{x}}(t, \boldsymbol{\mu}) = \sum_{i=1}^{n} \mathbf{v}_i \widehat{x}_i(t, \boldsymbol{\mu}) = \mathbf{V}\widehat{\mathbf{x}}(t, \boldsymbol{\mu}), \tag{2.12}$$

where $\mathbf{V} := [\mathbf{v}_1\, \mathbf{v}_2\, \cdots\, \mathbf{v}_n] \in \mathbb{R}^{N\times n}$. Substituting $\mathbf{x}(t, \boldsymbol{\mu})$ with the approximation $\widetilde{\mathbf{x}}(t, \boldsymbol{\mu})$ in Eq. (2.4) results in the following residual system:

$$\mathbf{r}(t, \boldsymbol{\mu}) = \mathbf{E}(\boldsymbol{\mu})\frac{d}{dt}\big(\mathbf{V}\widehat{\mathbf{x}}(t, \boldsymbol{\mu})\big) - \mathbf{A}(\boldsymbol{\mu})\mathbf{V}\widehat{\mathbf{x}}(t, \boldsymbol{\mu}) - \mathbf{B}(\boldsymbol{\mu})\mathbf{u}(t).$$

The above residual system is *over-determined*, with N equations and n unknown variables. In the next step, we seek a *test space* $\mathcal{S}_W$ and adopt a Petrov-Galerkin projection so that

$$\mathbf{r}(t, \boldsymbol{\mu}) \perp \mathcal{S}_W.$$

Let $\mathbf{W} := [\mathbf{w}_1\, \mathbf{w}_2\, \cdots\, \mathbf{w}_n] \in \mathbb{R}^{N\times n}$, $\mathcal{S}_W = \operatorname{range}(\mathbf{W})$ and $\mathbf{W}^\mathsf{T}\mathbf{V} = \mathbf{I}$, then the ROM for Eq. (2.4) is

$$\begin{aligned} \widehat{\mathbf{E}}(\boldsymbol{\mu})\frac{d}{dt}\widehat{\mathbf{x}}(t, \boldsymbol{\mu}) &= \widehat{\mathbf{A}}(\boldsymbol{\mu})\widehat{\mathbf{x}}(t, \boldsymbol{\mu}) + \widehat{\mathbf{B}}(\boldsymbol{\mu})\mathbf{u}(t), \qquad \widehat{\mathbf{x}}(0, \boldsymbol{\mu}) = \widehat{\mathbf{x}}_0(\boldsymbol{\mu}) = \mathbf{W}^\mathsf{T}\mathbf{x}_0(\boldsymbol{\mu}), \\ \widehat{\mathbf{y}}(t, \boldsymbol{\mu}) &= \widehat{\mathbf{C}}(\boldsymbol{\mu})\widehat{\mathbf{x}}(t, \boldsymbol{\mu}). \end{aligned} \tag{2.13}$$

Remark 2.8:
Care needs to be exercised while obtaining a ROM for a system with non-zero initial condition, i.e., $\mathbf{x}_0(\boldsymbol{\mu}) \neq 0$. As pointed out in [21], when the norm $\|\mathbf{x}_0\|_2$ of the non-zero initial condition is large, a significant error can be introduced when approximating the initial condition using projection. Proceeding in the manner described above, we have that

$$\mathbf{x}_0(\boldsymbol{\mu}) \neq \mathbf{V}\widehat{\mathbf{x}}_0(\boldsymbol{\mu}) = \mathbf{V}\mathbf{W}^{\mathsf{T}}\mathbf{x}_0(\boldsymbol{\mu}).$$

This incorrect approximation of the initial condition can lead to large deviations in the dynamics of the true and the reduced systems. A solution to this problem is to perform a coordinate transformation of the form

$$\mathbf{x}_c = \mathbf{x} - \mathbf{x}_0,$$

such that the transformed system has zero initial condition [21]. Besides this, other works such as [22, 109] also discuss the MOR of systems with non-zero initial conditions but the details are beyond the scope of this thesis. ♢

The reduced system matrices are obtained by making use of the affine parameter dependence of the system matrices.

$$\widehat{\mathbf{E}}(\boldsymbol{\mu}) := \mathbf{W}^{\mathsf{T}}\mathbf{E}_0\mathbf{V} + \sum_{i=1}^{Q_{\mathbf{E}}} \theta_{\mathbf{E}}^i\big(\mathbf{W}^{\mathsf{T}}\mathbf{E}_i\mathbf{V}\big), \tag{2.14a}$$

$$\widehat{\mathbf{A}}(\boldsymbol{\mu}) := \mathbf{W}^{\mathsf{T}}\mathbf{A}_0\mathbf{V} + \sum_{i=1}^{Q_{\mathbf{A}}} \theta_{\mathbf{A}}^i\big(\mathbf{W}^{\mathsf{T}}\mathbf{A}_i\mathbf{V}\big), \tag{2.14b}$$

$$\widehat{\mathbf{B}}(\boldsymbol{\mu}) := \mathbf{W}^{\mathsf{T}}\mathbf{B}_0 + \sum_{i=1}^{Q_{\mathbf{B}}} \theta_{\mathbf{B}}^i\big(\mathbf{W}^{\mathsf{T}}\mathbf{B}_i\big), \tag{2.14c}$$

$$\widehat{\mathbf{C}}(\boldsymbol{\mu}) := \mathbf{C}_0\mathbf{V} + \sum_{i=1}^{Q_{\mathbf{C}}} \theta_{\mathbf{C}}^i\big(\mathbf{C}_i\mathbf{V}\big). \tag{2.14d}$$

This is a major advantage in enabling a cheap computation of the ROM since the parameter-independent terms (e.g., $\big(\mathbf{W}^{\mathsf{T}}\mathbf{E}_i\mathbf{V}\big)$) can be precomputed once and for all, if $\mathbf{W}, \mathbf{V}$ are known.

The transfer function corresponding to the ROM in Eq. (2.13) is defined as

$$\widehat{\mathbf{H}}(\breve{\boldsymbol{\mu}}) := \widehat{\mathbf{C}}(\boldsymbol{\mu})\widehat{\mathbf{R}}(\breve{\boldsymbol{\mu}})^{-1}\widehat{\mathbf{B}}(\boldsymbol{\mu}), \tag{2.15}$$

where $\widehat{\mathbf{R}}(\breve{\boldsymbol{\mu}}) := \mathbf{W}^{\mathsf{T}}\mathbf{R}(\breve{\boldsymbol{\mu}})\mathbf{V} = \big(s\widehat{\mathbf{E}}(\boldsymbol{\mu}) - \widehat{\mathbf{A}}(\boldsymbol{\mu})\big)$. It is called the reduced transfer function throughout the thesis. A similar procedure can be used to obtain the ROMs for the time-harmonic system Eq. (2.10) and the fully discrete nonlinear system Eq. (2.11).

The ROM corresponding to the linear parametric system in Eq. (2.10) is given by:

$$\begin{aligned} \widehat{\mathbf{A}}(\breve{\boldsymbol{\mu}})\widehat{\mathbf{X}}(\breve{\boldsymbol{\mu}}) &= \widehat{\mathbf{B}}(\breve{\boldsymbol{\mu}}), \\ \widehat{\mathbf{Y}}(\breve{\boldsymbol{\mu}}) &= \widehat{\mathbf{C}}(\breve{\boldsymbol{\mu}})\widehat{\mathbf{X}}(\breve{\boldsymbol{\mu}}) \end{aligned} \tag{2.16}$$

with $\mathbf{X}(\breve{\boldsymbol{\mu}}) \approx \widetilde{\mathbf{X}}(\breve{\boldsymbol{\mu}}) = \mathbf{V}\widehat{\mathbf{X}}(\breve{\boldsymbol{\mu}})$ and $\mathbf{V} \in \mathbb{R}^{N\times n}$.

The ROM for the nonlinear system in Eq. (2.3) is of the form

$$\begin{aligned}\widehat{\mathbf{E}}(\boldsymbol{\mu})\frac{d}{dt}\widehat{\mathbf{x}}(t,\boldsymbol{\mu}) &= \widehat{\mathbf{A}}(\boldsymbol{\mu})\widehat{\mathbf{x}}(t,\boldsymbol{\mu}) + \widehat{\mathbf{f}}(\widehat{\mathbf{x}},\boldsymbol{\mu}) + \widehat{\mathbf{B}}(\boldsymbol{\mu})\mathbf{u}(t), \qquad \widehat{\mathbf{x}}(0,\boldsymbol{\mu}) = \widehat{\mathbf{x}}_0(\boldsymbol{\mu}) = \mathbf{W}^{\mathsf{T}}\mathbf{x}_0(\boldsymbol{\mu}),\\ \widehat{\mathbf{y}}(t,\boldsymbol{\mu}) &= \widehat{\mathbf{C}}(\boldsymbol{\mu})\widehat{\mathbf{x}}(t,\boldsymbol{\mu})\end{aligned} \tag{2.17}$$

where, in addition to the already defined variables, we have the reduced nonlinear quantity $\widehat{\mathbf{f}}(\widehat{\mathbf{x}},\boldsymbol{\mu}) := \mathbf{W}^{\mathsf{T}}\mathbf{f}(\mathbf{V}\widehat{\mathbf{x}},\boldsymbol{\mu})$. Furthermore, for the fully discrete nonlinear system Eq. (2.11) the ROM is of the form:

$$\begin{aligned}\widehat{\overline{\mathbf{E}}}(\boldsymbol{\mu})\widehat{\mathbf{x}}^{k+1} &= \widehat{\overline{\mathbf{A}}}(\boldsymbol{\mu})\widehat{\mathbf{x}}^{k} + \widehat{\overline{\mathbf{f}}}(\widehat{\mathbf{x}}^{k},\boldsymbol{\mu}) + \widehat{\overline{\mathbf{B}}}(\boldsymbol{\mu})\mathbf{u}(t^k),\\ \widetilde{\mathbf{y}}^{k+1} &= \widehat{\mathbf{C}}(\boldsymbol{\mu})\widehat{\mathbf{x}}^{k+1}\end{aligned} \tag{2.18}$$

where $\widehat{\overline{\mathbf{E}}}(\boldsymbol{\mu}) := \mathbf{W}^{\mathsf{T}}\overline{\mathbf{E}}(\boldsymbol{\mu})\mathbf{V} = \widehat{\mathbf{E}} - \Delta t\widehat{\mathbf{A}}$, $\widehat{\overline{\mathbf{A}}}(\boldsymbol{\mu}) := \mathbf{W}^{\mathsf{T}}\overline{\mathbf{A}}(\boldsymbol{\mu})\mathbf{V} = \widehat{\mathbf{E}}$ and $\widehat{\overline{\mathbf{f}}}(\widehat{\mathbf{x}}^k,\boldsymbol{\mu}) := \mathbf{W}^{\mathsf{T}}\overline{\mathbf{f}} = \Delta t\widehat{\mathbf{f}}(\widehat{\mathbf{x}},\boldsymbol{\mu})$.

Remark 2.9:
When the test space $\mathcal{S}_W$ is the same as the trial space $\mathcal{S}_V$, or equivalently, $\mathbf{W} = \mathbf{V}$, then the projection is denoted as a *Galerkin projection*. In applications such as circuit simulation, a Galerkin projection is preferred for its special properties such as stability preservation (see [148]). However, it has also been noticed [7] that Galerkin ROMs of some systems arising from fluid dynamics are unstable and a Petrov-Galerkin approach is preferred to ensure stability. ◊

Remark 2.10:
It is not always the case that the ROMs mentioned above are computationally cheaper to evaluate when compared to their corresponding FOMs. For nonlinear and non-affine systems, the ROM evaluation continues to incur a cost scaling with the high-dimension N. A solution to address this is the use of hyperreduction techniques, which will be addressed in more detail in Section 2.6. ◊

In order to obtain ROMs, we need to identify suitable left and right projection matrices $\mathbf{V}, \mathbf{W}$. A number of methods are available in the literature whose goal is the efficient construction of the two projection matrices in the frequency- or the time-domain. We only give a review of those MOR methods that are used in this thesis.

2.4. Frequency-domain MOR Methods

Frequency-domain MOR techniques are principally targeted towards the frequency-domain or input-output representation of systems (see, for example, Eqs. (2.4), (2.7) and (2.10)) and are popular in the systems and control theory community. Such techniques broadly fall under two families: Gramian-based approaches such as BT [29, 141, 194], Hankel-norm approximation [11, 205], Singular Perturbation Approximation [130], etc. and Krylov-subspace approaches such as Padé approximation, moment matching, rational interpolation [11, 101, 202]. Due to its relevance in this thesis, we give a detailed overview of the Krylov-subspace methods, in particular MMM.

2.4.1. Krylov-subspace Methods

Krylov-subspace methods seek projection matrices $\mathbf{W}, \mathbf{V}$ via interpolating the original transfer function (and its derivatives). They are applicable to problems with a fixed value of $\boldsymbol{\mu}$ and the transfer function reads $\mathbf{H}(s) := \mathbf{C}(s\mathbf{E}-\mathbf{A})^{-1}\mathbf{B}$. We write its power series at an expansion point $s_0 \in \mathbb{C}\backslash\mathcal{P}$ as

$$\begin{aligned}\mathbf{H}(s) &= \mathbf{C}\big((s-s_0+s_0)\mathbf{E}-\mathbf{A}\big)^{-1}\mathbf{B},\\ &= \mathbf{C}\big((s_0\mathbf{E}-\mathbf{A})+(s-s_0)\mathbf{E}\big)^{-1}\mathbf{B},\\ &= \mathbf{C}\big[\mathbf{I}+(s-s_0)(s_0\mathbf{E}-\mathbf{A})^{-1}\mathbf{E}\big]^{-1}(s_0\mathbf{E}-\mathbf{A})^{-1}\mathbf{B},\\ &= \sum_{i=0}^{\infty}\underbrace{\Big[\mathbf{C}[-(s_0\mathbf{E}-\mathbf{A})^{-1}\mathbf{E}]^i(s_0\mathbf{E}-\mathbf{A})^{-1}\mathbf{B}\Big]}_{=:\eta_i(s_0)}(s-s_0)^i\end{aligned} \tag{2.19}$$

with $\mathcal{P}$ denoting the set of all eigenvalues of $(s\mathbf{E}-\mathbf{A})$. Note that, in going from the second last to the last equality, we have made use of the Neumann lemma which states that for some $\|F\|<1$, $(I-F)^{-1}=\sum_{i=0}^{\infty}F^i$. Here, $\|\cdot\|$ is the induced matrix norm.

Definition 2.11 (Moments of the Transfer Function):
The quantity $\eta_i(s_0)$ in the power series expansion of $\mathbf{H}(s)$ in Eq. (2.19) is defined as the i-th *moment* of the transfer function around some point s_0, $i=0,1,\ldots$. Moreover, the moments are related to the derivatives of the transfer function $\mathbf{H}(s)$ w.r.t. s [11] as

$$\eta_i(s_0) = \frac{(-1)^i}{i!}\mathbf{H}^{(i)}(s_0). \tag{2.20}$$

where the superscript (i) of $\mathbf{H}$ represents the order of the derivative, for e.g., $\mathbf{H}^{(1)}(s) = \frac{d}{ds}\mathbf{H}(s)$, $\mathbf{H}^{(2)}(s)=\frac{d^2}{ds^2}\mathbf{H}(s)$ and so on. ♢

Moment-matching methods construct a ROM so that moments of its reduced transfer function $\widehat{\mathbf{H}}(s)$ (denoted $\hat{\eta}_i(s_0)$, $i=0,1,\ldots,\zeta$) match those of the original transfer function $\mathbf{H}(s)$ at s_0. Depending on the choice of the expansion point(s), the corresponding moment-matching method is known by different names. We highlight each below:

- For a single expansion point $s_0=0$, the moments are $\eta_i(0)=\mathbf{C}(-\mathbf{A}^{-1}\mathbf{E})^i\mathbf{A}^{-1}\mathbf{B}$. We seek $\eta_i(0)=\hat{\eta}_i(0)$, $i=0,1,\ldots,\zeta$. This approach is called *Padé approximation*. It preserves the DC gain $\mathbf{H}(0)$ and has good accuracy at low frequencies. For some problems in circuit simulation, it is even accurate at frequencies up to MHz.

- For a single expansion point $s_0=\infty$, the moments are obtained as $\mathbf{C}(\mathbf{E}^{-1}\mathbf{A})^{i-1}\mathbf{E}^{-1}\mathbf{B}$ and we require $\eta_i(\infty)=\hat{\eta}_i(\infty)$ $i=0,1,\ldots,\zeta$. This approach is known by the name of *partial realization* and the moments $\eta_i(\infty)$ are the *Markov parameters*. They are useful to characterize the impulse response of the system.

- Given a set of expansion points $\{s_j\}_{j=1}^{J}$ aiming at $\eta(s_j)=\hat{\eta}(s_j)$, the approach is termed *rational interpolation* or multipoint rational interpolation. Therefore, s_j are also called the *interpolation points*. We may alternatively use both names in

this thesis. For this case, the moments are given by $\eta_i = \mathbf{C}\big((\mathbf{A}-s_j\mathbf{E})^{-1}\mathbf{E}\big)^i(s_j\mathbf{E}-\mathbf{A})^{-1}\mathbf{B}$, $i = 1, 2, \ldots, \zeta_j$, with ζ_j referring to the number of moments to be matched at the j-th expansion/interpolation point s_j. We shall exclusively consider multipoint rational interpolation in the subsequent discussions. When the interpolation points are properly chosen, this approach is accurate for problems with wide frequency bands, e.g., $f \in [0, 10]$ GHz.

Constructing $\mathbf{V}, \mathbf{W}$ for the ROM is done based on a Krylov-subspace, which we define next.

Definition 2.12 (Krylov-subspace [139]):
Let $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{N\times N}$ be a non-singular matrix, $\mathbf{q} \in \mathbb{R}^N$ be a vector and $m \geq 1$ an integer. The m-th order Krylov subspace $\mathcal{K}_m(\boldsymbol{\mathcal{A}}, \mathbf{q})$ is defined as

$$\mathcal{K}_m(\boldsymbol{\mathcal{A}}, \mathbf{q}) = \text{span}\{\mathbf{q}, \boldsymbol{\mathcal{A}}\mathbf{q}, \boldsymbol{\mathcal{A}}^2\mathbf{q}, \ldots, \boldsymbol{\mathcal{A}}^{m-1}\mathbf{q}\}.$$

For $\mathbf{Q} \in \mathbb{R}^{N\times p}$, the m-th order block Krylov space is defined as

$$\mathcal{K}_m(\boldsymbol{\mathcal{A}}, \mathbf{Q}) = \text{span}\{\mathbf{Q}, \boldsymbol{\mathcal{A}}\mathbf{Q}, \boldsymbol{\mathcal{A}}^2\mathbf{Q}, \ldots, \boldsymbol{\mathcal{A}}^{m-1}\mathbf{Q}\}. \qquad \lozenge$$

The connection between interpolating the transfer function and the moments of the transfer function is made clear in the following theorem:

Theorem 2.13 ([101]):
Let $\mathbf{H}(s)$, $\widehat{\mathbf{H}}(s)$ be the original and reduced transfer function. Let $\{s_j\}_{i=1}^J \in \mathbb{C}\backslash\mathcal{P}$ be the set of interpolation points, if the projection matrices $\mathbf{V}$, $\mathbf{W}$ are chosen such that they span suitable Krylov-subspaces, i.e.,

$$\bigcup_{j=1}^{J}\mathcal{K}_{\zeta_j}\big((\mathbf{A}-s_j\mathbf{E})^{-1}\mathbf{E}, (\mathbf{A}-s_j\mathbf{E})^{-1}\mathbf{B}\big) \subseteq \text{span}(\mathbf{V}),$$
$$\bigcup_{j=1}^{J}\mathcal{K}_{\zeta_j}\big((\mathbf{A}-s_j\mathbf{E})^{-\mathsf{T}}\mathbf{E}^{\mathsf{T}}, (\mathbf{A}-s_j\mathbf{E})^{-\mathsf{T}}\mathbf{C}^{\mathsf{T}}\big) \subseteq \text{span}(\mathbf{W}),$$

then the first $2\zeta_j$ moments (starting from the zeroth moment) of $\widehat{\mathbf{H}}(s_j)$ match the first $2\zeta_j$ moments of $\mathbf{H}(s_j)$ for all $j = 1, 2, \ldots, J$. Usually, the same number of moments are matched at each interpolation point and we have $\zeta_1 = \zeta_2 = \cdots = \zeta_J$. $\lozenge$

Remark 2.14:
The Petrov-Galerkin approach involving the two projection matrices $\mathbf{V} \neq \mathbf{W}$ is called two-sided moment matching. A Galerkin approach, called one-sided moment matching, is also possible with $\mathbf{V} = \mathbf{W}$. For this case, only the first ζ moments (including the zeroth moment) of the reduced transfer function $\widehat{\mathbf{H}}(s)$ match those of the original transfer function $\mathbf{H}(s)$ for each interpolation point, i.e., we have

$$\eta_i(s_j) = \hat{\eta}_i(s_j), \qquad j = 1, 2, \ldots, J \;\&\; i = 0, 1, \ldots, (\zeta - 1). \qquad \lozenge$$

Solving the above shifted linear system of equations and generating the respective Krylov-subspaces can be achieved efficiently through the rational Krylov algorithm or the modified Gram-Schmidt process. Unlike BT, there are no general *a priori* error bounds for Krylov-subspace methods, but recent work has sought to address this [154]. Moreover, Krylov-subspace methods do not preserve stability automatically; special attempts have, however, been made to have Krylov-subspace ROMs that retain passivity or stability [79, 148]. The choice of the interpolation or expansion points $\{s_j\}_{j=1}^{J}$ plays a crucial role in determining the approximation quality of the reduced transfer function. The IRKA algorithm was proposed in [102] to identify the interpolation points iteratively by minimizing the $\mathcal{H}_2$-norm of the transfer function. Extensions to weakly-nonlinear systems have also been considered [25, 30]. Another approach to determine the interpolation points is to use *a posteriori* error bounds/estimators to pick a series of points from a training set Ξ [81, 85]. In Chapter 3, we discuss this approach in more detail.

Moment-matching techniques have also been extended to parametric systems $\mathbf{H}(\breve{\boldsymbol{\mu}})$ by considering the multivariate power series of the transfer function at an expansion point $\breve{\boldsymbol{\mu}}^0$ [35]. Next, we detail the MMM method, which we shall use extensively in Chapter 3.

2.4.2. Multi-moment Matching Method

While in the previous section we considered moment matching for the one-parameter case (s, the Laplace variable), multi-moment matching aims at moment matching for systems with more general parameter dependence. The first works to treat this problem include [39, 64]. In [82], a robust algorithm is proposed to realize multi-moment matching implicitly. In what follows, we adopt the setting in [82] to detail the MMM method.

The MMM method is applicable to parametric dynamical and steady/time-harmonic systems [84]. We use Galerkin projection to illustrate the method. For dynamical systems such as Eq. (2.4), the projection matrix $\mathbf{V}$ is obtained from its frequency-domain representation in Eq. (2.6), i.e.,

$$\mathbf{R}(\breve{\boldsymbol{\mu}})\mathbf{X}(\breve{\boldsymbol{\mu}}) = \mathbf{B}(\boldsymbol{\mu})\mathbf{u}(s), \tag{2.21a}$$

$$\mathbf{Y}(\breve{\boldsymbol{\mu}}) = \mathbf{C}(\boldsymbol{\mu})\mathbf{X}(\breve{\boldsymbol{\mu}}). \tag{2.21b}$$

Let $\mathbf{H}(\breve{\boldsymbol{\mu}}) := \mathbf{C}(\boldsymbol{\mu})\mathbf{R}(\breve{\boldsymbol{\mu}})^{-1}\mathbf{B}(\boldsymbol{\mu})$ be the associated transfer function for the system Eq. (2.21). The MMM method assumes that the parameter affine representation of $\mathbf{R}(\breve{\boldsymbol{\mu}})$ is available and is given by:

$$\mathbf{R}(\breve{\boldsymbol{\mu}}) := \mathbf{R}_0 + \sum_{i=1}^{h} \theta_i(\breve{\boldsymbol{\mu}})\mathbf{R}_i.$$

Expanding $\mathbf{X}(\breve{\boldsymbol{\mu}})$ into a Taylor series at an expansion point $\boldsymbol{\theta}^0 = [\theta_1^0, \cdots, \theta_h^0]^\mathsf{T}$, we obtain

$$\mathbf{X}(\breve{\boldsymbol{\mu}}) = \Big(\mathbf{I} - \big(\widetilde{\theta}_1\mathbf{M}_1 + \widetilde{\theta}_2\mathbf{M}_2 + \cdots + \widetilde{\theta}_h\mathbf{M}_h\big)\Big)^{-1}\mathbf{B}_\mathbf{M}\mathbf{u}(s), \tag{2.22}$$

where $\widetilde{\theta}_i := (\theta_i - \theta_i^0)$, $\widetilde{\mathbf{R}} := \mathbf{R}_0 + \theta_1^0\mathbf{R}_1 + \cdots + \theta_h^0\mathbf{R}_h$ and $\mathbf{M}_i := -\widetilde{\mathbf{R}}^{-1}\mathbf{R}_i$ for all $i = 1, 2, \ldots, h$. We set $\mathbf{B_M} := \widetilde{\mathbf{R}}^{-1}\mathbf{B}(\boldsymbol{\mu})$. Making use of the Neumann lemma, the solution may be expanded in terms of the infinite series below:

$$\begin{aligned}\mathbf{X}(\breve{\boldsymbol{\mu}}) &= \sum_{i=0}^{\infty} \big(\widetilde{\theta}_1\mathbf{M}_1 + \widetilde{\theta}_2\mathbf{M}_2 + \cdots + \widetilde{\theta}_h\mathbf{M}_h\big)^i \mathbf{B_M}\mathbf{u}(s), \\ &= \mathbf{B_M}\mathbf{u}(s) + \big(\widetilde{\theta}_1\mathbf{M}_1 + \widetilde{\theta}_2\mathbf{M}_2 + \cdots + \widetilde{\theta}_h\mathbf{M}_h\big)\mathbf{B_M}\mathbf{u}(s) + \cdots \\ &\quad + \big(\widetilde{\theta}_1\mathbf{M}_1 + \widetilde{\theta}_2\mathbf{M}_2 + \cdots + \widetilde{\theta}_h\mathbf{M}_h\big)^j \mathbf{B_M}\mathbf{u}(s) + \cdots\end{aligned} \tag{2.23}$$

With the definitions

$$\begin{aligned}&\mathbf{X}_0 := \mathbf{B_M}, \\ &\mathbf{X}_1 := \big(\widetilde{\theta}_1\mathbf{M}_1 + \widetilde{\theta}_2\mathbf{M}_2 + \cdots + \widetilde{\theta}_h\mathbf{M}_h\big)\mathbf{B_M} = \big(\widetilde{\theta}_1\mathbf{M}_1 + \widetilde{\theta}_2\mathbf{M}_2 + \cdots + \widetilde{\theta}_h\mathbf{M}_h\big)\mathbf{X}_0, \\ &\mathbf{X}_2 := \big(\widetilde{\theta}_1\mathbf{M}_1 + \widetilde{\theta}_2\mathbf{M}_2 + \cdots + \widetilde{\theta}_h\mathbf{M}_h\big)^2\mathbf{B_M} = \big(\widetilde{\theta}_1\mathbf{M}_1 + \widetilde{\theta}_2\mathbf{M}_2 + \cdots + \widetilde{\theta}_h\mathbf{M}_h\big)^2\mathbf{X}_1, \\ &\quad\vdots \qquad\qquad\qquad \vdots \qquad\qquad\qquad\qquad \vdots \\ &\mathbf{X}_j := \big(\widetilde{\theta}_1\mathbf{M}_1 + \widetilde{\theta}_2\mathbf{M}_2 + \cdots + \widetilde{\theta}_h\mathbf{M}_h\big)^j\mathbf{B_M} = \big(\widetilde{\theta}_1\mathbf{M}_1 + \widetilde{\theta}_2\mathbf{M}_2 + \cdots + \widetilde{\theta}_h\mathbf{M}_h\big)^j\mathbf{X}_{j-1}, \\ &\quad\vdots \qquad\qquad\qquad \vdots \qquad\qquad\qquad\qquad \vdots\end{aligned}$$

we get $\mathbf{X} = (\mathbf{X}_1 + \mathbf{X}_2 + \cdots)\mathbf{u}(s)$. Using the relations between successive terms in the above expansions, we can define the following recursive subspace sequence:

$$\begin{aligned}\boldsymbol{\mathcal{R}}_0 &= \mathbf{B_M}, \\ \boldsymbol{\mathcal{R}}_1 &= \big[\mathbf{M}_1\boldsymbol{\mathcal{R}}_0, \mathbf{M}_2\boldsymbol{\mathcal{R}}_0, \ldots, \mathbf{M}_h\boldsymbol{\mathcal{R}}_0\big], \\ \boldsymbol{\mathcal{R}}_2 &= \big[\mathbf{M}_1\boldsymbol{\mathcal{R}}_1, \mathbf{M}_2\boldsymbol{\mathcal{R}}_1, \ldots, \mathbf{M}_h\boldsymbol{\mathcal{R}}_1\big], \\ &\vdots \qquad\qquad \vdots \\ \boldsymbol{\mathcal{R}}_j &= \big[\mathbf{M}_1\boldsymbol{\mathcal{R}}_{j-1}, \mathbf{M}_2\boldsymbol{\mathcal{R}}_{j-1}, \ldots, \mathbf{M}_h\boldsymbol{\mathcal{R}}_{j-1}\big], \\ &\vdots \qquad\qquad \vdots\end{aligned}$$

For each term in the sequence, $\mathbf{C}(\breve{\boldsymbol{\mu}})\boldsymbol{\mathcal{R}}_j$ corresponds to the j-th multi-moment of $\mathbf{H}(\breve{\boldsymbol{\mu}})$ for all $j = 0, 1, \ldots$. We collect up to the ζ-th term (starting from zero) in the above sequence and set

$$\boldsymbol{\mathcal{R}} := \operatorname{span}\{\boldsymbol{\mathcal{R}}_0, \boldsymbol{\mathcal{R}}_1, \ldots, \boldsymbol{\mathcal{R}}_\zeta\}$$

such that $\mathbf{X} \approx \widetilde{\mathbf{X}} \in \boldsymbol{\mathcal{R}}$, $\widetilde{\mathbf{X}} = \mathbf{V}\widehat{\mathbf{X}}$. The orthonormal basis $\mathbf{V}$ of $\boldsymbol{\mathcal{R}}$ can be constructed using a modified Gram-Schmidt process. Algorithm 2.1 sketches the pseudocode, originally proposed in [82].

Remark 2.15:
For time-harmonic systems in Eq. (2.10) or steady systems ($\breve{\boldsymbol{\mu}} = \boldsymbol{\mu}$ in Eq. (2.10)) the derivation of $\mathbf{V}$ is similar and can be done by replacing $\mathbf{R}(\breve{\boldsymbol{\mu}})$ in Eq. (2.21) with $\mathbf{A}(\breve{\boldsymbol{\mu}})$ or $\mathbf{A}(\boldsymbol{\mu})$, respectively. ♢

The resulting ROM for Equation (2.4) is in the form of Equation (2.13), with $\mathbf{V} = \mathbf{W}$ computed from Algorithm 2.1. The reduced transfer function $\widehat{\mathbf{H}}(\breve{\boldsymbol{\mu}})$ of the ROM is then $\widehat{\mathbf{H}}(\breve{\boldsymbol{\mu}}) := \widehat{\mathbf{C}}(\boldsymbol{\mu})\widehat{\mathbf{R}}(\breve{\boldsymbol{\mu}})^{-1}\widehat{\mathbf{B}}(\boldsymbol{\mu})$. We state the following theorem:

Theorem 2.16 (Multi-Moment matching [82]):
Suppose the reduced transfer function $\widehat{\mathbf{H}}(\breve{\boldsymbol{\mu}})$ is obtained via Galerkin projection using the matrix $\mathbf{V}$. If $\mathbf{V}$ satisfies $\text{range}(\mathbf{V}) = \text{span}\{\boldsymbol{\mathcal{R}}_0, \boldsymbol{\mathcal{R}}_1, \ldots, \boldsymbol{\mathcal{R}}_\zeta\}$, then the multi-moments of $\widehat{\mathbf{H}}(\breve{\boldsymbol{\mu}})$ match those of $\mathbf{H}(\breve{\boldsymbol{\mu}})$ up to order ζ, that is, $\mathbf{C}(\boldsymbol{\mu})\boldsymbol{\mathcal{R}}_j = \widehat{\mathbf{C}}(\boldsymbol{\mu})\widehat{\boldsymbol{\mathcal{R}}}_j$, $j = 0, 1, \ldots, \zeta$. ◊

For a proof of the above theorem, we refer to [82]. Having reviewed some frequency-domain MOR techniques, we now turn to time-domain MOR methods.

2.5. Time-domain MOR Methods

Time-domain MOR techniques include methods such as the POD method and the RBM. In the fluid dynamics community, the Dynamic Mode Decomposition (DMD) [58, 127, 178, 184, 206] is a popular approach. A more recent time-domain MOR method is the operator-inference approach [31, 161]. Below, we only give a review of the POD method and the RBM as they will be used extensively in this thesis.

2.5.1. Proper Orthogonal Decomposition

POD has a long history in mathematics as a data analysis and compression tool. Its principal application is to transform and represent data in a new (preferably lower dimensional) coordinate system whose basis vectors are orthogonal and are obtained from the data itself. This underlying idea has been used in different contexts in several fields, under different names: Principal Component Analysis (PCA) in statistics, Karhunen-Loève Transform (KLT) in stochastics and signal processing, singular value decomposition (SVD) in numerical linear algebra, etc. As a MOR tool, POD was initially used in fluid dynamics [117] to decompose a flow field and represent its characteristics in a few generic (orthogonal) modes. It is usually the most preferred method to perform MOR of nonlinear dynamical systems.

POD works on *snapshots* of the solution to a FOM such as the system in Eq. (2.3) or Eq. (2.11). We define $\Xi := [\bar{\boldsymbol{\mu}}_1, \bar{\boldsymbol{\mu}}_2, \ldots, \bar{\boldsymbol{\mu}}_{n_s}]$ as the training set consisting of time and parameter samples, i.e., $\bar{\boldsymbol{\mu}} = t$, $\bar{\boldsymbol{\mu}} = \boldsymbol{\mu}$, or $\bar{\boldsymbol{\mu}} = (t, \boldsymbol{\mu})$ at which the data vectors $\{\mathbf{d}_i := \mathbf{d}(\bar{\boldsymbol{\mu}}_i)\}_{i=1}^{n_s}$ are obtained. We denote by $\mathbf{D} := [\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_{n_s}] \in \mathbb{R}^{N \times n_s}$ the snapshot matrix containing all the snapshots as its columns. POD first applies SVD to $\mathbf{D}$:

$$\mathbf{D} = \mathbf{L}\boldsymbol{\Sigma}\mathbf{K}^\mathsf{T}$$

where $\mathbf{L} \in \mathbb{R}^{N \times N}$, $\mathbf{K} \in \mathbb{R}^{n_s \times n_s}$ are orthogonal matrices. The columns of $\mathbf{L}, \mathbf{K}$ are, respectively, the left singular vectors and right singular vectors. The main diagonal of the rectangular matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{N \times n_s}$ contains non-negative elements arranged in a decreasing order: $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{n_d} > \sigma_{n_{d+1}} = \sigma_{n_{d+2}} = \cdots = \sigma_{n_s} = 0$, where $n_d \leq \min(N, n_s)$ is the rank of $\mathbf{D}$. The columns of $\mathbf{L}$, $\{\mathbf{l}_i\}_{i=1}^{N}$ are the POD basis or

Algorithm 2.1: `MMM`

Computes the projection basis $\mathbf{V}$ using the Multi-moment Matching Method.

Input: System matrices: $\mathbf{R}, \mathbf{B}$, Number of multi-moments to match ζ, Deflation tolerance ϵ_{def}.

Output: Projection matrix $\mathbf{V} := [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{n_{\text{mm}}}]$.

Initialization: $ncol = 0$, $indx\_start = 0$, $indx\_end = 0$.
Compute $\boldsymbol{\mathcal{R}}_0 = \widetilde{\mathbf{R}}^{-1}\mathbf{B}(\boldsymbol{\mu})$.
if $\texttt{size}(\mathbf{B}, 2) > 1$ **then**
    $\mathbf{V} = \texttt{mgs\_def}(\boldsymbol{\mathcal{R}}_0, \epsilon_{\text{def}})$; $ncol = \texttt{size}(\mathbf{V}, 2)$. `/* Algorithm A.1 (Appendix A.1) modified Gram-Schmidt with deflation to orthogonalize the columns of` $\boldsymbol{\mathcal{R}}_0$, *ncol* `is the number of columns in the orthogonal basis` $\mathbf{V}$ `*/`
else
    $\mathbf{V} = \frac{\boldsymbol{\mathcal{R}}_0}{\|\boldsymbol{\mathcal{R}}_0\|_2}$; $ncol = 1$.
end
for $iter = 1, 2, \ldots, \zeta$ **do**
    $indx\_end = ncol$.
    for $j = 1, 2, \ldots, h$ **do**
        if $indx\_start == indx\_end$ **then**
            `break`
        else
            for $k = indx\_start + 1, \ldots, indx\_end$ **do**
                $\mathbf{m} = \widetilde{\mathbf{R}}^{-1}\mathbf{R}_j\mathbf{v}_k$.
                $\mathbf{v}_{\text{orth}} = \texttt{orth\_def}(\mathbf{V}, \mathbf{m}, \epsilon_{\text{def}})$.
                $\mathbf{V} = \left[\mathbf{V}, \mathbf{v}_{\text{orth}}\right]$; $ncol = \texttt{size}(\mathbf{V}, 2)$.
            end
        end
    end
    $indx\_start = indx\_end$.
end

Function `orth_def`$(\mathbf{V}, \mathbf{m}, \epsilon_{\text{def}})$**:**
    $col = \texttt{size}(\mathbf{V}, 2)$
    for $k = 1, 2, \ldots, col$ **do**
        $h = \mathbf{v}_k^{\mathsf{T}}\mathbf{m}$.
        $\mathbf{m} = \mathbf{m} - h\mathbf{v}_k$.
    end
    if $\|\mathbf{m}\| > \epsilon_{\text{def}}$ **then**
        $\mathbf{v}_{\text{orth}} := \frac{\mathbf{m}}{\|\mathbf{m}\|_2}$.
    else
        $\mathbf{v}_{\text{orth}} = [\,]$.
    end
    return $\mathbf{v}_{\text{orth}}$.

Algorithm 2.2: POD
Computes the rank-n Proper Orthogonal Decomposition (POD) basis.

Input: Data matrix: $\mathbf{D}$, POD tolerance ϵ_{POD} or desired rank n.
Output: POD basis $\mathbf{V}$.

Compute the SVD of the data matrix: $\mathbf{D} \xrightarrow{SVD} \mathbf{L\Sigma K}^{\mathsf{T}}$.
if *n is not an input* **then**
  Find the smallest n for which RIL $< \epsilon_{\text{POD}}$ in Eq. (2.26).
end
return $\mathbf{V} = \mathbf{L}(:, 1:n)$.

POD modes of the data. The projection matrix $\mathbf{V}$ is obtained by selecting the first $n \leq n_d$ POD modes, i.e., $\mathbf{V} := \mathbf{L}(:, 1:n)$. The rank n POD basis has the following optimality property:

Theorem 2.17 (POD optimality [203]):
Let $\mathbf{D}$ be a given data matrix and let $\mathbf{D} = \mathbf{L\Sigma K}^{\mathsf{T}}$ be its SVD as described above. For any $n \in \{1, 2, \ldots, n_d\}$ the solution to the following optimization problem

$$\max_{\bar{\mathbf{l}}_1, \ldots, \bar{\mathbf{l}}_n \in \mathbb{R}^N} \sum_{i=1}^{n} \sum_{j=1}^{n_s} \left| \langle \mathbf{d}_j, \bar{\mathbf{l}}_i \rangle \right|^2, \qquad \text{such that, } \langle \bar{\mathbf{l}}_i, \bar{\mathbf{l}}_j \rangle = \delta_{ij},\ 1 \leq i, j \leq n \tag{2.24}$$

is given by the left singular vectors $\{\mathbf{l}_i\}_{i=1}^n$. Setting $\mathbf{V} := \mathbf{L}(:, 1:n)$ minimizes the least squares error in approximating the data matrix, that is,

$$\min_{\mathbf{V} \in \mathbb{R}^{N \times n}} \|\mathbf{D} - \mathbf{VV}^{\mathsf{T}}\mathbf{D}\|_F^2 = \min_{\mathbf{V} \in \mathbb{R}^{N \times n}} \sum_{i=1}^{n_s} \|\mathbf{d}_i - \mathbf{VV}^{\mathsf{T}}\mathbf{d}_i\|_2^2 = \sum_{i=n+1}^{n_d} \sigma_i^2. \tag{2.25}$$

In Eq. (2.24), δ_{ij} is the Kronecker delta: $\delta_{ij} = 1$, for $i = j$ and $\delta_{ij} = 0$, otherwise. ♢

Algorithm 2.2 sketches the pseudocode for constructing the rank n POD basis. For many systems, the rank n to get a good approximation of the data matrix $\mathbf{D}$ is very small, i.e., $n \ll N$. The appropriate choice of the reduced dimension n is often done through heuristics [164]. For example, in fluid dynamics, n is chosen as the smallest quantity for which the following expression is true:

$$\text{RIL} = \frac{\sum_{j=n+1}^{n_d} \sigma_j^2}{\sum_{j=1}^{n_d} \sigma_j^2} < \epsilon_{\text{POD}}. \tag{2.26}$$

RIL stands for the *relative information loss* and $((1 - \text{RIL})/100)$ means the percentage of 'energy' captured by the retained singular values $\{\sigma_i\}_{i=1}^n$. Here, ϵ_{POD} is a small, user-desired tolerance in approximation, for example, $10^{-5}, 10^{-10}$, etc. This expression has been adopted in other applications to good effect. Another recent approach to determine n is based on singular value thresholding [4].

In [87], the authors discuss an adaptive procedure to determine the reduced order n by starting from a small value and making iterative increments. The magnitude of the increments is determined using an *a posteriori* error estimator. We propose an extension of this in Chapter 4, in the form of a *two-way* adaptive scheme that can iteratively increment and decrement the reduced order n.

The choice of the snapshots $\{\mathbf{d}_i\}_{i=1}^{n_s}$ (equivalently, the choice of time and parameter instances in the training set Ξ where the solution is sampled) is crucial to obtain a good ROM. For non-parametric dynamical systems, a number of works discuss selection criteria to obtain optimal time snapshot locations [126, 151]. For data matrices containing many snapshots, the Adaptive Snapshot Selection (AdSS) method from [213] has been proposed to remove (almost) linearly dependent information from the trajectory. For general parametric systems, strategies such as Latin Hypercube sampling, Orthogonal sampling [41], sparse grid sampling [6], etc. have been considered. However, these approaches are often not very effective since they do not take into account the underlying model and its dynamics in the sampling process. Even if 'optimal' snapshot locations were available, POD still faces the issue of needing a FOM solution at each parameter sample to generate the data matrix. This is often prohibitive for large-scale systems, especially when the number of samples n_s is large.

Although the snapshots for POD may come from simulations of discretized (dynamical) systems such as those in Eq. (2.3), the method itself makes no assumptions about the underlying model which generated the data. The optimality property in Theorem 2.17 concerns optimal data reconstruction and does not carry over to the ROM generated using the POD basis $\mathbf{V}$. The work [171] gives a good analysis of this. For nonlinear systems, the ROM obtained through a Galerkin projection using the POD basis may not yield the desired computational speedup. This is caused by the reduced nonlinear term (e.g., $\widehat{\mathbf{f}}(\widehat{\mathbf{x}}, \boldsymbol{\mu})$), whose evaluation still depends on the dimension N of the FOM; see also Remark 2.10. Next, we detail the RBM, a time-domain MOR approach that incorporates knowledge about the underlying model in the sampling process through *a posteriori* error estimators.

2.5.2. Reduced Basis Method

RBM is a widely used MOR technique tailored towards parametrized PDEs. The origin of RBM as a model reduction approach dates all the way back to the 1970s, when it was initially used to reduce systems arising from linear and nonlinear structural analysis [5, 88, 142, 146, 147]. Succeeding decades saw further developments of RBM in theory [165, 175], and applications [122, 162]. In the early 2000s, several major breakthroughs in the RBM such as the Greedy algorithm for basis generation and rigorous *a posteriori* error estimation were made, establishing the method in a sound theoretical and numerical formulation; see [133, 166] and references therein. These works extended the RBM to (i) linear elliptic problems, both coercive and non-coercive [177, 201], (ii) linear parabolic problems [97, 100], (iii) nonlinear and/or non-affine elliptic, parabolic problems [99, 200]. Recent works have focussed on extending RBM to problems with high parameter dimension [75, 113], hyperbolic problems [63, 94], to name just a few. A comprehensive discussion may be found in the books [112, 167] and in the survey works [60, 180].

The RBM also relies on solution snapshots to identify a suitable projection matrix $\mathbf{V}$. However, unlike POD, $\mathbf{V}$ is enriched iteratively. The RBM has some salient features that sets it apart in comparison to other MOR methods. These are:

(a) *A posteriori* error estimation: an error estimator or indicator for the state or output variable is utilized to steer the parameter sampling for iteratively updating the basis $\mathbf{V}$ and to certify the accuracy of the ROM.

(b) Greedy sampling: RBM makes use of a greedy sampling strategy guided by the error estimator to efficiently explore the parameter space $\mathcal{P}$; this ensures construction of a good projection basis at a reasonable computational cost.

It is also worth mentioning here that the RBM was the first PMOR technique that formalized the *offline-online* paradigm, which involves performing computationally expensive tasks such as basis generation only once, at an offline stage. The online stage involves evaluating the ROM rapidly at any desired parameter with costs that do not scale with the order N of the FOM.

The RBM is applicable to both discretized time-independent (Eq. (2.8), Eq. (2.9), Eq. (2.10)) and time-dependent (Eqs. (2.3) and (2.11)) problems. For the former, the RBM uses a Greedy algorithm, while for the latter, an extension of the Greedy algorithm, known as POD-Greedy algorithm, is employed. Following the *a priori* convergence rates for the Greedy algorithm [37], the convergence of the POD-Greedy algorithm was established in [103], where it was proved that the POD-Greedy algorithm results in exponential or polynomial convergence rate, except for advection-dominated problems.

In this thesis, we apply RBM to the fully-discrete systems in the form given in Eq. (2.11) for the convenience of deriving *a posteriori* error estimators, therefore we focus on the main aspects of the RBM theory for the POD-Greedy algorithm. Nevertheless, for the sake of completeness, we have sketched the Greedy algorithm in Algorithm 2.3.

POD-Greedy Algorithm The POD-Greedy algorithm was introduced in [106] as a combination of POD data compression and the Greedy algorithm used in the RBM Algorithm 2.3. It requires two main ingredients: (i) a discretization of the parameter domain $\mathcal{P}$ in the form of a training set Ξ (this is similar to POD discussed earlier) and (ii) an *a posteriori* error estimator denoted by $\Delta(\boldsymbol{\mu})$

$$\|\mathbf{x}^k - \widetilde{\mathbf{x}}^k\| \leq \Delta(\boldsymbol{\mu}) \qquad \text{or} \qquad \|\mathbf{y}^k - \widetilde{\mathbf{y}}^k\| \leq \Delta(\boldsymbol{\mu}) \tag{2.27}$$

that estimates the approximation error of the state or that of the output quantity computed by the ROM. Algorithm 2.4 sketches the POD-Greedy algorithm.

Recall from the discussion in Section 2.5.1 that POD requires the FOM solutions at all n_s parameter samples in the training set Ξ to obtain the projection matrix $\mathbf{V}$ through an SVD. The RBM circumvents this bottleneck by performing instead, an iterative update of $\mathbf{V}$ while computing only a cheap and efficient error estimator over all the n_s parameters at each iteration. To achieve this, a greedy sampling of the training set is done. At the *iter*-th iteration, RBM selects one parameter ($\bar{\boldsymbol{\mu}}^*$) from the

Algorithm 2.3: GREEDY
Computes the rank-n basis using the Greedy algorithm for time-independent systems.

Input: Training set Ξ, ROM tolerance ϵ, Deflation tolerance ϵ_{def}, Maximum iterations *iter_max*.
Output: Projection matrix $\mathbf{V}$.

1 Initialization: $\mathbf{V} = [\,]$, $err\_max = 1 + \epsilon$, $iter = 1$, $\bar{\boldsymbol{\mu}}^*$ (chosen randomly from Ξ).
2 **while** $err\_max > \epsilon$ and $iter \leq iter\_max$ **do**
3 Solve FOM (e.g., Eq. (2.8), Eq. (2.9) or Eq. (2.10)) at $\bar{\boldsymbol{\mu}}^*$; obtain snapshots matrix $\mathbf{X}(\bar{\boldsymbol{\mu}}^*)$.
4 $\mathbf{V} = \texttt{orth\_def\_mat}(\mathbf{V}, \mathbf{X}(\bar{\boldsymbol{\mu}}^*), \epsilon_{\text{def}})$. `/* orthogonalize` $\mathbf{X}(\bar{\boldsymbol{\mu}}^*)$ `against the columns of` $\mathbf{V}$`; see Step 9 of Algorithm 2.4 */`
5 $iter = iter + 1$.
6 Find $\bar{\boldsymbol{\mu}}^* = \arg\max_{\boldsymbol{\mu} \in \Xi} \Delta(\boldsymbol{\mu})$.
7 Set $err\_max = \Delta(\bar{\boldsymbol{\mu}}^*)$.
8 **end**

training set and solves the FOM Eq. (2.11) at that parameter. The snapshots of the resulting solution at time instances $\{t^k\}_{k=0}^K$ are collected in a snapshot matrix $\mathbf{X}$ as

$$\mathbf{X}(\bar{\boldsymbol{\mu}}^*) := \left[\mathbf{x}^0, \mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^K\right] \in \mathbb{R}^{N \times N_t}, \qquad K = N_t + 1. \tag{2.28}$$

Then, the snapshots are projected onto the current basis $\mathbf{V}$ to obtain a new snapshot matrix $\overline{\mathbf{X}} := \mathbf{X} - \mathbf{V}\mathbf{V}^{\mathsf{T}}\mathbf{X}$ containing only new information. Following this, the POD (Algorithm 2.2) is applied to $\overline{\mathbf{X}}(\bar{\boldsymbol{\mu}}^*)$ to get $\mathbf{V}_{\text{POD}} \in \mathbb{R}^{N \times r_{\text{RB}}}$ and the projection matrix $\mathbf{V}$ is updated as

$$\mathbf{V} \leftarrow \left[\mathbf{V},\ \mathbf{V}_{\text{POD}}\right].$$

Here, r_{RB} is the number of POD modes used to enrich the projection matrix $\mathbf{V}$ at the *iter*-th iteration. Normally, r_{RB} is either set to 1 or is determined using a user-defined POD tolerance. It is important to orthonormalize $\mathbf{V}$ for good numerical performance. A modified Gram-Schmidt procedure is used to iteratively orthonormalize the newly added basis vectors, see Step 9 of Algorithm 2.4. The function `orth_def_mat` is used and is defined at Steps 15-22. The selection of the parameter $\bar{\boldsymbol{\mu}}^*$ is guided by the *a posteriori* error estimator $\Delta(\boldsymbol{\mu})$. The parameter $\bar{\boldsymbol{\mu}}^*$ at the $(iter + 1)$-th iteration is chosen as the one at which $\Delta(\boldsymbol{\mu})$ is the largest among all $\boldsymbol{\mu} \in \Xi$. This process is repeated until the estimated error goes below a user-specified tolerance (ϵ), or if the user-defined maximum number of iterations (*iter_max*) is reached. Note that, at Step 12 of the algorithm, the ROM (Eq. (2.18)) obtained with the updated $\mathbf{V}$, has to be computed for all $\boldsymbol{\mu} \in \Xi$ in order to evaluate the error estimator. For the POD-Greedy algorithm, the maximum number of FOM solutions required is the number of iterations of the POD-Greedy algorithm.

The POD-Greedy algorithm has been extended to nonlinear problems through the use of hyperreduction strategies (to be elaborated on in Section 2.6) such as the EIM, DEIM, etc. [19, 52, 71]. Initial attempts in this direction relied on precomputation

Algorithm 2.4: `PODGREEDY`
Computes the rank-n basis using POD-Greedy algorithm for time-dependent systems.

Input: Training set Ξ, ROM tolerance ϵ, Number of POD modes r_{RB}, Deflation tolerance ϵ_{def}, Maximum iterations $iter\_max$.
Output: Projection matrix $\mathbf{V}$.

1 Initialization: $\mathbf{V} = [\,]$, $err\_max = 1 + \epsilon$, $iter = 1$, $\bar{\boldsymbol{\mu}}^*$ (chosen randomly from Ξ).
2 **while** $err\_max > \epsilon$ and $iter \leq iter\_max$ **do**
3 Solve FOM Eq. (2.11) at $\bar{\boldsymbol{\mu}}^*$; obtain snapshots matrix $\mathbf{X}(\bar{\boldsymbol{\mu}}^*)$.
4 **if** $iter == 1$ **then**
5 $\mathbf{V} = \texttt{POD}(\mathbf{X}, r_{\text{RB}})$. `/* implement Algorithm 2.2 with inputs` $\mathbf{X}$ `and` $n = r_{\text{RB}}$ `*/`
6 **else**
7 Compute $\overline{\mathbf{X}} := \mathbf{X} - \mathbf{V}\mathbf{V}^{\mathsf{T}}\mathbf{X}$.
8 $\mathbf{V}_{\text{POD}} := \texttt{POD}(\overline{\mathbf{X}}, r_{\text{RB}})$.
9 $\mathbf{V} = \texttt{orth\_def\_mat}(\mathbf{V}, \mathbf{V}_{\text{POD}}, \epsilon_{\text{def}})$. `/* orthogonalize the columns of` $\mathbf{V}_{\text{POD}}$ `against those of` $\mathbf{V}$ `using the function orth_def in Algorithm 2.1 */`
10 **end**
11 $iter = iter + 1$.
12 Find $\bar{\boldsymbol{\mu}}^* = \arg\max_{\boldsymbol{\mu} \in \Xi} \Delta(\boldsymbol{\mu})$.
13 Set $err\_max = \Delta(\bar{\boldsymbol{\mu}}^*)$.
14 **end**

15 **Function** `orth_def_mat(`$\mathbf{V}$, $\mathbf{V}_{\text{POD}}$, ϵ_{def}`):`
16 $col = \texttt{size}(\mathbf{V}_{\text{POD}}, 2)$
17 **for** $k = 1, 2, \ldots, col$ **do**
18 $\mathbf{m} := \mathbf{V}_{\text{POD}}(:, k)$.
19 $\mathbf{v}_{\text{orth}} := \texttt{orth\_def}(\mathbf{V}, \mathbf{m}, \epsilon_{\text{def}})$. `/* orthogonalize the vector` $\mathbf{m}$ `against those of` $\mathbf{V}$ `using the function orth_def in Algorithm 2.1 */`
20 $\mathbf{V} = \left[\mathbf{V}, \mathbf{v}_{\text{orth}}\right]$.
21 **end**
22 **return** $\mathbf{V}$.

of the hyperreduction basis and suffered from a need to carry out FOM solutions at all training samples. In recent years, much attention has been paid to simultaneously enrich the RB and hyperreduction bases [24, 71, 209]. We discuss this issue in Chapter 4 and detail a new strategy for adaptive basis enrichment.

In Step 5 and Step 8 of Algorithm 2.4, the value of r_{RB} needs to be defined and the choices considered in literature have been mainly heuristic. We describe a more principled approach to make this choice as a part of the new adaptive basis enrichment algorithm to be discussed in Chapter 4.

A reliable, sharp estimation of the error is vital to generate good ROMs. While just one FOM solution is required per iteration, the POD-Greedy algorithm needs n_s

evaluations of the error estimator. Therefore, in addition to being sharp, $\Delta(\boldsymbol{\mu})$ should be cheap to evaluate. Error estimation has been an intensely researched topic in the RBM community; see [99, 107, 155, 197, 198, 210, 214]. Nevertheless, error estimation for general nonlinear systems is still a less explored area. In Chapter 4, we discuss this issue in greater depth and propose a novel output error estimator for a modified output quantity to improve the sharpness of the estimate. In addition to improving the error estimation, we also propose a RBF-based technique of reducing the computational costs of the new error estimator.

Another crucial ingredient for the POD-Greedy algorithm is the choice of the training set. For computational efficiency, fewer samples are preferred. At the same time, a *sufficiently large* number of samples is needed to capture all the variations over the parameter space $\mathcal{P}$. These contradictory requirements have motivated research that look to *optimally* sample Ξ [76, 105, 124, 134, 156, 187, 195]. We propose two approaches to sample the training set in Chapter 4. The first seeks to iteratively add new parameters to it, through a *coarse-fine* sampling strategy. The second approach is based on subsampling a large, fine training set by choosing only the samples most relevant to the problem.

For general nonlinear systems, MOR with POD or RBM alone is often not sufficient for efficient ROM evaluation. A possible remedy is through the use of hyperreduction strategies.

2.6. Hyperreduction

In Remarks 2.1 and 2.10, two challenges faced by projection-based MOR methods for nonlinear and non-affine systems were highlighted. For these two types of systems, the online computation costs for ROM evaluation scales with the dimension of the FOM. Unless treated, these two issues hinder the efficient evaluation of the ROM in real-time and multi-query simulations.

Hyperreduction methods[2] are a family of techniques that ensure efficient online evaluation of the ROM with nonlinear or non-affine terms. Hyperreduction was first employed in the context of image reconstruction, in the Gappy-POD method [77]. In the context of MOR, the EIM [19] was the earliest hyperreduction strategy and it was used together with the RBM to enable efficient evaluation of non-affine functions [19]. This was later extended to nonlinear problems [98]. Over the years, many related hyperreduction methods have been proposed. These include: the discrete EIM (DEIM) [52], the empirical operator interpolation method (EOIM) [71], the Best Point Interpolation Method (BPIM) [144], Missing Point Estimation (MPE) [17]; see also the recent survey [78] for a comprehensive summary.

Suppose $f(g(\mathbf{z}, t, \boldsymbol{\mu}), \boldsymbol{\mu})$ is some arbitrary Lipschitz continuous nonlinear function of $g \in \mathcal{V} \subset L^2(\Omega)$ and/or non-affine w.r.t the parameter $\boldsymbol{\mu} \in \mathcal{P} \subset \mathbb{R}^{N_p}$. Let

$$\mathcal{M}^f := \{f(\cdot, \boldsymbol{\mu}) | \boldsymbol{\mu} \in \mathcal{P}\}$$

[2]Many techniques classified as hyperreduction methods (such as Gappy-POD, EIM) predate the coining of the term 'hyperreduction'. The term was used in the specific context of strategies that employed a reduced integration domain for material problems. However, recent MOR literature retroactively applies the term as a catch-all notation for all reduction strategies.

be the function manifold resulting from varying the parameter $\boldsymbol{\mu}$. Hyperreduction techniques seek an approximation of f in the following form of variable separation:

$$f(g(\mathbf{z},t,\boldsymbol{\mu}),\boldsymbol{\mu}) \approx \mathscr{I}[f(g(\mathbf{z},t,\boldsymbol{\mu}),\boldsymbol{\mu})] := \sum_{i=1}^{n_{\mathrm{EI}}} \xi_i(t,\boldsymbol{\mu}) u_i^{\mathrm{EI}}(\mathbf{z}). \tag{2.29}$$

Here, $u_i^{\mathrm{EI}}(\mathbf{z})$, $i = 1,2,\ldots,n_{\mathrm{EI}}$ are a set of time, parameter-independent basis functions, $\xi_i \in \mathbb{R}$, $i = 1,2,\ldots,n_{\mathrm{EI}}$ are parameter-dependent coefficients and $\mathscr{I}[\cdot]$ is the interpolation operator. Adopting the *offline-online* paradigm of the RBM, all hyperreduction methods precompute the parameter/time-independent basis functions at an *offline* stage while the coefficients ξ_i are determined at the *online* stage. Notice that, Eq. (2.29) is precisely the parameter affine representation such as the ones in Remarks 2.1 and 2.10. For many problems, just a few well-chosen basis functions u_i^{EI} suffice for a good approximation.

Hyperreduction techniques vary based on how they identify the basis functions and what constraints they impose on Eq. (2.29) to determine the coefficients ξ_i. Methods such as EIM [19], DEIM [52], BPIM [144] seek an approximation that *interpolates* the original function at a small number (n_I) of carefully chosen spatial interpolation points. More precisely,

$$f(g(\mathbf{z}_{\wp_j},t,\boldsymbol{\mu}),\boldsymbol{\mu}) = \mathscr{I}[f(g(\mathbf{z}_{\wp_j},t,\boldsymbol{\mu}),\boldsymbol{\mu})], \qquad j = 1,2,\ldots,n_I = n_{\mathrm{EI}}. \tag{2.30}$$

Here, $\mathcal{I} := [\wp_1,\wp_2,\ldots,\wp_{n_I}]$ includes all the indices of the spatial grid points where the interpolation is enforced and the set $\mathcal{E} := \{\mathbf{z}_{\wp_1},\ldots,\mathbf{z}_{\wp_{n_I}}\}$ denotes the corresponding spatial locations.

Other hyperreduction techniques such as Gappy-POD result in an approximation that minimizes the least-square error over the n_I points, but now with $n_I \geq n_{\mathrm{EI}}$. The minimization problem reads

$$\min \left\| f(g(\mathbf{z}_{\wp_j},t,\boldsymbol{\mu}),\boldsymbol{\mu}) - \mathscr{I}[f(g(\mathbf{z}_{\wp_j},t,\boldsymbol{\mu}),\boldsymbol{\mu})] \right\|, \qquad j = 1,2,\ldots,n_I \geq n_{\mathrm{EI}}. \tag{2.31}$$

Irrespective of the method, we shall, hereafter, refer to the set of points $\mathcal{E}$ as the EI interpolation points, or alternatively, as EI sampling points.

2.6.1. Discrete setting

In the context of MOR, we apply hyperreduction to the discrete system Eq. (2.3). Therefore, we discretize the parameter space in the form of a training set $\Xi_{\mathrm{EI}} \subset \mathcal{P}$ with n_s^{EI} samples. The discretized nonlinear function is given by

$$\mathbf{f}(\mathbf{x}(t,\boldsymbol{\mu}),\boldsymbol{\mu}) := [f_1(\mathbf{x}(t,\boldsymbol{\mu}),\boldsymbol{\mu}),\ldots,f_N(\mathbf{x}(t,\boldsymbol{\mu}),\boldsymbol{\mu})]^{\mathsf{T}} \in \mathbb{R}^N$$

where $\mathbf{x}(t,\boldsymbol{\mu}) \in \mathbb{R}^N$ is the discrete representation of $g(\mathbf{z},t,\boldsymbol{\mu})$ defined previously and $\mathbf{x}(t,\boldsymbol{\mu}) := [x_1(t,\boldsymbol{\mu}),x_2(t,\boldsymbol{\mu}),\ldots,x_N(t,\boldsymbol{\mu})]^{\mathsf{T}}$. Let $\mathbf{u}_i^{\mathrm{EI}} := [u_{i1}^{\mathrm{EI}},\ldots,u_{iN}^{\mathrm{EI}}]^{\mathsf{T}} \in \mathbb{R}^N$ be the i-th discretized EI basis function and $\mathbf{U} := [\mathbf{u}_1^{\mathrm{EI}},\ldots,\mathbf{u}_{n_{\mathrm{EI}}}^{\mathrm{EI}}] \in \mathbb{R}^{N\times n_{\mathrm{EI}}}$ the EI projection basis. In addition, we define the snapshot matrix $\mathbf{F}$ consisting of snapshots of the discretized function $\mathbf{f}(\mathbf{x}(t,\boldsymbol{\mu}),\boldsymbol{\mu})$ at different parameter/time instances. We denote $\mathbf{f}^k(\boldsymbol{\mu}_i) \in \mathbb{R}^N$

as the snapshot of the function at the k-th time instant for the i-th parameter sample in Ξ_{EI}, i.e., $\mathbf{f}^k(\boldsymbol{\mu}_i) = \mathbf{f}(\mathbf{x}(t^k, \boldsymbol{\mu}_i), \boldsymbol{\mu}_i)$, we define the nonlinear snapshot matrix as

$$\mathbf{F} := \left[\mathbf{f}^k(\boldsymbol{\mu}_i)\right] \in \mathbb{R}^{N \times N_t \cdot n_s^{\text{EI}}}, \qquad i = 1, 2, \ldots, n_s^{\text{EI}}, \qquad k = 0, 1, \ldots, K. \tag{2.32}$$

A discrete version of Eq. (2.30) reads,

$$\mathbf{f}_{\wp_i}(\mathbf{x}(t, \boldsymbol{\mu}), \boldsymbol{\mu}) = \sum_{i=1}^{n_{\text{EI}}} \xi_i(\boldsymbol{\mu}, t) u_{i\wp_i}^{\text{EI}}. \tag{2.33}$$

To use matrix-vector form, we introduce the matrix

$$\mathbf{S} = [\mathbf{e}_{\wp_1}, \ldots, \mathbf{e}_{\wp_{n_I}}] \in \mathbb{R}^{N \times n_I}$$

where $\mathbf{e}_{\wp_i} := [0, \ldots, 0, 1, 0, \ldots, 0]^\mathsf{T} \in \mathbb{R}^N$ is the $\wp_i$-th column of the identity matrix $\mathbf{I} \in \mathbb{R}^{N \times N}$. Thus, Equation (2.33) can be represented equivalently as

$$\mathbf{S}^\mathsf{T} \mathbf{f}(\mathbf{x}(t, \boldsymbol{\mu}), \boldsymbol{\mu}) = \mathbf{S}^\mathsf{T} \mathbf{U} \boldsymbol{\xi}$$

with $\boldsymbol{\xi} := [\xi_1(\boldsymbol{\mu}, t), \ldots, \xi_{n_{\text{EI}}}(\boldsymbol{\mu}, t)]^\mathsf{T} \in \mathbb{R}^{n_{\text{EI}}}$. For the remainder of this thesis, we use the shortened notations $\mathbf{U}^\jmath := \mathbf{S}^\mathsf{T}\mathbf{U} \in \mathbb{R}^{n_I \times n_{\text{EI}}}$ and $\mathbf{f}^\jmath := \mathbf{S}^\mathsf{T}\mathbf{f}(\mathbf{x}(t, \boldsymbol{\mu}), \boldsymbol{\mu}) \in \mathbb{R}^{n_I}$. If $n_I = n_{\text{EI}}$, it can be proved [52, 167] that $\mathbf{U}^\jmath$ is nonsingular, so that the coefficients are obtained by simple matrix inversion as $\boldsymbol{\xi} = \left(\mathbf{U}^\jmath\right)^{-1} \mathbf{f}^\jmath$.

The EIM- or DEIM-based hyperreduction approximation for any $\mathbf{f}(\mathbf{x}(t, \boldsymbol{\mu}), \boldsymbol{\mu}) \in \mathbb{R}^N$ with $\boldsymbol{\mu} \in \Xi_{\text{EI}}$ is

$$\mathscr{I}[\mathbf{f}(\mathbf{x}(t, \boldsymbol{\mu}), \boldsymbol{\mu})] = \mathbf{U}\boldsymbol{\xi} = \mathbf{U}\left(\mathbf{U}^\jmath\right)^{-1} \mathbf{f}^\jmath. \tag{2.34}$$

For the case where the number of interpolation points is larger than the dimension of the EI basis ($n_I > n_{\text{EI}}$) the discretized minimization problem is

$$\min \left\| \mathbf{f}^\jmath - \mathbf{U}^\jmath \boldsymbol{\xi} \right\|$$

where the sampled EI basis $\mathbf{U}^\jmath \in \mathbb{R}^{n_I \times n_{\text{EI}}}$ is rectangular. The coefficients $\boldsymbol{\xi}$ are evaluated using a Moore-Penrose pseudoinverse as $\boldsymbol{\xi} = \left(\mathbf{U}^\jmath\right)^\dagger \mathbf{f}^\jmath$. The Gappy-POD approximation for any $\mathbf{f}(\mathbf{x}(t, \boldsymbol{\mu}), \boldsymbol{\mu})$ with $\boldsymbol{\mu} \in \Xi_{\text{EI}}$ reads

$$\mathscr{I}[\mathbf{f}(\mathbf{x}(t, \boldsymbol{\mu}), \boldsymbol{\mu})] = \mathbf{U}\left(\mathbf{U}^\jmath\right)^\dagger \mathbf{f}^\jmath \tag{2.35}$$

Next, we detail the EIM, DEIM strategies for identifying $\mathbf{U}$ and $\jmath$, since this thesis mainly uses these approaches.

2.6.2. EIM

EIM is an interpolatory hyperreduction method introduced in [19], which constructs the EI basis iteratively, through a greedy algorithm.

With a slight abuse of notation, let us denote by $\mathbf{F}$ also the set containing all the snapshots in the snapshot matrix Eq. (2.32), i.e., the columns of the matrix $\mathbf{F}$. The EIM begins by identifying the snapshot with the largest norm ($\|\cdot\|_2$ or $\|\cdot\|_\infty$)

$$\boldsymbol{\tau} := \arg\max_{\mathbf{f} \in \mathbf{F}} \|\mathbf{f}\|$$

Algorithm 2.5: `EIM`
Computes the EI basis and interpolation points using the EIM.

Input: $\mathbf{F}$: a set of all snapshots $\mathbf{f}^k(\boldsymbol{\mu}_j),\ j = 1, 2, \ldots, n_s^{\text{EI}},\ k = 0, 1, \ldots, K$, EI tolerance ϵ_{EI}, Maximum iterations $iter\_max$.
Output: EI basis $\mathbf{U}$ and interpolation indices $\mathcal{I}$.

Initialization: $\mathbf{U} = [\,]$, $\mathcal{I} = \varnothing$, $err\_max = 1 + \epsilon_{\text{EI}}$, $iter = 1$.
Set $\boldsymbol{\tau} = \arg\max_{\mathbf{f}\in\mathbf{F}} \|\mathbf{f}\|$ and $\wp_1 = \arg\max_{i\in\{1,2,\ldots,N\}} |\tau_i|$. /* $\boldsymbol{\tau} := [\tau_1, \ldots, \tau_N]^{\mathsf{T}}$ */
Compute the first EI basis vector $\mathbf{u}_1^{\text{EI}} := \frac{\boldsymbol{\tau}}{\tau_{\wp_1}}$.
Set $\mathbf{U} = \mathbf{u}_1^{\text{EI}}$, $\mathcal{I} := \mathcal{I} \cup \wp_1$.
while $err\_max > \epsilon_{\text{EI}}$ and $iter \leq iter\_max$ **do**
  $iter = iter + 1$.
  Form the interpolants $\mathscr{I}[\mathbf{f}] := \mathbf{U}\big(\mathbf{U}^{\mathcal{I}}\big)^{-1}(\mathbf{f})^{\mathcal{I}}$ for all $\mathbf{f} \in \mathbf{F}$.
  Compute the snapshot $\mathbf{f}^* := \arg\max_{\mathbf{f}\in\mathbf{F}} \|\mathbf{f} - \mathscr{I}[\mathbf{f}]\|$.
  Determine the error $\boldsymbol{\tau} := \mathbf{f}^* - \mathscr{I}[\mathbf{f}^*]$, let $err\_max = \|\boldsymbol{\tau}\|$.
  if $err\_max > \epsilon_{\text{EI}}$ **then**
    Find the next EI index $\wp_{iter} = \arg\max_{j\in\{1,2,\ldots,N\}} |\tau_j|$.
    Set $\mathbf{u}_{iter}^{\text{EI}} := \frac{\boldsymbol{\tau}}{\tau_{\wp_{iter}}}$.
    Enrich $\mathbf{U} := \big[\mathbf{U}\ ,\ \mathbf{u}_{iter}^{\text{EI}}\big]$ and $\mathcal{I} := \mathcal{I}\ \cup\ \wp_{iter}$.
  else
    terminate
  end
end

with $\boldsymbol{\tau} := [\tau_1, \ldots, \tau_N]^{\mathsf{T}} \in \mathbb{R}^N$. EIM then proceeds to find the index of the element with the largest absolute value in $\boldsymbol{\tau}$ and normalizes the vector w.r.t this value. We perform

$$\wp_1 = \arg\max_{j\in\{1,2,\ldots,N\}} |\tau_j|$$

Using this index, the first EI basis vector is determined as:

$$\mathbf{u}_1^{\text{EI}} := \frac{\boldsymbol{\tau}}{\tau_{\wp_1}}, \qquad \mathbf{U} = \mathbf{u}_1^{\text{EI}}.$$

Assuming at the end of the $(iter-1)$-th iteration we have the basis $\mathbf{U} = [\mathbf{u}_1^{\text{EI}}, \mathbf{u}_2^{\text{EI}}, \ldots, \mathbf{u}_{iter-1}^{\text{EI}}]$ and the set of indices $\mathcal{I} := [\wp_1, \wp_2, \ldots, \wp_{iter-1}]$, the $(iter-1)$-th EIM interpolant for each $\mathbf{f} \in \mathbf{F}$ is given by

$$\mathscr{I}[\mathbf{f}] := \mathbf{U}\big(\mathbf{U}^{\mathcal{I}}\big)^{-1}(\mathbf{f})^{\mathcal{I}}.$$

Based on this, we compute $\mathbf{f}^*$ at the $iter$-th iteration as

$$\mathbf{f}^* := \arg\max_{\mathbf{f}\in\mathbf{F}} \|\mathbf{f} - \mathscr{I}[\mathbf{f}]\|.$$

and compute the error $\boldsymbol{\tau} := \mathbf{f}^* - \mathscr{I}[\mathbf{f}^*]$. In the above scheme, $\mathbf{f}^*$ is the snapshot in $\mathbf{F}$ at which the corresponding $(iter-1)$-th EIM interpolant has the worst error. We then

check if the norm of the resulting error vector $\boldsymbol{\tau}$ is below a user-defined tolerance ϵ_{EI}, i.e., $\|\boldsymbol{\tau}\| < \epsilon_{\text{EI}}$. If this condition is true then the algorithm is terminated; if not, the *iter*-th index location is chosen and the corresponding EI basis vector is computed as below:

$$\wp_i = \arg \max_{j \in \{1,2,\ldots,N\}} |\tau_j|, \qquad \mathbf{u}_i^{\text{EI}} := \frac{\boldsymbol{\tau}}{\tau_{\wp_i}}$$

and the iteration continued. The EIM algorithm [167, 212] is sketched in Algorithm 2.5. Due to the construction, the matrix $\mathbf{U}^{\mathcal{I}}$ in Step 7 of the algorithm is lower-triangular [167].

EIM was originally introduced to efficiently treat non-affine functions in the context of the RBM. However, later it was extended to ensure efficient offline-online evaluation of nonlinear problems. *A priori* error bounds for the EIM approximation have been studied [19, 97]. When considering the $\|\cdot\|_\infty$ norm, the analysis involves the Lebesgue constant. For purpose of MOR, *a posteriori* error estimates/bounds are of greater interest and have been discussed in works such as [74]. We discuss a posteriori error estimates for EIM in Chapter 4.

2.6.3. DEIM

DEIM was introduced in [52] and is closely related to the EIM. Similar to EIM, DEIM uses a greedy scheme to determine the EI sampling indices in $\mathcal{I}$. However, the EI basis is computed differently. In fact, the EI basis used in the DEIM is determined by applying a POD to the nonlinear snapshot matrix $\mathbf{F}$:

$$[\mathbf{u}_1^{\text{EI}}, \ldots, \mathbf{u}_{n_{\text{EI}}}^{\text{EI}}] = \texttt{POD}(\mathbf{F}, n_{\text{EI}}).$$

Constructed this way, the EI basis is essentially the first n_{EI} left singular vectors of the matrix $\mathbf{F}$. Following this, the interpolation indices are chosen using a greedy algorithm. A pseudocode of the procedure can be found in [52] and is presented here in Algorithm 2.6.

A priori error bounds based on the neglected singular values or best 2-norm approximation exist for the DEIM [52]. A posteriori error estimates using the offline-online paradigm were proposed in [207]. We postpone an in-depth discussion to Chapter 4.

The combination of POD with DEIM has been used successfully in a number of applications. In contrast, only very few works have used EIM with POD for model reduction [173]. This is unsurprising since DEIM was initially proposed in a discrete setting, which can be straightforwardly applied. However, EIM was originally presented only in the continuous form Eq. (2.30) [19]. For the RBM, both DEIM and EIM have been used as hyperreduction strategies.

2.6.4. Hyperreduced ROMs from POD and RBM

As noted before, for nonlinear and/or non-affine systems, the full computational benefits of the ROM generated using POD or RBM can be enjoyed only if a suitable hyperreduction strategy is used. On the one hand, the hyperreduced, semi-discretized ROM will be finally simulated using numerical time integration, resulting in a fully-discretized form. On the other hand, our proposed *a posteriori* error estimator is based on the

Algorithm 2.6: DEIM
Computes the EI basis and interpolation points using the DEIM.

Input: Nonlinear snapshot matrix $\mathbf{F}$, EI tolerance ϵ_{EI} or DEIM rank n_{EI}.
Output: EI basis $\mathbf{U}$ and interpolation indices $\mathcal{I}$.

Initialization: $\mathbf{U} = [\,]$, $\mathcal{I} = \varnothing$.
Find the POD basis: $[\mathbf{u}_1^{\text{EI}}, \ldots, \mathbf{u}_{n_{\text{EI}}}^{\text{EI}}] = \texttt{POD}(\mathbf{F}, n_{\text{EI}}(\text{ or } \epsilon_{\text{EI}}))$.
Set $\wp_1 = \arg\max_{i \in \{1,2,\ldots,N\}} |u_{1,i}^{\text{EI}}|$; $\mathbf{u}_1^{\text{EI}} := [u_{1,1}^{\text{EI}}, \ldots, u_{1,N}^{\text{EI}}]^{\mathsf{T}} \in \mathbb{R}^N$.
$\mathbf{U} = \mathbf{u}_1^{\text{EI}}$; $\mathcal{I} := \mathcal{I} \cup \wp_1$.
for $j = 2$ to n_{EI} **do**
  Solve $\mathbf{U}^{\mathcal{I}}\boldsymbol{\xi} = (\mathbf{u}_j^{\text{EI}})^{\mathcal{I}}$ to obtain $\boldsymbol{\xi}$, where $(\mathbf{u}_j^{\text{EI}})^{\mathcal{I}} := \mathbf{S}^{\mathsf{T}}\mathbf{u}_j^{\text{EI}}$.
  Determine the error $\boldsymbol{\tau} := \mathbf{u}_j^{\text{EI}} - \mathbf{U}\boldsymbol{\xi}$.
  Compute the next EI index $\wp_j = \arg\max_{i \in \{1,2,\ldots,N\}} |\tau_i|$.
  Enrich $\mathbf{U} := \left[\mathbf{U}\ ,\ \mathbf{u}_j^{\text{EI}}\right]$ and $\mathcal{I} := \mathcal{I}\ \cup\ \wp_j$.
end

fully discrete Eq. (2.11) and its ROM. Thus, in the following, we discuss hyperreduced ROMs in the fully discrete form.

For Eq. (2.18), efficient evaluation of $\widehat{\overline{\mathbf{f}}}(\widehat{\mathbf{x}}^k, \boldsymbol{\mu}) := \mathbf{W}^{\mathsf{T}}\overline{\mathbf{f}}(\mathbf{V}\widehat{\mathbf{x}}^k, \boldsymbol{\mu})$ at each time step is required. Without any special treatment, the above evaluation involves, for a given $\boldsymbol{\mu}$ and at each time step:

(i) determining $\mathbf{V}\widehat{\mathbf{x}}^k \rightarrow$ with $O(N)$ cost,

(ii) evaluating $\overline{\mathbf{f}}(\mathbf{V}\widehat{\mathbf{x}}^k) \rightarrow$ with $O(N)$ cost and,

(iii) projecting $\overline{\mathbf{f}}(\mathbf{V}\widehat{\mathbf{x}}^k)$ onto the span of $\mathbf{W} \rightarrow$ again with $O(N)$ costs.

Clearly, this is not tractable in the online scenario. Using EIM or DEIM, we get

$$\widehat{\overline{\mathbf{f}}}(\widehat{\mathbf{x}}^k, \boldsymbol{\mu}) \approx \mathbf{W}^{\mathsf{T}}\mathscr{I}[\overline{\mathbf{f}}(\mathbf{V}\widehat{\mathbf{x}}^k, \boldsymbol{\mu})] = \underbrace{\mathbf{W}^{\mathsf{T}}\mathbf{U}(\mathbf{U}^{\mathcal{I}})^{-1}}_{\text{precomputed}} \overbrace{(\overline{\mathbf{f}}(\mathbf{V}\widehat{\mathbf{x}}^k, \boldsymbol{\mu}))^{\mathcal{I}}}^{\text{evaluated online}}.$$

In the above expression, we have $(\overline{\mathbf{f}}(\mathbf{V}\widehat{\mathbf{x}}^k, \boldsymbol{\mu}))^{\mathcal{I}} := \mathbf{S}^{\mathsf{T}}(\overline{\mathbf{f}}(\mathbf{V}\widehat{\mathbf{x}}^k, \boldsymbol{\mu}))$. The quantity $\boldsymbol{\mathcal{U}} := \mathbf{W}^{\mathsf{T}}\mathbf{U}(\mathbf{U}^{\mathcal{I}})^{-1} \in \mathbb{R}^{n \times n_{\text{EI}}}$ can be precomputed once and for all and stored offline, whereas $(\overline{\mathbf{f}}(\mathbf{V}\widehat{\mathbf{x}}^k, \boldsymbol{\mu}))^{\mathcal{I}} \in \mathbb{R}^{n_{\text{EI}}}$ can be cheaply computed online since $n_{\text{EI}} \ll N$. The hyperreduced ROM corresponding to Eq. (2.11) is

$$\begin{aligned} \widehat{\overline{\mathbf{E}}}(\boldsymbol{\mu})\widehat{\mathbf{x}}^{k+1} &= \widehat{\overline{\mathbf{A}}}(\boldsymbol{\mu})\widehat{\mathbf{x}}^k + \boldsymbol{\mathcal{U}}(\overline{\mathbf{f}}(\mathbf{V}\widehat{\mathbf{x}}^k, \boldsymbol{\mu}))^{\mathcal{I}} + \widehat{\overline{\mathbf{B}}}(\boldsymbol{\mu})\mathbf{u}(t^k), \\ \widetilde{\mathbf{y}}^{k+1}(\boldsymbol{\mu}) &= \widehat{\mathbf{C}}(\boldsymbol{\mu})\widehat{\mathbf{x}}^{k+1}. \end{aligned} \tag{2.36}$$

For Galerkin projection, we have $\mathbf{W} = \mathbf{V}$.

We sketch in Algorithms 2.7 and 2.8, the PODEI and RBMEI algorithms, respectively, where either the EIM or DEIM is employed to get the hyperreduced ROM, e.g., Eq. (2.36). In this thesis, PODEI will be applied to non-parametric systems while for parametric systems we use the RBMEI algorithm.

Algorithm 2.7: PODEI
Computes a ROM for (parametric) nonlinear dynamical systems using POD in combination with EIM or DEIM.

Input: System matrices and nonlinear term: $\mathbf{E}, \mathbf{A}, \mathbf{B}, \mathbf{C}$ and $\mathbf{f}$, tolerance ϵ_{POD} and ϵ_{EI} (or POD basis and EI basis dimension n, n_{EI}).

Output: Reduced system matrices $\widehat{\mathbf{E}}, \widehat{\mathbf{A}}, \widehat{\mathbf{B}}, \widehat{\mathbf{C}}$ (or $\widehat{\widehat{\mathbf{E}}}, \widehat{\widehat{\mathbf{A}}}, \widehat{\widehat{\mathbf{B}}}, \widehat{\widehat{\mathbf{C}}}$) and EI quantities $\boldsymbol{\mathcal{U}}, \mathcal{I}$.

1 Simulate the FOM, e.g., Eq. (2.11) to obtain state snapshots $\mathbf{X} = [\mathbf{x}^0, \ldots, \mathbf{x}^K]$ and corresponding nonlinear term snapshots $\mathbf{F} := [\overline{\mathbf{f}}^0, \ldots, \overline{\mathbf{f}}^K]$.
2 Compute POD basis $\mathbf{V} = \texttt{POD}(\mathbf{X}, \epsilon_{\text{POD}}(\text{or } n))$. `/* Algorithm 2.2 */`
3 Compute EI quantities $[\mathbf{U}, \mathcal{I}] = \texttt{EIM}(\mathbf{F}, \epsilon_{\text{EI}}, iter\_max)$ or $[\mathbf{U}, \mathcal{I}] = \texttt{DEIM}(\mathbf{F}, \epsilon_{\text{EI}}(\text{or } n_{\text{EI}}))$. `/* Algorithm 2.5 or Algorithm 2.6 */`
4 Determine reduced matrices $\widehat{\mathbf{E}}, \widehat{\mathbf{A}}, \widehat{\mathbf{B}}, \widehat{\mathbf{C}}$ (or $\widehat{\widehat{\mathbf{E}}}, \widehat{\widehat{\mathbf{A}}}, \widehat{\widehat{\mathbf{B}}}, \widehat{\widehat{\mathbf{C}}}$) through Galerkin projection using $\mathbf{V}$ as in Eq. (2.14). `/* assume parameter affine representation */`
5 Determine $\boldsymbol{\mathcal{U}} := \mathbf{V}^{\mathsf{T}}\mathbf{U}\mathbf{U}^{\mathcal{I}}$.

2.7. Radial Basis Functions

In this section, we review radial basis functions (RBFs) and the main ideas behind performing scattered-data interpolation using RBFs. In this thesis, RBFs are employed in two scenarios. Recall from the discussion in Section 2.5.2 that the error estimator used in the greedy algorithms (such as Algorithms 2.3 and 2.4), viz., $\Delta(\boldsymbol{\mu})$ should be cheap to evaluate. To address this, we derive a cheap data-driven surrogate of the error estimator based on RBF (see, for example, Chapter 3). Furthermore, we also use RBFs to efficiently interpolate the *inf-sup* constant, that appears in the output error estimator proposed in Chapter 4.

Suppose $\boldsymbol{\Lambda} := \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_\ell\} \subseteq \mathbb{R}^{N_p}$ be a set of pairwise distinct, non-collinear data locations. Let $D : \{d_1, d_2, \ldots, d_\ell\} \subseteq \mathbb{R}$ be the values of the data, corresponding to the locations in the set $\boldsymbol{\Lambda}$. The data may arise from some multivariate function $\mathfrak{g} : \mathbb{R}^{N_p} \to \mathbb{R}$. In practice, there may be cases for which the exact representation of this function is unknown and only the input-output data $(D, \boldsymbol{\Lambda})$ is known. Or, there may also be cases for which the exact form of $\mathfrak{g}$ is known but its evaluation is expensive or involves solving a high-dimensional PDE. In either case, it may be needed to generate an approximation $\chi : \mathbb{R}^{N_p} \to \mathbb{R}$ which interpolates the function $\mathfrak{g}$ at a set of locations $\boldsymbol{\Lambda}$ and whose evaluation is computationally inexpensive for any new value $\boldsymbol{\mu} \notin \boldsymbol{\Lambda}$ with $\|\mathfrak{g}(\boldsymbol{\mu}) - \chi(\boldsymbol{\mu})\| \leq tol$. Here, $tol \in (0, 1)$ is a small user-defined tolerance for the quality of approximation. This is a classical approximation problem known as *scattered-data interpolation* [204].

Standard approaches such as polynomial interpolation are unsuited to interpolate data in high-dimensional spaces due to several reasons [43, 80]. RBFs are an instance of kernel methods and were initially used in neural networks. Later, they became popular as a mesh-free method for data interpolation. We give a brief description of the RBF methodology to interpolate scattered data.

Algorithm 2.8: `RBMEI`
Computes a ROM for parametric dynamical systems using POD-Greedy in combination with EIM or DEIM.

Input: System matrices and nonlinear term: $\mathbf{E}, \mathbf{A}, \mathbf{B}, \mathbf{C}$ and $\mathbf{f}$, Training set Ξ, RB basis dimension r_{RB}, EI tolerance ϵ_{EI} or EI basis dimension n_{EI}, ROM tolerance ϵ, Deflation tolerance ϵ_{def}, Maximum iterations *iter_max*.

Output: Reduced system matrices $\widehat{\mathbf{E}}, \widehat{\mathbf{A}}, \widehat{\mathbf{B}}, \widehat{\mathbf{C}}$ (or $\widehat{\widehat{\mathbf{E}}}, \widehat{\widehat{\mathbf{A}}}, \widehat{\widehat{\mathbf{B}}}, \widehat{\widehat{\mathbf{C}}}$) and EI quantities $\boldsymbol{\mathcal{U}}, \mathcal{I}$.

1 Simulate the FOM, e.g., Eq. (2.11) for all $\boldsymbol{\mu} \in \Xi$ and obtain nonlinear term snapshots $\mathbf{F} \in \mathbb{R}^{N \times N_t \cdot n_s^{\text{EI}}}$ as in Eq. (2.32).
2 Obtain EI quantities $[\mathbf{U}, \mathcal{I}] = \texttt{EIM}(\mathbf{F}, \epsilon_{\text{EI}}, \mathit{iter\_max})$ or $[\mathbf{U}, \mathcal{I}] = \texttt{DEIM}(\mathbf{F}, \epsilon_{\text{EI}} (\text{or } n_{\text{EI}}))$. `/* Algorithm 2.5 or Algorithm 2.6 */`
3 Initialization: $\mathbf{V} = [\,]$, $\mathit{err\_max} = 1 + \epsilon$, $\mathit{iter} = 1$, $\bar{\boldsymbol{\mu}}^*$ (chosen randomly from Ξ).
4 **while** $\mathit{err\_max} > \epsilon$ and $\mathit{iter} \leq \mathit{iter\_max}$ **do**
5 Solve FOM Eq. (2.11) at $\bar{\boldsymbol{\mu}}^*$; obtain snapshots matrix $\mathbf{X}(\bar{\boldsymbol{\mu}}^*)$.
6 **if** $\mathit{iter} == 1$ **then**
7 $\mathbf{V} = \texttt{POD}(\mathbf{X}, r_{\text{RB}})$.
8 **else**
9 Compute $\overline{\mathbf{X}} := \mathbf{X} - \mathbf{V}\mathbf{V}^{\mathsf{T}}\mathbf{X}$.
10 $\mathbf{V}_{\text{POD}} := \texttt{POD}(\overline{\mathbf{X}}, r_{\text{RB}})$.
11 $\mathbf{V} = \texttt{orth\_def\_mat}(\mathbf{V}, \mathbf{V}_{\text{POD}}, \epsilon_{\text{def}})$. `/* see Step 15 of Algorithm 2.4 */`
12 **end**
13 $\mathit{iter} = \mathit{iter} + 1$.
14 Determine reduced matrices $\widehat{\mathbf{E}}, \widehat{\mathbf{A}}, \widehat{\mathbf{B}}, \widehat{\mathbf{C}}$ (or $\widehat{\widehat{\mathbf{E}}}, \widehat{\widehat{\mathbf{A}}}, \widehat{\widehat{\mathbf{B}}}, \widehat{\widehat{\mathbf{C}}}$) through Galerkin projection using $\mathbf{V}$ as in Eq. (2.14).
15 Solve the ROM (e.g., Eq. (2.36)) for all $\boldsymbol{\mu} \in \Xi$; find $\bar{\boldsymbol{\mu}}^* = \arg\max_{\boldsymbol{\mu} \in \Xi} \Delta(\boldsymbol{\mu})$.
16 Set $\mathit{err\_max} = \Delta(\bar{\boldsymbol{\mu}}^*)$.
17 **end**

Definition 2.18 ([80]):
Let $\boldsymbol{\mu} \in \mathbb{R}^{N_p}$ and let $\|\cdot\|$ be the Euclidean norm on $\mathbb{R}^{N_p}$. The function $\overline{\Phi}(\cdot)$ is a radial function if there exists a function $\Phi : [0, \infty) \to \mathbb{R}$ such that

$$\overline{\Phi}(\boldsymbol{\mu}) = \Phi(\|\boldsymbol{\mu}\|). \qquad \lozenge$$

The RBF interpolant $\chi : \mathbb{R}^{N_p} \to \mathbb{R}$ is given by

$$\chi(\boldsymbol{\mu}) := \sum_{i=1}^{\ell} \widetilde{w}_i \Phi(\|\boldsymbol{\mu} - \boldsymbol{\mu}_i\|). \tag{2.37}$$

The name RBF derives from the fact that radial functions $\Phi(\|\cdot\|)$ are used as the expansion basis to approximate the target function. The locations $\{\boldsymbol{\mu}_i\}_{i=1}^{\ell}$ are often called *centers*, $\{\widetilde{w}_i\}_{i=1}^{\ell}$ are the *weights* corresponding to the centers and $\widetilde{w}_i \in \mathbb{R}$. The

weights are determined by imposing the interpolation conditions

$$\chi(\boldsymbol{\mu}_i) = d_i, \qquad i = 1, 2, \ldots, \ell. \tag{2.38}$$

As a result, the following linear system of equations needs to be solved:

$$\underbrace{\begin{bmatrix} \Phi(\|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_1\|) & \Phi(\|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\|) & \cdots & \Phi(\|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_\ell\|) \\ \Phi(\|\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1\|) & \Phi(\|\boldsymbol{\mu}_2 - \boldsymbol{\mu}_2\|) & \cdots & \Phi(\|\boldsymbol{\mu}_2 - \boldsymbol{\mu}_\ell\|) \\ \vdots & \vdots & \ddots & \vdots \\ \Phi(\|\boldsymbol{\mu}_\ell - \boldsymbol{\mu}_1\|) & \Phi(\|\boldsymbol{\mu}_\ell - \boldsymbol{\mu}_2\|) & \cdots & \Phi(\|\boldsymbol{\mu}_\ell - \boldsymbol{\mu}_\ell\|) \end{bmatrix}}_{\mathbf{G}} \underbrace{\begin{bmatrix} \widetilde{w}_1 \\ \widetilde{w}_2 \\ \vdots \\ \widetilde{w}_\ell \end{bmatrix}}_{\widetilde{\mathbf{w}}} = \underbrace{\begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_\ell \end{bmatrix}}_{\mathbf{d}}. \tag{2.39}$$

The unique solution of the above system exists only if $\mathbf{G}$ is invertible. In RBF literature, this is achieved by ensuring that the matrix $\mathbf{G}$ is positive definite. The matrix $\mathbf{G}$ being positive definite is a sufficient condition for the unique solution of Eq. (2.39). To get a positive definite matrix, the radial basis function $\Phi(\|\cdot\|)$ should be chosen as a positive definite function.

Definition 2.19 (Positive definite function [80]):
A real-valued continuous function $\Phi : \mathbb{R}^{N_p} \to \mathbb{R}$ is positive definite on $\mathbb{R}^{N_p}$ if and only if it is *even* and

$$\sum_{j=1}^{\ell} \sum_{k=1}^{\ell} c_j c_k \Phi(\|\boldsymbol{\mu}_j - \boldsymbol{\mu}_k\|) \geq \mathbf{0}$$

for any pairwise different points $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_\ell \in \mathbb{R}^{N_p}$ and $\mathbf{c} = [c_1, \ldots, c_\ell]^\mathsf{T} \in \mathbb{R}^\ell$. Moreover, Φ is strictly positive definite on $\mathbb{R}^{N_p}$ if the above quadratic form is zero only for $\mathbf{c} \equiv \mathbf{0}$. ♢

A matrix $\mathbf{G}$ whose elements are defined as evaluations of a (strictly) positive definite function, i.e., $\mathbf{G}_{ij} := \Phi(\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|)$ is a (strictly) positive definite matrix. There exists a vast body of literature on positive definite radial functions. We do not offer an in-depth discussion here and refer to the books [43, 80, 204]. The radial functions that are positive definite include *Gaussian functions*, *Matérn functions*, *Generalized inverse multiquadrics*, etc.

It is worth noting that the requirement of strict positive definiteness on the functions $\Phi(\|\cdot\|)$ is rather restrictive and precludes the use of many functions that are radial, but not positive definite. A standard workaround adopted in the RBF literature is to consider the more relaxed notion of conditional positive definiteness which involves some additional constraints on the weights $\{\widetilde{w}_i\}_{i=1}^{\ell}$.

Definition 2.20 (Conditionally positive definite matrices [80]):
A real, symmetric matrix $\mathbf{G} \in \mathbb{R}^{\ell \times \ell}$ is called conditionally positive definite of order $d_0 + 1$ if its associated quadratic form satisfies $\widetilde{\mathbf{w}}^\mathsf{T} \mathbf{G} \widetilde{\mathbf{w}} > 0$ and the weights satisfy the additional constraint

$$\sum_{i=1}^{\ell} \widetilde{w}_i \pi(\boldsymbol{\mu}_i) = 0$$

where $\pi : \mathbb{R}^{N_p} \to \mathbb{R}$ is a polynomial function in p variables with degree at most d_0. ♢

To this end, Eq. (2.37) is augmented with the polynomial term

$$\chi(\boldsymbol{\mu}) := \sum_{i=1}^{\ell} \widetilde{w}_i \Phi(\|\boldsymbol{\mu} - \boldsymbol{\mu}_i\|) + \pi(\boldsymbol{\mu}), \tag{2.40}$$

where

$$\pi : \sum_{j=1}^{\nu} \lambda_j \psi_j(\boldsymbol{\mu}). \tag{2.41}$$

Here, the functions $\{\psi_j\}_{j=1}^{\nu}$ form a basis for $\Pi_{d_o}^p$ the space of polynomials in p variables with degree less than or equal to d_o and $\nu = \binom{d_0 + p}{d_0}$. Typically, a constant, linear or quadratic polynomial in p variables is added ($d_0 \in \{0, 1, 2\}$). The following additional weight constraint

$$\sum_{i=1}^{\ell} \widetilde{w}_i \psi_j(\boldsymbol{\mu}_i) = 0, \qquad j = 1, 2, \ldots, \nu. \tag{2.42}$$

is imposed in addition to the interpolation constraint Eq. (2.38), yielding the saddle-point system

$$\begin{bmatrix} \mathbf{G} & \mathbf{P} \\ \mathbf{P}^T & \mathbf{0}_{\nu\times\nu} \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{w}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{d} \\ \mathbf{0}_{\nu\times 1} \end{bmatrix} \tag{2.43}$$

with $\mathbf{P} := [\psi_1, \ldots, \psi_\nu] \in \mathbb{R}^{\ell\times\nu}$, $\mathbf{0}_{\nu\times\nu} \in \mathbb{R}^{\nu\times\nu}$ a zero matrix and $\mathbf{0}_{\nu\times 1} \in \mathbb{R}^{\nu}$ a zero vector. There exist many examples of radial functions that yield conditionally positive definite matrices including *Generalized multiquadrics*, *Thin-plate splines*, etc.

2.8. Conclusion

In this chapter, we have presented a broad mathematical background of the various state-of-the-art methodologies upon which this thesis is built. We introduced the general setting of the linear, nonlinear; steady-state, time-dependent systems of interest and reviewed MOR techniques used to obtain ROMs for each class. In the case of linear systems expressed in the frequency-domain, we paid special attention to the MMM method. For nonlinear systems, our main focus for MOR was the RBM. We also gave a short overview of radial basis functions that will be used in later chapters to perform interpolation of scattered data in high dimension.

In the following chapters, we propose new error estimation approaches and techniques for adaptive model reduction for linear and nonlinear systems.

CHAPTER 3

ERROR ESTIMATION AND ADAPTIVITY FOR LINEAR STEADY AND DYNAMICAL SYSTEMS

Contents

3.1. Introduction

In this chapter, we study *a posteriori* error estimation and propose adaptive algorithms to efficiently generate ROMs for two types of systems. The first type we consider is *linear* parametric systems, arising from discretization of PDEs (such as Eqs. (2.1) and (2.2)). For these systems, we introduce an efficient *inf-sup-constant-free* state error estimator and incorporate it within an adaptive algorithm to identify a good projection matrix to obtain the ROM. We illustrate the robustness of our state error estimator by comparing it with existing state-of-the-art error estimators and show the efficiency of the adaptive scheme on two real-life applications arising in the field of electromagnetics. Following this, we study the second type of systems, viz., input-output systems in the transfer function representation, see Eq. (2.7). For these systems, we discuss *a posteriori* output error estimation. For systems having many parameters or for systems with parameters that vary over a wide range, the computation of output error estimators poses a challenge due to the complexity of the training set. To enable a cheap computation of the error estimator, we develop a fast RBF-based surrogate model. We incorporate the surrogate error model within an adaptive scheme which (i) iteratively enriches the training set and (ii) adaptively constructs a projection matrix to achieve a robust ROM. Numerical tests on three benchmark examples are used to validate the proposed approach.

We begin this chapter by giving an overview of standard *a posteriori* error estimation techniques and existing works in Section 3.2. Following this, in Section 3.3 we discuss existing state error estimation techniques (Section 3.3.1), introduce the proposed *inf-sup-constant-free* error estimator (Section 3.3.2) and discuss its computational aspects (Section 3.3.3). We also compare our proposed method to the state-of-the-art techniques (Section 3.3.4). An adaptive algorithm for ROM generation using the *inf-sup-constant-free* estimator is detailed in Section 3.3.5. In Section 3.3.6, we validate the adaptive algorithm on several real-life electromagnetic device models. We then turn our focus to *a posteriori* output error estimation in Section 3.4. Existing dual-based and *inf-sup-constant-free* error estimation approaches are discussed in Sections 3.4.1 and 3.4.2. We then point out some drawbacks that arise during adaptive ROM construction based on the existing output error estimators in Section 3.4.4. To remedy this, we propose an adaptive ROM construction procedure that exploits a surrogate error estimator in Section 3.4.5 and analyse its computational advantages. We then demonstrate the benefits of the strategy on benchmark numerical models in Section 3.4.6.

Most of the topics discussed in this chapter are based on the results published in [56, 57].

3.2. A Posteriori Error Estimation

Over the years, MOR has demonstrated its robustness in reducing the complexity of parametric systems [11, 35, 180]. Besides the fact that a ROM offers considerable speedup of the computation, it is vital to certify the accuracy of the reduced models.

In order to certify the accuracy of the ROM (Eqs. (2.13), (2.15) and (2.16)) obtained using any MOR algorithm, the error of the approximation ($\|\mathbf{x}(\breve{\boldsymbol{\mu}}) - \widehat{\mathbf{x}}(\breve{\boldsymbol{\mu}})\|$,

$\|\mathbf{y}(\breve{\boldsymbol{\mu}}) - \widehat{\mathbf{y}}(\breve{\boldsymbol{\mu}})\|$) or ($\|\mathbf{H}(\breve{\boldsymbol{\mu}}) - \widehat{\mathbf{H}}(\breve{\boldsymbol{\mu}})\|$) needs to be estimated. These quantities refer to the *true errors*. Obviously, computing the true errors is not an option since it involves knowing the true solutions ($\mathbf{x}(\breve{\boldsymbol{\mu}}), \mathbf{y}(\breve{\boldsymbol{\mu}})$ or $\mathbf{H}(\breve{\boldsymbol{\mu}})$) at any parameter. In practice, an upper bound or a tight estimate of the true error is preferred. There exist *a priori* error bounds for some MOR methods, e.g., BT; but, such bounds are not very helpful in quantifying the accuracy of the generated ROM. Therefore, we desire *a posteriori* error bounds or estimates of the state error, output error, or transfer function error, respectively, i.e.,

$$\|\mathbf{x}(\breve{\boldsymbol{\mu}}) - \widehat{\mathbf{x}}(\breve{\boldsymbol{\mu}})\| \leq \Delta_{\mathbf{x}}(\breve{\boldsymbol{\mu}}), \tag{3.1a}$$

$$\|\mathbf{y}(\breve{\boldsymbol{\mu}}) - \widehat{\mathbf{y}}(\breve{\boldsymbol{\mu}})\| \leq \Delta_{\mathbf{y}}(\breve{\boldsymbol{\mu}}), \quad \text{or} \tag{3.1b}$$

$$\|\mathbf{H}(\breve{\boldsymbol{\mu}}) - \widehat{\mathbf{H}}(\breve{\boldsymbol{\mu}})\| \leq \Delta_{\mathbf{H}}(\breve{\boldsymbol{\mu}}). \tag{3.1c}$$

For both frequency- and time-domain MOR methods, *a posteriori* error estimation is an actively researched topic [81, 85, 98, 167, 191, 214]. In this chapter, we study *a posteriori* error estimators in Eq. (3.1), which can be applied to any projection-based MOR method.

A posteriori error bounds or indicators were initially proposed for non-parametric systems both for Gramian-based approaches [50] and Krylov-space approaches [23, 38, 101, 116]. The extension of frequency-domain MOR methods such as BT, MM to parametric systems [20, 64, 82] triggered the development of *a posteriori* error estimators for parametric systems [81, 83, 86]. These error estimators were inspired by similar error estimators for steady systems proposed in the RBM community. The most commonly considered error estimators are based on the residual, including the randomized approach [191]. Recent efforts towards more effective error estimation have exploited the use of adjoint or dual systems [85, 185].

In the following sections, we discuss *a posteriori* error estimation in detail. To ensure a clean notation, we do not explicitly show the parameter dependence of the matrices and vectors in the subsequent discussion. We emphasize that, unless stated otherwise, all systems considered in the discussion below are parametric.

3.3. A Posteriori State Error Estimation

In situations where the quantity of interest is the entire state variable, *a posteriori* state error estimators in Eq. (3.1a) are essential. This is often the case in applications such as fast frequency sweeps in electromagnetics and acoustics. Due to its importance in applications, a number of works have considered *a posteriori* state error estimation. The most common approach is simply using the norm of the residual as a heuristic error estimator [110, 116, 155, 193]. While computationally cheap, this approach often leads to a poor estimation of the error. Another widely used method is to use the residual norm divided by the *inf-sup* constant. This is denoted as the standard *a posteriori* error estimator. While this approach is rigorous, it is much more expensive as the computation of the *inf-sup* constant involves the solution of a large-scale eigenvalue problem at each parameter sample. The standard estimator is often even rougher than the heuristic error estimator using the residual norm, especially when the *inf-sup* constant is much smaller than 1. To alleviate the need of computing the *inf-sup*

constant, a randomization-based technique has been proposed recently [191]. This approach, as we shall see, results in more accurate error estimation. However, it still faces relatively high computational cost.

3.3.1. Standard State Error Estimation

Consider the discretized system in Eq. (2.10) arising from the discretization of a steady or time-harmonic PDE, i.e.,

$$\begin{aligned} \mathbf{A}\mathbf{X} &= \mathbf{B}, \\ \mathbf{Y} &= \mathbf{C}\mathbf{X}. \end{aligned}$$

Recall that $\mathbf{A} \in \mathbb{C}^{N\times N}$, $\mathbf{B} \in \mathbb{C}^{N\times N_I}$. $\mathbf{X} \in \mathbb{C}^{N\times N_I}$ is the state vector. Its corresponding ROM (given in Eq. (2.16) is):

$$\widehat{\mathbf{A}}\widehat{\mathbf{X}} = \widehat{\mathbf{B}}.$$

For the sake of easy presentation, we shall consider systems having only a single input, i.e., $N_I = 1$ and $\mathbf{X} = \mathbf{x} \in \mathbb{C}^N$. The below discussion can be extended in a straightforward manner to systems with multiple inputs.

The residual is obtained by replacing $\mathbf{x}$ in Eq. (2.10) with the approximate solution $\widetilde{\mathbf{x}} = \mathbf{V}\widehat{\mathbf{x}}$, i.e.,

$$\mathbf{r} = \mathbf{B} - \mathbf{A}\widetilde{\mathbf{x}}. \tag{3.2}$$

The state error estimator for the state variable used in [116] is simply defined as $\Delta_{\mathbf{x},\text{res}} := \|\mathbf{r}\|$ for every parameter $\breve{\boldsymbol{\mu}}$.

The approximation $\|\mathbf{x} - \widetilde{\mathbf{x}}\|$ can be obtained from Eqs. (2.10) and (3.2) as follows:

$$\begin{aligned} \mathbf{r} &= \mathbf{A}\mathbf{x} - \mathbf{A}\widetilde{\mathbf{x}}, \\ \implies \|\mathbf{e}\| := \|\mathbf{x} - \widetilde{\mathbf{x}}\| &= \|\mathbf{A}^{-1}\mathbf{r}\|, \end{aligned} \tag{3.3}$$

with $\mathbf{e} = \mathbf{x} - \widetilde{\mathbf{x}}$ denoting the true state error. In the subsequent discussion, we shall consider the $\|\cdot\|_2$ norm. Based on this, and invoking the sub-multiplicative property of the operator $\mathbf{A}$ we can show the following bound for the norm of the error.

Proposition 3.1:
The norm of the error $\mathbf{e}$ in Eq. (3.3) is bounded above and below by the norm of the residual $\mathbf{r}$ as

$$\frac{1}{\sigma_{\max}}\|\mathbf{r}\|_2 \leq \|\mathbf{e}\| \leq \frac{1}{\sigma_{\min}}\|\mathbf{r}\|_2$$

with $\sigma_{\max} := \|\mathbf{A}\|_2$ and $\sigma_{\min} := \|\mathbf{A}^{-1}\|_2$ the maximum and minimum singular values of the matrix $\mathbf{A}$, respectively.

Proof. From Eq. (3.3), we can show the following upper bound on the residual:

$$\begin{aligned} \|\mathbf{r}\|_2 &= \|\mathbf{A}\mathbf{e}\|_2, \\ &\leq \|\mathbf{A}\|_2 \|\mathbf{e}\|_2 = \sigma_{\max} \|\mathbf{e}\|_2. \end{aligned}$$

Next, starting once again from Eq. (3.3), we derive an upper bound for the error:

$$\begin{aligned} \mathbf{r} &= \mathbf{A}\mathbf{e}, \\ \implies \mathbf{e} &= \mathbf{A}^{-1}\mathbf{r}, \\ \|\mathbf{e}\|_2 &= \left\|\mathbf{A}^{-1}\mathbf{r}\right\|_2, \\ &\leq \left\|\mathbf{A}^{-1}\right\|_2 \|\mathbf{r}\|_2 = \frac{1}{\sigma_{\min}} \|\mathbf{r}\|_2 . \end{aligned}$$

Combining the above two bounds, the proposition is shown to be true. □

The above inequality is used to define the upper bound for the state approximation error, which we call the standard *a posteriori* state error estimator $\Delta_{\mathbf{x},\text{std}} := \frac{1}{\sigma_{\min}} \|\mathbf{r}\|_2$. In the numerically discretized setting, $\sigma_{\min}$ plays the role of the *inf-sup* constant [166].

The standard error estimator has two drawbacks: (i) It is expensive to compute the *inf-sup* constant for every value of the parameter $\breve{\boldsymbol{\mu}}$, and (ii) whenever the *inf-sup* constant is close to zero or is zero for some values of $\breve{\boldsymbol{\mu}}$, the error bound is either too rough to estimate the true error well, or simply goes to infinity. The latter scenario often occurs in electromagnetics, where the matrix $\mathbf{A}$ is close-to-singular or singular nearby or at resonance frequencies. To address the above two issues, we propose a new state error estimator.

3.3.2. An Inf-sup-constant-free State Error Estimator

From Eq. (3.3) we notice that

$$\mathbf{A}\mathbf{e} = \mathbf{r}. \tag{3.4}$$

The true error can be obtained by solving this linear system. However, the above system is of dimension N and therefore, it is not practical to solve it for every parameter. To enable fast error estimation, we propose to construct a reduced model for the above *error-residual* (ER) system. We denote by $\mathbf{V}_\text{e} \in \mathbb{R}^{N \times n_e}$ the projection matrix whose columns span the error subspace denoted by $\mathcal{S}_e := \{\mathbf{e}(\breve{\boldsymbol{\mu}}), \forall \breve{\boldsymbol{\mu}} \in (\mathbb{C} \times \mathcal{P})\}$. The ROM of the ER system in Eq. (3.4) obtained via Galerkin projection using $\mathbf{V}_\text{e}$ is given by

$$\widehat{\mathbf{A}}_e \widehat{\mathbf{e}} = \widehat{\mathbf{r}}, \tag{3.5}$$

with $\widehat{\mathbf{A}}_e := \mathbf{V}_\text{e}^\mathsf{T} \mathbf{A} \mathbf{V}_\text{e} \in \mathbb{C}^{n_e \times n_e}$, $\widehat{\mathbf{r}} := \mathbf{V}_\text{e}^\mathsf{T} \mathbf{r} \in \mathbb{C}^{n_e}$ and the variable $\widetilde{\mathbf{e}} := \mathbf{V}_\text{e} \widehat{\mathbf{e}}$ approximates the true error $\mathbf{e}$. Additionally, the residual of the ER system introduced by the approximation $\widetilde{\mathbf{e}}$ is given by:

$$\mathbf{r}_\text{e} := \mathbf{r} - \mathbf{A}\widetilde{\mathbf{e}}. \tag{3.6}$$

Our proposed error estimator for the state approximation error is $\|\widetilde{\mathbf{e}}\|$, i.e.,

$$\|\mathbf{x} - \widehat{\mathbf{x}}\| = \|\mathbf{e}\| \approx \|\widetilde{\mathbf{e}}\| =: \Delta_{\mathbf{x},\text{ER}}. \tag{3.7}$$

We present the following theorem on the rigorousness of the proposed error estimator:

Theorem 3.2:
The norm of the true error, $\|\mathbf{e}\|$ can be bounded from above and below by the proposed state error estimator $\Delta_{\mathbf{x},\mathrm{ER}}$ as follows:

$$\Delta_{\mathbf{x},\mathrm{ER}} - \gamma \leq \|\mathbf{e}\| \leq \Delta_{\mathbf{x},\mathrm{ER}} + \gamma.$$

Here, $\gamma := \|\mathbf{e} - \widetilde{\mathbf{e}}\| \geq 0$ is a small value whenever $\widetilde{\mathbf{e}}$ is a good approximation to the true error $\mathbf{e}$.

Proof. We have

$$\|\mathbf{e}\| = \|\mathbf{e}\| + \|\widetilde{\mathbf{e}}\| - \|\widetilde{\mathbf{e}}\|.$$

Applying the reverse triangle inequality to $\|\widetilde{\mathbf{e}}\| - \|\widetilde{\mathbf{e}}\|$ yields

$$\|\mathbf{e}\| \leq \|\widetilde{\mathbf{e}}\| + \|\mathbf{e} - \widetilde{\mathbf{e}}\| = \Delta_{\mathbf{x},\mathrm{ER}} + \gamma.$$

To show the lower bound, we start from the expression for the proposed error estimator

$$\Delta_{\mathbf{x},\mathrm{ER}} = \|\widetilde{\mathbf{e}}\| = \|\widetilde{\mathbf{e}}\| + \|\mathbf{e}\| - \|\mathbf{e}\|.$$

Once again, applying the reverse triangle inequality to $\|\widetilde{\mathbf{e}}\| - \|\mathbf{e}\|$ results in

$$\Delta_{\mathbf{x},\mathrm{ER}} \leq \|\mathbf{e}\| + \gamma$$

Combining the two inequalities and rewriting yields

$$\Delta_{\mathbf{x},\mathrm{ER}} - \gamma \leq \|\mathbf{e}\| \leq \Delta_{\mathbf{x},\mathrm{ER}} + \gamma.$$ □

The quantity γ is a measure of how well the ROM Eq. (3.5) approximates the FOM in Eq. (3.3). If γ is sufficiently small, a very tight estimation of the true error is achieved. In Section 3.3.5, we present an adaptive algorithm that ensures this. We next discuss the computation of the proposed error estimator.

3.3.3. Computing the Inf-sup-constant-free State Error Estimator

In order for the error estimator to be used, the error subspace $\mathcal{S}_e$ and an appropriate basis $\mathbf{V}_\mathrm{e}$ need to be identified. Looking at the primal ER system in Eq. (3.4), we observe that

$$\begin{aligned}
\mathbf{e} &= \mathbf{A}^{-1}\mathbf{r}, \\
&= \mathbf{A}^{-1}(\mathbf{B} - \mathbf{A}\widetilde{\mathbf{x}}) \qquad \text{(from Eq. (3.2))}, \\
&= \mathbf{A}^{-1}\mathbf{B} - \mathbf{V}\widehat{\mathbf{x}}.
\end{aligned}$$

Notice that $\mathbf{A}^{-1}\mathbf{B}$ is just the true solution $\mathbf{x}$. Suppose that there exists a basis $\mathbf{V}_s$ that approximates well the true solution, then we can make the following ansatz:

$$\mathbf{x} \approx \mathbf{V}_s\widehat{\mathbf{x}}_s.$$

Thus,

$$\mathbf{e} \approx \mathbf{V}_s \widehat{\mathbf{x}}_s - \mathbf{V}\widehat{\mathbf{x}}. \tag{3.8}$$

From Eq. (3.8), it can be seen that the true error is a linear combination of the columns of $\mathbf{V}$ and $\mathbf{V}_s$. Therefore,

$$\mathbf{V}_\mathrm{e} := \mathrm{colspan}\{\mathbf{V}, \mathbf{V}_s\}. \tag{3.9}$$

Since $\mathbf{e} \neq 0$, we must have range($\mathbf{V}_s$) different from range($\mathbf{V}$), so that $\mathbf{V}_e \neq \mathbf{V}$. In fact, we have the following theorem:

Theorem 3.3:
If $\mathbf{V}_e = \mathbf{V}$, then $\Delta_{\mathbf{x},\mathrm{ER}} = 0$.

Proof. Suppose that $\mathbf{V}_\mathrm{e} = \mathbf{V}$. The ROM of the ER system in Eq. (3.5) reads

$$\widehat{\mathbf{A}}\widehat{\mathbf{e}} = \widehat{\mathbf{r}},$$

where we have used the fact that $\widehat{\mathbf{A}}_e = \widehat{\mathbf{A}}$. Substituting Eq. (3.2) into $\widehat{\mathbf{r}} = \mathbf{V}^\mathsf{T}\mathbf{r}$, we get

$$\begin{aligned}
\widehat{\mathbf{A}}\widehat{\mathbf{e}} &= \mathbf{V}^\mathsf{T}\mathbf{r}, \\
&= \mathbf{V}^\mathsf{T}(\mathbf{B} - \mathbf{A}\widetilde{\mathbf{x}}), \\
&= \mathbf{V}^\mathsf{T}\mathbf{B} - \widehat{\mathbf{A}}\widehat{\mathbf{x}}, \\
&= \mathbf{0} \qquad \text{(Eq. (2.16))}.
\end{aligned}$$

In the last equality above, we have made use of Eq. (2.16). Thus, we infer from $\widehat{\mathbf{A}}\widehat{\mathbf{e}} = 0$ that $\widehat{\mathbf{e}} = 0$ and hence $\Delta_{\mathbf{x},\mathrm{ER}} = \|\widetilde{\mathbf{e}}\| = \|\mathbf{V}_e\widehat{\mathbf{e}}\| = 0$. □

Notice that the proposed error estimator involves direct approximation of the error using the ER system and does not involve the *inf-sup* constant. There are two advantages: (i) The computational cost of solving a large eigenvalue problem at each parameter is avoided, and (ii) since there is no involvement of the *inf-sup* constant, the estimator is expected to perform well for problems with small *inf-sup* constants such as those in electromagnetics. Next, we compare the proposed error estimator to other similar approaches in the literature.

3.3.4. Comparison to the state-of-the-art

For the estimation of the state approximation error, the residual-norm error estimator $\Delta_{\mathbf{x},\mathrm{res}}$ and the standard error estimator $\Delta_{\mathbf{x},\mathrm{std}}$ are the most popular. As already mentioned, the former can be quickly computed but tends to be inaccurate; whereas, the latter is computationally more expensive and is unsuitable when the *inf-sup* constant is very small. Recently, some new error estimator were proposed in the context of the RBM for accurate estimation of the state error [107, 191]. We briefly review both these approaches next and highlight their benefits and drawbacks in comparison to our proposed approach.

Hierarchical Error Estimator An hierarchical error estimator has been proposed in [107] for the RBM and it is defined as:

$$\Delta_{\mathbf{x},\text{hier}} := \|\widetilde{\mathbf{x}} - \widetilde{\mathbf{x}}^{(f)}\| \tag{3.10}$$

where $\widetilde{\mathbf{x}}$ is an approximation to the true solution in the subspace $\mathcal{V}^N$ spanned by the columns of $\mathbf{V}$ and $\widetilde{\mathbf{x}}^{(f)}$ is the approximation to the true solution in an *enriched* subspace $\mathcal{V}^{(f)}$ spanned by the columns of a matrix $\mathbf{V}^{(f)} \in \mathbb{R}^{N \times n^{(f)}}$ and $n^{(f)} > n$. Typically, the authors consider $n^{(f)} = n + i$, $i \in \{1, 2\}$. The hierarchical approach also avoids the computation of the *inf-sup* constant, is accurate and efficient at the online stage, as demonstrated in the numerical examples in [107]. A similar idea was previously proposed in [23] for dynamical systems, where the authors estimate the transfer function error in the non-parametric setting.

There are several key differences that makes the proposed error estimator $\Delta_{\mathbf{x},\text{ER}}$ more attractive to use at the offline stage. Firstly, the validity of the hierarchical error estimator relies on a *saturation assumption* and involves solving several nonlinear optimization problems at each RBM iteration, to determine the saturation constant. This incurs significant offline costs since the determination of the saturation constant involves solving a (possibly NP-hard) nonlinear optimization problem. Secondly, the enriched projection matrix is constructed as union of $\mathbf{V}$ and a few derivatives of the true solution $\mathbf{x}$ w.r.t the parameter. This is in contrast to the approach we adopt for $\Delta_{\mathbf{x},\text{ER}}$, where an enriched subspace $\mathbf{V}_\text{e}$ is computed by adding more snapshots of the solution $\mathbf{x}$ at different parameter samples.

Randomized Error Estimator Recently, a randomized state error estimator was proposed in [191] for Galerkin MOR. The main idea is to approximate the norm of the error by means of Monte Carlo estimation. It is stated that the norm of the error can be rewritten as

$$\|\mathbf{e}\|_\Sigma^2 = \mathbf{e}^\mathsf{T}\mathbb{E}[\mathfrak{z}\mathfrak{z}^\mathsf{T}]\mathbf{e} = \mathbb{E}[(\mathfrak{z}^\mathsf{T}\mathbf{e})^2] \tag{3.11}$$

with $\mathfrak{z} \in \mathbb{R}^N$ a zero mean, Gaussian random vector and $\Sigma \in \mathbb{R}^{N \times N}$ is a covariance matrix and $\|\cdot\|_\Sigma$ is defined as $\|\mathbf{e}\|_\Sigma^2 = \mathbf{e}^\mathsf{T}\Sigma\mathbf{e}$. For our discussion Σ is simply the identity matrix since we are interested in the Euclidean norm. Furthermore, $\mathbb{E}[\cdot]$ denotes the expected value. A Monte Carlo approximation to the above expression for the norm in Eq. (3.11) is sought as

$$\|\mathbf{e}\| = (\mathbb{E}[(\mathfrak{z}^\mathsf{T}\mathbf{e})^2])^{1/2} \approx \sqrt{\frac{1}{M_s}\sum_{i=1}^{M_s}\left(\mathfrak{z}_i^\mathsf{T}\mathbf{e}\right)^2} \tag{3.12}$$

where $\{\mathfrak{z}_i\}_{j=1}^{M_s}$ are the Monte Carlo samples of the random vector $\mathfrak{z}$ and M_s denotes the number of such samples. Next, the authors show the following equivalence:

$$\sqrt{\frac{1}{M_s}\sum_{i=1}^{M_s}\left(\mathfrak{z}_i^\mathsf{T}\mathbf{e}\right)^2} = \sqrt{\frac{1}{M_s}\sum_{i=1}^{M_s}\left(\mathfrak{y}_i^\mathsf{T}\mathbf{r}\right)^2}, \tag{3.13}$$

with $\mathbf{r}$ being the residual defined in Eq. (3.2) and the M_s random vectors $\{\mathfrak{y}_i\}_{i=1}^{M_s}$ are the solution vectors corresponding to the following M_s *random dual systems*:

$$\mathbf{A}^{\mathsf{T}}\mathfrak{y}_i = \mathfrak{z}_i, \qquad i = 1, 2, \ldots, M_s. \tag{3.14}$$

The authors of [191] then proceed to construct ROMs for the M_s random dual systems as:

$$\widehat{\mathbf{A}}_{\text{rand}}^{\mathsf{T}}\hat{\mathfrak{y}}_i = \hat{\mathfrak{z}}_i, \qquad i = 1, 2, \ldots, M_s \tag{3.15}$$

with $\widehat{\mathbf{A}}_{\text{rand}} := \mathbf{V}_{\text{rand}}^{\mathsf{T}}\mathbf{A}\mathbf{V}_{\text{rand}}$, $\hat{\mathfrak{z}}_i := \mathbf{V}_{\text{rand}}^{\mathsf{T}}\mathfrak{z}_i$ and $\widetilde{\mathfrak{y}}_i \approx \mathbf{V}_{\text{rand}}\hat{\mathfrak{y}}_i$. Combining Eqs. (3.11) and (3.13) and the above ROM approximation, the randomized state error estimator is of the form shown below:

$$\Delta_{\mathbf{x},\text{rand}} := \sqrt{\frac{1}{M_s}\sum_{i=1}^{M_s}\left(\widetilde{\mathfrak{y}}_i^{\mathsf{T}}\mathbf{r}\right)^2}. \tag{3.16}$$

The authors demonstrate that whenever $\mathbf{V}_{\text{rand}} = \mathbf{V}_{\text{e}}$ in Eq. (3.5) the following equality will hold:

$$\Delta_{\mathbf{x},\text{rand}} := \sqrt{\frac{1}{M_s}\sum_{i=1}^{M_s}\left(\widetilde{\mathfrak{y}}_i^{\mathsf{T}}\mathbf{r}\right)^2} = \sqrt{\frac{1}{M_s}\sum_{i=1}^{M_s}\left(\mathfrak{z}_i^{\mathsf{T}}\widetilde{\mathbf{e}}\right)^2} \tag{3.17}$$

When compared to the true error $\|\mathbf{e}\|$, the randomized error estimator is constructed using two layers of approximation. The first is the Monte Carlo approximation in Eq. (3.12) and the second is the ROM approximation of Eq. (3.15) or equivalently the ROM approximation Eq. (3.5). In contrast, our proposed error estimator $\Delta_{\mathbf{x},\text{ER}}$ is a direct approximation of the true error using the ROM Eq. (3.5). If a similar MOR accuracy is used to obtain both the error estimators, our proposed error estimator should be more accurate. In [191], two greedy algorithms are proposed to construct the matrix $\mathbf{V}_{\text{rand}}$. However, they incur significant computational effort. This is due to two reasons: (i) the tolerance for obtaining the ROM in Eq. (3.15) ϵ_{rand} is unknown beforehand and is, therefore, chosen heuristically. Adopting a conservative approach leads to slow convergence of the greedy algorithms, and (ii) the number of random vectors M_s is also not known. Overlooking the computational challenges, the randomized error estimator leads to more accurate estimation of the true error than the standard error estimator. Nevertheless, it tends to underestimate the true error as compared to our proposed error estimator. Our numerical results demonstrate this phenomenon.

Remark 3.4:
Although we discussed the SISO case above, the standard error estimator $\Delta_{\mathbf{x},\text{std}}$, the *inf-sup-constant-free* error estimator $\Delta_{\mathbf{x},\text{ER}}$, and the randomized error estimator $\Delta_{\mathbf{x},\text{rand}}$ are also applicable for multi-input systems, i.e., systems with $N_I > 1$. We can evaluate N_I different $\Delta_{\mathbf{x},\text{std}}$, $\Delta_{\mathbf{x},\text{ER}}$, and $\Delta_{\mathbf{x},\text{rand}}$, i.e., $\left(\{\Delta_{\mathbf{x},\text{std}}^{(i)}\}_{i=1}^{N_I}, \{\Delta_{\mathbf{x},\text{ER}}^{(i)}\}_{i=1}^{N_I}, \{\Delta_{\mathbf{x},\text{rand}}^{(i)}\}_{i=1}^{N_I}\right)$, corresponding to N_I different right-hand side vectors in Eq. (2.10) and pick the maximum. ♢

Algorithm 3.1: ROMGreedy-State
Computes a ROM for Eq. (2.10) using a greedy algorithm.

Input: System matrices: $\mathbf{A}, \mathbf{B}$, Training set Ξ, ROM tolerance ϵ, Maximum number of greedy iterations *iter_max*.
Output: Reduced system matrices $\widehat{\mathbf{A}}, \widehat{\mathbf{B}}, \widehat{\mathbf{C}}$.

Initialization: $\mathbf{V} = [\,]$, $\mathbf{V}_\mathrm{e} = [\,]$, $err\_max = 1 + \epsilon$, $iter = 1$, initial greedy parameters $\breve{\boldsymbol{\mu}}^*$ and $\breve{\boldsymbol{\mu}}^e$ (chosen randomly from Ξ) and $\breve{\boldsymbol{\mu}}^* \neq \breve{\boldsymbol{\mu}}^e$.
while $err\_max > \epsilon$ and $iter \leq iter\_max$ **do**
  Compute matrix $\mathbf{V}_{\breve{\boldsymbol{\mu}}^*}$ using a preferred MOR method applied to the FOM in Eq. (2.10); update projection matrix $\mathbf{V}$: $\mathbf{V} = \mathrm{orth}[\mathbf{V},\ \mathbf{V}_{\breve{\boldsymbol{\mu}}^*}]$; if a real $\mathbf{V}$ is preferred set $\mathbf{V} := [\mathrm{real}(\mathbf{V})\ \mathrm{imag}(\mathbf{V})]$.
  Compute matrix $\mathbf{V}_{\breve{\boldsymbol{\mu}}^e}$ using a preferred MOR method applied to the FOM Eq. (2.10); update projection matrix $\mathbf{V}_s$: $\mathbf{V}_s = [\mathbf{V}_s,\ \mathbf{V}_{\breve{\boldsymbol{\mu}}^e}]$; if a real $\mathbf{V}_s$ is preferred set $\mathbf{V}_s := [\mathrm{real}(\mathbf{V}_s)\ \mathrm{imag}(\mathbf{V}_s)]$.
  Set error projection matrix $\mathbf{V}_\mathrm{e} := [\mathbf{V}\ \mathbf{V}_s]$.
  $iter = iter + 1$.
  Obtain ROMs Eqs. (2.16) and (3.5) using $\mathbf{V}, \mathbf{V}_\mathrm{e}$; compute $\widetilde{\mathbf{e}}$ to determine the error estimator $\Delta_{\mathbf{x},\mathrm{ER}}$ for all $\breve{\boldsymbol{\mu}} \in \Xi$.
  Find $\breve{\boldsymbol{\mu}}^* = \arg\max_{\breve{\boldsymbol{\mu}} \in \Xi} \Delta_{\mathbf{x},\mathrm{ER}}$.
  Find $\breve{\boldsymbol{\mu}}^e = \arg\max_{\breve{\boldsymbol{\mu}} \in \Xi} \|\mathbf{r}_\mathrm{e}\|$, where $\mathbf{r}_\mathrm{e}$ is defined in Eq. (3.6).
  Set $err\_max := \Delta_{\mathbf{x},\mathrm{ER}}(\breve{\boldsymbol{\mu}}^*)$.
end

3.3.5. Greedy algorithm for ROM Construction with the State Error Estimator

In the section, we discuss adaptive construction of the ROM Eq. (2.16) for steady/time-harmonic systems. We consider Galerkin projection and therefore seek to identify an appropriate projection matrix $\mathbf{V} \in \mathbb{R}^{N \times n}$. Recall that the system is parametrized either by $\breve{\boldsymbol{\mu}} = \boldsymbol{\mu}$ or by the parameter $\breve{\boldsymbol{\mu}} = [s\,\boldsymbol{\mu}]$ where s is the complex Laplace variable and $\boldsymbol{\mu} \in \mathbb{R}^{N_p}$ is any general parameter. Therefore, our goal is to obtain $\mathbf{V}$ that results in a good ROM for any parameter in the parameter space $\mathcal{P}$ or $\mathcal{P}_c := \mathbb{C} \times \mathcal{P}$. To enable practical computation, usually a training set $\Xi := [\breve{\boldsymbol{\mu}}_1, \ldots, \breve{\boldsymbol{\mu}}_{n_s}]$ with a finite but large number of samples is chosen and we require the ROM to be a uniformly good approximation for all $\breve{\boldsymbol{\mu}} \in \Xi$.

We propose a greedy approach to construct the basis $\mathbf{V}$. The greedy algorithm is similar in spirit to the that used in the RBM [112, 167]. In our proposed scheme of adaptive ROM construction, the error subspace $\mathcal{S}_e$ (and hence the error estimator $\Delta_{\mathbf{x},\mathrm{ER}}$) is updated as a part of the algorithm unlike the randomized error estimation in [191] where two greedy algorithms are implemented in order to compute $\Delta_{\mathbf{x},\mathrm{rand}}$. All the computations needed to compute the proposed error estimator $\Delta_{\mathbf{x},\mathrm{ER}}$ are implemented in one greedy algorithm. The proposed method is sketched in Algorithm 3.1. As inputs, Algorithm 3.1 requires the system matrices in addition to the training set Ξ and a desired tolerance ϵ for the ROM. The algorithm is automatic, except the need

for a user-defined training set Ξ. However, this process can also be automated and will be described in detail in Section 3.4.5. Steps 3-4 of Algorithm 3.1 involve application of a preferred MOR method (RBM, MMM, etc.) to the FOM Eq. (2.10) to compute $\breve{\boldsymbol{\mu}}^*, \breve{\boldsymbol{\mu}}^e$-dependent matrices $\mathbf{V}_{\breve{\boldsymbol{\mu}}^*}, \mathbf{V}_{\breve{\boldsymbol{\mu}}^e}$. If employing the RBM, $\mathbf{V}_{\breve{\boldsymbol{\mu}}^*}$ in Step 3 is simply the snapshot: $\mathbf{A}(\breve{\boldsymbol{\mu}}^*)^{-1}\mathbf{B}(\breve{\boldsymbol{\mu}}^*)$ and $\mathbf{V}_{\breve{\boldsymbol{\mu}}^e}$ in Step 4 is the snapshot: $\mathbf{A}(\breve{\boldsymbol{\mu}}^e)^{-1}\mathbf{B}(\breve{\boldsymbol{\mu}}^e)$. The MMM method introduced in Section 2.4.2 can also be used; then $\mathbf{V}_{\breve{\boldsymbol{\mu}}^*}$ or $\mathbf{V}_{\breve{\boldsymbol{\mu}}^e}$ can be computed via Algorithm 2.1 by setting $\boldsymbol{\theta}^0$ in Eq. (2.22) as $\boldsymbol{\theta}^0 = \breve{\boldsymbol{\mu}}^*$ or $\boldsymbol{\theta}^0 = \breve{\boldsymbol{\mu}}^e$, respectively. It can be shown that, for steady systems, the RBM is a special case of the MMM method [82].

The initialization of $\breve{\boldsymbol{\mu}}^*, \breve{\boldsymbol{\mu}}^e$ in Step 1 and also their subsequent choices in Steps 8 and 9 are done to ensure that $\mathbf{V}_e \neq \mathbf{V}$; see Theorem 3.3. It is worth noting that the solution to the FOM Eq. (2.10) can be complex-valued. Therefore, if preferred, a *realification* of the projection matrices $\mathbf{V}, \mathbf{V}_s$ is done in Steps 3 and 4. In Step 8, the greedy parameter $\breve{\boldsymbol{\mu}}^*$ for the next iteration is determined as the one maximizing the error estimator $\Delta_{\mathbf{x},\mathrm{ER}}$. It is computed by solving the ROM Eq. (3.5). To identify the second greedy parameter $\breve{\boldsymbol{\mu}}^e$, we use the norm of the residual Eq. (3.6) corresponding to the ER system. As output of the algorithm, we obtain the reduced system matrices $\widehat{\mathbf{A}}, \widehat{\mathbf{B}}, \widehat{\mathbf{C}}$. If desired, the projection matrices $\mathbf{V}, \mathbf{V}_\mathrm{e}$ can be obtained as additional outputs, which can also be used to perform online error estimation whenever necessary.

3.3.5.1. Computational Costs

We briefly compare the computational costs of the adaptive ROM construction using the standard error estimator and the proposed *inf-sup-constant-free* error estimator. For simplicity, we assume the system matrices are dense. We perform the analysis for a system with N_I inputs.

- For Algorithm 3.1 using the proposed *inf-sup-constant-free* error estimator:
  - Steps 3 and 4 incur cost[1] scaling as $2 \cdot O(N^3)$ (assume a direct linear system solver is used, for e.g., LU decomposition),
  - In Step 7, the solution of the ROM Eq. (2.16) involves costs: $O\big(Q_\mathbf{A}(N^2 n + Nn^2) + Q_\mathbf{B} N N_I n\big)$ to form the reduced matrices (see Eq. (2.14)), $n_s \cdot O(n^3)$ to solve the linear system,
  - In Step 7, the solution of the other ROM Eq. (3.5) involves costs: $n_s \cdot O(NN_I n + N^2 N_I + NN_I)$ to obtain the residual Eq. (3.2), $O\big(Q_\mathbf{A}(N^2 n_e + Nn_e^2)\big) + n_s \cdot O(Nn_e N_I)$ to form the reduced matrix $\widehat{\mathbf{A}}_e$ and vector $\widehat{\mathbf{r}}$, respectively, in Eq. (3.2), $n_s \cdot O(n_e^3)$ to solve the linear system,
  - Computing $\widetilde{\mathbf{e}}$ costs $n_s \cdot O(NN_I n_e)$,
  - Step 8 computes $\Delta_{\mathbf{x},\mathrm{ER}}$ with cost: $n_s \cdot O(N)$.
  - Step 9 involves a matrix-vector product to evaluate the residual $\mathbf{r}_\mathrm{e}$ and its norm at a cost: $n_s \cdot O(N^2 N_I + NN_I)$.

[1] We use the big-O notation

- For a greedy ROM construction based on the standard error estimator (Algorithm B.1), the computational costs are:
  - In Step 2, the *inf-sup* constant $\sigma_{\min}(\breve{\boldsymbol{\mu}})$ is computed for all $\breve{\boldsymbol{\mu}} \in \Xi$ at cost $n_s \cdot O(N^3)$,
  - In Step 4 the FOM incurs cost: $O(N^3)$,
  - The solution of the ROM Eq. (2.16) in Step 7 involves costs: $O\big(Q_{\mathbf{A}}(N^2 n + Nn^2) + Q_{\mathbf{B}} N N_I n\big)$ to form the reduced matrices, $n_s \cdot O(n^3)$ to solve the linear system.
  - The computation of $\mathbf{r}$ using Eq. (3.2) costs $n_s \cdot O(N^2 N_I + N N_I n + N N_I)$.
  - Step 8 of computing $\Delta_{\mathbf{x},\text{std}}$ has costs: $n_s \cdot O(N)$.

Although Algorithm 3.1 involves the computation of an additional FOM (in Step 4) and ROM (in Step 7) at each iteration, it results in a overall reduction in cost when compared to Algorithm B.1. In particular, when $1 \ll n_s \ll N$, the dominant cost of Algorithm B.1 is $n_s \cdot O(N^3)$, which is definitely larger than the dominant cost $2 \cdot O(N^3)$ of Algorithm 3.1.

We now validate the proposed error estimator $\Delta_{\mathbf{x},\text{ER}}$ and the adaptive ROM construction method in Algorithm 3.1 on two models of electromagnetic devices.

3.3.6. Numerical Examples

Microwave devices, such as filters and antennas, are designed to operate in a particular range of frequencies. Frequency sweep studies play an important role in the design of microwave devices, especially to ensure qualities such as optimal gain, noise rejection. However, such studies are time consuming since a large-scale system needs to be simulated for each frequency sample. It is often the case that the range of operation is wide, spanning frequencies varying from mega-hertz (MHz) to giga-hertz (GHz). Therefore, ROMs play a key role in speeding up design studies.

In this section, we test the proposed *inf-sup-constant-free* state error estimator on two real-life electromagnetic devices. The first device is the dual-mode circular waveguide filter, a MIMO system. The other device is a wide-band antenna, a SISO system. The operation of both devices is governed by Maxwell's equations. After discretization using FEM, employing second-order first family of Nédélec's elements, each of the systems can be represented in the form of Eq. (2.10). The meshes are generated using the software `Gmsh` [92].

The systems we consider are one-parameter models, with $\breve{\boldsymbol{\mu}} = s := \jmath 2\pi f$. The system matrix $\mathbf{A}(\breve{\boldsymbol{\mu}})$ takes the following parameter affine form:

$$\mathbf{A}(s) = \mathbf{A}_0 + s\mathbf{A}_1 + s^2\mathbf{A}_2.$$

In the above form, $\mathbf{A}_0 \in \mathbb{R}^{N\times N}$ plays the role of a stiffness matrix, $\mathbf{A}_2 \in \mathbb{R}^{N\times N}$ is the mass matrix and $\mathbf{A}_1 \in \mathbb{R}^{N\times N}$ is the FEM matrix related to first-order absorbing boundary conditions. The matrix $\mathbf{B}(\breve{\boldsymbol{\mu}})$ takes the form

$$\mathbf{B}(s) := s\mathbf{B}_1$$

with $\mathbf{B}_1 \in \mathbb{R}^{N\times N_I}$. Recall that N_I is the number of inputs, which, in the context of the systems we consider, is the number of excitation ports on the device. The state variable $\mathbf{X}(s) \in \mathbb{C}^{N\times N_I}$ is the discretized electric field in the computational domain of the model.

Next, we describe the metrics that are used to quantify the performance of different algorithms we test. In our numerical tests, we consider the true error ϵ_{true} as the maximal difference among all columns of the true solution matrix $\mathbf{X}(s)$ and its approximation $\widetilde{\mathbf{X}}(s)$:

$$\epsilon_{\text{true}} = \max_{\substack{i\in\{1,2,\ldots,N_I\},\\ s\in\Xi}} \|\mathbf{x}_i(s) - \widetilde{\mathbf{x}}_i(s)\|_2. \tag{3.18}$$

Here, $\mathbf{x}_i$ and $\widetilde{\mathbf{x}}_i$ are the i-th columns of the true and approximate solution matrices, respectively. Further, Ξ is the training set consisting of samples of the parameter. We use the same notion for the estimated error as well and define the maximal estimated error (see Remark 3.4) ϵ_{est} as

$$\epsilon_{\text{est},\square} = \max_{\substack{i\in\{1,2,\ldots,N_I\},\\ s\in\Xi}} \Delta_{\mathbf{x},\square}(s) \tag{3.19}$$

with $\square$ serving as a place holder either for the residual error estimator (res), the standard error estimator (std), the randomized error estimator (rand), or the *inf-sup-constant-free* estimator (ER). We also define the effectivity (`eff`) as

$$\texttt{eff} := \frac{\epsilon_{\text{est}}}{\epsilon_{\text{true}}}.$$

The effectivity is a gauge for the closeness of the estimated error to the true error. The closer `eff` is to 1, the sharper or tighter the estimated error.

For each example in the numerical experiments, we evaluate the performance of the four error estimators, viz., the standard state error estimator ($\Delta_{\mathbf{x},\text{std}}$), the residual error estimator ($\Delta_{\mathbf{x},\text{res}}$), the randomized error estimator ($\Delta_{\mathbf{x},\text{rand}}$), and the proposed state error estimator ($\Delta_{\mathbf{x},\text{ER}}$). In what follows,

- Test A refers to the greedy algorithm Algorithm B.1 using the standard state error estimator,
- Test B refers to the greedy algorithm Algorithm B.2 which uses the norm of the residual as a heuristic error estimator,
- Test C stands for Algorithm B.3 using the randomized state error estimator,
- Test D refers to Algorithm 3.1.

Remark 3.5:
The matrix $\mathbf{B}(s)$ has entries with large magnitude due to the large value of s multiplying $\mathbf{B}_1$. However, the entries of the matrix $\mathbf{A}(s)$ are relatively small in magnitude. If not treated appropriately, the solution $\mathbf{X}(s)$ has a large norm, which may lead to large magnitude of the state error, i.e., ϵ_{true}. This distorts our measure of the actual accuracy of the ROM. Therefore, we first scale the matrix $\mathbf{B}$ with an appropriate scaling

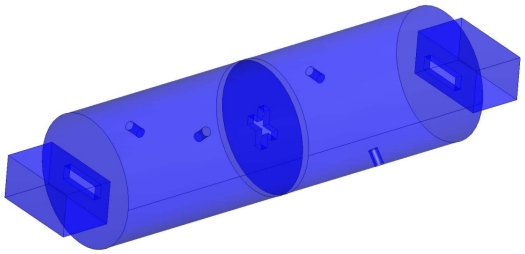


Figure 3.1.: Dual-mode Circular Waveguide Filter.

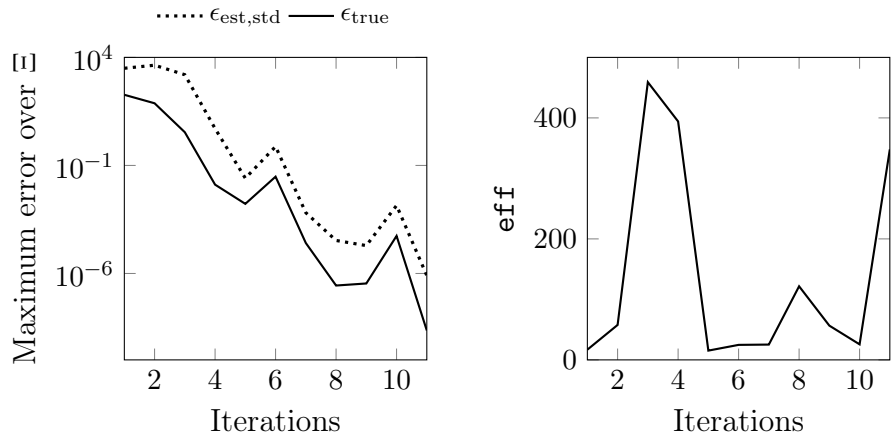
constant, i.e., we set $\mathbf{B} := (1/sc) \times \mathbf{B}$. The maximal true and estimated errors $\epsilon_{\text{true}}, \epsilon_{\text{est}}$ are reported for the scaled solutions. The scaling causes no distortion and the unscaled solution can be recovered by simple multiplication. The scaling strategy is not *ad-hoc*, but is actually determined by looking at the maximum entry in $s\mathbf{B}_1$. For the numerical tests below, we use $sc = 10^5$. An alternate approach would be to use a relative error estimator. However, note that the reduced solution $\widehat{\mathbf{X}}(s)$ varies with s. Therefore, a relative error estimator can not accurately reflect the absolute error estimator that we propose. Whereas, using a scaling strategy, we are still able to compute an absolute error estimator. ◊

Remark 3.6:
The two examples to be considered result in complex-valued solution vectors (or matrices). However, the system matrices $\mathbf{A}_i, i \in \{0, 1, 2\}$ are real. To obtain real reduced-order matrices, we perform a *realification* of the form $\mathbf{V} := \text{orth}\big([\text{real}(\mathbf{V}), \text{imag}(\mathbf{V})]\big)$, as highlighted in Step 3 and 4 of Algorithm 3.1. The above definition of $\mathbf{V}$ is motivated by the following reasons: first, we observe that $\text{span}\{\mathbf{V}\} \subset \text{span}\{\text{Re}(\mathbf{V}), \text{Im}(\mathbf{V})\} \subset \mathbb{C}^N$ over $\mathbb{C}$. From Theorem 2.13, we get the same moment matching property using $\mathbf{V} := \text{orth}\big([\text{real}(\mathbf{V}), \text{imag}(\mathbf{V})]\big)$, as using the original complex $\mathbf{V}$. ◊

For the first example, the numerical results were obtained using MATLAB®2015a, on a laptop running INTEL®CORE™i5-7200U, 2.5 GHz running with 8 GB of RAM. Owing to greater memory requirements, the simulation of the antenna example was done on a workstation with 3 GHz, INTEL®CORE™Xeon E5-2687W v4 processors and 256 GB or RAM, with MATLAB®2017a.

3.3.6.1. Dual-mode Circular Waveguide Filter

The first example we consider is a MIMO system arising from the model of a dual-mode circular waveguide filter [67] shown in Figure 3.1. It is a type of narrow bandpass filter widely used in satellite communication owing to its favourable properties such as efficient power handling [66, 68]. The operation of this device is governed by Maxwell's equations. The FEM discretization leads to a system of dimension $N = 36,426$ with $\mathbf{A}(s) = \mathbf{A}_0 + s^2\mathbf{A}_2$. This system has two input excitation ports ($N_I = 2$) and two outputs ($N_O = 2$). The frequency band of interest is [11.5, 12] GHz. We consider the training set Ξ consisting of 101 uniformly-spaced frequency samples from the band of interest. The ROM tolerance ϵ is 10^{-6}. We now perform Tests A-D for this example.



(a) Convergence of the greedy algorithm Algorithm B.1.

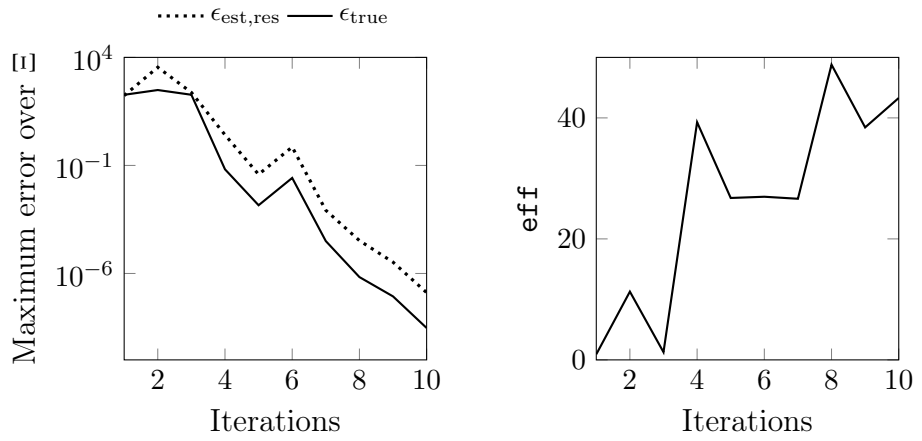
(b) Effectivity (`eff`).

Figure 3.2.: Dual-mode Circular Waveguide Filter: results for Test A.

Test A We use Algorithm B.1 to adaptively obtain a ROM for the filter. We recall that, this procedure involves computing the *inf-sup* constant for all the parameters in the training set Ξ and is expensive. In Fig. 3.2a we depict the convergence of the estimated error $\epsilon_{\text{est,std}}$ used in Algorithm B.1. It takes 11 iterations to converge. Further, in Fig. 3.2b we show the effectivity (`eff`) as a function of the iteration. The effectivity is of order $O(10^2)$, which informs us that $\Delta_{\mathbf{x},\text{std}}$ is not very sharp. This is mainly due to the fact that the *inf-sup* constants are very small for this example and they result in much overestimation of the true error. The dimension of the ROM resulting from this approach is $n = 20$.

Test B As our next experiment, we use Algorithm B.2 to obtain a ROM for the dual-mode filter. As done for the previous example, we illustrate the convergence of the estimated error $\epsilon_{\text{est,res}}$ in Fig. 3.3a and the effectivity in Fig. 3.3b. The greedy algorithm requires 10 iterations to converge and the resulting ROM dimension is $n = 20$. The effectivity is of order $O(10)$. The better effectivity here is due to the absence of the *inf-sup* constant.

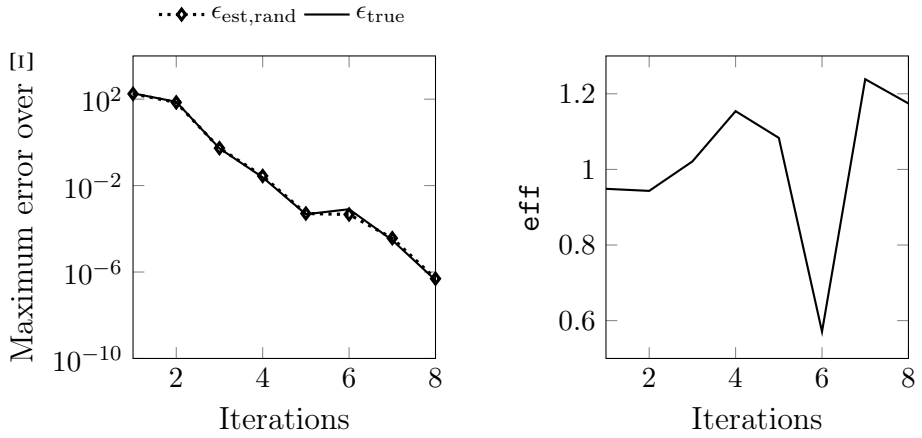
Test C We now apply Algorithm B.3 which uses the randomized error estimator $\Delta_{\mathbf{x},\text{rand}}$. To compute the error estimator we pick $M_s = 20$ random vectors from a random distribution with zero mean and unit covariance, using the MATLAB®command `mvnrnd` with the seed set to be zero. The random projection matrix $\mathbf{V}_{\text{rand}}$ is generated first using a separate greedy algorithm, and is the most time consuming part of Algorithm B.3. The generation of $\mathbf{V}_{\text{rand}}$ uses a tolerance ϵ_{rand} different from the one used for the ROM, viz., ϵ. We refer to [191] for the details. In our numerical test we use $\epsilon_{\text{rand}} = 5 \cdot 10^{-1}$. As shown in Figure 3.4, the algorithm converges in 8 iterations, yielding a ROM with dimension $n = 16$. Further, the effectivity of the randomized error estimator is very close to the ideal value of 1. Yet, there do occur underestimations for a few iterations.



(a) Convergence of the greedy algorithm Algorithm B.2.

(b) Effectivity (`eff`).

Figure 3.3.: Dual-mode Circular Waveguide Filter: results for Test B.



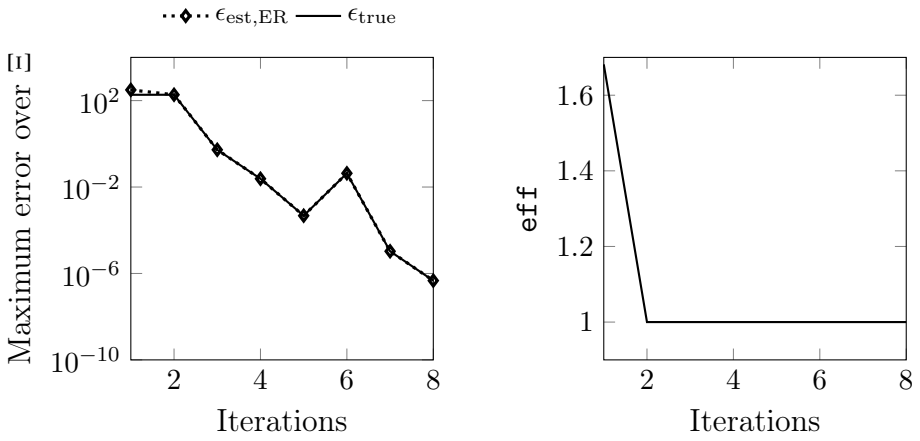
(a) Convergence of the greedy algorithm Algorithm B.3.

(b) Effectivity (`eff`).

Figure 3.4.: Dual-mode Circular Waveguide Filter: results for Test C.

Test D We now employ Algorithm 3.1 to adaptively generate a ROM for the dual-mode filter model. The convergence of the algorithm is displayed in Figure 3.5a and it is seen that 8 iterations to achieve the desired tolerance of $\epsilon = 10^{-6}$. The ROM dimension is $n = 16$. Evidently, the proposed error estimator leads to much better effectivity (see Figure 3.5b) than the standard state error estimator $\Delta_{\mathbf{x},\text{std}}$ and the residual error estimator $\Delta_{\mathbf{x},\text{res}}$. Although the randomized error estimator $\Delta_{\mathbf{x},\text{rand}}$ yields similar effectivity, the proposed estimator is more reliable with almost no underestimation of the true error.

To validate the ROMs derived from Tests A-D (Algorithms 3.1 and B.1 to B.3) to the dual-mode waveguide filter, we evaluate the true errors resulting from each of the ROMs



(a) Convergence of the greedy algorithm Algorithm 3.1.

(b) Effectivity (`eff`).

Figure 3.5.: Dual-mode Circular Waveguide Filter: results for Test D.

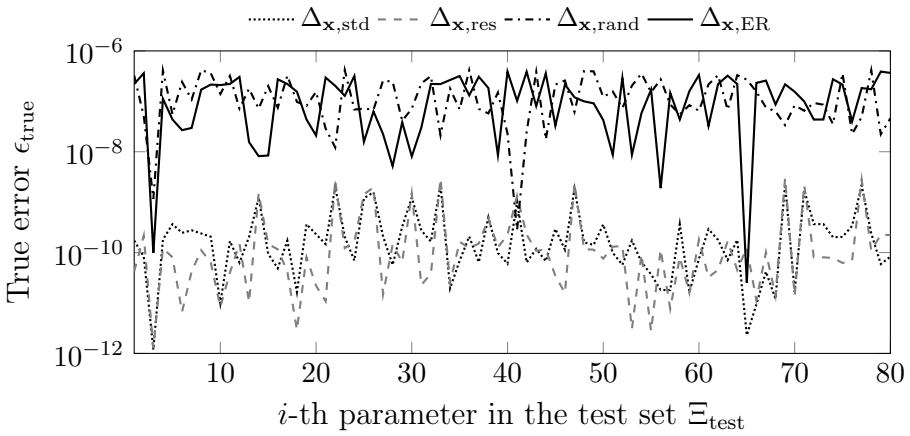


Figure 3.6.: Dual-mode Circular Waveguide Filter: error evaluated over a test set of parameters for ROMs obtained using different error estimators.

over a test set Ξ_{test} consisting of 80 randomly sampled parameters (different from the parameter samples in the training set Ξ) in the frequency range [11.5, 12] GHz; Figure 3.6 shows the results. We note that all the four ROMs result in error satisfying the desired tolerance of $\epsilon = 10^{-6}$. For the ROMs obtained with $\Delta_{\mathbf{x},\text{std}}, \Delta_{\mathbf{x},\text{res}}$ (Algorithms B.1 and B.2) the maximum error over the test set is much smaller when compared to the maximum test set errors for $\Delta_{\mathbf{x},\text{rand}}$ or $\Delta_{\mathbf{x},\text{ER}}$ ((Algorithms 3.1 and B.3)). This reflects the overestimation of $\Delta_{\mathbf{x},\text{std}}$ and $\Delta_{\mathbf{x},\text{resd}}$, which did not make the greedy algorithm converge as early as possible and led to ROMs with higher orders than the ROMs obtained using $\Delta_{\mathbf{x},\text{rand}}$ and $\Delta_{\mathbf{x},\text{ER}}$.

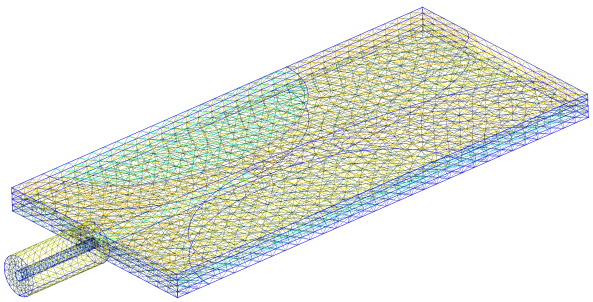


Figure 3.7.: Antipodal Vivaldi Antenna.

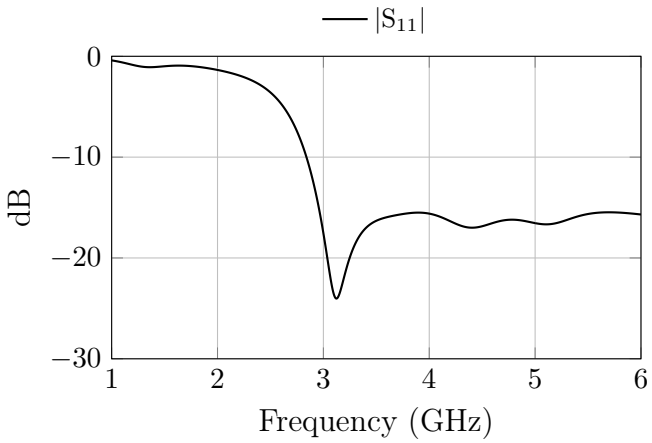
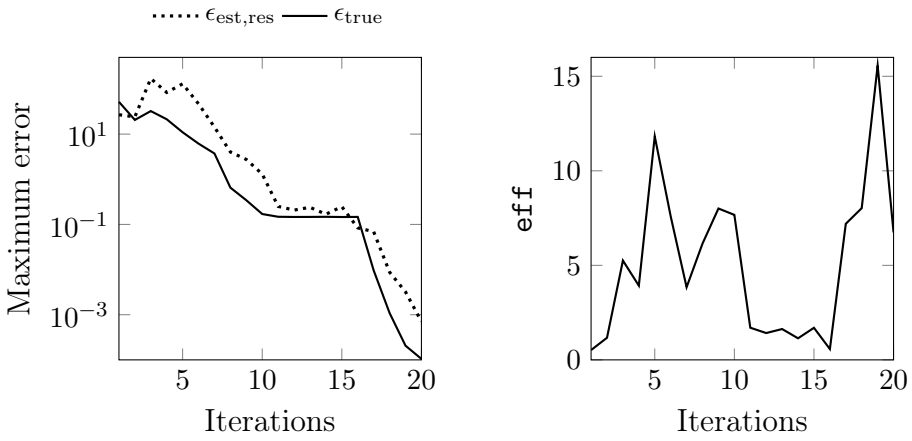


Figure 3.8.: s-parameter for the Antipodal Vivaldi Antenna.

3.3.6.2. Antipodal Vivaldi Antenna

The second example we consider is the antipodal Vivaldi antenna (AVA). This device is used in radar and wireless communication applications and is known for good wide-band performance [131]. The discretized model of the AVA is shown in Figure 3.7 and has dimension $N = 283,846$. The frequency range of interest for this antenna is $f \in [1, 6]$ GHz. Unlike the dual-mode waveguide filter, the AVA is a SISO device. The device characteristics are usually expressed in terms of the input reflection coefficient (or *s-parameter*) at the coaxial port and a typical response is shown in Figure 3.8. This model is particularly challenging to approximate with MOR techniques, owing to the larger number of in-band resonances. The tolerance for the ROM is set as $\epsilon = 10^{-3}$. The training set Ξ is made up of 51 uniform samples from the parameter range of interest. Unlike the previous case, we perform Tests B-D for the AVA. The results of Test A are not shown for two reasons: firstly, it is computationally very expensive due to the large-scale nature of the AVA; secondly, the resulting standard error estimator leads to poor estimation of the error.

Test B As noted previously, the AVA is a challenging model to approximate with MOR techniques. This is evident from Fig. 3.9a, where the convergence of the greedy



(a) Convergence of the greedy algorithm Algorithm B.2.

(b) Effectivity (`eff`).

Figure 3.9.: Antipodal Vivaldi Antenna: results for Test B.

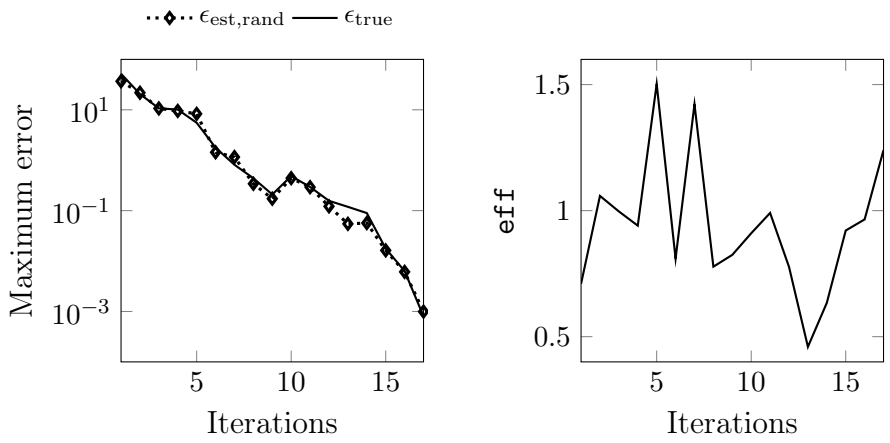
algorithm is shown. The algorithm requires up to 20 iterations to achieve the set tolerance. For this model, the norm of the residual overestimates the true error by roughly one order of magnitude, as seen in Fig. 3.9b, where the effectivity is illustrated. The resulting ROM has dimension $n = 40$.

Test C In this test, we use the randomized error estimator proposed in [191]. To construct $\mathbf{V}_{\text{rand}}$ for this example, we draw $M_s = 6$ random vectors using the `mvnrnd` command and set ϵ_{rand} to be 1[2]. Algorithm B.3 from [191] takes nearly 3 hours and 12 minutes to generate $\mathbf{V}_{\text{rand}}$. The results of the greedy algorithm are displayed in Fig. 3.10a. Compared to Test B for the AVA, Test C requires only 17 iterations to converge and results in a smaller ROM of dimension $n = 34$.

Test D Finally, we use Algorithm 3.1 to generate a ROM for the AVA. This results in a ROM of dimension $n = 36$, with the convergence achieved in 18 iterations. The convergence of the maximum error and the corresponding effectivity are shown in Fig. 3.11. Although in comparison to the randomized error estimator in Test C, the proposed approach takes one extra iteration to converge, the overall time for Test D is only 1 hour and 20 minutes. Moreover, the effectivity is nearly 1 for most of the iterations. The randomized estimator's tendency to underestimate the true error also explains why it takes fewer iterations to converge: the algorithm stops according to the already small error estimate, even before the true error is below the tolerance over the whole parameter domain.

We validate the ROMs derived from Tests B-D (Algorithms 3.1, B.2 and B.3) to the antenna model by evaluating the true errors of the ROMs over the test set Ξ_{test} consisting of 100 randomly sampled parameters (different from the parameter samples in the training set Ξ) in the frequency range [1, 6] GHz; Figure 3.12 shows the results. We note that all the three ROMs result in error satisfying the desired tolerance of $\epsilon = 10^{-3}$.

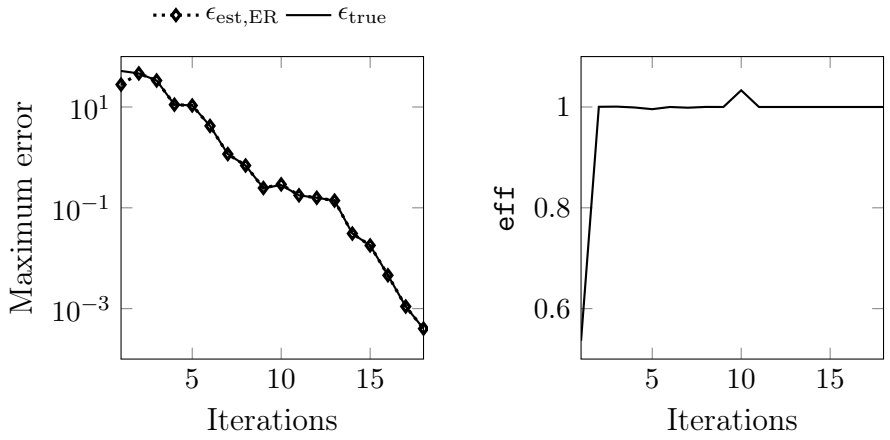
[2]The random number generator `MersenneTwister` was used with the seed set to 1.



(a) Convergence of the greedy algorithm Algorithm B.3.

(b) Effectivity (`eff`).

Figure 3.10.: Antipodal Vivaldi Antenna: results for Test C.



(a) Convergence of Algorithm 3.1.

(b) Effectivity (`eff`).

Figure 3.11.: Antipodal Vivaldi Antenna: results for Test D.

The larger dimension of the ROM ($n = 40$) explains the slightly better performance of Algorithm B.2 over the test set compared to the ROMs obtained using Algorithms 3.1 and B.3 for which the ROM dimensions are $n = 34$ and $n = 36$, respectively.

Remark 3.7:
The choice of the error tolerance ϵ is problem dependent. In case of the dual-mode filter in Section 3.3.6.1, the problem is relatively easier to approximate since the dynamics are not so rich. So, even with a higher tolerance ($\epsilon = 10^{-6}$) one can get a ROM of small dimension ($n = 16$ or $n = 20$). However, the antipodal Vivaldi antenna shows very rich dynamics under variation of the frequency. As a result, a large number of basis vectors are required for a smaller ϵ. The choice of $\epsilon = 10^{-3}$ is chosen as a balance and it leads to a ROM of dimension $n = 34$, $n = 36$, or $n = 40$, depending on the method used. Of course, one can set an even better tolerance, such as 10^{-5}; in this case

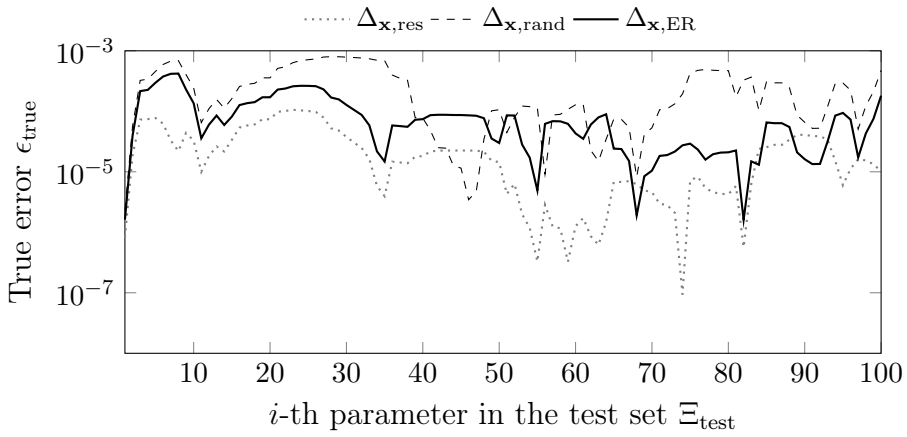


Figure 3.12.: Antipodal Vivaldi Antenna: error evaluated over a test set of parameters for ROMs obtained using different error estimators.

the resulting ROM will be of even larger size. Moreover, a tolerance of 10^{-3} is already accurate enough for many engineering applications, such as the Vivaldi antenna. ◊

3.4. A Posteriori Output Error Estimation

In this section, we discuss *a posteriori* output error estimation for linear steady/time-harmonic and dynamical systems having an output quantity of interest. Systems of this type are commonly encountered in applications such as feedback control, circuit simulation, etc. For linear dynamical systems (see Eq. (2.4)), we consider error estimation of the reduced transfer function defined in Eq. (2.15). Since $\mathbf{H}(s)$ is usually called the output response of Eq. (2.4) in the frequency domain, error estimation of $\widehat{\mathbf{H}}(\breve{\boldsymbol{\mu}})$ is also called output error estimation. For steady/time-harmonic systems in Eq. (2.10), we consider the output error estimation, i.e., $\|\mathbf{Y}(\breve{\boldsymbol{\mu}}) - \widehat{\mathbf{Y}}(\breve{\boldsymbol{\mu}})\|$ in Eq. (3.1b).

A posteriori error estimation for transfer functions has been an intensely researched topic in recent years; see [13, 23, 85, 86, 101, 154, 208]. Initial candidates for *a posteriori* error estimators were rather heuristic. In [23, 101] two types of error estimators are introduced. One was based on approximating the transfer function in two different subspaces, one richer than the other. The difference between the two approximations is used as a surrogate for the true error. Note that, this is very close to the idea of hierarchical error estimation discussed in Section 3.3.4. The second type of error estimator proposed was based on the residuals of the s-primal and s-dual systems Eqs. (2.8) and (2.9). Applying the norm of the residual as a heuristic error estimator has been proposed by other authors such as in [116]. Error bounds in the $\mathcal{H}_2$ or $\mathcal{H}_\infty$ norms were the subject in [154, 208].

Apart from its importance to certify ROMs, *a posteriori* output error estimation for reduced transfer functions is crucial for several more reasons. Firstly, for non-parametric systems, error estimators for transfer functions are used to adaptively steer the choice of interpolation points and/or number of moments to match in interpolatory MOR methods such as moment matching [82]. Secondly, for parametric systems, *a*

posteriori error estimators have been used as a part of an RBM-style greedy algorithm to automatically generate parametric ROMs; see [81, 86] and references therein. Finally, error estimation for the reduced transfer function can be applied in a straightforward manner to output error estimation of steady/time-harmonic systems.

For non-parametric systems, recall from the discussion in Chapter 2 that the choice of the interpolation points $\{s_i\}_{i=1}^{J}$ is crucial in determining how well the reduced transfer function $\widehat{\mathbf{H}}(s)$ approximates the original transfer function $\mathbf{H}(s)$. The early approaches to determine the interpolation points were mainly based on experience and were mostly heuristic. A logarithmic sampling of the interpolation points was the rule-of-thumb. An early work in [101] discussed approaches to adaptively sample interpolation points. A breakthrough in the choice of interpolation points for SISO systems was in [102], which proposed the IRKA algorithm for an 'optimal' choice interpolation points in the $\mathcal{H}_2$ norm. Extensions of IRKA to MIMO and other types of systems are available; see [12] and references therein.

For parametric systems, an RBM-style greedy algorithm has been proposed to automatically determine parametric ROMs [81]. The authors use an error estimator based on the residuals of the s-primal and s-dual systems. The error estimator is further used to choose the reduced basis vectors in order to enrich the projection basis. A drawback of this approach is the need to compute the *inf-sup* constant, whose computation involves solving large-scale eigenvalue problems at several parameters. This was remedied in the works [83, 85] where a residual-based error estimator avoiding this constant was proposed. The authors introduce an adaptive RBM-style greedy algorithm that iteratively builds the projection matrix $\mathbf{V}$. This approach is valid for both non-parametric and parametric systems and represents the state-of-the-art. However, it faces difficulties when the parameter space dimension is high or when the range of the parameters is large.

In the next subsection, we review the *a posteriori* error estimator from [83, 85] in detail and propose an extension of it. The proposed extension involves using a RBF-based data-driven surrogate model of the estimator. We detail an adaptive algorithm that: (i) adaptively builds a surrogate model by evaluating the error estimator at a few parameter samples, (ii) uses the surrogate model to explore the parameter space and enrich a training set, and (iii) adaptively enriches the projection basis $\mathbf{V}$ based on both the error estimator and its surrogate.

3.4.1. Existing Primal-Dual A Posteriori Error Estimator

We begin by briefly reviewing residual-based *a posteriori* output error estimation. Recall the s-primal system and s-dual system (Eqs. (2.8) and (2.9)) from Definitions 2.4 and 2.5. We reproduce them below for convenience.

$$\mathbf{R}\mathbf{X}_{\mathrm{pr}} = \mathbf{B},$$
$$\mathbf{R}^{\mathsf{T}}\mathbf{X}_{\mathrm{du}} = \mathbf{C}^{\mathsf{T}}.$$

Let us consider the ROMs for the above two systems in the Galerkin MOR framework. The reduced s-primal system is of the form

$$\widehat{\mathbf{R}}\widehat{\mathbf{X}}_{\mathrm{pr}} = \widehat{\mathbf{B}}, \tag{3.20}$$

where $\widehat{\mathbf{R}}$ is defined in Eq. (2.15) with $\mathbf{W} = \mathbf{V}$, i.e., $\widehat{\mathbf{R}} := \mathbf{V}^{\mathsf{T}}\mathbf{R}\mathbf{V} \in \mathbb{C}^{n\times n}$, $\widehat{\mathbf{B}}$ is defined in Eq. (2.14) with $\mathbf{W} = \mathbf{V}$, i.e., $\widehat{\mathbf{B}} := \mathbf{V}^{\mathsf{T}}\mathbf{B} \in \mathbb{R}^{n\times N_I}$ and the solution is approximated as $\mathbf{X}_{\mathrm{pr}} \approx \widetilde{\mathbf{X}}_{\mathrm{pr}} := \mathbf{V}\widehat{\mathbf{X}}_{\mathrm{pr}} \in \mathbb{C}^{N\times N_I}$. $\mathbf{V} \in \mathbb{R}^{N\times n}$ is the projection matrix that is used to derive the ROM in Eq. (2.13).

Remark 3.8:
For the sake of easy presentation, we shall base the following discussion on SISO systems ($N_I = N_O = 1$). Extension to MIMO systems is straightforward. We further remark on this when discussing the numerical examples. ◊

We denote by

$$\mathbf{r}_{\mathrm{pr}} := \mathbf{B} - \mathbf{R}\widetilde{\mathbf{x}}_{\mathrm{pr}} \in \mathbb{C}^N \tag{3.21}$$

the residual of the primal system. Analogously, the reduced s-dual system is given by

$$\widehat{\mathbf{R}}_{\mathrm{du}}^{\mathsf{T}}\widehat{\mathbf{x}}_{\mathrm{du}} = \widehat{\mathbf{C}}^{\mathsf{T}}, \tag{3.22}$$

where $\widehat{\mathbf{R}}_{\mathrm{du}} := \mathbf{V}_{\mathrm{du}}^{\mathsf{T}}\mathbf{R}\mathbf{V}_{\mathrm{du}} \in \mathbb{C}^{n\times n}$ and $\widehat{\mathbf{C}} := \mathbf{C}\mathbf{V}_{\mathrm{du}}$. The dual solution $\mathbf{x}_{\mathrm{du}}$ is approximated as $\mathbf{x}_{\mathrm{du}} \approx \widetilde{\mathbf{x}}_{\mathrm{du}} = \mathbf{V}_{\mathrm{du}}\widehat{\mathbf{x}}_{\mathrm{du}}$. $\mathbf{V}_{\mathrm{du}} \in \mathbb{R}^{N\times n}$ is the projection matrix for MOR of the dual system. Further, the residual corresponding to the s-dual systems is

$$\mathbf{r}_{\mathrm{du}} := \mathbf{C}^{\mathsf{T}} - \mathbf{R}^{\mathsf{T}}\widetilde{\mathbf{x}}_{\mathrm{du}} \in \mathbb{C}^N. \tag{3.23}$$

The following *a posteriori* error bound for the transfer function error is based on the residuals of the s-primal and s-dual systems. It reads:

$$|\mathbf{H}(\breve{\boldsymbol{\mu}}) - \widehat{\mathbf{H}}(\breve{\boldsymbol{\mu}})| \leq \frac{\|\mathbf{r}_{\mathrm{pr}}(\breve{\boldsymbol{\mu}})\|_2 \cdot \|\mathbf{r}_{\mathrm{du}}(\breve{\boldsymbol{\mu}})\|_2}{\sigma_{\min}(\mathbf{R}(\breve{\boldsymbol{\mu}}))} \tag{3.24}$$

For a proof of Eq. (3.24), see [81, 83, 101]. In the above expression, the quantity $\sigma_{\min}(\mathbf{R}(\breve{\boldsymbol{\mu}}))$ is the *inf-sup* constant mentioned earlier and for the Euclidean norm this simply turns out to be the smallest singular value of the matrix $\mathbf{R}(\breve{\boldsymbol{\mu}})$. The above error estimator is highly accurate for problems with $\sigma_{\min}$ away from zero. However, just like the standard *a posteriori* state error estimator, the cost of computing the *inf-sup* constant (involving the solution of a large-scale eigenvalue problem) is a severe restriction on using this error estimator for general parametric problems. Also, for problems whose *inf-sup* constants are zero or very close to zero, the above estimator is rather useless.

3.4.2. Inf-sup-constant-free A Posteriori Error Estimator for Dynamical Systems

A solution was proposed in the form of an *inf-sup-constant-free* output error estimator in [83]. For the s-dual system, the authors introduced the following *error-residual* (ER) system:

$$\mathbf{R}^{\mathsf{T}}\mathbf{e}_{\mathrm{du}} = \mathbf{r}_{\mathrm{du}} \tag{3.25}$$

A ROM for the dual ER system can be computed as

$$\widehat{\mathbf{R}}_e^{\mathsf{T}}\widehat{\mathbf{e}}_{\mathrm{du}} = \widehat{\mathbf{r}}_{\mathrm{du}}, \tag{3.26}$$

with $\widehat{\mathbf{R}}_e := \mathbf{V}_{\mathrm{e,du}}^{\mathsf{T}}\mathbf{R}\mathbf{V}_{\mathrm{e,du}} \in \mathbb{C}^{n_e \times n_e}$, $\widehat{\mathbf{r}}_{\mathrm{du}} := \mathbf{V}_{\mathrm{e,du}}^{\mathsf{T}}\mathbf{r}_{\mathrm{du}} \in \mathbb{C}^{n_e}$ and the variable $\widetilde{\mathbf{e}}_{\mathrm{du}} := \mathbf{V}_{\mathrm{e,du}}\widehat{\mathbf{e}}_{\mathrm{du}}$ approximates the true dual system error $\mathbf{e}_{\mathrm{du}}$. The following output error bound is proposed in [83]:

Theorem 3.9 (Inf-sup-constant-free Error Estimator [83]):
The approximation error of the reduced transfer function $\widehat{\mathbf{H}}$ can be bounded as

$$|\mathbf{H}(\breve{\boldsymbol{\mu}}) - \widehat{\mathbf{H}}(\breve{\boldsymbol{\mu}})| \leq |\mathbf{e}_{\mathrm{du}}(\breve{\boldsymbol{\mu}})^{\mathsf{T}}\mathbf{r}_{\mathrm{pr}}(\breve{\boldsymbol{\mu}})| + |\widetilde{\mathbf{x}}_{\mathrm{du}}(\breve{\boldsymbol{\mu}})^{\mathsf{T}}\mathbf{r}_{\mathrm{pr}}(\breve{\boldsymbol{\mu}})|. \tag{3.27}$$

♢

Proof. To prove the above assertion, we start from the expression for the transfer function error:

$$|\mathbf{H} - \widehat{\mathbf{H}}| = |\mathbf{C}\mathbf{R}^{-1}\mathbf{B} - \widehat{\mathbf{C}}\widehat{\mathbf{R}}^{-1}\widehat{\mathbf{B}}|.$$

By pulling out $\mathbf{C}\mathbf{R}^{-1}$ as a common factor, we get

$$|\mathbf{H} - \widehat{\mathbf{H}}| = |\mathbf{C}\mathbf{R}^{-1}\big(\mathbf{B} - \mathbf{R}\mathbf{V}\widehat{\mathbf{R}}^{-1}\widehat{\mathbf{B}}\big)|.$$

Noticing that $\widehat{\mathbf{R}}^{-1}\widehat{\mathbf{B}}$ is just the solution to Eq. (3.20) yields

$$\begin{aligned}|\mathbf{H}(\breve{\boldsymbol{\mu}}) - \widehat{\mathbf{H}}(\breve{\boldsymbol{\mu}})| = |\mathbf{C}\mathbf{R}^{-1}\big(\mathbf{B} - \mathbf{R}\mathbf{V}\widehat{\mathbf{x}}_{\mathrm{pr}}\big)| &= |\mathbf{C}\mathbf{R}^{-1}\big(\mathbf{B} - \mathbf{R}\widetilde{\mathbf{x}}_{\mathrm{pr}}\big)|,\\ &= |\mathbf{C}\mathbf{R}^{-1}\mathbf{r}_{\mathrm{pr}}|.\end{aligned}$$

Furthermore, we have

$$\begin{aligned}|\mathbf{H} - \widehat{\mathbf{H}}| - |\widetilde{\mathbf{x}}_{\mathrm{du}}^{\mathsf{T}}\mathbf{r}_{\mathrm{pr}}| &= |\mathbf{C}\mathbf{R}^{-1}\mathbf{r}_{\mathrm{pr}}| - |\widetilde{\mathbf{x}}_{\mathrm{du}}^{\mathsf{T}}\mathbf{r}_{\mathrm{pr}}|,\\ &\leq |\big(\mathbf{C}\mathbf{R}^{-1} - \widetilde{\mathbf{x}}_{\mathrm{du}}^{\mathsf{T}}\big)\mathbf{r}_{\mathrm{pr}}|\end{aligned}$$

where we have used the reverse triangle inequality and factored out $\mathbf{r}_{\mathrm{pr}}$. For the above expression, we further have

$$\begin{aligned}|\mathbf{H} - \widehat{\mathbf{H}}| - |\widetilde{\mathbf{x}}_{\mathrm{du}}^{\mathsf{T}}\mathbf{r}_{\mathrm{pr}}| &\leq |\big(\mathbf{R}^{-\mathsf{T}}\mathbf{C}^{\mathsf{T}} - \widetilde{\mathbf{x}}_{\mathrm{du}}\big)^{\mathsf{T}}\mathbf{r}_{\mathrm{pr}}|,\\ &= |\big(\mathbf{R}^{-\mathsf{T}}(\mathbf{C}^{\mathsf{T}} - \mathbf{R}^{\mathsf{T}}\widetilde{\mathbf{x}}_{\mathrm{du}})\big)^{\mathsf{T}}\mathbf{r}_{\mathrm{pr}}|,\\ &= |\big(\mathbf{R}^{-\mathsf{T}}\mathbf{r}_{\mathrm{du}}\big)^{\mathsf{T}}\mathbf{r}_{\mathrm{pr}}|.\end{aligned}$$

Finally, notice that $\mathbf{R}^{-\mathsf{T}}\mathbf{r}_{\mathrm{du}}$ is the solution to Eq. (3.25). Thus,

$$|\mathbf{H} - \widehat{\mathbf{H}}| \leq |\mathbf{e}_{\mathrm{du}}^{\mathsf{T}}\mathbf{r}_{\mathrm{pr}}| + |\widetilde{\mathbf{x}}_{\mathrm{du}}^{\mathsf{T}}\mathbf{r}_{\mathrm{pr}}|$$

which proves the assertion. □

The error bound in Eq. (3.27) does not involve the *inf-sup* constant. However, the computation of $\mathbf{e}_{\mathrm{du}}$ still involves solving the large-scale system Eq. (3.25) for each parameter and is not practical. In [83], it is proposed to approximate $\mathbf{e}_{\mathrm{du}}$ by $\widetilde{\mathbf{e}}_{\mathrm{du}}$ resulting in the following approximate error estimate:

$$|\mathbf{H} - \widehat{\mathbf{H}}| \lessapprox \big(|\widetilde{\mathbf{e}}_{\mathrm{du}}^{\mathsf{T}}\mathbf{r}_{\mathrm{pr}}| + |\widetilde{\mathbf{x}}_{\mathrm{du}}^{\mathsf{T}}\mathbf{r}_{\mathrm{pr}}|\big) =: \Delta_{\mathbf{y},\mathrm{ER}}. \tag{3.28}$$

Similar to Theorem 3.2, a sensitivity analysis of the effectivity of $\Delta_{\mathbf{y},\mathrm{ER}}$ is proposed in [85], and is given as below.

Theorem 3.10 ([85]):
The reduced transfer function error can be bounded from above and below as:

$$\Delta_{\mathbf{y},\mathrm{ER}} - \alpha_1 - \alpha_2 \leq |\mathbf{H} - \widehat{\mathbf{H}}| \leq \Delta_{\mathbf{y},\mathrm{ER}} + \gamma_1,$$

where $\gamma_1 := |(\mathbf{e}_{\mathrm{du}} - \widetilde{\mathbf{e}}_{\mathrm{du}})^{\mathsf{T}}\mathbf{r}| \geq 0$, $\alpha_1 := |(\mathbf{x}_{\mathrm{du}} - \widetilde{\mathbf{x}}_{\mathrm{du}})^{\mathsf{T}}\mathbf{r}|$ and $\alpha_2 := |\widetilde{\mathbf{e}}_{\mathrm{du}}^{\mathsf{T}}\mathbf{r}|$.

Proof. The proof of the above assertion is similar to the one in Theorem 3.2. To show the upper bound, we start with Eq. (3.27) in Theorem 3.9

$$|\mathbf{H} - \widehat{\mathbf{H}}| \leq |\mathbf{e}_{\mathrm{du}}^{\mathsf{T}}\mathbf{r}_{\mathrm{pr}}| + |\widetilde{\mathbf{x}}_{\mathrm{du}}^{\mathsf{T}}\mathbf{r}_{\mathrm{pr}}|.$$

In the above inequality, we add and subtract the term $|\widetilde{\mathbf{e}}_{\mathrm{du}}^{\mathsf{T}}\mathbf{r}|$ on the left leading to:

$$\begin{aligned}
|\mathbf{H} - \widehat{\mathbf{H}}| &\leq |\mathbf{e}_{\mathrm{du}}^{\mathsf{T}}\mathbf{r}_{\mathrm{pr}}| + |\widetilde{\mathbf{e}}_{\mathrm{du}}^{\mathsf{T}}\mathbf{r}| - |\widetilde{\mathbf{e}}_{\mathrm{du}}^{\mathsf{T}}\mathbf{r}| + |\widetilde{\mathbf{x}}_{\mathrm{du}}^{\mathsf{T}}\mathbf{r}_{\mathrm{pr}}|, \\
&= (|\widetilde{\mathbf{x}}_{\mathrm{du}}^{\mathsf{T}}\mathbf{r}_{\mathrm{pr}}| + \widetilde{\mathbf{e}}_{\mathrm{du}}^{\mathsf{T}}\mathbf{r}|) - |\widetilde{\mathbf{e}}_{\mathrm{du}}^{\mathsf{T}}\mathbf{r}| + |\mathbf{e}_{\mathrm{du}}^{\mathsf{T}}\mathbf{r}_{\mathrm{pr}}|, \\
&= \Delta_{\mathbf{y},\mathrm{ER}} + \underbrace{|(\mathbf{e}_{\mathrm{du}} - \widetilde{\mathbf{e}}_{\mathrm{du}})^{\mathsf{T}}\mathbf{r}|}_{\gamma_1}.
\end{aligned}$$

This yields the upper bound. To obtain the last inequality, we have made use of the definition of the *inf-sup-constant-free* error estimator from Eq. (3.28) and the reverse triangle inequality.

For the lower bound, we start with the expression for $\Delta_{\mathbf{y},\mathrm{ER}}$.

$$\Delta_{\mathbf{y},\mathrm{ER}} = |\widetilde{\mathbf{e}}_{\mathrm{du}}^{\mathsf{T}}\mathbf{r}_{\mathrm{pr}}| + |\widetilde{\mathbf{x}}_{\mathrm{du}}^{\mathsf{T}}\mathbf{r}_{\mathrm{pr}}|.$$

Adding and subtracting the term $|\mathbf{x}_{\mathrm{du}}^{\mathsf{T}}\mathbf{r}_{\mathrm{pr}}|$ results in

$$\begin{aligned}
\Delta_{\mathbf{y},\mathrm{ER}} &= |\widetilde{\mathbf{e}}_{\mathrm{du}}^{\mathsf{T}}\mathbf{r}_{\mathrm{pr}}| + |\widetilde{\mathbf{x}}_{\mathrm{du}}^{\mathsf{T}}\mathbf{r}_{\mathrm{pr}}| + |\mathbf{x}_{\mathrm{du}}^{\mathsf{T}}\mathbf{r}| - |\mathbf{x}_{\mathrm{du}}^{\mathsf{T}}\mathbf{r}|, \\
&= (|\widetilde{\mathbf{x}}_{\mathrm{du}}^{\mathsf{T}}\mathbf{r}_{\mathrm{pr}}| - |\mathbf{x}_{\mathrm{du}}^{\mathsf{T}}\mathbf{r}|) + |\widetilde{\mathbf{e}}_{\mathrm{du}}^{\mathsf{T}}\mathbf{r}_{\mathrm{pr}}| + |\mathbf{x}_{\mathrm{du}}^{\mathsf{T}}\mathbf{r}|.
\end{aligned}$$

It can be shown that $|\mathbf{H} - \widehat{\mathbf{H}}| = |\mathbf{x}_{\mathrm{du}}^{\mathsf{T}}\mathbf{r}|$. Using this and the reverse triangle inequality in the above expression yields

$$\Delta_{\mathbf{y},\mathrm{ER}} \leq \underbrace{|(\mathbf{x}_{\mathrm{du}} - \widetilde{\mathbf{x}}_{\mathrm{du}})^{\mathsf{T}}\mathbf{r}_{\mathrm{pr}}|}_{\alpha_1} + \underbrace{|\widetilde{\mathbf{e}}_{\mathrm{du}}^{\mathsf{T}}\mathbf{r}_{\mathrm{pr}}|}_{\alpha_2} + |\mathbf{H} - \widehat{\mathbf{H}}|.$$

resulting in the lower bound. □

3.4.3. Inf-sup-constant-free Error Estimator for Steady/Time-Harmonic Systems

An *inf-sup-constant-free* error bound similar to the one described in Theorem 3.9 is also applicable to a steady or time-harmonic system in the form of Eq. (2.10), that is,

$$\begin{aligned}
\mathbf{A}(\breve{\boldsymbol{\mu}})\mathbf{X}(\breve{\boldsymbol{\mu}}) &= \mathbf{B}(\breve{\boldsymbol{\mu}}), \\
\mathbf{Y}(\breve{\boldsymbol{\mu}}) &= \mathbf{C}(\breve{\boldsymbol{\mu}})\mathbf{X}(\breve{\boldsymbol{\mu}}).
\end{aligned}$$

This is easy to see by noting that $\mathbf{Y}(\breve{\boldsymbol{\mu}}) = \mathbf{C}(\breve{\boldsymbol{\mu}})\mathbf{A}(\breve{\boldsymbol{\mu}})^{-1}\mathbf{B}(\breve{\boldsymbol{\mu}})$ has the same form as the transfer function $\mathbf{H}(\breve{\boldsymbol{\mu}}) = \mathbf{C}(\breve{\boldsymbol{\mu}})\mathbf{R}(\breve{\boldsymbol{\mu}})^{-1}\mathbf{B}(\breve{\boldsymbol{\mu}})$. To derive the error estimator, we follow the same procedure as adopted in Sections 3.4.1 and 3.4.2. To keep the discussion simple, we derive the *inf-sup-constant-free* error estimator for a SISO system. Further, to keep the notation clean, we do not explicitly show the parameter dependence.

The corresponding ROM (see Eq. (2.16)) for the above system can be obtained via Galerkin projection with the matrix $\mathbf{V} \in \mathbb{C}^{N\times n}$ and is given by

$$\begin{aligned}\widehat{\mathbf{A}}\widehat{\mathbf{x}} &= \widehat{\mathbf{B}},\\ \widehat{\mathbf{y}} &= \widehat{\mathbf{C}}\widehat{\mathbf{x}},\end{aligned}$$

with $\widehat{\mathbf{A}} := \mathbf{V}^{\mathsf{T}}\mathbf{A}\mathbf{V} \in \mathbb{C}^{n\times n}$, $\widehat{\mathbf{B}} := \mathbf{V}^{\mathsf{T}}\mathbf{B} \in \mathbb{C}^{n}$. The approximated solution is $\widetilde{\mathbf{x}} \approx \mathbf{V}\widehat{\mathbf{x}}$. The residual is denoted by $\mathbf{r}_{\text{pr}} := \mathbf{B} - \mathbf{A}\widetilde{\mathbf{x}} \in \mathbb{C}^{N}$.

We define the dual system as

$$\mathbf{A}^{\mathsf{T}}\mathbf{x}_{\text{du}} = \mathbf{C}^{\mathsf{T}}.$$

Using the projection matrix $\mathbf{V}_{\text{du}} \in \mathbb{C}^{N\times n}$, we define the following ROM for the dual system:

$$\widehat{\mathbf{A}}_{\text{du}}^{\mathsf{T}}\widehat{\mathbf{x}}_{\text{du}} = \widehat{\mathbf{C}}^{\mathsf{T}},$$

where $\widehat{\mathbf{A}}_{\text{du}} := \mathbf{V}_{\text{du}}^{\mathsf{T}}\mathbf{A}\mathbf{V}_{\text{du}} \in \mathbb{C}^{n\times n}$, $\widehat{\mathbf{C}} := \mathbf{C}\mathbf{V}_{\text{du}} \in \mathbb{C}^{n}$. The dual solution is approximated as $\mathbf{x}_{\text{du}} \approx \widetilde{\mathbf{x}}_{\text{du}} = \mathbf{V}_{\text{du}}\widehat{\mathbf{x}}_{\text{du}}$. The residual of the dual system is $\mathbf{r}_{\text{du}} := \mathbf{C}^{\mathsf{T}} - \mathbf{A}^{\mathsf{T}}\widetilde{\mathbf{x}}_{\text{du}} \in \mathbb{C}^{N}$.

Similar to Eq. (3.25), we define the dual error-residual system as

$$\mathbf{A}^{\mathsf{T}}\mathbf{e}_{\text{du}} = \mathbf{r}_{\text{du}}. \tag{3.29}$$

A ROM for the dual error-residual system can be computed as

$$\widehat{\mathbf{A}}_{e}^{\mathsf{T}}\widehat{\mathbf{e}}_{\text{du}} = \widehat{\mathbf{r}}_{\text{du}}, \tag{3.30}$$

with $\widehat{\mathbf{A}}_{e} := \mathbf{V}_{\text{e,du}}^{\mathsf{T}}\mathbf{A}\mathbf{V}_{\text{e,du}} \in \mathbb{C}^{n_e\times n_e}$, $\widehat{\mathbf{r}}_{\text{du}} := \mathbf{V}_{\text{e,du}}^{\mathsf{T}}\mathbf{r}_{\text{du}} \in \mathbb{C}^{n_e}$ and the variable $\widetilde{\mathbf{e}}_{\text{du}} := \mathbf{V}_{\text{e,du}}\widehat{\mathbf{e}}_{\text{du}}$ approximates the true dual system error $\mathbf{e}_{\text{du}}$.

Taking up the same approach as adopted in Theorem 3.9, it can be shown that

$$|\mathbf{y}(\breve{\boldsymbol{\mu}}) - \widehat{\mathbf{y}}(\breve{\boldsymbol{\mu}})| \lessapprox \left(|\widetilde{\mathbf{e}}_{\text{du}}(\breve{\boldsymbol{\mu}})^{\mathsf{T}}\mathbf{r}_{\text{pr}}(\breve{\boldsymbol{\mu}})| + |\widetilde{\mathbf{x}}_{\text{du}}(\breve{\boldsymbol{\mu}})^{\mathsf{T}}\mathbf{r}_{\text{pr}}(\breve{\boldsymbol{\mu}})|\right) =: \Delta_{\mathbf{y}}. \tag{3.31}$$

The proof is straightforward and employs the exact same reasoning used to prove Theorem 3.9.

Computing the Inf-sup-constant-free Error Estimator We now discuss the computation of the output error estimator. The discussion is similar to what was previously mentioned for the state error estimation in Section 3.3.3.

For evaluating the *inf-sup-constant-free* error estimator Eq. (3.28), the quantities involved are $\mathbf{r}, \widetilde{\mathbf{e}}_{\text{du}}, \widetilde{\mathbf{x}}_{\text{du}}$ which can be obtained by solving the ROMs in Eqs. (3.20), (3.22) and (3.26). To this end, the projection matrices $\mathbf{V}, \mathbf{V}_{\text{du}}$ and $\mathbf{V}_{\text{e,du}}$ are required.

We now discuss the construction of $\mathbf{V}_{\text{e,du}}$. Looking at the dual ER system Eq. (3.25), we notice that

$$\begin{aligned}\mathbf{e}_{\text{du}} &= \mathbf{R}^{-\mathsf{T}}\mathbf{r}_{\text{du}},\\ &= \mathbf{R}^{-\mathsf{T}}(\mathbf{C}^{\mathsf{T}} - \mathbf{R}^{\mathsf{T}}\widetilde{\mathbf{x}}_{\text{du}}),\\ &= \mathbf{R}^{-\mathsf{T}}\mathbf{C}^{\mathsf{T}} - \mathbf{V}_{\text{du}}\widehat{\mathbf{x}}_{\text{du}}.\end{aligned}$$

Recall that $\mathbf{R}^{-\mathsf{T}}\mathbf{C}^{\mathsf{T}}$ is the true solution $\mathbf{x}_{\text{du}}$ for the s-dual system. Suppose there exists a basis $\mathbf{V}_d$ that is a good approximation to the true solution. We have

$$\mathbf{x}_{\text{du}} \approx \mathbf{V}_d\widehat{\mathbf{x}}_d.$$

Therefore,

$$\mathbf{e}_{\text{du}} \approx \mathbf{V}_d\widehat{\mathbf{x}}_d - \mathbf{V}_{\text{du}}\widehat{\mathbf{x}}_{\text{du}}.$$

Since the true error is not zero, i.e., $\mathbf{e}_{\text{du}} \neq 0$, range$(\mathbf{V}_d)$ should be different from range$(\mathbf{V}_{\text{du}})$. Hence, we set $\mathbf{V}_{\text{e,du}}$ to be a combination of $\mathbf{V}_d$ and $\mathbf{V}_{\text{du}}$, that is,

$$\mathbf{V}_{\text{e,du}} := \text{colspan}\{\mathbf{V}_{\text{du}}, \mathbf{V}_d\}. \tag{3.32}$$

Furthermore, in [85] it is proved that if $\mathbf{V}_d = \mathbf{V}_{\text{du}}$ (and hence $\mathbf{V}_{\text{e,du}} = \mathbf{V}_{\text{du}}$), the error estimator $\Delta_{\mathbf{y},\text{ER}}$ is identically zero for all $\breve{\boldsymbol{\mu}}$. In [83], an adaptive algorithm was proposed to construct the parametric ROM, making use of $\Delta_{\mathbf{y},\text{ER}}$. We discuss this approach next.

3.4.4. Greedy algorithm for ROM Construction

A greedy algorithm is proposed in [83] to adaptively construct the parametric ROM Eq. (2.13) for the linear dynamical system in Eq. (2.4). This method makes use of the *inf-sup-constant-free* error estimator in Eq. (3.28). Apart from the system matrices and the ROM tolerance ϵ, the algorithm requires as an input a suitable training set Ξ similar to the one defined in Section 3.3.5. At each iteration, the adaptive algorithm picks a parameter sample from Ξ, such that it maximizes $\Delta_{\mathbf{y},\text{ER}}$. The algorithm is sketched in Algorithm 3.2. Based on the selected greedy parameter samples, the projection matrices $\mathbf{V}, \mathbf{V}_{\text{du}}$ and $\mathbf{V}_{\text{e,du}}$ are updated iteratively (see Steps 3 - 5 in Algorithm 3.2). Any preferred MOR technique, such as BT, RBM, Multi-moment Matching, can be invoked to update the projection matrices. In [83], the authors discuss the usage of RBM and MMM. If MMM is used, the number of moments to be matched (ζ) needs to be specified as an additional input to the algorithm. In Step 5 of Algorithm 3.2, the dual error subspace is updated as a combination of the columns spans of the dual projection matrix $\mathbf{V}_{\text{du}}$ and the columns of the dual ER system projection matrix $\mathbf{V}_d$. It is critical that the greedy parameter at which $\mathbf{V}_{\breve{\boldsymbol{\mu}}^*}$ is computed, viz., $\breve{\boldsymbol{\mu}}^*$ is different from the parameter $\breve{\boldsymbol{\mu}}^e$ at which $\mathbf{V}_{\breve{\boldsymbol{\mu}}^e}$ is computed. Note that $\mathbf{V}_{\text{e,du}}$ contributes solely to the first part of the *inf-sup-constant-free* error estimator ($|\widetilde{\mathbf{e}}_{\text{du}}^{\mathsf{T}}\mathbf{r}_{\text{pr}}|$; see Eq. (3.28)). Hence, $|\widetilde{\mathbf{e}}_{\text{du}}^{\mathsf{T}}\mathbf{r}_{\text{pr}}|$ is used as an indicator in Step 10 to determine the choice of $\breve{\boldsymbol{\mu}}^e$ for the next iteration.

Algorithm 3.2: ROMGreedy-Output
Computes a ROM for linear dynamical systems Eq. (2.4) or steady/time-harmonic systems Eq. (2.10) using a greedy algorithm.

Input: System matrices: $\mathbf{E}, \mathbf{A}, \mathbf{B}, \mathbf{C}$, Training set Ξ, ROM tolerance ϵ, Maximum number of greedy iterations $iter\_max$.
Output: Reduced system matrices $\widehat{\mathbf{E}}, \widehat{\mathbf{A}}, \widehat{\mathbf{B}}, \widehat{\mathbf{C}}$.

Initialization: $\mathbf{V} = [\,]$, $\mathbf{V}_{\mathrm{du}} = [\,]$, $\mathbf{V}_{\mathrm{e,du}} = [\,]$, $err\_max = 1 + \epsilon$, $iter = 1$, initial greedy parameter $\breve{\boldsymbol{\mu}}^*$ and $\breve{\boldsymbol{\mu}}^e$ (chosen randomly from Ξ) and $\breve{\boldsymbol{\mu}}^* \neq \breve{\boldsymbol{\mu}}^e$.
while $err\_max > \epsilon$ and $iter \leq iter\_max$ **do**
  Compute matrix $\mathbf{V}_{\breve{\boldsymbol{\mu}}^*}$ using a preferred MOR method applied to the FOM Eq. (2.4) (or Eq. (2.10)); update $\mathbf{V} = \mathrm{orth}[\mathbf{V}, \mathbf{V}_{\breve{\boldsymbol{\mu}}^*}]$.
  Compute matrix $\mathbf{V}_{du,\breve{\boldsymbol{\mu}}^*}$ using a preferred MOR method applied to the dual system Eq. (2.9); update $\mathbf{V}_{\mathrm{du}} = \mathrm{orth}[\mathbf{V}_{\mathrm{du}}, \mathbf{V}_{du,\breve{\boldsymbol{\mu}}^*}]$.
  Compute matrix $\mathbf{V}_{d,\breve{\boldsymbol{\mu}}^e}$ using a preferred MOR method applied to the ER system Eq. (3.25) (or Eq. (3.29)); update $\mathbf{V}_{\mathrm{e,du}} := \mathrm{orth}[\mathbf{V}_{\mathrm{e,du}}\,, \mathbf{V}_{\mathrm{du}}\,, \mathbf{V}_{d,\breve{\boldsymbol{\mu}}^e}]$.
  $iter = iter + 1$.
  Determine reduced matrices $\widehat{\mathbf{E}}, \widehat{\mathbf{A}}, \widehat{\mathbf{B}}, \widehat{\mathbf{C}}$ through Galerkin projection using $\mathbf{V}$ as in Eq. (2.14).
  Solve the ROMs Eqs. (3.20), (3.22) and (3.26) for all $\breve{\boldsymbol{\mu}} \in \Xi$ using $\mathbf{V}, \mathbf{V}_{\mathrm{du}}, \mathbf{V}_{\mathrm{e,du}}$.
  Find $\breve{\boldsymbol{\mu}}^* = \arg\max_{\breve{\boldsymbol{\mu}} \in \Xi} \Delta_{\mathbf{y},\mathrm{ER}}(\breve{\boldsymbol{\mu}})$ or $\arg\max_{\breve{\boldsymbol{\mu}} \in \Xi} \Delta_{\mathbf{y}}(\breve{\boldsymbol{\mu}})$.
  Find $\breve{\boldsymbol{\mu}}^e = \arg\max_{\breve{\boldsymbol{\mu}} \in \Xi} |\widetilde{\mathbf{e}}_{\mathrm{du}}(\breve{\boldsymbol{\mu}})^{\mathsf{T}} \mathbf{r}_{\mathrm{pr}}(\breve{\boldsymbol{\mu}})|$.
  Set $err\_max := \Delta_{\mathbf{y},\mathrm{ER}}(\breve{\boldsymbol{\mu}}^*)$ or $\Delta_{\mathbf{y}}(\breve{\boldsymbol{\mu}}^*)$.
end

Remark 3.11:
The error estimator in Eq. (3.28) and the Algorithm 3.2 can be applied to MIMO systems with N_I inputs and N_O outputs. For MIMO systems, the transfer function $\mathbf{H}$ is matrix-valued. The entry-wise construction of the error estimator is given by:

$$|H_{ij} - \widehat{H}_{ij}| \lessapprox \left(|\widetilde{\mathbf{e}}_{\mathrm{du}}^{\mathsf{T}} \mathbf{r}_{\mathrm{pr}}| + |\widetilde{\mathbf{x}}_{\mathrm{du}}^{\mathsf{T}} \mathbf{r}_{\mathrm{pr}}|\right) =: \Delta_{\mathbf{y},\mathrm{ER}}^{ij}. \tag{3.33}$$

with the ij-th entry of both $\mathbf{H}, \widehat{\mathbf{H}}$ corresponding to the input signal at the j-th input port and the output signal at the i-th output port, with $1 \leq i \leq N_O$ and $1 \leq j \leq N_I$. The quantity $\widetilde{\mathbf{x}}_{\mathrm{du}}$ in Eq. (3.33) is the approximate solution to the s-dual system Eq. (2.9) by considering the right hand side as the i-th row vector of $\mathbf{C} \in \mathbb{R}^{N_O \times N}$, viz., $\mathbf{C}^{\mathsf{T}}(:, i)$. The residual $\mathbf{r}$ in Eq. (3.28) is obtained by considering the s-primal systems Eq. (2.8) with the right hand side being $\mathbf{B}(:, j)$, viz., the j-th column of the input matrix $\mathbf{B} \in \mathbb{R}^{N \times N_I}$. The MIMO version of the *inf-sup-constant-free* error estimator is then the maximum of $\Delta_{\mathbf{y},\mathrm{ER}}^{ij}$ in Eq. (3.33), among all i, j. That is:

$$\|\mathbf{H} - \widehat{\mathbf{H}}\|_{\max} := \max_{i,j} |H_{ij} - \widehat{H}_{ij}| \lessapprox \max_{i,j} \Delta_{\mathbf{y},\mathrm{ER}}^{i,j} =: \Delta_{\mathbf{y},\mathrm{ER}}. \tag{3.34}$$

◊

Drawbacks The efficiency and the reliability of the *inf-sup-constant-free* error estimator was demonstrated in [83, 85] on a range of benchmark problems arising in circuit simulation and microelectronic mechanical systems (MEMS). Nevertheless, it has a few drawbacks. The first is related to the extension of the *inf-sup-constant-free* error estimator to MIMO systems. As demonstrated in Remark 3.11, the error estimator needs to be computed elementwise. While this is robust, it incurs a large computational cost, especially when several inputs and/or outputs are present. A possible solution may exist in the form of *tangential interpolation* approaches such as in [72, 90]. However, this is beyond the scope of this thesis and will not be addressed further. Another drawback of the error estimator is the unknown choice of the training set and a possible need for a large number of samples in the training set. The training set Ξ for Algorithm 3.2 needs to be specified *a priori*. However, there exists no principled approach to do this. An inadequately sampled training set will result in a ROM that does not satisfy the desired tolerance. Usually, the rule-of-thumb is to take a finely sampled training set. However, this causes problems when the algorithm is applied to problems whose parameter space dimension is large ($N_p + 1 \gg 1$), or to problems whose parameters vary over a wide range of values. In both scenarios, the training set Ξ has a large number of samples.

To address this shortcoming, we propose to construct a surrogate for the *inf-sup-constant-free* error estimator using RBF. We further illustrate that computing this surrogate is cheaper than computing the error estimator. This allows for having a very finely sampled Ξ. We discuss this approach next.

3.4.5. Adaptive ROM Construction with Surrogate Error Estimator

As mentioned in the previous section, the greedy ROM construction technique Algorithm 3.2, based on the *inf-sup-constant-free* error estimator Eq. (3.28), faces some difficulties with relation to the choice of the training set Ξ and incurs large computational costs when the training set has many parameter samples. To remedy this, we propose an adaptive algorithm wherein, instead of a large training set, a coarsely-sampled training set with a small cardinality is chosen and iteratively updated with new parameter samples. Doing so ensures that the training set 'seen' by the greedy algorithm is small and hence has much less computational costs (we analyse this in detail below). At each greedy iteration, we *add* new parameter samples to the coarse training set. We also *remove* parameter samples which already satisfy the required tolerance. To add parameter samples, we propose a surrogate model for the *inf-sup-constant-free* error estimator, using the RBFs introduced in Section 2.7. Surrogate models for error estimators based on gaussian process regression and Kriging interpolation have been proposed in previous works [70, 156], in the context of the RBM applied to both steady and dynamical systems. Our approach extends this framework to frequency-domain interpolatory MOR methods. In particular, we build surrogate models of the output and transfer function approximation error estimators $\Delta_{\mathbf{y}}, \Delta_{\mathbf{y},\mathrm{ER}}$.

Remark 3.12:
A surrogate model for the state error estimator proposed in Section 3.3 can also be obtained similarly using the techniques to be proposed in the following sections. ♢

RBF Surrogate Estimator Recall that in Step 8 of Algorithm 3.2 three ROMs need to be solved for every $\breve{\boldsymbol{\mu}} \in \Xi$, in order to evaluate $\Delta_{\mathbf{y},\mathrm{ER}}$. Although this step involves solving ROMs, it can be expensive when the cardinality of the training set, i.e., $|\Xi|$ is large. To enable a cheap evaluation, we construct a *surrogate* model of $\Delta_{\mathbf{y},\mathrm{ER}}$ by learning the mapping $\Delta_{\mathbf{y},\mathrm{ER}} : \breve{\boldsymbol{\mu}} \to \mathbb{R}$ over the parameter domain. We build this surrogate model using RBFs. We consider two different training sets: a coarse training set $\Xi_c := \{\breve{\boldsymbol{\mu}}_1, \ldots, \breve{\boldsymbol{\mu}}_{n_c}\} \subseteq \mathbb{C} \times \mathcal{P}$ and a fine training set $\Xi_f := \{\mathring{\breve{\boldsymbol{\mu}}}_1, \ldots, \mathring{\breve{\boldsymbol{\mu}}}_{n_f}\} \subseteq \mathbb{C} \times \mathcal{P}$, with $n_f \gg n_c$. We use the coarse samples and the value of the error estimator evaluated at those samples as the *input-output data* pair for the RBF interpolation. More precisely, using the notations from Section 2.7, we have $\boldsymbol{\Lambda} = \Xi_c$ and $D = \{\Delta_{\mathbf{y},\mathrm{ER}}(\breve{\boldsymbol{\mu}}), \breve{\boldsymbol{\mu}} \in \Xi_c\}$. Using $(D, \boldsymbol{\Lambda})$, the linear system in Eq. (2.43) can be solved to obtain the RBF coefficients $\widetilde{\mathbf{w}}, \boldsymbol{\lambda}$.

To solve Eq. (2.43), the polynomial basis $\{\psi_j\}_{j=1}^{\nu}$ to express the side condition Eq. (2.42) and hence the matrix $\mathbf{P}$ needs to be specified. A linear polynomial is a common choice. For a linear polynomial, $\nu = N_p + 2$. We comment further on this aspect when discussing the numerical results. Note that Eq. (2.43) constitutes a small, dense system and solving it incurs costs of the order $O((\ell + \nu)^3)$ and $\ell = n_c$. However, since both n_c and ν are small, the cost remains low. Once the RBF coefficients are available, the RBF surrogate for the error estimator, $\chi(\breve{\boldsymbol{\mu}})$ in Eq. (2.40), can be evaluated for any $\breve{\boldsymbol{\mu}} \in \Xi_f$.

The pseudocode to adaptively construct the ROM using the surrogate model is given in Algorithm 3.3. Unlike Algorithm 3.2, the new approach requires two training sets Ξ_c and Ξ_f as inputs. Steps 3-7 are the same as in Algorithm 3.2. In Step 8, the ROMs are solved only for the parameters in the coarse training set. Thus, this step incurs lesser computational cost than Step 8 in Algorithm 3.2 since $|\Xi_c| \ll |\Xi|$. Step 9 determines the data-driven RBF surrogate using the samples in the coarse training set and their corresponding estimated errors as input data. In Steps 10-12 the coarse training set gets updated: we remove all parameter samples at which the estimated error is below the error tolerance ϵ. In addition, we identify n_{add} parameters from the fine training set Ξ_f which result in the largest values of the RBF error surrogate $\chi(\breve{\boldsymbol{\mu}})$. We then add these n_{add} new parameters to the coarse training set in Step 12. Finally, Steps 13-15 are similar to the last three steps from Algorithm 3.2.

Choice of kernel function for RBF The choice of the kernel function $\Phi(\|\cdot\|)$ determines the success of the resulting surrogate. Many kernels satisfying the positive definiteness (Definition 2.19) and the more relaxed conditional positive definiteness (see Definition 2.20) are known, such as inverse multiquadrics (IMQ), thin-plate splines (TPS), etc. There is no principled approach to identify the most appropriate kernel for a given problem and is rather heuristic. In the numerical experiments to be discussed, we have made use of both IMQ and TPS kernels. We have observed that IMQ kernels yield better surrogates when the error estimator $\Delta_{\mathbf{y},\mathrm{ER}}$ depends less smoothly on the parameter $\breve{\boldsymbol{\mu}}$. Also, we have noticed in our numerical tests that TPS kernels are best suited when the error estimator varies smoothly as a function of the parameter. IMQ kernels involve a hyperparameter which needs to be suitably tuned, while TPS have no such additional quantity to be determined. We use the Leave One Out Cross Validation

Algorithm 3.3: ROMGREEDYRBF
Computes a ROM for linear dynamical systems Eq. (2.4) or steady/time-harmonic systems Eq. (2.10) using a greedy algorithm and an RBF error surrogate.

Input: System matrices: $\mathbf{E}, \mathbf{A}, \mathbf{B}, \mathbf{C}$, Coarse training set Ξ_c, Fine training set Ξ_f, ROM tolerance ϵ, Maximum number of greedy iterations *iter_max*.
Output: Reduced system matrices $\widehat{\mathbf{E}}, \widehat{\mathbf{A}}, \widehat{\mathbf{B}}, \widehat{\mathbf{C}}$.

1 Initialization: $\mathbf{V} = [\,]$, $\mathbf{V}_{\text{du}} = [\,]$, $\mathbf{V}_{\text{e,du}} = [\,]$, $err\_max = 1 + \epsilon$, $iter = 1$, initial greedy parameter $\breve{\boldsymbol{\mu}}^*$ and $\breve{\boldsymbol{\mu}}^e$ (chosen randomly from Ξ_c) and $\breve{\boldsymbol{\mu}}^* \neq \breve{\boldsymbol{\mu}}^e$.
2 **while** $err\_max > \epsilon$ and $iter \leq iter\_max$ **do**
3 Compute matrix $\mathbf{V}_{\breve{\boldsymbol{\mu}}^*}$ using a preferred MOR method applied to the FOM Eq. (2.4) (or Eq. (2.10)); update $\mathbf{V} = \text{orth}[\mathbf{V}, \mathbf{V}_{\breve{\boldsymbol{\mu}}^*}]$.
4 Compute matrix $\mathbf{V}_{du,\breve{\boldsymbol{\mu}}^*}$ using a preferred MOR method to the dual system Eq. (2.9); update $\mathbf{V}_{\text{du}} = \text{orth}[\mathbf{V}_{\text{du}}, \mathbf{V}_{du,\breve{\boldsymbol{\mu}}^*}]$.
5 Compute $\mathbf{V}_{d,\breve{\boldsymbol{\mu}}^e}$ using a preferred MOR method applied to the ER system Eq. (3.25) (or Eq. (3.29)); update $\mathbf{V}_{\text{e,du}} := \text{orth}[\mathbf{V}_{\text{e,du}}\ \mathbf{V}_{\text{du}}\ \mathbf{V}_{d,\breve{\boldsymbol{\mu}}^e}]$.
6 $iter = iter + 1$.
7 Determine reduced matrices $\widehat{\mathbf{E}}, \widehat{\mathbf{A}}, \widehat{\mathbf{B}}, \widehat{\mathbf{C}}$ through Galerkin projection using $\mathbf{V}$ as in Eq. (2.14).
8 Solve the ROMs Eqs. (3.20), (3.22) and (3.26) for all $\breve{\boldsymbol{\mu}} \in \Xi_c$ using $\mathbf{V}, \mathbf{V}_{\text{du}}, \mathbf{V}_{\text{e,du}}$.
9 For $(D, \boldsymbol{\Lambda}) := (\Delta_{\mathbf{y},\text{ER}}, \Xi_c)$, determine the RBF coefficients by solving Eq. (2.43); Obtain the RBF surrogate $\chi(\breve{\boldsymbol{\mu}})$ for all $\breve{\boldsymbol{\mu}} \in \Xi_f$.
10 Update the coarse training set: Remove $\breve{\boldsymbol{\mu}} \in \Xi_c$ for which $\Delta_{\mathbf{y},\text{ER}}(\breve{\boldsymbol{\mu}}) < \epsilon$ from Ξ_c.
11 Find samples $\{\breve{\boldsymbol{\mu}}^{(1)}, \ldots, \breve{\boldsymbol{\mu}}^{(n_{\text{add}})}\} \in \Xi_f$ corresponding to the largest values of the surrogate $\chi(\breve{\boldsymbol{\mu}})$.
12 Update the coarse training set: $\Xi_c = \left[\Xi_c \cup \{\breve{\boldsymbol{\mu}}^{(1)}, \ldots, \breve{\boldsymbol{\mu}}^{(n_{\text{add}})}\}\right]$.
13 Find $\breve{\boldsymbol{\mu}}^* = \arg\max_{\breve{\boldsymbol{\mu}} \in \Xi_c} \Delta_{\mathbf{y},\text{ER}}(\breve{\boldsymbol{\mu}})$ or $\arg\max_{\breve{\boldsymbol{\mu}} \in \Xi_c} \Delta_{\mathbf{y}}(\breve{\boldsymbol{\mu}})$.
14 Find $\breve{\boldsymbol{\mu}}^e = \arg\max_{\breve{\boldsymbol{\mu}} \in \Xi_c} |\widetilde{\mathbf{e}}_{\text{du}}(\breve{\boldsymbol{\mu}})^{\mathsf{T}} \mathbf{r}(\breve{\boldsymbol{\mu}})|$.
15 Set $err\_max := \Delta_{\mathbf{y},\text{ER}}(\breve{\boldsymbol{\mu}}^*)$ or $\Delta_{\mathbf{y}}(\breve{\boldsymbol{\mu}}^*)$.
16 **end**

(LOOCV) [123, 176] method to identify the hyperparameter for the IMQ kernel. In terms of implementation, TPS kernels are more straightforward.

3.4.5.1. Computational Costs

We discuss the computational costs of the adaptive algorithm in Algorithm 3.3 and show how it is a considerable improvement over Algorithm 3.2. The major differences in Algorithm 3.3 compared to Algorithm 3.2 are: (a) The former uses the coarse training set in Step 8 to solve the ROMs and in Steps 13-14 to determine the next greedy parameters, (b) Steps 9-12 involve computing the RBF surrogate and updating of the

coarse training set. Let us first compare the costs of computing and evaluating the error estimator and the RBF surrogate in the two approaches.

- The cost of computing the error estimator $\Delta_{\mathbf{y},\mathrm{ER}}$ for all $\breve{\boldsymbol{\mu}} \in \Xi$ with cardinality n_s is:
  - Solving three small, dense ROMs (Step 8, Algorithm 3.2): $3n_s \cdot O(n^3)$,
  - Matrix-Vector product to evaluate $\mathbf{r}_{\mathrm{pr}}, \widetilde{\mathbf{e}}_{\mathrm{du}}, \widetilde{\mathbf{X}}_{\mathrm{du}}$ (Step 9, Algorithm 3.2): $3n_s \cdot O(Nn)$,
  - Inner-products in Eq. (3.28) to compute error estimator: $2n_s \cdot O(N)$,
  - Thus, the total cost is: $3n_s\big(O(n^3) + O(Nn) + \frac{2}{3}O(N)\big)$.
- The cost of computing the surrogate $\chi(\breve{\boldsymbol{\mu}})$ for all $\breve{\boldsymbol{\mu}} \in \Xi_c$ with cardinality n_c and determining $\chi(\breve{\boldsymbol{\mu}})$ for all $\breve{\boldsymbol{\mu}} \in \Xi_f$ is:
  - Solving three small, dense ROMs (Step 8, Algorithm 3.3): $3n_c \cdot O(n^3)$,
  - Matrix-Vector product to evaluate $\mathbf{r}_{\mathrm{pr}}, \widetilde{\mathbf{e}}_{\mathrm{du}}, \widetilde{\mathbf{X}}_{\mathrm{du}}$ (Step 13, Algorithm 3.3): $3n_c \cdot O(Nn)$,
  - Inner-products in Eq. (3.28) to compute error estimator: $2n_c \cdot O(N)$,
  - Solving Eq. (2.43) to obtain the RBF coefficients (Step 9, Algorithm 3.3): $O((n_c+\nu)^3)$,
  - Evaluating the RBF surrogate Eq. (2.40) for all $\breve{\boldsymbol{\mu}} \in \Xi_f$: $n_f \cdot O(n_c+\nu)$
  - Thus, the total cost is: $3n_c\big(O(n^3) + O(Nn) + \frac{2}{3}O(N)\big) + O((n_c+\nu)^3) + n_f \cdot O(n_c+\nu)$.

The quantity $(n_c+\nu)$ is small since the cardinality of the coarse training set is kept small. Moreover, since Ξ is usually finely sampled, $n_s \approx n_f \gg n_c$. Therefore, computing the error estimator for all the parameters in $\breve{\boldsymbol{\mu}} \in \Xi$ is much more expensive that computing the RBF surrogate at each iteration for all $\breve{\boldsymbol{\mu}} \in \Xi_c$.

3.4.6. Numerical Examples

In this section, we apply the proposed adaptive algorithm in Algorithm 3.3 to different benchmark examples and demonstrate its efficiency and robustness in identifying a good ROM for any parameter in the parameter domain. We also compare its performance against Algorithm 3.2 in terms of offline time taken and also the robustness of the resulting ROM.

We shall consider three examples. The first is a model of a large-scale RLC circuit while the second is a benchmark example of a microthruster device. The last example is the waveguide filter, introduced in Section 3.3.6.1. These three examples are carefully chosen to illustrate the performance of the proposed algorithm in various scenarios. The circuit example involves a wide range of values for the frequency parameter, while the microthruster is characterized by four different parameters. Finally, the waveguide filter is a MIMO system.

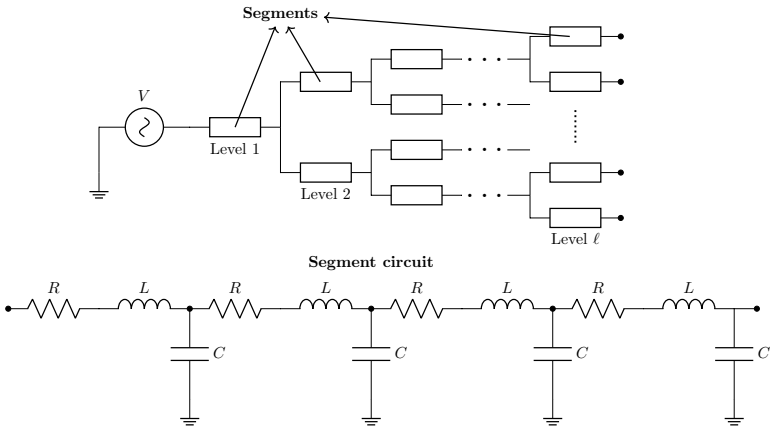


Figure 3.13.: RLC Interconnect Circuit.

To test the validity of the ROMs generated from Algorithms 3.2 and 3.3, we define a test set Ξ_{test} with n_t parameters different from those in Ξ, Ξ_c and Ξ_f. We further note that in Steps 3-5 of either algorithm, we make use of the MMM approach from Algorithm 2.1 to obtain the projection matrices. For this, the number of moments ζ to match and the deflation tolerance ϵ_{def} are provided as an additional input to the algorithms.

Next, we apply Algorithms 3.2 and 3.3 to each of the three examples and compare their respective performances. All numerical tests were performed in MATLAB®2015a, on a laptop with INTEL®CORE™i5-7200U @ 2.5 GHZ, with 8 GB of RAM.

3.4.6.1. RLC Interconnect Circuit

This example arises in large-scale interconnects in integrated circuit design. The circuit being modelled is shown in Fig. 3.13. It consists of a segment with four resistor-inductor pairs in series. The segment represents the wiring on a chip. The four capacitors model the interaction between the wire and the chip substrate. This segment can be instantiated in ℓ levels and between each level, the segment branches into two child segments leading to a total of $\sum_{i=0}^{\ell-1} 2^i$ segments for an ℓ-level circuit. The circuit is a SISO system [2, 86]. The model is constructed following modified nodal analysis. The

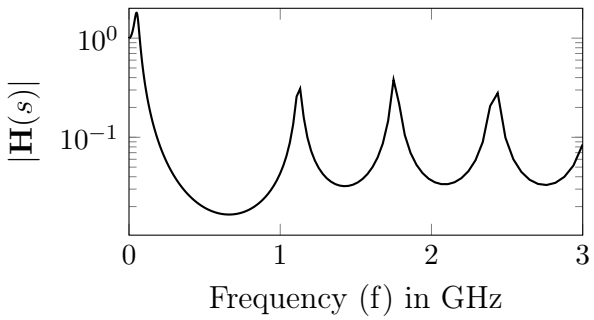
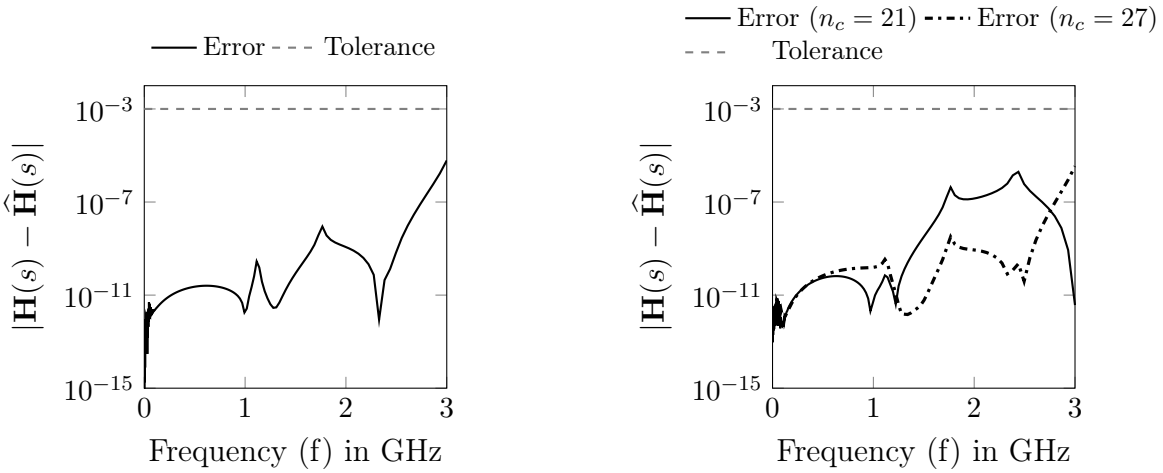


Figure 3.14.: Transfer function of the RLC Interconnect Circuit.

Table 3.1.: Simulation settings for the RLC Interconnect Circuit.

Setting	N	ϵ	$\lvert\Xi\rvert$	$\lvert\Xi_c\rvert$	$\lvert\Xi_f\rvert$	$\lvert\Xi_{\text{test}}\rvert$	ζ
Value	$6,134$	10^{-3}	90	$\{21, 27\}$	200	900	4



(a) Algorithm 3.2: Error $|\mathbf{H}(s) - \hat{\mathbf{H}}(s)|$ over the test set.

(b) Algorithm 3.3: Error $|\mathbf{H}(s) - \hat{\mathbf{H}}(s)|$ over the test set.

Figure 3.15.: Results of Test 1 and Test 2 for the RLC Interconnect Circuit.

resulting system is formulated as in Eq. (2.4) with $N = 6,134$. Its transfer function is in the form of Eq. (2.7) with $\breve{\boldsymbol{\mu}} = s$, and $f \in [0, 3]$ GHz. For this model, we are interested in transfer function simulation in the frequency domain.

We test both Algorithm 3.2 and Algorithm 3.3 on this model, using the MMM algorithm in Steps 3-5. Table 3.1 summarizes the settings for the different variables involved in each algorithm. Recall that ζ in Table 3.1 is the number of moments (including the zeroth moment) matched at each selected frequency sample; see Remark 2.14. For this example, we match the first four moments, i.e., $\zeta = 4$.

Test 1: Algorithm 3.2 applied to the RLC Interconnect Circuit This is the same test considered in [83]. The training set Ξ consists of $n_s = 90$ samples covering the frequency range of interest. The sampling is done as: $f_i := 3\times 10^{i/10}$ and $i = 1, 2, \ldots, 90$. The desired error tolerance is $\epsilon = 10^{-3}$. Algorithm 3.2 converges to the desired tolerance is 3 iterations and results in a ROM of dimension $n = 20$. We observe the average time for convergence to be 3.3 seconds, determined as the mean of 5 independent runs. To test the robustness of the resulting ROM, we solve the FOM and the ROM for the 900 samples contained in the test set Ξ_{test}.The resulting true error $|\mathbf{H}(s) - \widehat{\mathbf{H}}(s)|$ is plotted in Figure 3.15a.

Test 2: Algorithm 3.3 applied to the RLC Interconnect Circuit Having tested the efficiency of Algorithm 3.2, we apply Algorithm 3.3 to the example of the RLC Interconnect Circuit. To probe the sensitivity of the approach to the cardinality of

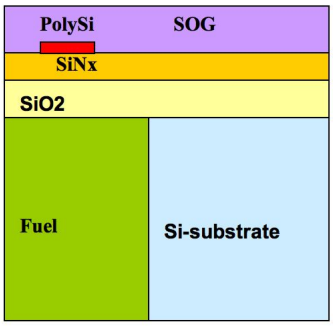


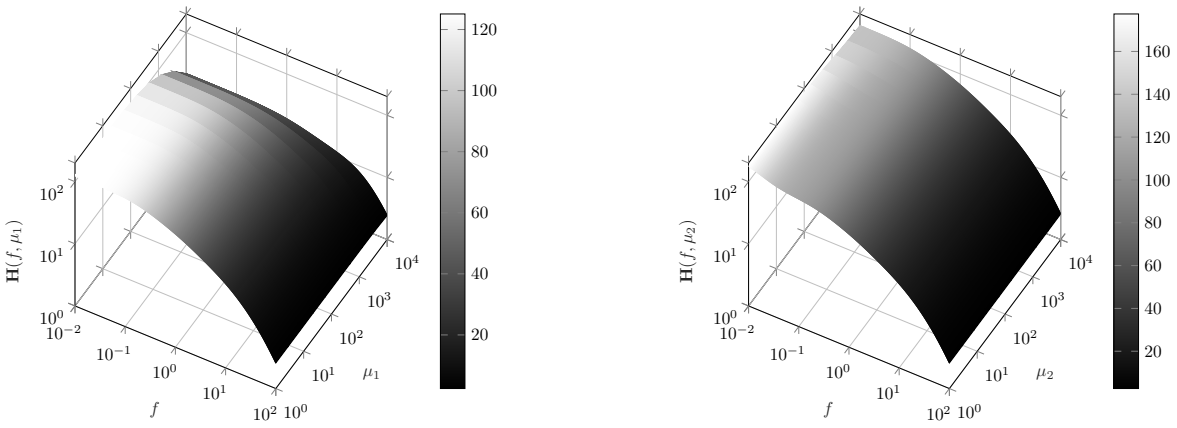
Figure 3.16.: Microthruster Unit.

the coarse training set, we consider two different values, viz., $n_c = 21, n_c = 27$. The frequency in either case is sampled as: $f_i := 3 \times 10^{(9 \cdot j)/21}$, $j = 1, 2, \ldots, 21$ and $f_i := 3 \times 10^{(9 \cdot j)/27}$, $j = 1, 2, \ldots, 27$. We also consider different samplings for the fine training set to numerically illustrate that Algorithm 3.3 is robust to the type of sampling adopted. The fine training set Ξ_f contains 200 logarithmically-spaced parameters[3] in the first case and consists of 200 parameters distributed as $f_i := 3 \times 10^{9 \cdot j/200}$, $j = 1, 2, \ldots, 200$ in the second case. For the RBF interpolation, we utilize the TPS kernel. Further, the number of new parameters n_{add} in Step 11 is set to be 1. Algorithm 3.3 converges to the specified tolerance in 3 iterations for both combinations of the coarse and fine training sets. In the first case, the ROM dimension is $n = 21$ with an average time of 1.6 seconds, while for the second case the ROM dimension was the same. However, the second case required a marginally higher time of 1.7 seconds to converge. This is not surprising since the size of the coarse training set for the second case was larger. For either case, the time taken by Algorithm 3.3 is roughly half of that required for Algorithm 3.2 to converge. Just as was done in Test 1, we test the robustness of the ROM by evaluating it over the test parameter set. The results of the true error are shown in Figure 3.15b. The ROMs obtained using different coarse and fine training set combinations are able to successfully meet the set tolerance.

3.4.6.2. Thermal Model

The second numerical example we use to illustrate the benefits from using Algorithm 3.3 is that of the heat transfer inside a microthruster unit [196]. The microthruster incorporates solid fuel within a silicon micromachined system and is used in applications such as space technology, gas generation. The goal is often to characterize the heat transfer within the microthruster device with the heat coefficients on the x-, y- and z-directions being the parameters. The heat equation with appropriate convection boundary conditions is discretized spatially using linear finite elements, resulting in a linear parametric

[3] using MATLAB® function `logspace`: $\Xi_{\text{test}} = 3 * \texttt{logspace}(0, 9, 200)$



(a) The transfer function of the thermal model $\mathbf{H}(f, \mu_1)$, $\mu_2 = \mu_3 = 10^4$.

(b) The transfer function of the thermal model $\mathbf{H}(f, \mu_2)$, $\mu_1 = 10, \mu_3 = 10^3$.

Figure 3.17.: Thermal Model.

Table 3.2.: Simulation settings for the Thermal Model.

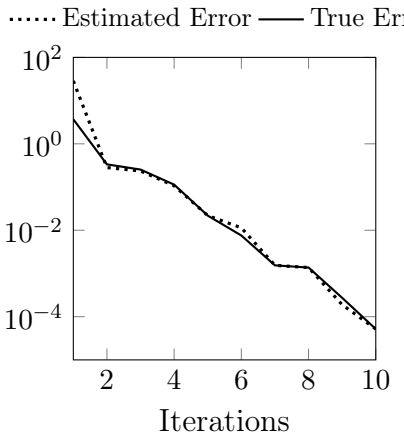
Setting	N	ϵ	$\lvert\Xi\rvert$	$\lvert\Xi_c\rvert$	$\lvert\Xi_f\rvert$	$\lvert\Xi_{\text{test}}\rvert$	ζ
Value	4257	10^{-4}	625	256	2401	1000	2

dynamical system of the form in Eq. (2.4). More specifically, we have the system

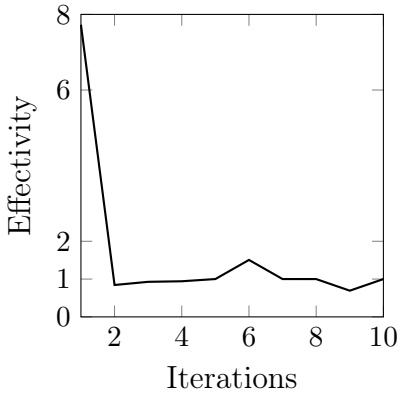
$$\mathbf{E}\frac{d}{dt}\mathbf{x}(t,\boldsymbol{\mu}) = \big(\mathbf{A}_0 - \sum_{i=1}^{3}\theta_{\mathbf{A}}^i(\boldsymbol{\mu})\mathbf{A}_i\big) + \mathbf{B}\mathbf{u}(t), \qquad \mathbf{x}(0,\boldsymbol{\mu}) = \mathbf{x}_0(\boldsymbol{\mu}),$$
$$\mathbf{y}(t,\boldsymbol{\mu}) = \mathbf{C}_7\mathbf{x}(t,\boldsymbol{\mu}).$$

where $\mathbf{E}, \mathbf{A}_0 \in \mathbb{R}^{N\times N}$ are sparse, symmetric matrices representing the heat capacity and heat conductivity, respectively. We have $N = 4,257$. Further, $\mathbf{A}_i \in \mathbb{R}^{N\times N}, i \in \{1,2,3\}$ are diagonal matrices enforcing the boundary conditions and the coefficients $\theta_{\mathbf{A}}^1, \theta_{\mathbf{A}}^2, \theta_{\mathbf{A}}^3$ are, respectively, the heat coefficients μ_1, μ_2, μ_3 along the x-, y- and z-directions. All three parameters lie in the range $[1, 10^4]$. The matrix $\mathbf{B} \in \mathbb{R}^N$ is the input matrix with $u(t) \equiv 1$ representing a constant input power of 15mW. Finally, $\mathbf{C}_7 \in \mathbb{R}^{7\times N}$ is the output matrix and measures the temperature distribution at seven different locations within the microthruster unit. For our purpose, we are interested in the average temperature distribution throughout the device. Hence, we consider $\mathbf{C} \in \mathbb{R}^{1\times N}$ as the sum of the seven rows of $\mathbf{C}_7$. The ROM Eq. (2.13) of the thermal model is obtained using either Algorithm 3.2 or Algorithm 3.3, employing the MMM method (Algorithm 2.1) with $\zeta = 2$ (zeroth and first moment are matched) in Steps 3-5. The parameter $\breve{\boldsymbol{\mu}} = [s, \mu_1, \mu_2, \mu_3]$, and $f \in [10^{-2}, 10^2]$ Hz. Figures 3.17a and 3.17b show the transfer functions of the thermal model. Table 3.2 summarizes the simulation parameters used.

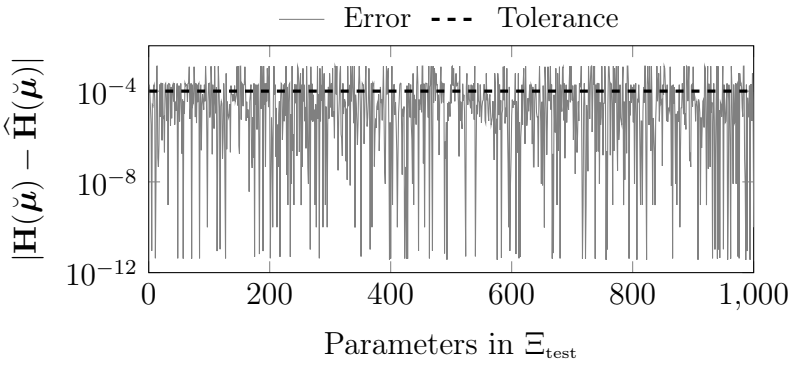
Test 3: Algorithm 3.2 applied to the Thermal Model Due to the larger range of values exhibited by the parameter, we consider a fixed training set Ξ formed as a



(a) Algorithm 3.2: Decay of $|\mathbf{H}(s) - \hat{\mathbf{H}}(s)|$ vs. $\Delta_{\mathbf{y},\mathrm{ER}}(s)$.



(b) Algorithm 3.2: Effectivity of the error estimator.

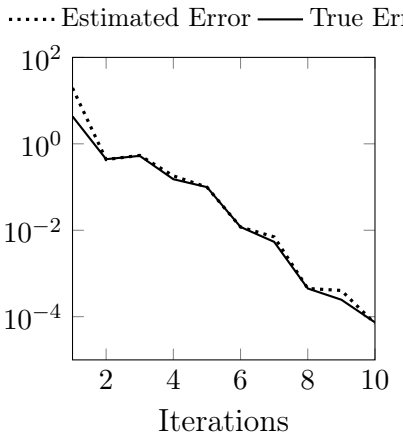


(c) Algorithm 3.2: Error over the test set.

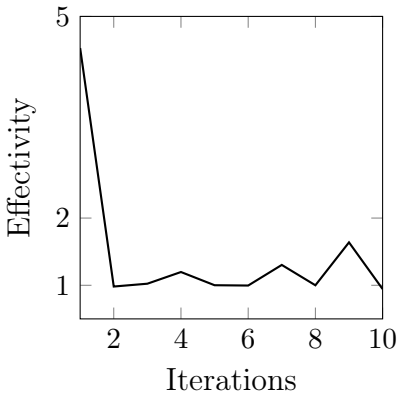
Figure 3.18.: Results of Test 3 for the Thermal model.

tensorial grid of five logarithmically-spaced samples for each of the four parameters. This leads to $n_s = 5^4 = 625$ samples in Ξ. The test set Ξ_{test} is formed by randomly sampling $n_t = 1000$ parameters from a tensorial grid having 8^4 parameter samples. This grid is formed by considering 8 logarithmically-spaced samples for the four parameters. Applying Algorithm 3.2 to this example results in a convergence to the desired tolerance $\epsilon = 10^{-4}$ in 10 iterations. The average algorithm runtime is 254s (averaged over 5 independent runs) and results in a ROM of size $n = 86$. Figures 3.18a and 3.18b show the convergence of the greedy algorithm and the effectivity of the error estimator Eq. (3.28), i.e., $\frac{\Delta_{\mathbf{y},\mathrm{ER}}(s)}{|\mathbf{H}(s)-\hat{\mathbf{H}}(s)|}$. In Figure 3.18c, we plot the error in approximating the ROM, for the parameters in the test set Ξ_{test}. As is evident, the ROM fails to meet the tolerance for several parameters. This is due to the fact that the training set used was not fine enough to capture all the variations across the parameter space.

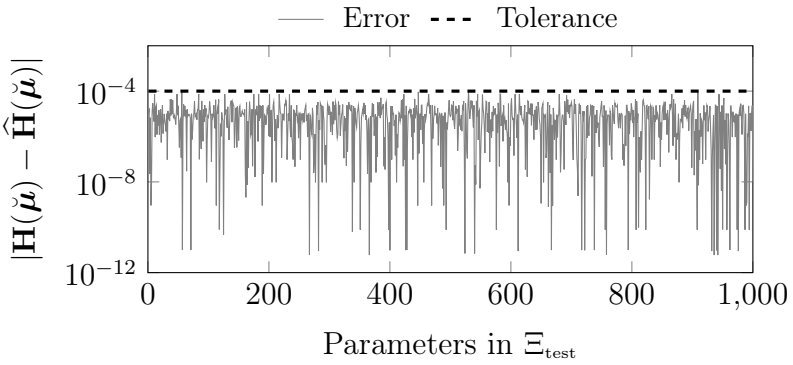
Test 4: Algorithm 3.3 applied to the Thermal Model We next apply Algorithm 3.3 to the Thermal model. For the coarse training set Ξ_c, a tensorial grid of 4^4, with four logarithmically-spaced samples per parameter is used. Similarly, the fine training set Ξ_f



(a) Algorithm 3.3: Decay of $|\mathbf{H}(s) - \hat{\mathbf{H}}(s)|$ vs. $\Delta_{\mathbf{y},\mathrm{ER}}(s)$.



(b) Algorithm 3.3: Effectivity of the error estimator.



(c) Algorithm 3.3: Error over the test set.

Figure 3.19.: Results of Test 4 for the Thermal model.

consists of 7^4 samples. For the RBF interpolation, we use TPS as the kernel function. The number of new parameters added to Ξ_c at each iteration is fixed as $n_{\mathrm{add}} = 1$. The test set Ξ_{test} is the same as the one used for Test 3. Algorithm 3.3 results in a ROM of size $n = 85$, taking 10 iterations. The average time to converge is 162 s. Figures 3.19a and 3.19b show, respectively, the convergence of the transfer function error and the effectivity of the error estimator Eq. (3.28). The ROM is validated over the test set and the error is plotted in Figure 3.19c. The ROM generated using Algorithm 3.3 requires roughly 36% lesser time and results in a uniformly good approximation over the test set.

3.4.6.3. Dual-mode Waveguide Filter

The last example we consider is that of a MIMO system arising from the model of a dual-mode circular waveguide filter shown in Figure 3.1 and the model is in the form of Eq. (2.10); details are the same as the ones in Section 3.3.6.1. The parameter is the complex Laplace variable s with the frequency $f \in [11.5, 12]$ GHz. Note that the system has two inputs and two outputs. The quantities of interest are the *scattering*

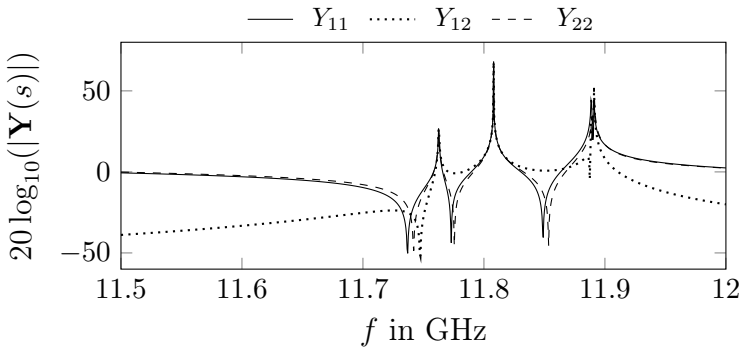


Figure 3.20.: Outputs of the Dual-mode Waveguide Filter.

Table 3.3.: Simulation settings for the Dual-mode Filter Model.

Setting	N	ϵ	$\lvert\Xi\rvert$	$\lvert\Xi_c\rvert$	$\lvert\Xi_f\rvert$	$\lvert\Xi_{\text{test}}\rvert$	ζ
Value	36,426	10^{-5}	51	17	500	101	1

parameters or *s-parameters*, which are obtained via a post-processing of the system output $\mathbf{Y}(\breve{\boldsymbol{\mu}}) := \mathbf{C}\mathbf{X}(s) \in \mathbb{C}^{2\times 2}$ and $\mathbf{C} := \mathbf{B}_0^{\mathsf{T}}$. Note that the output has the same expression as the transfer function $\mathbf{H}(\breve{\boldsymbol{\mu}})$. We denote by

$$\boldsymbol{\mathcal{S}} := \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix} \in \mathbb{C}^{2\times 2}.$$

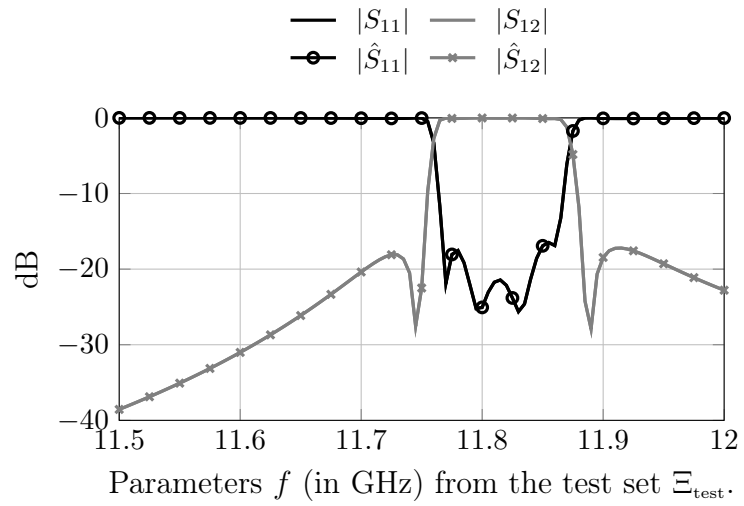
the scattering matrix for the system. The scattering matrix is symmetric, so $S_{12} = S_{21}$. In Figure 3.20, we show the output $\mathbf{Y}(s)$ for the filter.

We employ Algorithms 3.2 and 3.3 to generate ROM for the filter model, making use of the RBM in Steps 3-5[4]. Table 3.3 summarizes the simulation parameters used.

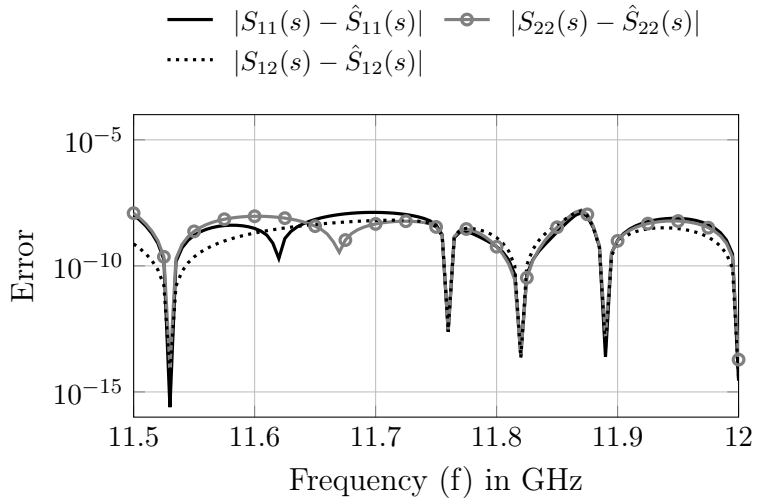
Test 5: Algorithm 3.2 applied to the Dual-mode Filter Application of Algorithm 3.2 to the filter model, using RBM at Steps 3-5 to obtain the projection matrix $\mathbf{V}$ results in a ROM of dimension $n = 10$. The training set Ξ has 51 uniformly-spaced parameters from $f \in [11.5, 12]$ GHz. The greedy algorithm converges in five iterations and requires, on average, 46 seconds to converge to the tolerance $\epsilon = 10^{-5}$. Since the system is MIMO, $\Delta_{\mathbf{y},\text{ER}}$ is computed following Eqs. (3.33) and (3.34). We plot in Fig. 3.21a the true *s-parameters* computed using the FOM solution and those obtained using the ROM, for the frequencies in the test set Ξ_{test} which contains 101 uniformly-spaced samples from the parameter domain. We note that the ROM generated by Algorithm 3.2 is a very good approximation. This is further supported by Fig. 3.21b where the errors of the components of the scattering parameter matrix are plotted.

Test 6: Algorithm 3.3 applied to the Dual-mode Filter To apply Algorithm 3.3, we first note that the filter model is a MIMO system. Therefore, in Step 9 of Algorithm 3.3, an RBF surrogate for each error estimator $\Delta^{ij}_{\mathbf{y},\text{ER}}$, $i, j \in \{1, 2\}$ in Eq. (3.33)

[4]RBM corresponds to matching the zeroth order moments at the selected frequency samples.



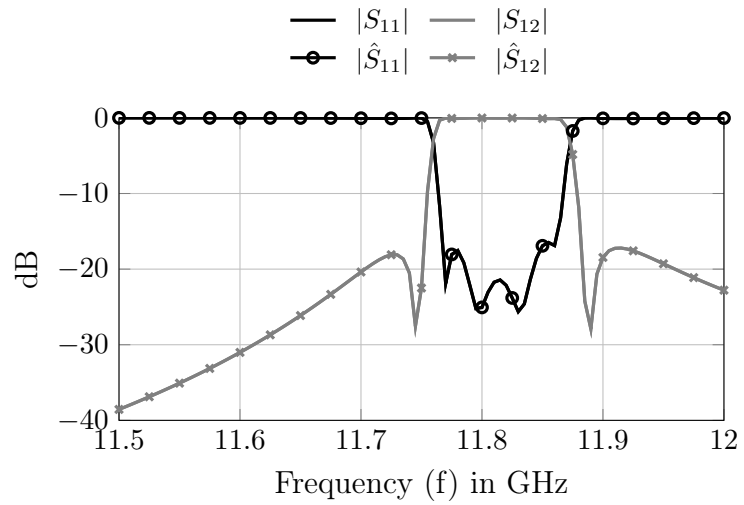
(a) Scattering parameters for the FOM and ROM evaluated on the test set.



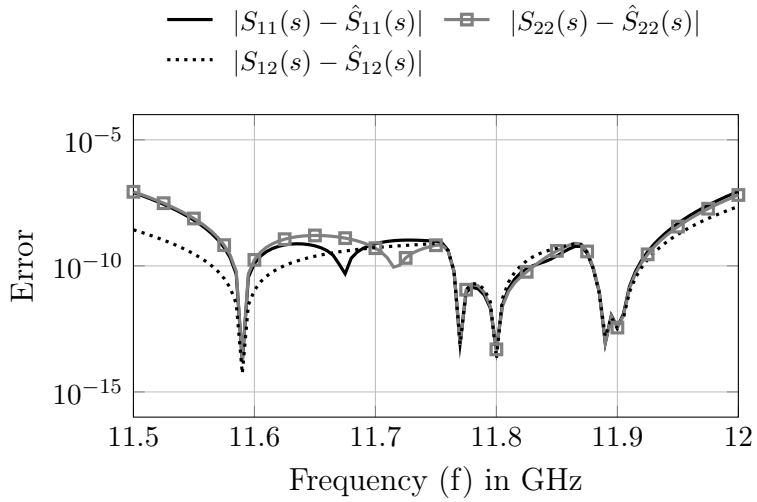
(b) Error in approximating the scattering parameters evaluated on the test set.

Figure 3.21.: Results of Test 5 for the Dual-mode Waveguide Filter.

needs to be constructed. We denote the surrogates by $\chi_{ij}(\breve{\boldsymbol{\mu}}), i, j \in \{1, 2\}$. Based on this, $\chi(\breve{\boldsymbol{\mu}})$ in Step 9 is simply defined as $\max_{i,j} \chi_{ij}(\breve{\boldsymbol{\mu}})$. For the kernel function, we use IMQ. The number of new parameters added to the coarse training set Ξ_c at each iteration is set to be $n_{\text{add}} = 1$. We pick the maximum among the four RBF surrogates and add the corresponding parameter to Ξ_c. The initial coarse training set contains 17 uniformly-spaced parameters of the frequency and the fine training set Ξ_f has 500 random parameters. Algorithm 3.3 produces a ROM of dimension $n = 10$, similar to Test 5. However, it needs only 24 s to converge, almost half that of the time take by Algorithm 3.2 in Test 5. The resulting ROM is validated by evaluating it over the test



(a) Scattering parameters for the FOM and ROM evaluated on the test set.



(b) Error in approximating the scattering parameters evaluated on the test set.

Figure 3.22.: Results of Test 6 for the Dual-mode Waveguide Filter.

set Ξ_{test}. As is clear from Figures 3.22a and 3.22b, the ROM is able to approximate the *s-parameters* very accurately. More importantly, the offline time for Algorithm 3.3 to identify a robust ROM is much smaller than that of Algorithm 3.2.

3.5. Conclusion

This chapter focussed on *a posteriori* error estimation and adaptivity for linear steady/time-harmonic and dynamical systems. We reviewed error estimation for both the state and output variables. In Section 3.3.2, we proposed a new *inf-sup-constant-free* error esti-

mator for the state variable approximation error. We further incorporated the proposed state error estimator in an adaptive algorithm (Algorithm 3.1) to generate a ROM in Section 3.3.5. In Section 3.3.6, we illustrated the efficiency and the robustness of Algorithm 3.1 using two real-life filter and antenna circuits. Turning to *a posteriori* error estimation in the output variable, we discussed existing approaches in Sections 3.4.1 and 3.4.2 and pointed out the associated drawbacks. As a remedy, we proposed a surrogate-based evaluation of the error estimator in Section 3.4.5 and used it within an adaptive algorithm to obtain parametric ROMs. Numerical tests were used in Section 3.4.6 to show the improved performance of the new surrogate-based approach.

CHAPTER 4

ERROR ESTIMATION AND ADAPTIVITY FOR NONLINEAR DYNAMICAL SYSTEMS

Contents

4.1. Introduction

In this chapter, we shift our focus to nonlinear dynamical systems. We shall consider both parametric and non-parametric systems. For non-parametric, nonlinear dynamical systems, the POD together with some hyperreduction strategy (PODEI algorithm, see Algorithm 2.7) is the most common MOR technique used, while for parametric systems, the RBM together with a suitable hyperreduction method (RBMEI, see Algorithm 2.8) is the predominant approach. To certify the accuracy of the ROMs generated either using PODEI or the RBMEI algorithm, we propose a new *a posteriori* output error

estimator, based on a modified output term in Section 4.2. We begin in Section 4.2.1 by detailing an existing state-of-the-art output error estimator and briefly mention its drawbacks. The proposed error estimator is introduced in Section 4.2.2. We discuss, at length, its efficient computation and advantages in Sections 4.2.3 and 4.2.4. Our approach involves using an *adjoint* or *dual* system to achieve tighter estimation.

The RBM has been successfully applied to a number of applications. Nevertheless in several applications where the parameter sets are large, or where the dependence of the solution on the parameter is complex, the dimension of the ROM identified by the RBM is large. Also, the computational time for the offline stage of the RBM can be significant. Therefore, adaptivity has been considered to address these issues. Actually, various notions of adaptivity have been considered for the RBM. To address the complexity of the parameter domain, adaptive sampling of the training set has been considered in a number of works [76, 105, 134]. Adaptive basis generation is discussed [24, 65, 71, 105] to address efficient enrichment of the RB projection matrix $\mathbf{V}$ and also the projection matrix $\mathbf{U}$ for hyperreduction [61]. Recent works have discussed adaptive snapshot selection to reduce the burden of computing the SVD of a large snapshot matrix [3, 28, 96].

In this thesis, we shall consider *adaptive basis enrichment* and *adaptive training set sampling*. Adaptive basis enrichment is discussed in Section 4.3. It is applicable to both non-parametric and parametric systems. Efficient adaptive algorithms to generate ROMs are proposed in Sections 4.3.5 and 4.3.6. Our basis enrichment method simultaneously enriches (a) the reduced basis, to approximate the solution manifold, and (b) the interpolation (hyperreduction) basis, for approximating the nonlinear quantity using approaches such as EIM or DEIM. To enrich either of the two bases, we make use of the proposed error estimator as a 'feedback' to determine, at each iteration, the number of new basis vectors to be added (or removed). In Section 4.3.7, numerical experiments on real-life systems arising in process engineering are used to demonstrate the reliability of the method and also the considerable savings in offline times it offers. The discussion on adaptive training set sampling is postponed to Chapter 5.

The results presented in this chapter are based on [53].

4.2. A Posteriori Output Error Estimation for Nonlinear Dynamical Systems

Error estimation is a topic of immense importance in the context of MOR in general and the RBM in particular. For parametric dynamical systems, the RBM introduced in Section 2.5.2 is a widely used MOR approach, highly dependent on efficient error estimators. In fact, it makes use of error estimation both at the offline stage and the online stage. At the offline stage, the error estimator is used to sample parameters in a greedy manner (see Algorithm 2.8). This allows for an automatic implementation of the RBM, while allowing for an efficient exploration of the parameter space at a reasonable cost. The error estimator is used at the online stage to quantify the accuracy of the ROM at any desired parameter.

The RBM community has, over the years, developed reliable, rapid error estimation

for both the state and the output variable. Most of the development was based on the representation of the PDE in the variational (or weak) form and using techniques from functional analysis to obtain guaranteed error bounds. *A posteriori* error bounds were initially proposed for linear, coercive, elliptic PDEs [177, 201] in the field variable, followed by extensions to non-coercive elliptic PDEs [186], nonlinear/non-affine parabolic PDEs [98, 100]. Some other recent works such as [198] have instead considered a combined space-time variational form of the PDE to derive error bounds for elliptic and parabolic problems [210]. We refer to the survey papers [180] and to the books [112, 167] for a deep discussion on various aspects of the RBM.

Error estimators for systems discretized using the finite volume method were considered in [106]. These error estimators were derived for the state or field variable, in the finite dimensional vector space. However, they often suffer from poor effectivity, especially for problems with a long simulation time. This is due to the accumulation of the residuals over time. Such an accumulation of the residual is a common pitfall associated with many error estimation approaches. In [214], the authors proposed a goal-oriented *a posteriori* error estimator RBM, that avoids the accumulation of the residual over time. This is achieved by considering an auxiliary dual system and its ROM, resulting in a tighter estimation of the error at each time instant. Such dual- or adjoint-based error estimation was originally introduced in [163], in the context of FEM and was popularized for error estimation of linear PDEs by the work in [97, 98].

In this work, we present an *a posteriori* output error estimator for nonlinear dynamical systems in a fully-discrete setting. Our subsequent discussion shall be based on Eq. (2.11) as the FOM for the RBM and Eq. (2.18) (or Eq. (2.36)) as the corresponding ROM. For these systems, we seek to bound/estimate the true output error at each time instant t^k and at each parameter $\boldsymbol{\mu}$ as

$$\|\mathbf{y}^k(\boldsymbol{\mu}) - \widetilde{\mathbf{y}}^k(\boldsymbol{\mu})\| \leq \Delta_{\mathbf{y}}^k(\boldsymbol{\mu}),$$

where $\Delta_{\mathbf{y}}^k(\boldsymbol{\mu}) \in \mathbb{R}_{\geq 0}$ is the estimated error at a time instance t^k for a given parameter $\boldsymbol{\mu}$. As done in Chapter 3, we drop the explicit dependence on the parameter in order to have a clean notation.

Our error estimator is closely aligned to the approach proposed in [214]. We propose two improvements to that approach and considerably improve the sharpness of the estimated error. We start by reviewing the output error estimator introduced in [214]. Following that, we highlight aspects that are ripe to be improved and propose an improved error estimator and discuss strategies for its efficient computation.

4.2.1. An Existing Approach

We discuss the procedure laid out in [214] in the context of a single output system (i.e., $\mathbf{C}(\boldsymbol{\mu}) \in \mathbb{R}^{1\times N}$ in Eq. (2.11)). The *a posteriori* output error bound is targeted at estimating the output error of the ROM for any system in the form of Eq. (2.11) (referred to hereafter as a t-primal system). We reproduce it below for easy reference:

$$\begin{aligned}\overline{\mathbf{E}}\mathbf{x}^{k+1} &= \overline{\mathbf{A}}\mathbf{x}^k + \overline{\mathbf{f}}(\mathbf{x}^k) + \overline{\mathbf{B}}\mathbf{u}(t^k),\\ \mathbf{y}^{k+1} &= \mathbf{C}\mathbf{x}^{k+1}.\end{aligned}$$

The ROM for the above t-primal system is given by Eq. (2.18) which we repeat below:

$$\begin{aligned}\widehat{\overline{\mathbf{E}}}\widehat{\mathbf{x}}^{k+1} &= \widehat{\overline{\mathbf{A}}}\widehat{\mathbf{x}}^{k} + \widehat{\overline{\mathbf{f}}}(\widehat{\mathbf{x}}^{k}) + \widehat{\overline{\mathbf{B}}}\mathbf{u}^{k},\\ \widetilde{\mathbf{y}}^{k+1} &= \widehat{\mathbf{C}}\widehat{\mathbf{x}}^{k+1}.\end{aligned}$$

The parameter dependency is not explicitly stated to avoid cluttered notation. The approximate solution to the t-primal system using the ROM is $\mathbf{x}^{k+1} \approx \widetilde{\mathbf{x}}^{k+1} = \mathbf{V}\widehat{\mathbf{x}}^{k+1}$.

The authors then introduce a discrete *adjoint* or *dual* system

$$\overline{\mathbf{E}}^{\mathsf{T}}\mathbf{x}_{\mathrm{du}}^{k+1} = -\mathbf{C}^{\mathsf{T}} \tag{4.1}$$

referred to hereafter as the t-dual system. Note that, in general, the t-dual system is time-dependent since the matrix $\overline{\mathbf{E}}(\boldsymbol{\mu})$ can vary with the time step $\Delta t^{(k)}$. Assuming a simplification in the form of constant time steps, i.e., $\Delta t^{(k)} = \Delta t$, the t-dual system becomes a steady system of the form

$$\overline{\mathbf{E}}^{\mathsf{T}}\mathbf{x}_{\mathrm{du}} = -\mathbf{C}^{\mathsf{T}}. \tag{4.2}$$

Further, the ROM corresponding to the t-dual system is given by

$$\widehat{\overline{\mathbf{E}}}^{\mathsf{T}}\widehat{\mathbf{x}}_{\mathrm{du}} = -\widehat{\mathbf{C}}^{\mathsf{T}}. \tag{4.3}$$

The approximate dual solution is $\widetilde{\mathbf{x}}_{\mathrm{du}} \approx \mathbf{V}_{\mathrm{du}}\widehat{\mathbf{x}}_{\mathrm{du}}$, with $\mathbf{V}_{\mathrm{du}} \in \mathbb{R}^{N\times n_{\mathrm{du}}}$ and n_{du} is the dimension of the reduced t-dual system.

The approximation of Eq. (2.11) and Eq. (4.2) by their corresponding ROMs results in corresponding residuals. The t-primal system residual $\mathbf{r}_{\mathrm{pr}}^{k+1} \in \mathbb{R}^N$ is defined as

$$\mathbf{r}_{\mathrm{pr}}^{k+1} := \overline{\mathbf{A}}\widetilde{\mathbf{x}}^{k} + \overline{\mathbf{f}}(\widetilde{\mathbf{x}}^{k}) + \overline{\mathbf{B}}\mathbf{u}^{k} - \overline{\mathbf{E}}\widetilde{\mathbf{x}}^{k+1}. \tag{4.4}$$

Additionally, an auxiliary residual $\check{\mathbf{r}}_{\mathrm{pr}}^{k+1} \in \mathbb{R}^N$ is obtained as the residual of Eq. (2.11) by inserting the true state vector $\mathbf{x}^k$ in the right-hand side. This results in

$$\check{\mathbf{r}}_{\mathrm{pr}}^{k+1} := \overline{\mathbf{A}}\mathbf{x}^{k} + \overline{\mathbf{f}}(\mathbf{x}^{k}) + \overline{\mathbf{B}}\mathbf{u}^{k} - \overline{\mathbf{E}}\widetilde{\mathbf{x}}^{k+1} = \overline{\mathbf{E}}\mathbf{x}^{k+1} - \overline{\mathbf{E}}\widetilde{\mathbf{x}}^{k+1}. \tag{4.5}$$

This yields a direct relation with the true error in the state vector $\mathbf{x}^{k+1} - \widetilde{\mathbf{x}}^{k+1}$ as

$$\check{\mathbf{r}}_{\mathrm{pr}}^{k+1} = \overline{\mathbf{E}}(\mathbf{x}^{k+1} - \widetilde{\mathbf{x}}^{k+1}). \tag{4.6}$$

Correspondingly, the residual of the t-dual system $\mathbf{r}_{\mathrm{du}} \in \mathbb{R}^N$ is

$$\mathbf{r}_{\mathrm{du}} := -\mathbf{C}^{\mathsf{T}} - \overline{\mathbf{E}}^{\mathsf{T}}\widetilde{\mathbf{x}}_{\mathrm{du}}. \tag{4.7}$$

Based on the primal and dual system residuals the following theorem is proved in [214]:

Theorem 4.1 ([214]):
For the t-primal system and its ROM, assuming that $\overline{\mathbf{E}}$ is invertible for every parameter $\boldsymbol{\mu}$, the true output error $\mathbf{y}^k - \widetilde{\mathbf{y}}^k$ at the k-th time instant t^k satisfies

$$\|\mathbf{y}^{k+1} - \widetilde{\mathbf{y}}^{k+1}\| \le \Phi^{k+1}\|\mathbf{r}_{\mathrm{pr}}^{k+1}\|, \qquad k = 1, 2, \ldots, K \tag{4.8}$$

with

$$\Phi^{k+1} := \varrho^k \big(\|\overline{\mathbf{E}}^{-\mathsf{T}}\| \|\mathbf{r}_{\mathrm{du}}\| + \|\widetilde{\mathbf{x}}_{\mathrm{du}}\| \big) \tag{4.9}$$

and

$$\varrho^{k+1} := \frac{\|\check{\mathbf{r}}_{\mathrm{pr}}^{k+1}\|}{\|\mathbf{r}_{\mathrm{pr}}^{k+1}\|}. \tag{4.10}$$

♢

A detailed proof can be found in [214]. There, it is also shown there that the constant ϱ^k is bounded above and below and that its value converges to 1 as the ROM dimension n increases.

While the above error bound is rigorous, it is not feasible to use as a part of the RBM POD-Greedy algorithm (Algorithm 2.4) because the true solution $\mathbf{x}^{k+1}$ is needed for evaluating $\|\check{\mathbf{r}}_{\mathrm{pr}}^{k+1}\|$. To address this, the authors estimate the value of ϱ^k based only on the data available at the snapshot locations for the current greedy parameter $\boldsymbol{\mu}^*$, i.e.,

$$\varrho^{k+1} \approx \frac{1}{K} \sum_{k=1}^{K} \varrho^{k+1}(\boldsymbol{\mu}^*) =: \tilde{\varrho}. \tag{4.11}$$

Based on this approximation, the following error estimator is proposed:

$$\|\mathbf{y}^{k+1} - \widetilde{\mathbf{y}}^{k+1}\| \lessapprox \tilde{\Phi} \|\mathbf{r}_{\mathrm{pr}}^{k+1}\| =: \widetilde{\Delta}_{\mathbf{y}}^{k+1}, \qquad k = 1, 2, \ldots, K \tag{4.12}$$

with $\tilde{\Phi} := \tilde{\varrho}\big(\|\overline{\mathbf{E}}^{-\mathsf{T}}\| \|\mathbf{r}_{\mathrm{du}}\| + \|\widetilde{\mathbf{x}}_{\mathrm{du}}\| \big)$. In [214], the above error estimator is used on two benchmark examples to demonstrate its robustness. Nevertheless, there are several issues associated with it. We elaborate on the details in the following sections.

4.2.1.1. Drawbacks

The error estimator in Eq. (4.12) consists of two parts:

$$\widetilde{\Delta}_{\mathbf{y}}^{k+1} = \tilde{\varrho}\big(\|\overline{\mathbf{E}}^{-\mathsf{T}}\| \cdot \|\mathbf{r}_{\mathrm{du}}\| \cdot \|\mathbf{r}_{\mathrm{pr}}^{k+1}\| \big) + \tilde{\varrho}\big(\|\widetilde{\mathbf{x}}_{\mathrm{du}}\| \cdot \|\mathbf{r}_{\mathrm{pr}}^{k+1}\| \big) = \widetilde{\Delta}_{\mathbf{y},1}^{k+1} + \widetilde{\Delta}_{\mathbf{y},2}^{k+1}. \tag{4.13}$$

The decay (with number of POD-Greedy iterations and ROM dimension) of $\widetilde{\Delta}_{\mathbf{y},1}^{k}$ is determined by the product of the norms of the primal and dual residuals, while that of $\widetilde{\Delta}_{\mathbf{y},2}^{k}$ depends on the product of the norm of the primal residual and the norm of the approximate dual solution $\|\widetilde{\mathbf{x}}_{\mathrm{du}}\|$.

For the first part ($\widetilde{\Delta}_{\mathbf{y},1}^{k+1}$), if both the primal and dual systems are approximated well, a quadratic convergence can be expected. In [214], the same projection matrix $\mathbf{V}$ used for obtaining the system ROM (Eq. (2.18)) is also used in obtaining the dual system ROM (Eq. (4.3)). In other words, $\mathbf{V}_{\mathrm{du}} = \mathbf{V}$. As a result, the approximation of the t-dual system is less than optimal. Furthermore, for the second part ($\widetilde{\Delta}_{\mathbf{y},2}^{k+1}$), even if $\|\mathbf{r}_{\mathrm{pr}}^{k+1}\|$ is small, there is no control over the norm of the approximated dual state vector $\|\widetilde{\mathbf{x}}_{\mathrm{du}}\|$ since that is entirely problem-dependent. Hence, the overall estimated error can potentially be large. Finally, the practical realization of the error estimator involves

computing the term $\|\overline{\mathbf{E}}(\boldsymbol{\mu})^{-\mathsf{T}}\|$ at all parameters $\boldsymbol{\mu}$ in a training set. When considering the Euclidean norm, this quantity turns out to be the smallest singular value of the matrix $\overline{\mathbf{E}}$ at a given parameter, whose computation requires solution of a large-scale eigenvalue problem. Although computed only once, this step can incur a significant portion of the overall computational cost at the offline stage especially for truly large-scale FOMs. We aim at addressing these drawbacks in our proposed error estimator, which we discuss next.

4.2.2. A New Error Bound with Modified Output

We consider the following modification of the reduced output term:

$$\overline{\mathbf{y}}^{k+1} := \widetilde{\mathbf{y}} - \widetilde{\mathbf{x}}_{\mathrm{du}}^{\mathsf{T}} \mathbf{r}_{\mathrm{pr}}^{k+1}. \tag{4.14}$$

The correction term employed is simply the product of the residual $\mathbf{r}_{\mathrm{pr}}^{k+1}$ with the approximate dual solution $\widetilde{\mathbf{x}}_{\mathrm{du}}$. Error estimators that make use of an adjoint/dual system have a long history in FEM, in the context of mesh adaptivity [163]. They have also been used previously in the context of MOR, mainly for steady-state systems [180] and linear PDEs [97]. Our proposed error estimator, to be introduced next, is targeted towards nonlinear dynamical systems.

Theorem 4.2:
Given the full-discrete FOM Eq. (2.11) and its ROM Eq. (2.18), assuming $\overline{\mathbf{E}}$ to be nonsingular at all parameters $\boldsymbol{\mu}$, the following error bound for the modified output Eq. (4.14) can be obtained:

$$\|\mathbf{y}^{k+1} - \overline{\mathbf{y}}^{k+1}\| \leq \|\overline{\mathbf{E}}^{-\mathsf{T}}\| \cdot \|\mathbf{r}_{\mathrm{du}}\| \cdot \|\mathbf{r}_{\mathrm{pr}}^{k+1}\| + \|\widetilde{\mathbf{x}}_{\mathrm{du}}\| \cdot \|\mathbf{r}_{\mathrm{pr}}^{k+1} - \check{\mathbf{r}}_{\mathrm{pr}}^{k+1}\|. \tag{4.15}$$

Proof. The broad scheme of the proof follows the structure of the proof in [214]. The error in the modified output is

$$\mathbf{y}^{k+1} - \overline{\mathbf{y}}^{k+1} = \mathbf{C}\big(\mathbf{x}^{k+1} - \widetilde{\mathbf{x}}^{k+1}\big) + \widetilde{\mathbf{x}}_{\mathrm{du}}^{\mathsf{T}} \mathbf{r}_{\mathrm{pr}}^{k+1}. \tag{4.16}$$

Multiplying by $\big(\mathbf{x}^{k+1} - \widetilde{\mathbf{x}}^{k+1}\big)^{\mathsf{T}}$ on both sides of Eq. (4.2) yields

$$\big(\mathbf{x}^{k+1} - \widetilde{\mathbf{x}}^{k+1}\big)^{\mathsf{T}} \overline{\mathbf{E}}^{\mathsf{T}} \mathbf{x}_{\mathrm{du}} = -\big(\mathbf{x}^{k+1} - \widetilde{\mathbf{x}}^{k+1}\big)^{\mathsf{T}} \mathbf{C}^{\mathsf{T}},$$

so that

$$\mathbf{x}_{\mathrm{du}}^{\mathsf{T}} \overline{\mathbf{E}} \big(\mathbf{x}^{k+1} - \widetilde{\mathbf{x}}^{k+1}\big) = -\mathbf{C}\big(\mathbf{x}^{k+1} - \widetilde{\mathbf{x}}^{k+1}\big),$$

where we have simply used the transpose to obtain the second equation. We further use the expression in Eq. (4.6) to obtain

$$\mathbf{x}_{\mathrm{du}}^{\mathsf{T}} \check{\mathbf{r}}_{\mathrm{pr}}^{k+1} = -\mathbf{C}\big(\mathbf{x}^{k+1} - \widetilde{\mathbf{x}}^{k+1}\big). \tag{4.17}$$

Now, we substitute Eq. (4.17) in Eq. (4.16) followed by addition and subtraction of the term $\widetilde{\mathbf{x}}_{\mathrm{du}}^{\mathsf{T}}\check{\mathbf{r}}_{\mathrm{pr}}^{k+1}$ to get

$$\begin{aligned}\mathbf{y}^{k+1}-\overline{\mathbf{y}}^{k+1} &= -\mathbf{x}_{\mathrm{du}}^{\mathsf{T}}\check{\mathbf{r}}_{\mathrm{pr}}^{k+1}+\widetilde{\mathbf{x}}_{\mathrm{du}}^{\mathsf{T}}\mathbf{r}_{\mathrm{pr}}^{k+1},\\ &= -\mathbf{x}_{\mathrm{du}}^{\mathsf{T}}\check{\mathbf{r}}_{\mathrm{pr}}^{k+1}+\widetilde{\mathbf{x}}_{\mathrm{du}}^{\mathsf{T}}\mathbf{r}_{\mathrm{pr}}^{k+1}+\widetilde{\mathbf{x}}_{\mathrm{du}}^{\mathsf{T}}\check{\mathbf{r}}_{\mathrm{pr}}^{k+1}-\widetilde{\mathbf{x}}_{\mathrm{du}}^{\mathsf{T}}\check{\mathbf{r}}_{\mathrm{pr}}^{k+1},\\ &= -\big(\mathbf{x}_{\mathrm{du}}-\widetilde{\mathbf{x}}_{\mathrm{du}}\big)^{\mathsf{T}}\check{\mathbf{r}}_{\mathrm{pr}}^{k+1}+\widetilde{\mathbf{x}}_{\mathrm{du}}^{\mathsf{T}}\big(\mathbf{r}_{\mathrm{pr}}^{k+1}-\check{\mathbf{r}}_{\mathrm{pr}}^{k+1}\big).\end{aligned} \tag{4.18}$$

Next, from Eq. (4.2) and Eq. (4.7). We get

$$\begin{aligned}&\mathbf{r}_{\mathrm{du}}=\overline{\mathbf{E}}^{\mathsf{T}}\mathbf{x}_{\mathrm{du}}-\overline{\mathbf{E}}^{\mathsf{T}}\widetilde{\mathbf{x}}_{\mathrm{du}}=\overline{\mathbf{E}}^{\mathsf{T}}\big(\mathbf{x}_{\mathrm{du}}-\widetilde{\mathbf{x}}_{\mathrm{du}}\big),\\ &\implies\big(\mathbf{x}_{\mathrm{du}}-\widetilde{\mathbf{x}}_{\mathrm{du}}\big)=\overline{\mathbf{E}}^{-\mathsf{T}}\mathbf{r}_{\mathrm{du}}.\end{aligned} \tag{4.19}$$

Substituting Eq. (4.19) into Eq. (4.18) yields

$$\mathbf{y}^{k+1}-\overline{\mathbf{y}}^{k+1}=-\mathbf{r}_{\mathrm{du}}^{\mathsf{T}}\overline{\mathbf{E}}^{-1}\check{\mathbf{r}}_{\mathrm{pr}}^{k+1}+\widetilde{\mathbf{x}}_{\mathrm{du}}^{\mathsf{T}}\big(\mathbf{r}_{\mathrm{pr}}^{k+1}-\check{\mathbf{r}}_{\mathrm{pr}}^{k+1}\big). \tag{4.20}$$

Taking the norm on either sides and using the triangle and Cauchy-Schwartz inequalities we obtain the error bound as

$$|\mathbf{y}^{k+1}-\overline{\mathbf{y}}^{k+1}| = \|-\mathbf{r}_{\mathrm{du}}^{\mathsf{T}}\overline{\mathbf{E}}^{-1}\check{\mathbf{r}}_{\mathrm{pr}}^{k+1}\|+\|\widetilde{\mathbf{x}}_{\mathrm{du}}^{\mathsf{T}}\big(\mathbf{r}_{\mathrm{pr}}^{k+1}-\check{\mathbf{r}}_{\mathrm{pr}}^{k+1}\big)\|, \tag{4.21}$$

$$\leq \|\overline{\mathbf{E}}^{-1}\|\|\mathbf{r}_{\mathrm{du}}\|\|\check{\mathbf{r}}_{\mathrm{pr}}^{k+1}\|+\|\widetilde{\mathbf{x}}_{\mathrm{du}}\|\|\big(\mathbf{r}_{\mathrm{pr}}^{k+1}-\check{\mathbf{r}}_{\mathrm{pr}}^{k+1}\big)\|. \tag{4.22}$$

□

Notice that, similar to the error bound of [214], the above error bound cannot be used for practical computations owing to the presence of the term $\|\check{\mathbf{r}}_{\mathrm{pr}}^{k+1}\|$ which needs the true solution. Therefore, instead of a rigorous upper bound, we propose the following computable error estimator.

4.2.3. A Computable Error Estimator

We define $\alpha := \|\mathbf{r}_{\mathrm{pr}}^{k+1}-\check{\mathbf{r}}_{\mathrm{pr}}^{k+1}\|$. The upper bound for α based on the triangle inequality is $\|\mathbf{r}_{\mathrm{pr}}^{k+1}\|+\|\check{\mathbf{r}}_{\mathrm{pr}}^{k+1}\| =: \alpha_u$. In addition, based on the reverse triangle inequality, we have a lower bound for α as $\big|\|\mathbf{r}_{\mathrm{pr}}^{k+1}\|-\|\check{\mathbf{r}}_{\mathrm{pr}}^{k+1}\|\big| =: \alpha_l$. We propose to approximate α using the lower bound, that is,

$$\|\mathbf{r}_{\mathrm{pr}}^{k+1}-\check{\mathbf{r}}_{\mathrm{pr}}^{k+1}\| \approx \big|\|\mathbf{r}_{\mathrm{pr}}^{k+1}\|-\|\check{\mathbf{r}}_{\mathrm{pr}}^{k+1}\|\big|.$$

The above proposal is motivated by the following observation. Notice that, in case we approximate α by the upper bound α_u, the second part of Equation (4.15) has the upper bound $(1+\tilde{\varrho})\|\widetilde{\mathbf{x}}_{\mathrm{du}}\|\|\mathbf{r}_{\mathrm{pr}}^{k+1}\|$ and this quantity which is even larger than $\Delta_{\mathbf{y},2}$ in Eq. (4.13). Additionally, note that

$$|\alpha_l-\alpha| \leq \begin{cases}2\varrho^{k+1}\|\mathbf{r}_{\mathrm{pr}}^{k+1}\|, & \varrho^{k+1}>1,\\ 2\|\mathbf{r}_{\mathrm{pr}}^{k+1}\|, & \varrho^{k+1}\leq 1,\end{cases} \tag{4.23}$$

while

$$|\alpha_u - \alpha| \leq (2 + 2\varrho^{k+1})\|\mathbf{r}_{\mathrm{pr}}^{k+1}\|. \tag{4.24}$$

Based on these observations, we conclude that $|\alpha_l - \alpha|$ possesses a *smaller* upper bound than $|\alpha_u - \alpha|$, implying that α could be better approximated by α_l than α_u. We use this to obtain the following expression

$$\begin{aligned} |\mathbf{y}^{k+1} - \overline{\mathbf{y}}^{k+1}| &\leq \| - \mathbf{r}_{\mathrm{du}}^{\mathsf{T}} \overline{\mathbf{E}}^{-1} \check{\mathbf{r}}_{\mathrm{pr}}^{k+1}\| + \|\widetilde{\mathbf{x}}_{\mathrm{du}}^{\mathsf{T}}\big(\mathbf{r}_{\mathrm{pr}}^{k+1} - \check{\mathbf{r}}_{\mathrm{pr}}^{k+1}\big)\|, \\ &\lessapprox \|\overline{\mathbf{E}}^{-1}\|\|\mathbf{r}_{\mathrm{du}}\|\|\check{\mathbf{r}}_{\mathrm{pr}}^{k+1}\| + \|\widetilde{\mathbf{x}}_{\mathrm{du}}\|\left|\|\mathbf{r}_{\mathrm{pr}}^{k+1}\| - \|\check{\mathbf{r}}_{\mathrm{pr}}^{k+1}\|\right|, \\ &= \big(\varrho^{k+1}\|\overline{\mathbf{E}}^{-1}\|\|\mathbf{r}_{\mathrm{du}}\| + |1 - \varrho^{k+1}|\|\widetilde{\mathbf{x}}_{\mathrm{du}}\|\big)\|\mathbf{r}_{\mathrm{pr}}^{k+1}\|, \\ &\approx \big(\tilde{\varrho}\|\overline{\mathbf{E}}^{-1}\|\|\mathbf{r}_{\mathrm{du}}\| + |1 - \tilde{\varrho}|\|\widetilde{\mathbf{x}}_{\mathrm{du}}\|\big)\|\mathbf{r}_{\mathrm{pr}}^{k+1}\|. \end{aligned} \tag{4.25}$$

We have used Eq. (4.10) to substitute for $\|\check{\mathbf{r}}_{\mathrm{pr}}^{k+1}\|$ and ϱ^{k+1} is approximated using $\tilde{\varrho}$ in Eq. (4.11). Finally, defining

$$\overline{\Phi} := \tilde{\varrho}\|\overline{\mathbf{E}}^{-1}\|\|\mathbf{r}_{\mathrm{du}}\| + |1 - \tilde{\varrho}|\|\widetilde{\mathbf{x}}_{\mathrm{du}}\| \tag{4.26}$$

results in the following error estimator:

$$\|\mathbf{y}^{k+1} - \overline{\mathbf{y}}^{k+1}\| \lessapprox \overline{\Phi}\|\mathbf{r}_{\mathrm{pr}}^{k+1}\| =: \Delta_{\mathbf{y}}^{k+1}, \qquad k = 1, 2, \ldots, K. \tag{4.27}$$

and $\Delta_{\mathbf{y}}^{k+1} = \Delta_{\mathbf{y},1}^{k+1} + \Delta_{\mathbf{y},2}^{k+1} = \tilde{\varrho}\|\overline{\mathbf{E}}^{-1}\|\|\mathbf{r}_{\mathrm{du}}\|\|\mathbf{r}_{\mathrm{pr}}^{k+1}\| + |1 - \tilde{\varrho}|\|\widetilde{\mathbf{x}}_{\mathrm{du}}\|\|\mathbf{r}_{\mathrm{pr}}^{k+1}\|$. We next highlight various aspects showing the advantages of the proposed error estimator Eq. (4.27) over $\widetilde{\Delta}_{\mathbf{y}}^{k+1}$ in Eq. (4.12).

4.2.4. Advantages of the Proposed Approach

Improving $\widetilde{\Delta}_{\mathbf{y},2}^{k+1}$ Compared to the error estimator Eq. (4.8) in [214], notice that for the proposed error estimator $\Delta_{\mathbf{y}}^{k+1}$, the second term is multiplied by the coefficient $|1 - \tilde{\varrho}|$. The quantity $\tilde{\varrho}$ tends to be 1 as the POD-Greedy algorithm converges, therefore $\Delta_{\mathbf{y},2}^{k+1}$ tends to be 0, resulting in tighter estimation of the error.

Improving $\widetilde{\Delta}_{\mathbf{y},1}^{k+1}$ For the error estimator $\widetilde{\Delta}_{\mathbf{y}}^{k+1}$ in Eq. (4.12), the first term $\widetilde{\Delta}_{\mathbf{y},1}^{k+1}$ contains the term $\|\mathbf{r}_{\mathrm{du}}\|$. This quantity is obtained by reducing the dual system (Eq. (4.2)) using the same projection matrix used to reduce the t-primal system, leading to a slow decay of $\widetilde{\Delta}_{\mathbf{y},1}^{k+1}$. To ensure a faster decay of the first term, we propose to make use of appropriate methods or projection matrices to compute the approximate dual solution $\widetilde{\mathbf{x}}_{\mathrm{du}}$. This is motivated by the observation that $\widetilde{\mathbf{x}}_{\mathrm{du}}$ need not necessarily be computed by reducing the t-dual system using a MOR method – any method that is able to approximate $\mathbf{x}_{\mathrm{du}}$ should be applicable. For the case of the dual system being parametric, we construct a separate dual RB space as considered in [97, 180]. Using this, the t-dual system can be much better approximated than using the primal RB matrix. Consequently, the approximate solution $\widetilde{\mathbf{x}}_{\mathrm{du}}$ will lead to a residual with a smaller norm, thus improving the decay rate of the first term $\Delta_{\mathbf{y},1}^{k+1}$. Naturally, this approach has

a higher computational cost. In case the dual system is non-parametric (when $\overline{\mathbf{E}}$ is non-parametric and time-independent), instead of reducing the dual system, we propose to use Krylov-space methods (such as GMRES, MINRES) [152, 181] to iteratively solve the t-dual system and obtain $\widetilde{\mathbf{x}}_{\text{du}}$. This procedure is done only once and leads to a smaller $\|\mathbf{r}_{\text{du}}\|$ compared to using the primal projection matrix to reduce the dual system.

Efficiently computing the inf-sup constant In Eqs. (4.12) and (4.27), $\|\overline{\mathbf{E}}(\boldsymbol{\mu})^{-1}\|$ needs to be computed at all parameters $\boldsymbol{\mu}$ in some training set. Actually, this quantity is the so-called *inf-sup* constant. In case of using the Euclidean norm (the matrix spectral norm) $\|\cdot\|_2$,

$$\|\overline{\mathbf{E}}(\boldsymbol{\mu})^{-1}\|_2 = \sigma_{\max}\big(\overline{\mathbf{E}}(\boldsymbol{\mu})^{-1}\big) = \frac{1}{\sigma_{\min}\big(\overline{\mathbf{E}}(\boldsymbol{\mu})\big)}$$

where $\sigma_{\min}$ is the smallest singular value of the matrix $\overline{\mathbf{E}}(\boldsymbol{\mu})$. For the RBM, the training set usually contains many parameters, thus, obtaining $\|\overline{\mathbf{E}}(\boldsymbol{\mu})^{-1}\|$ is expensive. The Successive Constraints Method (SCM) [119, 120] is an approach proposed to compute $\|\overline{\mathbf{E}}(\boldsymbol{\mu})^{-1}\|$ more efficiently. However, the SCM is known to suffer from slow convergence [188]. In [111], the *inf-sup* constant is approximated using Kriging interpolation [150]. Instead, we use an RBF-based interpolation to approximate the smallest singular values. This approach was originally introduced in [137] and it avoids the slow convergence rates seen in the SCM, while reducing the computational costs drastically.

4.2.5. Radial Basis Interpolation for the inf-sup constant

Recall from the discussion in Section 2.7 that RBFs are good choice to perform interpolation of scattered data in high-dimensional spaces. The training set Ξ in the RBM is usually densely-sampled, making it a large computational burden to evaluate $\sigma_{\min}$ for all samples in Ξ. To this end, we seek an RBF interpolant of the function $\sigma_{\min} : \mathbb{R}^{N_p} \to \mathbb{R}_{\geq 0}$, which can be cheaply evaluated. Our approach closely follows the procedure laid out in [137]. We start by considering a coarsely-sampled subset $\Xi_c := \{\boldsymbol{\mu}_1^c, \ldots, \boldsymbol{\mu}_{n_c}^c\} \subset \Xi$. We then solve the large-scale eigenvalue problem to determine the smallest singular values $\sigma_{\min}\big(\overline{\mathbf{E}}(\boldsymbol{\mu})\big)$ for all $\boldsymbol{\mu} \in \Xi_c$ and form the RBF interpolant for $\sigma_{\min}$. To achieve this, using the notations from Section 2.7, we then set $\boldsymbol{\Lambda} : \Xi_c$ and $D := \{\sigma_{\min}\big(\overline{\mathbf{E}}(\boldsymbol{\mu})\big), \boldsymbol{\mu} \in \Xi_c\}$. Using $(D, \boldsymbol{\Lambda})$, the RBF interpolant can be obtained by solving the linear system Eq. (2.43). Next, we follow a greedy procedure to enrich the coarse training set Ξ_c with new parameters from Ξ and update the RBF interpolant. The new parameter at each iteration is chosen as the one maximizing a pre-defined *criterion function* $\mathfrak{C}$ defined over Ξ. This function is chosen such that it promotes adding new samples in locations with highly varying response and also ensures the positivity of the interpolant. At the end of each greedy iteration, the relative error in the L^∞-norm between the current and the previous RBF interpolant is computed and this serves as the termination criterion. In the following algorithms, the RBF approximation $\widetilde{\sigma}_{\min}$ will replace $\sigma_{\min}$.

4.3. Adaptive Basis Enrichment

Now, we make use of the improved *a posteriori* error estimator in Section 4.2.2 to propose an adaptive scheme for basis enrichment. This scheme is applicable to both non-parametric and parametric systems.

4.3.1. Non-Parametric Systems

In the non-parametric case, the PODEI algorithm (Algorithm 2.7) is the state-of-the-art. However, this approach does not guarantee that the dimensions of the POD basis $\mathbf{V}$ or the hyperreduction basis $\mathbf{U}$ is as small as possible. In fact, in the worst case, the ROM can be unstable; see [87, 93]. This is due to the fact that the two bases are generated separately (Steps 2-3 of Algorithm 2.7). The choice of $\epsilon_{\text{POD}}, \epsilon_{\text{EI}}$ (or n, n_{EI}) is heuristic and uncorrelated to the actual output error. To address this, [87] introduced an adaptive enrichment of the bases $\mathbf{V}, \mathbf{U}$. The adaptivity was guided by an *a posteriori* error estimator. The method in [87] has several drawbacks, which we seek to improve. Firstly, it is applicable exclusively to non-parametric systems. Secondly, the adaptive scheme is *one-way*: the bases can only be extended with new basis vectors but can not be shrunk when needed. This may restrict the initial number of basis vectors in $\mathbf{V}, \mathbf{U}$ to be small and could result in many iterations to converge. Thirdly, the *a posteriori* error estimator used in [87] is the same as the one from [214] which is not optimally sharp (see Section 4.2.1.1) and could be further improved. As a consequence, a tighter error estimator may also lead to fewer number of iterations.

Our proposed extension to the PODEI algorithm is called the Two-way Adaptive PODEI algorithm. It is, in fact, a non-trivial extension of the ideas proposed in [87]. In terms of the enhancements, we make use of the improved *a posteriori* error estimator proposed in Section 4.2.2. The resulting tight estimation of the error allows for obtaining ideally small ROMs. Furthermore, we make the adaptive enrichment a *two-way* process – we allow for both addition and removal of basis vectors. This removes any limitation on the initial dimensions of the $\mathbf{V}, \mathbf{U}$, an issue faced in [87].

4.3.2. Parametric Systems

In the parametric setting, the RBMEI algorithm (Algorithm 2.8) represents the predominant approach to obtain parametric ROMs and it suffers from similar issues as the ones highlighted for the PODEI algorithm. In this form, this algorithm incurs large computational costs at the offline stage due to Step 1, where the FOM needs to be solved for every parameter in the training set Ξ in order to collect the snapshots of the nonlinear term. Moreover, the number of basis vectors in $\mathbf{U}$ is determined through a heuristic criterion of singular value decay in case of DEIM (see Eq. (2.26)), or through a greedy procedure in the case of EIM (see Algorithm 2.5). This often leads to an unnecessarily large dimension of the EI basis, since it is based on an artificial *singular value error measure* (for DEIM) or *interpolation error* (for EIM) and is not related to the approximation quality for the problem being considered. Apart from this, the dimension of the RB $\mathbf{V}$, viz., r_{RB} is determined heuristically (Steps 7, 10). The common choices are to simply set $r_{\text{RB}} = 1$ or to determine it using some tolerance ϵ_{POD} in

Algorithm 2.2. These shortcomings were also pointed out in [24, 65, 71].

In [65], the authors introduce the *simultaneous EIM-RB* (SER) algorithm for simultaneously enriching the RB and EIM basis for nonlinear, steady systems. The goal is to enrich the EIM and RB bases alternately, avoiding the computation of expensive FOM solutions for all the parameters in a given training set. At the first step, a FOM solution at a randomly chosen parameter is obtained. Based on this, the EIM basis $\mathbf{U}$ and interpolation points are computed. Furthermore, the first RB basis is built by orthogonalizing the available snapshot from the FOM solution. In the subsequent steps, the EIM and RB bases are enriched alternately, relying only on the approximate solutions computed from the ROM simulation. In essence, the approach requires only an initial FOM simulation at a single parameter. Since only the ROM is simulated for EIM and RB updates, the snapshots used for EIM and RB construction are approximate snapshots.

In [24], the *progressive EIM* (PREIM) algorithm is proposed as an extension of SER to time-dependent problems. The PREIM algorithm computes the nonlinear term based not only on the approximate FOM solution, but also on the high-fidelity FOM solution, whenever available. More precisely, if a new greedy parameter is chosen for enriching $\mathbf{V}$, the high-fidelity solution at this parameter needs to be computed. The corresponding snapshots of the nonlinear term at this greedy parameter are also readily available. For the other parameters in the training set, the nonlinear snapshots are obtained based on the approximate FOM solutions computed from the ROM.

Departing from the approaches in the above works, we extend the RBMEI algorithm in several directions. Our extension is called the Adaptive RBMEI algorithm.

- We consider a simultaneous enrichment of the RB and EI bases, starting from a single FOM solution at an initial parameter. This avoids the need for precomputing the FOM at all parameters in the training set.

- Our adaptive basis enrichment scheme is a flexible, two-way technique: we consider both adaptive basis extension and adaptive basis shrinking. The enrichment or shrinking is done using the estimated error at the greedy parameter as a *feedback*. If the error is large, we add more basis vectors and if the error is more than sufficient, we delete unnecessary basis vectors from $\mathbf{V}, \mathbf{U}$.

Also, we make use of the improved error estimator Eq. (4.28). The benefit of this is that we obtain a separation in the contribution of the error in terms of the RB and EI approximation. This is crucial to perform adaptive basis enrichment of $\mathbf{V}, \mathbf{U}$ individually. We highlight some key aspects in which our approach differs to the existing ones from [24, 65, 71]. When compared to the SER and PREIM algorithms in [24, 65] our method differs in the following ways:

- we do not use the approximate state solution to evaluate the nonlinear function, so that no extra errors are introduced during basis construction. For the hyperreduction (using EIM or DEIM), we only use the *high-fidelity* solutions that are needed for enriching $\mathbf{V}$ and which are computed by solving the FOM at the greedy parameters selected by the algorithm. The nonlinear functions are essentially evaluated using the FOM snapshots that are available *for free*.

- For SER and PREIM, a new greedy sample for basis enrichment is selected based on the EIM error, rather that the error of the ROM. Moreover, for PREIM, the selection process might be time consuming for dynamical systems since it is done at every parameter in the training set. In contrast, the adaptive scheme in the Adaptive RBMEI algorithm is based on an efficient output error estimator, that offers a separation in the RB and EI error contributions. At each iteration, a new parameter is selected based on both the RB and EIM errors.

- For SER and PREIM, at each iteration only one candidate basis vector is computed to enrich $\mathbf{U}$. But, our proposed method allows for adaptive EI basis construction. This means that the number of EI basis vectors to enrich $\mathbf{U}$ can vary at each iteration. In fact, both SER and PREIM do not consider an adaptive enrichment of the basis $\mathbf{V}$. However, our approach allows adaptive enrichment of both $\mathbf{V}$ and $\mathbf{U}$.

When compared to the method in [71] our approach has the following advantages:

- In [71], the initial size of the RB and EI bases are determined by solving the FOM for all parameters in a coarse training set, following which the bases are updated trivially with one new basis vector per greedy iteration. However, our approach avoids the need for choosing such a coarse training set. The initial bases $\mathbf{V}, \mathbf{U}$ are initialized with basis vectors obtained at a single FOM solution at an initial parameter, and this is more effective in the sense of adaptivity since it avoids many FOM solutions.

- The bases at each iteration are updated trivially with one new basis vector in [71]. Our approach allows for an adaptive update based on the estimated error.

Note that the estimator in Eq. (4.27) is applicable to estimating the error of the ROM in Eq. (2.18). However, in practice, we use the ROM in Eq. (2.36) using hyperreduction. Next, we introduce a modification of the error estimator in Eq. (4.27) when hyperreduction of the nonlinear term is used. That is, we propose an error estimator for the ROM in Eq. (2.36).

4.3.3. Error Estimation Considering Hyperreduction

To address the most general case of nonlinear systems, a hyperreduction procedure is used to efficiently evaluate the nonlinear term $\overline{\mathbf{f}}(\mathbf{x}^k)$ at the online stage and the ROM is of the form Eq. (2.36). Taking this into account, the residual term $\mathbf{r}_{\text{pr}}^{k+1}$(see Eq. (4.4)) in the estimator in Eq. (4.27) can be rewritten as

$$\begin{aligned}
\mathbf{r}_{\text{pr}}^{k+1} &= \overline{\mathbf{A}}\widetilde{\mathbf{x}}^k + \overline{\mathbf{f}}(\widetilde{\mathbf{x}}^k) + \overline{\mathbf{B}}\mathbf{u}^k - \overline{\mathbf{E}}\widetilde{\mathbf{x}}^{k+1}, \\
&= \overline{\mathbf{A}}\widetilde{\mathbf{x}}^k + \overline{\mathbf{f}}(\widetilde{\mathbf{x}}^k) + \mathscr{I}[\overline{\mathbf{f}}(\widetilde{\mathbf{x}}^k)] - \mathscr{I}[\overline{\mathbf{f}}(\widetilde{\mathbf{x}}^k)] + \overline{\mathbf{B}}\mathbf{u}^k - \overline{\mathbf{E}}\widetilde{\mathbf{x}}^{k+1}, \\
&= \underbrace{\big(\overline{\mathbf{A}}\widetilde{\mathbf{x}}^k + \mathscr{I}[\overline{\mathbf{f}}(\widetilde{\mathbf{x}}^k)] + \overline{\mathbf{B}}\mathbf{u}^k - \overline{\mathbf{E}}\widetilde{\mathbf{x}}^{k+1}\big)}_{=:\mathbf{r}_{\text{pr},\mathscr{I}}^{k+1}} + \underbrace{\big(\overline{\mathbf{f}}(\widetilde{\mathbf{x}}^k) - \mathscr{I}[\overline{\mathbf{f}}(\widetilde{\mathbf{x}}^k)]\big)}_{=:\mathbf{e}_{\mathsf{EI}}^k}.
\end{aligned}$$

In the above equation, $\mathbf{r}_{\mathrm{pr},\mathscr{I}}^{k+1}$ is the residual resulting from the ROM (Eq. (2.36)) by considering a hyperreduction step and $\mathbf{e}_{\mathsf{EI}}^{k}$ is the error introduced by the hyperreduction. By substituting this separation, we can rewrite Eq. (4.27) as a sum of (a) error resulting from approximating the state variable using $\mathbf{V}$ and (b) error resulting from approximating the nonlinear term using the hyperreduction basis $\mathbf{U}$. We have

$$\|\mathbf{y}^{k+1} - \overline{\mathbf{y}}^{k+1}\| \lessapprox \underbrace{\overline{\Phi}\|\mathbf{r}_{\mathrm{pr},\mathscr{I}}^{k+1}\|}_{=:\Delta_{\mathbf{y},\mathrm{RB}}^{k+1}} + \underbrace{\overline{\Phi}\|\mathbf{e}_{\mathsf{EI}}^{k}\|}_{=:\Delta_{\mathbf{y},\mathsf{EI}}^{k+1}}, \qquad k = 1, 2, \ldots, K. \tag{4.28}$$

The term $\Delta_{\mathbf{y},\mathrm{RB}}^{k+1} \in \mathbb{R}_{\geq 0}$ is the contribution to the overall estimated error $\Delta_{\mathbf{y}}^{k+1}$ from the RB approximation and $\Delta_{\mathbf{y},\mathrm{EI}}^{k+1} \in \mathbb{R}_{\geq 0}$ is an error estimator for the hyperreduction interpolation error $\mathbf{e}_{\mathsf{EI}}^{k}$.

The error estimator for the hyperreduction scheme is obtained based on the idea of *hierarchical error estimation* considered in works such as [19, 71, 100, 207]. We consider two different orders of hyperreduction (using EIM or DEIM) for the nonlinear term $\overline{\mathbf{f}}(\widetilde{\mathbf{x}})$: let $\mathbf{U}_{\mathring{\mathfrak{n}}} \in \mathbb{R}^{N\times\mathring{\mathfrak{n}}}, \mathcal{I}_{\mathring{\mathfrak{n}}} = [\mathring{\wp}_1, \ldots, \mathring{\wp}_{\mathring{\mathfrak{n}}}]$ denote the EI projection basis and the interpolation indices corresponding to an order-$\mathring{\mathfrak{n}}$ approximation and let $\mathbf{U}_{\mathfrak{n}} \in \mathbb{R}^{N\times\mathfrak{n}}, \mathcal{I}_{\mathfrak{n}} = [\wp_1, \ldots, \wp_{\mathfrak{n}}]$ denote the EI projection basis and the interpolation indices corresponding to an order-$\mathfrak{n}$ approximation, with $\mathring{\mathfrak{n}} > \mathfrak{n}$. Let us further define $\mathbf{S}_{\mathring{\mathfrak{n}}} := [\mathbf{e}_{\mathring{\wp}_1}, \ldots, \mathbf{e}_{\mathring{\wp}_{\mathring{\mathfrak{n}}}}] \in \mathbb{R}^{N\times\mathring{\mathfrak{n}}}$ and $\mathbf{S}_{\mathfrak{n}} := [\mathbf{e}_{\wp_1}, \ldots, \mathbf{e}_{\wp_{\mathfrak{n}}}] \in \mathbb{R}^{N\times\mathfrak{n}}$ with $\mathbf{e}_i \in \mathbb{R}^N$ denoting the i-th unit vector in $\mathbb{R}^N$. Based on these two different approximations, we have the following expression for the error incurred due to hyperreduction:

$$\mathbf{e}_{\mathsf{EI}}^{k} := \Pi_{\mathring{\mathfrak{n}}}\big(\mathbf{I} - \Pi_{\mathfrak{n}}\big)\overline{\mathbf{f}}(\widetilde{\mathbf{x}}^{k}). \tag{4.29}$$

The quantity $\Pi_{\mathring{\mathfrak{n}}}$ is defined as $\Pi_{\mathring{\mathfrak{n}}} := \big(\mathbf{I} - \Pi_{\mathfrak{n}}\big)\mathbf{U}_{\mathring{\mathfrak{n}}}\Big(\mathbf{S}_{\mathring{\mathfrak{n}}}^{\mathsf{T}}(\mathbf{I} - \Pi_{\mathfrak{n}})\mathbf{U}_{\mathring{\mathfrak{n}}}\Big)^{-1}\mathbf{S}_{\mathring{\mathfrak{n}}}^{\mathsf{T}}$ and we define $\Pi_{\mathfrak{n}} := \mathbf{U}_{\mathfrak{n}}\Big(\mathbf{S}_{\mathfrak{n}}^{\mathsf{T}}\mathbf{U}_{\mathfrak{n}}\Big)^{-1}\mathbf{S}_{\mathfrak{n}}^{\mathsf{T}}$, with $\mathbf{I} \in \mathbb{R}^{N\times N}$ being the identity matrix. A detailed derivation can be found in [207]. We will use this expression to obtain the empirical interpolation error that will be used for adaptive basis enrichment.

Mean Error Estimates The proposed error estimator Eq. (4.28) is given for every time instance t^k. For use in the proposed Two-way Adaptive PODEI and Adaptive RBMEI algorithms, we are interested in the mean error over time defined as below:

$$\frac{1}{K}\sum_{i=1}^{K}\|\mathbf{y}^{i} - \overline{\mathbf{y}}^{i}\| \lessapprox \frac{1}{K}\sum_{i=1}^{K}\big(\Delta_{\mathbf{y},\mathrm{RB}}^{i} + \Delta_{\mathbf{y},\mathrm{EI}}^{i}\big) = \Delta_{\mathbf{y},\mathrm{RB}} + \Delta_{\mathbf{y},\mathrm{EI}} =: \Delta_{\mathbf{y}}. \tag{4.30}$$

4.3.4. Adaptive update of RB and EI bases

We now discuss the main idea behind the adaptive basis enrichment. It is applicable to both non-parametric and parametric systems. To keep the discussion general, we illustrate the proposed approach for parametric systems.

Suppose that we have some desired tolerance for the ROM, denoted by ϵ. At each iteration of the POD-Greedy algorithm, we look at the ratio of the RB and EI estimated errors at the greedy parameter $\boldsymbol{\mu}^*$ w.r.t. the tolerance given by

$$\frac{\Delta_{\mathbf{y},\mathrm{RB}}(\boldsymbol{\mu}^*)}{\epsilon} \qquad \text{and} \qquad \frac{\Delta_{\mathbf{y},\mathrm{EI}}(\boldsymbol{\mu}^*)}{\epsilon}. \tag{4.31}$$

It is obvious that in Eq. (4.31), if either of the ratios is larger than 1, i.e., $\left(\frac{\Delta_{\mathbf{y},\mathrm{RB}}}{\epsilon}\right) > 1$ or $\left(\frac{\Delta_{\mathbf{y},\mathrm{EI}}}{\epsilon}\right) > 1$, the current RB or EI basis ($\mathbf{V}$ or $\mathbf{U}$) is not sufficient to meet the desired tolerance and needs to be extended. On the other hand, when either of the ratios is below 1, this implies that the current RB or EI basis is accurate and no new basis vectors need to be added. For the latter case, some basis vectors could be removed from $\mathbf{V}$ or $\mathbf{U}$ to make the bases as compact as possible. The magnitude of the ratios in Eq. (4.31) can vary over a large range, therefore, we instead use the logarithm (base 10) to map the magnitudes to a reasonable interval. We use the rounded, log-mapped values of the ratios in Eq. (4.31) as a 'feedback' for enriching or shrinking the RB, EI bases.

Remark 4.3:
The proposed adaptive basis enrichment scheme can also be applied to non-parametric systems. In this case, the corresponding form of Eq. (4.31) reads

$$\frac{\Delta_{\mathbf{y},\mathrm{RB}}}{\epsilon} \qquad \text{and} \qquad \frac{\Delta_{\mathbf{y},\mathrm{EI}}}{\epsilon}. \tag{4.32}$$

◊

Let r_{RB} and r_{EI} be the dimension of the RB, EI basis vectors, respectively, at the current iteration. We then update the dimensions based on the following rule:

$$\begin{aligned} \delta_{\mathrm{RB}} &:= \pm 1 + \left\lfloor \log_{10}\left(\frac{\Delta_{\mathbf{y},\mathrm{RB}}(\boldsymbol{\mu}^*)}{\epsilon}\right)\right\rfloor, \\ \delta_{\mathrm{EI}} &:= \pm 1 + \left\lfloor \log_{10}\left(\frac{\Delta_{\mathbf{y},\mathrm{EI}}(\boldsymbol{\mu}^*)}{\epsilon}\right)\right\rfloor. \end{aligned} \tag{4.33}$$

Based on these updates, the number of RB, EI basis vectors at each iteration is

$$\begin{aligned} r_{\mathrm{RB}} &= r_{\mathrm{RB}} + \delta_{\mathrm{RB}}, \\ r_{\mathrm{EI}} &= r_{\mathrm{EI}} + \delta_{\mathrm{EI}}. \end{aligned} \tag{4.34}$$

The basis update scheme is able to both add or remove basis vectors from $\mathbf{V}$ and $\mathbf{U}$. The quantity ± 1 in Eq. (4.33) tries to ensure that at least one basis vector is added ($+1$) or removed (-1) at each iteration, in case the rounded value $\gamma_{\mathrm{RB}} = \left\lfloor \log_{10}\left(\frac{\Delta_{\mathbf{y},\mathrm{RB}}(\boldsymbol{\mu}^*)}{\epsilon}\right)\right\rfloor$ or $\gamma_{\mathrm{EI}} = \left\lfloor \log_{10}\left(\frac{\Delta_{\mathbf{y},\mathrm{EI}}(\boldsymbol{\mu}^*)}{\epsilon}\right)\right\rfloor$ is zero, but the values $\log_{10}\left(\frac{\Delta_{\mathbf{y},\mathrm{RB}}(\boldsymbol{\mu}^*)}{\epsilon}\right)$ or $\log_{10}\left(\frac{\Delta_{\mathbf{y},\mathrm{EI}}(\boldsymbol{\mu}^*)}{\epsilon}\right)$ are non-zero (either slightly smaller or slightly larger than zero).

Overcoming Stagnation The goal for adaptivity is to add/delete basis vectors to/from the current RB, EI bases so that the ROM meets the given tolerance, while being kept as compact as possible. It is observed that in some cases, the convergence of the adaptive algorithm becomes slow when the estimated error is close to the tolerance. In such situations, the number of additional basis vectors to be added/deleted is usually one, resulting in a slow convergence. A second issue is that the error estimator could keep oscillating (below and above the tolerance ϵ) upon basis enriching/shrinking. We propose to define a *zone-of-acceptance* (zoa) for the output error. In particular, we set a new value $\epsilon^* < \epsilon$. Taken together, ϵ^* and ϵ define the zoa: $[\epsilon^*, \epsilon]$. Whenever the estimated error ($\Delta_{\mathbf{y}}$) falls within the zoa,the algorithm will terminate. We typically set $\epsilon^* = 0.1\epsilon$. The zoa helps address the second issue highlighted above (oscillations) for the case of basis enriching/shrinking and addresses the first issue (slow convergence) in case of basis shrinking.

(D)EIM Plateaus Previous works dealing with the simultaneous enrichment of RB and (D)EIM bases have noted the issue of (D)EIM plateaus. In fact, two different notions of plateauing have been observed and presented in [45, 71, 98, 121, 190, 197]. In [98, 190, 197], the authors note that when the number of basis elements of the EIM approximation is fixed at some small value, an increase in the number of RB vectors does not result in an improvement in the overall error. This is a plateauing due to *large error in the EI approximation*. However, in [71], it is observed that EIM plateaus occur when the error contribution is dominated by the RB approximation error and a further enrichment of the EIM basis is useless. This is a plateauing due to *large error in the RB approximation*. These observations suggest that simultaneous enrichment of the RB and the EI basis is critical to avoid plateaus in general. A solution proposed in [71] is to monitor an error estimator over the training set, that is, $\max_{\boldsymbol{\mu}\in\Xi} \Delta_{\mathbf{y}}$, between two successive iterations. If a newly added RB basis vector leads to an increase in the error, then it is dropped and only the EIM basis is updated. In [197], the author considers different tolerances for the RB, EIM approximations. However, the proposed algorithm involves some user-defined constants which makes it less straightforward to implement. From our experience in the numerical tests, setting different tolerances for the RB and EI basis updates in Eq. (4.34), viz., ϵ_{RB} and ϵ_{EI}, respectively, with $\epsilon_{\text{EI}} < \epsilon_{\text{RB}}$ proves to give the best approximation. A similar observation is also noted in [190]; however, no simultaneous enrichment is considered there. Still, it is not entirely clear how small the EI approximation tolerance has to be when compared to the RB tolerance. In our numerical experiments, we set $\epsilon_{\text{RB}} = \epsilon$, the ROM tolerance and use $\epsilon_{\text{EI}} = 0.01\epsilon_{\text{RB}}$ for the EI tolerance. We now detail the Two-way Adaptive PODEI algorithm and Adaptive RBMEI algorithm.

4.3.5. Two-way Adaptive PODEI algorithm

The Two-way Adaptive PODEI algorithm extends the one proposed in [87]. It offers the flexibility of being initialized with an arbitrary choice of dimensions of the POD and EI bases. The method is able to suggest the proper number of basis vectors to be added to or removed from the current basis, and yields a compact and stable ROM, for the

given tolerance. The pseudocode is sketched in Algorithm 4.1. The algorithm begins by solving the FOM Eq. (2.11) to collect snapshots. Then, the POD and EI tolerances $\epsilon_{\text{POD}}, \epsilon_{\text{EI}}$ are used to identify the POD projection matrix $\mathbf{V}_0$ and the hyperreduction quantities $\mathbf{U}_0, \mathcal{J}_0$. Usually, the tolerances are chosen conservatively small ($\sim 10^{-10}$). This potentially leads to an unnecessarily large dimension for the ROM. Moreover, there is no correlation to the actual desired ROM tolerance ϵ. To obtain a compact ROM, the adaptive scheme is initialized in Step 4 with an initial choice of dimensions $r_{\text{RB}}, r_{\text{EI}}$ and the initial projection matrices $\mathbf{V}_0, \mathbf{U}_0$ and initial interpolation indices $\mathcal{J}_0$ are identified. In Step 5, a Krylov-subspace method, such as GMRES, is used to obtain the approximate dual solution $\widetilde{\mathbf{x}}_{\text{du}}$ and in Step 6, the *inf-sup* constant σ_{min} is computed. As long as the estimated error $\Delta_{\mathbf{y}}$ does not fall within the zone-of-acceptance, the RB and EI bases dimensions are updated in Step 13, by adding or removing basis vectors. In Step 14, we ensure that the dimension of the EI projection matrix $\mathbf{U}$ is larger than that of the RB projection matrix $\mathbf{V}$. This is motivated by our observation in some numerical experiments that for $r_{\text{EI}} < r_{\text{RB}}$, the ROM is no longer stable.

As will be demonstrated in the numerical experiments, using Algorithm 4.1 leads to a much smaller ROM than the one resulting from applying Algorithm 2.7. Moreover, for the ROM obtained using Algorithm 2.7 there is no guarantee that it meets the desired tolerance ϵ; whereas this is ensured for the proposed approach.

4.3.6. Adaptive RBMEI algorithm

We now detail the Adaptive RBMEI algorithm, our proposed extension to Algorithm 2.8 with adaptive basis enrichment. The basic idea is to start from an initial greedy parameter $\boldsymbol{\mu}^*$ (chosen randomly), then iteratively and non-trivially update the RB and EI bases $\mathbf{V}$ and $\mathbf{U}$. In Step 1, in case the dual system Eq. (4.2) turns out to be non-parametric, we use a Krylov-subspace method (such as GMRES) to obtain the approximate dual solution $\widetilde{\mathbf{x}}_{\text{du}}$, since it will be required in the error estimator Eq. (4.28). In Step 2, we compute the approximate *inf-sup* constants $\widetilde{\sigma}_{\text{min}}(\boldsymbol{\mu}), \forall \boldsymbol{\mu} \in \Xi$ using the RBF-based interpolation approach discussed in Section 4.2.5. In Step 3, the quantities involved in the greedy algorithm are initialized. In our approach, we build the dual system projection matrix $\mathbf{V}_{\text{du}}$ as a part of the greedy algorithm. It is also possible to construct $\mathbf{V}_{\text{du}}$ separately, as done in [100], but with possibly more computational time. In Steps 5-16 of Algorithm 4.2, we update the RB matrix with r_{RB} basis vectors based on the POD applied to the snapshot matrix at every iteration. In Step 12, we use the *modified Gram-Schmidt* process to orthogonalize the most recently added columns against the existing columns of $\mathbf{V}$. For the case when $r_{\text{RB}} < 0$, i.e., when basis vectors need to be removed, we delete the last few columns of the matrix $\mathbf{V}$ in Step 15. The use of the modified Gram-Schmidt process allows for the direct removal of the last r_{RB} columns in $\mathbf{V}$. Step 17 involves collecting the snapshots of the nonlinear term at the current greedy parameter. We use this to update the matrix $\mathbf{F}$, which consists of all the snapshots of the nonlinear term obtained from the greedy parameters selected so far. In Step 18, the EI basis and interpolation points are updated based on the new information added in $\mathbf{F}$. The subroutine `update_ei` invokes either Algorithm 2.5 or Algorithm 2.6. The dual RB matrix $\mathbf{V}_{\text{du}}$ is enriched in Step 19 for the case of a parametric t-dual system. To do this, we invoke the subroutine `update_dual` where a greedy

Algorithm 4.1: `TWO-WAY_ADAPTIVE_PODEI`
Computes a ROM for a non-parametric nonlinear dynamical systems using adaptive basis enrichment.

Input: System matrices and nonlinear term: $\mathbf{E}, \mathbf{A}, \mathbf{B}, \mathbf{C}$ and $\mathbf{f}$, POD, EI tolerance ϵ_{POD} and ϵ_{EI}, error tolerance ϵ, zoa ϵ^*, Maximum number of greedy iterations *iter_max*.

Output: Reduced system matrices $\widehat{\mathbf{E}}, \widehat{\mathbf{A}}, \widehat{\mathbf{B}}, \widehat{\mathbf{C}}$ and EI quantities $\boldsymbol{\mathcal{U}}, \mathcal{I}$.

1 Simulate the FOM Eq. (2.11) to obtain state snapshots $\mathbf{X} = \{\mathbf{x}^k\}_{k=0}^K$ and corresponding nonlinear term snapshots $\mathbf{F} := \{\overline{\mathbf{f}}^k\}_{k=0}^K$.
2 Compute POD basis $\mathbf{V}_0 = \texttt{POD}(\mathbf{X}, \epsilon_{\text{POD}})$.
3 Compute EI quantities $[\mathbf{U}_0, \mathcal{I}_0] = \texttt{EIM}(\mathbf{F}, \epsilon_{\text{EI}}, \mathit{iter\_max})$ or $[\mathbf{U}_0, \mathcal{I}_0] = \texttt{DEIM}(\mathbf{F}, \epsilon_{\text{EI}})$.
4 Set initial choice of RB, EI basis $r_{\text{RB}}, r_{\text{EI}}$ (with $r_{\text{EI}} > r_{\text{RB}}$), form initial bases $\mathbf{V} := \mathbf{V}_0(:, 1 : r_{\text{RB}}), \mathbf{U} := \mathbf{U}_0(:, 1 : r_{\text{EI}})$ and $\mathcal{I} := \mathcal{I}_0(1 : r_{\text{EI}})$.
5 Determine the solution to the t-dual system Eq. (4.2) $\widetilde{\mathbf{x}}_{\text{du}}$ using GMRES.
6 Compute the *inf-sup* constant $\sigma_{\min}$ of the matrix $\mathbf{E}$.
7 Simulate the ROM Eq. (2.36) based on $\mathbf{V}, \mathbf{U}, \mathcal{I}$ and determine $\Delta_{\mathbf{y}} := \Delta_{\mathbf{y},\text{RB}} + \Delta_{\mathbf{y},\text{EI}}$.
8 **while** $\Delta_{\mathbf{y}} \notin$ *zoa* **do**
9 Determine $\delta_{\text{RB}} = \left\lfloor \log_{10}\left(\frac{\Delta_{\mathbf{y},\text{RB}}}{\epsilon}\right)\right\rfloor$ and $\delta_{\text{EI}} = \left\lfloor \log_{10}\left(\frac{\Delta_{\mathbf{y},\text{EI}}}{\epsilon_{\text{EI}}}\right)\right\rfloor$.
10 **if** $\delta_{\text{RB}} = 0$ or $\delta_{\text{EI}} = 0$ **then**
11 enforce trivial update $\delta_{\text{RB}} = \pm 1$, $\delta_{\text{EI}} = \pm 1$.
12 **end**
13 Update the basis vectors: $r_{\text{RB}} = r_{\text{RB}} + \delta_{\text{RB}}$, $r_{\text{EI}} = r_{\text{EI}} + \delta_{\text{EI}}$.
14 Ensure $r_{\text{EI}} > r_{\text{RB}}$.
15 Update the projection matrices and interpolation indices: $\mathbf{V} := \mathbf{V}_0(:, 1 : r_{\text{RB}}), \mathbf{U} := \mathbf{U}_0(:, 1 : r_{\text{EI}})$ and $\mathcal{I} := \mathcal{I}_0(1 : r_{\text{EI}})$.
16 Simulate ROM Eq. (2.36) based on updated $\mathbf{V}, \mathbf{U}$ and $\mathcal{I}$, compute error estimator in Eq. (4.30) $\Delta_{\mathbf{y}} = \Delta_{\mathbf{y},\text{RB}} + \Delta_{\mathbf{y},\text{EI}}$.
17 **end**
18 Determine reduced matrices $\widehat{\mathbf{E}}, \widehat{\mathbf{A}}, \widehat{\mathbf{B}}, \widehat{\mathbf{C}}$ through Galerkin projection using $\mathbf{V}$.
19 Determine $\boldsymbol{\mathcal{U}} := (\mathbf{V})^{\mathsf{T}}\mathbf{U}(\mathbf{U})^{\mathcal{I}}$.

algorithm is implemented. The error estimator $\Delta_{\mathbf{y},\text{du}}$ is the norm of the dual system residual Eq. (4.7). Once we have the RB matrix $\mathbf{V}$, the EI basis and sampling points $\mathbf{U}, \mathcal{I}$, and the dual projection matrix $\mathbf{V}_{\text{du}}$, we obtain the ROM and estimate the error using Eq. (4.30) for every parameter in the training set Ξ. The next greedy parameter is chosen in Step 22 based on the worst approximation error $\Delta_{\mathbf{y}}$ for all $\boldsymbol{\mu} \in \Xi$. Finally, in Step 24, the number of basis vectors for the RB and EI projection matrices (r_{RB} and r_{EI}) at a given iteration are determined by the update rule in Eq. (4.34). This part is implemented in the subroutine `adapt_basis_update`. Based on the values of $\delta_{\text{RB}}, \delta_{\text{EI}}$, we add a trivial update of ± 1 if required. It is worth noting that in Step 41, we ensure that the dimension of the EI basis is larger than $\mathbf{V}$. As before, this is to ensure a stable ROM. This can be also be assured by choosing different tolerances $\epsilon_{\text{RB}}, \epsilon_{\text{EI}}$. We

Algorithm 4.2: `ADAPTIVE_RBMEI`
Computes a ROM using the Adaptive POD-Greedy algorithm in combination with hyperreduction.

Input: System matrices and nonlinear term: $\mathbf{E}, \mathbf{A}, \mathbf{B}, \mathbf{C}$ and $\mathbf{f}$, Training set Ξ, EI tolerance ϵ_{EI}, ROM tolerance ϵ, zoa ϵ^*, Deflation tolerance ϵ_{def}, Maximum iterations $iter\_max$.
Output: Reduced system matrices $\widehat{\mathbf{E}}, \widehat{\mathbf{A}}, \widehat{\mathbf{B}}, \widehat{\mathbf{C}}$ and EI quantities $\boldsymbol{\mathcal{U}}, \mathcal{I}$.

For non-parametric t-dual system Eq. (4.2), obtain $\widetilde{\mathbf{x}}_{\text{du}}$ using a Krylov-space method.
Use RBF interpolation to approximate *inf-sup* constants as $\widetilde{\sigma}_{\min}(\boldsymbol{\mu})$ for all $\boldsymbol{\mu} \in \Xi$.
Initialization: $\mathbf{V} = [\,]$, $\mathbf{V}_{\text{du}} = [\,]$, $\mathbf{U} = [\,]$, $\mathcal{I} = \varnothing$, $err\_max = 1 + \epsilon$, $err\_max\_dual = 1 + \epsilon$, $iter = 1$, $\boldsymbol{\mu}^*, \boldsymbol{\mu}^*_{\text{du}}$ (chosen randomly from Ξ), $r_{\text{RB}} = r_{\text{EI}} = 1$.
while $err\_max \notin zoa$ and $iter \leq iter\_max$ **do**
  if $r_{\text{RB}} > 0$ **then**
    Solve FOM Eq. (2.11) at $\boldsymbol{\mu}^*$; obtain snapshots matrix $\mathbf{X}(\boldsymbol{\mu}^*)$.
    if $iter == 1$ **then**
      $\mathbf{V} = \texttt{POD}(\mathbf{X}, r_{\text{RB}})$
    else
      Compute $\overline{\mathbf{X}} := \mathbf{X} - \mathbf{V}\mathbf{V}^{\mathsf{T}}\mathbf{X}$.
      $\mathbf{V}_{\text{POD}} := \texttt{POD}(\overline{\mathbf{X}}, r_{\text{RB}})$.
      $\mathbf{V} = \texttt{orth\_def\_mat}(\mathbf{V}, \mathbf{V}_{\text{POD}}, \epsilon_{\text{def}})$. `/* see Step 15 of Algorithm 2.4 */`
    end
  else
    Remove the last r_{RB} columns from $\mathbf{V}$.
  end
  Form snapshot matrix for the nonlinear term: $\mathbf{F} := [\mathbf{F} \;\; \overline{\mathbf{f}}(\overline{\mathbf{X}})]$.
  Update EI basis and interpolation points $\mathbf{U}, \mathcal{I}$ using the subroutine `update_ei`.
  For a parametric t-dual system, update dual RB matrix $\mathbf{V}_{\text{du}}$ using the subroutine `update_dual`.
  $iter = iter + 1$.
  Determine reduced matrices $\widehat{\mathbf{E}}, \widehat{\mathbf{A}}, \widehat{\mathbf{B}}, \widehat{\mathbf{C}}$ through Galerkin projection using $\mathbf{V}$ as in Eq. (2.14).
  Solve the ROM Eq. (2.36) for all $\boldsymbol{\mu} \in \Xi$; find $\boldsymbol{\mu}^* = \arg\max_{\boldsymbol{\mu} \in \Xi} \Delta_{\mathbf{y}}(\boldsymbol{\mu})$.
  Set $err\_max = \Delta(\boldsymbol{\mu}^*)$.
  Update $r_{\text{RB}}, r_{\text{EI}}$ using the subroutine `adapt_basis_update`.
end

Function $[\mathbf{U}, \mathcal{I}]$ = `update_ei`$(\mathbf{F}, r_{\text{EI}}, \epsilon_{\text{EI}})$**:**
  Set $iter\_max = \texttt{size}(\mathbf{F}, 2)$.
  Compute the EI basis and interpolation points: $[\mathbf{U}, \mathcal{I}] = \texttt{EIM}(\mathbf{F}, r_{\text{EI}}, iter\_max)$ or $[\mathbf{U}, \mathcal{I}] = \texttt{DEIM}(\mathbf{F}, r_{\text{EI}})$.

Function $[\mathbf{V}_{\text{du}}, \boldsymbol{\mu}^*_{\text{du}}]$ = `update_dual`$(\mathbf{V}_{\text{du}}, \boldsymbol{\mu}^*_{\text{du}}, \epsilon, err\_max\_dual)$**:**
  if $err\_max\_dual > \epsilon$ **then**
    Solve Eq. (4.2) for the greedy parameter $\boldsymbol{\mu}^*_{\text{du}}$ to get snapshot $\mathbf{x}_{\text{du}}(\boldsymbol{\mu}^*_{\text{du}})$.
    Update $\mathbf{V}_{\text{du}}$ with new snapshot: $\mathbf{V}_{\text{du}} = [\mathbf{V}_{\text{du}}, \mathbf{x}_{\text{du}}(\boldsymbol{\mu}^*_{\text{du}})]$, orthogonalize the columns of $\mathbf{V}_{\text{du}}$.
    Find $\boldsymbol{\mu}^*_{\text{du}} := \arg\max_{\boldsymbol{\mu} \in \Xi} \Delta_{\mathbf{y},\text{du}}$.
  end

Function $[r_{\text{RB}}, r_{\text{EI}}]$ = `adapt_basis_update`$(r_{\text{RB}}, r_{\text{EI}}, \Delta(\boldsymbol{\mu}^*))$**:**
  Find $\delta_{\text{RB}} = \left\lfloor \log_{10}\left(\frac{\Delta_{\mathbf{y},\text{RB}}(\boldsymbol{\mu}^*)}{\epsilon}\right)\right\rfloor$ and $\delta_{\text{EI}} = \left\lfloor \log_{10}\left(\frac{\Delta_{\mathbf{y},\text{EI}}(\boldsymbol{\mu}^*)}{\epsilon_{\text{EI}}}\right)\right\rfloor$.
  if $\delta_{\text{RB}} = 0$ or $\delta_{\text{EI}} = 0$ **then**
    enforce trivial update $\delta_{\text{RB}} = \pm 1$, $\delta_{\text{EI}} = \pm 1$.
  end
  Update the basis vectors: $r_{\text{RB}} = \delta_{\text{RB}}$, $r_{\text{EI}} = r_{\text{EI}} + \delta_{\text{EI}}$.
  Ensure $r_{\text{EI}} > \text{rank}(\mathbf{V}) + r_{\text{RB}}$.

set $\epsilon_{\text{RB}} = \epsilon$ and $\epsilon_{\text{EI}} = 0.01\epsilon_{\text{RB}}$.

4.3.7. Numerical Examples

Having introduced the idea of adaptive basis enrichment for both non-parametric and parametric systems, we validate these approaches and compare them to the state-of-the-art methods through some benchmark examples. The examples we consider are

carefully chosen to demonstrate different aspects of our proposed algorithms. The first example is that of a fluidized bed crystallizer (FBC) [136], an apparatus used in chemical and process engineering for the separation of chemical compounds. This example is non-parametric. We use it to test Algorithm 4.1 and also compare the results to that obtained in [214]. Following that, we consider an example of the well-known Burgers' equation to test Algorithm 4.2 for parametric systems.

In the numerical examples, we compare the performance of the proposed error estimator Eq. (4.28) to the error estimator in Eq. (4.12) (with hyperreduction involved). For non-parametric systems, we compare the mean error over time for either error estimator. We have

$$\tilde{\Delta}_{\text{mean}} := \frac{1}{K}\sum_{i=1}^{K} \widetilde{\Delta}^i_{\mathbf{y}}, \qquad \text{and} \tag{4.35}$$

$$\Delta_{\text{mean}} := \frac{1}{K}\sum_{i=1}^{K} \Delta^i_{\mathbf{y}}. \tag{4.36}$$

For parametric systems, we compare the maximum of the mean errors $\tilde{\Delta}_{\text{mean}}$ or Δ_{mean}, evaluated over the training set Ξ. We define

$$\tilde{\Delta}_{\text{max}} := \max_{\boldsymbol{\mu}\in\Xi} \tilde{\Delta}_{\text{mean}}, \qquad \text{and} \tag{4.37}$$

$$\Delta_{\text{max}} := \max_{\boldsymbol{\mu}\in\Xi} \Delta_{\text{mean}}. \tag{4.38}$$

The true errors for the error estimator Eq. (4.12) are defined as:

$$\tilde{\epsilon}_{\text{mean}} := \frac{1}{K}\sum_{i=1}^{K} \|\mathbf{y}^i - \widetilde{\mathbf{y}}^i\|, \qquad \text{and} \tag{4.39}$$

$$\tilde{\epsilon}_{\text{max}} := \max_{\boldsymbol{\mu}\in\Xi} \tilde{\epsilon}_{\text{mean}}. \tag{4.40}$$

And, the true errors for the error estimator Eq. (4.27) are defined as:

$$\epsilon_{\text{mean}} := \frac{1}{K}\sum_{i=1}^{K} \|\mathbf{y}^i - \overline{\mathbf{y}}^i\|, \qquad \text{and} \tag{4.41}$$

$$\epsilon_{\text{max}} := \max_{\boldsymbol{\mu}\in\Xi} \epsilon_{\text{mean}}. \tag{4.42}$$

All numerical results in this section were obtained using MATLAB®2015a, on a laptop with INTEL®CORE™i5-7200U @ 2.5 GHZ, with 8 GB of RAM.

4.3.7.1. Fluidized Bed Crystallizer

The FBC is a setup used in the field of chemical engineering to separate chemical compounds known as enantiomers. Enantiomers are molecules that have the same physical and chemical properties but occur as mirror images of one another. Due to their similar properties, separation of the two components is not easily achieved using simple

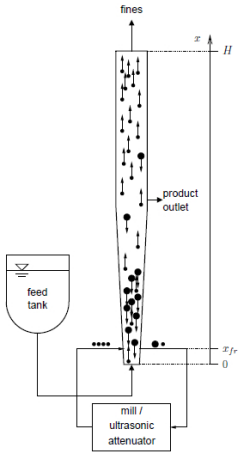


Figure 4.1.: Fluidized Bed Crystallizer.

techniques, but requires sophisticated methods such as adsorption, crystallization, and so on. For a more in-depth treatment, the reader is referred to [136]. Figure 4.1 shows the FBC, which is a long cylindrical column, with the walls tapering inwards as one approaches the bottom. The chemical mixture that has the two enantiomers dissolved in it (called racemate) is injected from the bottom. Some seed crystals need to be introduced into the crystallizer before it begins operation. Seed crystals are essentially the pure crystals of the enantiomer we want to isolate. The seeds are necessary for triggering the precipitation of the crystals in the racemate. During its operation, the smaller crystals move to the top of the crystallizer along with the fluid flow. Bigger crystals sink to the bottom from where they are collected and sent to a crushing device (such as an ultrasonic attenuator) to be crushed to an appropriate size and reintroduced as seed crystals. The crystallization process is governed by a set of conservation formulas, called the *population balance equations* which are PDEs. The PDE governing the FBC is given by

$$\begin{aligned} A_c(z)\frac{\partial n}{\partial t} = & -\frac{\partial}{\partial z}\big(A_c(z)v_p(z,L,t)n(z,L,t)\big) + D\frac{\partial}{\partial z}\Big(A_c(z)\frac{\partial n}{\partial z}\Big) - A_c(z)G\frac{\partial n}{\partial L} \\ & + \dot{V}_{\text{us}}\left(n_{\text{us}}(L)\frac{\int_0^\infty nl^3dl}{\int_0^\infty n_{us}l^3dl} - n(z,L,t)\right)\hat{\delta}(z-z_{\text{us}}), \end{aligned} \tag{4.43}$$

where

- z denotes the spatial coordinate, L denotes the particle size coordinate,
- $A_c(\cdot)$ is the area of cross-section, n is the number size density, i.e., the number of particles per volume of size L at coordinate z at time t,
- v_p is the plug flow velocity,
- D is the dispersion constant, G is the crystal growth factor,

- $\dot{V}_{\text{us}}$ is the volume flow to/from the attenuator,
- $n_{\text{us}}(L)$ is a constant describing the distribution of the crystals coming from the attenuator.

The state variable in the above equation is the quantity $n(z, L, t)$. We discretize the spatial coordinate z and the particle diameter coordinate L, using FVM. For time discretization, we adopt the IMEX first-order Euler scheme. Based on this, Eq. (4.43) can be written in the form of Eq. (2.11). Table 4.1 gives the model parameters that we consider for the full order simulation. The discretized FOM is of size $N = 18400$. The output quantity of interest is the volume fraction of particles. We plot this for the larger time interval of $10,000$ seconds in Figure 4.2. The model of the crystallizer takes a very long time to reach a cyclic, quasi steady-state, usually 5000 seconds. However, for snapshot generation, we need only the transient portion and the first cycle of the quasi steady-state, since the latter cycles behave very similarly. As a result we only need to simulate the FOM till 500 seconds (dashed gray line in Figure 4.2) for snapshot generation. To this end, we collect a snapshot every two seconds, resulting in 250 snapshots. For this model, we first apply Algorithm 2.7 with the tolerances $\epsilon_{\text{POD}} = \epsilon_{\text{EI}} = 10^{-10}$. The tolerance for the ROM is $\epsilon = 10^{-4}$. Although the tolerance for the desired ROM is 10^{-4}, we choose $\epsilon_{\text{POD}} = \epsilon_{\text{EI}}$ conservatively, since we do not know *a priori* how the singular values correlate to the actual output error. The ROM obtained from Algorithm 2.7 is of dimension $(r_{\text{RB}}, r_{\text{EI}}) = (60, 61)$. Following this, we make use of Algorithm 4.1. The tolerance for the ROM is the same $\epsilon = 10^{-4}$. We make use of GMRES to solve the associated dual system. It is implemented via the MATLAB®function `gmres`. Moreover, we use the Incomplete LU (ILU) factorization with a drop tolerance of 10^{-3} as a preconditioner. The GMRES tolerance is set to be 10^{-6}. To test Algorithm 4.1, we consider two cases. We denote the first case as INCREASE. It involves starting from a small initial choice of RB and DEIM bases dimensions $(r_{\text{RB}}, r_{\text{EI}})$, and iteratively adding new basis vectors to both. In the second case, denoted as DECREASE, we initialize Algorithm 4.1 with a larger number of initial basis vectors and adaptively remove basis vectors, till the ROM reaches the defined tolerance band zoa. For the zoa, we set $\epsilon^* = 10^{-5}$. Figure 4.3a shows the adaptive generation of POD and DEIM basis starting from small initial number of $(r_{\text{RB}}, r_{\text{EI}}) = (3, 8)$. The error estimator is below the tolerance after 9 iterations, showing that Algorithm 4.1 terminates.

The adaptive process results in a final ROM of $(r_{\text{RB}}, r_{\text{EI}}) = (16, 20)$ basis vectors. In Figure 4.4, we show the error landscape obtained by plotting the logarithm of the mean error estimator (Eq. (4.30)) corresponding to different combinations of $(r_{\text{RB}}, r_{\text{EI}})$. On the landscape, we mark the trajectory $\Delta_{\mathbf{y}}$, adaptively selected by Algorithm 4.1. We can see that, for several combinations of $r_{\text{RB}}, r_{\text{EI}}$ not present in the adaptive trajectory, the resulting ROMs are unstable. For ease of visualization, the $\Delta_{\mathbf{y}}$ resulting in unstable ROMs were set to be 1 in the log-scale. The figure clearly illustrates how the algorithm converges to the minimum in the landscape, while avoiding the combinations resulting in instabilities. Further, one can identify the plateaus in the error, whenever the RB approximation is too poor. An additional observation deserving attention is that the instabilities mainly occur at $(r_{\text{RB}}, r_{\text{EI}})$ combinations with $r_{\text{EI}} < r_{\text{RB}}$. For a pair of big

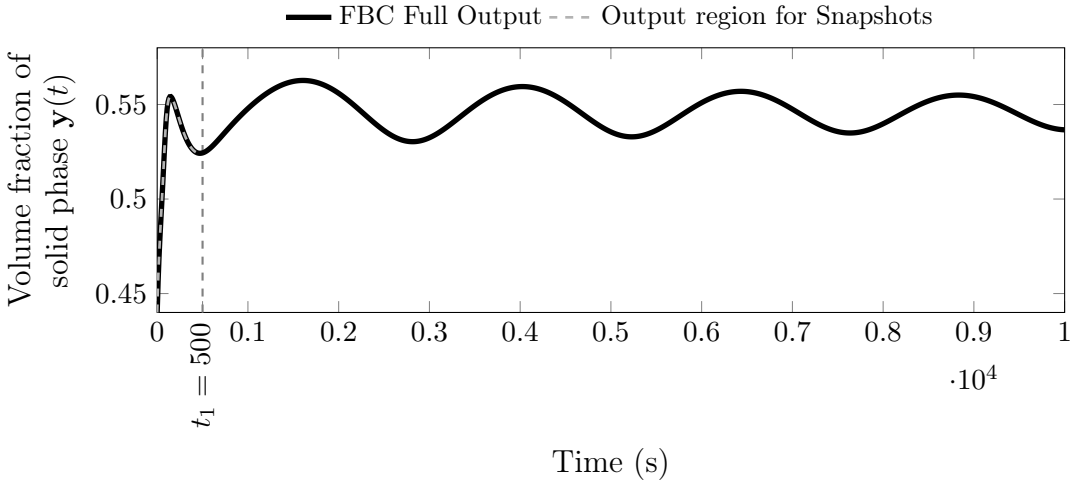
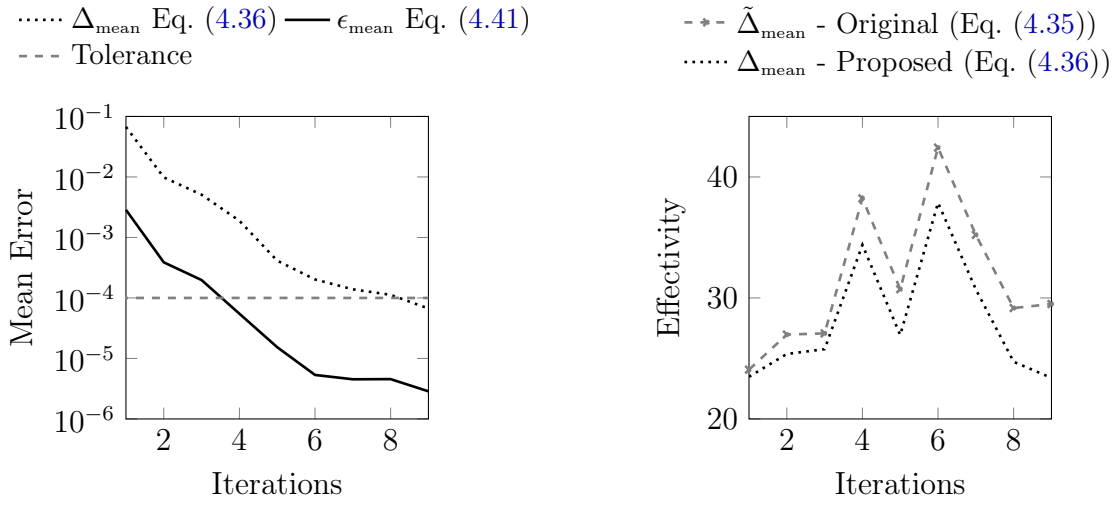


Figure 4.2.: Output quantity for the Fluidized Bed Crystallizer; the black line is the entire output until quasi steady-state; the dashed gray line shows the time until which snapshots are collected.



(a) Convergence of estimated error.

(b) Effectivity Comparison.

Figure 4.3.: FBC ROM obtained using INCREASE.

initial values: $(r_{\text{RB}}, r_{\text{EI}}) = (31, 39)$, the iterations of Algorithm 4.1 are shown in Figure 4.5a. In the beginning, the error estimator is below 10^{-5}, indicating that the ROM is very accurate and there is possibility to further reduce the size of the ROM. After 7 iterations, the reduced basis vectors from POD as well as the DEIM basis vectors are adaptively adjusted to $(r_{\text{RB}}, r_{\text{EI}}) = (17, 28)$. These results are summarised in Table 4.2. In Figures 4.3a and 4.5a, the true error is the mean error defined by the left hand side of Eq. (4.30) and the corresponding error estimator is defined by the right hand side of the same inequality. In Figures 4.3b and 4.5b, we compare the effectivities of the original and the modified error estimators. On the one hand, both error estimators show good effectivities and are relatively sharp. On the other hand, the modified error

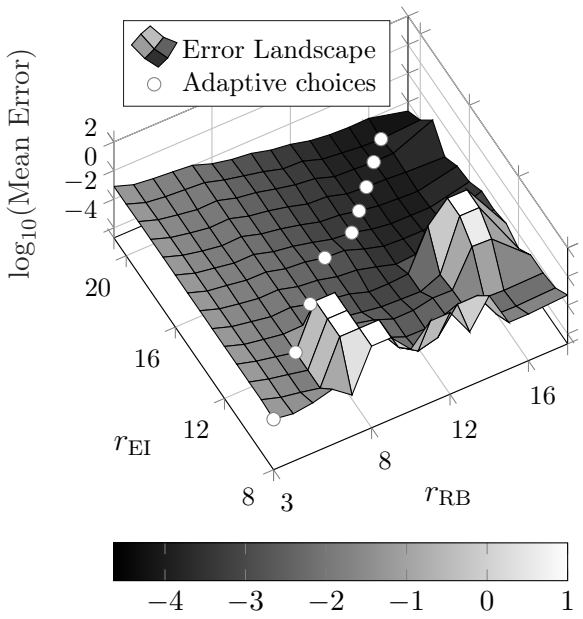


Figure 4.4.: Error landscape for the INCREASE procedure for the FBC example.

Table 4.1.: Simulation data for the FBC.

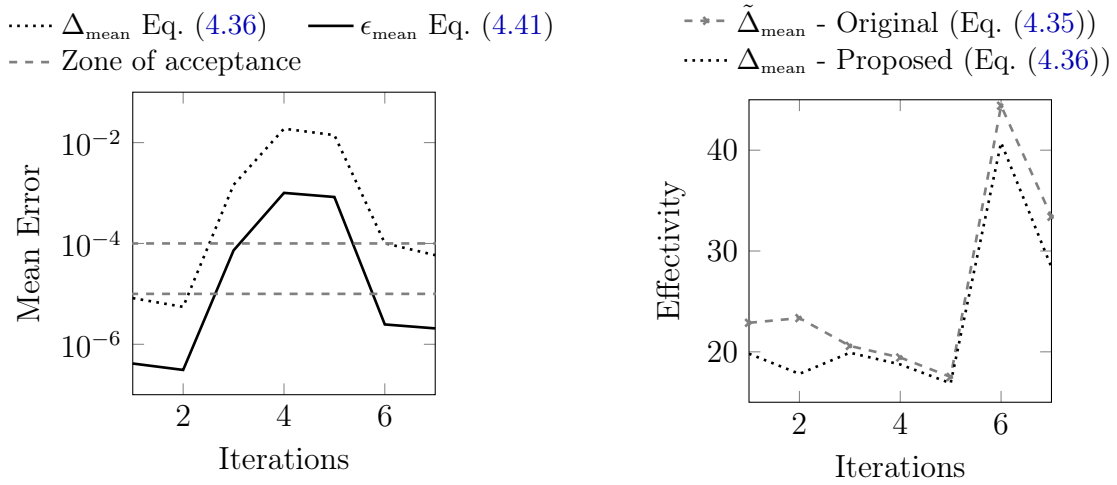
Interpolation	N	$[0, T]$ (s)	tolerance ϵ	ϵ_{POD}
DEIM	18400	[0, 500]	10^{-4}	10^{-10}

Table 4.2.: Simulation results for the FBC example.

Process	Initial		Final		Iterations
	r_{RB}	r_{EI}	r_{RB}	r_{EI}	
INCREASE	3	8	16	20	9
DECREASE	31	39	17	28	7

estimator clearly outperforms the original estimator, especially in the final step of the algorithm. Figure 4.6 shows the error landscape for the DECREASE case. Once again, one can see how Algorithm 4.1 avoids unstable RB, DEIM basis vector combinations and converges to a compact ROM.

Finally, Figures 4.7a and 4.7b compare the modified error estimator for the final ROM over all time steps t^k, with the true error, in the increasing and decreasing cases, respectively. Figure 4.7 not only shows the sharpness of the modified error estimator, especially for the cyclic state in the time interval $[200, 500]$s, but also verifies the reliability of the error estimator.



(a) Convergence of estimated error.

(b) Effectivity Comparison.

Figure 4.5.: FBC ROM obtained using DECREASE.

4.3.7.2. Burgers' Equation

We next test the proposed Adaptive PODEI algorithm on the one-dimensional viscous Burgers' equation defined in the domain $\Omega := [0, 1]$ and $z \in \Omega$ denoting the spatial variable. The parameter that varies is the viscosity. The equation and initial boundary

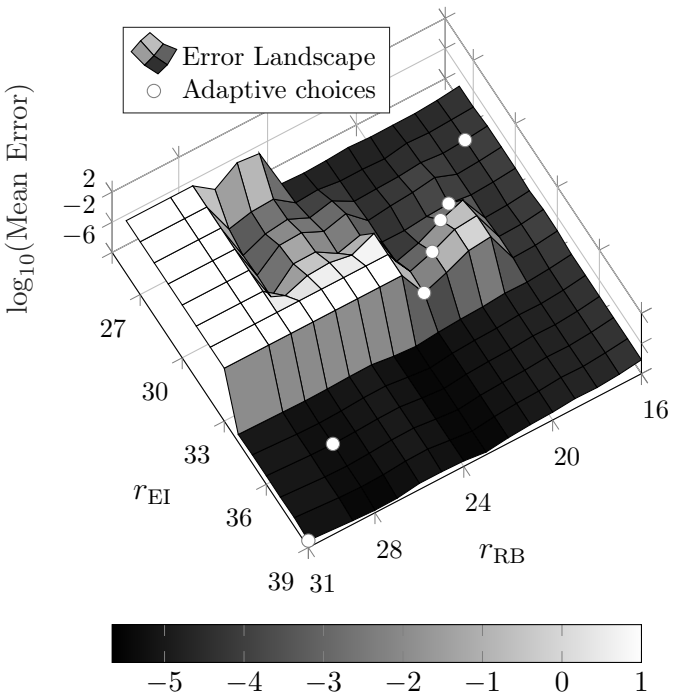
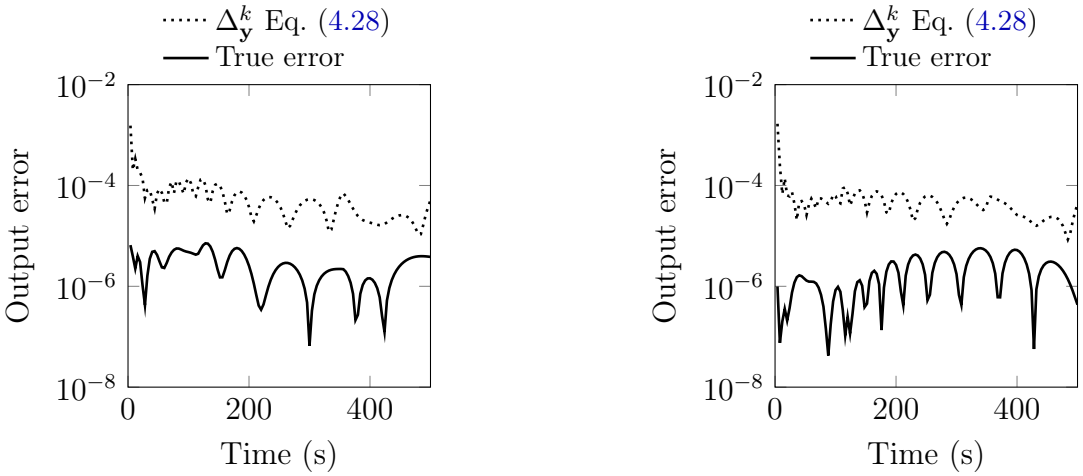


Figure 4.6.: Error landscape for the DECREASE procedure for the FBC example.



(a) Increasing.

(b) Decreasing.

Figure 4.7.: Final hyperreduced ROM error over all time for the FBC.

conditions are given as

$$\begin{aligned} \frac{\partial g}{\partial t} + g\frac{\partial g}{\partial z} &= \mu \frac{\partial^2 g}{\partial z^2} + s(z,t), \\ g(0,t) &= 0, \\ \frac{\partial g(1,t)}{\partial z} &= 0, \end{aligned} \tag{4.44}$$

where $g := g(z,t) \in \mathbb{R}$ is the state variable. $s(z,t)$ is the source/input term, μ is the viscosity. The output is taken at the last spatial point in the domain: $y = z(1,t)$. We consider $s(z,t) \equiv 1$. The initial condition is defined as $g(z,0) := 0$.

The simulation parameters are listed in Table 4.3. A training set Ξ is formed by 100 log-uniformly distributed samples in the parameter domain $\mathcal{P} := [0.0005\,, 1]$. The model has $N = 500$ equations after discretization in space. We employ the central difference scheme for both the diffusion and convection terms. A first-order IMEX Euler method is used to discretize the time variable. We make use of EIM (see Algorithm 2.5) to efficiently evaluate the nonlinearity. The EIM tolerance ϵ_{EI} is set to 10^{-10}. A time step of $\Delta t = 4 \cdot 10^{-4}$ was used, with the snapshots collected every 10^{th} time step. In Fig. 4.8, we compare the convergence of Algorithm 2.8 to that of Algorithm 4.2. It can be seen that using the latter leads to a much quicker convergence of the greedy loop: 10 iterations as compared to 16 iterations. We plot the convergence of the maximal error estimator Δ_{max} in Eq. (4.38) and the corresponding true error ϵ_{max} in Eq. (4.42), for both algorithms. Recall that, the maximal errors are defined over all the parameters in the training set Ξ. The improved convergence of Algorithm 4.2 is a direct consequence of enriching the basis in an adaptive manner. Further, in Fig. 4.9a, we plot the adaptive increments of the RB, EIM basis vectors. Starting from a value of 1 for each, we can see that the biggest jumps are at the first few steps where the maximal estimated output error Δ_{max} is large. Subsequent steps moderate the number of basis vectors to be added, as the algorithm converges. We end up with a final value of $(r_{\text{RB}}, r_{\text{EI}}) = (14, 40)$ for the RB, EIM basis respectively. As for the standard implementation, where the

Table 4.3.: Simulation parameters for the Burgers' equation.

Interpolation	N	$[0, T]$ (s)	Parameter training set (Ξ)	tolerance ϵ	ϵ_{EI}
EIM	500	[0, 2]	100 log-uniformly distributed samples in [0.0005,1]	10^{-3}	10^{-10}

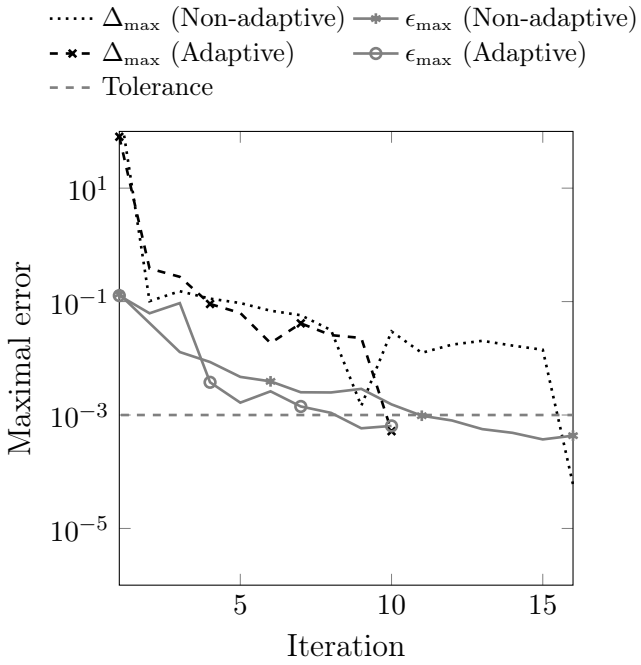


Figure 4.8.: Burgers' equation: convergence of the greedy algorithms - Algorithm 2.8 vs. Algorithm 4.2.

EIM basis is precomputed outside the greedy loop, the resulting ROM has dimension $(r_{\text{RB}}, r_{\text{EI}}) = (16, 154)$. Thus, our proposed algorithm not only produces a ROM that meets a certain tolerance, but also leads to a more compact ROM. Figure 4.9b compares the effectivities of the original error estimator from [214] (Eq. (4.37)) and the newly proposed error estimator in Eq. (4.38). For this example, the proposed error estimator performs slightly better.

Fig. 4.10a shows the output $\mathbf{y}(t)$ of the FOM and $\widetilde{\mathbf{y}}(t)$ of the ROM obtained using Algorithm 4.2 at viscosity $\mu = 5 \cdot 10^{-4}$. The ROM solution is nearly indistinguishable from the FOM solution. In Fig. 4.10b we plot the number of RB, EIM basis vectors as a function of the iteration number. Note that, both RB and EIM bases start from only one basis vector at the first iteration.

In Table 4.4, we show the runtime taken for Algorithms 2.8 and 4.2 till convergence. The adaptive algorithm needs much less time. The reduced runtime of the adaptive approach is mainly contributed by the reduced number of FOM simulations. For the inf-sup constant (the smallest singular value of the system matrix) we apply radial basis function interpolation. From Table 4.5, it is clear that the RBF approach is faster as compared to using SVD to determine the smallest singular value of the system matrix.

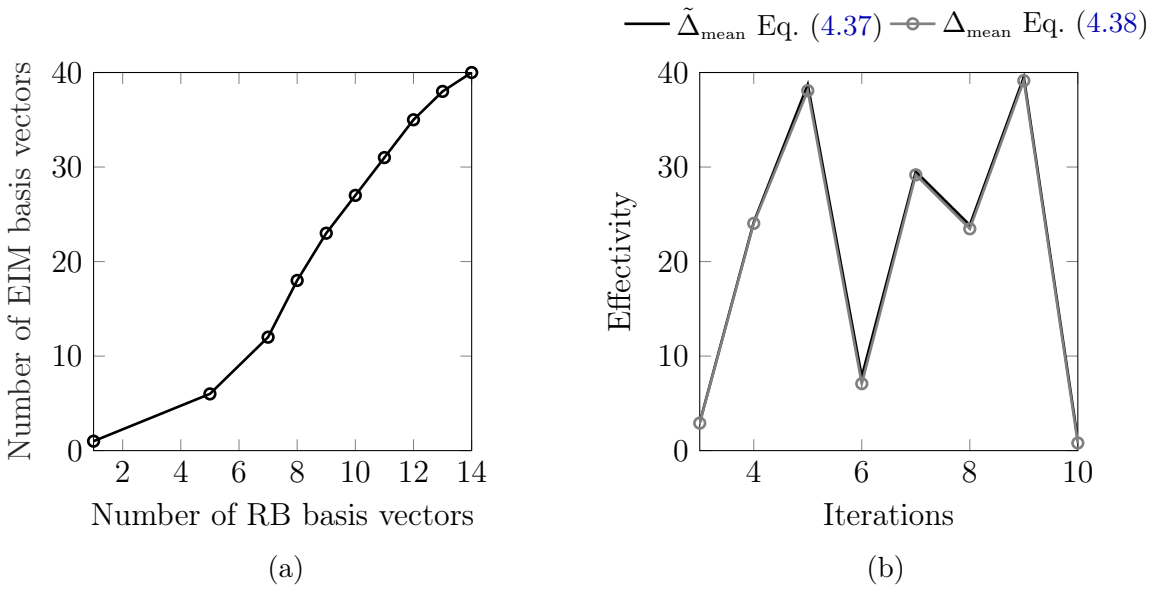


Figure 4.9.: Algorithm 4.2 for the Burgers' equation (a) Adaptive increment of RB vs. EIM basis vectors. (b) Effectivity: Original vs. modified estimator.

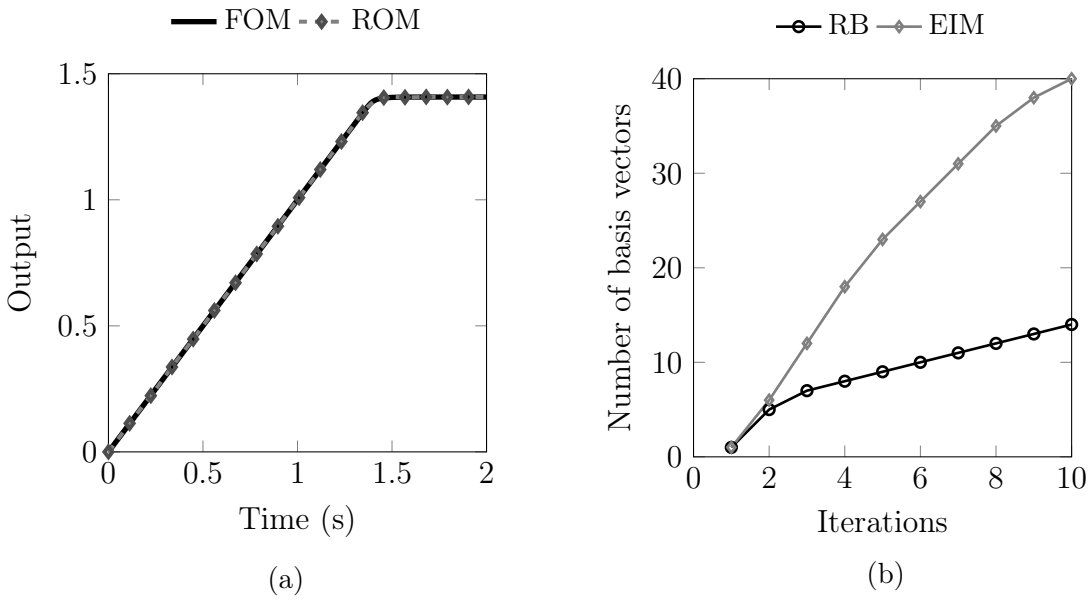


Figure 4.10.: Burgers' equation (a) Output at $\mu = 5.10^{-4}$. (b) RB, EIM basis vs. iteration number.

One can imagine, the savings in time would be much more significant for large-scale systems with $N \gg 500$.

Table 4.4.: Runtime for the Burgers' equation: Adaptive vs. non-adaptive.

Method	runtime (s)
Adaptive	606
Non-adaptive	933

Table 4.5.: Runtime for Burgers' equation: inf-sup constant computed over training set.

Method	runtime (s)
SVD	3.7
RBF	0.4

4.4. Conclusion

This chapter considered *a posteriori* error estimation and adaptivity for parametric nonlinear dynamical systems. We proposed a new *a posteriori* output error estimator in Section 4.2 that overcomes the disadvantages faced by an existing error estimator in literature. Furthermore, we discussed efficient computational strategies for computing the proposed error estimator. In Section 4.3 we discussed an adaptive basis enrichment scheme that makes use of the proposed output error estimator, for non-parametric and parametric systems. The adaptive basis enrichment scheme is able to iteratively enrich the RB (or POD) and EI bases based on a greedy strategy. The main benefits offered by the adaptive basis enrichment scheme are: significant reduction in offline cost and a better balance in the RB (POD) and EI bases, resulting in a more stable ROM. These advantages were numerically illustrated using two benchmark examples.

CHAPTER 5

ADAPTIVE TRAINING SET SAMPLING AND FULLY ADAPTIVE RBM

Contents

5.1. Adaptive Sampling for the Training Set in RBM

In Chapter 4, we introduced an approach for adaptive basis enrichment (Algorithm 4.2) in order to improve the offline computational cost for the RBM. We illustrated our approach on a one-parameter Burgers' equation. However, when the number of samples in the training set Ξ is large, the proposed approach may still incur a considerable cost.

Usually, the choice of a training set is non-trivial. On the one hand, if it includes too few parameters, the original solution manifold may not be adequately represented, leading to a poor ROM with large error. On the other hand, if it contains too many samples, the offline time can be prohibitively long. When the PDE involves several parameters, properly defining the training set can lead to a severe computational issue. In this chapter, we consider a further step of adaptivity, viz., *adaptive training set sampling*. Instead of fixing the training set involved in the RBM *a priori*, we adaptively update it as a part of the POD-Greedy algorithm.

Due to the criticality of the training set choice, a number of approaches have been considered within the RBM community to enable a good sampling of the training set. The earliest work to consider an adaptive sampling of the training set was [187], where the author proposes a Multi-Stage Greedy algorithm. The algorithm is run several times over randomly sampled small training sets to generate the RB projection matrix. Then, the ROM is tested over a much larger training set and the greedy algorithm is re-run only on those points failing the tolerance criterion. The authors of [105] address the issue of large training sets by two approaches. The first one is a procedure to monitor the error over an additional validation parameter set. If a large error is detected, then the training set is further refined, either uniformly or locally. The second approach, similar to the ones presented in [75, 76], is based on partitioning the parameter domain adaptively, and generating local bases $\mathbf{V}$ for each partition. Other approaches for adaptive training set sampling include [113] and [134]. [113] introduces the Adaptively Enriching Greedy algorithm where the authors propose a saturation criterion which is used to systematically remove parameters from a randomly-sampled training set. New random parameters are then added to the current training set. A larger training set serves as a safety check mechanism at every iteration. However, it may not be efficient, in general, to estimate a robust saturation criterion. In [115], the authors consider a two-stage approach that uses the analysis of variance (ANOVA) expansion [192] together with parameter domain decomposition to address training set complexity. The work [134] considers an anisotropic sampling of the parameter domain using an empirical norm derived from the truncated Hessian of the solution vector with respect to the parameter. No explicit partition of the parameter domain is considered. However, the basis vectors are determined at the online stage. Moreover, the method needs to compute the Hessian at each point in the training set in order to define a distance metric which is subsequently used to add more samples to the training set. The calculation of the Hessian can be very expensive, especially for time-dependent problems. The authors of [59] perform an eigendecomposition of the Hessian matrix of the output variable with respect to the parameter to identify a small subspace of the high-dimensional parameter space by truncation. The parameters that constitute the training set for the RBM are then sampled from the identified eigenspace. The method proposed in [125] makes only a subset of the finely sampled training set active at a given iteration of the greedy algorithm. A recent extension of this work [124] proposes hybrid strategies combining the ideas from [113, 187]. Different strategies are proposed to identify the set of active parameters.

An approach based on Kriging interpolation and clustering is proposed in [156], to tackle the problem of high-dimensional parameter spaces. An interpolant of the residual norm is calculated over a fine grid of parameters. Then, k-means clustering is used to

identify parameters that have high probability of presenting larger errors. In [195], the so-called active subspace [62] of the parameter space is identified by relying on gradient information of the output with respect to the parameter. Both these works are limited to scalar-valued outputs and steady problems.

Most of the existing work related to adaptive training set sampling focuses on steady or quasi steady-state problems. To the best of our knowledge, only [75, 105, 156] address training set adaptivity for time-dependent problems. The works [75, 105] propose a localization strategy that involves constructing multiple ROMs over local parameter domains, while [156] considers adaptively enriching a coarse training set by observing a cheap error surrogate over a fine training set.

We propose two different strategies to address the issue of training set sampling for the RBM. The first is an extension of the approach proposed in Section 3.4.5 to the RBM. It involves using an RBF-based surrogate model of the error estimator to adaptively update the training set, starting from a few samples. Two closely related works to our approach are [113, 156]. We highlight crucial distinctions of our approach compared to [113, 156] in the next section. Our second strategy for training set sampling takes a diametrically opposing view. We start from a finely sampled training set consisting of many parameter samples. From this, we identify a set of parameter samples that contribute the most to approximating the solution subspace, which will constitute the training set of the greedy algorithm. We do this by means of sparse sampling strategies based on the DEIM and its variants and also the pivoted QR decomposition.

Both our proposed strategies for training set sampling are tightly integrated with the adaptive basis enrichment scheme for the RBM (Algorithm 4.2). Thus, together they result in a fully adaptive POD-Greedy algorithm with minimal user interference, which (a) adaptively identifies a robust ROM, and (b) leads to considerable reduction in the offline computational costs. This is in perfect alignment with our stated objective of *automatic model order reduction* (see Section 1.4).

We present the combined RBMEI algorithm with adaptive basis enrichment and adaptive training set using the error surrogate in Section 5.2.1. We test it on several numerical examples in Section 5.2.2. Then, in Section 5.3 we present the combined RBMEI algorithm together with the subsampling-based strategy. Furthermore, we discuss two efficient sparse sampling strategies in Section 5.3.3. These methods are then validated using two benchmark examples in Section 5.3.4.

The material covered in this chapter is based on the works [54, 55].

5.2. Adaptive Training Set Sampling using Surrogate Error Estimator

The approach we propose here is based on interpolating a sharp *a posteriori* output error estimator. Recall that, for linear systems using frequency-domain MOR methods, a similar idea was discussed in Section 3.4.5 for adaptive ROM construction. In this section, we extend the approach to the RBM. For the standard RBM (Algorithm 2.8), we introduce a method based on a surrogate error estimator to efficiently sample the parameter domain, such that the training set is adaptively updated, starting from a

coarse set with a small number of parameter samples. The RBM is initialized with a coarse training set. We update this set iteratively by adding or removing points from it. At each iteration of the greedy algorithm, we estimate the error of the ROM at every parameter in the coarse training set. Note that the evaluation of the estimator does not involve the FOM. In order to further reduce the computational costs, we then interpolate the estimated error over a fine training set and use the interpolant as the surrogate estimator to estimate the ROM error over a fine training set. That is, at each iteration of the greedy algorithm, the ROM errors at the parameters in the fine training set are checked by the surrogate estimator. Those parameters corresponding to large values of the surrogate estimator are selected and added to the coarse training set. The surrogate estimator is much cheaper to compute than the error estimator. Therefore, using the former instead of the latter to check the ROM error over the fine training set, will reduce the computational cost. Additionally, if any parameter in the coarse training set achieves the required ROM accuracy, we remove it from the coarse training set. Such an approach is able to construct a small, representative training set by fully exploring the parameter domain with reduced computational cost.

As previously mentioned, the ideas proposed in [113, 156] are closely aligned with the approach we propose. Therefore, we briefly discuss how our approach differs from these earlier methods.

- We use a sharp *a posteriori* error estimator with the modified output term Eq. (4.30) on a coarse training set and a cheaply computable error surrogate on the fine training set. No cheap error estimator is used on the fine training set in [113], potentially leading to a larger computational effort.

- At each iteration of the greedy algorithm, a saturation assumption is introduced in [113] in order to avoid calculation of the ROM at certain parameters. However, this requires estimation of the saturation constant which needs to be defined *a priori* by the user. In our method, no such constant needs to be estimated.

Furthermore, when compared to [156]:

- We consider both addition and removal of parameter samples from the coarse training set. The method in [156] is restricted to adding new samples only.

- The use of Kriging interpolation involves the estimation of several hyperparameters. This can be a computational bottleneck [199]. In the case of the RBF, there is only one free parameter at most. If using some special kernel functions, e.g. polyharmonic spline kernels, there are no free parameters to tune.

- Finally, a nonlinear model was considered in [156], in order to demonstrate the adaptive sampling approach. An offline hyperreduction step was used to reduce the complexity of the online nonlinearity evaluations, where a second training set for the nonlinear function is used in the hyperreduction step, and is likely to be separately given and fixed. In our approach, we employ Algorithm 4.2. This avoids the need for a potentially separate training set for the hyperreduction phase. Instead, the EI basis generation is carried out within the same greedy loop. In this way, our method is a fully adaptive approach.

5.2.1. Fully Adaptive RBMEI algorithm with Surrogate Error Estimator

In this section, we extend and integrate the surrogate error estimator approach discussed in Section 3.4.5 to Algorithm 4.2.

For the POD-Greedy algorithm Algorithm 2.8 (or its extension Algorithm 4.2), Step 15 (or Step 22) involves solving the ROM at every parameter in the training set Ξ. Whenever the number of samples in Ξ is high, repeatedly solving for the ROMs at each iteration can quickly become expensive. Therefore, we construct a *surrogate* for the error estimator Eq. (4.30) by learning the map $\Delta_{\mathbf{y}} : \boldsymbol{\mu} \in \mathbb{R}^{N_p} \to \mathbb{R}$. We consider a coarse training set $\Xi_c := \{\boldsymbol{\mu}_1^c, \ldots, \boldsymbol{\mu}_{n_c}^c\}$ with n_c parameter samples. We also sample a fine training set $\Xi_f := \{\mathring{\boldsymbol{\mu}}_1, \ldots, \mathring{\boldsymbol{\mu}}_{n_f}\}$ with $n_f \gg n_c$ samples. We compute the error estimator only over Ξ_c and obtain the data $(D, \boldsymbol{\Lambda})$ for RBF interpolation with $D = \{\Delta_{\mathbf{y}}(\boldsymbol{\mu}), \forall \boldsymbol{\mu} \in \Xi_c\}$ and $\boldsymbol{\Lambda} = \Xi_c$. Based on this input data, the RBF interpolant is obtained by solving Eq. (2.43). The resulting interpolant $\chi(\boldsymbol{\mu})$ is then evaluated at all parameter samples present in Ξ_f. Note that, the cost of forming the RBF interpolant scales as $O((n_c + \nu)^3)$ and evaluating the interpolant incurs costs that scale as $n_f \cdot O(n_c + \nu)$ (see the analysis in Section 3.4.5.1). Since both n_c, ν are small, the cost of evaluating the surrogate model $\chi(\boldsymbol{\mu})$ is much lower than the cost of computing the error estimator $\Delta_{\mathbf{y}}$. The coarse training set Ξ_c is updated at each iteration by adding and/or removing samples. To add new parameters, we monitor the surrogate estimator $\chi(\boldsymbol{\mu})$ for the samples in Ξ_f. We then add those n_{add} parameters having the largest errors measured by $\chi(\boldsymbol{\mu})$. The value of n_{add} is user-defined. It can also be adaptively determined, for instance, by the following heuristic choice:

$$n_{\text{add}} := \log_{10} \frac{\max\limits_{\boldsymbol{\mu} \in \Xi_c} \Delta_{\mathbf{y}}(\boldsymbol{\mu})}{\epsilon}.$$

We observe the coarse training set to identify those parameter samples with $\Delta_{\mathbf{y}}(\boldsymbol{\mu}) < \epsilon$. These parameters are then removed from Ξ_c. This ensures that the ROM at these parameters need not be solved, thus controlling the offline computational costs. Algorithm 5.1 sketches the pseudocode for the proposed adaptive parameter sampling approach using surrogate error estimator.

Algorithm 5.1 is similar to Algorithm 4.2. The main differences are: we use two separate training sets Ξ_c, Ξ_f for the former. Moreover, the error estimator $\Delta_{\mathbf{y}}$ is evaluated only at the samples in the coarse training set in Step 22. This ensures that the offline cost is kept low since it avoids solving the ROM for every parameter in a finely sampled training set. In Step 23, the RBF surrogate for the estimator is determined. The error evaluated using the surrogate $\chi(\boldsymbol{\mu})$ is used to update Ξ_c in Steps 24-26. A similar computational cost analysis as done in Section 3.4.5.1 applies here as well.

Algorithm 5.1: `ADAPTIVE_RBMEI_TS1`
Computes a ROM using the Adaptive POD-Greedy algorithm in combination with hyperreduction and adaptive training set sampling using surrogate error estimator.

Input: System matrices and nonlinear term: $\mathbf{E}, \mathbf{A}, \mathbf{B}, \mathbf{C}$ and $\mathbf{f}$, Training sets Ξ_c, Ξ_f, EI tolerance ϵ_{EI}, ROM tolerance ϵ, zoa ϵ^*, Deflation tolerance ϵ_{def}, Maximum iterations *iter_max*.

Output: Reduced system matrices $\widehat{\mathbf{E}}, \widehat{\mathbf{A}}, \widehat{\mathbf{B}}, \widehat{\mathbf{C}}$ and EI quantities $\boldsymbol{\mathcal{U}}$, $\mathcal{I}$.

For non-parametric t-dual system Eq. (4.2), obtain $\widetilde{\mathbf{x}}_{\text{du}}$ using a Krylov-space method.
Use RBF interpolation to approximate *inf-sup* constants as $\widetilde{\sigma}_{\min}(\boldsymbol{\mu})$ for all $\boldsymbol{\mu} \in \Xi_c$.
Initialization: $\mathbf{V} = [\,]$, $\mathbf{V}_{\text{du}} = [\,]$, $\mathbf{U} = [\,]$, $\mathcal{I} = \varnothing$, $err\_max = 1 + \epsilon$, $err\_max\_dual = 1 + \epsilon$, $iter = 1$, $\boldsymbol{\mu}^*, \boldsymbol{\mu}^*_{\text{du}}$ (chosen randomly from Ξ_c), $r_{\text{RB}} = r_{\text{EI}} = 1$.
while $err\_max \notin zoa$ and $iter \leq iter\_max$ **do**
  if $r_{\text{RB}} > 0$ **then**
    Solve FOM Eq. (2.11) at $\boldsymbol{\mu}^*$; obtain snapshot matrix $\mathbf{X}(\boldsymbol{\mu}^*)$.
    if $iter == 1$ **then**
      $\mathbf{V} = \texttt{POD}(\mathbf{X}, r_{\text{RB}})$
    else
      Compute $\overline{\mathbf{X}} := \mathbf{X} - \mathbf{V}\mathbf{V}^{\mathsf{T}}\mathbf{X}$.
      $\mathbf{V}_{\text{POD}} := \texttt{POD}(\overline{\mathbf{X}}, r_{\text{RB}})$.
      $\mathbf{V} = \texttt{orth\_def\_mat}(\mathbf{V}, \mathbf{V}_{\text{POD}}, \epsilon_{\text{def}})$. `/* see Step 15 of Algorithm 2.4 */`
    end
  else
    Remove the last r_{RB} columns from $\mathbf{V}$.
  end
  Form snapshot matrix for the nonlinear term: $\mathbf{F} := [\mathbf{F} \;\; \overline{\mathbf{f}}(\overline{\mathbf{X}})]$.
  Update EI basis and interpolation points $\mathbf{U}, \mathcal{I}$ using the subroutine `update_ei`. `/* see Line 26 of Algorithm 4.2 */`
  For a parametric t-dual system, update dual RB matrix $\mathbf{V}_{\text{du}}$ using the subroutine `update_dual`. `/* see Line 29 of Algorithm 4.2 */`
  $iter = iter + 1$.
  Assemble the ROM Eq. (2.36) through Galerkin projection and hyperreduction using $\mathbf{V}, \mathbf{U}$.
  Solve the ROM Eq. (2.36) and compute $\Delta_{\mathbf{y}}(\boldsymbol{\mu})$ in Eq. (4.30) for all $\boldsymbol{\mu} \in \Xi_c$.
  Define $D := \{\Delta_{\mathbf{y}}(\boldsymbol{\mu}), \boldsymbol{\mu} \in \Xi_c\}$ and $\boldsymbol{\Lambda} := \Xi_c$, solve Eq. (2.43) to obtain RBF surrogate $\chi(\boldsymbol{\mu})$ for all $\boldsymbol{\mu} \in \Xi_f$.
  Update Ξ_c: remove $\boldsymbol{\mu} \in \Xi_c$ for which $\Delta_{\mathbf{y}} < \epsilon$.
  Find samples $\{\boldsymbol{\mu}^{(1)}, \ldots, \boldsymbol{\mu}^{(n_{\text{add}})}\} \in \Xi_f$ resulting in largest error for the surrogate $\chi(\boldsymbol{\mu})$.
  Update Ξ_c: $\Xi_c = \left[\Xi_c \cup \{\boldsymbol{\mu}^{(1)}, \ldots, \boldsymbol{\mu}^{(n_{\text{add}})}\}\right]$.
  Find $\boldsymbol{\mu}^* = \arg\max_{\boldsymbol{\mu} \in \Xi_c} \Delta_{\mathbf{y}}(\boldsymbol{\mu})$
  Set $err\_max = \Delta(\boldsymbol{\mu}^*)$.
  Update $r_{\text{RB}}, r_{\text{EI}}$ using the subroutine `adapt_basis_update`. `/* see Line 35 of Algorithm 4.2 */`
end

5.2.2. Numerical Examples

In this section, we validate the proposed adaptive training set sampling method using three examples. The first is the nonlinear Burgers' equation model with one parameter, already introduced in Section 4.3.7.2. The second example is a two-parameter linear convection-diffusion model. The last one is a three-parameter model of a microthruster unit, which was investigated in Section 3.4.6.2.

For the three examples, we implement the Adaptive RBMEI algorithm with training set sampling using the surrogate error estimator (Algorithm 5.1). For comparison, we also present the results of Algorithm 4.2 which uses a fixed training set. In both cases, the error estimator with modified output term (Eq. (4.30)) is employed. To accurately quantify the effects of our sampling, we make use of a test set Ξ_{test} that is different from the training sets (Ξ, Ξ_c, Ξ_f). We plot at the end of each greedy iteration, the maximal error $\Delta_{\max}$ (Eq. (4.38)) over the test set. We compute the true errors ϵ_{mean} (Eq. (4.41)) of ROMs resulting from Algorithms 4.2 and 5.1 over the samples present in Ξ_{test}. If not particularly pointed out, the initial coarse training set Ξ_c for Algorithm 5.1 is the same as the fixed training set for Algorithm 4.2.

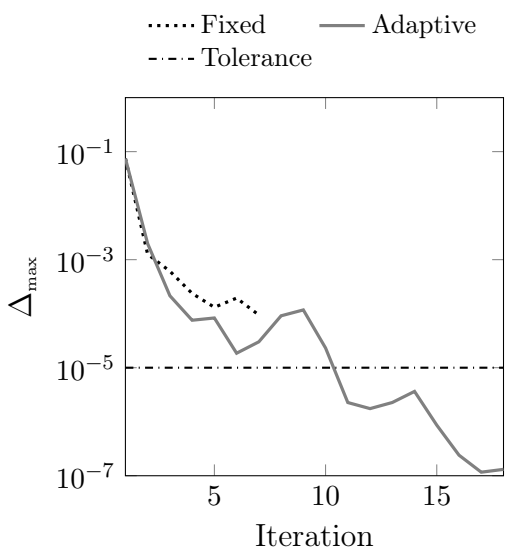
All numerical results in this section were obtained using MATLAB®2015a, on a laptop with INTEL®CORE™i5-7200U @ 2.5 GHZ, with 8 GB of RAM.

5.2.2.1. Burgers' Equation

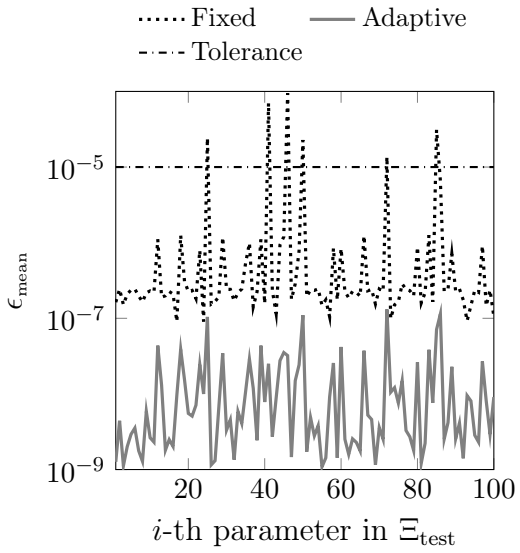
The model equations are same as those in Equation (4.44). We adopt the same space and time discretization. The tolerance is set to be $\epsilon = 10^{-5}$ and the parameter domain we consider for the viscosity is $\mu \in [0.001, 1]$. For Algorithm 4.2 we consider a training set Ξ with 10 randomly chosen samples[1]. The coarse training set for Algorithm 5.1 is set to be the same as Ξ, while the fine training set Ξ_f contains 300 randomly sampled[2] parameters. The test set Ξ_{test} used for validation contains 100 parameter samples[3]. To construct the surrogate error estimator $\chi(\boldsymbol{\mu})$ in Step 23 of Algorithm 5.1, the IMQ kernel function is used for the RBF interpolation. LOOCV has been applied to specify the shape parameter for the IMQ kernel. Since the model is nonlinear, the DEIM is used in order to efficiently compute the nonlinear term.

Applying Algorithm 4.2 to the Burgers' equation results in a convergence after 7 iterations and produces a ROM of dimension $n = 7$. Algorithm 5.1 takes 18 iterations to converge and the resulting ROM dimension is $n = 18$. In Figure 5.1a, the decay of the error estimator over the test set Ξ_{test} is shown for every iteration of both Algorithms 4.2 and 5.1. The reason to show the error decay over the test set is motivated by the fact that Algorithms 4.2 and 5.1 use different training sets in their respective greedy algorithms. The error measured over the test set serves to uniformly validate the accuracy at each iteration of either approaches. The earlier termination of Algorithm 4.2 is due to the insufficient density of the training set used, which contains only 10 samples. This has resulted in *overfitting*. This is evident from Figure 5.1b, where we see that the

[1] generated using `rng` command from MATLAB®with seed 12 and random number generator `twister`.
[2] generated using `rng` command from MATLAB®with seed 114 and random number generator `twister`.
[3] generated using `rng` command from MATLAB®with seed 200 and random number generator `simdTwister`.



(a) Greedy Convergence.



(b) Mean Error ϵ_{mean} of the final ROM over Ξ_{test}.

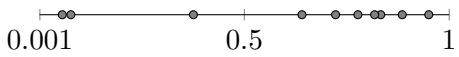
Figure 5.1.: Results for the Burgers' equation using Algorithm 4.2 (Fixed) and Algorithm 5.1 (Adaptive).

ROM resulting from Algorithm 4.2 fails to meet the desired tolerance over the test set. However, the ROM obtained from Algorithm 5.1 is successful in meeting the tolerance uniformly over the test set. In Figure 5.2, the evolution of the coarse training set Ξ_c for different iterations is shown. It is seen that the algorithm tends to pick parameters in the low-viscosity regions (close to 0.001), as expected, since the solution of the PDE tends to be 'less smooth', requiring more basis functions to approximate.

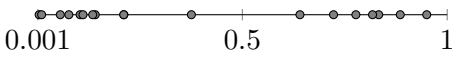
5.2.2.2. Convection-Diffusion Equation

The second example we consider is a 1-D model of brain transport, originally discussed in [18] and later in [97, 214]. The transport is modelled as a linear convection-diffusion PDE defined in the spatial domain $\Omega := [0, 1]$ and for time $t \in [0, 1]$,

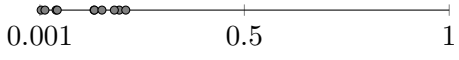
$$\frac{\partial g}{\partial t} = \mu_1 \frac{\partial^2 g}{\partial z^2} + \mu_2 \frac{\partial g}{\partial z} - \mu_2, \tag{5.1}$$



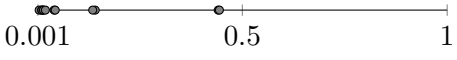
(a) Ξ_c at iteration 1.



(b) Ξ_c at iteration 5.

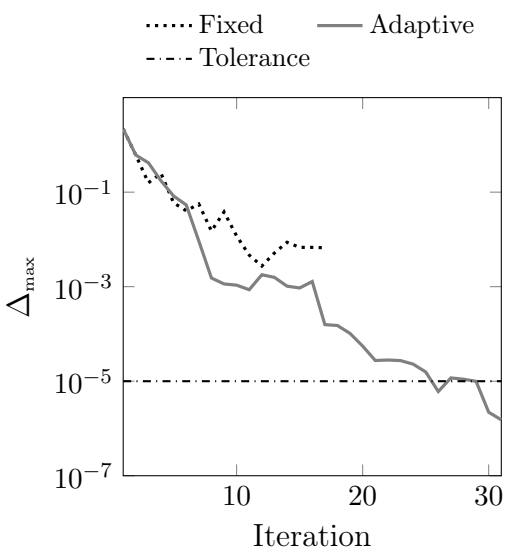


(c) Ξ_c at iteration 15.

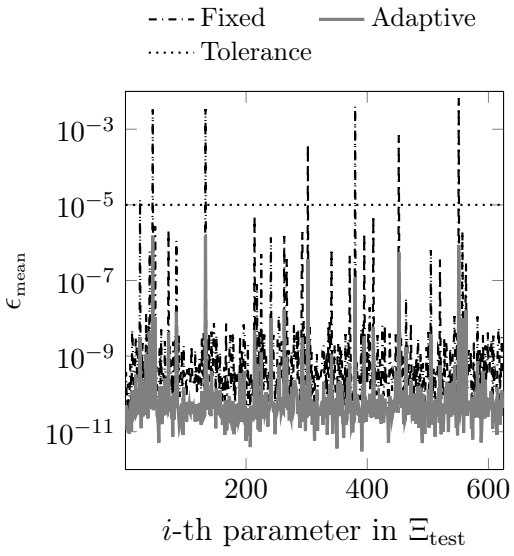


(d) Ξ_c at iteration 18.

Figure 5.2.: Burgers' equation: training set evolution for Algorithm 5.1.



(a) Greedy Convergence.



(b) Mean Error ϵ_{mean} of the final ROM over Ξ_{test}.

Figure 5.3.: Results for the Convection-diffusion equation using Algorithm 4.2 (Fixed) and Algorithm 5.1 (Adaptive).

where $g(z, t, \mu_1, \mu_2)$ is the field variable, $z \in \Omega$ is the spatial variable, and the two parameters $(\mu_1, \mu_2) \in \mathcal{P} := [0.001\,, 1] \times [0.5\,, 5]$ are the diffusion and convection constants, respectively. The boundary conditions are given by,

$$g(z, 0, \mu_1, \mu_2) = \begin{cases} 1, & z \leq 0.5 \\ 0, & \text{otherwise} \end{cases}, \qquad g(0, t, \mu_1, \mu_2) = g(1, t, \mu_1, \mu_2) = 0.$$

We discretize the equation using the FDM on a grid yielding $N = 800$. The output is calculated as the average value of the state in a small interval Ω_o centered around the midpoint of the domain at $z = 0.5$.

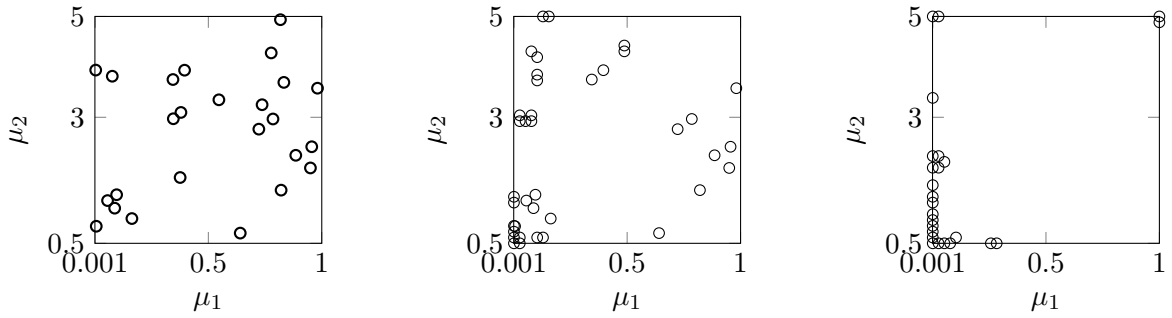
$$y(t) := \frac{1}{|\Omega_o|} \int_{\Omega_o} g(z, t, \mu_1, \mu_2) dz, \qquad \Omega_o := [0.495\,, 0.505].$$

We carry out the tests using Algorithms 4.2 and 5.1, just as in the case of the previous example. The tolerance is set as $\epsilon = 10^{-5}$. For Algorithm 4.2, we define Ξ as a set including 25 random samples from the parameter domain $\mathcal{P}$[4]. The fine training set Ξ_f for Algorithm 5.1 includes 1600 equidistant samples in $\mathcal{P}$. The test set Ξ_{test} consists of 625 random samples[5]. The error surrogate model $\chi(\boldsymbol{\mu})$ in Algorithm 5.1 is constructed using IMQ kernel function, where cross validation (LOOCV) has been applied to specify the shape parameter.

Figure 5.3 shows the decay of the maximal estimated error ($\Delta_{\max}$) over Ξ_{test}, computed at each iteration of Algorithm 4.2 and Algorithm 5.1. It is clear that adaptively

[4]picked using the `rng` command in MATLAB® and making use of `twister`, with a seed of 112.

[5]generated using the same random number generator as for the Burgers' equation example, viz., `simdTwister` with a seed of 200.



(a) Ξ_c at iteration 1. (b) Ξ_c at iteration 20. (c) Ξ_c at iteration 31.

Figure 5.4.: Convection-diffusion equation: training set evolution.

Table 5.1.: Convection-diffusion example: Runtime comparison between Algorithm 4.2 and Algorithm 5.1.

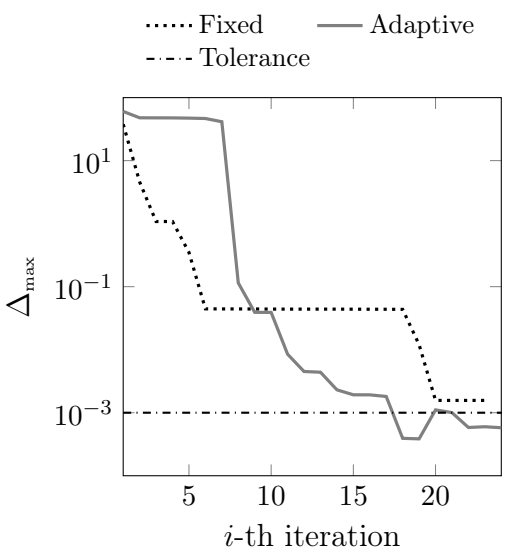
Algorithm	Runtime (seconds)
Fixed training set (Algorithm 4.2)	74.09
Adaptive training set (Algorithm 5.1)	41.52

enriching the training set leads to orders of magnitude faster convergence to the required accuracy. For Algorithm 4.2, the ROM dimension is $n = 38$ and for Algorithm 5.1, it is $n = 63$. Figure 5.3b plots the mean true error ϵ_{mean} (Eq. (4.41)) of the final ROM at every parameter in the test set Ξ_{test}, for the case of an adaptive training set and a fixed training set. We see that for some test parameters, the required tolerance is not met by the ROM computed using the fixed training set. In Figure 5.4, the evolution of the training set is shown at different stages of Algorithm 5.1. It can be seen that samples from the left boundary of the parameter domain are added or retained. This has physical sense as this corresponds to the lower viscosity regions where the convective part of the solution dominates and the solution is 'less smooth'. Thus, more basis functions are required for a good approximation.

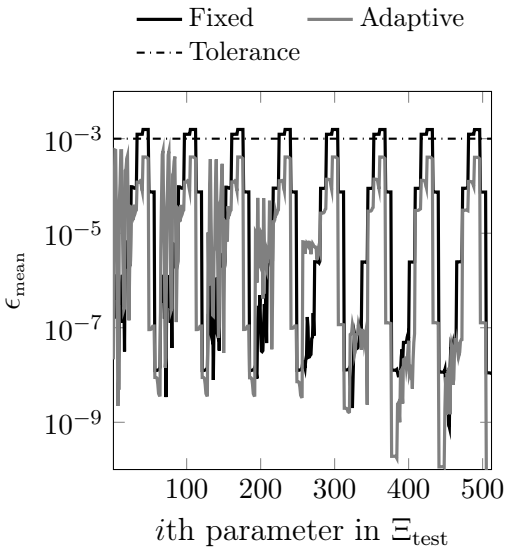
In order to achieve a better ROM by using the fixed training set, we tried a Ξ with 225 random samples for Algorithm 4.2, leading to a ROM with error below the tolerance. In Table 5.1, we provide the runtime results of both algorithms, where Algorithm 4.2 uses the refined training set with 225 samples. On the other hand, the initial coarse training set and the fine training set for Algorithm 5.1 remain the same. For this case, Algorithm 5.1 with adaptive training set sampling outperforms Algorithm 4.2, though both approaches produce accurate ROMs.

5.2.2.3. Thermal Model

The third and final example is that of the thermal model which has three parameters. This example was previously introduced in Section 3.4.6.2 in the frequency-domain representation. The dimension of the system is $N = 4,257$. The parameters $\{\mu_i\}_{i=1}^3$ are the heat coefficients at the top, bottom and side (x-, y- and z-directions) of the



(a) Greedy Convergence.



(b) Mean Error ϵ_{mean} of the final ROM over Ξ_{test}.

Figure 5.5.: Results for the Thermal model using Algorithm 4.2 (Fixed) and Algorithm 5.1 (Adaptive).

microthruster unit, respectively. All three have a range between $[1\,, 10^8]$ in our simulations. The output in this case is the temperature at the center of the polysilicon heater in the unit, which corresponds to the first row of the output matrix $\mathbf{C}_7(\boldsymbol{\mu})$.

For this example, the initial training sets Ξ, Ξ_c corresponding to Algorithm 4.2 and Algorithm 5.1, respectively, are different. Since this is a three-parameter problem with a very large parameter domain, we use a large fixed training set Ξ with 6^3 logarithmically-spaced parameter samples for Algorithm 4.2, to have a fair comparison. For Algorithm 5.1, the initial coarse training set includes only $n_c = 10$ randomly chosen samples[6]. The fine training set Ξ_f for Algorithm 5.1 is made up of 16^3 equidistant samples, while the test set Ξ_{test} consists of 8^3 logarithmically-spaced samples. For the RBF interpolation, TPS kernel is used. No shape parameter needs to be determined. The tolerance for this model is set as $\epsilon = 10^{-3}$.

In Figure 5.5a, we show the maximal error estimator $\Delta_{\max}$ over Ξ_{test} at each iteration of Algorithm 4.2 and Algorithm 5.1, respectively. Algorithm 4.2 takes 23 iterations to converge with the fixed training set Ξ. The resulting order of the ROM is $n = 44$. However, the maximal error estimator $\Delta_{\max}$ over the test set Ξ_{test} is still above the tolerance. Algorithm 5.1 converges in 24 iterations to a ROM of dimension $n = 74$. The maximum error estimator $\Delta_{\max}$ over Ξ_{test} is below the tolerance upon convergence. We note that, in the first few iterations, Algorithm 4.2 has a faster convergence in comparison to Algorithm 5.1. However, since the training set is fixed, the convergence eventually

[6]To generate these random samples, we first generate a uniform sampling of each parameter $\mu_j, j = 1, 2, 3$, given as, $\mu_{ij} = 10^{\frac{i}{N_c/8}}$, $i = 1, 2, \ldots, N_c$, which leads to a matrix $(\mu_{ij}) \in \mathbb{R}^{N_c \times 3}$. Then, the rows of the matrix are permuted using the `rng` and `randperm` commands in MATLAB®. The seeds and random number generators used are 100 (`twister`), 120 (`combRecursive`) and 600 (`combRecursive`), respectively. In this way, we get random samples which spread through the 3-D parameter domain.

Table 5.2.: Thermal model: Runtime comparison between Algorithm 4.2 and Algorithm 5.1.

Algorithm	Runtime (seconds)
Fixed training set (Algorithm 4.2)	151.16
Adaptive training set (Algorithm 5.1)	38.76

saturates. In Fig. 5.5b, we plot the error ϵ_{mean} of the final ROM over Ξ_{test}, computed by the two algorithms. Algorithm 5.1 once again outperforms Algorithm 4.2. In Table 5.2, we provide the runtime comparison between Algorithm 4.2 and Algorithm 5.1 for the thermal model, where an obvious speed-up by Algorithm 5.1 is observed. More importantly, with the reduced runtime, Algorithm 5.1 produces a ROM with sufficient accuracy, whereas, the ROM computed by Algorithm 4.2 still does not meet the accuracy requirement. This further justifies the motivation of using adaptive sampling for models with two or more parameters, especially when the parameter domain is very large.

5.3. A Subsampling Approach for Adaptive Training Set Sampling

In this section, we introduce a second, alternate strategy based on *subsampling* for choosing the training set in the RBM. This approach is a *goal-oriented* sampling strategy that relies on the output quantity of interest. We aim at identifying the structure of the parameter dependency of the output through the empirical interpolation algorithm (DEIM) or the pivoted QR decomposition and utilize this information to find out the parameter importance. Our proposed method is applicable to both steady and time-dependent problems with vector-valued outputs. Our central contribution is a two-stage algorithm to control the cardinality of the training set. In the first stage, a low-fidelity RBM approximation of the problem is obtained using a fine training set. Then, an approximate output snapshot matrix is derived by simulating the low-fidelity ROM at all the parameter samples in the fine training set. We apply the pivoted QR decomposition or, alternatively, the DEIM and its variants to the approximate output snapshot matrix. As we shall show, this procedure identifies regions of the parameter space that have a greater contribution to the RB approximation space. The set of sparse sampling points identified by the pivoted QR decomposition or the DEIM reveal the structure of the parametric dependence of the output variable. In the second stage, the fine training set is *subsampled* based on the parameter distribution identified using the pivoted QR decomposition or the DEIM algorithm, and leads to a subsampled coarse training set. The RBM is continued over the coarse training set, until a targeted error tolerance is met.

5.3.1. Sparse Sampling Strategies

In Section 2.6, we introduced hyperreduction strategies for efficient treatment of the nonlinear term in the context of MOR. We especially focused on EIM and DEIM, two popularly used approaches in the context of POD and RBM. As was described in Algorithms 2.5 and 2.6, the main criterion to choose the EI sampling points was a greedy selection strategy based on the residual. Recently, several extensions to DEIM have been introduced, which use alternate criteria to determine the interpolation indices [69, 159]. Our proposed training set subsampling strategy relies on these. We give a brief overview of these methods below.

QDEIM The QDEIM (QR-DEIM) was introduced in [69] as an extension of the DEIM. The QDEIM method relies on a *column-pivoted* QR decomposition to identify the interpolation indices. This is different from the sequential, greedy choice of interpolation points in DEIM. QDEIM is proven to have a sharper error bound and is also computationally more efficient and straightforward to implement. The pseudocode for QDEIM is sketched in Algorithm C.1.

KDEIM Another recent variant to the DEIM is the KDEIM. The K in KDEIM refers to the k-means clustering algorithm [108]. Recall the definition of the nonlinear snapshots matrix $\mathbf{F}$ in Eq. (2.32). In the KDEIM approach, an SVD is used to obtain the EI basis $\mathbf{U}$. Following this, the k-means clustering is used to cluster the rows of $\mathbf{U}$ such that rows having a similar response are assigned to the same cluster. The standard k-means objective function is recast as a relaxed trace maximization problem which is then solved using the QR decomposition. For an in-depth treatment we refer to [159]. The procedure in Algorithm C.2 sketches the basic KDEIM scheme, taken from [159].

Gappy-POD Recall from Section 2.6 that the number of interpolation points in the EIM and the DEIM algorithm (n_I) is equal to the dimension n_{EI} of their EI basis $\mathbf{U}$. However, in many cases it is beneficial to consider $n_I > n_{\mathrm{EI}}$ interpolation points. The Gappy-POD method and other related approaches fall into this category [47, 48, 77]. The Gappy-POD approximation for any function $\mathbf{f}(\boldsymbol{\mu})$ is given in Eq. (2.35), where $\mathbf{U}^{\mathcal{J}}$ is a non-square matrix with $n_I > n_{\mathrm{EI}}$ rows. Different sampling strategies are possible to choose the n_I EI sampling points. In [159], the authors discuss two different oversampling strategies. The first approach, called Gappy-POD Eigenvector chooses new interpolation points as those leading to the largest decrease of $\|(\mathbf{U}^{\mathcal{J}})^{\dagger}\|$. This is closely related to the sampling strategy in the DEIM, where the interpolation point is chosen based on the location leading to minimizing $\|(\mathbf{U}^{\mathcal{J}})^{-1}\|_2$, with $\mathbf{U}^{\mathcal{J}} \in \mathbb{R}^{n_{\mathrm{EI}} \times n_{\mathrm{EI}}}$ being square. The second oversampling strategy, Gappy-POD Clustering, proposed in [159] can be viewed as Gappy-POD based on different interpretation of the QR decomposition as a clustering algorithm. The additional samples from $n_I - n_{\mathrm{EI}}$ interpolation indices are identified based on the mutual entropy of the columns. The pseudocodes for the Gappy-POD Eigenvector and Gappy-POD Clustering schemes are sketched in Algorithms C.3 and C.4, respectively. For a more elaborate discussion we refer to [159, 211]. In the next sections, we aim to make use of DEIM and its variants

Table 5.3.: Greedy parameters picked by RBM for the Burgers' equation.

Parameter	0.005	0.0151	0.0251	0.0352	0.0553	1
Repetitions	14	1	1	1	1	1

(QDEIM, KDEIM), as well as Gappy-POD to select important parameter samples from the parameter domain for the RBM.

5.3.2. Motivating Observations

We detail two observations that pertain to the greedy algorithm in the RBM, the DEIM algorithm as well as the QR pivoting. We shall see that these two observations have motivated us to develop a subsampling strategy for the RBM training set.

5.3.2.1. Greedy Parameters, QR Pivots, and DEIM Interpolation Points

Our first observation concerns the parameters $\boldsymbol{\mu}^*$ selected by the greedy algorithm (See Algorithm 2.8). The second observation is their resemblance to the QR pivots and the DEIM interpolation points.

From our experience, the greedy algorithm tends to repeatedly pick parameter samples from a small subset of the training set, especially for time-dependent problems. This same phenomenon has been reported in other existing works [99, 104, 106, 134, 145, 169, 214]. The solution (or output) vectors at these parameter values usually exhibit large variability. While the greedy algorithm scans through all the parameter samples in the training set, a majority of those samples are never picked. The fact that a few parameters get repeatedly picked reveals that there are still unresolved modes and hence more POD modes, corresponding to the selected parameter, are needed to get a good approximation. These few parameters picked by the greedy algorithm, usually represent solutions that are less smooth and hence are more difficult to approximate. An example of this phenomenon occurs in fluid dynamics problems where the low viscosity solutions develop shock and need a large number of POD modes to approximate. We illustrate this observation through the standard greedy algorithm applied to the discretized 1-D viscous Burgers' equation with $N = 1000$. The details of the model were presented in Section 4.3.7.2. We use 100 equispaced parameter samples from the domain $\mathcal{P} := [0.005, 1]$ to form the training set Ξ. A ROM with error below the tolerance $\epsilon = 10^{-6}$ is requested from the greedy algorithm (Algorithm 4.2). In Table 5.3, we provide the parameters picked by the greedy algorithm at each iteration from the training set. Noticeably, among the 100 parameter samples in the training set, only 6 contribute to generating the basis $\mathbf{V}$ that approximates the solution manifold. Of these 6 samples, the sample $\mu = 0.005$ is picked fourteen times. This is not surprising since this parameter corresponds to the solution vector with the smallest viscosity and is the most difficult to approximate.

Next, we make an important connection between the parameters selected by the greedy algorithm and the pivots obtained through a pivoted QR decomposition of the transpose of the output snapshot matrix defined in Eq. (5.2).

Table 5.4.: First 10 pivots for the QR decomposition of the transposed true output snapshot matrix $\boldsymbol{\mathcal{Y}}$ of the Burgers' equation.

Pivots	0.005	0.0151	0.0251	0.0352	0.0553	0.1055	0.1859	0.3166	0.7487	1

Figure 5.6.: Greedy parameters for the Burgers' equation and QR pivots of the true output snapshots matrix $\boldsymbol{\mathcal{Y}}$.

For the same viscous Burgers' equation, we collect the snapshots of the scalar-valued outputs $\mathbf{y}$ at all the parameters in Ξ into a snapshot matrix given by

$$\boldsymbol{\mathcal{Y}} := \begin{bmatrix} \mathbf{y}^0(\boldsymbol{\mu}_1) & \cdots & \mathbf{y}^K(\boldsymbol{\mu}_1) \\ \vdots & \ddots & \vdots \\ \mathbf{y}^0(\boldsymbol{\mu}_{n_s}) & \cdots & \mathbf{y}^K(\boldsymbol{\mu}_{n_s}) \end{bmatrix} \in \mathbb{R}^{n_s \times N_t}, \tag{5.2}$$

Each row of the matrix consists of the snapshots of the output at $N_t = K + 1$ time instances corresponding to a given parameter. Consider first the well-known pivoted QR decomposition of a matrix $\boldsymbol{\mathcal{D}}$ given by

$$\boldsymbol{\mathcal{D}}\Pi = \mathbf{QR} = \begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ 0 & \mathbf{R}_{22} \end{bmatrix}, \tag{5.3}$$

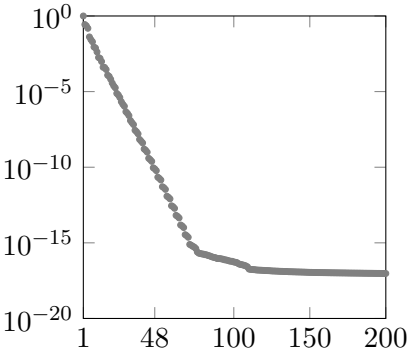
where $\mathbf{Q}$ is an orthogonal matrix and $\mathbf{R}$ is upper triangular. The pivots are given by the column permutation matrix Π. We apply the QR decomposition to $\boldsymbol{\mathcal{Y}}^\mathsf{T}$ and identify the pivots. A comparison of the parameters corresponding to the first ten pivots and the parameters selected in the greedy algorithm is shown in Table 5.4 and Fig. 5.6. Of the ten pivots, six are identical with the greedy parameters. This close connection between the pivots of the QR decomposition and the greedy parameters chosen in the RBM has been, to the best of our knowledge, discussed only in [10, 143], where the QR decomposition is interpreted as a greedy column selection procedure. Note that the application of a QR decomposition assumes the existence of the FOM solution for all the parameters in the training set. In practice, we do not have this information as part of the RBM. Instead, we propose to apply the pivoted QR decomposition to the transpose of an approximate output snapshot matrix, in order to identify *important* parameters which can then be used to subsample the fine training set in the RBM.

In Section 2.6, the usage of DEIM was discussed in the context of MOR. The DEIM algorithm uses a greedy, sparse sampling of the left singular vector matrix ($\mathbf{U}$) of the snapshots to identify interpolation points. The QDEIM approach, a variant of DEIM introduced in the previous section, performs a QR decomposition with column pivoting on $\mathbf{U}^\mathsf{T}$. This implicates a similar phenomenon as observed above: QR with pivoting could select points of importance on different demands.

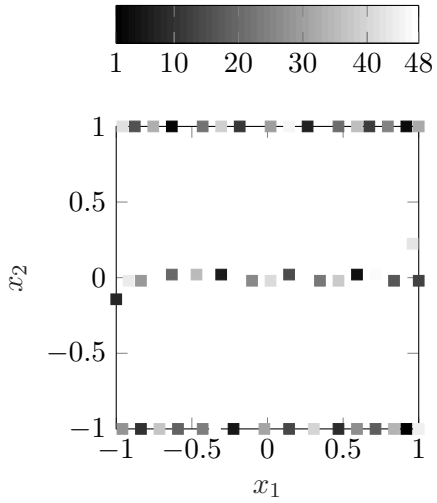
It is also noticed that QR with pivoting connects the greedy algorithm with DEIM (or QDEIM), which indicates that DEIM and QDEIM could also be used to select

representative parameter samples if either is applied to the output snapshot matrix. Fig. 5.6 shows that pivots of the QR decomposition on $\mathcal{Y}^\mathsf{T}$ gives similar points as those selected by the greedy algorithm. It then indicates that sample points selected by the greedy algorithm in a way are highly related to the interpolation points of QDEIM, if the same snapshot matrix is considered by both the greedy algorithm and QDEIM, which is, in our case, the output snapshot matrix $\mathcal{Y}$. By exploiting this interpretation, we propose to use DEIM or other variants of DEIM in order to adapt the training set during the greedy algorithm.

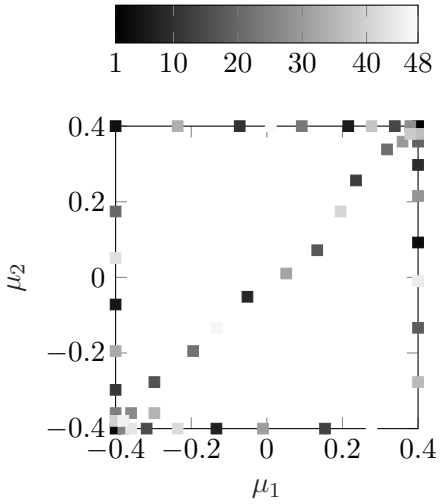
To further support and motivate our proposed scheme, in the next subsection, we show that DEIM also has the similar capability of identifying the most representative parameter samples for dynamics, as that exhibited by the greedy algorithm in the RBM.



(a) Singular value decay.



(b) Spatial DEIM interpolation points.



(c) Parameter DEIM interpolation points.

Figure 5.7.: Toy problem demonstrating anisotropic choice of interpolation points. The colourbars indicate the order of selection of the parameters. Points in the black end of the spectrum were selected earlier while those in the white regions of the spectrum were picked later during the course of the algorithm.

5.3.2.2. DEIM and Parametric Anisotropy

For a function of two variables $\mathbf{f}(\mathbf{x}, \boldsymbol{\mu}) : \mathbb{R}^N \times \mathbb{R}^{N_p} \to \mathbb{R}^N$, the DEIM algorithm first identifies a linear subspace $\mathbf{U}$ and a small subset of points in the $\mathbf{x}$ variable, based on the snapshot matrix $\mathbf{F}$ of $\mathbf{f}(\mathbf{x}, \boldsymbol{\mu})$. One can analogously consider the mapping $\mathbf{f}(\boldsymbol{\mu}, \mathbf{x}) : \mathbb{R}^{N_p} \times \mathbb{R}^N \to \mathbb{R}^N$ through a transpose of $\mathbf{F}$. However, now the DEIM algorithm identifies a small subset of points in the $\boldsymbol{\mu}$ variable. We illustrate this on a toy example from [1]. Consider the following nonlinear, two parameter function:

$$\mathbf{f}(\mathbf{x}_1, \mathbf{x}_2, \boldsymbol{\mu}) = \frac{1 + \frac{\pi^2}{4}(\mu_2 - \mu_1 - (\mu_1 + \mu_2)\mathbf{x}_2)^2 \sin^2(\frac{\pi}{2}(\mathbf{x}_1 + 1))}{1 + (\mu_1 + \mu_2)\cos(\frac{\pi}{2}(\mathbf{x}_1 + 1))}, \tag{5.4}$$

where $\mathbf{x} := [\mathbf{x}_1, \mathbf{x}_2] \in \mathbb{R}^{N \times 2}$ is the spatial variable obtained from the discretization of the two dimensional domain $\Omega := [-1\,, 1] \times [-1\,, 1]$ with 50 points in each spatial direction, resulting in $N = 2500$. The parameter $\boldsymbol{\mu} := (\mu_1, \mu_2) \in \mathbb{R}^2$ belongs to the domain $\mathcal{P} := [-0.4\,, 0.4] \times [-0.4\,, 0.4]$. We collect 1600 snapshots of the function into the snapshot matrix $\mathbf{F} \in \mathbb{R}^{2500 \times 1600}$, based on uniform, equally spaced samples of the parameter. In Fig. 5.7a, the singular values of the snapshot matrix $\mathbf{F}$ are plotted. The rapid decay clearly demonstrates the reducibility of this function. We apply a cut-off of $\epsilon_{\text{EI}} = 10^{-10}$ for Algorithm 2.6 applied to both $\mathbf{F}$ and $\mathbf{F}^\mathsf{T}$. The DEIM interpolation points in Fig. 5.7b are those corresponding to the indices in the set $\mathcal{I}$, obtained from applying Algorithm 2.6 to $\mathbf{F}$. The DEIM interpolation points in Fig. 5.7c are those $\boldsymbol{\mu}$ corresponding to the indices stored in $\mathcal{I}$, obtained from applying Algorithm 2.6 to $\mathbf{F}^\mathsf{T}$. The distribution of the points determined by DEIM for both the spatial and parameter variable have a characteristic structure. The number of interpolation points with SVD truncation tolerance $\epsilon_{\text{EI}} = 10^{-10}$ was 48, a mere 3% of the total points. The spatial interpolation points illustrate that while the variable x_1 is equally important over the entire range of $[-1\,, 1]$, the x_2 variable has almost all its variation concentrated at $x_2 \in \{-1, 0, 1\}$. For the parameter variable, the greedy algorithm picks most of the samples from the boundary of the domain and from the diagonal going from the lower left to the upper right. There is a dense concentration of points around the corners $(-0.4, -0.4)$ and $(0.4, 0.4)$. The choice of the greedy points is closely related to the structure of the function $\mathbf{f}$ being approximated. In most of the existing MOR literature, the DEIM algorithm has been used mainly as a tool to speed up evaluations of nonlinear or non-affine (parametric) functions in a ROM. However, through the toy example, we have demonstrated its capability to expose the nature of parametric dependence of a function. As seen in Fig. 5.7c, it is able to identify the regions in the parameter space where the function has large variations.

5.3.3. Subsampling the Training Set

Based on the observations in Sections 5.3.2.1 and 5.3.2.2, it is evident that a substantial computational effort can be saved at the offline stage of the RBM if we appropriately (optimally) sample the training set. The rationale for the proposed approach is the following: the standard greedy algorithm scans through the entire training set at each iteration and evaluates the error estimator at each parameter. This approach can incur significant computational cost for training sets with a large number of parameters. The

proposed algorithm aims at picking out a small subset of the training set containing the most informative parameters. As will be demonstrated numerically, the parameters match closely to those chosen by the standard greedy algorithm.

Based on our observation of Fig. 5.6 in Section 5.3.2.1 and Fig. 5.7 in Section 5.3.2.2, we propose to apply the pivoted QR decomposition and the DEIM algorithm (and its variants) to the snapshot matrix of the approximate output vector $\widetilde{\mathbf{y}}^k(\boldsymbol{\mu})$ (See Eq. (2.18)). More specifically, we consider the output snapshot matrix given by

$$\widetilde{\boldsymbol{\mathcal{Y}}} := \begin{bmatrix} [\widetilde{\mathbf{y}}^0(\boldsymbol{\mu}_1)]^\mathsf{T} & \cdots & [\widetilde{\mathbf{y}}^K(\boldsymbol{\mu}_1)]^\mathsf{T} \\ \vdots & \ddots & \vdots \\ [\widetilde{\mathbf{y}}^0(\boldsymbol{\mu}_{n_s})]^\mathsf{T} & \cdots & [\widetilde{\mathbf{y}}^K(\boldsymbol{\mu}_{n_s})]^\mathsf{T} \end{bmatrix} \in \mathbb{R}^{n_s \times N_O \cdot N_t} \tag{5.5}$$

with each row containing the snapshots of the approximated (vector-valued) output quantity at $K+1$ time instances corresponding to a given parameter sample.

Remark 5.1:
In case of steady systems with a single output we apply the proposed subsampling approach on the approximate state snapshots. For this, we define $\widetilde{\boldsymbol{\mathcal{Y}}} := \widetilde{\boldsymbol{\mathcal{X}}}^\mathsf{T}$, $\widetilde{\boldsymbol{\mathcal{X}}}$ being the snapshot matrix of the approximate state vector ($\widetilde{\mathbf{x}}(\boldsymbol{\mu}) = \mathbf{V}\widehat{\mathbf{x}}(\boldsymbol{\mu})$) such that

$$\widetilde{\boldsymbol{\mathcal{X}}} := [\widetilde{\mathbf{x}}(\boldsymbol{\mu}_1), \cdots, \widetilde{\mathbf{x}}(\boldsymbol{\mu}_{n_s})] \in \mathbb{R}^{N \times n_s},$$

with $\widetilde{\mathbf{x}}(\boldsymbol{\mu}_i)$ being the solution snapshot corresponding to the parameter $\boldsymbol{\mu}_i$, $i = 1, 2, \ldots, n_s$.$\Diamond$

Note that $\widetilde{\boldsymbol{\mathcal{Y}}}$ can be obtained from a coarse or low-fidelity ROM of the original system without doing FOM simulation at all the parameter samples. We propose two sampling strategies: (a) apply pivoted QR decomposition to $\widetilde{\boldsymbol{\mathcal{Y}}}^\mathsf{T}$, (b) apply DEIM or its variants to $\widetilde{\boldsymbol{\mathcal{Y}}}$, in order to identify the structure of the parametric dependence of the output variable. Once the distribution of the interpolation points is identified, we can then adapt the training set for subsequent iterations of the greedy algorithm. We now outline the proposed approach and discuss different computational strategies.

The proposed sampling procedure consists of two stages. The first stage is identical to the standard RBM procedure outlined in Algorithm 2.8. A finely sampled training set Ξ_f is used. We consider two different stopping criteria for the first stage — (a) the first stage runs until the maximum estimated error is below a coarse tolerance denoted by $\epsilon^c > \epsilon$, where ϵ is the desired error tolerance for the final ROM, or (b) at two successive iterations, the number of DEIM sampling points or QR pivots does not change. The value of ϵ^c is user-defined and is of order $O(1)$ in this work. Based on the two stopping criteria, two different schemes of training set subsampling are presented in Algorithms 5.2 and 5.3, respectively. For both algorithms, we do not reset the value of *iter* at the end of Stage 1, so the final value of *iter* upon convergence for Algorithms 5.2 and 5.3 is the total number of iterations required by either algorithms to converge to the desired tolerance. We employ the *a posteriori* error estimator in Eq. (4.30) in both stages of the proposed algorithms so that the parameter sample picked at each iteration is the one at which an estimated *output* error is the largest. Following the adaptive basis enrichment strategy from Section 4.3.6, the number of POD modes corresponding

to a selected parameter sample is adaptively decided: when the estimated error is large, a higher number of POD modes (r_{RB}) are added for the selected parameter; otherwise fewer POD modes are added. This reduces the chance of the same parameter sample being repeatedly chosen at subsequent iterations. This adaptive basis enrichment is implemented for both stages of our proposed method.

We now discuss several practical computational strategies in connection with Steps 28 and 29 in Algorithm 5.2 and Steps 26, 28 and 33 in Algorithm 5.3.

Remark 5.2:
The active subspaces method (ASM) is an approach for parameter space reduction that has been recently applied in the context of MOR [62, 179, 195] mainly for the case of scalar valued outputs. The ASM identifies a set of important directions in the parameter space onto which the parameter vectors are projected. This is done by means of Monte Carlo sampling of the gradients (with respect to the parameter) of the scalar-valued output quantity at selected parameter samples. The *active subspaces* are the eigenspaces of the (truncated) covariance matrix of the gradients. Compared to ASM, our approach differs in two significant ways. Firstly, the proposed subsampling strategy is applicable to vector-valued output quantities. Secondly, ASM requires calculation of the gradient of the output. Moreover, the user still has to define an additional training set over which the gradient samples are acquired. Our proposed approach does not require the calculation of any additional quantity. ◊

Remark 5.3:
Our proposed subsampling strategy occurring in Step 28 of Algorithm 5.2 and Step 26 of Algorithm 5.3 shares similarity with the column subset selection problem (CSSP) in the fields of numerical linear algebra and data mining [135]. For some general data matrix $\boldsymbol{\mathcal{D}} \in \mathbb{R}^{N\times M}$, the CSSP aims to identify $h < M$ independent columns of the matrix $\boldsymbol{\mathcal{D}}$ such that the residual $\|\boldsymbol{\mathcal{D}} - \mathscr{P}_h \boldsymbol{\mathcal{D}}\|$ is minimized. Here, $\mathscr{P}_h = \mathbf{S}\mathbf{S}^{\dagger}$ is a projection matrix and $\mathbf{S} \in R^{N\times h}$ consists of the h extracted columns from $\boldsymbol{\mathcal{D}}$. A number of algorithms, both deterministic and randomized, have been proposed to solve the CSSP [40, 42, 49]. One popular approach is to apply variants of the QR decomposition (column-pivoted, rank-revealing, hybrid, etc.) either to the data matrix $\boldsymbol{\mathcal{D}}$ or to the transpose of the (truncated) left (or right) singular matrix $\mathbf{L}$ (or $\mathbf{K}$) of $\boldsymbol{\mathcal{D}}$. If we consider $\boldsymbol{\mathcal{D}}$ as the approximate output snapshot matrix, then our proposed algorithm using pivoted QR (or QDEIM) can be seen as a special case of the CSSP. ◊

Algorithm 5.2: `ADAPTIVE_RBMEI_TS2_S1`
Computes a ROM using the Adaptive POD-Greedy algorithm in combination with hyperreduction and sparse subsampling of the training set.

Input: System matrices and nonlinear term: $\mathbf{E}, \mathbf{A}, \mathbf{B}, \mathbf{C}$ and $\mathbf{f}$, Training set Ξ_f, EI tolerance ϵ_{EI}, ROM tolerance ϵ, coarse tolerance ϵ^c, zoa ϵ^*, Deflation tolerance ϵ_{def}, Maximum iterations $iter\_max$.
Output: Reduced system matrices $\widehat{\mathbf{E}}, \widehat{\mathbf{A}}, \widehat{\mathbf{B}}, \widehat{\mathbf{C}}$ and EI quantities $\boldsymbol{\mathcal{U}}$, $\mathbb{J}$.

Initialization: $\mathbf{V} = [\,]$, $\mathbf{U} = [\,]$, $\mathbb{J} = \varnothing$, $err\_max = 1 + \epsilon$, $iter = 1$, $\boldsymbol{\mu}^*$ (chosen randomly from Ξ_f), $r_{\text{RB}} = r_{\text{EI}} = 1$.
For non-parametric t-dual system Eq. (4.2), obtain $\widetilde{\mathbf{x}}_{\text{du}}$ using a Krylov-space method.
Use RBF interpolation to approximate *inf-sup* constants as $\widetilde{\sigma}_{\min}(\boldsymbol{\mu})$ for all $\boldsymbol{\mu} \in \Xi_f$.

Stage 1
while $err\_max > \epsilon^c$ and $iter \leq iter\_max$ **do**
  if $r_{\text{RB}} > 0$ **then**
    Solve FOM Eq. (2.11) at $\boldsymbol{\mu}^*$; obtain snapshot matrix $\mathbf{X}(\boldsymbol{\mu}^*)$.
    if $iter == 1$ **then**
      $\mathbf{V} = \texttt{POD}(\mathbf{X}, r_{\text{RB}})$.
    else
      Compute $\overline{\mathbf{X}} := \mathbf{X} - \mathbf{V}\mathbf{V}^{\mathsf{T}}\mathbf{X}$.
      $\mathbf{V}_{\text{POD}} := \texttt{POD}(\overline{\mathbf{X}}, r_{\text{RB}})$.
      $\mathbf{V} = \texttt{orth\_def\_mat}(\mathbf{V}, \mathbf{V}_{\text{POD}}, \epsilon_{\text{def}})$. `/* see Step 15 of Algorithm 2.4 */`
    end
  else
    Remove the last r_{RB} columns from $\mathbf{V}$.
  end
  Form snapshot matrix of the nonlinear function: $\mathbf{F} := [\mathbf{F}\ \ \overline{\mathbf{f}}(\overline{\mathbf{X}})]$.
  Update EI basis and interpolation points $\mathbf{U}, \mathbb{J}$ using the subroutine `update_ei`. `/* see Line 26 of Algorithm 4.2 */`
  For a parametric t-dual system, update dual RB matrix $\mathbf{V}_{\text{du}}$ using the subroutine `update_dual`. `/* see Line 29 of Algorithm 4.2 */`
  $iter = iter + 1$.
  Assemble the ROM Eq. (2.36) through Galerkin projection and hyperreduction using $\mathbf{V}, \mathbf{U}$.
  Solve the ROM Eq. (2.36) and compute $\Delta_{\mathbf{y}}(\boldsymbol{\mu})$ for all $\boldsymbol{\mu} \in \Xi_f$.
  Find $\boldsymbol{\mu}^* = \arg\max_{\boldsymbol{\mu} \in \Xi_f} \Delta_{\mathbf{y}}(\boldsymbol{\mu})$; set $err\_max = \Delta(\boldsymbol{\mu}^*)$.
  Update $r_{\text{RB}}, r_{\text{EI}}$ using the subroutine `adapt_basis_update`. `/* see Line 35 of Algorithm 4.2 */`
end

Stage 2
Perform pivoted QR decomposition of $\widetilde{\mathcal{Y}}^{\mathsf{T}}$, or apply DEIM or a DEIM variant to $\widetilde{\mathcal{Y}}$ and identify the indices $\boldsymbol{\mathcal{I}}$ of the QR pivots or DEIM interpolation points.
Identify new training set Ξ using distribution of $\boldsymbol{\mathcal{I}}$.
while $err\_max > \epsilon$ and $iter \leq iter\_max$ **do**
  if $r_{\text{RB}} > 0$ **then**
    Solve FOM Eq. (2.11) at $\boldsymbol{\mu}^*$; obtain snapshot matrix $\mathbf{X}(\boldsymbol{\mu}^*)$.
    if $iter == 1$ **then**
      $\mathbf{V} = \texttt{POD}(\mathbf{X}, r_{\text{RB}})$
    else
      Compute $\overline{\mathbf{X}} := \mathbf{X} - \mathbf{V}\mathbf{V}^{\mathsf{T}}\mathbf{X}$.
      $\mathbf{V}_{\text{POD}} := \texttt{POD}(\overline{\mathbf{X}}, r_{\text{RB}})$.
      $\mathbf{V} = \texttt{orth\_def\_mat}(\mathbf{V}, \mathbf{V}_{\text{POD}}, \epsilon_{\text{def}})$. `/* see Step 15 of Algorithm 2.4 */`
    end
  else
    Remove the last r_{RB} columns from $\mathbf{V}$.
  end
  Form snapshot matrix of the nonlinear function: $\mathbf{F} := [\mathbf{F}\ \ \overline{\mathbf{f}}(\overline{\mathbf{X}})]$.
  Update EI basis and interpolation points $\mathbf{U}, \mathbb{J}$ using the subroutine `update_ei`.
  $iter = iter + 1$.
  Determine reduced matrices $\widehat{\mathbf{E}}, \widehat{\mathbf{A}}, \widehat{\mathbf{B}}, \widehat{\mathbf{C}}$ through Galerkin projection using $\mathbf{V}$ as in Eq. (2.14).
  Solve the ROM Eq. (2.36) and compute $\Delta_{\mathbf{y}}(\boldsymbol{\mu})$ for all $\boldsymbol{\mu} \in \Xi$.
  Find $\boldsymbol{\mu}^* = \arg\max_{\boldsymbol{\mu} \in \Xi} \Delta_{\mathbf{y}}(\boldsymbol{\mu})$; set $err\_max = \Delta(\boldsymbol{\mu}^*)$.
  Update $r_{\text{RB}}, r_{\text{EI}}$ using the subroutine `adapt_basis_update`.
end

5.3.3.1. Fully Adaptive RBMEI algorithm with Training Set Subsampling – Scheme 1

We describe the proposed approach for the first scheme detailed in Algorithm 5.2. In the first stage, a low-fidelity ROM is built with a coarse tolerance ϵ^c, over a finely sampled training set Ξ_f. The intuition is that a low-fidelity approximation is sufficient to discover the parametric dependence of the output variable. At the end of the first stage, DEIM (or one of its variants) is applied to $\widetilde{\boldsymbol{\mathcal{Y}}}$ to identify the interpolation points or a pivoted QR decomposition of $\widetilde{\boldsymbol{\mathcal{Y}}}^{\mathsf{T}}$ is used to identify the pivots. Once the set of interpolation points (or pivots) $\mathcal{I}$ is identified, we proceed to suitably subsample the finely sampled training set based on the distribution of the identified interpolation points or pivots. Different possibilities exist to achieve this. A simple approach is to consider the training set for the second stage Ξ populated only by the identified interpolation points or pivots. Consider the fine training set $\Xi_f := \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_{n_f}\}$ with the subscript denoting the index corresponding to the position of a parameter in the set. Let $\boldsymbol{\mathcal{I}}$ be the vector whose elements are the indices of the chosen interpolation points (in $\mathcal{I}$) or pivots. We define the subsampled training set Ξ as the one consisting of all those parameters $\boldsymbol{\mu}_z$ from Ξ_f such that their indices are present in $\boldsymbol{\mathcal{I}}$, i.e., $\Xi := \{\boldsymbol{\mu}_{z;z\in\boldsymbol{\mathcal{I}}}\}$. If there are only a few interpolation points, this approach would lead to a rapid second stage of the algorithm. However, there may exist the pitfall that it may result in an *overfit* by which we mean that the resulting ROM after Stage 2 satisfies the desired tolerance ϵ only over the subsampled training set and does not generalize to other parameters in the parameter domain. We illustrate this phenomenon in the numerical tests. Another possible approach is to define a training budget m for the second stage and use an oversampling strategy like the Gappy-POD to ensure that the training set for the second stage consists of a total of m parameter samples.

5.3.3.2. Fully Adaptive RBMEI algorithm with Training Set Subsampling – Scheme 2

The first scheme of our proposed training set adaptivity method requires a user-defined coarse tolerance ϵ^c. Choosing such a tolerance is rather heuristic. For some problems, a rough approximation may be enough to suitably capture all the parametric dependences, whereas a finer approximation may be needed for others. Therefore, for the second scheme we define a heuristic criterion that leads automatically to the second stage. To achieve this, we apply DEIM approximation (or a variant of DEIM) to the matrix $\widetilde{\boldsymbol{\mathcal{Y}}}$ or the pivoted QR factorization to $\widetilde{\boldsymbol{\mathcal{Y}}}^{\mathsf{T}}$, at each iteration of the greedy algorithm. Whenever the DEIM interpolations or pivoted QR decompositions at two successive iterations turn out to be equal, we terminate the first stage. This can be easily calculated by comparing the number of interpolation indices or the pivot indices at two successive iterations. Then, the subsampling of the training set for the second stage is carried out with similar approaches as discussed above. Following this, the second stage is run, until the required tolerance ϵ is met. In Algorithm 5.3, $\boldsymbol{\mathcal{I}}_{iter}$ and $\boldsymbol{\mathcal{I}}_{iter-1}$ in Stage 1 refer to the vector containing the interpolation indices identified by DEIM or its variants, or the indices of the QR pivots at the current iteration and the previous iteration, respectively.

Each of the two schemes has its own benefits or shortcomings. For Scheme 1, the burden of choosing an appropriate coarse tolerance lies with the user. This is highly problem-dependent. In the limit that $\epsilon^c = \epsilon$, Scheme 1 is just the POD-Greedy algorithm (Algorithm 4.2) with a fixed training set. If $\epsilon^c \gg \epsilon$, a very fast second stage can be ensured, leading to considerable speedup of the offline stage of the RBM. However, it is not known *a priori*, if the chosen coarse tolerance results in a sufficiently accurate ROM. Scheme 2, on the other hand, automatizes the switching from Stage 1 to Stage 2 of the subsampling strategy by considering a heuristic criterion. But, this comes with the additional cost of performing the DEIM algorithm or the pivoted QR decomposition, repeatedly. The success of both schemes is also highly dependent on the strategy adopted to construct the subsampled training set. In the numerical tests, we shall consider two approaches. In the first approach, we consider as many parameters in the subsampled training set as the number of DEIM interpolation points or QR pivots. For the second approach, we fix the cardinality of the subsampled training set to be m and then use oversampling strategies based on the Gappy-POD method to choose those m parameters in a principled way.

Remark 5.4 (Computational complexity):
The fine training set is used in Stage 1 of both Algorithms 5.2 and 5.3. However, the computational complexity is not high for Algorithm 5.2, since we use a coarse tolerance ϵ^c in Stage 1, so that the greedy algorithm converges within much fewer steps than when using the user desired tolerance in Stage 2. The computational complexity will grow with the decrease of the coarse tolerance used in Stage 1. However, as we have observed, a moderate value of ϵ^c is enough to figure out the parameter dependency of the output. The number of FOM simulations is indeed independent of the size of the fine training set, since the FOM simulation is implemented only at those "selected" parameter samples. The situation for Algorithm 5.3 is different since it involves the DEIM or QR algorithms at each iteration (see Step 26) to compute the interpolation points. A fine training set will indeed increase the cost of this step. Nevertheless, one can readily use recent techniques based on randomized linear algebra (such as randomized SVD, randomized QR, etc.) to keep the costs under control, see [73, 182].◊

Remark 5.5 (High-dimensional parameter spaces):
Both Algorithms 5.2 and 5.3, can be used when the parameter space is high-dimensional. The cost in Stage 2 will not be affected much, since a small training set identified from Stage 1 is used. The main increase in cost is due to the need to solve additional ROMs and estimate the error in Stage 1 of the proposed algorithms (Step 23 in Algorithms 5.2 and 5.3). If sparse grid sampling and cheap error estimator is used, the costs will not increase fast. Actually, we can go a step further and make use of cheaply computable surrogate models of the error estimator as done in Section 5.2.1. Such an approach involves a RBF surrogate for the error estimator that is adaptively updated during the greedy algorithm. We only evaluate the actual error estimator over a few parameter samples. We then use this data to form a surrogate model, which can be used to evaluate the error for different parameter samples in the fine training set in Stage 1. ◊

Algorithm 5.3: `ADAPTIVE_RBMEI_TS2_S2`

Computes a ROM using the Adaptive POD-Greedy algorithm in combination with hyperreduction and sparse subsampling of the training set.

Input: System matrices and nonlinear term: $\mathbf{E}, \mathbf{A}, \mathbf{B}, \mathbf{C}$ and $\mathbf{f}$, Training set Ξ_f, EI tolerance ϵ_{EI}, ROM tolerance ϵ, zoa ϵ^*, Deflation tolerance ϵ_{def}, Maximum iterations $iter\_max$.

Output: Reduced system matrices $\widehat{\mathbf{E}}, \widehat{\mathbf{A}}, \widehat{\mathbf{B}}, \widehat{\mathbf{C}}$ and EI quantities $\boldsymbol{\mathcal{U}}$, $\mathcal{I}$.

Initialization: $\mathbf{V} = [\,]$, $\mathbf{U} = [\,]$, $\mathcal{I} = \varnothing$, $err\_max = 1 + \epsilon$, $iter = 1$, $\boldsymbol{\mu}^*$ (chosen randomly from Ξ_f), $r_{\text{RB}} = r_{\text{EI}} = 1$.

For non-parametric t-dual system Eq. (4.2), obtain $\widetilde{\mathbf{x}}_{\text{du}}$ using a Krylov-space method.

Use RBF interpolation to approximate *inf-sup* constants as $\widetilde{\sigma}_{\min}(\boldsymbol{\mu})$ for all $\boldsymbol{\mu} \in \Xi_f$.

Stage 1

while not terminated **do**

  if $r_{\text{RB}} > 0$ **then**

    Solve FOM Eq. (2.11) at $\boldsymbol{\mu}^*$; obtain snapshot matrix $\mathbf{X}(\boldsymbol{\mu}^*)$.

    if $iter == 1$ **then**

      $\mathbf{V} = \texttt{POD}(\mathbf{X}, r_{\text{RB}})$

    else

      Compute $\overline{\mathbf{X}} := \mathbf{X} - \mathbf{V}\mathbf{V}^{\mathsf{T}}\mathbf{X}$.

      $\mathbf{V}_{\text{POD}} := \texttt{POD}(\overline{\mathbf{X}}, r_{\text{RB}})$.

      $\mathbf{V} = \texttt{orth\_def\_mat}(\mathbf{V}, \mathbf{V}_{\text{POD}}, \epsilon_{\text{def}})$. `/* see Step 15 of Algorithm 2.4 */`

    end

  else

    Remove the last r_{RB} columns from $\mathbf{V}$.

  end

  Form snapshot matrix of the nonlinear function: $\mathbf{F} := [\mathbf{F} \;\; \overline{\mathbf{f}}(\overline{\mathbf{X}})]$.

  Update EI basis and interpolation points $\mathbf{U}, \mathcal{I}$ using the subroutine `update_ei`. `/* see Line 26 of Algorithm 4.2 */`

  For a parametric t-dual system, update dual RB matrix $\mathbf{V}_{\text{du}}$ using the subroutine `update_dual`. `/* see Line 29 of Algorithm 4.2 */`

  $iter = iter + 1$.

  Assemble the ROM Eq. (2.36) through Galerkin projection and hyperreduction using $\mathbf{V}, \mathbf{U}$.

  Solve the ROM Eq. (2.36) and compute $\Delta_{\mathbf{y}}(\boldsymbol{\mu})$ for all $\boldsymbol{\mu} \in \Xi_f$.

  Find $\boldsymbol{\mu}^* = \arg\max_{\boldsymbol{\mu} \in \Xi_f} \Delta_{\mathbf{y}}(\boldsymbol{\mu})$; set $err\_max = \Delta(\boldsymbol{\mu}^*)$.

  Update $r_{\text{RB}}, r_{\text{EI}}$ using subroutine `adapt_basis_update`. `/* see Line 35 of Algorithm 4.2 */`

  Perform pivoted QR decomposition of $\widetilde{\boldsymbol{\mathcal{Y}}}^{\mathsf{T}}$, or apply DEIM or a DEIM variant to $\widetilde{\boldsymbol{\mathcal{Y}}}$ and identify the indices $\boldsymbol{\mathcal{I}}$ of the QR pivots or DEIM interpolation points.

  if $iter \geq 2$ **then**

    Check if length($\boldsymbol{\mathcal{I}}_{iter-1}$) == length($\boldsymbol{\mathcal{I}}_{iter}$).

    If true **break** and proceed to **Stage 2**.

  end

end

Stage 2

Identify new training set Ξ using distribution of $\boldsymbol{\mathcal{I}}$.

while $err\_max > \epsilon$ and $iter \leq iter\_max$ **do**

  if $r_{\text{RB}} > 0$ **then**

    Solve FOM Eq. (2.11) at $\boldsymbol{\mu}^*$; obtain snapshot matrix $\mathbf{X}(\boldsymbol{\mu}^*)$.

    if $iter == 1$ **then**

      $\mathbf{V} = \texttt{POD}(\mathbf{X}, r_{\text{RB}})$

    else

      Compute $\overline{\mathbf{X}} := \mathbf{X} - \mathbf{V}\mathbf{V}^{\mathsf{T}}\mathbf{X}$.

      $\mathbf{V}_{\text{POD}} := \texttt{POD}(\overline{\mathbf{X}}, r_{\text{RB}})$.

      $\mathbf{V} = \texttt{orth\_def\_mat}(\mathbf{V}, \mathbf{V}_{\text{POD}}, \epsilon_{\text{def}})$. `/* see Step 15 of Algorithm 2.4 */`

    end

  else

    Remove the last r_{RB} columns from $\mathbf{V}$.

  end

  Form snapshot matrix of the nonlinear function: $\mathbf{F} := [\mathbf{F} \;\; \overline{\mathbf{f}}(\overline{\mathbf{X}})]$.

  Update EI basis and interpolation points $\mathbf{U}, \mathcal{I}$ using the subroutine `update_ei`.

  $iter = iter + 1$.

  Assemble the ROM Eq. (2.36) through Galerkin projection and hyperreduction using $\mathbf{V}, \mathbf{U}$.

  Solve the ROM Eq. (2.36) and compute $\Delta_{\mathbf{y}}(\boldsymbol{\mu})$ for all $\boldsymbol{\mu} \in \Xi$.

  Find $\boldsymbol{\mu}^* = \arg\max_{\boldsymbol{\mu} \in \Xi} \Delta_{\mathbf{y}}(\boldsymbol{\mu})$; set $err\_max = \Delta(\boldsymbol{\mu}^*)$.

  Update $r_{\text{RB}}, r_{\text{EI}}$ using the subroutine `adapt_basis_update`.

end

Remark 5.6 (Role of output quantity of interest):
In many cases, there is often the requirement, based on the application, to have a good approximation for the entire state vector. However, in this work we have specifically focussed on a *goal-oriented* approach. For several applications, like in control systems or fluid dynamics, only a small number of state variables may be of interest. By focussing on those states alone, the resulting ROM dimension can be considerably lowered when compared to the case where the entire state needs to be well-approximated. Also, it is indeed true that different output QoIs may have different influences on the parameter samples chosen. However, if some QoIs give rise to quantities with similar emphases, then they may result in similar parameter samples being chosen. For example, a QoI defined by the mean of the solution over the spatial domain and a QoI defined by the sum of the solution over spatial domain should result in similar samples. But a QoI defined by the mean of the solution over space probably gives different samples from the QoI defined by the maximum of the state over the spatial domain. ◊

5.3.4. Numerical Examples

We test the proposed adaptive training set subsampling algorithm on two examples. They are:

1. viscous Burgers' equation with one parameter,
2. thermal block with four parameters.

The first example is a nonlinear system and the details were introduced in Section 4.3.7.2. The second example is a linear model of the heat equation. Next, we describe the metrics used in all the numerical tests:

- The results of the proposed algorithms (Algorithms 5.2 and 5.3) are compared against the Adaptive RBMEI algorithm (Algorithm 4.2) with a fixed training set. The implementation adopts Galerkin projection. The number of POD modes r_{RB}, r_{EI} to enrich the RB and EI bases is determined at each iteration based on the subroutine `adapt_basis_update` in Algorithm 4.2. The dual RB basis $\mathbf{V}_{\text{du}}$ required for the error estimator is generated separately.
- The fixed training set used for Algorithm 4.2 and the initial fine training set used for Algorithms 5.2 and 5.3 are the same.
- For Algorithms 5.2 and 5.3, we apply (i) the pivoted QR decomposition to the transposed approximate output snapshot matrix $\widetilde{\boldsymbol{\mathcal{Y}}}^{\mathsf{T}}$ and, (ii) the DEIM variants on the approximate output snapshot matrix $\widetilde{\boldsymbol{\mathcal{Y}}}$ as two approaches to subsample the fine training set.
- The cut-off criterion to determine the number of pivots (h) for the pivoted QR decomposition in Eq. (5.3) of the approximate output snapshot matrix in Eq. (5.5) is based on the magnitude of the diagonal elements in the upper triangular matrix $\mathbf{R}$, i.e., we set $h = q$ based on the smallest q such that $|\mathbf{R}(q{+}1, q{+}1)|/|\mathbf{R}(1,1)| < \epsilon_{\text{QR}}$, with $q \in \{1, 2, \ldots, \min(n_s, N_t)\}$. The pivoted QR decomposition can effectively

identify the rank of a matrix with a small diagonal $\mathbf{R}(q+1, q+1)$. Although there are cases when the column pivoted QR decomposition fails, they are rare in practice [51]. A more robust rank-revealing QR factorization [51] can also be straightforwardly applied to our proposed subsampling algorithm. However, in this work, we simply use column pivoted QR. The intrinsic MATLAB®command `qr` is used with the options `vector` enabled, i.e., we call $[\mathbf{Q}, \mathbf{R}, p_{\text{qr}}] = \texttt{qr}(\widetilde{\boldsymbol{\mathcal{Y}}}^{\mathsf{T}}, \texttt{'vector'})$. It returns the pivot indices p_{qr} as a vector, from which we select the first h as our subsampling indices, i.e., $\boldsymbol{\mathcal{I}} = p_{\text{qr}}(1:h)$. Here, $\boldsymbol{\mathcal{I}}$ is the vector whose elements are the indices of the QR pivots and it lets us choose the subsampled training set Ξ for Stage 2 of the proposed method, based on the fine training set Ξ_f from Stage 1.

- Our implementation of the k-means algorithm for KDEIM is based on the intrinsic MATLAB®function `kmeans`. We use five different initializations and pick the best configuration among the five.

- Reported runtimes for all the algorithms are obtained by considering the median value of five independent runs.

- The quantity Iterations reported in Tables 5.5 to 5.9 refers to the total number of iterations of the corresponding greedy algorithm (Algorithms 4.2, 5.2 and 5.3) to converge to the desired tolerance.

The reported results below were obtained using MATLAB®2015a, on a laptop with INTEL®CORE™i5-7200U @ 2.5 GHZ, with 8 GB of RAM.

5.3.4.1. Burgers' Equation

The model equations and discretization schemes are the same as those used to discretize Equation (4.44) in Section 4.3.7.2. However, for this example, we consider a finer spatial discretization resulting in a FOM with dimension $N = 1000$. The parameter domain of the viscosity μ is $\mathcal{P} := [0.005, 1]$. The training sets $\Xi_f = \Xi$ consists of 100 equally spaced samples in $\mathcal{P}$. For the RBM, we fix the tolerance to be $\epsilon = 1 \cdot 10^{-6}$. To validate the ROM, we use a test set Ξ_{test} containing 300 randomly sampled parameters, different from those in the training set.

Greedy Algorithm with Fixed Training Set We begin by applying the Adaptive RBMEI algorithm (Algorithm 4.2) to the discretized model of the Burgers' equation. The algorithm requires 505.47 seconds and 19 iterations to converge to the defined tolerance of $1 \cdot 10^{-6}$. The resulting ROM has RB dimension $r_{\text{RB}} = 32$ along with $r_{\text{EI}} = 33$ basis vectors for the DEIM projection matrix. The maximum true error over the test set is $\epsilon_{\max} = 2.07 \cdot 10^{-8}$, where $\epsilon_{\max}$ is as defined in Eq. (4.42). Although the POD-Greedy algorithm with a fixed training set results in a ROM that meets the specified tolerance, its offline time is high and there is scope for improvement by considering a subsampled training set.

Greedy Algorithm Schemes 1 and 2 We apply Algorithms 5.2 and 5.3 to the Burgers' equation, making use of both pivoted QR and two DEIM variants (QDEIM and KDEIM) to identify the interpolation points $\boldsymbol{\mathcal{I}}$. Further, we consider three different SVD and QR cut-off tolerances (ϵ_{EI}, ϵ_{QR}) for the pivoted QR and DEIM variants $\{1 \cdot 10^{-4}, 1 \cdot 10^{-6}, 1 \cdot 10^{-8}\}$ to highlight the progressive refinement of the adapted training set in 'difficult regions' of the parameter space. The results are summarized in Tables 5.5 and 5.6 for Algorithms 5.2 and 5.3, respectively. The matrix $\widetilde{\boldsymbol{\mathcal{Y}}} \in \mathbb{R}^{100\times 81}$, for either algorithms, was assembled by collecting the snapshots of the output vector at every 25^{th} time step. The fine training set Ξ_f for either algorithm is the same as Ξ, used for the fixed training set case above. The training set in Stage 2 of both algorithms consists of interpolation points identified by QR, QDEIM or KDEIM. As revealed in the results, this choice is sufficient to produce ROMs that meet the required tolerance over the test set. For this example, there is not a big difference between the results of the two algorithms. Both schemes produce ROMs of almost identical RB, DEIM basis sizes (r_{RB}, r_{EI}) and result in nearly the same maximum error over the test set.

We show the subsampled training sets resulting from Algorithm 5.3 using the pivoted QR, QDEIM and KDEIM variants, with different SVD, QR tolerances in Figs. 5.8 to 5.10. The black crosses denote those samples from the fine training set which were retained in Stage 2 of the algorithm. For the QDEIM variant, it is clear that the subsampled parameters are concentrated more around the lower viscosity regions of the parameter space. Thus, the method is able to successfully identify the *physically more relevant* points. Moreover, the parameter samples identified by QDEIM are very close to the ones identified by the method using a pivoted QR decomposition. This is not surprising since the former determines the interpolation points through a pivoted QR decomposition of $\mathbf{U}^{\mathsf{T}}$ ($\mathbf{U}$ is the left singular matrix of $\widetilde{\boldsymbol{\mathcal{Y}}}$) whereas the latter applies the pivoted QR decomposition directly to $\widetilde{\boldsymbol{\mathcal{Y}}}^{\mathsf{T}}$. We also show the results of KDEIM, where the subsampled (selected) parameter samples and their corresponding clusters are presented. The subsampled points in this case are the centroids of the clusters. The clusters are smaller in size for the low viscosity regions, while they are comparatively larger in the high viscosity zone. The resulting subsampled training sets from Algorithm 5.2 display a similar trend.

For a given ϵ_{EI} or ϵ_{QR}, the runtimes for the QDEIM and KDEIM based training set adaptation are very close. Using the pivoted QR leads to a subsampled training set with one sample more than that generated by using QDEIM or KDEIM. This results in a marginally higher offline time for this method. One observation worth remarking is that, for some instances, using the KDEIM approach to identify the adapted training set leads to the greedy algorithm converging in fewer iterations. This is most likely due to the fact that the identified parameters in this case represent cluster centroids and are more representative of the average behaviour. This yields a more uniform approximation throughout the parameter domain, with each cluster average being well-represented. The QDEIM version, on the other hand, tends to identify points based on the SVD of the output snapshot matrix and tends to favour points away from the mean behaviour. We illustrate this in Fig. 5.11 for the case of $\epsilon_{\text{EI}} = 10^{-8}$ for Algorithm 5.2. It is evident that while the subsampling strategy using QDEIM results in a smaller magnitude of the maximum error over the test set ($4.55 \cdot 10^{-8}$ vs. $6.88 \cdot 10^{-8}$), the

approach using the KDEIM-based sampling leads to a more uniform distribution of the error over the test set.

On average, for a given SVD tolerance, Algorithm 5.2 is faster than Algorithm 5.3. This can be attributed to the fact that for the latter, the DEIM variant or the pivoted QR decomposition needs to be performed repeatedly to check the criterion in Step 26 of Algorithm 5.3. Since this involves performing an SVD, the associated costs are higher. The proposed subsampling algorithms result in a noticeable speedup of the POD-Greedy algorithm. For Algorithm 5.2, the maximum achieved speedup was 5.6 for $\epsilon_{\text{EI}} = 1 \cdot 10^{-4}$ using QDEIM. However, the least speedup noticed was 4.0 for $\epsilon_{\text{EI}} = 1 \cdot 10^{-8}$ using QDEIM or KDEIM. Also, for Algorithm 5.3 the maximum achieved speedup was 5.7 for $\epsilon_{\text{EI}} = 1 \cdot 10^{-4}$ using KDEIM while the minimum speedup was 3.5 for $\epsilon_{\text{EI}} = 1 \cdot 10^{-8}$ using QDEIM.

5.3.4.2. Thermal Block

The second example is a benchmark model of the time-dependent heat transfer in a thermal block. The domain $\Omega := (0,1) \times (0,1) \in \mathbb{R}^2$ is partitioned into five regions — $\Omega = \Omega_0 \cup \Omega_1 \cup \Omega_2 \cup \Omega_3 \cup \Omega_4$ as shown in Fig. 5.12. The governing equation is given by:

$$\frac{\partial g(\mathbf{z}, t; \boldsymbol{\mu})}{\partial t} + \nabla.(-\kappa(\mathbf{z}; \boldsymbol{\mu}) \nabla g(\mathbf{z}, t; \boldsymbol{\mu})) = 0, \qquad t \in (0, T), \qquad \mathbf{z} \in \Omega. \tag{5.6}$$

The left boundary of the domain (Γ_{in}) is associated with an input heat flux, the top and bottom boundaries (Γ_{N}) are associated with a Neumann boundary condition with zero flux and finally the right boundary (Γ_{D}) is fixed at zero. The state variable g is the temperature at a given spatial location $\mathbf{z} \in \Omega$, for a given time t. The output is the average temperature measured at Ω_2. The problem is parametrized by the heat conductivity κ in the four subdomains ($\Omega_1, \Omega_2, \Omega_3$ and Ω_4). We have the parameter vector $\boldsymbol{\mu} = [\kappa_1, \kappa_2, \kappa_3, \kappa_4]$. The governing PDE is discretized in space using linear finite elements with respect to a simplicial triangulation of the domain Ω obtained via the software `gmsh` [92]. It is further discretized in time using the implicit Euler scheme for a time ranging from $t \in [0\,, 1]$, with step size $\Delta t = 0.01$. The spatially discretized system has dimension $N = 7,488$. For more details on the model and the spatial discretization, the reader is referred to [172]. The discretized heat equation can be written in the form of Eq. (2.11). Since the problem is linear, we have $\overline{\mathbf{f}} \equiv 0$. For the numerical results, the parameter $\boldsymbol{\mu}$ is sampled from the domain $\mathcal{P} := [1 \cdot 10^{-5}\,, 1 \cdot 10^{-2}] \times [1 \cdot 10^{-5}\,, 1 \cdot 10^{-2}] \times [1 \cdot 10^{-4}\,, 1] \times [1 \cdot 10^{-1}\,, 1]$. For purposes of illustration, we consider the three parameter version of the thermal block problem by fixing κ_4 to its mean value, i.e., $\kappa_4 = 0.5$. The training set Ξ consists of a tensor grid of $n_f = 6^3 = 216$ parameters, with 6 parameters sampled for each κ_i, $i = 1, 2, 3$. The test parameter set consists of 100 parameters, randomly sampled from $\mathcal{P}$. The tolerance for the greedy algorithm is set to be $\epsilon = 1 \cdot 10^{-3}$.

Greedy Algorithm with Fixed Training Set Applying Algorithm 4.2 with a fixed training set to the thermal block example results in a ROM of dimension $r_{\text{RB}} = 74$, taking 53 iterations to converge in 694.73 seconds. The maximum error over the test set is $\epsilon_{\text{max}} = 9.78 \cdot 10^{-4}$. In Fig. 5.13, the training set Ξ and the greedy parameters

identified by Algorithm 4.2 are shown. Of the 216 parameters in the training set, only 44 are chosen. The greedy parameters have a larger concentration at and around $(0.01, 0.01, 0.0001)$, the upper right corner of the figure. In fact, the regions around the vicinity of the upper and right wall of the grid posses many greedy samples near them.

Greedy Algorithm Schemes 1 and 2 Similar to the Burgers' equation, we now apply the proposed training set subsampling schemes to the thermal block example. For this model, we shall also illustrate the advantages of using oversampling. For both Algorithms 5.2 and 5.3, we consider $\epsilon_{\text{QR}}, \epsilon_{\text{EI}} = 1 \cdot 10^{-10}$ and a coarse tolerance $\epsilon^c = 1$. The approximation to $\boldsymbol{\mathcal{Y}}$ is obtained by taking snapshots at every time step of the implicit Euler scheme. The results are summarized in Table 5.7 and Table 5.8 for Algorithms 5.2 and 5.3, respectively. For both algorithms, the fine training set Ξ_f is the same as Ξ, the one used for the fixed training set. The first scheme does not lead to a ROM with sufficient accuracy for both QDEIM and KDEIM whereas using the pivoted QR decomposition on $\widetilde{\boldsymbol{\mathcal{Y}}}^{\mathsf{T}}$ to identify the subsampled training set produces a successful ROM. For the second scheme of the proposed algorithm, both QDEIM and KDEIM result in a subsampled training set of cardinality $n_s = 19$ while the pivoted QR approach gives $n_s = 20$. However, both QR and QDEIM are unsuccessful in meeting the required ROM tolerance for the test set. On the other hand, KDEIM results in a ROM satisfying the tolerance, taking a significantly smaller number of iterations (40) to converge. The results seem to indicate that the subsampling approach is not entirely able to capture the full range of features over the training set. This is mainly due to the smaller number of parameters $n_s = 19$, that the algorithm results in. Recall that for the standard greedy approach, 44 unique greedy parameters were determined. However, it is also to be noted that the performance of the ROMs resulting from either scheme on the test set is not bad. The maximum error is only slightly higher than the desired tolerance of $\epsilon = 1 \cdot 10^{-3}$.

Table 5.5.: Results of Algorithm 5.2 with varying ϵ_{EI}, ϵ_{QR} for Burgers' equation.

Method	Fixed	Adapted								
		$\epsilon_{\text{EI}}, \epsilon_{\text{QR}} = 1 \cdot 10^{-4}$			$\epsilon_{\text{EI}}, \epsilon_{\text{QR}} = 1 \cdot 10^{-6}$			$\epsilon_{\text{EI}}, \epsilon_{\text{QR}} = 1 \cdot 10^{-8}$		
		QDEIM	KDEIM	QR	QDEIM	KDEIM	QR	QDEIM	KDEIM	QR
n_s	100	7	7	8	11	11	12	15	15	15
ϵ_{max}	$2.07 \cdot 10^{-8}$	$4.22 \cdot 10^{-7}$	$4.15 \cdot 10^{-7}$	$3.43 \cdot 10^{-8}$	$1.87 \cdot 10^{-8}$	$1.87 \cdot 10^{-8}$	$3.43 \cdot 10^{-8}$	$1.87 \cdot 10^{-8}$	$1.84 \cdot 10^{-8}$	$3.28 \cdot 10^{-8}$
$(r_{\text{RB}}, r_{\text{EI}})$	(32,33)	(31,32)	(30,31)	(31,32)	(31,32)	(31,32)	(31,32)	(31,32)	(31,32)	(31,32)
Iterations	19	17	17	17	18	18	17	18	18	17
Offline time (s)	505.47	91.00	91.16	94.90	110.10	110.35	109.54	125.07	125.59	120.53
Speedup	-	5.6	5.5	5.3	4.6	4.6	4.6	4.0	4.0	4.2

Table 5.6.: Results of Algorithm 5.3 with varying ϵ_{EI}, ϵ_{QR} for Burgers' equation.

Method	Fixed	Adapted								
		$\epsilon_{\text{EI}}, \epsilon_{\text{QR}} = 1 \cdot 10^{-4}$			$\epsilon_{\text{EI}}, \epsilon_{\text{QR}} = 1 \cdot 10^{-6}$			$\epsilon_{\text{EI}}, \epsilon_{\text{QR}} = 1 \cdot 10^{-8}$		
		QDEIM	KDEIM	QR	QDEIM	KDEIM	QR	QDEIM	KDEIM	QR
n_s	100	7	7	8	11	11	12	15	15	15
ϵ_{max}	$2.07 \cdot 10^{-8}$	$4.22 \cdot 10^{-7}$	$3.43 \cdot 10^{-7}$	$3.43 \cdot 10^{-8}$	$1.87 \cdot 10^{-8}$	$5.96 \cdot 10^{-8}$	$3.43 \cdot 10^{-8}$	$1.87 \cdot 10^{-8}$	$2.59 \cdot 10^{-8}$	$3.28 \cdot 10^{-8}$
$(r_{\text{RB}}, r_{\text{EI}})$	(32,33)	(31,32)	(30,31)	(31,32)	(31,32)	(31,32)	(31,32)	(31,32)	(30,31)	(31,32)
Iterations	19	17	16	17	18	18	17	18	16	17
Offline time (s)	505.47	90.78	88.52	94.58	110.12	110.10	109.55	145.22	135.82	120.27
Speedup	-	5.6	5.7	5.3	4.6	4.6	4.6	3.5	3.7	4.2

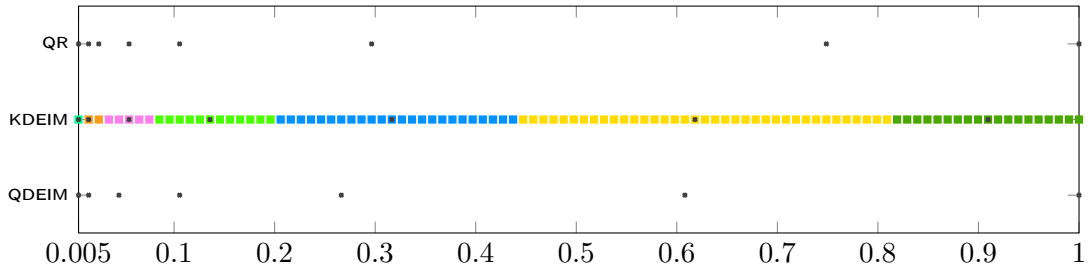


Figure 5.8.: Algorithm 5.3 for the Burgers' equation with SVD, QR tolerance $\epsilon_{\text{EI}}, \epsilon_{\text{QR}} = 1 \cdot 10^{-4}$. The crossmarks denote the parameters in the subsampled training set. For KDEIM each colour represents one cluster; the centroids of each of the clusters make up the subsampled training set.

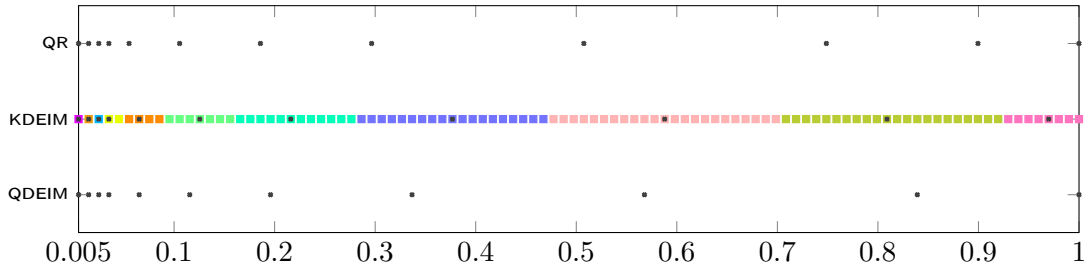


Figure 5.9.: Algorithm 5.3 for the Burgers' equation with SVD, QR tolerance $\epsilon_{\text{EI}}, \epsilon_{\text{QR}} = 1 \cdot 10^{-6}$. The crossmarks denote the parameters in the subsampled training set. For KDEIM each colour represents one cluster; the centroids of each of the clusters make up the subsampled training set.

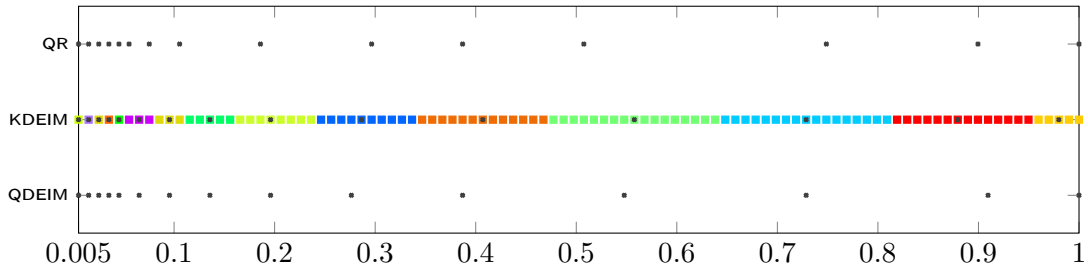
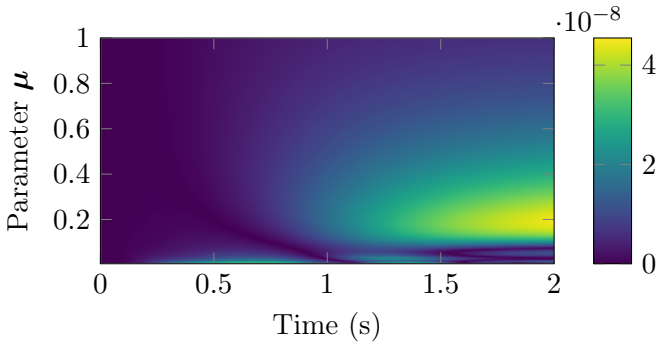
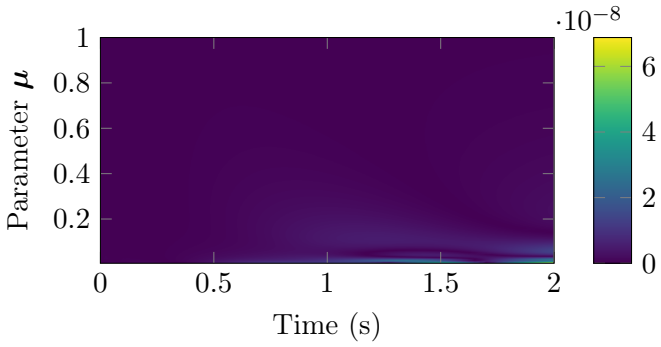


Figure 5.10.: Algorithm 5.3 for the Burgers' equation with SVD, QR tolerance $\epsilon_{\text{EI}}, \epsilon_{\text{QR}} = 1 \cdot 10^{-8}$. The crossmarks denote the parameters in the subsampled training set. For KDEIM each colour represents one cluster; the centroids of each of the clusters make up the subsampled training set.



(a) QDEIM.



(b) KDEIM.

Figure 5.11.: Error plot for Algorithm 5.2 with tolerance $\epsilon_{\text{EI}} = 10^{-8}$ applied to the Burgers' equation. The error between the true and reduced outputs $\|\mathbf{y}^k(\boldsymbol{\mu}) - \widetilde{\mathbf{y}}^k(\boldsymbol{\mu})\|$ is plotted over the duration of the simulation for all parameters in the test set.

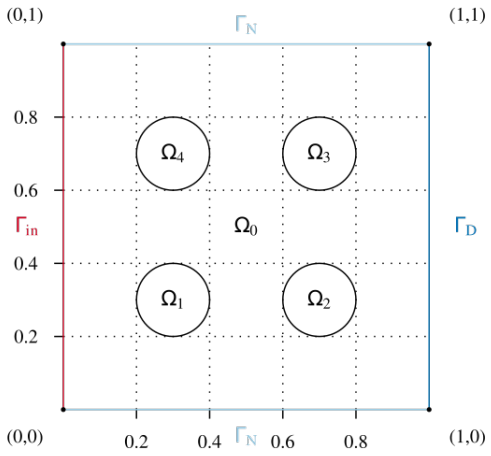


Figure 5.12.: Thermal Block: spatial domain and boundaries.

Next, we perform oversampling to identify more parameters. For the standard DEIM approach, the number of interpolation points is equal to the rank n_{EI} of the truncated left singular vectors of the snapshots matrix. For oversampling, we set the number of interpolation points to be $m = 2n_{\text{EI}}$ and test the approaches based on maximizing the

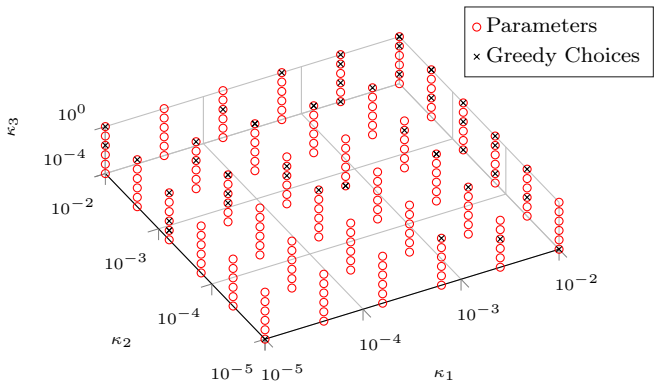


Figure 5.13.: Thermal Block: fine training set with 216 parameters and the 44 greedy parameters picked by Algorithm 4.2.

smallest singular value (Gappy-POD Eigenvector) and the approach based on clustering (Gappy-POD Clustering), both originally proposed in [159]. The results are given in Table 5.9. We see that both oversampling approaches result in ROMs that are validated to be accurate over the test set. The Gappy-POD Clustering method results in the smallest test error among the two and takes 40 iterations to converge. Notice that, compared to the previous two approaches based on QDEIM and KDEIM, the oversampling approach requires more time. This is not surprising, since a larger number of parameters is included in the coarser training set Ξ at Stage 2 of Algorithms 5.2 and 5.3. The speedup of the Gappy-POD Eigenvector variant is 3.9, while a speedup of 4.6 is achieved by the Gappy-POD Clustering variant. We show the subsampled training sets of both the approaches in Fig. 5.14. In particular, parameter samples anticipated by the Gappy-POD Clustering variant bear a close resemblance to the greedy parameter distribution in Fig. 5.13. In Fig. 5.15, we plot the mean error over time for each parameter in the test set Ξ_{test} which consists of 100 random samples from the parameter domain. It is evident that both the proposed oversampling strategies, Gappy-POD Eigenvector (Fig. 5.15a) and Gappy-POD Clustering (Fig. 5.15b) have been successful in keeping the error below the desired tolerance, uniformly for all the parameters in the test set.

5.4. Conclusion

In this chapter, we turned our focus to adaptive training set sampling for the RBM. We proposed two different approaches. Our first approach in Section 5.2 iteratively builds the training set, starting from a small number of samples. The procedure uses a surrogate model for the *a posteriori* error estimator introduced in Chapter 4. Three benchmark numerical examples were used to show the improved performance of this method.

Following this, in Section 5.3 we discussed a different strategy for adaptive training set sampling. This two-stage approach starts from a finely sampled training set and has the ability to identify a set of sparse parameter samples that are most important in approximating the solution to the given problem. The set of identified samples are

Table 5.7.: Thermal Block: results of Algorithm 5.2 with QDEIM, KDEIM and QR.

Method	Fixed	Adapted		
		$\epsilon_{\text{EI}}, \epsilon_{\text{QR}} = 1 \cdot 10^{-10}$		
		QDEIM	KDEIM	QR
n_s	216	19	19	20
ϵ_{max}	$9.78 \cdot 10^{-4}$	$1.10 \cdot 10^{-3}$	$2.10 \cdot 10^{-3}$	$6.45 \cdot 10^{-4}$
r_{RB}	74	64	64	66
Iterations	53	42	43	44
Offline time (s)	694.73	121.36	123.27	126.35
Speedup	-	5.7	5.6	5.5

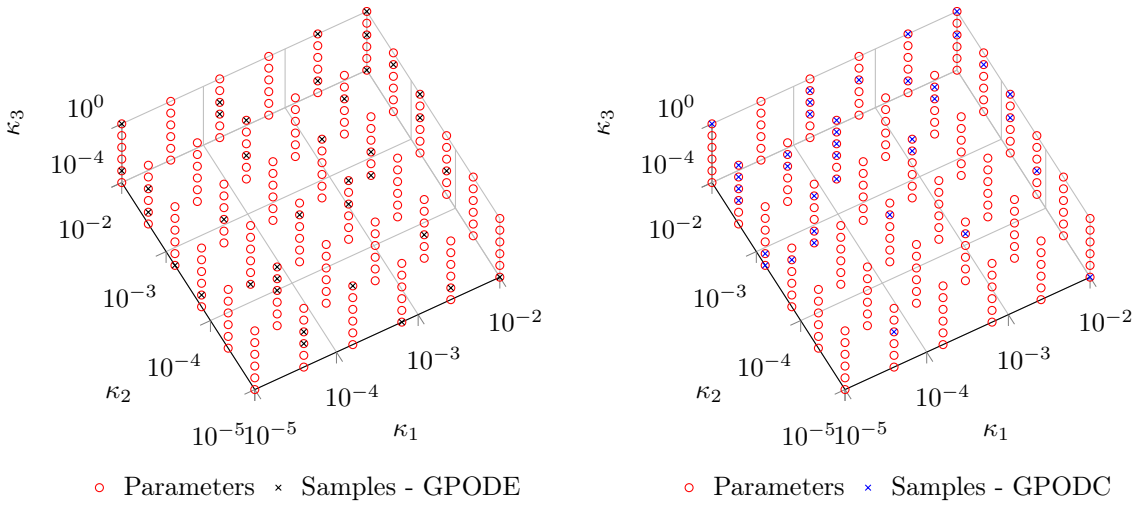
Table 5.8.: Thermal Block: results of Algorithm 5.3 with QDEIM, KDEIM and QR.

Method	Fixed	Adapted		
		$\epsilon_{\text{EI}}, \epsilon_{\text{QR}} = 1 \cdot 10^{-10}$		
		QDEIM	KDEIM	QR
n_s	216	19	19	20
ϵ_{max}	$9.78 \cdot 10^{-4}$	$1.10 \cdot 10^{-3}$	$9.36 \cdot 10^{-4}$	$1.60 \cdot 10^{-3}$
r_{RB}	74	65	62	57
Iterations	53	45	40	35
Offline time (s)	694.73	121.58	110.13	73.38
Speedup	-	5.7	6.3	9.5

Table 5.9.: Thermal Block: results of Algorithm 5.2 with oversampling.

Method	Oversampling	
	$m = 2n_{\text{EI}}$	
	Gappy-POD Eigenvector	Gappy-POD Clustering
n_c	38	38
ϵ_{max}	$9.91 \cdot 10^{-4}$	$7.96 \cdot 10^{-4}$
r_{RB}	70	62
Iterations	47	40
Offline time (s)	177.51	151.39
Speedup	3.9	4.6

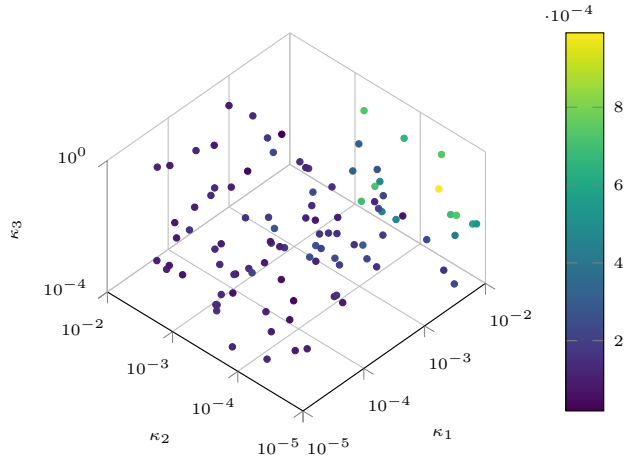
then used as a sparse training set for the RBM. We proposed different sparse sampling strategies and they were tested on two benchmark examples to show their capability to identify good training set samples.



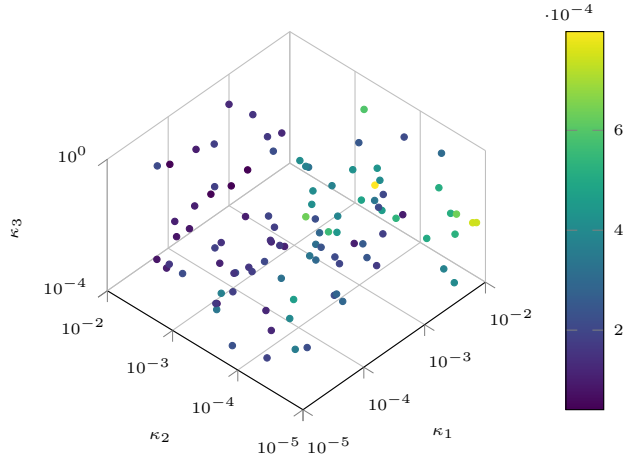
(a) Subsampled parameters using Gappy-POD Eigenvector.

(b) Subsampled parameters using Gappy-POD Clustering.

Figure 5.14.: Subsampling strategy using Gappy-POD with oversampling for the Thermal Block.



(a) Gappy-POD Eigenvector.



(b) Gappy-POD Clustering.

Figure 5.15.: Error plot for Algorithm 5.2 with coarse tolerance $\epsilon^c = 1$ and subsampling based on Gappy-POD Eigenvector and Gappy-POD Clustering applied to the thermal block example. The mean error over time between the true and reduced outputs - $(1/K+1)\sum_{i=0}^{K} \|\mathbf{y}^k(\boldsymbol{\mu}) - \widetilde{\mathbf{y}}^k(\boldsymbol{\mu})\|$ - is plotted for all parameters in the test set Ξ_{test}.

CHAPTER 6

FULLY ADAPTIVE RBM FOR COUPLED SYSTEMS

Contents

6.1. Coupled Systems

So far in this thesis, we focused on large-scale systems arising from PDEs of the form Eqs. (2.1) and (2.2). Both these PDEs are scalar-valued, in the sense that they model the dynamics of the state variable $g \in \mathbb{R}$. Moreover, they describe the dynamics of one physical quantity (for e.g., heat, temperature, velocity, etc.). In many physical and engineering systems, the underlying physical phenomenon is often in the form of a coupled system of PDEs, i.e, the PDE describes the relation between different physical quantities. A well-known example of this arises in multi-physics problems, which involve the coupling between different physical quantities. For example, fluid-structure interaction, electro-thermal coupling, etc. We now focus our attention on such coupled systems. The general time-dependent, nonlinear, parametric coupled system can be represented in the form

$$\frac{\partial}{\partial t}\mathbf{g}(\mathbf{z}, t, \boldsymbol{\mu}) = \mathscr{R}_c[\mathbf{g}(\mathbf{z}, t, \boldsymbol{\mu}), \mathbf{u}(t), \boldsymbol{\mu}], \tag{6.1}$$

similar to that of Eq. (2.1). For coupled systems, the field variable $\mathbf{g}$ is no longer scalar. We denote $\mathbf{g} := [g_1, g_2, \ldots, g_m]^\mathsf{T} \in \mathbb{R}^m$. If the PDEs model a fluid-structure interaction, then g_1 could denote the fluid pressure and g_2 the displacement of the solid structure from some mean location. Furthermore, $\mathscr{R}_c[\cdot, \cdot, \boldsymbol{\mu}]$ is a parametrized linear/nonlinear differential operator. Within the class of coupled systems, the coupling

can vary based on the physics being modelled. The PDEs can be coupled directly through the vector of field variables $\mathbf{g}$. In case of nonlinear PDEs, the coupling between the field variables can also occur indirectly, through the nonlinear function operating on the field variables. Other instances of coupled systems include interconnected systems, for which the coupling is through the inputs and the outputs: the output of one PDE is the input to the next, and so on.

Spatial and time discretization of state-coupled systems represented in the form of Eq. (6.1) can be carried out with similar standard techniques as explained previously for the PDEs in Eqs. (2.1) and (2.2). The fully discrete system can be modelled as below:

$$\underbrace{\begin{bmatrix} \overline{\mathbf{E}}_{11} & \overline{\mathbf{E}}_{12} & \cdots & \overline{\mathbf{E}}_{1m} \\ \overline{\mathbf{E}}_{21} & \overline{\mathbf{E}}_{22} & \cdots & \overline{\mathbf{E}}_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \overline{\mathbf{E}}_{m1} & \overline{\mathbf{E}}_{m2} & \cdots & \overline{\mathbf{E}}_{mm} \end{bmatrix}}_{\mathcal{E}} \underbrace{\begin{bmatrix} \mathbf{x}_1^{k+1} \\ \mathbf{x}_2^{k+1} \\ \vdots \\ \mathbf{x}_m^{k+1} \end{bmatrix}}_{\mathbf{x}^{k+1}} = \underbrace{\begin{bmatrix} \overline{\mathbf{A}}_{11} & \overline{\mathbf{A}}_{12} & \cdots & \overline{\mathbf{A}}_{1m} \\ \overline{\mathbf{A}}_{21} & \overline{\mathbf{A}}_{22} & \cdots & \overline{\mathbf{A}}_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \overline{\mathbf{A}}_{m1} & \overline{\mathbf{A}}_{m2} & \cdots & \overline{\mathbf{A}}_{mm} \end{bmatrix}}_{\mathcal{A}} \underbrace{\begin{bmatrix} \mathbf{x}_1^{k} \\ \mathbf{x}_2^{k} \\ \vdots \\ \mathbf{x}_m^{k} \end{bmatrix}}_{\mathbf{x}^{k}} + \underbrace{\begin{bmatrix} \overline{\mathbf{f}}_1(\mathbf{x}^k) \\ \overline{\mathbf{f}}_2(\mathbf{x}^k) \\ \vdots \\ \overline{\mathbf{f}}_m(\mathbf{x}^k) \end{bmatrix}}_{\mathcal{F}(\mathbf{x}^k)} + \underbrace{\begin{bmatrix} \overline{\mathbf{B}}_1 \\ \overline{\mathbf{B}}_2 \\ \vdots \\ \overline{\mathbf{B}}_m \end{bmatrix}}_{\mathcal{B}} \mathbf{u}^k,$$

$$\underbrace{\begin{bmatrix} \overline{\mathbf{y}}_1^{k+1} \\ \overline{\mathbf{y}}_2^{k+1} \\ \vdots \\ \overline{\mathbf{y}}_m^{k+1} \end{bmatrix}}_{\overline{\mathbf{y}}^{k+1}} = \underbrace{\begin{bmatrix} \mathbf{C}_1 & & & \\ & \mathbf{C}_2 & & \\ & & \ddots & \\ & & & \mathbf{C}_m \end{bmatrix}}_{\mathcal{C}} \begin{bmatrix} \mathbf{x}_1^{k+1} \\ \mathbf{x}_2^{k+1} \\ \vdots \\ \mathbf{x}_m^{k+1} \end{bmatrix} \tag{6.2}$$

Here, $\overline{\mathbf{E}}_{ij}, \overline{\mathbf{A}}_{ij} \in \mathbb{R}^{N^i \times N^j}$ are time- and/or parameter-dependent system matrices arising from the discretization of each coupled equation, with $1 \leq i, j \leq m$. $\overline{\mathbf{B}}_i \in \mathbb{R}^{N^i \times N_I^i}, \mathbf{C}_i \in \mathbb{R}^{N_O^i \times N^i}$ are the input and output matrices corresponding to the i-th coupled system, with $i \in \{1, 2, \ldots, m\}$. The total number of inputs and outputs are: $N_I = N_I^1 + \cdots + N_I^m$, $N_O = N_O^1 + \cdots + N_O^m$, respectively. The state vector $\mathbf{x}_i^k$ corresponds to the discretization of the field variable of the i-th equation, while $\overline{\mathbf{f}}_i$ is the nonlinear term associated to that equation. The overall dimension of the coupled system N is given by $N = N^1 + N^2 + \cdots + N^m$. Based on the coupling, some elements of the large matrices $\mathcal{E} \in \mathbb{R}^{N \times N}, \mathcal{A} \in \mathbb{R}^{N \times N}$ turn out to be zero, i.e., $\overline{\mathbf{E}}_{ij} = 0, \overline{\mathbf{A}}_{kl} = 0$ for some values of $i, j, k, l \in \{1, 2, \ldots, m\}$. This means that the corresponding subsystems are not coupled. However, in case of nonlinear systems, this is not entirely true. For nonlinear systems, a coupling between the state variables may arise indirectly through the operation of the nonlinear vector $\overline{\mathbf{f}}_i$ on the state vector $\mathbf{x}^k$.

6.1.1. MOR for Coupled Systems

A number of works in the past decade have considered reduced order modelling of coupled systems [27, 132, 174]. Early development was mainly driven by the need in the field of circuit theory, which called for the modelling and simulation of very large scale integration (VLSI) chips consisting of many interconnected units. ROMs for coupled systems were also considered in structural dynamics, where coupled systems

involve interconnection of different mechanical modules such as trusses and beams. Another prominent area where coupled systems are widely encountered is chemical and process engineering. Often, chemical reactions involve multiple components, governed by different physical laws. We shall especially focus on coupled systems arising in the process of batch chromatography in process engineering.

Broadly, two competing approaches have been considered in literature to obtain ROMs for coupled systems [27]. The first is the *structure-preserving approach*. For several reasons, it is important to preserve the block structure of the coupled system Eq. (6.2) in the reduced model. Therefore, ROMs are generated individually for each subsystem and they are finally assembled together based on the particular coupling law for the problem under consideration. Such an approach is adopted in many works, both for frequency- and time-domain MOR methods. Two sub-variants of this approach are discussed in [27]. The other approach for MOR of coupled systems, the *monolithic approach*, considers the system as a whole and extracts a single ROM [174]. While this approach is not so common, it is often adopted in systems and control theory. There, the coupled system is represented as a single system based on the closed-loop representation and MOR is carried out for the closed-loop system.

While MOR techniques have been extended to nonlinear coupled systems [213], only few works have considered parametric systems. Among them, only a handful have considered adaptive MOR techniques in the context of the RBM. To fill this gap, in this thesis, our goal is to extend adaptive MOR techniques introduced in the previous sections (adaptive basis enrichment, adaptive parameter sampling) to general nonlinear, parametric, time-dependent coupled systems.

6.1.2. Adaptive Structure-Preserving MOR for Coupled Systems

We adopt the *structure-preserving approach* in order to carry out adaptive MOR for coupled systems. For a coupled system involving m subsystems, we shall consider separate projection matrices $\mathbf{V}$ for each subsystem. For the FOM in Eq. (6.2), we begin with the following ansatz:

$$\mathbf{x}^k \approx \widetilde{\mathbf{x}}_i^k := \mathbf{V}_i \widehat{\mathbf{x}}_i, \qquad i = 1, 2, \ldots, m. \tag{6.3}$$

The matrix $\mathbf{V}_i \in \mathbb{R}^{N^i \times n^i}, i = 1, 2, \ldots, m$, is the RB projection matrix for the i-th subsystem. Here, $\{n^i\}_{i=1}^m$ are the reduced dimensions of each subsystem constituting the coupled system and $n = n^1 + n^2 + \cdots + n^m$. Substituting this in Eq. (6.2) and performing a Galerkin projection results in the reduced coupled system of the form

$$\begin{aligned} \widehat{\mathcal{E}}\widehat{\mathbf{x}}^{k+1} &= \widehat{\mathcal{A}}\widehat{\mathbf{x}}^k + \widehat{\mathcal{F}}(\widetilde{\mathbf{x}}^k) + \widehat{\mathcal{B}}\mathbf{u}^k, \\ \widetilde{\mathbf{y}}^{k+1} &= \widehat{\mathcal{C}}\widehat{\mathbf{x}}^{k+1}. \end{aligned} \tag{6.4}$$

with $\widehat{\mathcal{E}}, \widehat{\mathcal{A}} \in \mathbb{R}^{n \times n}$ being the reduced system matrices and $\widehat{\mathcal{B}} \in \mathbb{R}^{n \times N_I}, \widehat{\mathcal{C}} \in \mathbb{R}^{N_O \times n}$ are the input and output matrix, respectively, of the reduced coupled system. We denote

the block projection matrix as

$$\mathbf{V} := \begin{bmatrix} \mathbf{V}_1 & & & \\ & \mathbf{V}_2 & & \\ & & \ddots & \\ & & & \mathbf{V}_m \end{bmatrix}. \tag{6.5}$$

The reduced matrices of the coupled system have the forms as shown below:

$$\widehat{\mathcal{E}} := \mathbf{V}^{\mathsf{T}}\mathcal{E}\mathbf{V} = \begin{bmatrix} \mathbf{V}_1^{\mathsf{T}}\overline{\mathbf{E}}_{11}\mathbf{V}_1 & \mathbf{V}_1^{\mathsf{T}}\overline{\mathbf{E}}_{12}\mathbf{V}_2 & \cdots & \mathbf{V}_1^{\mathsf{T}}\overline{\mathbf{E}}_{1m}\mathbf{V}_m \\ \mathbf{V}_2^{\mathsf{T}}\overline{\mathbf{E}}_{21}\mathbf{V}_1 & \mathbf{V}_2^{\mathsf{T}}\overline{\mathbf{E}}_{22}\mathbf{V}_2 & \cdots & \mathbf{V}_2^{\mathsf{T}}\overline{\mathbf{E}}_{2m}\mathbf{V}_m \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{V}_m^{\mathsf{T}}\overline{\mathbf{E}}_{m1}\mathbf{V}_1 & \mathbf{V}_m^{\mathsf{T}}\overline{\mathbf{E}}_{m2}\mathbf{V}_2 & \cdots & \mathbf{V}_m^{\mathsf{T}}\overline{\mathbf{E}}_{mm}\mathbf{V}_m \end{bmatrix}, \tag{6.6}$$

$$\widehat{\mathcal{A}} := \mathbf{V}^{\mathsf{T}}\mathcal{A}\mathbf{V} = \begin{bmatrix} \mathbf{V}_1^{\mathsf{T}}\overline{\mathbf{A}}_{11}\mathbf{V}_1 & \mathbf{V}_1^{\mathsf{T}}\overline{\mathbf{A}}_{12}\mathbf{V}_2 & \cdots & \mathbf{V}_1^{\mathsf{T}}\overline{\mathbf{A}}_{1m}\mathbf{V}_m \\ \mathbf{V}_2^{\mathsf{T}}\overline{\mathbf{A}}_{21}\mathbf{V}_1 & \mathbf{V}_2^{\mathsf{T}}\overline{\mathbf{A}}_{22}\mathbf{V}_2 & \cdots & \mathbf{V}_2^{\mathsf{T}}\overline{\mathbf{A}}_{2m}\mathbf{V}_m \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{V}_m^{\mathsf{T}}\overline{\mathbf{A}}_{m1}\mathbf{V}_1 & \mathbf{V}_m^{\mathsf{T}}\overline{\mathbf{A}}_{m2}\mathbf{V}_2 & \cdots & \mathbf{V}_m^{\mathsf{T}}\overline{\mathbf{A}}_{mm}\mathbf{V}_m \end{bmatrix}, \tag{6.7}$$

$$\widehat{\mathcal{B}} := \mathbf{V}^{\mathsf{T}}\mathcal{B} = \begin{bmatrix} \overline{\mathbf{B}}_1^{\mathsf{T}}\mathbf{V}_1 & \overline{\mathbf{B}}_2^{\mathsf{T}}\mathbf{V}_2 & \cdots & \overline{\mathbf{B}}_m^{\mathsf{T}}\mathbf{V}_m \end{bmatrix}^{\mathsf{T}}, \tag{6.8}$$

$$\widehat{\mathcal{C}} := \mathcal{C}\mathbf{V} = \begin{bmatrix} \mathbf{C}_1\mathbf{V}_1 & & & \\ & \mathbf{C}_2\mathbf{V}_2 & & \\ & & \ddots & \\ & & & \mathbf{C}_m\mathbf{V}_m \end{bmatrix}. \tag{6.9}$$

The nonlinear term $\widehat{\mathcal{F}}$ is of the form

$$\widehat{\mathcal{F}} := \mathbf{V}^{\mathsf{T}}\mathcal{F} = \begin{bmatrix} \overline{\mathbf{f}}_1(\mathbf{x}^k)^{\mathsf{T}}\mathbf{V}_1 & \overline{\mathbf{f}}_2(\mathbf{x}^k)^{\mathsf{T}}\mathbf{V}_2 & \cdots \overline{\mathbf{f}}_m(\mathbf{x}^k)^{\mathsf{T}}\mathbf{V}_m \end{bmatrix}^{\mathsf{T}} \tag{6.10}$$

Since there exist nonlinear terms in Eq. (6.2), we invoke the hyperreduction techniques to enable efficient online computation. As considered for the RB projection matrix, we obtain separate EI projection matrices and interpolation indices corresponding to each subsystem forming the coupled system. The block EI projection matrix $\mathbf{U}$ is of the form

$$\mathbf{U} := \begin{bmatrix} \mathbf{U}_1 & & & \\ & \mathbf{U}_2 & & \\ & & \ddots & \\ & & & \mathbf{U}_m \end{bmatrix} \tag{6.11}$$

and the interpolation indices $\{\mathcal{I}_i\}_{i=1}^m$ are collected in the set $\mathcal{I} := \{\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_m\}$. Similar to Eq. (2.36), the hyperreduced ROM for Eq. (6.2) is given by

$$\begin{aligned} \widehat{\mathcal{E}}\widehat{\mathbf{x}}^{k+1} &= \widehat{\mathcal{A}}\widehat{\mathbf{x}}^k + \boldsymbol{\mathcal{U}}\mathfrak{f} + \widehat{\mathcal{B}}\mathbf{u}^k, \\ \widetilde{\mathbf{y}}^{k+1} &= \widehat{\mathcal{C}}\widehat{\mathbf{x}}^{k+1}. \end{aligned} \tag{6.12}$$

The quantity $\boldsymbol{\mathcal{U}} := \mathbf{V}^{\mathsf{T}}\mathbf{U}(\mathbf{U}^{\mathcal{J}})^{-1}$ can be precomputed once and for all and stored offline, whereas $\mathfrak{f} := (\overline{\mathbf{f}}(\mathbf{x}^k, \boldsymbol{\mu}))^{\mathcal{J}} \in \mathbb{R}^{n_{\mathrm{EI}}}$ can be cheaply computed online since $n_{\mathrm{EI}} \ll N$. The selection of the rows of the EI bases in the term $\mathbf{U}^{\mathcal{J}}$ is done for each subsystem as shown below:

$$\mathbf{U}^{\mathcal{J}} = \begin{bmatrix} \mathbf{U}_1^{\mathcal{J}_1} & & & \\ & \mathbf{U}_2^{\mathcal{J}_2} & & \\ & & \ddots & \\ & & & \mathbf{U}_m^{\mathcal{J}_m} \end{bmatrix}. \tag{6.13}$$

Similarly, the evaluation of the quantity $\mathfrak{f}$ in Eq. (6.12) is also done component-wise based on the respective EI sampling points:

$$\mathfrak{f} = \begin{bmatrix} \overline{\mathbf{f}}_1^{\mathcal{J}_1} \\ \overline{\mathbf{f}}_2^{\mathcal{J}_2} \\ \vdots \\ \overline{\mathbf{f}}_m^{\mathcal{J}_m} \end{bmatrix}. \tag{6.14}$$

Having discussed the structure-preserving ROM for coupled systems, we next apply the adaptive basis enrichment and adaptive training set sampling methods to efficiently obtain the RB and EI projection bases and sampling points.

Remark 6.1:
For the remainder of the discussion, we shall restrict ourselves to the case of the matrices $\mathcal{E}, \mathcal{A}$ being diagonal. We have $\overline{\mathbf{E}}_{ij} = \overline{\mathbf{A}}_{ij} = 0$ for all $i, j \in \{1, 2, \ldots, m\}$ & $i \neq j$, i.e., subsystems are coupled indirectly through the nonlinear term. The numerical example we shall deal with, viz., coupled systems arising in the process of batch chromatography are precisely in this form. ◊

6.1.2.1. Adaptive Basis Enrichment for Coupled Systems

To enable the application of adaptive strategies, we first discuss error estimation. We make use of the *a posteriori* output error estimator $\Delta_{\mathbf{y}}$ from Eq. (4.30). For coupled systems, we require m error estimators, one for each subsystem denoted by $\Delta_{\mathbf{y},i}, i \in \{1, 2, \ldots, m\}$.

Remark 6.2:
For MIMO systems, this can potentially become expensive, as in total, the error needs to be estimated for $m \cdot N_O$ output variables. However, in practice, not all subsystems/variables are important, and it is usually enough to estimate the errors for a total of m_c critical subsystems and $N_{O,c}$ critical variables. Therefore, the error estimation is computationally tractable. In our discussion, we assume that the error needs to be estimated for only $e_c = (m_c \cdot N_{O,c}) \ll (m \cdot N_O)$ variables in Eq. (6.12). We denote by $\{\alpha_1, \ldots, \alpha_{e_c}\}$ the indices of those variables in Eq. (6.2) which are critical. ◊

Recall from Section 4.2.2 that the t-dual system Eq. (4.1) needs to be solved to compute the error estimator in Eq. (4.30). This means, e_c dual systems

$$\overline{\mathbf{E}}_{\alpha_k \alpha_k}^{\mathsf{T}} \mathbf{x}_{\mathrm{du},k} = -\mathbf{C}_k^{\mathsf{T}}, \qquad k = 1, 2, \ldots, e_c \tag{6.15}$$

are to be solved. Let us denote the block dual projection matrix for coupled systems by

$$\mathbf{V}_{\mathrm{du}} := \begin{bmatrix} \mathbf{V}_{\mathrm{du},1} & & & \\ & \mathbf{V}_{\mathrm{du},2} & & \\ & & \ddots & \\ & & & \mathbf{V}_{\mathrm{du},e_c} \end{bmatrix}. \tag{6.16}$$

We define the estimated error as the maximum among the errors estimated for the e_c variables, i.e.,

$$\Delta_{\mathbf{y},c} := \max_{i \in \{1,2,\ldots,e_c\}} \Delta_{\mathbf{y},i}. \tag{6.17}$$

We first detail the adaptive basis enrichment scheme for coupled systems in Algorithm 6.1. The algorithm follows the essence of the procedure outlined previously in Algorithm 4.2; however, as we will describe below, it is suitably modified to treat coupled systems. In case the dual systems are non-parametric, Krylov-space methods are used in Step 1 to obtain the dual system solutions. Following this, the initialization of the RB, EI projection bases and interpolation points is carried out. It is worth mentioning that for coupled systems the number of POD modes $\mathbf{r}_{\mathrm{RB}}, \mathbf{r}_{\mathrm{EI}}$ to be added/removed at each iteration are now vectors defined as $\mathbf{r}_{\mathrm{RB}} := [r_{\mathrm{RB},1}, \ldots, r_{\mathrm{RB,m}}]^{\mathsf{T}} \in \mathbb{R}^m$, $\mathbf{r}_{\mathrm{EI}} := [r_{\mathrm{EI},1}, \ldots, r_{\mathrm{EI,m}}]^{\mathsf{T}} \in \mathbb{R}^m$ with the entries denoting the number of RB or EI basis vectors for each subsystem. Upon entering the greedy loop, in Step 5 we collect in the set S, the indices j of all entries in $\mathbf{r}_{\mathrm{RB}}$ for which $r_{\mathrm{RB},j} > 0$. In Step 6, the FOM Eq. (6.2) corresponding to each subsystem $j \in S$ is solved for the current greedy parameter and the snapshots of the state vectors $\mathbf{x}_j$ are collected and stored in the snapshot matrices $\mathbf{X}_j$. Based on the new information from the snapshots, the RB projection matrices $\mathbf{V}_j$ are updated in Step 8 or Step 12. For all entries $r_{\mathrm{RB},j}$, $j \notin S$, we then suitably delete columns from the corresponding $\mathbf{V}_j$ in Step 14. Steps 15-16 involve the simultaneous enrichment of the EI projection bases and sampling points. We first collect the snapshots of the nonlinear term corresponding to each subsystem into the matrix $\mathbf{F}_i$ in Step 15. Step 16 then invokes the subroutine `update_ei` (see Line 26 in Algorithm 4.2) for every subsystem, to enrich $\mathbf{U}_i, \mathfrak{I}_i$. The error estimation is done for the e_c output variables in Step 20. We pick the maximum estimated error among all variables. Finally, in Step 22, each entry in $\mathbf{r}_{\mathrm{RB}}, \mathbf{r}_{\mathrm{EI}}$ is updated through the subroutine `adapt_basis_update` (see Line 35 in Algorithm 4.2).

6.1.2.2. Fully Adaptive RBMEI algorithm for Coupled Systems

The adaptive training set sampling based on a surrogate error estimator was introduced in Section 5.2.1. We now extend this to coupled systems. Since the implementation and computational costs were already discussed in detail, we do not repeat it here and simply sketch the pseudocode in Algorithm 6.2. Concerning the surrogate error estimator, since we have e_c quantities of interest, we build RBF-surrogates for the estimated error corresponding to each of those variables in Step 21. Step 22 involves pruning the coarse training set Ξ_c. We monitor the maximum estimated error $\Delta_{\mathbf{y},c}$

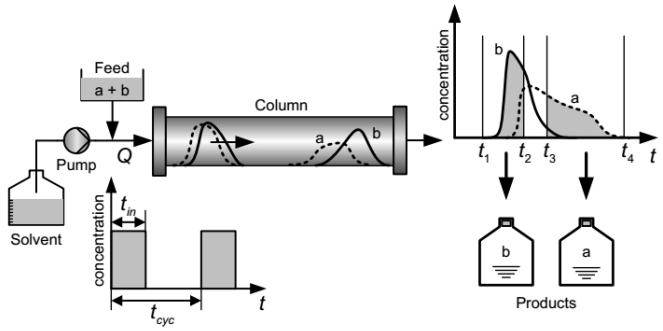


Figure 6.1.: The schematic of a Batch Chromatography for binary separation.

among all the subsystems, and remove samples from Ξ_c for which $\Delta_{\mathbf{y},c} < \epsilon$. In Steps 23 and 24, we add new parameters to Ξ_c. For this, we first identify the surrogate $\chi_i(\boldsymbol{\mu})$ which yields the maximum error among all subsystems. Then, we add n_{add} new parameters corresponding to largest errors identified by $\chi_i(\boldsymbol{\mu})$. The remaining steps are nearly the same as described for Algorithm 6.1.

6.1.3. Numerical Example: Batch Chromatography

The process of chromatography as a separation and purification is widely used in the fields of chemical and process engineering. It is a critical part of drug discovery and bio-chemical purification. There exist different types of chromatography such as discontinuous or batch chromatography, simulated moving bed chromatography, gas chromatography, etc. We shall mainly consider the batch elution chromatography, used for binary separation. Figure 6.1 shows the schematic of this process taken from [213]. The mixture containing two components that need to be separated is periodically injected at one end of the column. In the column, a static bed of a substance called the stationary phase is present. The injected mixture has to pass through this stationary phase. Batch chromatography relies on the phenomenon of adsorption. The components that need to be separated have different adsorption affinities toward the stationary phase and, hence, tend to move through the column with varying velocities. The separated components are then collected at the end of the column. The time of collecting the two components of the original mixture is determined based on the required purity specifications.

Algorithm 6.1: `ADAPTIVE_RBMEI_COUPLED`
Computes a ROM using the Adaptive POD-Greedy algorithm in combination with hyperreduction for coupled systems.

Input: System matrices and nonlinear term: $\mathcal{E}, \mathcal{A}, \mathcal{B}, \mathcal{C}$ and $\mathcal{F}$, Training set Ξ, EI tolerance ϵ_{EI}, ROM tolerance ϵ, zoa ϵ^*, Deflation tolerance ϵ_{def}, Maximum iterations *iter_max*.

Output: Reduced system quantities $\widehat{\mathcal{E}}, \widehat{\mathcal{A}}, \widehat{\mathcal{B}}, \widehat{\mathcal{C}}$ and EI quantities $\boldsymbol{\mathcal{U}}, \mathcal{I}$.

For non-parametric dual systems solve e_c t-dual systems of the form Eq. (4.2), obtain $\{\widetilde{\mathbf{x}}_{\text{du},1}, \ldots, \widetilde{\mathbf{x}}_{\text{du,ec}}\}$ using a Krylov-space method.
Use RBF interpolation to approximate *inf-sup* constants $\widetilde{\sigma}_{\min}(\boldsymbol{\mu})$ for all $\boldsymbol{\mu} \in \Xi$.
Initialization: $\{\mathbf{V}_i, \mathbf{V}_{\text{du},i}, \mathbf{U}_i\}_{i=1}^m = [\,]$, $\mathcal{I} = \varnothing$, $err\_max = 1 + \epsilon$, $err\_max\_dual = 1 + \epsilon$, $iter = 1$, $\boldsymbol{\mu}^*, \boldsymbol{\mu}^*_{\text{du}}$ (chosen randomly from Ξ), $\mathbf{r}_{\text{RB}} = \mathbf{r}_{\text{EI}} = 1$.
while $err\_max \notin zoa$ and $iter \leq iter\_max$ **do**
  Denote by S the set of all indices j such that $r_{\text{RB},j} > 0$.
  Solve FOM Eq. (6.2) at $\boldsymbol{\mu}^*$; obtain snapshot matrices $\{\mathbf{X}_j(\boldsymbol{\mu}^*)\}$, $j \in S$.
  if $iter == 1$ **then**
    $\mathbf{V}_j = \texttt{POD}(\mathbf{X}_j, r_{\text{RB,j}})$, $j \in S$.
  else
    Compute $\overline{\mathbf{X}}_j := \mathbf{X}_j - \mathbf{V}_j \mathbf{V}_j^{\mathsf{T}} \mathbf{X}_j$, $j \in S$.
    $\mathbf{V}_{\text{POD},j} := \texttt{POD}(\overline{\mathbf{X}}_j, r_{\text{RB},j})$, $j \in S$.
    $\mathbf{V}_j = \texttt{orth\_def\_mat}(\mathbf{V}_j, \mathbf{V}_{\text{POD},j}, \epsilon_{\text{def}})$, $j \in S$. `/* see Step 15 of Algorithm 2.4 */`
  end
  Remove the last $r_{\text{RB},j}$ columns from $\mathbf{V}_j$, $j \notin S$.
  Form snapshot matrices for the nonlinear function: $\mathbf{F}_i := [\mathbf{F}_i \;\; \overline{\mathbf{f}}_i(\overline{\mathbf{X}}_i)]$, $i = 1, 2, \ldots, m$.
  Update EI basis and interpolation points $\mathbf{U}_i, \mathcal{I}_i$ using the subroutine `update_ei` for each subsystem and form $\mathbf{U}^{\mathcal{I}}, \mathfrak{f}$ (see Eqs. (6.13) and (6.14)).
  For a parametric t-dual system, update dual RB matrix $\mathbf{V}_{\text{du,i}}$ using the subroutine `update_dual` for the subsystems.
  $iter = iter + 1$.
  Determine reduced matrices $\widehat{\mathcal{E}}, \widehat{\mathcal{A}}, \widehat{\mathcal{B}}, \widehat{\mathcal{C}}$ through Galerkin projection using $\mathbf{V}_i$ and hyperreduction using $\mathbf{U}_i$.
  Solve the ROM Eq. (6.12) for all $\boldsymbol{\mu} \in \Xi$; find $\boldsymbol{\mu}^* = \arg\max_{\boldsymbol{\mu} \in \Xi} \Delta_{\mathbf{y},c}(\boldsymbol{\mu})$.
  Set $err\_max = \Delta_{\mathbf{y},c}(\boldsymbol{\mu}^*)$.
  Update $\mathbf{r}_{\text{RB}}, \mathbf{r}_{\text{EI}}$ using the subroutine `adapt_basis_update`.
end

6.1.3.1. Mathematical Model

The model of the batch chromatography was originally proposed in [213]. Our discussion closely follows the presentation there. The process can be modelled using a system of coupled PDEs. There exists a coupling between the solid and liquid phase

concentrations of the two components. The coupled PDEs governing the process are given by

$$\frac{\partial \mathring{g}_\beta}{\partial t} + \frac{1-\mathring{p}}{\mathring{p}}\frac{\partial \mathring{g}_\beta}{\partial t} = -\frac{\partial \mathring{g}_\beta}{\partial z} + \frac{1}{Pe}\frac{\partial^2 \mathring{g}_\beta}{\partial z^2}, \qquad 0 < z < 1, \beta \in \{a, b\} \tag{6.18a}$$

$$\frac{\partial \overline{g}_\beta}{\partial t} = \frac{L}{Q/\mathring{p}A_c}\kappa_\beta(\overline{g}_\beta^{\mathrm{Eq}} - \overline{g}_\beta), \qquad 0 \le z \le 1. \tag{6.18b}$$

with $\mathring{g}_\beta, \overline{g}_\beta$ being, respectively, the solid and liquid phase concentrations of the two components $\beta \in \{a, b\}$ which need to be separated. For easy modelling, we have considered an idealized 1-D domain $\Omega : [0, 1]$ with z denoting the spatial variable. This assumption is not simplistic since there is generally a uniform concentration of the reactants along the cross-section of the column. $\mathring{p}$ denotes the column porosity. The porosity is crucial for the interaction between the solid and liquid phases. Further, we have the Péclet number Pe, which is the ratio of the rate of advection to the rate of diffusion. For the batch chromatography process, the advection tends to dominate the diffusion and we have $Pe \gg 1$. Concerning the geometry of the column, L denotes its length while A_c is the cross-section area. Q refers to the volumetric feed flow, the amount of solvent injected per second into the column. The injection of the solvent is done periodically, in batches. The time t_{in} is the duration for which the solvent is pumped while t_{cyc} is the duration of a single cycle of injection and flow through the column. The mass transfer coefficients for each component is κ_β.

The source of nonlinearity in this model is the adsorption equilibrium $\overline{g}_\beta^{\mathrm{Eq}}$ denoted by

$$\overline{g}_\beta^{\mathrm{Eq}} = \frac{H_{\beta 1}\mathring{g}_\beta}{1 + K_{a1}\mathring{g}_a^f\mathring{g}_a + K_{b1}\mathring{g}_b^f\mathring{g}_b} + \frac{H_{\beta 2}\mathring{g}_\beta}{1 + K_{a2}\mathring{g}_a^f\mathring{g}_a + K_{b2}\mathring{g}_b^f\mathring{g}_b} \tag{6.19}$$

where $H_{\beta 1}, H_{\beta 2}$ are Henry's constants, $K_{\beta 1}, K_{\beta 2}$ are thermodynamic coefficients, and $\mathring{g}_\beta^f$ are the feed concentrations of the two components $\beta \in \{a, b\}$. We impose the following initial and boundary conditions:

$$\mathring{g}_\beta(0, z) = 0, \qquad \overline{g}_\beta(0, z) = 0, \qquad 0 \le z \le 1, \tag{6.20a}$$

$$\frac{\partial \mathring{g}_\beta}{\partial z}|_{z=0} = Pe\big(\mathring{g}_\beta(t, 0) - u_{[0,t_{\mathrm{in}}]}(t)\big), \tag{6.20b}$$

$$\frac{\partial \mathring{g}_\beta}{\partial z}|_{z=1} = 0. \tag{6.20c}$$

Here, $u_{[0,t_{\mathrm{in}}]}(t)$ defines the periodic input $u(t) = 1, t \in [0, t_{in}]$ and $u(t) = 0, t \notin [0, t_{in}]$. The volumetric feed flow Q and the injection time t_{in} are considered as the parameters that can be varied.

The coupled system Eq. (6.18) is spatially discretized using FVM and a second-order Crank-Nicolson method is employed for time discretization. The discretized system can

be represented in the form of Eq. (6.2) as:

$$\underbrace{\begin{bmatrix}\overline{\mathbf{E}} & & & \\ & \overline{\mathbf{E}} & & \\ & & \mathfrak{J} & \\ & & & \mathfrak{J}\end{bmatrix}}_{\mathcal{E}}\underbrace{\begin{bmatrix}\mathbf{x}_1^{k+1}\\ \mathbf{x}_2^{k+1}\\ \mathbf{x}_3^{k+1}\\ \mathbf{x}_4^{k+1}\end{bmatrix}}_{\mathbf{x}^{k+1}} = \underbrace{\begin{bmatrix}\overline{\mathbf{A}} & & & \\ & \overline{\mathbf{A}} & & \\ & & \mathfrak{J} & \\ & & & \mathfrak{J}\end{bmatrix}}_{\mathcal{A}}\underbrace{\begin{bmatrix}\mathbf{x}_1^{k}\\ \mathbf{x}_2^{k}\\ \mathbf{x}_3^{k}\\ \mathbf{x}_4^{k}\end{bmatrix}}_{\mathbf{x}^{k}} + \underbrace{\begin{bmatrix}\frac{\mathring{p}-1}{\mathring{p}}\\ 1\end{bmatrix}\otimes\begin{bmatrix}\overline{\mathbf{f}}_a(\mathbf{x}^k)\\ \overline{\mathbf{f}}_b(\mathbf{x}^k)\end{bmatrix}}_{\mathcal{F}(\mathbf{x}^k)} + \underbrace{\begin{bmatrix}\overline{\mathbf{B}}\\ \overline{\mathbf{B}}\\ \mathbf{0}\\ \mathbf{0}\end{bmatrix}}_{\mathcal{B}}\mathbf{u}^k,$$

$$\underbrace{\begin{bmatrix}\overline{\mathbf{y}}_1^{k+1}\\ \overline{\mathbf{y}}_2^{k+1}\\ \overline{\mathbf{y}}_3^{k+1}\\ \overline{\mathbf{y}}_4^{k+1}\end{bmatrix}}_{\overline{\mathbf{y}}^{k+1}} = \underbrace{\begin{bmatrix}\mathbf{C} & & & \\ & \mathbf{C} & & \\ & & \mathbf{0} & \\ & & & \mathbf{0}\end{bmatrix}}_{\mathcal{C}}\begin{bmatrix}\mathbf{x}_1^{k+1}\\ \mathbf{x}_2^{k+1}\\ \mathbf{x}_3^{k+1}\\ \mathbf{x}_4^{k+1}\end{bmatrix}. \tag{6.21}$$

The vectors $\mathbf{x}_1^k, \mathbf{x}_2^k \in \mathbb{R}^{N^1}$ are the discretized version of the liquid phase concentrations, $\mathring{g}_a, \mathring{g}_b$, respectively; the vectors $\mathbf{x}_3^k, \mathbf{x}_4^k \in \mathbb{R}^{N^2}$ are, respectively, the discretized form of the solid phase concentrations $\overline{g}_a, \overline{g}_b$ of the two components a, b in the solvent. The overall state vector at some time $t = t^k$ is $\mathbf{x}^k \in \mathbb{R}^N$ with $N = 2N^1 + 2N^2$ and $N^1 = N^2$. The term $\mathcal{F}(\mathbf{x}^k) \in \mathbb{R}^{4N^1}$ is the discretized nonlinear vector where

$$\overline{\mathbf{f}}_a^k := \Delta t \frac{L}{Q/\mathring{p}A_c}\kappa_a\big(\mathbf{x}^{\mathrm{Eq}}(\mathbf{x}_1^k, \mathbf{x}_2^k) - \mathbf{x}_3^k\big) \in \mathbb{R}^{N^1},$$
$$\overline{\mathbf{f}}_b^k := \Delta t \frac{L}{Q/\mathring{p}A_c}\kappa_b\big(\mathbf{x}^{\mathrm{Eq}}(\mathbf{x}_1^k, \mathbf{x}_2^k) - \mathbf{x}_4^k\big) \in \mathbb{R}^{N^1}.$$

Here, $\mathbf{x}^{\mathrm{Eq}}(\mathbf{x}_1^k, \mathbf{x}_2^k)$ is the discretized representation of the adsorption equilibrium in Eq. (6.19). From the above equation, it is clear how there exists a coupling between the solid and liquid phase concentrations. The matrices $\overline{\mathbf{E}}, \overline{\mathbf{A}} \in \mathbb{R}^{N^1 \times N^1}$ are non-parametric and tridiagonal. $\mathfrak{J} \in \mathbb{R}^{N^2 \times N^2}$ is the identity matrix. The input matrix is given by $\overline{\mathbf{B}} := \begin{bmatrix} d_0 & 0 & \cdots & 0\end{bmatrix}^{\mathsf{T}} \in \mathbb{R}^{N^1}$ with

$$d_0 = \Delta z \cdot Pe \cdot \big(\frac{\Delta t}{2\Delta z} + \frac{\Delta t}{Pe \cdot \Delta z^2}\big) u_{[0,t_{\mathrm{in}}]}(t^k),$$

Δz and Δt being the space and time discretization step sizes. Although Eq. (6.21) is a coupling of $m = 4$ subsystems, there are only two output variables of interest, viz., $\overline{\mathbf{y}}_1^k, \overline{\mathbf{y}}_2^k$. These variables denote the liquid phase concentrations at the last node of the spatial domain, where the solvent exits the column. The output matrix is of the form $\mathbf{C} := \begin{bmatrix}0 & 0 & \cdots & 1\end{bmatrix} \in \mathbb{R}^{1 \times N^1}$. Table 6.1 summarizes the values of the various parameters used in the model.

Algorithm 6.2: `ADAPTIVE_RBMEI_COUPLED_TS1`
Computes a ROM using the Adaptive POD-Greedy algorithm in combination with hyperreduction and a training set sampling using surrogate error estimator for coupled systems.

Input: System matrices and nonlinear term: $\mathcal{E}, \mathcal{A}, \mathcal{B}, \mathcal{C}$ and $\mathcal{F}$, Training sets Ξ_c, Ξ_f, EI tolerance ϵ_{EI}, ROM tolerance ϵ, zoa ϵ^*, Deflation tolerance ϵ_{def}, Maximum number of iterations *iter_max*.

Output: Reduced system quantities $\widehat{\mathcal{E}}, \widehat{\mathcal{A}}, \widehat{\mathcal{B}}, \widehat{\mathcal{C}}$ and EI quantities $\boldsymbol{\mathcal{U}}$, $\mathcal{I}$.

For non-parametric dual systems solve e_c t-dual systems of the form Eq. (4.2), obtain $\{\widetilde{\mathbf{x}}_{\text{du},1}, \ldots, \widetilde{\mathbf{x}}_{\text{du},\text{ec}}\}$ using a Krylov-space method.

Use RBF interpolation to approximate *inf-sup* constants $\widetilde{\sigma}_{\min}(\boldsymbol{\mu})$ for all $\boldsymbol{\mu} \in \Xi_c$.

Initialization: $\{\mathbf{V}_i, \mathbf{V}_{\text{du},i}, \mathbf{U}_i\}_{i=1}^m = [\,]$, $\mathcal{I} = \varnothing$, $err\_max = 1 + \epsilon$, $iter = 1$, $\boldsymbol{\mu}^*$, $err\_max\_dual = 1 + \epsilon$, $\boldsymbol{\mu}^*_{\text{du}}$ (chosen randomly from Ξ_c), $\mathbf{r}_{\text{RB}} = \mathbf{r}_{\text{EI}} = 1$.

while $err\_max \notin zoa$ and $iter \leq iter\_max$ **do**

  Denote by S the set of all indices j such that $r_{\text{RB},j} > 0$.

  Solve FOM Eq. (6.2) at $\boldsymbol{\mu}^*$; obtain snapshot matrices $\{\mathbf{X}_j(\boldsymbol{\mu}^*)\}$, $j \in S$.

  if $iter == 1$ **then**

    $\mathbf{V}_j = \texttt{POD}(\mathbf{X}_j, r_{\text{RB},j})$, $j \in S$.

  else

    Compute $\overline{\mathbf{X}}_j := \mathbf{X}_j - \mathbf{V}_j \mathbf{V}_j^{\mathsf{T}} \mathbf{X}_j$, $j \in S$.

    $\mathbf{V}_{\text{POD},j} := \texttt{POD}(\overline{\mathbf{X}}_j, r_{\text{RB},j})$, $j \in S$.

    $\mathbf{V}_j = \texttt{orth\_def\_mat}(\mathbf{V}_j, \mathbf{V}_{\text{POD},j}, \epsilon_{\text{def}})$, $j \in S$. `/* see Step 15 of Algorithm 2.4 */`

  end

  Remove the last $r_{\text{RB},j}$ columns from $\mathbf{V}_j$, $j \notin S$.

  Form snapshot matrices for the nonlinear function: $\mathbf{F}_i := [\mathbf{F}_i \;\; \overline{\mathbf{f}}_i(\overline{\mathbf{X}_i})]$, $i = 1, 2, \ldots, m$.

  Update EI basis and interpolation points $\mathbf{U}_i, \mathcal{I}_i$ using the subroutine `update_ei` for each subsystem and form $\mathbf{U}^{\mathcal{I}}, \mathfrak{f}$ (see Eqs. (6.13) and (6.14)).

  For a parametric t-dual system, update dual RB matrix $\mathbf{V}_{\text{du},i}$ using the subroutine `update_dual` for the subsystems.

  $iter = iter + 1$.

  Determine reduced matrices $\widehat{\mathcal{E}}, \widehat{\mathcal{A}}, \widehat{\mathcal{B}}, \widehat{\mathcal{C}}$ through Galerkin projection using $\mathbf{V}_i$ and hyperreduction using $\mathbf{U}_i$.

  Solve the ROM Eq. (6.12) for all $\boldsymbol{\mu} \in \Xi_c$.

  Define $D_i := \{\Delta_{\mathbf{y},i}(\boldsymbol{\mu}), \boldsymbol{\mu} \in \Xi_c\}$ and $\boldsymbol{\Lambda}_i := \Xi_c$, solve Eq. (2.43) to obtain RBF surrogate $\chi_i(\boldsymbol{\mu})$ for all $\boldsymbol{\mu} \in \Xi_f$ for every variable with index $i \in \{\alpha_1, \ldots, \alpha_{e_c}\}$.

  Update Ξ_c: remove $\boldsymbol{\mu} \in \Xi_c$ for which $\Delta_{\mathbf{y},c} < \epsilon$ for all e_c subsystems.

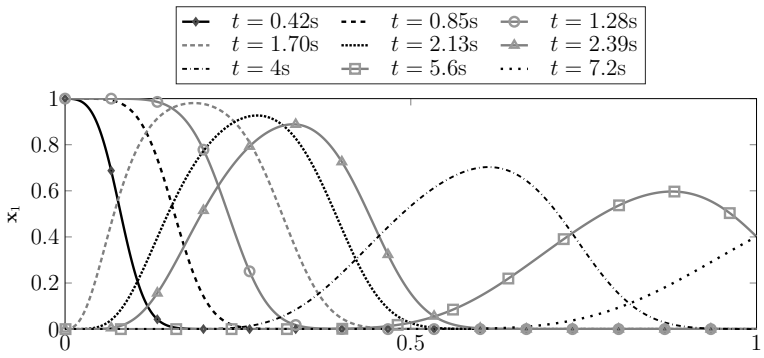
  Among all the e_c surrogate estimators, identify the one with maximal error; find samples $\{\boldsymbol{\mu}^{(1)}, \ldots, \boldsymbol{\mu}^{(n_{\text{add}})}\} \in \Xi_f$ corresponding to largest errors of the identified surrogate $\chi(\boldsymbol{\mu})$.

  Update Ξ_c: $\Xi_c = \left[\Xi_c \cup \{\boldsymbol{\mu}^{(1)}, \ldots, \boldsymbol{\mu}^{(n_{\text{add}})}\}\right]$.

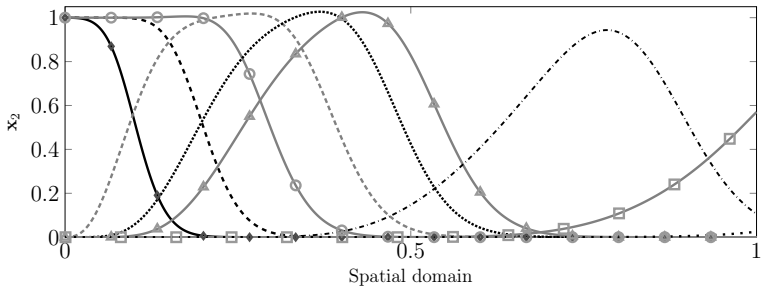
  Find $\boldsymbol{\mu}^* = \arg\max_{\boldsymbol{\mu} \in \Xi_c} \Delta_{\mathbf{y},c}(\boldsymbol{\mu})$. Set $err\_max = \Delta_{\mathbf{y},c}(\boldsymbol{\mu}^*)$.

  Update $\mathbf{r}_{\text{RB}}, \mathbf{r}_{\text{EI}}$ using the subroutine `adapt_basis_update`.

end



(a) Snapshots of component a.



(b) Snapshots of component b.

Figure 6.2.: Snapshots of liquid phase concentrations for components a and b.

Table 6.1.: Parameters for the Batch Chromatography Model.

Parameter	Magnitude	Parameter	Magnitude
Columns length L (cm)	10.5	Diameter D (cm)	2.6
Péclet number Pe	2000	Porosity $\mathring{p}$	0.4
Dimension N^1, N^2	1000	Time (T) (s)	11
Spatial step Δz	0.001	Time step Δt	$<= 8 \cdot 10^{-4}$
H_{a1}	3.728	H_{a2}	0.3
H_{b1}	2.688	H_{b2}	0.1
K_{a1} (ml/g)	46.6	K_{a2} (ml/g)	33.6
K_{b1} (ml/g)	3000	K_{b2} (ml/g)	1000
Mass transfer coeff. κ_a (1/s)	0.1	Mass transfer coeff. κ_b (1/s)	0.1
Feed concentration $\mathring{g}_a^f$ mg/l	$2.9 \cdot 10^{-3}$	Feed concentration $\mathring{g}_b^f$ mg/l	$2.9 \cdot 10^{-3}$
Volumetric feed flow Q (m^3/s)	[0.0667, 0.1167]	Injection time t_{in} (s)	[0.5, 2.0]

The parametric model of the batch chromatography process poses several challenges in its simulation. We list them below:

- To precisely capture the dynamics of the system, the dimension N^1 (or N^2) for each subsystem needs to be large; since the equations are coupled, the FOM is of dimension $4N^1$ making the simulation extremely challenging.
- The simulation time window T is large, in order to allow for flow of the entire

volume of solvent through the column. In addition, the step size Δt needs to be small to satisfy the CFL[1] condition. Moreover, the step size also depends on the parameter Q (the volumetric feed flow); for smaller values of Q, the step size needs to be reduced.

- The model is nonlinear (see Eq. (6.19)) with a rational-type nonlinearity. It is also non-affine in its parameter dependence.
- The Péclet number Pe for the model is usually large and of order $O(10^3)$. This means that the model is convection-dominant. It is well-known (see [44, 149]) that for convection-dominated problems approaches such as POD and RBM face challenges.

Figs. 6.2a and 6.2b show the snapshots of the liquid phase concentration $\mathbf{x}_1, \mathbf{x}_2$ at different time instances. It clearly illustrates the convection-dominated behaviour of the solution.

Our adaptive strategies (basis enrichment and training set sampling) aim at addressing some of the above challenges. To apply a parametric ROM for the batch chromatography model, we apply Algorithm 6.2 to the discretized batch chromatography system Eq. (6.21). To show the considerable benefits offered by the adaptive basis enrichment and adaptive training set sampling, we compare our results against those obtained by using the standard RBMEI algorithm (Algorithm 2.8).

For the numerical simulation of this example we used MATLAB®2020a, on a laptop with INTEL®CORE™i7-8565U @ 1.8 GHZ, with 16 GB of RAM.

Snapshot Selection As previously mentioned, the simulation time T is large while Δt is small. Hence, the number of snapshots collected is also big and it is time consuming to perform SVD on such a large snapshot matrix. To circumvent this, we use the adaptive snapshot selection (AdSS) technique [28]. It is essentially an algorithm to determine the linear dependency of successive vectors in the snapshot matrix. The angle between a new snapshot vector and the last selected vector is evaluated. If it falls below a tolerance it means that the new vector is almost linearly dependent on the previously added snapshot vector and thus can be discarded. For more details on this approach, we refer to [28]. Finally, a much thinner snapshot matrix is obtained, reducing the costs of SVD in either the DEIM or the (Adaptive) RBMEI algorithm.

The parameters used for the simulations are shown in Table 6.1. The system dimension is $N^1 = N^2 = 1000$, where the finite volume method is used for space discretization. The Lax-Friedrichs flux is used for the convection term, while a central difference scheme is applied for the diffusion term. The volumetric feed flow Q and the injection time t_{in} are the two parameters. For the fixed training set Ξ we sample 36 logarithmically distributed parameters from the parameter domain $\mathcal{P} := [0.0067, 0.0167] \times [0.5, 2.0]$. The coarse training set Ξ_c contains 16 logarithmically-spaced parameter samples whereas the fine training set Ξ_f has 100 equally-spaced parameters in the form of a 10×10 grid over $\mathcal{P}$. The test set Ξ_{test} used to validate the ROMs resulting from both algorithms consists of 49 randomly sampled parameters. The simulation time T varies depending

[1] Courant-Friedrich-Levy

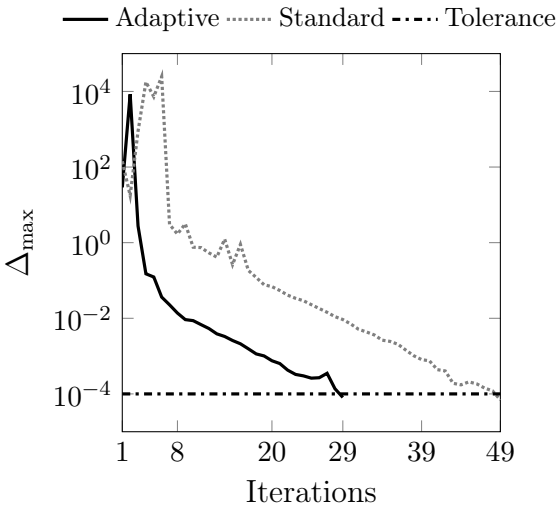
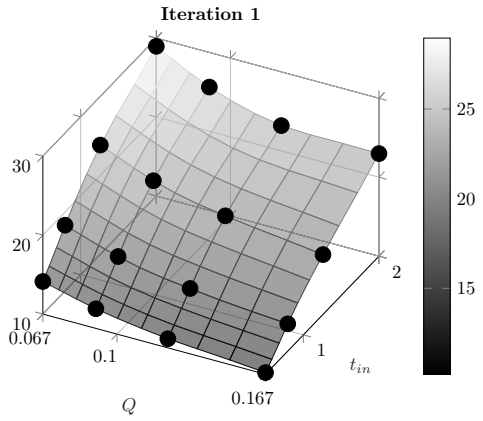


Figure 6.3.: Batch Chromatography: convergence of the greedy algorithms - Algorithm 2.8 vs. Algorithm 6.2.

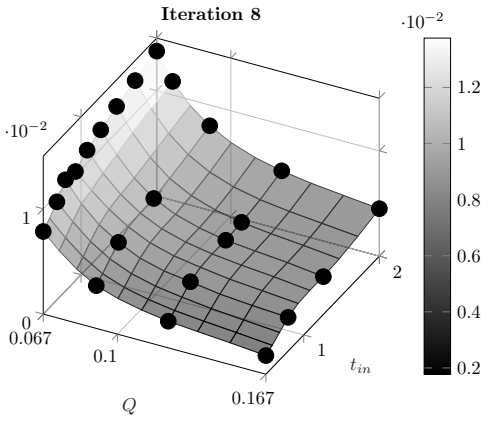
on the choice of Q. The tolerance for the ROM error is set as $\epsilon = 10^{-4}$. We use the DEIM for treating the nonlinear terms. The t-dual system for this example is non-parametric, therefore we use GMRES to compute the approximate dual solutions for the subsystems. The two subsystems whose output are required are denoted by the indices $\alpha_k \in \{1, 2\}$, i.e., the first and second subsystems. Luckily, since the matrices $\overline{\mathbf{E}}, \mathbf{C}$ are the same for these two subsystems, in essence, we need to solve only one dual system.

We apply Algorithm 6.2 to the batch chromatography model. In Figure 6.3, we plot the decay of the quantity $\Delta_{\max}$ (see Eq. (4.38)) at each iteration of the standard RBMEI Algorithm 2.8 and Algorithm 6.2, respectively. The proposed Adaptive RBMEI algorithm for coupled systems with adaptive basis enrichment and training set sampling results in a quicker convergence (29 iterations) as compared to the standard RBMEI algorithm (49 iterations). The dimension of the ROM resulting from application of Algorithm 2.8 is $(r_{\mathrm{RB}}, r_{\mathrm{EI}}) = (49, 99)$ while that from Algorithm 6.2 is $(r_{\mathrm{RB}}, r_{\mathrm{EI}}) = (48, 49)$. This is owing to the simultaneous enrichment of the DEIM projection bases and interpolation points together with the RB projection matrices. In addition to producing a smaller ROM, Algorithm 6.2 results in considerable savings at the offline training time. The overall time taken by Algorithm 2.8 was $2,421$ seconds, whereas only $1,436$ seconds are needed for the proposed Algorithm 6.2 to identify a ROM. The ROMs obtained from both algorithms satisfy the tolerance when evaluated over the samples in the test set Ξ_{test}, however, the benefit of using Algorithm 6.2 is its considerably reduced offline costs.

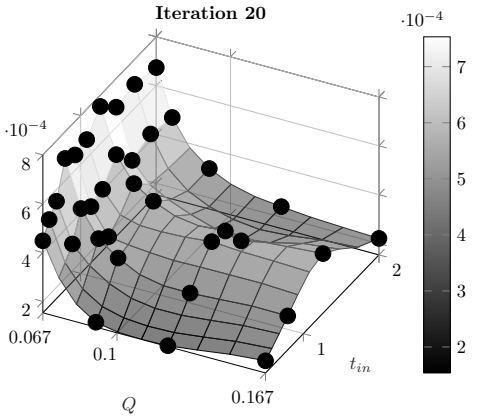
Finally, in Figure 6.4 we illustrate the effect of the adaptive training set sampling scheme. At the first iteration, the initial coarse training set Ξ_c consists of 16 uniformly sampled parameters (represented by the black dots) in Fig. 6.4a. The surface in the figure is the interpolated error over the fine training set Ξ_f obtained via the RBF surrogate. As the algorithm proceeds, newer parameters are added to the coarse training set, as seen in Figs. 6.4b and 6.4c. A large number of parameters are added in the region



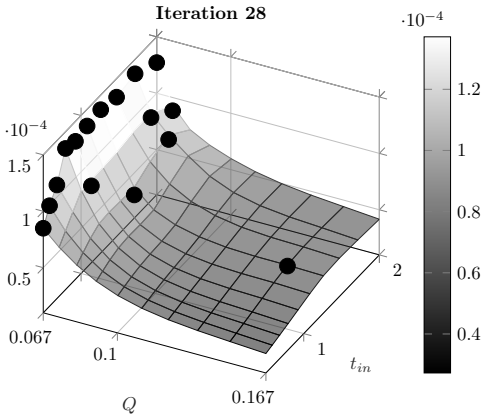
(a) Surrogate Error at Iteration 1.



(b) Surrogate Error at Iteration 8.



(c) Surrogate Error at Iteration 20.



(d) Surrogate Error at Iteration 28.

Figure 6.4.: Coarse training set Ξ_c and the maximum surrogate error $\chi(\boldsymbol{\mu})$ at different iterations for the batch chromatography model; black dots are the samples in Ξ_c.

where Q is small. At the very end, at iteration $iter = 28$ (Fig. 6.4d) it is seen that the algorithm has removed a number of parameters, since at these samples the required tolerance has already been achieved. The remaining parameters are mainly clustered around regions where the parameter Q is small. Thus, we see that the adaptive algorithm is able to adjust itself automatically based on the degree of approximation achieved in different regions of the parameter space $\mathcal{P}$.

6.2. Conclusion

The focus of this chapter is nonlinear coupled systems. We extended the adaptive methods proposed in Chapters 4 and 5 to address the challenges arising in building good ROMs for coupled systems. We used a real-life model of the batch chromatography separation process to show the vastly reduced offline time achieved by our adaptive MOR methods. Not only is the proposed method faster, it is also successful in identifying a ROM of smaller dimension when compared to the standard RBM. The construction of the ROM is almost automatic, with little requests from the user.

CHAPTER 7

CONCLUSIONS

Contents

7.1. Summary

In this thesis, we investigated *a posteriori* error estimation and adaptivity with the goal of automatic model reduction for several types of large-scale systems. The systems covered in this thesis can be broadly classified into linear and nonlinear systems. Within these we considered:

- Linear systems: parametric steady systems, parametric time-harmonic systems, parametric dynamical systems in the input-output representation,
- Nonlinear systems: non-parametric and parametric systems, parametric coupled systems.

To set the stage for the discussions, Chapter 2 reviewed mathematical preliminaries. We formally introduced the setting of the different kinds of systems we consider. This was followed by the introduction of the fundamentals of MOR and the mathematical forms of the ROMs of the different systems of interest. We highlighted the core ideas behind some prominent MOR methods in the frequency- and time-domain. Then we discussed hyperreduction strategies, an important tool to enable efficient ROMs for nonlinear systems. We also briefly looked at radial basis functions which is an important tool for interpolation of scattered data.

In Chapter 3 we focussed on *a posteriori* error estimation and adaptivity for linear systems. We employed frequency-domain MOR techniques to obtain the ROMs. We began by deriving an *inf-sup-constant-free a posteriori* state error estimator. We then proposed a greedy algorithm that uses the *inf-sup-constant-free* state error estimator for efficient ROM construction. Application of the proposed error estimator focused on electromagnetic systems, for which *inf-sup-constant*-based error estimators often result in poor error estimation. The superior performance of our approach over existing ones, in terms of robustness and computational cost is demonstrated via real-life models of an

antenna and a filter. Our contribution regarding *a posteriori* output error estimation is algorithmic. We introduced an efficient procedure for generating a surrogate model of the error estimator using RBF and integrated it as a part of a greedy algorithm to obtain parametric ROMs. The proposed approach is effective for systems with high-dimensional parameter space and systems with large parameter ranges. Tests on three benchmark numerical examples proved the efficiency of this method and the considerable improvement it brings compared to current approaches.

Chapter 4 looked at *a posteriori* error estimation and adaptive basis enrichment for nonlinear dynamical systems. We derived a residual-based *a posteriori* output error estimator by considering a modified output term. Our approach made use of an adjoint or dual system to improve the sharpness of the error estimation and we discussed various aspects involved in its efficient computation. The adaptive basis enrichment scheme aimed to bring down the large offline computational cost usually involved in the RBM and we also aimed to obtain ideally small ROMs. Using the proposed *a posteriori* output error estimator as a 'feedback', we proposed simultaneous enrichment of the reduced basis vectors that approximate the state variable and also the hyperreduction basis vectors that approximate the nonlinear term. The adaptive basis enrichment scheme we introduced is applicable to both non-parametric and parametric systems.

In Chapter 5, we addressed adaptive training set sampling for the RBM. We proposed two approaches: the first involves using a surrogate model of the *a posteriori* error estimator to adaptively add and/or remove samples from a coarse training set; the second approach invokes several state-of-the-art sparse sampling strategies to identify the most important parameter samples from a fine training set. Furthermore, the adaptive training set sampling was successfully combined with the adaptive basis enrichment scheme. Consequently, the RBM for (non)linear parametric dynamical systems is upgraded to an almost automatic version. We tested both training set sampling ideas on different numerical examples to bring out considerable improvements in performance and efficiency they offer for different types of systems.

Finally, in Chapter 6, we extended the previously proposed adaptive strategies to generate efficient ROMs for coupled systems. The proposed techniques are numerically demonstrated using a model of batch chromatography, a method widely used in process engineering. Large computational savings are obtained using our adaptive strategies.

7.2. Outlook

While we have tried to be as comprehensive as possible, this thesis is by no means the last effort on the topics of *a posteriori* error estimation and adaptivity. In fact, there exist a number of promising avenues for future research. We list the ones, which in our opinion, are of most interest:

- We discussed *inf-sup-constant-free a posteriori* error estimation in Chapter 3, in the context of frequency-domain MOR methods. A worthwhile line of research could be the extension of *inf-sup-constant-free* error estimation to the RBM for structured systems such as second-order systems or time-delay systems.

- While a considerable improvement over the state-of-the-art, the *a posteriori* error

estimator proposed in Chapter 4 still has some shortcomings. Firstly, it is derived exclusively for systems whose time discretization uses an implicit-explicit strategy (implicit for the linear term, explicit for the nonlinear term). Future work could be to derive error estimators for systems that involve an implicit discretization of the nonlinear term as well. Secondly, the estimation of the quantity $\tilde{\varrho}$ in Equation (4.27) is more or less heuristic. In our approach, only the FOM solution at the current greedy parameter is used to estimate the value of $\tilde{\varrho}$. It may be worth considering to use all available FOM solutions in order to better estimate the value of $\tilde{\varrho}$.

- We studied *a posteriori* error estimation and adaptivity in Chapter 4 for the RBM. The numerical time stepping technique is assumed to be known and to be the same for the FOM and ROM simulations. A challenge worth addressing would be scenarios where commercial adaptive time step solvers are used to solve the FOM and the particular numerical time integrator used is unknown or can not be easily extracted. For this case, the goal would be to *learn the mismatch* between the residual obtained from snapshots provided by the software and the residual available by imposing a self-defined numerical time integration method. If this mismatch information can be learned, then it is safe to use the snapshots available from the commercial software. One can then derive an error estimator at each time instance of the self-defined numerical time integration scheme by taking into consideration the mismatch information. Moreover, even if a user requests the error at a time t^k that is missing in the defined integration scheme, this can be computed, for e.g., through interpolation.

- The adaptive training set sampling approach in Chapter 5 is aimed at learning the parameter-to-output map through sparse sampling strategies. In many cases, the parameter-to-state map may be of interest. To address this, ideas from tensor theory such as higher-order interpolatory decompositions may be worth pursuing.

- The adaptive MOR of coupled systems was considered in Chapter 6. There, we assumed the coupling to arise exclusively from the nonlinear term. It is not yet clear how adaptive MOR methods proposed here could be applied to systems involving state-coupling or input-output coupling. This is another topic left for future work.

Appendices

APPENDIX A

MODIFIED GRAM-SCHMIDT ORTHOGONALIZATION WITH DEFLATION

A.1. MGS

Algorithm A.1: `MGS_DEF`
Determines an orthogonal basis of range $(\mathbf{D})$ using modified Gram-Schmidt algorithm with deflation.

Input: Matrix $\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_m] \in \mathbb{R}^{N \times m}$, Deflation tolerance ϵ_{def}.
Output: Orthogonal basis $\mathbf{V} := [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{n_{\text{mm}}}] \in \mathbb{R}^{N \times n_{\text{mm}}}$.

Initialization: $\mathbf{v}_1 := \mathbf{d}_1 / \|\mathbf{d}_1\|$, $\mathbf{V} = [\,]$.
for $iter = 2, \ldots, m$ **do**
  $\hat{\mathbf{v}} = \mathbf{d}_i$.
  for $j = 1, 2, \ldots, (i-1)$ **do**
    $h = \hat{\mathbf{v}}^{\mathsf{T}} \mathbf{v}_j$.
    $\hat{\mathbf{v}} = \hat{\mathbf{v}} - h\mathbf{v}_i$.
  end
  if $\|\hat{\mathbf{v}}\| > \epsilon_{\text{def}}$ **then**
    $\mathbf{v}_{iter} = \hat{\mathbf{v}} / \|\hat{\mathbf{v}}\|$, $\mathbf{V} = [\mathbf{V},\ \mathbf{v}_{iter}]$.
  else
    `break`.
  end
end

APPENDIX B

GREEDY ALGORITHMS FOR ADAPTIVE ROM CONSTRUCTION

B.1. Standard State Error Estimator

Algorithm B.1: ROMGreedy-State-Standard

Determines a ROM for Eq. (2.10) using a greedy algorithm with the standard state error estimator.

Input: System matrices: $\mathbf{A}, \mathbf{B}$, Training set Ξ, error tolerance ϵ, Maximum iterations *iter_max*.

Output: Reduced system matrices $\widehat{\mathbf{A}}, \widehat{\mathbf{B}}, \widehat{\mathbf{C}}$.

1 Initialization: $\mathbf{V} = [\,]$, $err\_max = 1 + \epsilon$, $iter = 1$, initial greedy parameter $\breve{\boldsymbol{\mu}}^*$

2 Compute the *inf-sup* constant $\sigma_{\min}(\breve{\boldsymbol{\mu}})$ for all $\breve{\boldsymbol{\mu}} \in \Xi$.

3 **while** $err\_max > \epsilon$ and $iter \leq iter\_max$ **do**

4 Compute matrix $\mathbf{V}_{\breve{\boldsymbol{\mu}}^*}$ using a preferred MOR method applied to the FOM Eq. (2.10); update projection matrix $\mathbf{V}$: $\mathbf{V} = \mathrm{orth}[\mathbf{V},\ \mathbf{V}_{\breve{\boldsymbol{\mu}}^*}]$; if a real $\mathbf{V}$ is preferred set $\mathbf{V} := [\mathrm{real}(\mathbf{V})\ \mathrm{imag}(\mathbf{V})]$.

5 $iter = iter + 1$.

6 Determine reduced matrices $\widehat{\mathbf{A}}, \widehat{\mathbf{B}}$ through Galerkin projection using $\mathbf{V}$ as in Eq. (2.14).

7 Solve the ROM Eq. (2.16) and compute the residual $\mathbf{r}$ for all $\breve{\boldsymbol{\mu}} \in \Xi$.

8 Find $\breve{\boldsymbol{\mu}}^* = \arg\max_{\breve{\boldsymbol{\mu}} \in \Xi} \Delta_{\mathbf{x},\mathrm{std}}$.

9 Set $err\_max := \Delta_{\mathbf{x},\mathrm{std}}(\breve{\boldsymbol{\mu}}^*)$.

10 **end**

B.2. Residual Error Estimator

Algorithm B.2: ROMGreedy-State-Residual

Computes a ROM for Eq. (2.10) using a greedy algorithm with the residual state error estimator.

Input: System matrices: $\mathbf{A}, \mathbf{B}$, Training set Ξ, error tolerance ϵ, Maximum iterations $iter\_max$.

Output: Reduced system matrices $\widehat{\mathbf{A}}, \widehat{\mathbf{B}}, \widehat{\mathbf{C}}$.

1 Initialization: $\mathbf{V} = [\,]$, $err\_max = 1 + \epsilon$, $iter = 1$, initial greedy parameter $\breve{\boldsymbol{\mu}}^*$

2 **while** $err\_max > \epsilon$ and $iter \leq iter\_max$ **do**

3 Compute matrix $\mathbf{V}_{\breve{\boldsymbol{\mu}}^*}$ using a preferred MOR method applied to the FOM Eq. (2.10); update projection matrix $\mathbf{V}$: $\mathbf{V} = \text{orth}[\mathbf{V},\ \mathbf{V}_{\breve{\boldsymbol{\mu}}^*}]$; if a real $\mathbf{V}$ is preferred set $\mathbf{V} := [\text{real}(\mathbf{V})\ \text{imag}(\mathbf{V})]$.

4 $iter = iter + 1$.

5 Determine reduced matrices $\widehat{\mathbf{A}}, \widehat{\mathbf{B}}$ through Galerkin projection using $\mathbf{V}$ as in Eq. (2.14).

6 Solve the ROM Eq. (2.16) and compute the residual $\mathbf{r}$ for all $\breve{\boldsymbol{\mu}} \in \Xi$.

7 Find $\breve{\boldsymbol{\mu}}^* = \arg\max_{\breve{\boldsymbol{\mu}} \in \Xi} \Delta_{\mathbf{x},\text{res}}$.

8 Set $err\_max := \Delta_{\mathbf{x},\text{res}}(\breve{\boldsymbol{\mu}}^*)$.

9 **end**

B.3. Randomized Error Estimator

Algorithm B.3: ROMGreedy-State-Randomized

Determines a ROM for Eq. (2.10) using a greedy algorithm with the randomized state error estimator.

Input: System matrices: $\mathbf{A}, \mathbf{B}$, Training set Ξ, error tolerance ϵ, Maximum iterations $iter\_max$.

Output: Reduced system matrices $\widehat{\mathbf{A}}, \widehat{\mathbf{B}}, \widehat{\mathbf{C}}$.

Initialization: $\mathbf{V} = [\,]$, $err\_max = 1 + \epsilon$, $iter = 1$, initial greedy parameter $\breve{\boldsymbol{\mu}}^*$

Compute the matrix $\mathbf{V}_{\text{rand}}$: $[\mathbf{V}_{\text{rand}}, \mathfrak{Z}] = \texttt{rand\_estm}(\mathbf{A}, \mathbf{B}, N, \Xi, M_s, \epsilon_{\text{rand}})$ with $\mathfrak{Z} := [\mathfrak{z}_1, \ldots, \mathfrak{z}_{M_s}] \in \mathbb{R}^{N \times M_s}$.

while $err\_max > \epsilon$ and $iter \leq iter\_max$ **do**

  Compute matrix $\mathbf{V}_{\breve{\boldsymbol{\mu}}^*}$ using a preferred MOR method applied to the FOM Eq. (2.10); update projection matrix $\mathbf{V}$: $\mathbf{V} = \text{orth}[\mathbf{V},\ \mathbf{V}_{\breve{\boldsymbol{\mu}}^*}]$; if a real $\mathbf{V}$ is preferred set $\mathbf{V} := [\text{real}(\mathbf{V})\ \text{imag}(\mathbf{V})]$.

  $iter = iter + 1$.

  Determine reduced matrices $\widehat{\mathbf{A}}, \widehat{\mathbf{B}}$ through Galerkin projection using $\mathbf{V}$ as in Eq. (2.14).

  Solve the ROM Eq. (2.16) and compute the residual $\mathbf{r}$ for all $\breve{\boldsymbol{\mu}} \in \Xi$.

  Solve the M_s ROMs (Eq. (3.15)) obtained using $\mathbf{V}_{\text{rand}}$ and obtain the corresponding approximate solution vectors $\{\mathfrak{y}_i\}_{i=1}^{M_s}$.

  Find $\breve{\boldsymbol{\mu}}^* = \arg\max_{\breve{\boldsymbol{\mu}} \in \Xi} \Delta_{\mathbf{x},\text{rand}}$, with $\Delta_{\mathbf{x},\text{rand}}$ computed as in Eq. (3.16).

  Set $err\_max := \Delta_{\mathbf{x},\text{rand}}(\breve{\boldsymbol{\mu}}^*)$.

end

Function $[\mathbf{V}_{\text{rand}}, \mathfrak{Z}]$ = `rand_estm`$(\mathbf{A}, \mathbf{B}, N, \Xi, M_s, \epsilon_{\text{rand}})$**:**

  Draw M_s random N-dimensional vectors $[\mathfrak{z}_1, \ldots, \mathfrak{z}_{M_s}]$ from a Gaussian distribution with zero mean and covariance identity.

  Define the augmented training set $\Xi^{\text{aug}} := \{1, 2, \ldots, M_s\} \times \Xi$.

  Initialization: $\mathbf{V}_{\text{rand}} = [\,]$, $err\_max = 1 + \epsilon_{\text{rand}}$, $iter = 1$, initial augmented greedy parameter $\breve{\boldsymbol{\mu}}^*_{aug} = (i^*, \breve{\boldsymbol{\mu}}^*)$

  while $err\_max > \epsilon_{\text{rand}}$ and $iter \leq iter\_max$ **do**

    Solve the FOM random dual system Equation (3.14) for the i^*-th random right hand side vector $\mathfrak{z}_{i^*}$ at the greedy parameter $\breve{\boldsymbol{\mu}}^*_{aug}$ and collect snapshot $\mathfrak{y}_{i^*}$.

    Update random projection matrix: $\mathbf{V}_{\text{rand}} := \text{orth}[\mathbf{V}_{\text{rand}}\ \mathfrak{y}_{i^*}]$. `/* orthogonalize` $\mathfrak{y}_{i^*}$ `against columns of` $\mathbf{V}_{\text{rand}}$ `*/`

    $iter = iter + 1$.

    Determine the reduced matrix $\widehat{\mathbf{A}}_{\text{rand}}$, right hand side vectors $\hat{\mathfrak{z}}_i$ (See Equation (3.15)) through Galerkin projection using $\mathbf{V}_{\text{rand}}$.

    Solve the ROM Eq. (3.15) and compute the residual vector $\mathbf{r}_{\text{rand}} := \mathfrak{z}_i - \mathbf{A}^{\mathsf{T}}\widetilde{\mathfrak{y}}_i$ for all $(i, \breve{\boldsymbol{\mu}}) \in \Xi^{\text{aug}}$.

    Find $\breve{\boldsymbol{\mu}}^*_{aug} = (i^*, \breve{\boldsymbol{\mu}}^*) := \arg\max_{(i,\breve{\boldsymbol{\mu}}) \in \Xi^{aug}} \|\mathbf{r}_{\text{rand}}\|_2$.

    Set $err\_max := \|\mathbf{r}_{\text{rand}}(i^*, \breve{\boldsymbol{\mu}}^*)\|_2$.

  end

APPENDIX C

ALGORITHMIC SKETCH OF DEIM VARIANTS

The algorithms pertaining to the implementation of the DEIM variants, namely, QDEIM, KDEIM and Gappy-POD are sketched below. While the QDEIM algorithm is based on the pseudocode in [69], KDEIM and the two Gappy-POD variants are based on the pseudocode provided in [159].

C.1. QDEIM

Algorithm C.1: QR Discrete Empirical Interpolation Method (QDEIM)

Input: Snapshots of the nonlinear vector $\mathbf{F}$, tolerance ϵ_{EI} (or rank n_{EI}).
Output: $\mathbf{U}$, $\mathbf{S}$ and $\mathcal{I}$.

1 Find the POD basis: $\mathbf{U} = \texttt{POD}(\mathbf{F}, n_{\text{EI}}(\text{ or } \epsilon_{\text{EI}}))$.
2 $[\sim, \sim, \mathcal{I}] = \texttt{qr}(\mathbf{U}^{\mathsf{T}}, \texttt{'vector'})$, with $\mathcal{I} = [\wp_1, \wp_2, \ldots, \wp_N]$.
3 Set $\mathcal{I} = \mathcal{I}(1 : n_{\text{EI}})$ and $\mathbf{S} = [e_{\wp_1}, \ldots, e_{\wp_{n_{\text{EI}}}}]$.

C.2. KDEIM

Algorithm C.2: `kmeans`-Discrete Empirical Interpolation Method (KDEIM)

Input: Snapshots of the nonlinear vector ($\mathbf{F}$), tolerance ϵ_{EI} (or rank n_{EI}).
Output: $\mathbf{U}$, $\mathbf{S}$ and $\mathcal{I}$.

1 Find the POD basis: $\mathbf{U} = \texttt{POD}(\mathbf{F}, n_{\text{EI}}(\text{ or } \epsilon_{\text{EI}}))$.
2 Set $[\sim, n_{\text{EI}}] = \texttt{size}(\mathbf{U})$ and $p = \texttt{zeros}(n_{\text{EI}}, 1)$.
3 Apply `k-means` algorithm and identify the clusters: $[\mathbf{Idx}, \mathbf{K}] = \texttt{kmeans}(\mathbf{U}, n_{\text{EI}})$.
4 **for** $i = 1, \ldots, n_{\text{EI}}$ **do**
5 $\quad J = \texttt{find}(\mathbf{Idx} == i)$; $j = \texttt{length}(J)$.
6 $\quad \mathbf{Z} = \mathbf{U}(J\,,:) - \texttt{ones}(j, 1)\,\mathbf{K}(i\,,:)$.
7 $\quad$ Define $s1 = \texttt{sum}(\mathbf{Z}. * \mathbf{Z}, 2)$ and $s2 = \texttt{sum}((\mathbf{U}(J\,,:).\hat{}2), 2)$.
8 $\quad [\sim, c] = \texttt{min}(s1./s2)$.
9 $\quad \wp_i = J(c(1))$.
10 **end**
11 Set $\mathcal{I} = [\wp_1, \ldots, \wp_{n_{\text{EI}}}]$ and $\mathbf{S} = [e_{\wp_1}, \ldots, e_{\wp_{n_{\text{EI}}}}]$.

C.3. Gappy-POD Eigenvector

Algorithm C.3: Gappy-POD Eigenvector

Input: Snapshots of the nonlinear vector $\mathbf{F}$, number of additional samples m, tolerance for the SVD ϵ_{EI} (or rank n_{EI}).

Output: $\mathbf{U}$, $\mathbf{S}$ and $\mathcal{I}$.

Find the POD basis: $\mathbf{U} = \texttt{POD}(\mathbf{F}, n_{\text{EI}}(\text{ or } \epsilon_{\text{EI}}))$.
$[\sim, \sim, \mathcal{I}] = \texttt{qr}(\mathbf{U}^{\mathsf{T}}, \texttt{'vector'})$, with $\mathcal{I} = [\wp_1, \wp_2, \ldots, \wp_N]$.
Set $\mathcal{I} = \mathcal{I}(1 : n_{\text{EI}})$.
for $i = n_{\text{EI}} + 1, \ldots, m$ **do**
    $[\sim, \mathbf{S}, \mathbf{W}] = \texttt{svd}(\mathbf{U}(p\,, :), 0)$.
    $g = \mathbf{S}(\text{end-1}\,, \text{end-1}).\hat{}2 - \mathbf{S}(\text{end}\,, \text{end})\hat{}2$.
    $\mathbf{U}_b = \mathbf{W}^{\mathsf{T}} * \mathbf{U}^{\mathsf{T}}$.
    $r = g + \texttt{sum}(\mathbf{U}_b.\hat{}2\,, 1)$.
    $r = r - \texttt{sqrt}\,((r).\hat{}2 - 4 * g * \mathbf{U}_b(\text{end}\,, :).\hat{}2)$.
    $[\sim, \mathbf{Q}] = \texttt{sort}(r\,, \texttt{'descend'})$.
    $e = 1$.
    while any $(\mathbf{Q}(e) == p)$ **do**
        $e = e + 1$.
    end
    $\mathcal{I}(\text{end} + 1) = \mathbf{Q}(e)$.
end
Set $\mathbf{S} = [e_{\wp_1}, \ldots, e_{\wp_m}]$.

C.4. Gappy-POD Clustering

Algorithm C.4: Gappy-POD Clustering

Input: Snapshots of the nonlinear vector ($\mathbf{F}$), number of additional samples m, tolerance for the SVD ϵ_{EI} (or rank n_{EI}).

Output: $\mathbf{U}$, $\mathbf{S}$ and $\mathcal{I}$.

Find the POD basis: $\mathbf{U} = \texttt{POD}(\mathbf{F}, n_{\text{EI}}(\text{ or } \epsilon_{\text{EI}}))$.
$[\sim, \mathbf{R}, \mathcal{I}] = \texttt{qr}(\mathbf{U}^{\mathsf{T}}, \texttt{'vector'})$, with $\mathcal{I} = [\wp_1, \wp_2, \ldots, \wp_N]$.
$[N, n_{\text{EI}}] = \texttt{size}(\mathbf{U})$.
Set $\phi = \mathcal{I}(1 : n_{\text{EI}})$.
$\widetilde{\mathbf{R}} = \mathbf{R}(1 : n_{\text{EI}}, 1 : n_{\text{EI}}) \backslash \mathbf{R}$; set $s = \texttt{sum}(\texttt{abs}(\widetilde{\mathbf{R}}), 1)$.
$\widetilde{\mathbf{R}} = \texttt{abs}(\widetilde{\mathbf{R}}) ./ \texttt{repmat}(s\,,\, \texttt{size}(\widetilde{\mathbf{R}}\,,\, 1)\,,\, 1)$.
$e = \texttt{zeros}(N\,,\, 1)$.
for $i = 1, \ldots, N$ **do**
    $K = \widetilde{\mathbf{R}}(:\,,\, i) > 0$.
    $e(i) = -\widetilde{\mathbf{R}}(K\,,\, i)^{\mathsf{T}} \log(\widetilde{\mathbf{R}}(K\,,\, i))$.
end
$[\sim, K] = \texttt{sort}(e, \texttt{'descend'})$.
Set $i = 1$.
while $\texttt{length}(\phi) < m$ **do**
    $\phi = \texttt{unique}([\phi\,,\, \mathcal{I}(K(i))])$.
    $i = i + 1$.
end
Set $\mathcal{I} = \phi$, where $\phi = [\wp_1, \ldots, \wp_m]$.
Set $\mathbf{S} = [e_{\wp_1}, \ldots, e_{\wp_m}]$.

BIBLIOGRAPHY

[1] T. Aanonsen, *Empirical interpolation with application to reduced basis approximations*, master's thesis, Master Thesis, Norwegian University of Science and Technology (NTNU), Trondheim, Norway, 2009, http://hdl.handle.net/11250/258487. 127

[2] R. Achar and M. Nakhla, *Simulation of high-speed interconnects*, Proceedings of the IEEE, 89 (2001), pp. 693–728, https://doi.org/10.1109/5.929650. 73

[3] M. Ali, K. Steih, and K. Urban, *Reduced basis methods with adaptive snapshot computations*, Adv. Comput. Math., 43 (2017), pp. 257–294, https://doi.org/10.1007/s10444-016-9485-9. 84

[4] A. Alla and J. N. Kutz, *Randomized model order reduction*, Adv. Comput. Math., 45 (2019), pp. 1251–1271, https://doi.org/10.1007/s10444-018-09655-9. 26

[5] B. O. Almroth, P. Stern, and F. A. Brogan, *Automatic choice of global shape functions in structural analysis*, AIAA J., 16 (1978), pp. 525–528, https://doi.org/10.2514/3.7539. 27

[6] F. Alsayyari, Z. Perkó, D. Lathouwers, and J. L. Kloosterman, *A nonintrusive reduced order modelling approach using Proper Orthogonal Decomposition and locally adaptive sparse grids*, J. Comput. Phys., 399 (2019), p. 108912, https://doi.org/10.1016/j.jcp.2019.108912. 27

[7] D. Amsallem and C. Farhat, *Stabilization of projection-based reduced-order models*, Internat. J. Numer. Methods Engrg., 91 (2012), pp. 358–377, https://doi.org/10.1002/nme.4274. 19

[8] D. Amsallem, M. J. Zahr, and C. Farhat, *Nonlinear model order reduction based on local reduced-order bases*, Internat. J. Numer. Methods Engrg., 92 (2012), pp. 891–916, https://doi.org/10.1002/nme.4371. 4, 5

[9] D. Amsallem, M. J. Zahr, and K. Washabaugh, *Fast local reduced basis updates for the efficient reduction of nonlinear systems with hyper-reduction*, Adv. Comput. Math., 41 (2015), pp. 1187–1230, https://doi.org/10.1007/s10444-015-9409-0. 4, 5

[10] H. Antil, D. Chen, and S. Field, *A note on qr-based model reduction: Algorithm, software, and gravitational wave applications*, Comput. Sci. Eng., 20 (2018), pp. 10–25, https://doi.org/10.1109/MCSE.2018.042781323. 125

[11] A. C. Antoulas, *Approximation of Large-Scale Dynamical Systems*, vol. 6 of Adv. Des. Control, SIAM Publications, Philadelphia, PA, 2005, https://doi.org/10.1137/1.9780898718713. 2, 3, 19, 20, 42

[12] A. C. Antoulas, C. A. Beattie, and S. Gugercin, *Interpolatory Methods for Model Reduction*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2020, https://doi.org/10.1137/1.9781611976083. 2, 3, 62

[13] A. C. Antoulas, P. Benner, and L. Feng, *Model reduction by iterative error system approximation*, Math. Comput. Model. Dyn. Syst., 24 (2018), pp. 103–118, https://doi.org/10.1080/13873954.2018.1427116. 61

[14] A. C. Antoulas, S. Lefteriu, and A. C. Ionita, *Model Reduction and Approximation: Theory and Algorithms*, vol. 15 of Computational Science & Engineering, SIAM Publications, Philadelphia, PA, 2017, ch. 8: A Tutorial Introduction to the Loewner Framework for Model Reduction, pp. 335–376, https://doi.org/10.1137/1.9781611974829.ch8. 2, 17

[15] S. R. Arridge, J. P. Kaipio, V. Kolehmainen, M. Schweiger, E. Somersalo, T. Tarvainen, and M. Vauhkonen, *Approximation errors and model reduction with an application in optical diffusion tomography*, Inverse Problems, 22 (2006), pp. 175–195, https://doi.org/10.1088/0266-5611/22/1/010. 17

[16] U. M. Ascher, S. J. Ruuth, and B. T. R. Wetton, *Implicit-explicit methods for time-dependent partial differential equations*, SIAM J. Numer. Anal., 32 (1995), pp. 797–823, https://doi.org/10.1137/0732037. 16

[17] P. Astrid, S. Weiland, K. Willcox, and T. Backx, *Missing point estimation in models described by proper orthogonal decomposition*, IEEE Trans. Autom. Control, 53 (2008), pp. 2237–2251, https://doi.org/10.1109/TAC.2008.2006102. 31

[18] H. T. Banks and K. Kunisch, *Estimation Techniques for Distributed Parameter Systems*, vol. 6 of Systems & Control: Foundations & Applications, Birkhäuser, Boston, 1989, https://doi.org/10.1007/978-1-4612-3700-6. 118

[19] 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*, C.R. Acad. Sci. Paris, 339 (2004), pp. 667–672, https://doi.org/10.1016/j.crma.2004.08.006. 29, 31, 32, 33, 35, 95

[20] U. Baur, C. A. Beattie, P. Benner, and S. Gugercin, *Interpolatory projection methods for parameterized model reduction*, SIAM J. Sci. Comput., 33 (2011), pp. 2489–2518, https://doi.org/10.1137/090776925. 43

[21] U. Baur, P. Benner, and L. Feng, *Model order reduction for linear and nonlinear systems: A system-theoretic perspective*, Arch. Comput. Methods Eng., 21 (2014), pp. 331–358, https://doi.org/10.1007/s11831-014-9111-2. 18

[22] C. Beattie, S. Gugercin, and V. Mehrmann, *Model reduction for systems with inhomogeneous initial conditions*, Syst. Control Lett., 99 (2017), pp. 99–106, https://doi.org/10.1016/j.sysconle.2016.11.007. 18

[23] T. Bechtold, E. Rudnyi, and J. Korvink, *Error indicators for fully automatic extraction of heat-transfer macromodels for MEMS*, Micromech. Microeng., 15 (2004), pp. 430–440, https://doi.org/10.1088/0960-1317/15/3/002. 43, 48, 61

[24] A. Benaceur, V. Ehrlacher, A. Ern, and S. Meunier, *A progressive reduced basis/empirical interpolation method for nonlinear parabolic problems*, SIAM J. Sci. Comput., 40 (2018), pp. A2930–A2955, https://doi.org/10.1137/17M1149638. 30, 84, 93

[25] P. Benner and T. Breiten, *Interpolation-based $\mathcal{H}_2$-model reduction of bilinear control systems*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 859–885, https://doi.org/10.1137/110836742. 22

[26] P. Benner, T. Breiten, H. Fassbender, M. Hinze, T. Stykel, and R. Zimmermann, eds., *Model Reduction of Complex Dynamical Systems*, vol. 171 of International Series of Numerical Mathematics, Springer Nature, 2021, https://doi.org/10.1007/978-3-030-72983-7. 2

[27] P. Benner and L. Feng, *Model order reduction for coupled problems*, Applied and Computational Mathematics: An International journal, 14 (2015), pp. 3–22. 148, 149

[28] P. Benner, L. Feng, S. Li, and Y. Zhang, *Reduced-order modeling and ROM-based optimization of batch chromatography*, in Numerical Mathematics and Advanced Applications - ENUMATH 2013, Springer International Publishing, 2015, pp. 427–435, https://doi.org/10.1007/978-3-319-10705-9_42. 84, 159

[29] P. Benner and P. Goyal, *Balanced truncation model order reduction for quadratic-bilinear systems*, e-prints 1705.00160, arXiv, 2017, https://arxiv.org/abs/1705.00160. math.OC. 19

[30] P. Benner, P. Goyal, and S. Gugercin, *$\mathcal{H}_2$-quasi-optimal model order reduction for quadratic-bilinear control systems*, SIAM J. Matrix Anal. Appl., 39 (2018), pp. 983–1032, https://doi.org/10.1137/16M1098280. 22

[31] P. Benner, P. Goyal, B. Kramer, B. Peherstorfer, and K. Willcox, *Operator inference for non-intrusive model reduction of systems with non-polynomial nonlinear terms*, Comp. Meth. Appl. Mech. Eng., 372 (2020), p. 113433, https://doi.org/10.1016/j.cma.2020.113433. 17, 24

[32] P. Benner, S. Grivet-Talocia, A. Quarteroni, G. Rozza, and L. M. Schilder, W. Silveira, eds., *Model Order Reduction. Volume 1: System- and Data-Driven Methods and Algorithms*, De Gruyter, 2021, https://doi.org/10.1515/9783110499001. 2, 3

[33] P. Benner, S. Grivet-Talocia, A. Quarteroni, G. Rozza, and L. M. Schilder, W. Silveira, eds., *Model Order Reduction. Volume 2: Snapshot-Based Methods and Algorithms*, De Gruyter, 2021, https://doi.org/10.1515/9783110671490. 2, 3

[34] P. Benner, S. Grivet-Talocia, A. Quarteroni, G. Rozza, and L. M. Schilder, W. Silveira, eds., *Model Order Reduction. Volume 3: Applications*, De Gruyter, 2021, https://doi.org/10.1515/9783110499001. 2

[35] P. Benner, S. Gugercin, and K. Willcox, *A survey of projection-based model reduction methods for parametric dynamical systems*, SIAM Rev., 57 (2015), pp. 483–531, https://doi.org/10.1137/130932715. 3, 16, 22, 42

[36] P. Benner, M. Ohlberger, A. Cohen, and K. Willcox, eds., *Model Reduction and Approximation: Theory and Algorithms*, Computational Science & Engineering, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2017, https://doi.org/10.1137/1.9781611974829. 2

[37] P. Binev, A. Cohen, W. Dahmen, R. DeVore, G. Petrova, and P. Wojtaszczyk, *Convergence rates for greedy algorithms in reduced basis methods*, SIAM J. Math. Anal., 43 (2011), pp. 1457–1472, https://doi.org/10.1137/100795772. 28

[38] A. Bodendiek and M. Bollhöfer, *Adaptive-order rational Arnoldi-type methods in computational electromagnetism*, BIT, 54 (2014), pp. 357–380, https://doi.org/10.1007/s10543-013-0458-9. 43

[39] B. Bond and L. Daniel, *Parameterized model order reduction of nonlinear dynamical systems*, in Proceedings of the 2005 IEEE/ACM International Conference on Computer-Aided Design, ICCAD '05, USA, 2005, IEEE Computer Society, p. 487–494. 22

[40] C. Boutsidis, M. W. Mahoney, and P. Drineas, *An improved approximation algorithm for the column subset selection problem*, in Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, Philadelphia, PA, 2009, pp. 968–977. 129

[41] T. Braconnier, M. Ferrier, J.-C. Jouhaud, M. Montagnac, and P. Sagaut, *Towards an adaptive POD/SVD surrogate model for aeronautic design*, Comput. & Fluids, 40 (2011), pp. 195–209, https://doi.org/10.1016/j.compfluid.2010.09.002. 27

[42] M. E. Broadbent, M. Brown, and K. Penner, *Subset selection algorithms: Randomized vs. deterministic*, SIAM Undergrad. Res. Online, 3 (2010), pp. 50–71, https://doi.org/https://dx.doi.org/10.1137/09S010435. 129

[43] M. D. Buhmann, *Radial Basis Functions: Theory and Implementations*, vol. 12 of Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, Cambridge, 2003. 37, 39

[44] N. Cagniart, Y. Maday, and B. Stamm, *Model order reduction for problems with large convection effects*, in Contributions to Partial Differential Equations and Applications, vol. 47 of Comput. Methods Appl. Sci., Springer, Cham, 2019, pp. 131–150. 159

[45] C. Canuto, T. Tonn, and K. Urban, *A posteriori error analysis of the reduced basis method for nonaffine parametrized nonlinear PDEs*, SIAM J. Numer. Anal., 47 (2009), pp. 2001–2022, https://doi.org/10.1137/080724812. 97

[46] K. Carlberg, *Adaptive h-refinement for reduced-order models*, Internat. J. Numer. Methods Engrg., 102 (2015), pp. 1192–1210, https://doi.org/10.1002/nme.4800. 4, 5

[47] K. Carlberg, D. Amsallem, P. Avery, M. Zahr, and C. Farhat, *The GNAT nonlinear model reduction method and its application to fluid dynamics problems*, 6th AIAA Theoretical Fluid Mechanics Conference, Honolulu, (2011), pp. 1–24, https://doi.org/10.2514/6.2011-3112. 123

[48] K. Carlberg, C. Bou-Mosleh, and C. Farhat, *Efficient non-linear model reduction via a least-squares Petrov-Galerkin projection and compressive tensor approximations*, Internat. J. Numer. Methods Engrg., 86 (2011), pp. 155–181, https://doi.org/10.1002/nme.3050. 123

[49] A. Çivril and M. Magdon-Ismail, *Column subset selection via sparse approximation of SVD*, Theoret. Comput. Sci., 421 (2012), pp. 1–14, https://doi.org/10.1016/j.tcs.2011.11.019. 129

[50] Y. Chahlaoui, *A posteriori error bounds for discrete balanced truncation*, Linear Algebra Appl., 436 (2012), pp. 2744–2763, https://doi.org/10.1016/j.laa.2011.07.025. 43

[51] T. F. Chan, *Rank revealing QR factorizations*, Linear Algebra Appl., 88/89 (1987), pp. 67–82, https://doi.org/10.1016/0024-3795(87)90103-0. 135

[52] S. Chaturantabut and D. C. Sorensen, *Nonlinear model reduction via discrete empirical interpolation*, SIAM J. Sci. Comput., 32 (2010), pp. 2737–2764, https://doi.org/10.1137/090766498. 29, 31, 32, 33, 35

[53] S. Chellappa, L. Feng, and P. Benner, *Adaptive basis construction and improved error estimation for parametric nonlinear dynamical systems*, Internat. J. Numer. Methods Engrg., 121 (2020), pp. 5320–5349, https://doi.org/10.1002/nme.6462. iii, 84, 200

[54] S. Chellappa, L. Feng, and P. Benner, *A training set subsampling strategy for the reduced basis method*, J. Sci. Comput., 89 (2021), pp. 1–34, https://doi.org/10.1007/s10915-021-01665-y. iii, 113, 200

[55] S. Chellappa, L. Feng, and P. Benner, *An adaptive sampling approach for the reduced basis method*, in Realization and Model Reduction of Dynamical Systems - A Festschrift in Honor of the 70th Birthday of Thanos Antoulas, Springer,

Cham, 2022, pp. 137–155, https://doi.org/10.1007/978-3-030-95157-3_8. iii, 113, 199, 200

[56] S. Chellappa, L. Feng, V. de la Rubia, and P. Benner, *Adaptive interpolatory MOR by learning the error estimator in the parameter domain*, in Model Reduction of Complex Dynamical Systems, vol. 171 of International Series of Numerical Mathematics, Birkhäuser, Cham, 2021, pp. 97–117, https://doi.org/10.1007/978-3-030-72983-7_5. iii, 42, 199

[57] S. Chellappa, L. Feng, V. de la Rubia, and P. Benner, *Inf-sup-constant-free state error estimator for model order reduction of parametric systems in electromagnetics*, e-prints 2104.12802, arXiv, 2021, https://arxiv.org/abs/2104.12802. math.NA. iii, 42, 199

[58] K. K. Chen, J. H. Tu, and C. W. Rowley, *Variants of dynamic mode decomposition: boundary condition, Koopman, and Fourier analyses*, J. Nonlinear Sci., 22 (2012), pp. 887–915, https://doi.org/10.1007/s00332-012-9130-9. 24

[59] P. Chen and O. Ghattas, *Hessian-based sampling for high-dimensional model reduction*, Int. J. Uncertain. Quantif., 9 (2019), pp. 103–121, https://doi.org/10.1615/Int.J.UncertaintyQuantification.2019028753. 112

[60] P. Chen, A. Quarteroni, and G. Rozza, *Reduced basis methods for uncertainty quantification*, SIAM/ASA J. Uncertain. Quantif., 5 (2017), pp. 813–869, https://doi.org/10.1137/151004550. 27

[61] Y. Choi, D. Coombs, and R. Anderson, *SNS: A solution-based nonlinear subspace method for time-dependent model order reduction*, SIAM J. Sci. Comput., 42 (2020), pp. A1116–A1146, https://doi.org/10.1137/19M1242963. 4, 5, 84

[62] P. G. Constantine, E. Dow, and Q. Wang, *Active subspace methods in theory and practice: applications to kriging surfaces*, SIAM J. Sci. Comput., 36 (2014), pp. A1500–A1524, https://doi.org/10.1137/130916138. 113, 129

[63] R. Crisovan, D. Torlo, R. Abgrall, and S. Tokareva, *Model order reduction for parametrized nonlinear hyperbolic problems as an application to uncertainty quantification*, Journal of Computational and Applied Mathematics, 348 (2019), pp. 466–489, https://doi.org/https://doi.org/10.1016/j.cam.2018.09.018. 27

[64] L. Daniel, O. C. Siong, L. S. Chay, K. H. Lee, and J. White, *A multiparameter moment-matching model-reduction approach for generating geometrically parameterized interconnect performance models*, Trans. Comp.-Aided Des. Integ. Cir. Sys., 23 (2006), p. 678–693, https://doi.org/10.1109/TCAD.2004.826583. 22, 43

[65] C. Daversin and C. Prud'homme, *Simultaneous empirical interpolation and reduced basis method for non-linear problems*, C.R. Acad. Sci. Paris, 353 (2015), pp. 1105–1109, https://doi.org/10.1016/j.crma.2015.08.003. 84, 93

[66] V. de la Rubia, *Reliable reduced-order model for fast frequency sweep in microwave circuits*, Electromagnetics, 34 (2014), pp. 161–170, https://doi.org/10.1080/02726343.2014.877735. 54

[67] V. de la Rubia and M. Mrozowski, *A compact basis for reliable fast frequency sweep via the reduced-basis method*, IEEE Trans. Microwave Theory Tech., 66 (2018), pp. 4367–4382, https://doi.org/10.1109/TMTT.2018.2865957. 54

[68] V. de la Rubia and J. Zapata, *Microwave circuit design by means of direct decomposition in the finite-element method*, IEEE Trans. Microwave Theory Tech., 55 (2007), pp. 1520–1530, https://doi.org/10.1109/TMTT.2007.900307. 54

[69] Z. Drmač and S. Gugercin, *A new selection operator for the discrete empirical interpolation method—improved a priori error bound and extensions*, SIAM J. Sci. Comput., 38 (2016), pp. A631–A648, https://doi.org/10.1137/15M1019271. 123, 175

[70] M. Drohmann and K. Carlberg, *The ROMES method for statistical modeling of reduced-order-model error*, SIAM/ASA Journal on Uncertainty Quantification, 3 (2015), pp. 116–145, https://doi.org/10.1137/140969841. 69

[71] M. Drohmann, B. Haasdonk, and M. Ohlberger, *Reduced basis approximation for nonlinear parametrized evolution equations based on empirical operator interpolation*, SIAM J. Sci. Comput., 34 (2012), pp. A937–A969, https://doi.org/10.1137/10081157X. 4, 29, 30, 31, 84, 93, 94, 95, 97

[72] V. Druskin, V. Simoncini, and M. Zaslavsky, *Adaptive tangential interpolation in rational Krylov subspaces for MIMO dynamical systems*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 476–498, https://doi.org/10.1137/120898784. 69

[73] J. A. Duersch and M. Gu, *Randomized QR with column pivoting*, SIAM J. Sci. Comput., 39 (2017), pp. C263–C291, https://doi.org/10.1137/15M1044680. 132

[74] J. L. Eftang, M. A. Grepl, and A. T. Patera, *A posteriori error bounds for the empirical interpolation method*, C. R. Math. Acad. Sci. Paris, 348 (2010), pp. 575–579, https://doi.org/10.1016/j.crma.2010.03.004. 35

[75] J. L. Eftang, D. J. Knezevic, and A. T. Patera, *An hp certified reduced basis method for parametrized parabolic partial differential equations*, Math. Comput. Model. Dyn. Syst., 17 (2011), pp. 395–422, https://doi.org/10.1080/13873954.2011.547670. 27, 112, 113

[76] J. L. Eftang, A. T. Patera, and E. M. Rønquist, *An "hp" certified reduced basis method for parametrized elliptic partial differential equations*, SIAM J. Sci. Comput., 32 (2010), pp. 3170–3200, https://doi.org/10.1137/090780122. 31, 84, 112

[77] R. Everson and L. Sirovich, *Karhunen–Loève procedure for gappy data*, J. Opt. Soc. Am. A, 12 (1995), pp. 1657–1664. 2, 31, 123

[78] C. Farhat, S. Grimberg, A. Manzoni, and A. Quarteroni, *5 Computational Bottlenecks for PROMs: Precomputation and Hyperreduction*, in Model Order Reduction. Volume 2: Snapshot-Based Methods and Algorithms, P. Benner, S. Grivet-Talocia, A. Quarteroni, G. Rozza, W. Schilders, and L. M. Silveira, eds., De Gruyter, 2020, pp. 181–244, https://doi.org/doi:10.1515/9783110671490-005. 31

[79] H. Fassbender and P. Benner, *Passivity preserving model reduction via a structured Lanczos method*, in 2006 IEEE Conference on Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control, 2006, pp. 8–13, https://doi.org/10.1109/CACSD-CCA-ISIC.2006.4776616. 22

[80] G. Fasshauer and M. McCourt, *Kernel-based Approximation Methods using MATLAB*, vol. 19 of Interdisciplinary Mathematical Sciences, World Scientific, 2015. 17, 37, 38, 39

[81] L. Feng, A. C. Antoulas, and P. Benner, *Some a posteriori error bounds for reduced order modelling of (non-)parametrized linear systems*, ESAIM: M2AN, 51 (2017), pp. 2127 – 2158, https://doi.org/10.1051/m2an/2017014. 4, 22, 43, 62, 63

[82] L. Feng and P. Benner, *Reduced Order Methods for modeling and computational reduction, MS&A Series*, vol. 9 of MS & A, Springer-Verlag, Berlin, Heidelberg, New York, 2014, ch. 6: A robust algorithm for parametric model order reduction based on implicit moment matching, pp. 159–186, https://doi.org/10.1007/978-3-319-02090-7_6. 3, 22, 23, 24, 43, 51, 61

[83] L. Feng and P. Benner, *A new error estimator for reduced-order modeling of linear parametric systems*, IEEE Trans. Microw. Theory Techn., 67 (2019), pp. 4848–4859, https://doi.org/10.1109/TMTT.2019.2948858. 4, 43, 62, 63, 64, 67, 69, 74, 199

[84] L. Feng and P. Benner, *Model order reduction based on moment-matching*, in Model Order Reduction. Volume 1: System- and Data-Driven Methods and Algorithms, P. Benner, S. Grivet-Talocia, A. Quarteroni, G. Rozza, W. H. A. Schilders, and L. M. Silveira, eds., De Gruyter, 2021, pp. 57–96, https://doi.org/10.1515/9783110498967-003. 22

[85] L. Feng and P. Benner, *On error estimation for reduced-order modeling of linear non-parametric and parametric systems*, ESAIM: Math. Model. Numer. Anal., 55 (2021), pp. 561–594, https://doi.org/10.1051/m2an/2021001. 4, 22, 43, 61, 62, 64, 65, 67, 69

[86] L. Feng, J. G. Korvink, and P. Benner, *A fully adaptive scheme for model order reduction based on moment-matching*, IEEE Transactions on Components, Packaging and Manufacturing Technology, 5 (2015), pp. 1872–1884, https://doi.org/10.1109/TCPMT.2015.2491341. 43, 61, 62, 73

[87] L. Feng, M. Mangold, and P. Benner, *Adaptive POD-DEIM basis construction and its application to a nonlinear population balance system*, AIChE J., 63 (2017), pp. 3832–3844, https://doi.org/10.1002/aic.15749. 4, 5, 27, 92, 97, 199

[88] R. L. Fox and H. Miura, *An approximate analysis technique for design calculations*, AIAA J., 9 (1971), pp. 177–179, https://doi.org/10.2514/3.6141. 27

[89] S. Fresca, L. Dedé, and A. Manzoni, *A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized PDEs*, J. Sci. Comput., 87 (2021), https://doi.org/10.1007/s10915-021-01462-7. 3

[90] K. Gallivan, A. Vandendorpe, and P. Van Dooren, *Model reduction of MIMO systems via tangential interpolation*, SIAM J. Matrix Anal. Appl., 26 (2004), pp. 328–349, https://doi.org/10.1137/S0895479803423925. 69

[91] S. García, V. de la Rubia, and M. Mrozowski, *Reduced basis approximations in microwave filters and diplexers: Inf-sup constant behavior*, in 2017 IEEE MTT-S International Conference on Numerical Electromagnetic and Multiphysics Modeling and Optimization for RF, Microwave, and Terahertz Applications (NEMO), IEEE, 2017, pp. 275–277, https://doi.org/10.1109/NEMO.2017.7964258. 3

[92] C. Geuzaine and J.-F. Remacle, *Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities*, Int. J. Numer. Methods Eng., 79 (2009), pp. 1309–1331, https://doi.org/10.1002/nme.2579. 52, 137

[93] F. Ghavamian, P. Tiso, and A. Simone, *POD-DEIM model order reduction for strain-softening viscoplasticity*, Comput. Methods Appl. Mech. Engrg., 317 (2017), pp. 458–479, https://doi.org/10.1016/j.cma.2016.11.025. 92

[94] S. Glas, A. T. Patera, and K. Urban, *A reduced basis method for the wave equation*, International Journal of Computational Fluid Dynamics, 34 (2020), pp. 139–146, https://doi.org/10.1080/10618562.2019.1686486. 27

[95] F. P. Gonzalez and M. Balajewicz, *Deep convolutional recurrent autoencoders for learning low-dimensional feature dynamics of fluid systems*, e-prints 1808.01346, arXiv, 2018, https://arxiv.org/abs/1808.01346. math.DS. 3

[96] C. Grässle, M. Hinze, J. Lang, and S. Ullmann, *POD model order reduction with space-adapted snapshots for incompressible flows*, Adv. Comput. Math., 45 (2019), pp. 2401–2428, https://doi.org/10.1007/s10444-019-09716-7. 4, 5, 84

[97] M. Grepl, *Reduced-basis approximation a posteriori error estimation for parabolic partial differential equations*, PhD thesis, Massachussetts Institute of Technology (MIT), Cambridge, USA, 2005, http://dspace.mit.edu/handle/1721.1/7582. 4, 27, 35, 85, 88, 90, 118

[98] M. Grepl, Y. Maday, N. C. Nguyen, and A. T. Patera, *Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equations*, ESAIM: Math. Model. Numer. Anal., 41 (2007), pp. 575–605, https://doi.org/10.1051/m2an:2007031. 4, 31, 43, 85, 97

[99] M. A. Grepl, *Certified reduced basis methods for nonaffine linear time-varying and nonlinear parabolic partial differential equations*, Math. Models Methods Appl. Sci., 22 (2012), pp. 1150015, 40, https://doi.org/10.1142/S0218202511500151. 27, 31, 124

[100] M. A. Grepl and A. T. Patera, *A posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations*, ESAIM: Math. Model. Numer. Anal., 39 (2005), pp. 157–181, https://doi.org/10.1051/m2an:2005006. 4, 27, 85, 95, 98

[101] E. J. Grimme, *Krylov Projection Methods for Model Reduction*, PhD thesis, University of Illinois at Urbana-Champaign, USA, 1997. 2, 4, 19, 21, 43, 61, 62, 63

[102] S. Gugercin, A. C. Antoulas, and C. Beattie, $\mathcal{H}_2$ *model reduction for large-scale linear dynamical systems*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 609–638, https://doi.org/10.1137/060666123. 3, 22, 62

[103] B. Haasdonk, *Convergence rates of the POD-greedy method*, ESAIM: Math. Model. Numer. Anal., 47 (2013), pp. 859–873, https://doi.org/10.1051/m2an/2012045. 28

[104] B. Haasdonk, *Model Reduction and Approximation: Theory and Algorithms*, vol. 15 of Computational Science & Engineering, SIAM Publications, Philadelphia, PA, 2017, ch. 2: Reduced Basis Methods for Parametrized PDEs — A Tutorial Introduction for Stationary and Instationary Problems, pp. 65–136, https://doi.org/10.1137/1.9781611974829. 124

[105] B. Haasdonk, M. Dihlmann, and M. Ohlberger, *A training set and multiple bases generation approach for parameterized model reduction based on adaptive grids in parameter space*, Math. Comput. Model. Dyn. Syst., 17 (2011), pp. 423–442, https://doi.org/10.1080/13873954.2011.547674. 3, 4, 5, 31, 84, 112, 113

[106] B. Haasdonk and M. Ohlberger, *Reduced basis method for finite volume approximations of parametrized linear evolution equations*, ESAIM: Math. Model. Numer. Anal., 42 (2008), pp. 277 – 302, https://doi.org/10.1051/m2an:2008001. 28, 85, 124

[107] S. Hain, M. Ohlberger, M. Radic, and K. Urban, *A hierarchical a posteriori error estimator for the reduced basis method*, Adv. Comput. Math., 45 (2019), pp. 2191–2214, https://doi.org/10.1007/s10444-019-09675-z. 31, 47, 48

[108] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, Springer Series in Statistics, Springer, New York, 2009, https://doi.org/10.1007/978-0-387-84858-7. 123

[109] M. Heinkenschloss, T. Reis, and A. C. Antoulas, *Balanced truncation model reduction for systems with inhomogeneous initial conditions*, Automatica J. IFAC, 47 (2011), pp. 559–564, https://doi.org/10.1016/j.automatica.2010.12.002. 18

[110] M. W. Hess and P. Benner, *Fast evaluation of time–harmonic maxwell's equations using the reduced basis method*, IEEE Trans. Microwave Theory Tech., 61 (2013), pp. 2265–2274, https://doi.org/10.1109/TMTT.2013.2258167. 43

[111] M. W. Hess, S. Grundel, and P. Benner, *Estimating the inf-sup constant in reduced basis methods for time-harmonic Maxwell's equations*, IEEE Trans. Microw. Theory Techn., 63 (2015), pp. 3549–3557, https://doi.org/10.1109/TMTT.2015.2473157. 91

[112] J. S. Hesthaven, G. Rozza, and B. Stamm, *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*, SpringerBriefs in Mathematics, Springer International Publishing, 2016, https://doi.org/10.1007/978-3-319-22470-1. 2, 27, 50, 85

[113] J. S. Hesthaven, B. Stamm, and S. Zhang, *Efficient greedy algorithms for high-dimensional parameter spaces with applications to empirical interpolation and reduced basis methods*, ESAIM: Math. Model. Numer. Anal., 48 (2014), pp. 259–283, https://doi.org/10.1051/m2an/2013100. 3, 27, 112, 113, 114, 200

[114] J. S. Hesthaven and S. Ubbiali, *Non-intrusive reduced order modeling of nonlinear problems using neural networks*, J. Comput. Phys., 363 (2018), pp. 55–78, https://doi.org/10.1016/j.jcp.2018.02.037. 3, 17

[115] J. S. Hesthaven and S. Zhang, *On the use of ANOVA expansions in reduced basis methods for parametric partial differential equations*, J. Sci. Comput., 69 (2016), pp. 292–313, https://doi.org/10.1007/s10915-016-0194-9. 112

[116] U. Hetmaniuk, R. Tezaur, and C. Farhat, *An adaptive scheme for a class of interpolatory model reduction methods for frequency response problems*, Internat. J. Numer. Methods Engrg., 93 (2013), pp. 1109–1124, https://doi.org/10.1002/nme.4436, https://doi.org/10.1002/nme.4436. 4, 43, 44, 61

[117] P. Holmes, J. L. Lumley, and G. Berkooz, *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*, Cambridge Monographs on Mechanics, Cambridge University Press, Cambridge, 1996, https://doi.org/10.1017/CBO9780511622700. 2, 24

[118] M. Hund, T. Mitchell, P. Mlinarić, and J. Saak, *Optimization-based parametric model order reduction via $\mathcal{H}_2 \otimes \mathcal{L}_2$ first-order necessary conditions*, Tech. Report 3, 2022, https://doi.org/10.1137/21M140290X. 3

[119] D. Huynh, D. Knezevic, Y. Chen, J. Hesthaven, and A. Patera, *A natural-norm successive constraint method for inf-sup lower bounds*, Comp. Meth. Appl. Mech. Eng., 199 (2010), pp. 1963–1975, https://doi.org/10.1016/j.cma.2010.02.011. 91

[120] D. Huynh, G. Rozza, S. Sen, and A. Patera, *A successive constraint linear optimization method for lower bounds of parametric coercivity and inf-sup lower bounds*, C.R. Acad. Sci. Paris, 345 (2007), pp. 473–478, https://doi.org/10.1016/j.crma.2007.09.019. 91

[121] L. Iapichino, A. Quarteroni, and G. Rozza, *Reduced basis method and domain decomposition for elliptic problems in networks and complex parametrized geometries*, Comput. Math. Appl., 71 (2016), pp. 408–430, https://doi.org/10.1016/j.camwa.2015.12.001. 97

[122] K. Ito and S. S. Ravindran, *A reduced basis method for control problems governed by PDEs*, in Control and Estimation of Distributed Parameter Systems (Vorau, 1996), vol. 126 of Internat. Ser. Numer. Math., Birkhäuser, Basel, 1998, pp. 153–168. 27

[123] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning : With Applications in R*, vol. 103 of Springer Texts in Statistics, Springer, New York, 2013. 71

[124] J. Jiang and Y. Chen, *Adaptive greedy algorithms based on parameter-domain decomposition and reconstruction for the reduced basis method*, Internat. J. Numer. Methods Engrg., 121 (2020), pp. 5426–5445, https://doi.org/10.1002/nme.6544. 4, 5, 31, 112

[125] J. Jiang, Y. Chen, and A. Narayan, *Offline-enhanced reduced basis method through adaptive construction of the surrogate training set*, J. Sci. Comput., 73 (2017), pp. 853–875, https://doi.org/10.1007/s10915-017-0551-3. 4, 5, 112

[126] K. Kunisch and S. Volkwein, *Optimal snapshot location for computing POD basis functions*, ESAIM: Math. Model. Numer. Anal., 44 (2010), pp. 509–529, https://doi.org/10.1051/m2an/2010011. 4, 5, 27

[127] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor, *Dynamic Mode Decomposition*, Society for Industrial and Applied Mathematics, 2016, https://doi.org/10.1137/1.9781611974508. 24

[128] K. Lee and K. T. Carlberg, *Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders*, J. Comput. Phys., 404 (2020), pp. 108973, 32, https://doi.org/10.1016/j.jcp.2019.108973. 3, 17

[129] R. M. Lewis and S. G. Nash, *Model problems for the multigrid optimization of systems governed by differential equations*, SIAM J. Sci. Comput., 26 (2005), pp. 1811–1837, https://doi.org/10.1137/S1064827502407792. 17

[130] Y. Liu and B. Anderson, *Singular perturbation approximation of balanced systems*, in Proceedings of the 28th IEEE Conference on Decision and Control,, vol. 2, 1989, pp. 1355–1360, https://doi.org/10.1109/CDC.1989.70360. 19

[131] Z. Lou and J.-M. Jin, *Modeling and simulation of broad-band antennas using the time-domain finite element method*, IEEE Trans. Antennas Propagat., 53 (2005), pp. 4099–4110, https://doi.org/10.1109/TAP.2005.859905. 58

[132] A. Lutowska, *Model order reduction for coupled systems using low-rank approximations*, PhD thesis, Mathematics and Computer Science, 2012, https://doi.org/10.6100/IR729804. 148

[133] L. Machiels, Y. Maday, and A. T. Patera, *Output bounds for reduced-order approximations of elliptic partial differential equations*, Comp. Meth. Appl. Mech. Eng., 190 (2001), pp. 3413–3426, https://doi.org/10.1016/S0045-7825(00)00275-9. 4, 27

[134] Y. Maday and B. Stamm, *Locally adaptive greedy approximations for anisotropic parameter reduced basis spaces*, SIAM J. Sci. Comput., 35 (2013), pp. A2417–A2441, https://doi.org/10.1137/120873868. 31, 84, 112, 124

[135] M. W. Mahoney, *Algorithmic and statistical perspectives on large-scale data analysis*, in Combinatorial Scientific Computing, Chapman & Hall/CRC Comput. Sci. Ser., CRC Press, Boca Raton, FL, 2012, pp. 427–469, https://doi.org/10.1201/b11644-17. 129

[136] M. Mangold, L. Feng, D. Khlopov, S. Palis, P. Benner, D. Binev, and A. Seidel-Morgenstern, *Nonlinear model reduction of a continuous fluidized bed crystallizer*, J. Comput. Appl. Math., 289 (2015), pp. 253–266, https://doi.org/10.1016/j.cam.2015.01.028. 101, 102

[137] A. Manzoni and F. Negri, *Heuristic strategies for the approximation of stability factors in quadratically nonlinear parametrized PDEs*, Adv. Comput. Math., 41 (2015), pp. 1255–1288, https://doi.org/10.1007/s10444-015-9413-4. 91, 200

[138] A. J. Mayo and A. C. Antoulas, *A framework for the solution of the generalized realization problem*, Linear Algebra Appl., 425 (2007), pp. 634–662, https://doi.org/10.1016/j.laa.2007.03.008. Special Issue in honor of P. A. Fuhrmann, Edited by A. C. Antoulas, U. Helmke, J. Rosenthal, V. Vinnikov, and E. Zerz. 2

[139] G. Meurant and J. Duintjer Tebbens, *Krylov Methods for Nonsymmetric Linear Systems: From Theory to Computations*, vol. 57 of Springer Series in Computational Mathematics, Springer, Cham, 2020, https://doi.org/10.1007/978-3-030-55251-0. 21

[140] P. Monk, *Finite Element Methods for Maxwell's Equations*, Numerical Mathematics and Scientific Computation, Oxford University Press, New York, 2003, https://doi.org/10.1093/acprof:oso/9780198508885.001.0001. 15

[141] B. C. Moore, *Principal component analysis in linear systems: controllability, observability, and model reduction*, IEEE Trans. Autom. Control, AC–26 (1981), pp. 17–32, https://doi.org/10.1109/TAC.1981.1102568. 2, 19

[142] D. A. Nagy, *Modal representation of geometrically nonlinear behavior by the finite element method*, Comput. & Structures, 10 (1979), pp. 683–688, https://doi.org/10.1016/0045-7949(79)90012-9. 27

[143] A. Narayan, *Reduced order modeling and numerical linear algebra.* ICERM Special Semester on Model and dimension reduction in uncertain and dynamic systems, Brown University, Providence, USA, 2020, https://icerm.brown.edu/materials/Slides/sp-s20/Tutorial-Introductory_Talk_-_Snapshot-based_model_reduction_and_numerical_linear_algebra_part_1_]_Akil_Narayan,_University_of_Utah.pdf. 125

[144] N. C. Nguyen, A. T. Patera, and J. Peraire, *A 'best points' interpolation method for efficient approximation of parametrized functions*, Internat. J. Numer. Methods Engrg., 73 (2008), pp. 521–543, https://doi.org/10.1002/nme.2086. 31, 32

[145] N.-C. Nguyen, G. Rozza, and A. T. Patera, *Reduced basis approximation and a posteriori error estimation for the time-dependent viscous Burgers' equation*, Calcolo, 46 (2009), pp. 157–185, https://doi.org/10.1007/s10092-009-0005-x. 124

[146] A. K. Noor, *Recent advances in reduction methods for nonlinear problems*, Comput. & Structures, 13 (1981), pp. 31–44, https://doi.org/10.1016/0045-7949(81)90106-1. 27

[147] A. K. Noor and J. M. Peters, *Reduced basis technique for nonlinear analysis of structures*, AIAA J., 18 (1980), pp. 455–462, https://doi.org/10.2514/3.50778. 27

[148] A. Odabasioglu, M. Celik, and L. T. Pileggi, *PRIMA: passive reduced-order interconnect macromodeling algorithm*, in Tech. Dig. 1997 IEEE/ACM Intl. Conf. Computer-Aided Design, IEEE Computer Society Press, Los Alamitos, CA, 1997, pp. 58–65. 19, 22

[149] M. Ohlberger and S. Rave, *Reduced basis methods: Success, limitations and future challenges*, Proceedings of the Conference Algoritmy, (2016), pp. 1–12, http://www.iam.fmph.uniba.sk/amuc/ojs/index.php/algoritmy/article/view/389. 159

[150] M. A. Oliver and R. Webster, *Basic Steps in Geostatistics: The Variogram and Kriging*, SpringerBriefs in Agriculture, Springer International Publishing, 2015, https://doi.org/10.1007/978-3-319-15865-5. 91

[151] G. M. Oxberry, T. Kostova-Vassilevska, W. Arrighi, and K. Chand, *Limited-memory adaptive snapshot selection for proper orthogonal decomposition*,

Internat. J. Numer. Methods Engrg., 109 (2017), pp. 198–217, https://doi.org/10.1002/nme.5283. 4, 5, 27

[152] C. Paige and M. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629, https://doi.org/10.1137/0712047. 91

[153] H. Panzer, J. Mohring, R. Eid, and B. Lohmann, *Parametric model order reduction by matrix interpolation*, at-Automatisierungstechnik, 58 (2010), pp. 475–484, https://doi.org/doi:10.1524/auto.2010.0863. 3

[154] H. K. F. Panzer, T. Wolf, and B. Lohmann, $\mathcal{H}_2$ *and* $\mathcal{H}_\infty$ *error bounds for model order reduction of second order systems by krylov subspace methods*, in 2013 European Control Conference (ECC), 2013, pp. 4484–4489, https://doi.org/10.23919/ECC.2013.6669657. 22, 61

[155] A. T. Patera and G. Rozza, *Reduced Basis Approximation and A Posteriori Error Estimation for Parametrized Partial Differential Equations*, MIT, 2007. MIT Pappalardo Graduate Monographs in Mechanical Engineering. 31, 43

[156] A. Paul-Dubois-Taine and D. Amsallem, *An adaptive and efficient greedy procedure for the optimal training of parametric reduced-order models*, Internat. J. Numer. Methods Engrg., 102 (2015), pp. 1262–1292, https://doi.org/10.1002/nme.4759. 31, 69, 112, 113, 114, 200

[157] B. Peherstorfer, *Model reduction for transport-dominated problems via online adaptive bases and adaptive sampling*, SIAM J. Sci. Comput., 42 (2020), pp. A2803–A2836, https://doi.org/10.1137/19M1257275. 3, 4

[158] B. Peherstorfer, D. Butnaru, K. Willcox, and H.-J. Bungartz, *Localized discrete empirical interpolation method*, SIAM J. Sci. Comput., 36 (2014), pp. A168–A192, https://doi.org/10.1137/130924408. 4, 5

[159] B. Peherstorfer, Z. Drmač, and S. Gugercin, *Stabilizing discrete empirical interpolation via randomized and deterministic oversampling*, e-prints 1808.10473v3, arXiv, 2018, https://arxiv.org/abs/1808.10473v3. 123, 142, 175

[160] B. Peherstorfer and K. Willcox, *Online adaptive model reduction for nonlinear systems via low-rank updates*, SIAM J. Sci. Comput., 37 (2015), pp. A2123–A2150, https://doi.org/10.1137/140989169. 4, 5

[161] B. Peherstorfer and K. Willcox, *Data-driven operator inference for nonintrusive projection-based model reduction*, Comp. Meth. Appl. Mech. Eng., 306 (2016), pp. 196–215, https://doi.org/10.1016/j.cma.2016.03.025. 2, 17, 24

[162] J. S. Peterson, *The reduced basis method for incompressible viscous flow calculations*, SIAM Journal on Scientific and Statistical Computing, 10 (1989), pp. 777–786, https://doi.org/10.1137/0910047. 27

[163] N. A. Pierce and M. B. Giles, *Adjoint recovery of superconvergent functionals from PDE approximations*, SIAM Rev., 42 (2000), pp. 247–264, https://doi.org/10.1137/S0036144598349423. 85, 88

[164] R. Pinnau, *Model reduction via proper orthogonal decomposition*, in Model Order Reduction: Theory, Research Aspects and Applications, vol. 13 of Math. Ind., Springer, Berlin, 2008, pp. 95–109, https://doi.org/10.1007/978-3-540-78841-6_5. 26

[165] T. A. Porsching, *Estimation of the error in the reduced basis method solution of nonlinear equations*, Math. Comp., 45 (1985), pp. 487–496, https://doi.org/10.2307/2008138. 27

[166] C. Prud'homme, D. V. Rovas, K. Veroy, L. Machiels, Y. Maday, A. T. Patera, and G. Turinici, *Reliable real-time solution of parametrized partial differential equations: Reduced-basis output bound methods*, ASME. J. Fluids Eng, 124 (2002), pp. 70–80, https://doi.org/10.1115/1.1448332. 27, 45

[167] A. Quarteroni, A. Manzoni, and F. Negri, *Reduced Basis Methods for Partial Differential Equations*, vol. 92 of La Matematica per il 3+2, Springer International Publishing, 2016, https://doi.org/10.1007/978-3-319-15431-2. 2, 27, 33, 35, 43, 50, 85

[168] A. Quarteroni and G. Rozza, eds., *Reduced Order Methods for Modeling and Computational Reduction*, vol. 9 of MS&A. Modeling, Simulation and Applications, Springer, Cham, 2014, https://doi.org/10.1007/978-3-319-02090-7. Selected papers from the workshop "Reduced Basis, POD and Reduced Order Methods for Model and Computational Reduction: Towards Real-Time Computing and Visualization" held at Ecole Polytechnique Fédérale de Lausanne, Lausanne, May 14–16, 2012. 2

[169] A. Quarteroni, G. Rozza, and A. Manzoni, *Certified reduced basis approximation for parametrized partial differential equations and applications*, J. Math. Ind., 1 (2011), pp. Art. 3, 44, https://doi.org/10.1186/2190-5983-1-3. 124

[170] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, Adaptive Computation and Machine Learning, MIT Press, Cambridge, MA, 2006. 17

[171] M. Rathinam and L. R. Petzold, *A new look at proper orthogonal decomposition*, SIAM J. Numer. Anal., 41 (2003), pp. 1893–1925, https://doi.org/10.1137/S0036142901389049. 27

[172] S. Rave and J. Saak, *A non-stationary thermal-block benchmark model for parametric model order reduction*, in Model Reduction of Complex Dynamical Systems, P. Benner, T. Breiten, H. Faßbender, M. Hinze, T. Stykel, and R. Zimmermann, eds., vol. 171 of International Series of Numerical Mathematics, Birkhäuser, Cham, 2021, pp. 349–356, https://doi.org/10.1007/978-3-030-72983-7_16. 137

[173] M. Redeker and B. Haasdonk, *A POD-EIM reduced two-scale model for precipitation in porous media*, Math. Comput. Model. Dyn. Syst., 22 (2016), pp. 323–344, https://doi.org/10.1080/13873954.2016.1198384. 35

[174] T. Reis and T. Stykel, *A survey on model reduction of coupled systems*, in Model Order Reduction: Theory, Research Aspects and Applications, W. H. A. Schilders, H. A. van der Vorst, and J. Rommes, eds., Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 133–155, https://doi.org/10.1007/978-3-540-78841-6_7. 148, 149

[175] W. C. Rheinboldt, *On the theory and error estimation of the reduced basis method for multi-parameter problems*, Nonlinear Analysis: Theory, Methods and Applications, 21 (1993), pp. 849–858, https://doi.org/https://doi.org/10.1016/0362-546X(93)90050-3. 27

[176] S. Rippa, *An algorithm for selecting a good value for the parameter c in radial basis function interpolation*, Adv. Comput. Math., 11 (1999), pp. 193–210, https://doi.org/10.1023/A:1018975909870. 71

[177] D. V. Rovas, *Reduced-Basis Output Bound Methods for Parametrized Partial Differential Equations*, PhD thesis, Massachussetts Institute of Technology (MIT), Cambridge, USA, 2003, https://dspace.mit.edu/handle/1721.1/16956. 4, 27, 85

[178] C. W. Rowley, I. Mezić, S. Bagheri, P. Schlatter, and D. S. Henningson, *Spectral analysis of nonlinear flows*, J. Fluid Mech., 641 (2009), pp. 115–127, https://doi.org/10.1017/S0022112009992059. 24

[179] G. Rozza, M. Hess, G. Stabile, M. Tezzele, and F. Ballarin, *Basic ideas and tools for projection-based model reduction of parametric partial differential equations*, in Snapshot-Based Methods and Algorithms, P. Benner, S. Grivet-Talocia, A. Quarteroni, G. Rozza, W. Schilders, and L. M. Silveira, eds., De Gruyter, 2021, pp. 1–47, https://doi.org/10.1515/9783110671490-001. 129

[180] G. Rozza, D. B. P. Huynh, and A. T. Patera, *Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations: application to transport and continuum mechanics*, Arch. Comput. Methods Eng., 15 (2008), pp. 229–275, https://doi.org/10.1007/s11831-008-9019-9. 27, 42, 85, 88, 90

[181] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869, https://doi.org/10.1137/0907058. 91

[182] A. K. Saibaba, *Randomized discrete empirical interpolation method for nonlinear model reduction*, SIAM J. Sci. Comput., 42 (2020), pp. A1582–A1608, https://doi.org/10.1137/19M1243270. 132

[183] W. H. A. Schilders, H. A. van der Vorst, and J. Rommes, *Model Order Reduction: Theory, Research Aspects and Applications*, Springer-Verlag, Berlin, Heidelberg, 2008. 2

[184] P. J. Schmid, *Dynamic mode decomposition of numerical and experimental data*, J. Fluid Mech., 656 (2010), pp. 5–28, https://doi.org/10.1017/S0022112010001217. 24

[185] A. Schmidt, D. Wittwar, and B. Haasdonk, *Rigorous and effective a-posteriori error bounds for nonlinear problems - application to RB methods*, Adv. Comput. Math., 46 (2020), pp. 1–30, https://doi.org/10.1007/s10444-020-09741-x. 43

[186] S. Sen, *Reduced basis approximation and a posteriori error estimation for non-coercive elliptic problems : applications to acoustics*, PhD thesis, Massachussetts Institute of Technology (MIT), Cambridge, USA, 2007, http://dspace.mit.edu/handle/1721.1/7582. 85

[187] S. Sen, *Reduced-basis approximation and a posteriori error estimation for many-parameter heat conduction problems*, Numerical Heat Transfer, Part B: Fundamentals, 54 (2008), pp. 369–389, https://doi.org/10.1080/10407790802424204. 31, 112

[188] P. Sirković and D. Kressner, *Subspace acceleration for large-scale parameter-dependent hermitian eigenproblems*, SIAM J. Matrix Anal. Appl., 37 (2016), pp. 695–718, https://doi.org/10.1137/15M1017181. 91

[189] L. Sirovich, *Turbulence and the dynamics of coherent structures. parts I-III*, Quart. Appl. Math., 45 (1987), pp. 561–590, http://www.jstor.org/stable/43637457. 2

[190] K. Smetana and M. Ohlberger, *Hierarchical model reduction of nonlinear partial differential equations based on the adaptive empirical projection method and reduced basis techniques*, ESAIM: Math. Model. Numer. Anal., 51 (2017), pp. 641–677, https://doi.org/10.1051/m2an/2016031. 97

[191] K. Smetana, O. Zahm, and A. T. Patera, *Randomized residual-based error estimators for parametrized equations*, SIAM J. Sci. Comput., 41 (2019), pp. A900–A926, https://doi.org/10.1137/18M120364X. 43, 44, 47, 48, 49, 50, 55, 59

[192] I. Sobol', *Theorems and examples on high dimensional model representation*, Reliability Engineering & System Safety, 79 (2003), pp. 187–193, https://doi.org/10.1016/S0951-8320(02)00229-6. 112

[193] A. Sommer, O. Farle, and R. Dyczij-Edlinger, *A new method for accurate and efficient residual computation in adaptive model-order reduction*, IEEE Trans. Magn., 51 (2015), pp. 1–4, https://doi.org/10.1109/TMAG.2014.2352812. 43

[194] N. Son, P.-Y. Gousenbourger, E. Massart, and T. Stykel, *Balanced truncation for parametric linear systems using interpolation of Gramians: a comparison of algebraic and geometric approaches*, in Model Reduction of Complex Dynamical Systems, P. Benner, T. Breiten, H. Faßbender, M. Hinze, T. Stykel, and R. Zimmermann, eds., vol. 171 of International Series of Numerical Mathematics, Birkhäuser, Cham, 2021, pp. 31–51, https://doi.org/10.1007/978-3-030-72983-7_2. 3, 19

[195] M. Tezzele, F. Ballarin, and G. Rozza, *Combined parameter and model reduction of cardiovascular problems by means of active subspaces and POD-Galerkin methods*, in Mathematical and Numerical Modeling of the Cardiovascular System and Applications, vol. 16 of SEMA SIMAI Springer Ser., Springer, Cham, 2018, pp. 185–207, https://doi.org/10.1007/978-3-319-96649-6_8. 31, 113, 129

[196] The MORwiki Community, *MORwiki - Model Order Reduction Wiki*. http://modelreduction.org. 75

[197] T. Tonn, *Reduced-Basis Method (RBM) for Non-Affine Elliptic Parametrized PDEs*, PhD thesis, Institut für Numerische Mathematik, Universität Ulm, Ulm, Germany, 2011. 31, 97

[198] K. Urban and A. T. Patera, *An improved error bound for reduced basis approximation of linear parabolic problems*, Math. Comp., 83 (2014), pp. 1599–1615, https://doi.org/10.1090/S0025-5718-2013-02782-2. 31, 85

[199] B. van Stein, H. Wang, W. Kowalczyk, M. T. M. Emmerich, and T. Bäck, *Cluster-based Kriging approximation algorithms for complexity reduction*, Appl. Intell., 50 (2020), pp. 778–791, https://doi.org/10.1007/s10489-019-01549-7. 114

[200] K. Veroy, C. Prud'homme, D. V. Rovas, and A. T. 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, Orlando, United States, 2003, https://hal.archives-ouvertes.fr/hal-01219051. 4, 27

[201] K. Veroy, D. V. Rovas, and A. T. Patera, *A posteriori error estimation for reduced-basis approximation of parametrized elliptic coercive partial differential equations: "convex inverse" bound conditioners*, ESAIM Control Optim. Calc. Var., 8 (2002), pp. 1007–1028, https://doi.org/10.1051/cocv:2002041. 4, 27, 85

[202] D. C. Villemagne and R. E. Skelton, *Model reduction using a projection formulation*, Internat. J. Control, 46 (1987), pp. 2141–2169, https://doi.org/10.1080/00207178708934040. 2, 19

[203] S. Volkwein, *Model reduction using proper orthogonal decomposition*, lecture notes, University of Konstanz, 2013. 26

[204] H. Wendland, *Scattered Data Approximation*, vol. 17 of Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, Cambridge, 2005. 37, 39

[205] S. Werner, *Hankel-norm approximation of descriptor systems*, master's thesis, Otto-von-Guericke-Universität, Magdeburg, Germany, 2016, https://doi.org/10.25673/4507. 19

[206] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, *A data-driven approximation of the Koopman operator: extending dynamic mode decomposition*, J. Nonlinear Sci., 25 (2015), pp. 1307–1346, https://doi.org/10.1007/s00332-015-9258-5. 24

[207] D. Wirtz, D. C. Sorensen, and B. Haasdonk, *A posteriori error estimation for DEIM reduced nonlinear dynamical systems*, SIAM J. Sci. Comput., 36 (2014), pp. A311–A338, https://doi.org/10.1137/120899042. 35, 95

[208] T. Wolf, H. Panzer, and B. Lohmann, *Gramian-based error bound in model reduction by Krylov subspace methods*, IFAC Proceedings Volumes, 44 (2011), pp. 3587–3592, https://doi.org/https://doi.org/10.3182/20110828-6-IT-1002.02809. 4, 61

[209] M. Yano and A. T. Patera, *An lp empirical quadrature procedure for reduced basis treatment of parametrized nonlinear pdes*, Comp. Meth. Appl. Mech. Eng., 344 (2019), pp. 1104–1123, https://doi.org/10.1016/j.cma.2018.02.028. 30

[210] M. Yano, A. T. Patera, and K. Urban, *A space-time hp-interpolation-based certified reduced basis method for Burgers' equation*, Math. Models Methods Appl. Sci., 24 (2014), pp. 1903–1935, https://doi.org/10.1142/S0218202514500110. 31, 85

[211] H. Zha, X. He, C. Ding, M. Gu, and H. D. Simon, *Spectral relaxation for k-means clustering*, in Advances in Neural Information Processing Systems 14, T. G. Dietterich, S. Becker, and Z. Ghahramani, eds., MIT Press, 2002, pp. 1057–1064, http://papers.nips.cc/paper/1992-spectral-relaxation-for-k-means-clustering.pdf. 123

[212] Y. Zhang, *Model order reduction for parameterized nonlinear evolution equations*, Dissertation, Otto-von-Guericke-Universität, Magdeburg, Germany, 2016, https://doi.org/10.25673/4434. 35

[213] Y. Zhang, L. Feng, S. Li, and P. Benner, *Accelerating PDE constrained optimization by the reduced basis method: application to batch chromatography*, Internat. J. Numer. Methods Engrg., 104 (2015), pp. 983–1007, https://doi.org/10.1002/nme.4950. 27, 149, 153, 154

[214] Y. Zhang, L. Feng, S. Li, and P. Benner, *An efficient output error estimation for model order reduction of parametrized evolution equations*, SIAM J. Sci. Comput., 37 (2015), pp. B910–B936, https://doi.org/10.1137/140998603. 4, 31, 43, 85, 86, 87, 88, 89, 90, 92, 101, 108, 118, 124

STATEMENT OF SCIENTIFIC COOPERATIONS

This work is based on articles and reports (published and unpublished) that have been obtained in cooperation with various coauthors. To guarantee a fair assessment of this thesis, this statement clarifies the contributions that each individual coauthor has made. The following people contributed to the content of this work:

- Dr. Lihong Feng (LF), Max Planck Institute for Dynamics of Complex Technical Systems, Magdeburg, Germany,
- Prof. Peter Benner (PB), Max Planck Institute for Dynamics of Complex Technical Systems, Magdeburg, Germany,
- Dr. Valentín de la Rubia (VR), Departamento de Matemática Aplicada a las TIC, ETSI de Telecomunicación, Universidad Politécnica de Madrid, Spain.

Chapter 3

The idea for an *inf-sup-constant-free* error estimator was developed by me with suggestions from LF. I worked by myself on its development, algorithmic implementation as part of a greedy algorithm (Algorithm 3.1), and obtained all the numerical results. Proposition 3.1 and Theorem 3.2 were derived and proved by myself, and were proofread later by LF. LF also suggested to show the result in Theorem 3.3, which I then proved. VR provided the filter and antenna models used in Section 3.3.6 and also provided access to the Madrid high-performance computer cluster to perform the simulations. LF, PB, and VR proofread and suggested several improvements to the manuscript [57] on which Section 3.3 is based.

I extended the data-driven RBF surrogate from the work [55] to the frequency-domain based on the error estimator from [83]. I worked on this idea by myself and developed the algorithmic implementation (Algorithm 3.3). The numerical experiments and corresponding results were also obtained by me. LF advised me during this stage on various aspects. LF, PB, and VR proofread and suggested several improvements to the manuscript [56] on which Section 3.4 is based.

Chapter 4

LF pointed out the work [87] and suggested me to extend the methodology to parametric systems. LF also suggested to use a modified output term and derive an improved output error estimator. I then independently developed the corresponding algorithms

and methods. Theorem 4.2 was proved by me. LF provided me advices on the computable error estimator in Section 4.2.3. I conceived and developed Algorithms 4.1 and 4.2 and also performed all numerical experiments. LF advised me during the exploration of the idea on using the RBF-based interpolation of the *inf-sup* constant in the output error estimator (Section 4.2.5) based on the work [137]. I independently developed the corresponding algorithm based on the literature. Both LF and PB proofread and suggested several improvements to the manuscript [53] on which this chapter is based.

Chapter 5

LF referred me to the works [113, 156] and suggested me to find a way to use RBFs and combine adaptive parameter sampling and the adaptive basis enrichment approach from Chapter 3 and Chapter 4, respectively. I worked by myself on the algorithmic development and implementations (Algorithm 5.1). LF advised on various details during this development. All numerical experiments were performed by me. LF and PB together proofread and improved the manuscript [55] on which Section 5.2 is based.
I conceived and developed the subsampling-based strategy for training set selection in Section 5.3. The algorithms (Algorithms 5.2 and 5.3) were developed by myself. LF advised during the development and suggested to also add the comparison with the QR-based sampling in the numerical results. LF and PB proofread and made revisions to the manuscript [54] which forms the basis for Section 5.3.

Chapter 6

LF suggested me to extend the adaptive basis enrichment and adaptive training set sampling methods from Chapters 4 and 5 to coupled systems. I independently developed the algorithms (Algorithms 6.1 and 6.2) and LF advised with several suggestions for improvement. The related numerical experiments and results were obtained by me.

DECLARATION OF HONOR

I hereby declare that I produced this thesis without prohibited assistance and that all sources of information that were used in producing this thesis, including my own publications, have been clearly marked and referenced.

In particular I have not willfully:

- Fabricated data or ignored or removed undesired results.
- Misused statistical methods with the aim of drawing other conclusions than those warranted by the available data.
- Plagiarized data or publications or presented them in a distorted way.

I know that violations of copyright may lead to injunction and damage claims from the author or prosecution by the law enforcement authorities.

This work has not previously been submitted as a doctoral thesis in the same or a similar form in Germany or in any other country. It has not previously been published as a whole.

Magdeburg, 25.04.2022

Sridhar Chellappa