RICE UNIVERSITY

Dimension Reduction for Unsteady Nonlinear Partial Differential Equations via Empirical Interpolation Methods

by

Saifon Chaturantabut

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree

Master of Arts

APPROVED, THESIS COMMITTEE:

Dr. Danny Sorensen, Chairman
Noah G. Harding Professor of Computational and Applied Mathematics

Dr. Matthias Heinischloss
Professor of Computational and Applied Mathematics

Dr. Mark Embree
Associate Professor of Computational and Applied Mathematics

Dr. Matteo Pasquali
Professor of Chemical and Biomolecular Engineering

HOUSTON, TEXAS

NOVEMBER, 2008
INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.
Abstract

Dimension Reduction for Unsteady Nonlinear Partial Differential Equations via Empirical Interpolation Methods

by

Saifon Chaturantabut

This thesis evaluates and compares the efficiencies of techniques for constructing reduced-order models for finite difference (FD) and finite element (FE) discretized systems of unsteady nonlinear partial differential equations (PDEs). With nonlinearity, the complexity for solving the reduced-order system constructed directly from the well-known Proper Orthogonal Decomposition (POD) technique alone still depends on the dimension of the original system. Empirical Interpolation Method (EIM), proposed in [2], and its discrete variation, Discrete Empirical Interpolation Method (DEIM), introduced in this thesis, are therefore combined with the POD technique to remove this inefficiency in the nonlinear terms of FE and FD cases, respectively. Numerical examples demonstrate that both POD-EIM and POD-DEIM approaches not only dramatically reduce the dimension of the original system with high accuracy, but also remove the dependence on the dimension of the original system as reflected in the decrease computational time compared to the POD approach.
Acknowledgements

I am truly grateful to my advisor, Dr. Danny C. Sorensen, for his thoughtful suggestions, support, understanding, encouragement, and patience throughout this thesis process. I really appreciate his prompt assistance with invaluable advices on this thesis. This thesis would never be complete without him.

I would like to thank the members of my thesis committee: Dr. Matthias Heinkenschloss, for his insightful guidance, particularly in providing the useful background references and the model problem used in this thesis; Dr. Mark Embree, for his helpful suggestion on the error analysis used in this thesis and for his incredibly detailed proof-reading with very useful comments; and Dr. Matteo Pasquali, for his helpful suggestions on the polymer modeling, which will be continued in my future work.
I also would like to thank Dr. Steve Cox and Anthony Kellemns, for providing the neuron model used for testing the algorithm in this thesis; Dr. Jan Hewitt, for her generous help on the thesis writing; and the CAAM staff, especially Daria Lawrence, for her help on the paper work.

My special thank goes to my family, my parents and my two sisters, for their endless encouragement, warm support, and unconditional love.
Contents

Abstract ii

Acknowledgements iii

List of Figures vii

1 Introduction 1

1.1 Motivation and Goal ........................................... 1

1.2 Previous Work and Choices of Techniques ................... 3

1.2.1 Techniques for Constructing the Reduced Basis ........... 3

1.2.2 Techniques for Nonlinearities ............................. 6

1.3 Thesis Scope and Outline ..................................... 9

2 Methods 11

2.1 Reduced-Order Modeling by Projection Method ............... 12

2.2 Proper Orthogonal Decomposition (POD) ...................... 14

2.3 Discrete Empirical Interpolation Method (DEIM) ............. 21
2.3.1 DEIM: Algorithm for Interpolation Indices ........................................ 26
2.3.2 Error Bound for DEIM ...................................................................... 33
2.3.3 Numerical Examples of the DEIM Error Bound ......................... 39

2.4 Empirical Interpolation Method (EIM) [2] ....................................... 46
2.4.1 EIM for interpolation points ......................................................... 49
2.4.2 Alternative Form of Algorithm for EIM ...................................... 52

3 Model Problem: Finite Difference Discretized System ......................... 55

3.1 FitzHugh-Nagumo (FN) System ....................................................... 56
3.2 Full Order Model of Finite Difference Discretized System ............ 56
3.3 Proper Orthogonal Decomposition (POD) Basis of the Snapshots ... 60
3.4 Reduced Order Model from POD-Galerkin Method ..................... 61
3.5 Reduced Model for Nonlinear Term: DEIM .................................... 62

4 Numerical Results: Finite Difference Discretized System .................... 65

4.1 FitzHugh-Nagumo System ............................................................. 66
4.2 A Variation of FitzHugh-Nagumo System with Periodic Solution ... 75

5 Model Problem: Finite Element Discretized System ......................... 86

5.1 Unsteady Burgers’ Equation (1D) .................................................... 87
5.2 Full Order Model of the Discretized System ................................... 87
5.3 Proper Orthogonal Decomposition (POD) Basis of the Snapshots ... 90
5.4 Reduced Order Model from POD-Galerkin Method ..................... 92
5.5 Reduced Model for Nonlinear Term .......................... 95

5.5.1 Empirical Interpolation Method (EIM) with POD basis for the
Nonlinear term of the Reduced System ......................... 95

5.5.2 Precomputing Technique for Nonlinear term of the Reduced
System .......................................................... 99

6 Numerical Results: Finite Element Discretized System 102

6.1 Burgers’ Equations (1D) with a Fixed Viscosity Parameter .... 103

6.2 Burgers’ Equations (1D) with Varied Viscosity Parameters .... 112

7 Conclusions and Future Work .................................. 120

A Computational Details for the Model Problem: Burgers’ Equation 126

A.1 Weak form of Burgers’ equation ................................ 126

A.2 Precomputing the nonlinear term .............................. 127

A.2.1 Varying Dimensions of the EIM for the Nonlinear Term ... 129

Bibliography ......................................................... 131
List of Figures

2.1 Illustration of the selection process of indices in Algorithm 1 for the DEIM approximation. The input basis vectors are the first 6 eigenvectors of the discrete Laplacian. From the plots, \( u = u_\ell, Uc \) and \( r = u_\ell - Uc \) are defined as in the iteration \( \ell \) of Algorithm 1. ............................................. 28

2.2 Singular values and the corresponding first 6 POD bases with DEIM points of snapshots from (2.73) ......................................................... 42

2.3 The approximate functions from DEIM of dimension 10 compared with the original functions (2.73) of dimension \( n = 100 \) at \( \mu = 1.17, 1.5, 2.3, 3.1 \). ............................................. 42

2.4 Comparison of the average errors of POD and DEIM approximations for (2.73) with the average error bounds and their approximations given in (2.70) and (2.71), respectively. ......................................................... 42

2.5 Singular values and the first 6 corresponding POD basis vectors of the snapshots of the nonlinear function (2.75). ......................................................... 44

2.6 First 20 points selected by DEIM for the nonlinear function (2.75). .................. 45
2.7 Comparison of the original nonlinear function (2.75) of dimension 400 with the 
POD and DEIM approximations of dimension 6 at parameter $\mu = [-0.05, -0.05]$. 45

2.8 Left: Average errors of POD and DEIM approximations for (2.75) with the average 
error bounds given in (2.70) and their approximations given in (2.71). Right: 
Average cpu time for evaluating the POD and DEIM approximations. 46

4.1 The singular values of the snapshots solution for $v$ and $w$ from the full finite dif-
ference system (dim = 1024). 67

4.2 Numerical Solutions $v$ and $w$ of the full-order system (dim=1024) 67

4.3 Numerical Solutions $v$ and $w$ of the reduced system from the $POD$ approach: using 
dim POD = 10 and 30; compared with full FD system (dim= 1024). 70

4.4 Numerical Solutions $v$ and $w$ of the reduced systems from the $POD-DEIM$ ap-
proach: using dim of POD= 30 with dim of DEIM = 10 and 30; compared with 
full FD system (dim 1024). 71

4.5 Average relative errors in (4.5) of POD reduced systems compared to the original 
FD discretized systems. 72
4.6 Average relative errors of the POD-DEIM (solid lines)/POD (dashed line) reduced systems and the original FD discretized system, as defined in (4.5) and in (4.6). Notice that (i) for each fixed dimension of the POD, the error of the POD approach (dashed line) is the lower bound of the error of POD-DEIM approach (solid lines); and (ii) for each fixed dimension of DEIM, at the beginning, the error decreases as the dimension of POD increases, then becomes constant due to the larger effect of the error from the DEIM.

73

4.7 Average relative errors in (4.7) of the POD-DEIM reduced systems and the corresponding POD reduced systems (same dimension for POD but without DEIM).

74

4.8 Computational time for solving reduced (POD and POD-DEIM) systems compared with the full system.

74

4.9 The eigenvalues of the 100 snapshot solutions for $v$, $w$, and $f(u)$ from the full-order FD system.

78

4.10 Numerical solutions $v$, $w$, and the phase-space diagram at different spatial points $x$ with its projection onto the $v$-$w$ plane from the full-order FD system (dim = 1024).

79

4.11 Numerical solutions $v$ and $w$ of the reduced-order systems from the POD approach: using POD bases of dimensions 3, 5, and 10; compared with the original FD system (dim 1024).

80

4.12 The phase-space diagram of $v$ and $w$ at different spatial points $x$ (left) and its projection onto the $v$-$w$ plane (right) from POD reduced-order systems: POD bases of dimensions 3, 5, and 10; compared with original FD system (dim 1024).
4.13 Numerical solutions $v$ and $w$ of the reduced systems from the $POD-DEIM$ approach:

using a POD basis of dimension 5 with DEIM of dimensions 1, 3, and 5; compared with the original FD system (dim 1024). .......................... 82

4.14 The phase-space diagram of $v$ and $w$ at different spatial points $x$ (left) and its projection onto the $v$-$w$ plane (right) from $POD-DEIM$ reduced systems: POD dimension 5 with DEIM dimensions 1, 3, and 5; compared with original FD system (dim 1024). .......................... 83

4.15 (a) Average relative errors of the solutions from the POD reduced systems and (b) from the POD-DEIM reduced system (solid lines) compared with the ones from POD reduced systems (dashed line). Notice that errors of POD (dashed line) are the lower bound for the errors of POD-DEIM for each fixed dimension of the POD basis. .......................... 84

4.16 Average relative errors of the reduced-order system from the POD-DEIM (solid lines) and POD (dashed line) approaches with the plots of the singular values (of the snapshot solutions $v, w, f(v)$). .......................... 85

4.17 Computational time (scaled with the CPU time of the sparse full-order system) for solving reduced (POD and POD-DEIM) systems compared with the full system. .......................... 85

5.1 Plots of the first 4 POD basis functions from the snapshots of the model problem (left). Notice that the first POD basis has the similar shape as the solution space of the problem (right). .......................... 93
5.2 The first 4 POD basis functions of the nonlinear term (top); and the corresponding basis functions after applying EIM (bottom). 98

6.1 Singular values of the 70 snapshot solutions from the original system (FE discretization of dimension 100). The plot shows the rapid decay of the first ten singular values. 105

6.2 Numerical solutions of the reduced-order system from the Direct POD approach: using a POD basis of dimension 1 to 5 at time $t = 0, 0.2029, 0.9855, 1.2174$, and 2 (2D on the top) and using a POD basis of dimension 6 (the 3D surface plot on the bottom left); compared with the original system (100 FE basis). 106

6.3 Numerical solutions of the reduced-order system from the Precompute POD approach: using a POD basis of dimension 1 to 5 at time $t = 0, 0.2029, 0.9855, 1.2174$, and 2 (the 2D plots on the top) and using a POD basis of dimension 6 (the 3D surface plot on the bottom left); compared with the original system (100 FE basis). 107

6.4 Numerical solutions of the reduced-order system from the POD-EIM approach: using the EIM dimension 5 with a POD basis of dimension 1 to 5 at time $t = 0, 0.2029, 0.9855, 1.2174$, and 2 (the 2D plots on the top) and using a POD basis of dimension 6 (the 3D surface plot on the bottom left); compared with the original system (100 FE basis). 108
6.5 (Top) Relative errors of the solution from reduced-order systems constructed from POD bases of size 1 to 15 with the three different approaches; (Bottom) CPU times used for solving the corresponding reduced-order systems constructed from POD bases of size 1 to 15. .................................................. 111

6.6 Singular values plot of the snapshots .................................................. 114

6.7 Top: Plots of first 4 POD basis functions constructed from snapshot solutions with various $\nu \in [0.005, 0.1]$. Bottom: Plots of solutions from FE with different $\nu$: $\nu = 0.005, 0.01, 0.03, 0.1$, which are used in the sampled snapshots. ............... 114

6.8 Numerical solutions from the FE systems of dim 200 with different values of viscosity parameters $\nu = 0.007, 0.01, 0.07$. .................................................. 115

6.9 Numerical solutions from reduced systems from POD approach with dim 1, 6, and 10. Each column corresponds to a fixed value of viscosity parameter: $\nu = 0.007, 0.01, and 0.07$. .................................................. 116

6.10 Numerical solutions from reduced systems from Precomputed-POD approach with dim 1, 6, and 10. Each column corresponds to a fixed value of viscosity parameter: $\nu = 0.007, 0.01, and 0.07$. .................................................. 117

6.11 Numerical solutions from reduced systems from POD-EIM approach with dim of POD = 1, 6, 10, and with a fixed dim of EIM = 10. Each column corresponds to a fixed value of viscosity parameter: $\nu = 0.007, 0.01, and 0.07$. .................................................. 118

6.12 Average relative errors of the three approaches with three different parameters $\nu = 0.007, 0.01, and 0.07$. .................................................. 119
A.1 Errors from using different dimensions for EIM-POD basis ........................................ 130

A.2 CPU times used for solving the reduced-order systems constructed from POD basis of size 1 to 70 with the different three approaches and different dimensions for the EIM-POD basis ......................................................... 130
Chapter 1

Introduction

1.1 Motivation and Goal

In many practical applications, such as in optimal control problems governed by unsteady nonlinear partial differential equations (PDEs), the discretized system of PDEs must be solved repeatedly in real-time simulations. The dimension of the discretized systems constructed from the Galerkin method with standard basis functions can get large very quickly in order to obtain higher accuracy. This is because these classical bases are designed to capture local phenomena and do not contain the global characteristics of the system. This thesis compares and evaluates techniques for constructing a reduced-order system that gives a good approximation to the original discretized system and can be re-solved efficiently in a reasonable computational time.

The systems to be considered in this thesis are obtained from the finite difference
(FD) and finite element (FE) discretized systems of \textit{unsteady nonlinear convection-diffusion differential equations}, which commonly arise in many practical applications, such as in gas dynamics modeling, turbulence modeling, and control problems. All of these time dependent systems share the need to solve the spatial approximation rapidly at each time step. The classical FD and FE methods involve basis functions with local support. Resolution of fine scale features involves higher order elements and/or mesh refinement leading to very large sparse linear equations.

The goal of this thesis is to construct reduced-order systems for these FD and FE discretized systems by projecting the systems and well as their solutions onto \textit{low dimensional} subspaces, which are expected to contain dominant characteristics of their corresponding solution spaces. Each of these subspaces is often determined by a set of \textit{reduced basis} functions with global support which are "learned"; they are constructed from high fidelity FD or FE approximation and as such, are problem dependent. Fine scale detail is \textit{embedded} in these global basis functions and this makes it possible to obtain good approximation with very few basis functions. The resulting linear systems are low dimensional but dense.

Since PDEs of interest are nonlinear, this thesis addresses the problem of obtaining a reduced order approximation to nonlinear terms that has a complexity that is on the order of the number of reduced variables. Such a reduction is generally not obtained with the standard Proper Orthogonal Decomposition (POD) approach. With POD, the computational complexity of the nonlinear term in the reduced system still
depends on the dimension of the original system. As a result, the dimension of the system is not truly reduced. Two types of techniques are considered in this thesis:

- Techniques for constructing a reduced basis that gives a good approximation of the space of the solutions;
- Techniques for removing the dependence on the dimension of the original discretized system due to the nonlinearities.

1.2 Previous Work and Choices of Techniques

1.2.1 Techniques for Constructing the Reduced Basis

Reduced-basis techniques originated in the 1980’s in the context of nonlinear structural analysis [18] and later in the context of fluid flow simulation [21], [11]. A primary motivation for the reduced-basis methods comes from the observation that the solution space is often embedded in a manifold that has much lower dimension than the dimension of the finite element basis or other standard basis used in the Galerkin projection method. Existing reduced-basis methods are problem dependent; the reduced basis can be used only for the solutions of a particular system. These reduced-basis methods would start from a set of selected solutions of the original full-order system. Examples of the early developments on reduced-basis techniques are the Lagrange, Taylor, and Hermite methods. These techniques construct global basis functions derived from snapshots, which are discrete samples of a trajectory associated with a
particular set of boundary conditions and inputs. Examples of the recent reduced-basis techniques are the Proper Orthogonal Decomposition (POD) and the greedy basis selection process (part of the Empirical Interpolation Method [2]).

Amongst the various reduced basis approximations, POD is optimal in the sense that a certain approximation error concerning the snapshots is minimized. Thus, POD usually gives a better approximation than other approaches with the same dimension. The POD technique is therefore used in this thesis to construct a reduced basis.

**Previous Work on POD**

The POD approach has provided reduced order models of systems in numerous applications such as compressible flow [27], computational fluid dynamics [14, 26, 32], aerodynamics [3], and optimal control [12]. Numerical results indicate POD approximation can provide a reduced order system with substantially lower dimension compared to the original system.

While POD methods have been quite successful in providing reduced order systems, they typically fail to reduce the actual complexity involved with evaluating nonlinear terms. Unless there is special structure, such as a bi-linear form, the evaluation of nonlinear terms has the same complexity as the full order system. Clearly, constructing reduced dimension approximations to the nonlinear terms that actually have complexity proportional to the number of reduced variables is of the highest priority. This thesis addresses this fundamental issue.
Analyses of stability and accuracy of POD appear in many recent works. Han and Park [19] showed that POD is robust to noise and can be used in conjunction with empirical data, which is typically characterized by noise. Prajna [22] provided the condition that guarantees preservation of stability and proposed a stability-preserving POD model reduction scheme. In [13], Kunisch and Volkwein gave rigorous error estimates and a convergence proof for POD applied to parabolic problems. They showed that the error of the reduced system depends not only on the number of POD basis functions, but also on the time discretization of the ordinary differential equation (ODE) arising from applying the Galerkin projection onto the space generated by the POD basis.

The choice of the snapshot ensemble is a crucial factor in constructing a POD basis, and this choice can greatly affect the approximation of the original space of solutions. Recently, in 2007, Kunisch and Volkwein [15] suggested a way to avoid the dependence on the choice of the snapshots in optimal control applications. In 2008, a model-constrained adaptive sampling is proposed in [4] for selecting the snapshots for large-scale systems with high-dimensional parametric input spaces. Although there is no general framework for selecting the snapshots for constructing the POD basis, the method works well in many applications and generally gives better approximations than any other known reduced-basis techniques in general. However, when POD used in conjunction with the Galerkin projection, effective dimension reduction is usually limited to the linear terms, as shown later in this thesis. A system with nonlinearities
needs additional treatment.

1.2.2 Techniques for Nonlinearities

Existing approaches for constructing a reduced order approximation to the nonlinear terms in a time-dependent PDE are mostly based on the schemes used in general function approximation. The common schemes are linearization and piecewise polynomial approximation [1, 5, 24, 6]. These approaches may not be efficient for systems with highly nonlinear functions which generally cannot be approximated well by using only lower degree polynomials. As the degree of the polynomial increases or the number of the piecewise polynomial increases, the computational cost of these methods can get extremely high. Moreover, these methods still face the problem of choosing linearization points [6].

Two alternative techniques based on interpolation schemes are the Empirical Interpolation Method (EIM) proposed by Barrault, Maday, Nguyen, and Patera in [2] and the Best Point Interpolation Method (BPIM) proposed by Nguyen, Patera, and Peraire in [16]. They are developed especially for removing the dependence on the dimension of the original FE discretized system. The approximation of nonlinear term from these methods is in the form of a separation of parameters (time variable) and spatial variable, i.e., linear combination of spatial basis functions with parameter (time) dependent coefficients. EIM and BPIM proposed different procedures for selecting interpolation points to compute the coefficient functions.
EIM selects the points in the spatial domain that explain the variation of the function. It is based on a heuristic greedy selection method, which worked well in general cases as shown in [2]. BPIM selects a set of points that are optimal over an approximate solution space of interest, and not over the space of interest itself. In particular, BPIM selects a set of points which minimize the error between the approximate function and the original function over the discrete set of arbitrary selected parameters \( \{\mu_1, \ldots, \mu_n\} \in \mathcal{D} \), instead of over the continuous parameter domain \( \mathcal{D} \) (which is claimed to be computationally infeasible to solve [16]).

In most numerical results shown in [16], the BPIM gives a relatively small improvement in accuracy of the approximate function from the EIM. The computational complexity of the BPIM is clearly much higher than the EIM, since it must solve a nonlinear constrained optimization problem. This thesis shall focus on EIM and avoid the high computational cost of the BPIM, which may not compensate for the gain in accuracy.

Note that, in this thesis, the term EIM will refer to the EIM procedure for constructing interpolation points\(^1\).

\(^1\)Originally, the paper [2] used the term EIM for its proposed two main methods: (i) the method for constructing a basis set and (ii) the method for constructing interpolation points used in the coefficient function approximation.
Previous work on EIM and BPIM

EIM and BPIM are recently developed methods and therefore they do not have as many demonstrated applications as the POD method. To my knowledge, most of the applications of EIM are found together with the greedy basis selection procedures from [2] in the recent works of Patera’s students, addressing the second order wave equation [30] in 2006, shape design by optimal flow control [28] in 2005, and parabolic partial differential equations [8] in 2005. In 2006, a longer version of [2] with more detailed explanation and numerical results is presented in [9]. In fact, there is book in progress along with a software package called Reduced Basis Approximation and A Posteriori Error Estimation for Parametrized Partial Differential Equations by Patera and Rozza [20], which is based on EIM with the greedy basis selection procedure. This book also includes a comparison study showing that the POD approach is better, but more expensive, than the greedy basis selection procedure.

BPIM is used together with POD to construct reduced order systems for nonlinear parametrized PDEs by Nguyen and Peraire in [17]. The objective of their work is closely related to this thesis. The main difference between their work and this thesis is the treatment of the nonlinearities. Their work uses the POD basis with BPIM for coefficient function approximation for nonlinear terms, whereas this thesis uses a POD basis with EIM for coefficient function approximation of nonlinear terms. Theoretically, BPIM is superior to EIM with respect to accuracy. In practice, BPIM
is far more computationally intense as compared to EIM with a small accuracy improvement. As a result, this thesis shall focus on the technique combining POD and EIM.

Note that EIM cannot be applied directly on a nonlinear term in the FD discretized system due to the continuity assumption in the spatial domain. This thesis therefore extends EIM by introducing its discrete variation, called Discrete Empirical Interpolation Method (DEIM). The formulation of DEIM allows an alternative approach for error analysis to be done rigorously in the aspect of numerical linear algebra. The derivation of this error bound also gives an insight into both DEIM and EIM selection processes. In this thesis, DEIM is used together with POD to approximate the nonlinear terms in FD discretized systems or general ODE systems.

1.3 Thesis Scope and Outline

This thesis evaluates and compares the effectiveness of model reduction techniques for finite difference (FD) and finite element (FE) discretized systems of unsteady nonlinear PDEs. It primarily focuses on (i) the technique combining POD and DEIM, called the POD-DEIM technique, for FD case and (ii) the technique combing POD and EIM, called the POD-EIM technique, for FE case. These techniques are compared with the standard POD-Galerkin method. In the case of FE, another benchmark technique considered here is the precomputing-POD technique, which employs the special (simple) structure of the nonlinearities to remove the dependence on the dimension of the
original discretized system by manipulating the order of computing.

This thesis is organized as follows. Chapter 2 reviews POD and EIM as well as presents an simpler alternative form of EIM algorithm for interpolation points. This chapter also introduces DEIM and the corresponding error bound. Chapter 3 demonstrates the application of POD-DEIM approach on the FD discretized system of a problem in neuron modeling, called FitzHugh-Nagumo system. The corresponding numerical results on accuracy and computational complexity of POD-DEIM reduced system compared with the standard POD reduced system are shown in Chapter 4. Similarly, Chapter 5 describes the general formulation for constructing reduced-order systems using direct POD, POD-EIM, and precomputing approaches for the FE discretized systems of unsteady 1D Burgers' Equation. The corresponding numerical results are illustrated in Chapter 6 with two different initial conditions: one is a simple initial condition with a fixed viscosity parameter; and the other is a more complicated initial condition with varied viscosity parameters in a certain range. Finally, the conclusions with suggested future work are discussed in Chapter 7.
Chapter 2

Methods

This chapter presents the techniques that are used in this thesis for constructing reduced-order models for finite different (FD) and finite element (FE) discretized systems of time-dependent nonlinear PDEs. First, the Galerkin projection scheme for constructing a reduced-order model for dynamical systems is introduced in Section 2.1. In this thesis, the basis of the subspace used in Galerkin projection is assumed to be generated from the Proper Orthogonal Decomposition (POD) which is presented formally in Section 2.2. In general, the reduced-order models derived from the POD-Galerkin scheme give good approximations to the original ones with much lower dimension. However, with general nonlinearities, computational complexities for solving these reduced-order systems usually still depend upon the dimension of their original full-order models. In order to overcome this difficulty, this chapter considers (i) *Discrete Empirical Interpolation Method (DEIM)* for the FD discretized
systems, and (ii) Empirical Interpolation Method (EIM) for the FE discretized systems. DEIM is a discrete variation of EIM proposed in [2]. DEIM has a simple formulation and the derivation of its error bound gives insight into the algorithms for both DEIM and EIM [2] and therefore it is presented first in Section 2.3. EIM with simplified notation and modified formulation is presented later in Section 2.4 for reference.

2.1 Reduced-Order Modeling by Projection Method

Let $X$ be an inner product space with inner product $\langle \cdot, \cdot \rangle_X$ and corresponding norm $\| \cdot \|_X = \sqrt{\langle \cdot, \cdot \rangle_X}$. Let $y : [0, T] \rightarrow X$ where $[0, T]$ is the time interval. Consider a simplified nonlinear dynamical system

$$
\frac{\partial}{\partial t} y(t) = A y(t) + \mathcal{F}(y(t)),
$$

where $A$ is a linear operator on $X$ and $\mathcal{F}$ is a nonlinear function on $X$.

In this chapter, we consider a reduced-order system of (2.1) that is obtained by applying Galerkin projection. In particular, the reduced system are constructed by first projecting the solution onto a subspace spanned by an orthonormal basis $\{ \phi_i \}_{i=1}^k \subset X$, i.e., replace $y(t)$ by $\tilde{y}(t) = \sum_{j=1}^k \tilde{y}_j(t) \phi_j$ and then imposing the conditions of Galerkin projection, i.e., we require $\langle \phi_i, \frac{\partial}{\partial t} \tilde{y}(t) - A \tilde{y}(t) - \mathcal{F}(\tilde{y}(t)) \rangle_X = 0$, for $i = 1, \ldots, k$. That
is,

\[
\langle \phi_i, \frac{\partial}{\partial t} \tilde{y}(t) \rangle_X = \langle \phi_i, A\tilde{y}(t) \rangle_X + \langle \phi_i, F(\tilde{y}(t)) \rangle_X,
\]
\[
\sum_{j=1}^{k} \frac{d}{dt} \tilde{y}_j(t) \langle \phi_i, \phi_j \rangle_X = \sum_{j=1}^{k} \tilde{y}_j(t) \langle \phi_i, A\phi_j \rangle_X + \langle \phi_i, F(\sum_{j=1}^{k} \tilde{y}_j(t) \phi_j) \rangle_X.
\]

(2.2)

(2.3)

The resulting reduced system is then of the form

\[
\frac{d}{dt} \tilde{y}(t) = \tilde{A}\tilde{y}(t) + \tilde{N}(\tilde{y}(t)),
\]

(2.4)

where \( \tilde{y}(t) = [\tilde{y}_1(t), \ldots, \tilde{y}_k(t)]^T \in \mathbb{R}^k \), \( \tilde{A} = [a_{ij}] = [\langle \phi_i, A\phi_j \rangle_X] \in \mathbb{R}^{k \times k} \), \( \tilde{N} \) is a nonlinear term with \( \tilde{N}(\tilde{y}(t))_i = \langle \phi_i, F(\sum_{j=1}^{k} \tilde{y}_j(t) \phi_j) \rangle_X \), for \( i = 1, \ldots, k \).

The projection scheme for constructing a reduced-order system shown above can be applied to both ODEs and time-dependent PDEs with a given initial condition. In the case of ODEs, this thesis particularly focuses on the ones arising from the finite difference discretized system of unsteady nonlinear PDEs as shown in Section 2.3.

In the case of PDEs, the system in (2.4) has to be modified according to the given boundary conditions as shown in a model problem in Chapter 5.

This thesis uses the basis \( \{\phi_i\}_{i=1}^{k} \) from the Proper Orthogonal Decomposition (POD) to construct the reduced system (2.4). The POD generates an optimally ordered, orthonormal basis in the least-squares sense for a given set of theoretical, experimental or computational data, which will be called snapshots in this thesis. The choice of snapshots is crucial, but it is a separate issue and will not be discussed here. The POD is introduced formally next in Section 2.2.
2.2 Proper Orthogonal Decomposition (POD)

POD is a method for constructing a low-dimensional approximation representation of subspaces in Hilbert space. It is essentially the same as the singular value decomposition (SVD) in a finite dimensional space or in Euclidean space. It efficiently extracts the basis elements that contain characteristics of the space of expected solutions of the PDE.

As in Section 2.1, throughout this section, $X$ shall denote a Hilbert space with inner product $\langle \cdot , \cdot \rangle_X : X \times X \to \mathbb{R}$ and with the induced norm $\| \cdot \|_X = \sqrt{\langle \cdot , \cdot \rangle_X}$. Define a snapshot $y$ to be a certain specified element of $X$. Let $\{y_1, y_2, \ldots, y_{n_s}\} \subset X$ be a given set of snapshots and $\mathcal{Y} := \text{span}\{y_1, y_2, \ldots, y_{n_s}\}$ with $r = \text{dim}(\mathcal{Y})$. Typically, a set of snapshots is comprised of approximations to points of a trajectory of the dynamical system at selected instances of time.

Suppose $\{\phi_i\}_{i=1}^r$ is an orthonormal basis for $\mathcal{Y}$. Then, each snapshot $y_j$ for $j = 1, \ldots, n_s$ can be written as

$$y_j = \sum_{i=1}^r \langle y_j, \phi_i \rangle_X \phi_i. \quad (2.5)$$

POD provides a way to construct an orthonormal (ordered) basis of dimension $k$, $\{\phi_i\}_{i=1}^k$ that gives the best approximation of the form

$$\tilde{y}_j = \sum_{i=1}^k \langle y_j, \phi_i \rangle_X \phi_i$$

to the snapshot $y_j$ for all $j = 1, \ldots, n_s$ (in the sense that the mean square error between $\tilde{y}_j$ and $y_j$ is minimized for all $j$). The set $\{\phi_i\}_{i=1}^k$ is called a POD basis.
The following is the formal definition of a POD basis [14].

**Definition 2.2.1** A POD basis of dimension $k < r$ is a set of orthonormal basis \( \{\phi_i\}_{i=1}^k \), which is a solution of

\[
\min_{\{\phi_i\}_{i=1}^k} \sum_{j=1}^n \| y_j - \sum_{i=1}^k \langle y_j, \phi_i \rangle_X \phi_i \|_X^2,
\]

s.t.

\[
\langle \phi_i, \phi_j \rangle_X = \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \quad i, j = 1, \ldots, k. \tag{2.6}
\]

The following theorem provides an explicit form of the POD basis generated by a given set of snapshots in a Hilbert space.

**Theorem 2.2.2** Define a linear operator \( \mathcal{F} : X \to X \) via

\[
\mathcal{F}(w) = \sum_{t=1}^n \langle w, y_t \rangle_X y_t, \quad \text{for } w \in X. \tag{2.7}
\]

Then the POD basis is the set of \( k \) eigenfunctions of \( \mathcal{F} \) corresponding to the \( k \) largest eigenvalues of \( \mathcal{F} \). That is, the POD basis is \( \{\phi_i\}_{i=1}^k \) where \( \phi_i \in X \) satisfies

\[
\sum_{t=1}^n \langle \phi_i, y_t \rangle_X y_t = \phi_i \lambda_i
\]

for \( i = 1, \ldots, k \) and \( \lambda_1 \geq \cdots \geq \lambda_k \) are the \( k \) largest eigenvalues of \( \mathcal{F} \).

**Proof:** To prove that \( \{\phi_i\}_{i=1}^k \) is a POD basis, we have to show that it is a solution to (2.6). To find the solution of this minimization problem (2.6), first form the Lagrange function:

\[
L(\phi_1, \ldots, \phi_k, \lambda_{11}, \ldots, \lambda_{ij}, \ldots, \lambda_{kk}) = E(\phi_1, \ldots, \phi_k) + \sum_{i,j=1}^k \lambda_{ij} (\langle \phi_i, \phi_j \rangle_X - \delta_{ij}),
\]
where $E$ is the objective function of (2.6) and $\lambda_{ij}$ are the Lagrange multipliers for $i, j = 1, \ldots, k$. The Karush-Kuhn-Tucker (KKT) necessary conditions for this problem gives:

1. $\frac{\partial}{\partial \phi_i} L = 0 \iff \begin{cases} \sum_{j=1}^{n} y_j \langle y_j, \phi_i \rangle_X = \lambda_{ii} \phi_i \\ \lambda_{ij} = 0, \text{ if } i \neq j \end{cases}$.

2. $\frac{\partial}{\partial \lambda_{ij}} L = 0 \iff \langle \phi_i, \phi_j \rangle_X = \delta_{ij}$

For condition (1), by setting $\lambda_i = \lambda_{ii}$, each element $\phi_i$ in the POD basis (optimal solution) then has to be an eigenfunction of $\mathcal{F}$ with corresponding eigenvalue $\lambda_i$. Condition (2) follows from the orthogonality of the eigenvectors of symmetric linear operator $\mathcal{F}$. If we consider also the objective function or the error of the POD basis in approximating the snapshots as shown in Lemma 2.2.5,

$$\sum_{j=1}^{n_s} \| y_j - \sum_{i=1}^{k} \langle y_j, \phi_i \rangle_X \phi_i \|_X^2 = \sum_{i=k+1}^{r} \lambda_i,$$

the POD basis (optimal solution) is the set of $k$ eigenfunctions of $\mathcal{F}$ corresponding to the $k$ largest eigenvalues of $\mathcal{F}$ (since we want to minimize the error which is the sum of the eigenvalues corresponding to the remaining unused eigenfunctions).

From Theorem 2.2.2, a POD basis can be obtained by solving for eigenfunctions of a linear operator. Alternatively, the following corollary gives a practical way for computing the POD basis by solving for eigenvectors in $\mathbb{R}^n$, instead of solving for eigenfunctions in space $X$, which can be infinite dimensional. This approach is often called the method of snapshots.
Corollary 2.2.3 Define a linear operator \( \mathcal{L} : \mathbb{R}^{n_s} \to \mathbb{R}^{n_s} \),

\[
\mathcal{L} \equiv [(y_i,y_j)_X] \in \mathbb{R}^{n_s \times n_s},
\]

for \( i, j = 1, \ldots, n_s \). Then a POD basis of dimension \( k < r \) is provided by a set of orthonormal basis functions \( \{\phi_i\}_{i=1}^k \), with

\[
\phi_i = \frac{1}{\sqrt{\lambda_i}} \sum_{\ell=1}^{n_s} (v_i)_\ell y_\ell,
\]

for \( i = 1, \ldots, k \), where \( k \leq r \) and \( v_i \in \mathbb{R}^{n_s} \) is the \( i^{th} \) eigenvector of \( \mathcal{L} \) with corresponding eigenvalue \( \lambda_i \), i.e.

\[
\mathcal{L}v_i = v_i\lambda_i,
\]

for \( i = 1, \ldots, k \) and \( \lambda_1 \geq \cdots \geq \lambda_k \) are the \( k \) largest eigenvalues of \( \mathcal{L} \).

Proof: First note that, since \( \mathcal{L} \in \mathbb{R}^{n_s \times n_s} \) is a symmetric positive semi-definite matrix, all the eigenvalues are non-negative. Also, since \( k < r = \text{rank}(\mathcal{Y}) = \text{rank}(\mathcal{L}) \), then the first \( k \) largest eigenvalues of \( \mathcal{L} \) are positive (\( \lambda_1, \ldots, \lambda_k > 0 \)). Therefore (2.9) is well-defined.

Substituting \( \phi_i \) of (2.9) into \( \mathcal{F} \) provides \( \lambda_i \) and \( \phi_i \) as defined in (2.9) as the eigenvalue and the corresponding eigenfunction of \( \mathcal{F} \). Hence, this Corollary follows directly from Theorem 2.2.2.

\[ \square \]

Lemma 2.2.4 Define a linear operator \( \mathcal{L} \):

\[
\mathcal{L} \equiv [(y_i,y_j)_X] \in \mathbb{R}^{n_s \times n_s},
\]
Then \( L \) is a Hermitian positive semi-definite matrix.

**Proof:** Recall from the beginning of this section that the inner product is defined as \( \langle \cdot, \cdot \rangle_X : X \times X \to \mathbb{R} \). This implies that \( \overline{\langle y_j, y_i \rangle_X} = \langle y_j, y_i \rangle_X \). So, by the conjugate symmetry of the inner product, it is clear that \( L \) is Hermitian because \( L_{ij} = \langle y_i, y_j \rangle_X = \overline{\langle y_j, y_i \rangle_X} = \langle y_j, y_i \rangle_X = L_{ji} \). For the positive semi-definiteness, let \( w \) be a non-zero vector in \( \mathbb{R}^{n_x} \), and then

\[
 w^T L w = \sum_{i,j=1}^{n_x} w_i w_j \langle y_i, y_j \rangle_X = \left( \sum_{i=1}^{n_x} w_i y_i, \sum_{j=1}^{n_x} w_j y_j \right)_X = \left\| \sum_{i=1}^{n_x} w_i y_i \right\|_X^2 \geq 0.
\]

\[ \square \]

**Lemma 2.2.5** Define a linear operator \( \mathcal{F} : \) for \( w \in X \),

\[
 \mathcal{F}(w) = \sum_{i=1}^{n_x} \langle w, y_i \rangle_X y_i. \tag{2.11}
\]

Let \( \phi_i \in X \) be an eigenfunction corresponding to the eigenvalue \( \lambda_i \) of \( \mathcal{F} \), i.e.,

\[
 \mathcal{F}(\phi_i) = \phi_i \lambda_i, \text{ or } \sum_{i=1}^{n_x} \langle \phi_i, y_i \rangle_X y_i = \phi_i \lambda_i
\]

for \( i = 1, \ldots, r \) where \( r \) is the rank of the operator \( \mathcal{F} \). Then, for \( k < r \),

\[
 \sum_{j=1}^{n_x} \| y_j - \sum_{i=1}^{k} \langle y_j, \phi_i \rangle_X \phi_i \|_X^2 = \sum_{i=k+1}^{r} \lambda_i. \tag{2.12}
\]

**Proof:** First note that, since \( \text{rank}(\mathcal{F}) = \text{rank}(Y) = r \), then \( \mathcal{F} \) has \( r \) orthonormal eigenfunctions \( \{ \phi_i \}_{i=1}^r \) (\( \mathcal{F} \) is Hermitian, i.e., it can be shown that \( \langle \mathcal{F}(w_1), w_2 \rangle_X = \langle w_1, \mathcal{F}(w_2) \rangle_X \), so eigenfunctions of \( \mathcal{F} \) are orthonormal) and these eigenfunctions
span the subspace \( \mathcal{V} = \text{span}(\{y_1, \ldots, y_{n_s}\}) \). Therefore each \( y_i \) can then be written in the expansion of \( \{\phi_i\}_{i=1}^r \) as

\[
y_j = \sum_{i=1}^r \langle y_j, \phi_i \rangle_X \phi_i,
\]

for \( j = 1, \ldots, n_s \). Also, since, \( (\lambda_i, \phi_i) \) is an eigenpair of \( \mathcal{F} \) for \( i = 1, \ldots, r \), i.e.

\[
\phi_i \lambda_i = \sum_{j=1}^{n_s} \langle y_j, \phi_i \rangle_X y_j,
\]

then taking the inner product with \( \phi_i \) on both sides of above equation and using orthonormality \( \langle \phi_i, \phi_j \rangle_X = \delta_{ij} \) gives

\[
\lambda_i = \sum_{j=1}^{n_s} \langle y_j, \phi_i \rangle_X^2.
\]

Hence, substituting \( y_j \) from (2.13), applying again orthonormality, \( \langle \phi_i, \phi_j \rangle_X = \delta_{ij} \), and using (2.14) gives

\[
\sum_{j=1}^{n_s} \| y_j - \sum_{i=1}^k \langle y_j, \phi_i \rangle_X \phi_i \|^2_X = \sum_{j=1}^{n_s} \sum_{i=k+1}^r \langle y_j, \phi_i \rangle_X^2 = \sum_{i=k+1}^r \lambda_i.
\]

\[
\square
\]

To summarize, given a set of snapshots: \( y_1, \ldots, y_{n_s} \in X \), where \( X \) is a Hilbert Space, the following is the steps for computing the POD basis of dimension \( n_m \leq n_s \) for this set of snapshots.
Steps for Computing the POD Basis

1. Construct a symmetric linear operator

\[ \mathcal{L} \equiv [(y_i, y_j)] \in \mathbb{R}^{n_s \times n_s}, \tag{2.15} \]

where \( y_i \) and \( y_j \) are the snapshots for \( i, j = 1, \ldots, n_s \).

2. Find \( k \leq n_s \) eigenvectors \( v_i \in \mathbb{R}^{n_s} \) of \( \mathcal{L} \) and corresponding eigenvalues \( \lambda_i \):

\[ \mathcal{L} v_i = v_i \lambda_i, \]

for \( i = 1, \ldots, k \) and \( \lambda_1 \geq \cdots \geq \lambda_k \) are the largest \( k \) eigenvalues of \( \mathcal{L} \).

3. Then each POD basis functions \( \phi_i \), for \( i = 1, \ldots, k \), is given by

\[ \phi_i = \frac{1}{\sqrt{\lambda_i}} \sum_{\ell=1}^{n_s} (v_i)_{\ell} y_\ell. \]

To write each POD basis function \( \phi_i \) in a matrix/vector form, let \( \mathcal{L}V = V \Lambda \) be the eigenvalue decomposition of \( \mathcal{L} \) where \( \Lambda = \text{diag}(\lambda_1, \ldots, \lambda_{n_s}) \) with \( \lambda_1 \geq \cdots \geq \lambda_{n_s} \geq 0 \).

Let \( V_k = [v_1, \ldots, v_k] \in \mathbb{R}^{n_s \times k} \) be the first \( k \) columns of \( V \) and \( \Lambda_k = \text{diag}(\lambda_1, \ldots, \lambda_k) \in \mathbb{R}^{k \times k} \).

Let \( Y \equiv [y_1, \ldots, y_{n_s}] \) be the 1-by-\( n_s \) matrix of the snapshots. Then, we can write 1-by-\( k \) matrix of POD basis functions: \( \Phi \equiv [\phi_1, \ldots, \phi_k] \) as

\[ \Phi = Y V_k \Lambda_k^{-\frac{1}{2}}, \tag{2.16} \]

where \( \Lambda_k^{-\frac{1}{2}} = \text{diag}(\frac{1}{\sqrt{\lambda_1}}, \ldots, \frac{1}{\sqrt{\lambda_k}}) \in \mathbb{R}^{k \times k} \).
In the case of Euclidean space, i.e. snapshots $y_1, \ldots, y_n, \in \mathbb{R}^n$, POD basis can be obtained directly from the left singular vectors of the snapshot matrix: $Y = [y_1, \ldots, y_n] \in \mathbb{R}^{n \times n}$.

POD can be used to construct a reduced-order system of a given system of discretized PDEs by first solving the original discretized system (full-order system) to obtain a set of solution snapshots, then constructing a POD basis as described above from these snapshot solutions, and finally projecting the original discretized system onto the first $k$ POD basis, where $k$ is the reduced dimension. Snapshot selection is a separate issue that is not addressed here.

As noted, although POD is an efficient method for extracting a basis for generating a reduced-order model, the computational complexity of the nonlinear term in this POD reduced-order model still depends on the dimension of the original full-order model. The next two sections discuss approaches to resolve this dependence in the nonlinear terms.

### 2.3 Discrete Empirical Interpolation Method (DEIM)

Recall the dynamical system (2.1) in Section 2.1. In the case of a system of nonlinear ODEs arising from finite difference discretization of a nonlinear PDE, (2.1) can be written as

$$
\frac{d}{dt}y(t) = Ay(t) + F(y(t)),
$$

(2.17)
where \( y : [0, T] \rightarrow \mathbb{R}^n, A \in \mathbb{R}^{n \times n} \) is a constant matrix and \( F \) is a nonlinear function evaluated at \( y(t) \) componentwise. Suppose \( V_k \in \mathbb{R}^{n \times k} \) is the matrix consisting of the first \( k \) dominant left singular vectors of the solution snapshots from (2.17): \( y_1, \ldots, y_{n_s} \), where \( y_j \) is the approximate solution of \( y(t_j) \), for \( t_j \in [0, T], j = 1, \ldots, n_s \). That is, the columns of \( V_k \) are the POD basis.

As shown in Section 2.1, a reduced order system of (2.17) can be obtained by projecting the solution onto the space spanned by columns of \( V_k \in \mathbb{R}^{n \times k} \), i.e., substitute \( y(t) \) in (2.17) by \( V_k \bar{y}(t) \), for \( k < n \) and applying the Galerkin projection onto \( V_k \) to the system (2.17). Then the reduced system is of the form

\[
\frac{d}{dt} \bar{y}(t) = V_k^T A V_k \bar{y}(t) + V_k^T F(V_k \bar{y}(t)).
\]  

(2.18)

Note that the computational complexity of the nonlinear term still depends on the dimension \( n \) of the original system. To illustrate this dependency, consider the nonlinear function of the reduced system (2.18) in a simplified notation given by

\[
f(t) := F(V_k \bar{y}(t)),
\]

(2.19)

so that we can define the nonlinear term of the reduced system (2.18) as

\[
N(t) := \underbrace{V_k^T}_{k \times n} \underbrace{f(t)}_{n \times 1}.
\]

(2.20)

Computing the nonlinear term \( N(t) \) requires \( n \) function evaluations of \( f(t) \) and matrix-vector multiplication of order \( n \). Therefore, computational complexity for solving the reduced system (2.18) still depends on the dimension \( n \) of the original system. As a
result, the dimension of system (2.18) is not truly reduced, and hence, solving this system might still be as inefficient as solving the original system. One way to recover from this inefficiency is to approximate the nonlinear function $f(t)$ by projecting it onto the subspace spanned by a basis $\{u_1, \ldots, u_m\} \subset \mathbb{R}^n$ of dimension $m < n$ so that the approximation is of the form

$$f(t) \approx Uc(t), \quad (2.21)$$

where $U = [u_1 \ldots u_m] \in \mathbb{R}^{n \times m}$ and $c(t)$ is the corresponding coefficient vector. From (2.21), the nonlinear term in (2.20) can be approximated by

$$N(t) \approx \underbrace{V_k^T U}_{\text{precomputed: } k \times m} \underbrace{c(t)}_{\text{m \times 1}}. \quad (2.22)$$

The term $V_k^T U$ does not depend on $t$ and therefore it can be precomputed before solving the ODE. Hence, by using this approximation, the computational complexity of the nonlinear term is independent of $n$. Note that, similar to the basis of the subspace used in Galerkin projection, the basis $\{u_1, \ldots, u_m\} \subset \mathbb{R}^n$ for the nonlinear function can be constructed effectively by applying the POD on the nonlinear snapshots $\{f(t_1), \ldots, f(t_{n_s})\}$, where $n_s$ is the number of snapshots. E.g., the nonlinear snapshots of the system (2.17) are $F(y_1), \ldots, F(y_{n_s})$ which are obtained directly from the snapshot solutions $y_1, \ldots, y_{n_s}$.

Interpolation can be used to determine the coefficient vector $c(t)$ by selecting $m$ rows, $\varphi_1, \ldots, \varphi_m$, of the overdetermined linear system:
to form $m$-by-$m$ linear system:

\[
\begin{bmatrix}
    f_1(t) \\
    \vdots \\
    f_n(t)
\end{bmatrix}
= \begin{bmatrix}
    u_{11} & \cdots & u_{1m} \\
    \vdots & \ddots & \vdots \\
    u_{n1} & \cdots & u_{nm}
\end{bmatrix}
\begin{bmatrix}
    c_1(t) \\
    \vdots \\
    c_m(t)
\end{bmatrix},
\]
\[ (2.23) \]

\[
\begin{bmatrix}
    f_{p_1}(t) \\
    \vdots \\
    f_{p_m}(t)
\end{bmatrix}
= \begin{bmatrix}
    u_{p_11} & \cdots & u_{p_1m} \\
    \vdots & \ddots & \vdots \\
    u_{p_m1} & \cdots & u_{p_m m}
\end{bmatrix}
\begin{bmatrix}
    c_1(t) \\
    \vdots \\
    c_m(t)
\end{bmatrix},
\]
\[ (2.24) \]

or

\[ U_{\varphi} \mathbf{c}(t) = f_{\varphi}(t), \]
\[ (2.25) \]

where

\[ U_{\varphi} = \begin{bmatrix}
    u_{p_11} & \cdots & u_{p_1m} \\
    \vdots & \ddots & \vdots \\
    u_{p_m1} & \cdots & u_{p_m m}
\end{bmatrix} \quad \text{and} \quad f_{\varphi}(t) = \begin{bmatrix}
    f_{p_1}(t) \\
    \vdots \\
    f_{p_m}(t)
\end{bmatrix}. \]
\[ (2.26) \]

If $U_{\varphi}$ is invertible, then we can uniquely determine $\mathbf{c}(t)$ from (2.25) by

\[ \mathbf{c}(t) = U_{\varphi}^{-1} f_{\varphi}(t), \]
\[ (2.27) \]
and the approximation from (2.21) becomes

\[ f(t) \approx Uc(t) = UU_\varphi^{-1}f_\varphi(t). \quad (2.28) \]

An alternative way to write \( U_\varphi \) and \( f_\varphi \) in terms of the original matrix \( U \) and \( f \) is to introduce a matrix

\[ P = [e_{p_1}, \ldots, e_{p_m}] \in \mathbb{R}^{n \times m}, \text{ where } e_{p_i} = [0, \ldots, 0, \underbrace{1}_{p_i}, 0, \ldots, 0]^T \in \mathbb{R}^n. \quad (2.29) \]

I.e., the vector \( e_{p_i} \) has one in the \( p_i \)-th entry and zero in the remaining entries. Then

\[ U_\varphi = P^T U \quad \quad f_\varphi(t) = P^T f(t), \quad (2.30) \]

and from (2.28),

\[ f(t) \approx U(P^TU)^{-1}P^T f(t). \quad (2.31) \]

Note that if we go back to the original form of the nonlinear function \( f(t) = F(V_k\tilde{y}(t)) \) in (2.19), then \( f_\varphi(t) = P^T f(t) = P^T F(V_k\tilde{y}(t)) = F(P^TV_k\tilde{y}(t)) \). The last equality follows from the fact that the function \( F \) evaluates componentwise at its input. Therefore, from (2.31), the final approximation of the nonlinear function and the nonlinear term in (2.18) can be written as

\[ F(V_k\tilde{y}(t)) \approx U(P^TU)^{-1}F(P^TV_k\tilde{y}(t)) \quad (2.32) \]

and

\[ V_k^T F(V_k\tilde{y}(t)) \approx V_k^T U(P^TU)^{-1} F(P^TV_k\tilde{y}(t)). \quad (2.33) \]
With this approximation, the complexity for computing the nonlinear term of the reduced system in each time step is now independent of the dimension \( n \) of the original full-order system. From (2.33), the only unknown needs to be specified in the final approximation in (2.32), is the matrix \( P \), or the indices \( \varphi_1, \ldots, \varphi_m \) used in (2.24). Next subsection introduces an algorithm, called *Discrete Empirical Interpolation Method (DEIM)*, for selecting these indices. This algorithm is based on an inductive selection process. The associate error bound of this approximation is given in Section 2.3.2.

### 2.3.1 DEIM: Algorithm for Interpolation Indices

DEIM is a discrete variant of the Empirical Interpolation Method (EIM) proposed in [2] for constructing an approximation of a non-affine parametrized function with spatial variable defined in a continuous bounded domain \( \Omega \). The DEIM algorithm treats the *continuous* domain \( \Omega \) as a *finite set of discrete points* in \( \Omega \). The DEIM algorithm selects an index at each iteration to limit growth of an error bound. This provides a derivation of a global error bound as presented in Section 2.3.2.

The notation \( \max \) in Algorithm 1 is the same as the function \( \max \) in MATLAB. Thus, \( |\rho| \varphi_\ell = \max_1 \{|r|\} \) implies \( |\rho| = |r_{\ell\ell}| = \max_{i=1, \ldots, n}\{|r_i|\} \), with the smallest index taken in case of a tie. Note that we define \( \rho = r_{\ell\ell} \) in each iteration \( \ell = 1, \ldots, m \).

From Algorithm 1, the DEIM procedure constructs a set of indices inductively on the input basis. Figure 2.1 illustrates the selection procedure in Algorithm 1 for DEIM interpolation indices. The order of the input basis \( \{u_\ell\}_{\ell=1}^m \) according to the
Algorithm 1: DEIM

**INPUT:** \( \{u_\ell\}_{\ell=1}^m \subset \mathbb{R}^n \) linearly independent

**OUTPUT:** \( \tilde{\varphi} = [\varphi_1, \ldots, \varphi_m]^T \in \mathbb{R}^m \)

1: \( ||\varphi_1|| = \max\{|u_1|\} \)

2: \( U = [u_1], P = [e_{p1}], \tilde{\varphi} = [\varphi_1] \)

3: for \( \ell = 2 \) to \( m \) do

4: Solve \( (P^T U)c = P^T u_\ell \) for \( c \)

5: \( r = u_\ell - Uc \)

6: \( ||\varphi_\ell|| = \max\{|r|\} \)

7: \( U \leftarrow [U \ u_\ell], P \leftarrow [P \ e_{p\ell}], \tilde{\varphi} \leftarrow \begin{bmatrix} \tilde{\varphi} \\ \varphi_\ell \end{bmatrix} \)

8: end for
Figure 2.1: Illustration of the selection process of indices in Algorithm 1 for the DEIM approximation. The input basis vectors are the first 6 eigenvectors of the discrete Laplacian. From the plots, $u = u_\ell$, $Uc$ and $r = u_\ell - Uc$ are defined as in the iteration $\ell$ of Algorithm 1.
dominant singular values is important and an error analysis indicates that the POD basis is a suitable choice for this algorithm. The process starts from selecting the first interpolation index $\varphi_1 \in \{1, \ldots, n\}$ corresponding to the entry of the first input basis $\mathbf{u}_1$ with largest magnitude. The remaining interpolation indices, $\varphi_\ell$ for $\ell = 2, \ldots, m$, are selected so that each of them corresponds to the entry with the largest magnitude of the residual $\mathbf{r} = \mathbf{u}_\ell - \mathbf{Uc}$ from line 5 of Algorithm 1. The term $\mathbf{r}$ can be viewed as the residual or the error between the input basis $\mathbf{u}_\ell$ and its approximation $\mathbf{Uc}$ from interpolating the basis $\{\mathbf{u}_1, \ldots, \mathbf{u}_{\ell-1}\}$ at the indices $\varphi_1, \ldots, \varphi_{\ell-1}$ in line 4 of Algorithm 1. Hence, $r_{\varphi_i} = 0$ for $i = 1, \ldots, \ell - 1$. However, the linear independence of the input basis $\{\mathbf{u}_\ell\}_{\ell=1}^m$ guarantees that, in each iteration, $\mathbf{r}$ is a nonzero vector. That is, $\rho$ and hence $r_{\varphi_\ell}$ are also nonzero for each selected interpolation index $\varphi_\ell$ in iteration $\ell$. In particular, suppose $\rho = 0$. Then, from line 6 of Algorithm 1,

$$\|\mathbf{u}_\ell - \mathbf{Uc}\|_{\infty} = \|\mathbf{r}\|_{\infty} = |\rho| = 0,$$

which implies that $\mathbf{u}_\ell = \mathbf{Uc}$ or $\mathbf{u}_\ell \in \text{span}\{\mathbf{u}_1, \ldots, \mathbf{u}_{\ell-1}\}$ and this contradicts the fact that $\{\mathbf{u}_i\}_{i=1}^m$ is linearly independent. This further implies that the output indices $\{\varphi_i\}_{i=1}^m$ are non-repeated. In particular, at iteration $\ell$, if the selected interpolation indices $\varphi_\ell = \varphi_j$ for some $j \in \{1, \ldots, \ell - 1\}$, then $\rho = r_{\varphi_\ell} = 0$ which is a contradiction as shown for (2.34). Also, these indices are hierarchical in the sense that if we wish to increase the dimension of the approximation from $m$ to $m + 1$, where $m < n$, then the first $m$ selected indices are the same as $\varphi_1, \ldots, \varphi_m$ and therefore they can be used to determine the index $\varphi_{m+1}$ without recomputing the initializing step and the
iterations 2 to \( m \). Moreover, Algorithm 1 is well-defined, since \( P^T U \) is non-singular in each iteration \( \ell \) as shown next.

**Claim 2.3.1** Let \( U = [u_1, \ldots, u_\ell] \in \mathbb{R}^{n \times \ell} \), and \( P = [e_{p_1}, \ldots, e_{p_\ell}] \in \mathbb{R}^{n \times \ell} \) be the matrices from Algorithm 1. Then \( P^T U \in \mathbb{R}^{\ell \times \ell} \) is nonsingular for each \( \ell = 1, \ldots, m \).

**Proof:**

This can be shown by induction on iteration \( \ell \). For \( \ell = 1 \), \( P^T U \) is nonsingular, since \( (P^T U)^{-1} = 1/\|u_1\|_\infty \in \mathbb{R} \) from the initialization step. It is clear that \( \|u_1\|_\infty \neq 0 \) otherwise \( u_1 \) is a zero vector which contradicts the linear independence of the input basis. For \( \ell = 2, \ldots, m \), to simplify notation, at iteration \( \ell \), define

\[
\begin{align*}
\tilde{U} & = [u_1, \ldots, u_{\ell-1}] \in \mathbb{R}^{n \times (\ell-1)}, \quad \tilde{P} = [e_{p_1}, \ldots, e_{p_{\ell-1}}] \in \mathbb{R}^{n \times (\ell-1)}, \\
u & = u_\ell \in \mathbb{R}^n, \quad p = e_{p_\ell} \in \mathbb{R}^n, \\
U & = [\tilde{U} \ u] \in \mathbb{R}^{n \times \ell}, \quad P = [\tilde{P} \ p] \in \mathbb{R}^{n \times \ell}.
\end{align*}
\]  

(2.35)

Assume that \( \tilde{P}^T \tilde{U} \in \mathbb{R}^{(\ell-1) \times (\ell-1)} \) is nonsingular. It is next shown that \( P^T U \in \mathbb{R}^{\ell \times \ell} \) is also nonsingular. First note that, from Algorithm 1, Step 4 gives \( \tilde{P}^T u = (\tilde{P}^T \tilde{U})c \); Steps 5 and 6 imply \( p^T u = (p^T \tilde{U})c + \rho \). Then \( P^T U \) can be written as

\[
P^T U = \begin{bmatrix} \tilde{P}^T \tilde{U} & \tilde{P}^T u \\ p^T \tilde{U} & p^T u \end{bmatrix} = \begin{bmatrix} \tilde{P}^T \tilde{U} & (\tilde{P}^T \tilde{U})c \\ p^T \tilde{U} & (p^T \tilde{U})c + \rho \end{bmatrix}. \tag{2.36}
\]

The following shows that \( (P^T U)d = 0 \) implies \( d = 0 \in \mathbb{R}^{\ell} \). Write \( d = \begin{bmatrix} d \\ \delta \end{bmatrix} \in \mathbb{R}^{\ell} \).
where \( \bar{d} \in \mathbb{R}^{\ell-1} \) and \( \delta \in \mathbb{R} \). Then
\[
0 = (P^T U) \bar{d} = \begin{bmatrix}
\bar{P}^T \bar{U} & (\bar{P}^T \bar{U}) c \\
p^T \bar{U} & (p^T \bar{U}) c + \rho
\end{bmatrix}
\begin{bmatrix}
\bar{d} \\
\delta
\end{bmatrix}.
\] (2.37)

From (2.37) and the fact that \( \bar{P}^T \bar{U} \) is nonsingular by the inductive hypothesis,
\[
(\bar{P}^T \bar{U}) \bar{d} + (\bar{P}^T \bar{U}) c \delta = 0 \quad \Rightarrow \quad (\bar{P}^T \bar{U})(\bar{d} + \delta c) = 0 \quad \Rightarrow \quad \bar{d} + \delta c = 0.
\] (2.38)

Also, from (2.37), (2.38), and from the fact that \( \rho \neq 0 \) as noted earlier, it follows that
\[
(p^T \bar{U}) \bar{d} + (p^T \bar{U}) \delta = 0 \quad \Rightarrow \quad (p^T \bar{U}) (\bar{d} + \delta c)^T + \rho \delta = 0 \quad \Rightarrow \quad \rho \delta = 0 \quad \Rightarrow \quad \delta = 0.
\] (2.39)

Hence, from (2.38) and (2.39), \( 0 = \bar{d} + \delta c = \bar{d} \). That is, \( \bar{d} = \begin{bmatrix}
\bar{d} \\
\delta
\end{bmatrix} = 0. \)

\( \square \)

To summarize, the DEIM approximation is given formally as follows.

**Definition 2.3.2** Let \( f : \mathcal{D} \rightarrow \mathbb{R}^n \) be a nonlinear vector-valued function with \( \mathcal{D} \subset \mathbb{R}^d \), for some positive integer \( d \). Let \( \{u_i\}_{i=1}^m \subset \mathbb{R}^n \) be a linearly independent set, for \( m = 1, \ldots, n \). For \( \tau \in \mathcal{D} \), the DEIM approximation of order \( m \) for \( f(\tau) \) in the space spanned by \( \{u_i\}_{i=1}^m \) is given by
\[
\hat{f}(\tau) := U(P^T U)^{-1} P^T f(\tau),
\] (2.40)

where \( U = [u_1, \ldots, u_m] \in \mathbb{R}^{n \times m} \) and \( P = [e_{p_1}, \ldots, e_{p_m}] \in \mathbb{R}^{n \times m} \) with \( \{\varphi_1, \ldots, \varphi_m\} \) being the output from Algorithm 1 with the input basis \( \{u_i\}_{i=1}^m \).
Note that, in this thesis, $\tau = t$. However, in general, $\tau$ can be considered as any parameter in the nonlinear term of the discretized system of a PDE. Notice that $\hat{f}$ in (2.40) is indeed an interpolation approximation for the original function $f$, since $\hat{f}$ is exact at the interpolation indices; i.e., for $\tau \in \mathcal{D}$,

$$P^T \hat{f}(\tau) = P^T (U(P^T U)^{-1} P^T f(\tau)) = (P^T U)(P^T U)^{-1} P^T f(\tau) = P^T f(\tau).$$

Notice also that the DEIM approximation is uniquely determined by the projection basis $\{u_i\}_{i=1}^m$ (the uniqueness follows from the invertibility of $P^T U$ in Lemma 2.3.1). This basis not only specifies the projection subspace used in the approximation, but also determines the interpolation indices used for computing the coefficient of the approximation. Hence, the choice of projection basis can greatly affect the accuracy of the approximation in (2.40) as shown also in the error bound of the DEIM approximation (2.43) in the next section. As noted, proper orthogonal decomposition (POD) introduced in Section 2.2 is an efficient method for constructing this projection basis, since it provides a set of optimal global basis that captures the dynamic of the space of nonlinear function. POD basis is therefore used in this paper for constructing a projection basis.

The selection of the interpolation points is basis dependent. However, once the set of DEIM interpolation indices $\{\varphi_\ell\}_{\ell=1}^m$ is determined from $\{u_i\}_{i=1}^m$, the DEIM approximation is independent of the choice of basis spanning the space $\text{Range}(U)$. In particular, let $\{q_\ell\}_{\ell=1}^m$ be any basis for $\text{Range}(U)$. Then

$$U(P^T U)^{-1}P^T f(\tau) = Q(P^T Q)^{-1}P^T f(\tau),$$

(2.41)
where \( Q = [q_1, \ldots, q_m] \in \mathbb{R}^{n \times m} \). To verify (2.41), note that \( \text{Range}(U) = \text{Range}(Q) \) so that \( U = QR \) for some nonsingular matrix \( R \in \mathbb{R}^{m \times m} \). This substitution gives

\[
U(P^T U)^{-1} P^T f(\tau) = (QR)(P^T Q R)^{-1} P^T f(\tau) = Q(P^T Q)^{-1} P^T f(\tau).
\]

### 2.3.2 Error Bound for DEIM

This section provides an error bound for the DEIM approximation. The bound is obtained recursively on the magnification factor of the best 2-norm approximation error. This error bound is given formally as follows.

**Lemma 2.3.3** Let \( f \in \mathbb{R}^n \) be an arbitrary vector. Let \( \{u_i\}_{i=1}^m \subset \mathbb{R}^n \) be a given orthonormal set of vectors. From Definition 2.3.2, the DEIM approximation of order \( m \) for \( f \) in the space spanned by \( \{u_i\}_{i=1}^m \) is

\[
\hat{f} = U(P^T U)^{-1} P^T f,
\]

where \( U = [u_1, \ldots, u_m] \in \mathbb{R}^{n \times m} \) and \( P = [e_{p_1}, \ldots, e_{p_m}] \in \mathbb{R}^{n \times m} \) with \( \{\varphi_1, \ldots, \varphi_m\} \) being the output from Algorithm 1 with the input basis \( \{u_i\}_{i=1}^m \). An error bound for \( \hat{f} \) is then given by

\[
\|f - \hat{f}\|_2 \leq \|P^T U\|_2 \|e_*(f)\| \leq C \|e_*(f)\|
\]

(2.43)

where

\[
e_*(f) = \|(I - UU^T)f\|_2
\]

(2.44)
is the error of the best 2-norm approximation for $f$ from the space $\text{Range}(U)$, and $C$ is a constant obtained recursively from the DEIM iteration as follows:

For $m = 1$,

$$C_1 = \frac{1}{|e_{\mu_1}^T u_1|} = \|u_1\|_\infty^{-1}. \tag{2.45}$$

For $\ell = 2, \ldots, m$,

$$C_\ell = \left[1 + \sqrt{2n}\right] C_{\ell-1}. \tag{2.46}$$

Finally, $C = C_m$.

Proof:

This proof provides motivation for DEIM selection process in terms of minimizing the local error growth for the approximation.

Consider the DEIM approximation $\hat{f}$ given by (2.42). We wish to determine a bound for the error $\|f - \hat{f}\|_2$ in terms of the optimal 2-norm approximation for $f$ from $\text{Range}(U)$. This best approximation is given by

$$f_* = UU^T f, \tag{2.47}$$

which minimizes the error $\|f - \hat{f}\|_2$ over $\text{Range}(U)$. Consider

$$f = (f - f_*) + f_* = w + f_*, \tag{2.48}$$

where $w = f - f_* = (I - UU^T) f$. Define the projector $P = U(P^T U)^{-1} P^T$. From (2.40) and (2.48),

$$\hat{f} = Pf = P(w + f_*) = Pw + Pf_* = Pw + f_* \tag{2.49}$$
Equations (2.48) and (2.49) imply $f - \hat{f} = (I - P)w$ and

$$
\|f - \hat{f}\|_2 = \|(I - P)w\|_2 \leq \|I - P\|_2 \|w\|_2
$$

(2.50)

Note that

$$
\|I - P\|_2 = \|P\|_2 = \|U(P^T U)^{-1}P^T\|_2 \leq \|U\|_2 \|(P^T U)^{-1}\|_2 \|P^T\|_2 = \|(P^T U)^{-1}\|_2.
$$

(2.51)

The first equality in (2.51) follows from the fact that $\|I - P\|_2 = \|P\|_2$, for any projector $P \neq 0$ or $I$ (see [29]). The last equality in (2.51) follows from the fact that $\|U\|_2 = \|P^T\|_2 = 1$, since each of the matrices $U$ and $P$ has orthonormal columns.

Note that $\delta_\ast(f) := \|w\|_2$ is the minimum 2-norm error for $f$, defined in (2.47). From (2.51), the bound for the error in (2.50) becomes

$$
\|f - \hat{f}\|_2 \leq \|(P^T U)^{-1}\|_2 \delta_\ast(f),
$$

(2.52)

and $\|(P^T U)^{-1}\|_2$ is the magnification factor needed to express the DEIM error in terms of the optimal approximation error. Hence determination of the bound for $\|f - \hat{f}\|_2$ is reduced to a problem of giving a bound for the matrix norm $\|(P^T U)^{-1}\|_2$.

The matrix norm $\|(P^T U)^{-1}\|_2$ depends on the DEIM selection of indices $\sigma_1, \ldots, \sigma_m$ through the permutation matrix $P$. We now show that each iteration of the DEIM algorithm aims to select an index to limit stepwise growth of $\|(P^T U)^{-1}\|_2$ and hence to limit size of the bound for the error $\|f - \hat{f}\|_2$.

To simplify notation, for $\ell = 2, \ldots, m$, we denote the relevant quantities at iteration
\ell \text{ by}

\begin{align*}
\bar{U} &= [u_1, \ldots, u_{\ell-1}] \in \mathbb{R}^{n \times (\ell-1)}, \quad \bar{P} = [e_{p_1}, \ldots, e_{p_{\ell-1}}] \in \mathbb{R}^{n \times (\ell-1)}, \\
u &= u_\ell \in \mathbb{R}^n, \quad p = e_{p_\ell} \in \mathbb{R}^n, \quad (2.53) \\
U &= [\bar{U} \ u] \in \mathbb{R}^{n \times \ell}, \quad P = [\bar{P} \ p] \in \mathbb{R}^{n \times \ell}.
\end{align*}

Put \( M = P^T U \) and consider the matrix norm \( \|M^{-1}\|_2 \) from Algorithm 1.

At the initial step of Algorithm 1, \( P = e_{p_1} \) and \( U = u_1 \). Thus,

\[ M = P^T U = e_{p_1}^T u_1 \quad \|M^{-1}\|_2 = \frac{1}{|e_{p_1}^T u_1|} = \|u_1\|_{\infty}^{-1} \geq 1, \quad (2.54) \]

which establishes condition (2.45). Note that the choice of the first interpolation index \( p_1 \) minimizes the matrix norm \( \|M^{-1}\|_2 \) and hence minimize the error bound in (2.52). Now consider a general step \( \ell \geq 2 \) with matrices defined by Equations (2.53).

With \( M = P^T U \), \( M = \begin{bmatrix} \bar{M} & \bar{P}^T u \\ p^T \bar{U} & p^T u \end{bmatrix} \), where \( \bar{M} = \bar{P}^T \bar{U} \) and \( M \) can be factored in the form of

\[ M = \begin{bmatrix} \bar{M} & \bar{P}^T u \\ p^T \bar{U} & p^T u \end{bmatrix} = \begin{bmatrix} \bar{M} & 0 \\ \bar{M} & 0 \end{bmatrix} \begin{bmatrix} I & c \\ a^T & \rho \end{bmatrix}, \quad (2.55) \]
where $a^T = P^T \bar{U}$, $c = \hat{M}^{-1} \bar{P}^T u$, and $\rho = p^T u - a^T c = p^T (u - \hat{U} \hat{M}^{-1} \bar{P}^T u)$. Note $|\rho| = \|r\|_\infty$ where $r$ is defined at Step 5 of Algorithm 1. Now, the inverse of $M$ is

\[
M^{-1} = \begin{bmatrix}
I - c \\
0 & 1
\end{bmatrix}
\begin{bmatrix}
\hat{M}^{-1} & 0 \\
-\rho^{-1} a^T \hat{M}^{-1} & \rho^{-1}
\end{bmatrix}
\]

(2.56)

\[
= \begin{bmatrix}
I - c \\
0 & 1
\end{bmatrix}
\begin{bmatrix}
I & 0 \\
-\rho^{-1} a^T & \rho^{-1}
\end{bmatrix}
\begin{bmatrix}
\hat{M}^{-1} & 0 \\
0 & 1
\end{bmatrix}
\]

(2.57)

\[
= \begin{bmatrix}
I & 0 \\
0 & 0
\end{bmatrix} + \rho^{-1}
\begin{bmatrix}
c \\
[a^T, -1]
\end{bmatrix}
\begin{bmatrix}
\hat{M}^{-1} & 0 \\
0 & 1
\end{bmatrix}
\]

(2.58)

A bound for the 2-norm of $M^{-1}$ is then given by

\[
\|M^{-1}\|_2 \leq \left\| \begin{bmatrix}
I & 0 \\
0 & 0
\end{bmatrix} \right\|_2 + |\rho|^{-1} \left\| \begin{bmatrix}
c \\
[a^T, -1]
\end{bmatrix} \right\|_2 \left\| \begin{bmatrix}
\hat{M}^{-1} & 0 \\
0 & 1
\end{bmatrix} \right\|_2
\]

(2.59)

Now, observe that

\[
\left\| \begin{bmatrix}
c \\
[a^T, -1]
\end{bmatrix} \right\|_2 = \left\| [\bar{U}, u] \begin{bmatrix}
c \\
[a^T, -1]
\end{bmatrix} \right\|_2
\]

(2.60)

\[
= \left\| \bar{U} c - u \right\|_2 \left\| [a^T, -1] \right\|_2
\]

(2.61)

\[
\leq \sqrt{1 + \|a\|^2_2 \sqrt{n}} \| \bar{U} c - u \|_\infty \leq \sqrt{2n} |\rho|.
\]

(2.62)

Substituting this into (2.59) gives

\[
\|M^{-1}\|_2 \leq \left[ 1 + \sqrt{2n} \|\hat{M}^{-1}\|_2 \right]
\]

(2.63)

to establish the recursive Formula (2.46) and conclude the proof. \qed
This error bound applies to any nonlinear vector-valued function \( f(\tau) = F(y(\tau)) \) approximated by DEIM. However, the bound is mainly of theoretical interest since it is very pessimistic and grows far more rapidly than the actual values of \( \| (P^TU)^{-1} \|_2 \) that I have observed in practice.

For a given dimension \( m \) of the DEIM approximation, the term \( \mathcal{C} \) is constant for any vector \( f \) and hence it applies to the approximation to any sample \( f(\tau) = F(V_k\tilde{y}(\tau)) \), where \( \tau \) represents time \( t \). However, the best approximation error

\[
\mathcal{E}_* = \mathcal{E}_*(f(\tau))
\]

is dependent upon \( f(\tau) \) and changes with each new value of \( \tau \). This would be quite expensive to compute, so an easily computable estimate is highly desirable. A reasonable estimate is available with the SVD of the nonlinear snapshot matrix

\[
\hat{F} = [f^1, f^2, \ldots, f^n].
\]

Let \( \mathcal{F} = \text{Range}(\hat{F}) \) and let \( \hat{F} = \hat{U}\hat{\Sigma}\hat{W}^T \) its SVD. where \( \hat{U} = [U, \tilde{U}] \) where \( U \) represents the leading \( m \) columns of the orthogonal matrix \( \hat{U} \). Partition \( \hat{\Sigma} = \begin{bmatrix} \Sigma & 0 \\ 0 & \tilde{\Sigma} \end{bmatrix} \) to conform with the partitioning of \( \hat{U} \). The singular values are ordered as usual with \( \sigma_1 \geq \sigma_2 \geq \cdots \sigma_m \geq \sigma_{m+1} \geq \cdots \geq \sigma_n \geq 0 \). The diagonal matrix \( \Sigma \) has the leading \( m \) singular values on its diagonal. The orthogonal matrix \( \hat{W} = [W, \tilde{W}] \) is partitioned accordingly. Any vector \( f \in \mathcal{F} \) may be written in the form

\[
f = \hat{F}\hat{g} = U\Sigma g + \tilde{U}\tilde{\Sigma}g,
\]
where \( g = W^T \hat{g} \) and \( \tilde{g} = \tilde{W}^T \hat{g} \). Thus

\[
\|f - f_*\|_2 = \|(I - UU^T)f\|_2 = \|\tilde{U}\tilde{\Sigma}\tilde{g}\|_2 \leq \sigma_{m+1}\|\tilde{g}\|_2.
\]

For vectors \( f \) nearly in \( F \), we have \( f = \hat{F}\hat{g} + w \) with \( w^T \hat{F}\hat{g} = 0 \), and thus

\[
\mathcal{E}_* = \mathcal{E}_*(f) \approx \sigma_{m+1}.
\]  
(2.64)

is reasonable so long as \( \|w\|_2 \) is small (\( \|w\|_2 = \mathcal{O}(\sigma_{m+1}) \) ideally). The POD approach (and hence the resulting DEIM approach) is most successful when the trajectories are attracted to a low dimensional subspace (or manifold). Hence, the vectors \( f(\tau) \) should nearly lie in \( F \) and this approximation will then serve for all of them.

To illustrate the error bound for DEIM approximation, we next present numerical examples on some examples of nonlinear parametrized functions defined on 1D and 2D discrete spatial points. These experiments show that the approximate error bound using \( \sigma_{m+1} \) in place of \( \mathcal{E}_* \) is quite reasonable in practice.

### 2.3.3 Numerical Examples of the DEIM Error Bound

This section demonstrates the accuracy and efficiency of the approximation from DEIM as well as its error bound given in Section 2.3.2. The examples here use the POD basis in the DEIM approximation. The POD basis is constructed from a set of snapshots corresponding to a selected set of elements in \( \mathcal{D} \). In particular, define

\[
\mathcal{D}^* = \{\mu_1^*, \ldots, \mu_{n_s}^*\} \subset \mathcal{D}
\]  
(2.65)
used for constructing a set of snapshots given by

\[ S = \{ f(\mu_1), \ldots, f(\mu_n) \}, \quad (2.66) \]

which is used for computing the POD basis \( \{ u_\ell \}_{\ell=1}^m \) for the DEIM approximation.

To evaluate the accuracy, apply the DEIM approximation \( \hat{f} \) in (2.40) to the elements in the set

\[ \tilde{\mathcal{D}} = \{ \tilde{\mu}_1, \ldots, \tilde{\mu}_n \} \subset \mathcal{D}, \quad (2.67) \]

which is different from and larger than the set \( \mathcal{D}^s \) used for the snapshots. Then consider the average error for DEIM approximation \( \hat{f} \) over the elements in \( \tilde{\mathcal{D}} \) given by

\[ \mathcal{E}(f) = \frac{1}{n} \sum_{i=1}^n \| f(\tilde{\mu}_i) - \hat{f}(\tilde{\mu}_i) \|_2, \quad (2.68) \]

the average POD error in (2.44) for POD approximation \( \hat{f}_* \) from (2.47) over the elements in \( \tilde{\mathcal{D}} \) given by

\[ \mathcal{E}_*(f) = \frac{1}{n} \sum_{i=1}^n \| f(\tilde{\mu}_i) - \hat{f}_*(\tilde{\mu}_i) \|_2 = \frac{1}{n} \sum_{i=1}^n \mathcal{E}(f(\tilde{\mu}_i)). \quad (2.69) \]

The error bound given in (2.43) guarantees the existence \( (P^T U)^{-1} \) and therefore, in this section, instead of using \( C \) given in Lemma 2.3.3, \( \tilde{C} := \| (P^T U)^{-1} \|_2 \) can be used directly to compute the average error bound which is then given by

\[ \frac{1}{n} \sum_{i=1}^n \| f(\tilde{\mu}_i) - \hat{f}(\tilde{\mu}_i) \|_2 \leq \tilde{C} \frac{1}{n} \sum_{i=1}^n \mathcal{E}(f(\tilde{\mu}_i)), \quad (2.70) \]

with the corresponding approximation using (2.64):

\[ \frac{1}{n} \sum_{i=1}^n \| f(\tilde{\mu}_i) - \hat{f}(\tilde{\mu}_i) \|_2 \lesssim \tilde{C} \sigma_{m+1}. \quad (2.71) \]
2.3.3.1 Ex: Nonlinear parametrized function with spatial points in 1D

The following numerical example illustrates the efficiency of DEIM in approximating a highly nonlinear function defined on a discrete 1D spatial domain. The nonlinear function used here is from an example in [16].

Consider a nonlinear parametrized function \( s : \Omega \times \mathcal{D} \mapsto \mathbb{R} \) defined by

\[
s(x; \mu) = (1 - x)\cos(3\pi \mu(x + 1))e^{-(1+x)^\mu},
\]

where \( x \in \Omega = [-1, 1] \) and \( \mu \in \mathcal{D} = [1, \pi] \).

Let \( \mathbf{x} = [x_1, \ldots, x_n]^T \in \mathbb{R}^n \), with \( x_i \) equidistantly spaced points in \( \Omega \), for \( i = 1, \ldots, n, n = 100 \). Define \( f : \mathcal{D} \mapsto \mathbb{R}^n \) by

\[
f(\mu) = [s(x_1; \mu), \ldots, s(x_n; \mu)]^T \in \mathbb{R}^n,
\]

for \( \mu \in \mathcal{D} \). This example uses 51 snapshots \( f(\mu_k) \) to construct POD basis \( \{u_e\}_{e=1}^{m} \) with \( \mu_1, \ldots, \mu_{51} \) are selected from equally spaced points in \([1, \pi]\). Figure 2.2 shows the singular values of these snapshots and the corresponding first 6 POD basis vectors with the first 6 spatial points selected from the DEIM algorithm using this POD basis as an input. Figure 2.3 compares the approximate functions from DEIM of dimension 10 with the original function of dimension 100 at different values of \( \mu \in \mathcal{D} \).

This demonstrates that DEIM gives a good approximation at arbitrary values \( \mu \in \mathcal{D} \).

Figure 2.4 illustrates the average errors defined in (2.68) and (2.69), the average error bound and its approximation computed from the right hand side of (2.70) and (2.71), respectively, with \( \bar{\mu}_1, \ldots, \bar{\mu}_{\bar{n}} \in \mathcal{D} \) selected uniformly over \( \mathcal{D} \) and \( \bar{n} = 101 \).
Figure 2.2: Singular values and the corresponding first 6 POD bases with DEIM points of snapshots from (2.73)

Figure 2.3: The approximate functions from DEIM of dimension 10 compared with the original functions (2.73) of dimension $n = 100$ at $\mu = 1.17, 1.5, 2.3, 3.1$.

Figure 2.4: Comparison of the average errors of POD and DEIM approximations for (2.73) with the average error bounds and their approximations given in (2.70) and (2.71), respectively.
2.3.3.2 Ex: Nonlinear parametrized function with spatial points in 2D

The following numerical example illustrates the DEIM approximation and its error bound for a nonlinear function defined on a discrete 2D spatial domain. The nonlinear function used in this example is modified from one given in [9]. In this example, DEIM gives more than 95% reduction in dimension with $\mathcal{O}(10^{-5})$ error.

Consider a nonlinear parametrized function $s : \Omega \times \mathcal{D} \mapsto \mathbb{R}$ defined by

$$s(x, y; \mu) = \frac{1}{\sqrt{(x - \mu_1)^2 + (y - \mu_2)^2 + 0.1^2}}, \quad (2.74)$$

where $(x, y) \in \Omega = [0.1, 0.9]^2 \subset \mathbb{R}^2$ and $\mu = (\mu_1, \mu_2) \in \mathcal{D} = [-1, -0.01]^2 \subset \mathbb{R}^2$.

Let $(x_i, y_j)$ be uniform grid points in $\Omega$, for $i = 1, \ldots, n_x$ and $j = 1, \ldots, n_y$. Define $s : \mathcal{D} \mapsto \mathbb{R}^{n_x \times n_y}$ by

$$s(\mu) = [s(x_i, y_j; \mu)] \in \mathbb{R}^{n_x \times n_y} \quad (2.75)$$

for $\mu \in \mathcal{D}$ and $i = 1, \ldots, n_x$, and $j = 1, \ldots, n_y$. In this example, the full dimension is $n = n_x n_y = 400$ ($n_x = n_y = 20$). Note that we can define a corresponding vector-valued function $\mathbf{f} : \mathcal{D} \mapsto \mathbb{R}^n$ for this problem, e.g. by reshaping the matrix $s(\mu)$ to a vector of length $n = n_x n_y$. The 225 snapshots constructed from uniformly selected parameters $\mu^s = (\mu^s_1, \mu^s_2)$ in parameter domain $\mathcal{D}$ are used for constructing the POD basis. The different 625 pairs of parameters $\mu$ are used for testing (error and cpu time). Figure 2.5 shows the singular values of these snapshots and the corresponding first 6 POD basis vectors. Figure 2.6 illustrates the distribution of the first 20 spatial points selected from the DEIM algorithm using this POD basis as
an input. Notice that most of the selected points cluster close to the origin, which
the singularity of the function \( s \). Figure 2.7 shows that the approximate functions
from DEIM of dimension 6 can reproduce the original function of dimension 400
very well at different values of \( \mu \in \mathcal{D} \). Figure 2.8 gives the average errors with the
bounds from the last section and the corresponding average CPU times for different
dimensions of POD and DEIM approximations. The average errors of POD and DEIM
approximations are computed from (2.68) and (2.69), respectively. The average error
bounds and their approximations are computed from the right hand side of (2.70)
and (2.71), respectively. This example uses \( \bar{\mu}_1, \ldots, \bar{\mu}_n \in \mathcal{D} \) selected uniformly over
\( \mathcal{D} \) and \( n = 625 \). The CPU times are averaged over the same set \( \mathcal{D} \).

![Singular Values of Snapshots](image)

Figure 2.5: Singular values and the first 6 corresponding POD basis vectors of the snapshots of
the nonlinear function (2.75).

The next section reviews the original EIM proposed in [2] as well as introduces an
alternative simplified form that is analogous to Algorithm 1 for DEIM. EIM is used
Figure 2.6: First 20 points selected by DEIM for the nonlinear function (2.75).

Figure 2.7: Comparison of the original nonlinear function (2.75) of dimension 400 with the POD and DEIM approximations of dimension 6 at parameter $\mu = [-0.05, -0.05]$. 
Figure 2.8: Left: Average errors of POD and DEIM approximations for (2.75) with the average error bounds given in (2.70) and their approximations given in (2.71). Right: Average cpu time for evaluating the POD and DEIM approximations.

for constructing an approximation of a non-affine parametrized function, which often arises in the FE discretized systems of unsteady nonlinear PDEs.

2.4 Empirical Interpolation Method (EIM) [2]

This section presents a scheme for removing the dependence on the dimension of the original system during the on-line computation due to the nonlinearities arising in the FE discretized systems. As noted in Section 2.1, this dependence occurs when the reduced-order system is constructed by projecting the original problem with a nonlinear term onto the space generated by the reduced basis. This problem in FE setting will be presented explicitly through the model problem in Chapter 5. One way to remove this dependency is to approximate the nonlinear function as a linear combination of spatial basis functions with time dependent coefficients by applying a method based on projection combined with interpolation. The time variable $t$ is
considered here as a parameter $\mu$.

Consider a parameterized function $f(x; \mu)$ with spatial variable $x$ defined on a physical domain $\Omega \subset \mathbb{R}^d$ and parameter $\mu$ in a parameter domain, $\mathcal{D} \subset \mathbb{R}^p$. For simplicity, assume here that $f(\cdot; \mu)$ is a real-valued function.

The approximation $\hat{f}(x; \mu)$ for $f(x; \mu)$ is of the form

$$\hat{f}(x; \mu) = \sum_{j=1}^{m} q_j(x)c_j(\mu), \quad (2.76)$$

where $\{q_j\}_{j=1}^{m}$ form a basis whose span gives a good approximation to $\mathcal{M}^f := \text{span}\{f(\cdot; \mu) : \mu \in \mathcal{D}\}$ and $\{c_j(\mu)\}_{j=1}^{m}$ are obtained from the coefficient function approximation using a set of pre-specified points, called interpolation points $\{z_i\}_{i=1}^{m}$, in $\Omega$ which are expected to capture the parameter variation. From the formula above, two ingredients for constructing $\hat{f}$ are:

- A set of basis functions $\{q_j\}_{j=1}^{m}$. The original paper [2] for EIM uses this basis set constructed from a greedy selection process on the set of snapshots. However, this thesis uses this basis set constructed from POD as described in Section 2.2.

- A set of interpolation points $\{z_i\}_{i=1}^{m}$ used in coefficient function approximation for $c_j(\mu)$. In particular, for a fixed value of $\mu$, $\{c_1(\mu), \ldots, c_m(\mu)\}$ in (2.76) satisfies

$$f(z_i; \mu) = \hat{f}(z_i; \mu) = \sum_{j=1}^{m} q_j(z_i)c_j(\mu), \quad (2.77)$$

for $i = 1, \ldots, m$. That is, $\{c_j(\mu)\}_{j=1}^{m}$ can be found by solving the $m$-by-$m$ linear system (2.77). This thesis uses the interpolation points constructed from the
EIM algorithm [2] presented later in this section. When this linear system is written in matrix form, the coefficient $c_j(\mu)$ can be written as a function of parameter $\mu$ explicitly as shown below in (2.82).

**MATRIX FORM**

For a fixed $\mu \in \mathcal{D}$, we define the following notations used in the explicit form of the approximation $\hat{f}$ in (2.76) and used in the steps for the EIM algorithm for the interpolation points.

For $\ell = 1, \ldots, m$, let

- **Basis (Input):** $U(x) = [u_1(x), \ldots, u_\ell(x)]$
- **Basis (Output):** $Q(x) = [q_1(x), \ldots, q_\ell(x)]$
- **Interpolation Points:** $z = [z_1, \ldots, z_\ell]^T \in \Omega^\ell$ (2.78)
- **Coefficient:** $c = [c_1, \ldots, c_\ell]^T \in \mathbb{R}^\ell$

$\ell^{th}$ Input Basis Function at $z$: $u_\ell(z) = [u_\ell(z_1), \ldots, u_\ell(z_\ell)]^T \in \mathbb{R}^\ell$

and

$$U(z) = \begin{bmatrix} u_1(z_1) & \ldots & u_\ell(z_1) \\ \vdots & \ddots & \vdots \\ u_1(z_\ell) & \ldots & u_\ell(z_\ell) \end{bmatrix}, Q(z) = \begin{bmatrix} q_1(z_1) & \ldots & q_\ell(z_1) \\ \vdots & \ddots & \vdots \\ q_1(z_\ell) & \ldots & q_\ell(z_\ell) \end{bmatrix} \in \mathbb{R}^{\ell \times \ell}, (2.79)$$

i.e., $[U(z)]_{ij} = u_j(z_i)$ and $[Q(z)]_{ij} = q_j(z_i)$ for $i, j = 1, \ldots, \ell$. Note that both $U(x)$ and $Q(x)$ are used as an input and output, respectively, in Algorithm 2 and it will be later shown that $\text{span}\{u_1, \ldots, u_\ell\} = \text{span}\{q_1, \ldots, q_\ell\}$. By setting $\ell = m$, the
approximation \( \hat{f} \) in (2.76) for the function \( f \) can be written as:

\[
\hat{f}(x; \mu) = Q(x)c(\mu),
\]

(2.80)

and (2.77) becomes

\[
f(z; \mu) = Q(z)c(\mu).
\]

(2.81)

Assume that we choose to use basis \( Q(x) \) and interpolation points \( z \) so that \( Q(z) \) is invertible. Then the coefficient can be written as a function of parameter \( \mu \) explicitly as follows:

\[
c(\mu) = (Q(z))^{-1}f(z; \mu).
\]

(2.82)

Therefore, from (2.80) and (2.82) the final function approximation can be written as

\[
\hat{f}(x; \mu) = Q(x)(Q(z))^{-1}f(z; \mu).
\]

(2.83)

### 2.4.1 EIM for interpolation points

EIM is proposed by Barrault, Maday, Nguyen, and Patera in [2]. It constructs a nested set of interpolation points \( \{z_i\}_{i=1}^m \) inductively on the input order basis. To simplify the notation, this section presents the EIM procedure, shown in Algorithm 2, by using matrix forms defined in (2.78) and (2.79). This setting is also useful for implementation purposes.

Similar to Algorithm 1, the explanation for Algorithm 2 is given as follows. The process starts from selecting the first interpolation point \( z_1 \in \Omega \) that maximizes
Algorithm 2: EIM [2] for Interpolation Points (Matrix Form)

INPUT: \( \{u_\ell\}_{\ell=1}^m \) linearly independent

OUTPUT: \( z \) and \( Q(x) \)

1. \( z_1 = \arg \text{ess sup}_{x \in \Omega} |u_1(x)| \)

2. \( q_1(x) = \frac{u_1(x)}{u_1(z_1)} \)

3. Initialize: \( z = [z_1], Q(x) = [q_1(x)], R = [u_1(z_1)] \)

4. for \( \ell = 2 \) to \( m \) do

5. Solve \( Q(z)c = u_\ell(z) \) for \( c \)

6. \( r_\ell(x) = u_\ell(x) - Q(x)c \)

7. \( z_\ell = \arg \text{ess sup}_{x \in \Omega} |r_\ell(x)| \)

8. \( q_\ell(x) = \frac{r_\ell(x)}{r_\ell(z_\ell)} \)

9. Update: \( z \leftarrow \begin{bmatrix} z \\ z_\ell \end{bmatrix} \), \( Q(x) \leftarrow [Q(x) \ q_\ell(x)] \), \( R = \begin{bmatrix} R & c \\ 0 & r_\ell(z_\ell) \end{bmatrix} \)

10. end for

Final Approx for \( f(x; \mu) \): \( \hat{f}(x; \mu) = Q(x)(Q(z))^{-1}f(z; \mu), \forall \mu \in \mathcal{D} \)
the magnitude of the first input basis \( u_1(x) \) over the points in the spatial domain \( \Omega \), and then normalizing \( u_1(x) \) by its value with maximum magnitude, i.e., \( u_1(z_1) \). This initialization step is well-defined since \( u_1(x) \) is a nonzero function (by the linear independence of the input basis). The remaining interpolation points, \( z_\ell \in \Omega \) for \( \ell = 2, \ldots, m \), are selected so that each of them maximizes \( r_\ell(x) = u_\ell(x) - Q(x)c \). In Step 6 of Algorithm 2, the term \( r_\ell(x) \) can be viewed as the residual between the input basis \( u_\ell(x) \) and its approximation from interpolating the basis \( \{ q_1(x), \ldots, q_{\ell-1}(x) \} \) at the points \( z_1, \ldots, z_{\ell-1} \) from the previous steps. Hence, \( r_\ell(z_i) = 0 \) for all \( i = 1, \ldots, \ell - 1 \).

However, the linear independence of the input basis guarantees that \( r_\ell(x) \) is a nonzero function and hence \( r_\ell(z_\ell) \) is nonzero. It is therefore valid to construct the corresponding transformed basis function \( q_\ell(x) \) in Step 8 by normalizing \( r_\ell(x) \) with its maximum magnitude on the domain \( \Omega \), i.e., \( r_\ell(z_\ell) \). This implies that \( q_j(z_i) = 0 \) for \( i = 1, \ldots, j - 1 \) and \( |q_j(z_j)| = 1 \) for \( j = 1, \ldots, \ell \). That is, \( Q(z) \) is invertible since it is an upper triangular matrix with nonzero determinant. Algorithm 2 is therefore well-defined. Note that the fact that \( r_\ell(x) \) is nonzero with \( r_\ell(z_i) = 0 \) for all \( i = 1, \ldots, \ell - 1 \) implies that Algorithm 2 generates non-repeated output interpolation points \( \{ z_i \}_{i=1}^m \). It is also clear that this procedure has a hierarchical property. The final approximation of the EIM is obtained from the general framework for the interpolation method described at the beginning of Section 2.4. A more detailed treatment of EIM, including the error estimates, may be found in [2] and [9].
2.4.2 Alternative Form of Algorithm for EIM

Notice that, besides the setting in function space, Algorithm 2 for EIM is mainly different from Algorithm 1 for DEIM at the steps that construct and use the transformed basis $Q(x)$, instead of the input basis $U(x)$. The linear transformation between $U(x)$ and $Q(x)$ in each iteration $\ell$ can be obtained through the matrix $R$ in Step 9 of Algorithm 2. This transformation is given by

$$U(x) = Q(x)R.$$  \hfill (2.84)

The matrix $R$ also can be written explicitly as

$$R = \begin{bmatrix}
    u_1(z_1) & c_1^1 & c_2^1 & \ldots & c_{\ell-1}^1 \\
    r_2(z_2) & c_1^2 & c_2^2 & \ldots & c_{\ell-1}^2 \\
    r_3(z_3) & \ldots & \ldots & \ldots & c_{\ell-1}^3 \\
    \vdots & \ddots & \ddots & \ddots & \ddots \\
    r_{\ell}(z_{\ell}) & & & & c_{\ell-1}^{\ell-1}
\end{bmatrix} \in \mathbb{R}^{\ell \times \ell}, \hfill (2.85)$$

where $c = [c_1^{j-1}, \ldots, c_{j-1}^{j-1}]^T \in \mathbb{R}^{j-1}$ is the solution from the linear system in Step 5 of Algorithm 2 at iteration $j$ for $j = 1, \ldots, \ell$. Equation (2.84) follows directly from Steps 6 and 8 of Algorithm 2. I.e., $r_\ell(x) = u_\ell(x) - Q(x)c$ and $q_\ell(x) = r_\ell(x)/r_\ell(z_\ell)$ imply

$$u_\ell(x) = r_\ell(x) + Q(x)c = q_\ell(x)r_\ell(z_\ell) + Q(x)c = [Q(x) \quad q_\ell(x)] \begin{bmatrix} c \\ r_\ell(z_\ell) \end{bmatrix}.$$
Since $u_j(z_j)$ and $r_j(z_j)$ are nonzero for all $j = 2, \ldots, \ell$ as noted earlier, then $R$ is nonsingular. From (2.84), this implies
\[
\text{span}\{q_1, \ldots, q_\ell\} = \text{span}\{u_1, \ldots, u_\ell\},
\] (2.86)

Also, from (2.84),
\[
U(z) = Q(z)R.
\] (2.87)

Hence, $U(z)$ is invertible, since both $Q(z)$ and $R$ are invertible. From (2.84) and (2.87),
\[
U(x)(U(z))^{-1} = (Q(x)R)(Q(z)R)^{-1} = Q(x)(Q(z))^{-1}.
\] (2.88)

The identity (2.88) is next used to show that the same interpolation point in each iteration of Algorithm 2 can be obtained from a procedure shown in Algorithm 3 that only involves the input basis without any basis transformation.

The first interpolation point $z_1$ from each of Algorithms 2 and 3 is clearly the same. For $\ell = 2, \ldots, m$, we note that each interpolation point $z_\ell$ is determined from the residual function $r_\ell(x)$ for both Algorithms 2 and 3. For Algorithm 2, combining Steps 5 and 6, the residual $r_\ell(x)$ is given by
\[
r_\ell(x) = u_\ell(x) - Q(x)c = u_\ell(x) - Q(x)(Q(z))^{-1}u_\ell(z).
\] (2.89)

Similarly, for Algorithm 3, the residual $r_\ell(x)$ is given by
\[
r_\ell(x) = u_\ell(x) - U(x)c = u_\ell(x) - U(x)(U(z))^{-1}u_\ell(z).
\] (2.90)
Applying (2.88), we see that \( r_\ell(x) \) in (2.89) and in (2.90) are indeed the same and therefore these two algorithms generate the same interpolation point at each iteration. Moreover, (2.88) implies that Algorithms 2 and 3 give the same final approximation:

\[
\hat{f}(x; \mu) = Q(x)(Q(z))^{-1}f(z; \mu) = U(x)(U(z))^{-1}f(z; \mu). \tag{2.91}
\]

Hence Algorithm 3 provides a simplified equivalent alternative to the original EIM [2] presented in Algorithm 2. Notice that Algorithm 1 for DEIM in the previous section is the discrete analogous procedure derived directly from Algorithm 3.

**Algorithm 3 : EIM without Transformed Basis (Matrix Form)**

**INPUT:** \( \{u_\ell\}^{m}_{\ell=1} \) linearly independent

**OUTPUT:** \( z \)

1: \( z_1 = \arg \max \sup_{x \in \Omega} |u_1(x)| \)

2: Initialize: \( z = [z_1], U(x) = [u_1(x)] \)

3: for \( \ell = 2 \) to \( m \) do

4: Solve \( U(z)c = u_\ell(z) \) for \( c \)

5: \( r_\ell(x) = u_\ell(x) - U(x)c \)

6: \( z_\ell = \arg \max \sup_{x \in \Omega} |r_\ell(x)| \)

7: Update: \( z \leftarrow \begin{bmatrix} z \\ z_\ell \end{bmatrix}, U(x) \leftarrow [U(x) \ u_\ell(x)] \)

8: end for

**Final Approx for** \( f(x; \mu) \): \( \hat{f}(x; \mu) = U(x)(U(z))^{-1}f(z; \mu), \forall \mu \in \mathcal{G} \)
Chapter 3

Model Problem: Finite Difference

Discretized System

This chapter illustrates how to apply Proper Orthogonal Decomposition (POD) with Discrete Empirical Interpolation Method introduced in Chapter 2 to a nonlinear system from a finite difference (FD) discretization.

The unsteady nonlinear PDEs called FitzHugh-Nagumo system, arising in neuron modeling, is considered here. This system is a simplified version of the Hodgkin-Huxley model, which models in a detailed manner activation and deactivation dynamics of a spiking neuron. More details of this system are provided in [23, 25]. The formulation of the FitzHugh-Nagumo system is given in (3.1)-(3.4). The numerical solutions of this system can be obtained from both finite difference (FD) and finite element (FE) methods. However, for the purpose of illustration, this chapter only
focuses on the FD discretization.

3.1 FitzHugh-Nagumo (FN) System

Let $x \in [0, L], t \geq 0$,

$$
\varepsilon v_t(x, t) = \varepsilon^2 v_{xx}(x, t) + f(v(x, t)) - w(x, t) \quad (3.1)
$$

$$
w_t(x, t) = b v(x, t) - \gamma w(x, t), \quad (3.2)
$$

where $f(v) = v(v - 0.1)(1 - v)$ with initial conditions and boundary conditions:

$$
v(x, 0) = 0, \quad w(x, 0) = 0, \quad x \in [0, L] \quad (3.3)
$$

$$
v_x(0, t) = -i_0(t), \quad v_x(L, t) = 0, \quad t \geq 0 \quad (3.4)
$$

where the parameters $L = 1, \varepsilon = 0.015, b = 0.5, \gamma = 2$, and a function $i_0(t)$, e.g.,

$$
i_0(t) = 100t^3 \exp(-15t); \quad i_0(t) = 50000t^3 \exp(-15t).
$$

The variable $v$ is voltage and the variable $w$ is recovery of voltage. The function $i_0(t)$ is the stimulus applied to the system.

3.2 Full Order Model of Finite Difference Discretized System

The finite difference (FD) method is used to obtain a discretized system of the PDE in (3.1) and (3.2). Consider first the discretization of the spatial domain $x_i = idx$
for \( i = 0, 1, \ldots, N + 1 \) with \( x_0 = 0 \) and \( x_{N+1} = L \) and the discretization of the
time domain \( t_j = jdt \) for \( j = 0, 1, \ldots \), where \( dx \) is the spatial stepsize and \( dt \) is the
time stepsize. Let \( v_i^j \) and \( w_i^j \) denote the solution of the discretized system at the
mesh point \((x_i, t_j)\) of \( v(x_i, t_j) \) and \( w(x_i, t_j) \), respectively. The the (forward Euler) FD
discretized system of (3.1) and (3.2) is shown below. Note that we can use different
numerical methods, e.g., backward Euler, which can be constructed in a similar way.

For \( i = 0, \ldots, N + 1 \), and \( j = 0, 1, \ldots \),

\[
\varepsilon \left( \frac{v_i^{j+1} - v_i^j}{dt} \right) = \varepsilon^2 \left( \frac{v_i^{j+1} - 2v_i^{j} + v_i^{j+1}}{dx^2} \right) + f(v_i^j) - w_i^j \tag{3.5}
\]

with initial conditions: \( v_i^0 = 0 \) and \( w_i^0 = 0 \) for all \( i = 1, \ldots, N + 1 \), and the boundary
conditions:

\[
\frac{v_1^j - v_0^j}{dx} = -i_0(t_j) \quad \Rightarrow \quad v_0^j = v_1^j + dx \cdot i_0(t_j), \tag{3.7}
\]

\[
\frac{v_{N+1}^j - v_N^j}{dx} = 0 \quad \Rightarrow \quad v_{N+1}^j = v_N^j, \tag{3.8}
\]

for \( j = 0, 1, \ldots \). The condition in (3.7) implies

\[
\frac{v_0^j - 2v_1^j + v_2^j}{dx^2} = \frac{(v_1^j + dx \cdot i_0(t_j)) - 2v_1^j + v_2^j}{dx^2} = -\frac{v_1^j + v_2^j}{dx^2} + \frac{i_0(t_j)}{dx}, \tag{3.9}
\]

and the condition in (3.8) implies

\[
\frac{v_{N-1}^j - 2v_N^j + v_{N+1}^j}{dx^2} = \frac{v_{N-1}^j - 2v_N^j + v_N^j}{dx^2} = \frac{v_{N-1}^j - v_N^j}{dx^2}. \tag{3.10}
\]
Let \( \mathbf{v}^j = [v_1^j, \ldots, v_N^j]^T \in \mathbb{R}^N \) and \( \mathbf{w}^j = [w_1^j, \ldots, w_N^j]^T \in \mathbb{R}^N \). Then we have

\[
\begin{bmatrix}
\varepsilon I_N & 0 \\
0 & I_N
\end{bmatrix} \frac{d}{dt} \begin{bmatrix}
\mathbf{v}^{j+1} \\
\mathbf{w}^{j+1}
\end{bmatrix} - \begin{bmatrix}
\mathbf{v}^j \\
\mathbf{w}^j
\end{bmatrix} = \begin{bmatrix}
-\frac{\varepsilon^2}{dx^2} \mathbf{K} & -I_N \\
\beta I_N & -\gamma I_N
\end{bmatrix} \begin{bmatrix}
\mathbf{v}^j \\
\mathbf{w}^j
\end{bmatrix} + \frac{\varepsilon^2}{dx} \begin{bmatrix}
g_2(t_j) \\
g_3(t_j)
\end{bmatrix} + \begin{bmatrix}
f(v^j) \\
0
\end{bmatrix};
\]

with initial condition:

\[
\begin{bmatrix}
\mathbf{v}^0 \\
\mathbf{w}^0 \\
y^0
\end{bmatrix} = 0. \text{ That is,}
\]

\[
\begin{align*}
\mathbf{v}^0 \\
\mathbf{w}^0 \\
y^0
\end{align*}
\]

\[
\mathbf{E} \frac{1}{dt} (y^{j+1} - y^j) = \mathbf{A} \mathbf{y}^j + g(t_j) + \mathbf{N}(y^j) \quad \text{and} \quad y^0 = 0, \tag{3.11}
\]
where

\[ y^j = \begin{bmatrix} v^j \\ w^j \end{bmatrix} \in \mathbb{R}^{2N} \quad \text{with} \quad v^j = \begin{bmatrix} v_1^j \\ \vdots \\ v_N^j \end{bmatrix}, \quad w^j = \begin{bmatrix} w_1^j \\ \vdots \\ w_N^j \end{bmatrix} \in \mathbb{R}^N, \ j = 0, 1, 2, \ldots \]

\[ E = \begin{bmatrix} \varepsilon I_N & 0 \\ 0 & I_N \end{bmatrix} \in \mathbb{R}^{2N \times 2N} \]

\[ A = \begin{bmatrix} -\frac{\varepsilon^2}{dx} K & -I_N \\ b I_N & -\gamma I_N \end{bmatrix} \in \mathbb{R}^{2N \times 2N} \]

\[ I_N = \text{identity matrix} \in \mathbb{R}^{N \times N} \]

\[ K = \begin{pmatrix} 1 & -1 \\ -1 & 2 & -1 \\ \vdots & \ddots & \ddots \\ -1 & 2 & -1 \\ -1 & 1 \end{pmatrix} \in \mathbb{R}^{N \times N} \]

\[ g(t) = \frac{\varepsilon^2}{dx} \begin{bmatrix} g_0(t) \\ 0 \end{bmatrix} \in \mathbb{R}^{2N} \quad \text{with} \quad g_0(t) = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{R}^N \]

\[ N(y^j) = \begin{bmatrix} f(v^j) \\ 0 \end{bmatrix} \in \mathbb{R}^{2N} \quad \text{with} \quad f(v^j) = \begin{bmatrix} f(v_1^j) \\ \vdots \\ f(v_N^j) \end{bmatrix} \in \mathbb{R}^N, \ j = 0, 1, 2, \ldots \]
3.3 Proper Orthogonal Decomposition (POD) Basis of the Snapshots

The POD basis used for constructing a reduced-order system can be computed from a given set of snapshots. In this setting, a snapshot is defined as the solution of (3.1)-(3.4) at a particular time $t$. Consider a set of $n_s$ snapshots at times $\{t_1, \ldots, t_{n_s}\} \subset \{t_1, \ldots, t_N\}$. Then, $v^\ell$ and $w^\ell$ are the $\ell^{th}$ snapshots at time $t_\ell$.

Define a matrix of snapshots:

$$
\hat{V} = \begin{bmatrix} v^1 & \ldots & v^{n_s} \end{bmatrix} \in \mathbb{R}^{N \times n_s}, \quad \hat{W} = \begin{bmatrix} w^1 & \ldots & w^{n_s} \end{bmatrix} \in \mathbb{R}^{N \times n_s}. \quad (3.12)
$$

The $\ell^{th}$ column of $\hat{V}$, $v^\ell$, corresponds to the solution of the FD discretized system at time $t_\ell$ and similarly for $\hat{W}$. Let $r = \min\{\text{rank}(\hat{V}), \text{rank}(\hat{W})\}$. The POD basis of dimension $k \leq r$ of the snapshots $\{v^\ell\}_{\ell=1}^{n_s}$ is the set of right singular vectors of $\hat{V}$ corresponding to the first $k$ largest singular values of $\hat{V}$ (similarly for the snapshots $\{w^\ell\}_{\ell=1}^{n_s}$). That is, the POD basis of the snapshots $\{v^\ell\}_{\ell=1}^{n_s}$ is the set of the first $k$ columns of $U^v$, where

$$
\hat{V} = U^v \Sigma^v (Z^v)^T
$$

is the singular value decomposition (SVD) of $\hat{V}$ with $\Sigma^v = \text{diag}(\sigma_1^v, \ldots, \sigma_r^v); \sigma_1^v \geq \cdots \geq \sigma_r^v > 0$. Similarly, the POD basis of the snapshots $\{w^\ell\}_{\ell=1}^{n_s}$ is set of the first $k$
columns of $U^w$, where

$$\tilde{V} = U^w \Sigma^w (Z^w)^T$$

is the singular value decomposition (SVD) of $\tilde{W}$ with $\Sigma^w = \text{diag}(\sigma_1^w, \ldots, \sigma_r^w)$; $\sigma_1^w \geq \cdots \geq \sigma_r^w > 0$.

### 3.4 Reduced Order Model from POD-Galerkin Method

Consider the matrices $U^v_k$ and $U^w_k$, which consist of the first $k$ column vectors of $U^v$ and $U^w$ from the previous section, respectively. Define

$$U_k = \begin{bmatrix} U^v_k & 0 \\ 0 & U^w_k \end{bmatrix} \in \mathbb{R}^{2N \times 2k}. \quad (3.13)$$

The reduced-order system of the discretized FD is obtained by projecting the system and the solution onto $U_k$. Let $y^j = U_k \tilde{y}^j$. The reduced system is of the form

$$\underbrace{U_k^T E U_k}_{\hat{E}} \frac{1}{dt} (\tilde{y}^{j+1} - \tilde{y}^j) = \underbrace{U_k^T A U_k}_{\hat{A}} \tilde{y}^j + \underbrace{U_k^T g(t_j)}_{\tilde{g}(t_j)} + \underbrace{U_k^T N(U_k \tilde{y}^j)}_{\tilde{N}(\tilde{y}^j)}; \quad \tilde{y}^0 = 0, \quad (3.14)$$

or,

$$\frac{1}{dt} (\tilde{y}^{j+1} - \tilde{y}^j) = \hat{A} \tilde{y}^j + \tilde{g}(t_j) + \tilde{N}(\tilde{y}^j) \quad \text{and} \quad \tilde{y}^0 = 0, \quad (3.15)$$

where $\hat{E} = U_k^T E U_k = \begin{bmatrix} \varepsilon I_k & 0 \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{2k \times 2k}$, $I_k \in \mathbb{R}^{k \times k}$ is the identity matrix;

$\hat{A} = U_k^T A U_k$; $\tilde{g}(t_j) = U_k^T g(t_j)$; and $\tilde{N}(\tilde{y}^j) = U_k^T N(U_k \tilde{y}^j)$. 

Notice that although in (3.15), the equation is expressed in the expansion of the reduced (POD) basis, the complexity in computing the nonlinear term still depends on the dimension $N$ of the full FD system. In particular, the nonlinear term is

$$\tilde{N}(\tilde{\phi}^j) = \begin{bmatrix} \tilde{N}^v(\tilde{\phi}^j) \\ 0 \end{bmatrix},$$

with $\tilde{N}^v(\tilde{\phi}^j)$ is of the form

$$\tilde{N}^v(\tilde{\phi}^j) = (U_k^v)^T f(U_k^w \tilde{\phi}^j) \in \mathbb{R}^k. \tag{3.16}$$

The problem here is that $f(U_k^w \tilde{\phi}^j)$ cannot be precomputed, since it depends on the unknown vector $\tilde{\phi}^j$.

### 3.5 Reduced Model for Nonlinear Term: DEIM

The previous discussion indicates that although the system (3.15) is expressed in the POD reduced basis, the complexity of on-line computation of the nonlinear term still depends on the dimension of the original full-order system, as shown in (3.16).

#### Discrete Empirical Interpolation Method (DEIM) with POD basis for Nonlinear term of the Reduced System

The (on-line) dependence on the dimension of the full FD discretized system in (3.15) can be removed by using DEIM described in Section 2.3 in Chapter 2. This section uses the POD basis/SVD basis of the nonlinear snapshots as an input basis for DEIM algorithm (see Algorithm 1). The POD basis of the nonlinear snapshots is constructed
from the solutions of the full FD system as follows. Let \( \{v^1, \ldots, v^{n_s}\} \) be solutions from the full FD system and let \( f(v^\ell) \) be the nonlinear function evaluated at \( v^\ell \) componentwise, for \( \ell = 1, \ldots, n_s \). Define

\[
\hat{F} = \begin{bmatrix}
| & | \\
| & |
\end{bmatrix}
\begin{bmatrix}
\begin{array}{c}
\vdots \\
f(v^1) \\
\vdots \\
f(v^{n_s}) \\
\vdots \\
\end{array}
\end{bmatrix} \in \mathbb{R}^{N \times n_s}.
\]

(3.17)

The POD basis of dimension \( m < \text{rank}(\hat{F}) \) of the nonlinear snapshots \( \{f(v^\ell)\}^{n_s}_{\ell=1} \) is set of the first \( m \) columns of \( U^f \), where

\[
\hat{F} = U^f \Sigma^f (Z^f)^T
\]

is the SVD of \( \hat{F} \) with \( \Sigma^f = \text{diag}(\sigma^f_1, \ldots, \sigma^f_r) \); \( \sigma^f_1 \geq \cdots \geq \sigma^f_r > 0 \).

With input basis vectors from the first \( m \) columns of \( U^f \), \( U^f_m \), Algorithm 1 in Chapter 2 for DEIM is then used to obtain the output interpolation indices \( \hat{\varphi} = [\varphi_1, \ldots, \varphi_m]^T \) for constructing matrix \( P \) defined in (2.29). The final approximation is obtained by considering \( y = U^f_k \hat{v} \), and so \( y_{\hat{\varphi}} = P^T U^f_k \hat{v} \) (note: \( P^T U^f_k = U^v_k (\hat{\varphi}, :) \) in MATLAB notation). Let \( D := P^T U^v_k \in \mathbb{R}^{m \times k} \). The approximation is then in the form of:

\[
f(U^f_k \hat{v}) \simeq Q Q_p^{-1} f(D \hat{v}),
\]

(3.18)

where \( Q = U^f_m \in \mathbb{R}^{n \times m} \) is a matrix of input basis vectors, and \( Q_p = P^T Q \in \mathbb{R}^{m \times m} \) is a matrix of input basis vectors with \textit{EIM interpolated rows}. 

Hence, the nonlinear term (3.16) is approximated by:

\[
\tilde{N}^\alpha(\tilde{v}) \simeq (U_k^\alpha)^T Q Q_p^{-1} f(D\tilde{v}) = C f(D\tilde{v}),
\]

where \( C = (U_k^\alpha)^T Q Q_p^{-1} \in \mathbb{R}^{k \times m} \), which can be precomputed and hence there is no dependence on dimension of original FD system.

Finally, from (3.15), the approximate reduced system is given by:

\[
\hat{E} \frac{1}{dt} (\tilde{y}^{j+1} - \tilde{y}^j) = \hat{A} \tilde{y}^j + \hat{g}(t_j) + \begin{bmatrix} C f(D\tilde{v}^j) \\ 0 \end{bmatrix} \quad \text{and} \quad \tilde{y}^0 = 0,
\]

where \( \tilde{y}^j = \begin{bmatrix} \tilde{v}^j \\ \tilde{w}^j \end{bmatrix} \); \( \hat{E}, \hat{A}, \hat{g}(t) \) are defined as in (3.15); \( C, D \) are defined as in (3.19); and \( f(D\tilde{v}^j) \in \mathbb{R}^m \) is evaluated at \( D\tilde{v}^j \in \mathbb{R}^m \) componentwise.

The next chapter compares the numerical results of the full-order and reduced-order FD discretized systems constructed in this chapter.
Chapter 4

Numerical Results: Finite Difference Discretized System

This chapter presents the numerical results from the reduced-order models of finite difference (FD) discretization of FitzHugh-Nagumo system introduced in Chapter 3 and of a variation of FitzHugh-Nagumo system, which has periodic solutions with limit cycles. These reduced-order systems are obtained from applying the POD and the POD with DEIM approaches described in Chapter 3. The numerical results demonstrate that POD-DEIM approach not only reduces the computational complexity, but also preserves the asymptotic behavior of the original full-order system.
4.1 FitzHugh-Nagumo System

Recall that FitzHugh-Nagumo (FN) system is given by: Let $x \in [0,L], t \geq 0,$

\[ \epsilon v_t(x,t) = \epsilon^2 v_{xx}(x,t) + f(v(x,t)) - w(x,t) \]  \hspace{1cm} (4.1)

\[ w_t(x,t) = bv(x,t) - \gamma w(x,t), \]  \hspace{1cm} (4.2)

where $f(v) = v(v-0.1)(1-v)$ with initial conditions and boundary conditions:

\[ v(x,0) = 0, \quad w(x,0) = 0, \quad x \in [0,L] \]  \hspace{1cm} (4.3)

\[ v_x(0,t) = -i_0(t), \quad v_x(L,t) = 0, \quad t \geq 0 \]  \hspace{1cm} (4.4)

where the parameters $L = 1, \epsilon = 0.015, b = 0.5, \gamma = 2,$ and a function $i_0(t), \text{ e.g.,}$

\[ i_0(t) = 100t^3 \exp(-15t); \quad i_0(t) = 50000t^3 \exp(-15t). \]

The variable $v$ is voltage and the variable $w$ is recovery of voltage. The function $i_0(t)$ is the stimulus applied to the system.

The following results are the numerical solutions from the reduced-order system of the FN system, constructed by applying the POD and the POD-DEIM approaches. The stimulus $i_0$ used in this section is: $i_0(t) = 50000t^3 \exp(-15t).$ This stimulus is large enough so that the system produces a traveling wave solution as shown in Figure 4.2 for the full-order system, which we consider to be the exact solution. The dimension of the full-order FD system is 1024. The POD basis functions are constructed from 100 snapshot solutions, which are obtained by solving the full-order
Figure 4.1: The singular values of the snapshots solution for $v$ and $w$ from the full finite difference system (dim = 1024).

Figure 4.2: Numerical Solutions $v$ and $w$ of the full-order system (dim=1024)
FD system at equally-spaced time steps in the interval \([0, 2]\). The singular values of the snapshots for \(v\) and \(w\) are shown in Figure 4.1.

Note that the dimension of the POD refers to the dimension of POD basis used for projecting the whole system; and the dimension of DEIM or EIM refers to the dimension of the nonlinear POD basis or the number of interpolation indices used in the approximation of the nonlinear term.

Figures 4.3 and 4.4 show the numerical solutions \(v\) and \(w\) obtained from the POD and the POD-DEIM reduced-order systems, respectively. These show that the standard POD approach with dimension 30, and the POD-DEIM approach with both POD and DEIM dimension 30, give good approximations to the original full-order system. Figures 4.5 and 4.6 show the average relative errors at different dimensions of POD and DEIM defined below in (4.5) and (4.6), respectively. The corresponding computational times are shown in Figure 4.8.

Let \(y^{FD}(t_j)\) be the solutions at time \(t_j\) from the full-order FD system. Let \(y^{POD}(t_j)\) and \(y^{DEIM}(t_j)\) be the solutions at time \(t_j\) from the POD reduced system and the POD-DEIM reduced system, respectively, after projecting back to the original system. In particular, if \(U_k\) is the matrix of the POD basis vectors as defined in (3.13) from the previous section, then \(y^{POD}(t_j) = y^j = U_k \tilde{y}^j\), where \(\tilde{y}^j\) is the solution at time \(t_j\) from the POD reduced system (and similarly for \(y^{DEIM}(t_j)\)). Note that \(y^{FD}(t_j), y^{POD}(t_j), y^{FD}(t_j) \in \mathbb{R}^{2N \times 2N}\). The average relative errors in 2-norm of the solutions from the reduced-order systems obtained from applying the POD and the POD-DEIM
approaches can be defined as follows:

\[ \mathcal{E}_1 := \frac{1}{n_t} \sum_{j=1}^{n_t} \frac{\|y^{FD}(t_j) - y^{POD}(t_j)\|_2}{\|y^{FD}(t_j)\|_2}, \]  \hspace{1cm} (4.5) 

and

\[ \mathcal{E}_2 := \frac{1}{n_t} \sum_{j=1}^{n_t} \frac{\|y^{FD}(t_j) - y^{DEIM}(t_j)\|_2}{\|y^{FD}(t_j)\|_2}. \]  \hspace{1cm} (4.6)

From Figure 4.5, the average relative error defined in (4.5) decreases as the dimension of the POD basis increases, as expected (from the decay of the singular values in Figures 4.1). However, notice from Figure 4.6 that, for a fixed DEIM dimension, the error decreases as the dimension of POD increases until when the dimension of the POD is large enough (error from the DEIM approximation has a larger effect than the error from POD), then the error becomes constant. On the other hand, for a fixed dimension of the POD basis, the errors from the POD-DEIM reduced systems do not decrease lower than the errors arising from applying POD. That is, once reaching the error contributed from the POD, the errors of the POD-DEIM systems do not further decrease, even though the dimension of DEIM increases.

To see the improvement of the DEIM approximation as its dimension increases, consider the POD-DEIM reduced system and the POD reduced system. These two systems are only different in their nonlinear terms, when the same POD basis is used for projecting both systems. Hence, by considering the solutions from the POD
Figure 4.3: Numerical Solutions $v$ and $w$ of the reduced system from the $POD$ approach: using $\dim POD = 10$ and $30$; compared with full FD system ($\dim = 1024$).
Figure 4.4: Numerical Solutions $v$ and $w$ of the reduced systems from the POD-DEIM approach: using dim of POD = 30 with dim of DEIM = 10 and 30; compared with full FD system (dim 1024).
Figure 4.5: Average relative errors in (4.5) of POD reduced systems compared to the original FD discretized systems.

reduced system to be "exact", the error arising from only the DEIM approximation can be defined as

$$\varepsilon^3 := \frac{1}{n_t} \sum_{j=1}^{n_t} \frac{\|y^{POD}(t_j) - y^{DEIM}(t_j)\|_2}{\|y^{POD}(t_j)\|_2}.$$ (4.7)

Figure 4.7 shows the errors defined in (4.7). It demonstrates that, when the effect of the error from truncating the POD basis is excluded, DEIM gives a better approximation as the dimension of its nonlinear POD basis and the number of interpolation entries increases. Also, it illustrates that the POD-DEIM reduced system gets closer to the corresponding POD reduced system as the dimension of DEIM basis increases.
Figure 4.6: Average relative errors of the POD-DEIM (solid lines)/POD (dashed line) reduced systems and the original FD discretized system, as defined in (4.5) and in (4.6). Notice that (i) for each fixed dimension of the POD, the error of the POD approach (dashed line) is the lower bound of the error of POD-DEIM approach (solid lines); and (ii) for each fixed dimension of DEIM, at the beginning, the error decreases as the dimension of POD increases, then becomes constant due to the larger effect of the error from the DEIM.
Figure 4.7: Average relative errors in (4.7) of the POD-DEIM reduced systems and the corresponding POD reduced systems (same dimension for POD but without DEIM).

Figure 4.8: Computational time for solving reduced (POD and POD-DEIM) systems compared with the full system.
4.2 A Variation of FitzHugh-Nagumo System with Periodic Solution

This section considers a variation of FitzHugh-Nagumo system, which has periodic solutions with limit cycles, to illustrate that the reduced-order systems from both POD and POD-DEIM approaches preserve the limit cycle of the original full-order system.

Let \( x \in [0, L], t \geq 0, \)
\[
\begin{align*}
\varepsilon v_t(x, t) &= \varepsilon^2 v_{xx}(x, t) + f(v(x, t)) - w(x, t) + c \quad (4.8) \\
w_t(x, t) &= bv(x, t) - \gamma w(x, t) + c, \quad (4.9)
\end{align*}
\]

where \( f(v) = v(v - 0.1)(1 - v) \) with initial conditions and boundary conditions:
\[
\begin{align*}
v(x, 0) &= 0, \quad w(x, 0) = 0, \quad x \in [0, L], \quad (4.10) \\
v_x(0, t) &= -i_0(t), \quad v_x(L, t) = 0, \quad t \geq 0, \quad (4.11)
\end{align*}
\]

where the parameters \( L = 1, \varepsilon = 0.015, b = 0.5, \gamma = 2, c = 0.05, \) and a function \( i_0(t), \) e.g.,
\[
i_0(t) = 100t^3 \exp(-15t); \quad i_0(t) = 50000t^3 \exp(-15t).
\]

The system (4.8)-(4.9) uses the same FD discretization setup and the model reduction techniques (the POD and the POD-DEIM techniques) as the ones described in Chapter 3. Note that the constant \( c \) is the only difference between the system (4.8)-(4.9) and the FN system in the previous section. The constant \( c \) makes the
solution of this system periodic. If $c$ is too small, the solution may not be periodic. This constant $c$ can be thought of as a stimulus. If it is below the threshold, it will have no effect to the system. In addition to the plots of the solutions, the numerical results in this section also include the phase-space diagrams or the phase portraits, which can be used to illustrate the qualitative information about the dynamics of the system, such as the limit-cycle.

The stimulus $i_0$ used in this section is $i_0(t) = 50000t^3 \exp(-15t)$. The dimension of the full-order system (discretized spatial points in finite difference) is 1024. The POD basis functions are constructed from 100 snapshot solutions, which are obtained by solving the full-order FD system at equally-spaced time steps in the interval $[0, 8]$.

Figure 4.9 shows the fast decay around the first 38 eigenvalues of the snapshot solutions for $v$, $w$, and the nonlinear snapshots $f(v)$. Figure 4.10 shows the periodic behavior of the solutions $v$ and $w$ with their stable limit cycle (i.e., there is a closed trajectory in the phase-space diagram which the trajectory spirals into it as time proceeds). Note that, since the solutions $v$ and $w$ are the functions depending on both the spatial variable $x$ and time variable $t$, the phase-space diagram is therefore considered for each fixed value of $x$ and presented in a 3D plot. Notice that in this system, the limit cycles for all points on the spatial domain turn out to be the same as shown in the 2D projection plots on the $v$-$w$ plane in Figure 4.10.

Figure 4.11 and Figure 4.12 are the solutions and the phase-space diagrams from the POD reduced system of dimension 3, 5, and 10. Similarly, Figure 6.4 and Fig-
ure 4.14 are the solutions and the phase-space diagrams from the POD-DEIM reduced system using a fixed POD dimension 5 with different DEIM dimensions: 1, 3, and 5. The average relative errors of the solutions of the reduced systems from both approaches with different dimensions of the POD (and the different dimensions of DEIM in the case of POD-DEIM) are presented in Figure 4.15. From Figure 4.15, the average relative error for the solutions of the reduced system constructed from POD of dimension 5 and DEIM of dimension 5 is $O(10^{-3})$ when compared to the original system of dimension 1024. These errors are defined as in (4.5) and (4.6) from the previous section. The corresponding computational times are shown in Figure 4.17.

In Figures 4.15, the similar behavior of errors for the POD and the POD-DEIM approaches can be observed as in the previous section; i.e., in Figure 4.15(b), the errors of POD (dashed line) are the lower bound for the errors of POD-DEIM for each fixed dimension of the POD basis. Notice also that the decay in the error of the POD approaches roughly follows the decay of the singular values of the snapshots as shown in Figure 4.16.
Figure 4.9: The eigenvalues of the 100 snapshot solutions for $v$, $w$, and $f(v)$ from the full-order FD system.
Figure 4.10: Numerical solutions $v$, $w$, and the phase-space diagram at different spatial points $x$ with its projection onto the $v$-$w$ plane from the full-order FD system (dim = 1024).
Figure 4.11: Numerical solutions $v$ and $w$ of the reduced-order systems from the POD approach: using POD bases of dimensions 3, 5, and 10; compared with the original FD system (dim 1024).
Figure 4.12: The phase-space diagram of $v$ and $w$ at different spatial points $x$ (left) and its projection onto the $v$-$w$ plane (right) from POD reduced-order systems: POD bases of dimensions 3, 5, and 10; compared with original FD system (dim 1024).
Figure 4.13: Numerical solutions $v$ and $w$ of the reduced systems from the POD-DEIM approach: using a POD basis of dimension 5 with DEIM of dimensions 1, 3, and 5; compared with the original FD system (dim 1024).
Figure 4.14: The phase-space diagram of $v$ and $w$ at different spatial points $x$ (left) and its projection onto the $v$-$w$ plane (right) from POD-DEIM reduced systems: POD dimension 5 with DEIM dimensions 1, 3, and 5; compared with original FD system (dim 1024).
Figure 4.15: (a) Average relative errors of the solutions from the POD reduced systems and (b) from the POD-DEIM reduced system (solid lines) compared with the ones from POD reduced systems (dashed line). Notice that errors of POD (dashed line) are the lower bound for the errors of POD-DEIM for each fixed dimension of the POD basis.
Figure 4.16: Average relative errors of the reduced-order system from the POD-DEIM (solid lines) and POD (dashed line) approaches with the plots of the singular values (of the snapshot solutions $v, w, f(v)$).

Figure 4.17: Computational time (scaled with the CPU time of the sparse full-order system) for solving reduced (POD and POD-DEIM) systems compared with the full system.
Chapter 5

Model Problem: Finite Element

Discretized System

This chapter demonstrates how to apply the Proper Orthogonal Decomposition (POD) as well as the Empirical Interpolation Method (EIM) [2] presented in Chapter 2 to construct a reduced-order model for the finite element (FE) discretized system for the 1D unsteady Burgers’ equation, which is a commonly used model problem to test an algorithm for a Navier-Stokes type equation. The 1D unsteady Burgers’ equation has a simple nonlinearity that is sufficient to test the algorithm. It is generally used in gas dynamics modeling, fluid flow modeling and control theory.
5.1 Unsteady Burgers’ Equation (1D)

Consider a one dimensional unsteady Burgers’ equation with homogeneous boundary conditions and initial conditions:

\[
\frac{\partial}{\partial t} y(x, t) - \nu \frac{\partial^2}{\partial x^2} y(x, t) + \frac{\partial}{\partial x} \left( \frac{(y(x, t))^2}{2} \right) = 0 \quad x \in [0, 1], \quad t \geq 0
\]  

(5.1)

\[y(0, t) = y(1, t) = 0, \quad t \geq 0\]  

(5.2)

\[y(x, 0) = y_0(x), \quad x \in [0, 1],\]  

(5.3)

where \(y(x, t)\) is the unknown function of time \(t\) and location \(x \in \Omega \equiv [0, 1]\); \(\nu\) is a diffusion coefficient; and \(y_0(x)\) is an initial condition.

5.2 Full Order Model of the Discretized System

The finite element method is used to obtain a discretized system of the above PDE in (5.1). The variational form (weak form) of (5.1) is shown in (5.4). It is obtained by requiring \(\langle v, y_t - f(y) \rangle = 0\) for all \(v \in H^1(\Omega)\) where \(\Omega \equiv [0, 1]\) and \(f(y(x, t)) = \nu \frac{\partial^2}{\partial x^2} y(x, t) - \frac{\partial}{\partial x} \left( \frac{(y(x, t))^2}{2} \right)\). This leads to multiplying \(v \in H^1(\Omega)\) throughout (5.1), integrating it over the domain \(\Omega\), and applying integration by parts (See details in Appendix A).

\[
\int_\Omega \frac{\partial}{\partial t} y(x, t) v(x) dx + \nu \int_\Omega \frac{\partial}{\partial x} y(x, t) \frac{d}{dx} v(x) dx - \int_\Omega s(y(x, t)) \frac{d}{dx} v(x) dx = 0,
\]  

(5.4)

where \(s(y(x, t)) = s(y) = \frac{1}{2} y^2\).
By dividing the spatial domain $\Omega = [0, 1]$ into $n_x$ equal elements, $T^h_i = [x_{i-1}, x_i]$ where $x_0 = 0, x_i = ih$ for $i = 1, \ldots, n_x$ and $h = \frac{1}{n_x}$, the discretized state space can be given by

$$X \equiv \{ w \in H^1(\Omega) ; w|_{T^h_i} \text{ is linear} \forall i = 1, \ldots, n_x \text{ and } w|_{\partial \Omega} = 0 \}.$$ 

Suppose $\{\psi_j\}_{j=1}^{n_y}$ forms a basis for $X$. Then this basis can be used for approximating the weak solution to (5.4); i.e. the approximate solutions can be written as

$$y_h(x, t) = \sum_{j=1}^{n_y} \psi_j(x)y_j(t). \quad (5.5)$$

From (5.4), by substituting $v = \psi_i$ for $i = 1, \ldots, n_y$ and $y_h(x, t) = \sum_{j=1}^{n_y} y_j(t)\psi_j(x)$, then finally the FE discretized system becomes the $n_y$-by-$n_y$ system of (nonlinear) ordinary differential equations (ODE):

$$M_h\frac{d}{dt}y(t) + \nu K_h y(t) - N_h(y(t)) = 0, \quad (5.6)$$

where $M_h, K_h \in \mathbb{R}^{n_y \times n_y}$, and $y(t), N_h(y(t)) \in \mathbb{R}^{n_y}$. $N_h : \mathbb{R}^{n_y} \to \mathbb{R}^{n_y}$ is a nonlinear functional of $y(t)$. The entries of these matrices and vectors are defined as follows.

For $i, j = 1, \ldots, n_y$,

$$[M_h]_{ij} = \int_{\Omega} \psi_i(x)\psi_j(x)dx,$$

$$[K_h]_{ij} = \int_{\Omega} \frac{d}{dx} \psi_i(x) \frac{d}{dx} \psi_j(x)dx,$$

$$[y(t)]_i = y_i(t),$$

$$[N_h(y(t))]_i = \int_{\Omega} \frac{d}{dx} \psi_i(x)s(y_h(x, t))dx = \int_{\Omega} \frac{d}{dx} \psi_i(x) \frac{1}{2} \left( \sum_{j=1}^{n_y} y_j(t)\psi_j(x) \right)^2 dx.$$
The ODE system in (5.6) can be solved by using an appropriate standard numerical method for solving an ODE system.

This model problem uses \( n_y \) piecewise linear elements on a uniform grid with \( n_x \) subintervals of length \( h \). Let \( n_y = n_x - 1, h = \frac{1}{n_x} \), and \( x_0 = 0, x_i = ih \) for \( i = 1, \ldots, n_x \).

The piecewise linear functions are defined as follows:

\[
\psi_i(x) = \begin{cases}
  \frac{x-(i-1)h}{h}, & x \in [x_{i-1}, x_i], \\
  \frac{-x+(i+1)h}{h}, & x \in [x_i, x_{i+1}], \quad i = 1, \ldots, n_y, \\
  0, & \text{otherwise.}
\end{cases} \tag{5.7}
\]

Note that \( \{\psi_1, \ldots, \psi_{n_y}\} \) is a basis for \( X \). With this choice of basis, matrices \( M_h, K_h \) in the discretized system (5.6) are given explicitly as

\[
M_h = \frac{h}{6} \begin{pmatrix}
  4 & 1 & \\
  1 & 4 & 1 \\
  \vdots & \vdots & \vdots \\
  1 & 4 & \\
  1 & 4
\end{pmatrix}
\quad \text{and} \quad
K_h = \frac{1}{h} \begin{pmatrix}
  2 & -1 \\
  -1 & 2 & -1 \\
  \vdots & \vdots & \vdots \\
  -1 & 2 & -1 \\
  -1 & 2
\end{pmatrix}.
\]

Note also that the linear function \( \psi_i(x) \) in (5.7) has a constant derivative for each subinterval as shown below:

\[
\frac{d}{dx} \psi_i(x) = \begin{cases}
  \frac{1}{h}, & x \in [x_{i-1}, x_i], \\
  -\frac{1}{h}, & x \in [x_i, x_{i+1}], \quad i = 1, \ldots, n_y, \\
  0, & \text{otherwise.}
\end{cases} \tag{5.8}
\]

Equation (5.8) is used to compute \( K_h \) and is also used to simplify the computation.
for the nonlinear term $N_h(y(t))$ as shown below. For $i = 1, \ldots, n_y$,

\[
[N_h(y(t))]_i = \int_0^1 \frac{d}{dx} \psi_i(x) s(y_h(x, t)) dx
\]

\[
= \frac{1}{h} \int_{x_{i-1}}^{x_i} s(y_h(x,t)) dx - \frac{1}{h} \int_{x_i}^{x_{i+1}} s(y_h(x,t)) dx. \quad (5.10)
\]

The solutions at a set of time steps from this FEM discretized system will be used to construct the POD basis as shown in the next section.

### 5.3 Proper Orthogonal Decomposition (POD) Basis of the Snapshots

The POD basis used for constructing a reduced-order system can be computed from a given set of snapshots as shown in Chapter 2. In this setting, a snapshot is defined as the solution of (5.6) at particular time $t$. Consider a set of $n_s$ snapshots at times $t_1, \ldots, t_{n_s}$. The $\ell$th snapshot at time $t_\ell$ is of the form

\[
y_h(x, t_\ell) = \sum_{j=1}^{n_y} \psi_j(x) \hat{y}_j(t_\ell) = \Psi(x)\hat{y}(t_\ell), \quad (5.11)
\]

for $\ell = 1, \ldots, n_s$, where $\Psi(x) = [\psi_1(x), \ldots, \psi_{n_y}(x)]$ is a $1$-by-$n_y$ matrix of FE basis functions and $\hat{y}(t) = [\hat{y}_1(t), \ldots, \hat{y}_{n_y}(t)]^T \in \mathbb{R}^{n_y}$.

Define a matrix of snapshot coefficients:

\[
\hat{Y} = \begin{bmatrix}
\vdots & \vdots \\
\hat{y}(t_1) & \cdots & \hat{y}(t_{n_s}) \\
\vdots & \vdots 
\end{bmatrix} \in \mathbb{R}^{n_y \times n_s}. \quad (5.12)
\]
The $j^{th}$ column of $\hat{Y}$, $\hat{y}(t_j)$, corresponds to coefficients in FE expansion of snapshot $j$.

Let $\{y_\ell\}_{\ell=1}^{n_s} \equiv \{y_h(x, t_\ell)\}_{\ell=1}^{n_s}$ be a set of snapshots as defined in (5.11). Let $r = \min\{n_y, n_s\}$. The POD basis of dimension $k \leq r$ of the snapshots $\{y_\ell\}_{\ell=1}^{n_s}$ can be constructed as follows.

- Form the inner product matrix of the snapshots $\mathcal{L}$, which is given in an explicit form as shown below:

$$
\mathcal{L} = [\langle y_i, y_j \rangle_x] = [\sum_{k, l=1}^{n_y} \hat{y}_k(t_i) \hat{y}_l(t_j) \langle \psi_i, \psi_j \rangle_x] = \hat{Y}^T M_h \hat{Y} \in \mathbb{R}^{n_s \times n_s},
$$

where $M_h = [\langle \psi_i, \psi_j \rangle_x] \in \mathbb{R}^{n_y \times n_y}$ is the matrix of inner products of FE basis functions as defined earlier in the case of hat functions.

- Find $k$ eigenvectors of $\mathcal{L}$ corresponding the first $k$ largest eigenvalues. That is, find $v_i$ such that

$$
\mathcal{L}v_i = v_i \lambda_i,
$$

for $i = 1, \ldots, k$ and $\lambda_1 \geq \cdots \geq \lambda_k$ are the largest $k$ eigenvalues of $\mathcal{L}$.

- Then each POD basis function $\phi_i$ is given by: for $i = 1, \ldots, k$,

$$
\phi_i(x) = \frac{1}{\sqrt{\lambda_i}} \sum_{\ell=1}^{n_s} (v_i)_\ell y_h(x, t_\ell).
$$

- To write each POD basis function $\phi_i$ in terms of the FE basis $\Psi(x) = [\psi_1(x), \ldots, \psi_{n_y}(x)]$ in a matrix/vector form, let $\mathcal{L}V = \Lambda V$ be the eigenvalue decomposition of $\mathcal{L}$ where $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_{n_s})$ with $\lambda_1 \geq \cdots \geq \lambda_{n_s} \geq 0$. Let $V_k =$
\[
\begin{bmatrix}
| & | \\
v_1 & \ldots & v_k \\
| & | \\
\mathbb{R}^{n \times k} & \mathbb{R}^{k \times k}
\end{bmatrix}
\in \mathbb{R}^{n \times k}
\text{ be the first } k \text{ columns of } V \text{ and } \Lambda_k = \text{diag}(\lambda_1, \ldots, \lambda_k) \in \mathbb{R}^{k \times k}.
\]

Then, the 1-by-\(k\) matrix of POD basis functions, \(\Phi(x) \equiv [\phi_1(x), \ldots, \phi_k(x)]\) can be written as

\[
\Phi(x) = \Psi(x) \tilde{Y} V_k \Lambda_k^{-\frac{1}{2}}.
\]  \(5.13\)

Figure 5.1 shows the plots of the first four POD basis functions obtained from applying (5.13) to the snapshots of this model problem. The POD basis extracts the essential characteristics of the solution space of the model problem. Figure 5.1 illustrates that the dominant eigenfunction (the first POD basis function), i.e., the eigenfunction corresponding to the largest singular value, has most of the qualitative information about the solution space.

### 5.4 Reduced Order Model from POD-Galerkin Method

By using the POD basis \(\{\phi_1(x), \ldots, \phi_k(x)\}\), the approximate solution can be written as

\[
\tilde{y}_h(x, t) = \sum_{j=1}^{k} \phi_j(x) \tilde{y}_j(t) = \Phi(x) \tilde{y}(t),
\]  \(5.14\)

where \(\tilde{y}(t) = [\tilde{y}_1(t), \ldots, \tilde{y}_k(t)]^T \in \mathbb{R}^k\) (for a fixed time \(t\)) and \(\Phi(x) \equiv [\phi_1(x), \ldots, \phi_k(x)]\) is a 1-by-\(k\) matrix of POD basis functions.
Figure 5.1: Plots of the first 4 POD basis functions from the snapshots of the model problem (left). Notice that the first POD basis has the similar shape as the solution space of the problem (right).

From the weak form (5.4), instead of using the larger set of finite element basis functions as in the last section, we now use \( \{ \phi_i(x) \}_{i=1}^k \) as the test functions. As shown in (5.13) from the previous section, the relationship between the reduced basis \( \Phi(x) \) and the FE basis \( \Psi(x) \) is given by

\[
\Phi(x) = \Psi(x) U_k,
\]

where \( U_k = \hat{Y} V_k \Lambda_k^{-\frac{1}{2}} \). The reduced order model of the discretized system corresponding to (5.6) then becomes

\[
\tilde{M}_k \frac{d}{dt} \tilde{y}(t) + \nu \tilde{K}_k \tilde{y}(t) - \tilde{N}(\tilde{y}(t)) = 0,
\]  
(5.15)
where

\[
\tilde{M}_k = U_k^T M_k U_k \in \mathbb{R}^{k \times k},
\]
\[
\tilde{K}_k = U_k^T K_k U_k \in \mathbb{R}^{k \times k},
\]
\[
\tilde{N}(\tilde{y}(t)) = U_k^T N_h(U_k \tilde{y}(t)) \in \mathbb{R}^k,
\]

or, \([\tilde{N}(\tilde{y}(t))]_i = \int_{\Omega} \frac{d}{dx} \phi_i(x)s(\tilde{y}_h(x,t))dx = \int_{\Omega} \sum_{t=1}^{n_y} (u_t) \frac{d}{dx} \psi_t(x)s \left( \sum_{j=1}^{k} \tilde{y}_j(t) \phi_j(x) \right) dx,\]

for \(i = 1, \ldots, k\). \(M_h, K_h \in \mathbb{R}^{n_y \times n_y}\) are defined as for (5.6) and \(N_h : \mathbb{R}^k \rightarrow \mathbb{R}^k\) is a nonlinear functional of \(\tilde{y}(t)\). Note that the POD basis given in (5.13) is orthonormal, so we have \(\tilde{M}_k = \int_{\Omega} \Phi(x)^T \Phi(x)dx = I_k\), where \(I_k\) is the \(k\)-by-\(k\) identity matrix.

Note also that although (5.15) is written in terms of the expansion in the reduced basis (POD basis) \(\{\phi_i\}_{i=1}^k\), the complexity in computing the nonlinear term still depends on the dimension \(n_y\) of the full system in the FE basis. In particular, the nonlinear term is of the form

\[
\tilde{N}(\tilde{y}(t)) = U_k^T \gamma(\tilde{y}(t)) \in \mathbb{R}^k \quad \text{where} \quad \gamma(\tilde{y}(t)) = \begin{bmatrix} \int_{\Omega} \frac{d}{dx} \psi_1(x)s(\Phi(x)\tilde{y}(t))dx \\ \vdots \\ \int_{\Omega} \frac{d}{dx} \psi_{n_y}(x)s(\Phi(x)\tilde{y}(t))dx \end{bmatrix} \in \mathbb{R}^{n_y}. \quad (5.16)
\]

The problem here is that \(\gamma(\tilde{y}(t))\) cannot be precomputed, since evaluating \(\gamma(\tilde{y}(t))\) depends on the unknown coefficient vector \(\tilde{y}(t)\). The next section shows how to remove this inefficiency.
5.5 Reduced Model for Nonlinear Term

The previous discussion indicates that although the system (5.15) is expressed in the POD reduced basis, the complexity in \textit{on-line} computation of the nonlinear term still depends on the dimension of the original full-order system as shown in (5.16).

This section illustrates two approaches to handle this problem in the nonlinear term. These two approaches are the Empirical Interpolation Method (EIM) with POD basis and the precomputing technique. EIM, introduced in Chapter 2, can be applied to general nonlinear problems, but the precomputing technique can be used only for quadratic nonlinearities. However, for this particular model problem which does have quadratic nonlinear term, the precomputing technique works better than EIM in terms of accuracy and complexity, as shown in the numerical results in the next chapter. The common idea of these approaches is to remove the dependence on the dimension of the original system during the on-line computation by separating the spatial variables from the time variable.

5.5.1 Empirical Interpolation Method (EIM) with POD basis for the Nonlinear term of the Reduced System

EIM gives an approximate function for the nonlinear term in the form

\[
\hat{s}(\tilde{y}_h(x, t)) = \sum_{m=1}^{n_m} q_m(x)\alpha_m(\tilde{y}_h(z, t)),
\]  

(5.17)
where $z = \{z_1, \ldots, z_{n_m}\}$ are the pre-specified (off-line) points in $\Omega$ that ‘represent’ the nonlinearity of $s(y)$; with the coefficient function $\alpha_m(\tilde{y}_h(z, t))$, for $m = 1, \ldots, n_m$ and the basis $\{q_m\}_{m=1}^{n_m}$.

To find a POD basis for the nonlinear term, first the snapshots of the nonlinear term are constructed by substituting the snapshot solutions from the full-order (FE) system into the nonlinear functions. The $\ell^{th}$ snapshot of the nonlinear term, $s_{\ell}$, is defined by

$$s_{\ell} \equiv s(y_h(x, t_\ell)) = s\left(\sum_{j=1}^{n_y} \psi_j(x)\tilde{y}_j(t_\ell)\right) = \frac{\left(\sum_{j=1}^{n_y} \psi_j(x)\tilde{y}_j(t_\ell)\right)^2}{2},$$

for $\ell = 1, \ldots, n_s$ where $y_h(x, t_\ell) = \sum_{j=1}^{n_y} \psi_j(x)\tilde{y}_j(t_\ell)$ is the $\ell^{th}$ snapshot solution from the FE system as defined in (5.11). Note that these snapshots of the nonlinear function can be computed concurrently with the snapshot solutions $y_h(x, t_\ell)$, for $\ell = 1, \ldots, n_s$.

Then the POD basis for snapshots of the nonlinear term is obtained by using the same procedure as for the POD basis for the snapshots of the FE system. In particular, first construct the matrix of inner products of the nonlinear snapshots:

$$\mathcal{L}^s \equiv [(s_i, s_j)] \in \mathbb{R}^{n_s \times n_s},$$

and then find its eigenvalue decomposition to obtain a POD basis as in the case of $\mathcal{L}$ defined in (2.15).

The approximation of the $s(y_h(x, t_\ell))$ can be found from EIM shown in Section 2.4 of Chapter 2 by setting $f(x; t) = s(y_h(x, t_\ell))$ and setting the POD basis for nonlinear term as an input basis $\{u_i\}_{i=1}^{m}$ for the EIM steps in Algorithm 2. Note that the first
four POD basis functions of the nonlinear term together with the corresponding basis
after applying the EIM (i.e. \( q_\ell(x) \)) are shown in Figure 5.2.

Then the EIM approximation of the nonlinear function \( s(\tilde{y}_h(x,t)) = s(\Phi(x)\tilde{y}(t)) \)
in (5.16) is then given by

\[
s(\Phi(x)\tilde{y}(t)) \simeq Q(x)\rho(t) = Q(x)(Q(z))^{-1}s(\Phi(z)\tilde{y}(t)) = Q(x)(Q(z))^{-1}s(\Phi(z)\tilde{y}(t)).
\]  

(5.19)

By substituting (5.19) into (5.16) and recalling that \( Q(x) = [q_1(x), \ldots, q_m(x)] \), the
unknown \( \tilde{y}(t) \) can be separated from the integral as shown below. It is important to
note that, being able to separate the unknown \( \tilde{y}(t) \) would allow the precomputation
of the integrals which then can be used in all of the time steps.

\[
\gamma(\tilde{y}(t)) = \begin{bmatrix}
\int_\Omega \frac{d}{dx} \psi_1(x)s(\tilde{y}_h(x,t))dx \\
\vdots \\
\int_\Omega \frac{d}{dx} \psi_{n_y}(x)s(\tilde{y}_h(x,t))dx
\end{bmatrix}
\simeq \begin{bmatrix}
\int_\Omega \frac{d}{dx} \psi_1(x)Q(x)(Q(z))^{-1}s(\Phi(z)\tilde{y}(t))dx \\
\vdots \\
\int_\Omega \frac{d}{dx} \psi_{n_y}(x)Q(x)(Q(z))^{-1}s(\Phi(z)\tilde{y}(t))dx
\end{bmatrix}
\]

\[
= \begin{bmatrix}
(\int_\Omega \frac{d}{dx} \psi_1(x)Q(x)dx)(Q(z))^{-1}s(\Phi(z)\tilde{y}(t)) \\
\vdots \\
(\int_\Omega \frac{d}{dx} \psi_{n_y}(x)Q(x)dx)(Q(z))^{-1}s(\Phi(z)\tilde{y}(t))
\end{bmatrix}
\]

\[
= \begin{bmatrix}
\int_\Omega \frac{d}{dx} \psi_1(x)q_1(x)dx & \ldots & \int_\Omega \frac{d}{dx} \psi_1(x)q_m(x)dx \\
\vdots & \ddots & \vdots \\
\int_\Omega \frac{d}{dx} \psi_{n_y}(x)q_1(x)dx & \ldots & \int_\Omega \frac{d}{dx} \psi_{n_y}(x)q_m(x)dx
\end{bmatrix}
\begin{bmatrix}
s(\Phi(z_1)\tilde{y}(t)) \\
\vdots \\
s(\Phi(z_m)\tilde{y}(t))
\end{bmatrix}
\]

So from (5.16), we write the nonlinear term as:

\[
\tilde{N}(\tilde{y}(t)) = \text{U}^T\text{E}(Q(z))^{-1}s(D\tilde{y}(t)) = \text{C}\ s(D\tilde{y}(t)),
\]
Figure 5.2: The first 4 POD basis functions of the nonlinear term (top); and the corresponding basis functions after applying EIM (bottom).
where for $i = 1, \ldots, n_y$, $j = 1, \ldots, k$, and $\ell = 1, \ldots, m$,

$$E = \left[ \int_\Omega \frac{d}{dx} \psi_i(x) q_\ell(x) dx \right] \in \mathbb{R}^{n_y \times m}$$

$$D = [\phi_j(z_\ell)] \in \mathbb{R}^{m \times k}$$

$$C = U_k^T E(Q(z))^{-1} \in \mathbb{R}^{k \times m}.$$  

Note that matrix $C$ is precomputed and hence there is no dependence on the dimension of finite element basis. The discretized system in (5.15) then becomes

$$\ddot{\bar{M}}_k \frac{d}{dt} \bar{\mathbf{y}}(t) + \nu \ddot{\bar{K}}_k \bar{\mathbf{y}}(t) - C s(D\tilde{\mathbf{y}}(t)) = 0,$$  \hspace{1cm} (5.20)

where $s(D\tilde{\mathbf{y}}(t)) \in \mathbb{R}^m$ is evaluated at $D\tilde{\mathbf{y}}(t) \in \mathbb{R}^m$ componentwise.

The next section shows a precomputing technique that can also remove the dependence on the dimension of the original FE system. However, it is only applicable for the problems with quadratic nonlinearities.

### 5.5.2 Precomputing Technique for Nonlinear term of the Reduced System

This approach is developed specially for removing the dependence of the dimension of the original full-order system in the quadratic nonlinear term after applying Galerkin projection during the on-line computation. The reduced-order model of the nonlinear term from this approach is still in the POD basis expansion as other linear terms in the discretized PDEs. The dependence of the full dimension is moved into the coefficient matrix that can be pre-computed. This approach is based on manipulating the
order of computation. Therefore, there is no loss in accuracy when compared to the reduced-order system constructed directly from POD. The following shows how to do the precomputing for the 1D case. A similar approach can be used for higher spatial dimensions.

The quadratic term \( s \) can be written as

\[
s(\tilde{y}_h(x,t)) = s(\sum_{j=1}^{k} \tilde{y}_j(t)\phi_j(x)) = \frac{1}{2}(\sum_{j=1}^{k} \tilde{y}_j(t)\phi_j(x))^2 = \frac{1}{2}(\Phi(x)\tilde{y}(t))(\Phi(x)\tilde{y}(t)) = \frac{1}{2}(\tilde{y}(t)^T\Phi(x)^T)(\Phi(x)\tilde{y}(t)).
\]

Recall the nonlinear term in (5.15):

\[
[\tilde{N}(\tilde{y}(t))]_i = \int_{\Omega} \frac{d}{dx} \Psi(x)u_i s(\tilde{y}_h(x,t))dx
\]

\[
= \int_{\Omega} \frac{d}{dx} \Psi(x)u_i \frac{1}{2}(\tilde{y}(t)^T\Phi(x)^T)(\Phi(x)\tilde{y}(t))dx
\]

\[
= \frac{1}{2} \int_{\Omega} (\tilde{y}(t)^T\Phi(x)^T) \frac{d}{dx} \Psi(x)u_i (\Phi(x)\tilde{y}(t))dx
\]

\[
= \tilde{y}(t)^T \left[ \frac{1}{2} \int_{\Omega} \Phi(x)^T \frac{d}{dx} \Psi(x)u_i (\Phi(x)dx) \right] \tilde{y}(t)
\]

\[
= \tilde{y}(t)^T G_i \tilde{y}(t),
\]

where \( G_i = \frac{1}{2} \int_{\Omega} \Phi(x)^T \frac{d}{dx} \Psi(x)u_i (\Phi(x)dx) \in \mathbb{R}^{k\times k} \) for \( i = 1, \ldots, k \). The matrix \( G_i \) is precomputed and hence the system in (5.15) can be truly reduced. The technique for replacing the integration in the full model can also be used here to make the precomputing more efficient. Here are the details. Using \( \Phi(x) = \Psi(x)U_k \), we can
expand $G_i$ for $i = 1, \ldots, k$ in terms of the finite element basis as:

$$G_i = \frac{1}{2} \int_{\Omega} \Phi(x)^T \left( \frac{d}{dx} \Psi(x)u_i \right) \Phi(x) dx = \frac{1}{2} U^T \left[ \int_{\Omega} \Psi(x)^T \left( \frac{d}{dx} \Psi(x)u_i \right) \Psi(x) dx \right] U_k.$$ 

Consider each entry of $T_i \in \mathbb{R}^{n_y \times n_y}$:

$$(T_i)_{pq} = \int_{\Omega} \psi_p(x) \psi_q(x) \left( \frac{d}{dx} \Psi(x)u_i \right) dx$$

$$= \int_{\Omega} \psi_p(x) \psi_q(x) \left( \frac{d}{dx} \psi_1(x)u_{i,1} + \cdots + \frac{d}{dx} \psi_{n_y}(x)u_{i,n_y} \right) dx.$$ 

If the hat functions are used as finite element basis functions, then the nonzero entries are on diagonal, upper- and lower-diagonal entries, i.e. when $p = q - 1$, $p = q$, and $p = q + 1$. In particular, let $u_i = [u_{i,1}, \ldots, u_{i,n_y}]^T$, then

$$(T_i)_{pq} = \begin{cases} 
\frac{u_{i,2}}{3}, & p = q = 1, \\
\frac{-u_{i,n_y-1}}{3}, & p = q = n_y, \\
\frac{-u_{i,p-1}+u_{i,p+1}}{3}, & p = q, p \in \{2, \ldots, n_y - 1\}, \\
\frac{-u_{i,p-1}+u_{i,p}}{6}, & p = q - 1, p \in \{2, \ldots, n_y\}, \\
\frac{-u_{i,p}+u_{i,p+1}}{6}, & p = q + 1, p \in \{1, \ldots, n_y - 1\}, \\
0, & \text{otherwise}. 
\end{cases}$$ (5.21)$$

See Appendix A for details. Note that $T_i$ is a symmetric tridiagonal matrix. Notice that once the matrix $G_i$ is precomputed in the off-line stage, the nonlinear term depends only on the dimension of the POD reduced basis.

The next chapter illustrates the numerical results of the techniques discussed in this section.
Chapter 6

Numerical Results: Finite Element

Discretized System

This chapter compares model reduction for the 1D Burgers’ Equation using EIM with the POD basis (the POD-EIM approach) with two other benchmark approaches: 
direct-POD method, which applies the standard POD directly to the discretized system without any treatment on nonlinearities, and precompute-POD method, which employs the formulations of the precomputing technique described in the previous chapter.

Consider again the 1D unsteady Burgers’ equation from Chapter 5:

\[ \frac{\partial}{\partial t} y(x, t) - \nu \frac{\partial^2}{\partial x^2} y(x, t) + \frac{\partial}{\partial x} \left( \frac{y(x, t)^2}{2} \right) = 0 \quad x \in [0, 1], t \geq 0 \quad (6.1) \]

\[ y(0, t) = y(1, t) = 0 \quad t \geq 0 \quad (6.2) \]

\[ y(x, 0) = y_0(x) \quad x \in [0, 1], \quad (6.3) \]
where $\nu$ is a viscosity parameter and $y_0$ is a given initial condition. Two numerical examples of the 1D Burgers’ equation are presented. The first example uses simple initial condition with a fixed viscosity parameter, which is large enough so that the solution has no oscillatory effect. The purpose of the first example is to compare the relative errors and the computational times of the POD-EIM approach to two other benchmark approaches, Direct POD and Precompute-POD approaches as noted earlier.

The second example uses a more complicated initial condition with varied viscosity parameters in a fixed interval. The purpose of the second example is to illustrate how to construct a single set of POD basis vectors that can be used for all parameters in a given range and to show the effectiveness of the POD basis for the case of an oscillatory solution.

6.1 Burgers’ Equations (1D) with a Fixed Viscosity Parameter

This section presents and compares the numerical solutions as well as the errors and the computational times from solving the reduced-order systems of the unsteady 1D Burgers’ equation introduced in the previous chapter. As discussed earlier, these reduced-order systems are obtained from applying POD to the linear terms and following three different approaches (direct POD; precompute POD; and EIM with POD
basis, called POD-EIM) to the nonlinear terms. The accuracy and the efficiency of each approach are different, as shown later in this section. The last part of this section also compares the errors and the computational times for solving the reduced-order systems using different dimensions of the POD and EIM.

The initial condition used in this section is \( y_0(x) = f(x) - f(0) \), where \( f(x) = e^{-(15(x-0.5))^2} \); the diffusion parameter \( \nu \) is 0.1; and the number of finite element (FE) basis functions (hat functions) used in the spatial discretization (on \([0,1]\)) for the full-order system is 100. The POD basis functions are constructed from 70 snapshots which are obtained by solving the full-order FE system at equally-spaced time steps in the interval \([0,2]\).

Before constructing a reduced-order system, the reduced dimension (i.e., the number of POD basis functions) has to be specified. The singular values reflect the error of the solutions of the reduced systems obtained from using different numbers of POD basis functions. Therefore, assuming that the snapshots are dense enough in the space of the solutions from the full-order system, this plot can be used to suggest the chosen dimension for the reduced system or even help to decide if the POD method should be used to construct a reduced-order system. Figure 6.1 is the plot of the singular values of the 70 snapshots from this model problem. The singular values of these snapshots decay rapidly at the beginning (i.e., the first 10 singular values) and hence the space of the solution could be expected to be much smaller than the number of the snapshots.
Figure 6.1: Singular values of the 70 snapshot solutions from the original system (FE discretization of dimension 100). The plot shows the rapid decay of the first ten singular values.

Figures 6.2 to 6.4 compare the numerical solutions of the reduced discretized systems (dimensions 1 to 6) and of the finite element (FE) discretization (dimension 100) of the 1D Burgers’ equation obtaining from the three approaches. For the POD-EIM approach, the nonlinear term of the reduced system uses 5 nonlinear POD basis vectors (from 70 nonlinear snapshots). From these figures, when the POD basis of dimension 6 is used, the numerical solutions of the reduced systems from all approaches (the POD-EIM approach with EIM dimension 5) appear to be indistinguishable from the numerical solution of the original FE discretized system. The comparison of the exact errors are shown in the next subsection.
Figure 6.2: Numerical solutions of the reduced-order system from the Direct POD approach: using a POD basis of dimension 1 to 5 at time $t = 0, 0.2029, 0.9855, 1.2174$, and 2 (2D on the top) and using a POD basis of dimension 6 (the 3D surface plot on the bottom left); compared with the original system (100 FE basis).
Figure 6.3: Numerical solutions of the reduced-order system from the Precompute POD approach: using a POD basis of dimension 1 to 5 at time $t = 0, 0.2029, 0.9855, 1.2174$, and 2 (the 2D plots on the top) and using a POD basis of dimension 6 (the 3D surface plot on the bottom left); compared with the original system (100 FE basis).
Figure 6.4: Numerical solutions of the reduced-order system from the POD-EIM approach: using the EIM dimension 5 with a POD basis of dimension 1 to 5 at time $t = 0, 0.2029, 0.9855, 1.2174$, and 2 (the 2D plots on the top) and using a POD basis of dimension 6 (the 3D surface plot on the bottom left); compared with the original system (100 FE basis).
The accuracy of the approaches can be analyzed by considering the relative error of the solutions from the reconstructed reduced-order systems from these approaches with the solution of the original FE system. The average relative error is given in the form below:

$$E_{avg} = \frac{1}{n_t} \sum_{i=1}^{n_t} \frac{\| \tilde{y}_h(\cdot, t_i) - y_h(\cdot, t_i) \|_X}{\| y_h(\cdot, t_i) \|_X},$$

where $\| \cdot \|_X$ is the induced norm in the Hilbert space $X$ as defined earlier; $\tilde{y}_h(x, t_i)$ is the solution of the reduced-order system from each approach; and $y_h(x, t_i)$ is the solution of the original FE system for $i = 1, \ldots, n_t$. The first plot in Figure 6.5 shows the average of the relative error of the solutions given in (6.4) at the time steps used for the selected snapshots.

The complexity is measured by the CPU time used for solving the reduced-order system. The two plots in Figure 6.5 show the trade-off between the accuracy and the complexity of these approaches. The fewer the number of POD basis functions, the less accurate the reduced model is going to be when compared to the FEM model. On the other hand, a larger number of POD basis functions would increase the complexity of the reduced model to an extent that it might be too slow to be used in real-time process control or other applications.

Despite the fact that it is only applicable to the problem with quadratic nonlinearities, the precomputing technique is more effective than the POD and the EIM-POD approaches in the sense that it has the highest accuracy with lowest computational
time. However, for the generalization purposes, the EIM-POD approach, which can be applied to any type of nonlinearity, shows a considerable improvement over the other two approaches. Clearly, the EIM-POD approach reduces the computational complexity while retaining a high accuracy approximation. For this model problem, from Figure 6.5, the average relative error given in (6.4) of the reduced system with the POD of dimension 6, and the EIM of dimension 5 (dimension of the full order FE system is 100), is only $O(10^{-5})$.

This chapter uses one fixed number of the EIM basis functions for approximating the nonlinear term in the EIM-POD approach to compare with the other two benchmark approaches (the POD and the precomputing techniques). Generally, the dimension of the EIM basis can be varied and is usually much smaller than the dimension of the POD basis used for the linear terms (preliminary numerical results are shown in Appendix A).
Figure 6.5: (Top) Relative errors of the solution from reduced-order systems constructed from POD bases of size 1 to 15 with the three different approaches; (Bottom) CPU times used for solving the corresponding reduced-order systems constructed from POD bases of size 1 to 15.
6.2 Burgers’ Equations (1D) with Varied Viscosity Parameters

This section demonstrates numerical results for 1D Burgers’ Equations with varied viscosity parameters in a fixed interval with more complicated initial condition than the one in the previous section. In particular, this section considers initial condition 
\[ y_0(x) = 0.3 \sin(\pi x) + \sin(4\pi x) \] and diffusion coefficient \( \nu \in \mathcal{D} = [0.005, 0.1] \). This is a modified version of the problem that has been used as a test example with small \( \nu \) for moving mesh methods by many authors [10, 7, 31]. This example aims to illustrate:

- How to construct a POD basis that can be used for all parameters \( \nu \in \mathcal{D} \);

- The efficiency of the POD basis for oscillatory solution phenomena (i.e., small \( \nu \) or large Péclet number).

One way to construct a set of the POD basis that works for all \( \nu \in \mathcal{D} \) is to take a set of snapshots over the intervals of both time and parameter \( \nu \). That is, we first have to select a discrete set of times

\[ \{t_1, \ldots, t_n\} \subset [0, T], \]

where \( T > 0 \) is a final time; and a discrete set of diffusion parameters

\[ \{\nu_1, \ldots, \nu_n\} \subset \mathcal{D}. \]
The following is an example of these snapshots $n_s \times n_p$.

$$
\{ y_1^{\nu_1}, \ldots, y_{n_s}^{\nu_1}, y_1^{\nu_2}, \ldots, y_{n_s}^{\nu_2}, \ldots, y_1^{\nu_p}, \ldots, y_{n_s}^{\nu_p}\},
$$

where $y_i^{\nu_j}$ is the FE solution at time $t_i$ of (5.1) with $\nu = \nu_j$ for $i = 1, \ldots, n_s$ and $j = 1, \ldots, n_p$. That is, $y_i^{\nu_j}$ is of the form

$$
y_i^{\nu_j} = y_h(x, t_i) = \sum_{\ell=1}^{n_y} \psi_\ell(x)y_\ell(t_i),
$$

where $\{\psi_\ell\}_{\ell=1}^{n_y}$ is a FE basis.

An oscillatory phenomenon of the solution can occur when the viscosity $\nu$ is small but not zero. To avoid this situation, the number of FE basis functions needed in the discretization has to be large enough or certain special technique, such as non-uniform mesh with progression factors, has to be used. However, the number of the POD basis functions needed to be used in this situation is relatively small when compared to FE basis (see the numerical results when $\nu = 0.007$).

The following results use $n_s = 50$, $n_p = 10$ (i.e., the total number of snapshots = 500), and $n_y = 200$. The snapshots are selected uniformly on the time interval $[0, 1]$, and selected uniformly on the parameter (diffusion) interval $\mathcal{D} = [0.005, 0.1]$. First we consider the plots of the SVD and the POD basis, which can be used for all the parameters in $\mathcal{D}$. 
Figure 6.6: Singular values plot of the snapshots

Figure 6.7: Top: Plots of first 4 POD basis functions constructed from snapshot solutions with various $\nu \in [0.005, 0.1]$. Bottom: Plots of solutions from FE with different $\nu$: $\nu = 0.005, 0.01, 0.03, 0.1$, which are used in the sampled snapshots.
Figure 6.2 shows the numerical solutions from the full-order system (FE) using 200 FE basis functions. Figures 6.2 to 6.2 shows the numerical solutions from reduced-order systems (Direct-POD, Precomputed-POD, and POD-EIM) with dimensions 1, 6, and 10. Each column in each figure corresponds to a fixed value of viscosity parameter: \( \nu = 0.007, 0.01, \) and 0.07. The numerical solutions from the reduced-order systems of dimension 10 give a good approximation to the ones from the full-order systems for all viscosity parameters \( \nu = 0.007, 0.01, 0.07. \) The average relative error, defined in (6.4), of these three different viscosity parameters are shown in Figure 6.2.

Figure 6.8: Numerical solutions from the FE systems of dim 200 with different values of viscosity parameters \( \nu = 0.007, 0.01, 0.07. \)
Figure 6.9: Numerical solutions from reduced systems from POD approach with dim 1, 6, and 10. Each column corresponds to a fixed value of viscosity parameter: $\nu = 0.007, 0.01, \text{ and } 0.07$. 
Figure 6.10: Numerical solutions from reduced systems from Precomputed-POD approach with dim 1, 6, and 10. Each column corresponds to a fixed value of viscosity parameter: $\nu = 0.007, 0.01, \text{and } 0.07$. 
Figure 6.11: Numerical solutions from reduced systems from POD-EIM approach with dim of POD = 1, 6, 10, and with a fixed dim of EIM = 10. Each column corresponds to a fixed value of viscosity parameter: $\nu = 0.007, 0.01, \text{ and } 0.07$. 
Figure 6.12: Average relative errors of the three approaches with three different parameters $\nu = 0.007, 0.01, \text{ and } 0.07$. 
Chapter 7

Conclusions and Future Work

This thesis compared the effectiveness of model reduction techniques for finite difference (FD) and finite element (FE) discretized systems of unsteady nonlinear PDEs. It primarily focused on (i) the technique combining Proper Orthogonal Decomposition (POD) and Discrete Empirical Interpolation Method (DEIM), called the POD-DEIM technique, for FD case and (ii) the technique combining POD and Empirical Interpolation Method (EIM), called the POD-EIM technique, for FE case.

EIM proposed by [2] is originally developed to handle the nonlinearities in FE discretized systems. It cannot be applied directly on a nonlinear term in the FD discretized system due to the continuity assumption in the spatial domain. This thesis therefore introduced DEIM, which is a discrete variation of EIM, as well as the error bound, as shown in Section 2.3 of Chapter 2. DEIM can be applied to approximate the nonlinear terms of FD discretized systems or general ODE systems.
Chapter 4 demonstrated the success of the POD-DEIM approach on a practical problem through FitzHugh-Nagumo (FN) system, which arises in models of the neuron systems and cardiac electrical activity. The numerical results demonstrate that the POD-DEIM approach not only gives an accurate reduced system that is substantially smaller than the original system with general nonlinearity (e.g., the dimension is reduced from 1024 to 30 with error of order $O(10^{-5})$), but it also preserves the steady-state behavior (e.g. the limit cycle) of the original system. This thesis also illustrated the application of the POD-EIM technique to the FE discretized system of Burgers' equation as shown in Chapter 6. The numerical results also demonstrated the effectiveness of the POD-EIM approach: e.g. it gave a 90% reduction in dimension with $O(10^{-5})$ error.

The numerical results in Chapter 4 and in Chapter 6 showed that both POD-DEIM and POD-EIM approaches can be used to construct an accurate reduced system with an improvement in efficiency (on-line computational time), compared to the reduced system constructed from the standard POD approach. This improvement reflects the effectiveness of both DEIM and EIM procedures for removing the dependency on the dimension of the original full-order system, which occurs when the POD approach is applied to a system with nonlinearity.

In the case of a system consisting of a parameter that can be varied, a POD basis constructed from this system with one fixed parameter may not work well for the same system with a different parameter, since POD produces a problem-dependent
basis. This thesis illustrated how to construct a single POD basis that can be used for a system with parameter varying in a certain range for Burgers’ equation with varying viscosity parameter as shown in Section 6.2 of Chapter 6. This technique is based on constructing a POD basis by using selected snapshots from the system with different parameters on the given range. The effectiveness of the POD-EIM approach was also shown in the case of problem with an oscillatory solution.

The average relative errors for POD with POD-DEIM approach and POD with POD-EIM approaches, shown in Chapter 4 and Chapter 6, demonstrated that (i) the decay in the errors of the POD approaches roughly follows the decay of the singular values of the snapshots as shown in Figure 4.16, and (ii) the error of the POD is the lower bound for the errors of the EIM-POD for each fixed dimension of the POD basis. Therefore, the singular values can be used to suggest the desired dimension of the POD basis.

**Future Work**

The previous chapters have shown the success of the DEIM and EIM on simple model problems. To generalize this method, further work needs to be done including:

- **Extending to higher dimensions (2D and 3D)**:

  Many realistic models are defined on physical domains in 2D and 3D. The framework of the DEIM and EIM algorithm is applicable to parametrized PDEs defined on any open and bounded physical domain $\Omega \subset \mathbb{R}^d$ for $d = 1, 2, 3$. 
Therefore, the analogous, but more complicated, procedures to the approach presented for the 1D case in this thesis can be applied to the 2D and 3D cases.

- **Extending and improving the existing error bound estimation:**

  For the EIM-POD approach, the quality of the approximation depends on both the POD basis and the EIM basis/interpolation points. The error contributed by POD can be seen from the singular values of the snapshots. For the error from EIM, the only available error bounds are presented in [2] as an _a priori_ and an _a posteriori_ error bounds. The given _a priori_ error bound, based on the Lebesgue constant, can be a very loose bound and hence it may not be useful in practice. The _a posteriori_ error bound is given in a rigorous fashion and it is claimed to be sharp in practice as shown in numerical results in [2, 20]. However, it is still limited to assumptions that may not hold in general. Therefore, the work on deriving an effective and generalized error bound estimate still needs to be done.

  In the case of the system from FD discretization shown in Chapter 3 with _DEIM_ introduced in Section 2.3 of Chapter 2, there is a more efficient and generalized _a posteriori_ error bound. The formulation of _DEIM_ allows an alternative approach for error analysis to be done rigorously in the aspect of numerical linear algebra for general FD systems. The derivation of this error bound also gives an insight into both DEIM and EIM selection processes. The procedures for _DEIM_ and the original EIM are very similar and therefore the error analysis given in Section 2.3
could be extended to the case of FE discretizations. However, this extension may not be straightforward due to the continuity of the spatial domain of the FE systems.

- **Applying and performing the comparative study for the Best Point Interpolation Method (BPIM):**
  The BPIM constructs a set of points which is the solution of a least-square minimization of the error and therefore it is superior in accuracy to EIM, which is based on the greedy selection process. However, the BPIM involves some optimization algorithms which make its computational cost higher than EIM. Further analysis is still needed in order to indicate whether or not the gain in accuracy would compensate for the loss of computational efficiency from using the BPIM instead of the EIM.

- **Constructing a reduced order system for a nonlinear model by employing the information from its corresponding linear model:**
  For nonlinear systems, it is sometimes impossible or too expensive to generate a set of snapshots by solving the original full-order system. In the case when a physical system can be modeled by using both a linear system as a simplified model and a nonlinear system as a realistic model, it might be useful to use some information from the linear problem to construct a reduced system for its corresponding nonlinear problem (i.e., using snapshots from the linear problem, instead of the nonlinear problem). This idea is more desirable when the analytic
solution of the linear system is known. The modeling of polymer chains using the bead-spring model and the modeling of neuron systems are examples that fit into this situation.

- **Developing software based on the EIM-POD procedure for constructing reduced order systems:**

  The challenging aspects in developing this software include (i) generalization of the procedure (e.g., selecting snapshots, constructing a projection basis and interpolation points, and computing coefficient function approximations) to a broad class of parametrized nonlinear systems both from FE and FD discretizations, (ii) incorporating automatic order selection (selection of the reduced dimension) based on error analysis, and (iii) integrating with PDE solvers (e.g. MATLAB) and FEM(finite element method) softwares (e.g. MATLAB PDE Toolbox, freeFEM, CosmosWorks).

  These extensions promise to enable the real-time simulation to general large-scale nonlinear (parametrized) systems, which currently are computationally intractable.
Appendix A

Computational Details for the

Model Problem: Burgers’ Equation

A.1 Weak form of Burgers’ equation

First, divide the domain \( \Omega = [0,1] \) into equal \( n_x \) elements, called \( T_h \). Let \( v \in X = \{ w \in H^1(\Omega); w|_{T_h} \in P_1(T_h) \forall T_h \text{ and } w = 0 \text{ on } \partial \Omega \} \). To write variational form of (5.1), multiply \( v \) through out (5.1), integrate over the domain \( \Omega \), and using integrate by parts:

\[
\int_\Omega \frac{\partial}{\partial t} y(x, t)v(x)dx - \nu \int_\Omega \frac{\partial^2}{\partial x^2} y(x, t)v(x)dx + \int_\Omega \frac{\partial}{\partial x} s(y(x, t))v(x)dx = 0 (A.1)
\]
Note that
\[
-\nu \int_\Omega \frac{\partial^2}{\partial x^2} y(x, t) v(x) dx = - \left( \underbrace{[\partial_x y(x, t) v(x)]_{\partial\Omega}}_{=0} - \nu \int_\Omega \frac{\partial}{\partial x} y(x, t) v'(x) dx \right)
\]
\[
= \nu \int_\Omega \frac{\partial}{\partial x} y(x, t) \frac{d}{dx} v(x) dx
\]
and
\[
\int_\Omega \frac{\partial}{\partial x} s(y(x, t)) v(x) dx = \left[ s(y(x, t)) v(x) \right]_{\partial\Omega} - \int_\Omega s(y(x, t)) \frac{d}{dx} v(x) dx
\]
\[
= - \int_\Omega s(y(x, t)) \frac{d}{dx} v(x) dx
\]

Above we used the fact that \( v|_{\partial\Omega} = 0 \).

### A.2 Precomputing the nonlinear term

The following facts mentioned earlier are used.

1. \[
\frac{d}{dx} \psi_i(x) = \begin{cases} 
\frac{1}{h}, & x \in [x_{i-1}, x_i], \\
-\frac{1}{h}, & x \in [x_i, x_{i+1}], \quad i = 1, \ldots, n_y \\
0, & \text{otherwise}
\end{cases}
\tag{A.2}
\]

2. \[
\int_{x_p}^{x_{p+1}} \psi_k(x) \psi_j(x) dx = \begin{cases} 
\frac{h}{3}, & \text{if } (k, j) = (p, p) \text{ or } (k, j) = (p + 1, p + 1) \\
\frac{h}{6}, & \text{if } (k, j) = (p + 1, p) \text{ or } (k, j) = (p, p + 1) \\
0, & \text{otherwise}
\end{cases}
\]

where \( p = 1, \ldots, n_y \).
Let \( u_i = [u_{i,1}, \ldots, u_{i,n_y}]^T \). The cases for the nonzero entries of \( T_i \) are:

- If \( p = q, p = 2, \ldots, n_y - 1 \)

\[
(T_i)_{pp} = \int_{x_{p-1}}^{x_p} \psi_p(x)\psi_p(x) \frac{d}{dx} \psi_{p-1}u_{i,p-1}dx \\
+ \int_{x_{p-1}}^{x_{p+1}} \psi_p(x)\psi_p(x) \frac{d}{dx} \psi_{p}u_{i,p}dx + \int_{x_p}^{x_{p+1}} \psi_p(x)\psi_p(x) \frac{d}{dx} \psi_{p+1}u_{i,p+1}dx \\
= -\frac{u_{i,p-1}}{h} \int_{x_{p-1}}^{x_p} \psi_p(x)\psi_p(x) dx + \frac{u_{i,p+1}}{h} \int_{x_p}^{x_{p+1}} \psi_p(x)\psi_p(x) dx \\
= -\frac{u_{i,p-1}}{3} + \frac{u_{i,p+1}}{3}.
\]

Note that

\[
\int_{x_{p-1}}^{x_{p+1}} \psi_p(x)\psi_p(x) \frac{d}{dx} \psi_{p}u_{i,p}dx = \int_{x_{p-1}}^{x_p} \psi_p(x)\psi_p(x) \frac{d}{dx} \psi_{p}u_{i,p}dx + \\
\int_{x_p}^{x_{p+1}} \psi_p(x)\psi_p(x) \frac{d}{dx} \psi_{p}u_{i,p}dx \\
= -\frac{u_{i,p}}{h} \int_{x_{p-1}}^{x_p} \psi_p(x)\psi_p(x) dx \\
+ \frac{u_{i,p}}{h} \int_{x_p}^{x_{p+1}} \psi_p(x)\psi_p(x) dx \\
= 0.
\]

- If \( p = q - 1 \), for \( p = 2, \ldots, n_y \)

\[
(T_i)_{p,p-1} = \int_{x_{p-1}}^{x_p} \psi_p(x)\psi_{p-1}(x) \frac{d}{dx} \psi_{p-1}u_{i,p-1}dx + \int_{x_{p-1}}^{x_p} \psi_p(x)\psi_{p-1}(x) \frac{d}{dx} \psi_{p}u_{i,p}dx \\
= -\frac{u_{i,p-1}}{h} \int_{x_{p-1}}^{x_p} \psi_p(x)\psi_{p-1}(x) dx + \frac{u_{i,p}}{h} \int_{x_{p-1}}^{x_p} \psi_p(x)\psi_{p-1}(x) dx \\
= -\frac{u_{i,p-1}}{6} + \frac{u_{i,p}}{6}.
\]
\[ (T_1)_{p,p+1} = \int_{x_p}^{x_{p+1}} \psi_p(x)\psi_{p+1}(x) \frac{d}{dx} \psi_p u_{i,p} \, dx + \int_{x_{p+1}}^{x_{p+1}} \psi_p(x)\psi_{p+1}(x) \frac{d}{dx} \psi_{p+1} u_{i,p+1} \, dx \]
\[ = -\frac{u_{i,p}}{h} \int_{x_p}^{x_{p+1}} \psi_p(x)\psi_{p+1}(x) \, dx + \frac{u_{i,p+1}}{h} \int_{x_{p-1}}^{x_p} \psi_p(x)\psi_{p+1}(x) \, dx \]
\[ = -\frac{u_{i,p}}{6} + \frac{u_{i,p+1}}{6}. \]

For \( p = 1 \),
\[ (T_1)_{1,1} = \int_{x_0}^{x_1} \psi_1(x)\psi_1(x) \frac{d}{dx} \psi_1 u_{i,1} \, dx + \int_{x_1}^{x_2} \psi_1(x)\psi_1(x) \frac{d}{dx} \psi_2 u_{i,2} \, dx \]
\[ = \frac{u_{i,2}}{3}. \]

For \( p = n_y \),
\[ (T_1)_{n_y,n_y} = \int_{x_{n_y-1}}^{x_{n_y}} \psi_{n_y}(x)\psi_{n_y}(x) \frac{d}{dx} \psi_{n_y} u_{i,n_y} \, dx + \int_{x_{n_y-1}}^{x_{n_y}} \psi_{n_y}(x)\psi_{n_y}(x) \frac{d}{dx} \psi_{n_y} u_{i,n_y} \, dx \]
\[ = -\frac{u_{i,n_y-1}}{3}. \]

### A.2.1 Varying Dimensions of the EIM for the Nonlinear Term

In Chapter 6, the EIM-POD approach uses one fixed dimension for the EIM, which is chosen heuristically. The followings are the preliminary numerical results comparing EIM with different dimensions.
Figure A.1: Errors from using different dimensions for EIM-POD basis

Figure A.2: CPU times used for solving the reduced-order systems constructed from POD basis of size 1 to 70 with the different three approaches and different dimensions for the EIM-POD basis
Bibliography


