Model Reduction of Nonlinear Fire Dynamics Models

Alan M. Lattimer

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Mathematics

Jeffrey T. Borggaard, Co-chair
Serkan Gugercin, Co-chair
John A. Burns
Lizette Zietsman

March 23, 2016
Blacksburg, Virginia

Keywords: Model Reduction, Fire Models, IRKA, POD, Discrete-Time Systems

Model Reduction of Nonlinear Fire Dynamics Models

Alan M. Lattimer

(ABSTRACT)

Due to the complexity, multi-scale, and multi-physics nature of the mathematical models for fires, current numerical models require too much computational effort to be useful in design and real-time decision making, especially when dealing with fires over large domains. To reduce the computational time while retaining the complexity of the domain and physics, our research has focused on several reduced-order modeling techniques. Our contributions are improving wildland fire reduced-order models (ROMs), creating new ROM techniques for nonlinear systems, and preserving optimality when discretizing a continuous-time ROM. Currently, proper orthogonal decomposition (POD) is being used to reduce wildland fire-spread models with limited success. We use a technique known as the discrete empirical interpolation method (DEIM) to address the slowness due to the nonlinearity. We create new methods to reduce nonlinear models, such as the Burgers' equation, that perform better than POD over a wider range of input conditions. Further, these ROMs can often be constructed without needing to capture full-order solutions *a priori*. This significantly reduces the off-line costs associated with creating the ROM. Finally, we investigate methods of time-discretization that preserve the optimality conditions in a certain norm associated with the input to output mapping of a dynamical system. In particular, we are able to show that the Crank-Nicholson method preserves the optimality conditions, but other single-step methods do not. We further clarify the need for these discrete-time ROMs to match at ∞ in order to ensure local optimality.

Model Reduction of Nonlinear Fire Dynamics Models

Alan M. Lattimer

(GENERAL AUDIENCE ABSTRACT)

Large fires, such as industrial, coal mine, and wildland fires, represent a significant impact worldwide with regards to property damage, loss of life, and suppression costs. According to the National Interagency Fire Center, the amount of land burned by wildland fires in the United States in 2015 exceeded 10 million acres. The fire suppression costs alone for these fires was greater than $2.1 billion. This cost continues to grow and was at its highest level in 2015. Due to the complexity, multi-scale, and multi-physics nature of the mathematical models for fires, current numerical models require too much computational effort to be useful in design and real-time decision making, especially when dealing with fires over large domains. To reduce the computational time, there has been some work done in the area of reduced-order modeling of fire models. However, the majority of the effort has been with either simplifying the domain (e.g. network models), or by simplifying the underlying physics. Reduced-order modeling offers a third approach - retain the complexity of the domain and physics, but find a more efficient way to expose the essential input-output behavior of the model. Our research focuses on techniques that lead toward a fully-realized strategy for reducing the mathematical models related to fires. Our contributions to the field are to improve existing model reduction techniques for fires, create new reduced-order model techniques for nonlinear systems, and show when time discretization of the continuous-time reduced-order model preserves optimality under a certain norm.

Dedicated to my family.

Preston and Miranda

For being there as your father pursued his dreams.

Kara

(I do not know what it is about you that closes

and opens;only something in me understands

the voice in your eyes is deeper than all roses)

nobody,not even the rain,has such small hands

e.e. cummings

Acknowledgments

An endeavor such as this would have never been successful without the incredible support I received from faculty, friends, and family. While there is no way to thank everyone that has helped along the way, I would like to point out several people that have provided immense support and guidance along the way.

First, I would like to thank NIOSH for their funding under contract number 200-2014-59669. Their funding was instrumental in the coal mine and plume fire research discussed in Chapter 5. I would also like to give a special thanks to Dr. Kray Luxbacher and Ali Haghighat for all of their support and for including me in this great research team.

To my committee members, Dr. John Burns and Dr. Lizette Zietsman, I would have never made it to this point without your guidance. I really appreciate the frank advice about my research and education. To Dr. John Rossi, your passion is infectious. I knew the moment that I sat down in the first class I had with you that this was my path. Thank you for the conversations, laughs, and encouragement along the way.

I cannot begin to thank my advisers, Dr. Serkan Gugercin and Dr. Jeff Borggaard, enough. Both professionally and personally, you have been extraordinary. Your patient guidance along the way was essential in my success. You allowed me the space to grow and learn, but

were still present with advice and suggestions whenever necessary. You seemed to always be willing to drop everything to help. I am excited about working together for many years to come.

From a personal standpoint, there is no way I would have made it without my family and friends. Mom, thanks for just letting me soar, even when it looked like I would crash. Jonathan, Bryson, and Lynne thanks for all the talks, comfort and prayers. Mother and Granddaddy, thank you for being the true spiritual giants in my life. To my brothers, Brian and Charles, I love you! Brian, thanks for teaching me all the engineering background that I was sorely missing. Charles, thanks for the wordsmithing. To Kelli, thanks for showing me the ropes in grad school. To George, Chris, Kevin, Joe, Anthony, and Mark; let's go raise a glass and celebrate. Who would have thought I would have made it here? Thanks to all the folks at The Next Door Bake Shop. I really appreciate you letting me set up my writing "office" there.

Finally, I want to give an extra-special thanks to my family, Kara, Preston, and Miranda. You guys are my rock and my refuge. You hold me up, and you keep me grounded. You love and support me unconditionally. It is the three of you that give my life's work meaning. Thank you!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Large fires, such as industrial, coal mine, and wildland fires, represent a significant impact worldwide with regards to property damage, loss of life, and suppression costs. According to the National Interagency Fire Center, the amount of land burned by wildland fires in the United States in 2015 exceeded 10 million acres. The fire suppression costs alone for these fires was greater than $2.1 billion. This cost continues to grow and was at its highest level in 2015. This cost presents and inherent need for fast, high-fidelity models that provide real-time or better than real-time analysis of a fire, so that timely decisions regarding evacuations, fire suppression strategies, and on-scene personnel deployments can be made. Further, as we look to leverage unmanned and autonomous vehicle technologies, the need for faster embedded models that capture the true dynamics of the fire just increase.

Due to the complexity, multi-scale, and multi-physics nature of mathematical models for fires, current numerical approaches are extremely costly, especially when dealing with fires over

large domains. To reduce the computation time, the majority of current efforts have been with either simplifying the domain (e.g. network models), or by simplifying the underlying physics. Reduced-order modeling takes a third approach; retain the complexity of the domain and physics, but reduce the underlying mathematical complexity of the model.

Our research focuses on techniques leading to a fully-realized strategy for reducing the mathematical models related to fires. Our contribution to the field is to improve existing model reduction techniques for fires, create new reduced-order model techniques for nonlinear systems, and show when time discretization of the continuous-time reduced-order model preserves optimality under a certain norm. In particular, we design new methods that do not depend on prior data to be collected for reducing the types of nonlinear models that arise when looking at fires. We also combine methods to leverage the advantages associated with each of the individual model reduction strategies. For certain models, we improve the projection techniques used to create the reduced-order models to significantly reduce the computation time of the reduced-order model. Finally, we provide the mathematical theory required to justify the preservation of optimality in a certain norm when applying a time discretization method to solve a system of time-dependent ordinary differential equations.

Over the coarse of this work, we provide several numerical examples applicable to fires that reinforce the mathematical concepts presented. Several new algorithms are presented to explain the methodologies used. In the next section, we cover a more complete outline of the trajectory of the research covered in this dissertation.

1.2 Outline

In Chapter 2, we present the fundamental concepts that provide the underpinnings for the work done in this dissertation. We begin by giving an overview of a mathematical model for

fires (2.1)-(2.5). This model represents a fully-realized physical model for fires. This model incorporates several physical processes such as diffusion, convection, species transport, and combustion. As we move through the dissertation, we will refer back to this model as the primary context for the various examples we present. We then present some basics on dynamical systems and how we will define them for our work. The last section of the chapter covers the basics of projection-based model reduction methods, including two fundamental model reduction techniques used in our research - the proper orthogonal decomposition (POD) and iterative rational Krylov algorithm (IRKA) methods.

Chapter 3 covers our first efforts towards providing reduced-order models for fires. We examine the natural circulation flow induced by a temperature difference on the walls of a unit square. We show how this model is related to the fire model in (2.1)-(2.5) under a certain set of assumptions. For this model we linearized the flow about a steady-state flow and then used a modified version of IRKA to reduce the model. We showed how the this technique does an excellent job mapping inputs to outputs, but with the obvious limitation that all of the interesting nonlinear flows are removed. We used this experience to help develop new techniques for nonlinear model reduction.

Chapter 4 introduces new techniques for performing model reduction on nonlinear systems. Specifically, we designed methods that were more accurate over a larger range of input conditions for a certain class of problems. For the POD method, the quality of the reduced-order model is dependent on the data that is used to create the reduced-order bases. So, POD tends to produce accurate reduced-order models for nearby conditions, but is not as accurate for distant input conditions or solutions. With our new methods, developed in this chapter, we have techniques that are more accurate over a wider range of input conditions. While these methods are not truly input independent, they are moving the in that direction. Further, one of the methods developed, IRKA V$\oplus$W, does not require data samples to be

computed to build the reduced-order model. This method significantly reduces the off-line costs of building the reduced-order model.

We present three scenarios in Chapter 5 that demonstrate the power of the reduced-order modeling techniques for problems associated with the management and decision making during fire disasters. In the first scenario, we demonstrate how reduced-order models can help make design decisions for airflow to mitigate the dangers of methane fires in coal mines. The next scenario covers the simulation of fire plumes using reduced-order models. Here we give a set of criteria used to determine the quality of a reduced-order model in representing a true fire. The last example is a wildland fire-spread model, in which the computational time required to solve this model was significantly reduced by addressing the *lifting bottleneck* issue using the discrete empirical interpolation method (DEIM).

In Chapter 6, we examine the time-discretization techniques used to solve continuous-time ODE models resulting from the spatial discretization of a partial differential equation. This chapter also addresses which of the discretization methods preserve the first-order necessary conditions for optimality, under a specific norm. The Crank-Nicholson method preserves these conditions, with certain additions to the optimality conditions; conversely, the forward and backward Euler methods do not preserve the conditions. We cover specific information about how these discrete-time systems are related to their continuous-time counterparts. Finally, we present several theorems that show the preservation, or lack of preservation, of the first-order necessary conditions for optimality.

Chapter 2

Background

In this chapter, we discuss the mathematical background for the models and model-reduction techniques that will be employed throughout this dissertation. The material presented in this chapter is not meant to be a complete or thorough treatment; rather, it is an overview of the methods we have used in our work. Where there is more detail to be garnered about a certain technique, we have listed recommended references as a foundation for exploration.

In Section 2.1, we discuss the development of our fundamental fire model. Throughout the work, we will reference back to this model as the foundation for many subsequent simplified models. The culmination of this work is to be able to provide accurate reduced-order models for small plume fires. Inherently, fires represent a very complex, multi-scale problem that spans thermally driven fluid flows, reaction kinetics, and species transport. We discuss how we constructed our model with the goal of providing an accurate, surrogate reduced-order model.

Section 2.2 covers the basics of dynamical systems as we will use them. We define linear and nonlinear dynamical systems and discuss some of the relationships between them. For linear

systems, we discuss the Laplace transformation of the dynamical system and the resulting input to output mappings in the frequency domain.

We close the chapter by discussing the basics of reduced-order modeling in Section 2.3. We first examine the basics of projection-based model reduction techniques, showing how we project a full-order dynamical system to generate a reduced-order model. We then discuss optimal interpolation-based techniques and the mathematical basis that explains why these methods provide an accurate approximation for the reduced-order model output. Finally, we look at the most common nonlinear technique for model reduction - the proper orthogonal decomposition (POD). We briefly look at the mathematics underlying POD and discuss a few techniques for projecting the system using the POD basis.

2.1 Mathematical Model for Fire Dynamics

Fire dynamics models consist of modeling three constitutive equations involving airflow, species and energy transport, and chemical reactions. To begin with, we couple the momentum and energy equations under the assumption that the air is incompressible, although we plan to remove this assumption in future research. For the two-dimensional model, we let $\mathbf{u}(t, \mathbf{x}) = \mathbf{u} = [u\ v]^T$ be the velocity vector at a given point, $\mathbf{x} = (x, y)$, and time, t. We similarly define the velocity and spatial vectors for the three-dimensional model by adding the velocity component w and the spatial component z. We define the pressure as $p = p(t, \mathbf{x})$ and temperature as $T = T(t, \mathbf{x})$ at a given point and time. To track the fire, we must also track the mass fraction of the air, fuel, and product lumped species, $Z^A = Z^A(t, \mathbf{x})$, $Z^F = Z^F(t, \mathbf{x})$, and $Z^P = Z^P(t, \mathbf{x})$ respectively. This particular mathematical model gives rise to the coupled set of partial differential equations (PDEs) in (2.1)-(2.6). In the development of our mathematical model for fire, several sources for modeling heat transfer, buoyancy-driven

flows, fluid dynamics, combustion, and chemical reaction kinetics were consulted. For more detailed explanations of the physics leading to the model we put forth, the reader is referred to [37, 43, 58, 65, 74, 82, 89] and the associated references therein. We present the model below and then exposit over the different terms and their mathematical justification. Thus, consider

$$\frac{\partial \mathbf{u}}{\partial t} = -\mathbf{u}\cdot\nabla\mathbf{u} - \nabla p + \nu\nabla^2\mathbf{u} - \beta\mathbf{g}(T - T_\infty), \tag{2.1}$$

$$0 = \nabla\cdot\mathbf{u}, \tag{2.2}$$

$$\frac{\partial T}{\partial t} = -\mathbf{u}\cdot\nabla T + \alpha\nabla^2 T + (\rho_r\eta)\dot{q}''', \tag{2.3}$$

$$\frac{\partial Z_A}{\partial t} = -\mathbf{u}\cdot\nabla Z_A + \gamma_a\nabla^2 Z_A + \dot{m}_A''', \tag{2.4}$$

$$\frac{\partial Z_F}{\partial t} = -\mathbf{u}\cdot\nabla Z_F + \gamma_f\nabla^2 Z_F + \dot{m}_F'''. \tag{2.5}$$

$$\frac{\partial Z_P}{\partial t} = -\mathbf{u}\cdot\nabla Z_P + \gamma_f\nabla^2 Z_P + \dot{m}_P'''. \tag{2.6}$$

The constants used in the formulation of the mathematical model (2.1)-(2.6) are given in Table 2.1. Further, we note that $\dot{m}_A'''$, $\dot{m}_F'''$, and $\dot{m}_P'''$ are the mass reduction rates per unit volume of air, fuel, and products, respectively, and $\dot{q}'''$ is the heat release rate per unit volume. These values arise from combustion in the domain and are discussed in more detail in Section 2.1.2.

2.1.1 Buoyancy Force

The main model equations for the airflow are based on the Boussinesq equations (2.1)-(2.3). These arise from conservation of momentum (2.1), conservation of mass (2.2), and conservation of energy (2.3), see e.g. [31, 77]. The coupling of the thermal energy back to the momentum equation is modeled through the body force term in the momentum equa-

Table 2.1: Constants used in the fire model.

Constant	Definition
$\mathbf{g}$	Gravitational acceleration
ν	Kinematic viscosity
α	Thermal diffusivity
γ_a	Diffusion constant for air
γ_f	Diffusion constant for fuel
κ	Mass stoichiometry constant for air
η	Ratio of the heat of combustion $\Delta h_{c,F}$ to the specific heat C_p
ρ_r	Ratio of the density of a cell before combustion to after combustion

tion (2.1). Thus, the primary force acting on the air is the temperature-induced buoyancy, $-\beta\mathbf{g}(T-T_\infty)$, where we define the temperature contraction coefficient by $\beta(T(t,\mathbf{x})) = \frac{1}{T(t,\mathbf{x})}$ with the temperature, $T(t,\mathbf{x})$ given in Kelvin. It is worth noting that when the temperature does not vary much over the domain, *e.g.* $< 20°K$, we can treat β as a constant. This assumption greatly simplifies the computation. However, for large variations in temperature, such as those observed in a fire, a general nonlinear term, $\beta(T)$ must be retained to correctly model the dynamics in a fire.

2.1.2 Combustion

At the heart of all mathematical modeling of fires is the combustion, or chemically reactive flow, component. Our full-order fire models were computed using the Fire Dynamics Simulator (FDS) software developed at the National Institute of Standards and Technology (NIST). Therefore, to best match our reduced-order models (ROM) to the full-order model (FOM), we developed our combustion model based on the technical documents for FDS [74]. We simplified the model as informed by [43, 58, 89] to minimize the computational time while still maintaining the inherent physics of the model. To simplify the chemical reaction calculations, we utilize a lumped species approach to balancing the reaction equations

as described in [74]. Here the primitive species are lumped together, and we consider the simplified reaction equation

$$\text{Air} \ + \ \text{Fuel} \ \rightarrow \ \text{Products}. \tag{2.7}$$

For example, the chemical reaction associated with the burning of methane is given by the balanced reaction equation

$$CH_4 + 2O_2 + 7.52N_2 \rightarrow CO_2 + 2H_2O + 7.52N_2. \tag{2.8}$$

With the lumped species approach, we group terms together and handle the reaction stoichiometry at the lumped level. So for the methane reaction in (2.8), the lumped species approach yields

$$9.52(\underbrace{0.21O_2 + 0.79N_2}_{:=Z_A}) + \underbrace{CH_4}_{:=Z_F} \rightarrow 10.52(\underbrace{0.095CO_2 + 0.19H_2O + 0.715N_2}_{:=Z_P}) \tag{2.9}$$

In this way, we just need to track the lumped species as opposed to every individual species. Additionally, when we balance the equation we always set the stoichiometric coefficient for fuel to one. Therefore we can always write the reaction using the volume fractions as follows:

$$\kappa Z_A + Z_F \rightarrow (\kappa + 1)Z_P. \tag{2.10}$$

A correct balance of fuel and air need to exist for a fire to occur. Further, a sufficient amount of heat must be present to cause the reaction; more specifically, a correct balance of heat, fuel and air is required for combustion to occur. We only consider models with a single fuel for this research. It is possible to have several fuels, and this case is handled in detail in the FDS Technical Guide [74]. To determine whether the reaction is fuel or air limited, we determine the amount of air and fuel that will react by equation (2.11) where

the stoichiometry coefficient for air is given by κ. This assumption leads to:

$$\hat{Z}_F = (Z_A/\kappa, Z_F), \tag{2.11}$$
$$\hat{Z}_A = \kappa Z_F.$$

where $\hat{Z}_F$ and $\hat{Z}_A$ are the mass fractions of fuel and air, respectively, which will react during combustion. We consider excess air or fuel as inert when calculating the reaction. Next, we must determine if enough heat exists to initiate a reaction. We test to see if the reaction would bring the current temperature above the critical flame temperature, T_{CFT}. If it will not, then we set $\dot{q}''' = 0$ in (2.3), $\dot{m}_A''' = 0$ in (2.4), and $\dot{m}_F''' = 0$ in (2.5). Otherwise, $\dot{m}_F''' = \hat{Z}_F/\tau_{\text{mix}}$ in (2.5), $\dot{m}_A''' = \hat{Z}_A/\tau_{\text{mix}}$ in (2.4), and $\dot{q}''' = \dot{m}_F'''\Delta h^0_{f,F}$ in (2.3), where τ_{mix} is the time-scale for mixing and $\Delta h^0_{f,F}$ is the heat of formation for the fuel F.

2.2 Dynamical Systems

Throughout this dissertation, we will discuss many techniques used to reduce large-scale dynamical systems. These systems often arise from applying a spatial discretization, such as finite differences or finite elements, to a partial differential equation model with time dependence [6, 10, 12, 13, 17, 32, 33].

We first look at systems that are linear in both the state and input variables. Further, we assume that the measured output is some linear mapping from the state and input. If we define $\mathbf{x}(t) \in \mathbb{R}^n$ as the time-dependent state variable, $\mathbf{u}(t) \in \mathbb{R}^m$ as the time-dependent input or control function, and $\mathbf{y}(t) \in \mathbb{R}^p$ as the measured output over time, then we can define the linear dynamical system by Definition 2.1.

Definition 2.1. The *linear time-invariant (LTI) dynamical system*, $\boldsymbol{\Sigma}(\mathbf{E}, \mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$, is

given by equation (2.12)

$$\boldsymbol{\Sigma}: \begin{cases} \mathbf{E}\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \\ \mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) \end{cases}, \tag{2.12}$$

where $\mathbf{E}, \mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{B} \in \mathbb{R}^{n\times m}$, $\mathbf{C} \in \mathbb{R}^{p\times n}$, and $\mathbf{D} \in \mathbb{R}^{p\times m}$.

We note that $\mathbf{D}$ is simply a linear mapping from the input directly to the output, and for most of the systems we investigate, this will be $\mathbf{D} = \mathbf{0}$. For LTI systems (2.12), we often desire to look at a direct mapping from the input to the output. It is noted that taking the Laplace transform of the LTI system $\boldsymbol{\Sigma}(\mathbf{E}, \mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$ results in the *frequency-domain* description of the system [4, 83, 96] given by equation (2.13).

$$\hat{\mathbf{y}}(s) = \mathbf{C}(s\mathbf{E} - \mathbf{A})^{-1}\mathbf{B}\hat{\mathbf{u}}(s) + \mathbf{D}\hat{\mathbf{u}}(s), \tag{2.13}$$

where $\hat{\mathbf{y}}$ and $\hat{\mathbf{u}}$ are the Laplace transforms of the input and output respectively. We can now define the transfer function.

Definition 2.2. The finite-dimensional system $\boldsymbol{\Sigma}(\mathbf{E}, \mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$ defined by (2.12) has the *transfer function* $G(s)$ given by

$$\mathbf{G}(s) = \mathbf{C}(s\mathbf{E} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}. \tag{2.14}$$

For the multiple input/multiple output (MIMO) case, $\mathbf{G}(s)$ is a matrix valued rational function, whereas in the single input, single output (SISO) case it is scalar valued. For a more thorough treatment of linear systems theory and the role of the transfer function, the reader is referred to [4, 45, 83, 86, 96] and the references contained therein.

The discretization of certain PDE models results in a dynamical system that is not linear. As we will see in Section 2.3, the type of nonlinearity often determines what methods are

available to reduce the order of the model. For our purposes, we will generally categorize these nonlinear systems as bilinear, quadratic bilinear, or general nonlinear systems. While all of the nonlinear systems could be lumped together and defined as a general unstructured nonlinearity, we will separate them to leverage the structure of certain nonlinearities when applying model reduction techniques later in the dissertation.

Definition 2.3. The *bilinear dynamical system*, $\boldsymbol{\zeta}(\mathbf{E}, \mathbf{A}, \mathbf{N}, \mathbf{B}, \mathbf{C}, \mathbf{D})$, is bilinear in state and control (or input) and is given by

$$\boldsymbol{\zeta} : \begin{cases} \mathbf{E}\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \sum_{k=1}^{m} \mathbf{N}_k \mathbf{x}(t) u_k(t) + \mathbf{B}\mathbf{u}(t) \\ \mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) \end{cases}, \tag{2.15}$$

where $\mathbf{E}, \mathbf{A} \in \mathbb{R}^{n \times n}$, $\mathbf{B} \in \mathbb{R}^{n \times m}$, $\mathbf{C} \in \mathbb{R}^{p \times n}$, $\mathbf{D} \in \mathbb{R}^{p \times m}$, and $\mathbf{N}_k \in \mathbb{R}^{n \times n}$ for all $k = 1, \ldots, m$. Further the vector valued input function $\mathbf{u}(t)$ is defined by $\mathbf{u}(t) = [u_1(t), \ldots, u_m(t)]$, where each $u_k(t)$ is a scalar-valued function.

Definition 2.4. The *quadratic bilinear dynamical system*, $\boldsymbol{\eta}(\mathbf{E}, \mathbf{A}, \mathbf{H}, \mathbf{N}, \mathbf{B}, \mathbf{C}, \mathbf{D})$, is characterized by being quadratic in state and bilinear in state and control (or input) and is given by

$$\boldsymbol{\eta} : \begin{cases} \mathbf{E}\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{H}(\mathbf{x}(t) \otimes \mathbf{x}(t)) + \sum_{k=1}^{m} \mathbf{N}_k \mathbf{x}(t) u_k(t) + \mathbf{B}\mathbf{u}(t) \\ \mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) \end{cases}, \tag{2.16}$$

where $\mathbf{E}, \mathbf{A} \in \mathbb{R}^{n \times n}$, $\mathbf{H} \in \mathbb{R}^{n \times n^2}$, $\mathbf{B} \in \mathbb{R}^{n \times m}$, $\mathbf{C} \in \mathbb{R}^{p \times n}$, $\mathbf{D} \in \mathbb{R}^{p \times m}$, and $\mathbf{N}_k \in \mathbb{R}^{n \times n}$ for all $k = 1, \ldots, m$. Further the vector valued input function $\mathbf{u}(t)$ is defined by $\mathbf{u}(t) = [u_1(t), \ldots, u_m(t)]$, where each $u_k(t)$ is a scalar-valued function.

Definition 2.5. The *general nonlinear dynamical system* where there could be time-dependent

nonlinearities in the state and/or control (or input) given by $\mathbf{f}[\mathbf{x}, \mathbf{u}, t]$ is defined by the

$$\begin{aligned} \mathbf{E}\dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{f}[\mathbf{x}(t), \mathbf{u}(t), t], \\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t), \end{aligned} \tag{2.17}$$

where $\mathbf{E}, \mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{B} \in \mathbb{R}^{n\times m}$, $\mathbf{C} \in \mathbb{R}^{p\times n}$, and $\mathbf{D} \in \mathbb{R}^{p\times m}$. Further we restrict the order of $\mathbf{f}$ such that $\mathbf{f} = o(\|x\|, \|u\|)$. If we group the linear and control terms with the nonlinearity as $\mathbf{g}[\mathbf{x}(t), \mathbf{u}(t), t] = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{f}[\mathbf{x}(t), \mathbf{u}(t), t]$, then we can simplify (2.17) to

$$\begin{aligned} \mathbf{E}\dot{\mathbf{x}}(t) &= \mathbf{g}[\mathbf{x}(t), \mathbf{u}(t), t], \\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t). \end{aligned} \tag{2.18}$$

2.3 Reduced-Order Modeling

Model reduction or reduced-order modeling is a process of taking a very large system of equations and projecting it onto a much smaller subspace. The main goals of the projection are to significantly reduce the size of the system, often by several orders of magnitude, and to project the system in such a way as to preserve underlying behaviors of the original system. For systems with linear dynamics, there are numerous methodologies for effective, optimal model reduction. For example, one can apply Gramian-based methods such as Balanced Truncation [80, 81] or rational interpolation-based methods such Iterative Rational Krylov Algorithm [53]. These transfer function-based methods have been recently extended to systems with special nonlinearities, for example, to bilinear [10, 14, 39] and quadratic nonlinearities [15, 48]. We refer the reader to [6, 13, 17] for recent surveys on model reduction in general. For general nonlinearities, model reduction is most commonly achieved by Proper

Orthogonal Decomposition (POD) [8, 18, 60, 63, 69, 70, 80]. In the following sections, we discuss the basics of model reduction and the general techniques that were used to reduce systems related to the fire model given in (2.1)-(2.6).

2.3.1 Basic Framework

Consider an LTI dynamical system $\mathbf{\Sigma}(\mathbf{E}, \mathbf{A}, \mathbf{B}, \mathbf{C})$ defined by (2.12), where $\mathbf{D} = \mathbf{0}$. The goal is to find a reduced-order model of size $r \ll n$ such that

$$\begin{aligned} \mathbf{E}_r \dot{\mathbf{x}}_r(t) &= \mathbf{A}_r \mathbf{x}_r(t) + \mathbf{B}_r \mathbf{u}(t), \\ \mathbf{y}_r(t) &= \mathbf{C}_r \mathbf{x}_r(t), \end{aligned} \tag{2.19}$$

where $\mathbf{E}_r, \mathbf{A}_r \in \mathbb{R}^{r\times r}$, $\mathbf{B}_r \in \mathbb{R}^{r\times m}$, $\mathbf{C}_r \in \mathbb{R}^{p\times r}$. Further we seek a model such that $\mathbf{y}_r(t) \in \mathbb{R}^p$ is close to $\mathbf{y}(t)$ over the time interval $(0, t_f)$ and/or $\mathbf{x}_r(t)$ is close to $\mathbf{x}(t)$ over the domain for the given time interval and a class of inputs $u(t)$. We will discuss what we mean by "close" depending on the context and method used. We create left and right projection vectors, $\mathbf{W}, \mathbf{V} \in \mathbb{R}^{n\times r}$, to satisfy the Petrov-Galerkin condition

$$\mathbf{W}^T(\mathbf{E}\mathbf{V}\dot{\mathbf{x}} - \mathbf{A}\mathbf{V}\mathbf{x} - \mathbf{B}\mathbf{u}) = 0. \tag{2.20}$$

If the full state of the system is approximated as $\mathbf{x} = \mathbf{V}\mathbf{x}_r$, then (2.19) holds with

$$\begin{aligned} \mathbf{E}_r &= \mathbf{W}^T\mathbf{E}\mathbf{V}, & \mathbf{A}_r &= \mathbf{W}^T\mathbf{A}\mathbf{V}, \\ \mathbf{B}_r &= \mathbf{W}^T\mathbf{B}, & \mathbf{C}_r &= \mathbf{C}\mathbf{V}. \end{aligned} \tag{2.21}$$

We note that when $\mathbf{V} = \mathbf{W}$, this is a Galerkin projection. With this framework, each distinct method becomes a new way to build $\mathbf{V}$ and $\mathbf{W}$, and the choice of $\mathbf{V}$ and $\mathbf{W}$ depends on the

goals of the reduced-order model. If we are concerned with minimizing the output error over a wide range of bounded inputs, then we will seek input-independent methods such as the interpolatory method described in Section 2.3.2. These input-independent methods are well suited to applications, such as controls or circuits, where we are not necessarily concerned with the entire state space but in matching output conditions. If we are concerned with predicting the state space for a nonlinear system, then another method, such as proper orthogonal decomposition (POD), may be more applicable. We focused on POD for our research of fire models, and an overview of this technique is given in section 2.3.3.

2.3.2 Interpolatory Model Reduction

An interpolatory method utilizing projections with multiple interpolation points was first suggested by Skelton et al. [34, 93, 94]. A computationally efficient extension to this method using Krylov subspaces based on the work by Ruhe [84] was developed in [46]. Choosing the interpolation for optimality in the $\mathcal{H}_2$ norm, defined later by Equation (2.26), has since been developed for several classes of systems. [14, 52, 53, 54, 90]

As stated in the Section 2.3.1, the goal of this technique is to minimize the error between the full model output $\mathbf{y}(t)$ and the reduced-order model output $\mathbf{y}_r(t)$ over a wide range of inputs $\mathbf{u}(t)$. For simplicity of the discussion, consider the SISO system $\boldsymbol{\Sigma}(\mathbf{E}, \mathbf{A}, \mathbf{b}, \mathbf{c}^T)$ defined by (2.12) where $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^n$. In this case, input and output functions, $u(t)$, $y(t)$, and $y_r(t)$, are now scalar valued. If needed, this can be extended to the multi-input, multi-output case as shown by Gugercin et al. [53]. For the SISO system, the full-order scalar-valued transfer function is given by

$$G(s) = \mathbf{c}^T(s\mathbf{E} - \mathbf{A})^{-1}\mathbf{b}. \tag{2.22}$$

Combining (2.13) and (2.22), the relationship between the input and output is given by

$$\hat{y}(s) = G(s)\hat{u}(s). \tag{2.23}$$

Using the same procedure on the reduced linear system (2.19), the transfer function for the reduced system $\widetilde{G}$ is

$$G_r(s) = \mathbf{c}_r^T(s\mathbf{E}_r - \mathbf{A}_r)^{-1}\mathbf{b}_r, \qquad \hat{y}_r(s) = G_r(s)\hat{u}(s). \tag{2.24}$$

We are able to keep $\|y - y_r\|$ small in a suitable norm by ensuring that the transfer functions are close in an appropriate norm. In particular the relationship of the L^∞ error in the output functions to the $\mathcal{H}_2$ error in the transfer functions is shown in [17, 53] to be

$$\max_{t>0} |y(t) - y_r(t)| = \|y - y_r\|_{L^\infty} \leq \|G - G_r\|_{\mathcal{H}_2} \|u\|_{L^2}, \tag{2.25}$$

where the error in the $\mathcal{H}_2$ norm is given by

$$\|G - G_r\|_{\mathcal{H}_2} = \left(\frac{1}{2\pi}\int_{-\infty}^{\infty} |G(i\omega) - G_r(i\omega)|^2 \, d\omega\right)^{1/2}. \tag{2.26}$$

So we see that minimizing $\|G - G_r\|_{\mathcal{H}_2}$ also minimizes $\|y - y_r\|_{L^\infty}$ and thus provides the optimal reduced-order model (ROM) of dimension r provided we can use this fact to determine the $\mathbf{V}$ and $\mathbf{W}$ that generate this ROM.

Theorem 2.6 (**First order necessary conditions (SISO)**). *Given the SISO LTI dynamical system with the transfer function* $G(s) = \mathbf{c}^T(s\mathbf{E} - \mathbf{A})^{-1}\mathbf{b}$, *the optimal* $\mathcal{H}_2$ *reduced-order*

transfer function $G_r(s) = \mathbf{c}_r^T(s\mathbf{E}_r - \mathbf{A}_r)^{-1}\mathbf{b}_r$ *satisfies*

$$G(-\lambda_i) = G_r(-\lambda_i), \tag{2.27}$$

$$G'(-\lambda_i) = G_r'(-\lambda_i), \tag{2.28}$$

where λ_i, $i = 1, \cdots, r$ *are the poles of* $G_r(s)$.

Proof. See Meier III and Luenberger [76]. □

Theorem 2.7 (First order necessary conditions (MIMO)). *Given the MIMO LTI dynamical system with the transfer function* $\mathbf{G}(s) = \mathbf{C}(s\mathbf{E} - \mathbf{A})^{-1}\mathbf{B}$, *then the optimal* $\mathcal{H}_2$ r*-dimensional reduced-order transfer function is* $\mathbf{G}_r(s) = \mathbf{C}_r(s\mathbf{E}_r - \mathbf{A}_r)^{-1}\mathbf{B}_r$. *Let* $\widetilde{\lambda}_1, \ldots, \widetilde{\lambda}_r$ *be the poles of the* $\mathbf{G}_r(s)$ *with the associated rank-1 matrix residues,* $\text{res}[\mathbf{G}_r(s), \widetilde{\lambda}_k] = \widetilde{\mathbf{c}}_k\widetilde{\mathbf{b}}_k^T$. *Then the reduced-order transfer function satisfies*

$$\mathbf{G}(-\widetilde{\lambda}_k)\widetilde{\mathbf{b}}_k = \mathbf{G}_r(-\widetilde{\lambda}_k)\widetilde{\mathbf{b}}_k, \tag{2.29}$$

$$\widetilde{\mathbf{c}}_k^T\mathbf{G}(-\widetilde{\lambda}_k) = \widetilde{\mathbf{c}}_k^T\mathbf{G}_r(-\widetilde{\lambda}_k), \tag{2.30}$$

$$\widetilde{\mathbf{c}}_k^T\mathbf{G}'(-\widetilde{\lambda}_k)\widetilde{\mathbf{b}}_k = \widetilde{\mathbf{c}}_k^T\mathbf{G}_r'(-\widetilde{\lambda}_k)\widetilde{\mathbf{b}}_k, \tag{2.31}$$

for $k = 1, \cdots, r$.

Proof. See Gugercin et al. [53]. □

Given a linear dynamical system $\boldsymbol{\Sigma}(\mathbf{E}, \mathbf{A}, \mathbf{b}, \mathbf{c}^T)$ as defined in (2.12) with the associated transfer function (2.22) for the SISO case, Algorithm 2.8 provides a method for iteratively solving for the optimal interpolation points $\{\sigma_i\}_{i=1}^r = \{-\lambda_i\}_{i=1}^r$ from Theorem 2.6.

Algorithm 2.8 (Iterative Rational Krylov Algorithm (IRKA), Gugercin et al. [52]).

1. Make an initial selection of σ_k for $k = 1, \ldots, r$ that is closed under conjugation. Select a convergence tolerance `tol`.
2. Choose $\mathbf{V}$ and $\mathbf{W}$ such that

$$\begin{aligned} \text{span}(\mathbf{V}) &= \text{span}\{(\sigma_1\mathbf{E} - \mathbf{A})^{-1}\mathbf{b} \cdots (\sigma_r\mathbf{E} - \mathbf{A})^{-1}\mathbf{b}\}, \\ \text{span}(\mathbf{W}) &= \text{span}\{(\sigma_1\mathbf{E} - \mathbf{A})^{-T}\mathbf{c} \cdots (\sigma_r\mathbf{E} - \mathbf{A})^{-T}\mathbf{c}\}. \end{aligned} \tag{2.32}$$

3. While the relative change $\|\sigma_{curr} - \sigma_{last}\| >$ `tol`
   (a) $\mathbf{E}_r = \mathbf{W}^T\mathbf{E}\mathbf{V}$ and $\mathbf{A}_r = \mathbf{W}^T\mathbf{A}\mathbf{V}$
   (b) Assign $\sigma_k \leftarrow -\lambda_k(\mathbf{E}_r, \mathbf{A}_r)$ for $k = 1, \ldots, r$.
   (c) Update $\mathbf{V}$ and $\mathbf{W}$ using the new σ_k such that the equations from (2.32) are satisfied.
4. Set $\mathbf{E}_r = \mathbf{W}^T\mathbf{E}\mathbf{V}$, $\mathbf{A}_r = \mathbf{W}^T\mathbf{A}\mathbf{V}$, $\mathbf{b}_r = \mathbf{W}^T\mathbf{b}$, and $\mathbf{c}_r^T = \mathbf{c}^T\mathbf{V}$.

We extend this to the MIMO case where $\mathbf{B} \in \mathbb{R}^{n\times m}$ and $\mathbf{C} \in \mathbb{R}^{p\times n}$, by additionally determining the tangential directions $\widetilde{\mathbf{b}}_k$ and $\widetilde{\mathbf{c}}_k^T$ for $k = 1, \ldots, r$ associated with the shifts σ_k. These are updated at each time step, and instead of satisfying (2.32), $\mathbf{V}$ and $\mathbf{W}$ must satisfy (2.33).

$$\begin{aligned} \text{span}(\mathbf{V}) &= \text{span}\{(\sigma_1\mathbf{E} - \mathbf{A})^{-1}\mathbf{B}\widetilde{\mathbf{b}}_1 \cdots (\sigma_r\mathbf{E} - \mathbf{A})^{-1}\mathbf{B}\widetilde{\mathbf{b}}_r\}, \\ \text{span}(\mathbf{W}) &= \text{span}\{(\sigma_1\mathbf{E} - \mathbf{A})^{-T}\mathbf{C}^T\widetilde{\mathbf{c}}_1 \cdots (\sigma_r\mathbf{E} - \mathbf{A})^{-T}\mathbf{C}^T\widetilde{\mathbf{c}}_r.\} \end{aligned} \tag{2.33}$$

The reader is referred to Gugercin et al. [53] and Gugercin et al. [52] for a more thorough explanation of IRKA including the necessary extensions to the MIMO case.

2.3.3 Proper Orthogonal Decomposition

In POD, one first obtains a matrix of snapshots using a number of benchmark simulations and then computes the model reduction basis from a truncated singular value decomposition (SVD) of this snapshot matrix. Finally, one applies a Galerkin projection of the governing equations onto the low-dimensional using POD basis. The result is a small system of ordinary differential equations (ODE) that can replace the complexity of the full-order models while retaining nearly the same accuracy as long as the physics of the system remain "close" to one of the benchmark simulations.

We will cover a brief overview of POD and how to project the system. The reader is referred to [8, 18, 60, 63, 69, 70, 80], and the references therein, for a more complete treatment of this topic.

We start by looking at how the POD projection matrix is created. We must first capture a series of full-order solutions to our dynamical system over some time interval. These solutions could be from a full-order simulation or data collected by a physical experiment. Since these solutions are typically captured at specific time intervals, we refer to them as *full-order data snapshots* or simply *data snapshots*. Then, we create a matrix of N data snapshots, $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$, where $\mathbf{x} \in \mathbb{R}^{n \times N}$ represent the individual full-order solutions that were previously generated. For this matrix, we create an orthonormal basis, $\mathbf{U}$ such that $\mathbf{u}_j \in \mathbb{R}^n$, for $j = 1, \ldots, N$ and

$$\mathbf{x}_i = \sum_{j=1}^{N} \gamma_{ji} \mathbf{u}_j, \qquad i = 1, \ldots, N. \tag{2.34}$$

Choosing $k < N$ of these basis vectors, we build an approximation $\hat{\mathbf{X}}$ to $\mathbf{X}$ such that

$$\hat{\mathbf{x}}_i = \sum_{j=1}^{k} \gamma_{ji} \mathbf{u}_j, \qquad i = 1, \ldots, N, \tag{2.35}$$

and $\hat{\mathbf{X}}$ is close to $\mathbf{X}$ in some average sense. If we want to minimize the 2-norm error between the full and approximated solutions at the snapshot times, i.e.

$$\mathbf{X}_* = \arg\min_{\hat{\mathbf{X}}} \|\mathbf{X} - \hat{\mathbf{X}}\|_2, \tag{2.36}$$

then this orthonormal basis can be obtained from the singular value decomposition (SVD)

Theorem 2.9 (**Singular Value Decomposition (SVD)**). *If* $\mathbf{A}$ *is a real* m*-by-*n *matrix then there exist orthogonal matrices*

$$\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_m] \in \mathbb{R}^{m \times m} \quad \textit{and} \quad \mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_n] \in \mathbb{R}^{n \times n}, \tag{2.37}$$

such that

$$\mathbf{U}^T \mathbf{A} \mathbf{V} = \operatorname{diag}(\sigma_1, \ldots, \sigma_p) \in \mathbb{R}^{m \times n} \quad \textit{and} \quad p = \min\{m, n\}, \tag{2.38}$$

where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$.

Proof. See Golub and Van Loan [44, p. 70]. □

Definition 2.10. Given a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ with $\operatorname{rank}(\mathbf{A}) = r \leq \min\{m, n\}$ has a singular value decomposition (SVD) where

$$\mathbf{A} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T, \quad \mathbf{U} \in \mathbb{R}^{n \times r}, \ \mathbf{\Sigma} \in \mathbb{R}^{r \times r}, \ \text{and } \mathbf{V} \in \mathbb{R}^{N \times r}. \tag{2.39}$$

This decomposition is called the *thin SVD*. Σ is a square diagonal matrix $\operatorname{diag}\{\sigma_1, \sigma_2, \ldots, \sigma_r\}$

with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$. See for example Golub and Van Loan [44].

A useful property of the SVD is the dyadic decomposition defined below.

Definition 2.11. Assume that $\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T$ is the thin SVD decomposition from (2.39), where $\mathbf{U} = [\mathbf{u}_1 \dots \mathbf{u}_r]$ and $\mathbf{V} = [\mathbf{v}_1 \dots \mathbf{v}_r]$. The *dyadic decomposition* of $\mathbf{A}$ is the sum of rank one outer products

$$\mathbf{A} = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + \ldots + \sigma_r \mathbf{u}_r \mathbf{v}_r^T. \tag{2.40}$$

Now with those terms defined we present a theorem from [44] that is critical in the justification of the POD technique. Theorem 2.12 essentially tells us that the first k vectors in $\mathbf{U}$ will give us the optimal approximation $\mathbf{X}_*$ from the minimization problem (2.36).

Theorem 2.12. *Let the SVD $\mathbf{A} \in \mathbb{R}^{m\times n}$ be given by Theorem 2.9. If $k < r = \text{rank}(A)$ and*

$$\mathbf{A}_k = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^T, \tag{2.41}$$

then

$$\min_{\text{rank}(\mathbf{B})=k} \|\mathbf{A} - \mathbf{B}\|_2 = \|\mathbf{A} - \mathbf{A}_k\|_2 = \sigma_{k+1}. \tag{2.42}$$

Proof. See Golub and Van Loan [44, p. 73, Thm 2.5.3]. □

We note that $\mathbf{A}_k$ is the dyadic decomposition (2.40) truncated to the first k terms. Further, if we choose the leading k columns of $\mathbf{U}$, written $\mathbf{U}_k$, then we can obtain an orthogonal basis that will be able to generate a minimizer to (2.42). For the remainder of this section, we let $\Phi = \mathbf{U}_k$ and refer to it as the POD basis for the system. It is worth noting that the quality of this basis largely depends on the quality of the data. In other words, the more snapshots and the richer the information contained in those snapshots is, the better the POD basis can be at reconstructing solutions for diverse initial conditions and inputs. The trade off here

is that if the snapshots become too rich or diverse, then it is difficult to obtain a low order POD basis. As discussed in [4, 18, 69], the quality of the basis is largely dependent on the decay of the singular values.

Once we have created the POD basis matrix, Φ, we project the system to a low order subspace. When we have a LTI dynamical system we can apply the Galerkin projection technique discussed in Section 2.3.1 with $\mathbf{V} = \mathbf{W} = \Phi$. In this section, we discuss the extension of this technique to nonlinear systems along with the issue of the *lifting bottleneck*. To obtain computational efficiency and reduce or eliminate the lifting bottleneck issue for nonlinear systems, we must employ a numerical algorithm that mitigates the repeated matrix-vector product of the form $\Phi^T\mathbf{g}(\Phi\mathbf{x})$. There are several numerical methods that can be employed to address this issue, such as collocation [7, 66], gappy POD [7, 19, 24, 25, 38], discrete empirical interpolation method (DEIM) [3, 11, 27, 35, 36], and finite-element subassembly methods [2], to name a few. Here we discuss a method to avoid this bottleneck issue for systems with polynomial nonlinearities by treating the POD bases as basis functions evaluated at each node.

Nonlinear Model Reduction Using POD

Suppose we have an order n nonlinear system as defined by (2.18). For simplification of the discussion, we consider nonlinear time-invariant systems where the nonlinearity can be written as $\mathbf{g}(\mathbf{x})$. Given a steady-state solution $\bar{\mathbf{x}}$, we approximate the state by

$$\mathbf{x}(t) \approx \widetilde{\mathbf{x}}(t) = \bar{\mathbf{x}} + \Phi\mathbf{x}_r(t), \tag{2.43}$$

where $\Phi \in \mathbb{R}^{n\times r}$ is the matrix of r POD basis vectors created using an SVD of the deviations of the $p > r$ snapshots, $\widetilde{\mathbf{S}} = [(\mathbf{s}_1 - \bar{\mathbf{x}}) \dots (\mathbf{s}_p - \bar{\mathbf{x}})]$. It is worth noting that the choice of $\bar{\mathbf{x}}$

depends on the goals of the reduced-order model and the problem itself. For example, $\bar{\mathbf{x}}$ could represent the mean of the snapshots, the initial state, or the final state. The details of how $\bar{\mathbf{x}}$ should be chosen are beyond the scope of this discussion, and the reader is referred to [8, 9, 18, 64, 70] for a more thorough treatment of the subject. Substituting (2.43) into (2.17) and performing a Galerkin projection to ensure that the residual is orthogonal to Φ and thus in the null space of the POD basis vectors, we obtain

$$\Phi^T \left(\frac{d[\bar{\mathbf{x}} + \Phi \mathbf{x}_r(t)]}{dt} - \mathbf{g}(\bar{\mathbf{x}} + \Phi \mathbf{x}_r(t)) \right) = 0. \tag{2.44}$$

Using the fact that $\Phi^T \Phi = \mathbf{I}_r$, rearranging and simplifying equation (2.44) results in the following r-dimensional reduced-order model for the nonlinear system,

$$\dot{\mathbf{x}}_r(t) = \Phi^T \mathbf{g}(\bar{\mathbf{x}} + \Phi \mathbf{x}_r(t)). \tag{2.45}$$

For $r \ll n$, (2.45) should provide a significant computational speedup over the full-order differential equation. However, in practice, this is may not the be case. Upon closer inspection, we see that that $\mathbf{g}$ is defined as a vector valued function, i.e. $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^n$. Therefore, to evaluate $\mathbf{g}$ from the r coefficients of $\mathbf{x}_r$, we must first lift $\mathbf{x}_r$ up to the full state space by $\Phi \mathbf{x}_r$, add it to the centering vector $\bar{\mathbf{x}}$, evaluate this using $\mathbf{g}$, and finally project that result back to the r-dimensional subspace by $\Phi^T \mathbf{g}$. This so-called *lifting bottleneck* is covered in more detail in [26, 27, 28], but the result is that, for the nonlinear portion of the system, the computational complexity returns to order n. As mentioned earlier, there are several techniques that can be used to mitigate this issue. We cover one technique in the next section that can be used when we have a polynomial nonlinearity.

Projecting the PDE using POD

To demonstrate this technique, we examine the two-dimensional Navier-Stokes equations for isothermal, incompressible flows given by

$$\frac{\partial \mathbf{u}}{\partial t} - \mu \nabla^2 \mathbf{u} + (\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla p = 0, \tag{2.46}$$

$$\nabla \cdot \mathbf{u} = 0. \tag{2.47}$$

This model is well-suited for typical flow speeds and temperatures of airflows found in coal mines and was used as a starting point in examining mine fires. For any given time, we can write the velocity flow of the air, $\mathbf{u} = [u \ v]^T \in \mathbb{R}^2$ in the two-dimensional domain $\mathbf{x} = [x \ y]^T$, as a linear combination of a centering vector $\bar{\mathbf{u}}(\mathbf{x}, t)$ and an infinite set of basis vectors. Using the approximation discussed in the previous section, we create an r-dimensional approximation of the velocity, (2.48)-(2.49), using r POD basis vectors, i.e. $\Phi = [\boldsymbol{\phi}_1(\mathbf{x}) \ \boldsymbol{\phi}_2(\mathbf{x}) \cdots \boldsymbol{\phi}_r(\mathbf{x})]$. Here we have decomposed the basis into its u and v components as $\boldsymbol{\phi}_i = [(\boldsymbol{\phi}_i^u)^T \ (\boldsymbol{\phi}_i^v)^T]^T$.

$$u(\mathbf{x}, t) \approx \widetilde{u}(\mathbf{x}, t) = \bar{u}(\mathbf{x}) + \sum_{i=1}^{r} \boldsymbol{\phi}_i^u(\mathbf{x})\mathbf{a}(t), \tag{2.48}$$

$$v(\mathbf{x}, t) \approx \widetilde{v}(\mathbf{x}, t) = \bar{v}(\mathbf{x}) + \sum_{j=1}^{r} \boldsymbol{\phi}_j^v(\mathbf{x})\mathbf{a}(t). \tag{2.49}$$

Using the same coefficient $\mathbf{a}(t)$ for both approximations allows us to enforce the incompressibility condition (2.47) in the reduced-order model. Then using these POD modes Φ as test functions, we can build the weak formulation of equation (2.46) by multiplying each side by the test functions and integrating. Applying a straight-forward integration by parts and gathering terms for each of the velocity directions, we obtain the weak form of the equation.

For example, the weak form in the u direction is given by

$$(u_t, \boldsymbol{\phi}^u) = -(uu_x + vu_y, \boldsymbol{\phi}^u) - 2\mu(u_x, \boldsymbol{\phi}^u_x) - \mu(u_y + v_x, \boldsymbol{\phi}^u_y). \tag{2.50}$$

We can now substitute the approximation for $\mathbf{u}$ from (2.48) and (2.49) into (2.50) for each $\boldsymbol{\phi}_i$, $i = 1, \ldots, r$ and apply the linearity properties of the inner product to obtain

$$\begin{aligned} a_i = &- (\bar{u}\bar{u}_x, \boldsymbol{\phi}^u_i) - \left(\bar{u} \sum \boldsymbol{\phi}^u_{k,x} a_k, \boldsymbol{\phi}^u_i\right) \\ &- \left(\bar{u}_x \sum \boldsymbol{\phi}^u_j a_j, \boldsymbol{\phi}^u_i\right) - \left(\sum \boldsymbol{\phi}^u_j a_j \sum \boldsymbol{\phi}^u_{k,z} a_k, \boldsymbol{\phi}^u_i\right) \\ &- (\bar{v}\bar{u}_y, \boldsymbol{\phi}^u_i) - \left(\bar{v} \sum \boldsymbol{\phi}^u_{k,y} a_k, \boldsymbol{\phi}^u_i\right) \\ &- \left(\bar{v}_y \sum \boldsymbol{\phi}^v_j a_j, \boldsymbol{\phi}^u_i\right) - \left(\sum \boldsymbol{\phi}^v_j a_j \sum \boldsymbol{\phi}^u_{k,z} a_k, \boldsymbol{\phi}^u_i\right) \\ &- 2\mu \left(\bar{u}_x, \boldsymbol{\phi}^u_{i,x}\right) - 2\mu \sum \left(\boldsymbol{\phi}^u_{k,x}, \boldsymbol{\phi}^u_{i,x}\right) a_k \\ &- \mu \left(\bar{u}_y, \boldsymbol{\phi}^u_{i,y}\right) - \mu \sum \left(\boldsymbol{\phi}^u_{k,y}, \boldsymbol{\phi}^u_{i,y}\right) a_k \\ &- \mu \left(\bar{v}_x, \boldsymbol{\phi}^u_{i,y}\right) - \mu \sum \left(\boldsymbol{\phi}^v_{k,x}, \boldsymbol{\phi}^u_{i,y}\right) a_k. \end{aligned} \tag{2.51}$$

Similar formulations are done for each velocity direction. The power in this method is that we can now apply a finite element discretization and precompute much of the inner product so that our discretized system is of order r, where we are solving for the coefficients of the POD basis, $\mathbf{a}(t)$. To see how this might be done, we look at the inner product $(\bar{u}\bar{u}_x, \boldsymbol{\phi}^u_i)$. First, let h represent the N finite element basis functions. Then we fully expand the inner

product and the functions in the finite element basis.

$$
\begin{aligned}
(\bar{u}\bar{u}_x, \boldsymbol{\phi}_i^u) &= \int_\Omega \sum_{m=1}^{N} \bar{u}_m h_m \sum_{n=1}^{N} \bar{u}_n h_{n,x} \boldsymbol{\phi}_i^u \, dA \\
&= \sum_{m=1}^{N}\sum_{n=1}^{N} \bar{u}_m \bar{u}_n \underbrace{\int_\Omega h_m h_{n,x} \boldsymbol{\phi}_i^u \, dA}_{[\mathbf{T}_{ux}^{(i)}]_{mn}}
\end{aligned} \tag{2.52}
$$

With $[\mathbf{T}_{ux}^{(i)}]_{mn}$ computed for each i, we compute the inner product for each i and assemble it into an r dimensional constant vector `cucux` as given by

$$
\texttt{cucux}^{(i)} = (\bar{u}\bar{u}_x, \boldsymbol{\phi}_i^u) = [\bar{\mathbf{u}}]_{m1}[\mathbf{T}_{ux}^{(i)}]_{mn}[\bar{\mathbf{u}}]_{n1} = (\bar{\mathbf{u}})^T \mathbf{T}_{ux}^{(i)} \bar{\mathbf{u}}, \quad i = 1, \ldots, r. \tag{2.53}
$$

where $\bar{\mathbf{u}} \in \mathbb{R}$ are the values of $\bar{u}(\mathbf{x})$ at n distinct points in the domain determined by the spatial discretization. For equation (2.53), repeated indices in the third term are considered to be implied summations over that index. All the constant terms are computed and then summed together into a single vector, $\mathbf{c}$. Additionally, we can construct constant matrices associated with the linear and quadratic terms. For example, the computation below shows how we precompute

$$
\begin{aligned}
\left(\bar{u}\sum \boldsymbol{\phi}_{j,x}^u \mathbf{a}_j, \boldsymbol{\phi}_i^u\right) &= [\bar{\mathbf{u}}]_{m1}[\mathbf{T}_{ux}^{(i)}]_{mn}[\boldsymbol{\phi}^u]_{nj}\mathbf{a}_j \\
&\Rightarrow \underbrace{\left(\bar{\mathbf{u}}^T \mathbf{T}^{(i)} \boldsymbol{\phi}\right)}_{\text{Precomputed}} \mathbf{a}(t), \quad i = 1, \ldots, r \text{ and } t \in [0, t_f]
\end{aligned} \tag{2.54}
$$

For the linear terms, we simply sum all the precomputed $r \times r$ matrices together to construct the reduced linear matrix, $\widetilde{\mathbf{A}}$, for the time-dependent ODE. The quadratic term is a little more complicated. First we precompute the matrices $\widetilde{\mathbf{Q}}_i$, $i = 1, \ldots, r$, that are a sum of $r \times r$ matrices that arise from terms like $\left(\sum \boldsymbol{\phi}_j^u a_j \sum \boldsymbol{\phi}_{k,z}^u a_k, \boldsymbol{\phi}_i^u\right)$. During the time integration, we assemble the reduced-order quadratic term using Algorithm 2.13.

Algorithm 2.13 (Quadratic Term Subassembly). Given matrices $\widetilde{\mathbf{Q}}_i$, $i = 1, \ldots, r$, and the POD coefficient vector $\mathbf{a}^n = \mathbf{a}(t^n)$ at time step t^n, the quadratic term $\widetilde{\mathbf{q}}(t^n) = [q_1^n, \ldots, q_r^n]^T$ is constructed as follows:

1. `for` $i = 1$ to r
   - $q_i^n = (\mathbf{a}^n)^T \widetilde{\mathbf{Q}}_i \mathbf{a}^n$;
2. end `for`

Now we have the following r-dimensional system of ordinary differential equations that can be solved to obtain the POD basis coefficients $a(t)$ for a given time interval $t = [0, t_f]$:

$$\dot{\mathbf{a}}(t) = \widetilde{\mathbf{q}}(t) + \widetilde{\mathbf{A}}\mathbf{a}(t) + \mathbf{c}. \tag{2.55}$$

Once constructed, this can be solved using a standard ODE solver. The advantage is that we never have to lift the problem back to the n-dimensional state space for the nonlinearity under consideration in equation (2.46). Using this technique, we have completely removed the lifting bottleneck for this nonlinearity while maintaining an exact computation of the nonlinear terms. The cost of this method is primarily in the precomputation and storage of the intermediate matrices needed to assemble the reduced-order ODE. These intermediate matrices are sparse and multiplied by thin $n \times r$ matrices, so the precomputation (off-line) cost is practical. Moreover, since the resulting matrices are all dimension r, the storage burden is minimal. Solving the ODE (2.55) and plugging the coefficients $\mathbf{a}(t)$ into our approximation in Equation (2.48), we can recover an approximation to the full-order model.

Chapter 3

A Natural Convection Flow Model

3.1 Introduction

Recall that Equations (2.1)-(2.3) represent the momentum and energy equations coupled together with the assumption that the fluid is incompressible. Under certain physical assumptions we can use these same equations to model the natural circulation of a fluid induced by a thermally-driven flow. Discretizing this system of PDEs in space results in a nonlinear dynamical system. For this chapter, we discuss the simplifying physical assumptions about the model that we have used. Linearizing the system around a steady-state flow results in a coupled set of linear differential algebraic equations (DAEs) that we can then reduce using interpolation-based methods. Linearizing a system like this is one approach that is taken to simplify the computation for large nonlinear systems. Our goal was to investigate the efficacy of applying reduced-order modeling techniques to these types of systems. Further, we wanted to see how well we could capture the input to output behavior of the system.

We first cover a description of the problem, and the physical assumptions that we made to

simplify our model. We then discuss the linearized DAE that is obtained and how the unique characteristics of the DAE system necessitate certain modifications to the IRKA technique described in Section 2.3.2. Finally, we present some numerical results of the accuracy of the reduced-order model for this problem.

3.2 The Boussinesq Equations

As a simplification to the fire model, we ignore the species transport in Equations (2.4)-(2.6) and focus on reducing a thermally-driven airflow given by Equations (2.1)-(2.3). For this problem, the primary body force is the temperature difference on opposing wall of the domain. Further, we assume that the temperature changes across the domain are less than $20°K$. This allows us to treat β as a constant as described in Section 2.1.1. Additionally, under these assumptions for temperature, we can consider the kinematic viscosity, ν, and thermal diffusivity, α, as constants. Finally, we assume there is no fuel present and therefore there is no combustion contribution to the energy equation. These assumptions lead to the well-known Boussinesq equations given by

$$\frac{\partial \mathbf{u}}{\partial t} = -\mathbf{u} \cdot \nabla \mathbf{u} - \nabla p + \nu \nabla^2 \mathbf{u} - \beta \mathbf{g}(T - T_\infty), \tag{3.1}$$

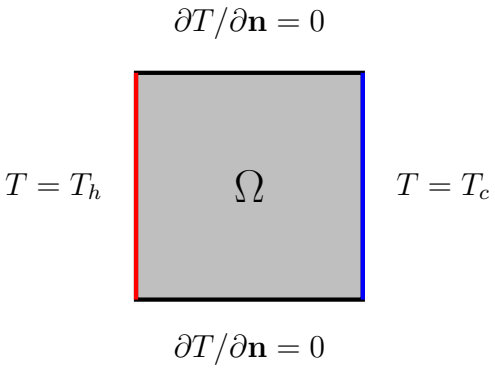
$$0 = \nabla \cdot \mathbf{u}, \tag{3.2}$$

$$\frac{\partial T}{\partial t} = -\mathbf{u} \cdot \nabla T + \alpha \nabla^2 T. \tag{3.3}$$

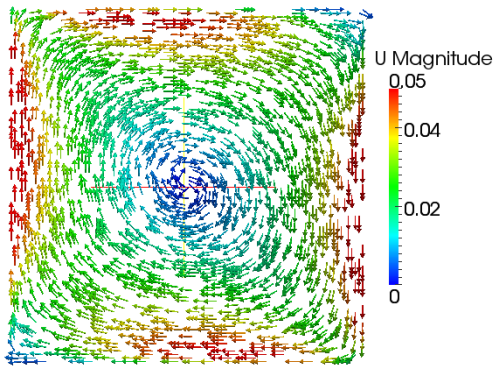
3.3 Example Problem Description

Consider the natural convection model in a two-dimensional square where there is a temperature difference between the right and left boundaries. The top and bottom boundaries

have a zero normal gradient, no-slip boundary condition as shown in Figure 3.1a. The temperature differential between the walls generates a rotational flow in the square, such as the flow shown in Figure 3.1b. Here the fluid rises near the hot wall, T_h, and sinks along the cold wall, T_c.



(a) Problem setup.



(b) Representative flow.

Figure 3.1: Rayleigh convection on the domain $[0, 1] \times [0, 1]$.

The flow of the fluid inside the square is induced by the buoyant force caused by the temperature difference on opposing walls. The Boussinesq equations (3.1)-(3.3) provide a means for modeling this type of flow [79]. Another characterization of this model is that it is simply the incompressible Navier-Stokes equations coupled to the convection-diffusion equation for temperature via a buoyancy term in the momentum equation.

We define the domain for this particular problem as $t \in (0, t_f)$ and $\mathbf{x} = (x, y) \in \Omega = [0, 1] \times [0, 1]$ with the boundary defined as $\partial\Omega = \partial\Omega_c \cup \partial\Omega_h \cup \partial\Omega_{tb}$. $\partial\Omega_h$ and $\partial\Omega_c$ are the hot and cold boundaries defined on the $x = 0$ and $x = 1$ faces respectively, and $\partial\Omega_{tb}$ is the combined top and bottom boundary $y = 0$ and $y = 1$ when $x \neq 0$ and $x \neq 1$. We set the

boundary conditions as follows:

$$\begin{aligned} u(\mathbf{x},t) &= 0, && \mathbf{x} \in \partial\Omega, \\ v(\mathbf{x},t) &= 0, && \mathbf{x} \in \partial\Omega, \\ \frac{\partial \mathbf{T}}{\partial \mathbf{n}}(\mathbf{x},t) &= 0, && \mathbf{x} \in \partial\Omega_{tb} \\ T(\mathbf{x},t) &= T_h(t) = T_\infty + \frac{1}{2}u(t), && \mathbf{x} \in \partial\Omega_h \\ T(\mathbf{x},t) &= T_c(t) = T_\infty - \frac{1}{2}u(t), && \mathbf{x} \in \partial\Omega_c. \end{aligned} \tag{3.4}$$

The reference room temperature is given by T_∞. By subtracting the last two lines, the control function is given by (3.5) centered around the reference temperature.

$$u(t) = T_h(t) - T_c(t), \tag{3.5}$$

The partial differential equations given in (3.1)-(3.2) were linearized about a mean flow and discretized using a Taylor-Hood finite element approximation. This discretization results in the Stokes-type descriptor system of index 2,

$$\mathbf{E}_{11}\dot{\mathbf{x}}_1 = \mathbf{A}_{11}\mathbf{x}_1 + \mathbf{A}_{12}\mathbf{x}_2 + \mathbf{B}_1\mathbf{u}(t), \tag{3.6}$$

$$\mathbf{0} = \mathbf{A}_{21}\mathbf{x}_1 + \mathbf{B}_2\mathbf{u}(t), \tag{3.7}$$

$$\mathbf{y} = \mathbf{C}_1\mathbf{x}_1 + \mathbf{C}_2\mathbf{x}_2 + \mathbf{D}\mathbf{u}(t), \tag{3.8}$$

where the state is $\mathbf{x}_1 = [\mathbf{u},\ \mathbf{T}]^T, \mathbf{x}_2 = [\mathbf{p}]^T$. Additionally, for the Boussinesq model with the given control function $u(t)$, $\mathbf{B}_2 = \mathbf{0}, \mathbf{C}_2 = \mathbf{0}$, and $\mathbf{D} = \mathbf{0}$. The output $\mathbf{y}(t)$ is the average vorticity for the cavity. The state and output equations in (3.6)-(3.8) can be written in the form of (2.12) if we embed E_{11} into a larger matrix with zero blocks and note that the block matrix A_{22} is zero. For a Stokes-type descriptor system of index 2, $\mathbf{E}$ is singular, and the

system is characterized by the facts that $\mathbf{E}_{11}$ is nonsingular, $\mathbf{A}_{12}$ and $\mathbf{A}_{21}^T$ have full column rank, and $\mathbf{A}_{21}\mathbf{E}_{11}^{-1}\mathbf{A}_{12}$ is nonsingular.

3.4 Model Reduction of Descriptor Systems

The focus for reducing this model is to maintain the input to output mapping. In terms of model reduction, we seek to minimize the error between the full-order and reduced-order outputs. Given the model in (3.6)-(3.8), we note that $\mathbf{E}$ is singular, being of the form

$$\mathbf{E} = \begin{bmatrix} \mathbf{E}_{11} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}. \tag{3.9}$$

Reducing this type of system is further complicated by the fact that the transfer function is not a strictly proper rational function. So, the transfer function $\mathbf{G}(s)$ can be decomposed into its strictly proper portion $\mathbf{G}_{sp}(s)$ and its polynomial portion $\mathbf{P}(s)$.

$$\mathbf{G}(s) = \mathbf{G}_{sp}(s) + \mathbf{P}(s), \qquad \widetilde{\mathbf{G}}(s) = \widetilde{\mathbf{G}}_{sp}(s) + \widetilde{\mathbf{P}}(s). \tag{3.10}$$

Because of this form, particular care must be taken to match the polynomial parts. As is demonstrated in Gugercin et al. [54], to have bounded $\mathcal{H}_2$ or $\mathcal{H}_\infty$ errors between the full-order and reduced-order transfer functions, the polynomial portions of the full-order and reduced-order systems must match exactly, i.e. $\mathbf{P}(s) = \widetilde{\mathbf{P}}(s)$. This is accomplished by first decoupling the system into a differential equation to solve for $\mathbf{x}_1$ and an algebraic equation to solve for $\mathbf{x}_2$, as shown in [54]. As described in [54, 90], this decoupling is done implicitly and thus the deflating projectors, which are expensive to compute, are never computed. Rather, we use a series of sparse indefinite linear matrix solves to accomplish the reduction and thus

the implicit decoupling. The differential equation system for $\mathbf{x}_1$ is then reduced using an iterative interpolatory algorithm similar to Algorithm 2.8. Gugercin et al. [54] discuss the specifics on how to enforce a Hermite interpolation of the transfer function of this descriptor system. In addition, to improve computational efficiency, Algorithm 3.1 was developed to find the projection matrices needed to provide a bi-tangential Hermite interpolation.

Algorithm 3.1 (**Interpolatory model reduction for Stokes-type descriptor systems of index 2; Gugercin et al. [54]).**

1. Given shifts $\{\sigma_i\}_{i=1}^r$ and tangent directions $\{\mathbf{b}_i\}_{i=1}^r$ and $\{\mathbf{c}_i\}_{i=1}^r$.
2. For $i = 1, \ldots, r$, solve

$$\begin{bmatrix} \sigma_i \mathbf{E}_{11} - \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{v}_i \\ \mathbf{z} \end{bmatrix} = \begin{bmatrix} \mathbf{B}_1 \mathbf{b}_i \\ \mathbf{0} \end{bmatrix}$$

$$\begin{bmatrix} \sigma_i \mathbf{E}_{11}^T - \mathbf{A}_{11}^T & \mathbf{A}_{21}^T \\ \mathbf{A}_{12}^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{w}_i \\ \mathbf{q} \end{bmatrix} = \begin{bmatrix} \mathbf{C}_1^T \mathbf{c}_i \\ \mathbf{0} \end{bmatrix}$$

3. $\mathbf{V} = [\mathbf{v}_1 \ \cdots \ \mathbf{v}_r]$, $\mathbf{W} = [\mathbf{w}_1 \ \cdots \ \mathbf{w}_r]$
4. $\mathbf{E}_r = \mathbf{W}^T \mathbf{E}_{11} \mathbf{V}$, $\mathbf{A}_r = \mathbf{W}^T \mathbf{A}_{11} \mathbf{V}$, $\mathbf{B}_r = \mathbf{W}^T \mathbf{B}_1$, $\mathbf{C}_r = \mathbf{C}_1 \mathbf{V}$, $\mathbf{D}_r = \mathbf{D}$

Now that we have a computationally effective method to compute $\mathbf{V}$ and $\mathbf{W}$, we can embed Algorithm 3.1 into the IRKA Algorithm 2.8 to find the optimum shifts and directions. This leads to the version of IRKA that is described in Algorithm 3.2.

Algorithm 3.2 (**IRKA for Stokes-type descriptor systems of index 2;Gugercin et al. [54]**).

1. Select initial shifts $\{\sigma_i\}_{i=1}^r$ and tangent directions $\{\mathbf{b}_i\}_{i=1}^r$ and $\{\mathbf{c}_i\}_{i=1}^r$.
2. Apply Algorithm 3.1 to obtain $\mathbf{E}_r$, $\mathbf{A}_r$, $\mathbf{B}_r$, $\mathbf{C}_r$, and $\mathbf{D}_r$
3. while (not converged)
   (a) Compute $\mathbf{Y}^*\mathbf{A}_r\mathbf{Z} = \text{diag}(\lambda_1, \ldots, \lambda_r)$ and $\mathbf{Y}^*\mathbf{E}_r\mathbf{Z} = \mathbf{I}$, where the columns of $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_r]$ and $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_r]$ are, respectively, the right and left eigenvectors of $\lambda\mathbf{E}_r - \mathbf{A}_r$.
   (b) $\sigma_i \leftarrow -\lambda_i$, $\mathbf{b}_i^T \leftarrow \mathbf{y}_i^*\widetilde{\mathbf{B}}$, and $\mathbf{c}_i \leftarrow \widetilde{\mathbf{C}}\mathbf{z}_i$ for $i = 1, \ldots, r$.
   (c) Apply Algorithm 3.1 to obtain $\mathbf{E}_r$, $\mathbf{A}_r$, $\mathbf{B}_r$, $\mathbf{C}_r$, and $\mathbf{D}_r$

   end while.

With this algorithm, we can now create an $\mathcal{H}_2$-optimal r-dimensional reduced-order model via projection as given by

$$\mathbf{E}_r\dot{\widetilde{\mathbf{x}}}_1(t) = \mathbf{A}_r\widetilde{\mathbf{x}}_1(t) + \mathbf{B}_r u(t), \tag{3.11}$$

$$y(t) = \mathbf{C}_r\widetilde{\mathbf{x}}_1(t), \tag{3.12}$$

that will allow us to solve for $\widetilde{\mathbf{x}}_1$, where $\mathbf{x}_1 \approx \mathbf{V}\widetilde{\mathbf{x}}_1$. Once we have calculated $\widetilde{\mathbf{x}}_1$, then $\mathbf{x}_2$ can be recovered using the algebraic equations given in Gugercin et al. [54], Wyatt [90].

3.5 Numerical Results

The temperature induced natural circulation in a 2D square was modeled using the Boussinesq equations and then linearized about a mean flow to create an index-2 Stokes-type

descriptor system. See Section 3.3 for specific details. For this example, $n_1 = 7301$ and $r = 15$, and the control function defined by (3.5) is given by the step function in Figure 3.2. The goal of these particular interpolatory methods is to minimize the error between

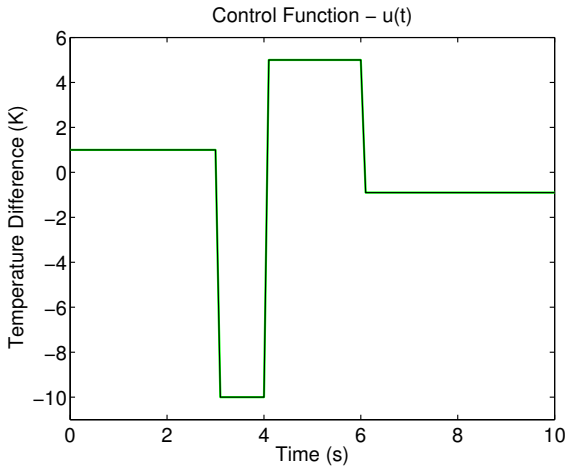


Figure 3.2: Control function for the natural circulation square.

the full-order and reduced-order transfer functions, and as a consequence minimize the error between the full and reduced order system response. As Figure 3.3 illustrates, the reduced-order transfer function is a very good approximation of the full-order transfer function for frequencies along the imaginary axis.

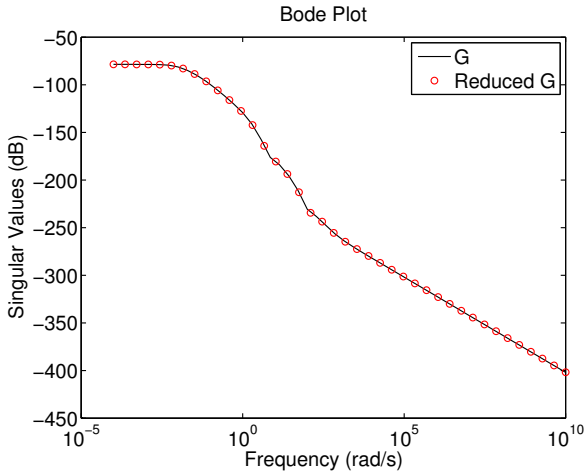
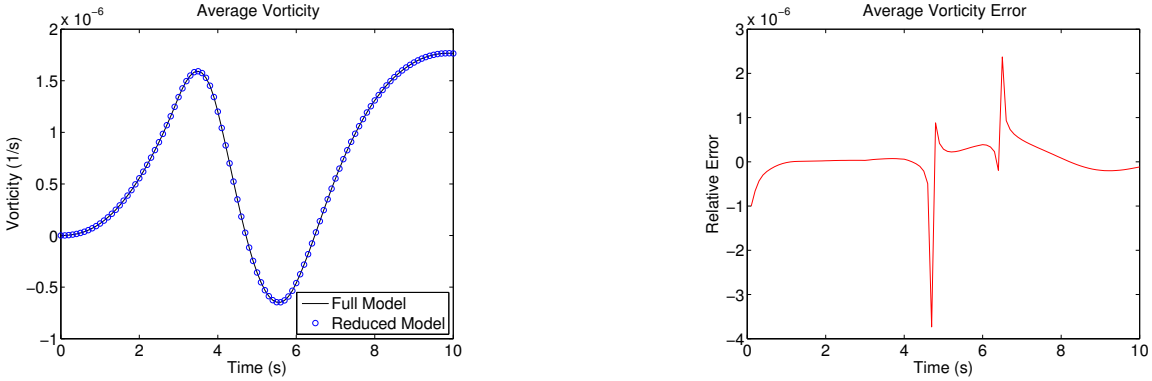


Figure 3.3: Frequency response of the full and reduced-order transfer functions.

As the theory in [53] and [54] suggest, this leads to a very small error of $\mathcal{O}(10^{-6})$ for the measured response, $y(t)$, as seen in Figure 3.4.



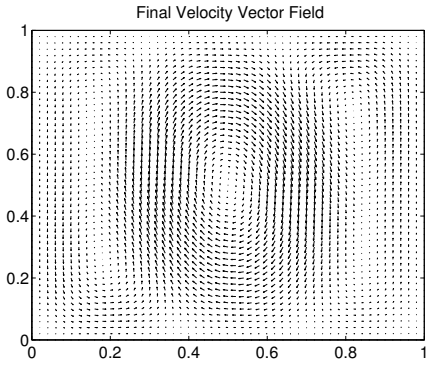
(a) System response.

(b) System response error.

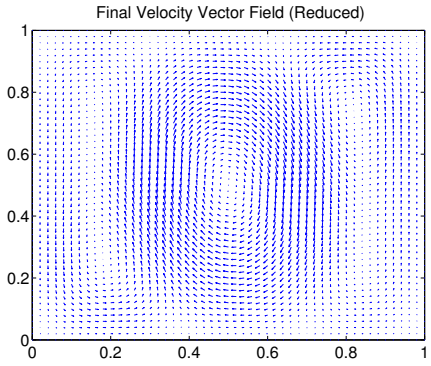
Figure 3.4: Comparison of the full-order and reduced-order system response, $y(t)$ for the natural circulation problem.

Somewhat unexpectedly, the reduced-order model also provided a good approximation for the state space as is seen in Figure 3.5.

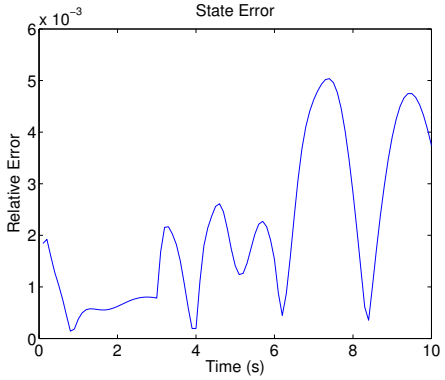
While the results of this example are excellent, this method is approximating a linearized system instead of the full nonlinear system. As such, the nonlinearity is ignored, and the approximations do not account for the nuances developed by the nonlinear terms of the original system. To capture this nonlinear behavior using input-independent techniques, new methods must be developed to handle the nonlinearity.



(a) State space generated using the full-order model $\mathbf{x}_1$.



(b) State space approximated by the reduced-order model $\widetilde{\mathbf{x}}_1 = \mathbf{V}\mathbf{x}_{1_r}$.



(c) Relative error over the first 10 seconds.

Figure 3.5: Comparison of the state spaces generated via the full and reduced-order models for the natural circulation problem.

Chapter 4

Towards Input-Independent Methods for Nonlinear Model Reduction

Accurately modeling fires inherently requires the use of nonlinear models. While the results for reducing linear models are excellent, we must move in the direction of nonlinearity to truly represent the behavior of a fire. In moving from linear to nonlinear systems, we are able to investigate a larger class of problems. However, this shift often results, except for very special nonlinearities, in the inability to employ $\mathcal{H}_2$-optimal input-independent methods to reduce the system order.

As stated earlier, large nonlinear dynamical systems typically require methods such as POD to create the reduced-order models. However, one of the limitations of these methods is that the reduced-order models are only as good as the full-order samples (i.e. snapshot matrices) that were used to generate the POD basis used to create the reduced-order model. Further, even if the ROM performs well for nearby solutions, it may not provide accurate solutions when input conditions are chosen sufficiently far away from the original full-order data snapshot samples. This behavior is in contrast to the $\mathcal{H}_2$-optimal methods used for

linear and bilinear systems where the reduced-order model is designed to perform well over a wide range of bounded input conditions. One of the long term goals of this research is to link these models to low-resolution, large-scale models, such as network models, to provide local high-fidelity predictions for the fire behavior. Therefore it is important to provide accurate input to output mappings to and from the nodal connection points.

Another limitation of POD, is the large off-line cost associated with capturing the data snapshots. We must capture a sufficiently rich set of data snapshots to build a set of basis vectors that are able to represent the dynamics of the system well. Additionally, this set must be targeted enough to allow for a relatively small number of basis vectors to represent the system [18, 80]. Recalling Theorem 2.12, we note that if the singular values of the snapshot matrix do not decay quickly, then it is difficult to reduce the system to a desired accuracy without utilizing a large number of basis vectors.

In this chapter, we begin to look at truly nonlinear dynamical systems, and we examine new methods to handle those nonlinear dynamical systems. As a test system, we chose the Burgers' equation (4.1). In our context, the Burgers' equation can be seen as a simplified version of the momentum equation in (2.1), where there is no external body force and the pressure remains constant over the spatial domain. This is a simple nonlinear system that embodies some of the challenges that we face in reducing the full fire model. Ignoring external forces, we see that the components, namely a convection term $v(x,t) \cdot v_x(x,t)$ and a diffusion term $\nu v_{xx}(x,t)$, are present in equations (2.1), (2.3)-(2.6) of the fire model. In this way we can test certain methods on a simpler system first to see if they are viable for the full fire model . After discussing the model generated from the Burgers' equation, we will examine three possible techniques for extending input independent methods to nonlinear equations. The first is a method to reduce the discretized quadratic-bilinear system using multi-moment matching (QBMOR) [15, 47, 48]. Next we investigate combining POD with an

input-independent method in an attempt improve the accuracy and stability of the reduced-order model over a larger range input functions. Finally, we discuss a technique where we ignore the nonlinear terms and attempt to determine the projection matrices using only the linear portion of the system.

4.1 Burgers' Equation

For our research, we solved the 1D Burgers' equation (4.1) using several different input functions (see Appendix B). A POD reduced-order model was creating using data snapshots from the control function $u_1(t)$ shown in figure B.1a. This reduced-order model was used as the standard with which we compared the reduced-order models generated using other techniques.

4.1.1 Problem Definition

Specifically, we looked at the 1D Burgers' equation over $[0,1] \times [0, t_f]$. For these tests, we set Neumann boundary conditions on the right side of the domain and a Dirichlet boundary control on the left. This leads to the boundary conditions

$$v_t(x,t) + v(x,t) \cdot v_x(x,t) = \nu v_{xx}(x,t), \tag{4.1}$$

$$v(0,t) = u(t), \tag{4.2}$$

$$v_x(1,t) = 0, \tag{4.3}$$

$$v(x,0) = v_0(x) = 0. \tag{4.4}$$

Examining the convection term, we see that the nonlinearity is actually the derivative of the quadratic state variable leading to the equivalent form

$$v_t(x,t) + \frac{1}{2}\frac{\partial}{\partial x}\left[(v(x,t) \otimes v(x,t)\right] = \nu \cdot v_{xx}(x,t). \tag{4.5}$$

This system can be spatially discretized by a few different methods. We discuss the group finite element method (GFEM) and finite difference method below.

4.1.2 Group Finite Element Method (GFEM)

We refer the reader to Krämer [62] for a complete treatment of the GFEM for solving the 1D Burgers' equation. We will mention a couple of points that have a computational impact when generating the reduced-order models. Using the weak formulation on (4.1), integrating by parts, and then projecting to the finite element basis, we are left with the matrix system

$$\mathbf{E}\dot{\alpha}(t) = \mathbf{A}\alpha(t) + \mathbf{B}u(t) + \mathcal{J}(\alpha(t)), \tag{4.6}$$

where given the finite element basis $\varphi_k(x)$, $k = 1, \ldots, N$, we can approximate the state space by (4.7)

$$v(x,t) \approx \sum_{k=1}^{N} \alpha_k(t)\varphi_k(x). \tag{4.7}$$

Further, the nonlinearity in the system is defined by (4.8) for each $j = 1, \ldots, N$.

$$\mathcal{J}(\alpha(t)) = -\sum_{i=1}^{N}\sum_{k=1}^{N}\int_0^1 \alpha_i(t)\alpha_k(t)\varphi_i(x)\varphi_k'(x)\varphi_j(x)\,dx. \tag{4.8}$$

This nonlinearity depends on t and must be recalculated at each time step. This is a major bottleneck for the computation.

The GFEM uses the conservative form of the Burgers equation (4.5) when generating the weak form to eliminate this bottleneck. Looking at the nonlinear term

$$v^2(x,t) \approx \left(\sum_{k=1}^{N} \alpha_k(t)\varphi_k(x) \right)^2, \tag{4.9}$$

$$\approx \sum_{k=1}^{N} \alpha_k^2(t)\varphi_k(x). \tag{4.10}$$

The last step in (4.10) is justified by the fact that at the nodes x_k, $k = 1, \ldots, N$, $\varphi_i(x_k)\varphi_j(x_k) = \delta_{ij}$. We then linearly interpolate between the nodes. Now our nonlinear system is a quadratic nonlinear system of the form

$$\mathbf{E}\dot{\alpha}(t) = \mathbf{A}\alpha(t) + \mathbf{H}(\alpha(t) \odot \alpha(t)) + \mathbf{B}u(t), \tag{4.11}$$

where $\odot$ represents the component-wise multiplication below

$$\alpha \odot \beta = [\ \alpha_1\beta_1,\ \cdots,;\ \alpha_n\beta_n\ ]^T.$$

Further, $\mathbf{H} \in \mathbb{R}^{N\times N}$ is defined by

$$(\mathbf{H})_{i,j} = -\frac{1}{2}\int_0^1 \varphi_i'(x)\varphi_j(x)\,dx,$$

which is constant and does not need to be computed at each time step. This system can now be reduced using the QBMOR method where $\mathbf{N} = \mathbf{0}$.

4.1.3 Finite Difference Method

In this method for discretizing the Burgers' equation, the spatial derivatives are approximated using backward finite difference scheme. For the convective term we approximate the first derivative using a backward difference

$$\frac{\partial v(x,t)}{\partial x}\Big|_{x=x_i} \approx \frac{v(x_i,t) - v(x_{i-1},t)}{h}, \tag{4.12}$$

where $h = x_i - x_{i-1}$. For the diffusion term, the second derivative is approximated using

$$\frac{\partial^2 v(x,t)}{\partial x^2}\Big|_{x=x_i} \approx \frac{v(x_{i+1},t) - 2v(x_i,t) + v(x_{i-1},t)}{h^2}. \tag{4.13}$$

The domain is then uniformly discretized as $a = x_0, x_1, \cdots, x_N, x_{N+1} = b$ and the boundary conditions are applied. This results in an N-dimensional system of ordinary differential equations to solve for $\mathbf{v}(t) = [v(x_1,t), \cdots, v(x_N,t)]^T$ in the discretized state space x^N

$$\begin{aligned} \dot{\mathbf{v}}(x^N,t) &= \mathbf{A}\mathbf{v}(x^N,t) + \mathbf{H}\left(\mathbf{v}(x^N,t) \otimes \mathbf{v}(x^N,t)\right) + \mathbf{N}\mathbf{v}(x^N,t)u(t) + \mathbf{B}u(t), \\ y(t) &= \mathbf{C}^T\mathbf{v}(x^N,t). \end{aligned} \tag{4.14}$$

where $y(t)$ is some output of interest. For this study, the output was simply considered as the value at $x = 1$.

It is worth noting that due to sparsity and structure of $\mathbf{H} \in \mathbb{R}^{n\times n^2}$, the nonlinear term in the full model can be simplified by

$$\mathbf{H}(\mathbf{v} \otimes \mathbf{v}) = \left((\hat{\mathbf{H}}\mathbf{v}) \odot \mathbf{v}\right),$$

where $\hat{\mathbf{H}} \in \mathbb{R}^{n\times n}$ is a diagonal matrix. Since the corresponding matrix in the reduced-order

system $\mathbf{H}_r \in \mathbb{R}^{r \times r^2}$ is dense, we cannot make this same simplification. However, since $r \ll n$, this is not a significant computational concern.

4.1.4 Projecting the Quadratic-Bilinear System

Once we have constructed a discrete system for the Burgers' equation using either the method described in section 4.1.2 or section 4.1.3, we are left with the quadratic bilinear dynamical system $\boldsymbol{\eta}(\mathbf{A}, \mathbf{H}, \mathbf{N}, \mathbf{b}, \mathbf{c})$ as defined by (2.16). In general, we will construct our system via projection similar to what was described for linear systems in section 2.3.1. However, we now must also project the quadratic and bilinear terms. Again, we start by approximating the state variable by $\mathbf{v} \approx \mathbf{V}\mathbf{v}_r$. Performing a Petrov-Galerkin projection on the state equation in (2.16) leads to equation

$$\mathbf{W}\left[\mathbf{V}\dot{\mathbf{v}}_r(t) - \mathbf{A}\mathbf{V}\mathbf{v}_r(t) - \mathbf{H}(\mathbf{V}\mathbf{v}_r(t) \otimes \mathbf{V}\mathbf{v}_r(t)) - \mathbf{N}\mathbf{V}\mathbf{v}_r(t)u(t) - \mathbf{b}u(t)\right] = 0. \tag{4.15}$$

We now provide a definition for the projection-based reduced-order model of the quadratic bilinear system.

Definition 4.1. Given the quadratic bilinear dynamical system $\boldsymbol{\eta}(\mathbf{A}, \mathbf{H}, \mathbf{N}, \mathbf{b}, \mathbf{c})$ given by (2.16), the projection-based reduced-order model is given by

$$\begin{aligned} \dot{\mathbf{v}}_r(t) &= \mathbf{A}_r\mathbf{v}_r(t) + \mathbf{H}_r(\mathbf{v}_r(t) \otimes \mathbf{v}_r(t)) + \mathbf{N}_r\mathbf{v}_r(t)u(t) + \mathbf{b}_r u(t), \\ y_r &= \mathbf{c}_r\mathbf{v}_r(t), \end{aligned} \tag{4.16}$$

where we define

$$\mathbf{A}_r = \mathbf{W}^T\mathbf{A}\mathbf{V} \qquad \mathbf{H}_r = \mathbf{W}^T\mathbf{H}\left(\mathbf{V}\otimes\mathbf{V}\right) \qquad \mathbf{N}_r = \mathbf{W}^T\mathbf{N}\mathbf{V} \tag{4.17}$$
$$\mathbf{b}_r = \mathbf{W}^T\mathbf{b} \qquad \mathbf{c}_r = \mathbf{c}\mathbf{V} \qquad \mathbf{W}^T\mathbf{V} = \mathbf{I}_r.$$

When $\mathbf{W} \neq \mathbf{V}$, we consider this a *two-sided* or *Petrov-Galerkin* projection. Further, if we let $\mathbf{W} = \mathbf{V}$, then this called a *one-sided* or *Galerkin* projection.

For the remainder of this chapter, we will use this projection to determine the reduced-order model. The only difference in each of the methods is how $\mathbf{V}$ and $\mathbf{W}$ are determined.

4.1.5 Tensor Product Computational Strategies

Before discussing model reduction strategies for the Burgers' equation, we must discuss the computational challenge of reducing the quadratic matrix $\mathbf{H}$ in this quadratic-bilinear system. To understand this issue, we look at the sizes of each of the matrices and the size of the resulting matrix products. As seen in (4.17), $\mathbf{H} \in \mathbb{R}^{n\times n^2}$, $\mathbf{W}, \mathbf{V} \in \mathbb{R}^{n\times r}$ are required to compute $\mathbf{H}_r \in \mathbb{R}^{r\times r^2}$. Since $\mathbf{H}$ is sparse and contains $\mathcal{O}(n)$ nonzero columns, it only requires about $\mathcal{O}(n^2)$ to store the matrix. Further, $\mathbf{W}^T\mathbf{H}$ will result in a sparse matrix $\hat{\mathbf{H}} \in \mathbb{R}^{r\times n^2}$ that requires $\mathcal{O}(r\times n)$ entries to store. The issue comes in the $(\mathbf{V}\otimes\mathbf{V})$ term which results in a dense $\mathbb{R}^{n^2\times r^2}$ matrix. Even for moderately sized systems, this matrix may exceed storage capacities of many desktop computers. For example, if we reduced a system from $n = 1500$ to $r = 20$ on a double precision machine, it would require almost 2 gigabytes to store $(\mathbf{V}\otimes\mathbf{V})$, not to mention that we still have to pre-multiply this matrix by $\hat{\mathbf{H}}$. Due to the density of this matrix we are not able to leverage sparse matrix multiplications. Fortunately, Benner and Breiten [15] suggest a method that leverages several well known tensor structure and multiplication properties, which can be found in [61].

Suppose we have a tensor $\mathcal{H} \in \mathbb{R}^{m\times n\times p}$, then we can create a matricization, $\mathbf{H}^{(1)}$ or simply $\mathbf{H}$, of the tensor $\mathcal{H}$ such that $\mathbf{H}^{(1)} \in \mathbb{R}^{m\times np}$. We call this the mode-1 unfolding of the tensor $\mathcal{H}$. We can also form the mode-2 unfolding $\mathbf{H}^{(2)} \in \mathbb{R}^{n\times mp}$ and the mode-3 unfolding $\mathbf{H}^{(3)} \in \mathbb{R}^{p\times mn}$ [61]. Using the properties of tensors and their matricizations, we now present Theorem 4.2 that will be critical in our being able to reduce these quadratic systems via projection.

Theorem 4.2. *The Hessian* $\mathbf{H}$ *in* (2.16) *can be symmetrized as shown in [15] without changing the dynamics of the operator. Further, if it is symmetrized, then the following properties hold*

1. $\mathbf{H}^{(2)} = \mathbf{H}^{(3)}$
2. $\mathbf{H}(u \otimes v) = \mathbf{H}(v \otimes w)$
3. $w^T\mathbf{H}^{(1)}(u \otimes v) = u^T\mathbf{H}^{(2)}(v \otimes w)$

Proof. See Benner and Breiten [15] for details. □

Using the properties from Theorem 4.2, we are able to significantly reduce the computation times and memory requirements for calculating $\mathbf{H}_r = \mathbf{W}^T\mathbf{H}(\mathbf{V} \otimes \mathbf{W})$. In particular, using property 3 from Theorem 4.2, we get Algorithm 4.3 which was suggested by [15].

Algorithm 4.3 (Projecting the Quadratic Matrix [15]). Given a symmetrized matrix $\mathbf{H}$, we calculate the reduced-order projection $\mathbf{H}_r = \mathbf{W}^T\mathbf{H}(\mathbf{V} \otimes \mathbf{V})$ as follows:

1. Compute tensor $\mathcal{Y} \in \mathbb{R}^{r\times n\times n}$ via $\mathbf{Y}^{(1)} = \mathbf{W}^T\mathbf{H}$.
2. Compute tensor $\mathcal{Z} \in \mathbb{R}^{r\times r\times n}$ via $\mathbf{Z}^{(2)} = \mathbf{V}^T\mathbf{Y}^{(2)}$.
3. Compute tensor $\widetilde{\mathcal{H}} \in \mathbb{R}^{r\times r\times r}$ via $\widetilde{\mathbf{H}}^{(3)} = \mathbf{V}^T\mathbf{Z}^{(3)}$.
4. Form the mode-1 unfolding $\mathbf{H}_r$ of the tensor $\widetilde{\mathcal{H}}$.

Using Algorithm 4.3, we can see that $(\mathbf{V} \otimes \mathbf{V})$ is never actually calculated. Further the computational costs of the matrix multiplications are significantly reduced. Finally, since the formation of the mode-2 and mode-3 unfoldings is a trivial re-indexing of the mode-1 unfolding, shifting between modes is very efficient.

4.2 Quadratic-Bilinear Model Reduction

In [47] and [48], a method (QLMOR) is developed to reduce quadratic models of the form given in (2.16) using moment matching and projections. In this method, the reduced-order model will match up to the q-th order moments of the first two transfer functions $H_1(s_1)$ and $H_2(s_1, s_2)$ at two given frequencies. The method only provides a technique for performing a one-sided projection. Benner and Breiten [15] use the ideas from Gu [48], but extend the approach to provide for moment matching at multiple frequencies and allow for two-sided projections. Benner and Goyal [16] have very recently extended the idea of Gramians and balanced truncation to model reduction of quadratic-bilinear dynamical systems.

In the following sections, we discuss the basics of two-sided quadratic-bilinear model order reduction (QBMOR) and then present our results using the techniques described. Finally, we discuss the stability issues associated with this method when using two-sided projections.

4.2.1 QBMOR Framework

For linear (2.12) and bilinear (2.15) dynamical systems, much is known about how to efficiently determine $\mathcal{H}_2$-optimal reduced-order models [14, 22, 39, 53, 54]. However, for quadratic systems, the criteria for guaranteeing this $\mathcal{H}_2$ optimality are not yet well understood. In a sense, the research for quadratic systems is in an early stage akin to when

the linear techniques were first being developed in [5, 46, 50, 51].

While the work by Gu [47, 48] set the stage for using moment matching techniques, it was focused on one-sided projections. We focus on the two-sided techniques presented by Benner and Breiten [15]. We start by choosing an appropriate sequence of nested Krylov subspaces that extends the known results for one-sided projections specified in [48] to implement a two-sided version of QLMOR defined by Benner and Breiten as defined by Theorem 4.6, which we refer to here as QBMOR. First, a couple terms are defined to clarify later notation.

Definition 4.4. Given a matrix $A \in \mathbb{R}^{n\times n}$ and starting vector $\mathbf{b} \in \mathbb{R}^n$, the associated Krylov space is defined as

$$\mathcal{K}(\mathbf{A}, \mathbf{b}) := \text{span}\{\mathbf{b},\ \mathbf{Ab},\ \mathbf{A}^2\mathbf{b},\ \mathbf{A}^3\mathbf{b},\ \ldots\}. \tag{4.18}$$

Further the nth order Krylov subspace is defined as

$$\mathcal{K}_n(\mathbf{A}, \mathbf{b}) := \text{span}\{\mathbf{b},\ \mathbf{Ab},\ \ldots,\ \mathbf{A}^n\mathbf{b}\}. \tag{4.19}$$

Definition 4.5. Let $\mathbf{E}, \mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{b} \in \mathbb{R}^n$, $q \in \mathbb{N}$, and $\sigma \in \mathbb{C}$. We define the associated rational Krylov subspace by

$$\mathcal{K}_q(\mathbf{E}, \mathbf{A}, \mathbf{b}, \sigma) := \mathcal{K}_q\left((\sigma\mathbf{E} - \mathbf{A})^{-1}\mathbf{E}, (\sigma\mathbf{E} - \mathbf{A})^{-1}\mathbf{b}\right). \tag{4.20}$$

Theorem 4.6 (Benner and Breiten, 2012)**.** *Let $\Sigma = (\mathbf{E}, \mathbf{A}, \mathbf{H}, \mathbf{N}, \mathbf{b}, \mathbf{c})$ denote a quadratic bilinear control system of dimension n. Let $q_1, q_2 \in \mathbb{N}$ with $q_2 \leq q_1$. Assume that a reduced QBDAE is constructed by a Petrov-Galerkin type projection*

$$\mathbf{E}_r = \mathcal{W}^T\mathbf{E}\mathcal{V}, \qquad \mathbf{A}_r = \mathcal{W}^T\mathbf{A}\mathcal{V}, \qquad \mathbf{H}_r = \mathcal{W}^T\mathbf{H}(\mathcal{V}\otimes\mathcal{V})$$
$$\mathbf{N}_r = \mathcal{W}^T\mathbf{N}\mathcal{V}, \qquad \mathbf{b}_r = \mathcal{W}^T\mathbf{b}, \qquad \mathbf{c}_r = \mathcal{V}^T\mathbf{c}$$

where $\mathcal{V}$ and $\mathcal{W}$ are the orthonormal bases for the span of the union of the following column spaces

$$\mathbf{V}_1 = \mathcal{K}_{q_1}(\mathbf{E}, \mathbf{A}, \mathbf{b}, \sigma)$$

$$\mathbf{W}_1 = \mathcal{K}_{q_1}(\mathbf{E}^T, \mathbf{A}^T, \mathbf{c}, 2\sigma)$$

for $i = 1 : q_2$

$$\mathbf{V}_2^i = \mathcal{K}_{q_2-i+1}(\mathbf{E}, \mathbf{A}, \mathbf{N}\mathbf{V}_1(:, i), 2\sigma)$$

$$\mathbf{W}_2^i = \mathcal{K}_{q_2-i+1}(\mathbf{E}^T, \mathbf{A}^T, \mathbf{N}^T\mathbf{W}_1(:, i), \sigma)$$

for $j = 1 : \min(q_2 - i + 1, i)$

$$\mathbf{V}_3^{i,j} = \mathcal{K}_{q_2-i-j+2}(\mathbf{E}, \mathbf{A}, \mathbf{H}\mathbf{V}_1(:, i) \otimes \mathbf{V}_1(:, j), 2\sigma)$$

$$\mathbf{W}_3^{i,j} = \mathcal{K}_{q_2-i-j+2}(\mathbf{E}^T, \mathbf{A}^T, \mathcal{H}^{(2)}\mathbf{V}_1(:, i) \otimes \mathbf{W}_1(:, j), \sigma),$$

i.e.

$$\operatorname{span}(\mathcal{V}) = \operatorname{span}(\mathbf{V}_1) \cup \bigcup_i \operatorname{span}(\mathbf{V}_2^i) \cup \bigcup_{i,j} \operatorname{span}(\mathbf{V}_3^{i,j}),$$

$$\operatorname{span}(\mathcal{W}) = \operatorname{span}(\mathbf{W}_1) \cup \bigcup_i \operatorname{span}(\mathbf{W}_2^i) \cup \bigcup_{i,j} \operatorname{span}(\mathbf{W}_3^{i,j}).$$

then the following conditions hold

$$\frac{\partial^i G_1}{\partial s_1^i}(\sigma) = \frac{\partial^i \tilde{G}_1}{\partial s_1^i}(\sigma) \qquad i = 0, \ldots, q_1 - 1,$$

$$\frac{\partial^i G_1}{\partial s_1^i}(2\sigma) = \frac{\partial^i \tilde{G}_1}{\partial s_1^i}(2\sigma) \qquad i = 0, \ldots, q_1 - 1,$$

$$\frac{\partial^{i+j} G_2}{\partial s_1^i s_2^j}(\sigma) = \frac{\partial^{i+j} \tilde{G}_2}{\partial s_1^i s_2^j}(\sigma) \qquad i + j \leq 2q_2 - 1.$$

4.2.2 Results

Our initial approach was to look at the Burgers' equation as set up in Section 4.1. These equations were first solved using the GFEM method described in 4.1.2 and Krämer [62]. As was suggested in [15], IRKA was run on the linear portion of the system in order to determine the shift points to use with QBMOR. Using the projections in 4.6, $\mathcal{V}$ and $\mathcal{W}$ were calculated and used to create reduced-order models via Galerkin (one-sided) and Petrov-Galerkin (two-sided) projections. We observed that the two-sided systems were often not stable.

To test the stability of the one-sided versus two-sided projections, tests were run when using the various values for q_1 and q_2. As seen in the data the two-sided projection was unstable for several different configurations of q_1 and q_2, while the one-sided projections produced a bounded output in every case. This is not surprising. Unless the Petrov-Galerkin bases are chosen by effective/optimal methods such as Balanced Truncation and IRKA, even for linear systems, one might expect one-sided projections to preserve stability more often the two-sided ones. Examples of the cases where the two-sided projections converge are shown in Figures 4.1 and 4.2. As can be seen in the Figures and in Table 4.1, when the two-sided projection did converge, it was a much better approximation to the full-order solution.

We investigated to see if there was some correlation between the eigenvalues of $\mathbf{E}^{-1}\mathbf{A}$ and the stability (see Table 4.1) of the reduced-order model. While there does not seem to be a direct correlation, it is worth noting that one-sided projections never produced unstable eigenvalues, but as more moments are matched for the two-sided projections, we start to see unstable eigenvalues appear in the linear portion of the reduced-order model. Further, while the condition number of the projected mass matrix of the one-sided projection is always nearly 1, the condition number for the mass matrix seems to grow when the projection

Table 4.1: Summary data for $\mathbf{E}_{r1} = \mathcal{V}^T\mathbf{E}\mathcal{V}$, $\mathbf{E}_{r2} = \mathcal{W}^T\mathbf{E}\mathcal{V}$, $\mathbf{A}_{r1} = \mathcal{V}^T\mathbf{A}\mathcal{V}$, and $\mathbf{A}_{r2} = \mathcal{W}^T\mathbf{A}\mathcal{V}$.

			Condition Number		# Unstable Eigenvalues		Relative Error	
q_1	q_2	r	$\mathbf{E}_{r1}$	$\mathbf{E}_{r2}$	$\lambda_1(\mathbf{E}_{r1}^{-1}\mathbf{A}_{r1})$	$\lambda_2(\mathbf{E}_{r2}^{-1}\mathbf{A}_{r2})$	$\|y - y_{r1}\|/\|y\|$	$\|y - y_{r2}\|/\|y\|$
3	0	3	1.002	6.114	0	0	0.34942	∞
3	1	4	1.003	2.718	0	0	0.21131	∞
3	2	6	1.006	4.621	0	0	0.12979	0.40050
3	3	10	1.014	7.429	0	0	0.04798	0.00806
4	0	4	1.003	19.427	0	0	0.17225	∞
4	1	5	1.004	6.521	0	0	0.17514	∞
4	2	7	1.007	4.980	0	0	0.09402	0.08103
4	3	11	1.017	8.129	0	0	0.04118	0.00444
4	4	17	1.494	14.026	0	1	0.02506	∞
5	0	5	1.005	69.437	0	0	0.15804	∞
5	1	6	1.006	14.425	0	0	0.11997	∞
5	2	8	1.010	12.986	0	0	0.05434	0.03009
5	3	12	1.019	8.617	0	0	0.03058	0.00144
5	4	18	1.570	40.088	0	3	0.03338	∞
5	5	27	1.986	171.625	0	6	0.01393	∞
6	0	6	1.007	264.469	0	0	0.09385	∞
6	1	7	1.008	36.903	0	0	0.08121	∞
6	2	9	1.011	8.532	0	0	0.05957	0.01333
6	3	13	1.020	13.560	0	1	0.03450	∞
6	4	19	1.784	13.368	0	2	0.02021	∞
6	5	28	1.985	53.223	0	4	0.01274	∞
6	6	40	2.184	264.982	0	5	0.01017	∞

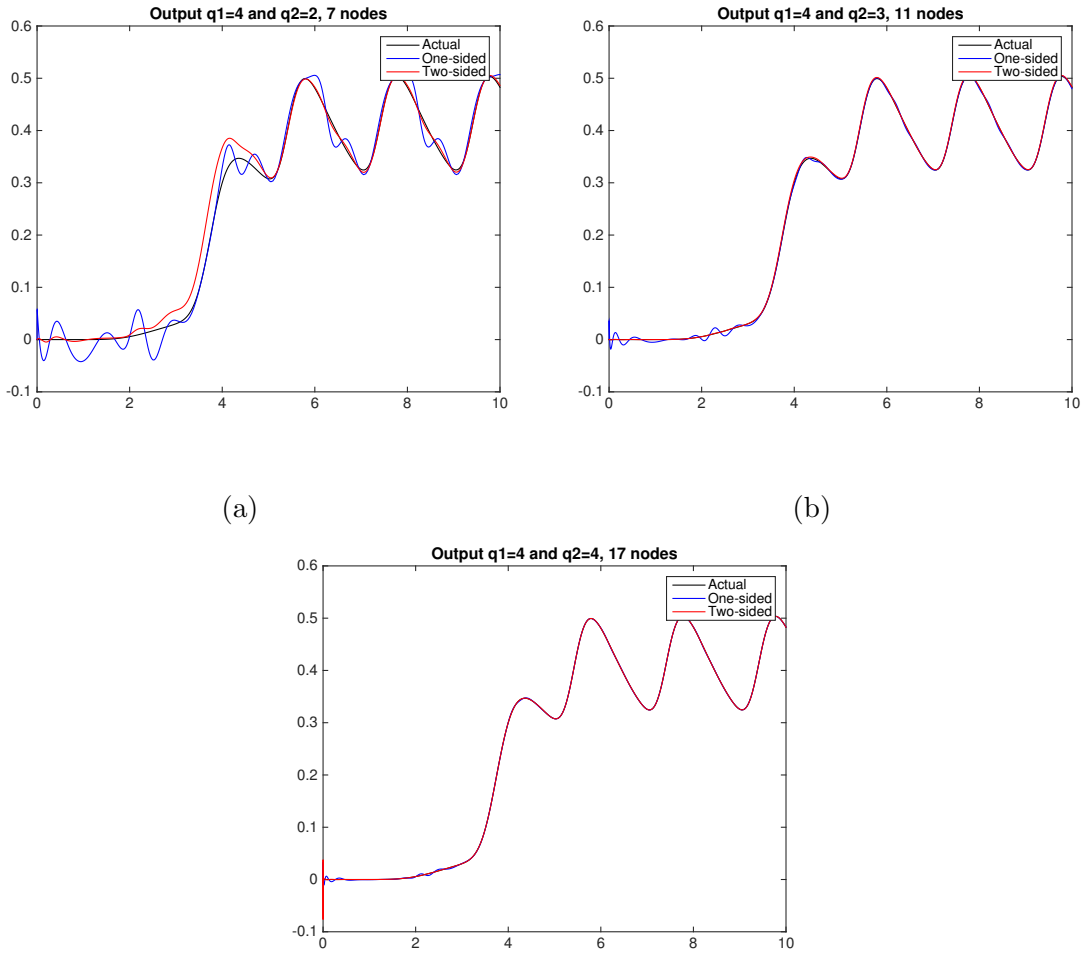
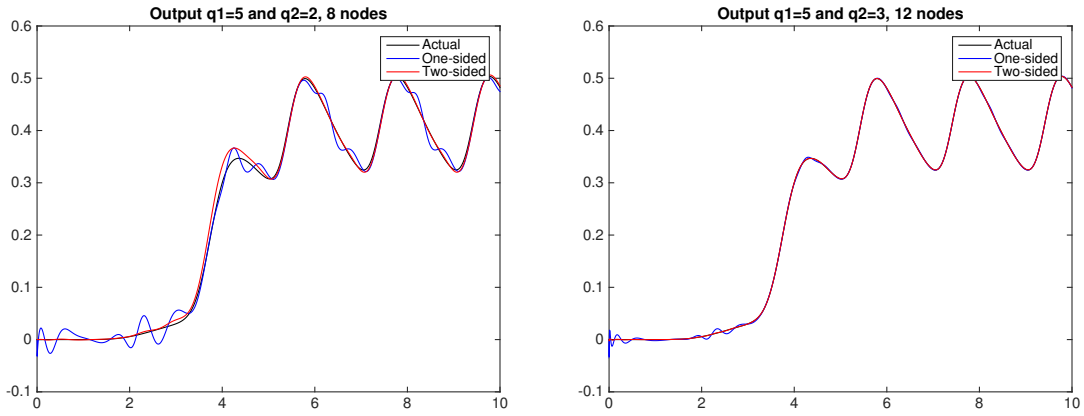


Figure 4.1: Output for reduced models versus the full state output where $q_1 = 4$.

matrices are dominated either by the linear or the quadratic term.

Attempt to Control Stability

Benner and Breiten [15] generated the interpolation points for QBMOR using IRKA applied to the linear portion of the system. In an attempt to improve the stability results, we extended this idea of using QBMOR and IRKA together by combining the QBMOR basis with



(a) (b)

Figure 4.2: Output for reduced models versus the full state output where $q_1 = 5$.

four basis vectors generated by using IRKA. The difference between these two frameworks is that the first is only seeking the interpolation points to use for QBMOR, whereas the second merges the bases created using two techniques into a single basis. Our goal was to use the IRKA vectors to increase the order of the reduced system while still maintaining stability. These IRKA vectors generated using just the linear portion of the system are known as $\mathcal{H}_2$-optimal vectors since they would be optimal for the linear system. However, they are not optimal for the full nonlinear system [29]. For QBMOR-IRKA, the IRKA projection matrices were appended to the QBMOR projection matrices and orthonormalized via QR. This resulted in a significant improvement in the accuracy over QBMOR alone, with the output error being almost an order of magnitude less when both converged. However, the stability of the system was not improved; and for almost every case if the QBMOR method diverged, then so would the IRKA-QBMOR method, as seen in Table 4.2 and Figure 4.3.

Table 4.2: Summary data for reduced-order models using QBMOR.

				Solution Error $\|y - y_r\|$			
q_1	q_2	r	r_{IQ}	1-sided	QB-IRKA 1S	2-sided	QB-IRKA 2S
2	0	2	6	5.16e+00	6.08e-01	7.10e+00	Inf
2	1	4	8	1.45e+00	3.05e-01	1.36e+00	3.70e-01
2	2	8	12	4.61e-01	8.13e-02	2.80e-01	6.63e-03
3	0	3	7	2.26e+00	4.24e-01	Inf	Inf
3	1	5	9	1.09e+00	1.91e-01	7.68e-01	2.04e-01
3	2	9	13	2.79e-01	7.78e-02	1.44e-01	Inf
3	3	16	20	7.29e-02	2.72e-02	Inf	2.52e-03
4	0	4	8	1.08e+00	2.68e-01	Inf	Inf
4	1	6	10	6.88e-01	1.90e-01	5.42e-01	Inf
4	2	10	14	1.87e-01	7.55e-02	5.15e-02	7.57e-03
4	3	17	21	3.96e-02	3.07e-02	Inf	Inf
4	4	27	31	3.01e-02	1.76e-02	Inf	Inf
5	0	5	9	1.09e+00	2.49e-01	Inf	Inf
5	1	7	11	4.30e-01	1.19e-01	6.33e-01	Inf
5	2	11	15	1.39e-01	4.71e-02	2.65e-02	4.27e-03
5	3	18	22	4.88e-02	1.65e-02	Inf	Inf
5	4	28	32	2.88e-02	1.60e-02	Inf	Inf
5	5	42	46	1.59e-02	8.09e-03	Inf	Inf

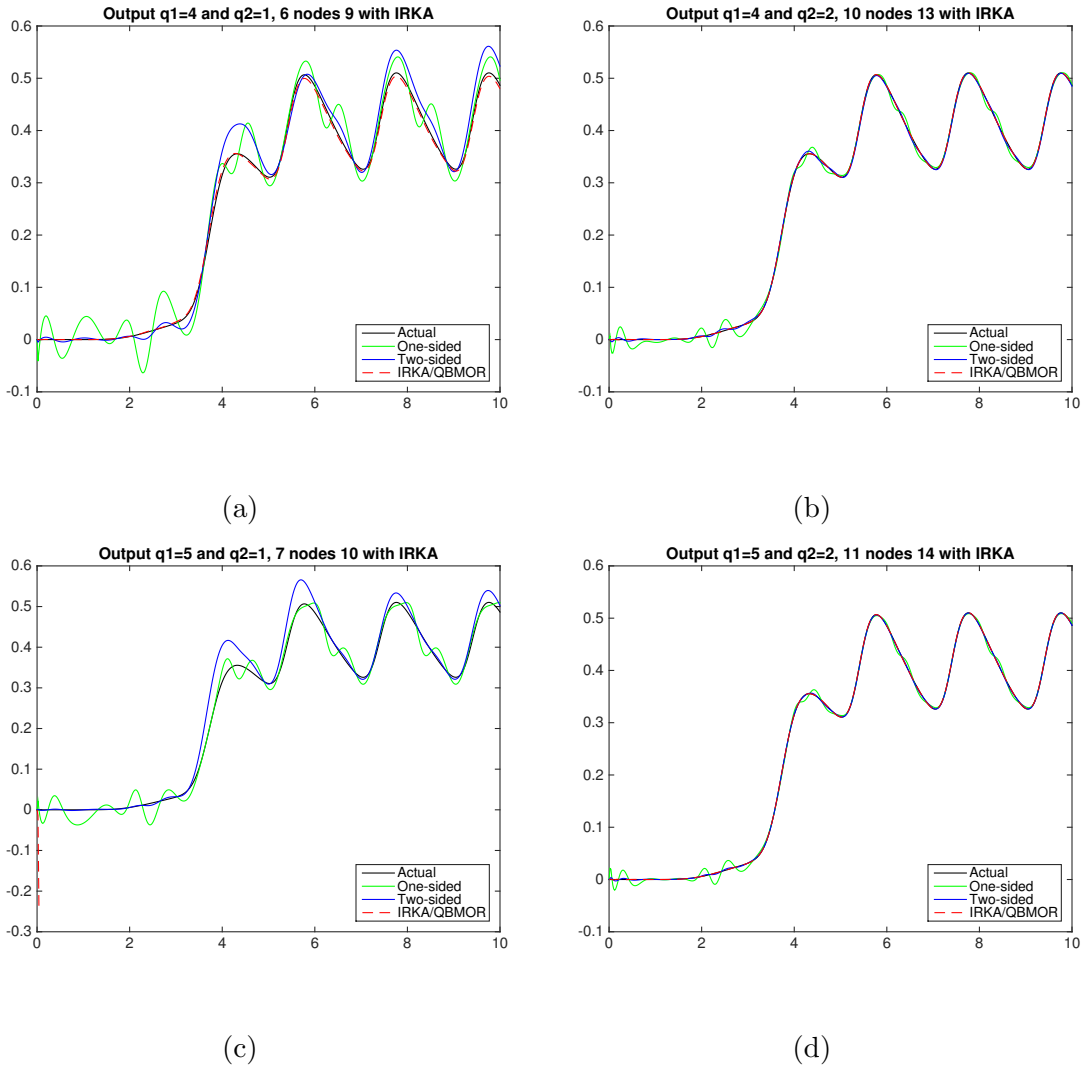


Figure 4.3: Output for reduced models versus the full state finite difference output.

4.2.3 Stability Preservation

In looking at the QBMOR method for reducing the quadratic-bilinear system, it was noted that the oblique Petrov-Galerkin projection often produced unstable reduced-order models. While these two-sided projections theoretically provide more accurate models, they also may result in asymptotic instabilities [15, 21]. As noted by [15], there may be a Lyapunov-based

method to help ensure stability. However, this method requires solving large-scale Lyapunov equations and may be impractical for very large systems. For our research, we did not investigate the use of this method, and we refer the reader to [15] for a complete discussion of the method, including benefits and limitations.

4.3 Combining IRKA and POD

As was shown by Chaturantabut et al. [29], POD and quasi-$\mathcal{H}_2$-optimal basis vectors have been successfully employed to improve the approximations for structure-preserving reduced-order models of port-Hamiltonian systems. The basis vectors are considered quasi-$\mathcal{H}_2$-optimal because they are generated using IRKA on the linear portion, ignoring the non-linear terms. Our research shows this could potentially be a new technique for leveraging the best characteristics of different model reduction techniques. At this point, however, there are no real guidelines on how these subspaces should be combined.

4.3.1 Methodology

Our first investigation into these methods was combining the linear $\mathcal{H}_2$-optimal basis vectors with two-sided QBMOR in an attempt to improve stability as described in Section 4.2.2. As said in that Section, we did not see an improvement in the stability, but we did notice an improvement in accuracy of the approximations when the linear $\mathcal{H}_2$-optimal IRKA basis was augmented to the QBMOR basis. This improvement along with the results from [29] motivated combining POD with the linear $\mathcal{H}_2$-optimal IRKA basis to approximate the quadratic bilinear system given by (2.16). Initially, we developed Algorithm 4.7 to build our projection vectors $\mathbf{V}$ and $\mathbf{W}$. With this method, we first pick a reduced-order size r. Then a number of

IRKA vectors, r_I, and POD vectors, r_P, are chosen such that $r = r_I + r_P$. To build $\mathbf{V}$ and $\mathbf{W}$, the IRKA and POD vectors are combined, and an orthogonal basis is generated that spans the range of the combined basis.

Algorithm 4.7 (POD+IRKA - Method 1). Let $\Sigma = (\mathbf{A}, \mathbf{H}, \mathbf{N}, \mathbf{b}, \mathbf{c})$ be a SISO quadratic bilinear control system (2.16) of size n. Let r be the desired size of the reduced-order model and r_I be the number of desired IRKA vectors where $1 \leq r_I \leq r$.

1. Generate k POD orthonormal basis vectors $\Phi = [\varphi_1, \cdots, \varphi_k]$ from full-order snapshots created using the baseline input function $u_1(t)$, where $r = r_I + k$.
2. Apply IRKA using just the linear terms, $\mathbf{A}, \mathbf{b}, \mathbf{c}$, to produce $\mathbf{V}_I, \mathbf{W}_I \in \mathbb{R}^{n \times r_I}$.
3. Create orthonormal bases $\mathcal{V}$ and $\mathcal{W}$ such that
$$\begin{aligned} \text{span}\{\mathbf{V}\} &= \text{span}[\mathbf{V}_I, \varphi_1, \ldots, \varphi_k], \\ \text{span}\{\mathbf{W}\} &= \text{span}[\mathbf{W}_I, \varphi_1, \ldots, \varphi_k]. \end{aligned}$$
4. Create reduced-order model $\zeta = (\mathbf{A}_r, \mathbf{H}_r, \mathbf{N}_r, \mathbf{b}_r, \mathbf{c}_r)$ of size r using a Petrov-Galerkin (two-sided) projection:
$$\begin{aligned} \mathbf{A}_r &= \mathbf{W}^T\mathbf{A}\mathbf{V}, & \mathbf{H}_r &= \mathbf{W}^T\mathbf{H}(\mathbf{V} \otimes \mathbf{V}), & & \\ \mathbf{N}_r &= \mathbf{W}^T\mathbf{N}\mathbf{V}, & \mathbf{b}_r &= \mathbf{W}^T\mathbf{b}, & \mathbf{c}_r &= \mathbf{V}^T\mathbf{c}. \end{aligned}$$

As we will see in the results section 4.3.2, if too many IRKA vectors are chosen, the system becomes unstable. Therefore, we attempted to control the stability of the reduced-order model by building a larger space consisting of r POD basis vectors and $1 \leq r_I \leq r$ IRKA basis vectors. We then create an orthogonal basis from these using SVD and truncate back to the desired reduced-order dimension, r. This idea was motivated by the proper orthogonal decomposition technique of computing the SVD and then truncating to the correct dimension. This technique is summarized in Algorithm 4.8. We do note that with this second method we attempted both Galerkin (one-sided) and Petrov-Galerkin (two-sided) projections to see if one method would perform better than the other.

Algorithm 4.8 (POD+IRKA - Method 2). Let $\Sigma = (\mathbf{A}, \mathbf{H}, \mathbf{N}, \mathbf{b}, \mathbf{c})$ be a SISO quadratic bilinear control system (2.16) of size n. Let r be the desired size of the reduced-order model and r_I be the number of desired IRKA vectors where $1 \leq r_I \leq r$.

1. Generate POD orthonormal basis vectors $\Phi = [\varphi_1, \cdots, \varphi_r]$ from from full-order snapshots created using the baseline input function $u_1(t)$.
2. Apply IRKA using just the linear terms, $\mathbf{A}, \mathbf{B}, \mathbf{C}$, to produce $\mathcal{V}_I$ and $\mathcal{W}_I$.
3. Create orthonormal bases $\mathcal{V}$ and $\mathcal{W}$ using SVD such that

$$\begin{aligned} \text{span}\{\mathbf{V}_f\} &= \text{span}[\mathcal{V}_I, \varphi_1, \ldots, \varphi_r], \\ \text{span}\{\mathbf{W}_f\} &= \text{span}[\mathcal{W}_I, \varphi_1, \ldots, \varphi_r]. \end{aligned}$$

   Since $\mathbf{V}_f, \mathbf{W}_f \in \mathbb{R}^{n \times (r+r_I)}$ are column ordered by SVD based on the associated singular vectors, we let $\mathbf{V}$ and $\mathbf{W}$ be the leading r vectors of $\mathbf{V}_f$ and $\mathbf{W}_f$, respectively.
4. Create reduced-order model $\zeta = (\mathbf{A}_r, \mathbf{H}_r, \mathbf{N}_r, \mathbf{b}_r, \mathbf{c}_r)$ of size r using a Petrov-Galerkin (two-sided) projection:

$$\begin{aligned} \mathbf{A}_r &= \mathbf{W}^T\mathbf{A}\mathbf{V}, & \mathbf{H}_r &= \mathbf{W}^T\mathbf{H}(\mathbf{V} \otimes \mathbf{V}), & & \\ \mathbf{N}_r &= \mathbf{W}^T\mathbf{N}\mathbf{V}, & \mathbf{b}_r &= \mathbf{W}^T\mathbf{b}, & \mathbf{c}_r &= \mathbf{V}^T\mathbf{c}. \end{aligned}$$

5. We let $\mathbf{W} = \mathbf{V}$ for the Galerkin (one-sided) projection.

4.3.2 Results

For our tests we ran full-order simulations of the Burgers' equation using multiple input functions, $u_1(t), \ldots, u_6(t)$ (see Figure B.1). We used the state-space solutions from the simulation with input $u_1(t)$ to create our snapshot matrix. From this matrix, we built an r-dimensional POD basis, Φ_r. After using this to project the full-order system, we have the

following reduced-order model

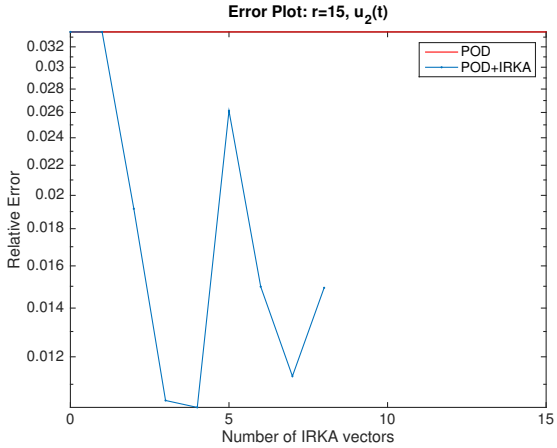
$$\dot{\mathbf{x}}_r(t) = \mathbf{A}_P\mathbf{x}_r(t) + \mathbf{H}_P(\mathbf{x}_r(t) \otimes \mathbf{x}_r(t)) + \mathbf{N}_P\mathbf{x}_r(t)u(t) + \mathbf{b}_P u(t), \tag{4.21}$$
$$y_r(t) = \mathbf{c}_P^T\mathbf{x}_r(t),$$

where $\mathbf{A}_P, \mathbf{H}_P, \mathbf{N}_P, \mathbf{b}_P$, and $\mathbf{c}_P$ are created via projection as in Definition 4.1 where $\mathbf{W} = \mathbf{V} = \Phi_r$. We then create a second reduced-order model by choosing $1 < k \leq r$ and building k-dimensional IRKA projecting vectors $\mathbf{V}_I$ and $\mathbf{W}_I$. Using Algorithm 4.7, we create a reduced-order model

$$\dot{\mathbf{x}}_r(t) = \mathbf{A}_k\mathbf{x}_r(t) + \mathbf{H}_k(\mathbf{x}_r(t) \otimes \mathbf{x}_r(t)) + \mathbf{N}_k\mathbf{x}_r(t)u(t) + \mathbf{b}_k u(t), \tag{4.22}$$
$$y_k(t) = \mathbf{c}_k^T\mathbf{x}_r(t).$$

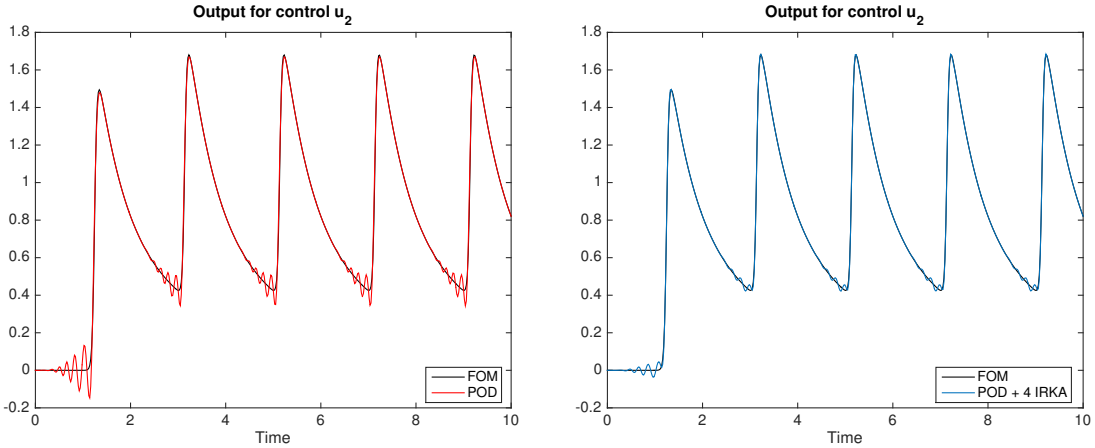
Table 4.3: Error for $r = 15$ using r_{I} number of IRKA vectors and $(r - r_{\mathrm{I}})$ number of POD vectors in a combined subspace.

r_{I}	$u_1(t)$	$u_2(t)$	$u_3(t)$	$u_4(t)$	$u_5(t)$	$u_6(t)$
POD	2.0395e-03	3.3547e-02	2.0889e-02	7.7171e-02	1.9559e-02	3.6554e-02
2	1.1813e-03	1.9170e-02	1.0549e-02	4.9792e-02	1.0588e-02	2.0199e-02
3	1.0157e-03	1.0451e-02	5.2608e-03	3.7456e-02	6.4086e-03	1.6919e-02
4	8.2954e-04	1.0224e-02	5.3244e-03	3.2765e-02	6.0958e-03	1.3249e-02
5	5.6006e-04	2.6137e-02	5.1035e-02	NaN	9.8819e-03	NaN
6	3.3443e-04	1.4983e-02	5.7398e-03	1.7622e-01	5.4606e-03	NaN
7	2.1545e-04	1.1286e-02	5.2093e-03	1.3122e-01	6.1630e-03	NaN
8	1.9037e-04	1.4928e-02	6.8101e-03	3.0147e-01	5.4235e-03	NaN
9	6.1983e-04	NaN	NaN	NaN	3.9650e-02	NaN
10	7.8468e-04	NaN	NaN	NaN	1.0426e-01	NaN
11	1.8322e-03	NaN	NaN	NaN	NaN	NaN



(a) Input $u_2(t)$

Figure 4.4: Output error combining POD and IRKA vectors with $r = 15$ and input $u_2(t)$, (Method 1).



(a) POD for input $u_2(t)$ (b) POD and IRKA for input $u_2(t)$

Figure 4.5: Output combining POD and IRKA vectors with $r = 15$, (Method 1).

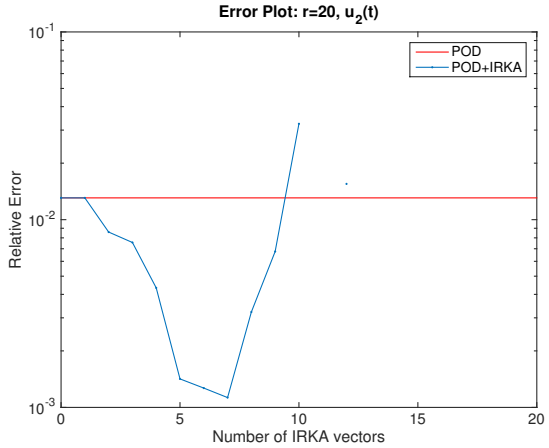
We then tested each reduced-order model using the various input functions. We calculated the relative error of each of the results by taking the norm of the difference between the full-order models and reduced-order models as:

$$err_{POD} = \frac{\|y(\hat{t}) - y_P(\hat{t})\|_2}{\|y(\hat{t})\|_2} \qquad err_{POD-IRKA} = \frac{\|y(\hat{t}) - y_k(\hat{t})\|_2}{\|y(\hat{t})\|_2}, \tag{4.23}$$

where $\hat{t}$ are the uniformly spaced solution times from the ODE solver, $\hat{t} = \{0 = t_0, t_1, \ldots, t_f = 10\}$. The error results when $r = 15$ are summarized in Table 4.3, where NaN indicates a solution that is asymptotically unstable. Figure 4.4 gives a graphical representation of these results for input function $u_4(t)$. It is difficult to see any patterns in the error results, but we can see an improvement over POD alone for certain combinations of POD and IRKA. Further, Figure 4.5 shows comparisons between the output for the POD model and the best combined POD/IRKA reduced-order model for the input function $u_2(t)$. Here we see that augmenting the POD basis with IRKA basis vectors improves the accuracy of the approximation, especially at the transition points where there are oscillations in the reduced-order model. See Appendix C for all the error and output plots associated with this test.

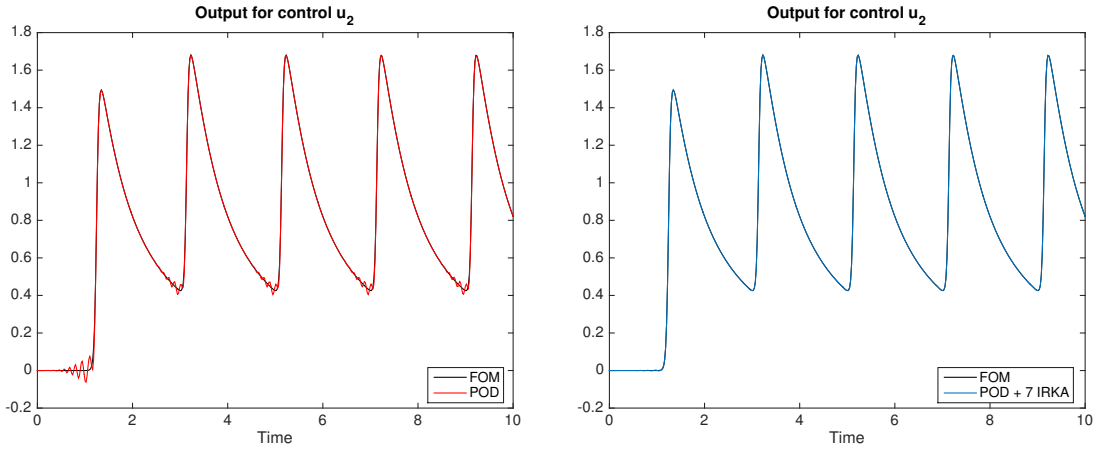
Table 4.4: Error for $r = 20$ using r_I number of IRKA vectors and $(r - r_\mathrm{I})$ number of POD vectors in a combined subspace.

r_I	$u_1(t)$	$u_2(t)$	$u_3(t)$	$u_4(t)$	$u_5(t)$	$u_6(t)$
POD	3.0994e-04	1.3063e-02	7.7132e-03	4.3534e-02	6.1436e-03	1.7336e-02
2	4.1667e-04	8.5916e-03	4.5444e-03	3.4318e-02	4.2229e-03	1.2376e-02
3	4.0444e-04	7.5586e-03	3.6971e-03	3.7349e-02	4.2320e-03	1.1147e-02
4	2.4889e-04	4.3315e-03	2.2410e-03	1.7807e-02	2.3858e-03	6.2902e-03
5	1.9050e-04	1.4183e-03	1.6518e-03	1.2586e-02	6.1097e-04	2.1014e-02
6	1.8958e-04	1.2675e-03	1.1171e-03	9.9517e-03	6.4955e-04	3.5521e-03
7	1.8925e-04	1.1281e-03	1.1015e-03	1.5459e-02	4.4534e-04	6.3730e-03
8	1.8915e-04	3.2229e-03	3.2438e-03	7.0775e-02	5.6806e-04	7.5582e-02
9	1.8954e-04	6.7668e-03	2.5300e-03	2.6656e-01	1.0726e-03	3.4157e-02
10	1.8885e-04	3.2414e-02	2.3428e-02	NaN	3.4075e-03	NaN
11	2.8407e-04	NaN	NaN	NaN	3.2713e-01	NaN
12	1.8995e-04	1.5509e-02	1.0009e-02	NaN	2.0734e-03	NaN
13	NaN	NaN	NaN	NaN	NaN	NaN
14	1.4427e-03	NaN	NaN	NaN	NaN	NaN



(a) Input $u_2(t)$

Figure 4.6: Output error combining POD and IRKA vectors for $r = 20$ and input $u_2(t)$ (Method 1).



(a) POD for input $u_2(t)$ (b) POD and IRKA for input $u_2(t)$

Figure 4.7: Output combining POD and IRKA vectors for input $u_2(t)$ (Method 1).

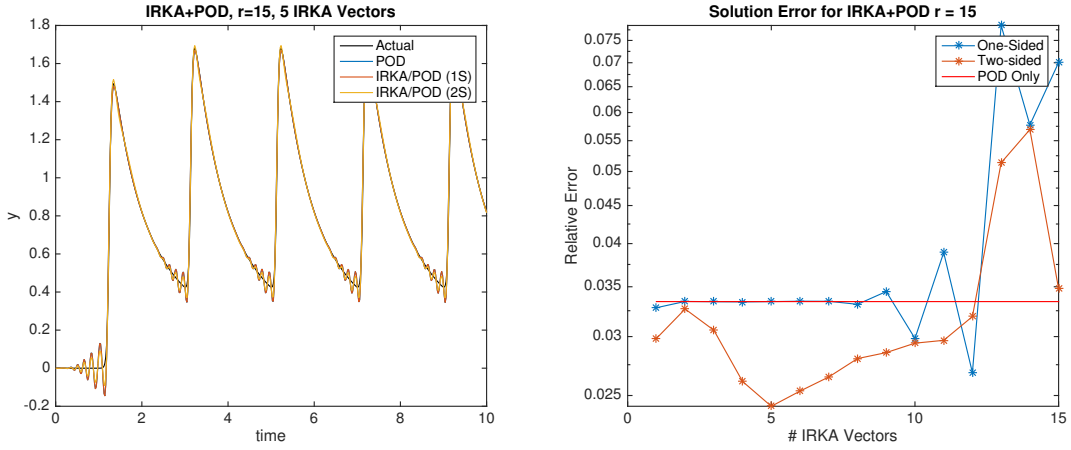
We then set the reduced-order model size to $r = 20$ and the results are shown in 4.8. Again, we show the graphical depiction of the error for the selected function $u_2(t)$ in Figure 4.6 and the output in 4.7. Unlike when we looked at the reduced-order models of size $r = 15$, we start to see a pattern emerge where there seems to be an optimal number of IRKA vectors that should be added for each input function. Unfortunately, this varies from input

function to input function. We also observe that while the there is a slight improvement in the amplitude of the oscillations when moving from fifteen to twenty POD vectors, there is a drastic reduction in the oscillations for the IRKA/POD models. Again the plots for all the output and error results can be seen in Appendix C.

Finally, we ran a full set of tests after creating reduced-order models using Algorithm 4.8. As can be seen in Table 4.5 and Figure 4.8, the two-sided projection remained stable, but the approximations were not as good as we saw with Method 1. The tests using the one-sided projections were essentially equivalent to using POD alone. While, the reduced-order models remained asymptotically stable, the errors became worse than POD for a large number of IRKA vectors relative to the reduced-order model size. That is to say that as $r_I \to r$, the relative error of the output for the combined method exceeds the relative error of just using POD alone.

Table 4.5: Errors for ROM ($r = 15$) combining POD and IRKA (Method 2)

	Relative Error: $\|y_r - y\|/\|y\|$	
# of IRKA bases	One-sided	Two-sided
1	3.28e-02	2.98e-02
2	3.34e-02	3.27e-02
3	3.34e-02	3.06e-02
4	3.34e-02	2.61e-02
5	3.34e-02	2.42e-02
6	3.35e-02	2.53e-02
7	3.35e-02	2.65e-02
8	3.31e-02	2.80e-02
9	3.45e-02	2.86e-02
10	2.98e-02	2.94e-02
11	3.89e-02	2.97e-02
12	2.68e-02	3.19e-02
13	7.86e-02	5.14e-02
14	5.77e-02	5.69e-02
15	7.00e-02	3.48e-02



(a) Output (b) Approximation errors

Figure 4.8: Combining POD and IRKA vectors with input function $u_2(t)$ (Method 2).

4.4 Combining Left and Right IRKA Vectors

After looking at combining POD and IRKA linear $\mathcal{H}_2$-optimal basis, we wanted to see if projecting the quadratic-bilinear system using the linear $\mathcal{H}_2$-optimal projection matrices alone could successfully reduce the model. Specifically, we were looking to see if we could obtain similar or better errors than the POD models while maintaining the system stability in the reduced-order model. The main advantage would be that since the snapshot matrix would not be needed, the off-line cost of building the reduced-order model would be significantly decreased. Additionally, we were hoping that we would see the benefits of using an input independent method, i.e. our reduced-order model would perform better over a larger range of input functions. Unfortunately, when utilizing a two-sided projection we lost asymptotic stability in the reduced-order models. When we used the right projection vectors in a one-sided projection, POD easily out-performed the linear $\mathcal{H}_2$-optimal method. We saw similar results when using B-IRKA. Therefore we sought to find a new input-independent projection method for reducing the system arising from the Burgers' equation.

4.4.1 Methodology

Algorithm 4.9 (IRKA $\mathbf{V} \oplus \mathbf{W}$). Let $\Sigma = (\mathbf{A}, \mathbf{H}, \mathbf{N}, \mathbf{b}, \mathbf{c})$ be a SISO quadratic bilinear control system (2.16) of size n. Let r_W be the desired number of left projection vectors, r_V be the desired number of right projection vectors and $r_W + r_V = r$ where r is the desired size of the reduced-order model.

1. Apply IRKA using just the linear terms, $\mathbf{A}, \mathbf{B}, \mathbf{C}$, to produce r_W left projection vectors $\mathbf{W}$.
2. Apply IRKA using just the linear terms, $\mathbf{A}, \mathbf{B}, \mathbf{C}$, to produce r_V right projection vectors $\mathbf{V}$.
3. Create orthonormal basis $\mathcal{V}$ using SVD such that
$$\text{span}\{\mathcal{V}\} = \text{span}[\mathbf{V}, \mathbf{W}].$$
4. Create reduced-order model $\zeta = (\mathbf{A}_r, \mathbf{H}_r, \mathbf{N}_r, \mathbf{b}_r, \mathbf{c}_r)$ of size r using a Galerkin projection:
$$\mathbf{A}_r = \mathcal{V}^T\mathbf{A}\mathcal{V}, \qquad \mathbf{H}_r = \mathcal{V}^T\mathbf{H}(\mathcal{V} \otimes \mathcal{V}),$$
$$\mathbf{N}_r = \mathcal{V}^T\mathbf{N}\mathcal{V}, \qquad \mathbf{b}_r = \mathcal{V}^T\mathbf{b}, \qquad \mathbf{c}_r = \mathcal{V}^T\mathbf{c}.$$

Since previous methods had shown that two-sided methods performed better when asymptotic stability was maintained and that one-sided methods always maintained asymptotic stability, we created Algorithm 4.9 in an attempt to capture the benefits of each method. In essence, we select a size r for the reduce order model. We then build r_W left projection vectors $\mathbf{W}$ and r_V right projection vectors $\mathbf{V}$ such that $r_W + r_V = r$. These are combined, and an orthonormal basis is built to span the range of the combined right and left projection vectors. We emphasize the fact that this method does not depend on any prior solutions, nor does it depend on the nonlinear matrices $\mathbf{H}$ and $\mathbf{N}$ associated with the quadratic-bilinear dynamical system.

4.4.2 Results

Similar to the tests run for combining POD and IRKA described in section 4.3.1, we started by building a POD reduced-order model of size $r = 15$ and $r = 20$. Additionally, we built r-dimensional reduced-order models using IRKA, B-IRKA, and QBMOR. To maintain asymptotic stability, all of these models were built using a one-sided Galerkin projection. All of these models were run using the six input functions $u_1(t), \ldots, u_6(t)$. The relative output error was then computed using the same method as equation (4.23). Then a series of tests were run combining the left and right linear $\mathcal{H}_2$-optimal IRKA into a single projection matrix using Algorithm 4.9 to create the reduced-order model. After selecting the reduced-order model size, $r = 15$ or $r = 20$, we created linear $\mathcal{H}_2$-optimal right and left projection vectors using IRKA for sizes $j = 1, 2, \ldots, r$. We built a reduced-order model using Algorithm 4.9 for every possible combination for the number of left projection vectors r_W and right projection vectors r_V such that $r_W + r_V = r$. We then solved each of these reduced order models using each input function $u_1(t), \ldots, u_6(t)$.

Table 4.6: Relative errors for ROM ($r = 15$) for each control function given the number of $\mathbf{V}$ vectors (r_V) and $\mathbf{W}$ vectors (r_W). Combinations not shown or with an NaN did not converge.

r_V	r_W	$u_1(t)$	$u_2(t)$	$u_3(t)$	$u_4(t)$	$u_5(t)$	$u_6(t)$
2	13	3.3608e-01	NaN	NaN	NaN	NaN	NaN
3	12	1.0926e-01	NaN	1.0789e-01	NaN	1.5231e-01	NaN
4	11	3.1494e-02	8.6793e-02	4.4581e-02	NaN	9.5301e-02	7.4208e-02
5	10	8.7638e-03	3.8002e-02	1.7153e-02	1.3407e-01	2.7604e-02	4.6459e-02
6	9	2.5361e-03	1.2243e-02	5.7546e-03	5.4041e-02	1.0876e-02	2.0137e-02
7	8	6.7241e-04	5.2004e-03	2.5799e-03	3.8746e-02	4.4921e-03	8.6424e-03
8	7	5.7876e-04	3.4269e-03	1.7649e-03	3.0796e-02	2.6200e-03	5.4555e-03
9	6	5.8503e-04	3.8505e-03	1.7381e-03	3.0579e-02	2.8272e-03	5.4379e-03
10	5	6.2066e-04	4.5373e-03	1.9250e-03	2.7459e-02	3.3594e-03	7.0218e-03
11	4	7.5621e-04	6.3001e-03	2.5758e-03	2.7180e-02	4.2327e-03	8.8640e-03
12	3	1.3276e-03	8.8344e-03	3.5321e-03	2.8747e-02	5.6981e-03	1.0070e-02
13	2	1.1310e-02	5.3339e-02	3.2766e-02	8.4221e-02	3.7449e-02	3.6137e-02

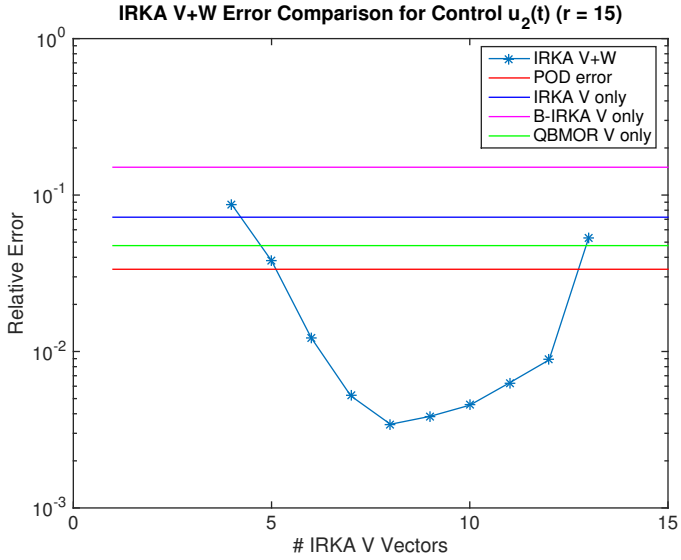
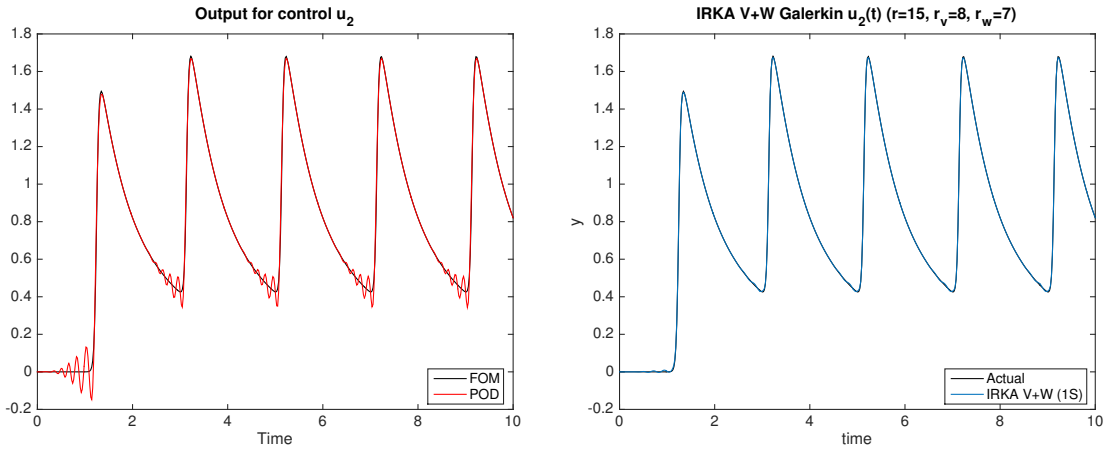


Figure 4.9: Comparison of relative output error of IRKA $\mathbf{V} \oplus \mathbf{W}$ to other methods for $r = 15$.

Table 4.7: Relative errors for all ROMs ($r = 15$).

Method	$u_1(t)$	$u_2(t)$	$u_3(t)$	$u_4(t)$	$u_5(t)$	$u_6(t)$
POD	2.0395e-03	3.3547e-02	2.0889e-02	7.7171e-02	1.9559e-02	3.6554e-02
QBMOR	8.7004e-03	4.7521e-02	2.5424e-02	1.1485e-01	3.1968e-02	3.8208e-02
B-IRKA	6.9133e-02	1.5061e-01	1.0764e-01	1.8894e-01	1.2208e-01	9.4322e-02
IRKA	1.1751e-02	7.2188e-02	4.7789e-02	1.1534e-01	5.0353e-02	5.5483e-02
IRKA $\mathbf{V} \oplus \mathbf{W}$	5.7876e-04	3.4269e-03	1.7381e-03	2.7180e-02	2.6200e-03	5.4379e-03

For our first set of tests, with $r = 15$, Table 4.6 shows the results for each input function $u_j(t)$, $j = 1, \ldots, 6$, and combination of left and right projection vectors, $\mathbf{W}$ and $\mathbf{V}$, respectively. The first observation is that the reduced-order model was stable when almost all the vectors comprising the deflation space came from either the right or left projection vectors. Adding a single left projection vector to the right projection vectors causes the system to lose asymptotic stability. However, if we continue to add left projection vectors until there are approximately the same number of right and left projection vectors, then the reduced-order model is both stable and more accurate than POD. We see a graphical depiction of this in Figure 4.9 where the error is minimized when the right and left projection vectors



(a) POD for input $u_2(t)$ (b) IRKA $\mathbf{V} \oplus \mathbf{W}$

Figure 4.10: Comparison of POD to IRKA $\mathbf{V} \oplus \mathbf{W}$ output for $r = 15$ and input $u_2(t)$.

are balanced. For the Burgers' equation, when the solution tends towards a shock, the POD solution begins to exhibit spurious oscillations. We can see this at times $t \approx 1, 3, 5, 7, 9$ for input function $u_2(t)$ in Figure 4.10a. As can be seen in Appendix D, these oscillations are the primary source of error for the POD models. For the IRKA $\mathbf{V} \oplus \mathbf{W}$ models, the amplitude of these oscillations are dampened when we balance the number of right and left projection vectors. We can see evidence of this in Figure 4.10b. Further, as seen in Table 4.7 and Figure 4.9, IRKA $\mathbf{V} \oplus \mathbf{W}$ is much better than the other one-sided techniques, often reducing the relative error by more than an order of magnitude.

By increasing the reduced-order model size to $r = 20$, the relative error continued to improve as can be seen in Table 4.8. This trend is completely anticipated, but the details are worth discussing. For the POD models, there is a reduction in the amplitude of the previously discussed oscillations. However, the oscillations are still pronounced and noticeable in the POD models (see Figure 4.12a). For IRKA $\mathbf{V} \oplus \mathbf{W}$, the oscillations are almost completely eliminated for $r = 20$, as is apparent in Figure 4.12b. In looking further into this issue, we see from Table 4.9 that it requires significantly more POD basis vectors than IRKA $\mathbf{V} \oplus \mathbf{W}$

Table 4.8: Relative errors for ROM ($r = 20$) for each control function given the number of $\mathbf{V}$ vectors (r_V) and $\mathbf{W}$ vectors (r_W). Combinations not shown or with an NaN did not converge.

r_V	r_W	$u_1(t)$	$u_2(t)$	$u_3(t)$	$u_4(t)$	$u_5(t)$	$u_6(t)$
2	18	3.3522e-01	NaN	NaN	NaN	NaN	NaN
3	17	1.0856e-01	NaN	1.0686e-01	NaN	1.5151e-01	NaN
4	16	3.1186e-02	8.5943e-02	4.4119e-02	NaN	9.2337e-02	7.1059e-02
5	15	8.6703e-03	3.7222e-02	1.6868e-02	1.2331e-01	2.7329e-02	4.2000e-02
6	14	2.5294e-03	1.2193e-02	5.7197e-03	5.1789e-02	1.0802e-02	1.9243e-02
7	13	6.6779e-04	5.1826e-03	2.5694e-03	3.7794e-02	4.4715e-03	8.5274e-03
8	12	4.8096e-04	1.8670e-03	1.3849e-03	1.8826e-02	1.4777e-03	4.3300e-03
9	11	1.9965e-04	8.1184e-04	1.0471e-03	1.0889e-02	4.8614e-04	1.8296e-03
10	10	2.0426e-04	4.7672e-04	9.7037e-04	6.4960e-03	3.1731e-04	1.1618e-03
11	9	2.0023e-04	5.0618e-04	9.7744e-04	6.4252e-03	3.2124e-04	1.1688e-03
12	8	1.9932e-04	5.4304e-04	1.0025e-03	6.3297e-03	3.2189e-04	1.2003e-03
13	7	1.9426e-04	1.2335e-03	1.0643e-03	1.1081e-02	7.7241e-04	2.2312e-03
14	6	2.0990e-04	1.7699e-03	1.0843e-03	1.3283e-02	1.0994e-03	3.2566e-03
15	5	6.9779e-04	5.3562e-03	2.0825e-03	1.8263e-02	3.3030e-03	5.9685e-03
16	4	1.7395e-03	1.1348e-02	5.2856e-03	2.7913e-02	7.3868e-03	1.0233e-02
17	3	8.5211e-03	6.2610e-02	4.1098e-02	1.0390e-01	4.2642e-02	4.9908e-02
18	2	8.1664e-03	6.2213e-02	4.0871e-02	1.0421e-01	4.2340e-02	5.0144e-02

vectors to achieve the same accuracy in the output. Except for the input function that was used to generate the POD snapshots, it required over double the number of POD vectors, $r_P > 40$, to match the performance of IRKA $\mathbf{V} \oplus \mathbf{W}$ with $r = 20$. Further, the IRKA $\mathbf{V} \oplus \mathbf{W}$ method compares quite favorably to the other one-sided methods in Table 4.10, especially when we have balanced the number of right and left projection vectors in the final deflation space.

4.5 Summary

In this chapter, we presented three potential input-independent methods to reduce a nonlinear system by projection. Our goal was to match the stability of POD while meeting or

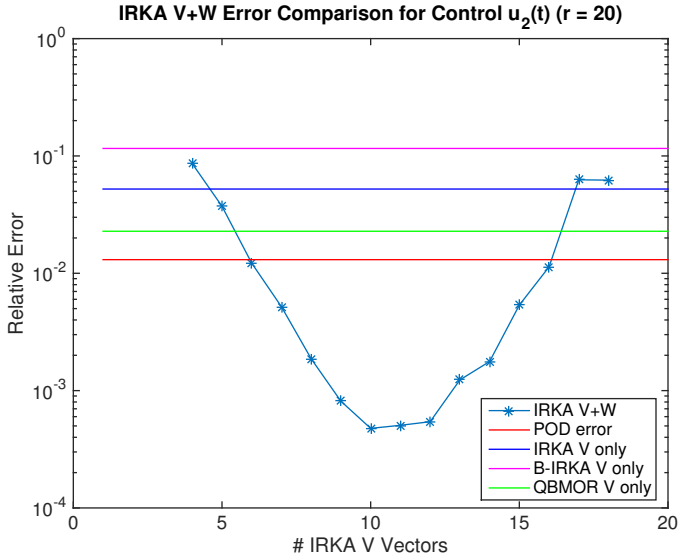


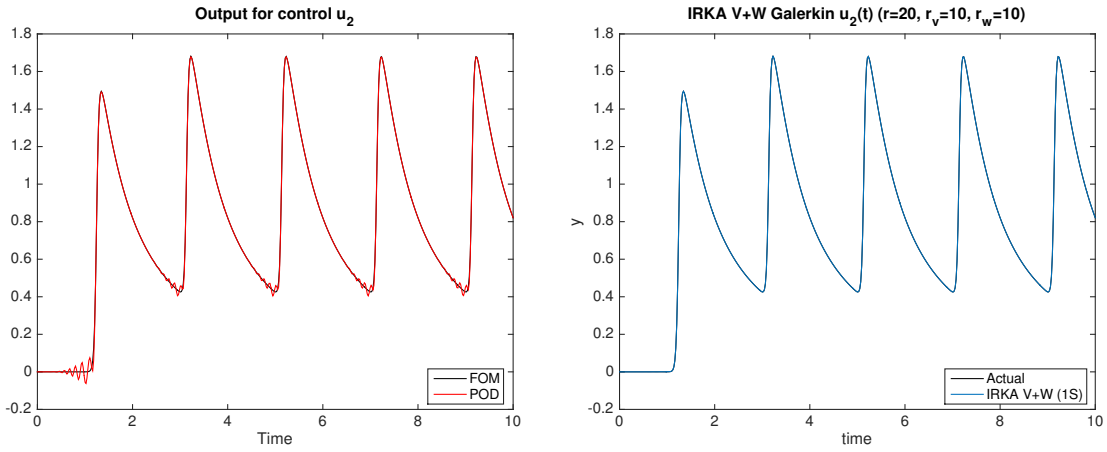
Figure 4.11: Comparison of relative output error of IRKA $\mathbf{V} \oplus \mathbf{W}$ to other methods for $r = 20$.

exceeding the accuracy over a large set of input functions. We chose POD as our benchmark since it is the most commonly used method for reducing nonlinear systems.

For the first method, QBMOR, we noted that when the solution for the reduced-order model built by a two-sided projection converged, it compared favorably with POD. However, the two-sided projections tended to create reduced-order models that did not maintain asymptotic stability. The one-sided projections were stable, but were not as accurate as the POD

Table 4.9: Relative output error for POD over various sizes and input functions.

r	$u_1(t)$	$u_2(t)$	$u_3(t)$	$u_4(t)$	$u_5(t)$	$u_6(t)$
20	3.2651e-04	1.3789e-02	8.1725e-03	4.5189e-02	6.5918e-03	1.8459e-02
25	1.9566e-04	6.5398e-03	3.7455e-03	2.9637e-02	2.5513e-03	1.1450e-02
30	1.8920e-04	2.9968e-03	1.7827e-03	1.8217e-02	1.0229e-03	7.6764e-03
35	1.8910e-04	1.2722e-03	1.1203e-03	1.0488e-02	4.6636e-04	4.0524e-03
40	1.8887e-04	7.0075e-04	9.9613e-04	6.3618e-03	3.2416e-04	1.8659e-03
45	1.8915e-04	3.8085e-04	9.5396e-04	3.0627e-03	2.7753e-04	1.4988e-03
50	1.8915e-04	3.3627e-04	9.4874e-04	2.0341e-03	2.6627e-04	1.0788e-03
IRKA $\mathbf{V} \oplus \mathbf{W}$	1.9426e-04	4.7672e-04	9.7037e-04	6.3297e-03	3.1731e-04	1.1618e-03



(a) POD for input $u_2(t)$ (b) IRKA $\mathbf{V} \oplus \mathbf{W}$

Figure 4.12: Comparison of POD to IRKA $\mathbf{V} \oplus \mathbf{W}$ output for $r = 20$ and input $u_2(t)$.

Table 4.10: Relative errors for all ROMs ($r = 20$).

Method	$u_1(t)$	$u_2(t)$	$u_3(t)$	$u_4(t)$	$u_5(t)$	$u_6(t)$
POD	3.0994e-04	1.3063e-02	7.7132e-03	4.3534e-02	6.1436e-03	1.7336e-02
QBMOR	2.4013e-03	2.2814e-02	1.1544e-02	7.5351e-02	1.4869e-02	2.3034e-02
B-IRKA	3.7288e-02	1.1584e-01	8.0186e-02	1.5691e-01	8.8710e-02	7.7514e-02
IRKA	5.4244e-03	5.2232e-02	3.3940e-02	9.3593e-02	3.4612e-02	4.4689e-02
IRKA $\mathbf{V} \oplus \mathbf{W}$	1.9426e-04	4.7672e-04	9.7037e-04	6.3297e-03	3.1731e-04	1.1618e-03

models. An attempt to augment the QBMOR basis with $\mathcal{H}_2$-optimal basis vectors did not improve stability.

We next attempted to augment the POD basis with $\mathcal{H}_2$-optimal vectors generated by IRKA. This method remained stable when the deflating space contained more POD vectors than IRKA vectors, but was not stable otherwise. There was an improvement in the accuracy of the reduced-order model over a wide range of inputs using this method when the model remained stable. However, it was difficult to determine an optimal number of each basis to use, and these numbers varied depending on the input function. Further, since POD was at the heart of this method, a full snapshot matrix needed to be built, which did allow for the

cost savings in the off-line calculation.

The final method, IRKA $\mathbf{V} \oplus \mathbf{W}$, proved to be the best of the three. Since it did not depend on POD, there was no need to run a full-order simulation prior to building the reduced-order model. When the deflating space contained roughly an even number of right and left projection vectors, then not only was the system stable, but it was significantly more accurate than POD, often decreasing the error by more than an order of magnitude. It typically required over twice as many POD vectors as IRKA $\mathbf{V} \oplus \mathbf{W}$ vectors to achieve the same accuracy for input functions other than the one used to generate the snapshot matrix. Even for the input function used to generate the snapshot matrix, the IRKA $\mathbf{V} \oplus \mathbf{W}$ method was more accurate than POD for a given reduced-order model size r. This method appears to be a promising substitute for POD on certain nonlinear systems. In general, roughly an equal number of right and left projection vectors, generated using IRKA, should be used to build the deflating space to ensure stability and produce an accurate model.

Chapter 5

Reduction of Fire Models

The modeling and simulation of fires is a complex, multi-scale problem. The need to create accurate models extends from prevention to prediction to damage control management. Since these problems are so complex, even models of moderately-sized fires in relatively small domains require significant computational resources and time to solve. Further, the actual physics are often difficult to model exactly, e.g. the exact fuel loading and distribution in a burning room or wildland fire. To this end, much of the focus in the fire community has been on simplifying the physics of the problem to obtain reasonable models. Current techniques being used for fire models, such as POD for wildland fires [49, 85], often have limited effectiveness due in large part to the inherent nonlinearities that exist in fire models.

In this chapter, we focus on three problems involving fire where we use reduced-order modeling to reduce the computational time while maintaining a suitable level of accuracy. We begin by discussing the issue of coal mine ventilation and its relationship to the prevention of fires due to methane build up at a coal mine face. The need here is to provide surrogate models with a sufficient accuracy that can be solved quickly enough to assist in making design and best practice decisions for curtain placement and required inlet mass flow

rates. Next we examine small plume fires in an open environment. Here, we want to ensure that we can incorporate the full spectrum of physics, such as airflow, buoyant convection, species transport, and combustion, in a ROM and still provide accurate results. We close the chapter by discussing a phenomenological model for the radial spread of a wildfire over flat land with a known fuel density [49, 71, 72, 85]. POD has been applied to these models, e.g. [49, 85], but the computational gains have been modest. By addressing the lifting bottleneck, as discussed in Section 2.3.3, using the discrete empirical interpolation method (DEIM) [27, 28, 35], we show that the time required to solve this reduced-order system can be significantly shortened.

5.1 Airflow in a Mine

Here we motivate the reduction of computational models of coal mine fires. In a large number of coal mines today, coal is mined using a machine known as a continuous miner (CM). The CM "shaves" coal from the face using tungsten teeth mounted to a rotating drum on the front of the machine. Methane entrained in the coal is released at the CM face where the coal is being mined. Normally, ventilation airflow at the CM face removes the methane in sufficient quantities to prevent a fire or explosion. However, depending on the geometry, including the position of a ventilation curtain and the magnitude of the flow, isolated pockets of methane can build up to an ignition threshold and result in a catastrophic event. Therefore understanding airflow is critical to the prevention of fires and/or explosions in and around the CM face. Figure 5.1 shows the typical configuration at the CM face.

Current techniques for simulating airflows in a mine fall into two categories: computational fluid dynamics (CFD) models (ANSYS Fluent , COMSOL, OpenFOAM, FDS) [88, 91, 92] and network models (Ventsim, VUMA, MFIRE) [20, 23, 30, 68, 95]. The resolutions required

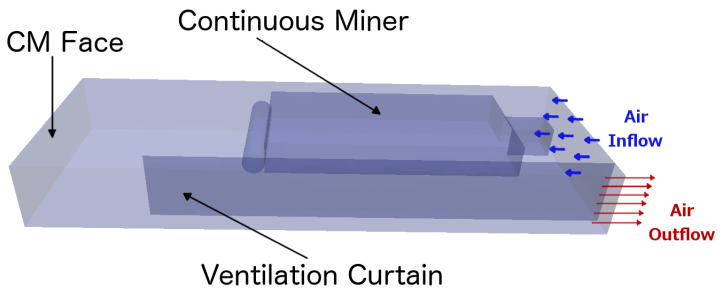


Figure 5.1: Typical configuration at the CM face showing the positioning of the ventilation curtain and CM.

for CFD solutions to be accurate result in mathematical models which are so large that it is not practical to simulate the entire mine at once. Further, even for computationally tractable domain sizes, the computational solve times often exceed the required time constraints for design or decision making. To overcome these limitations, most designers and decision makers look to network models as an acceptable alternative to the CFD models. However, the resolutions of these network models are so coarse that eddy points and gas accumulation pockets are not able to be realized in the model.

We focus on a third alternative that, up to this point, has seen little exposure in the mining community. Using model reduction techniques, we seek to obtain a computational solve time that meets the desired time requirements while maintaining enough system accuracy to be able to reliably predict potential problems resulting from methane gas build up.

5.1.1 The Model Reduction Technique

We used POD with Galerkin projection to reduce a finite volume approximation of our mine airflow model. This method is well-suited for our problem size and structure. In particular, the mathematical model discussed in the next section lends itself to being solved using the technique covered in Section 2.3.3. When using this method, we are able to circumvent the

lifting bottleneck issue by precomputing several small matrices. While this method is a good fit for our problem, we explore some of the other advantages and disadvantages associated with using this method.

POD excels as a model-order reduction technique used to simulate flow solutions around known solutions at a given set of inputs. So, if there is a given flow where the data is generated either experimentally or computationally via a full-order simulation, then POD can be used to approximate flows sufficiently close to the given flow [18]. POD is one of the most used model-order reduction techniques for non-linear systems [4]. While some model order reduction techniques are focused on input to output mappings, POD is designed to recreate the entire state space [18] as a linear combination of the individual POD modes. Thus, it is a good technique to use if the goal is to obtain an approximation of what the flow is doing in the entire space.

POD is very dependent on the quality and quantity of snapshots. First, the snapshot solutions must be representative of the types of solutions being modeled. For example, the kinematic viscosity and inlet velocity used to generate snapshots should be close to values being used in the reduced-order model (ROM). Additionally, since POD determines the most prominent flow modes by using an eigenvalue computation, the number of reliable modes is about ten percent of the total number of snapshots in most cases [44]. So, if there are one hundred snapshots, then we can expect to obtain on the order of ten reliable POD modes. Finally, creating the actual ROM from the POD modes can be challenging for complex fluid flows. In particular, solving the ROM quickly requires the computation of several matrices. These matrices must be stored and then loaded when the ROM is being solved.

5.1.2 Basic Description

We modeled the airflow in the mine using the Navier-Stokes equations for isothermal, incompressible flows presented here in (5.1)-(5.2). Again, this model is well-suited for typical flow speeds and temperatures encountered in a coal mine during normal operations [88]. Further, we can see that these equations are simply the first two equations (2.1)-(2.2) of the general fire model without the body force term.

$$\frac{\partial \mathbf{u}}{\partial t} - \mu \nabla^2 \mathbf{u} + (\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla p = 0 \tag{5.1}$$

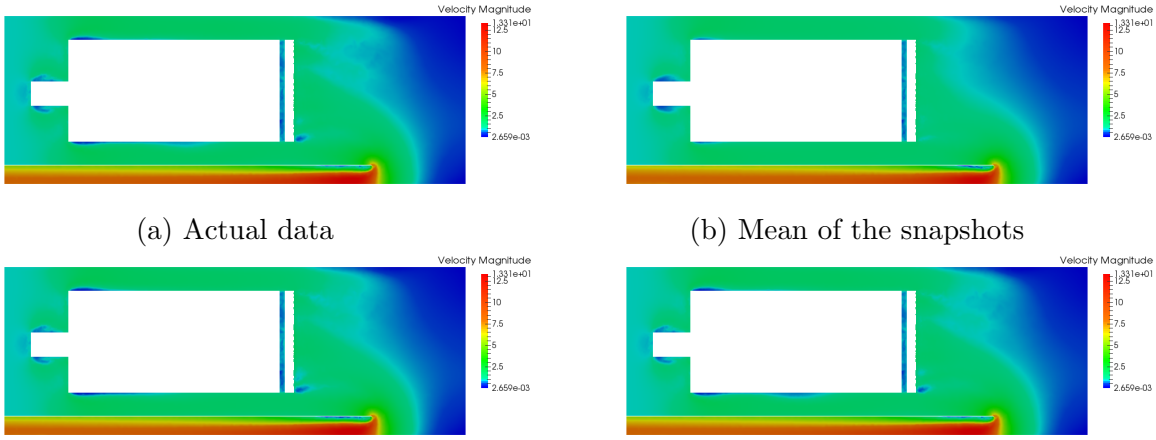
$$\nabla \cdot \mathbf{u} = 0. \tag{5.2}$$

Reviewing Section 2.3.3, for any given time and n spacial nodes in the domain, we can write the velocity flow of the air, $\mathbf{u} = [u\ v\ w]^T \in \mathbb{R}^{3n}$, as a combination of a centering vector $\bar{\mathbf{u}}(\mathbf{x})$ and an infinite set of basis vectors, $\phi_i(\mathbf{x})$, as

$$\mathbf{u}(t, \mathbf{x}) = \bar{\mathbf{u}}(\mathbf{x}) + \sum_{i=1}^{\infty} \phi_i(\mathbf{x})\alpha_i(t). \tag{5.3}$$

Then an r-dimensional approximation of the velocity, (5.4), can be created by using r basis vectors. Here we have decomposed the basis into its u, v, and w components as $\phi_i(\mathbf{x}) = [\phi_i^u(\mathbf{x})\ \phi_i^v(\mathbf{x})\ \phi_i^w(\mathbf{x})]^T$.

$$\begin{aligned} u(t, \mathbf{x}) \approx \widetilde{u}(t, \mathbf{x}) &= \bar{u}(\mathbf{x}) + \sum_{i=1}^{r} \phi_i^u(\mathbf{x})\alpha_i(t), \\ v(t, \mathbf{x}) \approx \widetilde{v}(t, \mathbf{x}) &= \bar{v}(\mathbf{x}) + \sum_{i=1}^{r} \phi_i^v(\mathbf{x})\alpha_i(t), \\ w(t, \mathbf{x}) \approx \widetilde{w}(t, \mathbf{x}) &= \bar{w}(\mathbf{x}) + \sum_{i=1}^{r} \phi_i^w(\mathbf{x})\alpha_i(t). \end{aligned} \tag{5.4}$$



(a) Actual data

(b) Mean of the snapshots

(c) 3 POD modes

(d) 10 POD modes

Figure 5.2: Comparison of the airflow in the full model versus the airflow at the same time step generated using various POD modes.

To find an appropriate basis for the system, we first generate $k > r$ full-order snapshot solutions $\mathbf{S} = [\mathbf{u}_1\ \mathbf{u}_2 \dots \mathbf{u}_k]$ for the velocity field using ANSYS Fluent as suggested in [88]. We generated the POD modes, as described in Section 2.3.3, by subtracting the mean flow from each snapshot in $\mathbf{S}$, performing a singular value decomposition (SVD) of the resulting matrix, and then selecting first r singular vectors $\Phi = [\phi_1 \dots \phi_r]$. Using the POD modes, Φ, as the basis functions for the approximation in (5.4), we are guaranteed the best r-dimensional representation of the snapshot solutions. In Figure 5.2, we show an example of this approximation at the CM face for a given time snapshot. We see that as the number of POD basis vectors we use increases, the approximation to the actual airflow improves. Here, we are just demonstrating how well the POD basis can represent a given solution.

In practical terms, we are not interested in simply reproducing an existing solution; rather, we would like to build a reduced-order model that can then be used to solve for new solutions given a different set of parameters and initial conditions. If the anticipated behavior of the system is close to the snapshot solutions, and thus can be accurately represented in the POD

Table 5.1: Error between the actual and reduced-order model velocity profiles at 10.0 seconds.

$\overline{U}_{in}$	$\overline{U}_{out}^{Act}$	$\overline{U}_{out}^{ROM}$	$\% \, \overline{U}_{out}^{Err}$	$\% \, U^{Err}$
0.90	7.0903	7.0912	0.0121	4.9970

basis, we can expect to represent the behavior of the nearby system reasonably well with the POD basis functions given by Φ. Using these POD modes as test functions, the weak formulation of equations (5.1)-(5.2) can be used to perform Galerkin projection as we covered in Section 2.3.3. Applying a finite element discretization, we obtain an r-dimensional system of ODEs that can be solved to obtain the POD basis coefficients $a(t)$ for a given time interval $t = [0, t_f]$. Substituting these coefficients into our approximation (5.4), we can recover an approximation to the distributed flow states in the full-order model.

5.1.3 Results

The POD vectors and reduced-order model were created using the snapshots generated from an ANSYS Fluent simulation run with a 0.9 m/s input velocity. The data was then exported into Matlab® . We note that ANSYS Fluent solves for the velocity at the center of each volume element, but exports the data averaged to the nodes. The consequence of this is that average output velocity as calculated from exported data varies slightly from the internal ANSYS Fluent calculation. There were 100 data snapshots created from 0.1 to 10.0 seconds. Due to the accuracy of the eigenvalue solver, we restricted our reduced-order model to ten POD modes. The total time required to generate the POD modes from the data snapshots and to create the associated reduced-order model was approximately 7.5 minutes. Using the initial condition from the snapshots to initialize the reduced-order model, the system was integrated forward in time from 0.1 to 10 seconds to generate an approximate solution. This computation took approximately 0.5 seconds to complete. The resulting error between the

actual system and the ROM is given in Table 5.1. Further we looked at the average velocity at the output face over time and compared it to the reduced-order model. As is shown in Figure 5.3, the ROM is a good approximation of the actual system over the entire time interval.

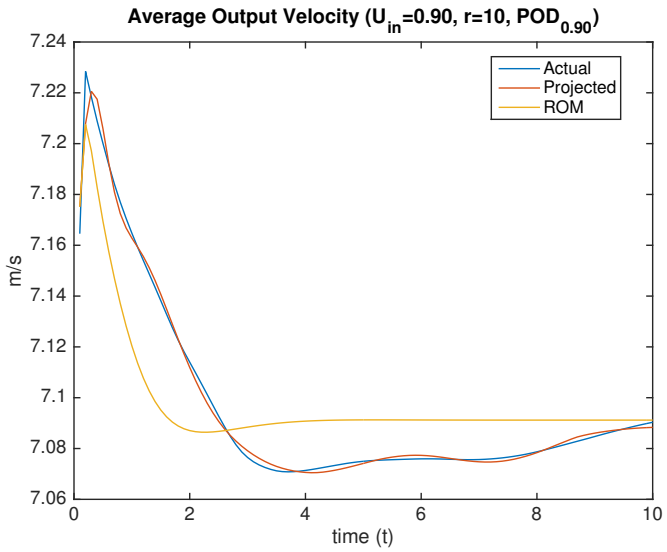


Figure 5.3: Average outlet velocity magnitude given an average input velocity magnitude of 0.88 m/s.

While these results are good, it is ultimately just an approximation to simulation data that we already had. Therefore the total time to create this ROM was the two hours required to generate the actual data plus the seven to eight minutes necessary to generate the ROM. However, the benefit of reduced order modeling is the ability to use the ROM created for one initial condition to approximate nearby solutions without having to generate the full solution.

The time savings, as shown in Table 5.2, compound as each subsequent solution is generated from the original ROM. While the time savings are obvious, it would not matter if the nearby solutions created using the ROM were not good approximations of the full solutions. To test the accuracy, we generated full solutions at input velocities of 0.88 m/s and 0.92

Table 5.2: Approximate computation times in hh:mm:ss for full-order versus reduced-order models.

	Computation Time	
$\overline{U}_{in}$ m/s	Full Model	ROM
0.90	2:00:00	2:08:00
0.88	2:00:00	0:00:03
0.92	2:00:00	0:00:03
Total	6:00:00	2:08:06

m/s. Each of the solutions took approximately two hours to compute on a 35 processor high-performance computer. We then computed the solutions using the previously generated ROM, but changed the initial conditions to the alternate input velocities. To load the ROM required about 2 seconds and creating the approximate solution took about 0.5 seconds.

Table 5.3: Error between the actual and reduced-order model velocity profiles at 10.0 seconds.

$\overline{U}_{in}$	$\overline{U}_{out}^{Act}$	$\overline{U}_{out}^{ROM}$	% $\overline{U}_{out}^{Err}$	% U^{Err}
0.88	6.9315	6.9328	0.0180	5.0221
0.92	7.2491	7.2494	0.0048	4.9951

The error comparison in Table 5.3 shows that the approximate solutions compare favorably to the actual solutions. Further, as with the earlier simulation, we can see in Figures 5.4 and 5.5 that the ROM solution does a good job approximating the average output velocity magnitude for nearby initial conditions. We also see that it does a decent job capturing the output airflow dynamics over the entire time of the simulation.

5.2 Fire Plumes

With our focus still on providing reduced-order models for physical processes inside a coal mine, we look at fire plumes. For our discussions, these are relatively small fires (approx-

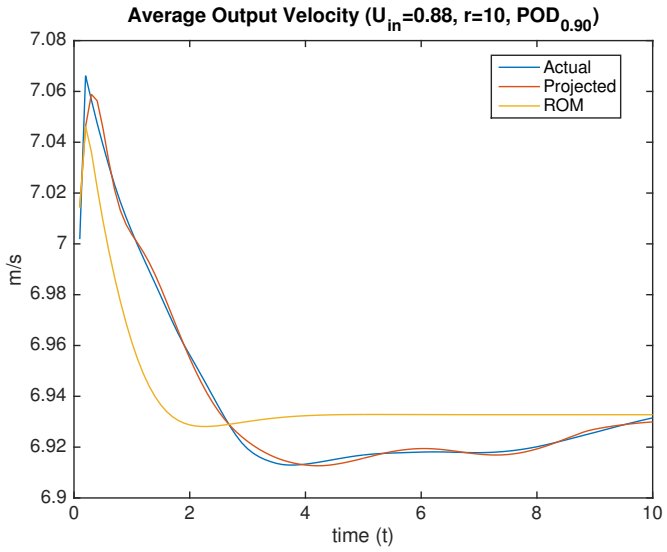


Figure 5.4: Average outlet velocity magnitude given an average input velocity magnitude of 0.88 m/s.

imately 40 kw) that are localized and unconstrained by obstructions in the environment. Our goal here was to understand the dynamics of a fire and to determine if a reduced-order model could produce a suitable reproduction of the fire. To define a suitable reproduction, we focused on three aspects of the fire to compare.

Remark 5.1 (**Suitability of a ROM for Modeling a Fire).** Fire simulations involve fine scale discretizations of coupled nonlinear PDEs. The complexity of the simulations are such that one cannot expect reduced-order models to accurately capture every feature of the simulations. In fact, a fundamental premise of reduced-order modeling in these cases is an underlying low-dimensional manifold for the full-order simulation. However, many of the applications of fire modeling do not require knowledge of the states at every point in time and space, but rather they need to capture certain characteristics of the fire that would be useful for either making safety design decisions or to evaluate real-time fire suppression strategies. Thus, we present qualitative measures of reduced-order models that are important to capture.

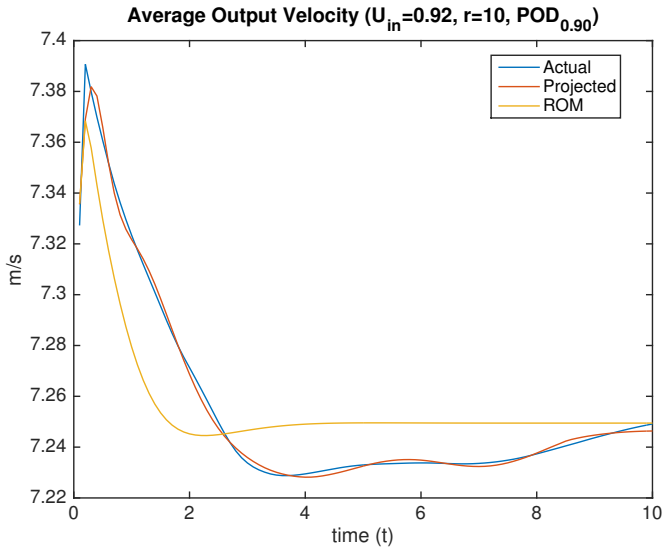


Figure 5.5: Average outlet velocity magnitude given an average input velocity magnitude of 0.92 m/s.

When considering how well a ROM of a fire matches the full-order fire model we will consider the following criteria. The idea here is that we are not necessarily interested in matching the fine scale details in a fire in time and space, but rather at certain characteristics of the fire that would be useful for either making safety design decisions or for real-time fire suppression strategies.

1. **Dynamics:** First, the ROM had to capture the "essential" dynamics of the fire. In other words, the ROM should actually look "like a fire". Thus, large-scale dynamical structures such as vortex shedding, flow oscillations, and other qualities present in a full-order model need to be captured. Further the general shape of the fire plume as measured by the velocity and temperature needed to match the FOM.

2. **Magnitude:** The ROM needs to generally match the magnitudes of the fire (to predict spread and ignition). We consider two quantitative measures to track our success. First of all, we track the maximum temperature or velocity magnitudes over the entire domain. The precise location of the maximum magnitudes are not as important in this

measure. The knowledge of these magnitudes can indicate whether or not new combustion can take place or whether or not a fire suppression strategy can be successful. We also consider the mean of the temperature and square of velocity magnitudes over the entire domain as an indication of energy. In particular, let $T(\xi,t)$, $\mathbf{u}=(u(\xi,t),v(\xi,t))$ be the solutions for temperature and velocity over the domain, Ω, respectively, of the fire model. Then, we define the following

$$\mathcal{F}_T(t)=\max_{\xi_k\in\Omega} T(\xi_k,t) \qquad \mathcal{F}_{\mathbf{u}}(t)=\max_{\xi_k\in\Omega}\|\mathbf{u}(\xi_k,t)\|^2, \tag{5.5}$$

$$\mathcal{G}_T(t)=\frac{1}{N}\sum_{k=1}^{N} T(\xi_k,t) \qquad \mathcal{G}_{\mathbf{u}}(t)=\frac{1}{N}\sum_{k=1}^{N}\|\mathbf{u}(\xi_k,t)\|^2. \tag{5.6}$$

where $\xi_k\in\Omega$ are the discrete solution points for the FOM. Similarly, for the ROM solutions of temperature, $T_r(\xi,t)$ and velocity $\mathbf{u}(\xi,t)$ we have

$$\mathcal{F}_{T_r}(t)=\max_{\xi_k\in\Omega} T_r(\xi_k,t) \qquad \mathcal{F}_{\mathbf{u}_r}(t)=\max_{\xi_k\in\Omega}\|\mathbf{u}_r(\xi_k,t)\|^2, \tag{5.7}$$

$$\mathcal{G}_{T_r}(t)=\frac{1}{N}\sum_{k=1}^{N} T_r(\xi_k,t) \qquad \mathcal{G}_{\mathbf{u}_r}(t)=\frac{1}{N}\sum_{k=1}^{N}\|\mathbf{u}_r(\xi_k,t)\|^2. \tag{5.8}$$

We compare the FOM to the ROM both graphically and numerically. For the graphical comparison, we simply plot the functions in (5.5)-(5.6) alongside the corresponding functions in (5.7)-(5.8). For the numerical comparison, we take the average over a selected time range for each of the functions (5.5)-(5.6) and compare them to the averages of (5.5)-(5.6) over the same time range.

3. **Oscillation Frequency and Amplitude:** This is a more esoteric quantity but can be mathematically quantified by looking at the dominant Fourier frequencies in the amplitude coefficients and comparing these to those of the FOM projected onto the same basis functions.

We see that these criteria do not seek to strictly minimize an error between the FOM and ROM at some particular time step. The issue is that due to the complexity and dynamics of the fire, it would be almost impossible to produce a ROM that would be able to exactly capture the behavior. Instead we seek to match the essence of the fire in an effort to determine the feasibility of ROMs for fires. In this section, we first discuss our general methodology and then look at some of the initial challenges we faced when attempting to reduce the FOM. Finally, we discuss the factors above with regards to the quality of the ROM.

5.2.1 Description and Methods

As stated earlier, our focus for this section is on creating reduced-order models of small plume fires. We used the Fire Dynamics Simulator (FDS) software to generate the full-order fire model. Our domain was a 1 m $\times$ 1 m $\times$ 2 m hexahedron open to the atmosphere on all four sides and the top. The base of the domain was a solid gypsum plaster surface with a 0.2 m $\times$ 0.2 m methane burner centered on the base. The burner was set to release methane at a rate to generate a 40 kw fire. Figure 5.6 shows few image snapshots of the fire.

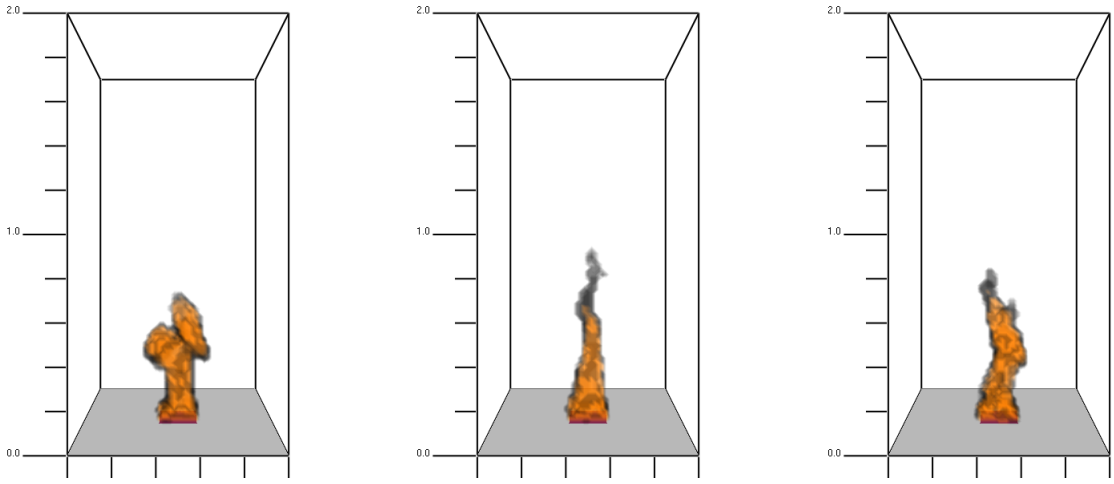


Figure 5.6: FDS full-order fire model used to generate the various ROMs.

With the base geometry determined, the system was discretized to 50 elements in the x direction, 40 in the y direction and 100 in the z direction. We simulated the system over 20 seconds. We then took a slice of the data at $y = 0.5$ for temperature and velocity to create a two dimensional snapshot over the 1 m $\times$ 2 m rectangular domain $\mathbf{x} = [x, y]$[1]. This gives a mesh of nodal values evenly spaced every 0.02 m in both directions, i.e. 51 $\times$ 101 nodes. This gives a full-order size of $n = 5151$. We then captured 500 evenly spaced snapshots over the time domain $t = [0, 20]$ to represent the entire full-order plume fire model. To produce a ROM from this data, we used the PDE in (5.9)-(5.11) for our model.

$$\frac{\partial \mathbf{u}}{\partial t} = -\mathbf{u} \cdot \nabla \mathbf{u} - \nabla p + \nu \nabla^2 \mathbf{u} - \beta \mathbf{g}(T - T_\infty), \tag{5.9}$$

$$0 = \nabla \cdot \mathbf{u}, \tag{5.10}$$

$$\frac{\partial T}{\partial t} = -\mathbf{u} \cdot \nabla T + \alpha \nabla^2 T. \tag{5.11}$$

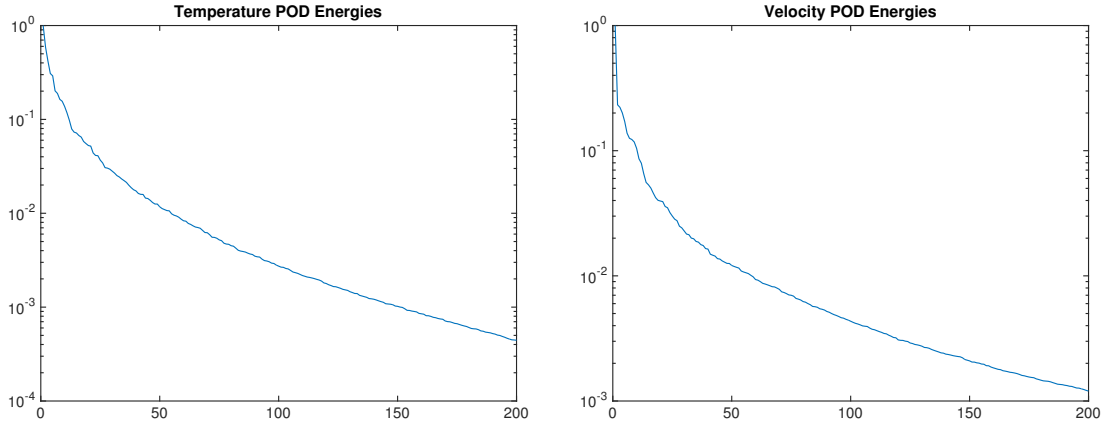
We see that these are the same equations as the general fire model equations (2.1)-(2.3) with the chemical reaction term removed. In particular we wanted to investigate how close we could get to the actual fire model without incorporating the combustion. From the snapshots we created r POD basis modes for the u velocity, v velocity, and temperature T. We used the technique from Section 2.3.3 to generate the ROM. We then ran several tests to compare the ROM to the FOM generated by FDS. To best match the FOM and provide asymptotic stability in the ROM, we adjusted the diffusion constants ν and α. While this process was done manually for our research, this process can be automated using sensitivity analysis [56, 57].

[1]To bring the notation in line with previous discussions, we will refer to the z direction as y for the rest of the section when discussing the two-dimensional domain, i.e. $\mathbf{x} = [x, y]$.

5.2.2 Buoyancy-Driven Flows

Initially, we assumed that temperature contraction coefficient, β, was constant over the domain. We had used this constant parameter when we were working on the natural circulation problem, see Chapter 3. However, when we attempted to adjust the value for β, ν, and α to build a stable ROM that accurately reflected the FOM, we were unable to create any ROMs that resembled the FOM. In particular, any stable ROM that we obtained did not match any of the suitability criteria given earlier in Remark 5.1. In particular, the maximum and mean values for the ROM were not close to the FOM. Additionally, the ROM quickly produced a constant solution with none of the dynamics that are apparent in the FOM.

After reconsidering the model, we determined that fundamentally (5.9)-(5.11) were correct, but that β was actually temperature dependent over the range of temperatures present in the domain. As seen in [43], β is actually given by $\beta = \frac{1}{T}$. With temperature variations of over 1000 K across the domain, β varies by over three orders of magnitude across the domain at a given time step. By simply changing the body force term to be nonlinear by modifying β to its actual definition, we were able to produce significantly better ROMs, especially with respect to the dynamics of the ROM. The downside is that the body force term is no longer a term that can be handled by precomputing the inner product. We are handling the term by lifting the temperature to the full state space and then projecting it back down. For our testing of these ROMs, this is currently not an issue, because of the state space sizes we have. However, when we move to three-dimensional models this will need to be addressed, most likely by using DEIM.



(a) Temperature (b) Velocity

Figure 5.7: Decay of the singular values associated with the first 200 POD modes.

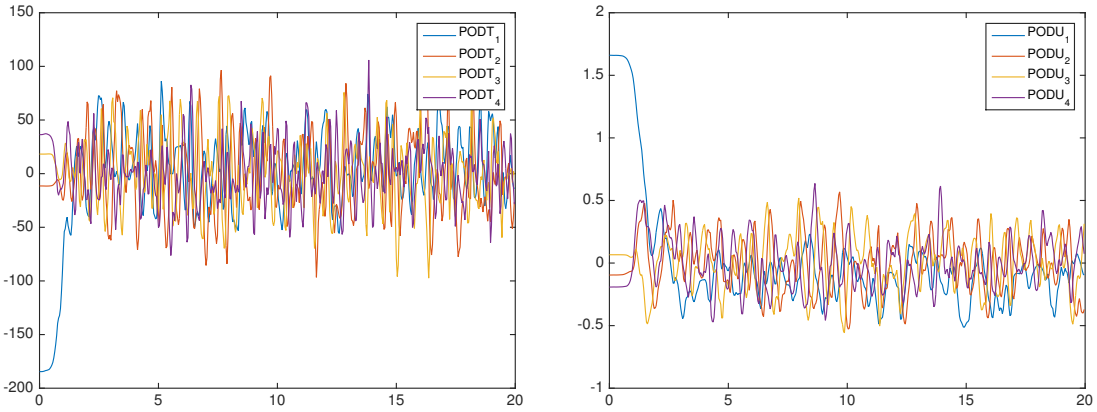
5.2.3 Numerical Results

When creating a ROM POD as discussed in Section 2.3.3, there are essentially five main steps that we must accomplish as listed in the following high-level algorithm.

Algorithm 5.2 (High-Level Steps to Building a ROM from Projecting the PDE using POD).

1. Capture data snapshots.
2. Build POD basis functions from the snapshots.
3. Compute and store the intermediate inner product matrices required to create the ROM ODE to be solved.
4. Load the inner product matrices and assemble the ROM ODE.
5. Solve the time-dependent ROM ODE using a standard ODE solver.

We will consider each of these steps in turn and look at the numerical results along with any challenges that we needed to manage. It is worth noting that steps 1-3 in Algorithm 5.2 are considered off-line costs. Once they have been accomplished then we do not need to



(a) Temperature (b) Velocity

Figure 5.8: Dynamics of the fire model shown with respect to the first four POD coefficients for temperature and velocity.

repeat them unless we change the size of our ROM or alter the data snapshots. Steps 4-5 are the real-time costs for solving the problem. We emphasize that we change the diffusion constants or the initial conditions without having to repeat steps 1-3. As we discuss each step, we will cover the average time required to complete that step.

Capture Data Snapshots

As discussed before, we used FDS to create the data snapshots. It is a finite volume based solver, and therefore every domain is considered to be three-dimensional (3D). The system can handle two-dimensional (2D) domains by setting the y direction to be one cell deep. Due to some of the numerics associated with FDS the depth of cells must be kept small, see [75]. While there are some challenges correctly setting the size of the fire when using this 2D mode of FDS, it is much quicker than running an appropriately sized and discretized 3D model. In general, computation times for the 3D model were about 40 minutes for a 20 second simulation. In contrast, the 2D models required 8-9 minutes to compute a 20 second simulation. After FDS, it took approximately one minute to extract the data into snapshots

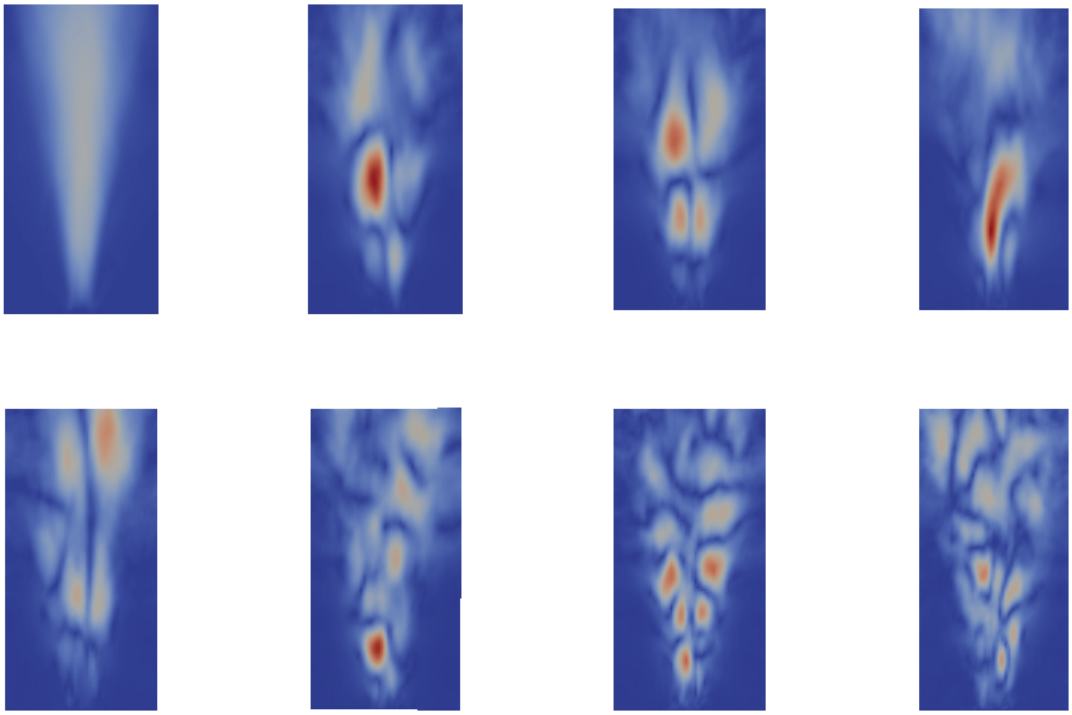


Figure 5.9: Magnitude of the velocity POD modes for the small plume fire. Here we show POD modes 1, 2, 3, 4, 5, 10, 15, and 20 from left to right, top to bottom.

and convert it to a format usable by MATLAB® .

Build POD Basis

For our ROM we set the size to $r = 20$ based on the decay of the singular values associated with the velocity and temperature snapshots as seen in Figure 5.7. Specifically, we create 20 POD basis functions each for the u velocity, v velocity, and temperature. This process requires less than a minute to complete. Looking at the POD modes in Figures 5.9 and 5.10, we see that even though the fire seems to have a random behavior, there are some underlying structures in the velocity and temperature. Further, the initial POD modes capture the overall shape and distribution of the velocity and temperature, whereas the

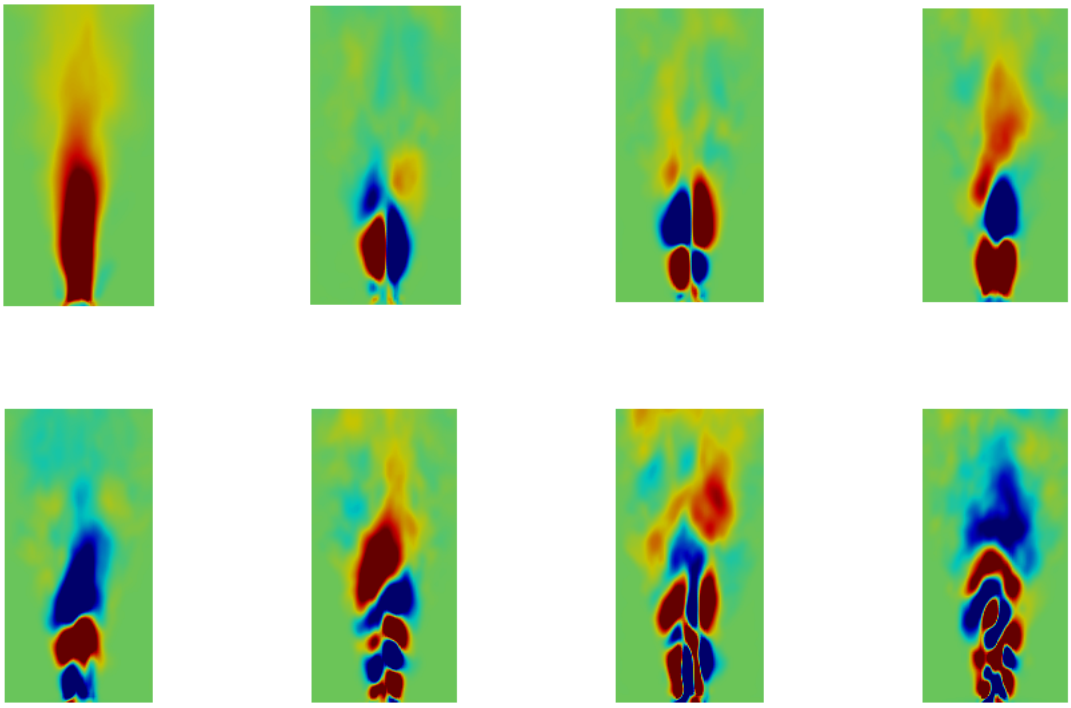


Figure 5.10: Temperature POD modes for the small plume fire. Here we show POD modes 1, 2, 3, 4, 5, 10, 15, and 20 from left to right, top to bottom.

higher modes capture the finer details. Further, in Figure 5.8, we show the temperature and velocity coefficients for the first four POD modes for the 20 second simulation. We can see from the oscillations just how complex the dynamics are for the fire model.

We did build larger ROMs to see how well they would perform, but as we will see in the coming steps, they did perform as well when taking into account the quality of the ROM and the time required to solve the model. In general, we were seeking models that were better than real-time.

Table 5.4: Sample times to build and store the inner product matrices required to construct the ROM ODE.

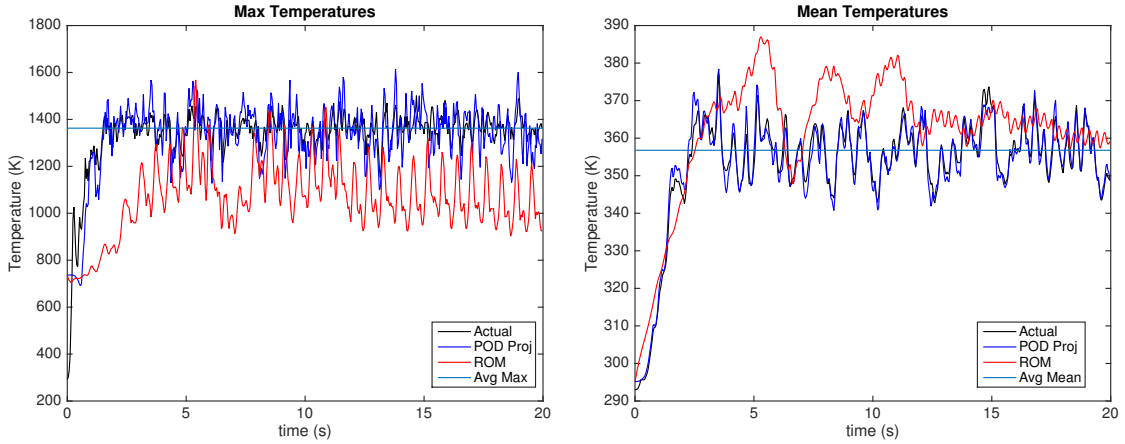
r	Time to build (s)	Time to store (s)
20	13.97	0.55
50	34.25	2.61
100	70.05	12.65
200	154.26	61.09

Compute Inner Products

While there are no visual results to show at this stage, this is the step that requires the longest to complete other than the initial generation of the snapshots. The time required is related to the size of the reduced-order model. The times in Table 5.4 give an indication of the typical amount of time required to compute and store the matrices. These times do vary by a second or two, but remain approximately the same. Additionally, the routines used to compute the inner products had to be compiled into MATLAB® MEX routines. When not compiling the routines, it took as much as ten to twenty times longer to perform the calculations.

Assemble the ROM ODE

In general this step is very quick to complete, and even for ROMs of size $r = 200$, this only took less than two seconds. Of note here is that while the linear matrix and constant vector can be computed prior to solving the ROM ODE, β and the quadratic terms must be solved at each ODE solver time-step. For the quadratic term, this is simply a series of r-dimensional matrix-vector multiplications, and therefore is fairly quick. As mentioned before, a more efficient solution for calculating β may need to be done when moving to larger original systems, such as 3D fire models.



(a) Maximum $\widetilde{\mathbf{T}}(t)$ (b) Maximum $\widetilde{\mathbf{T}}(t)$

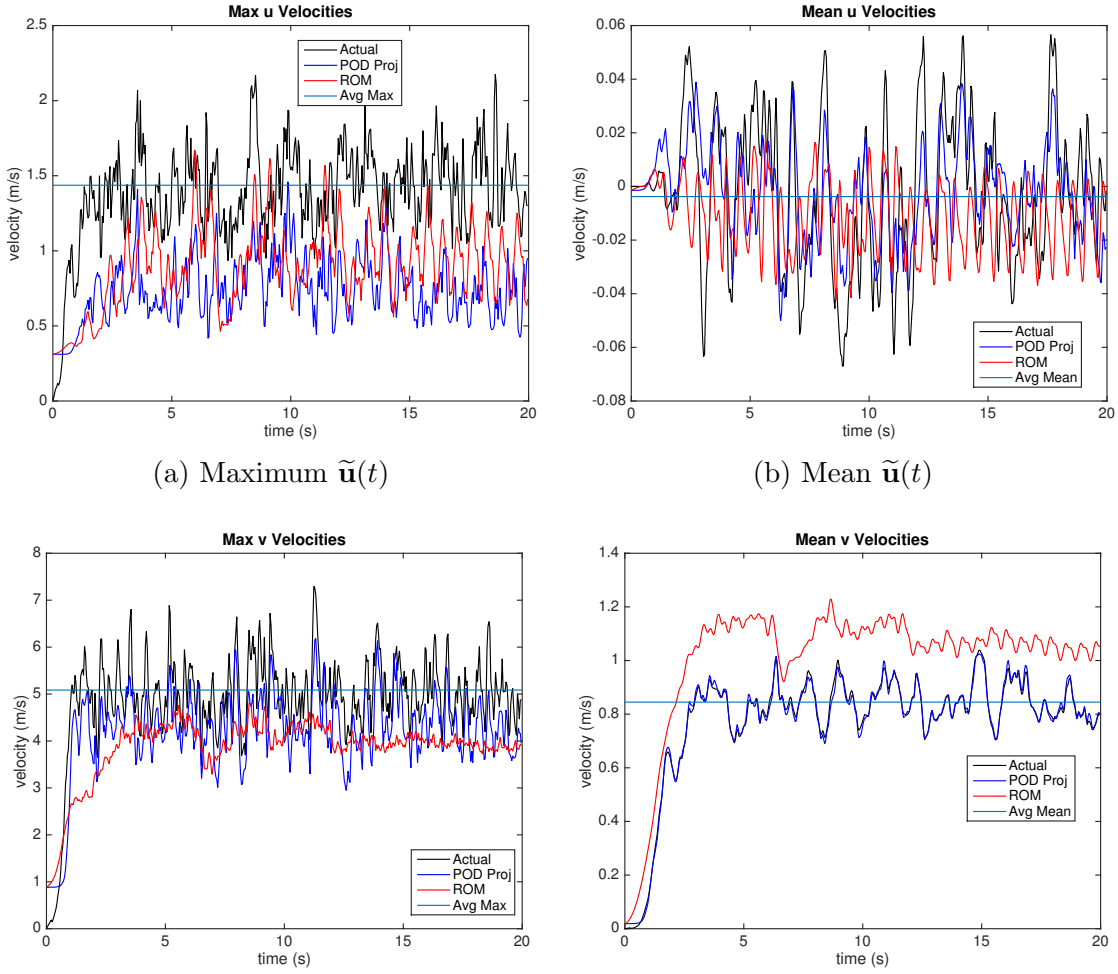
Figure 5.11: Mean and maximum temperature comparison between ROM and FOM with $r = 20$.

Solve the ROM ODE

Table 5.5: Values used for diffusion constants ν and α based on ROM size along with the typical solution time for the ROM ODE using those values.

r	ν	α	Solution time (s)
20	1.0e-02	1.0e-02	3.7
50	9.0e-03	9.0e-03	14.4
100	3.0e-03	3.0e-03	72.3

Our final step is actually solving the ROM ODE. As shown in Section 2.3.3, we are solving for the coefficients of the POD basis functions Φ over the simulation time. Once we have numerically solved the ROM ODE, we can apply the coefficients to the basis functions and add the centering vector back to obtain our approximate solution. To provide the best solution, we adjusted the diffusion parameters based on the size of the ROM. Typically, those constants became smaller as the size of the ROM became larger. This is consistent with typical computational fluid dynamics' techniques used to stabilize the flow on coarse meshes, see for example [37, 55, 59, 65]. Table 5.5 shows the constants that were used for various ROM sizes. Also, as one would expect, the time required to solve the ROM



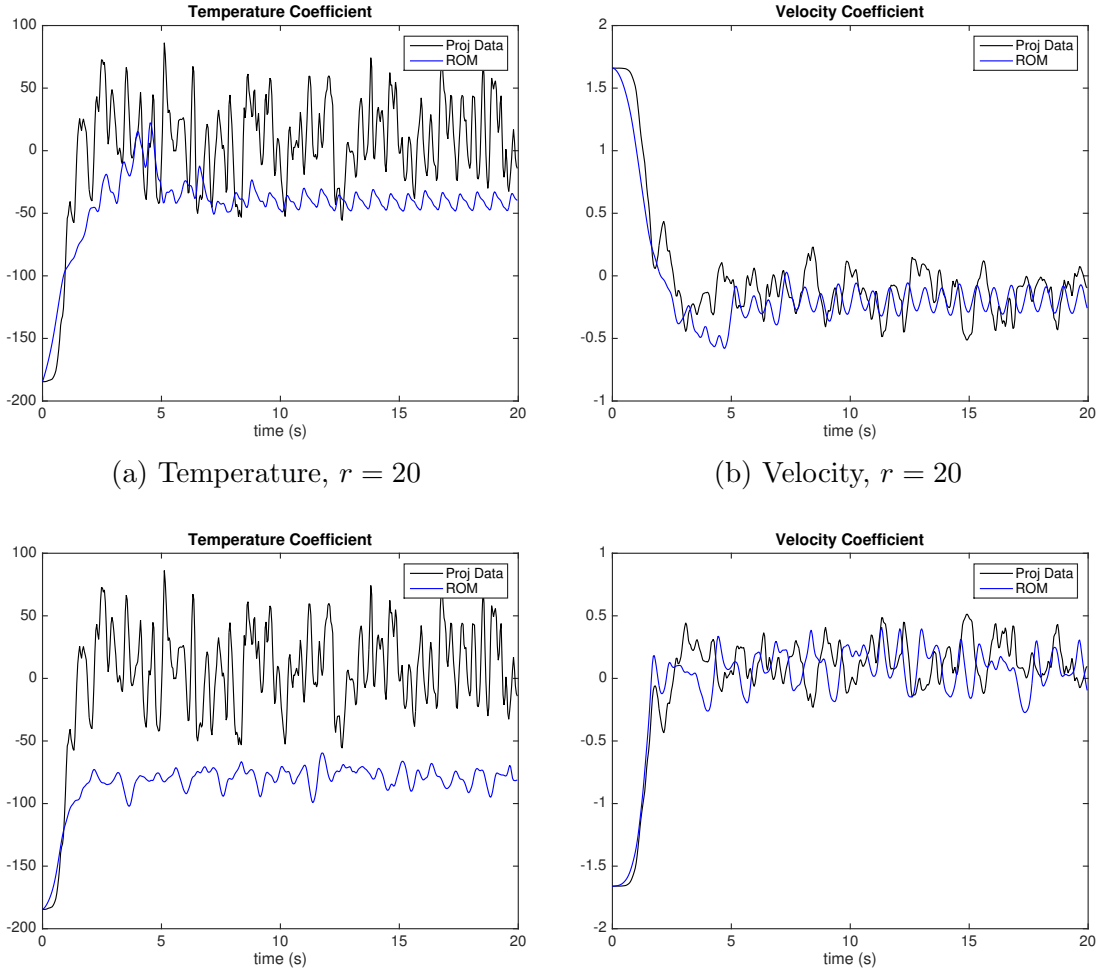
(a) Maximum $\widetilde{\mathbf{u}}(t)$ (b) Mean $\widetilde{\mathbf{u}}(t)$

(c) Maximum $\widetilde{\mathbf{v}}(t)$ (d) Mean $\widetilde{\mathbf{v}}(t)$

Figure 5.12: Mean and maximum velocity comparison between ROM and FOM with $r = 20$.

ODE increased as the size of the ROM was increased. What was not as obvious was that the computation times were affected by changing the diffusion parameters. This was not entirely predictable as it depended on how quickly the coefficients were changing for the given model. However, there was a general trend that when the diffusion constants were too large for the ROM, the model tended towards a steady state solution and thus required less time to solve. Table 5.5 shows the typical solution times for each size model. Again, these did vary some from one simulation to the next, but were generally close to the values given in Table 5.5.

We will now examine the ROM with $r = 20$ using the criteria set forth in Remark 5.1. We



(a) Temperature, $r = 20$

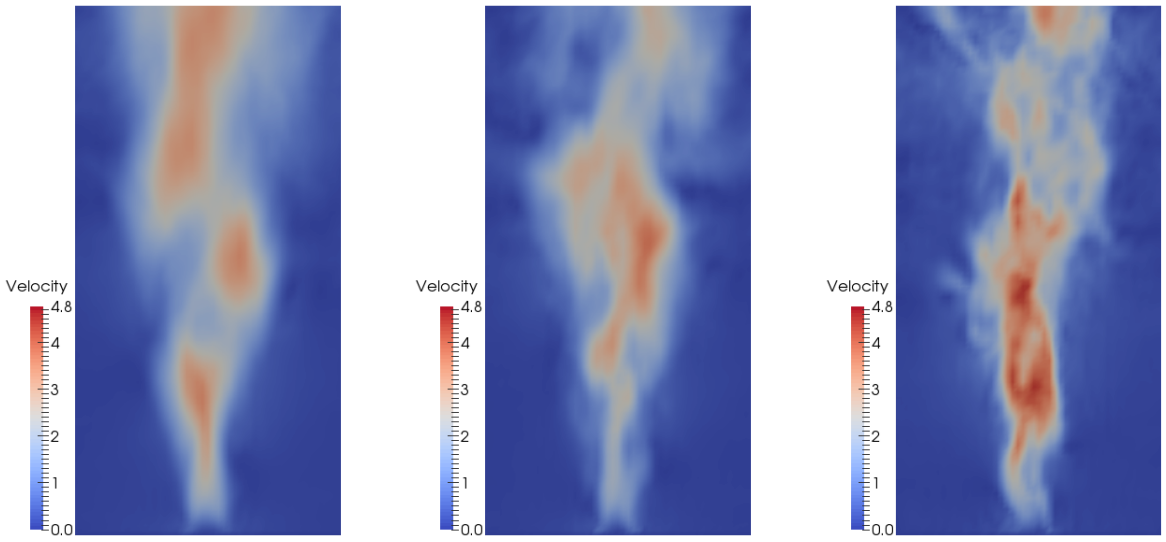
(b) Velocity, $r = 20$

(c) Temperature, $r = 50$

(d) Velocity, $r = 50$

Figure 5.13: POD coefficients for temperature and velocity.

will focus on Figures 5.11 and 5.12 as we look at our criteria. The first thing to note is that the ROM seems to do a good job of capturing the dynamics of the system. While not exact, we do see the types of behavior seen in a fire as evidenced by the mean and maximum temperatures and velocities. The model was not quite as good at approximating the mean or maximum of the temperature or velocity. This result is most likely due to the contribution of combustion in the FOM. Finally, the frequency of the temperature and velocity oscillations matches the FOM very well. However, the amplitude of the oscillations was not as large as the FOM. Overall, the ROM does a good job of capturing the fire dynamics, but to truly



(a) POD ROM $r = 20$ (b) POD ROM $r = 50$ (c) Full-Order Model

Figure 5.14: Comparison of the velocity profiles for the POD ROMs and the FOM.

match the magnitude of the fire, the combustion aspect will need to be incorporated.

Finally, we compare some of the aspects of the $r = 20$ ROM to the $r = 50$ ROM. As Figure 5.14 shows, the more modes the finer the details in the ROM. This behavior is to be expected, based on Figures 5.10 and 5.9 where with the higher modes, more of the finer details are included. However, for the larger ROM, it was not as successful in capturing the magnitude or the oscillation behavior. This is likely due to overcompensation for the higher modes. It could also be due to a greater effect being realized due the absence of the combustion term.

5.3 Wildland Fires

Large-scale models for real-time simulations, as required for predicting wildland fires, are avoided due to limited computational resources. On the other hand, lower spatial resolution limits the physics that can be captured by the models. In our research, a reduced-order

model (ROM) technique that retains the underlying physics was employed to predict the solution to the advection-reaction-diffusion equation, a common component in fire models. This technique also projects the nonlinearity, resulting in ROMs that *decrease the on-line computational cost by 2-3 orders of magnitude*. This ROM technique can be applied in other large-scale fire applications, such as ventilation for mine fires. The accuracy and improved computational efficiency are demonstrated by building a ROM for a wildland fire spread model.

5.3.1 Basic Description

Proper orthogonal decomposition (POD) is the most commonly used technique to produce reduced-order models for complex nonlinear dynamics. This technique has been applied to wildland fire models (e.g. [71, 72]) in [49, 85], where their ROMs exhibited an order of magnitude speedup in computational time while maintaining a high degree of accuracy. Though ROMs using only POD are effective, they can be significantly improved by addressing the so-called *lifting bottleneck*. The lifting bottleneck occurs when computing the reduced nonlinear term, since standard POD first lifts the reduced variables up to the full-order dimension, evaluates the nonlinear term, then projects the result back down to reduced dimension. Since nonlinear terms are computed at full-order dimension, the computational gains compared to the original model are limited. To improve the performance, the discrete empirical interpolation method (DEIM) [27, 28] is used to perform an additional projection on the nonlinear function so that the nonlinearity is calculated on this new projected space instead of the full-order space. A basic description of DEIM and how it was used in this study is given in Section 5.3.2. We apply this technique to simulate advection-reaction-diffusion equations that can be represented as

$$\frac{\partial \theta}{\partial t} = \kappa \nabla^2 \theta - \mathbf{v} \cdot \nabla \theta + \mathcal{S}(\theta) \tag{5.12}$$

where θ could be a concentration of chemical species or a temperature that transported throughout the domain. Typically, $\mathcal{S}(\theta)$ is a nonlinear function representing source and/or sink terms such as species production, species consumption, temperature increases due to reaction kinetics, etc. This type of equation is present in the phenomenological model given by equations (5.13-5.14) suggested in [71] to predict flame front propagation in wildland fires. The one-dimensional version of this coupled system was studied to assess the benefits of POD with DEIM over the standard POD approach.

$$\frac{\partial T}{\partial t} = \kappa \frac{\partial^2 T}{\partial x^2} - v \frac{\partial T}{\partial x} + \alpha \left(S e^{-\widetilde{\beta}/(T - T_0)} - \gamma (T - T_0) \right) \tag{5.13}$$

$$\frac{\partial S}{\partial t} = -\gamma_S S e^{-\widetilde{\beta}/(T - T_0)} \tag{5.14}$$

These equations model the temperature (T) and mass fraction of fuel (S) in a propagating fire. Using constant parameters and wind velocity, the nonlinearity of this model occurs via a reaction term.

5.3.2 Discrete Empirical Interpolation Method (DEIM)

Suppose we have a nonlinear system (2.17) where the state equation is given by

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{F}(\mathbf{x}(t)), \tag{5.15}$$

where $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{x} : \mathbb{R} \to \mathbb{R}^n$, and $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^n$. Suppose that $\mathbf{V} \in \mathbb{R}^{n\times r}$ is the projection matrix that we determine using POD. Then using a Galerkin projection we see that the ROM would be

$$\dot{\mathbf{x}}_r(t) = \underbrace{\mathbf{V}^T\mathbf{A}\mathbf{V}}_{\mathbf{A}_r:r\times r}\mathbf{x}_r(t) + \underbrace{\mathbf{V}^T}_{r\times n}\underbrace{\mathbf{F}(\mathbf{V}\mathbf{x}_r(t))}_{n\times 1}, \tag{5.16}$$

where $\mathbf{x}_r : \mathbb{R} \to \mathbb{R}^r$. As can be seen in (5.16), $\mathbf{A}_r$ can be precomputed and thus the linear term is strictly r-dimensional. This is not the case for the nonlinear term where $\mathbf{x}_r(t)$ must be lifted back to the original size of the state space, i.e. $\mathbf{V}\mathbf{x}_r(t) \in \mathbb{R}^n$, before evaluating $\mathbf{F}$. This implies that the computational complexity of calculating the nonlinear term is order n. Therefore potential gains for reducing the system to r dimensions are lost when computing the nonlinear term. DEIM is a method that we can use to help reduce this problem. In this section, we will cover an overview of DEIM and how that method is applied to the wildland fire model. We refer the reader to [27, 28, 35] for a more thorough coverage of DEIM and its associated error bounds.

We first define the nonlinear function from the ROM as $\mathbf{f}(t) := \mathbf{F}(\mathbf{V}\mathbf{x}_r(t))$ to simplify the notation. Now, the nonlinear term for the ROM can be defined as

$$\mathbf{N}(t) := \mathbf{V}^T\mathbf{F}(\mathbf{V}\mathbf{x}_r(t)) = \mathbf{V}^T\mathbf{f}(t). \tag{5.17}$$

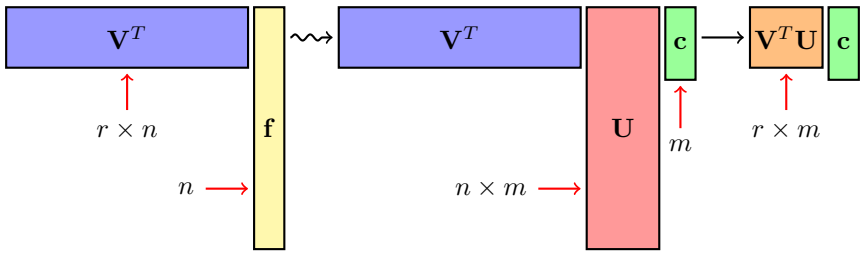


Figure 5.15: Visual depiction of the approximation $\mathbf{V}^T\mathbf{f}(t) \approx \mathbf{V}^T\mathbf{U}\mathbf{c}(t)$.

We then seek to approximate $\mathbf{f}(t)$ by finding a matrix $\mathbf{U} \in \mathbb{R}^{n\times m}$ where $m \ll n$ such that

$$\mathbf{f}(t) \approx \mathbf{U}\mathbf{c}(t), \tag{5.18}$$

where $\mathbf{c} : \mathbb{R} \to \mathbb{R}^m$. Applying this to (5.17), we now have the approximation to the nonlinear term given by

$$\underbrace{\mathbf{N}(t)}_{r} \approx \widetilde{\mathbf{N}}(t) = \underbrace{\mathbf{V}^T\mathbf{U}}_{r\times m}\underbrace{\mathbf{c}(t)}_{m}. \tag{5.19}$$

Figure 5.15 gives a visual depiction of the approximation. Here we also see that $\mathbf{V}^T\mathbf{U} \in \mathbb{R}^{r\times m}$ can be computed in advance, thus decreasing the overall computational complexity of $\widetilde{\mathbf{N}}(t)$ to be $\max\{r, m\} \ll n$. We can calculate $\mathbf{U}$ by evaluating the snapshots using the nonlinear function, $\mathbf{F}$, and then truncating the thin SVD (2.39) to m columns or basis functions. However, we still do not know how to create $\mathbf{c}(t)$ from the original nonlinear function $\mathbf{f}(t)$. This selection matrix is at the heart of the DEIM approximation. Using the method given in [35], we can create a selection matrix $\mathbf{P}$ that will select rows $\mathcal{P} := [\mathcal{P}_1, \ldots, \mathcal{P}_m]$ such that the following holds

$$\mathbf{P}^T\mathbf{f}(t) = \mathbf{P}^T\mathbf{U}\mathbf{c}(t). \tag{5.20}$$

In other words, borrowing from MATLAB® notation, $\mathbf{P}^T\mathbf{U} = \mathbf{U}(\mathcal{P}, :)$. Assuming that $\mathbf{P}^T\mathbf{U}$ is nonsingular, we obtain the following calculation $\mathbf{c}(t) = (\mathbf{P}^T\mathbf{U})^{-1}\mathbf{P}^T\mathbf{f}(t)$. We are now define our approximation for the nonlinear function $\mathbf{f}(t)$ as

$$\hat{\mathbf{f}}(t) = \mathbb{P}\mathbf{f}(t), \tag{5.21}$$

where $\mathbb{P}$ is the projector given by $\mathbb{P} = \mathbf{U}(\mathbf{P}^T\mathbf{U})^{-1}\mathbf{P}^T$. Using (5.21) and using the notation

in (5.16), we have the following approximation

$$\mathbf{F}(\mathbf{V}\mathbf{x}_r(t)) \approx \mathbf{U}(\mathbf{P}^T\mathbf{U})^{-1}\mathbf{P}^T\mathbf{F}(\mathbf{V}\mathbf{x}_r(t)) \tag{5.22}$$

From (5.22), we look specifically at the term $\mathbf{P}^T\mathbf{F}(\mathbf{V}\mathbf{x}_r(t)$. If $\mathbf{F}$ is a component-wise function, as it is in the wildland fire model, then we can move $\mathbf{P}^T$ into the function. This gives $\mathbf{F}(\mathbf{P}^T\mathbf{V}\mathbf{x}_r(t))$ which has computational complexity of $\mathcal{O}(m)$. Further, the matrix $\mathbf{P}^T\mathbf{V} \in \mathbb{R}^{m\times r}$ only has to be computed once. Returning to (5.16), the full nonlinear term is then approximated by

$$\mathbf{V}^T\mathbf{F}(\mathbf{V}\mathbf{x}_r(t)) \approx \mathbf{V}^T\mathbf{U}(\mathbf{P}^T\mathbf{U})^{-1}\mathbf{F}((\mathbf{P}^T\mathbf{V})\mathbf{x}_r(t)). \tag{5.23}$$

Define $\mathbf{E}_P := \mathbf{V}^T\mathbf{U}(\mathbf{P}^T\mathbf{U})^{-1}$ and $\mathbf{F}_P(t) := \mathbf{F}((\mathbf{P}^T\mathbf{V})\mathbf{x}_r(t))$, where $\mathbf{E}_P \in \mathbb{R}^{r\times m}$ only has to be computed once and $\mathbf{F}_P(t) : \mathbb{R} \to \mathbb{R}^m$.

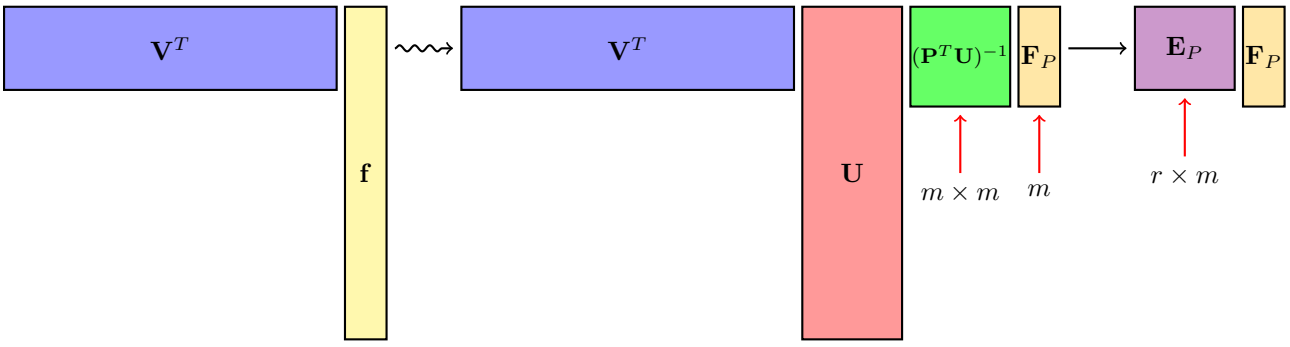


Figure 5.16: Visual depiction of the approximation $\mathbf{V}^T\mathbf{f}(t) \approx \mathbf{E}_P\mathbf{F}_P(t)$.

For the nonlinear term in the ROM (5.16), Figure 5.16 shows the full reduction using DEIM. We note that the complexity of this term is $\mathcal{O}(\max\{m, r\})$. We can now give the DEIM ROM definition as follows

Definition 5.3. Given (5.15), $\mathbf{V}$ determined using POD, and (5.19), we precompute $\mathbf{A}_r =$

$\mathbf{V}^T\mathbf{A}\mathbf{V}$, $\mathbf{V}_P = \mathbf{P}^T\mathbf{V}$, and $\mathbf{E}_P := \mathbf{V}^T\mathbf{U}(\mathbf{P}^T\mathbf{U})^{-1}$. Then the DEIM ROM is given by

$$\dot{\mathbf{x}}_r(t) = \mathbf{V}^T\mathbf{A}\mathbf{V}\mathbf{x}_r(t) + \mathbf{E}_P\mathbf{F}(\mathbf{V}_P\mathbf{x}_r(t)), \tag{5.24}$$

For the wildland fire-spread model (5.13)-(5.14), we can discretize the system using finite differences to create the following discretized model

$$\begin{bmatrix} \dot{\mathbf{T}}(t) \\ \dot{\mathbf{S}}(t) \end{bmatrix} = \begin{bmatrix} \mathbf{A}_T & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{T}(t) \\ \mathbf{S}(t) \end{bmatrix} + \begin{bmatrix} \alpha\mathbf{F}\left[\mathbf{T}(t), \mathbf{S}(t)\right] \\ -\gamma_S\mathbf{F}\left[\mathbf{T}(t), \mathbf{S}(t)\right] \end{bmatrix}, \tag{5.25}$$

where $\mathbf{F}\left[\mathbf{T}(t), \mathbf{S}(t)\right] = \mathbf{S}(t)e^{-\beta/(\mathbf{T}(t)-T_0)}$. For standard POD, an orthonormal basis is built for $\mathbf{T}$ and $\mathbf{S}$, given by $\mathbf{V}_T \in \mathbb{R}^{n\times r}$ and $\mathbf{V}_S \in \mathbb{R}^{n\times p}$ respectively. Let $q = r + p$ and $N = 2n$. We define the following

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_T & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \qquad \mathbf{V} = \begin{bmatrix} \mathbf{V}_T & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_S \end{bmatrix} \tag{5.26}$$

$$\Theta(t) = \begin{bmatrix} \mathbf{T}(t) \\ \mathbf{S}(t) \end{bmatrix} \qquad \widetilde{\mathbf{F}}(\Theta(t)) = \begin{bmatrix} \alpha\mathbf{F}\left[\mathbf{T}(t), \mathbf{S}(t)\right] \\ -\gamma_S\mathbf{F}\left[\mathbf{T}(t), \mathbf{S}(t)\right] \end{bmatrix}, \tag{5.27}$$

where $\Theta(t) \in \mathbb{R}^N$, $\mathbf{V} \in \mathbb{R}^{q\times N}$, and $\widetilde{\mathbf{F}} : \mathbb{R}^N \to \mathbb{R}^N$. We note that $\mathbf{V}^T\mathbf{V} = \mathbf{I}_q$. Projecting (5.25) using the definitions in (5.26)-(5.27) results in following POD ROM

$$\dot{\Theta}_q(t) = \mathbf{A}_q\Theta_q(t) + \mathbf{V}^T\widetilde{\mathbf{F}}(\mathbf{V}\Theta_q(t)), \tag{5.28}$$

where $\mathbf{A}_q = \mathbf{V}^T\mathbf{A}\mathbf{V} \in \mathbb{R}^{q\times q}$, $\Theta_q(t) \in \mathbb{R}^q$ and $\Theta(t) \approx \mathbf{V}\Theta_q(t)$. Now we need to apply DEIM to our model. We first note that the nonlinear function for $\mathbf{T}$ and $\mathbf{S}$ are just scalar multiples of each other. Therefore, our DEIM projection matrix $\mathbf{U}$ and selection matrix $\mathbf{P}$ will be the

same for both $\mathbf{T}$ and $\mathbf{S}$. So we will first determine $\mathbf{U} \in \mathbb{R}^{n\times m}$ and $\mathbf{P} \in \mathbb{R}^{n\times m}$ as described above using just the nonlinear function $\mathbf{F}\left[\mathbf{T}(t), \mathbf{S}(t)\right]$ and then we will bring the constants back into the system when we calculate out DEIM projection matrices. To obtain the full nonlinear DEIM projection (5.23), we define

$$\mathbf{E}_F = \begin{bmatrix} \mathbf{E}_T & \mathbf{0} \\ \mathbf{0} & \mathbf{E}_S \end{bmatrix} \qquad \mathbf{P}_F = \begin{bmatrix} \mathbf{P} & \mathbf{0} \\ \mathbf{0} & \mathbf{P} \end{bmatrix}, \tag{5.29}$$

where $\mathbf{E}_T = \alpha \mathbf{V}_T \mathbf{U}(\mathbf{P}^T\mathbf{U})^{-1}$ and $\mathbf{E}_S = -\gamma \mathbf{V}_S \mathbf{U}(\mathbf{P}^T\mathbf{U})^{-1}$. Further, we can let $\Phi = (\mathbf{P}^T\mathbf{V})$. This gives us the full nonlinear projection approximated as $\mathbf{V}^T\widetilde{\mathbf{F}}(\mathbf{V}\Theta(t) \approx \mathbf{E}_F\widetilde{\mathbf{F}}\left(\Phi\Theta(t)\right)$ which has a computational complexity of $\mathcal{O}(\max\{2m, q\})$. Applying this approximation and the previously defined matrices to (5.28) results in the POD/DEIM ROM ODE used to solve the system

$$\dot{\Theta}_q(t) = \mathbf{A}_q\Theta_q(t) + \mathbf{E}_F\widetilde{\mathbf{F}}\left(\Phi\Theta_q(t)\right). \tag{5.30}$$

5.3.3 Methods and Numerical Results

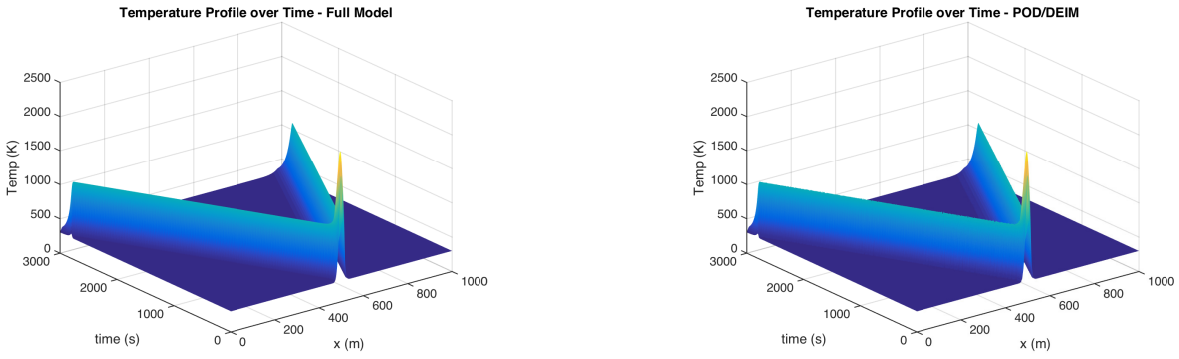


Figure 5.17: FOM versus the POD/DEIM ROM where $r_T = 250, r_S = 150$, and $r_{DEIM} = 250$.

For our testing we used the parameter values specified in Table 5.6, and the system was

Table 5.6: Parameter values for the wildland fire spread model given in [71].

Parameter	Value
κ	2.1360e-01
α	1.8793e02
$\widetilde{\beta}$	5.5849e02
γ	4.8372e-05
γ_S	1.6250e-01
T_0	300 K
v	0.00 m/s

discretized from zero to a 1000 m in 0.2 m increments and solved over 3000 s. The initial condition has a fire at the 500 m location. The fire then propagates across the domain towards both boundaries based on equations (5.13-5.14) as seen in Figure 5.17.

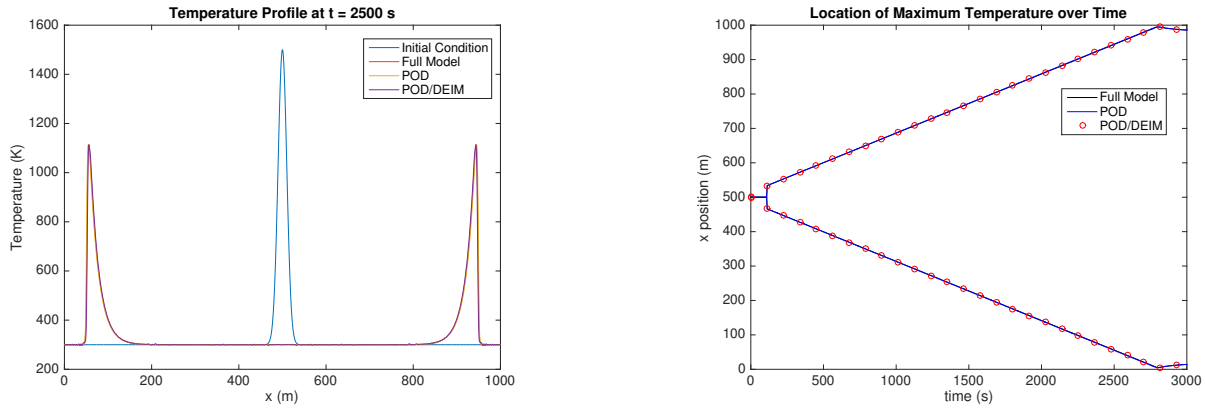


Figure 5.18: Fire spread for FOM, POD, and POD/DEIM.

Using the data snapshots created, r_T and r_S number of POD bases for T and S, respectively, were created. Further, POD was enhanced by projecting the nonlinearity using 250 DEIM vectors. Figures 5.17 and 5.18 show that when $r_T = 250$ and $r_S = 150$, the POD/DEIM ROM provides an excellent approximation of the full-order model (FOM) matching both the flame front location and temperature profile quite well.

As shown in Table 5.7, the solution times increase when more POD/DEIM vectors are used, but even the largest ROM using POD only was five times faster than the FOM. Further the

Table 5.7: Results for the ROM. Solution time for the FOM was 99.1 s

# POD Vectors	POD			POD/DEIM		
$r_T/r_S/r_{DEIM}$	Time (s)	Speed up	Rel Error	Time (s)	Speed up	Rel Error
70/35/250	1.33	74.6	1.7471e-02	0.13	760.1	1.7594e-02
200/150/250	15.4	6.38	4.3329e-03	0.72	137.6	5.2877e-04
200/200/250	18.5	5.31	1.8478e-02	0.84	118.2	1.3873e-02

relative error between the full-order and reduced-order model solutions was less 2% in all cases with a minimum of 0.43% when using 200 POD vectors for T and 150 POD vectors for S. When using POD with DEIM, the solution times were significantly better than POD alone while maintaining essentially the same error. The results demonstrate that using POD with DEIM can reduce the computational time by 2-3 orders of magnitude while retaining the physics and prediction accuracy.

5.4 Summary

We presented three scenarios in this chapter along with the methodology used to improve the performance of the numerical computation. These examples lead us towards providing reduced-order models for fully-realized fire models that include all of the fundamental physics involved.

For our first example, we showed how we could use POD to project the airflow model inside a coal mine. Our technique allows us to create and store a reduced-order model that can then be recalled to simulate nearby starting airflows. There is a potential to embed these models into network model at critical nodes. By having a stored ROM, the network model can conceivably be set to call the ROM to provide more detailed information at a given node.

Next, we created a ROM for a plume fire using POD. In this example, we put forth a set

of criteria to use when judging the quality of a ROM for a fire. Further, we note that even without the combustion being specifically modeled, the ROM provides a good representation of the full model.

Finally, we looked at the fire spread of a wildland fire. There have been previous attempts to reduce this model using POD [49, 85], but due to the Arrhenius kinetics associated with the full model, the reduced-order models were only able to realize modest gains in numerical computation times. We explained the DEIM technique and used it on this model. Our results show a 2-3 order of magnitude reduction in computation time while maintaining the fidelity of the POD. These results represented a significant improvement to existing reduced-order models generated using POD alone.

Chapter 6

Analysis of Discrete Time Model Reduction

The typical method for solving a time dependent partial differential equation (PDE) begins with discretizing the spatial domain and then using a method, e.g. finite elements, finite differences, or finite volumes, to create an N-dimensional system of ordinary differential equations (FOM ODE). The size of N depends on the density of the spatial mesh and the number of parameters and routinely becomes very large. This large FOM ODE is then discretized in time using an ODE solver and results in a fully discretized matrix system FOM OΔE where the solution at each time step is dependent on some number of solutions at previous time steps and the corresponding derivatives at those points.

Since these large problems can become computationally intractable to solve in the time scales necessary for the practical applications needing the solutions, we often need to employ some method to simplify the problem. As discussed before, we turn to model reduction in order to produce surrogate models that are computationally tractable and retain the accuracy required for a particular application. Carlberg et al. [26] investigated the preservation of POD

state-space optimality properties when using general multi-step and Runge-Kutta methods to discretize the ODE systems. In particular, for these systems, [26] looks to preserve optimality of the data in a least-squares sense, as was discussed in Section 2.3.3. For our research, we extend this analysis for linear systems, and investigate optimality in terms of input-independent, transfer-function based optimality conditions.

Typically, we apply input-independent model-reduction techniques, such as IRKA, to the FOM ODE to create a reduced-order ODE model (ROM ODE). This ROM ODE is then discretized in time to create the ROM$_r$ OΔE. However, we could first discretize the FOM ODE to obtain the FOM OΔE, and then reduce the FOM OΔE to produce a ROM$_d$ OΔE. Figure 6.1 shows a graphical representation of these different paths. In this chapter, we will limit our discussion to $\mathcal{H}_2$-optimal methods for reducing the FOM ODE or the FOM OΔE. Again, we refer the reader to [26] and the references therein for general nonlinear systems.

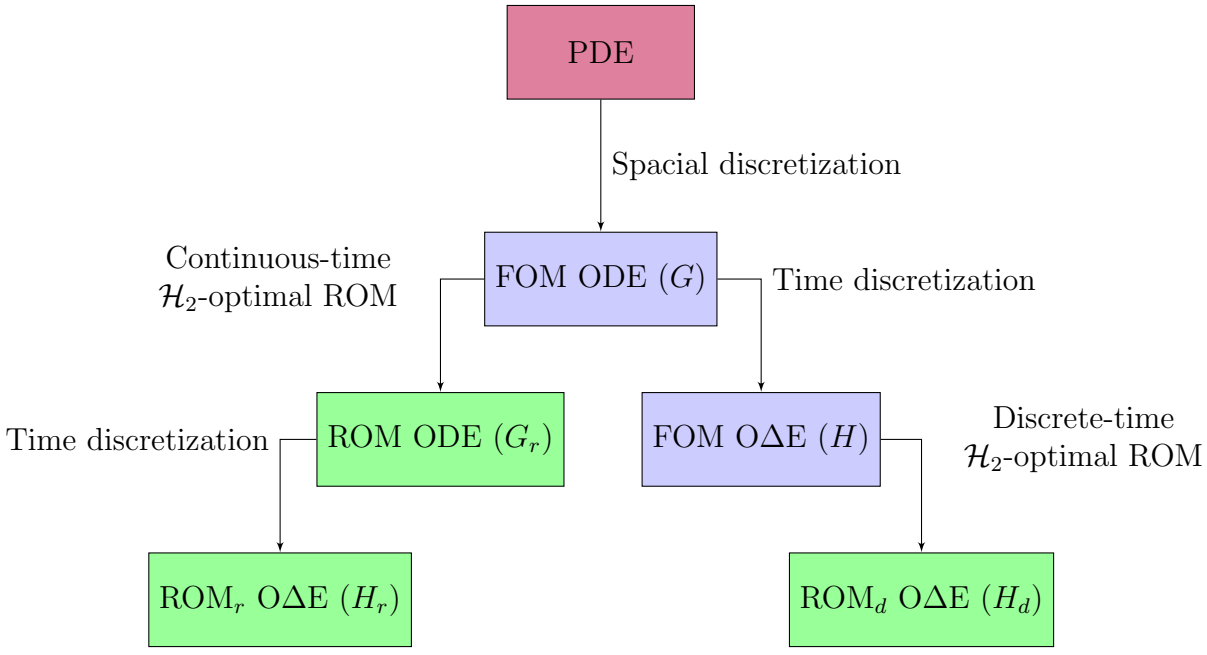


Figure 6.1: Comparison of space and time discretized reduced-order models.

The first section in this chapter discusses the $\mathcal{H}_2$ optimality conditions for continuous and

discrete time systems. While we have discussed continuous-time systems previously, we present it here to parallel the discussion of the discrete-time systems for the particular dynamical systems on which we are focusing. We then present the connections between the continuous time systems and discrete time systems for a certain class of ODE solvers. In Section 6.3, we discuss the relationship between the associated transfer functions of the continuous and discrete time systems. We show that there exists a transformation that we can use to connect the continuous-time system to the discrete-time system. We show that this transformation depends on the method used for the time discretization. We look, in Section 6.4, at whether or not the $\mathcal{H}_2$ optimality conditions are preserved after applying the time discretization. We conclude the chapter with a few numerical examples.

6.1 Optimality Conditions for the ROM

Many of the concepts for continuous-time linear time-invariant systems were covered in Section 2.3.2. We revisit them in here to investigate the conditions necessary to provide a local minimum for the $\mathcal{H}_2$-optimality problem. In addition, we will examine discrete-time systems through the lens of $\mathcal{H}_2$-optimality with a focus on satisfying the first-order necessary conditions of the $\mathcal{H}_2$-optimality problem.

6.1.1 Continuous-time Systems

Let the following be the continuous-time linear time-invariant SISO dynamical system

$$G : \begin{cases} \dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{b}u(t), \\ y(t) = \mathbf{c}^T\mathbf{x}(t), \end{cases} \tag{6.1}$$

where $\mathbf{x} \in \mathbb{R}^N$, $\mathbf{A} \in \mathbb{R}^{N \times N}$, $\mathbf{b}, \mathbf{c} \in \mathbb{R}^N$, and $y(t), u(t)$ are scalar valued functions of t. We will refer to this as the ODE FOM (ordinary differential equation full-order model). These systems often arise from the spatial discretization of a PDE using, for example, finite elements or finite differences. By taking the Laplace transform of this continuous-time system we can map the input to the output via the transfer function $G(s)$ as follows

$$\tilde{y}(s) = G(s)\tilde{u}(s) \tag{6.2}$$

where $\tilde{u}(s), \tilde{y}(s)$ are the Laplace transforms of the input and output functions, respectively. Additionally, we define the transfer function $G(s)$ by

$$G(s) = \mathbf{c}^T(sI - A)^{-1}\mathbf{b} \tag{6.3}$$

Often times, for these models, the dimension, N, is so large that it becomes impossible to solve the problem in time constraints needed. Therefore we desire to create a reduced-order model that closely approximates the full-order model. We create projection matrices $\mathbf{V}, \mathbf{W} \in \mathbb{R}^{N \times r}$ where $r \ll N$. Using these we create the following reduced-order model

$$G_r : \begin{cases} \dot{\mathbf{x}}_r(t) = \mathbf{A}_r\mathbf{x}_r(t) + \mathbf{b}_r u(t), \\ y_r(t) = \mathbf{c}_r^T\mathbf{x}_r(t), \end{cases} \tag{6.4}$$

where $\mathbf{A}_r = \mathbf{W}^T\mathbf{A}\mathbf{V}$, $\mathbf{b}_r = \mathbf{W}^T\mathbf{b}$, and $\mathbf{c}_r^T = \mathbf{c}^T\mathbf{V}$. Similarly, we have the following input to output mapping

$$\tilde{y}_r(s) = G_r(s)\tilde{u}(s), \tag{6.5}$$

$$= \left[\mathbf{c}_r^T(s\mathbf{I} - \mathbf{A}_r)^{-1}\mathbf{b}_r\right]\tilde{u}(s). \tag{6.6}$$

So, we now look at the error between the full and reduced-order models, $\tilde{y}_{err}(s)$, in the frequency domain.

$$\begin{aligned}\tilde{y}_{err}(s) &= \tilde{y}(s) - \tilde{y}_r(s) && (6.7)\\ &= G(s)\tilde{u}(s) - G_r(s)\tilde{u}(s) && (6.8)\\ &= \left[G(s) - G_r(s)\right]\tilde{u}(s) && (6.9)\\ &= \left[\mathbf{c}^T(s\mathbf{I}-\mathbf{A})^{-1}\mathbf{b} - \mathbf{c}_r^T(sI-\mathbf{A}_r)^{-1}\mathbf{b}_r\right]\tilde{u}(s) && (6.10)\end{aligned}$$

So if we would like to minimize the error between the output of the full and reduced-order models we keep $\|y-y_r\|$ small in an appropriate norm by ensuring that the transfer functions are close, again in an appropriate norm. In particular Gugercin et al. [53] show that

$$\|y(t) - y_r(t)\|_{L^\infty} = \max_{t>0} |y(t) - y_r(t)| \leq \|G - G_r\|_{\mathcal{H}_2} \|u(t)\|_{L^2}\,, \tag{6.11}$$

where the error in the $\mathcal{H}_2$ norm is given by (6.12).

$$\left\|G - \widetilde{G}\right\|_{\mathcal{H}_2} = \left(\frac{1}{2\pi}\int_{-\infty}^{\infty} \left|G(i\omega) - \widetilde{G}(i\omega)\right|^2 d\omega\right)^{1/2}. \tag{6.12}$$

In other words we would like to solve the minimization problem

$$G_r(s) = \arg\min_{\dim(\widetilde{G}_r)=r} \left\|G - \widetilde{G}_r\right\|_{\mathcal{H}_2}, \tag{6.13}$$

where $\widetilde{G}_r$ is stable. As is detailed [53], solving this minimization problem also minimizes $\|y - y_r\|_{L^\infty}$ and thus provides the optimal reduced-order model (ROM) of dimension r provided we can use this fact to determine $\mathbf{V}$ and $\mathbf{W}$ that generates this ROM. We note here that it is difficult to find a global minimum for this problem. Therefore, we seek a local minimum that satisfies the first-order necessary conditions for optimality. Now we tie these

together with the following theorem and lemma.

Lemma 6.1 (**Corollary 2.2 from [53]**). *Consider the system in (6.1) defined by* $\mathbf{A}, \mathbf{b}, \mathbf{c}$*, a set of distinct shifts given by* $\{\sigma_k\}_{k=1}^r$*, that is closed under conjugation, and subspaces spanned by the columns of* $\mathbf{V}_r$ *and* $\mathbf{W}_r$ *with*

$$\text{Ran}(\mathbf{V}) = \text{span}\{(\sigma_1\mathbf{I} - \mathbf{A})^{-1}\mathbf{b} \cdots (\sigma_r\mathbf{I} - \mathbf{A})^{-1}\mathbf{b}\}, \tag{6.14}$$

$$\text{Ran}(\mathbf{W}) = \text{span}\{(\sigma_1\mathbf{I} - \mathbf{A})^{-T}\mathbf{c} \cdots (\sigma_r\mathbf{I} - \mathbf{A})^{-T}\mathbf{c}.\}. \tag{6.15}$$

Then $\mathbf{V}_r$ *and* $\mathbf{W}_r$ *can be chosen to be real matrices with* $\mathbf{W}^T\mathbf{V} = \mathbf{I}$*, and the reduced-order system in (6.4) defined by* $\mathbf{A}_r = \mathbf{W}_r^T\mathbf{A}\mathbf{V}_r$*,* $\mathbf{b}_r = \mathbf{W}_r^T\mathbf{b}$*, and* $\mathbf{c}_r^T = \mathbf{c}^T\mathbf{V}_r$ *is itself real and matches the first two moments of* $G(s)$ *at each of the interpolation points* σ_k*, i.e.* $G(\sigma_k) = G_r(\sigma_k)$ *and* $G'(\sigma_k) = G_r'(\sigma_k)$ *for* $k = 1, \ldots, r$*.*

Proof. See [34, 46, 53, 93, 94] □

Theorem 6.2 (**SISO: Meier III and Luenberger [76]**). *Given the transfer function* $G(s) = \mathbf{c}^T(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{b}$*, the optimal* $\mathcal{H}_2$ *reduced-order transfer function* $G_r(s) = \mathbf{c}_r^T(s\mathbf{I} - \mathbf{A}_r)^{-1}\mathbf{b}_r$ *satisfies*

$$G(-\lambda_k) = G_r(-\lambda_k),$$

$$G'(-\lambda_k) = G_r'(-\lambda_k),$$

where λ_k*,* $k = 1, \cdots, r$*, are the eigenvalues of* $\mathbf{A}_r$*.*

Proof. See Meier III and Luenberger [76]. □

6.1.2 Discrete-time Systems

We consider the linear time-invariant discrete-time dynamical system defined by the following set of difference equations

$$H : \begin{cases} \mathbf{x}^{k+1} = \widetilde{\mathbf{A}}\mathbf{x}^k + \widetilde{\mathbf{b}}u^k, \\ y^k = \widetilde{\mathbf{c}}^T\mathbf{x}^k, \end{cases} \tag{6.16}$$

where $\widetilde{\mathbf{A}} \in \mathbb{R}^{N\times N}$ and $\widetilde{\mathbf{b}}, \widetilde{\mathbf{c}} \in \mathbb{R}^N$. We can consider $\mathbf{x}^k \in \mathbb{R}^N$ to be the state at the discrete time step t^k where the time $[0, t_f]$ is discretized as $\{0 = t^0, t^1, \ldots, t^k, t^{k+1}, \ldots, t_f\}$, and $u^k, y^k \in \mathbb{R}$ are the input and output, respectively, at time t^k. Applying a Z-transform to H, we obtain an input to output mapping in the frequency domain as

$$\check{y}(z) = H(z)\check{u}(s) = \widetilde{\mathbf{c}}^T(z\mathbf{I} - \widetilde{\mathbf{A}})^{-1}\widetilde{\mathbf{b}}\check{u}(z) \tag{6.17}$$

where $\check{u}(z)$ and $\check{y}(z)$ are the Z-transforms of the input and output functions respectively and $H(z)$ is the discrete-time transfer function. Similarly to before, we can construct a reduced-order model via projection matrices, $\widetilde{\mathbf{V}}, \widetilde{\mathbf{W}} \in \mathbb{R}^{N\times r}$, such that

$$H_d : \begin{cases} \mathbf{x}_d^{k+1} = \widetilde{\mathbf{A}}_d\mathbf{x}_d^k + \widetilde{\mathbf{b}}_d u^k, \\ y_d^k = \widetilde{\mathbf{c}}_d^T\mathbf{x}_d^k, \end{cases} \tag{6.18}$$

with $\widetilde{\mathbf{A}}_d = \widetilde{\mathbf{W}}^T\widetilde{\mathbf{A}}\widetilde{\mathbf{V}}$, $\widetilde{\mathbf{b}}_d = \widetilde{\mathbf{W}}^T\widetilde{\mathbf{b}}$, and $\widetilde{\mathbf{c}}_d = \widetilde{\mathbf{c}}^T\widetilde{\mathbf{V}}$. Then the reduced-order transfer function is given by

$$H_d(z) = \widetilde{\mathbf{c}}_d^T(z\mathbf{I} - \widetilde{\mathbf{A}}_d)^{-1}\widetilde{\mathbf{b}}_d. \tag{6.19}$$

Similar to the continuous-time case, we would like to find a reduced-order model H_r, that satisfies the first-order necessary conditions for $\mathcal{H}_2$ optimality. In order to do so, we first define the $\mathcal{H}_2$-norm for a discrete-time system.

Definition 6.3. The $\mathcal{H}_2$-norm for SISO discrete-time system H is given by

$$\|H\|_{\mathcal{H}_2} = \left(\frac{1}{2\pi} \int_0^{2\pi} \overline{H(e^{i\theta})} H(e^{i\theta}) \, d\theta \right)^{\frac{1}{2}} \tag{6.20}$$

As is shown in [22, 53], the conditions are slightly different for the discrete time case. Here we need to interpolate the transfer function at the shifts $\sigma_k = 1/\mu_k$ where μ_k are the poles of $H_d(z)$. If we assume that the poles of $H_d(z)$ are closed under conjugation then $1/\mu_k$ are the mirror images of the poles of $H_d(z)$ with respect to the unit disk. We now present a couple theorems that summarize this result.

Theorem 6.4 (First-order Necessary Conditions for $\mathcal{H}_2$ Optimality of a Discrete–Time ROM (Version 1)). *Given the transfer function $H(z) = \widetilde{\mathbf{c}}^T(s\mathbf{I} - \widetilde{\mathbf{A}})^{-1}\widetilde{\mathbf{b}}$ for the discrete-time system H (6.16), the transfer function $H_d(z) = \widetilde{\mathbf{c}}_d^T(s\mathbf{I} - \widetilde{\mathbf{A}}_d)^{-1}\widetilde{\mathbf{b}}_d$ satisfies the first-order necessary conditions for $\mathcal{H}_2$ optimality in the norm given by Definition 6.3 when*

$$H(1/\mu_k) = H_d(1/\mu_k), \tag{6.21}$$

$$H'(1/\mu_k) = H_d'(1/\mu_k), \tag{6.22}$$

where μ_k, $k = 1, \ldots, r$, are the eigenvalues of $\widetilde{\mathbf{A}}_d$, closed under conjugation.

Proof. See [22, 53]. □

It important to note here that the first-order necessary conditions given in Theorem 6.4 apply to discrete time systems with no d term. For systems with a d term, our transfer function becomes

$$\widetilde{H}(z) = \mathbf{c}^T(z\mathbf{I} - \mathbf{A})^{-1}\mathbf{b} + d. \tag{6.23}$$

In order for the reduced-order model to be optimal for these types of systems, we need to also

match the d terms. Gaier [41] has already showed this result for the case of fixed poles; i.e., the only unknowns are the residues of the transfer function. In [41], the poles are assumed outside the unit disc and the interpolation points are inside the disc. Of course, as expected, this result can be directly applied in our setting as well. To make this chapter self-contained, we include this proof here as a corollary to Gaier's [41] result.

Corollary 6.5 (First-order Necessary Conditions for $\mathcal{H}_2$ Optimality of a Discrete–Time ROM (Version 2)). *Given the n-dimensional transfer function $\widetilde{H}(z) = \widetilde{\mathbf{c}}^T(s\mathbf{I} - \widetilde{\mathbf{A}})^{-1}\widetilde{\mathbf{b}} + d$ for the full-order discrete-time system H (6.16), the transfer function $\widetilde{H}_d(z) = \widetilde{\mathbf{c}}_d^T(s\mathbf{I} - \widetilde{\mathbf{A}}_d)^{-1}\widetilde{\mathbf{b}}_d + d_d$ of dimension r, satisfies the first-order necessary conditions for $\mathcal{H}_2$ optimality in the norm given by Definition 6.3 when*

$$\widetilde{H}(1/\lambda_k) = \widetilde{H}_d(1/\lambda_k), \tag{6.24}$$

$$\widetilde{H}'(1/\lambda_k) = \widetilde{H}_d'(1/\lambda_k), \tag{6.25}$$

where λ_k, $k = 1, \cdots, r$, are the eigenvalues of $\widetilde{\mathbf{A}}_d$, closed under conjugation. Further, $d = d_d$ must also be true.

Proof. The conditions given by (6.24) and (6.25) have already been shown for the case where $d = 0$ in Theorem 6.4. So we focus on the case where $d \neq 0$. Using the pole-residue expansion, we write $\widetilde{H}_d(z)$ as

$$\widetilde{H}_d(z) = d_d + \sum_{k=1}^{r} \frac{\phi_k}{z - \mu_k}, \tag{6.26}$$

From this expansion, we can see that $\widetilde{H}(z)$ is generated by the basis

$$\left\{1, \frac{1}{z - \mu_1}, \cdots, \frac{1}{z - \mu_r}\right\}, \tag{6.27}$$

where the coefficient of the constant basis function, 1, is d_d and the coefficient for each basis element $\frac{1}{z-\mu_k}$ is ϕ_k. For $\mathcal{H}_2$ optimality, the error between $\widetilde{H}(z)$ and $\widetilde{H}_d(z)$ must be orthogonal to each of the basis vectors. So for the constant basis, 1, in (6.27) we need

$$\langle \widetilde{H} - \widetilde{H}_d, 1 \rangle_{\mathcal{H}_2} = 0, \tag{6.28}$$

where the $\mathcal{H}_2$ inner product is given by

$$\langle H, G \rangle_{\mathcal{H}_2} = \frac{1}{2\pi} \int_0^{2\pi} H(e^{i\theta}) \overline{G(e^{i\theta})}\, d\theta. \tag{6.29}$$

So now (6.28) becomes

$$\langle \widetilde{H} - \widetilde{H}_d, 1 \rangle_{\mathcal{H}_2} = 0, \tag{6.30}$$

$$\frac{1}{2\pi} \int_0^{2\pi} \widetilde{H}(e^{i\theta}) - \widetilde{H}_d(e^{i\theta})\, d\theta = 0 \tag{6.31}$$

$$\frac{1}{2\pi} \int_0^{2\pi} \widetilde{H}(e^{i\theta})\, d\theta = \frac{1}{2\pi} \int_0^{2\pi} \widetilde{H}_d(e^{i\theta})\, d\theta \tag{6.32}$$

We now do a change of variables by letting $z = e^{i\theta}$ and $dz = ie^{i\theta} d\theta = iz\, d\theta$. This gives

$$\frac{1}{2\pi i} \oint_\gamma \frac{1}{z} \widetilde{H}(z)\, dz = \frac{1}{2\pi i} \oint_\gamma \frac{1}{z} \widetilde{H}_d(z)\, dz, \tag{6.33}$$

where γ is a contour integral which, in our case, is the unit circle. Now recall the inverse Z-transform is given by

$$h[n] = \frac{1}{2\pi i} \oint_\gamma H(z) z^{n-1}\, dz, \tag{6.34}$$

where γ is a contour that encircles the poles of $H(z)$. From this, we see that both sides of

Equation (6.33) are simply the inverse Z-transforms at $n = 0$. Therefore,

$$\widetilde{h}[0] = \widetilde{h}_d[0]. \tag{6.35}$$

For discrete-time systems with $H(z) = \mathbf{c}^T(z\mathbf{I} - \mathbf{A})^{-1}\mathbf{b} + d$, the impulse response of $h[n]$ at $n = 0$ is the first Markov parameter and this is given by d. This implies that for $\mathcal{H}_2$ optimality we must have $d = d_d$ and the result is shown. □

Theorem 6.6. *Given shifts* $\{\sigma_k\}_{k=1}^r$ *and* $\widetilde{\mathbf{V}}$ *and* $\widetilde{\mathbf{W}}$ *chosen such that* $\widetilde{\mathbf{W}}^T\widetilde{\mathbf{V}} = \mathbf{I}$ *and*

$$\mathrm{Ran}(\widetilde{\mathbf{V}}) = \mathrm{span}\{(\sigma_1\mathbf{I} - \widetilde{\mathbf{A}})^{-1}\widetilde{\mathbf{b}} \cdots (\sigma_r\mathbf{I} - \widetilde{\mathbf{A}})^{-1}\widetilde{\mathbf{b}}\}, \tag{6.36}$$

$$\mathrm{Ran}(\widetilde{\mathbf{W}}) = \mathrm{span}\{(\sigma_1\mathbf{I} - \widetilde{\mathbf{A}})^{-T}\widetilde{\mathbf{c}} \cdots (\sigma_r\mathbf{I} - \widetilde{\mathbf{A}})^{-T}\widetilde{\mathbf{c}}.\}. \tag{6.37}$$

If the reduced-order model H_d *in* (6.18) *is produced using* $\widetilde{\mathbf{A}}_d = \widetilde{\mathbf{W}}^T\widetilde{\mathbf{A}}\widetilde{\mathbf{V}}$, $\widetilde{\mathbf{b}}_d = \widetilde{\mathbf{W}}^T\widetilde{\mathbf{b}}$, *and* $\widetilde{\mathbf{c}}_d^T = \widetilde{\mathbf{c}}^T\widetilde{\mathbf{V}}$, *and further if* $\sigma_k = 1/\mu_k$ *where the* μ_k *are the eigenvalues of* $\widetilde{\mathbf{A}}_d$, *closed under conjugation, then* H_d *satisfies the first-order necessary conditions for* $\mathcal{H}_2$ *optimality given in Theorem 6.4.*

Proof. See [22, 53]. □

6.2 Time Discretization of the ODE

We consider a system of N ODEs that was created using some spatial discretization (e.g finite elements or finite differences) of a time dependent PDE. Let this system can be represented as the linear time-invariant dynamical system G. To solve this system, we must also discretize the model in time. While there are several methods that can be used, we will focus on the linear multistep methods defined below.

Definition 6.7. A linear k-step method to solve the continuous time dynamical system G in (6.1) is given by

$$\sum_{j=0}^{k} \alpha_j \mathbf{x}^{n+k-j} = h \sum_{j=0}^{k} \beta_j (\mathbf{A}\mathbf{x}^{n+k-j} + \mathbf{b}u^{n-j}), \tag{6.38}$$

where $\mathbf{x}^n$ is the result at time t^n, $u^n = u(t^n)$, h is the time step, the coefficients α_j and β_j define the scheme, $\alpha_0 \neq 0$, and $\sum_{j=0}^{k} \alpha_j = 0$ and $\sum_{j=0}^{k-1} (k-j)\alpha_j = \sum_{j=0}^{k} \beta_j$ is necessary for consistency. Additionally, if $\beta_0 = 0$, the method is considered *explicit*. Otherwise, we say the method is *implicit*, [1, 42].

While not explicitly required, it is customary to set $\alpha_k = 1$ and then scale the other coefficients as necessary. Here, we refer to the time discretized ODE as H. Then H is solved by the system of algebraic equations at each time step $n = \mathbb{N}(T/h)$ where $T = [0, t_f]$ given by:

$$\mathbf{r}^n(\mathbf{z}^n) = 0, \tag{6.39}$$

where $\mathbf{z}^n \in \mathbb{R}^N$ is the unknown variable and $\mathbf{r}^n : \mathbb{R}^N \to \mathbb{R}^N$ denotes the multistep residual defined by

$$\mathbf{r}^n(\mathbf{z}) := \alpha_0 \mathbf{z} - h\beta_0(\mathbf{A}\mathbf{z} + \mathbf{b}u^n) + \sum_{j=1}^{k} \alpha_j \mathbf{x}^{n-j} - h \sum_{j=1}^{k} \beta_j (\mathbf{A}\mathbf{x}^{n-j} + \mathbf{b}u^{n-j}). \tag{6.40}$$

Then the state can be explicitly updated as $\mathbf{x}^n = \mathbf{z}^n$. If $\beta_0 \neq 0$, the method is considered implicit. We extend the results in [26] by showing that given any bi-orthogonal right and left projection matrices, $\mathbf{V}$ and $\mathbf{W}$, respectively, the Petrov-Galerkin projection and time discretization are commutative.

Theorem 6.8 (Petrov-Galerkin: commutivity of the projection and time discretization)**.** *Given right and left projection matrices $\mathbf{V}, \mathbf{W} \in \mathbb{R}^{N \times r}$ with $\mathbf{W}^T\mathbf{V} = \mathbf{I}$, then performing a Petrov-Galerkin projection on the governing ODE and subsequently applying a time*

discretization using a multistep method yields the same model as first applying the time discretization and then subsequently performing a Petrov-Galerkin projection.

Proof. We approximate $\mathbf{x}$ by $\mathbf{V}\mathbf{x}_r$ and then enforce the Petrov-Galerkin condition that the residual is orthogonal to $\mathbf{W}$ as follows

$$\mathbf{W}^T\left(\mathbf{V}\dot{\mathbf{x}}_r - \mathbf{A}\mathbf{V}\mathbf{x}_r - \mathbf{b}u\right) = 0 \tag{6.41}$$

From this, we obtain the reduced-order model G_r, possibly not $\mathcal{H}_2$-optimal,

$$\dot{\mathbf{x}}_r = \mathbf{W}^T\mathbf{A}\mathbf{V}\mathbf{x}_r + \mathbf{W}^T\mathbf{b}u. \tag{6.42}$$

Applying the multistep method to discretize G_r, we have the following residual

$$\mathbf{r}_r^n(\mathbf{z}_r) := \alpha_0\mathbf{z}_r - h\beta_0(\mathbf{W}^T\mathbf{A}\mathbf{V}\mathbf{z}_r + \mathbf{W}^T\mathbf{b}u^n) + \sum_{j=1}^{k}\alpha_j\mathbf{x}_r^{n-j} - h\sum_{j=1}^{k}\beta_j(\mathbf{W}^T\mathbf{A}\mathbf{V}\mathbf{x}^{n-j} + \mathbf{W}^T\mathbf{b}u^{n-j}). \tag{6.43}$$

where we solve $\mathbf{r}_r^n(\mathbf{z}_r^n) = 0$ and then let $\mathbf{x}_r^n = \mathbf{z}_r^n$. We now want to show that if we first apply the multistep method to obtain the full-order discrete-time system H and then apply the projection that, we will obtain the same result. Set $\mathbf{z} = \mathbf{V}\mathbf{z}_r$ and $\mathbf{x}^i = \mathbf{V}\mathbf{x}_r^i$ and then apply the multistep method. Again we want to enforce the Petrov-Galerkin condition which now gives $\mathbf{W}^T\mathbf{r}^n(\mathbf{V}\mathbf{z}_r) = 0$. Substituting, we obtain

$$\mathbf{W}^T\mathbf{r}^n(\mathbf{V}\mathbf{z}_r) = \mathbf{W}^T\left[\alpha_0\mathbf{V}\mathbf{z}_r - h\beta_0(\mathbf{A}\mathbf{V}\mathbf{z}_r + \mathbf{b}u^n) + \sum_{j=1}^{k}\alpha_j\mathbf{V}\mathbf{x}_r^{n-j} - h\sum_{j=1}^{k}\beta_j(\mathbf{A}\mathbf{V}\mathbf{x}^{n-j} + \mathbf{b}u^{n-j})\right]$$

$$\begin{aligned}\mathbf{W}^T\mathbf{r}^n(\mathbf{V}\mathbf{z}_r) = {}& \alpha_0\mathbf{z}_r - h\beta_0(\mathbf{W}^T\mathbf{A}\mathbf{V}\mathbf{z}_r + \mathbf{W}^T\mathbf{b}u^n) + \\ & \sum_{j=1}^{k}\alpha_j\mathbf{x}_r^{n-j} - h\sum_{j=1}^{k}\beta_j(\mathbf{W}^T\mathbf{A}\mathbf{V}\mathbf{x}^{n-j} + \mathbf{W}^T\mathbf{b}u^{n-j})\end{aligned} \tag{6.44}$$

We see that

$$\mathbf{r}_r^n(\mathbf{z}_r) = \mathbf{W}^T \mathbf{r}^n(\mathbf{V}\mathbf{z}_r) \tag{6.45}$$

at each time step. Therefore, the projection and time discretization commute. □

Suppose we let $\mathbf{W} = \mathbf{W}_r$ and $\mathbf{V} = \mathbf{V}_r$ be the $\mathcal{H}_2$-optimal projection matrices defined earlier by Theorem 6.2 and Lemma 6.1. Then we can see that while we have created an optimal reduced-order model, we cannot guarantee that we maintain that optimality once we have discretized the system in time. However, if we use the discrete-time projection criteria given in Theorem 6.6 on H, we can obtain an $\mathcal{H}_2$-optimal projection of the discrete-time FOM to the discrete-time ROM H_r.

6.2.1 Single-Step Discretization Schemes

For this research, we focused on three single-step time discretization schemes. We first define each method using the α and β coefficients for the general multistep method defined by 6.7. For each of the time discretization schemes, we then discuss how the eigenvalues of $\mathbf{A}$ and the step size, h affect the stability of the discrete system. We present the following definitions and theorems on stability to establish connections between the stability of the ODE and its associated time discretization.

Definition 6.9. A continuous-time system of ordinary differential equations (6.1) is said to be *asymptotically stable* if for every bounded initial condition and input function, i.e. $\|\mathbf{x}(0)\| < \infty$ and $\|u\|_{L^\infty} < \infty$ respectively, every solution $\widetilde{\mathbf{x}} \to 0$ as $t \to \infty$. Similarly, a discrete-time system is *asymptotically stable* if for every solution $\widetilde{\mathbf{x}}$, $\widetilde{\mathbf{x}}^n \to 0$ as $t \to \infty$ [1, 4, 44, 45, 78].

Theorem 6.10 (Asymptotic Stability of Continuous Systems)**.** *The system of ordinary*

differential equations

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) \quad \textit{with} \quad \mathbf{x}(0) = \mathbf{x}_0, \tag{6.46}$$

is asymptotically stable $t = [0, \infty)$*, if* $\Re(\lambda_k) < 0$ *for all* $\lambda_k \in \lambda(\mathbf{A})$*, the eigenvalues of* $\mathbf{A}$*.*

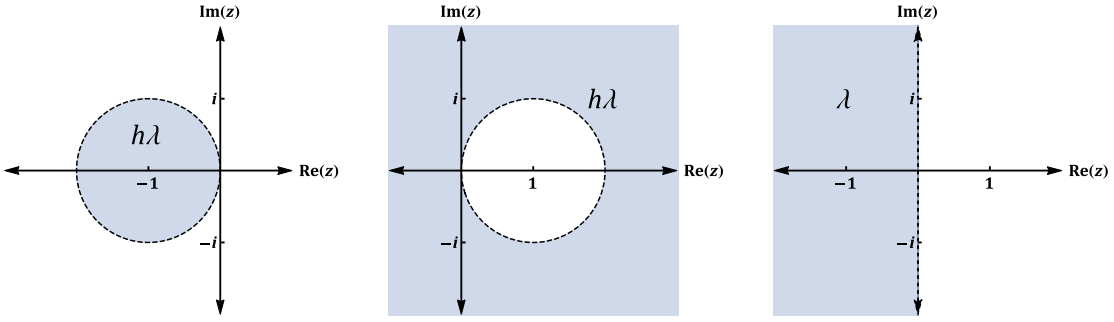
Proof. See any rigorous book on ODEs, e.g. [78]. □

Theorem 6.11 (Asymptotic Stability of Discrete Systems)**.** *The discrete evolution system given by*

$$\mathbf{x}^{n+1} = \mathbf{A}_d\mathbf{x}^n \quad \textit{with initial state} \quad \mathbf{x}^0, \tag{6.47}$$

is asymptotically stable for $n = 0, 1, \ldots, \infty$*, if* $|\lambda_k| < 1$ *for all* $\lambda_k \in \lambda(\mathbf{A}_d)$*, the eigenvalues of* $\mathbf{A}_d$*.*

Proof. See [4, 45, 87]. □



(a) Forward Euler (b) Backward Euler (c) Crank-Nicolson

Figure 6.2: The shaded region is the region of stability for selected single-step methods given eigenvalues λ of the continuous system $\mathbf{A}$.

Explicit Forward Euler Method

The Forward Euler method is defined by letting $k = 1$ and setting $\alpha_0 = 1$, $\alpha_1 = -1$, $\beta_0 = 0$, and $\beta_1 = 1$ for the multistep method in Definition 6.7. Since $\beta_0 = 0$, this is an explicit

method. Applying this method to the equation in (6.1) results in the following discrete time system.

$$\begin{aligned}
\mathbf{x}^{n+1} - \mathbf{x}^n &= h\mathbf{A}\mathbf{x}^n + h\mathbf{b}u(t^n) \\
\mathbf{x}^{n+1} &= \mathbf{x}^n + h\mathbf{A}\mathbf{x}^n + h\mathbf{b}u(t^n) \\
\mathbf{x}^{n+1} &= (\mathbf{I} + h\mathbf{A})\mathbf{x}^n + h\mathbf{b}u(t^n) \qquad (6.48) \\
\mathbf{x}^{n+1} &= \mathbf{A}_d\mathbf{x}^n + \mathbf{b}_d u(t^n) \qquad (6.49)
\end{aligned}$$

Proposition 6.12. *The stability of a Forward Euler discretization of an asymptotically stable ODE 6.46 is dependent on $\lambda(\mathbf{A})$ and the step size h. Specifically, all $\lambda_k \in \lambda(\mathbf{A})$ must satisfy*

$$|1 + h\lambda_k| < 1 \qquad (6.50)$$

Proof. Let $\mu_k \in \lambda(\mathbf{A}_d)$ and $\lambda_k \in \lambda(\mathbf{A})$. Then Theorem 6.11 says that this system is stable if for all μ_k and λ_k, the following condition holds

$$\begin{aligned}
|\mu_k| &< 1 \\
|1 + h\lambda_k| &< 1 \qquad (6.51)
\end{aligned}$$

Therefore if all $h\lambda_k$ are inside the circle of radius one centered at $z = -1$ in the complex plane, as shown in Figure 6.2a, then the system is stable. For a stable continuous system, the stability of the Forward Euler method is dependent on the step size and magnitude and argument of $\lambda(\mathbf{A})$. □

This stability requirement can lead to choosing h very small to maintain stability when it does not need to be nearly that small for accuracy. For example if $\lambda_{max} = \max|\lambda(\mathbf{A})|$ is real, then the step size, h must satisfy $h < 2/\lambda_{max}$. Additionally, if there are eigenvalues λ_k such

that $\Im(\lambda_k) \gg \Re(\lambda_k)$, then the step size h will need to be much smaller than $2/\lambda_k$ to satisfy the stability requirements. A numerical example of this behavior is presented in Section 6.5.

Implicit Backward Euler Method

The Backward Euler method is defined by setting $\alpha_0 = 1$, $\alpha_1 = -1$, $\beta_0 = 1$, and $\beta_1 = 0$ for the multistep method in Definition 6.7. Since $\beta_0 \neq 0$, this is an implicit method. Applying this method to the equation in (6.1) results in the following discrete time system.

$$\begin{aligned}
\mathbf{x}^{n+1} - \mathbf{x}^n &= h\mathbf{A}\mathbf{x}^{n+1} + h\mathbf{b}u(t^{n+1}) \\
\mathbf{x}^{n+1} &= \mathbf{x}^n + h\mathbf{A}\mathbf{x}^{n+1} + h\mathbf{b}u(t^{n+1}) \\
(\mathbf{I} - h\mathbf{A})\mathbf{x}^{n+1} &= \mathbf{x}^n + h\mathbf{b}u(t^{n+1}) \\
(\mathbf{I} - h\mathbf{A})\mathbf{x}^{n+1} &= \mathbf{x}^n + h\mathbf{b}u(t^{n+1}) \\
\mathbf{x}^{n+1} &= (\mathbf{I} - h\mathbf{A})^{-1}\mathbf{x}^n + h(\mathbf{I} - h\mathbf{A})^{-1}\mathbf{b}u(t^n) \qquad (6.52) \\
\mathbf{x}^{n+1} &= \mathbf{A}_d\mathbf{x}^n + \mathbf{b}_d u(t^n) \qquad (6.53)
\end{aligned}$$

Proposition 6.13. *The Backward Euler discretization of an asymptotically stable ODE given by 6.46 is always stable regardless of the time step size h.*

Proof. Let $\mu_k \in \lambda(\mathbf{A}_d)$ and $\lambda_k \in \lambda(\mathbf{A})$. Then Theorem 6.11 says that this system is stable if for all μ_k and λ_k, the following condition holds

$$\begin{aligned}
|\mu_k| &< 1 \\
\frac{1}{|1 - h\lambda_k|} &< 1 \qquad (6.54) \\
|1 - h\lambda_k| &> 1 \qquad (6.55)
\end{aligned}$$

Therefore if all $h\lambda_k$ are outside the circle of radius one centered at $z = 1$ in the complex plane, as shown in Figure 6.2b, then the system is stable. For a stable continuous system, $\Re(\lambda_k) < 0$ and thus the Backward Euler method is always stable, regardless of the choice of step size. □

Implicit Crank-Nicolson Method

The Crank-Nicolson method is defined by setting $\alpha_0 = 1$, $\alpha_1 = -1$, $\beta_0 = \frac{1}{2}$, and $\beta_1 = \frac{1}{2}$ for the multistep method in Definition 6.7. Since $\beta_0 \neq 0$, this is an implicit method. Applying this method to the equation in (6.1) results in the following discrete time system.

$$\begin{aligned}
\mathbf{x}^{n+1} - \mathbf{x}^n &= \tfrac{h}{2}\left(\mathbf{A}\mathbf{x}^n + \mathbf{b}u^n + \mathbf{A}\mathbf{x}^{n+1}\mathbf{b}u^{n+1}\right) \\
\mathbf{x}^{n+1} &= \mathbf{x}^n + \tfrac{h}{2}\left(\mathbf{A}\mathbf{x}^n + \mathbf{b}u^n + \mathbf{A}\mathbf{x}^{n+1}\mathbf{b}u^{n+1}\right) \\
\mathbf{x}^{n+1} - \tfrac{h}{2}\mathbf{A}\mathbf{x}^{n+1} &= \mathbf{x}^n + \tfrac{h}{2}\mathbf{A}\mathbf{x}^n + \tfrac{h}{2}\mathbf{b}(u^n + u^{n+1}) \\
\left(\mathbf{I} - \tfrac{h}{2}\mathbf{A}\right)\mathbf{x}^{n+1} &= \left(\mathbf{I} + \tfrac{h}{2}\mathbf{A}\right)\mathbf{x}^n + \tfrac{h}{2}\mathbf{b}\left(u^n + u^{n+1}\right) \\
\mathbf{x}^{n+1} &= \left(\mathbf{I} - \tfrac{h}{2}\mathbf{A}\right)^{-1}\left(\mathbf{I} + \tfrac{h}{2}\mathbf{A}\right)\mathbf{x}^n + \tfrac{h}{2}\left(\mathbf{I} - \tfrac{h}{2}\mathbf{A}\right)^{-1}\mathbf{b}\left(u^n + u^{n+1}\right) \qquad (6.56) \\
\mathbf{x}^{n+1} &= \mathbf{A}_d\mathbf{x}^n + \mathbf{b}_d u(t^n) + \mathbf{b}_d u(t^{n+1}) \qquad (6.57)
\end{aligned}$$

Proposition 6.14. *The Crank-Nicolson discretization (6.56) of an ODE given by (6.46) is asymptotically stable if and only if the ODE is asymptotically stable.*

Proof. We note that

$$\mu = T(\lambda) = \frac{1 + (h/2)\lambda}{1 - (h/2)\lambda} \qquad (6.58)$$

is a conformal map from the left half of the complex plane to the unit disk [40, 73]. Suppose that the ODE (6.46) is asymptotically stable. Then Theorem 6.10 says that for all $\lambda_k \in \lambda(\mathbf{A})$,

$\Re(\lambda_k) < 0$. Then $|T(\lambda_k)| < 1$ and

$$T(\lambda_k) = \frac{1 + (h/2)\lambda_k}{1 - (h/2)\lambda_k}. \tag{6.59}$$

Since $\mathbf{A}_d = (\mathbf{I} - (h/2)\mathbf{A})^{-1}(\mathbf{I} + (h/2)\mathbf{A})$, the eigenvalues $\mu_k \in \lambda(\mathbf{A}_d)$ are given by

$$\mu_k = \frac{1 + (h/2)\lambda}{1 - (h/2)\lambda} = T(\lambda_k) \tag{6.60}$$

Therefore $|\mu_k| < 1$ for all $\mu_k \in \lambda(\mathbf{A}_d)$. Thus, by Theorem 6.11 the discretized system (6.56) is asymptotically stable.

For the other direction, let the discretized system (6.56) be asymptotically stable. Then $|\mu_k| < 1$ for all $\mu_k \in \lambda(\mathbf{A}_d)$. Since conformal maps are invertible [40, 73], we have that $T^{-1}(\mu_k) = \lambda_k$ and $\Re(\lambda_k) < 0$. Therefore the conditions of Theorem 6.10 are satisfied and the ODE given by (6.46) is asymptotically stable. □

6.3 Relationship of Transfer Functions

When we compute the transfer function for a continuous time system, we take the Laplace transform; but when we determine the transfer function for a discrete time system, we are taking the Z-transform. For continuous time systems we let $G : S \subset \mathbb{C} \to \mathbb{C}$ be the transfer function and let $H : Z \subset \mathbb{C} \to \mathbb{C}$ be the transfer function for discrete time systems. We seek to find an invertible mapping $T : S \to Z$ such that $G(s) = H(T(s))$ and $G(T^{-1}(z)) = H(z)$. It turns out that this mapping is dependent on the method that is used to discretize the continuous time system. We will present results for a couple 1-step methods below.

6.3.1 Explicit Forward Euler Method

Given our standard definition a continuous time linear SISO system below:

$$\Sigma : \begin{cases} \dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x} + \mathbf{b}u(t) \\ y(t) &= \mathbf{c}^T\mathbf{x}. \end{cases} \tag{6.61}$$

As seen before, this system has the transfer function

$$G(s) = \mathbf{c}^T(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{b} \tag{6.62}$$

For the forward Euler discretization in Equation (6.48), the discrete time linear SISO dynamical system is given by

$$\Sigma_d : \begin{cases} \mathbf{x}^{n+1} &= (\mathbf{I} + h\mathbf{A})\mathbf{x}^n + h\mathbf{b}u^n \\ y^n &= \mathbf{c}^T\mathbf{x}^n \end{cases} \tag{6.63}$$

We define the transfer function for this discrete time system as

$$H(z) = \mathbf{c}^T(z\mathbf{I} - (\mathbf{I} + h\mathbf{A}))^{-1}h\mathbf{b} \tag{6.64}$$

Theorem 6.15. *Suppose that the linear SISO system (6.61) is discretized in time using the forward Euler method resulting in the discrete-time dynamical system given by (6.63). There is an invertible mapping T from the domain of (6.62), S, to the domain of (6.64), Z, such*

that

$$G(s) = H(T(s)) \tag{6.65}$$

$$H(z) = G(T^{-1}(z)) \tag{6.66}$$

Further, this map is given by

$$T : S \to Z \quad \textit{where} \quad s \mapsto 1 + hs = z. \tag{6.67}$$

Proof. Using the transfer functions (6.62) and (6.64), we develop the mapping for T.

$$\begin{aligned} G(s) &= \mathbf{c}^T(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{b} \\ &= \mathbf{c}^T(s\mathbf{I} - \mathbf{A})^{-1}h^{-1}h\mathbf{b} \\ &= \mathbf{c}^T(hs\mathbf{I} - h\mathbf{A})^{-1}h\mathbf{b} \\ &= \mathbf{c}^T(hs\mathbf{I} + \mathbf{I} - \mathbf{I} - h\mathbf{A})^{-1}h\mathbf{b} \\ &= \mathbf{c}^T((1 + hs)\mathbf{I} - (\mathbf{I} + h\mathbf{A}))^{-1}h\mathbf{b} \\ \implies G(s) &= H(1 + hs) = H(T(s)) \end{aligned} \tag{6.68}$$

This gives the mapping in (6.67) and shows (6.65). It is easy to show that $T^{-1} : Z \to S$ is

defined by $z \mapsto (z-1)/h = s$. Now we show that $G(T^{-1}(z)) = H(z)$.

$$\begin{aligned} H(z) &= \mathbf{c}^T \left(z\mathbf{I} - (\mathbf{I} + h\mathbf{A})\right)^{-1} h\mathbf{b} \\ &= \mathbf{c}^T \left((z-1)\mathbf{I} - h\mathbf{A}\right)^{-1} h\mathbf{b} \\ &= \left[h\left(\left(\tfrac{z-1}{h}\right)\mathbf{I} - \mathbf{A}\right)\right]^{-1} h\mathbf{b} \\ &= \left(\left(\tfrac{z-1}{h}\right)\mathbf{I} - \mathbf{A}\right)^{-1} \mathbf{b} \\ \implies H(z) &= G(\tfrac{z-1}{h}) = G(T^{-1}(z)) \end{aligned} \tag{6.69}$$

□

6.3.2 Implicit Backward Euler Method

Given the continuous time linear SISO system from (6.61) and its associated transfer function (6.62), we apply the Backward Euler method from (6.52) to obtain the following discrete dynamical system

$$\Sigma_d : \begin{cases} \mathbf{x}^{n+1} &= (\mathbf{I} - h\mathbf{A})^{-1}\mathbf{x}^n + (\mathbf{I} - h\mathbf{A})^{-1} h\mathbf{b} u^{n+1} \\ y^n &= \mathbf{c}^T \mathbf{x}^n, \end{cases} \tag{6.70}$$

and its associated transfer function

$$H(z) = \mathbf{c}^T (z\mathbf{I} - (\mathbf{I} - h\mathbf{A})^{-1})^{-1} (\mathbf{I} - h\mathbf{A})^{-1} h\mathbf{b} z. \tag{6.71}$$

Theorem 6.16. *Suppose that the linear SISO system (6.61) is discretized in time using the forward Euler method resulting in the discrete-time dynamical system given by (6.70). There is an invertible mapping T from the domain of (6.62), S, to the domain of (6.71), Z, such*

that

$$G(s) = H(T(s)) \tag{6.72}$$

$$H(z) = G(T^{-1}(z)) \tag{6.73}$$

Further, this map is given by

$$T : S \to Z \quad \textit{where} \quad s \mapsto \frac{1}{1-hs} = z. \tag{6.74}$$

Proof. Using the transfer functions (6.62) and (6.71), we develop the mapping for T.

$$\begin{aligned}
G(s) &= \mathbf{c}^T(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{b} \\
&= \mathbf{c}^T(h(s\mathbf{I} - \mathbf{A}))^{-1}h\mathbf{b} \\
&= \mathbf{c}^T(\mathbf{I} - h\mathbf{A} - \mathbf{I} + hs\mathbf{I})^{-1}h\mathbf{b} \\
&= \mathbf{c}^T\left[\left(\tfrac{1}{1-hs}\right)((\mathbf{I} - h\mathbf{A}) - (1-hs)\mathbf{I})\right]^{-1}h\mathbf{b}\left(\tfrac{1}{hs}\right) \\
&= \mathbf{c}^T\left[\left(\tfrac{1}{1-hs}\right)(\mathbf{I} - h\mathbf{A}) - \mathbf{I}\right]^{-1}h\mathbf{b}\left(\tfrac{1}{1-hs}\right) \\
&= \mathbf{c}^T\left[(\mathbf{I} - h\mathbf{A})^{-1}\left(\left(\tfrac{1}{1-hs}\right)(\mathbf{I} - h\mathbf{A}) - \mathbf{I}\right)\right]^{-1}(\mathbf{I} - h\mathbf{A})^{-1}h\mathbf{b}\left(\tfrac{1}{1-hs}\right) \\
&= \mathbf{c}^T\left(\left(\tfrac{1}{1-hs}\right)\mathbf{I} - (\mathbf{I} - h\mathbf{A})^{-1}\right)^{-1}(\mathbf{I} - h\mathbf{A})^{-1}h\mathbf{b}\left(\tfrac{1}{1-hs}\right) \\
s \implies G(s) &= H(\tfrac{1}{1-hs}) = H(T(s))
\end{aligned} \tag{6.75}$$

This gives the mapping in (6.74) and shows (6.72). It is easy to show that $T^{-1} : Z \to S$ is

defined by $z \mapsto \frac{z-1}{zh} = s$. Now we show that $H(z) = G(T^{-1}(z))$.

$$\begin{aligned} H(z) &= \mathbf{c}^T \left(z\mathbf{I} - (\mathbf{I} - h\mathbf{A})^{-1}\right) h(\mathbf{I} - h\mathbf{A})^{-1}\mathbf{b}z \\ &= \mathbf{c}^T \left(z(\mathbf{I} - h\mathbf{A}) - \mathbf{I}\right)^{-1} (\mathbf{I} - h\mathbf{A})(\mathbf{I} - h\mathbf{A})^{-1}h\mathbf{b}z \\ &= \mathbf{c}^T \left(z\mathbf{I} - zh\mathbf{A} - \mathbf{I}\right)^{-1} h\mathbf{b}z \\ &= \mathbf{c}^T \left((z-1)\mathbf{I} - zh\mathbf{A}\right)^{-1} hz\mathbf{b} \\ &= \mathbf{c}^T \left[zh\left(\left(\tfrac{z-1}{zh}\right)\mathbf{I} - \mathbf{A}\right)\right]^{-1} zh\mathbf{b} \\ &= \mathbf{c}^T \left(\left(\tfrac{z-1}{zh}\right)\mathbf{I} - \mathbf{A}\right)^{-1} \mathbf{b} \\ \implies H(z) &= G(\tfrac{z-1}{zh}) = G(T^{-1}(z)) \end{aligned} \tag{6.76}$$

□

6.3.3 Implicit Crank-Nicolson Method

Again suppose we have the linear SISO dynamical system (6.61) with the associated transfer function (6.62). Applying the implicit Crank-Nicolson method with time step h to discretize the state equation in time results in the time discretized state equation (6.56). This leads to the discrete dynamical system

$$\begin{cases} \mathbf{x}^{n+1} &= \left(\mathbf{I} - \frac{h}{2}\mathbf{A}\right)^{-1} \left(\mathbf{I} + \frac{h}{2}\mathbf{A}\right) \mathbf{x}^n + \frac{h}{2} \left(\mathbf{I} - \frac{h}{2}\mathbf{A}\right)^{-1} \mathbf{b} \left(u^n + u^{n+1}\right) \\ \mathbf{y}^n &= \mathbf{c}^T \mathbf{x}^n \end{cases} \tag{6.77}$$

By taking the Z-transform of this system, we obtain the following transfer function:

$$H(z) = \mathbf{c}^T \left(z\mathbf{I} - \left(\mathbf{I} - \tfrac{h}{2}\mathbf{A}\right)^{-1} \left(\mathbf{I} + \tfrac{h}{2}\mathbf{A}\right)\right)^{-1} \left(\mathbf{I} - \tfrac{h}{2}\mathbf{A}\right)^{-1} \tfrac{h}{2}\mathbf{b}(1+z) \tag{6.78}$$

Theorem 6.17. *Suppose that the linear SISO system (6.61) is discretized in time using the implicit Crank-Nicolson method resulting in the discrete-time dynamical system given by (6.77). There is an invertible mapping T from the domain of (6.62), S, to the domain of (6.78), Z, such that*

$$G(s) = H(T(s)) \tag{6.79}$$

$$H(z) = G(T^{-1}(z)) \tag{6.80}$$

Further, this map is given by

$$T : S \to Z \quad \text{where} \quad s \mapsto \frac{2 + hs}{2 - hs} = z. \tag{6.81}$$

Proof. Using the transfer functions (6.62) and (6.78), we develop the mapping for T.

$$
\begin{aligned}
G(s) &= \mathbf{c}^T(s\mathbf{I}-\mathbf{A})^{-1}\mathbf{b} \\
&= \mathbf{c}^T(s\mathbf{I}-\mathbf{A})^{-1}(2h)^{-1}2h\mathbf{b} \\
&= \mathbf{c}^T(2hs\mathbf{I}-2h\mathbf{A})^{-1}2h\mathbf{b} \\
&= \mathbf{c}^T\left(2\mathbf{I}-2\mathbf{I}+2hs\mathbf{I}-2h\mathbf{A}-\tfrac{h^2s}{2}\mathbf{A}+\tfrac{h^2s}{2}\mathbf{A}\right)^{-1}2h\mathbf{b} \\
&= \mathbf{c}^T\left[\left(2\mathbf{I}+hs\mathbf{I}-h\mathbf{A}-\tfrac{h^2s}{2}\mathbf{A}\right)-\left(2\mathbf{I}-hs\mathbf{I}+h\mathbf{A}-\tfrac{h^2s}{2}\mathbf{A}\right)\right]^{-1}2h\mathbf{b} \\
&= \mathbf{c}^T\left[(2+hs)\left(\mathbf{I}-\tfrac{h}{2}\mathbf{A}\right)-(2-hs)\left(\mathbf{I}+\tfrac{h}{2}\mathbf{A}\right)\right]^{-1}2h\mathbf{b} \\
&= \mathbf{c}^T\left[(2+hs)\mathbf{I}-(2-hs)\left(\mathbf{I}-\tfrac{h}{2}\mathbf{A}\right)^{-1}\left(\mathbf{I}+\tfrac{h}{2}\mathbf{A}\right)\right]^{-1}\left(\mathbf{I}-\tfrac{h}{2}\mathbf{A}\right)^{-1}2h\mathbf{b} \\
&= \mathbf{c}^T\left[(2-hs)\left(\left(\tfrac{2+hs}{2-hs}\right)\mathbf{I}-\left(\mathbf{I}-\tfrac{h}{2}\mathbf{A}\right)^{-1}\left(\mathbf{I}+\tfrac{h}{2}\mathbf{A}\right)\right)\right]^{-1}\left(\mathbf{I}-\tfrac{h}{2}\mathbf{A}\right)^{-1}2h\mathbf{b} \\
&= \mathbf{c}^T\left(\left(\tfrac{2+hs}{2-hs}\right)\mathbf{I}-\left(\mathbf{I}-\tfrac{h}{2}\mathbf{A}\right)^{-1}\left(\mathbf{I}+\tfrac{h}{2}\mathbf{A}\right)\right)^{-1}\left(\mathbf{I}-\tfrac{h}{2}\mathbf{A}\right)^{-1}\mathbf{b}\tfrac{h}{2}\tfrac{4}{2-hs} \\
&= \mathbf{c}^T\left(\left(\tfrac{2+hs}{2-hs}\right)\mathbf{I}-\left(\mathbf{I}-\tfrac{h}{2}\mathbf{A}\right)^{-1}\left(\mathbf{I}+\tfrac{h}{2}\mathbf{A}\right)\right)^{-1}\left(\mathbf{I}-\tfrac{h}{2}\mathbf{A}\right)^{-1}\mathbf{b}\tfrac{h}{2}\tfrac{2-hs+2+hs}{2-hs} \\
&= \mathbf{c}^T\left(\left(\tfrac{2+hs}{2-hs}\right)\mathbf{I}-\left(\mathbf{I}-\tfrac{h}{2}\mathbf{A}\right)^{-1}\left(\mathbf{I}+\tfrac{h}{2}\mathbf{A}\right)\right)^{-1}\left(\mathbf{I}-\tfrac{h}{2}\mathbf{A}\right)^{-1}\mathbf{b}\tfrac{h}{2}\left(1+\tfrac{2+hs}{2-hs}\right) \\
\implies G(s) &= H\left(\tfrac{2+hs}{2-hs}\right) = H(T(s)) \qquad (6.82)
\end{aligned}
$$

This gives the mapping in (6.81) and shows (6.79). It is straight-forward to show that

$T^{-1} : Z \to S$ is defined by $z \mapsto \frac{2}{h}\left(\frac{z-1}{z+1}\right) = s$. Now we show that $G(T^{-1}(z)) = H(z)$.

$$\begin{aligned}
H(z) &= \mathbf{c}^T \left(z\mathbf{I} - \left(\mathbf{I} - \tfrac{h}{2}\mathbf{A}\right)^{-1} \left(\mathbf{I} + \tfrac{h}{2}\mathbf{A}\right) \right)^{-1} \left(\mathbf{I} - \tfrac{h}{2}\mathbf{A}\right)^{-1} \tfrac{h}{2}\mathbf{b}(1+z) \\
&= \mathbf{c}^T \left[\left(\mathbf{I} - \tfrac{h}{2}\mathbf{A}\right)^{-1} \left(z\left(\mathbf{I} - \tfrac{h}{2}\mathbf{A}\right) - \left(\mathbf{I} + \tfrac{h}{2}\mathbf{A}\right) \right) \right]^{-1} \left(\mathbf{I} - \tfrac{h}{2}\mathbf{A}\right)^{-1} \tfrac{h}{2}\mathbf{b}(1+z) \\
&= \mathbf{c}^T \left(z\mathbf{I} - \tfrac{h}{2}z\mathbf{A} - \mathbf{I} - \tfrac{h}{2}\mathbf{A} \right)^{-1} \tfrac{h}{2}\mathbf{b}(1+z) \\
&= \mathbf{c}^T \left((z-1)\,\mathbf{I} - \tfrac{h}{2}\,(1+z)\,\mathbf{A} \right)^{-1} \tfrac{h}{2}\mathbf{b}(1+z) \\
&= \mathbf{c}^T \left[(1+z)\left(\left(\tfrac{z-1}{z+1}\right)\mathbf{I} - \tfrac{h}{2}\mathbf{A} \right) \right]^{-1} \tfrac{h}{2}\mathbf{b}(1+z) \\
&= \mathbf{c}^T \left[\tfrac{h}{2}\left(\tfrac{2}{h}\left(\tfrac{z-1}{z+1}\right)\mathbf{I} - \mathbf{A} \right) \right]^{-1} \tfrac{h}{2}\mathbf{b} \\
&= \mathbf{c}^T \left(\tfrac{2}{h}\left(\tfrac{z-1}{z+1}\right)\mathbf{I} - \mathbf{A} \right)^{-1} \mathbf{b} \\
\implies H(z) &= G\left(\tfrac{2}{h}\left(\tfrac{z-1}{z+1}\right) \right) = G(T^{-1}(z))
\end{aligned} \tag{6.83}$$

□

6.3.4 Relationship of Discrete-Time Transfer Functions

For the single-step methods above, we have shown that function values are preserved under the transformation T. Using this information, we present the following theorem relating the function values and the derivatives at the $\mathcal{H}_2$-optimal interpolation points of the continuous-time ROM for the discrete-time systems created from the full-order and reduced-order systems.

Corollary 6.18. *Suppose the transfer function $H(z)$ for the discrete-discrete time system, H, is created from the full-order system, G, with transfer function, $G(s)$. Further, the transfer function $H_r(z)$ for the reduced-order discrete-time system, H_r, is created from the ROM, G_r which satisfies the first order necessary conditions for $\mathcal{H}_2$ optimality with respect*

to G. Let λ_k, $k = 1, \ldots, r$, be the poles for the reduced-order transfer function, $G_r(s)$. Then

$$H(T(-\lambda_k) = H_r(T(-\lambda_k)), \tag{6.84}$$

$$H'(T(-\lambda_k) = H_r'(T(-\lambda_k)). \tag{6.85}$$

Proof. From Theorem 6.2, we have that

$$G(-\lambda_k) = G_r(-\lambda_k). \tag{6.86}$$

Based on Theorems 6.15, 6.16, and 6.17, there exist mappings $T : S \to Z$ for each single-step method such that

$$G(-\lambda_k) = H(T(-\lambda_k)), \tag{6.87}$$

$$G_r(-\lambda_k) = H_r(T(-\lambda_k)). \tag{6.88}$$

Therefore

$$H(T(-\lambda_k)) = G(-\lambda_k) = G_r(-\lambda_k) = H(T(-\lambda_k)), \tag{6.89}$$

and the first equality (6.84) is shown.

Taking the derivative with respect to s and applying the chain rule, we have

$$G'(s) = \frac{d}{ds}[H(T(s))] = H'(T(s))T'(s) \tag{6.90}$$

$$G_r'(s) = \frac{d}{ds}[H_r(T(s))] = H_r'(T(s))T'(s) \tag{6.91}$$

Looking at the $\mathcal{H}_2$-optimal interpolation points λ_k, $k = 1, \ldots, r$, we have that

$$H'(T(-\lambda_k))T'(-\lambda_k) = G'(-\lambda_k), \tag{6.92}$$

$$= G_r'(-\lambda_k), \tag{6.93}$$

$$= H_r'(T(-\lambda_k))T'(-\lambda_k), \tag{6.94}$$

which gives

$$H'(T(-\lambda_k))T'(-\lambda_k) = H_r'(T(-\lambda_k))T'(-\lambda_k), \tag{6.95}$$

$$H'(T(-\lambda_k)) = H_r'(T(-\lambda_k)), \tag{6.96}$$

and the second result is shown. □

6.4 Extension of $\mathcal{H}_2$ Optimality

As stated earlier, it is typical to apply the $\mathcal{H}_2$-optimal model reduction directly to the FOM ODE and then to apply the time discretization to the ROM ODE to create a ROM$_c$ OΔE. We would like to determine whether or not the ROM$_c$ OΔE created is an $\mathcal{H}_2$-optimal reduced-order model for the FOM OΔE created from the FOM ODE, and, by extension, is equivalent to ROM$_d$ OΔE. In other words, the time discretization method preserves the $\mathcal{H}_2$ optimality condition of the reduced-order model, and therefore the order with which the model reduction and time discretization are completed does not matter. It turns out that this preservation of the $\mathcal{H}_2$ optimality condition depends on the time discretization method as we will see in the following theorems.

For the remainder of this chapter, we will refer to the FOM ODE system as G with the

associated transfer function $G(s)$ and the ROM ODE as G_r with transfer function $G_r(s)$. The time discretization of G is given by H with its transfer function given by $H(z)$. The discrete-time system built from G_r is given by H_r with its transfer function $H_r(z)$. Finally, the discrete-time $\mathcal{H}_2$-optimal ROM built from H will be referred to as H_d with transfer function $H_d(z)$.

6.4.1 Forward and Backward Euler

In the following two theorems, we show that the Forward and Backward Euler methods do not preserve the $\mathcal{H}_2$-optimality of the continuous time ROM.

Theorem 6.19. *Given an asymptotically-stable full-order system of ODEs, G, the Forward Euler method applied to the reduced-order model, G_r, satisfying the first-order necessary conditions for $\mathcal{H}_2$ optimality produces a discrete-time system that does not preserve the first-order necessary conditions for $\mathcal{H}_2$ optimality in the discrete-time system.*

Proof. It suffices to show that the H_r does not satisfy the first-order necessary conditions for $\mathcal{H}_2$ optimality with respect to H. Let G be an n dimensional system of ODEs, and G_r be the r dimensional $\mathcal{H}_2$-optimal approximation of that system. Since the original system is stable then so is the ROM and thus $\mathbf{A}$ and $\mathbf{A}_r$ are nonsingular. G_r is the $\mathcal{H}_2$-optimal reduced-order model given by

$$G_r = \begin{cases} \dot{\mathbf{x}}_r(t) &= \mathbf{W}_r^T\mathbf{A}\mathbf{V}_r\mathbf{x}_r(t) + \mathbf{W}_r^T\mathbf{b}u(t) \\ y_r(t) &= \mathbf{c}^T\mathbf{V}_r\mathbf{x}_r(t), \end{cases} \tag{6.97}$$

with the degree r rational transfer function

$$G_r(s) = \mathbf{c}_r^T(\mathbf{I} - \mathbf{A}_r)^{-1}\mathbf{b}_r, \tag{6.98}$$

where $\mathbf{A}_r = \mathbf{W}_r^T \mathbf{A} \mathbf{V}_r$, $\mathbf{b}_r = \mathbf{W}^T \mathbf{b}$ and $\mathbf{c}_r^T = \mathbf{c}^T \mathbf{V}_r$. Let λ_k be the eigenvalues of A_r, $\lambda_k \in \lambda(\mathbf{A}_r)$, which are also the poles of $G_r(s)$. Applying the Forward Euler method to G_r yields a discrete dynamical system H_r with the r dimensional rational transfer function

$$H_r(z) = \mathbf{c}_r^T (z\mathbf{I} - (\mathbf{I} + h\mathbf{A}_r))^{-1} h\mathbf{b}_r, \tag{6.99}$$

with poles at $\mu_k = 1 + h\lambda_k$. By Theorem 6.2, we have that for all $\lambda_k \in \lambda(\mathbf{A}_r)$

$$G(-\lambda_k) = G_r(-\lambda_k) \quad \text{and} \quad G'(-\lambda_k) = G_r'(-\lambda_k) \tag{6.100}$$

By Theorem 6.15, we have that

$$G_r(-\lambda_k) = H_r(1 - h\lambda_k) \quad \text{and} \quad G_r'(-\lambda_k) = H_r'(1 - h\lambda_k), \tag{6.101}$$

and for the transfer function $H(z)$ defined by (6.64)

$$G(-\lambda_k) = H(1 - h\lambda_k) \quad \text{and} \quad G'(-\lambda_k) = H'(1 - h\lambda_k). \tag{6.102}$$

Combining (6.100), (6.101), and (6.102) gives that

$$H(1 - h\lambda_k) = H_r(1 - h\lambda_k) \quad \text{and} \quad H'(1 - h\lambda_k) = H_r'(1 - h\lambda_k). \tag{6.103}$$

By Theorem 6.4, for H_r to be a ROM that satisfies the $\mathcal{H}_2$ optimality conditions for H, $H_r(z)$ must be a Hermite interpolate of $H(z)$ at $1/\mu_k$ where μ_k are the poles of $H_r(z)$. From (6.99), $\mu_k = 1 + h\lambda_k$. Since $H(z)$ is exactly a degree n rational function and $H_r(z)$ is exactly a degree r rational function, by uniqueness of the interpolation, $H_r(z)$ will only match the function and derivatives at r points [1, 40]. From (6.103), those points are $1 - h\lambda_k$. Assume

that these points are the optimal points from Theorem 6.4.

$$\begin{aligned} & \frac{1}{\mu_k} = 1 - h\lambda_k, \\ \implies & \frac{1}{1+\lambda_k} = 1 - h\lambda_k, \\ \implies & 1 = 1 - h^2\lambda_k^2, \\ \implies & h^2\lambda_k^2 = 0, \\ \implies & \lambda_k = 0. \end{aligned} \tag{6.104}$$

for $k = 1, \ldots, r$. Clearly, this condition cannot be met in our case. Therefore H_r does not provide a ROM that satisfies the first-order necessary conditions for $\mathcal{H}_2$ optimality with respect to the discrete-time system H when G_r and G are discretized in time using the explicit Forward Euler method. □

Theorem 6.20. *Given an asymptotically-stable full-order system of ODEs, G, the Backward Euler method applied to the reduced-order model, G_r, satisfying the first-order necessary conditions for $\mathcal{H}_2$ optimality produces a discrete-time system that does not preserve the first-order necessary conditions for $\mathcal{H}_2$ optimality in the discrete-time system.*

Proof. This proof follows the same format as the proof for Theorem 6.19. Again, it suffices to show that the H_r does not satisfy the first-order necessary conditions for $\mathcal{H}_2$ optimality with respect to H. Following the same setup we have the ROM satisfying the first-order necessary conditions for $\mathcal{H}_2$ optimality given by (6.97) and its associated transfer function (6.98) with all the same properties. Applying the Backward Euler method to the ROM ODE yields a discrete dynamical system with the r dimensional rational transfer function

$$H_r(z) = \mathbf{c}_r^T(z\mathbf{I} - (\mathbf{I} - h\mathbf{A}_r)^{-1})^{-1}(\mathbf{I} - h\mathbf{A}_r)^{-1}h\mathbf{b}_r z, \tag{6.105}$$

with poles at $\mu_k = \frac{1}{1-h\lambda_k}$ where $\lambda_k \in \lambda(\mathbf{A}_r)$. By Theorem 6.2, we have that for all $\lambda_k \in \lambda(\mathbf{A}_r)$, (6.100) holds. Further, by Theorem 6.16, we have that

$$G_r(-\lambda_k) = H_r(\tfrac{1}{1+h\lambda_k}) \quad \text{and} \quad G_r'(-\lambda_k) = H_r'(\tfrac{1}{1+h\lambda_k}), \tag{6.106}$$

and for the transfer function $H(z)$ defined by (6.71)

$$G(-\lambda_k) = H(\tfrac{1}{1+h\lambda_k}) \quad \text{and} \quad G'(-\lambda_k) = H'(\tfrac{1}{1+h\lambda_k}). \tag{6.107}$$

Combining (6.100), (6.106), and (6.107) gives that

$$H(\tfrac{1}{1+h\lambda_k}) = H_r(\tfrac{1}{1+h\lambda_k}) \quad \text{and} \quad H'(\tfrac{1}{1+h\lambda_k}) = H_r'(\tfrac{1}{1+h\lambda_k}). \tag{6.108}$$

We now show that these interpolation points are not equal to $1/\mu_k$, where μ_k are the poles of $H_r(z)$, which is required by Theorem 6.4 for $H_r(z)$ to be an $\mathcal{H}_2$ optimal interpolation of $H(z)$. Assume that the points are equal, then

$$\begin{aligned}
&\frac{1}{\mu_k} = \frac{1}{1+h\lambda_k}, \\
\implies &1 - \lambda_k = \frac{1}{1+h\lambda_k}, \\
\implies &1 - h^2\lambda_k^2 = 1, \\
\implies &h^2\lambda_k^2 = 0, \\
\implies &\lambda_k = 0,
\end{aligned} \tag{6.109}$$

for $k = 1, \ldots, r$. By earlier arguments in Theorem 6.19, H_r does not provide a ROM that satisfies the first-order necessary conditions for $\mathcal{H}_2$ optimality with respect to the discrete-time system H when G_r and G are discretized in time using the implicit Backward Euler

method. □

6.4.2 Crank-Nicolson Method

As we pointed out in the proof of Proposition 6.14, the Crank-Nicholson method amounts to a conformal mapping from left-half plane to the unit disc. Since the reduced-order poles and the $\mathcal{H}_2$-optimal interpolation points are connected by $\lambda = -\sigma$, one might intuitively expect that after the Crank-Nicholson discretization, the $\mathcal{H}_2$-optimality will be preserved as these points will be mapped as λ and $1/\lambda$ together with the fact that the Hermite interpolation is preserved as we discussed in Corollary 6.18. However, this argument would not take into consideration the fact that the Crank-Nicholson discretization will introduce a d term that was not present in the original transfer function. Therefore, one needs to make sure that the condition of Corollary 6.5 is also met. Below, we will work explicitly with the transfer function and dynamical system framework to establish these facts.

Theorem 6.21 ($\mathcal{H}_2$ **Optimal Equivalency for Crank-Nicolson).** *Let H be the discrete-time system generated from G using the Crank-Nicholson method. Let H_r be the reduced-order discrete-time model generated from G_r, where $G_r(s)$ satisfies the first-order necessary conditions for $\mathcal{H}_2$ optimality with respect to $G(s)$. Then $H_r(z)$ satisfies the conditions defined by Theorem 6.4. That is*

$$H(1/\lambda_k) = H_d(1/\lambda_k), \tag{6.110}$$

$$H'(1/\lambda_k) = H'_d(1/\lambda_k), \tag{6.111}$$

where λ_k, $k = 1, \cdots, r$, are the eigenvalues of $\widetilde{\mathbf{A}}_d$, closed under conjugation.

Proof. Following the same initial setup as with Theorem 6.19, we have the continuous-time

ROM given by (6.97) satisfying the first-order necessary conditions for $\mathcal{H}_2$ optimality and its associated transfer function (6.98) with all the same properties. Applying the Crank-Nicolson method from (6.56) to G_r yields a discrete-time dynamical system

$$H_r : \begin{cases} \mathbf{x}_r^{n+1} & = \left(\mathbf{I} - \left(\frac{h}{2}\right)\mathbf{A}_r\right)^{-1}\left(\mathbf{I} + \left(\frac{h}{2}\right)\mathbf{A}_r\right)\mathbf{x}_r^n + \frac{h}{2}\left(\mathbf{I} - \left(\frac{h}{2}\right)\mathbf{A}_r\right)^{-1}\mathbf{b}_r(u^n + u^{n+1}) \\ y_r^n & = \mathbf{c}_r^T\mathbf{x}^n \end{cases} \tag{6.112}$$

with the r dimensional rational transfer function

$$H_r(z) = \mathbf{c}_r^T\left[z\mathbf{I} - \left(\mathbf{I} - \left(\frac{h}{2}\right)\mathbf{A}_r\right)^{-1}\left(\mathbf{I} + \left(\frac{h}{2}\right)\mathbf{A}_r\right)\right]^{-1}\left(\mathbf{I} - \left(\frac{h}{2}\right)\mathbf{A}_r\right)^{-1}\left(\mathbf{I} + \left(\frac{h}{2}\right)\mathbf{A}_r\right)\frac{h}{2}\mathbf{b}_r(1+z), \tag{6.113}$$

with poles at $\mu_k = \frac{1+(h/2)\lambda_k}{1-(h/2)\lambda_k}$ where $\lambda_k \in \lambda(\mathbf{A}_r)$. By Theorem 6.2, we have that for all $\lambda_k \in \lambda(\mathbf{A}_r)$, (6.100) holds. Further, by Theorem 6.17, we have that

$$G_r(-\lambda_k) = H_r(\tfrac{2-h\lambda_k}{2+h\lambda_k}) \qquad G_r'(-\lambda_k) = H_r'(\tfrac{2-h\lambda_k}{2+h\lambda_k}), \tag{6.114}$$

and for the transfer function $H(z)$ defined by (6.78)

$$G(-\lambda_k) = H(\tfrac{2-h\lambda_k}{2+h\lambda_k}) \quad \text{and} \quad G'(-\lambda_k) = H'(\tfrac{2-h\lambda_k}{2+h\lambda_k}). \tag{6.115}$$

Combining (6.100), (6.114), and (6.115) gives that

$$H(\tfrac{2-h\lambda_k}{2+h\lambda_k}) = H_r(\tfrac{2-h\lambda_k}{2+h\lambda_k}) \quad \text{and} \quad H'(\tfrac{2-h\lambda_k}{2+h\lambda_k}) = H_r'(\tfrac{2-h\lambda_k}{2+h\lambda_k}). \tag{6.116}$$

We now show that the interpolation points in (6.116) are equal to $1/\mu_k$, where μ_k are the poles of $H_r(z)$, which is required by Theorem 6.4 for $H_r(z)$ to be an $\mathcal{H}_2$-optimal interpolation

of $H(z)$.

$$\frac{1}{\mu_k} = \frac{1}{\frac{1+(h/2)\lambda_k}{1-(h/2)\lambda_k}} = \frac{1-(h/2)\lambda_k}{1+(h/2)\lambda_k} = \frac{2-h\lambda_k}{2+h\lambda_k} \tag{6.117}$$

Therefore, the Crank-Nicholson method preserves the conditions given in Theorem 6.4. □

We note here that the Crank-Nicholson method produces a d term in the discrete-time transfer function $H(z)$. Specifically, when we look at the behavior of $H(z)$ as $z \to \infty$, we see that $H(z) \not\to 0$. To show this fact, we first define a new transfer function $F(z)$ by

$$F(z) = \widetilde{\mathbf{c}}^T(z\mathbf{I} - \widetilde{\mathbf{A}})^{-1}\widetilde{\mathbf{b}} = \mathbf{c}^T\left[z\mathbf{I} - (\mathbf{I} - \tfrac{h}{2}\mathbf{A})^{-1}(\mathbf{I} + \tfrac{h}{2}\mathbf{A})\right]^{-1}(\mathbf{I} - \tfrac{h}{2}\mathbf{A})\tfrac{h}{2}\mathbf{b}. \tag{6.118}$$

We can write this function in pole-residue form as

$$F(z) = \sum_{k=1}^{n} \frac{\phi_k}{z - \mu_k}, \tag{6.119}$$

where μ_k are the poles of F and ϕ_k are the residues. Further, we see that the sum of the residues is just $\mathbf{c}^T\widetilde{\mathbf{b}}$, as shown in [4, 67].

Proposition 6.22. *When a continuous-time system G is discretized using the Crank-Nicholson method the resulting transfer function $H(z)$ contains a d term. In addition this term is given by*

$$d = \mathbf{c}^T\widetilde{\mathbf{b}} \tag{6.120}$$

where $\widetilde{\mathbf{c}}^T = \mathbf{c}^T$ *and* $\widetilde{\mathbf{b}} = \left(\mathbf{I} - \frac{h}{2}\mathbf{A}\right)^{-1}\frac{h}{2}\mathbf{b}$.

Proof. Recall the transfer function for the Crank-Nicholson discretization is given by

$$\begin{aligned} H(z) &= \mathbf{c}^T \left(z\mathbf{I} - \left(\mathbf{I} - \tfrac{h}{2}\mathbf{A}\right)^{-1} \left(\mathbf{I} + \tfrac{h}{2}\mathbf{A}\right) \right)^{-1} \left(\mathbf{I} - \tfrac{h}{2}\mathbf{A}\right)^{-1} \tfrac{h}{2}\mathbf{b}(1+z), \\ &= \widetilde{\mathbf{c}}^T (z\mathbf{I} - \widetilde{\mathbf{A}})^{-1} \widetilde{\mathbf{b}}(1+z), \end{aligned} \tag{6.121}$$

where $\widetilde{\mathbf{c}}^T = \mathbf{c}^T$ and $\widetilde{\mathbf{b}} = \left(\mathbf{I} - \frac{h}{2}\mathbf{A}\right)^{-1} \frac{h}{2}\mathbf{b}$. Now, we see that for any discrete-time transfer function $K(z) = \mathbf{c}^T(z\mathbf{I} - \mathbf{A})^{-1}\mathbf{b} + d$, we can find the d term by looking at the behavior of $K(z)$ at ∞ as seen by

$$\lim_{z\to\infty} K(z) = \lim_{z\to\infty} [\mathbf{c}^T(z\mathbf{I} - \mathbf{A})^{-1}\mathbf{b} + d] = d. \tag{6.122}$$

If the limit is 0, then there is no d term. Otherwise, the limit is the d term. From (6.118), we have that $H(z) = F(z)(1+z) = F(z) + zF(z)$. Using the pole-residue form of $F(z)$ and taking the limit as $z \to \infty$ gives

$$\begin{aligned} \lim_{z\to\infty} H(z) &= \lim_{z\to\infty} [F(z) + zF(z)] \\ &= \lim_{z\to\infty} \left(\sum_{k=1}^{n} \frac{\phi_k}{z - \mu_k} + \sum_{k=1}^{n} \frac{z\phi_k}{z - \mu_k} \right) \\ &= \sum_{k=1}^{n} \phi_k \\ \lim_{z\to\infty} H(z) &= \mathbf{c}^T \widetilde{\mathbf{b}} \end{aligned} \tag{6.123}$$

So we have that $d = \mathbf{c}^T\widetilde{\mathbf{b}}$ and the result is shown. □

Corollary 6.23. *Let H and H_r be the stable discrete-time systems created by applying Crank-Nicholson to the stable continuous-time systems G and G_r, respectively where $G_r(s)$ satisfies the first-order necessary conditions for $\mathcal{H}_2$ optimality with respect to $G(s)$. Let $F(z)$ be defined as in (6.118) and similarly define $F_r(z)$ for the system H_r. Then the following are*

true

1. *$F_r(z)$ satisfies the first order conditions for $\mathcal{H}_2$ optimality with respect to $F(z)$.*

2. *Further, if $H_r(z)$ has a pole at 0, then $H_r(z)$ satisfies the first-order necessary conditions for optimality given by Corollary 6.5 with respect to $H(z)$.*

Proof. **Result 1**

Since

$$\lim_{z\to\infty} F(z) = \lim_{z\to\infty} \sum_{k=1}^{n} \frac{\phi_k}{z-\mu_k} = 0, \tag{6.124}$$

we need to satisfy the conditions in Theorem 6.4 that

$$F(1/\mu_k) = F_r(1/\mu_k) \qquad F'(1/\mu_k) = F_r'(1/\mu_k), \tag{6.125}$$

where μ_k are the poles of $F_r(z)$ closed under conjugation. We note that the poles of $H_r(z)$ are the same as the poles of $F_r(z)$. From Theorem 6.21, we have that $H(1/\mu_k) = H_r(1/\mu_k)$. Since H_r is stable, $0 < |\mu_k| < 1$. Then

$$H\left(\tfrac{1}{\mu_k}\right) = H_r\left(\tfrac{1}{\mu_k}\right) \tag{6.126}$$

$$\implies F\left(1/\mu_k\right) + \left(\tfrac{1}{\mu_k}\right) F\left(\tfrac{1}{\mu_k}\right) = F_r\left(\tfrac{1}{\mu_k}\right) + \left(\tfrac{1}{\mu_k}\right) F_r\left(\tfrac{1}{\mu_k}\right) \tag{6.127}$$

$$\implies F\left(\tfrac{1}{\mu_k}\right)\left(1+\tfrac{1}{\mu_k}\right) = F_r\left(\tfrac{1}{\mu_k}\right)\left(1+\tfrac{1}{\mu_k}\right) \tag{6.128}$$

$$\implies F\left(\tfrac{1}{\mu_k}\right) = F_r\left(\tfrac{1}{\mu_k}\right) \tag{6.129}$$

for all μ_k, the poles of $F(z)$. Using this result and Theorem 6.21, we have

$$H'\left(\tfrac{1}{\mu_k}\right) = H_r'\left(\tfrac{1}{\mu_k}\right) \tag{6.130}$$

$$\implies F'\left(\tfrac{1}{\mu_k}\right) + \tfrac{1}{\mu_k}F'\left(\tfrac{1}{\mu_k}\right) + F\left(\tfrac{1}{\mu_k}\right) = F_r'\left(\tfrac{1}{\mu_k}\right) + \tfrac{1}{\mu_k}F_r'\left(\tfrac{1}{\mu_k}\right) + F_r\left(\tfrac{1}{\mu_k}\right) \tag{6.131}$$

$$\implies F'\left(\tfrac{1}{\mu_k}\right)\left(1+\tfrac{1}{\mu_k}\right) + F\left(\tfrac{1}{\mu_k}\right) = F_r'\left(\tfrac{1}{\mu_k}\right)\left(1+\tfrac{1}{\mu_k}\right) + F_r\left(\tfrac{1}{\mu_k}\right) \tag{6.132}$$

$$\implies F'\left(\tfrac{1}{\mu_k}\right)\left(1+\tfrac{1}{\mu_k}\right) = F_r'\left(\tfrac{1}{\mu_k}\right)\left(1+\tfrac{1}{\mu_k}\right) \tag{6.133}$$

$$\implies F'\left(\tfrac{1}{\mu_k}\right) = F_r'\left(\tfrac{1}{\mu_k}\right) \tag{6.134}$$

The first-order necessary conditions for $\mathcal{H}_2$ optimality for Theorem 6.4 are satisfied by (6.129) and (6.134) and the result is shown.

Result 2

First note that

$$\lim_{z\to 0} H(1/z) = \lim_{z\to\infty} H(z) = d. \tag{6.135}$$

Let μ_k be the poles of $H_r(z)$. From Theorem 6.21, we have $H(1/\mu_k) = H_r(1/\mu_k)$. Since $H_r(z)$ has a pole at 0, we have that

$$\lim_{z\to\infty} H(z) = \lim_{z\to\infty} H_r(z). \tag{6.136}$$

Applying Theorem 6.21 and the result in Equation (6.136), we satisfy the first-order necessary conditions from Corollary 6.5.

□

At this point, we need to address whether or not there can be a pole of $H_z(z)$ at 0. It turns out that if we choose h correctly, then $H(z)$ does indeed have a pole at 0.

Proposition 6.24. *Let G_r be the continuous-time reduced-order system given by 6.4. If h*

is chosen such that there is an eigenvalue, λ, of $\mathbf{A}_r$ such that $\lambda = -\frac{2}{h}$, then the discrete-time system H_r generated by applying Crank-Nicholson to G_r satisfies the first-order necessary conditions for $\mathcal{H}_2$ optimality in Corollary 6.5.

Proof. Let $\widetilde{\mathbf{A}}_r = \left(I - \frac{h}{2}\mathbf{A}_r\right)^{-1}\left(I + \frac{h}{2}\mathbf{A}_r\right)$. Then we can see that the poles of $H_r(z)$ are the eigenvalues of $\widetilde{\mathbf{A}}_r$. Given the eigenvalues, λ, of $\mathbf{A}_r$ we can calculate the eigenvalues, μ, of $\widetilde{\mathbf{A}}_r$ as follows

$$\mu = \left(1 - \tfrac{h}{2}\lambda\right)^{-1}\left(1 + \tfrac{h}{2}\lambda\right) \tag{6.137}$$

Suppose that $\lambda = -\frac{h}{2}$ for some eigenvalue of $\mathbf{A}_r$. Then

$$\mu = \left(1 - \tfrac{h}{2}\left(-\tfrac{2}{h}\right)\right)^{-1}\left(1 + \tfrac{h}{2}\left(-\tfrac{2}{h}\right)\right), \tag{6.138}$$

$$= \frac{1-1}{1+1}, \tag{6.139}$$

$$= 0. \tag{6.140}$$

Therefore, in this case, there is a zero pole of $H_r(z)$ and the result follows directly from Corollary 6.23. □

To conclude this section, we look at the optimality preservation given by Carlberg et al. [26]. We note that the optimality in [26] refers to the least-squares optimality tailored toward approximating an observed trajectory and is input-dependent. In [26], the focus is on showing when the discretized optimal least-squares Galerkin projection, i.e. POD, is equivalent to the optimal least-squares Petrov-Galerkin projection performed on the discretized system. For our research, both the continuous-time and discrete-time $\mathcal{H}_2$-optimal projections are Petrov-Galerkin projections, and we show the conditions under which our input-independent $\mathcal{H}_2$-optimality conditions are preserved for linear dynamical systems. This optimality condition is in terms of the transfer function and is valid for any bounded input. Carlberg et al. [26]

showed that to preserve the least-squares optimality when discretizing the system, there are certain restrictions on the continuous-time system or the discretization scheme. They demonstrated that for linear systems, any multi-step method will preserve the least-squares optimality. For nonlinear systems discretized using a multi-step scheme, Carlberg et al. [26] show that all single-step methods and explicit higher order methods preserve the least-squares optimality. In our setting, for $\mathcal{H}_2$-optimality conditions to be preserved, we found that we needed a method that would provide a conformal map from the left half of the complex plane to the unit disk. We then had to control the time step h in order to match the behavior of the constant d term. In analogy, this is in direct agreement with [26] where it was shown that to decrease the error for the least-squares Petrov-Galerking ROM, the time step should be matched to the spectral content of the reduced basis. The important information here is that in order to maintain optimality in a ROM, one must be cautious in how the resulting time-dependent ODE system is solved. Regardless of the reduction technique chosen, the effectiveness of that technique, with respect to the goals of the reduced-order model, can be reduced by a poor choice of the time-discretization scheme.

6.5 Numerical Results

To investigate the relationships from the previous section, we ran several tests to numerically verify the theorems. We studied two separate linear dynamical systems; the 1D heat equation with boundary control and a flex control system from the international space station.

We demonstrate the Crank-Nicolson method does indeed preserve the Hermite interpolation of the full-order transfer function regardless of whether the Crank-Nicolson method discretization is applied to the continuous-time ROM or the ROM is built from the full-order Crank-Nicolson method discretization. Further we show the forward and backward Euler

Table 6.1: Relative error between $H_r(1/\mu_k)$ and $H(1/\mu_k)$ where μ_k are the poles of the discrete time transfer function $H_r(z)$

	Relative Error		
Method	Heat Equation ($r = 6$)	Heat Equation ($r = 10$)	ISS12A ($r = 40$)
Forward Euler	4.4433e-06	1.5256e-02	1.3726e-01
Backward Euler	1.2598e-06	2.8978e-09	5.0455e-03
Crank-Nicolson	2.4769e-13	2.2393e-13	4.5262e-10

methods do not preserve this Hermite interpolation. However, while not $\mathcal{H}_2$ optimal, these methods may still provide an adequate approximation for the system.

6.5.1 1D Heat Equation

For this test, we examined the 1D heat equation on the spatial domain $\Omega = [0, 1]$, and time domain $t \in [0, 10]$. The PDE model describes the temperature T over the domain as follows

$$\frac{\partial \boldsymbol{\theta}(t,x)}{\partial t} = \alpha \frac{\partial^2 \boldsymbol{\theta}(t,x)}{\partial x^2}, \tag{6.141}$$

$$\boldsymbol{\theta}(t,0) = 0, \tag{6.142}$$

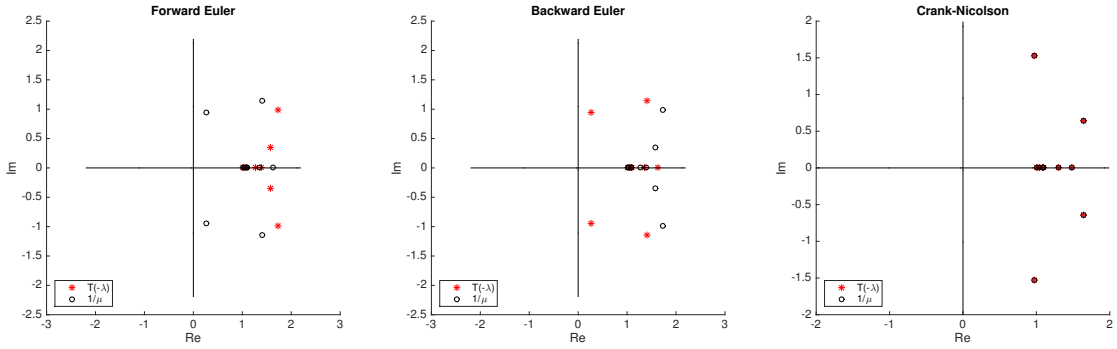
$$\boldsymbol{\theta}(t,1) = u(t), \tag{6.143}$$

$$\boldsymbol{\theta}(0,x) = \boldsymbol{\theta}_0(x), \tag{6.144}$$

where $x \in \Omega$, $\boldsymbol{\theta}_0(x)$ is the initial state, there is a zero Dirichlet boundary condition at $x = 0$, and there is a Dirichlet input $u(t)$ at $x = 1$. The thermal diffusivity constant was set to $\alpha = 0.02$.

Further, we define the output to be the value of $\boldsymbol{\theta}$ at $x = 0.5$

$$y(t) = \boldsymbol{\theta}(t, 0.5). \tag{6.145}$$



(a) Forward Euler (b) Backward Euler (c) Crank-Nicolson

Figure 6.3: Plot $T(-\lambda_k)$ and $1/\mu_k$ for the reduced heat equation model.

Applying a finite element discretization to the spatial domain, we obtained a SISO linear dynamical system G as in (6.61). We then used an $\mathcal{H}_2$-optimal IRKA to produce the r-dimensional ROM G_r. Each of the time-discretization methods was applied to G_r and G producing H_r and H, respectively. By a common abuse of notation, we will refer to the dynamical system that each transfer function represents by the same letter, e.g. $G(s)$ is the transfer function for the dynamical system G.

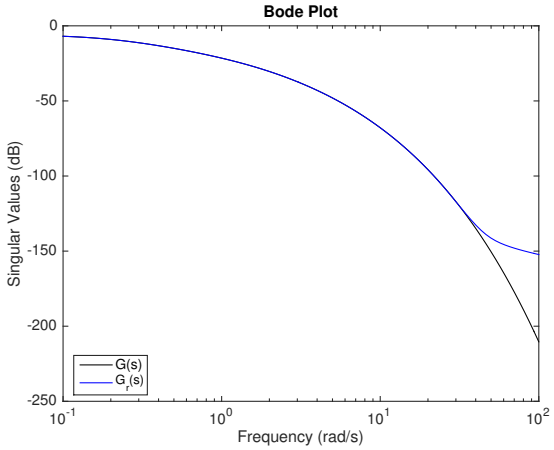


Figure 6.4: Heat equation Bode plot comparing the FOM and ROM

One method for looking at the complexity of the underlying model is to examine the Bode plot which gives the frequency response of the transfer function along the imaginary axis. Figure 6.4 gives a Bode plot for the various transfer functions associated with the heat

equation. We see that the response curve is fairly simple. We will contrast this later with the flex control system model for the ISS12A module.

Since G_r is an $\mathcal{H}_2$-optimal ROM for G, $G_r(s)$ is a Hermite interpolant of $G(s)$ at $-\lambda_k$, $k = 1, \ldots, r$ where λ_k are the poles of $G_r(s)$. From Theorems 6.15, 6.16, and 6.17, we know that the following relationship holds for each of the discretization methods

$$H_r(T(-\lambda)) = G_r(-\lambda) = G(-\lambda) = H(T(-\lambda)), \tag{6.146}$$

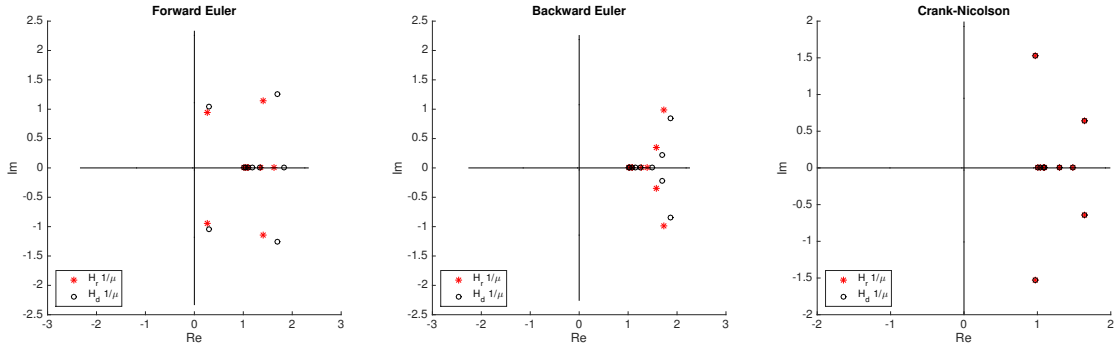
$$H_r'(T(-\lambda)) = G_r'(-\lambda) = G'(-\lambda) = H'(T(-\lambda)), \tag{6.147}$$

where $T : S \to Z$ as given by (6.67), (6.74), or (6.81) depending on the time-discretization method used. However, for H_r to be an $\mathcal{H}_2$-optimal reduced-order model for H, then $H_r(z)$ must be a Hermite interpolant of $H(z)$ at $\frac{1}{\mu_k}$, $k = 1, \ldots, r$ where μ_k are the poles of $H_r(z)$. Our first test is to see if

$$T(-\lambda_k) = \frac{1}{\mu_k} \tag{6.148}$$

for $k = 1, \ldots, r$ since (6.146)-(6.147) would imply that H_r is an $\mathcal{H}_2$-optimal ROM for H. It is clear from Figure 6.3, that (6.148) holds for the Crank-Nicolson method, but not for the forward or backward Euler methods. This supports the theory from Section 6.4.

Even though (6.148) did not hold, we were still interested in whether or not the Hermite interpolation would still hold. The error between $H_r(\frac{1}{\mu})$ and $H(\frac{1}{\mu})$ is shown in Table 6.2. We see that the error for the Crank-Nicolson method is effectively machine precision for the heat equation, whereas $H_r(\frac{1}{\mu})$ and $H(\frac{1}{\mu})$ do not match for the Euler methods. Further, Figure 6.5 illustrates how the value at the optimal interpolation points for the full and reduced-order discrete-time systems generated using the Crank-Nicolson method are essentially equal and all lie on the real axis, but for the Euler methods, some of the evaluations of the transfer function are even off the real axis.



(a) Forward Euler (b) Backward Euler (c) Crank-Nicolson

Figure 6.6: Plot $1/\mu_k$ from $H_r(z)$ and $H_d(z)$ for the heat equation model.

Next, we created an $\mathcal{H}_2$-optimal ROM, H_d, using the discrete time version of IRKA. We evaluated whether or not the $\mathcal{H}_2$-optimal interpolation points from $H_r(z)$ matched the $\mathcal{H}_2$-optimal interpolation points from $H_d(z)$. Figure 6.6 again supports the theory that these points match when using the Crank-Nicolson method, but do not when using the other two methods.

h	d	d_d	Relative Error
0.25748	5.7409030610182e-05	5.7409030610183e-05	1.794129e-14
0.05000	2.0851634758729e-10	1.5674850145348e-10	2.482676e-01

Table 6.2: Relative error between d and d_r for the heat equation.

Finally, we numerically validated the optimality conditions from Corollary 6.23 and Proposition 6.22 for the 1D heat equation when discretizing the system using Crank-Nicholson. For the continuous-time reduced-order model G_r with size $r = 10$, there was a real eigenvalue of $\mathbf{A}_r$ at $\lambda \approx -7.767$. Proposition 6.22 suggests that if we select $h = -2/\lambda$ for some eigenvalue of A_r then we should have a zero pole in $H_r(z)$. Indeed, when we selected $h = -2/\lambda \approx 0.25748$, there is a pole of $H_r(z)$ at zero. We then selected $h = 0.05$ since there are no eigenvalues *lambda* of $\mathbf{A}_r$ close to $\lambda = -2/h$. In Table 6.2, we compare the d_r terms for the reduced order discrete-time systems H_r to the d terms for the full-order discrete-time

systems H. As can be seen, by choosing the h appropriately, we are able to match d terms within machine precision. When we do not select the h according to the eigenvalues of A_r, then we do not match the d terms.

6.5.2 International Space Station Control (ISS12A)

The International Space Station (ISS) is a complex structure made up of several subassembly modules that were brought to space over the course of several missions of the Space Shuttle. The ISS Assembly Mission 12A brought, among other items, the second port truss segment. The flex of this module is tracked by 1412 states. [4] We focused on a single output and single out to the control system for this module. We will refer to this control system as the ISS12A model.

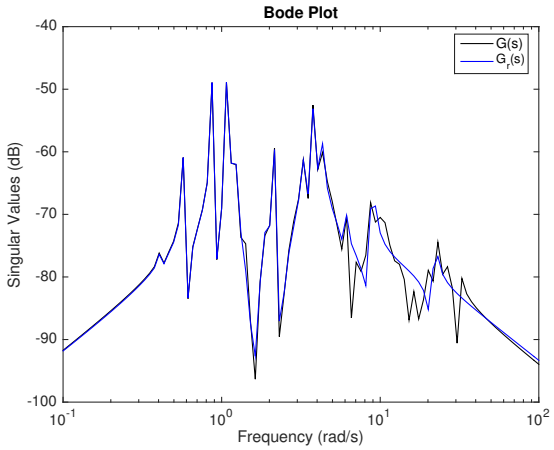
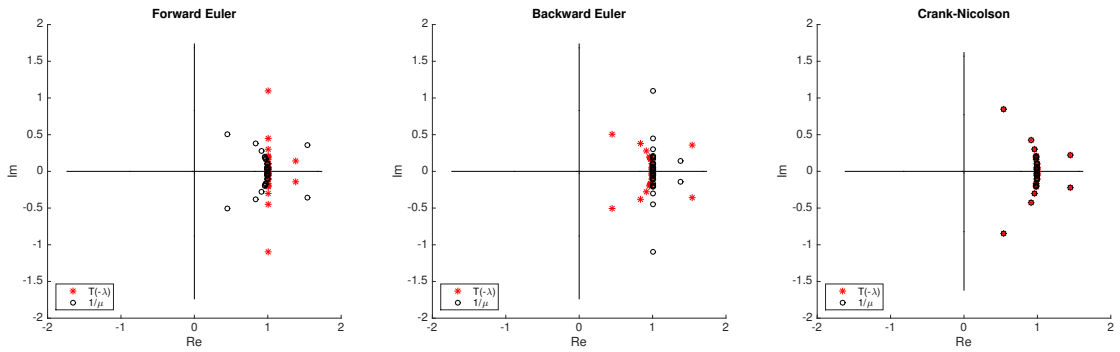


Figure 6.7: ISS12A Bode plot comparing the FOM and ROM

This model is much more challenging to discretize in time and reduce than the heat equation modeled earlier. First, the eigenvalues of the system are complex and close to the imaginary axis. Thus, the system is much more susceptible to perturbations than the heat equation. Further, since the eigenvalues are relatively large and close to the imaginary axis, the time step size required for the Forward Euler method to create a stable discrete-time model is be

prohibitively small to create a stable discrete-time model. Also, as seen in 6.7, the frequency response for the transfer function of the ISS12A is much more complex than the heat equation model as seen in 6.4.



(a) Forward Euler (b) Backward Euler (c) Crank-Nicolson

Figure 6.8: Plot $T(-\lambda_k)$ and $1/\mu_k$ for the ISS12A space station model.

We followed the same series of tests for this model as we did for the heat model. In particular, we looked to see if Equation (6.148) was true for $k = 1, \ldots, r$ since (6.146)-(6.147) would imply that H_r is an $\mathcal{H}_2$-optimal ROM for H. We can see in Figure 6.8, that the Crank-Nicolson method provides a good match, whereas for the other two methods $1/\mu_k$ and $T(-\lambda_k)$ are not a good match at all. In fact, they are much worse than we saw with the heat equation. This has to to with the fact that T for the Crank-Nicolson method maps the imaginary axis to the unit circle.

As with the heat equation, we are interested in whether or not $H_r(z)$ is a Hermite interpolant of $H(z)$ at the optimal points $1/\mu_k$ where μ_k are the poles of $H_r(z)$. We see in Table 6.2, that, as with the heat equation, the Crank-Nicolson matches the function values at the interpolation points much better.

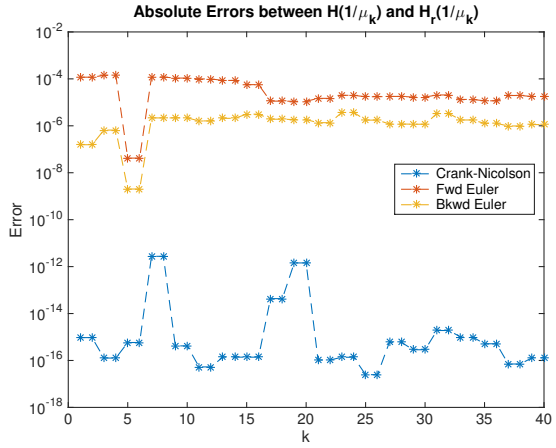
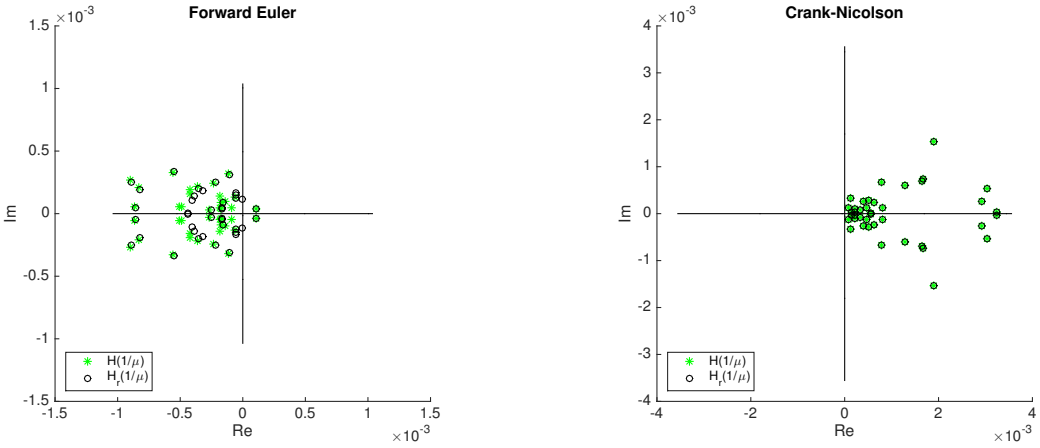


Figure 6.9: Absolute error between $H(1/\mu_k)$ and $H_r(1/\mu_k)$ for the ISS12A control system.

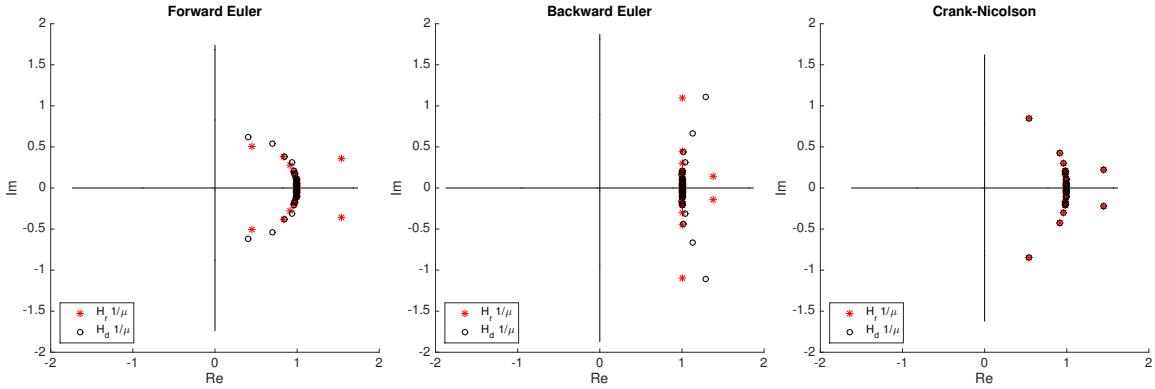
Further, when looking at the absolute errors at each interpolation point in Figure 6.9, we see that the error for the Crank-Nicolson method at each point is close to machine epsilon (1×10^{-16}) for almost all the interpolation points. However, for the other two methods, the error is several magnitudes higher. We can see how these differ for the Crank-Nicolson method and Forward Euler method in Figure 6.10. There is a pronounced shift in the values between the two transfer functions at the interpolation points for the Forward Euler method. In contrast, for the Crank-Nicolson method, the function values are visibly identical.



(a) Forward Euler (b) Crank-Nicolson

Figure 6.10: Plot of $H_r(1/\mu_k)$ and $H(1/\mu_k)$ for ISS12A, where μ_k are the poles of $H_r(z)$

Finally, we created an $\mathcal{H}_2$-optimal reduced-order model from the full-order discrete-time system and compared its transfer function $H_d(z)$ to the transfer function from the discrete-time system built from the Htw-optimal continuous-time reduced-order model, $H_r(z)$. Specifically, we looked at the optimal interpolation points, $1/\mu_k$, for each discrete-time ROM, where μ_k are the $k = 1, \ldots, r$ poles of the ROM transfer functions.



(a) Forward Euler (b) Backward Euler (c) Crank-Nicolson

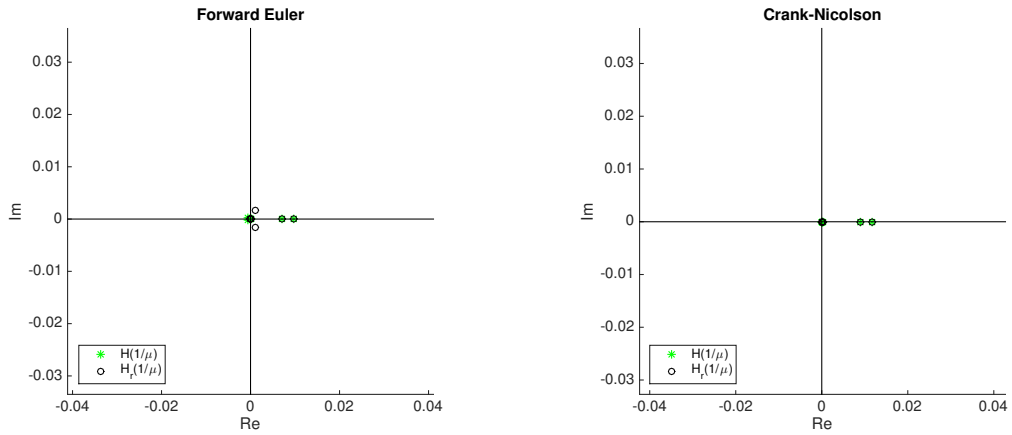
Figure 6.11: Plot $1/\mu_k$ from $H_r(z)$ and $H_d(z)$ for the ISS12A space station model.

In Figure 6.11, we plot these optimal points for each method, comparing how changing the order of reducing the model and discretizing in time affects the optimal interpolation points. For the Crank-Nicolson method, the order does not matter, but for the Euler methods, switching the order has a significant impact on the positioning of the interpolation points.

6.5.3 Summary

We see that numerical results support the theory presented in 6.19, 6.19, and 6.21. For the Crank-Nicolson method we have seen that both branches in Figure 6.1 are equivalent. We have also seen that care must be taken when choosing discretization methods to apply to the $\mathcal{H}_2$-optimal continuous-time ROM, because the method chosen may remove the first-order necessary conditions needed for $\mathcal{H}_2$ optimality of the fully discretized model. We have

focused on single-step methods in this chapter, but there may be extensions to higher order methods that should be investigated in the future.



(a) Forward Euler (b) Crank-Nicolson

Figure 6.5: Plot of $H_r(1/\mu_k)$ and $H(1/\mu_k)$ for the heat equation, where μ_k are the poles of $H_r(z)$. Plots zoomed to show details near origin.

Chapter 7

Conclusions and Future Research

In this dissertation, we studied and developed reduced-order modeling techniques leading towards an effective model reduction of fire models. While there is still research to be done in this area, this research yielded several new techniques and mathematical ideas for model reduction. In particular, we developed a new method for providing model reduction of a system without needing sample data sets. We showed that by combining DEIM with POD for wildland fire models, we could significantly improve performance of the reduced-order models. We examined the time-discretization of time-dependent ODE models and showed when the first-order necessary conditions for $\mathcal{H}_2$-optimality of the reduced-order model were preserved in the associated discrete-time system and when they were not.

7.1 Conclusions

We began by giving an overview of the mathematical models and model reduction techniques that are used throughout the dissertation. We came back to our initial mathematical model of a fire throughout the discussion to show how each of these techniques were connected back

to that model. From this seminal effort, we contributed three major advancements leading to model reduction for fire-related models.

Our first contribution was in the development of the IRKA V⊕W method. We showed that given certain nonlinear systems, we could ignore the nonlinear portion of the system when building the reduced-order model, and still perform better than POD. This is remarkable since the POD method requires that full-order data snapshots are created *a priori*, which can be expensive. However, for this method, we only require the dynamical system, i.e. no data snapshots are needed. This significantly reduces the off-line costs. Further, since there is no dependency on a specific set of data, this method performs much better over a wider range of input conditions. This is an especially effective technique to provide an input to output map for a nonlinear dynamical system (2.17), as we would use in a control system or as a detail node in a network model.

The second major contribution is in improving the numerical computation times for wildland fire-spread models. By using Q-DEIM on these models, we were able to reduce the computation times by an order of magnitude over POD models and by 2-3 orders of magnitude over full-order models, while still providing high-fidelity output results. In our research, the error associated with the POD/Q-DEIM reduced-order models were typically as accurate as the POD models of the same size. This is a significant development in the field of wildland fire modeling.

Finally, we have proved that the Crank-Nicolson method for time-discretization of an ODE model preserves the first-order necessary conditions for $\mathcal{H}_2$-optimality and thus ensures that the fully-discretized reduced-order model retains the property of being at least a local minimum with respect to the $\mathcal{H}_2$ norm. Further, we proved that the forward and backward Euler methods do not preserve this property. This is crucial information when attempting to solve a reduced-order model developed using IRKA. It specifically says that you should not use

the forward or backward Euler method to solve a ROM that was created using IRKA, rather the Crank-Nicholson method should be used.

7.2 Future Work

We have identified four major areas that should be explored in the near future: 1) further testing of IRKA V⊕W, 2) adding combustion to the current reduced-order plume-fire models, 3) extending the wildland fire-spread models to include two-dimensional domains and variable initial fuel conditions, and 4) extending the work on discrete-time systems to identify other methods that preserve the first-order necessary conditions for $\mathcal{H}_2$ optimality.

For the IRKA V⊕W technique, we have currently only tested this against the Burgers' equation. There needs to be continued research to determine to what larger class of problems this method can be extended. Further, while in general we know that this method performs best when the number of $\mathbf{V}$ and $\mathbf{W}$ vectors are balanced, we would like to determine a specific criteria that determines the number of vectors from $\mathbf{V}$ and $\mathbf{W}$ that should be used.

To continue to improve the plume-fire models, we need to incorporate the combustion terms. As discussed in the background section, this requires the tracking of lumped species and the determination of whether or not the fuel-air concentration is sufficient for combustion to take place. Since, this behavior is localized to each cell, it is difficult to replicate at the ROM level.

We plan to couple the species together to preserve the property that all the species must sum to one. Recalling the fire-model equations for species transport (2.4)-(2.6), the full-order solutions have the following property

$$Z_A(\mathbf{x}, t) + Z_F(\mathbf{x}, t) + Z_P(\mathbf{x}, t) = 1, \tag{7.1}$$

for all $t \in [0, t_f]$ and $\mathbf{x} \in \Omega$ for the domain Ω. We would like to preserve this property in the ROM. Using the method described in Section 2.3.3, we first calculate the mean of the data snapshots, $\bar{Z}_A$, $\bar{Z}_F$, and $\bar{A}_P$, for the mass fractions of air, fuel, and products, respectively, as given by

$$\bar{Z}_A(\mathbf{x}) = \frac{1}{N}\sum_{k=1}^{N} Z_A(\mathbf{x}, t_k), \tag{7.2}$$

$$\bar{Z}_F(\mathbf{x}) = \frac{1}{N}\sum_{k=1}^{N} Z_F(\mathbf{x}, t_k), \tag{7.3}$$

$$\bar{Z}_P(\mathbf{x}) = \frac{1}{N}\sum_{k=1}^{N} Z_P(\mathbf{x}, t_k), \tag{7.4}$$

where $0 = t_1, t_2, \ldots, t_N = t_f$ are the N evenly distributed time steps from 0 to t_f where the data snapshots are captured. We see that the mean values preserve the property given in (7.1).

$$\bar{Z}_A(\mathbf{x}) + \bar{Z}_F(\mathbf{x}) + \bar{Z}_P(\mathbf{x}) = \frac{1}{N}\sum_{k=1}^{N} Z_A(\mathbf{x}, t_k) + \frac{1}{N}\sum_{k=1}^{N} Z_F(\mathbf{x}, t_k) + \frac{1}{N}\sum_{k=1}^{N} Z_P(\mathbf{x}, t_k), \tag{7.5}$$

$$= \frac{1}{N}\sum_{k=1}^{N} \left(Z_A(\mathbf{x}, t_k) + Z_F(\mathbf{x}, t_k) + Z_P(\mathbf{x}, t_k)\right), \tag{7.6}$$

$$= \frac{1}{N}\sum_{k=1}^{N} 1 = 1. \tag{7.7}$$

Now, we can define our variable of interest as the deviation from the mean for each of the species by

$$\widehat{Z}_A(\mathbf{x}, t) = Z_A(\mathbf{x}, t) - \bar{Z}_A(\mathbf{x}), \tag{7.8}$$

$$\widehat{Z}_F(\mathbf{x}, t) = Z_F(\mathbf{x}, t) - \bar{Z}_F(\mathbf{x}), \tag{7.9}$$

$$\widehat{Z}_P(\mathbf{x}, t) = Z_P(\mathbf{x}, t) - \bar{Z}_P(\mathbf{x}), \tag{7.10}$$

The sum of the deviations for $t = [0, t_f]$ must then be zero as seen by

$$\begin{aligned}\widehat{Z}_A(\mathbf{x},t) + \widehat{Z}_F(\mathbf{x},t) + \widehat{Z}_P(\mathbf{x},t) &= [Z_A(\mathbf{x},t) - \bar{Z}_A(\mathbf{x})] + [Z_F(\mathbf{x},t) - \bar{Z}_F(\mathbf{x})] + [Z_P(\mathbf{x},t) - \bar{Z}_P(\mathbf{x})], \\ &= Z_A(\mathbf{x},t) + Z_F(\mathbf{x},t) + Z_P(\mathbf{x},t) - [\bar{Z}_A(\mathbf{x}) + \bar{Z}_F(\mathbf{x}) + \bar{Z}_P(\mathbf{x})], \\ &= 1 - 1 = 0. \end{aligned} \tag{7.11}$$

Let $\widehat{\mathbf{Z}}(\mathbf{x},t) = [\widehat{Z}_A(\mathbf{x},t), \widehat{Z}_F(\mathbf{x},t), \widehat{Z}_P(\mathbf{x},t)]^T$. We then define the associated $3r$-dimensional POD basis as $\Phi(\mathbf{x}) = [\phi^{(A)}(\mathbf{x}), \phi^{(F)}(\mathbf{x}), \phi^{(P)}(\mathbf{x})]^T$. This leads to the approximation

$$\widehat{\mathbf{Z}}(\mathbf{x},t) \approx \widehat{\mathbf{Z}}_r(\mathbf{x},t) = \sum_{i=1}^{r} \Phi_i(\mathbf{x})\eta_i(t). \tag{7.12}$$

Since the POD basis preserves the linearity properties, we are guaranteed to maintain the zero property given in (7.11) in the ROM. This, in turn, ensures that the sum of the species created using the ROM will sum to one. Additionally, we are investigating methods to ensure positive solutions for all species. We currently believe that by restricting the coefficients to a certain manifold, we can ensure that the solutions will remain positive. By combining these two techniques, we should be able to ensure that the mass fraction of each species will remain between zero and one. Once the species is in the correct band, we will be able to implement the combustion. We note that the nonlinear behavior associated with combustion is no longer quadratic. Therefore, we must look to other techniques, such as Q-DEIM, to keep the computation times from growing to an impractical level.

With the significant improvements to the reduced-order wildland fire-spread model computation times, we seek to extend the research in a few specific ways. First we will extend our research to two-dimensional models. Next, we will investigate methods to handle discontinuous fuel and geographic properties. This may require "stitching" several models together at the boundaries. In practical terms, these conditions could occur when transitioning between

fuel sources, such as moving from fields to forests, or across geographic formations such as cliffs or mountains. By creating several ROMs specifically tailored to handling certain fuel loadings and geographic conditions, we could piece them together to simulate a larger domain.

We have shown that the Crank-Nicholson method preserves the first-order necessary conditions for $\mathcal{H}_2$ optimality. However, there may be times when we need to use a method with better convergence properties. We will extend our current research to see what other methods may also preserve the first-order Necessary conditions. Specifically, we will investigate other multi-step methods and Runge-Kutta methods to see when those methods preserve the first-order necessary conditions for $\mathcal{H}_2$ optimality.

Bibliography

[1] A. S. Ackleh, E. J. Allen, R. B. Kearfott, and P. Seshaiyer. *Classical and Modern Numerical Analysis: Theory, Methods and Practice*. CRC Press, 2009.

[2] S. S. An, T. Kim, and D. L. James. Optimizing cubature for efficient integration of subspace deformations. In *ACM Transactions on Graphics (TOG)*, volume 27, page 165. ACM, 2008.

[3] H. Antil, M. Heinkenschloss, and D. C. Sorensen. Application of the discrete empirical interpolation method to reduced order modeling of nonlinear and parametric systems. In *Reduced Order Methods for Modeling and Computational Reduction*, pages 101–136. Springer, 2014.

[4] A. C. Antoulas. *Approximation of Large-Scale Dynamical Systems (Advances in Design and Control)*. SIAM, 2005.

[5] A. C. Antoulas, D. C. Sorensen, and S. Gugercin. A survey of model reduction methods for large-scale systems. *Contemporary Mathematics*, 280:193–220, 2001.

[6] A. C. Antoulas, C. A. Beattie, and S. Gugercin. Interpolatory model reduction of large-scale dynamical systems. In *Efficient Modeling and Control of Large-Scale Systems*, pages 3–58. Springer, 2010.

[7] P. Astrid, S. Weiland, K. Willcox, and T. Backx. Missing point estimation in models described by proper orthogonal decomposition. *Automatic Control, IEEE Transactions on*, 53(10):2237–2251, 2008.

[8] J. A. Atwell. *Proper Orthogonal Decomposition for Reduced Order Control of Partial Differential Equations*. PhD thesis, Virginia Polytechnic Institute and State University, 2000.

[9] J. A. Atwell and B. B. King. Reduced order controllers for spatially distributed systems via proper orthogonal decomposition. 1999.

[10] Z. Bai and D. Skoogh. A projection method for model reduction of bilinear dynamical systems. *Linear Algebra and its Applications*, 415(2):406–425, 2006.

[11] M. Barrault, Y. Maday, N. C. Nguyen, and A. T. Patera. An 'empirical interpolation' method: Application to efficient reduced-basis discretization of partial differential equations. *Comptes Rendus Mathematique*, 339(9):667–672, 2004.

[12] U. Baur, C. Beattie, P. Benner, and S. Gugercin. Interpolatory projection methods for parameterized model reduction. *SIAM Journal on Scientific Computing*, 33(5):2489–2518, 2011.

[13] U. Baur, P. Benner, and L. Feng. Model order reduction for linear and nonlinear systems: A system-theoretic perspective. *Archives of Computational Methods in Engineering*, 21 (4):331–358, 2014.

[14] P. Benner and T. Breiten. Interpolation-based $\mathcal{H}_2$ model reduction of bilinear control systems. *SIAM Journal on Matrix Analysis and Applications*, 33(3):859–885, 2012.

[15] P. Benner and T. Breiten. Two-sided projection methods for nonlinear model order reduction. *SIAM Journal on Scientific Computing*, 37(2):B239–B260, 2015.

[16] P. Benner and P. Goyal. Algebraic Gramians for quadratic-bilinear systems and their application in model order reduction. In *Proceedings of the 22nd International Symposium on Mathematical Theory of Networks and Systems*, 2016.

[17] P. Benner, S. Gugercin, and K. Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM Review*, 57(4):483–531, 2015.

[18] G. Berkooz, P. Holmes, and J. L. Lumley. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual Review of Fluid Mechanics*, 25(1):539–575, 1993.

[19] R. Bos, X. Bombois, and P. Van den Hof. Accelerating large-scale non-linear models for monitoring and control using spatial and temporal correlations. In *American Control Conference, 2004. Proceedings of the 2004*, volume 4, pages 3705–3710. IEEE, 2004.

[20] D. J. Brake. Fire modelling in underground mines using Ventsim Visual VentFIRE Software. Australian Mine Ventilation Conference/Adelaide, SA, Australia, 2013.

[21] T. Breiten and T. Damm. Krylov subspace methods for model order reduction of bilinear control systems. *Systems & Control Letters*, 59(8):443–450, 2010.

[22] A. Bunse-Gerstner, D. Kubalińska, G. Vossen, and D. Wilczek. H_2-norm optimal model reduction for large scale discrete dynamical MIMO systems. *Journal of Computational and Applied Mathematics*, 233(5):1202–1216, 2010.

[23] M. Şuvar, D. Cioclea, I. Gherghe, and V. Păsculescu. Advanced software for mine ventilation networks solving. *Environmental Engineering and Management Journal*, 11 (7):1235–1239, 2012.

[24] K. Carlberg, C. Bou-Mosleh, and C. Farhat. Efficient non-linear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations. *International Journal for Numerical Methods in Engineering*, 86(2):155–181, 2011.

[25] K. Carlberg, J. Cortial, C. Farhat, and D. Amsallem. The GNAT nonlinear model-reduction method with application to large-scale turbulent flows. Technical report, Sandia National Laboratories (SNL-CA), Livermore, CA (United States), 2013.

[26] K. Carlberg, M. Barone, and H. Antil. Galerkin v. discrete-optimal projection in non-linear model reduction. *arXiv preprint arXiv:1504.03749*, 2015.

[27] S. Chaturantabut and D. C. Sorensen. Discrete empirical interpolation for nonlinear model reduction. In *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pages 4316–4321. IEEE, 2009.

[28] S. Chaturantabut and D. C. Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal on Scientific Computing*, 32(5):2737–2764, 2010.

[29] S. Chaturantabut, C. Beattie, and S. Gugercin. Structure-preserving model reduction for nonlinear port-Hamiltonian systems. *arXiv preprint arXiv:1601.00527*, 2016.

[30] L. H. Cheng, T. H. Ueng, and C. W. Liu. Simulation of ventilation and fire in the underground facilities. *Fire Safety Journal*, 36(6):597–619, Sept. 2001.

[31] A. J. Chorin, J. E. Marsden, and J. E. Marsden. *A Mathematical Introduction to Fluid Mechanics*, volume 3. Springer, 1990.

[32] R. Curtain and K. Morris. Transfer functions of distributed parameter systems: A tutorial. *Automatica*, 45(5):1101–1116, May 2009.

[33] R. F. Curtain and H. Zwart. *An Introduction to Infinite-Dimensional Linear Systems Theory*, volume 21. Springer, 1995.

[34] C. de Villemagne and R. E. Skelton. Model reductions using a projection formulation. *International Journal of Control*, 46(6):2141–2169, 1987.

[35] Z. Drmac and S. Gugercin. A new selection operator for the discrete empirical interpolation method–improved a priori error bound and extensions. *arXiv preprint arXiv:1505.00370*, 2015.

[36] M. Drohmann, B. Haasdonk, and M. Ohlberger. Reduced basis approximation for nonlinear parametrized evolution equations based on empirical operator interpolation. *SIAM Journal on Scientific Computing*, 34(2):A937–A969, 2012.

[37] H. C. Elman, D. J. Silvester, and A. J. Wathen. *Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics*. Oxford University Press, 2014.

[38] R. Everson and L. Sirovich. Karhunen–Loeve procedure for gappy data. *JOSA A*, 12 (8):1657–1664, 1995.

[39] G. Flagg and S. Gugercin. Multipoint Volterra series interpolation and $\mathcal{H}_2$ optimal model reduction of bilinear systems. *SIAM Journal on Matrix Analysis and Applications*, 36(2):549–579, 2015.

[40] F. J. Flanigan. *Complex Variables: Harmonic and Analytic Functions*. Courier Corporation, 1972.

[41] D. Gaier. *Lectures on Complex Approximation*. Springer, 1987.

[42] W. Gautschi. *Numerical analysis*. Springer Science & Business Media, 2011.

[43] B. Gebhart, Y. Jaluria, R. L. Mahajan, and B. Sammakia. *Buoyancy-Induced Flows and Transport*. 1988.

[44] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The John Hopkins University Press, 1996.

[45] M. Green and D. J. N. Limebeer. *Linear Robust Control*. Courier Corporation, 2012.

[46] E. J. Grimme. *Krylov Projection Methods for Model Reduction*. PhD thesis, Citeseer, 1997.

[47] C. Gu. *Model Order Reduction of Nonlinear Dynamical Systems*. PhD thesis, University of California, Berkeley, 2011.

[48] C. Gu. QLMOR: A projection-based nonlinear model order reduction approach using quadratic-linear representation of nonlinear systems. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 30(9):1307–1320, 2011.

[49] E. Guelpa, A. Sciacovelli, V. Verda, and D. Ascoli. Model reduction approach for wildfire multi-scenario analysis. *Parte: http://hdl. handle. net/10316.2/34013*, 2014.

[50] S. Gugercin. *Projection Methods for Model Reduction of Large-Scale Dynamical Systems*. Thesis, Rice University, 2003.

[51] S. Gugercin and A. C. Antoulas. A comparative study of 7 algorithms for model reduction. In *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, volume 3, pages 2367–2372, 2000.

[52] S. Gugercin, C. Beattie, and A. C. Antoulas. Rational Krylov methods for optimal $\mathcal{H}_2$ model reduction. *ICAM Technical Report*, 2006.

[53] S. Gugercin, A. C. Antoulas, and C. Beattie. $\mathcal{H}_2$ model reduction for large-scale linear dynamical systems. *SIAM Journal on Matrix Analysis and Applications*, 30(2):609–638, 2008.

[54] S. Gugercin, T. Stykel, and S. Wyatt. Model reduction of descriptor systems by interpolatory projection methods. *SIAM Journal on Scientific Computing*, 35(5):B1010–B1033, 2013.

[55] M. D. Gunzburger. *Finite Element Methods for Viscous Incompressible Flows: A Guide to Theory, Practice, and Algorithms*. Elsevier, 2012.

[56] A. Hay, J. T. Borggaard, and D. Pelletier. Local improvements to reduced-order models using sensitivity analysis of the proper orthogonal decomposition. *Journal of Fluid Mechanics*, 629:41–72, 2009.

[57] A. Hay, I. Akhtar, and J. T. Borggaard. On the use of sensitivity analysis in model reduction to predict flows for varying inflow conditions. *International Journal for Numerical Methods in Fluids*, 68(1):122–134, 2012.

[58] F. Incropera and D. DeWitt. *Introduction to Heat Transfer*. 1985.

[59] C. Johnson. *Numerical Solution of Partial Differential Equations by the Finite Element Method*. Courier Corporation, 2012.

[60] G. Kerschen and J.-C. Golinval. Physical interpretation of the proper orthogonal modes using the singular value decomposition. *Journal of Sound and Vibration*, 249(5):849–865, 2002.

[61] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

[62] B. Krämer. *Model Reduction of the Coupled Burgers Equation in Conservation Form*. PhD thesis, Virginia Polytechnic Institute and State University, 2011.

[63] K. Kunisch and S. Volkwein. Galerkin proper orthogonal decomposition methods for parabolic problems. *Numerische Mathematik*, 90(1):117–148, 2001.

[64] K. Kunisch and S. Volkwein. Galerkin proper orthogonal decomposition methods for a general equation in fluid dynamics. *SIAM Journal on Numerical analysis*, 40(2): 492–515, 2002.

[65] W. Layton. *Introduction to the Numerical Analysis of Incompressible Viscous Flows*, volume 6. SIAM, 2008.

[66] P. A. LeGresley. *Application of Proper Orthogonal Decomposition (POD) to Design Decomposition Methods*. PhD thesis, Citeseer, 2005.

[67] J. Leyva-Ramos. Partial-fraction expansion in system analysis. *International Journal of Control*, 53(3):619–639, 1991.

[68] I. S. Lowndes, S. A. Silvester, D. Giddings, S. Pickering, A. Hassan, and E. Lester. The computational modelling of flame spread along a conveyor belt. *Fire Safety Journal*, 42 (1):51–67, Feb. 2007.

[69] J. L. Lumley. The structure of inhomogeneous turbulent flows. *Atmospheric Turbulence and Radio Wave Propagation*, pages 166–178, 1967.

[70] H. V. Ly and H. T. Tran. Modeling and control of physical processes using proper orthogonal decomposition. *Mathematical and Computer Modelling*, 33(1):223–236, 2001.

[71] J. Mandel, L. S. Bennethum, J. D. Beezley, J. L. Coen, C. C. Douglas, M. Kim, and A. Vodacek. A wildland fire model with data assimilation. *Mathematics and Computers in Simulation*, 79(3):584–606, 2008.

[72] J. Mandel, J. D. Beezley, J. L. Coen, and M. Kim. Data assimilation for wildland fires. *Control Systems, IEEE*, 29(3):47–65, 2009.

[73] J. E. Marsden and M. J. Hoffman. *Basic Complex Analysis*. Macmillan, 1999.

[74] K. McGrattan, S. Hostikka, R. McDermott, J. Floyd, C. Weinschenk, and K. Overholt. Fire dynamics simulator (version 6), technical reference guide. *NIST Special Publication*, 1018-1(6), 2015.

[75] K. B. McGrattan and G. P. Forney. *Fire Dynamics Simulator: User's Manual*. US Department of Commerce, Technology Administration, National Institute of Standards and Technology, 2000.

[76] L. Meier III and D. G. Luenberger. Approximation of linear constant systems. *Automatic Control, IEEE Transactions on*, 12(5):585–588, 1967.

[77] R. E. Meyer. *Introduction to Mathematical Fluid Dynamics*, volume 24. Courier Corporation, 1971.

[78] R. K. Miller and A. N. Michel. *Ordinary Differential Equations*. Courier Dover Publications, Nov. 2007. ISBN 978-0-486-46248-6.

[79] M. Molla and M. M. A. Sarker. Natural convection flow in a square cavity with temperature dependent heat generation. In *Proceedings of the 3rd BSME-ASME International Conference on Thermal Engineering, December*, volume 20, page 22, 2006.

[80] B. C. Moore. Principal component analysis in linear systems: Controllability, observability, and model reduction. *Automatic Control, IEEE Transactions on*, 26(1):17–32, 1981.

[81] C. T. Mullis and R. Roberts. Synthesis of minimum roundoff noise fixed point digital filters. *Circuits and Systems, IEEE Transactions on*, 23(9):551–562, 1976.

[82] M. Renardy and R. C. Rogers. *An Introduction to Partial Differential Equations*, volume 13. Springer Science & Business Media, 2006.

[83] W. J. Rugh. *Mathematical Description of Linear Systems*. M. Dekker, 1975.

[84] A. Ruhe. Rational Krylov sequence methods for eigenvalue computation. *Linear Algebra and its Applications*, 58:391–405, 1984.

[85] B. R. Sharma, M. Kumar, and K. Cohen. Spatio-temporal estimation of wildfire growth. In *ASME 2013 Dynamic Systems and Control Conference*, pages V002T25A005–V002T25A005. American Society of Mechanical Engineers, 2013.

[86] A. Sinha. *Linear Systems: Optimal and Robust Control*. CRC Press, 2007.

[87] N. K. Sinha. *Linear Systems*. Wiley, New York, 1991. ISBN 978-0-471-62341-0.

[88] C. Solnordal, P. Witt, M. Prakash, P. Liovic, K. Tanguturi, and R. Balusu. CFD modeling of methane gas distribution and control strategies in a gassy coal mine. *The Journal of Computational Multiphase Flows*, 6(1):65–78, 2014.

[89] S. R. Turns. *An Introduction to Combustion*, volume 287. McGraw-hill New York, 1996.

[90] S. Wyatt. *Issues in Interpolatory Model Reduction: Inexact Solves, Second-order Systems and DAEs*. PhD thesis, Virginia Polytechnic Institute and State University, 2012.

[91] G. Xu, K. D. Luxbacher, S. Ragab, and S. Schafrik. Development of a remote analysis method for underground ventilation systems using tracer gas and CFD in a simplified laboratory apparatus. *Tunnelling and Underground Space Technology*, 33:1–11, 2013.

[92] G. Xu, K. D. Luxbacher, S. Ragab, J. Xu, and X. Ding. Computational fluid dynamics applied to mining engineering: A review. *International Journal of Mining, Reclamation and Environment*, pages 1–25, 2016.

[93] A. Yousouff and R. E. Skelton. Covariance equivalent realizations with applications to model reduction of large-scale systems. *Control and Dynamic Systems*, 22:273–348, 1985.

[94] A. Yousuff, D. A. Wagie, and R. E. Skelton. Linear system approximation via covariance equivalent realizations. *Journal of Mathematical Analysis and Applications*, 106(1):91–115, 1985.

[95] L. Yuan, R. J. Mainiero, J. H. Rowland, R. A. Thomas, and A. C. Smith. Numerical and experimental study on flame spread over conveyor belts in a large-scale tunnel. *Journal of Loss Prevention in the Process Industries*, 30:55–62, July 2014.

[96] K. Zhou, J. C. Doyle, and K. Glover. *Robust and Optimal Control*, volume 40. Prentice hall New Jersey, 1996.

Appendices

Appendix A

Notation

Table A.1: Mathematical Notation

Symbol	Description
$\mathbf{A}^T$	Transpose of a matrix
$\mathbf{A}^*$	Conjugate transpose of a matrix
$v_t(x,t)$	Partial derivative with with respect to time, e.g. $\frac{\partial v}{\partial t}$
$v_x(x,t)$	Partial derivative with respect to the spatial component, e.g. $\frac{\partial v}{\partial x}$
$v_{xx}(x,t)$	Second partial derivative with respect to the spatial component, e.g. $\frac{\partial^2 v}{\partial x^2}$
$\dot{x}(t)$	A dotted function indicates the time derivative, e.g. $\frac{dx}{dt}$.
∇	Gradient of a function
∇^2	Laplacian of a function
$\Re(z)$	Real part, a, of the complex number $z = a + bi$.
$\Im(z)$	Imaginary part, b, of the complex number $z = a + bi$.
$\lambda(\mathbf{A})$	The set of eigenvalues of $\mathbf{A}$
$G(s)$	Continuous-time transfer functions
$H(z)$	Discrete-time transfer functions

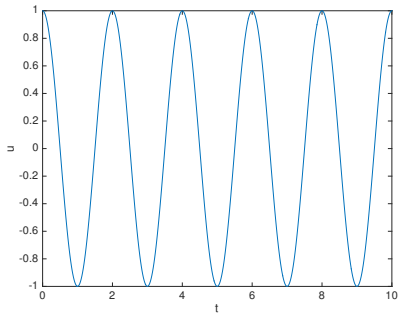
For this dissertation, we will use the conventions listed in Table A.1. Where ambiguities are possible, we will elucidate the meaning in the discussion. In general, we will adhere to the convention that bold uppercase letters refer to matrices, e.g. $\mathbf{A} \in \mathbb{R}^{m\times n}$ or $\mathbf{B} \in \mathbb{C}^{m\times n}$, and bold lowercase letters refer to column vectors in $\mathbb{R}^n$ or $\mathbb{C}^n$. Scalars will simply be italic

lowercase or uppercase letters depending on the context. Constants will normally be written using Greek letters and will be considered greater than or equal to zero unless otherwise noted.

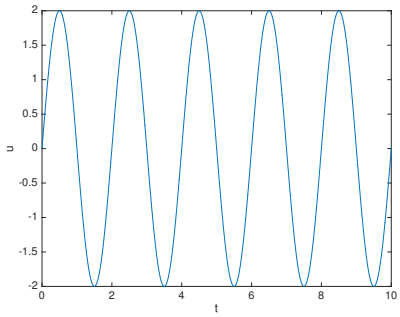
Appendix B

Burgers' Equation Control Functions

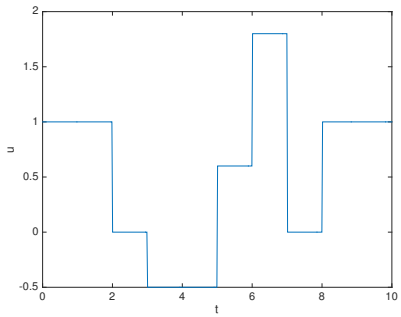
When testing various model reduction techniques on the Burgers' equation, we first build the reduced-order model using the control function $u_1(t) = \cos(\pi t)$.



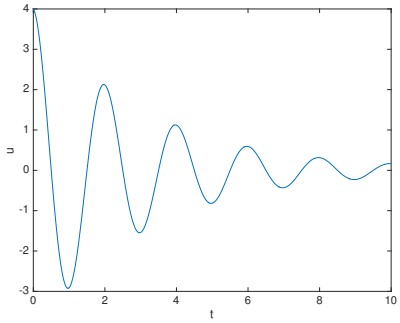
(a) $u_1(t) = \cos(\pi t)$



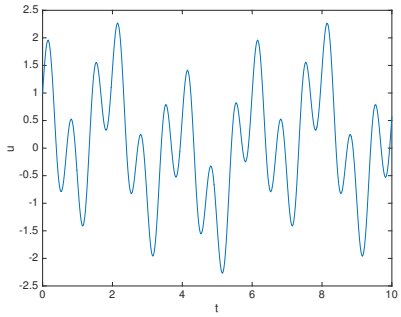
(b) $u_2(t) = \sin(\pi t)$



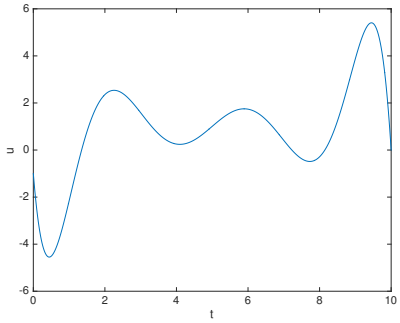
(c) $u_3(t)$: Step function



(d) $u_4(t) = 4e^{-t/\pi}\cos(\pi t)$



(e) $u_5(t)\cos(\pi t) + \sin(3\pi t) + \frac{1}{2}\sin(\frac{\pi t}{3})$



(f) $u_6(t) = P(t)$[1]

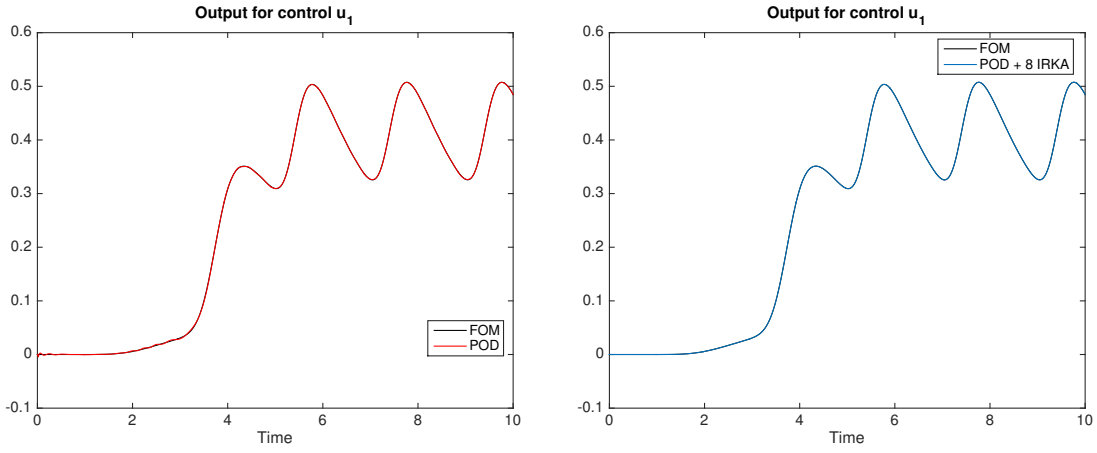
Figure B.1: Control functions for Burgers equation.

[1] $$P(t) = -(0.001519632917)t^7 + (0.05294205)t^6 - (0.729703917)t^5 + (5.0421)t^4 - (18.1461291)t^3 + (31.3845)t^2 - (18.765)t - 1$$

Appendix C

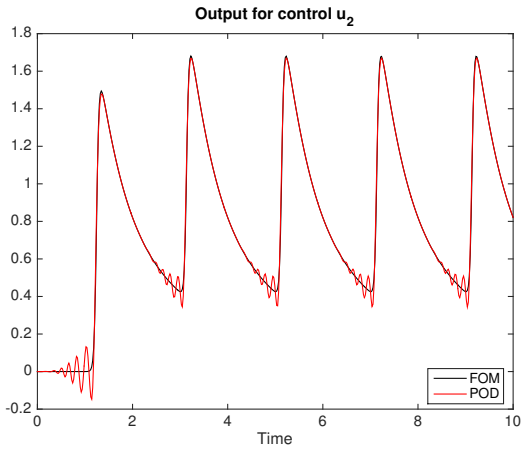
Full Results for POD+IRKA

The appendix gives the plots from all of the tests run using Algorithm 4.7 to combine the POD and IRKA projection vectors. We tested reduced order model sizes of $r = 15$ and $r = 20$ for all six input functions, $u_1(t), \ldots, u_6(t)$. We present the results here to minimize the flow of the actual text.

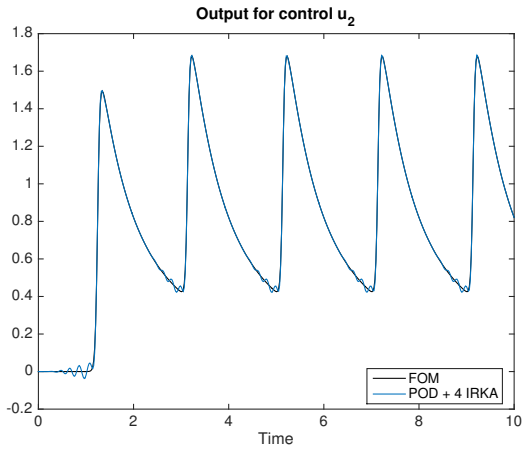


(a) POD for input $u_1(t)$ (b) Combined POD and IRKA

Figure C.1: Output combining POD and IRKA vectors with $r = 15$ and input $u_1(t)$.

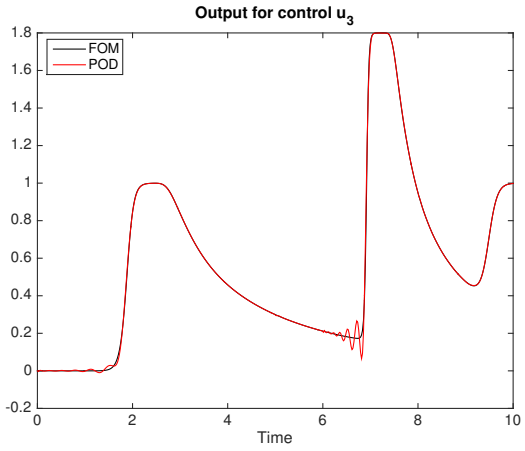


(a) POD for input $u_2(t)$

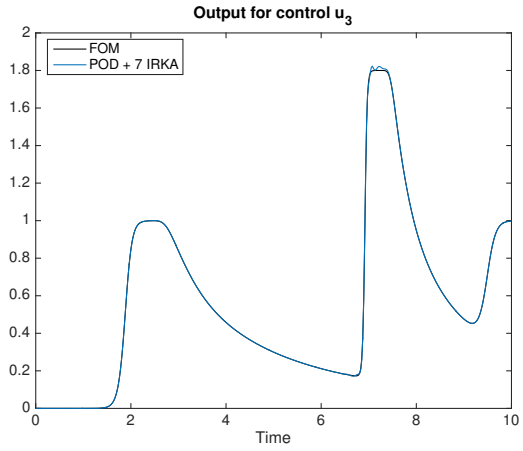


(b) Combined POD and IRKA

Figure C.2: Output combining POD and IRKA vectors with $r = 15$ and input $u_2(t)$.

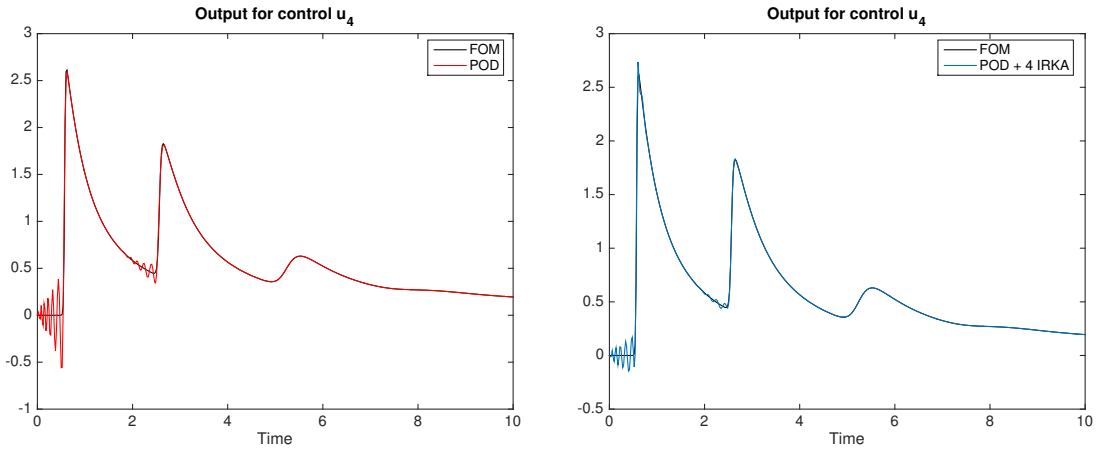


(a) POD for input $u_3(t)$



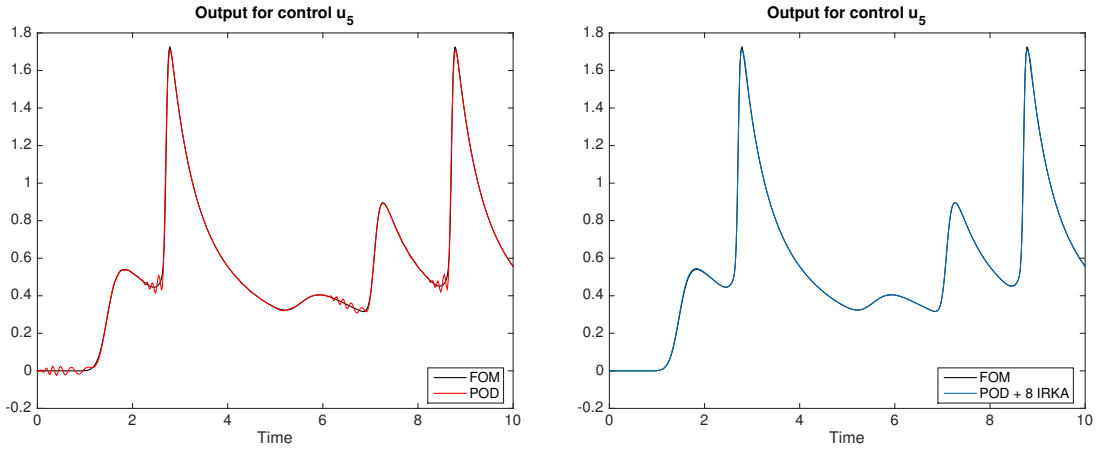
(b) Combined POD and IRKA

Figure C.3: Output combining POD and IRKA vectors with $r = 15$ and input $u_3(t)$.



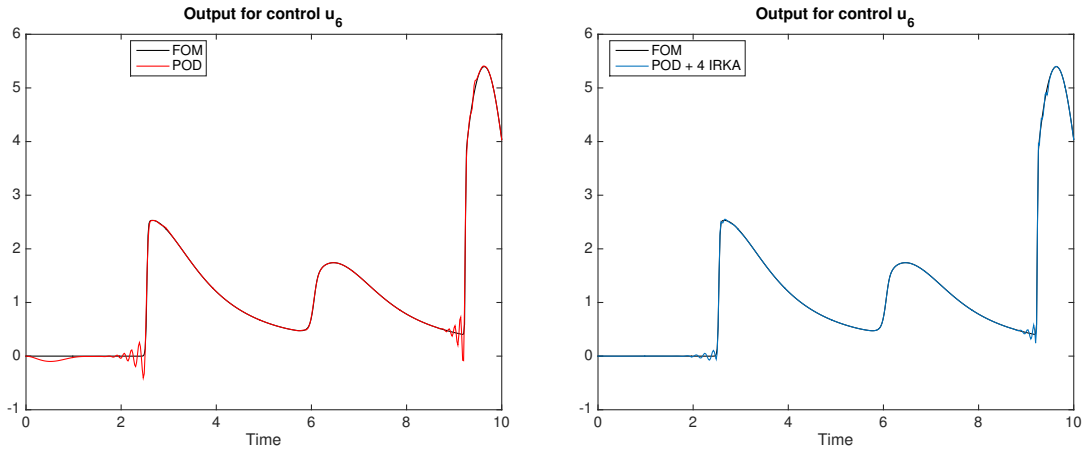
(a) POD for input $u_4(t)$ (b) Combined POD and IRKA

Figure C.4: Output combining POD and IRKA vectors with $r = 15$ and input $u_4(t)$.



(a) POD for input $u_5(t)$ (b) Combined POD and IRKA

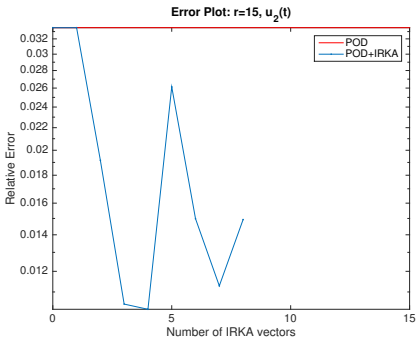
Figure C.5: Output combining POD and IRKA vectors with $r = 15$ and input $u_5(t)$.



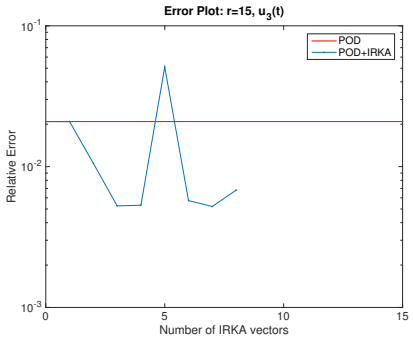
(a) POD for input $u_6(t)$ (b) Combined POD and IRKA

Figure C.6: Output combining POD and IRKA vectors with $r = 15$ and input $u_6(t)$.

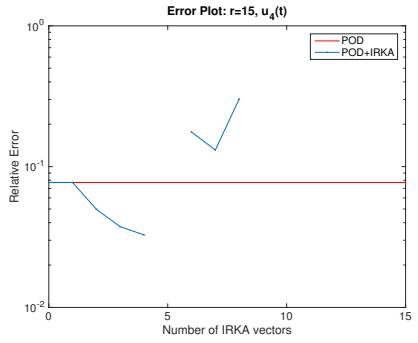
(a) Input $u_1(t)$



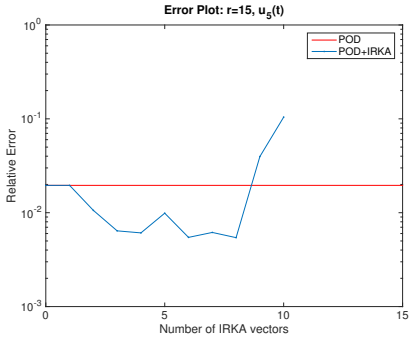
(b) Input $u_2(t)$



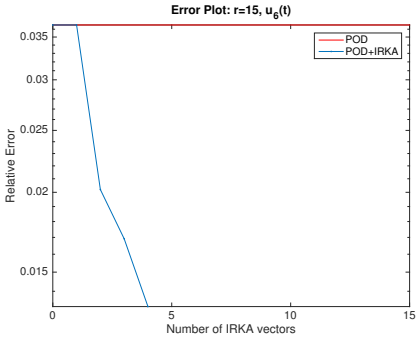
(c) Input $u_3(t)$



(d) Input $u_4(t)$

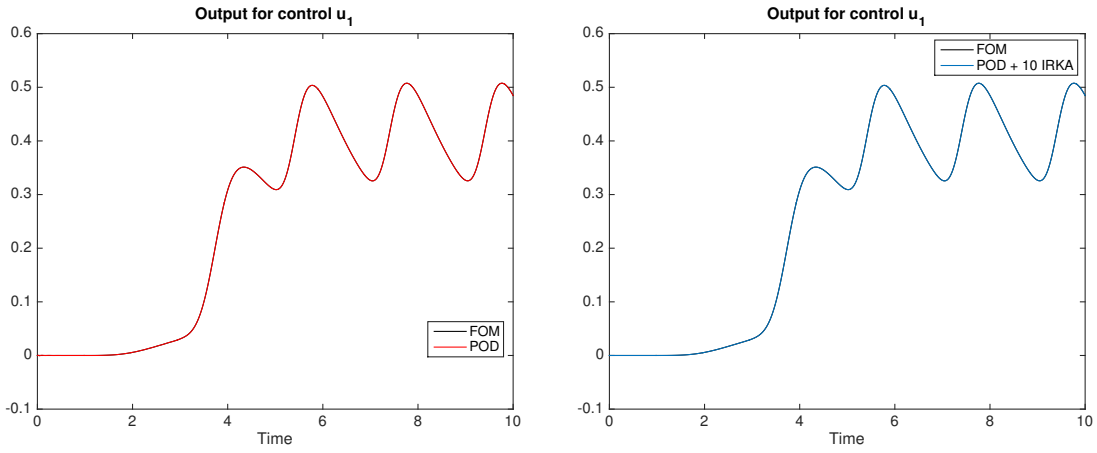


(e) Input $u_5(t)$



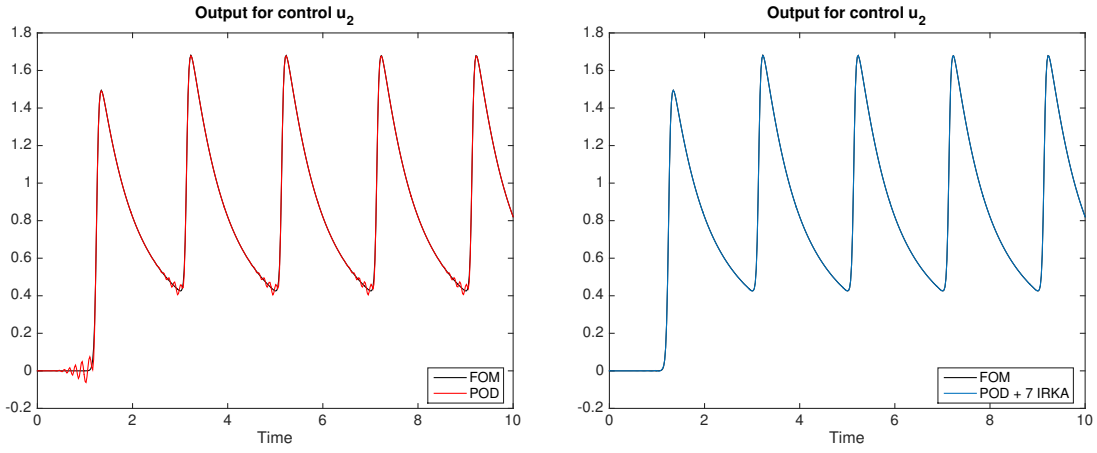
(f) Input $u_6(t)$

Figure C.7: Output error combining POD and IRKA vectors for $r = 15$.



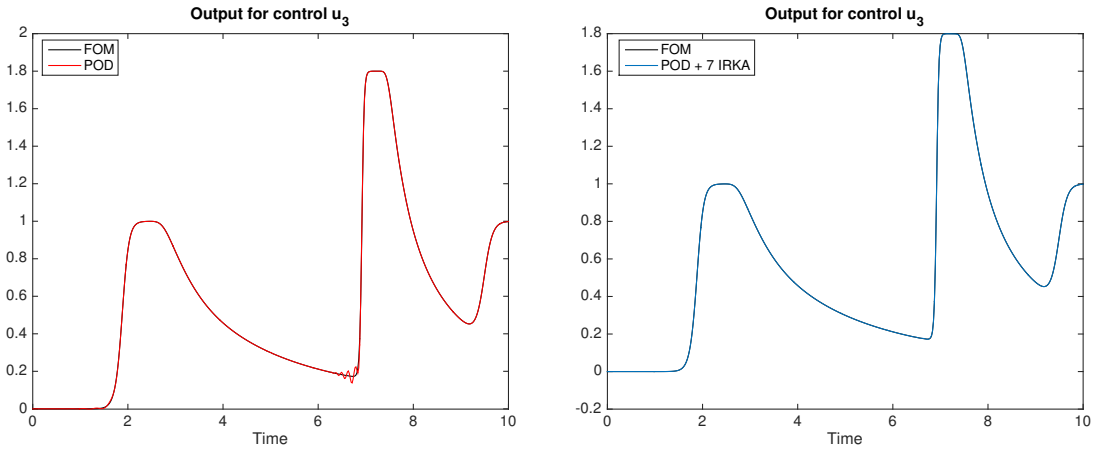
(a) POD for input $u_1(t)$ (b) Combined POD and IRKA

Figure C.8: Output combining POD and IRKA vectors with $r = 20$ and input $u_1(t)$.



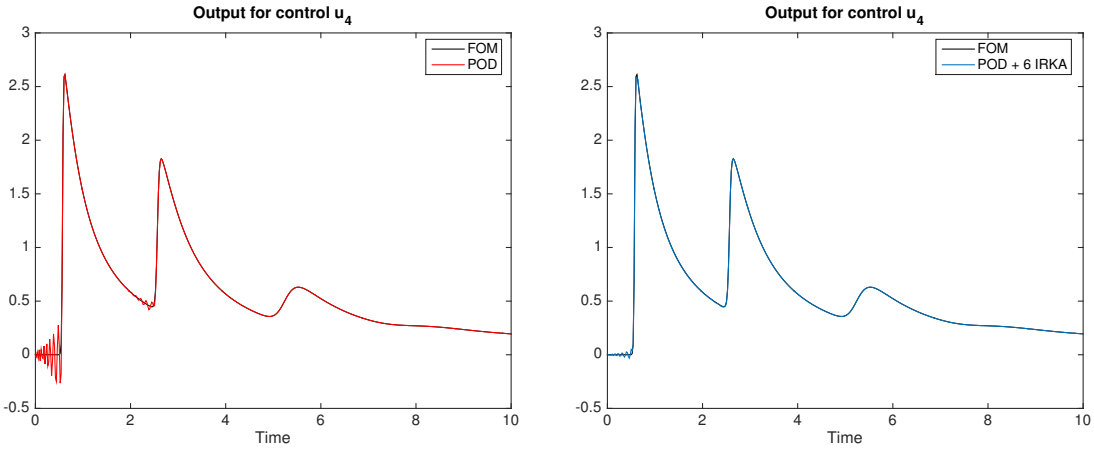
(a) POD for input $u_2(t)$ (b) Combined POD and IRKA

Figure C.9: Output combining POD and IRKA vectors with $r = 20$ and input $u_2(t)$.



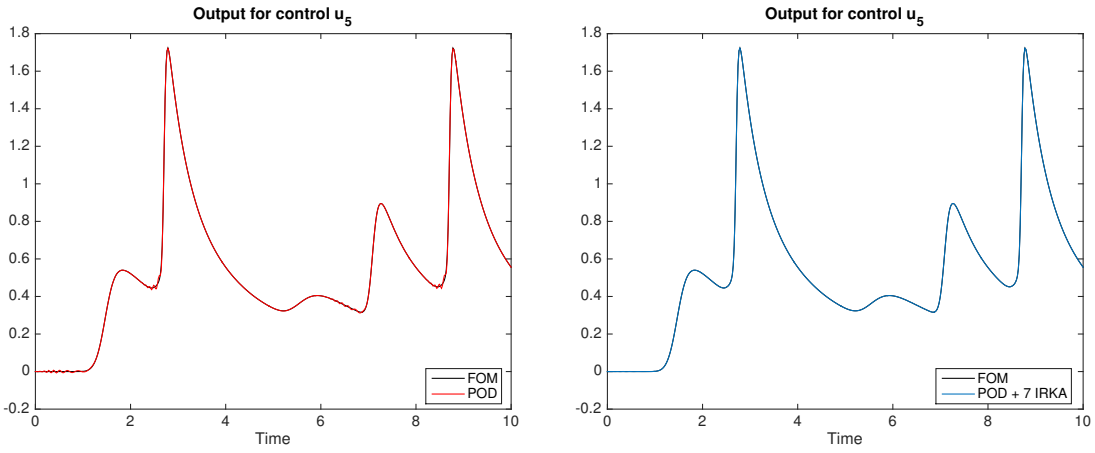
(a) POD for input $u_3(t)$ (b) Combined POD and IRKA

Figure C.10: Output combining POD and IRKA vectors with $r = 20$ and input $u_3(t)$.



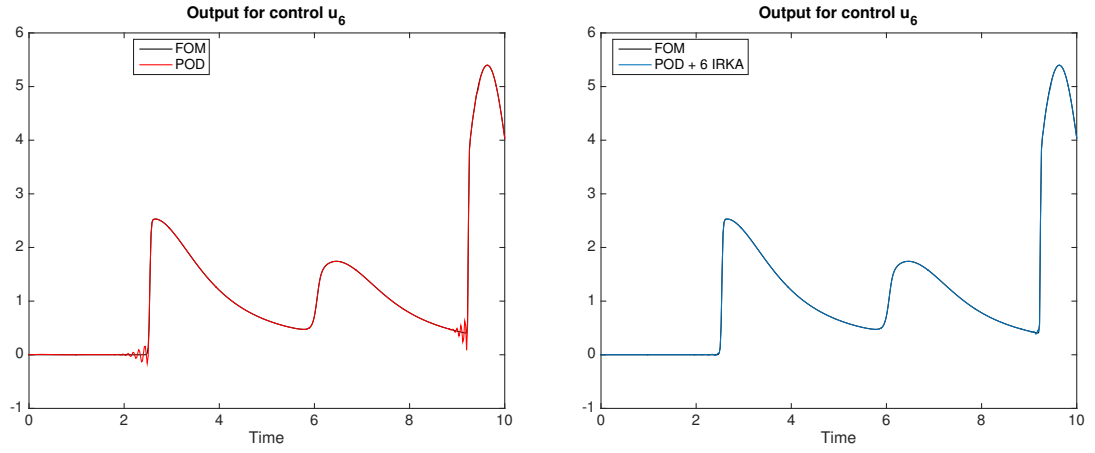
(a) POD for input $u_4(t)$ (b) Combined POD and IRKA

Figure C.11: Output combining POD and IRKA vectors with $r = 20$ and input $u_4(t)$.



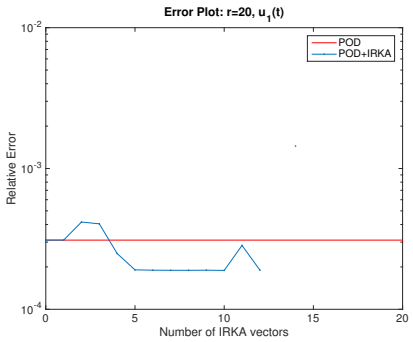
(a) POD for input $u_5(t)$ (b) Combined POD and IRKA

Figure C.12: Output combining POD and IRKA vectors with $r = 20$ and input $u_5(t)$.

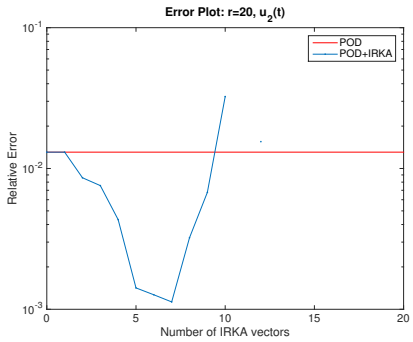


(a) POD for input $u_6(t)$ (b) Combined POD and IRKA

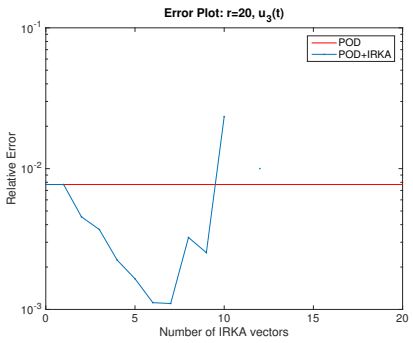
Figure C.13: Output combining POD and IRKA vectors with $r = 20$ and input $u_6(t)$.



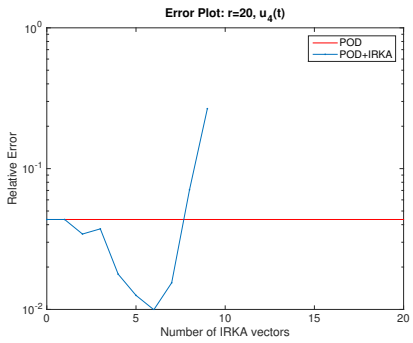
(a) Input $u_1(t)$



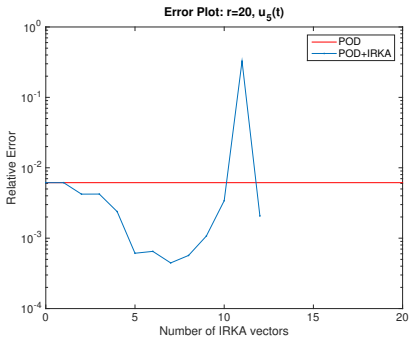
(b) Input $u_2(t)$



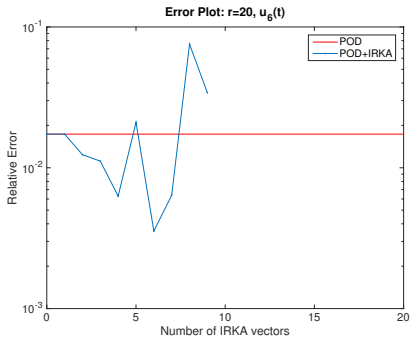
(c) Input $u_3(t)$



(d) Input $u_4(t)$



(e) Input $u_5(t)$



(f) Input $u_6(t)$

Figure C.14: Selected output error combining POD and IRKA vectors for $r = 20$.

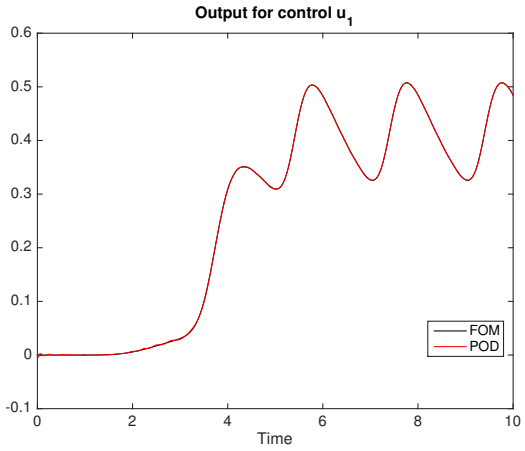
Appendix D

Full Results for IRKA $\mathbf{V} \bigoplus \mathbf{W}$

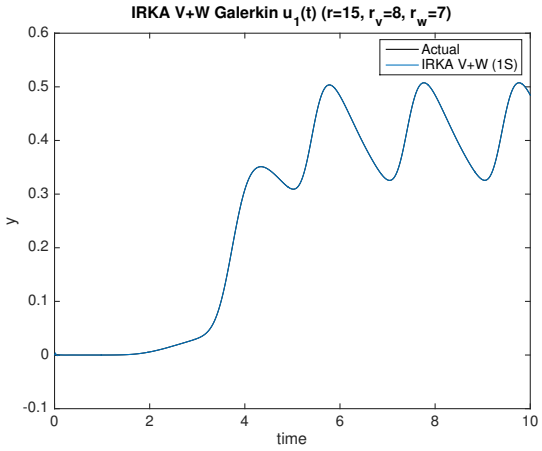
The appendix gives the plots from all of the tests run using Algorithm 4.9 to combine the IRKA right and left projection vectors. We tested reduced-order model sizes of $r = 15$ and $r = 20$ for all six input functions, $u_1(t), \ldots, u_6(t)$. We present the results here to minimize the flow of the actual text.

D.1 Details for $r = 15$

D.1.1 Output Plots

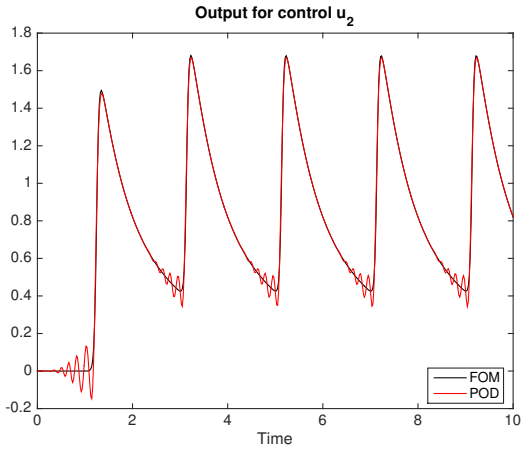


(a) POD for input $u_1(t)$

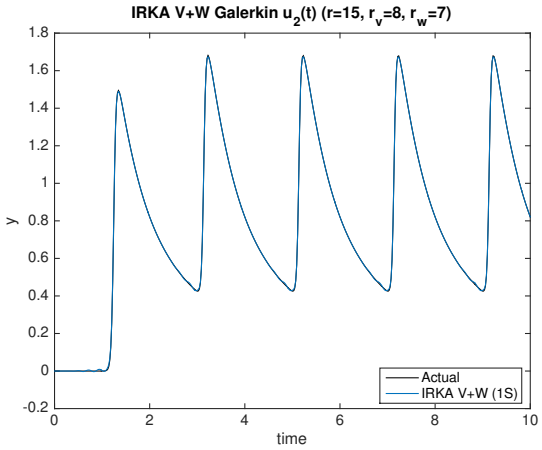


(b) IRKA $\mathbf{V} \oplus \mathbf{W}$

Figure D.1: Output for combined IRKA right and left projection vectors with $r = 15$ and input $u_1(t)$.

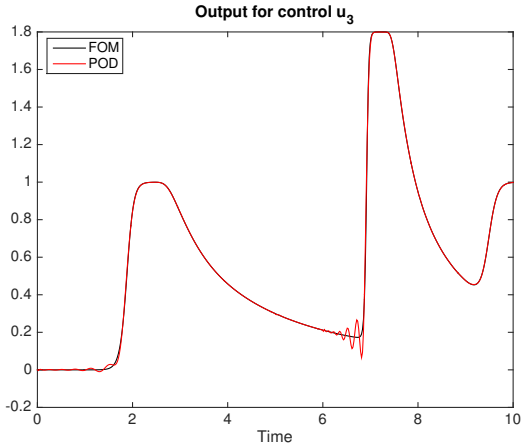


(a) POD for input $u_2(t)$

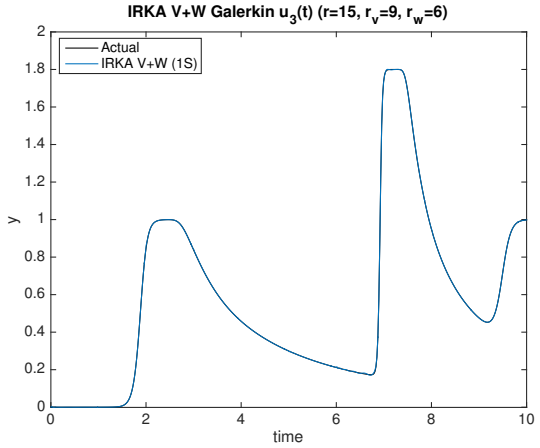


(b) IRKA $\mathbf{V} \oplus \mathbf{W}$

Figure D.2: Output for combined IRKA right and left projection vectors with $r = 15$ and input $u_2(t)$.

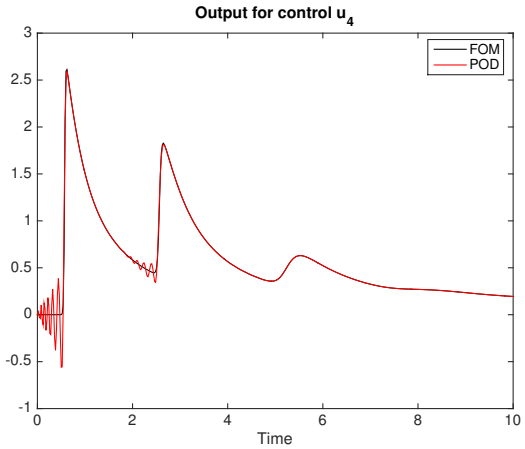


(a) POD for input $u_3(t)$

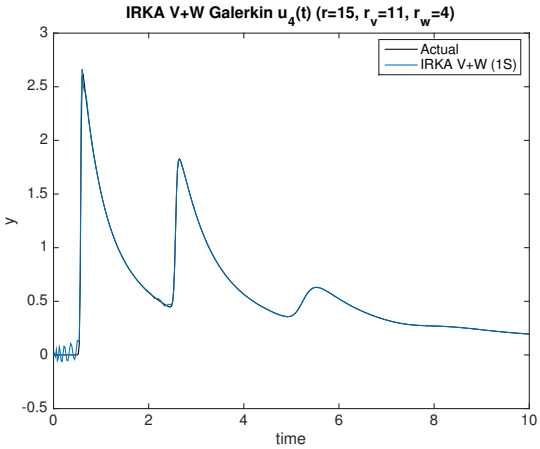


(b) IRKA $\mathbf{V} \oplus \mathbf{W}$

Figure D.3: Output for combined IRKA right and left projection vectors with $r = 15$ and input $u_3(t)$.

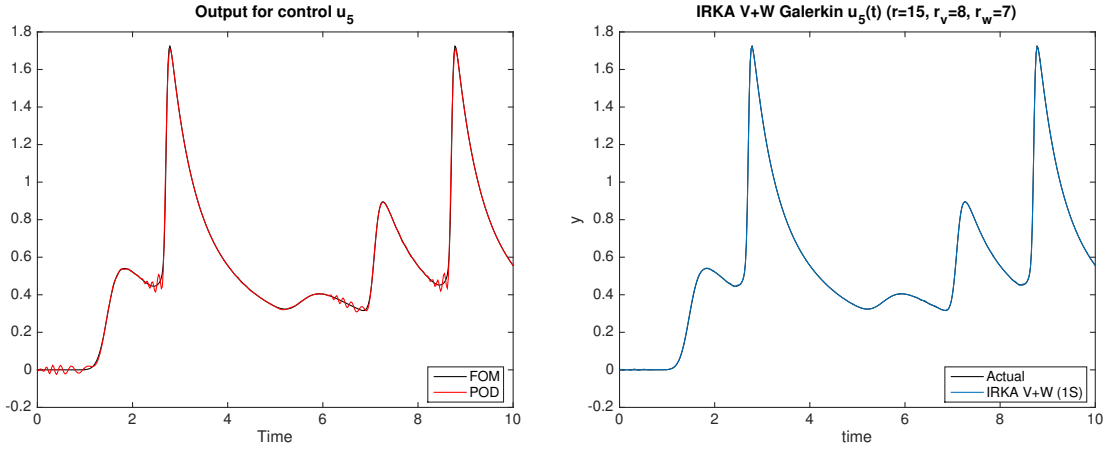


(a) POD for input $u_4(t)$



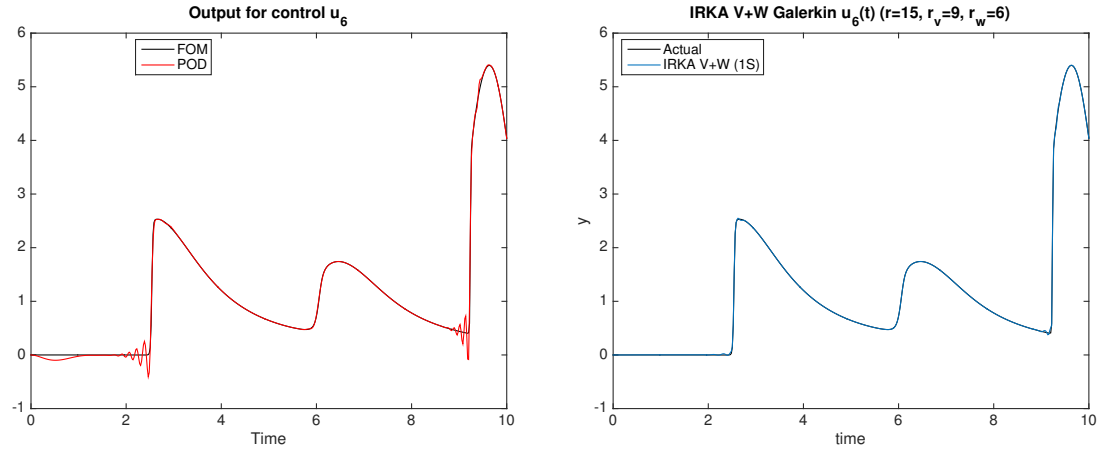
(b) IRKA $\mathbf{V} \oplus \mathbf{W}$

Figure D.4: Output for combined IRKA right and left projection vectors with $r = 15$ and input $u_4(t)$.



(a) POD for input $u_5(t)$ (b) IRKA $\mathbf{V} \oplus \mathbf{W}$

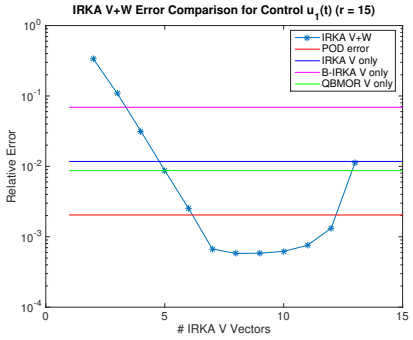
Figure D.5: Output for combined IRKA right and left projection vectors with $r = 15$ and input $u_5(t)$.



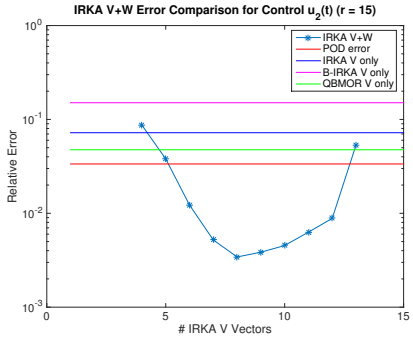
(a) POD for input $u_6(t)$ (b) IRKA $\mathbf{V} \oplus \mathbf{W}$

Figure D.6: Output for combined IRKA right and left projection vectors with $r = 15$ and input $u_6(t)$.

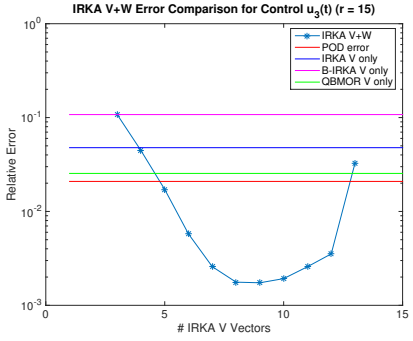
D.1.2 Error Plots



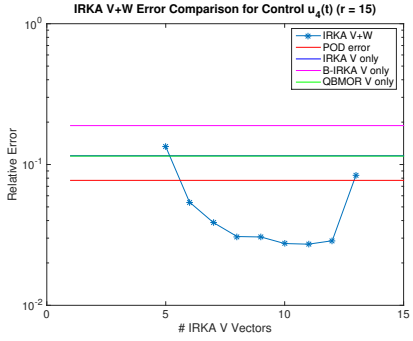
(a) Input $u_1(t)$



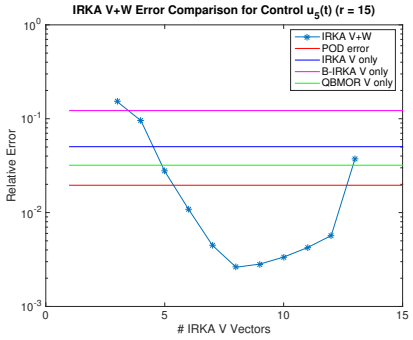
(b) Input $u_2(t)$



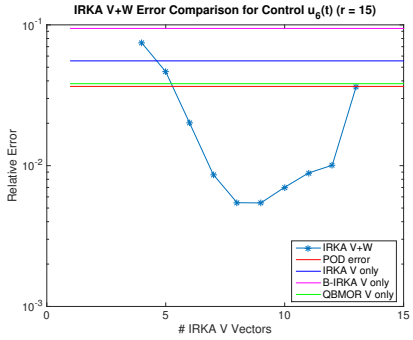
(c) Input $u_3(t)$



(d) Input $u_4(t)$



(e) Input $u_5(t)$

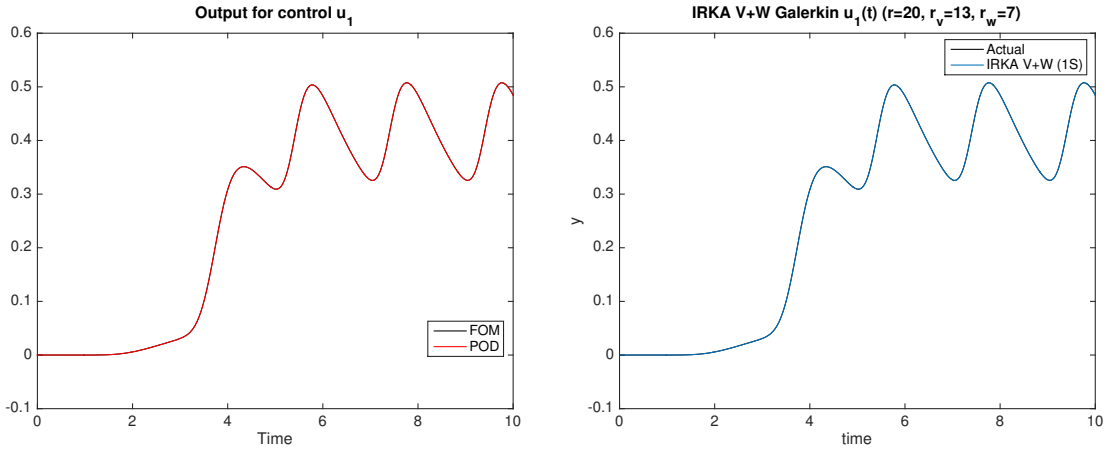


(f) Input $u_6(t)$

Figure D.7: Relative output error combining left and right IRKA projection vectors for $r = 15$.

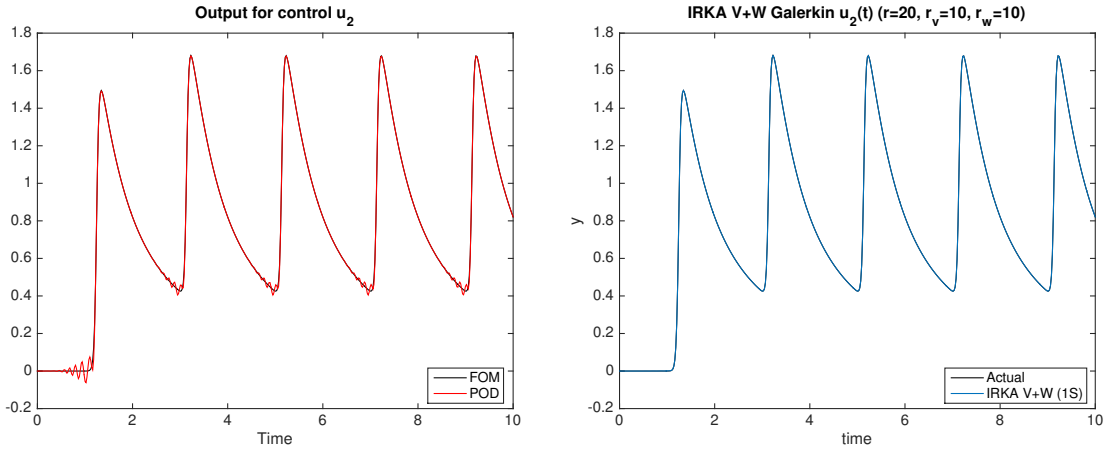
D.2 Details for $r = 20$

D.2.1 Output Plots



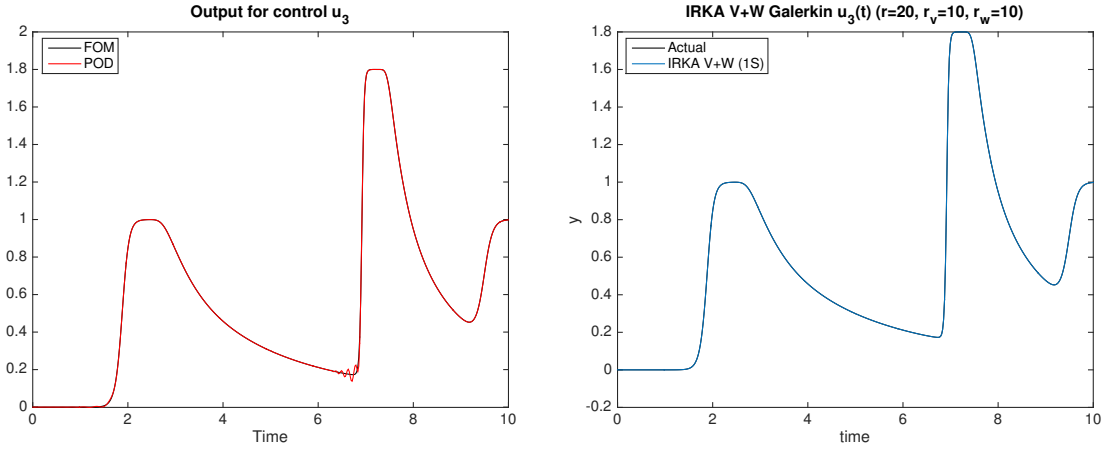
(a) POD for input $u_1(t)$ (b) IRKA $\mathbf{V} \oplus \mathbf{W}$

Figure D.8: Output for combined IRKA right and left projection vectors with $r = 20$ and input $u_1(t)$.



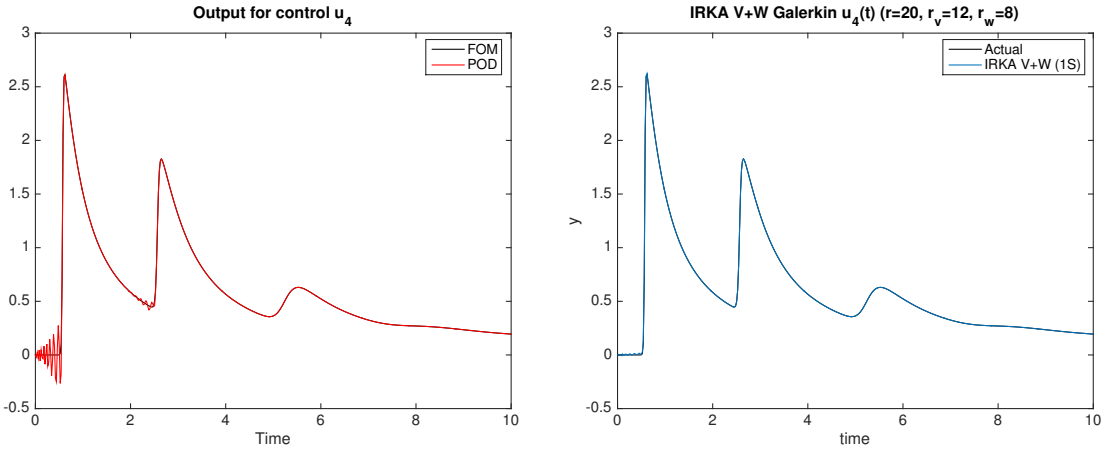
(a) POD for input $u_2(t)$ (b) IRKA $\mathbf{V} \oplus \mathbf{W}$

Figure D.9: Output for combined IRKA right and left projection vectors with $r = 20$ and input $u_2(t)$.



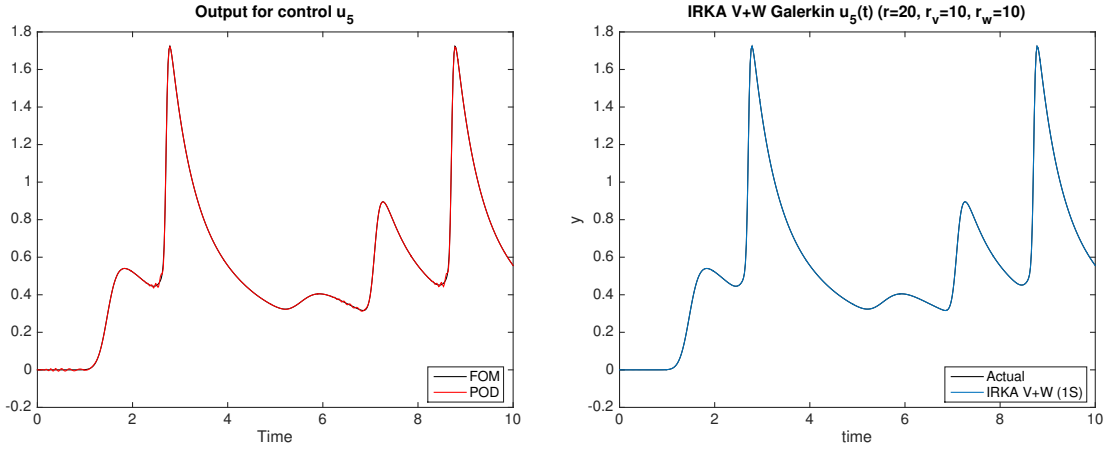
(a) POD for input $u_3(t)$ (b) IRKA $\mathbf{V} \oplus \mathbf{W}$

Figure D.10: Output for combined IRKA right and left projection vectors with $r = 20$ and input $u_3(t)$.



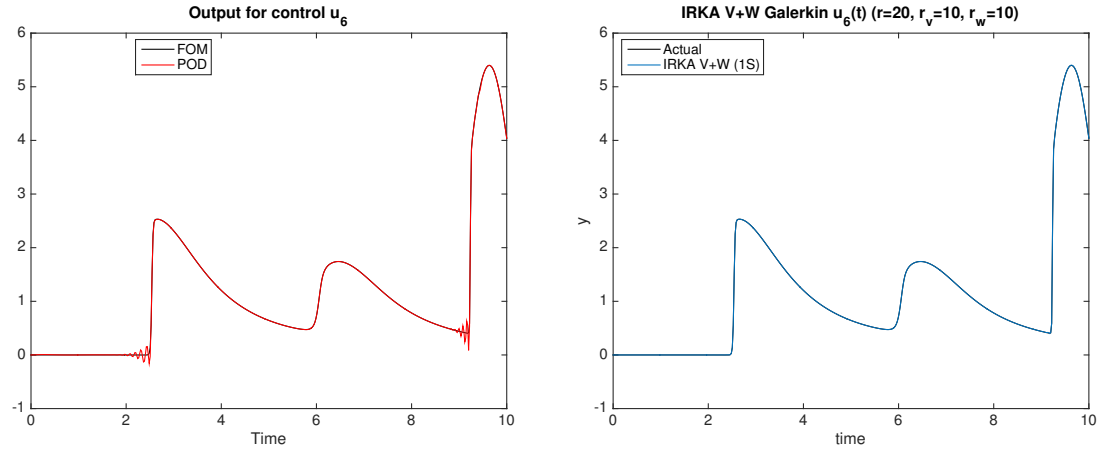
(a) POD for input $u_4(t)$ (b) IRKA $\mathbf{V} \oplus \mathbf{W}$

Figure D.11: Output for combined IRKA right and left projection vectors with $r = 20$ and input $u_4(t)$.



(a) POD for input $u_5(t)$ (b) IRKA $\mathbf{V} \oplus \mathbf{W}$

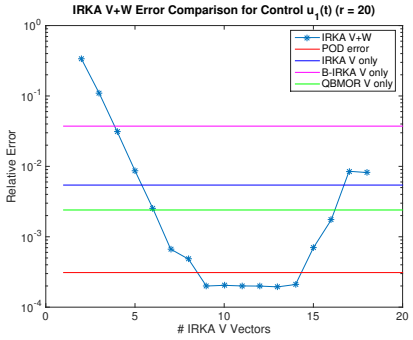
Figure D.12: Output for combined IRKA right and left projection vectors with $r = 20$ and input $u_5(t)$.



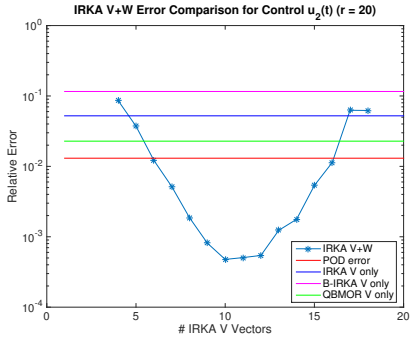
(a) POD for input $u_6(t)$ (b) IRKA $\mathbf{V} \oplus \mathbf{W}$

Figure D.13: Output for combined IRKA right and left projection vectors with $r = 20$ and input $u_6(t)$.

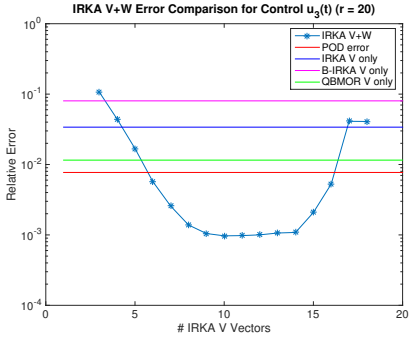
D.2.2 Error Plots



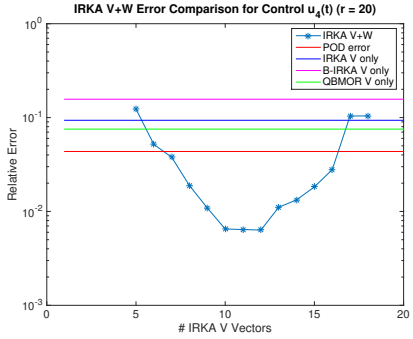
(a) Input $u_1(t)$



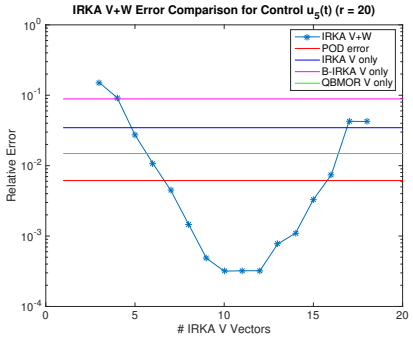
(b) Input $u_2(t)$



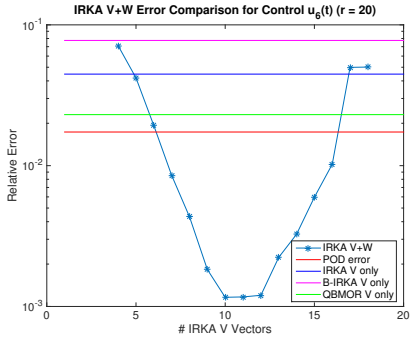
(c) Input $u_3(t)$



(d) Input $u_4(t)$



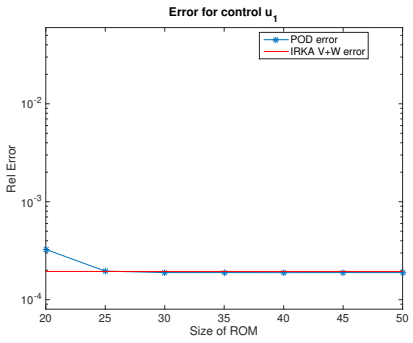
(e) Input $u_5(t)$



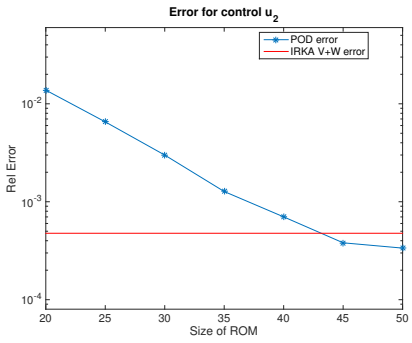
(f) Input $u_6(t)$

Figure D.14: Relative output error for IRKA $\mathbf{V} \oplus \mathbf{W}$ compared to other methods with $r = 20$.

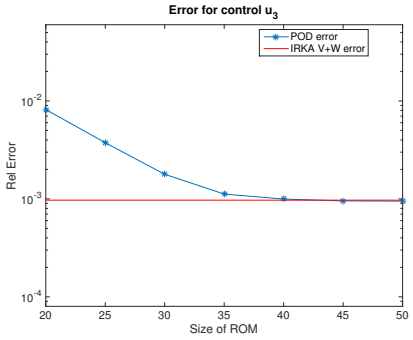
D.3 POD Model Error Plots



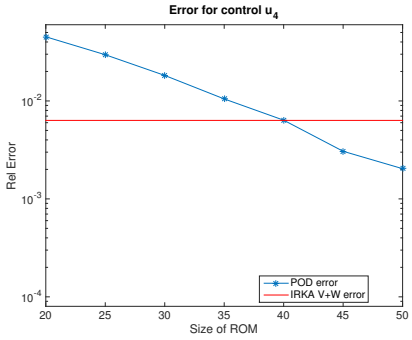
(a) Input $u_1(t)$



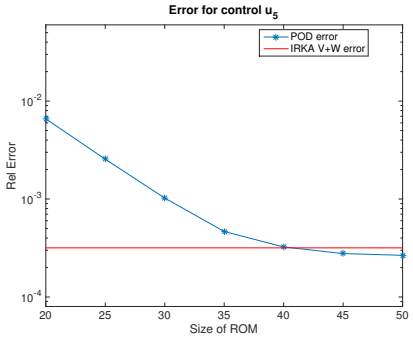
(b) Input $u_2(t)$



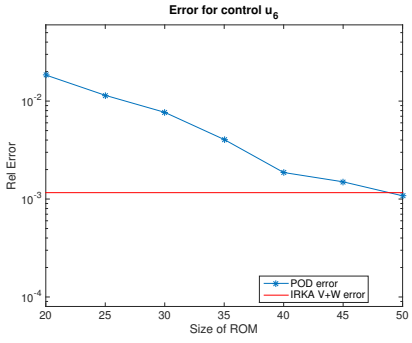
(c) Input $u_3(t)$



(d) Input $u_4(t)$



(e) Input $u_5(t)$



(f) Input $u_6(t)$

Figure D.15: Relative output error comparison between POD models of various sizes and IRKA $\mathbf{V} \oplus \mathbf{W}$ at $r = 20$