Adjoint and Acceleration Methods for Projection-based Reduced Order Modeling

by

Gary Collins

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Aerospace Engineering)
in the University of Michigan
2022

Doctoral Committee:

Professor Carlos E. S. Cesnik, Co-Chair
Professor Krzysztof J. Fidkowski, Co-Chair
Professor Karthik Duraisamy
Assistant Professor Aaron Towne

Gary Collins

ggggggggg@umich.edu

ORCID iD: 0000-0003-2476-3208

DEDICATION

To my family: Mom, Dad, and Hayley.

And to Natalie.

ACKNOWLEDGEMENTS

Writing this dedication has been a difficult exercise because every time I think back through this journey I find more and more people that I would like to thank. I find that it will be easiest to go through these acknowledgements chronologically; however, I would like to first thank Professor Krzysztof Fidkowski (Chris) and Professor Carlos E.S. Cesnik. Chris and Carlos have been absolutely fantastic advisors – endlessly helpful, supportive, and approachable. I am amazed just as much by Chris's ability to explain and reason through almost everything as I am by his good-nature and warm affability. Carlos cares deeply about his students, their work and their lives. I am extremely grateful for that support, his vision, and especially his humor through my time at Michigan. I am truly blessed to have had such incredible and good people as my advisors. I would also like to thank the rest of my committee: Professor Karthik Duraisamy and Assistant Professor Aaron Towne. Karthik is a fantastic researcher and educator. I have many fond memories from his courses, and he carries a welcoming attitude and warm persona. Aaron has been a helpful committee member despite the circumstances, and I am grateful for his effort and knowledge. Additionally, I would like to thank Hans Bleecke and Reik Thormann who were my primary collaborators at Airbus that helped oversee this work. Our Monday meetings were always a bright start to the weekend.

Before everything else, I would also like to acknowledge my family. Mom, Dad, and Hayley, no matter what happens we only have each other. You have all supported me immensely: Dad from taking me to science camps as a kid to dropping me off at college, Mom from teaching me basic math and science to helping me choose graduate schools, Hayley from bringing me to middle school everyday to having my back throughout my entire academic career. Thank you and I love you.

From my hometown, I would like to acknowledge my friends Michael and Meyer. You guys are truly friends for life and I am so happy that we continue to stay in touch and hang-out. Some of my favorite memories revolve around you: playing board games, watching UTK and Vandy football, going to the Ren Faire to watch the juggling act for the n^{th} time, etc; and even despite our distances and lack of social media, we still find a way to meet up. Next I would like to thank Dr. Hartwig Anzt. I met Hartwig as an undergrad at the University of Tennessee where he was a high performance computing researcher with the

Innovative Computing Laboratory. He saw a lot of potential in me and without him and his encouragement, I do not know if I would have pursued this degree. Furthermore, I would also like to thank from my time at the University of Tennessee my roommates (Ben, Joseph, Stephen, and Alex); Profs. Bond, Dongarra, Ekici, Kit, Lyne, and Pionke; and my many friends (Daniel, Autumn, Kyle, Jared, Kaleigh, Sam, Paulson, and many others). Among these people are the Scott's Tots, the "best" trivia team at the Black Horse Pub and Brewery. I looked forward to trivia night every week. Beyond winning and losing, it was the only consistent time we got to disconnect from our phones/work and focus on being friends and grow as people.

There are many people I would also like to thank from the University of Michigan. Office 2001 will forever be the best office in the entire FXB building, and I was extremely happy to share it with many people: Vivek Ojha, Qingzhao Wang, Kevin Doetsch, Matteo Franciolini, and Gustavo Halila. Vivek, who was a part of my cohort and also co-advised by Chris and Carlos, is a friend for life. Helping him practice driving will still be some of my fear-filled moments and going out to Detroit with him for shows will still be some of my most fond. Vivek and I would not be complete without our partner in crime, Matteo. He's an incredibly impressive person. I remember watching him do over a hundred push-ups in a row one day like we were on Mars. He is a person who is honest about his tastes and preferences, especially about his coffee (to Italian, Americans drink "acqua sporca" (*i.e.*, *dirty water*) and a lot of my understanding of Italian culture comes from him. I miss him as a roommate. Qingzhao has been an incredible co-worker and friend. We would often debrief after our Monday meetings in Office 2001 with equal parts chit-chatting and planning how to tackle the work ahead. I can say that without her help and advise, much of this work would not have been possible. I want everyone who has read this far to know that Gustavo is one of the nicest people I know and I feel like I can talk with him endlessly. Our video calls have been a great relief during the pandemic.

In addition to my Office 2001 friends, I also want to acknowledge how grateful I am to have interacted with the rest of the research groups under Chris and Carlos: Devina Sanjaya, Yukiko Shimizu, Guodong Chen, Kyle Ding, Rahul Halder, Bilal Sharqi, Braden Frigoletto, Divya Sanghi, Cristina Riso, among others. Devina Sanjay was my advocate for going to the University of Michigan and is a great friend. I first met her on a visit day in 2016 and am still in touch with her now as she is a professor at my old Alma Mater. She is the kind of person who is always honest and is always looking out for what is best for others. Also she is a fantastic baker! Yuki Shimizu is good friend, kind soul, and someone I bonded a lot over our experiences as semi-Korean-Americans and only wish for her to meet others as good and kind as she is. Divya and I wrote significant portions of our

respective theses during writing sessions over video, and I got to learn a lot more about her and how we share a lot of common values. I am very grateful for her keeping me dedicated to writing and wish her the best of luck in her future career. I would also like to thank my friends from Professor Boyd's group for still letting me enjoy your ultra-exclusive coffee breaks and also letting me join the **second** best trivia team this side of the Mississippi: Alex Vazsonyi, Mike Holloway, Kaelan Hansson, Pawel Sawicki, and Ross Chaudhry. Lastly I would like to thank Chris Wentland, Ben Brelje, Anıl Yıldırım, Puneet Singh, Brittany Essink, and Francisco "Panchito" Ramirez Rueda.

Lastly, I would like to acknowledge Natalie Belew and our fur friends, Cleo and Livi, for supporting and loving me for the last leg of this journey. I am so extremely blessed to have you in my life and look forward to our future together.

This material is based upon work supported by Airbus in the frame of the Airbus-Michigan Center for Aero-Servo-Elasticity of Very Flexible Aircraft (CASE-VFA).

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Computational modeling is a pillar of modern aerospace science and is increasingly becoming more important as computer technology and numerical methods grow more powerful and sophisticated. However, computational modeling remains expensive for many aerospace engineering problems, including high-fidelity solutions to highly-complex, three-dimensional unsteady vehicle simulations, large-scale aeroservoelastic control problems, and multidisciplinary design optimization. Reduced-order models (ROMs) have therefore garnered interest as an alternative means of preserving high fidelity at a much lower computational cost. Among these methods is the class of projection-based ROMs, which utilizes the original physics and equations of the high-fidelity system but resolves the state projected onto a lower-order trial manifold with the system projected onto a low-order test manifold. The low-order trial manifold is typically chosen to be a linear basis, and this thesis focuses on ROMs that use proper orthogonal decomposition (POD) and the method of snapshots in order to formulate the linear state basis. Hyper-reduction is also necessary to reduce the complexity of nonlinear problems, and this thesis is focused on the the discrete empirical interpolation method (DEIM), an interpolation method that uses a sparse sampling of the nonlinear function values. The benefit of the method of snapshots is that given a representative set of solution samples, a linear basis that projects the solution space with very low error can be constructed. However, accuracy in state projection is only one part requirement for a ROM to be useful for engineering applications.

Low errors in state projection do not necessarily mean that outputs of interest are accurately predicted as the proportion of the domain that is used to calculate the output may be very small relative to the entire domain. Quantification of the output error is thus important to assess the quality of a ROM. Methods for estimating output error for POD-DEIM

models exist; however, the application of these methods to fine-grain adaptation is limited. Furthermore, the commonly used Galerkin formulation of ROMs, where the test and trial spaces are the same, is known to be inaccurate for many problems. These inaccuracies arise from the inability of a state basis to appropriately project the physical system. However, the construction of a tailored Petrov-Galerkin test basis that yields the appropriate dynamics from system projection is not trivial. Finally, the implementation of ROMs for engineering applications may be difficult due to their intrusive nature. Projection-based models require the ability to call subroutines of the original high-fidelity model; however, code modification may be complex and hindered by intellectual property and export control protections.

The research presented investigates the use of adjoint-based methods to achieve those goals. For error estimation, this thesis applies adjoint-weighted residuals in order to assess output error, presents methods for localizing the contribution of output error to individual ROM degrees of freedom, and derives adaptation schemes using those error localizations. For constructing dynamically useful test bases, this thesis derives a novel test basis that yields dynamics from system projection that minimize the state error of the ROM. This test basis is composed of the reduced state adjoints and stability and convergence studies of this test basis are presented. Additionally, to overcome issues of portability and intrusive implementation, a novel hybridization of machine learning and hyper-reduced methods is presented that supplants the intrusive portions of projection-based model reduction with element-level neural networks. The implementation of the neural networks at a low level allows for ROM time-marching to be used, which appropriately propagates the influence of states on future times.

CHAPTER 1

Introduction

This introductory chapter provides the motivation for the work contained in this dissertation as well as includes a literature review of several reduced-order modeling methods. Although a more in-depth literature review is provided with each technical chapter, the articles presented here provide the context and motivation for the research endeavors undertaken here. Following these sections, an overview of the structure of this dissertation is provided.

1.1 Motivation

A revolution in computational aviation has taken place over the last three decades. In the early 1990s, the first generation of commercial aircraft to be "fully-developed" with the aid of computational fluid dynamics (CFD) and computer-aided design (CAD) software was entering the market, with the Airbus A330/340 in 1993 and the Boeing 777 in 1994 [2]. For the Boeing 777 aircraft, the fuselage and cabin were entirely designed with CFD. However, for the more complicated components of the aircraft, like the wing-body fairing, engine placement, control surfaces, and tail, wind tunnel tests were still instrumental, with CFD serving to correct mounting and wall effects [3]. The design process cycled between using CAD to orientate and verify realizability, using CFD-augmented wind tunnel tests to obtain the aerodynamic characteristics of the design, then iterating on the design to remove undesirable characteristics. Difficulties in meshing, complications with modeling flow separation, and limited memory prevented the use of CFD for the entire aircraft. However, the modern age of high-performance computing was only beginning. The fastest machines listed by the TOP500, at the time a newly created organization for listing the world's most powerful supercomputers, just broke the 100GFLOPS ceiling [4]. Since that time, the scale of computing power has exponentially increased 10 million fold as shown by Figure 1.1. Today, even the most affordable "commodity" processors would top the Top500 list. Furthermore, in June 2022, Oak Ridge National Lab's "Frontier" supercomputer became first

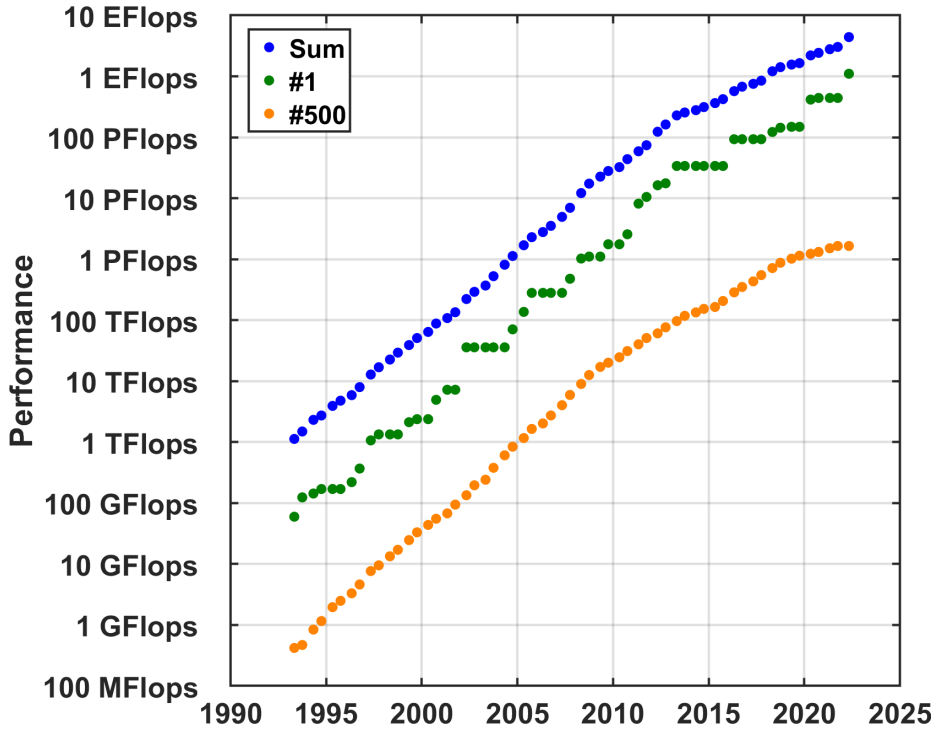


Figure 1.1: History of the performance of the TOP500 supercomputers by year. This chart was created by the TOP500 organization [1].

true exascale machine listed on the semi-annual TOP500 list [1].

The scale of computing power and advances in numerical methods available for engineering applications today tower over what was available when the Boeing 777 and A330/340 aircraft were developed. As a result, the number of wind tunnel tests needed for the design of aircraft has decreased substantially [3]. Computational fluid dynamics (CFD), computational structural mechanics (CSM), and fluid-structure interaction (FSI) have become important pillars in the design process of modern aircraft. Yet, despite these advances in computation and numerics, the demand for high-fidelity aerodynamic analysis still outweighs the speed at which they can be developed. Among NASA's 2030 Grand Challenges is the use of high-fidelity multidisciplinary design analysis and optimization for highly flexible aircraft configurations [5]. The solution to this Grand Challenge will need to be able to generate time-accurate CFD solutions to coupled aero-servo-elastic problems for highly flexible aircrafts in an engineering time-frame to allow for the optimization of complex aircraft configurations. The process of designing, analyzing the design, and iterating on the design is exactly the same process that was undertaken to design the Boeing 777 and the A330/340, but with a wider set of possible configurations and using fully-automated/computerized methods. Many other types of problems exist in computational aerospace, including real-time model predictive control, aero-structural-combustion

settings, and flight parameter sweep, all of which require fast-turnaround time of highly-complex solutions. Model reduction techniques have therefore garnered a large amount of interest for efficiently generating high-fidelity solutions. Model reduction leverages mathematical and computational methods in order to generate approximate solutions to high-fidelity problems at a higher speed while maintaining accuracy. However, for a reduced-order model (ROM) to be practically applicable, it must have several characteristics:

1. **Accuracy**
2. **Stability**
3. **Speed**

While these points seem obvious, achieving them is not trivial. In many settings, increasing the accuracy of a reduced-order model by increasing training or degrees of freedom may harm its stability, while at the same time increasing the stability of a ROM by reducing its degrees of freedom may harm its accuracy. A key to achieving accurate ROMs is to define what sort of accuracy is demanded. For engineering applications, output accuracy and state accuracy are hand in hand. Currently, with regards to projection-based models, methods for precise output error estimates and output-driven adaptation schemes are under-investigated. Further, although projection-based models are able to span the state space with the construction of a basis through decomposition of solution samples, obtaining the correct dynamics to drive the ROMs towards accurate state solutions is difficult. This thesis will demonstrate methods for achieving output and state accuracy for even coarse ROMs. This is approached through adjoint-weighted residual error estimation, localization, and adaptation, and with the use of adjoints-based ROM construction. Additionally, increasing accuracy through increasing the size of the ROM or through the inclusion of modeling physical constraints impacts the efficiency that a model can produce solutions. Further, implementation of projection-based ROMs requires access to the original high-fidelity system, which can be difficult or even impossible to access. However, machine learning can accelerate projection-based ROMs without accessing the high-fidelity system and, when combined with the adjoint-based methods introduced in this thesis, are able to efficiently produce accurate solutions. The remainder of this introduction is meant to introduce basic model reduction concepts and identify problems with current methods as well as establish an outline for how this thesis will approach these problems.

1.2 Model Reduction

Model reduction is a class of mathematical techniques that reduces the complexity of generating high-fidelity solutions to expensive physical models. The term *full-order model*

(FOM) is often used to referred to the high-fidelity model that is reduced. For this thesis, the full-order model is the fully-discretized model with all of its degrees of freedom and governed by partial differential equations (PDEs); however, model reduction can be performed in continuous settings and also on real-world data. Most approaches are divided into *offline* and *online* stages which serve to construct and deploy the ROM, respectively. During the offline stage, solutions of the FOM are computed at various configurations and then used for constructing the ROM. During the online stage, trained ROM can be used to predict solutions for new configurations not encountered in training. Although not universal, a common categorization of reduced-order models is into those that are interpolation-based and those that are projection-based. Interpolation-based model reduction attempts to create surrogate models that map system inputs to outputs of interest. These methods can be seen as "black-box" models that do not invoke the original physical equations or develop any intermediate physical state between the inputs and outputs. These methods include polynomial fitting [6], radial basis functions [7, 8], Volterra series [9, 10], Kriging interpolation [11, 12], and machine learning [13, 14]. Interpolation-based models offer highly efficient, stable solutions. However, because surrogate models do not invoke the underlying physical system, substantially more training data is require to construct these models. In addition, the error of these models is difficult to quantify, and often statistical techniques are used to infer the error of the model [15].

1.2.1 Projection-based Models

On the other hand, projection-based techniques attempt to satisfy the underlying physical equations with a solution that is restricted to a lower-order trial manifold while using a low-order test manifold for projecting the physical model. These manifolds are typically linear, using bases obtained from solutions of the FOM. Effectively the degrees of freedom of the problem are reduced while retaining information of the entire physical state vector that can be used to obtain outputs of interest. However, these models are usually highly-intrusive, requiring access to FOM codes in order to invoke the same physical models. Projection-based methods include the proper orthogonal decomposition method (POD) [16, 17], moment matching [18, 19], balanced truncation [20, 21], dynamic mode decomposition [22], and spectral proper orthogonal decomposition [23]. For linear problems, the projection of the state and the system onto the trial and test manifolds also effectively reduces the complexity of the system of equations; however, this is not true for nonlinear problems. For these applications, the nonlinear portions of the physical equations are evaluated on the low-rank, full-sized reduced state; this also applies to the formation of linearizations of the system that are needed for a nonlinear solver, such as Newton-Raphson.

These calculations dominate the computational cost, and the savings from projection are effectively lost. To remedy this issue, many hyper-reduced methods have been introduced to reduce the cost of nonlinear function evaluations through sparse sampling and interpolation. These include continuous and discrete interpolation methods (EIM/DEIM) [24, 25], the missing point approximation [26, 27], the best points interpolation method [28], and the Gauss-Newton with approximate tensors method [29]. This thesis uses the POD and DEIM methods for model reduction and the method of snapshots [16] for constructing the linear bases for the ROMs. In this approach, a singular value decomposition (SVD) of a set of full-order model solution samples, called snapshots, is applied, as it satisfies the L2 minimization of the projection error of all possible bases of the snapshot set. The basis vectors from SVD are ordered with their corresponding singular values, and the basis vectors with very small associated singular values are neglected to keep the ROM small and to remove potential numerical error. POD uses a linear basis to project the state and system, and DEIM hyper-reduction approximates the nonlinear portions of the system with a projected interpolation constructed from nonlinear function samples at degrees of freedom that best estimate the system. The linear basis for DEIM hyper-reduction is a truncated set of singular vectors from an SVD of the nonlinear function snapshots.

1.2.2 Error Estimation for Projection-based ROMs

Given that the solution of the ROM is constructed from a low-rank approximation, quantification of the error of projection-based models has garnered significant attention. Sharp *a posteriori* error bounds on the state errors have been studied extensively within the reduced basis method community [30, 31, 32, 33, 34]. In other procedures, error bounds are formed based on the trajectory of the kernel of the trial basis [35, 20, 36]. Reduced-order model analogs of adjoint-weighted residual error estimates have been used for POD [37], h-refinement of a POD system through vector splitting [38], and adaptation of the empirical quadrature procedure [39, 40] – a hyper-reduction method for the reduced basis formulation.

Current approaches to DEIM error estimation rely on establishing *a priori* [41] or *a posteriori* [42, 43, 44] error bounds. Accurate output predictions are important for engineering applications; however, for DEIM-ROMs, adaptation that specifically targets output prediction is under-investigated. Feng, Mangold, and Benner presented an approach to output adaptation that uses an error bound that approximates the adjoint-weighted residual equation [44]. The approximation of the adjoint solution is performed in order to avoid the cost of resolving time-coupled adjoints. However, resolving the time-coupled adjoints is important in order for error to appropriately propagate through time. Furthermore, while

the error bounds for DEIM-ROMs are rigorously derived, error bounds are unable to finely attribute sources of error. Rather, adaptation mechanisms update state and nonlinear function bases with priority on basis vectors with higher associated singular values. Thus, even *a posteriori* error estimates use *a priori* information to update the model without incorporating information about online performance of the ROM beyond the formation of error bounds. In addition, given that the singular values are known *a priori* and also used for ROM construction, it is typical that ROMs are already constructed with a vast proportion of the available singular values. Thus, adapting based on singular values may be inefficient, as basis vectors that do not contribute significantly to either state or output prediction may be added. Additionally, higher frequency or numerical error polluted basis vectors may be added, making the ROM stiffer as it must dissipate the effects of these basis vectors. However, adjoint-based methods [45, 46, 47] can be used to identify specific degrees of freedom that contribute to output error and thus can be used as an additional metric for adaptation. Chapter 3 presents the time-coupled adjoint equations for DEIM-ROMs, develops ways to finely assess output prediction error contribution from various DEIM-ROM characteristics, and demonstrates adaptive methods using those estimates. By doing so, DEIM-ROMs are constructed that accurately predict outputs, while maintaining coarseness and stability. Some of the work presented here has already been published and presented, notably at the American Institute of Aeronautics and Astronautics (AIAA) Aviation 2019 conference [48], and at the 15$^{\text{th}}$ U.S. National Congress on Computational Mechanics.

1.2.3 Petrov-Galerkin Test Basis Formulation

For most applications of projection-based model reduction, stability properties of the FOM do not necessarily carry over to the ROM; however, a wealth of literature exists on the stabilization of projection-based ROMs [49, 36, 50, 51, 52, 53]. Chapter 4 of this thesis is interested in the formation of Petrov-Galerkin models for improving stability and state accuracy. Galerkin formulations, where the trial and test manifolds are the same, are especially prone to instability as the trial manifold may be a poor projection of the system and enforcing the system to be only resolved in the trial manifold can allow for unbounded residuals in the kernal of the manifold [20]. Thus Petrov-Galerkin constructions, where separate trial and test manifolds do not equal, are considered more a viable and stable approach for projection-based models for aerospace settings mainly due to the test basis being better suited for projecting the system than the trial basis [54]. A common approach is to use a test basis that makes the ROM solve the FOM residual minimization where the state exists on the trial manifold [55, 56]. This formulation, referred to as the minimum residual and least squares Petrov-Galerkin (LSPG) method, is the basis for the

Gauss-Newton with approximate tensors hyper-reduced method. LSPG has been shown to be more accurate and stable than the Galerkin ROM approach for many problems and guarantees stability for linear, unsteady problems as the temporal linearization of the ROM is made symmetric, yielding only real eigenvalues [29, 57, 58]; however, LSPG-ROMs are shown to be sensitive to time step size and become equivalent to Galerkin ROMs for explicit time-marching schemes [57]. Recent work by Parish, Wentland, and Duraisamy investigated the construction of a Petrov-Galerkin test basis for closure of a POD-ROM model in order to improve stabilization and accuracy, named the Adjoint Petrov-Galerkin method (APG) [59]. Closure in this sense comes from the turbulence-modeling inspired formalization of projection-based ROMs into resolved and unresolved scales. The resolved scales are the solution to the ROM, while the unresolved scales (orthogonal to the trial basis) are modeled [60, 61, 62]. The chosen model arises from the Mori-Zwanzig formulation, where an integrated memory term serves to close the model [63]. This memory term is typically intractably expensive to evaluate, and the Petrov-Galerkin test basis arises through its approximation.

Taking a step back, it is important to recognize that Galerkin ROMs fail due to the inability of the test space to yield the appropriate dynamics to drive the state towards the optimal solution. If a snapshot set is a good representation of the solution space and given a basis that is able to span the snapshot set with very low error, then that basis should be able to span the entire solution space with very low error. However, there is no guarantee that such a basis is able to yield the correct dynamics of the system when used as a test space. Recognizing that the test basis is the mapping of the entire system to a reduced space, the test basis defines what the reduced system is ultimately trying to achieve. For APG, the goal is to obtain dynamics from the unresolved state. For the case of LSPG, the goal is to obtain dynamics that will solve the minimization of the residual. However, a test basis can also be constructed such that the projection of the system yields dynamics that minimizes the state error. Chapter 4 presents the novel Petrov-Galerkin test basis that is constructed in order to optimally obtain state accuracy. This test basis is formulated through the use of reduced state adjoints, and thus this approach is called the state adjoint Petrov-Galerkin method (SAPG). This is inspired by work from Demkowicz and Gopalakrishnan [64] and Kast and Fidkowski [65, 66] on the formulation of optimal discontinous Petrov-Galerkin finite element models. The SAPG test basis incorporates new information from the entire system by requiring a linear system to be solved. However, a restriction to the ROM is that the linear system is of the full-order size. To keep the benefits of incorporating new information into the test space while making the SAPG model usable for model reduction, a reduced-order model of the SAPG test space is presented. This reduced-order model

allows for the use of a test search basis, composed of any basis such as the truncated modes or the adjoints, to be used to form the SAPG test basis. Work first presented at the AIAA Scitech 2020 conference for POD-ROMs [67] is expanded in this thesis to hyper-reduced models, with analysis of the stability and convergence effects of the model.

1.2.4 Hybridized Projection-based ROMs and Machine Learning

Machine learning is a field of computational and mathematical techniques that utilizes computers to efficiently apply mathematical algorithms for constructing sophisticated surrogate models, such as neural networks, from observed quantities. The use of machine learning to enhance projection-based ROMs has garnered a lot of recent interest due to their ability to identify nonlinear relationships between system features and outputs of interest and also due to the availability of many open-source, fast machine learning libraries. These methods include the use of neural networks for POD-ROM closure modeling [60, 68, 69, 61, 62]. However, the final section of this thesis is focused on the creation of hybridized ROMs that avoid the need of accessing the high-fidelity system that is required for projection-based ROMs. Additionally, implementation of hyper-reduced methods often requires additional low-level code to be written in order to obtain sparsely sampled nonlinear function evaluations. Direct access to CFD code is not guranteed as barriers, such as intellectual copyright protections and export controls, may prevent any amount of low-level access that is needed for implementing projection-based model reduction. Additionally, the age ("legacy codes") and complexity of CFD codes can also hinder the implementation of projection-based ROMs. Issues of portability are also a concern, as being intrusively implement also means that projection-based ROM codes are difficult to reuse for different codes without whole rewriting of many subroutines. However, many approaches have used neural networks to create non-intrusive projection-based ROMs that avoid direct code access. A common method is to map the system inputs to POD coefficients [70, 71, 72, 73, 74, 75, 76]. In this fashion, the neural networks are trained to learn the dynamics of the physical model but act as black-boxes online. Other approaches attempt to use machine learning to improve the underlying physical modeling of projection-based ROMs but not for directly computing POD coefficients. This includes the use of unsupervised learning for POD-ROM operator inference [77, 78, 79], convolutional autoencoders for nonlinear trial manifolds [80, 76], and for residual/error modeling [81, 82, 71]. In the same vein, Chapter 5 pertains to the application of machine learning methods to non-intrusive ROMs that do not only generate POD coefficients. Time-marching the physical system is important in order for errors to be appropriately dissipated and for the influence of a state to propagate appropriately through time. When neural networks are used for POD coefficients in an unsteady setting, special

care has to be taken to allow for information to be propagated correctly. Recurrent neural networks (RNNs) use a feedback of an output hidden state in order to propagate information through time. However, these systems are susceptible to the vanishing and exploding gradient phenomena, where the influence of the inputs and hidden state of a time node quickly diminishes or the hidden state of the system grows non-physically large. Certain RNN formulations, such as the long short term memory unit, can alleviate some of these issues but have many limitations such as a fixed time step size, sensitivity to memory length, and the need of initial memory accumulation. Instead, this thesis presents a non-intrusive ROM model that replaces the FOM residual and Jacobian computation, lowest-level intrusive subroutines of the DEIM method, with neural networks. These computations are then used in the DEIM-ROM for time-marching a state. In this fashion, the mechanism for the state propagation in time is unmodified. The neural networks are embedded at the elemental level and only receive as inputs the local and neighboring states as well as system parameters. Thus, the neural networks are trained to learn only the physical model that maps states to spatial residuals for a particular element. These quantities are already available for ROM construction, so no modification of the FOM is needed. The resulting model is compared to the traditional DEIM and shows comparable accuracy and $\mathcal{O}(10)$ improvement in speedup.

1.2.5 Recent Model Reduction Achievements

Several recent applications of model reduction have had noteworthy results:

- Washabaugh, Zahr, and Farhat applied projection-based techniques to developing the steady solutions to the NASA Common Research Aircraft Model with shape variations in viscous, transonic flow. They attained a reduction of the degrees of freedom from 69 million to only 23, and a speedup from 2 hours on a 1024 core system to 3 minutes on a Apple Macbook [83].

- Ripepi *et al.* formulated various intrusive and non-intrusive, interpolation-based and projection-based models for predicting the aerodynamic response for a fully-coupled CFD-CSM solver under the DLR Digital-X project. The parameterized aerodynamic ROMs were able to predict the integrated and distributed loads with less than 1% error for various flight conditions (Mach, angle of attack, slip angle, and altitude) with accurate shape deflection for a wing-fuselage half-body model obtained from FSI [84].

- Zhou *et al.* used linear genetic programming to design a controller to optimize the mixing of a turbulent air jet. The controller was trained with real-time experimental

data and provided real-time inputs to radial-placed actuators on the nozzle of the jet. The controller sequentially learned various known mixing patterns in the order of effectiveness before settling on a previously unknown but highly effective pattern. This result is included as it is an example of machine learning that is not based on a neural-network and that the training and testing data comes from observed experimental data rather than a computer model. [85]

1.3 Outline

This thesis consists of four technical chapters. Chapter 2 details the projection-based models that are used for the development of the algorithms central to the thesis: the POD method and the DEIM method. The remaining three technical chapters are focused on uses of adjoint-based and machine learning methods for improving on the three characteristics mentioned above, *i.e.*,

1. **Accuracy** $\rightarrow$ Output error estimates and adaptation for DEIM-ROM model (Chapter 3). Construction of state optimal Petrov-Galerkin models (Chapter 4).

2. **Stability** $\rightarrow$ Adaptive techniques that increase accuracy but retain coarseness (Chapter 3 and 4).

3. **Speed** $\rightarrow$ Element-embedded neural networks (Chapter 5).

The applications of the methods developed in this thesis pertain primarily to unsteady loads prediction for aeroelastic analysis and design with Euler and unsteady Reynolds average Navier-Stokes physics. Applications to unsteady, resolved turbulent flows and combustion problems are of interest but not addressed here. Chapter 3 focuses on the extension of adjoint-weighted residual error estimates to DEIM-ROMs, developments of fine-grain output error localization methods for DEIM-ROMs, and adaptive methods based on that quantification. Chapter 4 presents a novel Petrov-Galerkin test basis that yields dynamics from the projected system that drive the ROM to the optimal state solution. Formulations for hyper-reduced problems and a reduced model of the test space are also provided. Chapter 5 provides a new method for using machine learning for non-intrusive ROMs that incorporates the time marching of the ROM, which allows for appropriate state propagation in time and drastically increases ROM speed up. Finally, Chapter 6 summarizes the main observations from the earlier chapters, the key contributions of this dissertation, and presents ideas for related future work.

CHAPTER 2

Projection-based Reduced Order Models

Projection-based reduced-order modeling is a key concept for this thesis. Qualitatively, reduced-order modeling is a class of systematic techniques for reducing the complexity of generating high-fidelity solutions to computationally expensive problems. Further, projection-based ROMs are mathematical approaches to reducing the complexity of systems of differential equations, and in a common setting, discrete formulations of PDEs. As a consequence of the mathematical foundation of projection-based models, even though this thesis focuses on a variety of computational fluid dynamic problems, ultimately the ROMs studied in this thesis can be applied to problems arising from most scientific formulations of the same form. Therefore, for this section, a general dynamical system of equations will be defined and used for deriving the various projection-based ROMs of interest. These are the proper orthogonal decomposition (POD) and the discrete empirical interpolation method (DEIM).

2.1 Generic Dynamical System

A system of ordinary differential equations, generally arising from spatially-discretized partial differential equations, can be written as

$$G : \begin{cases} \boldsymbol{M}\frac{d\boldsymbol{x}}{dt} + \boldsymbol{R}(\boldsymbol{x}(t), \boldsymbol{\mu}(t), t) = \boldsymbol{0}, \\ \boldsymbol{y}(t) = \boldsymbol{J}(\boldsymbol{x}(t), \boldsymbol{\mu}(t), t). \end{cases} \tag{2.1}$$

This definition serves as the full-order model (FOM) that is to be reduced, and generally, most scientific problems can be written in this setting, with $\boldsymbol{x} \in \mathbb{R}^{n_{\boldsymbol{x}}}$ a vector of state variables that are dynamically changing in time driven by the function $\boldsymbol{R} \in \mathbb{R}^{n_{\boldsymbol{x}}}$, whose inputs are the state itself, a set of inputs $\boldsymbol{\mu} \in \mathbb{R}^{n_{\boldsymbol{\mu}}}$, and time t. For many applications, the trajectories of the outputs of the system ($\boldsymbol{y} \in \mathbb{R}^{n_{\boldsymbol{y}}}$) are of concern and are defined here as a vector function $\boldsymbol{J}$ of the state, any relevant inputs, and time. For fluid dynamic problems, $\boldsymbol{x}$ can be the vector of conserved variables (density, specific momentum, specific energy,

etc.) throughout the spatial discretization; $\boldsymbol{R}$ can be the balance of advective and diffusive fluxes of the state throughout the discretized space; and $\boldsymbol{\mu}$ can be the boundary conditions, initial conditions, geometric constraints, etc. of the problem being reduced.

Combining the temporal and spatial terms into a single term leaves a single, temporal residual,

$$\bar{\boldsymbol{R}}(\boldsymbol{x}, \boldsymbol{\mu}, t) = \boldsymbol{0}, \tag{2.2}$$

which can be discretized and minimized in time. Generally, the use of the word "*residual*" can be applied to different terms in the above definitions in different contexts. Thus, for the sake of clarity, in this thesis "*residual*" always refers to the dynamical function $\boldsymbol{R}$ in the generic model from Equation (2.1) whereas "*temporal residual*" is used for the balance of temporal and spatial dynamics, which is $\bar{\boldsymbol{R}}$ in Equation (2.2).

The usefulness of this generic dynamical model is that all of the model reduction derivations can be performed without formally defining the physics. Thus, once these ROMs are defined in the proceeding sections, their application can be performed to many sets of problems.

2.2 Projection-based ROMs

The process of spatially and temporally discretizing the generic model creates a system of equations that can be solved for the state and outputs. Practically, the use of computers makes these discretizations tractable for scientific and engineering analysis. Often the sizes of the spatial and temporal discretizations have an effect of the dynamics in $\boldsymbol{R}$. An example from CFD is the effect of grid size on the presence and strength of turbulence. Any turbulent phenomena occurring at scales smaller than what can be resolved in the grid effectively disappear. Finer discretizations of the problem usually result in greater resolution in the physics being modeled; however, the immediate issue with the finer resolution is the increase in degrees of freedom in the system. The increase in cost for is typically super-linear for most problems, as the cost of inversion of the dynamical system increases super-linearly.

A question relevant to model reduction is: are there certain degrees of freedom that can be neglected while maintaining a high-fidelity solution? The answer for many practical engineering problems is typically yes and serves as the justification of projection-based ROMs. There are many configurations of the state $\boldsymbol{x}$ that are not possible. For example, most physical models do not allow for negative density or energy. Additionally, the variation of the solution in certain domains may be too small to have a substantial effect on

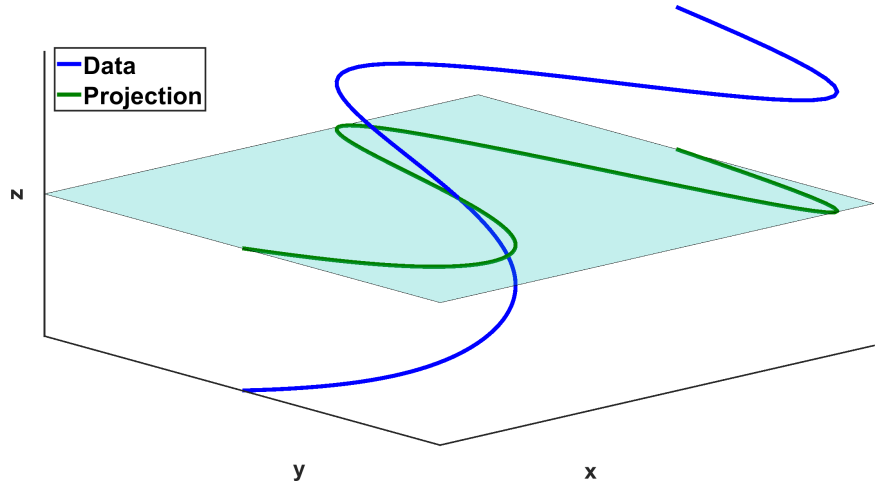


Figure 2.1: Example of projecting data from 3 dimensions into 2 dimensions.

the overall solution. Therefore, we can seek the solution of $\boldsymbol{x}$ in only a subspace of the full-order discretization. This is formally defined with the low-rank approximation,

$$\boldsymbol{x} \approx \boldsymbol{V}\hat{\boldsymbol{x}}, \tag{2.3}$$

where $\boldsymbol{V} \in \mathbb{R}^{n_{\boldsymbol{x}} \times n_{\boldsymbol{V}}}$ is a set of basis vectors (also called the state basis vectors), $\hat{\boldsymbol{x}} \in \mathbb{R}^{n_{\boldsymbol{V}}}$ is a set of weights on those basis vectors (also called the reduced state), and the rank of the state basis is substantially smaller than the original full degrees of freedom $n_{\boldsymbol{V}} \ll n_{\boldsymbol{x}}$. Essentially, the solutions of the model are now sought in a projection of the FOM space, and if the projecting basis spans the FOM solution space well, the approximation will be accurate. Figure 2.1 demonstrates this concept visually with the projection of 3D data onto a 2D plane.

Making this substitution into the general model results in a system with significantly fewer degrees of freedom but still utilizing the fundamental physics-based equations,

$$\boldsymbol{M}\boldsymbol{V}\frac{d\hat{\boldsymbol{x}}}{dt} + \boldsymbol{R}(\boldsymbol{V}\hat{\boldsymbol{x}}(t), \boldsymbol{\mu}(t), t) = \boldsymbol{0}, \tag{2.4}$$

$$\boldsymbol{y}(t) = J(\boldsymbol{V}\hat{\boldsymbol{x}}(t), \boldsymbol{\mu}(t), t). \tag{2.5}$$

The problem is now overdetermined, as we have $n_{\boldsymbol{x}}$ equations for $n_{\boldsymbol{V}}$ unknowns. Consider the temporal residual with the state approximation defined by

$$\bar{\boldsymbol{R}}(\boldsymbol{V}\hat{\boldsymbol{x}}(t), \boldsymbol{\mu}(t), t) \in \mathbb{R}^{n_{\boldsymbol{x}}}. \tag{2.6}$$

Since this temporal residual is driven to zero, the cost of the time-marching is effectively

unchanged since the actual problem itself is still of FOM problem size. Reduction of the problem size can be performed via projection of the entire system with another set of basis vectors ($\boldsymbol{W} \in \mathbb{R}^{n_{\boldsymbol{x}} \times n_{\boldsymbol{V}}}$), resulting in

$$\begin{aligned} \boldsymbol{W}^T \left[\boldsymbol{M}\boldsymbol{V}\frac{d\hat{\boldsymbol{x}}}{dt} + \boldsymbol{R}(\boldsymbol{V}\hat{\boldsymbol{x}}(t), \boldsymbol{\mu}(t), t) \right] &= \boldsymbol{W}^T \boldsymbol{0}, \\ \hat{\boldsymbol{M}}\frac{d\hat{\boldsymbol{x}}}{dt} + \boldsymbol{W}^T \boldsymbol{R}(\boldsymbol{V}\hat{\boldsymbol{x}}(t), \boldsymbol{\mu}(t), t) &= \boldsymbol{0}. \end{aligned} \tag{2.7}$$

where $\hat{\boldsymbol{M}} = \boldsymbol{W}^T \boldsymbol{M}\boldsymbol{V}$ is the reduced mass matrix.

The approach of seeking the solution in the span of a basis while restricting the system to another basis is analogous to the Galerkin approach to PDEs where a solution to a continuous problem is restricted to the span of a set of linear basis vectors ("trial space") and the system itself is subjected to constraint via projection with a basis ("test space"). When the system is projected with the same basis used for representing the state (*i.e.*, $\boldsymbol{V} = \boldsymbol{W}$) it is referred to as the Bubnov-Galerkin (or simply Galerkin) formulation. When these two bases are not equal (*i.e.*, $\boldsymbol{W} \neq \boldsymbol{V}$) one has the Petrov-Galerkin formulation. The same nomenclature is adopted when referring to the formulation of a projection-based ROM, and this distinction will be important in Chapter 4, where the formulation of a novel Petrov-Galerkin ROM is presented. However, for this thesis, unless otherwise stated, the Galerkin formulation is used.

In summary, the formulation of a projection-based ROM requires a reduction of degrees of freedom via the use of the low-rank approximation in Equation (2.3) and the projection of the system itself in Equation (2.7). Earlier, a justification for the use of a low-rank approximation of the state was that given a state basis that spans the solution space fairly well, the approximation will be accurate; however, the actual construction of the state basis is left to the discussion in the next section, which details the proper orthogonal decomposition method.

2.3 Proper Orthogonal Decomposition

The following section derives the discrete proper orthogonal decomposition (POD) ROM. The derivation here follows the idea of the POD basis constructed to maximize the *information* it carries as presented in [86]. A projection-based ROM was constructed via substitution of the state with a low-rank approximation and projection of the entire system with an orthogonal basis in the preceding section. Although these bases can consist of any set of orthogonal vectors (*e.g.*, Fourier modes, Lagrange polynomials, etc.), the immediate benefit of using a tailored basis is the reduction of the size of the state projection and the

minimization of irrelevant information that needs to be dissipated in an untailored basis. The proper orthogonal decomposition aims to construct a basis that maximizes the amount of relevant information that the POD basis carries in the solution space of interest,

$$\mathcal{X} \subset \mathbb{R}^{n_{\boldsymbol{x}}}. \tag{2.8}$$

The information contained in a projecting basis of a solution space refers to the amount that basis can span that solution space. The larger amount of the solution space that can be represented by a basis, the more information the basis carries. This can be quantified for any instance of the state solution $\boldsymbol{x} \in \mathcal{X}_i$ and some normal basis vector $\boldsymbol{V}_i$ with the square magnitude of their inner product $\langle \cdot, \cdot \rangle$,

$$\mathcal{I}_i = \left| \langle \boldsymbol{V}, \boldsymbol{x}_i \rangle \right|^2. \tag{2.9}$$

POD attempts to maximize the amount of information carried by the projecting basis across all possible $\boldsymbol{x} \in \mathcal{X}$. Fully solving for this basis is intractable and self-defeating as all possible solutions of $\boldsymbol{x}$ is needed. However, this basis approximated efficiently with the method of snapshots first introduced by Sirovich [16]. First, an ensemble of snapshots from the full-order system is collected, defined by

$$\mathcal{S} = \begin{bmatrix} \boldsymbol{s}_0 & \boldsymbol{s}_1 & \dots & \boldsymbol{s}_{k-1} \end{bmatrix}, \tag{2.10}$$

where $\boldsymbol{s}_i \in \mathcal{X}$ for $i \in [0, k-1]$). Then the POD basis is constructed to maximize the information carried across the snapshot ensemble – *i.e.*,

$$\boldsymbol{V} = \arg\max_{\tilde{\boldsymbol{V}}} \sum_{i=0}^{k-1} \left| \langle \tilde{\boldsymbol{V}}, \boldsymbol{s}_i \rangle \right|^2. \tag{2.11}$$

The solution to this maximization problem is the set of left singular vectors of the snapshot set $\mathcal{S}$, according to

$$\mathcal{S} = \boldsymbol{V} \boldsymbol{\Sigma} \boldsymbol{Z}^T, \tag{2.12}$$

where $\boldsymbol{\Sigma} = \text{diag}(\sigma_i)$ is the diagonal matrix containing the singular values, σ_i, which are all real, non-negative and listed in decreasing order. The columns of the matrix $\boldsymbol{Z}$ are the right singular vectors.

Although for completeness the entire set of left singular vectors are needed to maximize the total information in $\mathcal{S}$ carried by $\boldsymbol{V}$, truncation of the resulting basis can be done for

various reasons. Singular vectors with sufficiently small singular values carry very small amounts of information from $\mathcal{S}$; additionally, these basis vectors are more prone to larger amounts of inauspicious numerical error, which affects convergence, accuracy, and stability. A criterion for the truncation index (r) is the partial sum of singular value energy being above some tolerance (*e.g.*, 99.9%), defined by

$$\frac{\sum_0^r \sigma_i}{\sum_0^{k-1} \sigma_i} \geq \text{TOL}_\sigma. \tag{2.13}$$

However, as discussed in Chapter 3, the energy criterion is not universally sufficient for generating an accurate POD-ROM. Small-scale behavior, while indeed small, can be very influential on outputs of interest for a problem. This is true for many physics-based models. Consider for example the production of forces on an aerodynamic surface. These forces are of primary interest to engineers and are dictated significantly by a thin *boundary layer* of flow just outside the wetted-surface of a structure. Thus, even if the solution is well resolved in the majority of the fluid-domain, inaccuracy in the boundary layer results in poor load predictions. Thus for the sake of accurate output prediction, a large effort in the aerospace industry is placed on producing accurate small-scale behavior [87, 66]. In subsequent sections, the exploration of selecting basis vectors based on targeted output error estimates is presented.

For many discretizations and problem settings, a singular value decomposition of the snapshot matrix suffices for computing the state basis vector set. However, for finite-element discretizations, such as the discontinuous Galerkin (DG) method used throughout this thesis, the discrete singular value decomposition will depend on the finite-element basis functions used in the discretization. To eliminate this dependence, a continuous spatial inner product is used, and the singular vectors and values are computed from an eigendecomposition of the corresponding normal matrix. This normal matrix takes the form

$$\boldsymbol{K}_{\text{norm}} = \mathcal{S}^T \boldsymbol{M} \mathcal{S}, \tag{2.14}$$

where $\boldsymbol{M}$ is the mass matrix. For finite element models, the state is approximated by the linear combination of basis functions, according to

$$\boldsymbol{x}_\Omega(\vec{r}) \approx \sum_j^{n_p} \boldsymbol{X}_\Omega \Phi_j(\vec{r}), \tag{2.15}$$

where $\boldsymbol{x}_\Omega$ is the state in a domain Ω, $\vec{r}$ are physical coordinates inside the domain, and

$\tilde{\boldsymbol{x}} \in \mathbb{R}^{n_p}$ are coefficients for the n_p basis functions $\boldsymbol{\Phi}$. The mass matrix then is

$$\boldsymbol{M}_{i,j} = \int_{\Omega} \Phi_i^T \Phi_j d\Omega. \tag{2.16}$$

Rowely *et al.* [17], Barone *et al.* [53], and Kalashnikova and Barone [52] have demonstrated that utilizing the mass matrix normalized inner product can improve stability of compressible Navier-Stokes systems. The eigenvectors of this system represent the singular vectors of the snapshot set, and the corresponding eigenvalues are the squares of the corresponding singular values ($\sigma_i = \sqrt{\lambda_i(\boldsymbol{K}_{\text{norm}})}$).

Generally, the POD-ROM models are constructed centered about some reference state $\boldsymbol{x}_r$. This is similar to Reynolds averaging of the Navier-Stokes equations, where the state is broken into static and fluctuating components and is defined by

$$\boldsymbol{x} = \boldsymbol{x}_r + \boldsymbol{V}_f \hat{\boldsymbol{x}}_f, \tag{2.17}$$

$$[\mathcal{S} - \boldsymbol{x}_r] = \boldsymbol{V}_f \boldsymbol{\Sigma}_f \boldsymbol{Z}_f^T. \tag{2.18}$$

For POD-ROMs, the ensemble average of the snapshot set is typically used to define the static component,

$$\boldsymbol{x}_r = \frac{1}{k} \sum_{i=0}^{k-1} \boldsymbol{s}_i. \tag{2.19}$$

In this thesis, both the average state and the steady-state solution are used as the reference state. For simplicity, in the remainder of the thesis, we will drop the f subscript for the fluctuating portion of the POD-ROM. The resulting centered POD-ROM has the form

$$\hat{\boldsymbol{M}} \frac{d\hat{\boldsymbol{x}}}{dt} + \boldsymbol{W}^T \boldsymbol{R}(\boldsymbol{x}_r + \boldsymbol{V}\hat{\boldsymbol{x}}(t), \boldsymbol{\mu}(t), t) = \boldsymbol{0}. \tag{2.20}$$

There are many reasons to use the mean as the reference state to center the POD basis. Jarvis demonstrated that the first eigenvector of uncentered snapshots equals the normalization of the mean [88]. Given that the mean should be time/parameter invariant, inclusion of the mean in the POD basis can create perturbations in the invariant component and introduce significant error [89]. In addition, centered ROMs for problems with consistent boundary conditions between snapshots inherently satisfy static boundary conditions, since they are contained in the unchanging reference state [36]. This is typically not the case for parameterized ROMs, as the variation in parameters is often implemented through boundary conditions. For multiparameter problems, a single, *global* ROM that is constructed

from all of the snapshots; however, for a solution space that is highly sensitive to the parameter space, global ROMs may be difficult to construct due to the need of a large number of snapshots and inaccurate due to the inclusion of irrelevant dynamics in the state basis [90, 91].

An alternative approach is to construct localized ROMs for partitions of the parameter space. ROMs are then interpolated for configurations that are not already modeled locally. Interpolation between basis vectors does not maintain orthogonality. Instead interpolation between bases can be performed. A method by Amsallem and Farhat [92, 93], uses tangent spaces of the Grassman manifold containing each of the bases. The Grassman manifold is the space of subspaces of size r (the rank of the bases) of the space of size N.

Let $\boldsymbol{V}_k$ be the set of local ROM bases where $k \in [1, K]$ corresponds to a sampled parameter set $\vec{p}_k$ for each local ROM. The interpolation is performed about a reference basis, which for simplicity is $\boldsymbol{V}_1$. The location ($\boldsymbol{T}_k$) of all other local bases on the interpolation space is found with a logarithmic mapping with respect to the reference basis, defined by

$$\boldsymbol{U}_k \boldsymbol{\Sigma}_k \boldsymbol{Z}_k^T = (\boldsymbol{I} - \boldsymbol{V}_1 \boldsymbol{V}_1^T) \boldsymbol{V}_k (\boldsymbol{V}_1^T \boldsymbol{V}_k)^{-1} \longrightarrow \text{(thin SVD)}, \tag{2.21}$$

$$\boldsymbol{T}_k = \boldsymbol{U}_k \arctan(\boldsymbol{\Sigma}_k) \boldsymbol{Z}_k^T. \tag{2.22}$$

For a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ with rank r, the thin SVD refers to a singular value decomposition, according to,

$$\boldsymbol{A} = \boldsymbol{U}_{\boldsymbol{A}} \boldsymbol{\Sigma}_{\boldsymbol{A}} \boldsymbol{V}_{\boldsymbol{A}}^T, \tag{2.23}$$

$$\boldsymbol{U} = \begin{bmatrix} \boldsymbol{u}_1 & \dots & \boldsymbol{u}_r \end{bmatrix} \in \mathbb{R}^{m \times r}, \tag{2.24}$$

$$\boldsymbol{V} = \begin{bmatrix} \boldsymbol{v}_1 & \dots & \boldsymbol{v}_r \end{bmatrix} \in \mathbb{R}^{n \times r}, \tag{2.25}$$

and $\boldsymbol{\Sigma}_{\boldsymbol{A}}$ is an $r \times r$ diagonal matrix of the singular values.

For a set of parameters $\vec{p}_q$, the mapping onto this space can be interpolated with

$$\boldsymbol{T}(\vec{p}_q) = \boldsymbol{\Sigma}_{k=2}^{K} L(\vec{p}_k) \boldsymbol{T}_k, \tag{2.26}$$

where $L(\vec{p})$ is an interpolation weight dependent on the problem parameters. The mapping from the interpolation space back to the space where the bases originated from is performed with

$$\boldsymbol{T}(\vec{p}_q) = \boldsymbol{U}_q \boldsymbol{\Sigma}_q \boldsymbol{Z}_q^T \longrightarrow \text{(thin SVD)}, \tag{2.27}$$

$$\boldsymbol{V}_q = \boldsymbol{V}_1 \boldsymbol{Z}_q \cos \boldsymbol{\Sigma}_q + \boldsymbol{U}_q \sin \boldsymbol{\Sigma}_q. \tag{2.28}$$

The basis $\boldsymbol{V}_q$ becomes the basis for the POD-ROM and is used online for prediction.

Returning to Equation (2.20), note that if the function $\boldsymbol{R}$ is linear with respect to $\boldsymbol{x}$ and $\boldsymbol{\mu}$, the complexity of the POD-ROM model can be effectively reduced, resulting in

$$\begin{aligned}\hat{\boldsymbol{M}}\frac{d\hat{\boldsymbol{x}}}{dt} + \boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V}\hat{\boldsymbol{x}} + \boldsymbol{W}^T\boldsymbol{C}\Delta\boldsymbol{\mu} &= \boldsymbol{0},\\ \hat{\boldsymbol{M}}\frac{d\hat{\boldsymbol{x}}}{dt} + \hat{\boldsymbol{A}}\hat{\boldsymbol{x}} + \hat{\boldsymbol{C}}\Delta\boldsymbol{\mu} &= \boldsymbol{0},\end{aligned} \tag{2.29}$$

where $\boldsymbol{A}$ is the linearization of $\boldsymbol{R}$ about the reference state, $\boldsymbol{C}$ is the linearization of $\boldsymbol{R}$ about some reference input, $\Delta\boldsymbol{\mu}$ is the change in the input, and $\hat{\boldsymbol{A}}$ and $\hat{\boldsymbol{C}}$ are the reduced forms of the linearizations.

However, most problems of interest in the aerospace community are nonlinear. Due to this nonlinearity, the sizes of the residual and its Jacobian often used in updating the state to solve for $\hat{\boldsymbol{x}}$ are still computed at full-order complexity (*i.e.*, the input to $\boldsymbol{R}$ is still the same full-order size). The cost of computing these values often dominates the cost of CFD simulations, and thus the POD-ROM model, although reducing the number of degrees of freedom, has a small effect on the actual computational complexity of nonlinear problems [90]. This motivated the development of hyperreduced ROMs, which aim to reduce the cost of the residual and Jacobian calculations. The following section covers these techniques with a particular focus on the discrete empirical interpolation method.

2.4 Hyper-reduction

Hypper-reduction refers to procedures that reduce the computational complexity of the system itself, in addition to any reduction to the degrees of freedom through projection. These include continuous and discrete interpolation methods (EIM/DEIM) [24, 25], the missing point approximation [26, 27], the best points interpolation method [28], and the Gauss-Newton with approximate tensors method [29]. The focus of this thesis is the discrete empirical interpolation method, first introduced by Chaturantabut and Sorensen, which is the discrete formulation the empirical interpolation method and uses techniques found in sparse-sensing theory [25, 94]. Suppose that there is a limitation to the number of discrete locations where the residual can be observed, and that those indices are contained in a sampling matrix $\boldsymbol{P} = \begin{bmatrix}\boldsymbol{e}_0 & \boldsymbol{e}_1 & \dots & \boldsymbol{e}_i & \dots & \boldsymbol{e}_{n_{\boldsymbol{P}}-1}\end{bmatrix}$, where $\boldsymbol{e}_i$ is an elementary vector with a one at the i^{th} observed residual and $n_{\boldsymbol{P}}$ is the total number of observed discrete residuals. Storing the elementary vectors in this manner allows the product of the sampling matrix

and the residual to contain only the residual values at those locations, *i.e.*,

$$\boldsymbol{R}_{\boldsymbol{P}} = \boldsymbol{P}^T \boldsymbol{R} = \begin{bmatrix} \boldsymbol{R}_{\boldsymbol{P}_0} \\ \vdots \\ \boldsymbol{R}_{\boldsymbol{P}_{n_{\boldsymbol{P}}-1}} \end{bmatrix}. \tag{2.30}$$

Also suppose that there is a linear basis ($\boldsymbol{U} \in \mathbb{R}^{n_{\boldsymbol{x}} \times n_{\boldsymbol{U}}}$) that can be used as a low-rank approximation of the residual, weighted by $\boldsymbol{c} \in \mathbb{R}^{n_{\boldsymbol{U}} \times 1}$, given by

$$\boldsymbol{R} \approx \boldsymbol{U}\boldsymbol{c}. \tag{2.31}$$

Applying the sampling matrix to both sides yields

$$\boldsymbol{P}^T \boldsymbol{R} \approx \boldsymbol{P}^T \boldsymbol{U}\boldsymbol{c} \tag{2.32}$$

If $n_{\boldsymbol{P}} \geq n_{\boldsymbol{U}}$, then there is a solution for $\boldsymbol{c}$ such that the approximation is correct at the sampling points, according to

$$\begin{aligned} \boldsymbol{P}^T \boldsymbol{R} &= \boldsymbol{P}^T \boldsymbol{U}\boldsymbol{c}, \\ \boldsymbol{c} &= \left[\boldsymbol{P}^T \boldsymbol{U}\right]^{\dagger} \boldsymbol{P}^T \boldsymbol{R}, \end{aligned} \tag{2.33}$$

where $\dagger$ indicates the inverse for $n_{\boldsymbol{P}} = n_{\boldsymbol{\mu}}$ and the left pseudo-inverse for $n_{\boldsymbol{P}} > n_{\boldsymbol{\mu}}$. The left pseudo-inverse is member of a class of generalized matrix inverses [95] and is defined for a real matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ with $m \geq n$ by

$$\boldsymbol{A}^{\dagger} = \left[\boldsymbol{A}^T \boldsymbol{A}\right]^{-1} \boldsymbol{A}^T, \tag{2.34}$$

$$\boldsymbol{A}^{\dagger} \boldsymbol{A} = \mathbb{I}. \tag{2.35}$$

Multiplying both sides by $\boldsymbol{U}$ and substituting the low-rank approximation of Equation (2.31) yields the DEIM approximation for the residual, *i.e.*,

$$\boldsymbol{R}_{\text{DEIM}} = \boldsymbol{U} \left[\boldsymbol{P}^T \boldsymbol{U}\right]^{\dagger} \boldsymbol{P}^T \boldsymbol{R}, \tag{2.36}$$

$$\boldsymbol{R}_{\text{DEIM}} \approx \boldsymbol{R}. \tag{2.37}$$

Although in sparse sensor theory, any set of basis functions can serve as the residual basis, the vectors in $\boldsymbol{U}$ are typically derived via the method of snapshots, similarly to how the state basis vectors are derived. The benefit of using tailored basis vectors is that they usually allow for fewer overall interpolation points in $\boldsymbol{P}$ [94]. Additionally, the residual

snapshots can be obtain in tandem with the state snapshots $\boldsymbol{S}$.

Combining the DEIM approximation from Equation (2.37) with the POD-ROM from Equation (2.20), yields the DEIM-ROM model, defined with

$$\hat{\boldsymbol{M}}\frac{d\hat{\boldsymbol{x}}}{dt} + \boldsymbol{W}^T\boldsymbol{R}_{\text{DEIM}}(\boldsymbol{x}_r + \boldsymbol{V}\hat{\boldsymbol{x}}(t), \boldsymbol{\mu}(t), t) = \boldsymbol{0}. \tag{2.38}$$

The selection of sampling indices is important to keeping the reduced system accurate and tractable. Generally, using more sampling indices results in better residual approximation but also at a higher cost. Thus, it is important to choose sampling points that best approximate the residual. A greedy algorithm was originally used for selecting the DEIM-ROM residual sampling indices where the sampling indices were chosen based on the error of the sampling of the residual basis [25]:

Algorithm 1 Greedy Selection of Sampling Indices

Find sample index: $p_0 \leftarrow \arg\max |\boldsymbol{U}_0|$
Update $\boldsymbol{P}$: $\boldsymbol{P} \leftarrow \boldsymbol{P} \cup e_{\boldsymbol{P}_0}$
Update $\boldsymbol{U}$: $\boldsymbol{U} \leftarrow \boldsymbol{U} \cup \boldsymbol{U}_0$
for $i = 1 \ldots n_{\boldsymbol{\mu}} - 1$ **do**
  $\boldsymbol{c} \leftarrow \arg\min_{\tilde{\boldsymbol{c}}} \boldsymbol{P}^T\boldsymbol{U}\tilde{\boldsymbol{c}} - \boldsymbol{P}^T\boldsymbol{U}_i$
  Compute Error: $\boldsymbol{r} = \boldsymbol{U}_i - \boldsymbol{U}\boldsymbol{c}$
  Find sample index: $p_i \leftarrow \arg\max |\boldsymbol{r}|$
  Update $\boldsymbol{P}$: $\boldsymbol{P} \leftarrow \boldsymbol{P} \cup e_{\boldsymbol{P}_i}$
  Update $\boldsymbol{U}$: $\boldsymbol{U} \leftarrow \boldsymbol{U} \cup \boldsymbol{U}_i$
end for

This process can be carried out multiple times in order to create an oversampling of the residual.

Although this process generates a sampling matrix that performs better than just randomized sampling points [96], Drmac and Gugercin provide a more optimal method for generating sampling indices by using a pivoted-QR decomposition (QDEIM) [97]. First, the sampling indices of the greedy algorithm depends on the order of the residual basis. Reordering the vectors in $\boldsymbol{U}$ may result in different sampling indices being chosen. Secondly, Drmac and Gugercin demonstrate that the sampling error is minimized by their pivoted-QR based method.

Column pivoting QR decomposition is a variant of the QR decomposition which decomposes a matrix $\boldsymbol{B}$ into the product of an orthogonal matrix ($\boldsymbol{Q}$) and an upper right

triangular matrix ($\boldsymbol{R}$), *i.e.*,

$$\boldsymbol{B} = \boldsymbol{Q}\boldsymbol{R}. \tag{2.39}$$

Column pivoting QR performs this procedure on the matrix $\boldsymbol{B}$, whose columns are permuted with the column permutation $\boldsymbol{C}_{\text{QR}}$, such that the diagonal components of $\boldsymbol{R}$ are non-increasing, according to

$$\boldsymbol{B}\boldsymbol{C}_{\text{QR}} = \boldsymbol{Q}\boldsymbol{R}. \tag{2.40}$$

This procedure is rank-revealing and numerically more stable than the standard QR decomposition [98].

A product of the sampling error and the residual basis error is an upper bound of the L2 error of the DEIM residual, defined by

$$||\boldsymbol{R} - \boldsymbol{R}_{\text{DEIM}}||_2 \leq \underbrace{||(\boldsymbol{P}^T\boldsymbol{U})^\dagger||_2}_{\text{Sampling Error}} \underbrace{||(\boldsymbol{I} - \boldsymbol{U}\boldsymbol{U}^T)\boldsymbol{R}||_2}_{\text{Projection Error}}. \tag{2.41}$$

It is clear that the sampling indices only affect the first component of this upper bound. For the case where $n_{\boldsymbol{P}} = n_{\boldsymbol{\mu}}$ and $\dagger$ is the inverse operator, the sampling error is the inverse of the spectral radius of $\left(\boldsymbol{P}^T\boldsymbol{U}\right)$. Thus choosing $\boldsymbol{P}$ to maximize the first eigenvalue of $\left(\boldsymbol{P}^T\boldsymbol{U}\right)$ minimizes the error in the DEIM residual. The ingenuity of the QDEIM method is the recognition that this condition is met if $\boldsymbol{P}$ are the column pivots arising from the pivoted-QR decomposition of $\boldsymbol{U}^T$, according to

$$\boldsymbol{U}^T\boldsymbol{P} = \boldsymbol{Q}\boldsymbol{R}. \tag{2.42}$$

An additional benefit to using pivoted-QR for obtaining the sampling indices is that highly efficient and parallel pivoted-QR algorithms are standard in most scientific libraries (SCLAPACK, MLK, etc.) [94, 97]. In most applications of DEIM, many more sampling indices are needed for an accurate residual prediction than the rank of the residual basis, and *oversampling* refers to the use of a DEIM-ROM where $n_{\boldsymbol{P}} > n_{\boldsymbol{U}}$. For oversampling, the pivoted-QR decomposition of $\boldsymbol{U}\boldsymbol{U}^T$ yields the optimal sampling indices; however, it should be noted that computational and memory costs of performing the pivoted-QR decomposition of $\boldsymbol{U}\boldsymbol{U}^T$ is immense. The computational complexity of performing a column pivoted QR decomposition of a matrix $\boldsymbol{B}$ is roughly $O(\text{rank}(\boldsymbol{B})^3)$. For $\boldsymbol{U}^T$, this cost is small as long as the number of basis vectors in $\boldsymbol{U}$ is kept small. However, the rank of $\boldsymbol{U}\boldsymbol{U}^T$ is equal to the high-fidelity degrees of freedom. Additionally $\boldsymbol{U}\boldsymbol{U}^T$ is a dense ma-

trix, making its storage infeasible for most problems of interest. Thus for this thesis, two separate methods are used for obtaining the sampling indices. Both begin with a pivoted-QR of $\boldsymbol{U}^T$ to obtain $n_{\boldsymbol{\mu}}$ sampling indices. The remaining indices are obtained with either randomized sampling or with iteratively performing the pivoted-QR of $\boldsymbol{U}^T$ with zeros in the columns corresponding to the already chosen sampling.

CHAPTER 3

Adjoint-weighted Residual Error Estimation for POD and DEIM-ROMs

Reduced-order models inherently suffer losses in accuracy due to the low-rank approximations made for the state and projection of the system. In addition, while *a priori* snapshots are used for designing ROMs, *a priori* information may not be suitable for assessing the quality of ROM solutions online and adaptation thereof. Thus online error quantification is an important feature for ROMs as it uses *a posteriori* information to assess the quality of ROM solutions and drive any subsequent adaptation. This chapter introduces output-based error estimates for DEIM-ROMs that are based on the adjoint-weighted residual error estimation method used for CFD analysis. In addition, fine-grain error analysis allows for these error estimates to be used as an adaptation mechanism that prioritizes improving the output prediction of a completed online solution efficiently. Several example applications are demonstrated including unsteady and 3D simulations.

3.1 Current Approaches for ROM Error Quantification

Error quantification is an important factor in determining the usefulness of any ROM. For projection-based ROMs, substantial effort has been placed in developing *a posterori* error bounds [30, 32, 31, 33, 20, 43, 34]. These error bounds are constructed with many different approaches: utilizing the error in the snapshot projection, the evolution of the truncated portion of the basis in the FOM system, and/or the error of the output of the solution on the original snapshot set. With a first-order Taylor expansion of the residual, LeGresley and Alonso [35] estimated the error between a POD solution and a solution to the same problem solved with the union of the original POD basis and addition basis vectors. Applied to a transonic flow problem, this technique was used to determine spatial regions for domain decomposition. With a focus on control problems, Homescu, Petzold, and Serban [99] developed an error estimate based on a combination of adjoint solutions and the small sample statistical condition estimation method. Using this estimate, they were able to determine

regions of validity for perturbations about initial conditions where the use of POD ROMs would be appropriate.

Error quantification for DEIM-ROMs has been restricted to the formulation of error bounds. Chaturantabut and Sorensen first provided an *a priori* upper bound for the state error in terms of the neglected singular values of the SVD decomposition of the state and nonlinear function snapshots [41]. While their approach is useful for developing error bounds for other sparsely interpolated methods and is theoretically insightful, it lacks application for evaluating ROM solutions, as it is not able to distinguish between different online solutions. The need for *a posteriori* error estimates for DEIM models has produced a few approaches. Wirtz, Sorensen, and Haasdonk developed rigorous *a posteriroi* state and output error bounds for a general, unsteady, parameterized nonlinear system [42]. Their method relies on integrating the upper bound of the gradient of the error with the ROM error approximated with the truncated components of the ROM. This creates sharp error bounds for DEIM ROMs of general nonlinear PDEs; however, it lacks a mechanism for adaptation. Feng, Mangold, and Benner applied output-based error bounds to adapting the POD-DEIM model [44]. Based on an approach by Zhang *et al.* [43], the output error bound is constructed from an inequality involving an adjoint-weighted residual, where projections of the snapshots are used to approximate a scaling of the error bound. The use of the inequality is to avoid producing time-coupled adjoint solutions, and the need of the projected snapshots arises from lacking access to the FOM solution for the scaling parameter for the inequality. The adaptation distinguishes error estimate contributions for the POD and DEIM components of the model in order to adapt the model with additional basis vectors. The methods developed in this thesis improve upon these ideas in several ways. Resolving the time-coupled adjoint is important for accurately propagating error through time. Instead of utilizing an approximation, This chapter presents the full adjoint equation for the DEIM-ROM model and solves it to produced time-coupled adjoints and output error estimates. Error bounds indicate the need for adaptation, but the manner of adaptation is undetermined. The adaptation approach taken by Feng, Mangold, and Benner, as well as many other approaches using error bounds, update the ROM system with neglected singular vectors in the order of their singular values. This can lead to inefficient adaptation. Although the ordering of the singular values represents the added accuracy of each basis vector for projecting the state snapshots, the ordering does not necessarily represent the added accuracy for output prediction. By fully solving the adjoint equations, the error provided by each neglected basis vector can be localized, and the updating of the DEIM-ROM can be performed to prioritize improving output prediction. Additionally, error contributions for the nonlinear function sampling can be provided to allow for adaptation of the

sampling mesh.

Adjoint-weighted residual error estimation and adaptation methods quantify the error of an output between two different model resolutions without needing to resolve the model on the higher resolution, which can be too expensive to solve. Assuming that the model approaches the "true" solution with increasing resolution, the error estimate between a coarse and fine-resolution model is an approximation of the true error of the coarse model. Errors can be localized in the high-resolution space in order to identify locations where the resolution of the coarse-space model can be increased in order to efficiently adapt the model. Adjoint-weighted residuals have been applied to a variety of fluid dynamics problems [45, 46, 47, 100, 101, 102] and have found particular usefulness for adapting mesh resolution, approximation order, and the basis used in the finite element method. For projection-based models, the *resolution* of a model is dictated primarily by the rank of the state basis $\boldsymbol{V}$, residual basis $\boldsymbol{U}$, and the sampling matrix $\boldsymbol{P}$. Thus, the derivation of adjoint-weighted residual error estimation and adaptation is made with these spaces. The first application of adjoint-weighted residuals for POD models was performed by Meyers and Matthies but by solving the full-order sized adjoint problem and without adaptation [37]. Carlberg also apply adjoint-weighted error estimates for ROM adaptation through vector splitting but without hyper-reduction [38]. Yano utilized adjoint-weighted residual error estimates to formulate goal-oriented ROMs, composed of the reduced basis method and the empirical quadrature procedure [39, 40]. This thesis extends those ideas to the DEIM-ROM, formulates efficient error localization of the three DEIM-ROM metrics, and demonstrates error estimation and adaptation of POD and DEIM ROMs for several problems.

3.2 Adjoint-weighted Error Estimation for Finite Element Problems

Up to this point, a generic model defined with Equation (2.1) was used to derive the POD and DEIM-ROMs. This is useful as it allows for these methods to be applicable to any problems of the same structure but arising from different physics and discretizations. However, for this section, a basic finite element discretization will be defined in order to give context to the discussion of adjoint-based error estimation and the problem setting of the FOMs used in this thesis. Fluid dynamic problems are dictated by the conservation of mass,

momentum, and energy, which all take the same form of

$$\frac{\partial \boldsymbol{x}}{\partial t} + \sum_{d}^{D} \frac{\partial \boldsymbol{F}(\boldsymbol{x}, \nabla \boldsymbol{x})}{\partial r_d} = \mathbf{0}, \tag{3.1}$$

where $\boldsymbol{x}$ represents the conserved quantities (mass, momentum, energy, etc.), $\boldsymbol{F}$ is the flux of the conserved variable, D is the number of spatial dimensions, and r_d is the d^{th} spatial direction. Furthermore, the flux can be broken into convective and diffusive components, resulting in

$$\boldsymbol{F}(\boldsymbol{x}, \nabla \boldsymbol{x}) = \boldsymbol{F}^c(\boldsymbol{x}) - \boldsymbol{F}^d(\boldsymbol{x}, \nabla \boldsymbol{x}). \tag{3.2}$$

Qualitatively, these equations mean that the rate of change of a conserved quantity in a control volume is equal to the balance of the rate at which the conserved quantity enters and exits that space due to the bulk movement of the quantity and the diffusion of the quantity in the presence of a spatial gradient. Many CFD solvers utilize these equations to derive solutions for these quantities to fluid dynamic problems in some space of interest – *i.e.*, an area for 2D or a volume for 3D. The finite element method (FEM) breaks the space of interest into a composition of *finite elements* that are each modeled with Equation (3.1) along with equations of state and equations for non-conserved variables such as pressure and temperature for closure. The state $\boldsymbol{x}$ in some element Ω_i is approximated with the linear combination of basis functions $\boldsymbol{\Phi}(\vec{r})$ (*e.g.*, Legendre polynomials) spatially defined for the coordinates $\vec{r}$. A globally continuous solution can be obtained if the continuity of the observed quantities is enforced across neighboring elements with a globally continuous set of basis functions, which then yields the continuous Galerkin (CG) method; however, the continuous Galerkin method may not be the best choice for many problems in the field of aerospace. For problems with strongly advective physics, continuous Galkerin formulations may lose accuracy and stability without additional modifications.The discontinuous Galerkin method (DG) does not enforce continuity across neighboring elements; instead, "jumps" of the state across elements are permitted and dealt with special treatment of the flux terms. For the DG formulation, the basis functions are defined within an element, according to

$$\boldsymbol{x}_{\Omega_i}(\vec{r}) \approx \sum_{j}^{n_p} \boldsymbol{X}_{\Omega,j} \Phi_j(\vec{r}), \tag{3.3}$$

where n_p is the number of basis functions, $\boldsymbol{X}_{\Omega,j}$ are the coefficients on the j^{th} basis function of the domain Ω, and Φ_j is the j^{th} basis function. Equation (3.1) and Equation (3.2) are transformed into their weak formulation by first substituting the basis representation of the state, multiplying the system by test functions to obtain a square system which equal the trial basis for the Galerkin formulation, and then integrating by parts over each element. Since the state is discontinuous across element boundaries, the fluxes across element boundaries are modeled as Riemann problems across each interface of neighboring elements. For this thesis, the convective portion of the flux is modeled with the Roe flux formulation [103], and the diffusive portion of the flux is modeled with the second form of the Bassi-Rebay formulation (BR2) [104]. The resulting system is

$$\boldsymbol{M}\frac{d\boldsymbol{X}_\Omega}{dt} + \boldsymbol{R}(\boldsymbol{X}_\Omega) = \boldsymbol{0}, \tag{3.4}$$

where $\boldsymbol{M}_{j,k} = \int_\Omega \Phi_j^T \Phi_k d\Omega$ is the mass matrix and $\boldsymbol{R}$ is the spatial residual formed by the balancing of the convective and diffusive fluxes.

Equation (3.4) is in a semi-discrete form, where the spatial terms have been discretized (*i.e.*, with the DG state representation) and the temporal term remains continuous. There are various approaches to discretizing the time term; however, for this thesis, the temporal derivative is approximated with finite differencing and the solution is marched forward in time using the second-order backwards difference formulation (BDF2). BDF2 is part of a class of multistep time-marching methods. A mutlistep method applied to Equation (3.4) yields

$$\boldsymbol{M}\sum_{k=1-K}^{1}\left[\frac{\alpha^k \boldsymbol{X}_\Omega^{n+k}}{\Delta t}\right] + \sum_{k=1-K}^{1}\left[\beta^k \boldsymbol{R}(\boldsymbol{X}_\Omega^{n+k}, t^{n+k})\right] = \boldsymbol{0}, \tag{3.5}$$

where k is the time node index, α^k and β^k are the corresponding , K is the number of steps taken, $\boldsymbol{X}_\Omega^n$ is the latest state information, and the $\boldsymbol{X}_\Omega^{n+1}$ the next state to be solved. The choices of α^k and β^k determine the nature of the time-marching scheme, but for generality $\alpha^1 = 1$. For $\beta^1 = 0$, the residual of the $\boldsymbol{X}_\Omega^{n+1}$ state is not involved. These methods are known as an explicit scheme, as a solution for $\boldsymbol{X}_\Omega^{n+1}$ can be explicitly obtained. Although explicit scheme can produce solutions efficiently, the time step required for stability is constrained to spatial discretizations *i.e.*, a finer mesh size will require finer time steps. This is problematic for fluid dynamics problems as important fluid phenomena occur at very large and very small spatial and time scales. Thus, to accurately predict CFD solutions, a very fine mesh needs to be resolved at a very fine time step, which can be prohibitively expensive. Implicit time marching schemes resolve Equation (3.5) with involvement of the

residual at the next time step (*i.e.*, $\beta^n \neq 0$). This makes the solution $\boldsymbol{X}_\Omega^{n+1}$ arise from solving a nonlinear problem involving the spatial residual and makes a spatial coupling across elements. This can be defined as the spatial-temporal residual, defined by

$$\bar{\boldsymbol{R}}(\boldsymbol{X}_\Omega^n) = \boldsymbol{M}\frac{\alpha^n \boldsymbol{X}_\Omega^n}{\Delta t} + \boldsymbol{R}(\boldsymbol{X}_\Omega^n) + \boldsymbol{b}, \tag{3.6}$$

where $\boldsymbol{b}$ is a constant vector that is a function of the states at previous time nodes. This allows for larger time steps to be taken, even for finer meshes. The coefficients for the multistep scheme are determined by the finite differencing approximation used for the temporal derivatives in (3.5). Table 3.1 contains the coefficients for various multi-step schemes that are discuss in this thesis.

Method	Order	α^{-2}	α^{-1}	α^0	α^1	β^{-2}	β^{-1}	β^0	β^1
FE	1	0	0	-1	1	0	0	1	0
BDF1	1	0	0	-1	1	0	0	0	1
BDF2	2	0	$\frac{1}{2}$	-2	$\frac{3}{2}$	0	0	0	1

Table 3.1: Coefficients for multistep schemes discussed in this thesis.

For first-order schemes, an initial condition is given that defines the state at the start of the simulation. Because all of the required previous time-node information is available, these schemes are considered self-starting schemes. Higher-order schemes that use more than one previous time node state for constructing the temporal finite difference are not self-starting, but a self-starting scheme can be used to generate the requisite stencil needed by the higher-order scheme to begin. For this thesis, the backwards Euler method is used to generate the state stencil to start BDF2.

3.2.1 Discrete Adjoints for Steady Problems

The adjoint is the sensitivity of some scalar output y from Equation (2.1) to perturbations of the residual. This is defined mathematically as

$$\boldsymbol{\Psi} = \frac{\delta y}{\delta \bar{\boldsymbol{R}}}. \tag{3.7}$$

where each entry of $\boldsymbol{\Psi}$ represents the sensitivity of the output y to perturbation of the same entry in $\boldsymbol{R}$. For steady problems, only the sensitivity of the output with respect to the spatial residual is of concern, and the bar is dropped.

An equation for the adjoint can be derived by considering the cascading effect of small perturbations. A change in the input to the system, $\boldsymbol{\mu}$, creates a change in the residual,

which requires a change in the state to drive the residual to zero, and this change in the state creates a change in the output,

$$\boldsymbol{\mu}_0 = \boldsymbol{\mu}_0 + \delta\boldsymbol{\mu} \tag{3.8}$$

$$\downarrow$$

$$\delta\boldsymbol{R} = \boldsymbol{R}(\boldsymbol{x}_0, \boldsymbol{\mu}_0 + \delta\boldsymbol{\mu}) - \boldsymbol{R}(\boldsymbol{x}_0, \boldsymbol{\mu}_0) \tag{3.9}$$

$$\downarrow$$

$$\boldsymbol{x}_0 = \boldsymbol{x}_0 + \delta\boldsymbol{x} \tag{3.10}$$

$$\downarrow$$

$$\delta y = y(\boldsymbol{x}_0 + \delta\boldsymbol{x}, \boldsymbol{\mu}_0 + \delta\boldsymbol{\mu}) - y(\boldsymbol{x}_0, \boldsymbol{\mu}_0) \tag{3.11}$$

Beginning with a steady system, perturbations to the input of the problem will create a change of the residual away from zero. This perturbation of the residual can be approximated linearly with

$$\boldsymbol{R}(\boldsymbol{x}_0, \boldsymbol{\mu}_0) + \left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{\mu}}\right|_{\boldsymbol{x},\boldsymbol{\mu}} \delta\boldsymbol{\mu} = \delta\boldsymbol{R}, \tag{3.12}$$

where $\boldsymbol{x}_0$ is the solution of the model with the initial input $\boldsymbol{\mu}_0$ and $\delta\boldsymbol{\mu}$ is the perturbation of the input.

Driving the residual to zero yields the solution to the system at the perturb input,

$$\boldsymbol{R}(\boldsymbol{x}_0, \boldsymbol{\mu}_0) + \left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{\mu}}\right|_{\boldsymbol{x},\boldsymbol{\mu}} \delta\boldsymbol{\mu} + \left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x},\boldsymbol{\mu}} \delta\boldsymbol{x} = \boldsymbol{0}. \tag{3.13}$$

Subtracting Equation (3.13) from Equation (3.12) reveals that the change in the state caused by perturbations to the input can be approximated by the inverse of the Jacobian of the system, according to

$$-\left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x},\boldsymbol{\mu}} \delta\boldsymbol{x} = \delta\boldsymbol{R},$$

$$\delta\boldsymbol{x} = -\left[\left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}_h}\right|_{\boldsymbol{x},\boldsymbol{\mu}}\right]^{-1} \delta\boldsymbol{R}. \tag{3.14}$$

Combining this result with the linearization of the output yields the change in the output

that the residual perturbation causes,

$$y(\boldsymbol{x}_0 + \delta\boldsymbol{x}, \boldsymbol{\mu}_0 + \delta\boldsymbol{\mu}) = y(\boldsymbol{x}_0, \boldsymbol{\mu}_0) + \left.\frac{\partial y}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x},\boldsymbol{\mu}} \delta\boldsymbol{x}, \tag{3.15}$$

$$y(\boldsymbol{x}_0 + \delta\boldsymbol{x}, \boldsymbol{\mu}_0 + \delta\boldsymbol{\mu}) = y(\boldsymbol{x}_0, \boldsymbol{\mu}_0) - \left.\frac{\partial y}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x},\boldsymbol{\mu}} \left[\left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x},\boldsymbol{\mu}}\right]^{-1} \delta\boldsymbol{R}, \tag{3.16}$$

$$y(\boldsymbol{x}_0 + \delta\boldsymbol{x}, \boldsymbol{\mu}_0 + \delta\boldsymbol{\mu}) - y(\boldsymbol{x}_0, \boldsymbol{\mu}_0) = \underbrace{-\left.\frac{\partial y}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x},\boldsymbol{\mu}} \left[\left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x},\boldsymbol{\mu}}\right]^{-1}}_{\boldsymbol{\Psi}^T} \delta\boldsymbol{R}. \tag{3.17}$$

Recognizing that the change in residual is equal to only the residual of the original solution with the perturbed input, since the baseline residual is zero, we have

$$\delta\boldsymbol{R} = \boldsymbol{R}(\boldsymbol{x}_0, \boldsymbol{\mu}_0 + \delta\boldsymbol{\mu}) \cancel{- \boldsymbol{R}(\boldsymbol{x}_0, \boldsymbol{\mu}_0)}, \tag{3.18}$$

$$y(\boldsymbol{x}_0 + \delta\boldsymbol{x}, \boldsymbol{\mu}_0 + \delta\boldsymbol{\mu}) - y(\boldsymbol{x}_0, \boldsymbol{\mu}_0) = \boldsymbol{\Psi}^T \boldsymbol{R}(\boldsymbol{x}_0, \boldsymbol{\mu}_0 + \delta\boldsymbol{\mu}). \tag{3.19}$$

The adjoint equation for steady systems can be obtained from Equation (3.17),

$$\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}\right]^T \boldsymbol{\Psi} + \left[\frac{\partial y}{\partial \boldsymbol{x}}\right]^T = \boldsymbol{0}. \tag{3.20}$$

Up to this point, the perturbed inputs are not defined. Parameters that influence the resolution of the problem can take the place of these inputs and these derivations will still hold. Given the adjoint in the finer resolution space, an estimate of the change of the output caused by changing the problem resolution is given by an adjoint-weighted residual,

$$y_h(\boldsymbol{x}_h) - y_h(\boldsymbol{x}_h^H) = \boldsymbol{\Psi}_h^T \boldsymbol{R}_h(\boldsymbol{x}_h^H), \tag{3.21}$$

$$\left[\frac{\partial \boldsymbol{R}_h}{\partial \boldsymbol{x}_h}\right]^T \boldsymbol{\Psi}_h + \left[\frac{\partial y}{\partial \boldsymbol{x}_h}\right]^T = \boldsymbol{0}. \tag{3.22}$$

where h and H refer to the fine and coarse-spaces, $\boldsymbol{x}_h^H$ is the injection of the coarse-space solution $\boldsymbol{x}_H$ into the fine-space, and $\boldsymbol{x}_h$ is the fine-space solution. The key benefit of utilizing adjoint-weighted residual error estimates is that only the coarse-space solution and fine-space adjoint are required to obtain an estimate of the output difference between the fine and coarse-spaces. With the assumption that the "true" (infinite-dimensional) solution is approached with finer resolution, the error estimate between a coarse and fine-space is also an estimate of the error between the coarse-space solution and the "true" solution. The following section derives adjoint-weighted residual error estimates for unsteady systems.

3.2.2 Discrete Adjoints for Unsteady Problems

In a fully-discrete, multi-step time integration setting, Equation (3.1) becomes

$$\boldsymbol{M}\frac{\sum_{n=0}^{m} a^n \boldsymbol{x}^n}{\Delta t} + \boldsymbol{R}(\boldsymbol{x}^m, \boldsymbol{\mu}) = \boldsymbol{0}, \tag{3.23}$$

where Δt is the time-step size, the superscripts $n \geq m$ refer to time-nodes in the temporal discretiziation, and a^n refers to the weight used for the specific time-marching scheme used. For example, the backwards Euler method uses, $a^{m-1} = -1$, $a^m = 1$, and $a^n = 0$ for all $n < m - 1$. The second-order backwards Euler method uses $a^{m-2} = \frac{1}{2}$, $a^{m-1} = -2$, $a^m = \frac{3}{2}$, and $a^n = 0$ for all $n < m - 2$. These weights are derived from finite-difference approximations. For a problem being solved between times t_0 and t_f, the size of the time step is

$$\Delta t = \frac{t_f - t_0}{N_t - 1}, \tag{3.24}$$

where N_t is the number of time-nodes. Equation (3.23) can be written as a temporal residual with

$$\bar{\boldsymbol{R}}^m(\boldsymbol{x}^n, \boldsymbol{\mu}) = \boldsymbol{0}. \tag{3.25}$$

Additionally, for this thesis, unsteady adjoint-weighted residuals error estimates are made for time-integrated outputs. These can be defined by

$$\bar{y} = \int_{t=t_0}^{t_f} b(t) y(\boldsymbol{x}, \boldsymbol{\mu}) dt, \tag{3.26}$$

where the integral for fully-discrete problems is typically temporally-discretized with the same discretization as the primal problem,

$$\bar{y} = \sum_{n=0}^{N_t} b^n \boldsymbol{x}^n \Delta t, \tag{3.27}$$

where $b(t)$ is a time-dependent integration weight function, and b^n are a weights at time node n composed of the product of the time-dependent weight function and quadrature weights. As in the steady case, the unsteady adjoint is the sensitivity of the unsteady output

to perturbations of the unsteady residual,

$$\bar{\boldsymbol{\Psi}} = \frac{\partial \bar{y}}{\partial \bar{\boldsymbol{R}}}. \tag{3.28}$$

Similarly to Section 3.2, a perturbation analysis can be used to derive the discrete adjoint for unsteady problems. Perturbations to the input of the system create perturbations to the spatial-temporal residual necessitating changes in states at all times after the perturbation, creating a change in the output trajectory,

$$\delta\boldsymbol{\mu} \longrightarrow \delta\bar{\boldsymbol{R}}^m \longrightarrow \delta\boldsymbol{x}^n \longrightarrow \delta\bar{y}. \tag{3.29}$$

The adjoint equation is similar to its steady equivalent Equation (3.20) but is substantially larger as there are N_t adjoints of size $n_{\boldsymbol{x}}$, according to

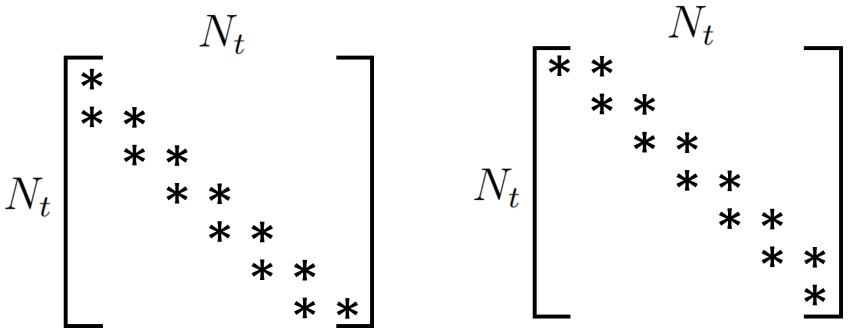
$$\sum_{m=1}^{N_t} \left[\frac{\partial \bar{\boldsymbol{R}}^m}{\partial \boldsymbol{x}^n}\right]^T \bar{\boldsymbol{\Psi}}^m + \left[\frac{\partial \bar{y}}{\partial \boldsymbol{x}^n}\right]^T = \boldsymbol{0}. \tag{3.30}$$

The intractability of this problem can be overcome by taking advantage of the structure of the linear system, as is done with the initial primal solution. Although Equation (3.23) forms N_t $n_{\boldsymbol{x}} \times n_{\boldsymbol{x}}$ linear systems, solutions are obtained for each time node sequentially, beginning at the initial time node and going to N_t. This is done as solutions at any time node only depend on solutions from previous time nodes, and thus the full linear system is a lower-triangular matrix, as seen in Figure 3.1. In the adjoint system there is a transpose of the linearization of the state. Thus, the linear operator in the time-dependent adjoint system is an upper-triangular matrix and is most efficiently solved with backwards substitution – *i.e.*, marching backwards in time starting from N_t and going to the initial time node. Backwards-in-time marching of the adjoint solution can be conveniently performed with a forward-in-time marching framework by substituting a reverse time variable $\tau = t_f - t$,

$$\boldsymbol{M}\frac{\partial \bar{\boldsymbol{\Psi}}_h(\tau)}{\partial \tau} + \left[\frac{\partial \boldsymbol{R}_h}{\partial \boldsymbol{x}_h}\right]^T \bar{\boldsymbol{\Psi}}_h(\tau) + \left[\frac{\partial \bar{y}(\tau)}{\partial \boldsymbol{x}_h}\right]^T = \boldsymbol{0}, \tag{3.31}$$

with the terminal condition,

$$\bar{\boldsymbol{\Psi}}_h(t = t_f) = \boldsymbol{M}^{-1}\frac{\partial \bar{y}}{\partial \boldsymbol{x}_h}^T. \tag{3.32}$$



(a) Primal Jacobian $\frac{\partial \bar{\boldsymbol{R}}}{\partial \boldsymbol{x}}$ **(b)** Adjoint Jacobian $\left[\frac{\partial \bar{\boldsymbol{R}}}{\partial \boldsymbol{x}}\right]^T$

Figure 3.1: Jacobian of primal and adjoint systems. Each $*$ is a $n_{\boldsymbol{x}} \times n_{\boldsymbol{x}}$ block matrix. Due to the structure of these Jacobians, the primal system is most efficiently solved forward in time (*i.e.*, forward substitution) and the adjoint system is most efficiently solved backwards in time (backwards substitution).

Finally, the unsteady output error estimate is,

$$\bar{y}_h^H - \bar{y}_h = \sum_{n=0}^{N_t} w^n \left[\bar{\boldsymbol{\Psi}}^T \boldsymbol{R}_h\right]^n \Delta t, \tag{3.33}$$

where w^n are weights determined by the quadrature rule used (*e.g.*, trapezoidal rule).

3.3 Adjoint-weighted Error Estimation for Projection-based ROMs

The previous section presented the derivation of adjoints for a generic conservative system. A similar approach can be taken with projection-based ROMs. This was first introduced for POD by Meyers and Matthies [37]. This thesis extends these methods to DEIM hyperreduction. First, recognize that in the previous discussion, the error estimates pertain to different mesh resolution or approximation order in the discretization. For projection-based ROMs the analogous resolution of the problem can be taken as the rank of the state basis ($n_{\boldsymbol{V}}$), the rank of the residual basis ($n_{\boldsymbol{U}}$), and the rank of the DEIM sampling matrix ($n_{\boldsymbol{P}}$). For the sake of simplicity, consider a general projection-based ROM residual defined by

$$\boldsymbol{R}_{\text{ROM}} = \boldsymbol{C}_{\text{ROM}} \boldsymbol{R}(\boldsymbol{V}\hat{\boldsymbol{x}}) : \begin{cases} \boldsymbol{C}_{\text{ROM}} = \boldsymbol{V}^T \ \text{(POD)}, \\ \boldsymbol{C}_{\text{ROM}} = \boldsymbol{V}\boldsymbol{U}^T \left[\boldsymbol{P}^T \boldsymbol{U}\right]^\dagger \boldsymbol{P}^T \ \text{(DEIM)}. \end{cases} \tag{3.34}$$

where $\boldsymbol{C}_{\text{ROM}} \in \mathbb{R}^{n_{\boldsymbol{V}} \times n_{\boldsymbol{x}}}$ is a linear operator that transforms the FOM residual into the ROM residual. A coarse ROM solution ($\hat{\boldsymbol{x}}_{H_{n_{\boldsymbol{V}}}}$) arises from driving the coarse ROM residual to

zero, *i.e.*,

$$\boldsymbol{R}_{\text{ROM},H}(\boldsymbol{V}_{H_{\boldsymbol{V}}}\hat{\boldsymbol{x}}_{H_{\boldsymbol{V}}}) = \boldsymbol{0}. \tag{3.35}$$

The coarse solution can be injected into a finer POD approximation space. This is a lossless injection, as the coarse POD basis is a subspace of the finer POD basis ($\boldsymbol{V}_H \subset \boldsymbol{V}_h$),

$$\begin{aligned} \hat{\boldsymbol{x}}^{h_{\boldsymbol{V}}}_{H_{\boldsymbol{V}}} &= \boldsymbol{V}^T_{h_{\boldsymbol{V}}}\boldsymbol{V}_{H_{\boldsymbol{V}}}\hat{\boldsymbol{x}}_{H_{\boldsymbol{V}}}, \\ \hat{\boldsymbol{x}}^{h_{\boldsymbol{V}}}_{H_{\boldsymbol{V}}} &= \begin{bmatrix} \hat{\boldsymbol{x}}_{H_{\boldsymbol{V}}} \\ \boldsymbol{0}_{h_{\boldsymbol{V}}-H_{\boldsymbol{V}}} \end{bmatrix}. \end{aligned} \tag{3.36}$$

Additionally, for DEIM, the system itself can be injected to a finer system space by formulating a ROM linear projector with a greater number of residual basis vectors and/or sampling indices. This is also a lossless injection as the state representation is unchanged.

All three types of fine-space injections cause a perturbation of the residual through modifying $\boldsymbol{C}_{\text{ROM}}$ while $\boldsymbol{R}$ remains unchanged. This can be approximated linearly resulting in

$$\boldsymbol{R}_{\text{ROM},h}(\boldsymbol{V}_{h_{\boldsymbol{V}}}\hat{\boldsymbol{x}}^{h_{\boldsymbol{V}}}_{H_{\boldsymbol{V}}}) + \frac{\partial \boldsymbol{R}_{\text{ROM},h}}{\partial h_{\boldsymbol{V}}}\delta h_{\boldsymbol{V}} + \frac{\partial \boldsymbol{R}_{\text{ROM},h}}{\partial h_{\boldsymbol{U}}}\delta h_{\boldsymbol{U}} + \frac{\partial \boldsymbol{R}_{\text{ROM},h}}{\partial h_{\boldsymbol{P}}}\delta h_{\boldsymbol{P}} = \delta \boldsymbol{R}_{\text{ROM}}. \tag{3.37}$$

The fine-space ROM residual can be made zero with the solution to the fine-space ROM problem. The linearization of the fine-space ROM with respect to the fine-space state can be used to approximate the change in the perturbed residual needed to drive the problem to zero,

$$\delta\hat{\boldsymbol{x}}_{h_{\boldsymbol{V}}} = \hat{\boldsymbol{x}}_{h_{\boldsymbol{V}}} - \hat{\boldsymbol{x}}^{h_{\boldsymbol{V}}}_{h_{\boldsymbol{V}}}, \tag{3.38}$$

$$\begin{aligned} \boldsymbol{R}_{\text{ROM},h}(\boldsymbol{V}_{h_{\boldsymbol{V}}}\hat{\boldsymbol{x}}^{h_{\boldsymbol{V}}}_{H_{\boldsymbol{V}}}) + \frac{\partial \boldsymbol{R}_{\text{ROM},h}}{\partial h_{\boldsymbol{V}}}\delta h_{\boldsymbol{V}} + \frac{\partial \boldsymbol{R}_{\text{ROM},h}}{\partial h_{\boldsymbol{U}}}\delta h_{\boldsymbol{U}} \\ + \frac{\partial \boldsymbol{R}_{\text{ROM},h}}{\partial h_{\boldsymbol{P}}}\delta h_{\boldsymbol{P}} + \frac{\partial \boldsymbol{R}_{\text{ROM},h}}{\partial \hat{\boldsymbol{x}}_h}\delta\hat{\boldsymbol{x}}_{h_{\boldsymbol{V}}} = \boldsymbol{0}. \end{aligned} \tag{3.39}$$

This state perturbation also causes a change in the output,

$$y(\boldsymbol{V}_{h_{\boldsymbol{V}}}\hat{\boldsymbol{x}}^{h_{\boldsymbol{V}}}_{H_{\boldsymbol{V}}}) + \frac{\partial y}{\partial \hat{\boldsymbol{x}}_{h_{\boldsymbol{V}}}}\delta\hat{\boldsymbol{x}}_{h_{\boldsymbol{V}}} = y(\boldsymbol{V}_{h_{\boldsymbol{V}}}\hat{\boldsymbol{x}}_{h_{\boldsymbol{V}}}). \tag{3.40}$$

Subtracting Equation (3.37) from Equation (3.39) and solving for the change in the state

yields

$$\delta\hat{\boldsymbol{x}}_{h_{\boldsymbol{V}}} = -\left[\frac{\partial \boldsymbol{R}_{\text{ROM},h}}{\partial \hat{\boldsymbol{x}}_h}\right]^{-1} \delta \boldsymbol{R}_{\text{ROM}}. \tag{3.41}$$

Like with the FOM derivation, this equation can be used with the linear approximation of the fine-space output to obtain an output error estimate. Recognizing that the change in the residual is equal to the residual evaluated with the coarse ROM solution injected into the fine-space yields the output error estimate for the ROM system, defined by,

$$y(\boldsymbol{V}_{h_{\boldsymbol{V}}}\hat{\boldsymbol{x}}^{h_{\boldsymbol{V}}}_{H_{\boldsymbol{V}}}) - y(\boldsymbol{V}_{h_{\boldsymbol{V}}}\hat{\boldsymbol{x}}_{h_{\boldsymbol{V}}}) = \underbrace{-\frac{\partial y}{\partial \hat{\boldsymbol{x}}_{h_{\boldsymbol{V}}}}\left[\frac{\partial \boldsymbol{R}_{\text{ROM},h}}{\partial \hat{\boldsymbol{x}}_h}\right]^{-1}}_{\boldsymbol{\Psi}^T_{\text{ROM}}} \delta \boldsymbol{R},$$

$$y(\boldsymbol{V}_{h_{\boldsymbol{V}}}\hat{\boldsymbol{x}}^{h_{\boldsymbol{V}}}_{H_{\boldsymbol{V}}}) - y(\boldsymbol{V}_{h_{\boldsymbol{V}}}\hat{\boldsymbol{x}}_{h_{\boldsymbol{V}}}) = \boldsymbol{\Psi}^T_{\text{ROM}} \boldsymbol{R}_{\text{ROM},h}(\boldsymbol{V}_{h_{\boldsymbol{V}}}\hat{\boldsymbol{x}}^{h_{\boldsymbol{V}}}_{H_{\boldsymbol{V}}}), \tag{3.42}$$

$$y(\boldsymbol{V}_{h_{\boldsymbol{V}}}\hat{\boldsymbol{x}}^{h_{\boldsymbol{V}}}_{H_{\boldsymbol{V}}}) - y(\boldsymbol{V}_{h_{\boldsymbol{V}}}\hat{\boldsymbol{x}}_{h_{\boldsymbol{V}}}) = \boldsymbol{\Psi}^T_{\text{ROM}} \boldsymbol{C}_{\text{ROM}} \boldsymbol{R}_h(\boldsymbol{V}_{h_{\boldsymbol{V}}}\hat{\boldsymbol{x}}^{h_{\boldsymbol{V}}}_{H_{\boldsymbol{V}}}). \tag{3.43}$$

Finally, the ROM adjoint equation is

$$\left[\frac{\partial\left[\boldsymbol{C}_{\text{ROM}}\boldsymbol{R}_h\right]}{\partial \hat{\boldsymbol{x}}_{h_{\boldsymbol{V}}}}\right]^T \boldsymbol{\Psi}_h + \left[\frac{\partial y}{\partial \hat{\boldsymbol{x}}_{h_{\boldsymbol{V}}}}\right]^T = \boldsymbol{0}. \tag{3.44}$$

Applying the product and chain rules to the terms in Equation (3.20) with $\frac{\partial \boldsymbol{C}_{\text{ROM}}}{\partial \hat{\boldsymbol{x}}_h} = \boldsymbol{0}$, yielding

$$\frac{\partial\left[\boldsymbol{C}_{\text{ROM}}\boldsymbol{R}_h\right]}{\partial \hat{\boldsymbol{x}}_{h_{\boldsymbol{V}}}} = \boldsymbol{C}_{\text{ROM}}\frac{\partial \boldsymbol{R}_h}{\partial \boldsymbol{x}}\boldsymbol{V}_{h_{\boldsymbol{V}}},$$

$$\frac{\partial y}{\partial \hat{\boldsymbol{x}}_{h_{\boldsymbol{V}}}} = \frac{\partial y}{\partial \boldsymbol{x}}\boldsymbol{V}_{h_{\boldsymbol{V}}},$$

which further yields

$$\left[\boldsymbol{C}_{\text{ROM}}\frac{\partial \boldsymbol{R}_h}{\partial \boldsymbol{x}}\boldsymbol{V}_{h_{\boldsymbol{V}}}\right]^T \boldsymbol{\Psi}_{\text{ROM}} + \left[\frac{\partial y}{\partial \boldsymbol{x}}\boldsymbol{V}_{h_{\boldsymbol{V}}}\right]^T = \boldsymbol{0}. \tag{3.45}$$

Equation (3.45) is a useful expression as the first term is the Jacobian of the ROM system and the second term is a product of the FOM output linearilzation and the ROM state basis.

Deriving unsteady adjoints for ROM systems can be approached similarly to how unsteady adjoints were obtained for the FOM. Changes to the relative metrics for the ROMs has influence on the residual, solution, and output trajectory at all times after the perturba-

tion,

$$\delta(n_{\boldsymbol{V}}, n_{\boldsymbol{U}}, n_{\boldsymbol{P}}) \longrightarrow \delta\boldsymbol{R}^m \longrightarrow \delta\boldsymbol{x}^n \longrightarrow \delta y(t). \tag{3.46}$$

This forms the analogous discrete adjoint system for unsteady ROMs, according to

$$\sum_{m=1}^{N_t} \left[\frac{\partial \left[\boldsymbol{C}_{\text{ROM}} \bar{\boldsymbol{R}}_h^m \right]}{\partial \hat{\boldsymbol{x}}_{h_{n_{\boldsymbol{V}}}}^n} \right]^T \bar{\boldsymbol{\Psi}}_{\text{ROM}}^m + \left[\frac{\partial \bar{y}}{\partial \hat{\boldsymbol{x}}_{h_{n_{\boldsymbol{V}}}}^n} \right]^T = \boldsymbol{0}. \tag{3.47}$$

Applying the chain and product rules to Equation (3.47) yields

$$\sum_{m=1}^{N_t} \left[\boldsymbol{C}_{\text{ROM}} \frac{\partial \bar{\boldsymbol{R}}_h^m}{\partial \boldsymbol{x}^n} \boldsymbol{V}_{h_{n_{\boldsymbol{V}}}} \right]^T \bar{\boldsymbol{\Psi}}_{\text{ROM}}^m + \left[\frac{\partial \bar{y}}{\partial \boldsymbol{x}^n} \boldsymbol{V}_{h_{n_{\boldsymbol{V}}}} \right]^T = \boldsymbol{0}. \tag{3.48}$$

Backwards-in-time marching efficiently solves the above system due to having the same upper triangular structure that characterizes the FOM unsteady discrete adjoint system. Using the reverse time variable $\tau = t_f - (t - t_0)$ yields

$$\hat{\boldsymbol{M}}^T \frac{d\boldsymbol{\Psi}_{\text{ROM}}}{d\tau} + \left[\boldsymbol{C}_{\text{ROM}} \frac{\partial \boldsymbol{R}_h}{\partial \boldsymbol{x}} \boldsymbol{V}_{h_{n_{\boldsymbol{V}}}} \right]^T \boldsymbol{\Psi}_{\text{ROM}} - \left[\frac{\partial y}{\partial \boldsymbol{x}} \boldsymbol{V}_{h_{n_{\boldsymbol{V}}}} \right]^T = \boldsymbol{0}, \tag{3.49}$$

which is solved with the terminal condition,

$$\bar{\boldsymbol{\Psi}}_h(t = t_f) = \hat{\boldsymbol{M}}^{-T} \left[\frac{\partial \bar{y}}{\partial \boldsymbol{x}} \boldsymbol{V}_{h_{n_{\boldsymbol{V}}}} \right]^T. \tag{3.50}$$

For the Galerkin formulation of a finite-element discretization, the mass matrix is symmetric (*i.e.*, $\boldsymbol{M}^T = \boldsymbol{M}$. This is also true for the Galerkin formulation of POD and DEIM; however, if a Petrov-Galerkin ROM is used, the reduced mass matrix is not symmetric, and the adjoint system is solved with this distinction taken into account.

Finally, the unsteady output error estimate for the ROM is

$$\bar{y}_h^H - \bar{y}_h = \sum_{n=0}^{N_t} w^n \left[\bar{\boldsymbol{\Psi}}_{\text{ROM}}^T \boldsymbol{C}_{\text{ROM}} \boldsymbol{R}_h \right]^n \Delta t, \tag{3.51}$$

where w^n are weights determined by the quadrature rule used (*e.g.*, trapezoidal rule).

3.4 POD and DEIM Adaptation

Various POD and DEIM adaptation techniques have been employed to modify the state basis, residual basis, and sampling indices. These techniques can be separated into those

that utilize precomputed adaptation search spaces for additional basis vectors and/or sampling indices, and those that compute new basis vectors and/or sampling indices to better approximate the problem being solved. Feng, Mangold, and Benner [44] give an example of the former by adapting the POD state and DEIM residual basis by shifting the truncating index based on output error bounds defined by Zhang *et al.* [43]. Pehertorfer and Willcox showed that adaptation for DEIM models can be done efficiently for the interpolation points by comparing the error of the residual approximation at randomly chosen indices [105], as an example of the latter. For this thesis, candidate adaptation vectors and indices are obtained from the finer space of basis vectors and sampling indices that are truncated when constructing the POD/DEIM ROMs. However, unlike many adaptation methods, this approach selects additional basis vectors and sampling indices based on their output prediction quality instead of their singular value energy content.

Error estimates defined in Section 3.3 provide a mechanism for adaptation. For full-order steady adjoints, the inner product between the adjoints and the vector of residuals in Equation (3.19) can be separated into its additive parts, *i.e.*,

$$e_i = \left| \boldsymbol{\Psi}_{h,i}^T \boldsymbol{R}_h(\boldsymbol{x}_h^H)_i \right|, \tag{3.52}$$

where $\Psi_{h,i}$ and $R_h(\boldsymbol{x}_h^H)_i$ are the i^{th} entry of the fine-space adjoint and fine-space residual of the injected solution, respectively. Each weighted residual indicates the amount of error that is contributed by the corresponding degree of freedom in the fine-space discretiziation. For unsteady adjoint error estimation, the error contributions for the fine-space degrees of freedom can be obtained from collecting the discrete error contributions in time,

$$\bar{e}_i = \sum_{n=0}^{N_t} w^n \left| [\boldsymbol{\Psi}_{h,i}^n]^T \boldsymbol{R}_h^n(\boldsymbol{x}_h^H)_i \right| \Delta t. \tag{3.53}$$

The degrees of freedom i contributing the largest error are targeted for adaptation to obtain a new model. This is referred to as *error localization* in the literature and has been used for various applications, such as for mesh refinement, p-adaptation in finite-element methods, and temporal refinement [106, 107, 108, 109].

This thesis uses analogs to error localization for POD and DEIM models for adaptation. For steady POD, the injection of the coarse-space ROM solution into the fine space is lossless, and the resulting error localization produces error contributions on the truncated POD basis vectors, defined by

$$e_i = \left| \Psi_{i,\text{ROM}} \left[\boldsymbol{v}_i^T \boldsymbol{R} \right] \right|, \tag{3.54}$$

where $\boldsymbol{v}_i$ is the i^{th} POD basis vector. For unsteady POD, the state basis error contribution takes the form

$$\bar{e}_i = \sum_{n=0}^{N_t} \left| \Psi_{i,\text{ROM}}^n \left[\boldsymbol{v}_i^T \boldsymbol{R}^n \right] \right| \Delta t. \tag{3.55}$$

To adapt POD-ROMs, first error localization is performed; then the weights e_i are ordered by magnitude and the basis vectors that contribute the most to the error estimate are added to the online ROM basis.

Unlike POD error localization, DEIM error localization is produced on multiple different metrics: the state basis size, the residual basis size, and the total number of sampling indices. The error estimates and adjoints for DEIM are constructed with the injection of the solution and system into the full fine-space (*i.e.*, where all three metrics are finer). These adjoints are then used for error localization; however to isolate the error produced by the individual metrics, error localization only occurs in partial fine-spaces – *i.e.*, where only the metric being analyzed is fine and all other metrics are kept coarse. For steady-state, this is defined as

$$e_{i,\boldsymbol{V}} = \left| \Psi_{i,\text{ROM}} \left[\boldsymbol{v}_i \boldsymbol{U}_{H_U} \left[\boldsymbol{P}_{H_P} \boldsymbol{U}_{H_U}^T \right]^\dagger \boldsymbol{P}_{H_P}^T \boldsymbol{R} \right] \right|, \tag{3.56}$$

$$e_{i,\boldsymbol{U}} = \left| \left[\Psi_{i,\text{ROM}} \boldsymbol{V}_{H_V} \right]_i \left[\boldsymbol{u}_i^T \left[\boldsymbol{P}_{H_P} \boldsymbol{U}_{h_U}^T \right]^\dagger \boldsymbol{P}_{H_P}^T \boldsymbol{R} \right]_i \right|, \tag{3.57}$$

$$e_{i,\boldsymbol{P}} = \left| \left[\Psi_{i,\text{ROM}} \boldsymbol{V}_{H_V} \boldsymbol{U}_{H_U} \left[\boldsymbol{P}_{h_P} \boldsymbol{U}_{H_U}^T \right]^\dagger \right]_j \left[\boldsymbol{p}_i^T \boldsymbol{R} \right] \right|, \tag{3.58}$$

and for unsteady cases as

$$\bar{e}_{i,\boldsymbol{V}} = \sum_{n=0}^{N_t} \left| \Psi_{i,\text{ROM}}^n \left[\boldsymbol{v}_i \boldsymbol{U}_{H_U} \left[\boldsymbol{P}_{H_P} \boldsymbol{U}_{H_U}^T \right]^\dagger \boldsymbol{P}_{H_P}^T \boldsymbol{R}^n \right] \right| \Delta t, \tag{3.59}$$

$$\bar{e}_{i,\boldsymbol{U}} = \sum_{n=0}^{N_t} \left| \left[\Psi_{i,\text{ROM}}^n \boldsymbol{V}_{H_V} \right]_i \left[\boldsymbol{u}_i^T \left[\boldsymbol{P}_{H_P} \boldsymbol{U}_{h_U}^T \right]^\dagger \boldsymbol{P}_{H_P}^T \boldsymbol{R}^n \right]_i \right| \Delta t, \tag{3.60}$$

$$\bar{e}_{i,\boldsymbol{P}} = \sum_{n=0}^{N_t} \left| \left[\Psi_{i,\text{ROM}}^n \boldsymbol{V}_{H_V} \boldsymbol{U}_{H_U} \left[\boldsymbol{P}_{h_P} \boldsymbol{U}_{H_U}^T \right]^\dagger \right]_j \left[\boldsymbol{p}_i^T \boldsymbol{R}^n \right] \right| \Delta t, \tag{3.61}$$

where $\boldsymbol{u}_i$ is the i^{th} residual basis vector, $\boldsymbol{p}_i$ is the i^{th} column of $\boldsymbol{P}$, and i and j on the outside

of matrix products indicate the i^{th} or j^{th} column or row of that product. The full fine-space adjoint has been used in adjoint-weighted residual error estimation for fluid systems [110]. The overall error estimate can be used to assess the quality of the ROM solutions and criteria for adaptation. Once the need for adaptation is determined, the relative sizes of the error contributions of the metrics derived from error localization are more important for adaptation than the absolute sizes, which would require adjoint solutions for each partial fine space. The process for error estimation and adaptation of DEIM models is shown in Algorithm 2.

Algorithm 2 Error Estimation and Adaptation for DEIM-ROM models

Beginning with coarse-space ROM: rank $(\boldsymbol{V}) = H_{\boldsymbol{V}}$, rank $(\boldsymbol{U}) = H_{\boldsymbol{U}}$, and rank $(\boldsymbol{P}) = H_{\boldsymbol{P}}$
Iteration $\leftarrow 0$
while Iteration $<$ Max Iteration **do**
  Solve ROM and obtain coarse-solution, $\boldsymbol{x}_H$
  Inject ROM into overall fine-space and obtain overall fine-space adjoints with Equation (3.45), $\Psi_{h,\text{ROM}}$
  Use overall fine-space adjoint and state injection to find error estimate E_y with Equation (3.40)
  if $E_y >$ tolerance **then**
    break
  end if
  Localize errors using Equation (3.56) –Equation (3.58) or Equation (3.59)–Equation (3.61)
  Individually sort localized errors $e_{i,\boldsymbol{V}}$, $e_{i,\boldsymbol{U}}$, and $e_{i,\boldsymbol{P}}$ by magnitude: $s_{i,\boldsymbol{V}}$, $s_{i,\boldsymbol{U}}$, and $s_{i,\boldsymbol{P}}$.
  Update ROM metrics:
  $\boldsymbol{V} \leftarrow \begin{bmatrix}\boldsymbol{V} & \boldsymbol{v}_{s_{0,\boldsymbol{V}}} & \dots & \boldsymbol{v}_{s_{k,\boldsymbol{V}}}\end{bmatrix}$
  $\boldsymbol{U} \leftarrow \begin{bmatrix}\boldsymbol{U} & \boldsymbol{u}_{s_{0,\boldsymbol{U}}} & \dots & \boldsymbol{u}_{s_{l,\boldsymbol{U}}}\end{bmatrix}$
  $\boldsymbol{P} \leftarrow \begin{bmatrix}\boldsymbol{P} & \boldsymbol{p}_{s_{0,\boldsymbol{P}}} & \dots & \boldsymbol{p}_{s_{m,\boldsymbol{P}}}\end{bmatrix}$
  Update coarse-space sizes: $H_{\boldsymbol{V}} \leftarrow H_{\boldsymbol{V}} + k$, $H_{\boldsymbol{U}} \leftarrow H_{\boldsymbol{U}} + l$, $H_{\boldsymbol{P}} \leftarrow H_{\boldsymbol{P}} + m$
  Iteration $\leftarrow$ Iteration $+1$
end while

3.5 Example Applications of Error Estimation

The following section demonstrates the techniques developed in this chapter on three problems: error estimation of a 2D scalar advection-diffusion problem, error estimation and adaptation on a 3D wing undergoing forced rigid plunging, and error estimation and adaptation of a pitching and plunging airfoil undergoing flow with compressible Navier-Stokes physics. The first of these exercises is meant to verify the theory and implementation of

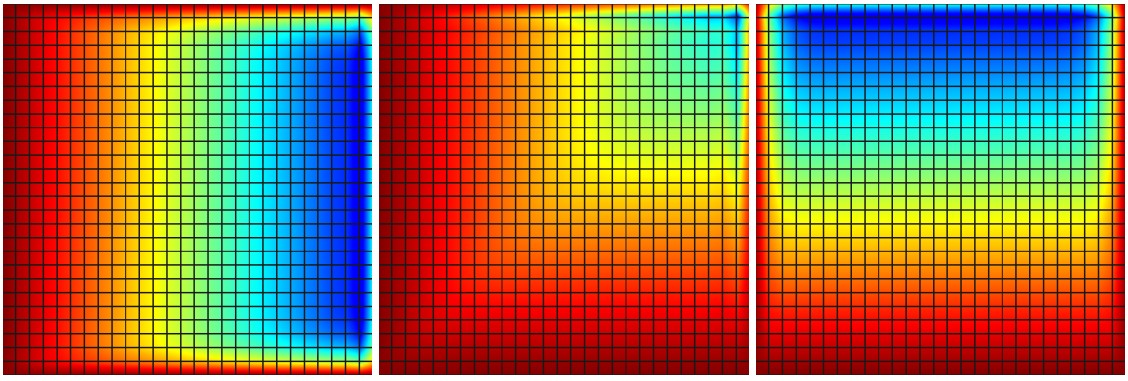
these adjoint error estimates for a problem whose output and residual are linear with respect to the state. The second exercise is meant to test the error estimates and adaptation mechanism on a problem whose outputs and residual are nonlinear with respect to the state. The final exercise is meant to demonstrate the benefits of refinement driven by output prediction over singular value energy content.

3.5.1 Steady-state Scalar Model

The first test case to demonstrate our ROM error estimation technique is a scalar advection-diffusion system in a square domain, on a uniform grid. The governing equation is

$$\nabla \cdot (\vec{v}x) - \nu \nabla^2 x + S = 0, \tag{3.62}$$

where $\vec{v}$ is the velocity field of the scalar quantity x which has a diffusivity of ν and a source S. The mesh consists of 27^2 quadrilateral elements, on which the state is approximated with bi-linear Lagrangian polynomials. 2D bi-linear DG states have 4 degrees of freedom, resulting in 2916 degrees of freedom for the entire problem. The length of the sides of the domain is 3 units. State snapshots were obtained by varying the flow direction, α, in the range $[0°, 90°]$ with a velocity magnitude of 1. The boundary conditions are all homogeneous Dirichlet (zero), and a constant unit source is added to the equation. The diffusivity is $\nu = 0.01$, for a domain-based Pèclèt number of 100. The target output is heat flux through the right boundary.

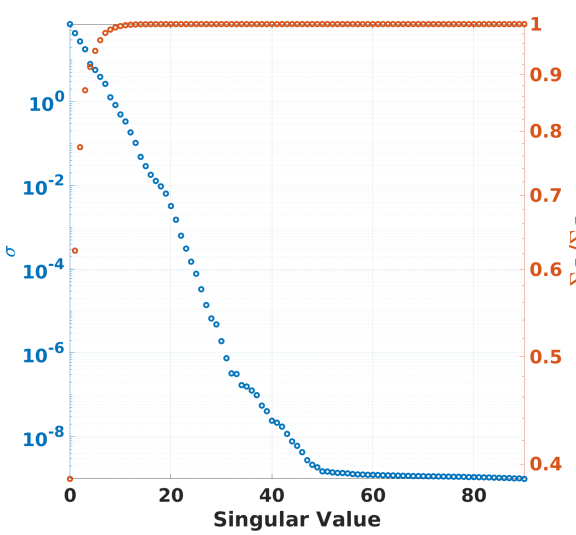


(a) $\alpha = 0°$ **(b)** $\alpha = 45°$ **(c)** $\alpha = 90°$

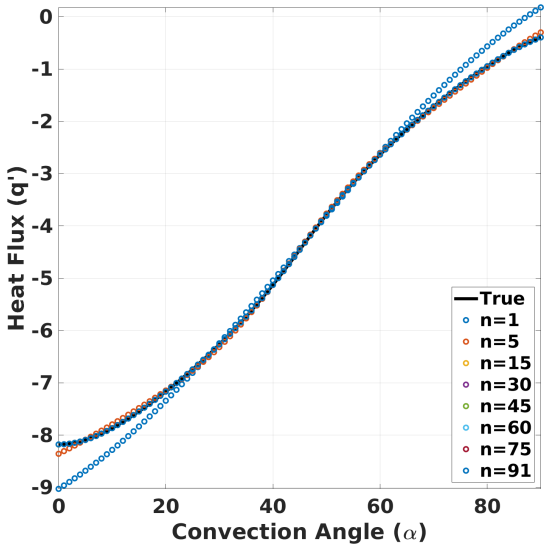
Figure 3.2: Solutions to scalar test problem.

Solutions to this system were generated with `xflow` – an in-house, high-order DG solver [111]. 91 snapshots were taken with even sampling of the internal flow angle. 99.9% of the total singular value energy is contained in the first 15 singular values. Each half-angle within the training domain is used for testing the POD and DEIM ROMs. Figure 3.3

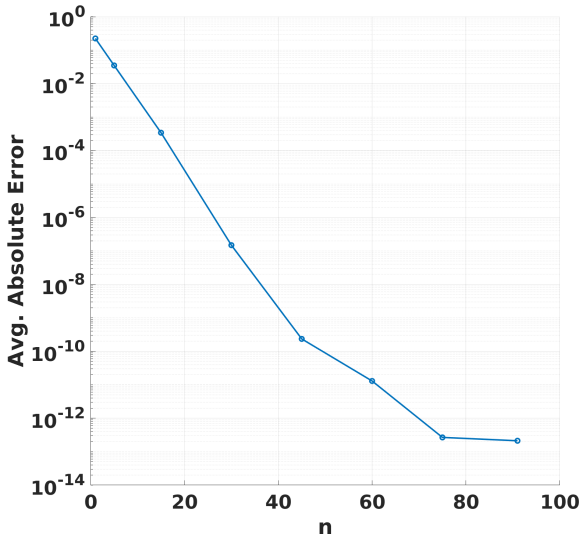
shows a plot of the singular values as well as the heat flux of the full-order system and its prediction by a POD-ROM using various numbers of basis vectors.



(a) σ and cumulative sum



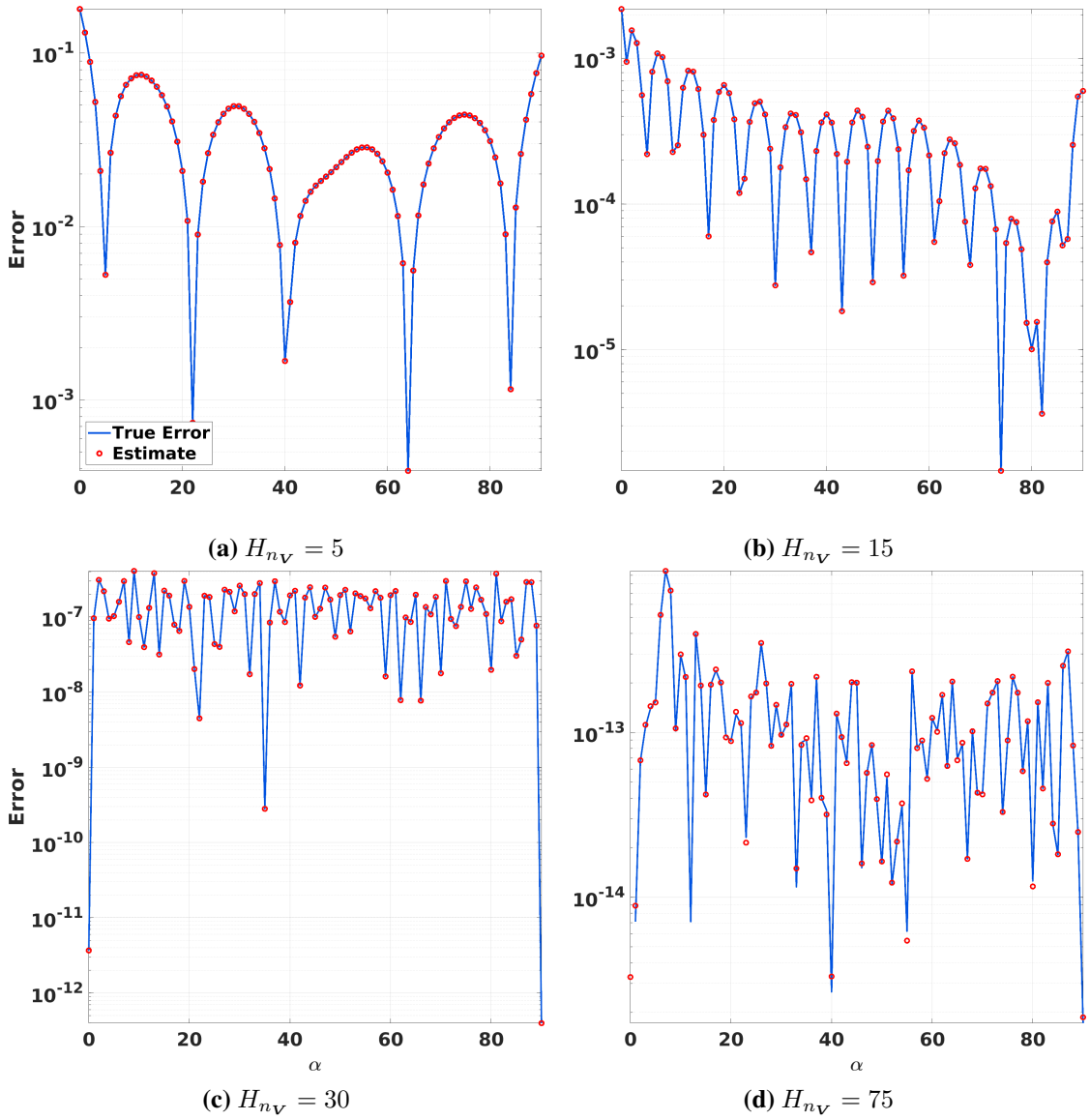
(b) Output predictions



(c) Average ROM output error

Figure 3.3: Solution for the scalar advection diffusion problem. The heat flux of the right boundary for each ROM model is displayed; additionally the average absolute heat flux error of each model is shown.

The adjoint-weighted residual techniques shown earlier in this chapter are used to construct heat flux errors for the solutions to different size POD with a fine-space of 91 basis vectors. Since the residual and output are linear with respect to the state, the true errors and the estimated errors should match exactly, as shown in Figure 3.4.

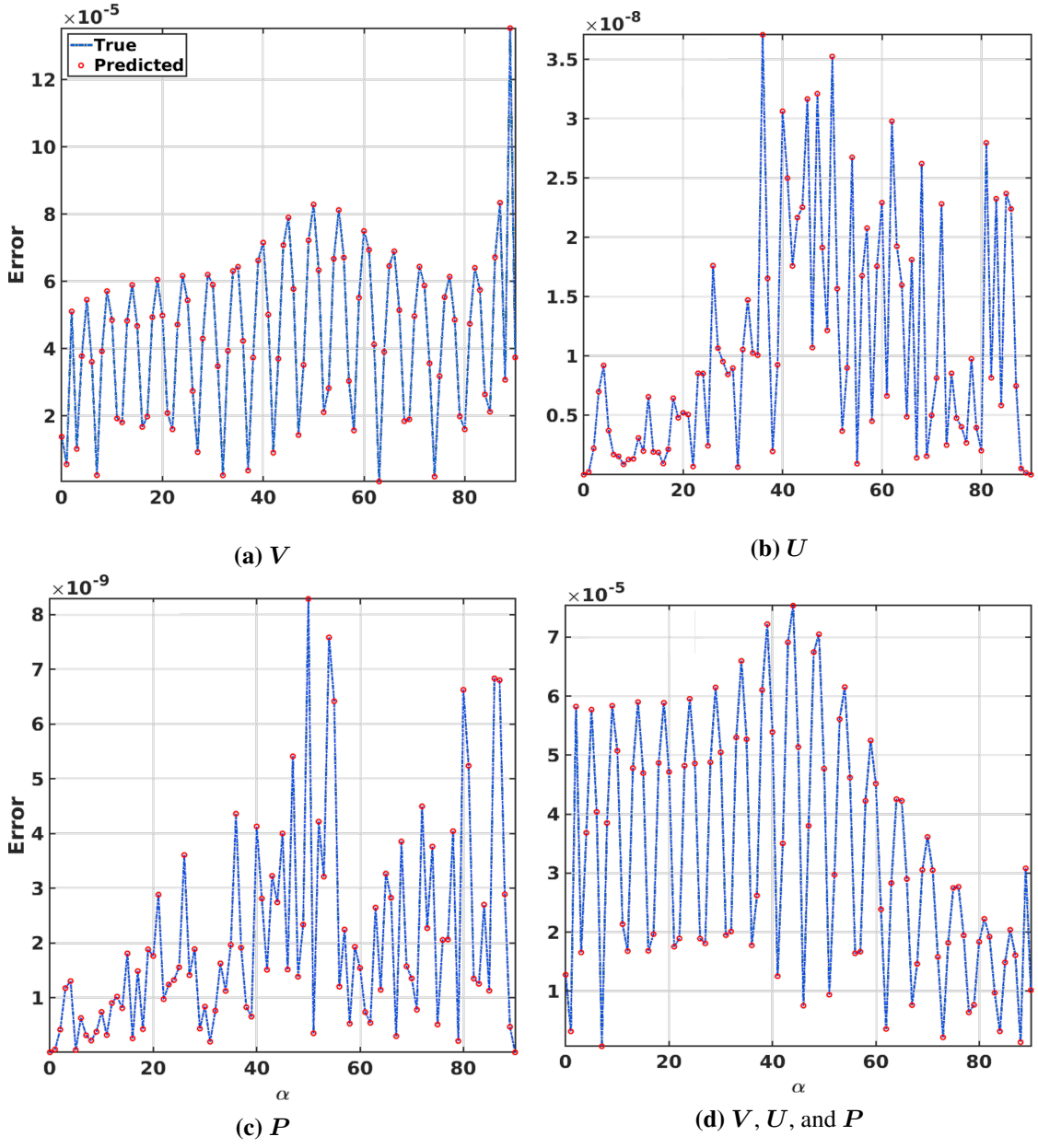


(a) $H_{n_V} = 5$ **(b)** $H_{n_V} = 15$

(c) $H_{n_V} = 30$ **(d)** $H_{n_V} = 75$

Figure 3.4: Error estimates of the output between coarse-space reduced models with H_{n_V} basis vectors and a fine-space reduced model with 91 basis vectors.

A DEIM model was made with the state basis from POD, a residual basis arising from the decomposition of residual snapshots, and sampling points obtained from the greedy procedure in Algorithm 1. Error estimates for these models are then obtained with respect to fine-space DEIM models, which increased each of the metrics individually and all three together. The true error between the coarse-space DEIM outputs and the fine-space DEIM solution was also computed, and as can be seen in Figure 3.5, the error estimates match the true error exactly. Once again, the exactness of the error estimates on this problem verifies both the implementation and theory underlying this chapter.



(a) $\boldsymbol{V}$ **(b)** $\boldsymbol{U}$

(c) $\boldsymbol{P}$ **(d)** $\boldsymbol{V}$, $\boldsymbol{U}$, and $\boldsymbol{P}$

Figure 3.5: Error estimates of the output between coarse-space DEIM models. The caption of the plot indicates which metric the error estimation was made. A DEIM model with those metrics coarse was solve while the other metrics were their fine-space size. The coarse-space sizes were $H_{n_{\boldsymbol{V}}} = 20$, $H_{n_{\boldsymbol{U}}} = 40$, and $H_{n_{\boldsymbol{P}}} = 80$. The fine-space sizes were $h_{n_{\boldsymbol{V}}} = 40$, $h_{n_{\boldsymbol{U}}} = 80$, and $h_{n_{\boldsymbol{P}}} = 160$.

3.5.2 3D Wing with Motion

3.5.2.1 Mesh Description

In the previous test case, the implementation of error estimates for POD and DEIM models was verified on a relatively tractable problem with benign physics. The following example explores the error estimation and adaptation methods on a more complex fluid model. This problem uses a high aspect-ratio wing configuration of the XRF1 model (XRF1-HARW). The XRF1 is an Airbus provided industrial standard multi-disciplinary research testcase representing a typical configuration for a long range wide body aircraft. The XRF1 research testcase is used by Airbus to engage with external partners on development and demonstration of relevant capabilities/technologies. The XRF1-HARW wing configuration was initially obtained with geometry optimization targeting the fuel efficiency, selected stress measurements, and buckling [112]. Several maneuvers and steady flight conditions were used as the testing conditions for the optimization with solutions obtained with RANS aerodynamics. The wing from the optimized full-body configuration was then isolated, and a mesh was generated for its use in this model problem. ADFlow [113], a multi-block, structured mesh, fluids-structure solver was used for the optimization. Images of the wing can be seen in Figure 3.6. The mesh has 14670 hexahedral elements, with large off-wall elements to facilitate efficient Euler solutions. The state inside each element is approximated with a second-order Lagrangian polynomial basis. The number of terms for each of the state polynomials is 10. With five states (density, three momenta, and energy) each element has 50 degrees of freedom or 733,500 total degrees of freedom in the entire mesh.

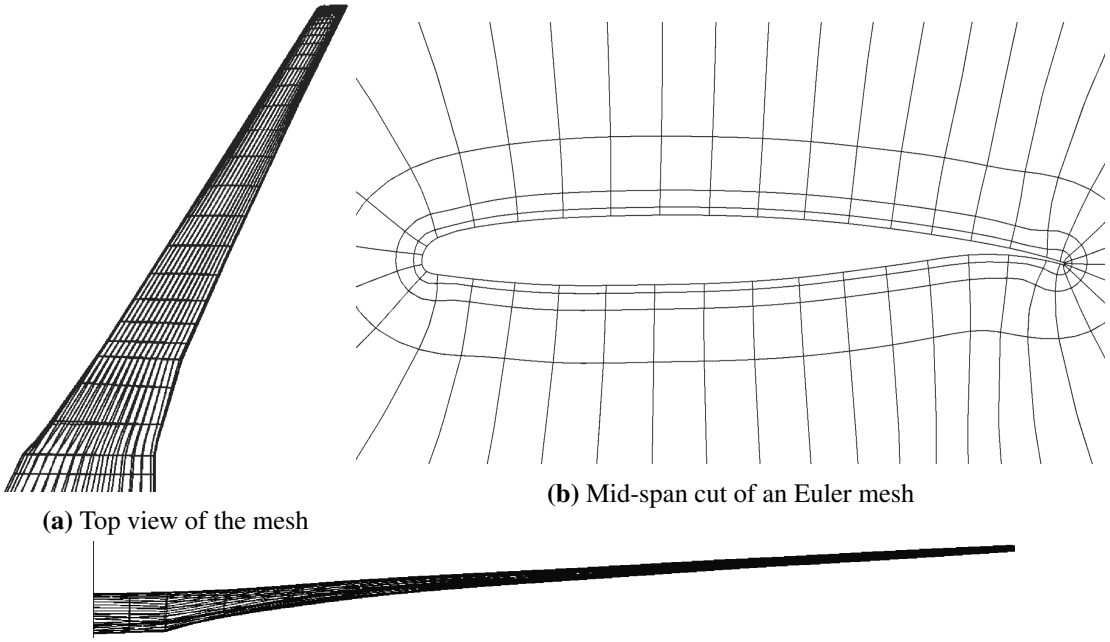
3.5.2.2 Model Description

Using `xflow` , unsteady inviscid solutions of the XRF1-HARW wing model were generated under transonic conditions and vertical plunging of various depths and frequencies. The freestream conditions were Mach 0.7 and 0° angle of attack. Two reduced frequencies were used for vertical plunging in order to compare the ROMs performance on more and less excited flow: 0.025 (low frequency) and 0.1 (high frequency). The reduced frequencies are based on the root semi-chord c_{root}. The period of the oscillation T is related to the reduced frequency by

$$T = \frac{2\pi c_{\text{root}}}{k\left|\vec{V}\right|} = \frac{1}{f}, \tag{3.63}$$

where $\left|\vec{V}\right|$ is the freestream velocity, and f is the linear frequency.

Each frequency dataset had five different plunging depths (a_{plunge}): 1.562%, 7.639%,



(a) Top view of the mesh

(b) Mid-span cut of an Euler mesh

(c) Horizontal view of the wing

Figure 3.6: XRF1-HARW wing-only (distorted) mesh.

13.02%, 18.63%, and 25% of the root chord. The simulations started from a steady-state solution of the wing without any motion or displacement. The motion was then exponentially ramped up to the full motion, achieving 63.21% of the full motion by the end of the first oscillation and then 99.97% of the full motion by the end of the second oscillation. The full description of the motion is

$$\Delta y(t) = a(t)\left[1 - \exp^{-\left(\frac{t}{T}\right)^3}\right], \tag{3.64}$$

$$a(t) = a_{\text{plunge}} \sin\left(2\pi f t + \phi\right), \tag{3.65}$$

where ϕ is the phase of the plunging. Figure 3.7 displays an example of the exponential ramp function used for this demonstration. Each training configuration was solved for four oscillations with 50 time steps per oscillation. The final oscillation was used for training, and the first three transient cycles were discarded. With the 50 samples for each of the five plunging depths, all ROMs were trained with 250 snapshots. The ROMs were tested on a plunging depth of 21.82% of the root chord, which is not in the training set. This value was chosen as it is the midpoint between the two largest sampled plunging depths. The steady solution of the wing-only model contains regions of supersonic flow and, as shown in Figure 3.8, a resulting shock near the tip of the wing.

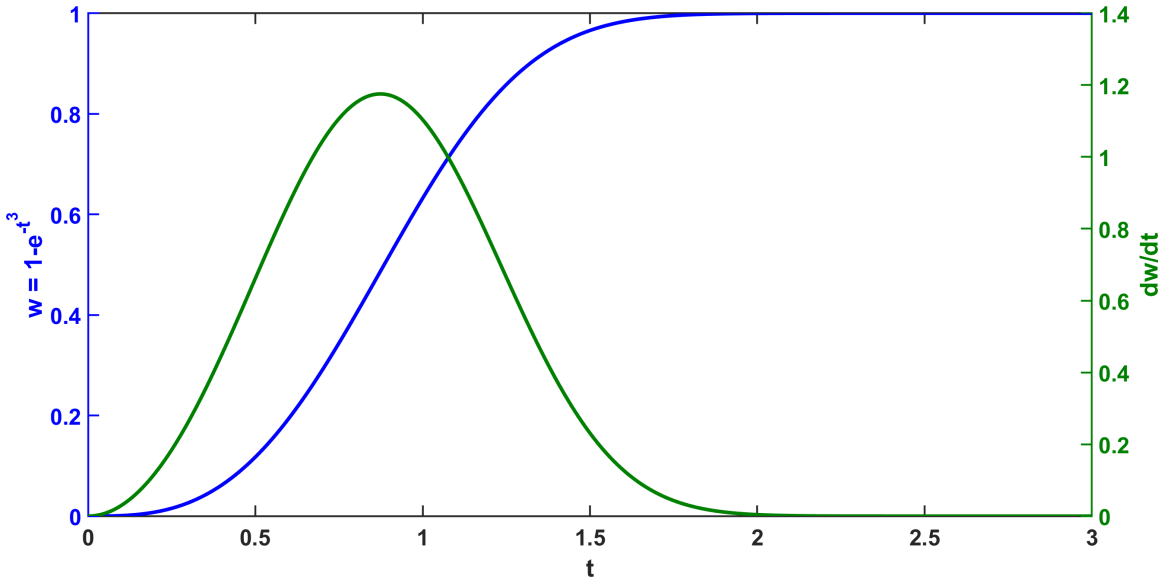
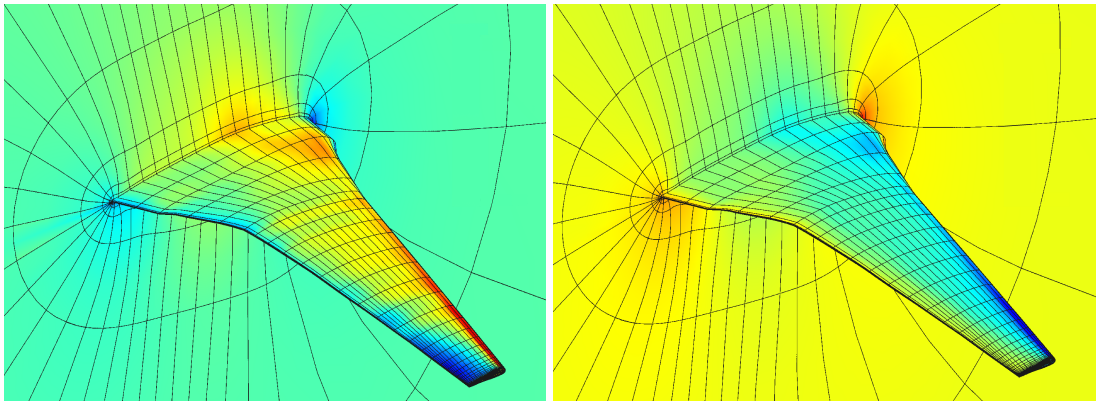
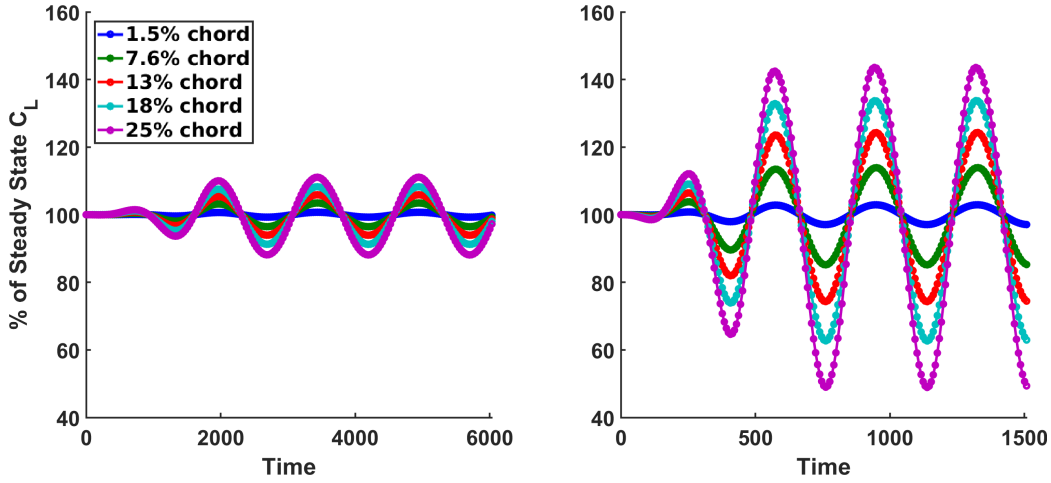


Figure 3.7: An example of the exponential function that was used to ramp the motion and its derivative.

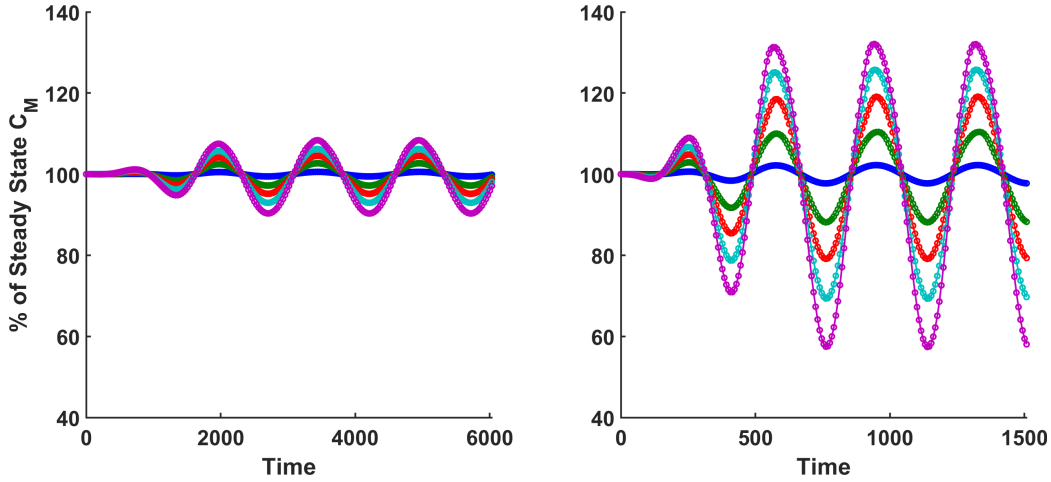


(a) Mach number **(b)** Pressure

Figure 3.8: Steady-state solution of XRF1 HARW wing only model with freestream flow conditions of 0.7 Mach number and 0° angle of attack.



(a) Lift



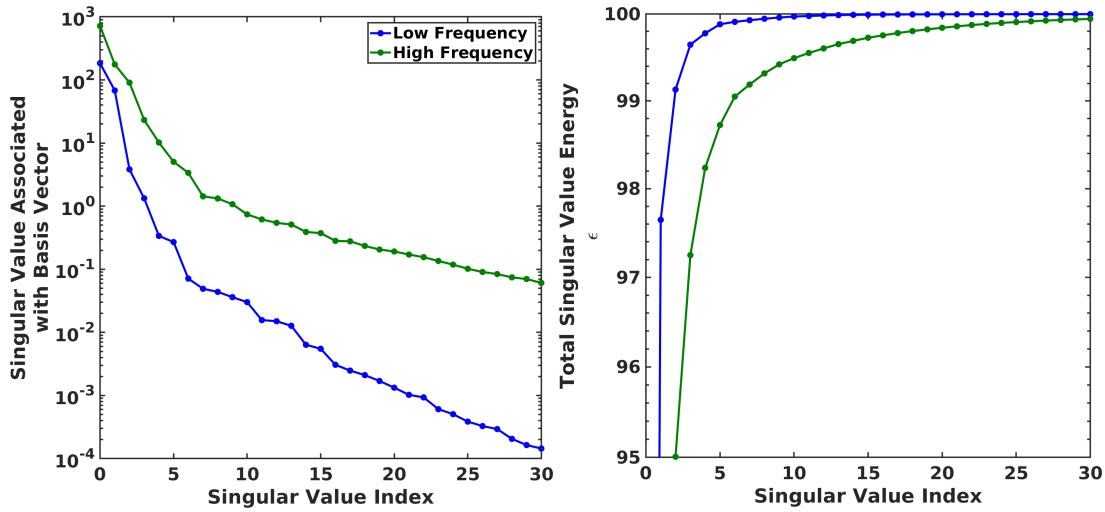
(b) Moment

Figure 3.9: Loads from the $k = 0.025$ and the $k = 0.10$ tests from training configurations. The set of snapshots consisted of the final period of the four period simulation shown here. The root chord is used as the reference chord for the plunging depth of the motion.

The instantaneous lift and moment coefficients of the plunge-excited motion oscillate about these values for both the low and high frequency cases, as seen in Figure 3.9. The immediately noticeable effect of increasing the reduced frequency of the plunging motion is an overall increase in load peak-to-peak amplitudes. The phase difference between the two datasets is caused by a phase difference in the induced motions.

3.5.2.3 ROM Results

Figure 3.10 shows plots of the singular values of the POD bases of the datasets. The singular values of the low frequency basis fall off in magnitude much earlier than the high frequency basis singular values. This is reflected in the size of the high frequency basis, which needs to be larger than the size of the low frequency basis to achieve the same



(a) Singular value **(b)** Energy content of basis vectors

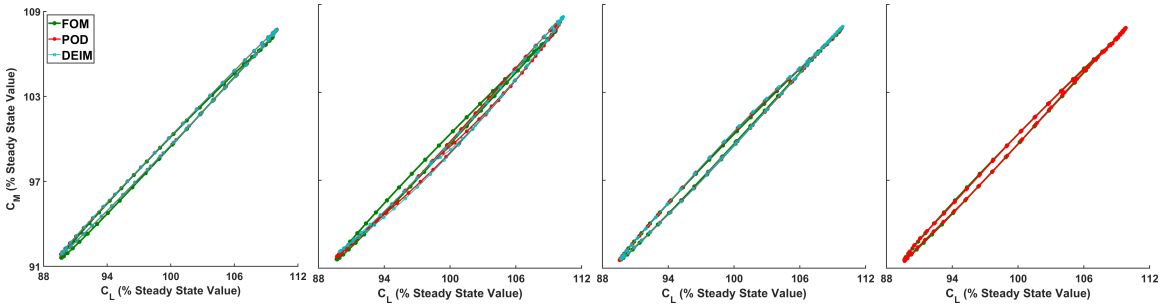
Figure 3.10: Singular values and percent of total singular value energy associated with basis vectors of the low frequency and high frequency datasets.

singular value energy level, ϵ. To capture $\epsilon = 95\%$ of the singular value energy, the first 2 and 3 basis vectors are needed for the low and the high frequency bases, respectively. This gap grows with energy level: for $\epsilon = 99\%$ the low frequency basis needs 3 vectors and the high frequency basis needs 7 vectors; for $\epsilon = 99.9\%$ the bases need 7 and 26 vectors; for $\epsilon = 99.99\%$ the bases need 15 and 51 vectors, respectively. This phenomenon is not caused specifically by the increase in reduced frequency but rather by the indirect increase in nonlinear physical behavior of the system. Apart from the small increase in computational cost, an issue that arises with the use of more basis vectors is the stability of the model may be reduced [20].

3.5.2.4 ROM Solutions

Due to the ROMs being constructed from only the oscillatory portion of the motion, an initial guess that had little or no transient features is needed. However, this initial guess cannot be obtained from the FOM as it would defeat the purpose of the ROM. Thus, interpolation was used to obtain an initial guess. The initial time-steps of the snapshots (*i.e.*, the first time-step of the 4$^{\text{th}}$ oscillation in Figure 3.9) were projected with the state bases, and then the initial guesses for the tests were obtained with a 2$^{\text{nd}}$ order polynomial fit the coefficients of these projections with respect to their plunging depths. ROM solutions were generated for the high frequency and low frequency models with $\epsilon = 95\%$, 99%, 99.9% and 99.99%.

Instantaneous lift and out-of-plane moment coefficients for the POD and DEIM solutions of different state basis singular value energy content are shown in Figure 3.11 and



(a) $\epsilon = 95\%$ **(b)** $\epsilon = 99\%$ **(c)** $\epsilon = 99.9\%$ **(d)** $\epsilon = 99.99\%$

Figure 3.11: POD and DEIM solutions for the $k = 0.0250$ testing configurations presented as C_M versus C_L plots.

Figure 3.12. The DEIM models were constructed from 50 residual basis vectors and 10000 interpolation points. The snapshots for the residual basis were obtained by computing the residual on the projection of the FOM snapshots with the state bases. To obtain more residual snapshots, interpolated states between the projected FOM solutions were used for computing residuals (*i.e.*, after each state was projected with the state basis, interpolations of their POD coefficients were made). An additional state for computing residual snapshots was obtained by perturbing to each coefficient of the state projection with a random weight, according to

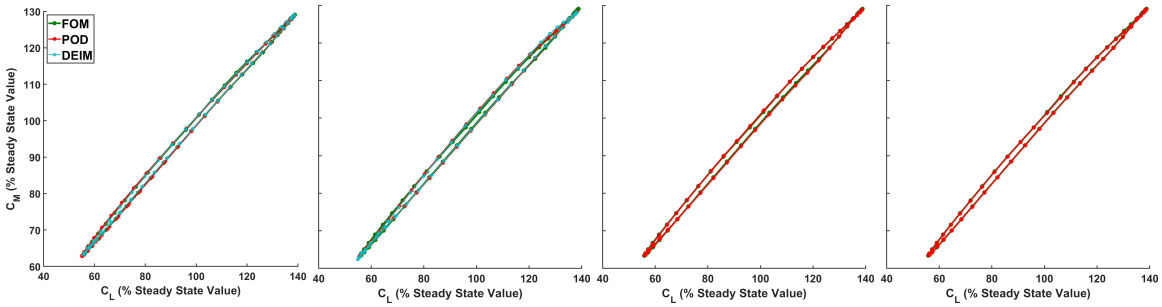
$$x_{i,\text{new}} = x_{i,\text{snapshot}} \left(1 + r\right), \tag{3.66}$$

$$r \in [-0.05, 0.05]. \tag{3.67}$$

Generally speaking, the ROMs improve in accuracy as the energy in the state basis increases. The final time lift and moment errors, as seen in Figure 3.13, in the DEIM models were on par with those in the POD models for the same state basis configuration despite the approximated residual. As mentioned before, increasing state basis size can lead to instability in the ROM. These instabilities are exacerbated in DEIM models with the reduced resolution of the residual. Consequentially, DEIM solutions could not be made for the low frequency and high frequency models with $\epsilon = 99.99\%$ and for the high frequency model with $\epsilon = 99.9\%$.

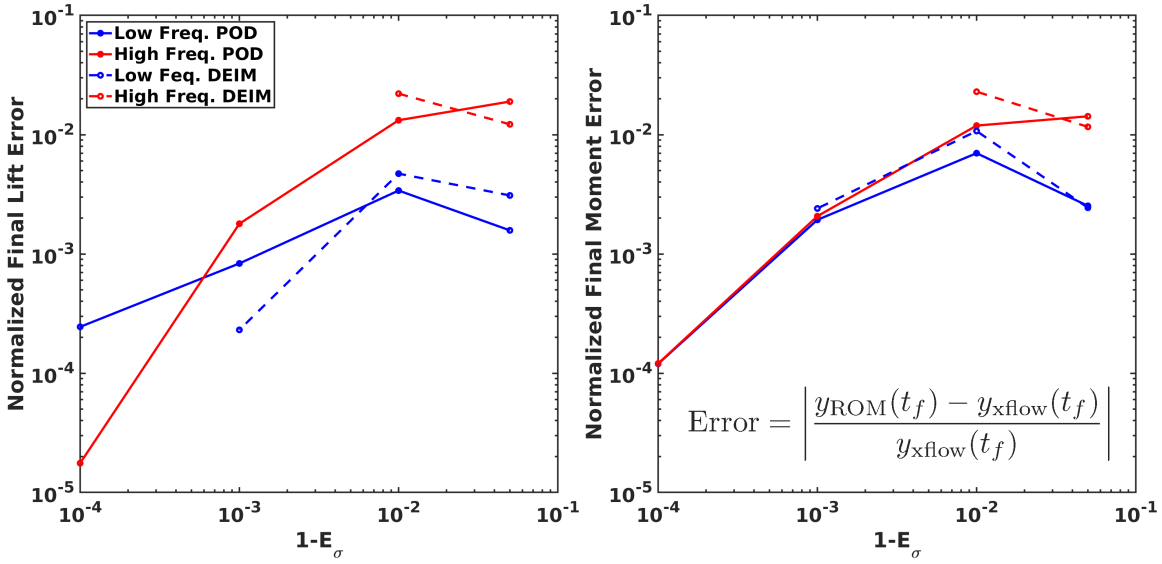
3.5.2.5 Error Estimates

The error estimates of the final-time lift load can be seen in Figure 3.14, and the configurations for these models and the fine-spaced size used for error estimation are summarized in Table 3.2. Overall, the error estimation technique described earlier in this deliverable has good agreement with the true error between the ROM solutions and the FOM; however,



(a) $\epsilon = 95\%$ **(b)** $\epsilon = 99\%$ **(c)** $\epsilon = 99.9\%$ **(d)** $\epsilon = 99.99\%$

Figure 3.12: POD and DEIM solutions for the $k = 0.10$ testing configurations presented as C_M versus C_L plots.



(a) Final Lift Error **(b)** Final Moment Error

Figure 3.13: Normalized final load errors of ROM models compared to FOM.

ROM	POD Basis Size	Residual Basis Size	Number of Interpolation Points
Low-frequency POD	2,3,7 (15)	N/A	N/A
High-frequency POD	3,7,26 (51)	N/A	N/A
Low-frequency DEIM	2,3,7 (15)	50(100)	10000(20000)
High-frequency DEIM	3,7 (51)	50(100)	10000(20000)

Table 3.2: ROM configurations for solving and error estimation. The values in parentheses represent the fine-space values of the parameters used in error estimation.

instability and inaccuracy occasionally made obtaining the error estimates more difficult.

Two key factors affecting the stability of the error estimation: the stability of the fine-space ROM configuration when used for generating solutions and the size of the coarse-space ROM. For example, for the high-frequency POD model, a fine-space of 51 state basis vectors ($\epsilon = 99.99\%$) was sufficient to generate adjoints but blew up for the high-frequency DEIM model; likewise, the high-frequency DEIM model blew up when using a state basis of 26 vectors ($\epsilon = 99.9\%$) to compute adjoints. Both configurations were also unstable for generating DEIM solutions, and in order to generate stable adjoints for the high-frequency DEIM model the state basis size was reduced to 20 basis vectors.

The low-frequency DEIM model also was unable to use the largest state basis ($\epsilon = 99.99\%$) for its fine space when performing error estimation. Additionally, the size of the coarse space affected the stability of the adjoint computation for the low-frequency DEIM model. When the fine-space state basis was 15 vectors ($\epsilon = 99.99\%$), the 7 state basis vector DEIM model ($\epsilon = 99.9\%$) was able to generate stable adjoints whereas the adjoints for the 3 and the 2 basis vector DEIM models ($\epsilon = 99/95\%$) blew up. However, one benefit to reducing the fine-space state basis to 7 was the demonstration of the error estimation for DEIM, where only the residual basis size and the number of interpolation points increase in the fine space and the state basis remains fixed. As with the other configurations, this error estimate is still fairly good, which shows potential benefits to residual basis and interpolation point refinement for increased accuracy.

The low-frequency POD model had the poorest error estimates. This may be due to some of the state basis vectors in the fine space adding instability to the basis. This may also have caused the low-frequency DEIM model being unable to generate stable adjoints when using a fine space of 15 state basis vectors.

3.5.2.6 Adaptation Results

The 95% POD and DEIM solutions were adapted using state basis and nonlinear basis error localization of their final lift error estimates. The highest error contributing state basis

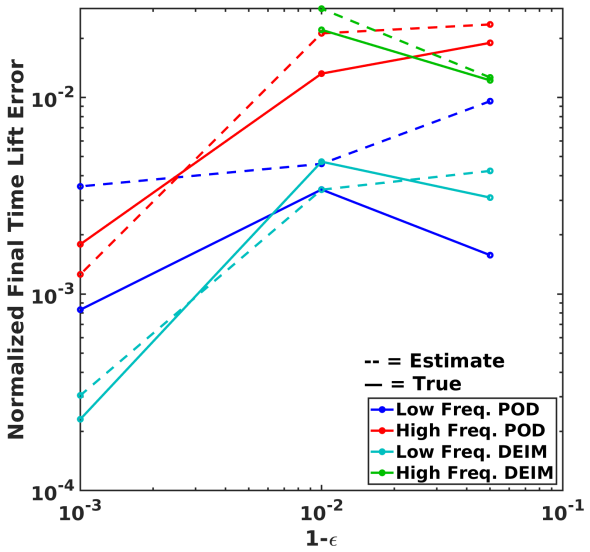
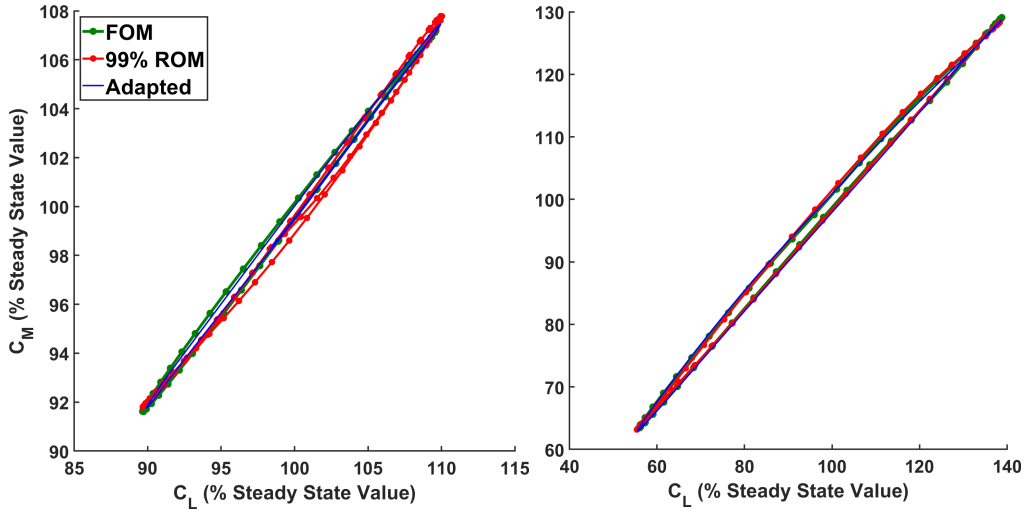


Figure 3.14: Error estimation results. The "true" value is the FOM final-time lift value. The true error and the estimated error are normalized with the "true" final lift value (*i.e.*, $\left|\frac{\text{Error}}{C_L(t_f)}\right|$).

vectors were added such that the adapted state bases ranks equalled the ranks of the 99% state bases. For DEIM. the residual basis rank was increased from 50 to 75 and 5000 additional sampling indices were added with the same criterion. The problem was then resolved with the adapted models. Overall the output trajectories of the adapted models were more accurate than ROM models with 99% of the POD singular value energy. A comparison of these models can be see in Figure 3.15. This demonstrates that singular value energy may not be the only criterion for selecting basis vectors. This is especially prominent for the output predictions of the POD and DEIM solutions on the low-frequency testing configurations. The quality of the output prediction decreases when the singular value energy increases from 95% to 99%; however, selecting basis vectors to take into account their effects on output predictions maintains quality output predictions for relatively coarse ROM systems. This is important for configurations when increasing the size of the ROM creates instability in the ROM solutions. For this problem, the ROMs are able to produce decent output trajectories before the ROM size affects their stability. The next example application demonstrates how adjoint-weighted residual error estimates can be useful when a ROM becomes more unstable with larger sized state bases.



(a) Low-Frequency POD

(b) High-Frequency POD

(c) Low-Frequency DEIM

(d) High-Frequency DEIM

Figure 3.15: Comparison of C_M versus C_L trajectories for the 99% singular value energy models and models with the same rank but obtained from adaptation of the 95% singular value energy models.

3.5.3 Pitching and Plunging Airfoil with Compressible Navier-Stokes

A pitching and plunging NACA 0012 airfoil is the last example demonstrating the use of adjoint-weighted residual error estimates and adaptation of POD and DEIM-ROMs. The mesh consists of 728 quadrilateral elements with linear states represented by a first-order Lagrangian polynomial basis. The freestream flow has a Reynolds number of $Re = 500,000$ and zero angle of attack. The fluid system is modeled with compressible URANS physics, using the Spalart-Allmaras turbulence model [114]. For this exercise,

"seconds" refers to the nondimensional, simulation time. This is defined by

$$\tilde{t} = t\frac{c}{\left|\vec{V}\right|} \tag{3.68}$$

where c is the chord length of the airfoil and $\left|\vec{V}\right|$ is the freestream flow speed. Beginning from a steady-state solution, forced pitching and plunging is imposed. The pitching and plunging amplitude trajectory is defined by

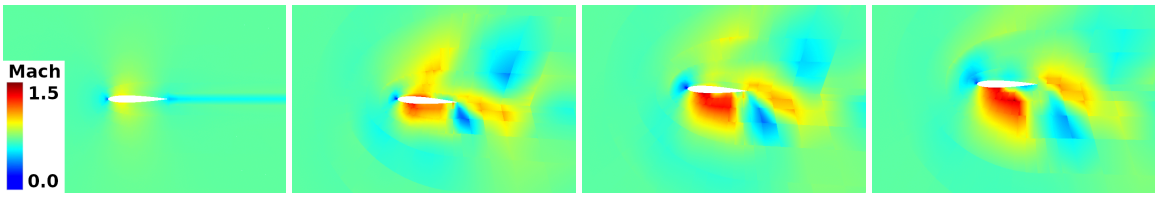
$$\alpha(t) = [3^\circ \sin(2kt)]\left[1 - e^{-\frac{t}{T}^3}\right] \tag{3.69}$$

$$h(t) = \underbrace{\left[0.25c\sin(2kt + \frac{\pi}{2})\right]}_{\text{sinusoidal}}\underbrace{\left[1 - e^{-\frac{t}{T}^3}\right]}_{\text{smooth ramp}}, \tag{3.70}$$

where c is the chord of the airfoil, k is the reduced frequency of the motion, and $T = 3$sec is the time constant of the exponential ramping. The freestream Mach number and the coupled pitching and plunging reduced frequency are sampled to generate a snapshot set for a ROM that can generate solutions to different Mach and reduced frequency configurations. Four different Mach and reduced frequency configurations are used for the training signals, and for each configuration, 10 seconds worth of simulation time data are generated with 1000 time intervals. Only the initial 6 seconds of each training signal data are used, and a global ROM is generated from the resulting snapshot set of 2404 samples with the mean of the snapshot set used as a reference state. In addition, the mean of the training configurations is used as the conditions for the testing set. Beginning with the initial steady-state solution to the testing configuration, the ROMs are used to solve 10 seconds of simulation time. Figure 3.16 displays Mach number distributions about the airfoil for different moments in time for the testing solution. Table 3.3 summarizes all of the training and testing configurations, and Figure 3.17 shows the coefficient of lift trajectory for each of the training and testing configurations.

Case	Mach	k
Training 1	0.6000	1.3805
Training 2	0.6733	3.1416
Training 3	0.7394	0.5000
Training 4	0.8000	2.2611
Testing	**0.7032**	**1.8208**

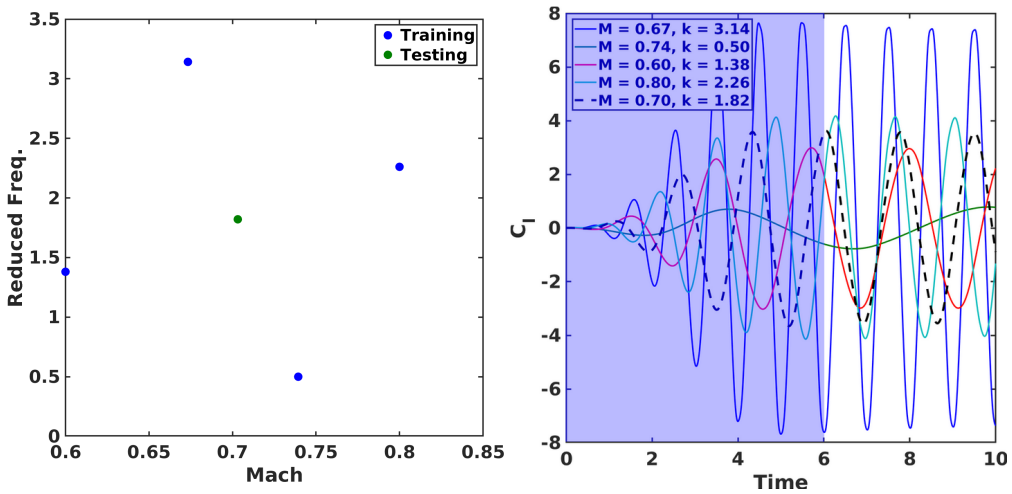
Table 3.3: Training and testing configurations for a NACA 0012 airfoil undergoing forced mesh motion with compressible Navier-Stokes physics.



(a) $t = 0s$ **(b)** $t = 5.17s$ **(c)** $t = 5.38s$ **(d)** $t = 5.60s$

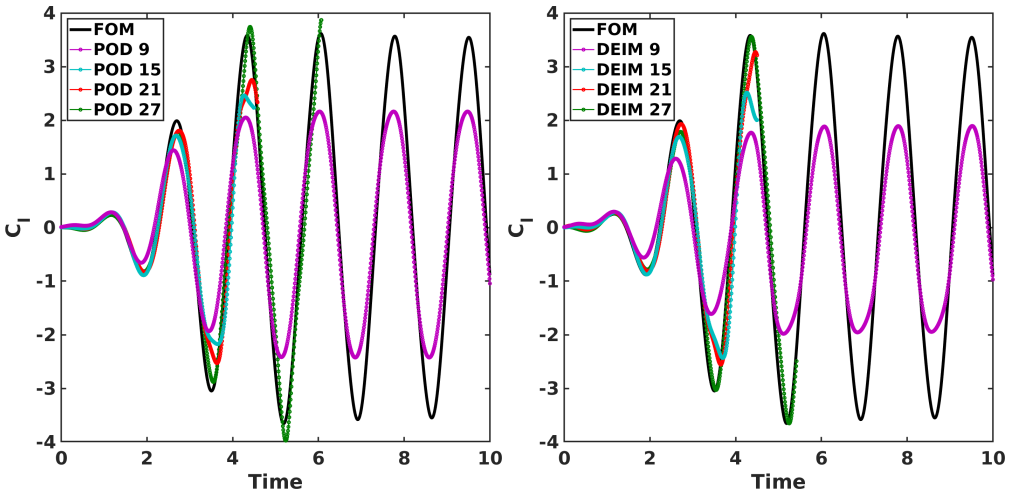
(e) $t = 5.81s$ **(f)** $t = 6.03s$ **(g)** $t = 6.25s$ **(h)** $t = 6.46s$

Figure 3.16: Solution trajectory for the testing case.



(a) Configurations **(b)** Lift

Figure 3.17: A plot of the training and testing configurations and a plot of the lift trajectories for their solutions. Only the first 6 seconds are used for training, as indicated by the shading.



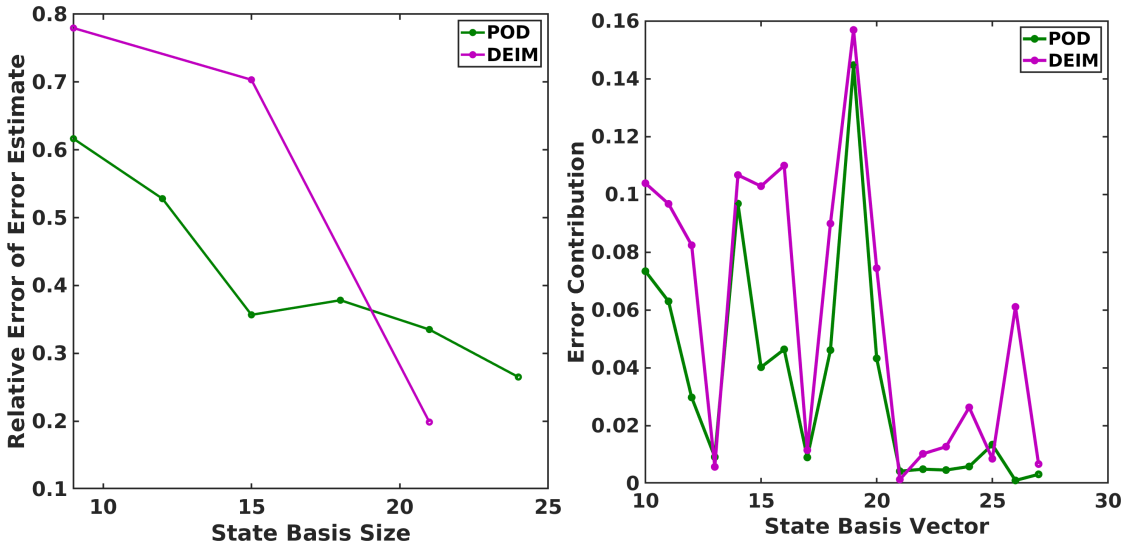
(a) POD **(b)** DEIM

Figure 3.18: Solutions from global POD and DEIM models for the testing configuration.

3.5.3.1 ROM Solutions

The number of basis vectors needed to obtain a state basis that captures 99%, 99.5%, 99.9% and 99.99% singular value energy is 9, 15, 21, and 27, respectively. Residual snapshots were obtained for each state basis by computing the residual for the projected state snapshots and for perturbed state snapshots. The resulting DEIM models all contain 120 residual basis vectors and 200 sampling elements (27.5% of the total dofs). Figure 3.18 shows the solutions for each of the POD and DEIM models. The performance of the ROMs is poor. Coarse POD and DEIM solutions may be stable, but they are inaccurate. Conversely, the fine POD and DEIM solutions begin with higher accuracy, but once the smooth ramping finishes, they quickly begin to have nonphysical errors, which result in the solution becoming unstable. This demonstrates a robustness issue with ROMs for multi-parameter, nonlinear systems. The solutions for this problem are nonlinear with respect to the inputs: increasing the Mach number will inevitably result in the appearance of a discontinuity in the flow, while the reduced frequency dictates which of the horizontal or vertical components of the flow will dominate the solutions. As mentioned before, global ROMs will perform poorly for this type of problem as the state basis introduces irrelevant flow characteristics to problems that need to be dissipated effectively.

However, error estimation on the coarsest ROMs reveals the basis vectors that influence the output error the most. An error estimate for the first peak lift value of each ROM is computed. The relative error of those error estimates and error localization for POD and DEIM for the coarsest ROMs are shown in Figure 3.19. The 4 largest error-contributing basis vectors were added to the state basis of the coarsest POD-ROM and DEIM-ROM; additionally,



(a) Error in error Estimate

(b) Error Localization

Figure 3.19: Error of error estimate for increasing basis sizes and error localization for the state basis of the POD and DEIM ROMs.

800 sampling indices were added to the sampling matrix of the coarsest DEIM-ROM.

The purpose of this problem is to demonstrate the ability of error estimation and adaptation to overcome robustness issues experienced by multi-parameter ROMs. A basis with 99% singular value energy is sufficient to span the snapshot space fairly well, and additional basis vectors do not contribute much to the projection error of the state basis. Thus, a takeaway from this exercise is that a larger number of singular vectors can be chosen by prioritizing output prediction over singular value energy. In doing so, the state basis and ROM system can remain coarse, and hence stable, but accurate for the output of interest.

3.6 Conclusion

This chapter derives and demonstrates a novel approach to POD and DEIM error quantification through adjoint-weighted residual error estimation. The error estimates are obtained from solving the reduced-order versions of the full-order adjoint equations. In full-order adjoint-based error estimation, the solutions from a coarse resolution model are injected into a fine-resolution model. The injection inherently generates nonzero residuals that can be used with adjoints to estimate the error between the coarse and fine space models without needing to solve the fine-space models. Instead of mesh size or FEM state approximation order, the reduced-order adjoint system considers the ranks of the POD state basis, the DEIM residual basis, and the DEIM sampling matrix for enrichment and adaptation.

Three example applications are shown in this chapter. The first is a steady, scalar advection-diffusion problem. The residual and output of this example are linear with re-

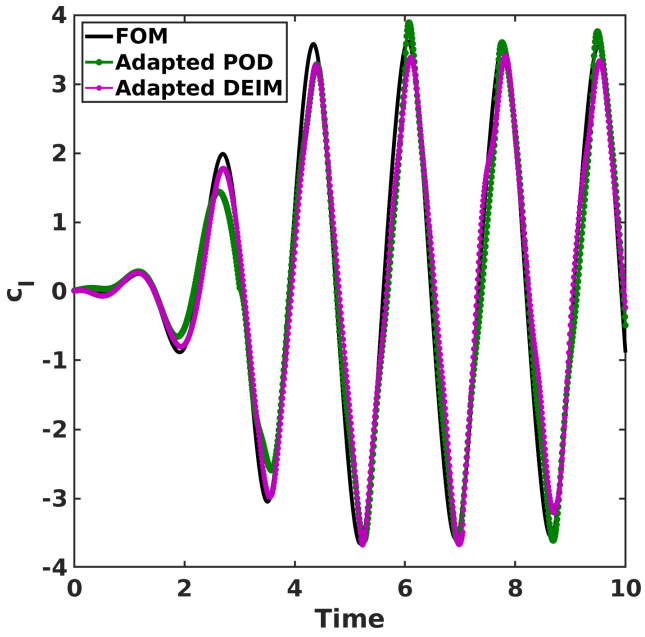


Figure 3.20: Adapted ROM solutions.

spect to the state. Because the adjoint-weighted residual error estimates are built on linear approximations, the expected and observed error estimates are exact. The second example application is meant to demonstrate the application of adjoint-weighted residual error estimates and adaptation on a large, 3D problem with nonlinear physics. After the output adjoints for the reduced-order models are obtained, error localization can be used to determine the relative error contributions among fine-space degrees of freedom. Those degrees of freedom with the largest error contributions are added to the coarse model.

This adaptation procedure prioritizes state and residual basis vectors based on the error in an output of interest and not purely singular value energy content. Since the POD procedure minimizes projection error, a relatively well spanning basis may not be able to produce good output predictions, as the reduced system might not be able to drive the reduced solution to the desired solution. In addition, it is possible that the basis vectors that are most crucial to the output predictions have mode numbers that are too large for the ROM system to remain stable, while including all larger singular value energy basis vectors before it. This was shown with the third example application, where the trade-off between stability and accuracy prevented any stable ROM from being accurate and any potentially accurate ROM from being stable. This was despite the coarse yet stable ROMs being able to span the solution of the testing configuration. In order to keep the ROM system stable and coarse but accurate for output predictions, adjoint-weighted error estimation with error localization is able to select and add the basis vectors that are the most important to the output predictions.

As mentioned before, after a majority of the singular value energy has been included in the state and residual basis, additional basis vectors do not significantly improve the projection error of the model. This means that solutions to the ROM do not improve due to the additional spanning quality of the ROM, but instead due to how the adapted basis can push the state towards the desired solution. This adaptation technique improves how the ROM projects the system itself (*i.e.*, enrichment of the test space). In fact, the adjoint weights themselves are weights on the test space and not the trial space. The next chapter explores a method for modifying the test space in a Petrov-Galerkin ROM to drive the coarse basis towards approximating the desired solution rather than refining all spaces to retain a Galerkin formulation.

CHAPTER 4

The State Adjoint Petrov-Galerkin ROM

The examples in Chapter 3 demonstrate a phenomenon that is common with projection-based ROMs that use the same basis for the low-rank state approximation and the system projection: although the basis is capable of high-fidelity state representation, the projection of the system alone is unable to yield the dynamics that obtain high-fidelity solutions online. This was ameliorated with basis adaptation driven by *a posteriori* output-based error weights. Although yielding improved output predictions, the method in Chapter 3 still makes the poor assumption that the state basis is able to accurately project the system. Rather, using a Petrov-Galerkin approach, where the state and system projections are allowed to differ, may yield more accurate and stable solutions. This chapter formulates a novel approach for Petrov-Galerkin ROMs by using reduced-state adjoints to form the system projection online. Following the derivation of the method, analysis of its stability, computational complexity, and convergence properties is presented; applications to unsteady problems are demonstrated; and comparisons to a commonly used Petrov-Galerkin method is made.

4.1 Petrov-Galerkin Methods

Galerkin-ROMs (GROMs) are commonly used and have been shown to be successful in several engineering applications [115, 116, 53]. They are also the formulation for several other model reduction techniques, such as the continuous and the discrete empirical interpolation methods [117, 118] and the missing point estimation technique [27]. However, GROMs can be susceptible to instability and inaccuracy. GROMs typically have no *a priori* stability guarantees [53, 52]. Additionally, numerical error in the construction of POD bases can also be problematic for stability as the numerical error is typically non-physical in nature [119]. Further, by enforcing that the residual only be resolved in the space of the reduced state, GROMs can have arbitrarily large residual values in the orthogonal space of the POD basis [55]. These unresolved residuals can create erroneous solutions, which may

accumulate and cause instability. These are among the reasons Galerkin projection is not seen as a viable approach to model reduction for many nonlinear systems. [54].

Because of the accuracy and stability issues present in pure Galerkin ROMs, great interest is placed in studying Petrov-Galerkin ROMs. A common approach to constructing Petrov-Galerkin ROMs, known as the least-squares Petrov-Galerkin method (LSPG), is to formulate the test basis vectors such that the ROM residual minimization also minimizes the L_2 norm of the full-order residual [56, 55]. This method has been shown to be beneficial over GPOD for a variety of problems [120, 121, 56] and is an underlying framework for the Gauss-Newton with Approximated Tensors method [29]. However, the benefits of LSPG are limited to implicit schemes and are sensitive to time step size. Another approach to stabilization is through closure modeling. ROM closure attempts to model the effects of the truncated state basis vectors on the modes that are used for the ROM, similar to concepts in turbulence modeling where the solution is divided in to resolved (POD basis) and modeled (kernal of POD basis) scales. Parish, Wentland, and Duraisamy developed a closure Petrov-Galerkin, called the Adjoint Petrov-Galerkin method (APG), test basis that arises from the Mori-Zwanzig formulation [59]. The Mori-Zwanzig formulation attempts to model the influence of truncated modes through time with a integrated memory term [63]; however, this memory term is typically intractably expensive to evaluate, and the Petrov-Galerkin test basis arises through its approximation.

The underlying issue of GROMs is not that they are unable to represent the solution space accurately. This is evident as the singular value energy criterion from Equation (2.13) will generate a basis that well-spans the solution snapshots, and if the solution snapshots are a well-representative sample of the solution space, then one should expect that the POD state basis is sufficient at projecting any state in the solution space with high accuracy. Rather, instances where GROMs fail to obtain an accurate or stable solution are attributable to the failure of Galerkin projection to properly obtain the correct dynamics to drive the state towards an optimal solution. Ultimately, the choice of the test space dictates the dynamics that are retained from the system projection. Instead of targeting the L2 minimization of the residual or the closure of the ROM model, a test space can be derived that yields dynamics of the system that solves for the minimization of the **state error** directly.

Drawing inspiration from the discontinuous Petrov-Galerkin (DPG) work of Demkowicz and Gopalakrishnan [64, 122] and the optimal test function DPG work of Kast and Fidkowski [65, 66], this chapter introduces a novel method for formulating test basis vectors for Petrov-Galerkin ROM systems, which are designed to optimally predict the state. These optimal test basis vectors are formulated by solving an adjoint-like system and transform the residual error minimization problem to be equivalent to a least-squares state error

minimization problem. It is shown that this basis is the negative reduced state adjoints of the FOM, and thus this method is referred to as the State-Adjoint Petrov-Galerkin (SAPG) method for this thesis. Example applications on linear problems demonstrate that using the SAPG test basis in a Petrov-Galerkin ROM results in ROM solutions that **exactly** equal the full-order solutions projected with the state basis; however, the cost of constructing the SAPG test basis is on the order of the FOM system. Thus, to decrease the cost of computing the test basis vectors online, an approximate method of computing the optimal basis vector is derived. In addition, the formulation of the SAPG method in a hyper-reduced setting is presented where a Petrov-Galerkin DEIM model is constructed with the SAPG test basis. Finally, the stability of the SAPG method and its effects on convergence are examined.

4.1.1 Least-squares Petrov-Galerkin Test Space

The LSPG test space arises from transforming the minimization of the L_2 norm of the reduced residual such that it is equivalent to the full-order residual minimization on the state constrained to the reduced space.

The discrete general model defined in Section 2.1 is the starting point of this derivation, defined by

$$\boldsymbol{M}\frac{d\boldsymbol{x}}{dt} + \boldsymbol{R}(\boldsymbol{x}(t), \boldsymbol{\mu}(t), t) = \boldsymbol{0}, \tag{2.1}$$

$$\bar{\boldsymbol{R}}(\boldsymbol{x}, \boldsymbol{\mu}, t) = \boldsymbol{0}. \tag{2.2}$$

It is useful to consider the linearization of Equation (2.2). Nonlinear residuals are dominated by linear dynamics near their minimum, and the direction of the linearization is often used to update the state when a system is solved iteratively. This linearization arises from a Taylor series of the system about some reference state $\boldsymbol{x}_0$, which minimizes the residual,

$$\bar{\boldsymbol{R}}(\boldsymbol{x}, \boldsymbol{\mu}, t) = \bar{\boldsymbol{R}}(\boldsymbol{x}_0, \boldsymbol{\mu}, t) + \frac{\partial \bar{\boldsymbol{R}}}{\partial \boldsymbol{x}}\left[\boldsymbol{x} - \boldsymbol{x}_0\right] + \underbrace{\cancel{\mathcal{O}(\Delta \boldsymbol{x}^2)}}_{\text{for small } \Delta\boldsymbol{x}} = \boldsymbol{0},$$

$$\bar{\boldsymbol{R}}(\boldsymbol{x}, \boldsymbol{\mu}, t) = \frac{\partial \bar{\boldsymbol{R}}}{\partial \boldsymbol{x}}\boldsymbol{x} - \left[\underbrace{\frac{\partial \bar{\boldsymbol{R}}}{\partial \boldsymbol{x}}\boldsymbol{x}_0 - \bar{\boldsymbol{R}}(\boldsymbol{x}_0, \boldsymbol{\mu}, t)}_{\boldsymbol{b}}\right],$$

$$\bar{\boldsymbol{R}}(\boldsymbol{x}, \boldsymbol{\mu}, t) = \left[c\boldsymbol{M} + \boldsymbol{A}\right]\boldsymbol{x} - \boldsymbol{b} = \boldsymbol{0}. \tag{4.1}$$

where $\mathcal{O}(\Delta \boldsymbol{x}^2)$ are the higher order terms of the Taylor series that can be neglected for small $\Delta\boldsymbol{x}$, c is some constant based on the temporal discretization and time step, $\boldsymbol{b}$ are the constant constraints of the system that arise from the linearization, and $\boldsymbol{A}$ is the spatial

Jacobian of the system. The reduced form of Equation (4.1) can be constructed by substituting the reduced representation of the state from Equation (2.3) and projecting the entire system via a left-multiplication by the test basis, resulting in

$$\hat{\tilde{\boldsymbol{R}}}(\hat{\boldsymbol{x}}) = \boldsymbol{W}^T \left[c\boldsymbol{M} + \boldsymbol{A}\right] \boldsymbol{V}\hat{\boldsymbol{x}} - \boldsymbol{W}^T\boldsymbol{b} = \boldsymbol{0}, \tag{4.2}$$

where $\boldsymbol{V}$ is a set of orthonormal basis vectors.

The minimum full-order residual of the state constrained by the state basis can be defined in an L_2 sense according to

$$\hat{\boldsymbol{x}} = \arg\min_{\hat{z}} ||\boldsymbol{R}(\boldsymbol{V}\hat{\boldsymbol{z}})||_2^2 = \arg\min_{\hat{z}} || \left[c\boldsymbol{M} + \boldsymbol{A}\right] \boldsymbol{V}\hat{\boldsymbol{z}} - \boldsymbol{b}||_2^2, \tag{4.3}$$

which is equivalent to

$$\hat{\boldsymbol{x}} = \arg\min_{\hat{z}} \left[\left[c\boldsymbol{M} + \boldsymbol{A}\right] \boldsymbol{V}\hat{\boldsymbol{z}}\right]^T \left[c\boldsymbol{M} + \boldsymbol{A}\right] \boldsymbol{V}\hat{\boldsymbol{z}} - 2\left[\left[c\boldsymbol{M} + \boldsymbol{A}\right] \boldsymbol{V}\hat{\boldsymbol{z}}\right]^T \boldsymbol{b} + \boldsymbol{b}^T\boldsymbol{b}. \tag{4.4}$$

This minimum can be found by setting the gradient of the above statement with respect to $\hat{\boldsymbol{x}}$ equal to zero, *i.e.*,

$$\nabla_{\boldsymbol{x}} \left[\left[\left[c\boldsymbol{M} + \boldsymbol{A}\right] \boldsymbol{V}\hat{\boldsymbol{x}}\right]^T \left[c\boldsymbol{M} + \boldsymbol{A}\right] \boldsymbol{V}\hat{\boldsymbol{x}} - 2\left[\left[c\boldsymbol{M} + \boldsymbol{A}\right] \boldsymbol{V}\hat{\boldsymbol{z}}\right]^T \boldsymbol{b} + \boldsymbol{b}^T\boldsymbol{b}\right] = \boldsymbol{0}. \tag{4.5}$$

This gradient reduces to

$$\left[\left[c\boldsymbol{M} + \boldsymbol{A}\right] \boldsymbol{V}\right]^T \left[c\boldsymbol{M} + \boldsymbol{A}\right] \boldsymbol{V}\hat{\boldsymbol{x}} - \left[\left[c\boldsymbol{M} + \boldsymbol{A}\right] \boldsymbol{V}\hat{\boldsymbol{x}}\right]^T \boldsymbol{b} = \boldsymbol{0}. \tag{4.6}$$

The LSPG test basis is then chosen as

$$\boldsymbol{W} = \left[c\boldsymbol{M} + \boldsymbol{A}\right] \boldsymbol{V}, \tag{4.7}$$

which makes Equation (4.6) equivalent to Equation (4.2). Thus, the minimum residual problem is equivalent to a Petrov-Galerkin POD formulation with the LSPG test basis [55].

4.1.2 Test Basis Construction for Optimal State Prediction

Let $\boldsymbol{x}_{\text{FOM}} \in \mathbb{R}^{n_{\boldsymbol{x}} \times 1}$ be the solution derived from Equation (2.7), and let $\hat{\boldsymbol{x}}_{\text{ROM}} \in \mathbb{R}^{n_{\boldsymbol{x}} \times 1}$ be the solution derived from the reduced problem Equation (4.2). Ultimately, the minimization of the error between $\boldsymbol{x}_{\text{FOM}}$ and $\hat{\boldsymbol{x}}_{\text{ROM}}$ is desired. Thus, rather than seeking a residual minimization, one can seek the state error minimization. If this minimization is defined in

an L_2 sense, then the error of the POD-ROM solution is

$$e_{\hat{\boldsymbol{x}}_{\text{ROM}}} = ||\boldsymbol{V}\hat{\boldsymbol{x}}_{\text{ROM}} - \boldsymbol{x}_{\text{FOM}}||^2. \tag{4.8}$$

which is equal to

$$e_{\hat{\boldsymbol{x}}_{\text{ROM}}} = \hat{\boldsymbol{x}}_{\text{ROM}}^T\boldsymbol{V}^T\boldsymbol{V}\hat{\boldsymbol{x}}_{\text{ROM}} - 2\hat{\boldsymbol{x}}_{\text{ROM}}^T\boldsymbol{V}^T\boldsymbol{x}_{\text{FOM}} + \boldsymbol{x}_{\text{FOM}}^T\boldsymbol{x}_{\text{FOM}}, \tag{4.9}$$

$\boldsymbol{V}^T\boldsymbol{V}$ is the identity, as $\boldsymbol{V}$ is an orthonormal basis. The minimization of this error can be found by setting the gradient of the error with respect to $\hat{\boldsymbol{x}}_{\text{ROM}}$ equal to zero, *i.e.*,

$$\frac{\partial e_{\hat{\boldsymbol{x}}_{\text{ROM}}}}{\partial \hat{\boldsymbol{x}}_{\text{ROM}}} = 2\hat{\boldsymbol{x}}_{\text{ROM}}^T - 2\boldsymbol{x}_{\text{FOM}}^T\boldsymbol{V} = \boldsymbol{0}^T, \tag{4.10}$$

$$\Rightarrow \left[\hat{\boldsymbol{x}}_{\text{ROM}} - \boldsymbol{V}^T\boldsymbol{x}_{\text{FOM}}\right] = \boldsymbol{0}. \tag{4.11}$$

Thus, the closest that $\hat{\boldsymbol{x}}_{\text{ROM}}$ can be to $\boldsymbol{x}_{\text{FOM}}$ is the projection of $\boldsymbol{x}_{\text{FOM}}$ onto the space spanned by the vectors in $\boldsymbol{V}$. One can consider $\boldsymbol{V}^T\boldsymbol{x}_{\text{FOM}}$ to be the *ideal* solution ($\hat{\boldsymbol{x}}_{\text{Ideal}}$) of a POD model. Here, *ideal* is understood in the sense that it is the most faithful replication of the full-order model that is possible, constrained by the information contained in the basis $\boldsymbol{V}$. This is an unsurprising outcome, but it does allow one to consider formulating the POD-ROM in a way such that minimizing the residual of the reduced system would be equivalent to minimizing the error between $\hat{\boldsymbol{x}}_{\text{ROM}}$ and $\hat{\boldsymbol{x}}_{\text{Ideal}}$.

If the goal of the model reduction is to satisfy Equation (4.11) and Equation (4.2) for $\hat{\boldsymbol{x}}_{\text{ROM}}$, then $\boldsymbol{V}^T\boldsymbol{x}_{\text{FOM}}$ must also satisfy Equation (4.2). Thus the error in the residual defined in Equation (4.2) when the input is $\hat{\boldsymbol{x}}_{\text{ROM}}$ and when the input is $\boldsymbol{V}^T\boldsymbol{x}_{\text{FOM}}$ must also equal zero.

$$\begin{aligned} e_{\hat{\bar{\boldsymbol{R}}}} &= \boldsymbol{W}^T\bar{\boldsymbol{R}}(\boldsymbol{V}\hat{\boldsymbol{x}}_{\text{ROM}}) - \boldsymbol{W}^T\bar{\boldsymbol{R}}(\boldsymbol{V}\boldsymbol{V}^T\boldsymbol{x}_{\text{FOM}}), \\ &= \left[\boldsymbol{W}^T\left[c\boldsymbol{M} + \boldsymbol{A}\right]\boldsymbol{V}\hat{\boldsymbol{x}}_{\text{ROM}} - \boldsymbol{W}^T\boldsymbol{b}\right] - \left[\boldsymbol{W}^T\left[c\boldsymbol{M} + \boldsymbol{A}\right]\boldsymbol{V}\boldsymbol{V}^T\boldsymbol{x}_{\text{FOM}} - \boldsymbol{W}^T\boldsymbol{b}\right], \\ &= \boldsymbol{W}^T\left[c\boldsymbol{M} + \boldsymbol{A}\right]\boldsymbol{V}\left[\hat{\boldsymbol{x}}_{\text{ROM}} - \boldsymbol{V}^T\boldsymbol{x}_{\text{FOM}}\right]. \end{aligned} \tag{4.12}$$

If the test basis is chosen such that $\boldsymbol{W} = [c\boldsymbol{M} + \boldsymbol{A}]^{-T} \boldsymbol{V}$, then

$$\begin{aligned} e_{\hat{\bar{\boldsymbol{R}}}} &= \left[[c\boldsymbol{M} + \boldsymbol{A}]^{-T} \boldsymbol{V} \right]^T [c\boldsymbol{M} + \boldsymbol{A}] \boldsymbol{V} \left(\hat{\boldsymbol{x}}_{\text{ROM}} - \boldsymbol{V}^T \boldsymbol{x}_{\text{FOM}} \right), \\ &= \boldsymbol{V}^T [c\boldsymbol{M} + \boldsymbol{A}]^{-1} [c\boldsymbol{M} + \boldsymbol{A}] \boldsymbol{V} \left(\hat{\boldsymbol{x}}_{\text{ROM}} - \boldsymbol{V}^T \boldsymbol{x}_{\text{FOM}} \right), \\ &= \boldsymbol{V}^T I_{N \times N} \boldsymbol{V} \left(\hat{\boldsymbol{x}}_{\text{ROM}} - \boldsymbol{V}^T \boldsymbol{x}_{\text{FOM}} \right), \\ &= \boldsymbol{V}^T \boldsymbol{V} \left(\hat{\boldsymbol{x}}_{\text{ROM}} - \boldsymbol{V}^T \boldsymbol{x}_{\text{FOM}} \right), \\ &= I_{n \times n} \left(\hat{\boldsymbol{x}}_{\text{ROM}} - \boldsymbol{V}^T \boldsymbol{x}_{\text{FOM}} \right), \\ &= \left(\hat{\boldsymbol{x}}_{\text{ROM}} - \boldsymbol{V}^T \boldsymbol{x}_{\text{FOM}} \right) = \boldsymbol{0}. \end{aligned} \tag{4.13}$$

This choice for the test basis therefore makes the problem of finding $\hat{\boldsymbol{x}}_{\text{ROM}}$ where $e_{\hat{\bar{\boldsymbol{R}}}} = \hat{\bar{\boldsymbol{R}}} = \boldsymbol{0}$ equivalent to Equation (4.11), and the SAPG test basis for a linear POD system is found by solving

$$[c\boldsymbol{M} + \boldsymbol{A}]^T \boldsymbol{W} = \boldsymbol{V}. \tag{4.14}$$

This definition of SAPG test basis vectors is only exact for linear PDEs. For nonlinear PDEs, the following approximation is made,

$$\left. \frac{\partial \bar{\boldsymbol{R}}}{\partial \boldsymbol{x}} \right|_{\hat{\boldsymbol{x}}^\eta}^T \boldsymbol{W} = \boldsymbol{V}, \tag{4.15}$$

where $\boldsymbol{x}^\eta$ is the current value of $\boldsymbol{x}$ in an iterative solver of the POD-ROM.

The solutions to the above equation are the negative reduced state adjoints. Considering the negative reduced state as the outputs of the system,

$$\boldsymbol{y} = -\hat{\boldsymbol{x}}, \tag{4.16}$$

and noting that $\hat{\boldsymbol{x}} = \boldsymbol{V}^T \boldsymbol{x}$, the linearization of $\boldsymbol{y}$ with respect to the state is

$$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} = \boldsymbol{V}^T. \tag{4.17}$$

Substituting this into the adjoint Equation (3.20) yields

$$\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}\right]^T \boldsymbol{\Psi} + \left[\frac{\partial y}{\partial \boldsymbol{x}}\right]^T = \boldsymbol{0}. \tag{3.20}$$

$$\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}\right]^T \boldsymbol{\Psi}_{-\hat{\boldsymbol{x}}} = \boldsymbol{V} \tag{4.18}$$

$$\Rightarrow \boldsymbol{W} = \boldsymbol{\Psi}_{-\hat{\boldsymbol{x}}}. \tag{4.19}$$

For this reason, this test space is referred to as the state adjoint Petrov-Galerkin (SAPG) test space. Before proceeding with the hyper-reduced forms of the SAPG method, it should be noted that the time-coupled adjoint solutions are not resolved here. Obtaining the reduced state adjoints for each unsteady solution would require an independent adjoint solution to be resolved backwards in time for each state basis vector. As these adjoint solutions each require different terminal conditions, reusing of solutions between adjoint solves is not possible. Thus, for a solution with N_t time nodes, $\frac{n_{\boldsymbol{V}} N_t(N_t+1)}{2}$ unsteady adjoint solutions would be required. Rather, resolving the reduced state adjoints in the fashion described above is equivalent to applying one block-Jacobi iteration to the unsteady residual block system.

4.1.3 Hyper-reduction

A state adjoint Petrov-Galerkin test space can be derived for DEIM using the same process. First, the residual error between the DEIM solution and the ideal solution is

$$\begin{aligned} e_{\hat{\boldsymbol{R}}_{\text{DEIM}}} = \boldsymbol{W}_{\text{DEIM}}^T &\left[\boldsymbol{M}\boldsymbol{V}\frac{d\hat{\boldsymbol{x}}_{\text{ROM}}}{dt} + \boldsymbol{U}\left[\boldsymbol{P}^T\boldsymbol{U}\right]^\dagger \boldsymbol{P}^T\boldsymbol{R}(\boldsymbol{V}\hat{\boldsymbol{x}}_{\text{ROM}})\right] \\ &- \boldsymbol{W}_{\text{DEIM}}^T \left[\boldsymbol{M}\boldsymbol{V}\frac{\partial \boldsymbol{V}^T\boldsymbol{x}_{\text{FOM}}}{\partial t} + \boldsymbol{U}\left[\boldsymbol{P}^T\boldsymbol{U}\right]^\dagger \boldsymbol{P}^T\boldsymbol{R}(\boldsymbol{V}\boldsymbol{V}^T\boldsymbol{x}_{\text{FOM}})\right]. \end{aligned} \tag{4.20}$$

Linearization of these terms yields

$$e_{\hat{\boldsymbol{R}}} = \boldsymbol{W}_{\text{DEIM}}^T \left[\boldsymbol{M} + \boldsymbol{U}\left[\boldsymbol{P}^T\boldsymbol{U}\right]^\dagger \boldsymbol{P}^T\boldsymbol{A}\right]\boldsymbol{V}\left(\hat{\boldsymbol{x}}_{\text{ROM}} - \boldsymbol{V}^T\boldsymbol{x}_{\text{FOM}}\right). \tag{4.21}$$

$\boldsymbol{W}_{\text{DEIM}}$ can be chosen such that the error in the residual is equivalent to the error between the ideal and DEIM solutions, according to

$$\boldsymbol{W}_{\text{DEIM}}^T = \left[\left[\boldsymbol{M} + \boldsymbol{U}\left[\boldsymbol{P}^T\boldsymbol{U}\right]^\dagger \boldsymbol{P}^T\boldsymbol{A}\right]^{-T}\boldsymbol{V}\right]^T, \tag{4.22}$$

$$\Rightarrow e_{\hat{\boldsymbol{R}}} = \hat{\boldsymbol{x}}_{\text{ROM}} - \boldsymbol{V}^T\boldsymbol{x}_{\text{FOM}}. \tag{4.23}$$

$\boldsymbol{W}_{\text{DEIM}}$ is constructed by solving

$$\left[\boldsymbol{M} + \boldsymbol{U}\left[\boldsymbol{P}^T\boldsymbol{U}\right]^\dagger \boldsymbol{P}^T\boldsymbol{A}\right]^T \boldsymbol{W}_{\text{DEIM}} = \boldsymbol{V}. \tag{4.24}$$

For nonlinear DEIM systems, the state adjoint Petrov Galerkin test space solves

$$\left[\boldsymbol{M} + \boldsymbol{U}\left[\boldsymbol{P}^T\boldsymbol{U}\right]^\dagger \boldsymbol{P}^T\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}\bigg|_{\hat{\boldsymbol{x}}^\eta}\right]^T \boldsymbol{W}_{\text{DEIM}} = \boldsymbol{V}, \tag{4.25}$$

where $\hat{\boldsymbol{x}}^\eta$ is the current value of the state during the η iteration of the ROM iterative solver. Like with conventional DEIM, the state adjoint Petrov-Galerkin DEIM model reduces the complexity of residual computations through sparse sampling and interpolation of the Jacobian. Additionally, the sparse Jacobian can be reused for the state update to further reduce the impact of the added cost of state adjoint computations.

4.1.4 Effects of SAPG on Convergence

When used in an iterative solver for a residual minimization problem, the SAPG method transforms the reduced state update into the FOM state update projected with the POD state basis vectors. As an example, the Newton method attempts to solve the residual minimization problem with state updates in a direction and magnitude (with no relaxation) that would solve the system linearized about the unresolved state. A simple implementation of the Newton method can be seen in Algorithm 3. A key takeaway from this algorithm is

Algorithm 3 The Newton Method

Given a guess $\boldsymbol{x}_g$ to the problem $\bar{\boldsymbol{R}}(\boldsymbol{x}) = \boldsymbol{0}$
while $\left|\bar{\boldsymbol{R}}(\boldsymbol{x}_g)\right| >$ TOL **do**
  Solve $\frac{\partial \bar{\boldsymbol{R}}}{\partial \boldsymbol{x}}\Big|_{\boldsymbol{x}_g} \Delta\boldsymbol{x} + \bar{\boldsymbol{R}}(\boldsymbol{x}_g) = \boldsymbol{0}$ for $\Delta\boldsymbol{x}$
  Update guess: $\boldsymbol{x}_g \leftarrow \boldsymbol{x}_g + \Delta\boldsymbol{x}$
end while

that the state updates take the form

$$\Delta\boldsymbol{x} = -\left[\frac{\partial \bar{\boldsymbol{R}}}{\partial \boldsymbol{x}}\bigg|_{\boldsymbol{x}_g}\right]^{-1} \bar{\boldsymbol{R}}(\boldsymbol{x}_g). \tag{4.26}$$

A similar algorithm can be defined for solving the residual minimization problem for ROMs. The reduced state update for that algorithm is

$$\Delta\hat{\boldsymbol{x}}_{\text{ROM}} = -\left[\boldsymbol{W}^T \frac{\partial \bar{\boldsymbol{R}}}{\partial \boldsymbol{x}}\bigg|_{\hat{\boldsymbol{x}}_g} \boldsymbol{V}\right]^{-1} \boldsymbol{W}^T \bar{\boldsymbol{R}}(\hat{\boldsymbol{x}}_g). \tag{4.27}$$

The specific state updates for each of the ROMs considered in this section can be obtained by substituting the corresponding value of $\boldsymbol{W}$. For a Galerkin system, the test basis equals the trial basis, and the state updates are

$$\Delta\hat{\boldsymbol{x}}_{\text{GROM}} = -\left[\boldsymbol{V}^T \frac{\partial \bar{\boldsymbol{R}}}{\partial \boldsymbol{x}}\bigg|_{\hat{\boldsymbol{x}}_g} \boldsymbol{V}\right]^{-1} \boldsymbol{V}^T \bar{\boldsymbol{R}}(\hat{\boldsymbol{x}}_g). \tag{4.28}$$

Obtaining the inverse of $\left[\boldsymbol{V}^T \frac{\partial \bar{\boldsymbol{R}}}{\partial \boldsymbol{x}}\Big|_{\hat{\boldsymbol{x}}_g} \boldsymbol{V}\right]$ has to be done numerically, and the updates to the reduced state are obtained from the minimization of the residual projected by $\boldsymbol{V}$. Taking $\boldsymbol{W} = \frac{\partial \bar{\boldsymbol{R}}}{\partial x}\boldsymbol{V}$, the LSPG state updates are

$$\Delta\hat{\boldsymbol{x}}_{\text{LSPG}} = -\left[\boldsymbol{V}^T \frac{\partial \bar{\boldsymbol{R}}}{\partial \boldsymbol{x}}\bigg|_{\hat{\boldsymbol{x}}_g}^T \frac{\partial \bar{\boldsymbol{R}}}{\partial \boldsymbol{x}}\bigg|_{\hat{\boldsymbol{x}}_g} \boldsymbol{V}\right]^{-1} \boldsymbol{V}^T \frac{\partial \bar{\boldsymbol{R}}}{\partial \boldsymbol{x}}\bigg|_{\hat{\boldsymbol{x}}_g}^T \bar{\boldsymbol{R}}(\hat{\boldsymbol{x}}_g). \tag{4.29}$$

By setting $\boldsymbol{D} = \frac{\partial \bar{\boldsymbol{R}}}{\partial x}\Big|_{\hat{\boldsymbol{x}}_g} \boldsymbol{V}$, Equation (4.29) becomes

$$\Delta\hat{\boldsymbol{x}}_{\text{LSPG}} = -\left[\boldsymbol{D}^T\boldsymbol{D}\right]^{-1} \boldsymbol{D}^T \bar{\boldsymbol{R}}(\hat{\boldsymbol{x}}_g). \tag{4.30}$$

$\left[\boldsymbol{D}^T\boldsymbol{D}\right]^{-1}\boldsymbol{D}^T$ is the left inverse of the rectangular matrix $\boldsymbol{D}$ and represents the solution to the least-squares minimization problem defined by

$$\begin{aligned} \left\|\boldsymbol{D}\Delta\hat{\boldsymbol{x}}_{\text{LSPG}} + \bar{\boldsymbol{R}}\right\|_2^2 &= 0, \\ \left\|\left[\frac{\partial \bar{\boldsymbol{R}}}{\partial \boldsymbol{x}}\bigg|_{\hat{\boldsymbol{x}}_g} \boldsymbol{V}\right]\Delta\hat{\boldsymbol{x}}_{\text{LSPG}} + \bar{\boldsymbol{R}}\right\|_2^2 &= 0. \end{aligned} \tag{4.31}$$

Finally, taking $\boldsymbol{W} = \frac{\partial \bar{\boldsymbol{R}}}{\partial \boldsymbol{x}}^{-T} \boldsymbol{V}$, the state updates for the SAPG method are

$$\begin{aligned}
\Delta \hat{\boldsymbol{x}}_{\text{SAPG}} &= -\left[\boldsymbol{V}^T \frac{\partial \bar{\boldsymbol{R}}}{\partial \boldsymbol{x}}\bigg|_{\hat{\boldsymbol{x}}_g}^{-1} \frac{\partial \bar{\boldsymbol{R}}}{\partial \boldsymbol{x}}\bigg|_{\hat{\boldsymbol{x}}_g} \boldsymbol{V} \right]^{-1} \boldsymbol{V}^T \frac{\partial \bar{\boldsymbol{R}}}{\partial \boldsymbol{x}}\bigg|_{\hat{\boldsymbol{x}}_g}^{-1} \bar{\boldsymbol{R}}(\hat{\boldsymbol{x}}_g), \\
\Delta \hat{\boldsymbol{x}}_{\text{SAPG}} &= -\left[\boldsymbol{V}^T \boldsymbol{V} \right]^{-1} \boldsymbol{V}^T \frac{\partial \bar{\boldsymbol{R}}}{\partial \boldsymbol{x}}\bigg|_{\hat{\boldsymbol{x}}_g}^{-1} \bar{\boldsymbol{R}}(\hat{\boldsymbol{x}}_g), \\
\Delta \hat{\boldsymbol{x}}_{\text{SAPG}} &= -\boldsymbol{V}^T \frac{\partial \bar{\boldsymbol{R}}}{\partial \boldsymbol{x}}\bigg|_{\hat{\boldsymbol{x}}_g}^{-1} \bar{\boldsymbol{R}}(\hat{\boldsymbol{x}}_g).
\end{aligned} \tag{4.32}$$

Furthermore, substituting Equation (4.26) into Equation (4.32) shows that the right hand side is essentially the FOM state updates but projected in the reduced space, *i.e.*,

$$\Delta \hat{\boldsymbol{x}}_{\text{SAPG}} = -\boldsymbol{V}^T \Delta \boldsymbol{x}_{\text{FOM}}. \tag{4.33}$$

We recognize that Equation (4.29) and Equation (4.32) are similar but arise from different formulations. Equation (4.29) drives the reduced state in the direction dictated by the minimization of Equation (4.31), whereas Equation (4.32) arises purely from the FOM state updates.

4.1.5 Reduced Form of SAPG Test Basis Vectors

The computational costs for generating the SAPG test space become problematic for model reduction, as each of the basis vectors is constructed with a FOM size linear system. In addition, the SAPG test basis has to be solved multiple times at each time node, as the Jacobian of the problem may change as the state solution is updated. However, there are two ways to reduce this cost. First, freezing the Jacobian and the SAPG test basis vectors will reduce the cost of computing the test functions by the fraction of the inverse of the length of the freezing. While the Jacobian may vary within a time step, the variations are likely going to be small and may have little effect on the test basis.

Another way to reduce the cost of solving for the SAPG test basis vectors is to use reduced models of Equation (4.15) and Equation (4.25). This is done by representing $\boldsymbol{W}$ as the combination of another set of basis vectors, according to

$$\boldsymbol{W} = \boldsymbol{\Phi} \boldsymbol{T}, \tag{4.34}$$

where $\boldsymbol{\Phi} \in \mathbb{R}^{n_{\boldsymbol{x}} \times n_{\boldsymbol{\Phi}}}$ is the test search space and $\boldsymbol{T} \in \mathbb{R}^{n_{\boldsymbol{\Phi}} \times n_{\boldsymbol{V}}}$ is a matrix of coefficients.

For the POD formulation, making the above substitution into Equation (4.14) results in

$$\left[c\boldsymbol{M}+\boldsymbol{A}\right]^{T}\boldsymbol{\Phi}\boldsymbol{T}=\boldsymbol{V}. \tag{4.35}$$

The degrees of freedom have been reduced from $n_{\boldsymbol{x}}$ to $n_{\boldsymbol{\Phi}}$, which is significantly smaller. However, like in formulating projection-based ROMs, the system is overdetermined as it is still of full-order size and will need to be projected to reduce the cost of solving for $\boldsymbol{T}$. Using $\boldsymbol{\Lambda}\in\mathbb{R}^{n_{\boldsymbol{x}}\times n_{\boldsymbol{\Lambda}}}$ to project the SAPG adjoint system yields

$$\boldsymbol{\Lambda}^{T}\left[c\boldsymbol{M}+\boldsymbol{A}\right]^{T}\boldsymbol{\Phi}\boldsymbol{T}=\boldsymbol{\Lambda}^{T}\boldsymbol{V}. \tag{4.36}$$

The only restriction to the choice of the projecting basis $\boldsymbol{\Lambda}$ is that it has a rank equal to or larger than the test search space. This is clear as Equation (4.36) is a set of $n_{\boldsymbol{\Lambda}}$ equations for $n_{\boldsymbol{\Phi}}$ unknowns. Furthermore, it is convenient to make the projecting basis equal to a finer set of state POD basis vectors $\boldsymbol{V}_h$ for several reasons. Firstly, construction of $\boldsymbol{V}_h$ can occur offline with the construction of $\boldsymbol{V}$. Secondly, due to orthogonality the right-hand side product of $\boldsymbol{V}_h^{T}\boldsymbol{V}$ has the straightforward form of

$$\boldsymbol{V}_h^{T}\boldsymbol{V}=\mathbb{I}_{h_{\boldsymbol{V}}\times n_{\boldsymbol{V}}}, \tag{4.37}$$

$$\mathbb{I}_{h_{\boldsymbol{V}}\times n_{\boldsymbol{V}}}=\begin{bmatrix}\mathbb{I}_{n_{\boldsymbol{V}}}\\ \boldsymbol{0}_{(h_{\boldsymbol{V}}-n_{\boldsymbol{V}})\times n_{\boldsymbol{V}}}\end{bmatrix}, \tag{4.38}$$

where $\mathbb{I}_{n_{\boldsymbol{V}}}$ is a $n_{\boldsymbol{V}}\times n_{\boldsymbol{V}}$ identity matrix and $\boldsymbol{0}_{(h_{\boldsymbol{V}}-n_{\boldsymbol{V}})\times n_{\boldsymbol{V}}}$ is a $(h_{\boldsymbol{V}}-n_{\boldsymbol{V}})\times n_{\boldsymbol{V}}$ zero matrix.

Making the substitution of Equation (4.37) into Equation (4.36) yields

$$\begin{aligned}\boldsymbol{V}_h^{T}\left[c\boldsymbol{M}+\boldsymbol{A}\right]^{T}\boldsymbol{\Phi}\boldsymbol{T}&=\mathbb{I}_{h_{\boldsymbol{V}}\times n_{\boldsymbol{V}}},\\ \Rightarrow\left[\boldsymbol{\Phi}^{T}\left[c\boldsymbol{M}+\boldsymbol{A}\right]\boldsymbol{V}_h\right]^{T}\boldsymbol{T}&=\mathbb{I}_{h_{\boldsymbol{V}}\times n_{\boldsymbol{V}}}.\end{aligned} \tag{4.39}$$

Equation (4.39) is the reduced form of Equation (4.14) and transforms the cost of computing the SAPG test basis vectors from solving $n_{\boldsymbol{V}}$ linear problems of size $n_{\boldsymbol{x}}$ to solving $n_{\boldsymbol{V}}$ linear problems of size $h_{\boldsymbol{V}}$. If $\boldsymbol{\Phi}=\boldsymbol{V}_h$, then

$$\left[\boldsymbol{V}_h^{T}\left[c\boldsymbol{M}+\boldsymbol{A}\right]\boldsymbol{V}_h\right]^{T}\boldsymbol{T}=\mathbb{I}_{n_{\boldsymbol{\Phi}}\times n_{\boldsymbol{V}}}, \tag{4.40}$$

and

$$\boldsymbol{W}=\boldsymbol{V}_h\boldsymbol{T}, \tag{4.41}$$

which is convenient as the right-hand side term is a finer space reduced Jacobian.

The reduced formulation, Equation (4.40), seeks to find SAPG test basis vectors for each trial basis vector as a linear combination of a finer set of trial basis vectors. This has some advantages. First, the cost of finding the SAPG test basis vectors is reduced. Second, $\boldsymbol{V}_h$ is produced in tandem with $\boldsymbol{V}$. However, a disadvantage will be that $\boldsymbol{W}$ will be restricted to exist in the fine projection space. If this space is not chosen appropriately, the benefits of the SAPG method are lost.

A similar process can be applied to obtain the reduced model of the SAPG-DEIM test space equation (Equation (4.25)). The first step is the substitution of a low-rank representation of $\boldsymbol{W}_{\text{DEIM}}$, resulting in

$$\left[c\boldsymbol{M} + \boldsymbol{U}\left[\boldsymbol{P}^T\boldsymbol{U}\right]^\dagger \boldsymbol{P}^T\boldsymbol{A}\right]^T \boldsymbol{\Phi}\boldsymbol{T} = \boldsymbol{V}. \tag{4.42}$$

Second is the projection of the entire system to reduce its size, yielding

$$\boldsymbol{\Lambda}^T\left[c\boldsymbol{M} + \boldsymbol{U}\left[\boldsymbol{P}^T\boldsymbol{U}\right]^\dagger \boldsymbol{P}^T\boldsymbol{A}\right]^T \boldsymbol{\Phi}\boldsymbol{T} = \boldsymbol{\Lambda}^T\boldsymbol{V}. \tag{4.43}$$

Finally, taking $\boldsymbol{\Lambda} = \boldsymbol{V}_h$ and $\boldsymbol{\Phi} = \boldsymbol{V}_h$ for convenience yields

$$\begin{aligned}\boldsymbol{V}_h^T\left[c\boldsymbol{M} + \boldsymbol{U}\left[\boldsymbol{P}^T\boldsymbol{U}\right]^\dagger \boldsymbol{P}^T\boldsymbol{A}\right]^T \boldsymbol{\Phi}\boldsymbol{T} &= \mathbb{I}_{h_{\boldsymbol{V}}\times n_{\boldsymbol{V}}},\\ \left[\boldsymbol{V}_h^T\left[c\boldsymbol{M} + \boldsymbol{U}\left[\boldsymbol{P}^T\boldsymbol{U}\right]^\dagger \boldsymbol{P}^T\boldsymbol{A}\right]\boldsymbol{V}_h\right]^T \boldsymbol{T} &= \mathbb{I}_{h_{\boldsymbol{V}}\times n_{\boldsymbol{V}}}.\end{aligned} \tag{4.44}$$

The nonlinear version of the reduced SAPG equations are

$$\left[\boldsymbol{V}_h^T\left[c\boldsymbol{M} + \left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}^\eta}\right]\boldsymbol{V}_h\right]^T \boldsymbol{T} = \mathbb{I}_{h_{\boldsymbol{V}}\times n_{\boldsymbol{V}}}, \tag{4.45}$$

$$\left[\boldsymbol{V}_h^T\left[c\boldsymbol{M} + \boldsymbol{U}\left[\boldsymbol{P}^T\boldsymbol{U}\right]^\dagger \boldsymbol{P}^T\left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}^\eta}\right]\boldsymbol{V}_h\right]^T \boldsymbol{T} = \mathbb{I}_{h_{\boldsymbol{V}}\times n_{\boldsymbol{V}}}. \tag{4.46}$$

4.1.6 Stability Analysis

The following section performs eigenvalue analysis of GPOD, LSPG-POD, and SAPG-POD to study their effects on stability. This analysis is performed on a 1D advection-diffusion problem with periodic boundary conditions. Eigenvalue analysis is typically done to analyze the stability of spatial and temporal discretization schemes, and although the setting for this problem is fairly simple, schemes that fail to remain stable for this problem will likely not be stable for more complex problems. The 1D advection-diffusion problem

takes the form

$$\frac{\partial x(d)}{\partial t} + a\frac{\partial x(d)}{\partial d} - \nu\frac{\partial^2 x(d)}{\partial d^2} = 0, \tag{4.47}$$

where $x(d)$ is the value of the scalar at position d, a is the advection speed, and ν is the diffusivity of the scalar. The domain of the system can be discretized, and finite difference approximations can be used to approximate the spatial derivatives. Backwards differencing to approximate the advection term and central differencing to approximate the diffusive term transforms Equation (4.47) into a semi-discrete system,

$$\frac{\partial x_i^n}{\partial t} + a\frac{x_i - x_{i-1}}{\Delta d} - \nu\frac{x_{i+1} - 2x_i + x_{i-1}}{(\Delta d)^2} = 0, \tag{4.48}$$

where x_i is the value of the scalar at discretization node i and Δd is the size of the spatial discretization. For this exercise, the length of the domain (L) is discretized evenly to give N spatial nodes, making the size of the spatial discretization equal to

$$\Delta d = \frac{L}{N-1}. \tag{4.49}$$

Once discretized, Equation (4.48) can be rewritten in vector form with

$$\boldsymbol{M}\frac{d\boldsymbol{x}}{dt} + \boldsymbol{A}\boldsymbol{x} = 0, \tag{4.50}$$

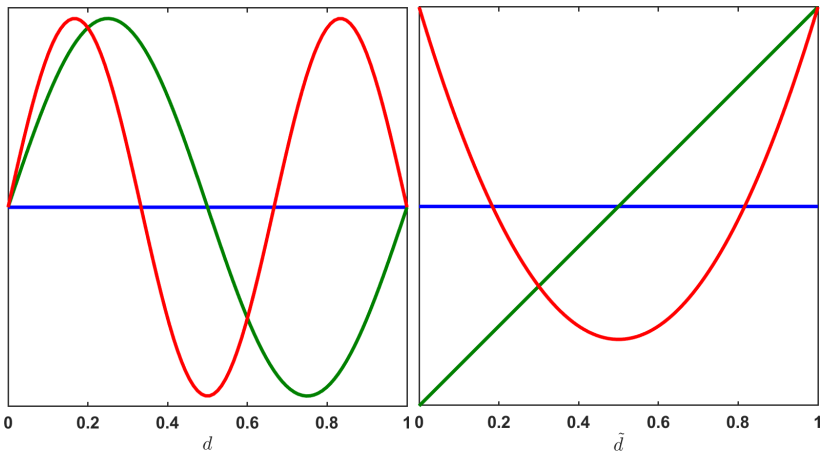
where $M = \mathbb{I}$ is the mass matrix and $\boldsymbol{A}$ is the Jacobian of the problem with 3-banded structure.

The trajectory of the system to be analyzed is then

$$\frac{d\boldsymbol{x}}{dt} = -\underbrace{\boldsymbol{M}^{-1}\boldsymbol{A}}_{\boldsymbol{C}}\boldsymbol{U}. \tag{4.51}$$

Specifically, if any of the eigenvalues of $\boldsymbol{C}$ have a positive real component, then the system will grow unbounded in the shape of the associated eigenvector, whereas if the real component of an eigenvalue is negative, then the system scalar will decrease in magnitude with the associated eigenvector. Substituting the low-rank approximation of $\boldsymbol{x}$ and then projecting the entire system formulates the reduced equivalent to this equation,

$$\frac{d\hat{\boldsymbol{x}}}{dt} = -\underbrace{\left[\boldsymbol{W}^T\boldsymbol{M}\boldsymbol{V}\right]^{-1}\left[\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V}\right]}_{\hat{\boldsymbol{C}}}\hat{\boldsymbol{x}}. \tag{4.52}$$



(a) Fourier **(b)** Legendre

Figure 4.1: The first 3 basis vectors for a Fourier and a Legendre basis. The scale of these basis vectors is not relevant, as they are superimposed with the appropriate weights when used online.

In this case, $\boldsymbol{W} = \boldsymbol{V}$ if a Galerkin ROM is formed, $\boldsymbol{W} = [\boldsymbol{M} + \boldsymbol{A}]\,\boldsymbol{V}$ if LSPG is used, and $\boldsymbol{W} = [\boldsymbol{M} + \boldsymbol{A}]^{-T}\,\boldsymbol{V}$ if SAPG is used. Two separate bases are used for this exercise: a Fourier basis and a Legendre basis. The Fourier basis is composed of harmonics of the scalar value in the domain, and the i^{th} Fourier basis vector is the i^{th} harmonic, according to

$$v_i = \sin \frac{di\pi}{L}. \tag{4.53}$$

The Legendre basis is composed of Legendre polynomials, which are orthonormal polynomials defined for $d \in [-1, 1]$ and with an output range of $v_n \in [-1, 1]$. For this exercise, the domain was normalized to the input range of the Legendre polynomials, according to

$$\tilde{d} = 2d - 1, \tag{4.54}$$

and the Legendre polynomials were generated using Bonnet's recursive formula [123], defined by

$$v_1 = 1, \tag{4.55}$$

$$v_2 = \tilde{d}, \tag{4.56}$$

$$v_n = \frac{(2n-1)\tilde{d}v_{n-1} - (n-1)v_{n-2}}{n}. \tag{4.57}$$

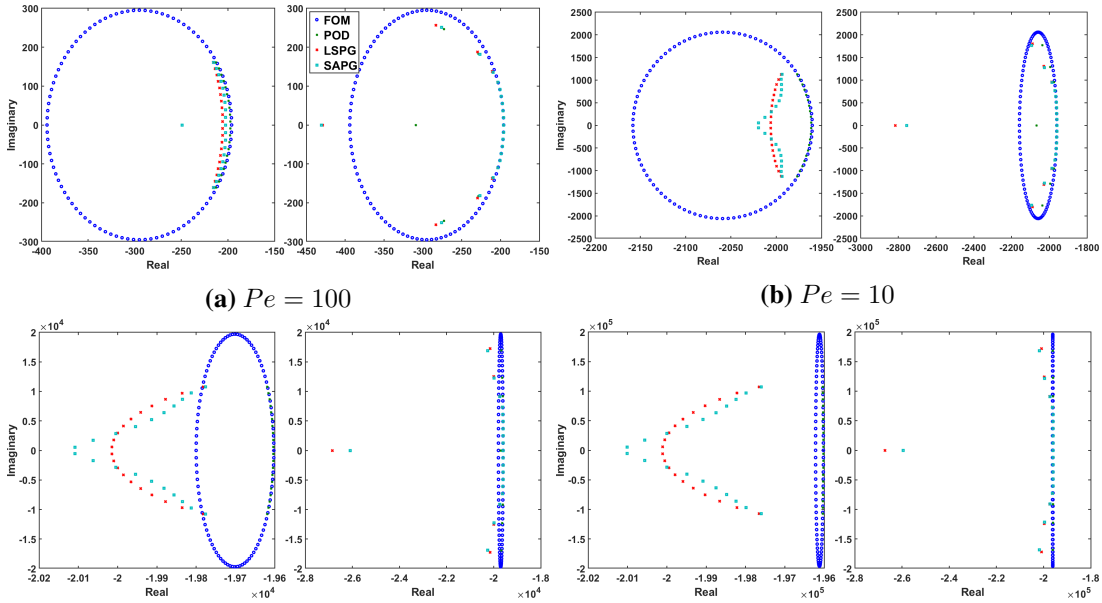
Figure 4.1 displays the first 3 basis vectors for the Fourier and Legendre bases. 20 basis vectors are used for each basis in this exercise.

The Péclet number characterizes the ratio of the advection and the diffusion of the

system and is defined by

$$Pe = \frac{aL}{\nu}. \tag{4.58}$$

The Péclet number is useful for determining the discretization schemes deployed throughout a larger CFD system, as the stability of a scheme is often directly influenced by the strength of local advection transport relative to the strength of local diffusion transport. The eigenvalues of the Galerkin ROM, LSPG, and SAPG systems constructed with Legendre and Fourier bases are shown in Figure 4.2 for different values of the Péclet number, $N = 100$, and $a = 1$. Overall, the magnitude and distribution of the eigenvalues for the LSPG and SAPG systems are similar. For highly advective systems, their values are bounded by the eigenvalues of the FOM system. As the system becomes more diffusive, the magnitude of the largest negative eigenvalue for both systems grows larger than the largest negative FOM eigenvalue. This indicates a higher stiffness of the system due to a higher dissipation.



(a) $Pe = 100$ **(b)** $Pe = 10$

(c) $Pe = 1$ **(d)** $Pe = 0.1$

Figure 4.2: Eigenvalues for the linear advection-diffusion problem at different Péclet numbers. Fourier and Legendre bases are used in the left and right columns, respectively.

4.1.7 Computational Complexity

This section examines the computational complexity of formulating the SAPG and LSPG test bases by computing the total number of floating point operations (FLOPs) needed for their construction. According to Equation (4.7), LSPG requires a general matrix multipli-

cation (GEMM) of the linearization of the system (an $n_{\boldsymbol{x}} \times n_{\boldsymbol{x}}$ matrix) and the state basis (an $n_{\boldsymbol{x}} \times n_{\boldsymbol{V}}$ matrix). This incurs a cost of ${n_{\boldsymbol{x}}}^2$ multiplications and ${n_{\boldsymbol{x}}}^2$ additions for each of the $n_{\boldsymbol{V}}$ basis vectors. The sum of these operations yields the total additional computational cost of forming the LSPG test basis, according to

$$\text{Cost of LSPG} = 2n_{\boldsymbol{V}}{n_{\boldsymbol{x}}}^2. \tag{4.59}$$

The reduced construction of the SAPG test basis has three stages. The first formulates the reduced SAPG equations, defined by Equation (4.36) and Equation (4.43). The right-hand side of the reduced equations is invariant and can be precomputed; however, the left-hand side is constructed with each SAPG test basis formulation. For this section, the ranks of the test search space ($\boldsymbol{\Phi}$) and the system projector of the reduced SAPG equation ($\boldsymbol{\Lambda}$) are assumed to be the same and equal to $n_{\boldsymbol{\Phi}}$. The left-hand side consists of two GEMM products. The first is defined by

$$\boldsymbol{\Lambda}^T \frac{\partial \bar{\boldsymbol{R}}}{\partial \boldsymbol{x}} = \mathbb{J}, \tag{4.60}$$

where $\mathbb{J}$ is the first product and is of size $\mathbb{R}^{n_{\boldsymbol{\Phi}} \times n_{\boldsymbol{x}}}$. This product is similar to the GEMM used to form the LSPG test basis, incurring a cost of $2{n_{\boldsymbol{x}}}^2 n_{\boldsymbol{\Phi}}$ FLOPs. The second product is defined by

$$\mathbb{J}\boldsymbol{\Phi} = \mathbb{A}, \tag{4.61}$$

where $\mathbb{A}$ is the second product and is of size $\mathbb{R}^{n_{\boldsymbol{\Phi}} \times n_{\boldsymbol{\Phi}}}$. This product requires $2n_{\boldsymbol{x}}{n_{\boldsymbol{\Phi}}}^2$ FLOPs. Thus, the total cost of formulating the SAPG equation is $2{n_{\boldsymbol{x}}}^2 n_{\boldsymbol{\Phi}} + 2n_{\boldsymbol{x}}{n_{\boldsymbol{\Phi}}}^2$. The second stage produces the solution of the SAPG equation, defined by

$$\mathbb{A}\boldsymbol{T} = \boldsymbol{\Lambda}^T \boldsymbol{V}. \tag{4.62}$$

An efficient way to solve the $n_{\boldsymbol{V}}$ linear systems is to construct the LU-factorization of $\mathbb{A}$ followed by forward and backward substitution for each system. The LU-factorization decomposes a matrix into a lower and an upper-triangular matrix ($\boldsymbol{L}$ and $\boldsymbol{U}$, respectively), according to

$$\mathbb{A} = \boldsymbol{L}\boldsymbol{U}. \tag{4.63}$$

LU-factorization is equivalent to Gaussian elimination and has a computational complexity

of $\frac{2}{3}{n_{\boldsymbol{\Phi}}}^3 - {n_{\boldsymbol{\Phi}}}^2 - \frac{n_{\boldsymbol{\Phi}}}{6}$ [124]. The resulting SAPG system is defined by

$$\boldsymbol{LUT} = \boldsymbol{\Lambda}^T \boldsymbol{V}. \tag{4.64}$$

Taking advantage of the structure of the upper and lower triangular matrices, forward and backward substitution is used to solve each linear system of Equation (4.64) efficiently. The computational complexities of forward substitution to invert $\boldsymbol{L}$ and backward substitution to invert $\boldsymbol{U}$ are equivalent and require $n_{\boldsymbol{\Phi}}$ divisions, $\frac{{n_{\boldsymbol{\Phi}}}^2 - n_{\boldsymbol{\Phi}}}{2}$ multiplications, and $\frac{{n_{\boldsymbol{\Phi}}}^2 - n_{\boldsymbol{\Phi}}}{2}$ additions each. Thus the total cost of solving Equation (4.62) is $\frac{2}{3}{n_{\boldsymbol{\Phi}}}^3 - {n_{\boldsymbol{\Phi}}}^2 - \frac{1}{6}n_{\boldsymbol{\Phi}} + 2{n_{\boldsymbol{\Phi}}}^2 n_{\boldsymbol{V}}$.

Partial pivoting may be necessary for numerical stability when computing the LU-factorization of $\mathbb{A}$. The PLU-factorization is defined by

$$\boldsymbol{P}\mathbb{A} = \boldsymbol{LU}, \tag{4.65}$$

where $\boldsymbol{P}$ is a matrix of elementary vectors used for row pivoting. The algorithm used to compute the partial pivots is applied intermittently during the Gaussian elimination steps and chooses the row index containing the largest magnitude column value. For an $n_{\boldsymbol{\Phi}} \times n_{\boldsymbol{\Phi}}$ matrix, this requires $\frac{{n_{\boldsymbol{\Phi}}}^2 + n_{\boldsymbol{\Phi}}}{2}$ FLOPs. Thus the computational complexity to solve the SAPG test basis weights using PLU-factorization is $\frac{2}{3}{n_{\boldsymbol{\Phi}}}^3 - \frac{1}{2}{n_{\boldsymbol{\Phi}}}^2 + \frac{1}{3}n_{\boldsymbol{\Phi}} + 2{n_{\boldsymbol{\Phi}}}^2 n_{\boldsymbol{V}}$.

The final stage of the SAPG model is to construct the SAPG test basis vectors using Equation (4.34) which is a GEMM of the test search space (an $n_{\boldsymbol{x}} \times n_{\boldsymbol{\Phi}}$ matrix) and the test basis weights (an $n_{\boldsymbol{\Phi}} \times n_{\boldsymbol{V}}$ matrix). This incurs a cost of $2n_{\boldsymbol{\Phi}} n_{\boldsymbol{V}} n_{\boldsymbol{x}}$ FLOPs. Adding the cost of formulating (4.62), the cost of PLU-factorization, and the cost of the GEMM to form the test basis yields the total additional cost of the reduced SAPG test basis,

$$\begin{aligned}\text{Cost of Reduced-SAPG} = {} & 2n_{\boldsymbol{\Phi}}{n_{\boldsymbol{x}}}^2 \\ & + (2{n_{\boldsymbol{\Phi}}}^2 + 2n_{\boldsymbol{V}} n_{\boldsymbol{\Phi}})n_{\boldsymbol{x}} + \left[\frac{2}{3}{n_{\boldsymbol{\Phi}}}^3 + (2n_{\boldsymbol{V}} - \frac{1}{2}){n_{\boldsymbol{\Phi}}}^2 + \frac{1}{3}n_{\boldsymbol{\Phi}}\right]\end{aligned} \tag{4.66}$$

Once the SAPG and LSPG test bases are formed, the corresponding ROMs are constructed and a reduced state update arises from an iterative solver. For the Newton-Raphson method described in Algorithm 3, the computational costs for the LSPG, SAPG and Galerkin ROMs are equivalent. Neglecting the full-order sized residual and Jacobian computation, each reduced state update consists of three stages. The first is the formulation of the reduced residual, which is the product of the transpose of test basis (an $n_{\boldsymbol{x}} \times n_{\boldsymbol{V}}$ matrix) and the full-order sized residual (an $n_{\boldsymbol{x}}$ vector). This operation costs $2n_{\boldsymbol{x}} n_{\boldsymbol{V}}$ FLOPs.

The second stage constructs the reduced Jacobian which is similar to the construction of the SAPG test basis Jacobian and costs $2{n_{\boldsymbol{x}}}^2 n_{\boldsymbol{V}} + 2n_{\boldsymbol{x}}{n_{\boldsymbol{V}}}^2$ FLOPs. The final stage inverts the reduced Jacobian to yield the reduced state update in Equation (4.28). PLU-factorization with forward and backward substitution can be used at the cost of $\frac{2}{3}{n_{\boldsymbol{V}}}^3 + \frac{3}{2}{n_{\boldsymbol{V}}}^2 + \frac{1}{3}n_{\boldsymbol{V}}$. Thus the per-iteration cost of the Newton-Raphson method for each reduced state update is

$$\begin{aligned}\text{Cost of ROM Update} = {} & 2n_{\boldsymbol{V}}{n_{\boldsymbol{x}}}^2 \\ & + (2{n_{\boldsymbol{V}}}^2 + 2n_{\boldsymbol{V}})n_{\boldsymbol{x}} + \left[\frac{2}{3}{n_{\boldsymbol{V}}}^3 + \frac{3}{2}{n_{\boldsymbol{V}}}^2 + \frac{1}{3}n_{\boldsymbol{V}}\right]. \quad (4.67)\end{aligned}$$

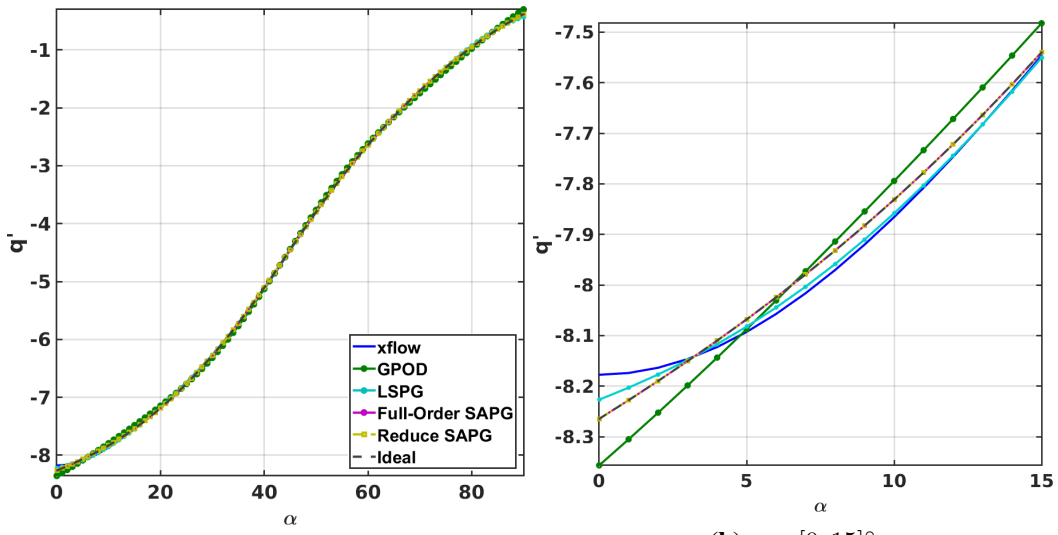
The leading terms of Equation (4.59), Equation (4.66), and Equation (4.67) are $\mathcal{O}(2n_{\boldsymbol{V}}{n_{\boldsymbol{x}}}^2)$, $\mathcal{O}(2n_{\boldsymbol{\Phi}}{n_{\boldsymbol{x}}}^2)$, and $\mathcal{O}(2n_{\boldsymbol{V}}{n_{\boldsymbol{x}}}^2)$, respectively. For the reduced formulation of the SAPG model to be well-conditioned, $n_{\boldsymbol{\Phi}} > n_{\boldsymbol{V}}$; however, the size of $n_{\boldsymbol{\Phi}}$ needs to not be substantially greater than $n_{\boldsymbol{V}}$ for practicality – *e.g.*, $n_{\boldsymbol{\Phi}} = 2n_{\boldsymbol{V}}$. Thus the ratio of additional costs of the LSPG and reduced SAPG test bases is approximately proportional to $\frac{n_{\boldsymbol{V}}}{n_{\boldsymbol{\Phi}}}$. Furthermore, when considering the cost of using the Newton-Raphson method to solve the ROMs and neglecting the cost of residual and Jacobian calculation, the total costs of forming and solving the LSPG and SAPG ROMs are $\mathcal{O}(4n_{\boldsymbol{V}}{n_{\boldsymbol{x}}}^2)$ and $\mathcal{O}((2n_{\boldsymbol{V}}+2n_{\boldsymbol{\Phi}})n_{\boldsymbol{x}}{}^2)$, respectively. Thus the scaling to larger sized problems should be similar between LSPG, SAPG, and Galerkin ROMs – *i.e.*, quadratic with respect to $n_{\boldsymbol{x}}$; however, the cost of SAPG will still be higher than LSPG and POD, which can be approximated by

$$\frac{\text{Cost of SAPG}}{\text{Cost of LSPG}} \approx \frac{n_{\boldsymbol{V}} + n_{\boldsymbol{\Phi}}}{2n_{\boldsymbol{V}}}, \quad (4.68)$$

$$\frac{\text{Cost of SAPG}}{\text{Cost of PROM}} \approx \frac{n_{\boldsymbol{V}} + n_{\boldsymbol{\Phi}}}{n_{\boldsymbol{V}}}. \quad (4.69)$$

4.2 Example Applications

The following section demonstrates the use of the SAPG method on various problems. These examples include problems from Section 3.5, specifically the steady scalar advection diffusion, the pitching and plunging airfoil, and the XRF1-HARW plunging problem. In addition, this section includes a scalar advection-diffusion problem with a nonlinear source component. These examples will verify the theory and implementation of the SAPG method, demonstrate the exactness between the *ideal* and SAPG solutions, make comparisons to the LSPG test space, demonstrate the effectiveness of the SAPG method in a nonlinear and a hyper-reduced settings, and study the limitations of the reduced SAPG



(a) $\alpha \in [0, 90]^\circ$ **(b)** $\alpha \in [0, 15]^\circ$

Figure 4.3: Comparisons between *Ideal*, GPOD, SAPG-POD, reduced SAPG-POD, and LSPG-POD reconstructions.

model.

4.2.1 Scalar Advection-diffusion

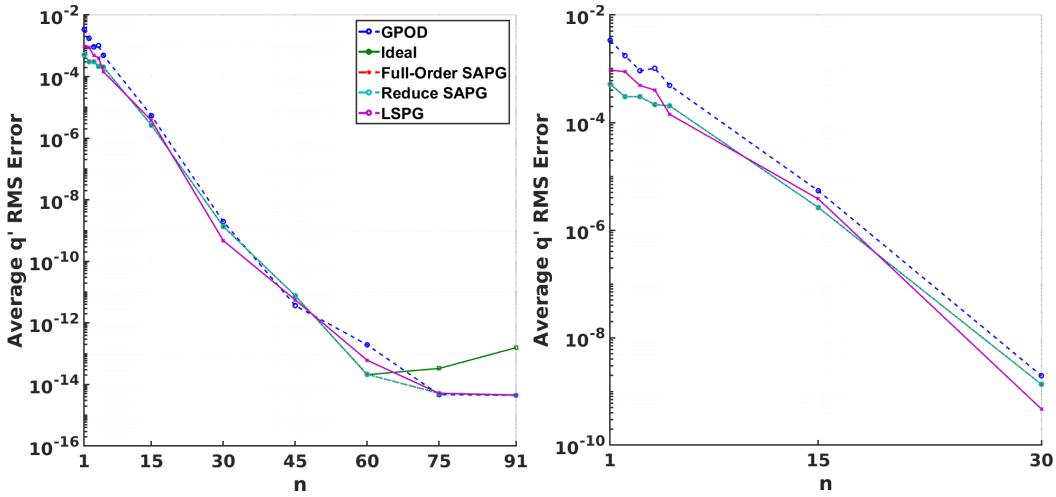
The following example is described in Section 3.5.1. To summarize, steady-solutions to a 2D scalar advection-diffusion problem compose the snapshot set where the advection angle at the domain boundaries is varied according to $\alpha \in [0^\circ, 90^\circ]$. The governing equation for the scalar advection-diffusion model is

$$\nabla \cdot (\vec{v}x) - \nu \nabla^2 x + S = 0, \tag{3.62}$$

where S is a source term, and ν is the diffusivity. For this example, $\nu = 0.001$ and $S = 1.0$. The domain is a 3×3 mesh uniformly discretized into 27×27 elements and with a bilinear state basis in each element.

As a small comparative study, the GPOD, LSPG-POD, and the full-order SAPG-POD ROMs were solved with 5 basis vectors in the trial space – containing 99.7% of the total singular value energy – for the snapshot set configurations. The results in Figure 4.3 demonstrate that the SAPG-POD predictions and the *ideal* solutions align exactly, while the GPOD and the LSPG-POD models do not align with the *ideal* solution.

Taking a more comprehensive view, the errors of several GPOD, LSPG-POD, SAPG-POD, and *ideal* solutions at various numbers of POD basis sizes are shown in Figure 4.4. With the exception of the POD basis sizes of 75 and 91, all of the SAPG-POD models align with the *ideal* solutions exactly. Numerical error is suspected to have caused the rise



(a) rank $(\boldsymbol{V}) \in [1, 91]$ **(b)** rank $(\boldsymbol{V}) \in [1, 30]$

Figure 4.4: Errors of the *Ideal*, GPOD, SAPG-POD, reduced SAPG-POD, and LSPG-POD reconstructions.

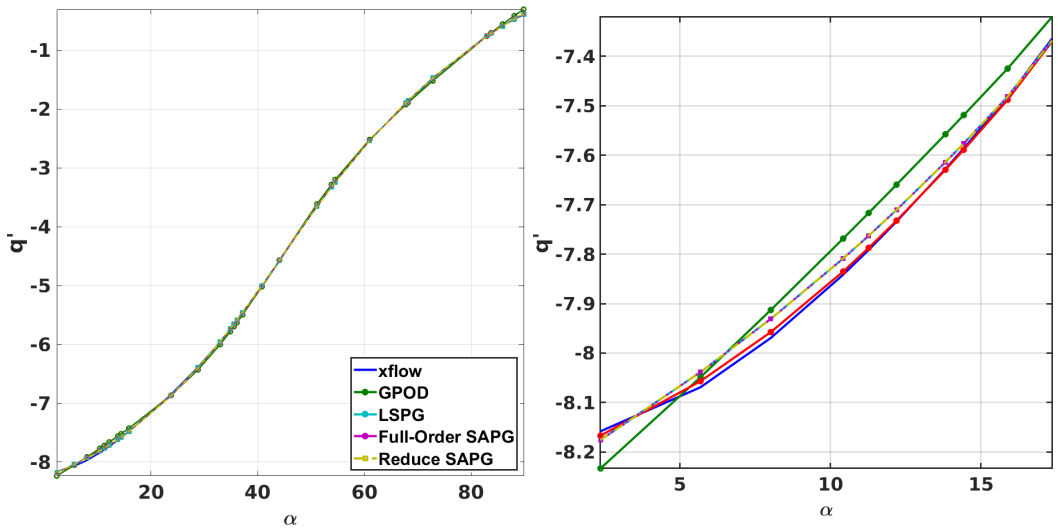
in error of the *ideal* solutions at the finest basis sizes. In addition, for the coarser ROMs, the SAPG-POD solutions are an order of magnitude more accurate than the GPOD solutions. However, the LSPG-POD model is fairly comparable with the SAPG-POD model.

The results from the reduced SAPG-POD model (reduced SAPG-POD) are also presented in Figures 4.3 and 4.4. The reduced SAPG-POD solutions were generated using all 91 basis vectors as the fine-space basis with every update to the state. The reduced SAPG-POD solutions match the full-order SAPG-POD solutions and the *ideal* solutions.

For at least reconstruction, the results of the full-order SAPG-POD formulation comport with the earlier derived theory. In addition, the information in the fine-space POD basis vectors is sufficient to produce SAPG test basis vectors for the reduced model. However, we note that all of the characteristics of the desired reconstructed solutions are already contained in the fine POD basis. To test the robustness of these approaches, predicting solutions for convection angles not already contained in the snapshot set is considered.

Thirty randomly-generated, non-snapshot convection angles are chosen for ROM predictions. The same POD-ROMs employed for snapshot reconstruction are used for prediction. The predictions from the POD-ROMs consisting of 5 basis vectors are shown in Figure 4.5. As before, the SAPG-POD, reduced SAPG-POD, and the *ideal* solutions align exactly, while the GPOD and LSPG-POD solutions do not. The greatest errors in the GPOD solution occur near the boundaries; however, the LSPG-POD solution aligns better with the output of the full-order model than the SAPG-POD model and the *ideal* solutions.

While the predicted solutions appear relatively similar to the reconstruction solutions, the errors with respect to the number of basis vectors, shown in Figure 4.6, are more reveal-



(a) $\alpha \in [0, 90]^\circ$ **(b)** $\alpha \in [0, 30]^\circ$

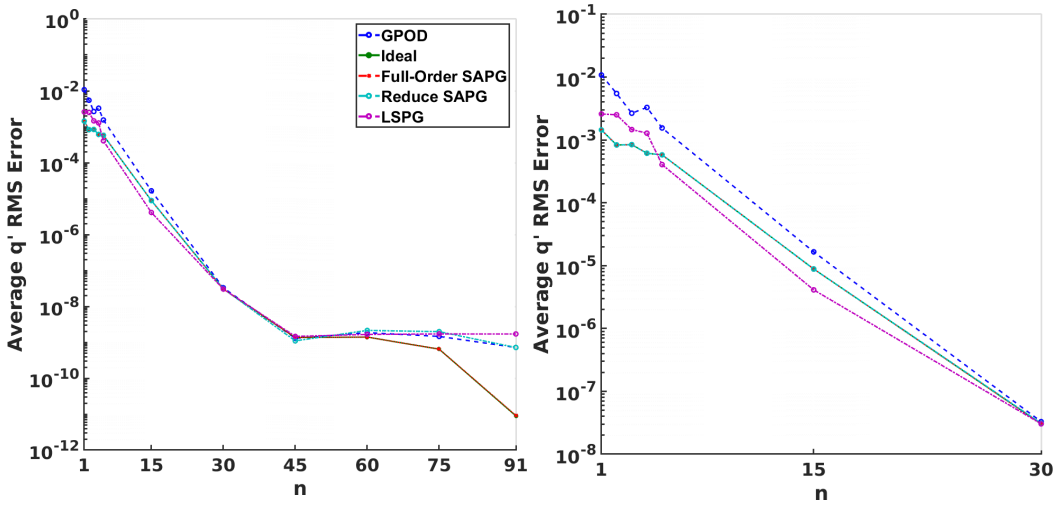
Figure 4.5: *Ideal*, GPOD, SAPG-POD, reduced SAPG-POD, and LSPG-POD predictions.

ing. For the coarser ROMs, the SAPG-POD and the reduced SAPG-POD predictions match the *ideal* solution. However, divergence of the reduced SAPG-POD model is observed, beginning at about $n > 40$ as the trial space becomes finer. The reduced SAPG-POD predictions begin to follow the GPOD predictions, becoming exactly equal to the GPOD model at $n = 91$.

This clearly shows the limitations of the reduced formulation using a fine set of state basis vectors. The reduced SAPG-POD searches the fine projection space for the solution to the SAPG test basis equation. Although the POD model increases in size, the test search space remains constant. With increases in the size of the trial space, the unique content of the test search space and the additional information that can be obtained by the test space decreases. Once the trial space and the test search space are identical, there is no additional information that the reduced SAPG-POD test space can obtain that is not already brought out by Galkerin projection. The effects of an insufficient size of the test search space is further investigated in Section 4.2.4.

Conversely, the full-order SAPG-POD model does not diverge and stays aligned with the *ideal* solution throughout the entire domain. This is because the full-order SAPG-POD model is able to search the entire state space for the SAPG test basis vectors. However, in most cases, it is not necessary to have a trial space size that approaches the size of the snapshot set, so these results still bode well for the reduced SAPG method as the coarser ROMs demonstrate the ability of the reduced SAPG method to produce good approximations of the SAPG test space.

The performance of the GPOD model on this scalar example, while not matching the

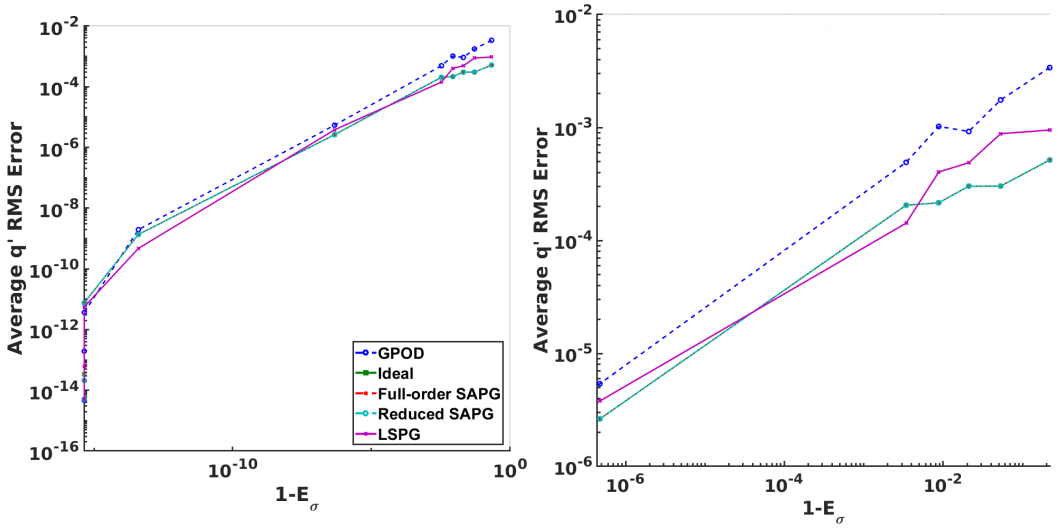


(a) rank$(\boldsymbol{V}) \in [1, 91]$ **(b)** rank$(\boldsymbol{V}) \in [1, 15]$

Figure 4.6: Errors of the *Ideal*, GPOD, SAPG-POD, the reduced SAPG-POD, and LSPG-POD predictions.

ideal solution, is still reasonable; additionally, the performance of the LSPG-POD model is not only competitive but also beats the SAPG and ideal solutions for output prediction. However, the flux on the right boundary is not fully-representative of the entire solution domain. State space domain errors are displayed in 4.8 for the prediction case with $n = 5$; additionally, the figure shows that the LSPG-POD model has the worst prediction of the entire domain of the system. Furthermore, the full-order SAPG-POD, reduced SAPG-POD, and *ideal* solution are all aligned and have the lowest error. It is observed that the *ideal* solution has the lowest error and the full-order SAPG-POD aligns with the *ideal* solution for all of the tested problems; and where the reduced SAPG-POD has sufficient test search space information, its domain solution also matches the *ideal* solution.

Two additional sets of SAPG-POD solutions are constructed in order to assess the balance of added accuracy and added cost of SAPG-POD over LSPG-POD and Galerkin POD. The first uses a fixed trial space of 15 POD state basis vectors while varying the size of the test search space with $n_{\boldsymbol{\Phi}} \in [15, 45]$ state basis vectors in increments of five basis vectors. The second fixes the size of the test search space to be twice the size of the trial space and varies the size of the trial space from one POD basis vector to 45 POD basis vectors with increments of one basis vector for $n_{\boldsymbol{V}} \in [1, 5]$ and five basis vectors for $n_{\boldsymbol{V}} \in [10, 45]$. The total costs of these additional SAPG-POD models and the original ROM solutions are estimated with Equation (4.59), Equation (4.66), and Equation (4.67). The average root-mean



(a) Reconstruction: Full Domain

(b) Reconstruction: Partial Domain

(c) Prediction: Full Domain

(d) Prediction: Partial Domain

Figure 4.7: Errors of the *Ideal*, GPOD, SAPG-POD, the reduced SAPG-POD, and LSPG-POD reconstruction and prediction for linear problem with respect to $1 - E_\sigma$. The ROM size decreases moving left to right.

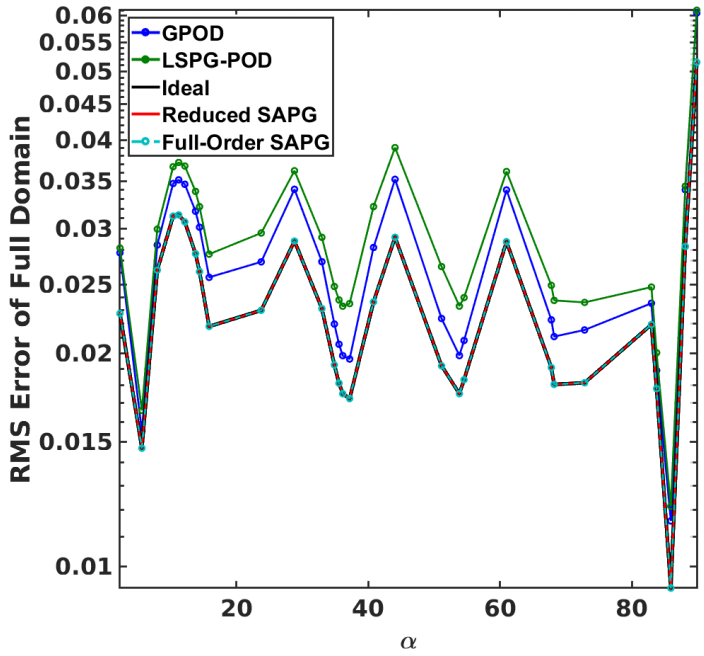


Figure 4.8: Root mean square error inside the entire domain for solutions generated with 5 basis vectors.

square (RMS) reduced state error is computed for each solution, according to

$$e_{\text{RMS},\hat{\boldsymbol{x}},i} = \sqrt{\frac{\left[\hat{\boldsymbol{x}}_i - \boldsymbol{V}^T\boldsymbol{x}_{\text{FOM},i}\right]^T\left[\hat{\boldsymbol{x}}_i - \boldsymbol{V}^T\boldsymbol{x}_{\text{FOM},i}\right]}{n_{\boldsymbol{V}}}}, \tag{4.70}$$

$$\bar{e}_{\text{RMS},\hat{\boldsymbol{x}}} = \sum_{i=1}^{N} \frac{e_{\text{RMS},\hat{\boldsymbol{x}},i}}{N}, \tag{4.71}$$

where N is the number of solutions. Figure 4.9 shows that for this linear problem the cost to obtain a given accuracy with SAPG models is less expensive than the cost to obtain the same accuracy with LSPG. Furthermore, the reduced state accuracy and cost are strongly dependent on the size of the test search space. With finer test search spaces, the accuracy and cost of the SAPG models increase; furthermore, with the full set of POD state basis vectors, the accuracy of the SAPG model is fairly constant. However, the errors of the SAPG model are highest when the test search space provides very little or no additional information. This occurs when the test search space is equal to the trial space for the SAPG $n_{\boldsymbol{\Phi}} = 90$ and the SAPG $n_{\boldsymbol{V}} = 15$ cases and when the trial basis and test search space are both very coarse as in the coarser SAPG $n_{\boldsymbol{\Phi}} = 2n_{\boldsymbol{V}}$ case. These accuracy issues disappear with refinement of the test search space. Although for this problem SAPG is more accurate than the LSPG model for similar costs, solving a finer trial space Galerkin POD model is more efficient than both SAPG and LSPG up to a certain level of accuracy. However,

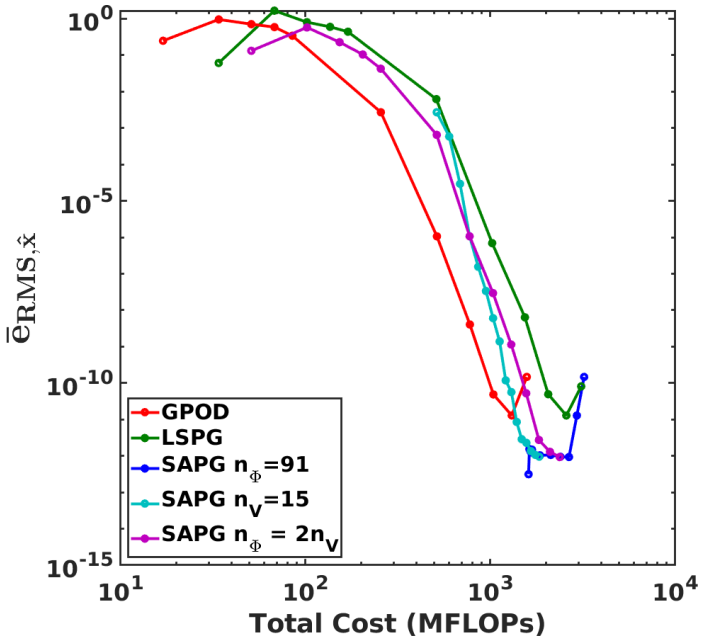


Figure 4.9: RMS state error of different ROM solutions versus the cost of formulating the ROM. There are three SAPG models, each using the reduced formulations with the fine-space POD state basis vectors as the test search space. SAPG $n_{\boldsymbol{\Phi}} = 91$ varies the trial basis while keeping the test space equal to 91 state basis vectors. SAPG $n_{\boldsymbol{V}} = 15$ holds the trial basis constant while varying the size of the test search space. SAPG $n_{\boldsymbol{\Phi}} = 2n_{\boldsymbol{V}}$ varies the trial basis with the rank of the test search space fixed at twice the size of the trial basis.

SAPG is able to achieve more accurate solutions, and later examples in this chapter show that SAPG models can be more stable and accurate for problems where Galerkin ROMs and LSPG ROMs fail, justifying the cost. Additionally, the number of test search space basis vectors may be reduced if they are tailored for representing solutions to the full-order SAPG equation – *i.e.*, arising from snapshots of the full-order SAPG model – as tailored basis vectors should be able to span the reduced state adjoint space of interest more accurately than a set of state basis vectors of the same size.

4.2.2 Nonlinear Model Example

Building on the linear example above, nonlinearity is introduced by replacing the constant source term with one that is a nonlinear function of the state. The entire system can be written as

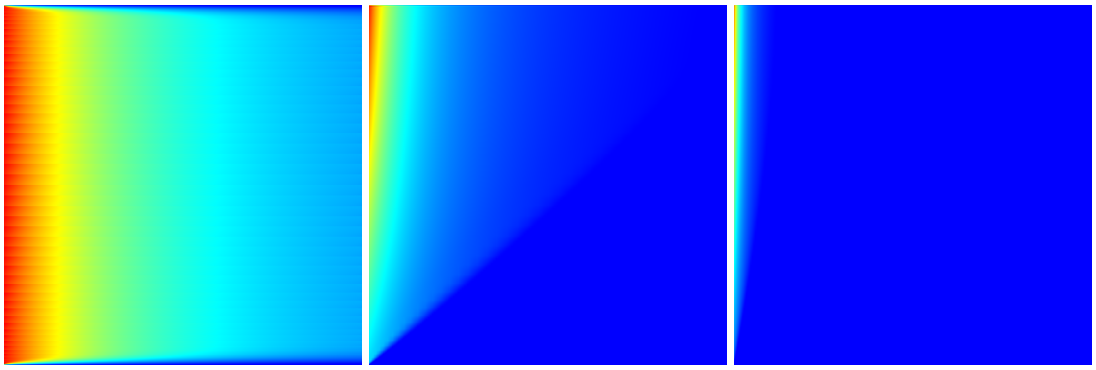
$$\nabla \cdot (\vec{v}x) - \nu \nabla^2 x + S(x) = 0, \tag{4.72}$$

$$S(x) = S_0 x^2, \tag{4.73}$$

with S_0 being a constant. For this example, the quadratic source coefficient is $S_0 = 1.0$, and the Péclet number is increased to 1000 by lowering the diffusion coefficient, ν, to 0.001. To maintain physical solutions, the mesh is refined such that 54 elements lie in the horizontal and vertical directions of the domain. In both directions, the length of each element decreases from the center, according to a cosine spacing. The element sizes are determined by $l_i = L\frac{1-\cos(\theta_i)}{2}$ where l_i is the width/height of element i numbered from the top/left to bottom/right, θ_i is the corresponding angle from a uniformly spaced array $[0, \pi]$ rad, and L is the entire domain's width/height (in this case $L = 3$ for both width and height). α is sampled from 0° to 80° at 81 evenly distributed points. The GPOD, LSPG-POD, and reduced SAPG-POD models are tested on both reconstruction of the original set of parameters and prediction of the midpoints between the sampled convection angles.

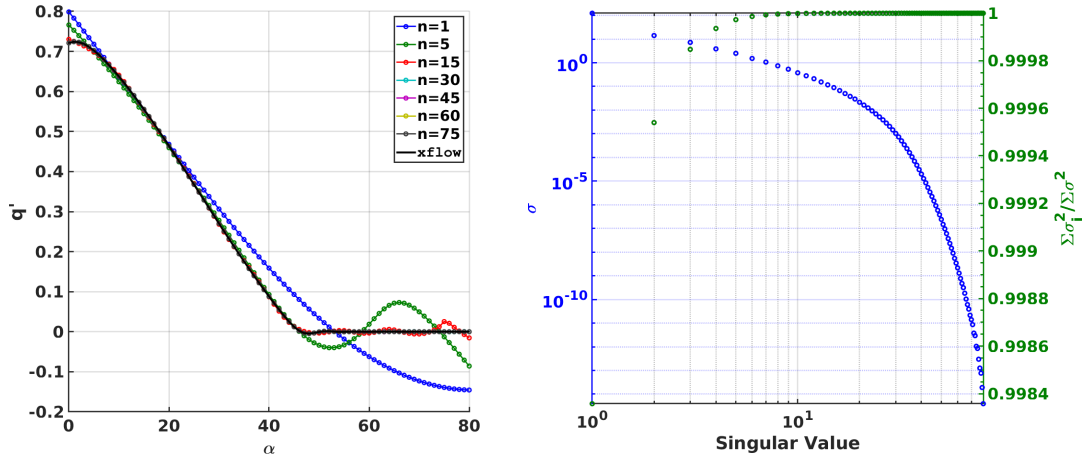
From Figure 4.10, we can tell that the additional nonlinear source term drastically changes the solution; additionally, as seen in Figure 4.10, the output changes drastically. Most importantly, changes of α above approximately 50° have an insignificant effect on the flux of x.

Unsurprisingly, the GPOD model performs the worst of the three methods and is unable to fully resolve the right boundary, as seen in Figure 4.12, where the solution from the GPOD model with 5 basis vectors oscillates about the FOM model solution. Like before, the LSPG-POD model performs better, with a less pronounced oscillation, and the reduced SAPG-POD model solution follows the *ideal* solution quite well.



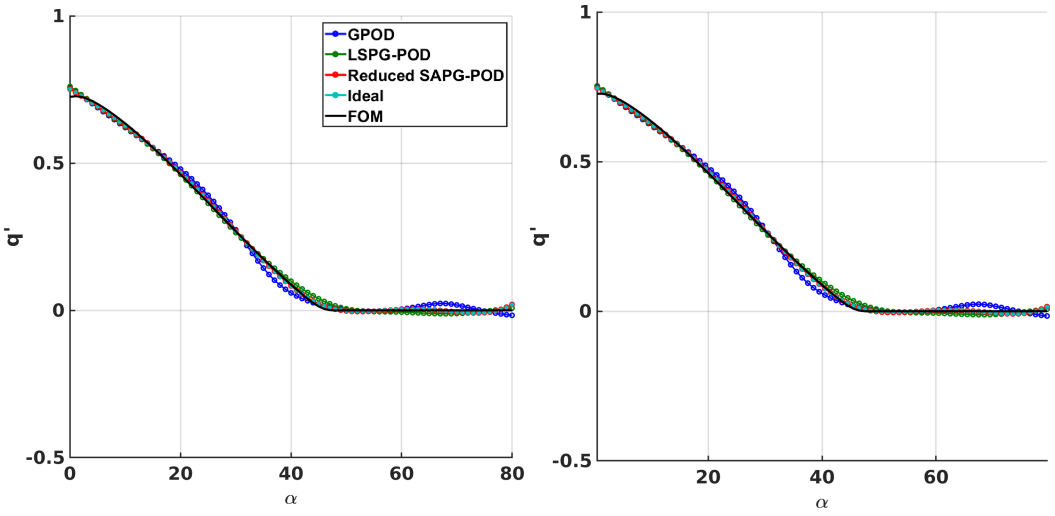
(a) $\alpha = 0°$ **(b)** $\alpha = 40°$ **(c)** $\alpha = 80°$

Figure 4.10: Solutions to the nonlinear test problem.



(a) Output of FOM and GPOD **(b)** POD Singular Values

Figure 4.11: GPOD results and singular values.



(a) Reconstructed Solutions **(b)** Predicted Solutions

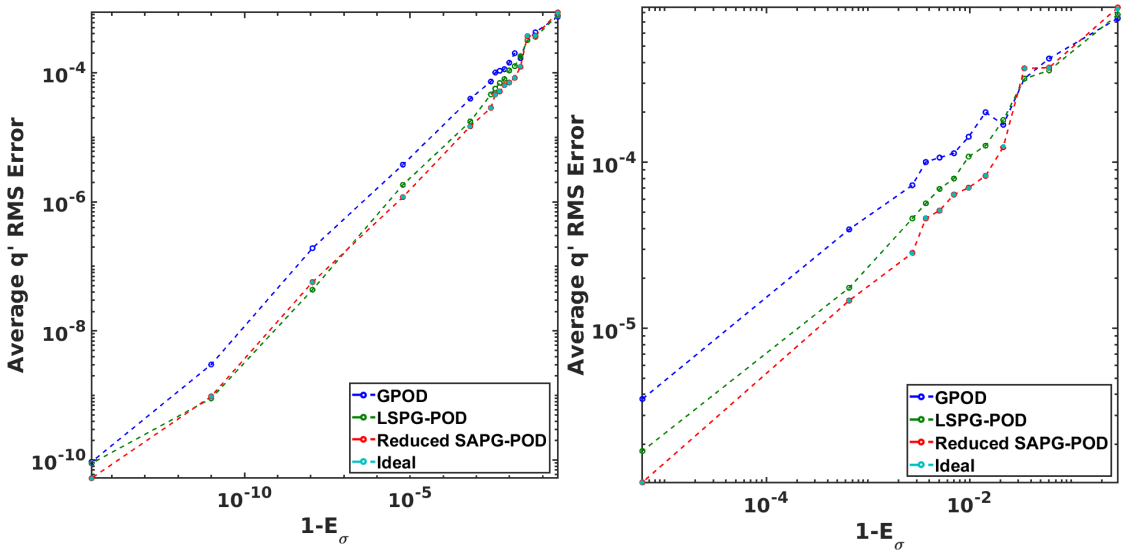
Figure 4.12: Reconstruction and predicted results for $n = 5$ models.

The error of the ROMs here are slightly different than in the linear case, in a comparative sense. From Figure 4.13, there is a slight discrepancy between the *ideal* solution and the reduced SAPG solution. This is even true for smaller ROMs, where a large amount of information can be introduced from the test search space. Overall, in this nonlinear case, the reduced SAPG-POD model performs better in comparison to the GPOD model and the LSPG-POD models than in the linear example in Section 4.2.1. All of the reduced SAPG-POD models performed better than the LSPG-POD and the GPOD models within a range of singular value energies often used in an industrial setting ($99.9\% - 99.99\%$) in both prediction and reconstruction. These improvements are typically an order of magnitude greater than the GPOD model and 1-4 times greater than the LSPG-POD model. This is similar to the results seen in the previous section. Likewise, for prediction, the reduced SAPG-POD model loses fidelity with the *ideal* solution as the trial space size approaches the test search space size, becoming equivalent to the GPOD model once the test search space equals the trial space.

The answer to why the SAPG-POD model performs better than the LSPG-POD model for the nonlinear problem than for the linear problem lies in the construction of the test space. One may think that the LSPG-POD model and the SAPG-POD model would have a similar outcome but from different mechanisms, the logic being that by minimizing the state reconstruction error, one should expect that the full-order residual would also be minimized. Likewise, minimizing the full-order residual would also minimize the state reconstruction error. However, for nonlinear problems, neither is necessarily true, and as a result, the state solutions will differ. The proceeding section demonstrates the application of the SAPG-POD and SAPG-DEIM models in an unsteady setting and makes comparisons with the LSPG and Grassmann manifold generated POD models.

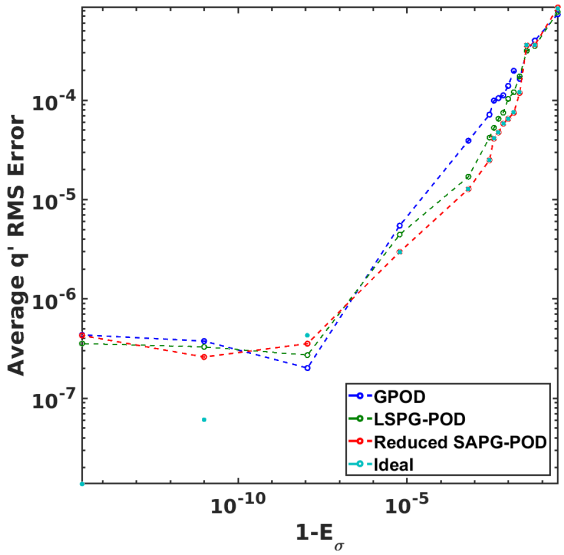
4.2.3 Unsteady Pitching and Plunging Airfoil

The pitching and plunging airfoil from Section 3.5.3 is a multiparameter problem that is used in this section to demonstrate the SAPG method, to make comparisons between the LSPG test space and solutions using the Grassmann manifold interpolated local ROMs, and to demonstrate the use of the hyper-reduced version of the SAPG method. This example utilizes the same unsteady pitching and plunging airfoil from Section 3.5. The pitching and plunging of a NACA 0012 airfoil is sampled at different reduced frequencies and Mach numbers. The choice of parameters creates a problem whose output varies nonlinearly with respect to its sampled parameters. As discussed before, global ROMs are poor candidates for multiparameter, nonlinear systems due to the inclusion of flow characteristics in the state basis that may not be relevant for the testing configuration being solved. Additionally,

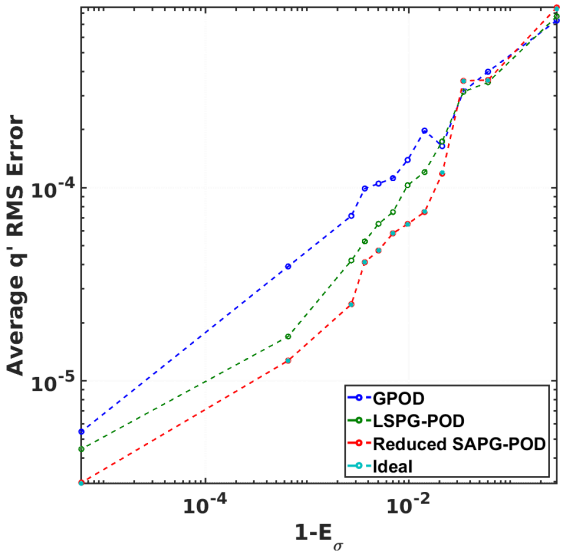


(a) Reconstruction: Full Domain

(b) Reconstruction: Partial Domain

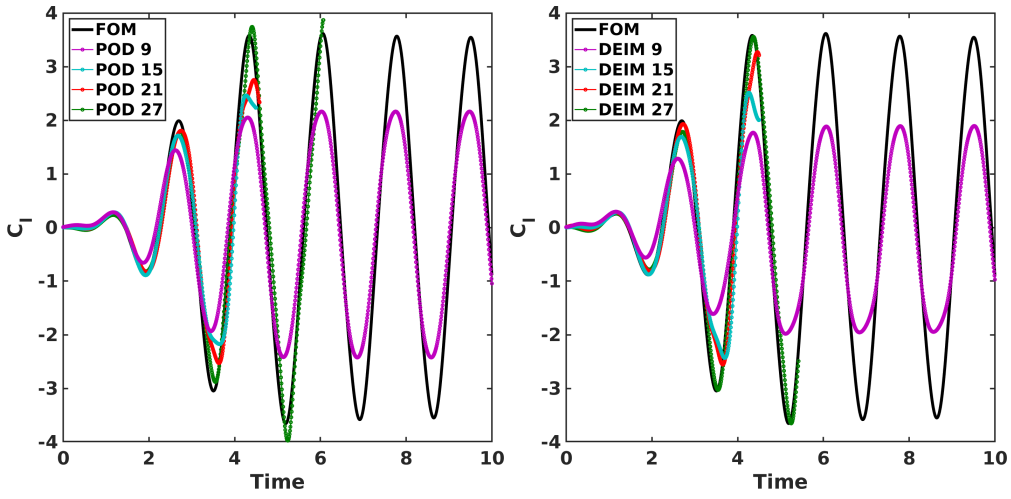


(c) Prediction: Full Domain



(d) Prediction: Partial Domain

Figure 4.13: Errors of the *Ideal*, GPOD, SAPG-POD, the reduced SAPG-POD, and LSPG-POD reconstruction and prediction for nonlinear problem with respect to $1-E_\sigma$. The ROM size decreases moving left to right.



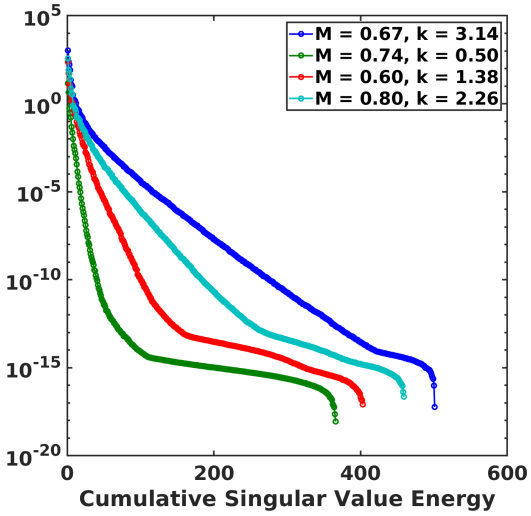
(a) POD **(b)** DEIM

Figure 4.14: Solutions from global POD and DEIM models for the testing configuration.

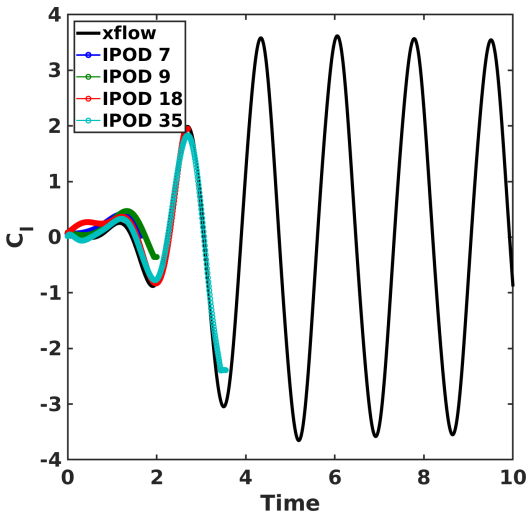
projection-based ROMs tend to grow in stiffness when their size increases, leading to a trade-off between accuracy and stability. This is apparent in the solutions in Figure 4.14 from Section 3.5.3, where only the coarsest, most inaccurate global projection-based ROMs are able to remain stable throughout the simulation.

A typical remedy for this robustness issue is to locally construct state bases at the training configurations with only relevant snapshots and interpolating state bases on the Grassmann manifold at the testing configurations [125]. However, choosing the weights for the Grassmann manifold interpolation is not straightforward. For multiparameter problems, one needs to address the relative strength those individual parameters have on the interpolation and the size of the sphere of influence that local ROMs have in the space of parameters that are used for interpolation.

For comparison, local ROM interpolation produced a Galerkin ROM for use on the testing configuration. The distribution of the singular values of the local state bases had a greater dependency on the reduced frequency than the Mach number, as shown in Figure 4.15a. Due to this dependency, the local ROMs differed greatly on the number of basis vectors required to satisfy the singular value energy criterion shown in Equation (2.13). For 99.9% of the cumulative singular value energy, the highest frequency configuration has a local basis composed of 18 basis vectors while the lowest frequency configuration has a local basis that contains only 7 basis vectors. Given that the testing configuration lies at the mean of the training configurations, a naïve approach is taken, where all of the local ROMs were weighted equally and normalized such that the weights had a unit L2 norm. Four different sized interpolated local ROMs were constructed from these configurations. The size



(a) Singular value energy distribution for the locally constructed state bases.



(b) Solutions from locally interpolated ROMs.

Figure 4.15: The first plot contains singular value energy content of the local POD bases which displays a dependency on motion frequency. The second plot displays Galerkin ROM solutions using Grassmann manifold interpolation of the local bases to construct the online state basis. The weights for interpolation were based on normalized distance of in the parameter space.

for each corresponded to the state basis rank requirement for the highest frequency local ROM basis to have 99%, 99.5%, 99.9%, and 99.99% of the singular value energy, which corresponds to 7, 9, 18, and 35 basis vectors, respectively. Unfortunately, each of these configurations was unable to generate a full solution for the testing problem, as can be seen in Figure 4.15b where the trajectory of the coefficient of lift for each of these interpolated bases is shown. It should be noted that although the naïve approach to weighting the local ROMs is unable to generate a test basis, there may be a better approach to weighting the system that may produce a more robust test basis.

LSPG-POD solutions were generated with various sized state bases. As can be seen in Figure 4.16, none of these configurations was able to produce a test space that yielded a system that remained stable throughout the simulation duration. Additionally large amplitude and frequency errors pollute the solutions.

The SAPG-POD and SAPG-DEIM methods were applied to the system with 99% of the singular value energy contained in the state basis. The solutions in Figure 4.17 demonstrate that unlike the Grassmann manifold interpolation and LSPG test space methods, the SAPG method is able to produce a stable and accurate solution to this problem. As noted in the preceding chapter, given the high singular value energy content retained in the POD process, the state basis should be sufficient to approximate the solution space fairly well. However, a Galerkin projection lacks the dynamics to drive the reduced solution towards

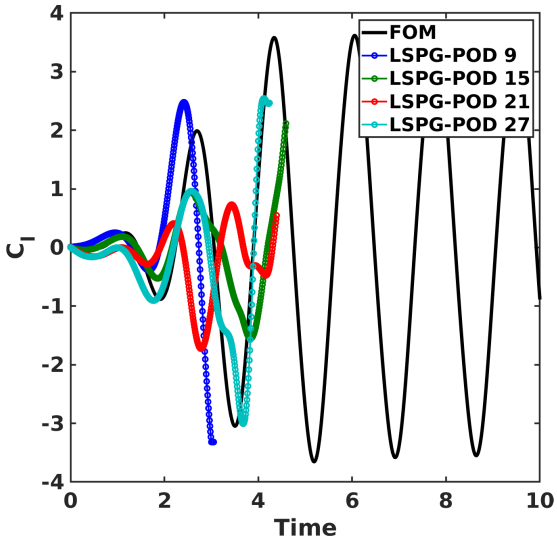


Figure 4.16: LSPG test space solutions.

the ideal solution. The SAPG method extracts those dynamics through the formulation of the test space such that the state error and residual error converges in tandem. This creates a form of adaptation that takes place online, in contrast to the adjoint-weight residual error estimation driven adaptation that adapts the system only *a posteriori*.

4.2.4 XRF1 with Unsteady Deformations

Although the examples contained above demonstrate unique and novel capabilities of the SAPG method for generating solutions that resemble the ideal solution for a particular state basis, it is important to discuss the current limitations of this method, and when the SAPG method is applied to the XRF1 with the unsteady plunging example from Section 3.5.2 these issues are apparent. The reduced formulations presented in Equation (4.45) and Equation (4.46) are necessary to keep the cost of the SAPG method feasible for model reduction, but the reduced formulations may not accurately produce the SAPG test space if the test search space does not adequately span the reduced state adjoint space. In these situations, the SAPG solution may become very erroneous.

For this demonstration, the SAPG test space for the 99% singular value energy POD-ROM was constructed using the reduced formulation with the 99.99% singular value energy POD basis as the test search space. The test space was updated with each iteration of the linear solver for the reduced solution at each time step. Figure 4.18a shows that the lift of this reduced SAPG solution diverges from the FOM lift trajectory significantly; additionally, Figure 4.18b shows the average error of the SAPG test space at the final sub-iteration

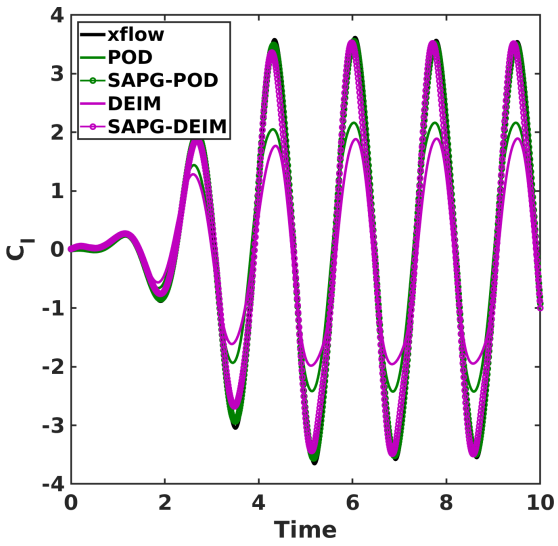


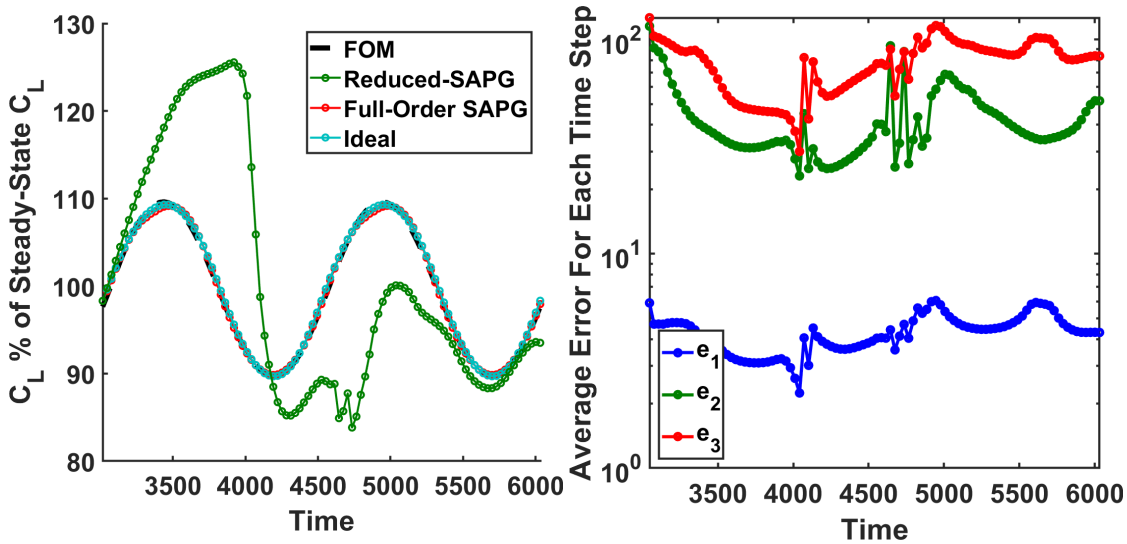
Figure 4.17: SAPG test space solutions.

at each time step. This error is computed with

$$e_{i,j} = \frac{||\frac{\partial \bar{\boldsymbol{R}}}{\partial \boldsymbol{x}}\Big|_{\hat{\boldsymbol{x}}^n}^T \boldsymbol{w}_{i,j} - \boldsymbol{v}_i||_1}{n_{\boldsymbol{x}}}, \tag{4.74}$$

where $e_{i,j}$ is the average L1-norm of the error of the test space basis vector corresponding to the state basis vector $\boldsymbol{v}_i$ at the j^{th} time step of the solution, $\boldsymbol{w}_{i,j}$ is corresponding the test basis vector, and $n_{\boldsymbol{x}}$ is the number of FOM degrees of freedom. For comparison, a SAPG solution is developed with test spaces obtained with the full-order formulation and only at the initial sub-iteration of each time step. This model performs significantly better than the reduced SAPG model using a finer set of basis vectors and stays fairly faithful to the ideal solution, which was obtained by projecting the true solution with the 99% singular value energy state basis. This illustrates that the quality of the test space constructed in the reduced SAPG formulation can have a significant effect on the quality of the ROM solution. The quality of the reduced test space is primarily influenced by the test search space, and the finer set of POD state basis vectors may not be the best choice for the test search space.

An improved reduced SAPG solution can be obtained if the test search space is better suited for representing state adjoints. Snapshots of the reduced state adjoints were used for constructing a basis for the SAPG test search space. The reduced state adjoint snapshots arise from projecting the snapshots of the full-order model and solving Equation (4.15). These snapshots are then decomposed into the test search space basis that contains 99.99% of the singular value energy of decomposition of the adjoint snapshot set. As shown in Figure 4.19a, the reduced SAPG solution improves dramatically; additionally, Figure 4.19b



(a) SAPG Solutions

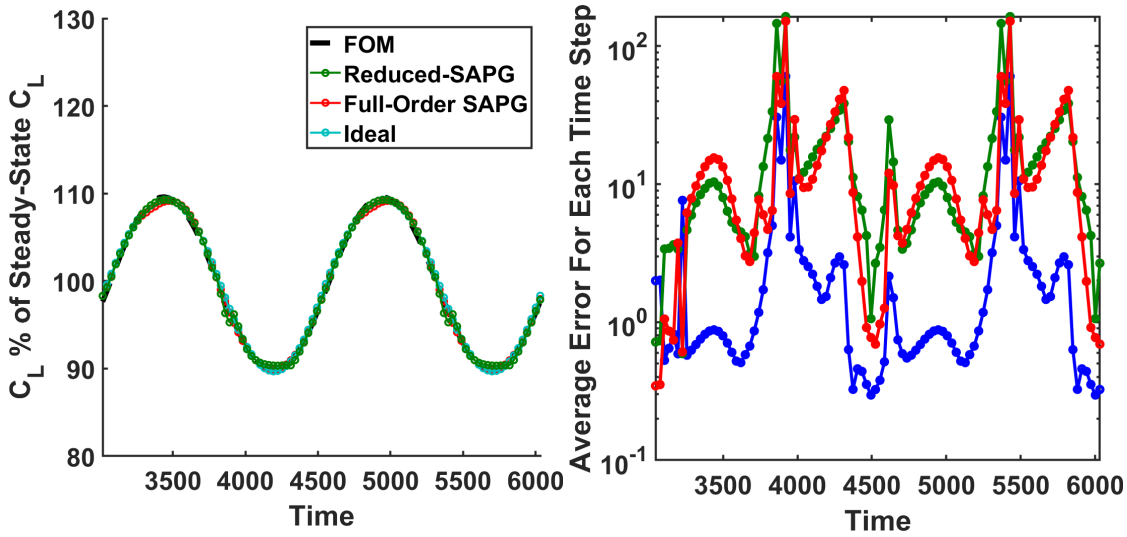
(b) Average Error

Figure 4.18: ROM solutions using the SAPG test space with the reduced and full-order formulation for the low-frequency configuration. The high test space errors for the reduced formulation and corresponding high solution error are indicative of the effects of a poor test search space.

shows that the errors for each test space basis vector decrease by an order of magnitude for most of the time steps. The time steps when the error of the SAPG test basis vectors increased correspond to time steps in the solution where the SAPG model diverges momentarily from the FOM trajectory. The cost of generating the snapshots of the reduced state adjoints increases the cost of the offline stage significantly, as each reduced state adjoint requires solving a linear system of the same size as the FOM; however, the improvement to the solution may justify using an adjoint basis for generating the SAPG test space if the error of SAPG test space constructed from a fine set of basis vectors is too high.

4.3 Conclusion

This chapter demonstrated a novel approach to generating Petrov-Galerkin reduced order models by using reduced state adjoints of the ROM system. The resulting ROMs drive the reduced solutions towards their *ideal* values, which are the solutions from the FOM projected with the state basis used in the projection-based ROM. These test bases are developed for both POD and hyper-reduced DEIM models, and examples shown in this chapter demonstrate advantages in accuracy yielded by these test bases over the more commonly used least-squares Petrov-Galerkin methods and Grassmann manifold interpolation of local ROMs. However, the increases in accuracy come at the cost of increases in computation, as each of the reduced state adjoints requires a full-order sized linear system to be solved at each iteration of the ROM. To remedy this additional computational cost, reduced models



(a) SAPG Solutions

(b) Average Error

Figure 4.19: ROM solutions using the SAPG test space with the reduced and full-order formulation for the low-frequency configuration. The reduced SAPG test space is constructed with a state adjoint basis.

of the state adjoint Petrov-Galerkin test bases were developed that seek the solution of the test bases as a linear combination of another basis called the test search space. For many problems, the finer set of state basis vectors that were truncated is sufficient to serve as a test search space; however, if the test bases is not well-represented by the finer set of state basis vectors, the ROM solution will be inaccurate and a different test search space should be used. For example, snapshots of the reduced state adjoints can be decomposed to form a more accurate test search space, which results in a more accurate ROM, as demonstrated with the XRF1 example. Finally, an examination of the stability, convergence, and cost of the SAPG test space was presented and yielded interesting contrasts between Galerkin ROMs and the LSPG method. The eigenvalues arising from the use of Fourier and Legendre bases on an advective-diffusive problem for the LSPG and SAPG methods had similar magnitudes, but differed in distribution; implying similar stability behavior but differing dynamics. When applied to the Newton-Rapshon method for obtaining a solution to a time step in a nonlinear, unsteady model, LSPG produces the solution of the L2 minimization of the problem, while SAPG produces the projection of the equivalent full-order model solution to the problem. Estimations of the computational complexity of projection-based ROMs, LSPG, and SAPG reveal that the cost of LSPG will be approximately twice that of GROMs, and the cost of SAPG will be even greater than the cost of LSPG. The additional cost of SAPG is dependent on the number of basis vectors used for the test search space as shown by Equation (4.68) and Equation (4.69), which can be minimized by using

a basis spanning the reduced state adjoints but at the cost of solving additional full-order sized snapshots offline. However, when considering the trade-off between state accuracy and ROM cost, SAPG yields more accurate solutions than LSPG solutions at the same cost.

Applications of SAPG to larger scale problems are of interest, but many factors need to be considered, including the scaling of the cost of SAPG with increasing problem size and the stability and accuracy of SAPG for singular value energy deficient problems. Neglecting the cost of residual and Jacobian computation, the computational costs in terms of FLOPs of constructing the SAPG model using the reduced formulation are inherently larger than the costs of constructing the Galerkin ROM and LSPG models; however, given that the leading order of the cost of forming and solving Galerkin ROMs, LSPG, and SAPG are all quadratic with respect to the dimension of the FOM, their scaling to larger problems should be similar and the added costs may be worth the additional accuracy. The state basis for larger-sized problems may not necessarily have a high singular value energy content due to the larger number of snapshots and the richness of the solution space. Favorably though, no assumptions of the state trial basis is made with the derivation of the SAPG test basis; further, the linear and nonlinear scalar transport problems in this chapter demonstrate that the SAPG test basis is able to arrive at the ideal solution even with rank one state bases. Thus its application to singular value energy deficient problems should be possible; however, an investigation into such applications is needed.

CHAPTER 5

Element-Embedded Neural Networks for DEIM Hyper-reduction

In the introduction of this dissertation, two categories for characterizing model reduction techniques are discussed: interpolation-based, where a "black box" model is used to map relevant inputs to relevant outputs, and projection-based, where the FOM model is projected onto a tailored subspace to efficiently reduce the degrees of freedom required for resolving the system. Making this distinction is useful as the models in these broad categories often share approaches, arise from common scientific fields, and have similar benefits and trade-offs. However, the hybridization of interpolation-based and projection-based models is a growing branch of model reduction. This chapter introduces a hybridization of machine learning surrogates and projection-based techniques that enable for a non-intrusive approach to projection-based model reduction. This method, the element-embedded neural network for DEIM hyper-reduction (EENN-DEIM), uses neural-network models to replace the intrusive portions of the DEIM method – *i.e.*, the formulation of the CFD residual. Following the derivation of the EENN-DEIM model, a discussion of approaches for Jacobian construction is presented and comparisons of the traditional DEIM and EENN-DEIM methods on example applications are made.

5.1 Approaches of Non-intrusive Projection-based ROMs

The intrusive implementation of many projection-based techniques is a major drawback to their use. Often, projection-based models are designed for specific systems and solvers, limiting their portability. Even so, many of the generic libraries for model reduction still require FOM code modification to access the specific physics model that is being reduced. Further, barriers to modifying the FOM model may make implementing projection-based ROMs difficult or even impossible, such as intellectual property laws, export control restrictions, and commonly the difficulty of working on legacy codes and on codes initially designed without the intention of being used with ROM techniques. To circumvent the

modification of the FOM code, non-intrusive ROMs have been develop that use machine learning to replaces the need for FOM subroutines.

A common approach for non-intrusive ROMs is to map system inputs (such as time, geometric configurations, and physical conditions) to outputs or POD coefficients in distribution and field approximations, which then can be expanded to obtain high-fidelity solutions. In this fashion, the POD portion is more similar to unsupervised learning than it is to a projection-based ROM as the POD basis is used to compress features of the system while the underlying physical equations are not invoked. For example, Renganathan, Maulik, and Rao used deep feed-forward neural networks to predict POD solutions for a steady, shape-parameterized airfoil problem [70]. Two approaches for parameterized unsteady flows by San, Maulik, and Ahmed apply recurrent neural-network models to map instantaneous POD coefficients and system parameters to the POD coefficients at the next time node and also the change in the POD coefficients from the input to the next time node [71]. This technique was demonstrated with a single-hidden layer recurrent neural network for predicting solutions for a 2D viscous Burgers equation model. Many other exercises exist with the common approach of developing a linear trial space of the state solutions, training artificial/recurrent neural networks to map system parameters to the reduced state solution, and then deploying the neural networks for prediction [72, 73, 74, 75]. Instead of mapping system inputs to coefficients for a linear POD basis, Fresca, Dede, and Manzoni used a two step process, where a convolutional autoencoder mapped the outputs of a deep, feed-forward neural network to the full-order state [76]. The use of autoencoders as a nonlinear trial manifold is based on work by Lee and Carlberg, although their formulation still requires access to the FOM model [80]. Peherstorfer and Willcox used unsupervised learning for operator inference in order to construct the linear and quadratic operators for a POD-ROM model [77]. In their method, snapshots of the FOM are used to construct the state basis, the POD-ROM is assumed to be quadratic, and the minimization of the quadratic POD-ROM residual over all of the reduced states is used to construct the POD-ROM operators. Khodabakhshi and Willcox are able to apply operator inference to nonlinear, non-quadratic problems by reformulating the FOM partial differential equations into partial differential algebraic equations with the addition of non-conserved variables, like pressure, in the ROM state – a technique called polynomial lifting [79].

In the same vein, this chapter introduces a non-intrusive ROM method, the element-embedded neural network for DEIM hyper-reduction (EENN-DEIM), that uses machine learning but not for predicting POD coefficients. Time marching of a physical system is important in order to allow for the dissipation of errors and the appropriate propagation of the state. The application of neural networks for directly predicting solutions for unsteady

systems requires careful construction of the model in order for proper information propagation. Recurrent neural networks (RNNs) use a feedback of an outputted hidden state in order to propagate information through time; however, these systems are susceptible to the vanishing and exploding gradient phenomena, where the influence of the inputs and hidden state of a time node quickly diminishes or the hidden state of the system grows non-physically large [126]. Certain RNN formulations, such as the long term short term memory unit [126], can alleviate some of these issues but have many limitations such as a fixed time step size, sensitivity to memory length, and the need of initial memory accumulation. Rather, the EENN-DEIM model only uses neural networks to map the relevant full-state information to residuals at the DEIM sampling indices, and the gradients of the networks are used to construct the Jacobian of the system. These values are then used in the DEIM-ROM which supplants the need for FOM subroutines. In this fashion, the time marching of the state is performed appropriately through the same time discretization and mass matrix, and errors in the system are allow to dissipate through the minimization of the neural-network predicted residuals. The only requisite data needed from the FOM model are the state and residual vectors, which are already supplied to construct the POD and DEIM model. In addition to the non-intrusive and portable nature of this method, the EENN-DEIM model has a substantial speedup over the traditional DEIM model. The proceeding sections discuss supervised machine learning and the EENN-DEIM method and provide example applications.

5.2 Supervised Machine Learning

Machine learning concerns the application of computational methods to accelerate the use of mathematical techniques that discover structure and features of an underlying system. There are three major classes of machine learning problems: unsupervised learning, supervised learning, and reinforcement learning. Of these three, this chapter is concerned with developing supervised learning models for use in projection-based models. Supervised learning attempts to construct a sophisticated mapping of the inputs to outputs of interest of a system. Classic applications include computer vision, handwriting analysis, and facial recognition. The learning is "supervised" in that the user provides known input-output pairs to train a surrogate model while monitoring a loss function based on the accuracy of the model. Once trained, the model can be deployed on untrained inputs to generate predicted outputs. The model can take many forms; however a common choice for many static problems is a feed-forward neural network, which is a layered mathematical struc-

ture, according to

$$g_{\text{net}}(\boldsymbol{x}_{\text{NN}} \rightarrow \boldsymbol{y}_{\text{NN}}) = L_o \circ L_{o-1} \circ ...L_j... \circ L_1(\boldsymbol{x}_{\text{NN}}), \tag{5.1}$$

where $\boldsymbol{x}_{\text{NN}}$ are the inputs to the network, $\boldsymbol{y}_{\text{NN}}$ are the outputs of the network, L_j is the j^{th} layer of the neural network, and $\cdot \circ \cdot$ indicates the composition of the layers where the output of one layer is the input of another. Each layer is a gated linear system, defined by

$$L_j(\boldsymbol{x}_{\text{NN}j} \rightarrow \boldsymbol{x}_{\text{NN}j+1}) : \boldsymbol{x}_{\text{NN}j+1} = \sigma_j\left(\boldsymbol{z}_{\text{NN}j}\right), \tag{5.2}$$

$$\boldsymbol{z}_{\text{NN}j} = \boldsymbol{W}_j \boldsymbol{x}_{\text{NN}j} + \boldsymbol{b}_j, \tag{5.3}$$

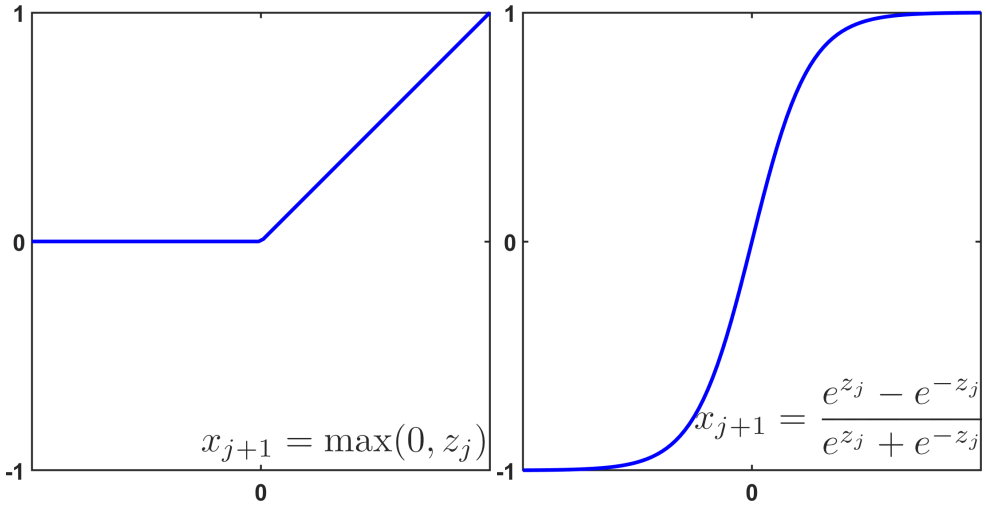
where σ_j is the j^{th} nonlinear function gate – called an activation function – $\boldsymbol{z}_{\text{NN}j}$ is the output of the linear stage of the network, and $\boldsymbol{W}_j$ and $\boldsymbol{b}_j$ are the weights and biases of the linear portion of the network. The weights and biases are defined by

$$\boldsymbol{W}_j = \begin{bmatrix} w1,1 & \dots & w_{1,k_j} \\ & \ddots & \\ wk_{j+1},1 & \dots & w_{k_{j+1},k_j} \end{bmatrix} \in \mathbb{R}^{k_{j+1} \times k_j}, \tag{5.4}$$

$$\boldsymbol{b}_j = \begin{bmatrix} b_1 \\ \vdots \\ b_{k_{j+1}} \end{bmatrix} \in \mathbb{R}^{k_{j+1}}. \tag{5.5}$$

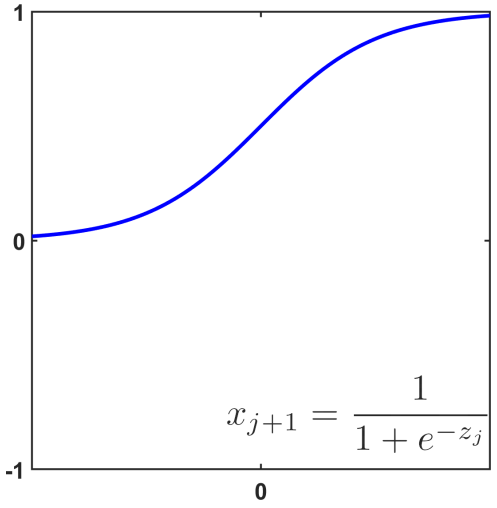
The size of the inputs of a layer (k_j) is used to characterize the size of the layer – also referred to as the number of neurons. Activation functions map scalar inputs to scalar output values, usually bounded by $[0, 1]$ or $[-1, 1]$ for numerical accuracy. Common activation functions include the sigmoid activation function, hyperbolic tangent function (TanH), rectified linear unit (ReLu), and a Gaussian distribution activation function, which are displayed in Figure 5.1. With the exception of the output layer, activation functions are typically used as gates on layers and add nonlinearity to the neural network. Figure 5.2 displays a single hidden layer feed-forward neural network. For clarity, the output layer is the "highest" layer, the input layer is the "lowest layer", and the closer a layer is to the input layer the "deeper" that layer is.

Training is required for the neural network to be accurate. Although many different training algorithms exist, they all require the minimization of some loss function which measures the error of the output predictions of a network on a given set of training input-output pairs. As an example, the squared L2-norm of the error of a network is defined

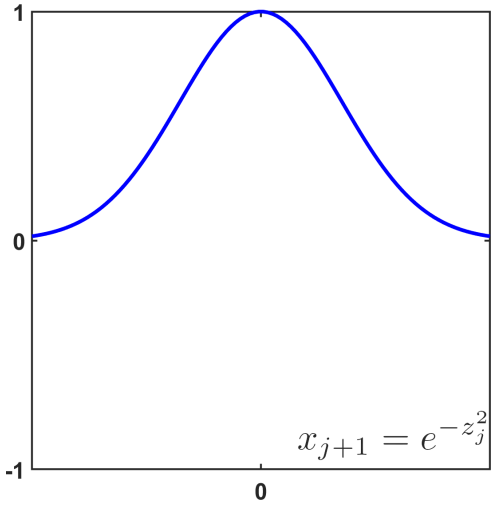


(a) ReLu

(b) TanH



(c) Sigmoid



(d) Gaussian

Figure 5.1: Examples of activation functions.

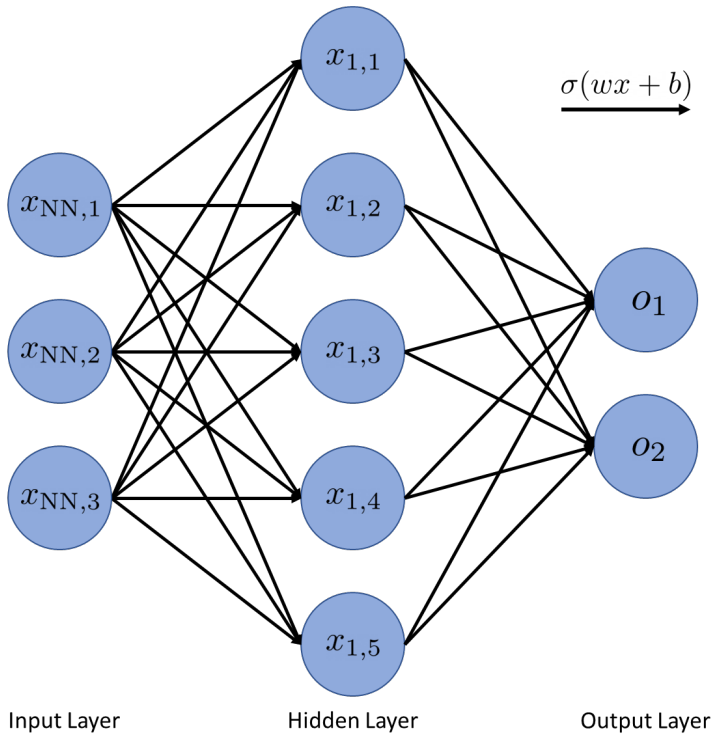


Figure 5.2: Single hidden layer neural network. Each unit contains a scalar, and every arrow is a composition of a linear operation and nonlinear activation function.

by

$$J(g(\boldsymbol{x}_{\text{NN}t}), \boldsymbol{y}_{\text{NN}t}) = ||\boldsymbol{y}_{\text{NN}t} - g(\boldsymbol{x}_{\text{NN}t})||_2^2,$$
$$J(g(\boldsymbol{x}_{\text{NN}t}), \boldsymbol{y}_{\text{NN}t}) = \sum_{i=1}^{N_t} (\boldsymbol{y}_{\text{NN}t,i} - g(\boldsymbol{x}_{\text{NN}t,i}))^T (\boldsymbol{y}_{\text{NN}t,i} - g(\boldsymbol{x}_{\text{NN}t,i})), \tag{5.6}$$

where $(\boldsymbol{x}_{\text{NN}t}, \boldsymbol{y}_{\text{NN}t})$ are a set of N_t known input-output pairs. Minimization of the loss function can be done iteratively by updating the weights and biases of the network with a gradient descent based update, defined by

$$\boldsymbol{W}_{i,\text{new}} = \boldsymbol{W}_{i,\text{old}} - \epsilon_l \nabla_{\boldsymbol{W}_{i,\text{old}}} J(g(\boldsymbol{x}_{\text{NN}t}), \boldsymbol{y}_{\text{NN}t}), \tag{5.7}$$
$$\boldsymbol{b}_{i,\text{new}} = \boldsymbol{b}_{i,\text{old}} - \epsilon_l \nabla_{\boldsymbol{b}_{i,\text{old}}} J(g(\boldsymbol{x}_{\text{NN}t}), \boldsymbol{y}_{\text{NN}t}), \tag{5.8}$$

where ϵ_l is the learning rate (*i.e.*, the step size) of the update. The gradient of the loss

function with respect to the network parameters is formulated with the chain-rule, *i.e.*,

$$\nabla_{\boldsymbol{W}_{i,\text{old}}} J(g(\boldsymbol{x}_{\text{NN}t}), \boldsymbol{y}_{\text{NN}t}) = \frac{\partial J(g(\boldsymbol{x}_{\text{NN}t}), \boldsymbol{y}_{\text{NN}t})}{\partial g} \frac{\partial g(\boldsymbol{x}_{\text{NN}})}{\partial \boldsymbol{W}_{i,\text{old}}}, \tag{5.9}$$

$$\nabla_{\boldsymbol{b}_{i,\text{old}}} J(g(\boldsymbol{x}_{\text{NN}t}), \boldsymbol{y}_{\text{NN}t}) = \frac{\partial J(g(\boldsymbol{x}_{\text{NN}t}), \boldsymbol{y}_{\text{NN}t})}{\partial g} \frac{\partial g(\boldsymbol{x}_{\text{NN}})}{\partial \boldsymbol{b}_{i,\text{old}}}. \tag{5.10}$$

Likewise, the gradients of the network with respect to the weights and biases are formed using the chain-rule; furthermore, the application of the chain-rule results in the recursive formation of the gradients of the weights and biases, as portions of the chain-rule used to form the gradient for a higher layer are required for the chain-rule used to form the gradients for deeper layers. For a given layer j of an o deep network, the gradients of the network with respect to its weights and biases are

$$\frac{\partial g}{\partial \boldsymbol{W}_j} = \left[\prod_{i=o}^{i=j+1} \left[\text{diag} \left(\frac{d\boldsymbol{x}_{\text{NN}i+1}}{d\boldsymbol{z}_{\text{NN}i}} \right) \boldsymbol{W}_i \right] \text{diag} \left(\frac{d\boldsymbol{x}_{\text{NN}j+1}}{d\boldsymbol{z}_{\text{NN}j}} \right) \right] \frac{d\boldsymbol{z}_{\text{NN}j}}{d\boldsymbol{W}_j}, \tag{5.11}$$

$$\frac{\partial g}{\partial \boldsymbol{b}_j} = \left[\prod_{i=o}^{i=j+1} \left[\text{diag} \left(\frac{d\boldsymbol{x}_{\text{NN}i+1}}{d\boldsymbol{z}_{\text{NN}i}} \right) \boldsymbol{W}_i \right] \text{diag} \left(\frac{d\boldsymbol{x}_{\text{NN}j+1}}{d\boldsymbol{z}_{\text{NN}j}} \right) \right] \mathbf{1}_{k_{j+1}}, \tag{5.12}$$

where

$$\left(\text{diag} \left(\frac{d\boldsymbol{x}_{\text{NN}i+1}}{d\boldsymbol{z}_{\text{NN}i}} \right) \right)_{k,l} = \delta_{k,l} \frac{d(\boldsymbol{x}_{\text{NN}i+1})_k}{d(\boldsymbol{z}_{\text{NN}i})_l}, \tag{5.13}$$

$$\left(\frac{d\boldsymbol{z}_{\text{NN}j}}{d\boldsymbol{W}_j} \right)_{k,l} = \left(\boldsymbol{x}_{\text{NN}j} \right)_l, \tag{5.14}$$

$\delta_{k,l}$ is the Kronecker delta, and $\mathbf{1}_{k_{j+1}}$ is a k_{j+1} length ones vector. The gradient for the weights and biases of the $j-1$ layer is the same as Equation (5.11) and Equation (5.12) but with an extra term in the product. Programmers can take advantage of this observation by writing *automatic differentiation* algorithms that recursively compute the gradients of the system layer by layer, which is computationally more efficient than the naïve implementation. Further carrying out this product to the deepest layer yields the derivative of the network with respect to its inputs, defined by

$$\frac{\partial g}{\partial \boldsymbol{x}_{\text{NN}}} = \prod_{i=o}^{i=1} \left[\text{diag} \left(\frac{d\boldsymbol{x}_{\text{NN}i+1}}{d\boldsymbol{z}_{\text{NN}i}} \right) \boldsymbol{W}_i \right]. \tag{5.15}$$

The next section details the use of automatic differentiation for computing system Jacobians. These system Jacobians are then used in the DEIM algorithm for constructing Jaco-

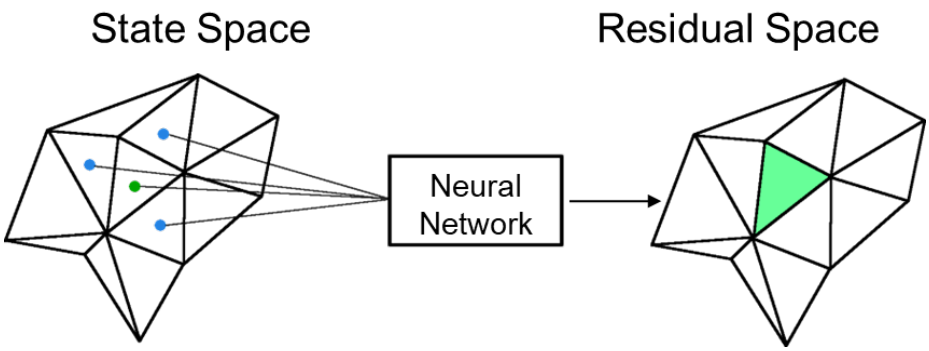


Figure 5.3: EENN-DEIM model applied to a mesh. The key idea of the EENN-DEIM model is to use the states of the embedded element and the neighboring elements to formulate the residuals instead of invoking the FOM code.

bian approximations. There is a wealth of literature dedicated to artificial neural-network design, training, and applications; however, the amount covered in this section is sufficient for the proceeding introduction of the element-embedded neural networks for DEIM hyper-reduction.

5.3 Element-Embedded Neural Networks

As mentioned earlier, sparse sensing based hyper-reduction ROMs have two major practical implementation issues: they are highly intrusive and their coding frameworks are typically solver exclusive. For projection-based models in general, access to state, the FOM functions for computing the residual and Jacobian, and additional subroutines for constructing the deploying the ROMs are necessary. DEIM sparsely samples the residual at specific degrees of freedom in order to interpolate the residual everywhere. The subroutines that compute the residual for a dispersed set of degrees of freedom often need to be written for FOM codes that were originally not designed for model reduction. These unique codes make projection-based techniques tied to the FOM solver that is being reduced, and changing the FOM solvers may require re-writing of codes. Additionally, there are many barriers to accessing FOM codes to begin with: governmental classifications, intellectual property protections, export controls, and even legacy or very complex code structure obscuring the implementation.

However, the implementation of DEIM can be non-intrusive and portable with the use of neural networks for residual and Jacobian computation. The element-embedded neural network for DEIM hyper-reduction uses individual neural networks at each DEIM sampling element and maps internal and neighboring state data to internal spatial residuals, as demonstrated in Figure 5.3. Recalling from Section 3.2, the spatial residual is computed by balancing the convective and diffusive fluxes. These fluxes for an element are often only

functions of the neighboring states, internal state, and boundary conditions. Thus a surrogate model with the same inputs can be used to replace the FOM residual computations. For the EENN-DEIM method, the approximation of the sparsely sampled residual is

$$\boldsymbol{R}_{\text{EENN}} = \begin{bmatrix} g_{\text{DEIM},1}(\boldsymbol{Q}_1\hat{\boldsymbol{x}}, \boldsymbol{\mu}(t)_1, t) \\ \vdots \\ g_{\text{DEIM},n_{\boldsymbol{P}}}(\boldsymbol{Q}_{n_{\boldsymbol{P}}}\hat{\boldsymbol{x}}, \boldsymbol{\mu}(t)_{n_{\boldsymbol{P}}}, t) \end{bmatrix} \approx \boldsymbol{P}^T\boldsymbol{R}(\boldsymbol{x}, \boldsymbol{\mu}, t), \tag{5.16}$$

where $g_{\text{DEIM},i}$ is the i^{th} neural network, $\boldsymbol{\mu}(t)_i$ are element specific inputs such as boundary conditions, and

$$\boldsymbol{Q}_i = \boldsymbol{G}_i^T\boldsymbol{V}. \tag{5.17}$$

$\boldsymbol{G}_i$ is similar to $\boldsymbol{P}$ – *i.e.*, the sampling matrix from DEIM – in that it contains elementary vectors indicating the states to keep for the i^{th} neural network. Making the substitution of the neural networks into Equation (2.38) yields

$$\hat{\boldsymbol{M}}\frac{d\hat{\boldsymbol{x}}}{dt} + \boldsymbol{W}^T\boldsymbol{U}\left[\boldsymbol{P}^T\boldsymbol{U}\right]^\dagger \begin{bmatrix} g_{\text{DEIM},1}(\boldsymbol{Q}_1\hat{\boldsymbol{x}}, \boldsymbol{\mu}(t)_1, t) \\ \vdots \\ g_{\text{DEIM},n_{\boldsymbol{P}}}(\boldsymbol{Q}_{n_{\boldsymbol{P}}}\hat{\boldsymbol{x}}, \boldsymbol{\mu}(t)_{n_{\boldsymbol{P}}}, t) \end{bmatrix} = \boldsymbol{0}, \tag{5.18}$$

which is the EENN-DEIM model.

There are two approaches to constructing an approximation to the DEIM Jacobian that are demonstrated in this thesis: finite differencing and analytical automatic differentiation. The straightforward application of finite differencing is to perturb each input of each network to formulate the sensitivity of the EENN residual to each degree of freedom. However, this is inefficient as the number of inputs for a neural network can be high, given that each residual value is constructed from the states of a collection of neighboring elements. For example, for triangular elements with a conforming tessellation, the total number of state inputs is

$$\text{d.o.f.} = 4\frac{(p+1)(p+2)}{2}r_{\text{state}}n_{\boldsymbol{P}}, \tag{5.19}$$

where p is the spatial approximation order, r_{State} is the state rank of the system, and $n_{\boldsymbol{P}}$ is the total number of residuals being computed. For 2D inviscid simulations, the state rank is 4, and Table 5.1 shows the total degrees of freedom that the finite differencing is applied to for each residual degree of freedom sampled in DEIM. Additionally, for central differencing,

two neural-network evaluations are needed, and in order to assess the quality of the gradient approximation, additional finite differences of smaller step sizes may be needed. A more

Spatial Order, p	0	1	2	3
d.o.f./$n_{\boldsymbol{P}}$	16	48	96	160

Table 5.1: The total number of degrees of freedom for an interior element per neural network for the EENN-DEIM applied to a system with inviscid physics, triangular mesh conforming elements, and increasing spatial approximation order.

straightforward approach would be to perturb the reduced state coefficients, then apply the chain-rule. For the l^{th} network, this finite difference is defined by

$$\frac{\partial g_{\text{DEIM},l}(\boldsymbol{x},\boldsymbol{\mu}(t)_k,t)}{\partial \boldsymbol{x}} = \left[\frac{g_{\text{DEIM},i}(\boldsymbol{Q}_k\left(\hat{\boldsymbol{x}}+\boldsymbol{e}_{n_{\boldsymbol{V}},i}\epsilon\right),\boldsymbol{\mu}(t)_k,t)-g_{\text{DEIM},l}(\boldsymbol{Q}_k\left(\hat{\boldsymbol{x}}-\boldsymbol{e}_{n_{\boldsymbol{V}},i}\epsilon\right),\boldsymbol{\mu}(t)_k,t)}{2\epsilon}\right]\boldsymbol{Q}^T\boldsymbol{G}_k^T, \tag{5.20}$$

where the finite difference term is a $1\times n_{\boldsymbol{V}}$ matrix of the finite differences for each reduced state coefficient, $\boldsymbol{e}_{n_{\boldsymbol{V}},i}$ is an elementary vector of length $n_{\boldsymbol{V}}$ with a 1 at the i^{th} index and zeros elsewhere, and ϵ is the step size of the finite difference. The total number of finite differences is now $n_{\boldsymbol{V}}n_{\boldsymbol{P}}$ which may be less than the number of finite differences needed when perturbing the states individually; however, the number of network evaluations is still very high, which impacts the performance of the EENN-DEIM model.

An alternative approach is to use analytical gradients of the network. Equation (5.15) can be used to compute the gradient of the networks with respect to the state inputs. Recognizing that an individual network has zero gradients for any state value that is not used as an input, the gradient of the k^{th} network with respect to the full state is defined by

$$\frac{\partial g_{\text{DEIM},l}(\boldsymbol{Q}_k\hat{\boldsymbol{x}},\boldsymbol{\mu}(t)_k,t)}{\partial \boldsymbol{x}} = \left[\prod_{i=o}^{i=1}\left[\text{diag}\left(\frac{d\boldsymbol{x}_{\text{NN}i+1}}{d\boldsymbol{z}_{\text{NN}i}}\right)\boldsymbol{W}_i\right]\right]\boldsymbol{G}_k^T. \tag{5.21}$$

The product needed for the gradients of the network can be computed while computing the residual of the EENN-DEIM model. Therefore, the gradient formed from automatic differentiation only needs a single network evaluation. This has a substantial effect on the performance of the EENN-DEIM model, which is demonstrated in the example applications.

Collecting the spatial gradients of the entire network system from either finite differencing or an analytical, automatic differentiation yields the spatial Jacobian, which can be

used to approximate the DEIM Jacobian,

$$\frac{\partial \boldsymbol{R}_{\text{EENN}}}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial g_{\text{DEIM},1}(\boldsymbol{Q}_1 \hat{\boldsymbol{x}}, \boldsymbol{\mu}(t)_1, t)}{\partial \boldsymbol{x}} \\ \vdots \\ \frac{\partial g_{\text{DEIM},n_{\boldsymbol{P}}}(\boldsymbol{Q}_{n_{\boldsymbol{P}}} \hat{\boldsymbol{x}}, \boldsymbol{\mu}(t)_{n_{\boldsymbol{P}}}, t)}{\partial \boldsymbol{x}} \end{bmatrix} \approx \boldsymbol{P}^T \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}. \tag{5.22}$$

5.4 Example Applications

Two applications of the EENN-DEIM method are demonstrated in this section: the nonlinear steady example from Section 4.2.2 and the unsteady pitching and plunging airfoil from Section 3.5.3. These solutions give insight to the use and benefits of EENN-DEIM and as well as its limitations.

5.4.1 Scalar Advection-Diffusion with Nonlinear Source

The first application of the EENN-DEIM hyper-reduction is to the scalar advection-diffusion with a nonlinear source example defined in Section 4.2.2. To summarize the problem settings, on a square domain with rectangular elements, the following scalar advection-diffusion equations are solved:

$$\nabla \cdot (\vec{v}x) - \nu \nabla^2 x + S(x) = 0, \tag{4.72}$$

$$S(x) = S_0 x^2, \tag{4.73}$$

where the speed of the advection is $|\vec{v}| = 1.0$, the diffusivity is $\nu = 0.001$, and the source constant is $S_0 = 1.0$. In total there are 2916 elements with bilinear basis for the state. The total degrees of freedom of the problem is 11664. 81 FOM steady solutions with the advection angle sampled evenly within $\alpha \in [0, 80]^\circ$ are taken to form the snapshots for the POD state basis. To formulate the DEIM model, spatial residual snapshots are computed by perturbing the state via projection of the state snapshots with the POD basis vectors and then calculating the difference between the resulting advective and diffusive fluxes. POD state bases with 5, 10, and 15 basis vectors are used to perturb each of the snapshots. To obtain additional snapshots, for each of the state projections, three sets of randomly generated values between $\pm 5\%$ are used to perturb the weights of the projections, and the residual of the perturbed projection is computed. In total 972 residual snapshots are used to obtain a residual basis where only 45 basis vectors are kept for the DEIM approximation. 90 elements from the FOM are sampled to form the DEIM residual. The locations of these elements are shown in Figure 5.4. The half angles of attack between the training samples used to develop the ROMs are used to test the models. The DEIM-ROM solutions for

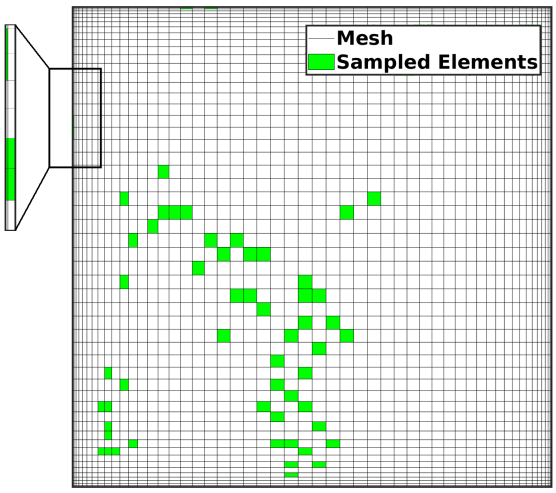
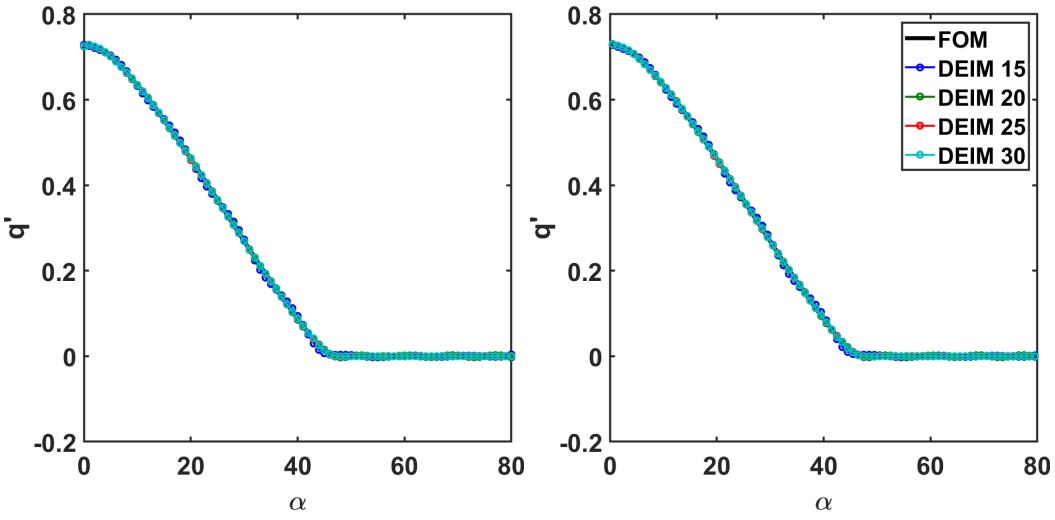


Figure 5.4: Sampling elements for DEIM approximation. A zoom-in on the right wall for $y \in [2.1, 2.6]$ shows that boundary elements are sampled as well.

various state basis sizes are shown in Figure 5.5 for both reconstruction of the training configurations and prediction on the testing configurations, and clearly the model improves with increasing POD state basis size.

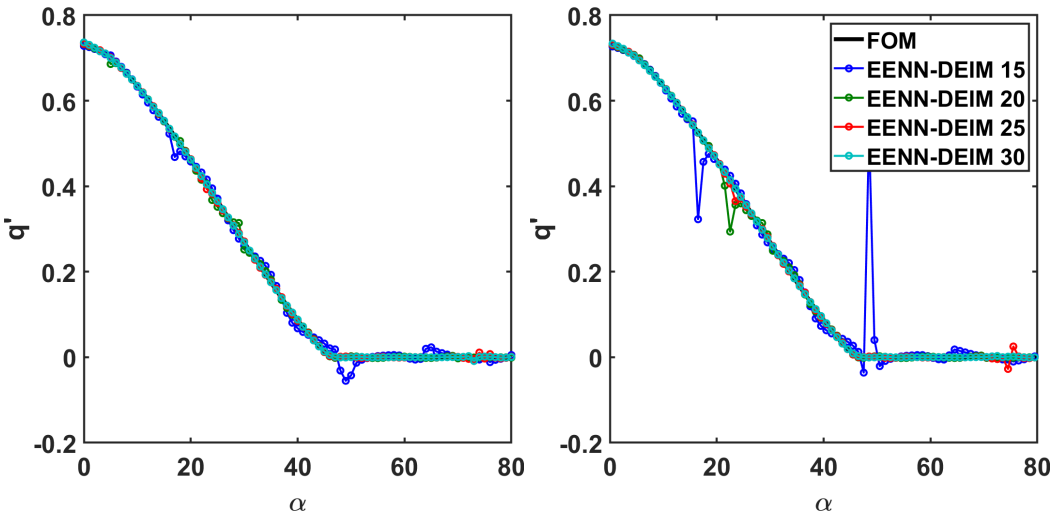
The EENN-DEIM model is written in MATLAB for this example problem as a demonstration of the portability of the EENN-DEIM ROM. All networks are designed with a single hidden layer containing 12 neurons and a sigmoid activation function. Networks for interior elements have 21 inputs: the Discontinuous Galerkin state basis has 4 weights per element, each interior element has the states of their 4 element neighbors and itself as inputs, and the convection angle. The Jacobian is approximated with the finite differencing formulation defined by Equation (5.20). The solutions for the reconstruction and testing configurations are shown in Figure 5.6.

The performance of the EENN-DEIM model is worse than the traditional DEIM implementation for the coarser DEIM-ROM solutions. However, with POD state basis refinement, the EENN-DEIM solution performs comparably to the traditional DEIM implementation. The number of residual snapshots used is much larger than what would be typically needed for hyper-reduction for a problem of this simplicity. However, the high number of snapshots arises from the need to train the element-embedded neural networks. As mentioned before, with 21 inputs and one hidden layer containing 12 neurons, the networks for the interior elements are composed of 277 weights and biases that need to be trained. Thus, a larger number of training snapshots is needed for constructing EENN-DEIM models. Additionally, the configurations on which the EENN-DEIM models are most erroneous correspond to the configurations on which the finite differencing of the networks did not produce a high quality Jacobian. When constructing the Jacobian with finite differencing,



(a) DEIM reconstruction. **(b)** DEIM testing configurations.

Figure 5.5: DEIM reconstruction and testing solutions for the scalar advection-diffusion problem with a nonlinear source. The ROM sizes are: $n_{\boldsymbol{V}} = [15, 20, 25, 30]$, $n_{\boldsymbol{U}} = 45$, and $n_{\boldsymbol{P}} = 90$.



(a) EENN-DEIM reconstruction **(b)** EENN-DEIM testing configurations

Figure 5.6: EENN-DEIM reconstruction and testing solutions for the scalar advection-diffusion problem with a nonlinear source. The ROM sizes are: $n_{\boldsymbol{V}} = [15, 20, 25, 30]$, $n_{\boldsymbol{U}} = 45$, and $n_{\boldsymbol{P}} = 90$. The element-embedded neural networks contain one hidden layer with 12 neurons and sigmoid activation functions.

the step size is incrementally decreased and consecutive finite differences are compared. These finite differences are defined with,

$$\boldsymbol{j}_i = \frac{\boldsymbol{R}_{\text{EENN}}(\boldsymbol{Q}\left(\hat{\boldsymbol{x}}\right) + \boldsymbol{e}_{n_{\boldsymbol{V}},i}\epsilon_i, \boldsymbol{\mu}, t) - \boldsymbol{R}_{\text{EENN}}(\boldsymbol{Q}\left(\hat{\boldsymbol{x}}\right) - \boldsymbol{e}_{n_{\boldsymbol{V}},i}\epsilon_i, \boldsymbol{\mu}, t)}{2\epsilon_i}, \tag{5.23}$$

where ϵ_i is the step size from the i^{th} finite difference, with $e_i > e_{i+1}$. If the error between the two is below a set tolerance, then the Jacobian approximation is saved and the next finite difference is computed. The error used to compare finite differences for this demonstration is the L2 norm of the residual error, defined by

$$e_{\text{FD}} = ||\left(\boldsymbol{j}_2 - \boldsymbol{j}_1\right) \oslash \boldsymbol{j}_1||_2, \tag{5.24}$$

where $\oslash$ is an element-wise division operation. For these configurations, the error of the finite differencing was on the order of 10%, whereas the allowed tolerance was set to 0.01%. This clearly affected the performance of the EENN-DEIM ROM at those configurations in comparison to the traditional DEIM model. In terms of speedup, the performance of this implementation of the EENN-DEIM model is lacking in comparison to the traditional DEIM model. The average run time for a traditional DEIM-ROM solution was 2.125 seconds, whereas the average run time for a EENN-DEIM solution was 22.154 seconds. The key factors that impacted the performance of the EENN-DEIM model were the chosen coding framework and the cost of the finite differencing. The tradition DEIM model is implemented with the C programming language, which is a compiled language. For C programs, the plain-text code goes through pre-processing and compiling stages that optimize the code before becoming an executable computer program. MATLAB, on the other hand, is an interpreted language, which does not benefit from the optimizations that compiled languages receive; instead, MATLAB code is translated to machine code at run time, line-by-line, which is almost always less efficient than a compiled language. The other factor impacting the performance of the EENN-DEIM model is the cost of producing the Jacobian through finite differences. The computation of the finite differences dominated the simulation cost of the EENN-DEIM model as 84% of the run time was spent on formulating the Jacobian approximations. If automatic differentiation were used instead, the entirety of the gradient formulation and residual evaluation would be encompassed into a single execution of all the networks, substantially reducing the computational cost. Despite these issues, the good performance of the EENN-DEIM model for finer state bases and the implementation of the EENN-DEIM model in MATLAB demonstrates the practicality and portability of this technique. The proceeding example will demonstrate the immense speedup and

improvements to accuracy that is obtained from the implementation of EENN-DEIM in a compiled language and the use of automatic differentiation for Jacobian construction.

5.4.2 Unsteady Pitching and Plunging Airfoil

The previous example is meant to demonstrate the portability and accuracy of the EENN-DEIM method. The model was prototyped entirely in MATLAB, only needing the FOM model for state and residual snapshots. Finite differencing was used for approximating the Jacobian; however, the inaccuracies of this approach to Jacobian formulation impacted the fidelity of the EENN-DEIM model. Additionally, the lack of optimization due to MATLAB being an interpreted language and the cost of finite differencing severely impacted the speedup of the EENN-DEIM model. For this example, the EENN-DEIM model is implemented in C, and automatic differentiation of the network with respect to its inputs is used for Jacobian calculations. These changes result in an immense improvement to the speedup of the EENN-DEIM model over the FOM and DEIM-ROM. The pitching and plunging airfoil, first defined in Section 3.5.3, is the subject of this demonstration.

To summarize, the forced pitching and plunging of a NACA 0012 airfoil is implemented with ALE mesh motion for different reduced frequencies and freestream Mach numbers. The fluid system is modeled with compressible URANS physics, using the Spalart-Allmaras turbulence model. The freestream conditions are $Re = 500K$ and zero angle of attack. EENN-DEIM models are built for the 99% singular value energy state basis. The DEIM residual basis contains 120 basis vectors, and 200 elements are sampled for generating the sparse residual. The locations of these elements are shown in Figure 5.8. Given that the FOM uses DG and a linear spatial basis, the number of degrees of freedom of the DEIM residual is 3000, and an element-embedded neural network is created for each element. These neural networks consist of a single hidden layer with 15 neurons and hyperbolic tangent activation functions. The number of inputs varied between the three different kinds of elements: 79 for interior elements, 69 for freestream boundary elements, and 64 for airfoil boundary elements. Four mesh motion variables are provided to the networks as inputs: the instantaneous pitch and plunge, and the temporal derivatives of the pitch and plunge. The DG state basis coefficients are also provided to the networks from the four neighboring elements and the embedded element. This totals to 75 state inputs for interior elements and 60 state inputs for the two different boundary elements. The boundary elements have additional inputs based on the boundary conditions of the adjacent boundary. For the freestream boundaries, these are the full-state vector for the boundary conditions: density, specific momentum in both Cartesian directions, specific energy, and the resolved viscous variable in the Spalart–Allmaras turbulence model ($\tilde{\nu}$). The airfoil boundary is

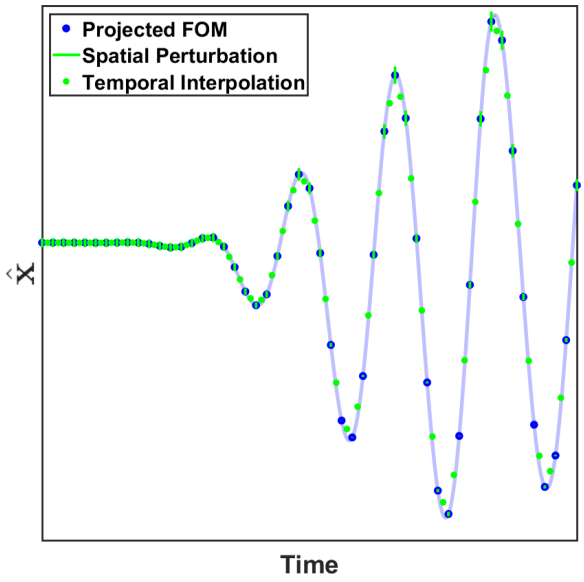
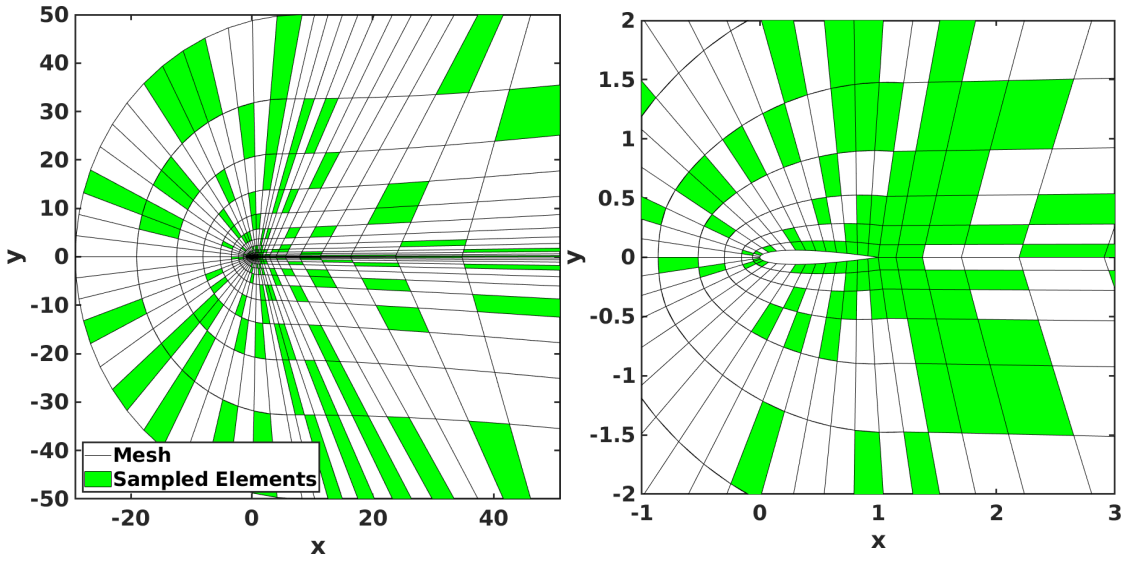


Figure 5.7: Mock-up of the trajectory of the i^{th} state basis weight and the spatial/temporal perturbations used to generate residual snapshots. The coarseness of the time discretization is for illustrative purposes, while the true discretization used is 12 times finer.

treated as a no-slip, impenetrable wall with the heat-flux and Spalart-Allmaras variable specified to be 0. Thus no additional input is given to the embedded neural networks for the airfoil elements, as these boundary conditions are invariant.

Overall, the largest networks have 1216 weights and biases. There are only 2404 state snapshots; thus merely using the residuals that arise from projecting the FOM solutions with the 99% singular value energy basis may not be sufficient for training. Therefore, additional snapshots were generated to train the network model. These residuals arise from both temporal and spatial perturbations of the projected FOM snapshot coefficients. Similar to the previous examples, the coefficients of the projected snapshots are perturbed $\pm 2.5\%$ twice to generate two new residual snapshots for each residual, and linear interpolation of the projection is used to construct coefficients at the mid-points between projected FOM solutions to generate additional residual snapshots. An mock-up of these perturbations is shown in Figure 5.7. Overall, the 6006 residuals snapshots are used to train the element-embedded networks in MATLAB.

A custom neural-network library is used in `xflow` to generate the EENN-DEIM solutions; additionally, automatic differentiation is used to generate the Jacobian for the EENN-DEIM ROM in tandem with the EENN-DEIM residuals. Using Galerkin projection, the EENN-DEIM model is unable to capture the load trajectory of the airfoil; however, the use of a state adjoint Petrov-Galerkin test basis yields far more accurate predictions. The SAPG test basis arises from the reduced formulation and uses the 99.9% singular value



(a) Full-domain **(b)** Near airfoil

Figure 5.8: Sampling elements for DEIM approximation. For each element, all of the DG state basis weights are used in the DEIM approximation.

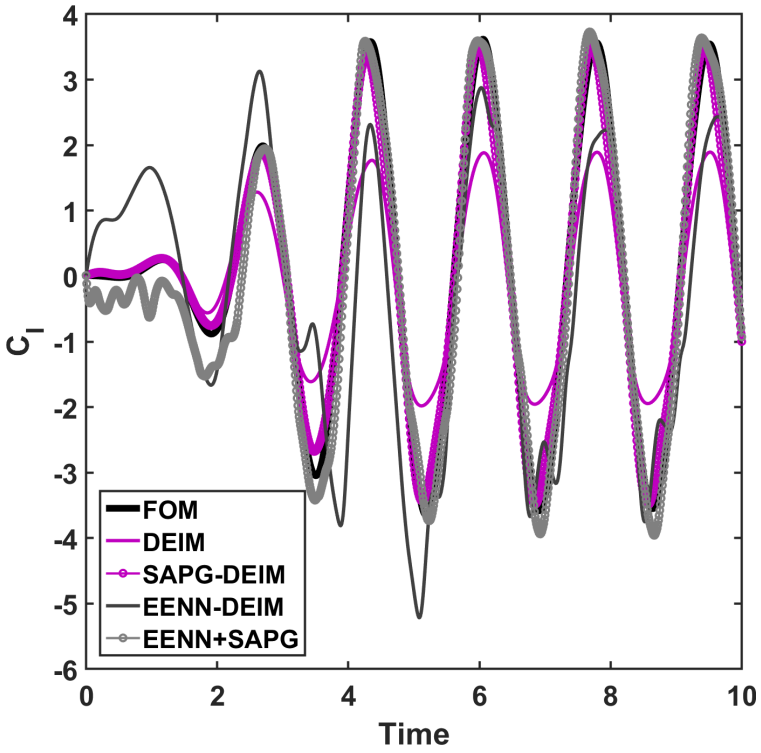


Figure 5.9: EENN-DEIM solutions for the unsteady airfoil problem. SAPG test basis vectors are used for improving the predictions of the EENN-DEIM model.

energy basis as the test search space. The solutions for the models in comparison to the DEIM-ROM solutions are shown in Figure 5.9. The poor EENN-DEIM solution for the Galerkin projection and the improved solution obtained with a SAPG test basis is similar to the behavior of the DEIM-ROM and SAPG-DEIM models and indicates that neural networks can be used at lower levels to develop accurate residuals for non-intrusive model reduction. The speedups of the DEIM and EENN-DEIM models are an order of magnitude higher than the speedups for the POD and DEIM-ROM models. The higher speedups are likely due to the reduced complexity of the element-embedded neural network compared to the DEIM implementation and the advantages of automatic differentiation. Additionally, the speedups for the SAPG formulations are less than the Galerkin formulations due to the construction of the SAPG test basis vectors and the additional amounts of residual that are reduced – the norm of the projection of the residual with the SAPG test basis is typically O(1-2) times larger than the norm of the Galerkin projection of the residual. Overall, this example demonstrates the potential of the EENN-DEIM model to generate accurate residuals for use in non-intrusive model reduction and can substantially improve performance when implemented with automatic differentiation.

However, this problem demonstrates three major issues that need to be addressed for the application of EENN-DEIM to larger, more complex problems. First, the need for many more residual snapshots is a major obstacle due to the significantly higher cost of their production for larger scale problems. This can be address with more sophisticated, general networks and is discussed in the conclusion of this chapter and the conclusion of this thesis. Second, the EENN-DEIM solutions are **highly** sensitive to the design of the neural networks. For this problem, the number of neurons and hidden layers have a dramatic effect on the stability and accuracy of the EENN-DEIM model, which facilitated multiple attempts at constructing and training the neural network models. This process is costly and will only be more expensive and prohibitive for larger problems. The use of more sophisticated neural-networks structures such as a convolutional neural-network may improve the consistency of trained surrogate models. The spatial residual is calculated from the balance of diffusive and convective fluxes integrated along cell boundaries. For FEM-DG, each flux is only dependent on the embedded cell and neighboring cell of the integrated boundary. Thus a convolutional network is more appropriate for identifying this relationship offline and may result in more consistent training. A convolutional structure may also scale better to 3D cases as the relationship between spatial residuals, advective and diffusive fluxes, and adjacent state information still holds. Finally, the training cost for 3000 individual networks was immensely expensive and was performed using distributed computing. For larger-scale problems this approach may not possible as the number of

interpolating elements for DEIM modeling scales with the problem degrees of freedom. Once again, using fewer more general networks is necessary for EENN-DEIM application on larger problems.

Method	Speedup
FOM	1.0
POD	2.7993
SAPG-POD	1.1012
DEIM	8.1421
SAPG-DEIM	4.4376
EENN-DEIM	66.8178
EENN-SAPG	31.8576

Table 5.2: Speed ups for all of the ROMs using the 99% singular value energy state basis.

5.5 Conclusion

This chapter introduced element-embedded neural networks for DEIM hyper-reduction, which is a non-intrusive model reduction technique that supplants intrusive portions of the DEIM algorithm with neural networks. DEIM-ROMs often require access to the FOM code for computing sparsely sampled residuals and Jacobians that are used to approximate the full-order residuals and Jacobians through sparse sensing interpolation. However, several hindrances can make the development of DEIM subroutines impractically complex or even impossible; additionally, the resulting DEIM subroutines are often not portable, leading to the need to rewrite them when applied to new FOM frameworks. The EENN-DEIM method uses element-level neural networks in place of these subroutines by mapping relevant states to residuals with neural networks. These neural networks are "element-embedded" in that each network is associated with an individual discretization element as opposed to other methods that use the neural networks on the projected residuals and states. Further, the use of automatic differentiation can substantially increase the speedup over traditional projection-based models, as shown in Section 5.4.2. EENN-ROMs can also be used with state adjoint Petrov-Galerkin test spaces, developed in Chapter 4, to yield more accurate solutions. Finally, EENN-DEIM models are non-intrusive to the FOM model as long as state and residual snapshots are able to be collected and decomposed.

There are some drawbacks to using EENN-DEIM that should be noted. First, to account for the high number of weights and biases that are trained for each element-embedded network, the number of residual snapshots needed for developing the EENN-DEIM models is

significantly higher than the number of snapshots needed for constructing the DEIM residual basis and the DEIM sampling matrix. This is more impactful for higher spatial order schemes where the number of state degrees of freedom grows quickly. A possible solution is to use a single network for each element type (interior, boundary, etc.) as opposed to each element individually. To do so, information about the embedded and neighboring element geometry needs to be provided. Thus, under this framework, the number of inputs increases; however, all of the sampling elements will share training data, significantly increasing the number of sampling points. Another open issue with the EENN-DEIM method is how to design deeper element-embedded neural networks to obtain possibly higher accuracy with maintaining smoothness of the gradients for stability when deployed online. All of the EENN-DEIM models demonstrated in this thesis are kept shallow to keep gradients of the network smooth for stability. Deeper networks can provide more accurate solutions, but the added nonlinearity can create oscillatory and large gradients which are undesirable for minimization. This issue may be ameliorated by more complex training frameworks and by training the network on both residuals and Jacobians. This is detailed more in the concluding chapter of this thesis.

CHAPTER 6

Concluding Remarks

This concluding chapter summarizes the preceding chapters, highlights the key-contributions of this research, and presents potential future work that expands on the ideas presented.

6.1 Summary

This thesis focused on three main subjects: 1) the use of adjoint-weighted residual error estimates for projection-based reduced-order models and hyper-reduced order models, 2) the construction of state adjoint Petrov-Galerkin reduced-order models for optimal state prediction, and 3) the application of element-level embedded neural networks for non-intrusive DEIM hyper-reduction. This chapter serves to highlight the major accomplishments in these subjects and propose additional work to be done.

Adjoint-weighted residual error estimation techniques were adapted for the discrete-empirical interpolation method (DEIM) to quantify the error of the output prediction of a DEIM-ROM, which can then be used for adaptation. These error estimates arise from a sensitivity analysis of the output with respect to the residual and a perturbation of the residual realized upon injection of the DEIM solution into a finer-space DEIM setting. The error estimation was verified on a steady scalar transport problem before being applied onto an inviscid, 3D wing undergoing forced rigid-body motions with ALE mesh deformations and onto a viscous, 2D, multiparameter airfoil problem with dynamic pitching and plunging. Overall, the quantification of the output error was accurate and improved with increased fidelity of the ROM. Chapter 3 also develops a method for localizing the error of the DEIM-ROM to specific degrees of freedom of the fine-space DEIM, which can then be targeted for adaptation. These degrees of freedom are the metrics that determine the "size" of the DEIM-ROM: the rank of the state basis, the rank of the residual basis, and the number of sampling indices for residual and Jacobian approximations. With this mechanism for driving adaptation, coarse DEIM-ROMs for the 3D wing and 2D airfoil were

adapted to generate models with more accurate output predictions while maintaining relatively coarse sizes. Key insights from these exercises are that adaptation is more beneficial in enriching the test basis rather than improving the state space representation and that the singular value energy criterion used to construct a basis is more beneficial for selecting the leading basis components and is less important for the higher basis vectors. As a general observation, for a basis at or above 99% of the singular value energy of the decomposition of a well-constructed snapshot set, an additional state basis vector adds relatively little to its overall ability to span the state space. However, the ordering of the singular values does not correlate with how much the associated basis vectors span the residual. Thus, adjoint-weighted residual error estimates and adaptation are beneficial because they are able to identify unused degrees of freedom that can drive the system towards a more accurate output prediction.

However, in many cases the test basis that can obtain the appropriate dynamics to drive the solution towards the correct solution is not well represented by the state basis. To address this, Chapter 4 introduces a novel method for constructing a Petrov-Galerkin ROM test basis that optimally drives the solution towards the *ideal* solution – *i.e.*, the full-order model solution projected with the state basis. This test basis transforms the residual error minimization problem to be equivalent to the minimization of the error of the reduced state with respect to the ideal solution. It was shown that this test basis consists of the reduced state adjoints of the system. Thus, this method is referred to as the state adjoint Petrov-Galerkin method (SAPG). Formations for the SAPG test basis are given for linear, nonlinear, and hyper-reduction settings. Additionally, to ameliorate the cost of the construction of the SAPG test basis, reduced-order formulations are also developed. Insights into the stability and convergence of the SAPG test basis are provided and compared to the popular least-squares Petrov-Galerkin test basis (LSPG). Overall, the SAPG test basis has eigenvalues similar to those of the LSPG test basis. The Newton-Raphson updates for the reduced state for the two test bases differ in that the LSPG test basis forms a least squares solution whereas the SAPG test basis forms the projection of the full-order model update. Comparisons between Galerkin ROMs, the LSPG-POD ROM, and the SAPG ROMs are made on several different models. Each demonstrates the ability of the SAPG method and its reduced formulation to push the solution towards the *ideal* solution and to produce more accurate output predictions for most of the tested problems; however, limitations of the reduced formulation are shown and alternative means of constructing the SAPG test basis are shown.

Chapter 5 moves away from the use of adjoint-based techniques and focuses on the construction of non-intrusive DEIM-ROMs via the use of element-embedded neural networks

(EENN-DEIM). The intrusiveness of DEIM is a major drawback of its use, especially in situations where access to the FOM code is not available. Instead, the EENN-DEIM method replaces the intrusive portions of the DEIM method with neural networks – *i.e.*, the sparse calculation of residuals and Jacobians for the DEIM interpolation. These neural networks are each associated with an individual degree of freedom of an individual element. Training of the neural networks only requires access to state and residual snapshots, which are necessary for DEIM-ROM construction. However, the number of residual and state snapshots is significantly larger than what is needed to form a traditional DEIM model in order to facilitate network training. The inputs to the EENN networks are the same inputs that are used to construct the FOM residual: the internal and neighboring states, boundary conditions for boundary elements, and any system parameters. The output of the EENN DEIM model is the residual of the corresponding degree of freedom. Chapter 5 also provides two separate methods for computing the system Jacobian: finite differencing and automatic differentiation. Of the two, automatic differentiation is significantly more accurate and efficient, as it is analytical and is constructed in tandem with the residual computation rather than needing to form multiple residuals for finite differencing. The EENN-DEIM ROM is compared to the traditional DEIM model for a steady, nonlinear, scalar transport problem and a viscous airfoil problem with forced deformations. The results of the EENN-DEIM model are promising, especially when combined with the SAPG test basis; however, investigation into the training and construction of EENN-DEIM may provide solutions to the issues noted above.

6.2 Key Contributions

The key contributions of this thesis can be summarized as:

- Developed time-coupled adjoint equations for DEIM hyper-reduced models in order to produce adjoint-weighted residual output error estimates (Chapter 3).
- Formulated an approach for localization of the error to specific degrees of freedom (Chapter 3).
- Demonstrated fine-grained DEIM-ROM adaptation methods based on output error estimates (Chapter 3).
- Derived a novel Petrov-Galerkin test basis for minimizing state errors of POD-ROMs and DEIM hyper-reduced ROMs (Chapter 4).
- Derived reduced-order models for the SAPG test basis (Chapter 4).

- Compared SAPG and LSPG test bases convergence, stability, and prediction quality for several example applications (Chapter 4).
- Developed a novel hybridized DEIM-neural network model for non-intrusive projection-based model reduction (Chapter 5).
- Demonstrated $\mathcal{O}(10)$ speedup of the hybridized model on an aerodynamic problem of interest (Chapter 5).

6.3 Future Work

There are several open areas of research that are left for future work:

- **Application of adjoint-weighted residual error estimates and adaptation to SAPG-ROMs:** The SAPG test space drives the solution towards the ideal solution only in a linear sense. This is demonstrated in Section 4.2 where for linear problems, the SAPG test space does drive the reduced solution to be equal to the ideal solution, but for the nonlinear examples the reduced solutions did not exactly equal the ideal solution; therefore, quantifying the error in the solution is still necessary. For application of adjoint-weighted residual error estimation techniques on the reduced formulation of the SAPG-ROMs, the rank of the test search space would need to be increased for constructing the fine-space ROM, in addition to the rank of state basis, the rank of the residual basis, and the number of sampling indices. Additionally, the solutions for the final SAPG test space for each time step would need to be saved during the primal solve so that the backwards in time marching could be properly constructed. The state basis error localizations (Equation (3.54), Equation (3.55), Equation (3.56), and Equation (3.61)) will change as the errors are localized to the test basis, which is no longer the state basis in the Petrov-Galerkin formation. However, those formulas now represent the localization for the test search space, thus other means of error estimates would need to be used to formulate the error of the state basis.
- **Application of adjoint-weighted residual error estimation and adaptation to EENN-DEIM ROMs:** Quantification of the error of EENN-DEIM ROMs is arguably more important than for the SAPG-ROMs as the EENN-DEIM ROM removes the physical model underlying the FOM system. Doing so requires an additional pool of unused, trained element-embedded neural networks that act as a fine-space for the EENN-DEIM residual. Errors can be localized to specific fine-space neural

networks that can be targeted for adaptation using Equation (3.58) for steady problems or Equation (3.61) for unsteady problems. In addition, adaptive techniques for the DEIM sampling matrix developed by Peherstorfer and Willcox [105] can be incorporated with the pool of fine-space element-embedded neural networks.

- **Generation of stabilized ROMs with the SAPG method:** Chapter 4 performs stability analysis on the SAPG test basis, but it does not incorporates methods for generating quantified stable ROMs for unsteady simulations. However, clearly from Example 4.2.4, the choice of the test space reduced-order model has an impact on the stability of the ROM; an investigation into constructing SAPG test search spaces and more broadly SAPG test space reduced-order models to guarantee stability is of interest.

- **Utilization of deep element-embedded neural networks for DEIM:** The networks used in Chapter 5 are shallow, single-hidden layer networks. This was done to keep the gradients of the EENN-DEIM model smooth for minimization. Improvements to the accuracy of the EENN-DEIM model are possible through the use of deeper networks, but the added nonlinearity of deep neural networks can lead to less smooth, irregular gradients that can harm the residual minimization process. However, changes to the training model may allow for deeper element-embedded neural networks with smooth gradients. Two possible changes are the regularization of the gradient and training the neural networks on the residuals and gradients. In the first, the loss function can take into account errors from the network prediction as well as the magnitude of the gradient, according to

$$\tilde{J}(g(\boldsymbol{x}_{\text{NN}t}), \boldsymbol{y}_{\text{NN}t}) = \alpha J_{\boldsymbol{y}_{\text{NN}}}(g(\boldsymbol{x}_{\text{NN}t}), \boldsymbol{y}_{\text{NN}t}) + \beta J_{\nabla}(g(\boldsymbol{x}_{\text{NN}t}), \boldsymbol{y}_{\text{NN}t}), \tag{6.1}$$

$$J_{\nabla}(g(\boldsymbol{x}_{\text{NN}t}), \boldsymbol{y}_{\text{NN}t}) = \sqrt{\frac{\sum_{i=1}^{N_t} ||\frac{dg(\boldsymbol{x}_{\text{NN}t,i})}{d\boldsymbol{x}_{\text{NN}t,i}}||_2^2}{N_t}}, \tag{6.2}$$

where $J_{\boldsymbol{y}_{\text{NN}}}$ is the traditional loss function based on the error of the network on the training data, J_{∇} is the loss function based on the norm of the gradients – taken for this example to be the root mean square but can be any loss function – and α and β are weights for the loss functions. In the second, the snapshots of the Jacobian can also be used to train both the output and gradients of the networks. The loss function is similar to Equation (6.1) but with a different definition for the gradient

loss function, according to

$$\tilde{J}(g(\boldsymbol{x}_{\text{NN}t}), \boldsymbol{y}_{\text{NN}t}) = \alpha J_{\boldsymbol{y}_{\text{NN}}}(g(\boldsymbol{x}_{\text{NN}t}), \boldsymbol{y}_{\text{NN}t}) + \beta J_{\nabla}(g(\boldsymbol{x}_{\text{NN}t}), \boldsymbol{y}_{\text{NN}t}), \tag{6.3}$$

$$J_{\nabla}(g(\boldsymbol{x}_{\text{NN}t}), \boldsymbol{y}_{\text{NN}t}) = \sqrt{\frac{\sum_{i=1}^{N_t} || \frac{dg(\boldsymbol{x}_{\text{NN}t,i})}{d\boldsymbol{x}_{\text{NN}}t,i} - \boldsymbol{p}_j^T \frac{\partial \boldsymbol{R}(\boldsymbol{x}_{\text{NN}t})}{\partial \boldsymbol{x}} ||_2^2}{N_t}}, \tag{6.4}$$

where $\boldsymbol{p}_j$ is the sparse sensing vector that is used for the network being trained. Training the gradients of the system may lead to higher accuracy in the gradients; however, the computational and memory cost of collecting snapshots of the Jacobian may make Equation (6.1) a more attractive option, unless the Jacobian can be sampled in a sparse fashion, defeating the initial motivation of this method.

- **Generalized element-embedded neural networks for DEIM:** A crucial drawback to the use of element-embedded neural networks is the significantly larger number of state and residual snapshots that are needed to accurately train the networks. The inputs for each network include the entire state representation of the embedded element and its neighbors, possible boundary conditions, and system parameters. The large input layer means that, even for a single hidden layer network, the total number of weights and biases is very large. Thus, a sufficient training set will need to be even larger. When each network is only associated with a single element, every state-residual snapshot pair will only account for a single training observation. However, a generalized element-embedded neural network that can operate on a subset of elements is able to collect training observations from each element for each state-residual snapshot pair, substantially increasing the amount of training data for each network. An example of the categorization of elements is one based on its neighboring cells: interior elements and distinct boundary elements. Additional inputs regarding the size and orientation of the embedded and neighboring elements would need to be provided to allow the network to differentiate between different elements. Quantification of the geometric configurations of the elements can take the form of a various metric tensors [127, 128], elemental bounding and minimum bounding boxes, normal vectors at element edges, etc.

BIBLIOGRAPHY

[1] Dongarra, J. J., "Performance of various computers using standard linear equations software," *Computer Science Technical Report Number CS*, Vol. 89, No. 85, 2022.

[2] Newhouse, J., *Boeing Versus Airbus: The Inside Story of the Greatest International Competition in Business*, Vintage Books, 2008.

[3] Johnson, F. T., Tinoco, E. N., and Yu, N. J., "Thirty years of development and application of CFD at Boeing Commercial Airplanes, Seattle," *Computers & Fluids*, Vol. 34, No. 10, 2005, pp. 1115–1151.

[4] Dongarra, J. J., "Performance of various computers using standard linear equations software," *Computer Science Technical Report Number CS*, Vol. 89, No. 85, 1994.

[5] Slotnick, J. P., Khodadoust, A., Alonso, J., Darmofal, D., Gropp, W., Lurie, E., and Mavriplis, D. J., "CFD vision 2030 study: a path to revolutionary computational aerosciences," Tech. rep., 2014.

[6] Giunta, A. A., Dudley, J. M., Narducci, R., Grossman, B., Haftka, R. T., Mason, W. H., and Watson, L. T., "Noisy aerodynamic response and smooth approximations in HSCT design," *Proceedings in 5th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary and Structural Optimization*, 1994, pp. 1117–1128.

[7] Mullur, A. A. and Messac, A., "Extended radial basis functions: more flexible and effective metamodeling," *AIAA Journal*, Vol. 43, No. 6, 2005, pp. 1306–1315.

[8] Chandrashekarappa, P. and Duvigneau, R., "Radial basis functions and kriging metamodels for aerodynamic optimization," *INRIA Document*, 2007, pp. 26–30.

[9] Silva, W., "Reduced-order models based on linear and nonlinear aerodynamic impulse responses," *40th Structures, Structural Dynamics, and Materials Conference and Exhibit*, 1999, p. 1262.

[10] Skujins, T. and Cesnik, C. E. S., "Reduced-order modeling of unsteady aerodynamics across multiple mach regimes," *Journal of Aircraft*, Vol. 51, No. 6, 2014, pp. 1681–1704.

[11] Simpson, T. W., Mauery, T. M., Korte, J. J., and Mistree, F., "Kriging models for global approximation in simulation-based multidisciplinary design optimization," *AIAA Journal*, Vol. 39, No. 12, 2001, pp. 2233–2241.

[12] Falkiewicz, N. J. and Cesnik, C. E. S., "Enhanced modal solutions for structural dynamics in aerothermoelastic analysis," *Journal of Aircraft*, Vol. 54, No. 3, 2017, pp. 870–889.

[13] Brunton, S., Noack, B., and Koumoutsakos, P., "Machine learning for fluid mechanics," *arXiv preprint arXiv:1905.11075*, 2019.

[14] Raissi, M., Perdikaris, P., and Karniadakis, G. E., "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, Vol. 378, 2019, pp. 686–707.

[15] Wang, Q., Medeiros, R. R., Cesnik, C. E. S., Fidkowski, K., Brezillon, J., and Bleecke, H. M., "Techniques for improving neural network-based aerodynamics reduced-order models," *AIAA Scitech 2019 Forum*, 2019, p. 1849.

[16] Sirovich, L., "Method of snapshots," *Quarterly of Applied Mathematical*, Vol. 45, No. 3, 1987, pp. 561–571.

[17] Rowley, Clarence W. and Colonius, T. and Murray, R. M., "Model reduction for compressible flows using POD and Galerkin projection," *Physica D: Nonlinear Phenomena*, Vol. 189, No. 1-2, 2004, pp. 115–129.

[18] Bai, Z., "Krylov subspace techniques for reduced-order modeling of large-scale dynamical systems," *Applied Numerical Mathematics*, Vol. 43, No. 1-2, 2002, pp. 9–44.

[19] Freund, R. W., "Model reduction methods based on Krylov subspaces," *Acta Numerica*, Vol. 12, 2003, pp. 267–319.

[20] Benner, P., Gugercin, S., and Willcox, K., "A survey of projection-based model reduction methods for parametric dynamical systems," *SIAM Review*, Vol. 57, No. 4, 2015, pp. 483–531.

[21] Lall, S., Marsden, J. E., and Glavaški, S., "A subspace approach to balanced truncation for model reduction of nonlinear control systems," *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, Vol. 12, No. 6, 2002, pp. 519–535.

[22] Schmid, P. J., "Dynamic mode decomposition of numerical and experimental data," *Journal of Fluid Mechanics*, Vol. 656, 2010, pp. 5–28.

[23] Towne, A., Schmidt, O. T., and Colonius, T., "Spectral proper orthogonal decomposition and its relationship to dynamic mode decomposition and resolvent analysis," *Journal of Fluid Mechanics*, Vol. 847, 2018, pp. 821–867.

[24] Barrault, M., Maday, Y., Nguyen, N. C., and Patera, A. T., "An 'empirical interpolation'method: application to efficient reduced-basis discretization of partial differential equations," *Comptes Rendus Mathematique*, Vol. 339, No. 9, 2004, pp. 667–672.

[25] Chaturantabut, S. and Sorensen, D. C., "Nonlinear model reduction via discrete empirical interpolation," *SIAM Journal on Scientific Computing*, Vol. 32, No. 5, 2010, pp. 2737–2764.

[26] Bui-Thanh, T., Damodaran, M., and Willcox, K., "Aerodynamic data reconstruction and inverse design using proper orthogonal decomposition," *AIAA Journal*, Vol. 42, No. 8, 2004, pp. 1505–1516.

[27] Astrid, P., Weiland, S., Willcox, K., and Backx, T., "Missing point estimation in models described by proper orthogonal decomposition," *IEEE Transactions on Automatic Control*, Vol. 53, No. 10, 2008, pp. 2237–2251.

[28] Nguyen, N. C., Patera, A. T., and Peraire, J., "A 'best points' interpolation method for efficient approximation of parameterized functions," *International Journal for Numerical Methods in Engineering*, Vol. DOI: 10.1002/nme.2086, 2007.

[29] Carlberg, K., Farhat, C., Cortial, J., and Amsallem, D., "The GNAT method for nonlinear model reduction: effective implementation and application to computational fluid dynamics and turbulent flows," *Journal of Computational Physics*, Vol. 242, 2013, pp. 623–647.

[30] Veroy, K. and Patera, A., "Certified real-time solution of the parametrized steady incompressible Navier–Stokes equations: rigorous reduced-basis a posteriori error bounds," *International Journal for Numerical Methods in Fluids*, Vol. 47, No. 8-9, 2005, pp. 773–788.

[31] Deparis, S. and Rozza, G., "Reduced basis method for multi-parameter-dependent steady Navier–Stokes equations: Applications to natural convection in a cavity," *Journal of Computational Physics*, Vol. 228, No. 12, 2009, pp. 4359–4378.

[32] Rozza, G., Huynh, D. B. P., and Patera, A. T., "Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations," *Archives of Computational Methods in Engineering*, Vol. 15, No. 3, 2008, pp. 229–275.

[33] Yano, M., "A space-time Petrov–Galerkin certified reduced basis method: Application to the Boussinesq equations," *SIAM Journal on Scientific Computing*, Vol. 36, No. 1, 2014, pp. A232–A266.

[34] Manzoni, A., "An efficient computational framework for reduced basis approximation and a posteriori error estimation of parametrized Navier–Stokes flows," *ESAIM: Mathematical Modelling and Numerical Analysis*, Vol. 48, No. 4, 2014, pp. 1199–1226.

[35] LeGresley, P. and Alonso, J., "Dynamic domain decomposition and error correction for reduced order models," *41st Aerospace Sciences Meeting and Exhibit*, 2003, p. 250.

[36] Wells, D. R., *Stabilization of POD-ROMs*, Ph.D. thesis, Virginia Tech, 2015.

[37] Meyer, M. and Matthies, H. G., "Efficient model reduction in non-linear dynamics using the Karhunen-Loève expansion and dual-weighted-residual methods," *Computational Mechanics*, Vol. 31, No. 1, May 2003, pp. 179–191.

[38] Carlberg, K., "Adaptive h-refinement for reduced-order models," *International Journal for Numerical Methods in Engineering*, Vol. 102, No. 5, 2015, pp. 1192–1210.

[39] Yano, M., "Discontinuous Galerkin reduced basis empirical quadrature procedure for model reduction of parametrized nonlinear conservation laws," *Advances in Computational Mathematics*, Vol. 45, No. 5, 2019, pp. 2287–2320.

[40] Yano, M., "Goal-oriented model reduction of parametrized nonlinear partial differential equations: Application to aerodynamics," *International Journal for Numerical Methods in Engineering*, Vol. 121, No. 23, 2020, pp. 5200–5226.

[41] Chaturantabut, S. and Sorensen, D. C., "A state space error estimate for POD-DEIM nonlinear model reduction," *SIAM Journal on Numerical Analysis*, Vol. 50, No. 1, 2012, pp. 46–63.

[42] Wirtz, D., Sorensen, D. C., and Haasdonk, B., "A posteriori error estimation for DEIM reduced nonlinear dynamical systems," *SIAM Journal on Scientific Computing*, Vol. 36, No. 2, 2014, pp. A311–A338.

[43] Zhang, Y., Feng, L., Li, S., and Benner, P., "An efficient output error estimation for model order reduction of parametrized evolution equations," *SIAM Journal on Scientific Computing*, Vol. 37, No. 6, 2015, pp. B910–B936.

[44] Feng, L., Mangold, M., and Benner, P., "Adaptive POD–DEIM basis construction and its application to a nonlinear population balance system," *AIChE Journal*, Vol. 63, No. 9, 2017, pp. 3832–3844.

[45] Venditti, D. A. and Darmofal, D. L., "Adjoint error estimation and grid adaptation for functional outputs: Application to quasi-one-dimensional flow," *Journal of Computational Physics*, Vol. 164, No. 1, 2000, pp. 204–227.

[46] Becker, R. and Rannacher, R., "An optimal control approach to a posteriori error estimation in finite element methods," *Acta Numerica*, edited by A. Iserles, Cambridge University Press, 2001, pp. 1–102.

[47] Fidkowski, K. J. and Darmofal, D. L., "Review of output-based error estimation and mesh adaptation in computational fluid dynamics," *AIAA Journal*, Vol. 49, No. 4, 2011, pp. 673–694.

[48] Collins, G., Fidkowski, K., and Cesnik, C. E. S., "Output Error Estimation for Projection-Based Reduced Models," *AIAA Aviation 2019 Forum*, 2019, p. 3528.

[49] Ballarin, F., Manzoni, A., Quarteroni, A., and Rozza, G., "Supremizer stabilization of POD–Galerkin approximation of parametrized steady incompressible Navier–Stokes equations," *International Journal for Numerical Methods in Engineering*, Vol. 102, No. 5, 2015, pp. 1136–1161.

[50] Amsallem, D. and Farhat, C., "Stabilization of projection-based reduced-order models," *International Journal for Numerical Methods in Engineering*, Vol. 91, No. 4, 2012, pp. 358–377.

[51] Huynh, D. B. P., Rozza, G., Sen, S., and Patera, A. T., "A successive constraint linear optimization method for lower bounds of parametric coercivity and inf–sup stability constants," *Comptes Rendus Mathematique*, Vol. 345, No. 8, 2007, pp. 473–478.

[52] Kalashnikova, I. and Barone, M., "On the stability and convergence of a Galerkin reduced order model (ROM) of compressible flow with solid wall and far-field boundary treatment," *International Journal for Numerical Methods in Engineering*, Vol. 83, No. 10, 2010, pp. 1345–1375.

[53] Barone, M. F., Kalashnikova, I., Segalman, D. J., and Thornquist, H. K., "Stable Galerkin reduced order models for linearized compressible flow," *Journal of Computational Physics*, Vol. 228, No. 6, 2009, pp. 1932–1946.

[54] Lassila, T., Manzoni, A., Quarteroni, A., and Rozza, G., "Model order reduction in fluid dynamics: challenges and perspectives," *Reduced Order Methods for Modeling and Computational Reduction*, Springer, 2014, pp. 235–273.

[55] Bui-Thanh, T., Willcox, K., and Ghattas, O., "Model reduction for large-scale systems with high-dimensional parametric input space," *SIAM Journal on Scientific Computing*, Vol. 30, No. 6, 2008, pp. 3270–3288.

[56] Carlberg, K., Bou-Mosleh, C., and Farhat, C., "Efficient non-linear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations," *International Journal for Numerical Methods in Engineering*, Vol. 86, No. 2, 2011, pp. 155–181.

[57] Carlberg, K., Barone, M., and Antil, H., "Galerkin v. least-squares Petrov–Galerkin projection in nonlinear model reduction," *Journal of Computational Physics*, Vol. 330, 2017, pp. 693–734.

[58] Choi, Y. and Carlberg, K., "Space–time least-squares Petrov–Galerkin projection for nonlinear model reduction," *SIAM Journal on Scientific Computing*, Vol. 41, No. 1, 2019, pp. A26–A58.

[59] Parish, E. J., Wentland, C. R., and Duraisamy, K., "The Adjoint Petrov–Galerkin method for non-linear model reduction," *Computer Methods in Applied Mechanics and Engineering*, Vol. 365, 2020, pp. 112991.

[60] Ahmed, S. E., Pawar, S., San, O., Rasheed, A., Iliescu, T., and Noack, B. R., "On closures for reduced order models—A spectrum of first-principle to machine-learned avenues," *Physics of Fluids*, Vol. 33, No. 9, 2021, pp. 091301.

[61] San, O. and Maulik, R., "Machine learning closures for model order reduction of thermal fluids," *Applied Mathematical Modelling*, Vol. 60, 2018, pp. 681–710.

[62] Wang, Q., Ripamonti, N., and Hesthaven, J. S., "Recurrent neural network closure of parametric POD-Galerkin reduced-order models based on the Mori-Zwanzig formalism," *Journal of Computational Physics*, Vol. 410, 2020, pp. 109402.

[63] Parish, E. J. and Duraisamy, K., "A dynamic subgrid scale model for large eddy simulations based on the Mori–Zwanzig formalism," *Journal of Computational Physics*, Vol. 349, 2017, pp. 154–175.

[64] Demkowicz, L. and Gopalakrishnan, J., "A class of discontinuous Petrov–Galerkin methods. Part I: The transport equation," *Computer Methods in Applied Mechanics and Engineering*, Vol. 199, No. 23-24, 2010, pp. 1558–1572.

[65] Kast, S. M., *Methods for Optimal Output Prediction in Computational Fluid Dynamics.*, Ph.D. thesis, University of Michigan, Department of Aerospace Engineering, 2016.

[66] Kast, S. M., Dahm, J. P., and Fidkowski, K. J., "Optimal test functions for boundary accuracy in discontinuous finite element methods," *Journal of Computational Physics*, Vol. 298, 2015, pp. 360–386.

[67] Collins, G., Fidkowski, K., and Cesnik, C. E., "Petrov-Galerkin projection-based model reduction with an optimized test space," *AIAA Scitech 2020 Forum*, 2020, p. 1562.

[68] Wan, Z. Y., Vlachas, P., Koumoutsakos, P., and Sapsis, T., "Data-assisted reduced-order modeling of extreme events in complex dynamical systems," *PloS one*, Vol. 13, No. 5, 2018, pp. e0197704.

[69] San, O. and Maulik, R., "Extreme learning machine for reduced order modeling of turbulent geophysical flows," *Physical Review E*, Vol. 97, No. 4, 2018, pp. 042322.

[70] Renganathan, S. A., Maulik, R., and Rao, V., "Machine learning for nonintrusive model order reduction of the parametric inviscid transonic flow past an airfoil," *Physics of Fluids*, Vol. 32, No. 4, 2020, pp. 047110.

[71] San, O., Maulik, R., and Ahmed, M., "An artificial neural network framework for reduced order modeling of transient flows," *Communications in Nonlinear Science and Numerical Simulation*, Vol. 77, 2019, pp. 271–287.

[72] Wang, Q., Hesthaven, J. S., and Ray, D., "Non-intrusive reduced order modeling of unsteady flows using artificial neural networks with application to a combustion problem," *Journal of Computational Physics*, Vol. 384, 2019, pp. 289–307.

[73] Guo, M. and Hesthaven, J. S., "Data-driven reduced order modeling for time-dependent problems," *Computer Methods in Applied Mechanics and Engineering*, Vol. 345, 2019, pp. 75–99.

[74] Mohan, A. T. and Gaitonde, D. V., "A deep learning based approach to reduced order modeling for turbulent flow control using LSTM neural networks," *arXiv preprint arXiv:1804.09269*, 2018.

[75] Swischuk, R., Mainini, L., Peherstorfer, B., and Willcox, K., "Projection-based model reduction: Formulations for physics-based machine learning," *Computers & Fluids*, Vol. 179, 2019, pp. 704–717.

[76] Fresca, S., Dede, L., and Manzoni, A., "A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized PDEs," *Journal of Scientific Computing*, Vol. 87, No. 2, 2021, pp. 1–36.

[77] Peherstorfer, B. and Willcox, K., "Data-driven operator inference for nonintrusive projection-based model reduction," *Computer Methods in Applied Mechanics and Engineering*, Vol. 306, 2016, pp. 196–215.

[78] Swischuk, R., Mainini, L., Peherstorfer, B., and Willcox, K., "Projection-based model reduction: Formulations for physics-based machine learning," *Computers & Fluids*, Vol. 179, 2019, pp. 704–717.

[79] Khodabakhshi, P. and Willcox, K. E., "Non-intrusive data-driven model reduction for differential–algebraic equations derived from lifting transformations," *Computer Methods in Applied Mechanics and Engineering*, Vol. 389, 2022, pp. 114296.

[80] Lee, K. and Carlberg, K. T., "Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders," *Journal of Computational Physics*, Vol. 404, 2020, pp. 108973.

[81] Pan, S. and Duraisamy, K., "Physics-informed probabilistic learning of linear embeddings of nonlinear dynamics with guaranteed stability," *SIAM Journal on Applied Dynamical Systems*, Vol. 19, No. 1, 2020, pp. 480–509.

[82] Kani, J. N. and Elsheikh, A. H., "DR-RNN: A deep residual recurrent neural network for model reduction," *arXiv preprint arXiv:1709.00939*, 2017.

[83] Washabaugh, K. M., Zahr, M. J., and Farhat, C., "On the use of discrete nonlinear reduced-order models for the prediction of steady-state flows past parametrically deformed complex geometries," *54th AIAA Aerospace Sciences Meeting*, 2016, p. 1814.

[84] Ripepi, M., Verveld, M. J., Karcher, N. W., Franz, T., Abu-Zurayk, M., Görtz, S., and Kier, T., "Reduced-order models for aerodynamic applications, loads and MDO," *CEAS Aeronautical Journal*, Vol. 9, No. 1, 2018, pp. 171–193.

[85] Zhou, Y., Fan, D., Zhang, B., Li, R., and Noack, B. R., "Artificial intelligence control of a turbulent jet," *Journal of Fluid Mechanics*, Vol. 897, 2020.

[86] Gräßle, C., Hinze, M., and Volkwein, S., "Model order reduction by proper orthogonal decomposition," *Model Order Reduction: Volume 2: Snapshot-Based Methods and Algorithms*, De Gruyter, 2021, pp. 47–96.

[87] Bui-Thanh, T. and Willcox, K., "Model reduction for large-scale CFD applications using the balanced proper orthogonal decomposition," Vol. 4617, 2005, p. 2005.

[88] Unger, B., *Impact of Discretization Techniques on Nonlinear Model Reduction and Analysis of the Structure of the POD Basis*, Ph.D. thesis, Virginia Tech, 2013.

[89] Jarvis, C. H., *Parameter Dependent Model Reduction for Complex Fluid Flows*, Ph.D. thesis, Virginia Tech, 2014.

[90] Yano, M., "Model reduction in computational aerodynamics," *Model Order Reduction Volume 3: Applications, De Gruyter*, 2021, pp. 201–236.

[91] Epureanu, B., "A parametric analysis of reduced order models of viscous flows in turbomachinery," *Journal of Fluids and Structures*, Vol. 17, No. 7, 2003, pp. 971–982.

[92] Amsallem, D. and Farhat, C., "Interpolation method for adapting reduced-order models and application to aeroelasticity," *AIAA Journal*, Vol. 46, No. 7, 2008, pp. 1803–1813.

[93] Amsallem, D., Cortial, J., and Farhat, C., "Towards real-time computational-fluid-dynamics-based aeroelastic computations using a database of reduced-order information," *AIAA Journal*, Vol. 48, No. 9, 2010, pp. 2029–2037.

[94] Manohar, K., Brunton, B. W., Kutz, J. N., and Brunton, S. L., "Data-driven sparse sensor placement for reconstruction: Demonstrating the benefits of exploiting known patterns," *IEEE Control Systems Magazine*, Vol. 38, No. 3, 2018, pp. 63–86.

[95] Penrose, R., "A generalized inverse for matrices," *Mathematical Proceedings of the Cambridge Philosophical Society*, Vol. 51, Cambridge University Press, 1955, pp. 406–413.

[96] Wentland, C. R., Huang, C., and Duraisamy, K., "Investigation of sampling strategies for reduced-order models of rocket combustors," *AIAA Scitech 2021 Forum*, 2021, p. 1371.

[97] Drmac, Z. and Gugercin, S., "A new selection operator for the discrete empirical interpolation method—Improved a priori error bound and extensions," *SIAM Journal on Scientific Computing*, Vol. 38, No. 2, 2016, pp. A631–A648.

[98] Gloub, G. H. and Van Loan, C. F., "Matrix Computations," *Johns Hopkins Universtiy Press, 3rd edtion*, 1996.

[99] Homescu, C., Petzold, L. R., and Serban, R., "Error estimation for reduced-order models of dynamical systems," *SIAM Journal on Numerical Analysis*, Vol. 43, No. 4, 2005, pp. 1693–1714.

[100] Ojha, V., Fidkowski, K. J., and Cesnik, C. E. S., "Adaptive high-order fluid-structure interaction simulations with reduced mesh-motion errors," *AIAA Journal*, Vol. 59, No. 6, 2021.

[101] Doetsch, K. and Fidkowski, K. J., "Combined entropy and output-based adjoint approach for mesh refinement and error estimation," *AIAA Journal*, Vol. 57, No. 8, 2019.

[102] Fidkowski, K. J., Ceze, M. A., and Roe, P. L., "Entropy-based drag error estimation and mesh adaptation in two dimensions," *AIAA Journal of Aircraft*, Vol. 49, No. 5, September-October 2012, pp. 1485–1496.

[103] Roe, P. L., "Approximate Riemann solvers, parameter vectors, and difference schemes," *Journal of Computational Physics*, Vol. 43, No. 2, 1981, pp. 357–372.

[104] Bassi, F. and Rebay, S., "Numerical evaluation of two discontinuous Galerkin methods for the compressible Navier–Stokes equations," *International Journal for Numerical Methods in Fluids*, Vol. 40, No. 1-2, 2002, pp. 197–207.

[105] Peherstorfer, B. and Willcox, K., "Online adaptive model reduction for nonlinear systems via low-rank updates," *SIAM Journal on Scientific Computing*, Vol. 37, No. 4, 2015, pp. A2123–A2150.

[106] Hartmann, R., Held, J., Leicht, T., and Prill, F., "Error estimation and adaptive mesh refinement for aerodynamic flows," *ADIGMA-A European Initiative on the Development of Adaptive Higher-Order Variational Methods for Aerospace Applications*, Springer, 2010, pp. 339–353.

[107] Kast, S. M. and Fidkowski, K. J., "Output-based mesh adaptation for high order Navier-Stokes simulations on deformable domains," *Journal of Computational Physics*, Vol. 252, No. 1, 2013, pp. 468–494.

[108] Fidkowski, K. J., "Output-based space–time mesh optimization for unsteady flows using continuous-in-time adjoints," *Journal of Computational Physics*, Vol. 341, 2017, pp. 258–277.

[109] Ding, K. and Fidkowski, K., "Acceleration of Adjoint-Based Adaptation through Sub-Iterations for Unsteady Simulations," *AIAA Scitech 2021 Forum*, 2021, p. 0155.

[110] Fidkowski, K. J. and Luo, Y., "Output-based space-time mesh adaptation for the compressible Navier-Stokes equations," *Journal of Computational Physics*, Vol. 230, 2011, pp. 5753–5773.

[111] Fidkowski, K. J. and Roe, P. L., "An entropy adjoint approach to mesh refinement," *SIAM Journal on Scientific Computing*, Vol. 32, No. 3, 2010, pp. 1261–1287.

[112] Kenway, G. K. W. and Martins, J. R. R., "High-Fidelity Aerostructural Optimization of the Airbus XRF1 Aircraft Configuration," Tech. rep., Multidisciplinary Design Optimization Laboratory, University of Michigan, 2016.

[113] Mader, C. A., Kenway, G. K. W., Yildirim, A., and Martins, J. R. R. A., "ADflow—An open-source computational fluid dynamics solver for aerodynamic and multidisciplinary optimization," *Journal of Aerospace Information Systems*, 2020.

[114] Spalart, P. and Allmaras, S., "A one-equation turbulence model for aerodynamic flows," *30th Aerospace Sciences Meeting and Exhibit*, 1992, p. 439.

[115] Stabile, G., Hijazi, S., Mola, A., Lorenzi, S., and Rozza, G., "POD-Galerkin reduced order methods for CFD using finite volume discretisation: vortex shedding around a circular cylinder," *Communications in Applied and Industrial Mathematics*, 2017.

[116] Lorenzi, S., Cammi, A., Luzzi, L., and Rozza, G., "POD-Galerkin method for finite volume approximation of Navier–Stokes and RANS equations," *Computer Methods in Applied Mechanics and Engineering*, Vol. 311, 2016, pp. 151–179.

[117] Galbally, D., *Nonlinear model reduction for uncertainty quantification in large-scale inverse problems: application to nonlinear convection-diffusion-reaction equation*, Ph.D. thesis, Massachusetts Institute of Technology, 2008.

[118] Chaturantabut, S., *Dimension Reduction for Unsteady Nonlinear Partial Differential Equations via Empirical Interpolation Methods*, Ph.D. thesis, Rice University, 2009.

[119] Lee, M. W., *On Improving the Predictable Accuracy of Reduced-order Models for Fluid Flows*, Ph.D. thesis, Duke University, 2020.

[120] Li, J., Cai, J., and Qu, K., "Adjoint-based two-step optimization method using proper orthogonal decomposition and domain decomposition," *AIAA Journal*, Vol. 56, No. 3, 2018, pp. 1133–1145.

[121] He, J. and Durlofsky, L. J., "Constraint reduction procedures for reduced-order subsurface flow models based on POD–TPWL," *International Journal for Numerical Methods in Engineering*, Vol. 103, No. 1, 2015, pp. 1–30.

[122] Demkowicz, L. and Gopalakrishnan, J., "A class of discontinuous Petrov–Galerkin methods. II. Optimal test functions," *Numerical Methods for Partial Differential Equations*, Vol. 27, No. 1, 2011, pp. 70–105.

[123] Clapham, C., Nicholson, J., and Nicholson, J. R., *The Concise Oxford Dictionary of Mathematics*, Oxford University Press, 2014.

[124] Trefethen, L. N. and Bau III, D., *Numerical Linear Algebra*, Vol. 50, SIAM, 1997.

[125] Amsallem, D. and Farhat, C., "An online method for interpolating linear parametric reduced-order models," *SIAM Journal on Scientific Computing*, Vol. 33, No. 5, 2011, pp. 2169–2198.

[126] Hochreiter, S. and Schmidhuber, J., "Long short-term memory," *Neural Computation*, Vol. 9, No. 8, Nov. 1997, pp. 1735–1780.

[127] Castro-Díaz, M., Hecht, F., Mohammadi, B., and Pironneau, O., "Anisotropic unstructured mesh adaption for flow simulations," *International Journal for Numerical Methods in Fluids*, Vol. 25, No. 4, 1997, pp. 475–491.

[128] Fidkowski, K. J. and Chen, G., "Metric-based, goal-oriented mesh adaptation using machine learning," *Journal of Computational Physics*, Vol. 426, 2021, pp. 109957.