RICE UNIVERSITY

Projection-Based Model Reduction in the Context of Optimization with Implicit PDE Constraints

by

Caleb Magruder

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE

Doctor of Philosophy

THESIS COMMITTEE:

Matthias Heinkenschloss, Chairman
Noah G. Harding Chair and Professor of Computational and Applied Mathematics

Beatrice Riviere
Noah G. Harding Chair and Professor of Computational and Applied Mathematics

Matthew Knepley
Assistant Professor of Computational and Applied Mathematics

Athanasios Antoulas
Professor of Electrical and Computer Engineering

HOUSTON, TEXAS

MAY, 2017
Abstract

I use reduced order models (ROMs) to substantially decrease the computational cost of Newton’s method for large-scale time-dependent optimal control problems in settings where solving the implicit constraints and their adjoints can be prohibitively expensive. Reservoir management in particular can take weeks to solve the state and adjoint equations necessary to generate one gradient evaluation. Model order reduction has potential to reduce the cost; however, ROMs are valid only nearby their training and must be retrained as the optimization progresses. I will demonstrate that in the case of reservoir management, frequent retraining defeats the purpose of using the reduced order model for the state equation altogether.

Rather than generating a reduced order model to replace the original implicit constraint, I use the structure of Hessian and subspace-based ROMs to compute approximate reduced order Hessian information by recycling data from the full order state and adjoint solves in the gradient computation. The resulting method often enjoys convergence rates similar to those of Newton’s Method, but at the computational cost of the gradient method.

I demonstrate my approach on nonlinear parabolic optimal control problems on two cases: (1) a semilinear parabolic case with nonlinear reaction terms and (2) the well rate optimization problem in reservoir engineering. For semilinear parabolic case, I consider a cubic reaction term and an exponential reaction term modeling solid fuel injection. My results show a dramatic speedup in wall clock time for the entire optimization procedure. This overall acceleration includes the training and precomputation of the ROMs that is conventionally discounted as ”offline” computational costs.
Acknowledgements

This work was supported in part through a sponsored research agreement with the ExxonMobil Upstream Research Company.
Contents

Abstract ii

Acknowledgements iii

1 Introduction 2
  1.1 Model Reduction in Optimization . . . . . . . . . . . . . . . . . . . 3
  1.2 Model Reduction and Flow in Porous Media . . . . . . . . . . . . . 4
  1.3 Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

2 Background 9
  2.1 Optimization with Implicit PDE Constraints . . . . . . . . . . . . . 9
    2.1.1 Adjoint-Based Derivative Computation . . . . . . . . . . . . . 11
  2.2 Subspace-Based Model Order Reduction . . . . . . . . . . . . . . . . 13
    2.2.1 Performance Guarantees . . . . . . . . . . . . . . . . . . . . . 15
  2.3 Proper Orthogonal Decomposition . . . . . . . . . . . . . . . . . . . 18
    2.3.1 Numerical Computation of POD Subspace . . . . . . . . . . . . 21
    2.3.2 Performance Guarantees via POD . . . . . . . . . . . . . . . . 22
  2.4 Empirical Interpolation Method . . . . . . . . . . . . . . . . . . . . 23
  2.5 Discrete Empirical Interpolation Method . . . . . . . . . . . . . . . 25
  2.6 Two-Phase, Incompressible Flows in Porous Media . . . . . . . . . . 28
    2.6.1 Darcy’s Law and Permeability . . . . . . . . . . . . . . . . . . 28
    2.6.2 Porosity . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 29
5.2 Finite Element Method Discretization ........................................ 84
  5.2.1 Finite Element Error Estimates ....................................... 85
  5.2.2 Matrix Representation of the Finite Element Method ............ 88
5.3 POD-ROM Approximation ...................................................... 91
  5.3.1 Proper Orthogonal Decomposition .................................... 91
  5.3.2 Proper Orthogonal Decomposition Error ............................. 93
5.4 The DEIM for Finite Elements .............................................. 96
  5.4.1 DEIM Applied Directly to Nonlinearity ............................... 97
  5.4.2 Unassembled DEIM ..................................................... 97
  5.4.3 A nonintrusive EIM+FEM precomputation ............................. 99
5.5 Error Due to Piecewise Polynomial Interpolation of Nonlinearity ... 101
5.6 Example Problem - Solid Fuel Ignition Model ............................ 105

6 Semilinear Parabolic Optimal Control ...................................... 110
  6.1 Problem Formulation ....................................................... 111
    6.1.1 Finite Element Discretization .................................... 111
  6.2 Piecewise Polynomial Approximation of the Nonlinear Reaction Term and DEIM ................................................................. 116
    6.2.1 ROM Precomputation of First Derivative of Nonlinearity ........ 117
    6.2.2 ROM Precomputation of the Second Derivative of Nonlinearity 118
    6.2.3 Computation of the DEIM Subspaces ............................... 119
  6.3 Results - Cubic Reaction .................................................. 121
  6.4 Results - Solid Fuel Ignition Model .................................... 127
  6.5 Discussion of Results ..................................................... 132

7 Well Rate Optimization ......................................................... 133
  7.1 Problem Formulation ....................................................... 133
  7.2 Projected Gradient Method ................................................. 135
  7.3 An Implementation Newton’s Method with Mixed Constraints ........ 137
**Notation**

\[ y(x, t) \] lower-case letter denote function values

\[ V \] mathcal capital letter denotes vector spaces and subspaces

\[ y \] boldface lower-case letters denote vectors in \( \mathbb{R}^n \)

\[ y_k \] discrete time is denoted as subscript

\[ A \] boldface capital letters denote matrices

\[ N(y) \] vector-valued function with vector-valued argument, \( y \)

\[ \vec{y} \] vec boldface denotes vector of vectors in \( (\mathbb{R}^n)^m \)

\[ J(\vec{y}, \vec{u}) \] full space objective function

\[ J(\vec{u}) \] reduced space objective function, \( J(\vec{y}(\vec{u}), \vec{u}) \)
Chapter 1

Introduction

The dominant cost of optimization with partial differential equations (PDEs) as implicit constraints is that of solving large-scale numerical solutions to the governing PDE and its adjoint. Objective function evaluations and adjoint-based derivatives necessitate many PDE simulations. In the context of high-fidelity PDEs solutions, this amounts to large-scale simulations that can quickly become prohibitive.

The core concept in this project is to approximate Hessian-vector computation via reduced order modeling. Conventional approaches to model reduction in optimization seek to approximate the implicit constraint with a reduced order model. Doing so generates a surrogate model whose solution is considerably cheaper to approximate, and consequently, so is the cost of derivative computations.

However, in my experience with reduced order modeling and optimization, generating reduced order models (ROMs) that are consistently representative of their original full order models (FOMs) throughout the optimization is expensive, if not impossible, depending on the application. Indeed, for the reservoir management problem discussed in Chapter 3, the surrogate model needs to be reconstructed frequently during the optimization (at the cost of full order solves, state and adjoint, at each iteration in the optimization). As the cost of “retraining” is at least the cost of the state and adjoint simulations, I propose using the solution trajectories to generate the
true gradient information of the problem. While full order state and adjoint solves are costly in reservoir simulation, Hessian-vector computations dominate the computational cost of solving the Newton subproblem via Conjugate Gradient. Hence, I propose to approximate the Hessian-vector computation via subspace-based model reduction, while using the full order gradient. I will demonstrate that this approach can dramatically speed up the Newton subproblem with little loss of fidelity.

1.1 Model Reduction in Optimization

I propose a method to dramatically decrease the computational cost of Newton’s method in large-scale time-dependent optimization with implicit partial differential equation (PDE) constraints. Conventional approaches to model reduction in optimization contexts seek to significantly lessen the cost of solving the implicit constraint by replacing it with a surrogate via subspace-based model order reduction, multi-scale modeling, etc., generating a new optimization problem whose solution is considerably less expensive to compute. Doing so significantly reduces the cost of gradient and Hessian evaluation and accelerates the optimization method, making Newton’s method, and the Gradient Method considerably less expensive to implement. Examples of this approach are numerous (see, e.g. [7, 16, 18, 19, 38, 40, 45, 57]).

Projection-based reduced order models attempt to systematically extract the important dynamics of a large-scale system and to construct substantially smaller systems by projecting the original large-scale system onto small dimensional subspaces that nearly contain the important dynamics. Ideally, the resulting reduced order model approximates the input / output mapping of the full order model but can be run at a fraction of the computational cost. The potential for optimization in reservoir management is this: if a reduced reservoir simulator exists, the large-scale reservoir equations can be replaced by the small dimensional reduced order model and the resulting optimization problem can be solved with substantially reduced cost. However,
because of the nonlinearities in the reservoir model and because of the large number of inputs (e.g., well rates at every time step) it is not clear that such reduced order models can be computed efficiently or how they can be applied in a broader range of operating scenarios (e.g. choices in control). For the reservoir management problem posed, subspace-based reduced order model quickly become inaccurate or even unstable when simulated under operating conditions modestly outside the ROM’s “training.”

I am interested in the possibility of reliably using reduced order models in a broader range of problems without manually tailoring the reduced order model to a specific problem instance – a challenge that is pathologically difficult in reservoir optimization. My approach attempts to circumvent this shortcoming of reduced order modeling in an optimization context.

Optimization algorithms explore a wide range of inputs in potentially high dimensional spaces. Based on my observations it is not clear that reliable reduced order models can replace the full order models in a cost-saving way when both online and offline computational costs are considered. Therefore, I propose using reduced order model, not to replace reservoir simulations, but to use them to reduce computational tasks within the optimization such as Hessian-vector computations. In my approach, reduced order models accelerate the optimization, but will not impact the accuracy of the simulation or optimization.

1.2 Model Reduction and Flow in Porous Media

Recent papers report successful applications of projection-based model reduction to specific reservoir simulation and reservoir optimization problems [11, 21, 26, 45]. Despite the literature on the subject, several questions about these approaches remain:

1. What can be done to improve reliability of the ROM away from the operating scenarios for which it was designed to do well?
2. Is it possible to detect failure of the ROM in a computationally affordable way: either in stability of the surrogate or fidelity of its simulation?

3. What guarantees do model reduction schemes have that the ROM saturation values remain within the physical bounds?

Unfortunately, these questions may not have sufficient answers. To address the first question, recent papers introduce the concept of localized Proper Orthogonal Decomposition and Discrete Empirical Interpolation [2, 25, 43]. These papers acknowledge that building a single ROM that is effective over a broad range of inputs is improbable in many applications settings. I demonstrate this shortcoming is certainly the case in oil reservoir management. To circumvent this issue, these papers generate a collection of ROMs that are locally effective in their respective regions of the control space (as opposed to being globally effective). Additionally, the approaches provide an online schedulers to determine which ROM or ROMs are needed for a given control.

Note that saturations bounds are not preserved when implementing methods proposed in Drohmann et al. (2012) [20, 21]. This is surprising given that a particular finite volume discretization of the two-phase porous media flow equations has been proven to keep saturation within bounds up to machine precision. See Eymard et al. (2003) and Michel (2003) [24, 41]. In my experience, ROM simulations violate the saturation constraints any time the water saturation is initialized at the minimum saturation, regardless of the dimension of the reduced system. Indeed, all literature that I am aware of on the subject does not guarantee that saturation stays within the physical bounds. However, if the requirement that reduced-order saturations must obey saturation constraints relaxed, it is important to note that approach proposed in Drohmann et al. (2012) yields accurate reduced-order models in operating scenarios for which they are trained [20, 21].

First introduced in Rewienski and White (2003) [44], Trajectory Piecewise Linearization (TPWL) simulates linearizations constructed around precomputed solution
trajectories. In the TPWL framework, linearizations are computed offline in previously simulation in a process called “training”. Online computations then approximate the simulation with the linearization most representative of the dynamics of interest.

Storage of these linearizations can be prohibitive in their size as Jacobians dependent on time and space need to be stored. To alleviate both the storage costs as well as the simulation cost of the linearization, model order reduction is introduced to make an approximate evaluation of the linearization affordable. Proper Orthogonal Decomposition with Trajectory Piecewise Linear approximations (POD-TPWL) is a popular method for this approach; see Ansari (2014) [3], Cardoso and Durlofsky (2010) [10], and Rousset et al. (2014) [45]. Jansen and Durlofsky (2017) [33] compare both approaches, POD-TPWL and POD, to accelerate well rate optimization for two-phase porous media flows.

Trehan and Durlofsky (2016) [53] extend the POD-TPWL to a Trajectory Piecewise Quadratic approximation with POD model reduction (POD-TPWQ) for optimization with implicit constraints. Their approach uses the POD-TPWL approximation for fast simulation and the POD-TPWQ approximation to indicate when retraining of both is necessary. They implement their method for a well rate optimization with a reservoir simulation as the implicit constraint.

Finally, the third question remains largely unanswered for reservoir simulation to the best of my understanding. Developing an online method to detect failure of the ROM to approximately reproduce FOM results in the reservoir management setting is an open challenge. A posteriori error bounds have been developed in many papers for linear/quadratic problem formulations with linear elliptic or linear parabolic implicit constraints by Karcher and Grepl (2014) [35] and Tröltzsch and Volkwein (2009) [54]. A generalization of this approach has been applied to semilinear parabolic problems with nonlinear reaction terms by Kamman et al. (2013) [34]. However, generalization of these results to the reservoir simulation problem is far from trivial.
Therefore, with these open questions regarding the fidelity and reliability of reduced-order models for porous media flows and reservoir simulation, it is not clear that reduced-order models can be used to replace the implicit constraint in optimization contexts effectively. Instead, my approach approximates Hessian-vector computations with subspace-based model reduction techniques. In doing so, I compute the gradient which requires full-order state and adjoint solves and recycle their solution trajectories to approximate the action of the Hessian. Using my approach the optimizations exhibit fast convergence to the minimizer for the cost of the Gradient Method.

1.3 Overview

In Chapter 2, I present an abbreviated discussion of the prerequisite background of my project: large-scale optimization with implicit PDE constraints, subspace-based model order reduction, Proper Orthogonal Decomposition and the Discrete Empirical Interpolation Method, two-phase incompressible flows in porous media and finite volume discretization of the reservoir equations. I also introduce an elementary discussion of the modeling two-phase incompressible flow through porous media and a frequently used finite volume discretization of the reservoir model.

Chapter 3 introduces a motivating negative result in model order reduction in reservoir simulation. This result is generated from a benchmark synthetic dataset in reservoir simulation, the top layer of the SPE10 model. I show that construction of effective ROMs over a range of well rates and operating conditions is challenging, if not impossible. I argue that the existing approaches to introduce model order reduction and optimization are not well-suited for reservoir simulation, motivating my alternative approach to model reduction in optimization.

Chapter 4 develops the theory and justification for my approach. I develop performance guarantees for the error introduced by the ROM Hessian using a subspace-based error analysis akin to the error analyses found in Kunisch and Volkwein (2001)
Chapter 5 proposes a non-intrusive implementation of DEIM for semilinear parabolic PDEs discretized via finite elements. To make the implementation of DEIM precomputation nonintrusive, I introduce a Lagrange interpolation operator for the nonlinear reaction term. I develop \textit{A priori} error analysis for this approach and present numerical results to support it. Using the non-intrusive precomputation proposed in Chapter 5, I implement my ROM Hessian Newton’s Method on a semilinear parabolic optimal control problem in Chapter 6. Results for the semilinear optimal control problem demonstrate convergence comparable to Newton’s Method for the cost of the Gradient method.

Finally, Chapter 7 describes the model reduction techniques currently proposed for oil-water reservoir flooding simulations. I formulate the well rate optimization problem and an implementation of my ROM Hessian Newton’s Method for this application. Preliminary results demonstrate a significant acceleration of Newton’s method for the well rate optimization problem when using the modified Hessian.
Chapter 2

Background

This chapter serves as an amalgamation of existing prerequisite material for the proposed methods. Each section is intended to be individually contained and used as a reference for discussions in subsequent chapters. I address the following topics in this chapter: large-scale optimization with implicit PDE constraints, subspace-based model reduction, Proper Orthogonal Decomposition, the Discrete Empirical Interpolation Method, and two-phase incompressible flow through porous media.

2.1 Optimization with Implicit PDE Constraints

Large-scale optimization poses particular challenges in computational demand. In the optimization settings I am interested in, typically the number of optimization variables scale with the discretization of the problem. Instead, it is enough to observe the action of the Jacobian time vector to compute gradient and Hessian information. To introduce these ideas, I will adopt the notation and formulation from [29].

Minimize the unconstrained function $J(u)$:

$$\min_u J(u) := J(y(u), u)$$
where $y(u) \in \mathbb{R}^n_y$ is the solution of an implicit constraint,

$$c(y, u) = 0$$

given smooth objective function $J : \mathbb{R}^n_y \times \mathbb{R}^n_u \to \mathbb{R}$ and implicit constraint $c : \mathbb{R}^n_y \times \mathbb{R}^n_u \to \mathbb{R}^n_y$. To make the gradient and Hessian calculation rigorous, I restate the following assumptions from Heinkenschloss (2008) [29]:

1. For all $u$ there exists a unique $y \in \mathbb{R}^n_y$ such that $c(y, u) = 0$.

2. There exists an open set $D \subset \mathbb{R}^n_y \times \mathbb{R}^n_u$ with $\{(y, u) : c(y, u) = 0\} \subset D$ such that $J$ and $c$ are twice continuously differentiable on $D$.

3. The inverse $c_y^{-1}(y, u)$ exists for all $(y, u) \in \{(y, u) : c(y, u) = 0\}$.

With high-dimensional control spaces, explicit construction of the Hessian is infeasible due to both computational cost of construction and storage. The conjugate gradient method allows the user to instead observe only the action of the Hessian via Hessian-vector products. The Newton’s method when implemented with CG can be found in Algorithm 2.1.

Algorithm 2.1: Newton-CG Method with Armijo Line-Search

1: Compute $\nabla J(u)$.

2: while $\|\nabla J(u)\| < tol$ do

3: Apply Conjugate Gradient (CG) method to solve the Newton subproblem

$$\nabla^2 J(u)d = -\nabla J(u)$$

4: Perform Armijo line-search to find $\alpha \in (0, 1]$ sufficiently small enough that

$$J(u + \alpha d) \leq J(u) + 10^{-4} \alpha d^T \nabla J(u)$$

5: Set $u + \alpha d \rightarrow u$.

6: Compute $\nabla J(u)$.

7: end while
An important note here is that the Newton subproblem in step 3 is not solved exactly; rather, it solved iteratively via the Conjugate Gradient Method which does not require explicit construction of the Hessian \( \nabla^2 J(u) \). However, via inexact Newton theory from Dembo et al. (1982) [17], the stopping criterion for Conjugate Gradient can guarantee local convergence behavior. Specifically, I use

\[
\|\nabla^2 J(u) \cdot d + \nabla J(u)\| \leq \min\{\|\nabla J(u)\|^2, 0.01\|\nabla J(u)\|\},
\]

which guarantees quadratic local convergence. Alternative stopping criterion exist that guarantee linear and superlinear convergence instead. For a summary of Newton-CG and its variants, see Nocedal and Wright (2006) [42].

2.1.1 Adjoint-Based Derivative Computation

The gradient can be computed via the sensitivity matrix (a.k.a. Jacobian),

\[ S := y_u(u), \text{ where } S \in \mathbb{R}^{n_y \times n_u} \text{ satisfies} \]

\[ c_y(y(u), u)S = -c_u(y(u), u). \]

However, if \( n_u \) is large this can very expensive as the system of equations needs to be solved for each column of \( c_u(y(u), u) \). Instead, observe that the action of the adjoint, \( S^T \) is needed without its explicit construction: the gradient can be computed as the action of the sensitivity matrix,

\[ \nabla J(u) = S^T \nabla_y \mathcal{J}(y(u), u) + \nabla_u \mathcal{J}(y(u), u). \]

Hence, the action of \( S^T \) is sufficient without it’s explicit construction. The gradient is computed using the adjoint equation in Algorithm 2.2 below.
Algorithm 2.2: Gradient Computation using Adjoint

1: Given $u$, solve $c(y, u) = 0$ for $y$.
2: Solve the adjoint equation for $\lambda$.

$$c_y(y(u), u)^T \lambda = -\nabla_y J(y(u), u).$$

3: Compute

$$\nabla J(u) = \nabla_u J(y(u), u) + c_u(y(u), u)^T \lambda(u).$$

In a similar fashion, the Hessian $\nabla^2 J(u)$ does not need to be explicitly constructed. Instead, its action on an arbitrary vector $v$ is needed in the Conjugate Gradient method. Like the gradient computation using adjoints in Algorithm 2.2, the action of the Hessian on vector $v$ is expressed as the solution of PDE solves. See Algorithm 2.3. Recall the Lagrangian is defined $L(y, u, \lambda) := J(y, u) + \lambda^T c(y, u)$.

Algorithm 2.3: Hessian-vector Product

1: Given $u$, solve $c(y, u) = 0$ for $y$.
2: Solve the adjoint equation for $\lambda$.

$$c_y(y(u), u)^T \lambda = -\nabla_y J(y(u), u).$$

3: Solve the equation for $w$

$$c_y(y(u), u)w = c_u(y(u), u)v.$$

4: Solve the equation for $p$

$$c_y(y(u), u)^T p = \nabla_{yy} L(y(u), u, \lambda(u))w - \nabla_{yu} L(y(u), u, \lambda(u))v.$$

5: Compute

$$\nabla^2 J(u)v = c_u(y(u), u)^T p - \nabla_{yu} L(y(u), u, \lambda(u))w$$
$$+ \nabla_{uu} L(y(u), u, \lambda(u))v.$$
Note that the first two steps of Algorithm 2.3 do not depend on \( v \). Further, in the Newton-CG method, Algorithm 2.1, these first two steps will already be completed in the computation of the gradient \( \nabla J(u) \) in Algorithm 2.2. Hence, for a fixed control \( u \) and solutions to the state and adjoint equations, \( y(u) \) and \( \lambda(u) \), it’s only necessary to solve two PDE solves: steps 3 and 4 per \( v \) (per CG iteration).

This framework for large-scale optimization scales better than traditional Newton’s Method implementations in both storage and computation cost. A more detailed discussion of these ideas can be found in Heinkenschloss (2008) [29] where the computations are derived carefully. The above methodology for large-scale optimization with implicit PDE constraints will become the benchmark by which I will compare the method proposed in this project.

### 2.2 Subspace-Based Model Order Reduction

Model order reduction seeks to alleviate the computational burden of multiple simulations by replacing the full order system with a problem of lesser dimension.

**Model Problem** Denote the full order semi-discretized system with \( y(t) \in \mathbb{R}^{n_y} \), \( F: \mathbb{R}^{n_y} \to \mathbb{R}^{n_y} \), \( G: \mathbb{R}^{n_u} \to \mathbb{R}^{n_y} \) and invertible matrix \( M \in \mathbb{R}^{n_y \times n_y} \),

\[
M \frac{dy(t)}{dt} = F(y(t)) + G(u(t)), \quad y(0) = y_0.
\] (2.2.1)

The dynamical system above can represent a wide range of discretizations (e.g. Finite Difference, Finite Element, Finite Volume, etc.) for a variety of problems. The formulation assumes that mixed derivatives vanish (i.e. no terms depend on state \( y(t) \) and \( u(t) \) simultaneously. We are concerned with computational settings where solving the full order system can be prohibitively expensive.

Subspace-based reduced order modeling assumes that the full-order solution trajectory \( y(t) \) is well approximated by a projection of the system dynamics onto a
low-dimensional manifold. Projection based reduced order models compute low dimensional subspaces for the state trajectory \( y(t) \) and project the equations on these subspaces. More specifically, they compute \( M \)-orthogonal matrices \( V \in \mathbb{R}^{n_y \times n_r} \) with \( n_r \ll n_y \), such that the columns of these matrices are bases for the solution subspace. The methods seek approximations of the form

\[
y(t) \approx V \hat{y}(t), \quad t \in [0, T].
\]

where \( \hat{y}(t) \in \mathbb{R}^{n_r} \) is a low dimensional trajectory. Without loss of generality, columns of \( V \) are chosen to be \( M \)-orthogonal so that \( V^T M V = I \).

Projection-based model order reduction occurs in two steps. First, substitute \( y(t) \) with its approximations \( V \hat{y}(t) \) full order system, (2.2.1). Second, force the error this introduces to be perpendicular to the subspace \( \text{Ran}(V) \),

\[
\frac{d}{dt} MV \hat{y}(t) - F(V \hat{y}(t)) - G(u(t)) \perp \text{Ran}(V).
\]

This leads to the reduced order formulation: given \( \hat{y}_0 = V^T M y_0 \), find \( \hat{y}(t) \in \mathbb{R}^{n_r} \) such that

\[
\frac{d}{dt} \hat{y}(t) = V^T F(V \hat{y}(t)) + V^T G(u(t)), \quad t \in [0, T]. \tag{2.2.2}
\]

**Backward Euler Discretization in Time** Given the full order system,

\[
M \frac{y_{k+1} - y_k}{\Delta t} = F(y_{k+1}) + G(u_{k+1}), \quad y_0 = y_{\text{given}}, \tag{2.2.3}
\]

we construct a reduced order system by the usual subspace-based model reduction. That is, we seek \( V \in \mathbb{R}^{n \times \ell} \) such that \( V^T M V = I \) and the error

\[
\sum_{k=1}^{n_t} \| y_k - V \hat{y}_k \|_M^2
\]

is small, where \( (\hat{y}_1, \ldots, \hat{y}_{n_t}) \in (\mathbb{R}^\ell)^{n_r} \) is the solution to the reduced order problem,

\[
\frac{\hat{y}_{k+1} - \hat{y}_k}{\Delta t} = V^T F(V \hat{y}_{k+1}) + V^T G(u_{k+1}), \quad \hat{y}_0 = V^T M y_{\text{given}}
\]
Define
\[
\hat{F}(y) := V^T F(V\hat{y}),
\]
\[
\hat{G}(u) := V^T G(u).
\]

Then the reduced order system can be written,
\[
\frac{\hat{y}_{k+1} - \hat{y}_k}{\Delta t} = \hat{F}(\hat{y}_{k+1}) + \hat{G}(u_{k+1}), \quad \hat{y}_0 = V^T M y_{\text{given}}.
\] (2.2.4)

The natural question is which \( V \in \mathbb{R}^{ny \times nr} \) will result in high-fidelity approximations, \( \hat{y}_k \approx V \hat{y}_k \). First we propose a performance guarantee for subspace-based model reduction adapted from Chaturantabut and Sorensen [14]. Then we introduce a popular approach for selecting these subspaces, the Proper Orthogonal Decomposition, (POD).

### 2.2.1 Performance Guarantees

Chaturantabut and Sorensen (2012) [14] provide an error analysis for the subspace-based model reduction method outlined in the previous subsection when the nonlinearity is simplified further with an interpolation method they develop call the Discrete Empirical Interpolation Method which is discussed later in this chapter. Their analysis provides an error bound for the unweighted Euclidean norm \( \mathbb{R}^n \) for dynamical systems with no mass matrix (e.g. \( M \equiv I \)). Theorem 2.1 walks through their analysis for dynamical systems that include a mass matrix \( M \) in the \( \mathbb{R}^n(M) \)-norm and ignoring the interpolation necessary to accelerate model reduction for now.

Define the *logarithmic Lipschitz constant* for the weighted inner product in \( \mathbb{R}^n \),
\[
\langle u, v \rangle_M = u^T M v \quad \text{for all} \quad u, v \in \mathbb{R}^n.
\]

Define a function \( F \) to be *locally Lipschitz continuous* if for every compact set \( K \subset R^n \), there exists a Lipschitz constant \( L \) such that for \( u, v \in K \),
\[
\|F(u) - F(v)\| \leq L\|u - v\|.
\]
Theorem 2.1. Let \( \{y_k\} \) and \( \{\hat{y}_k\} \) denote the full and reduced solutions to (2.2.3) and (2.2.4) respectively where \( V \in \mathbb{R}^{n \times \ell} \) satisfies \( V^T MV = I \). Define \( \hat{F}(\hat{z}) := V^T F(V\hat{z}) \) and assume that \( \Delta t M_1(\hat{F}) < 1 \). Then

\[
\sum_{k=1}^{n_t} \|y_k - V\hat{y}_k\|_M^2 \leq \left[ 1 + \Delta t L_M^2 \zeta^2 \left( \frac{1 - \zeta}{1 - \zeta^2} \right) \right] \sum_{k=1}^{n_t} \|(I - VV^T M)y_k\|_M^2
\]

where \( \zeta = (1 - \Delta t M_1(\hat{F}))^{-1} > 0 \) and \( L_M \) denotes the Lipschitz constant of \( \tilde{F}(y) := M^{-1/2} F(M^{-1/2} y) \) for the union of the line segments,

\[
\bigcup_{k=1}^{n_t} \{\alpha y_k + (1 - \alpha) VV^T M y_k : \alpha \in [0,1]\},
\]

for \( k = 0, \ldots, n_t \). That is,

\[
\|F(y_k) - F(VV^T M y_k)\| \leq L_M \|y_k - VV^T M y_k\|_M, \quad \text{for } k = 1, \ldots, n_t.
\]

Proof. First, by construction, we choose \( V \in \mathbb{R}^{n \times r} \) so that \( V^T MV = I \). Define the error

\[
e_k := y_k - V\hat{y}_k = y_k - VV^T M y_k + VV^T M y_k - V\hat{y}_k = \rho_k + \theta_k
\]

where

\[
\rho_k := y_k - VV^T M y_k,
\]
\[
\theta_k := VV^T M y_k - V\hat{y}_k.
\]

By construction, \( \rho_k \) and \( \theta_k \) are M-conjugate, \( \langle \rho_k, \theta_k \rangle_M = 0 \). Define

\[
\hat{\theta}_k := V^T M \theta_k.
\]

Then

\[
V\hat{\theta}_k = VV^T M \theta_k = \theta_k
\]

and consequently \( \|\hat{\theta}_k\| = \|\theta_k\|_M \). To see this,

\[
\|\hat{\theta}_k\|^2 = \theta_k^T M V V^T M \theta_k = \theta_k^T M \theta_k = \|\theta_k\|_M^2.
\]
\[ V^T M \left( \frac{y_{k+1} - y_k}{\Delta t} \right) = V^T F(y_{k+1}) + V^T G(u_{k+1}) \]
\[ = \hat{F}(V^T M y_{k+1}) + V^T F(y_{k+1}) - \hat{F}(V^T M y_{k+1}) + G(u_{k+1}) \]
\[ = \hat{F}(V^T M y_{k+1}) + \hat{R}_{k+1} + G(u_{k+1}), \]
where \( \hat{R}_{k+1} := V^T F(y_{k+1}) - \hat{F}(V^T M y_{k+1}) \). Then using,
\[ \hat{\theta}_k = V^T M \theta_k = V^T M (V V^T M y_k - V \hat{y}_k) \]
we can take the finite difference quotient,
\[ \frac{\hat{\theta}_{k+1} - \hat{\theta}_k}{\Delta t} = V^T M \left( \frac{y_{k+1} - y_k}{\Delta t} \right) - \left( \frac{\hat{y}_{k+1} - \hat{y}_k}{\Delta t} \right) \]
\[ = \hat{F}(V^T M y_{k+1}) + \hat{R}_{k+1} - \hat{F}(\hat{y}_{k+1}). \]

By the Cauchy-Schwarz inequality, \(-\| \hat{\theta}_k \| \leq -\langle \hat{\theta}_{k+1}, \hat{\theta}_k \rangle \| \hat{\theta}_{k+1} \| \),
\[ \frac{\| \hat{\theta}_{k+1} \| - \| \hat{\theta}_k \|}{\Delta t} \leq \frac{1}{\Delta t} \left( \frac{\langle \hat{\theta}_{k+1}, \hat{\theta}_{k+1} \rangle}{\| \hat{\theta}_{k+1} \|} - \frac{\langle \hat{\theta}_{k+1}, \hat{\theta}_k \rangle}{\| \hat{\theta}_{k+1} \|} \right) \]
\[ = \frac{1}{\| \hat{\theta}_{k+1} \|} \left( \hat{\theta}_{k+1}, \frac{\hat{\theta}_{k+1} - \hat{\theta}_k}{\Delta t} \right) \]
\[ = \frac{1}{\| \hat{\theta}_{k+1} \|} \left( \hat{\theta}_{k+1}, \hat{F}(V^T M y_{k+1}) - \hat{F}(\hat{y}_{k+1}) + \hat{R}_{k+1} \right) \]
\[ \leq M_1 \| \hat{F} \| \| \hat{\theta}_{k+1} \| + \| \hat{R}_{k+1} \|. \]

Fix \( \zeta = \frac{1}{1 - \Delta t M_1 \| \hat{F} \|} \). Then using \( \| \theta_k \|_M = \| \hat{\theta}_k \| \), we get
\[ \| \theta_{k+1} \|_M = \| \hat{\theta}_{k+1} \| \leq \zeta (\| \hat{\theta}_k \| + \Delta t \| \hat{R}_{k+1} \|) \]
\[ \leq \zeta^k \| \hat{\theta}_0 \| + \Delta t \sum_{\ell=1}^{k+1} \zeta^\ell \| \hat{R}_{k-\ell+2} \| \]
\[ \leq \Delta t \left( q_{k+1} \sum_{\ell=1}^{k+1} \| \hat{R}_{\ell} \|^2 \right)^{1/2}, \]
where $q_k = \sum_{\ell=1}^{k} \zeta^{2\ell}$. Then we can bound the term $\|\hat{R}_\ell\|$, 

$$
\|\hat{R}_\ell\| = \|V^T F(y_\ell) - \hat{F}(V^T M y_\ell)\| 
\leq \|M^{1/2} V\| \|M^{-1/2} [F(y_\ell) - F(VV^T M y_\ell)]\|_M 
\leq L_M \|y_\ell - VV^T M y_\ell\|_M 
= L_M \|\rho_\ell\|_M.
$$

where $L_M$ is the Lipschitz constant in the theorem statement and we know $\|M^{1/2} V\| = 1$ since the matrix is subunitary ($V^T MV = I$). Define 

$$
q := q_{nt} = \sum_{\ell=1}^{nt} \zeta^{2\ell} = \zeta^2 \left( \frac{1 - \zeta^{2nt}}{1 - \zeta^2} \right).
$$

Then 

$$
\|\theta_k\|_M^2 \leq \Delta t^2 q \left( \sum_{\ell=1}^{nt} \|\hat{R}_\ell\|^2 \right) \leq \Delta t^2 q L_M^2 \left( \sum_{\ell=1}^{nt} \|\rho_\ell\|^2 \right).
$$

Finally, we arrive at the bound, 

$$
\sum_{k=0}^{nt} \|e_k\|_M^2 = \sum_{k=0}^{nt} \|\rho_k\|_M^2 + \|\theta_k\|_M^2 \leq (1 + \Delta t^2 q L_M^2) \left( \sum_{k=1}^{nt} \|\rho_k\|_M^2 \right).
$$

The key observation is that the subspace-based error $\sum_{k=1}^{nt} \|y_k - V\hat{y}_k\|_M^2$ is bounded by a multiple of the projection error in the state $\sum_{k=1}^{nt} \|(I - VV^T M)y_k\|_M^2$. The Proper Orthogonal Decomposition method seeks to minimize the latter.

### 2.3 Proper Orthogonal Decomposition

The most popular model reduction scheme for nonlinear models is via Proper Orthogonal Decomposition (POD). See, e.g., Hinze and Volkwein (2005) [31].
This thesis uses POD in the context of the discretized, finite dimensional equation (2.2.3). Recall the weighted inner product
\[ \langle v, w \rangle_M = v^T M w. \]
Given samples of the solution, \( \{ y_1, \ldots, y_{n_t} \} \subset \mathbb{R}^n \), define the snapshot matrix
\[ Y = [y_1, \ldots, y_{n_t}] \in \mathbb{R}^{n \times n_t}. \]
For given integer \( \ell \ll \min\{\ell, n_t\} \), POD computes an \( M \)-orthogonal matrix \( V \in \mathbb{R}^{n \times \ell} \) as the solution of
\[ \min_{V \in \mathbb{R}^{n \times \ell}} \sum_{k=1}^{n_t} \| (I - VV^T M) y_k \|_M^2. \]  
A necessary condition for optimality for the minimization (2.3.1) takes the form of a generalized eigenvalue problem,
\[ M Y Y^T M v = \lambda M v, \quad Y = [y_1, \ldots, y_{n_t}] \in \mathbb{R}^{n \times n_t}. \]  
Notice the columns of \( V \in \mathbb{R}^{n \times \ell} \) are the dominant generalized eigenvectors of (2.3.2). Further, the global basis function \( \psi_j \) are orthogonal in \( X \) which implies that \( V \in \mathbb{R}^{n \times \ell} \) satisfies \( V^T M V = I_\ell \).

The solution to the generalized eigenvalue problem (2.3.2) can be computed via a singular value decomposition (SVD) of \( M^{1/2} Y \) where the singular values \( \sigma_i^2(M^{1/2} Y) \) are the eigenvalues of (2.3.2). If \( M^{1/2} Y = L \Sigma R^T \) is the SVD with singular values \( \sigma_1 \geq \sigma_2 \geq \ldots \geq 0 \) and \( \Sigma = \text{diag}(\sigma_1, \sigma_2, \ldots) \), then a solution of (2.3.1) is given by \( V \in \mathbb{R}^{n \times \ell} \) is the submatrix consisting of the first \( \ell \) columns of \( L \in \mathbb{R}^{n \times n} \).

Furthermore, for a solution \( V \) of (2.3.1),
\[ \min_{V \in \mathbb{R}^{n \times \ell}} \sum_{k=1}^{n_t} \| (I - VV^T M) y_k \|_M^2 = \sum_{i=\ell+1}^{\min\{n, n_t\}} \sigma_i^2(M^{1/2} Y), \]
This identity can be used to determine the dimension \( \ell \) of the subspace. Given a
tolerance $\text{tol}_{\text{POD}} > 0$ choose the smallest $\ell$ such that the relative error

$$
\frac{\sum_{k=1}^{n_t} \| (I - VV^T M) y_k \|^2_M}{\sum_{k=1}^{n_t} \| y_k \|^2_M} = \frac{\| M^{1/2}(I - VV^T M) Y \|^2_F}{\| M^{1/2} Y \|^2_F} = \frac{\sum_{i=\ell+1}^{\min\{n,n_t\}} \sigma_i^2(M^{1/2}Y)}{\min\{n,n_t\} \sum_{i=1}^{\min\{n,n_t\}} \sigma_i^2(M^{1/2}Y)} < \text{tol}_{\text{POD}}. \tag{2.3.3}
$$

In practice, computation of $\text{svd}(M^{1/2}Y)$ is prohibitive in large-scale simulations for two reasons: (1) the size of $M^{1/2}Y \in \mathbb{R}^{n \times n_t}$ is enormous, and (2) the square root factorization of the mass matrix $M^{1/2}$ may not be feasible for non-diagonal mass matrices (e.g., finite element discretizations). Therefore only the leading singular values are computed in practice and many of the neglected singular values in (2.3.3) are expensive to compute. As an alternative, we can choose $\ell$ by replacing the Frobenius norms in (2.3.3) with 2-norms,

$$
\frac{\| M^{1/2}(I - VV^T M) Y \|_2}{\| M^{1/2} Y \|_2} = \frac{\sigma_{\ell+1}}{\sigma_1} < \text{tol}_{\text{POD}}.
$$

In my numerical results I choose the latter, since in practice this does not require computation of the entire SVD, but only of the first few singular values and corresponding singular vectors, which for large problems can be done using, e.g., ARPACK [37]. The POD basis computation is summarized in Algorithm 2.4, but note that in practice it is not necessary to compute the full SVD of $M^{1/2}Y$. 
Algorithm 2.4: POD Basis Computation

1: Given snapshot matrix $Y = [y_1, \ldots, y_{n_t}] \in \mathbb{R}^{n \times n_t}$, and a tolerance $\text{tol}_{\text{POD}} > 0$, compute the square root of the mass matrix, $M^{1/2}$.

2: Compute the SVD $M^{1/2}Y = L \Sigma R^T$. Let $\sigma_1 \geq \sigma_2 \geq \ldots \geq 0$ denote the singular values.

3: Find the smallest integer $\ell$ such that $\sigma_{\ell+1}/\sigma_1 < \text{tol}_{\text{POD}}$.

4: Return $\ell$ and the matrix $V \in \mathbb{R}^{n \times \ell}$ consisting of the first $\ell$ columns of $R \in \mathbb{R}^{n \times n}$.

2.3.1 Numerical Computation of POD Subspace

The following discussion summarizes Hinze and Volkwein [31]. Denote the snapshot matrix $Y \in \mathbb{R}^{n \times n_t}$. Depending on the dimension of $Y$, we consider two different approaches to solving the generalized eigenvalue problem.

Case $n \leq n_t$: We solve the generalized eigenvalue problem (2.3.2) by computing the generalized SVD. By Golub and Van Loan [27, p. 466; Theorem 8.7.4] we know there exist orthogonal $W \in \mathbb{R}^{n_t \times n_t}$ and $U \in \mathbb{R}^{n \times n}$ and invertible $R \in \mathbb{R}^{n \times n}$ such that

$$W(Y^T M)R = \Sigma_1$$  \hspace{1cm} (2.3.4)

$$UM^{1/2}R = \Sigma_2$$  \hspace{1cm} (2.3.5)

with $\Sigma_1 \in \mathbb{R}^{n_t \times n}$ and $\Sigma_2 \in \mathbb{R}^{n \times n}$ diagonal and nonnegative. Then substituting $R = M^{-1/2}U^T \Sigma_2$ into (2.3.4), we get

$$U^T M^{1/2}YW^T = \Sigma_2^{-1} \Sigma_1^T$$

Hence, we have computed an SVD of $M^{1/2}Y$ and we choose $V$ to be the columns of $M^{-1/2}U$ associated with the largest values of $\{\sigma_i\} = \text{diag} (\Sigma_2^{-1} \Sigma_1^T)$.

Case $n \geq n_t$: Instead, we solve the smaller eigenvalue problem, $Y^T MYu = \lambda u$, we can post-compute the left singular vectors of $M^{1/2}Y$. 
1. Solve the eigenvalue problem $Y^TMY = W\Sigma W^T$ with $W \in \mathbb{R}^{n_t \times n_t}$.

2. Truncate $W = [W_1, W_2]$ and $\Sigma = \text{blkdiag}(\Sigma_1, \Sigma_2)$ where $W_1 \in \mathbb{R}^{n_t \times \ell}$ and $\Sigma_1 \in \mathbb{R}^{\ell \times \ell}$.

3. Compute $V = YW_1\Sigma_1^{-1/2}$.

Both methods yield solutions to the generalized eigenvalue problem (2.3.2). However, is preferable depending on the dimensions of the snapshot matrix $Y$.

### 2.3.2 Performance Guarantees via POD

Performance guarantees for POD is a corollary of Theorem 2.1 on subspace-based model reduction in Section 2.2. As the POD method chooses to minimize the latter term in the performance guarantee,

$$\sum_{k=1}^{n_t} \|(I - VV^T)Y_k\|_M^2 = \sum_{i=1}^{\ell} \sigma_i^2(M^{1/2}Y),$$

(2.3.6)

where $V \in \mathbb{R}^{n \times \ell}$ satisfies $V^TMV = I$ and $\sigma_i^2(M^{1/2}Y)$ are the eigenvalues of the generalized eigenvalue problem in descending order,

$$MYY^TMu = \lambda Mu.$$

We get the error bound when $V$ is chosen via POD by applying (2.3.6) in Theorem 2.1.

**Corollary 2.2.** Let $V \in \mathbb{R}^{n \times \ell}$ satisfying $V^TMV = I$ be chosen via POD. Define $\hat{F}(\hat{z}) := V^TF(V\hat{z})$ and assume that $\Delta t\mathcal{M}_1[\hat{F}] < 1$. Then

$$\sum_{k=1}^{n_t} \|y_k - V\hat{y}_k\|_M^2 \leq \left[1 + \Delta tL_M^2\zeta^2 \left(\frac{1 - \zeta^{2n_t}}{1 - \zeta^2}\right)\right] \left(\sum_{i=1}^{\ell} \sigma_i^2(M^{1/2}Y)\right),$$

where $\zeta = (1 - \Delta t\mathcal{M}_1[\hat{F}])^{-1} > 0$ and $L_M$ denotes the Lipschitz constant of $F$ for the union of the line segments $\bigcup_{k=1}^{n_t} \{\alpha y_k + (1-\alpha)VV^TMy_k : \alpha \in [0, 1]\}$ for $k = 0, \ldots, n_t$. That is,

$$\|F(y_k) - F(VV^TMy_k)\| \leq L_M\|y_k - VV^TMy_k\|_M, \quad \text{for } k = 1, \ldots, n_t.$$

**Proof.** The proof immediately follows from Theorem 2.1 and (2.3.6).
2.4 Empirical Interpolation Method

Originally introduced in Barrault et al. (2004) [5], the Empirical Interpolation Method (EIM) was rapidly extended to model reduction frameworks for non-affine dependence on parameter and nonlinear PDEs in Grepl et al. (2007) [28] and Maday et al. (2007) [39] respectively.

The following description of the EIM summarizes the introduction to EIM found in Chapter 5 of Hesthaven et al. [30]. Given a compact parameter space \( \mathbb{P} \) and domain \( \Omega \) such that \( \overline{\Omega} \) is compact, EIM seeks to approximate a bivariate function \( g : \Omega \times \mathbb{P} \to \mathbb{R} \) where for fixed \( \mu \in \mathbb{P} \), the function \( g_\mu = g(\cdot, \mu) \in C^0(\overline{\Omega}) \).

EIM selects \( Q \) interpolation points in space and parameter denoted \( \{ x_1, \ldots, x_Q \} \) and \( \{ \mu_1, \ldots, \mu_Q \} \) and constructs an interpolation operator \( I_Q \) that satisfies,

\[
I_Q [g_\mu](x) = g_\mu(x), \quad i = 1, \ldots, Q, \quad \forall x \in \Omega,
\]
\[
I_Q [g_\mu](x_i) = g_\mu(x_i), \quad i = 1, \ldots, Q, \quad \forall \mu \in \mathbb{P}.
\]

The EIM interpolant is constructed as follows,

\[
I_Q [g_\mu](x_i) = \sum_{q=1}^Q c_q(\mu) h_q(x), \quad x \in \Omega.
\]

Here \( \{ c_1(\mu), \ldots, c_Q(\mu) \} \) is defined implicitly to satisfy the interpolation condition,

\[
g_\mu(x_j) = I_Q [g_\mu](x_j) = \sum_{q=1}^Q c_q(\mu) h_q(x_j),
\]

which can be conveniently expressed as the \( Q \)-dimensional linear system, \( \mathbf{T} \mathbf{c}_\mu = \mathbf{g}_\mu \),

\[
\begin{bmatrix}
  h_1(x_1) & \cdots & h_Q(x_1) \\
  \vdots & \ddots & \vdots \\
  h_1(x_Q) & \cdots & h_Q(x_Q)
\end{bmatrix}
\begin{bmatrix}
  c_1(\mu) \\
  \vdots \\
  c_Q(\mu)
\end{bmatrix}
= \begin{bmatrix}
  g_\mu(x_1) \\
  \vdots \\
  g_\mu(x_Q)
\end{bmatrix}.
\]

(2.4.1)

Note here that the functions \( \{ h_1(x), \ldots, h_Q(x) \} \) are chosen to form a basis for the family of function \( \{ g_{\mu_1}(x), \ldots, g_{\mu_Q}(x) \} \). As the matrix \( \mathbf{T} \) does not depend on
parameter or space, it need only be computed once and each value of $c_{\mu}$ can be computed by evaluating the function $g_{\mu}$ on $Q$ interpolation points and by solving (2.4.1) for each choice of parameter $\mu$.

The choice of basis functions $\{h_1(x), \ldots, h_Q(x)\}$ is significant and a key feature of the EIM framework. The EIM method chooses interpolation points and basis function in with a greedy selection process. See Algorithm 2.5.

**Algorithm 2.5: EIM, Following Hesthaven et al. (2016) [30, Section 5.2]**

1: Set $q \leftarrow 1$
2: while $\nu < \text{tol}$ do
3: Pick the sample point 
   \[ \mu_q = \arg \max_{\mu \in \mathcal{P}} \| g_{\mu} - I_{q-1}[g_{\mu}] \|_{L^\infty(\Omega)} \]
   and the corresponding interpolation point
   \[ x_q = \arg \max_{x \in \Omega} | g_{\mu_q}(x) - I_{q-1}[g_{\mu_q}](x) |. \]
4: Define the next basis function as the scaled error function
   \[ h_q = \frac{g_{\mu_q} - I_{q-1}[g_{\mu_q}]}{g_{\mu_q}(x_q) - I_{q-1}[g_{\mu_q}](x_q)}. \]
5: Define the error
   \[ \nu = \| \eta(\mu) \|_{L^\infty(\mathcal{P})}, \quad \text{with} \quad \eta(\mu) := \| g_{\mu} - I_{q-1}[g_{\mu}] \|_{L^\infty(\Omega)} \]
6: Set $q \leftarrow q + 1$.
7: end while

The basis functions $\{h_1(x), \ldots, h_Q(x)\}$ are chosen to be linearly independent and so that the matrix $T \in \mathbb{R}^{Q \times Q}$ is invertible and well-conditioned. Specifically, when chosen via Algorithm 2.5, $T$ is triangular and has a unit diagonal,

\[ T_{ii} = h_i(x_i) = 1, \quad i = 1, \ldots, Q, \]
\[ T_{ij} = h_j(x_i) = 0, \quad 1 \leq i < j \leq Q. \]
The EIM has been extended to other settings. This includes a Generalized Empirical Interpolation Method (GEIM) that replaces the spatial interpolation operator with linear functionals and matrix-valued implementations. However, by far the most popular extension of EIM is the Discrete Empirical Interpolation Method (DEIM). Originally introduced by Chaturantabut and Sorensen (2010, 2011, 2012) [12, 13, 14] as an extension of EIM for dynamical systems whose state is represented as a vector-valued function, the DEIM can be understood as a special case of EIM with the following changes made:

1. the family of functions \( \{g_\mu\} \subset \mathbb{R}^n \) is vector-valued; hence, the spatial interpolation operator is replaced with a row-selecting functional on \( \mathbb{R}^n \), and

2. basis functions \( \{h_1(x), \ldots, h_Q(x)\} \) are prescribed as the orthogonal vectors generated from POD whose span approximates the family of vectors \( \{g_\mu\} \subset \mathbb{R}^n \) up to some level of precision.

As the basis vectors are chosen from POD instead of Algorithm 2.5, the matrix \( T \in \mathbb{R}^{Q \times Q} \) is not triangular. However, Chaturantabut and Sorensen prove that \( T \) is orthogonal when interpolation indices are chosen via their greedy selection method [12, 14]. The following section outlines the DEIM framework when \( \mu \) and \( P \) are chosen to be temporal variables \( t \) and \( [0, T] \) respectively.

### 2.5 Discrete Empirical Interpolation Method

Since \( \mathbf{\hat{y}}(t) \in \mathbb{R}^{n_r} \) and with \( n_r \ll n_y \), the size of the system (2.2.2) is substantially smaller than that of (2.2.1). Note that if \( \mathbf{F} \) is nonlinear, then the cost of the evaluation \( \mathbf{\hat{y}}(t) \mapsto \mathbf{V}^T \mathbf{F}(\mathbf{V\hat{y}}(t)) \) still depends on the size \( n_y \) of the full order model. Thus to arrive at a reduced order system in which the cost of evaluation of all terms depends only on \( n_r \ll n_y \), but not on \( n_y \), further approximations such as the Discrete Empirical Interpolation Method (DEIM) (see, e.g., Chaturantabut and Sorensen (2010) [12], need to be applied).
The original formulation for DEIM in Chaturantabut and Sorensen (2010) [12] considers the vector-valued function $f(t) := F(y(t)) \in \mathbb{R}^{n_y}$. The method projects $f(t)$ onto the subspace spanned by the basis $\{u_1, \ldots, u_m\}$ by oblique projection $U(P^TU)^{-1}P^T$ so that

$$\hat{f}(t) := U(P^TU)^{-1}P^T f(t) \approx f(t),$$

where $U = [u_1, \ldots, u_m] \in \mathbb{R}^{n_y \times m}$ has orthogonal columns and $P = [e_1, \ldots, e_{n_y}]$ consists of columns of the identity. The advantage of introducing this approximation is that the product $P^T f(t)$ need only be evaluated on $m \ll n_y$ elements of the vector $f(t)$ as $P^T$ selects $m$ rows of the vector.

In practice, $U$ is chosen so that its range approximately contains the trajectory $f(t)$ by collecting snapshots of the nonlinear term and constructing the snapshot matrix $[f(t_1), \ldots, f(t_{n_t})]$ and performing Proper Orthogonal Decomposition via a truncated SVD. The matrix $U$ then is the $m$ leading left singular vectors of the snapshot matrix. In doing so, the error guarantee is determined by the neglected singular values,

$$\sum_{k=1}^{n_t} \|f(t_k) - UU^T f(t_k)\|^2 = \sum_{i=m+1}^{\min\{n_y, n_t\}} \sigma_i^2([f(t_1), \ldots, f(t_{n_t})]).$$

The DEIM method provides an algorithm to construct $P \in \mathbb{R}^{n_y \times m}$ so that $f(t) - \hat{f}(t)$ is small and $P$ remains columns of the identity; See Chaturantabut and Sorensen (2010, 2012) [12, 14]. These papers provide (1) a guarantee that $P^TU$ is invertible and (2) a bound on the error $\|\hat{f}(t) - f(t)\|$. 
Algorithm 2.6: Discrete Empirical Interpolation Method (DEIM)

1: \([|\rho|, \varrho_1]\) = max \{ |u_1| \}.
2: \(U = [u_1], P = [e_{\varrho_1}], \vec{\varrho} = [\varrho_1] \).
3: for \(\ell = 2\) to \(m\) do
4: Solve \((P^T U)c = P^T u_\ell\) for \(c\)
5: \(r = u_\ell - Uc\)
6: \([|\rho|, \varrho_1]\) = max \{ |r| \}.
7: \(U \leftarrow [U u_\ell], P \leftarrow [P e_{\varrho_\ell}], \vec{\varrho} \leftarrow \begin{bmatrix} \vec{\varrho} \\ \varrho_\ell \end{bmatrix} \)
8: end for

Lemma 2.3 (Chaturantabut and Sorensen (2010, 2012) [12, 14]). If \(U^T U = I\), and \(P\) is chosen according to Algorithm 2.6 then

\[
\|f(t) - \hat{f}(t)\| \leq C \| (I - UU^T) f(t) \|
\]

where \(C = \| (P^T U)^{-1} \| < (1 + \sqrt{2n})^{m-1} \|u_1\|_\infty^{-1} \).

While DEIM works well for vector-valued functions like \(f(t)\), in practice, DEIM is used in conjunction with POD to lessen the cost of the nonlinear function evaluations \(V^T F(V\hat{y}_k)\) whose cost can be large evaluating the nonlinearity on the vector \(V\hat{y}_k \in \mathbb{R}^{n_y}\). The approximation of the nonlinear term becomes,

\[
V^T F(V\hat{y}(t)) \approx V^T U (P^T U)^{-1} P^T F(V\hat{y}(t)),
\]

where the cost of evaluating \(P^T F(V\hat{y}(t))\) is small evaluating only \(m\) elements of the \(\mathbb{R}^{n_y}\)-valued function \(F\) and the matrices \(V^T U \in \mathbb{R}^{n_r \times m}\) and \((P^T U)^{-1} \in \mathbb{R}^{m \times m}\) can be precomputed.

The assumption here is that each row of \(F(y)\) is sparsely dependent on the elements of \(y\). Therefore relatively few elements of \(y\) need be considered, resulting in the speed-up demonstrated in these methods. In practice, if \(F\) represents the nonlinear term in any standard discretization scheme (finite difference, finite element, finite volume, etc.) this is certainly the case.
As a special case, some nonlinear mappings $y \mapsto F(y)$ can be evaluated element-wise, $F_i(y) = F(y_i)$ (e.g. nonlinear reaction terms finite difference and finite volume discretization of parabolic PDEs). Then the row selection matrix $P$ distributes, $P^T F(y) = F(P^T y)$ making the nonlinear evaluations $O(m)$. However, for the finite element and finite volume discretization found in here, this is not the case.

More efficient implementations of DEIM for finite elements have been proposed authors such as Tiso et al. (2013) [52] and comparisons of the implementations have been discussed by Antil et al. (2014) [4]. A more detailed discussion of this can be found in Section 5.4 of this document.

2.6 Two-Phase, Incompressible Flows in Porous Media

Excellent introductions to computational modeling of porous media flows can be found in Aarnes et al. (2007) and Chen et al. (2006) [1, 15]. In this section, I introduce a consolidated version of the governing equations for multi-phase flow in porous media as it pertains to the reservoir conditions and simulations found later. The following assumptions are made in this section: (1) there are only two phases, oil and water, (2) the phases are incompressible, and (3) permeability and porosity are not uniform, but depend on space only.

2.6.1 Darcy’s Law and Permeability

Darcy’s law states that the fluid velocity through a porous medium is proportional to the pressure across the volume. The constant of proportionality is defined to be the intrinsic permeability of the medium, $\kappa(x)$ divided by the viscosity of the fluid, $\mu$. Permeability measured in SI units ($m^2$) or Darcy’s ($D$), ($1D = 9.869233 \times 10^{13} \, m^2$).
Viscosity is measured in SI units, \( kg/(s \cdot m) \). Darcy’s law then reads,

\[
v = \frac{\kappa(x)}{\mu} \nabla p. \tag{2.6.1}
\]

where pressure is measured in SI units \( (N/m^2) \) or \( (kg/(m \cdot s^2)) \) and velocity is measured in SI units \( (m/s) \). Darcy’s Law is stated for 1-dimensional flows through volumes uniform in cross-sectional area. However, porous media flows are formulated in \( d \in \{1, 2, 3\} \) dimensions depending on the application. Consequently, the permeability field \( \kappa(x) \) is an \( \mathbb{R}^{d \times d} \) diagonal matrix instead, written

\[
K(x) = \begin{bmatrix}
\kappa_{x_1}(x) \\
\vdots \\
\kappa_{x_d}(x)
\end{bmatrix}
\]

where \( x \) is defined on a rectangular domain, \( \Omega \subset \mathbb{R}^d \) and \( \kappa_{x_i}(x) \) denotes the directional permeability in the Cartesian \( x_i \) direction. Then the velocity term \( v \) becomes vector-valued in \( \mathbb{R}^d \) and the multidimensional analog of Darcy’s Law, (2.6.1) for anisotropic permeability is

\[
v = \frac{1}{\mu} K(x) \nabla p. \tag{2.6.2}
\]

### 2.6.2 Porosity

The porosity \( \phi(x) \in [0, 1] \) is defined to be the ratio of the pore volume to the total volume. Hence, porosity measures of void space in the media at a particular point in space. The porosity and permeability fields are determined by the rock material characteristics and are often fixed input data in reservoir simulations.
2.6.3 Relative Permeability and Mobility

In the presence of two-phase flows the permeability fields are scaled by a factor called relative permeability. Relative permeability takes on (unitless) scalar values between 0 and 1 depending on the saturation of the two-phases. In practice, the relative permeability curves are determined experimentally via a process called core flooding. In the absence of this experimental data, oftentimes the relative permeability curves are assumes to be quadratic.

Denote the effective saturation, $s^* \in [0, 1]$ to be a rescaling of the saturation values from the minimum water saturation, $s_{wr} \in [0, 1)$, and the minimum oil saturation, $s_{or} \in [0, 1)$,

$$s^* = (s_w - s_{wr})/(1 - s_{or} - s_{wr})$$
where $s_w$ is the water saturation. Define quadratic relative permeability curves,

$$kr_o(s_w) = (1 - s^*)^2, \quad kr_w(s_w) = (s^*)^2,$$

when $s_{or} = 0.2, s_{wr} = 0.2$. For values $s_w$ outside $[0.2, 0.8]$ the relative permeabilities are extended continuously by zero or one. See Figure 2.2.

Incorporating the relative permeabilities in the modified version of Darcy’s Law, there are now two equations for fluid velocity, one for each phase, water and oil respectively

$$v_w = \frac{kr_w(s_w)}{\mu_w} K(x) \nabla p = \lambda_w(s_w) K(x) \nabla p, \quad \lambda_w(s_w) := \frac{kr_w(s_w)}{\mu_w}, \quad (2.6.3)$$

$$v_o = \frac{kr_o(s_w)}{\mu_o} K(x) \nabla p = \lambda_o(s_w) K(x) \nabla p, \quad \lambda_o(s_w) := \frac{kr_o(s_w)}{\mu_o}. \quad (2.6.4)$$

where $\lambda_w(s_w)$ and $\lambda_o(s_w)$ are the mobility of water and mobility of oil respectively.

### 2.6.4 Conservation of Mass and Volume Balance

First, assume a volume balance: all available space in the media is occupied by some combination of the two phases,

$$s_w + s_o = 1,$$
where $s_w$ and $s_o$ are the saturations of water and oil respectively. Conservation of mass guarantees that the net flow of each phase (divergence of the velocity) must be in equilibrium with changes in saturation and sources/sinks for the phase. That is,
\[
\phi(x) \frac{\partial s_w}{\partial t} + \nabla \cdot v_w = q_w(s_w), \quad \text{on } \Omega \times (0, T) \quad (2.6.5)
\]
\[
\phi(x) \frac{\partial (1 - s_w)}{\partial t} + \nabla \cdot v_o = q_o(s_w) \quad \text{on } \Omega \times (0, T). \quad (2.6.6)
\]
where source terms $q_w$ and $q_o$ account for injection and production of water and oil respectively,
\[
q_w(s_w) = \begin{cases} 
s_{in} q(x, t), & q(x, t) > 0 \\
 f(s_w) q(x, t), & q(x, t) < 0
\end{cases}
\]
\[
q_o(s_w) = \begin{cases} 
(1 - s_{in}) q(x, t), & q(x, t) > 0 \\
 f(1 - s_w) q(x, t), & q(x, t) < 0
\end{cases}
\]
with $0 \leq s_{in} \leq 1$ denoting the saturation of the injected water and $q(x, t)$ denoting the injection and production rates in space and $f$ denotes the fractional flow,
\[
f(s_w) = \frac{\lambda_w(s_w)}{\lambda_w(s_w) + \lambda_o(s_w)}.
\]
Next, under the assumption of incompressibility, the flow rates must be in equilibrium to maintain the net volume in the reservoir,
\[
\int_{\Omega} q(x, t) d\mathbf{x} = 0.
\]
\section{2.6.5 PDE System}

Impose a no-flow boundary condition by setting the flux to 0 on the boundary,
\[
\mathbf{v}(x, t) \cdot \mathbf{n} = 0 \quad \text{on} \quad \partial \Omega \times (0, T),
\]
where the total velocity $\mathbf{v}(x, t) : \mathbb{R}^d \times \mathbb{R} \rightarrow \mathbb{R}^d$ is defined to be the sum of velocities of both phases,
\[
\mathbf{v}(x, t) := v_w(x, t) + v_o(x, t).
\]
To enforce a unique solution in pressure, fix the average pressure throughout the volume:

\[ \int_{\Omega} p_w(x, \cdot) dx = 0 \quad \text{on} \quad (0, T). \]

Fix the initial condition of the system, \( s_w(x, 0) = s_0(x) \), is within the feasible saturation constraints,

\[ 0 \leq s_0(x) \leq 1 \]

Defining the total mobility,

\[ \lambda(s_w) := \lambda_w(s_w) + \lambda_o(s_w), \]

the conservation of mass equations (2.6.5) combine to form a mixed formulation. Then, by combining, the conservation of mass equations (2.6.5) arrives at a mixed formulation. This forms a system of equations written in terms of saturation, velocity and pressure that must be solved simultaneously.

### 2.6.6 Formulation without Capillary Pressure

This forms a system of equations written in terms of saturation and pressure that must be solved simultaneously,

\[ \phi(x) \frac{\partial}{\partial t} s_w(x, t) + \nabla \cdot v_w(x, t) = q_w(x, t), \quad x \in \Omega, \ t > 0, \quad (2.6.7a) \]

\[ v_w(x, t) + K(x) \lambda_w(s_w(x, t)) \nabla p(x, t) = 0, \quad x \in \Omega, \ t > 0, \quad (2.6.7b) \]

\[ v(x, t) + K(x) \lambda(s_w(x, t)) \nabla p(x, t) = 0, \quad x \in \Omega, \ t > 0, \quad (2.6.7c) \]

\[ \nabla \cdot v = q(x, t), \quad x \in \Omega, \ t > 0, \quad (2.6.7d) \]

\[ v(x, t) \cdot n = 0, \quad x \in \partial \Omega, \ t > 0, \quad (2.6.7e) \]

\[ s_w(x, 0) = s_w_{\text{init}}(x), \quad x \in \Omega, \quad (2.6.7f) \]

\[ \int_{\Omega} p(x, t) dx = 0, \quad t > 0. \quad (2.6.7g) \]

Before discussing the discretization of the reservoir equations, I will consider an equivalent formulation that is popular in the numerical simulation of reservoir equations.
Total Velocity Formulation  An alternative formulation of (2.6.7) called the total velocity formulation uses an equivalent formulation for the flux term in the saturation equation,

\[ v_w(x, t) = v(x, t)f(s_w(x, t)) \]

resulting in the modified total velocity formulation of the saturation equation,

\[ \phi(x) \frac{\partial}{\partial t} s_w(x, t) + \nabla \cdot [v(x, t)f(s_w(x, t))] = q_w(x, t), \quad x \in \Omega, \ t > 0. \quad (2.6.8) \]

While the two formulations are equivalent at the differential equation level, depending on the discretization used, they can yield different results in simulations. When the velocity term is fixed, despite its dependence on saturation, the saturation equation takes the form of a multi-dimensional Buckley-Leverett equation. Since the velocity term \( v(x, t) \) has total mobility \( \lambda(s_w(x, t)) \) as a factor and the fractional flow \( f(s_w(x, t)) \) has total mobility in the denominator, at PDE level, the total mobility terms cancel and \( v_w(x, t) = v(x, t)f(s_w(x, t)) \) by definition. However, for some spatial-temporal discretizations, the two formulations are no longer equivalent.

Spillette et al. (1973) introduce the total velocity formulation for a sequential solver with nonzero capillary pressure [49]. Their approach is an operator splitting method, solving the saturation and pressure equations separately update saturation and pressure variables. The authors note that without the total velocity formulation, sequential solvers have ”given rise to anomalous results such as changing water saturation in regions of the reservoir where water relative permeability is zero”. They attribute the superiority of the total velocity formulation to the fact that total velocity does not depend on the capillary pressure, whereas, individual phase velocities do and demonstrate superior numerical stability and accuracy than the aqueous (water) velocity formulation.

However, for fully implicit schemes, the gains from the total velocity formulation are not realized. In fully implicit schemes, the velocity term depends implicitly on the new saturation update instead of the time-lagged saturation in the sequential
framework. The ratio of the two total mobilities cancel, save differences in their calculation across interfaces (i.e. in our simulator, we use arithmetic mean for the total mobilities in the pressure equation and upwind total mobility in the saturation equation). Hence, I use the aqueous (water) phase velocity $v_w(x,t)$ for the flux term in the saturation equation.

### 2.6.7 Formulation with Capillary Pressure

The difference in pressure of the two phases is defined as the capillary pressure, $p_c := p_o - p_w$. Capillary pressure is proportional to the interfacial tension between the two fluids (analogous to friction of the two phases). Like relative permeability, capillary pressure is often determined experimentally and defined as a function of saturation of water. Expect the capillary pressure to be monotonically increasing in saturation of water and become asymptotically unbounded at residual oil and water saturations. See Figure 2.3.

![Figure 2.3: Representative shape of capillary pressure curves – monotonically increasing functions with asymptotes at water and oil retentions, $s_{wr}$ and $s_{or}$.](image)

The resulting PDE system for water saturation $s_w$, total velocity $v$, and global
pressure $p$

$$p(x, t) := p_o(x, t) - \int_{1}^{s_w(x,t)} f(\xi) \frac{dp_c}{ds_w}(\xi) d\xi.$$

The extension of (2.6.7) to include capillary pressure is given as follows,

$$\phi(x) \frac{\partial}{\partial t} s_w(x, t) + \nabla \cdot \left[ f(s_w(x, t)) \left[ v(x, t) + d(s_w(x, t)) \right] \right] = q_w(x, t), \quad x \in \Omega, \ t > 0,$$

(2.6.9a)

$$v(x, t) + K(x) \lambda(s_w(x, t)) \nabla p(x, t) = 0, \quad x \in \Omega, \ t > 0,$$

(2.6.9b)

$$\nabla \cdot v(x, t) = q(x, t), \quad x \in \Omega, \ t > 0,$$

(2.6.9c)

$$v(x, t) \cdot \vec{n} = 0, \quad x \in \partial \Omega, \ t > 0,$$

(2.6.9d)

$$s_w(x, 0) = s_{w,\text{init}}(x), \quad x \in \Omega.$$

(2.6.9e)

where

$$d(s_w(x, t)) := K(x) \lambda_o(s_w(x, t)) \frac{d}{ds} p_c(s_w(x, t)) \nabla s_w(x, t)$$

and $q(x, t) := q_w(x, t) + q_o(x, t)$ is the total injection / production. Because of the no-flow boundary condition, the sources must satisfy $\int_{\Omega} q(x, t) dx = 0$ for all $t$.

We are now ready to discretize the system of partial differential equations (2.6.7). I will begin with an introduction to the Finite Volume method for the linear parabolic case and quickly move to discretization of the reservoir equations.

## 2.7 Finite Volume Discretization

The Finite Volume Method (FVM) is a popular method used in the computational fluid dynamics community due to conservation properties of the discretization. FVM
is a preferred method in the computational fluid dynamics community as the flux leaving one volume matches the flux entering its neighbor, a feature of the discretization called "locally conservative". Note that FVM is not the only locally conservative discretization – Discontinuous Galerkin and a variations of the Finite Element Method (i.e. Mixed Finite Elements) are also locally conservative.

2.7.1 Linear Parabolic in 2D

We begin by considering a the 2D diffusion-advection-reaction parabolic PDE as a model problem,

$$y_t(x, t) + \nabla \cdot [\epsilon(x, t)\nabla y(x, t)] + \nabla y(x, t) \cdot \nu(x, t) + r(x, t)y(x, t) = u(x, t),$$  \hspace{1cm} (2.7.1a)

where \((x, t) \in \Omega \times [0, T]\) and no-flow (Neumann) boundary conditions are imposed,

$$\nabla y(x, t) \cdot n = 0, \quad (x, t) \in \partial \Omega \times [0, T],$$  \hspace{1cm} (2.7.1b)

and with initial condition,

$$y(x, 0) = y_0(x), \quad x \in \Omega \hspace{1cm} (2.7.1c)$$

2.7.2 Finite Volume Mesh

We assume a rectangular domain \(\Omega \subset \mathbb{R}^2\). Further, we construct a uniform rectangular mesh. Let \(\mathcal{T} := \{e_i\}_{i=1}^{n_v}\) denote a family of rectangular volumes of equal dimension and volume where \(n_v\) is number of volumes. Let \(\mathcal{E} := \{\sigma_{ij} : e_i \text{ and } e_j \text{ are adjacent volumes}\}\) denote a family of edges, also known as interfaces, and \(\{x_i\}_{i=1}^{n_v}\) the volume centers with \(x_i \in e_i\). Following Drohmann et al. (2012) we assume the following mesh properties are satisfied [21]:

1. \(\bigcup_{i=1}^{n_v} \bar{e}_i = \bar{\Omega}\).

2. For any \(e, f \in \mathcal{T}\) with \(e \neq f\), either \(m(\bar{e} \cap \bar{f}) = 0\) or \(\bar{e} \cap \bar{f} = \bar{\sigma}\). The set of neighboring volumes is denoted \(\mathcal{N}(e) := \{f \in \mathcal{T} | m(\bar{e} \cap \bar{f}) \neq 0\}\).
3. For any cell \( e \in \mathcal{T} \) its boundary is given by a set of edges \( \mathcal{E}(e) \subset \mathcal{E} \), i.e., \( \partial e = \cap_{\sigma \in \mathcal{E}(e)} \bar{\sigma} \).

4. The connection of two points \( x_i, x_j \) is orthogonal to the edge \( \sigma_{ij} := \bar{\sigma}_i \cap \bar{\sigma}_j \) and its length is denoted by \( d_{ij} := |x_i - x_j| \).

We generate a rectangular mesh containing volumes \( e_i, \mathcal{T} := \{e_i\}_{i=1}^{n_e} \) satisfying

\[
\bigcup_{i=1}^{n_e} e_i = \bar{\Omega}, \quad e_i \cap e_j = \emptyset.
\]

![Figure 2.4: Mesh enumerated in row-major format.](image)

Denote centers \( x_i \in e_i \) so that the connection \( x_i \) to \( x_j \) is perpendicular to the common edge of two adjacent volumes,

\[
\sigma_{ij} := \bar{e}_i \cap \bar{e}_j.
\]

Next define two Hilbert spaces on \( \Omega \) and the skeleton, \( \mathcal{S} := \cap_{\sigma \in \mathcal{E}} \bar{\sigma} \) respectively,

\[
\mathcal{W}_h^T := \{ f : \Omega \rightarrow \mathbb{R} \text{ piecewise const on all } e_i \in \mathcal{T} \}
\]

\[
\mathcal{W}_h^\mathcal{E} := \{ f : \mathcal{S} \rightarrow \mathbb{R} \text{ piecewise const on all } \sigma_{ij} \in \mathcal{E} \}
\]

where \( \mathcal{W}_h^T \) is for pressure and saturation and \( \mathcal{W}_h^\mathcal{E} \) for fluid velocity across interfaces.

The spaces are endowed with the standard \( L^2(\Omega) \) and \( L^2(\mathcal{S}) \) inner products,

\[
\langle u_h, v_h \rangle_{\mathcal{W}_h^T} = \int_{\Omega} u_h v_h \quad \text{and} \quad \langle u_h, v_h \rangle_{\mathcal{W}_h^\mathcal{E}} = \int_{\mathcal{S}} u_h v_h.
\]
Since $W^T_h$ and $W^S_h$ are finite dimensional, they can be represented by vectors in $\mathbb{R}^{n_v}$ and $\mathbb{R}^{n_e}$ where $n_v$ and $n_e$ are the number of volumes and edges respectively:

\[
\begin{align*}
    u_h(x) &= \sum_{i=1}^{n_v} u_i \chi_{e_i}(x) \in W^T_h, \quad (u_i)_{i=1}^{n_v} \in \mathbb{R}^{n_v}, \\
v_h(x) &= \sum_{\sigma_{ij} \in E} v_{ij} \chi_{\sigma_{ij}}(x) \in W^E_h, \quad (v_{ij})_{\sigma_{ij} \in E} \in \mathbb{R}^{n_e}.
\end{align*}
\]

**Discretization** Fix $\bar{y}_i^k$ to denote the average of $y(x,t)$ over the volume $e_i$ at time $t = t_k$,

\[
\bar{y}_i^k \approx \frac{1}{m(e_i)} \int_{e_i} y(x,t_k) dx, \quad \bar{u}_i^k \approx \frac{1}{m(e_i)} \int_{e_i} u(x,t_k) dx.
\]

We approximate the diffusion, advection and reaction coefficients as piecewise constant functions on $e_i$. Note that when (2.7.1a) is a linearization of a nonlinear FV discretization, this approximation becomes exact.

\[
\begin{align*}
    \epsilon_i^k &:= \epsilon(x_i,t_k) \\
    \nu_i^k &:= \nu(x_i,t_k) \\
r_i^k &:= r(x_i,t_k)
\end{align*}
\]

If we integrate (2.7.1a) over $e_i$, via the Divergence Theorem at $t = t_{k+1}$,

\[
\begin{align*}
    m(e_i) \frac{\bar{y}_i^{k+1} - \bar{y}_i^k}{\Delta t} + \int_{\partial e_i} \epsilon(x,t_{k+1}) \nabla y(x,t_{k+1}) \cdot n \, dx &+ \int_{e_i} \nabla y(x,t_{k+1}) \cdot \nu_i^{k+1} \, dx + r_i^{k+1} \bar{y}_i^{k+1} \approx \bar{u}_i^{k+1}.
\end{align*}
\]

While $\nabla y(x,t)$ is defined on the interfaces $\partial e_i$, the finite volume solution is piecewise constant and therefore is not defined on the interfaces. Instead, we approximate the gradient computation with the finite difference of the averages between two adjacent volumes,

\[
\nabla y(x,t_k) \bigg|_{\partial e_i} \cdot n \approx \frac{\bar{y}_j^k - \bar{y}_i^k}{\|x_j - x_i\|}
\]
Likewise, we have assumed $\epsilon(x, t_k)$ to be constant on the volumes therefore we define the diffusion coefficient on the interface as the harmonic average of the coefficients on the adjacent volumes,

$$
\epsilon(x, t_k) \bigg|_{\partial e_i} \approx \{\epsilon(x, t_k)\}_{ij} := \frac{2\epsilon_i^k \epsilon_j^k}{\epsilon_i^k + \epsilon_j^k}.
$$

For the advection term, we need to approximate $\nabla y(x, t_k)$ on a volume $e_i$. To accomplish this, we use upwinding,

$$
\nabla y(x, t_k) \cdot \nu_i^k \approx \sum_{j \in N(e_i)} - \frac{\bar{y}_j^k - y_i^k}{\|x_j - x_i\|} \max\{0, -\nu_i^k \cdot n\}.
$$

Then the finite volume scheme approximates the volumes averages $\bar{y}_i^k$ with $y_i^k$ satisfying the dynamical system,

$$
m(e_i) \frac{y_i^{k+1} - y_i^k}{\Delta t} + \sum_{j \in N(e_i)} m(\sigma_{ij}) \{\epsilon(x, t_{k+1})\}_{ij} \frac{y_j^{k+1} - y_i^{k+1}}{\|x_j - x_i\|}
$$

$$
+ m(e_i) \sum_{j \in N(e_i)} \frac{y_j^{k+1} - y_i^{k+1}}{\|x_j - x_i\|} \max\{0, -\nu_i^{k+1} \cdot n\} + m(e_i)r_i^{k+1}y_i^{k+1} = m(e_i)u_i^{k+1}.
$$

The dynamical system can then be written in the form,

$$
M \frac{y^{k+1} - y^k}{\Delta t} + Ay^{k+1} = Bu^{k+1}.
$$

This framework lays the foundation for the reservoir discretization that follows.

### 2.7.3 Discretization of the Reservoir Equations

In this section, I summarize the finite volume scheme found in Drohmann et al. (2012) and Michel (2003) [21, 41]. However, the following changes have been made to their formulation in this section:

1. I assume vanishing capillary pressure, $p_c(s_w) \equiv 0$. This implies that $d(s_w) \equiv 0$ and pressure of the two phases are in equilibrium, $p = p_w = p_o$. 


2. I allow for heterogeneous media in both porosity \( \phi(x) \) and absolute permeability, \( K(x) \).

Integrating the first two conservation laws in (2.6.7) on each element individually yields

\[
\int_{e_i} \phi(x) \frac{\partial s_w(x,t)}{\partial t} + \int_{\partial e_i} v_w(x,t) \cdot n = \int_{e_i} q_w(s_w(x,t)) \\
\int_{\partial e_i} v(x,t) \cdot n = \int_{e_i} q_w(s_w(x,t)) + q_o(s_w(x,t))
\]

Assuming uniform porosity and permeability on each volume, we can state the semi-discretized formulation as follows. The finite volume method consists of three approximations to the equations (2.6.7),

1. approximate the saturation and pressure with piecewise constant functions,

\[
s_w(x,t) \approx s_h(x,t) = \sum_{i=1}^{n_v} s_i(t) \chi_{e_i}(x), \quad p(x,t) \approx p_h(x,t) = \sum_{i=1}^{n_v} p_i(t) \chi_{e_i}(x),
\]

2. approximate the velocity with piecewise constant functions on the mesh skeleton (union of interfaces),

\[
v_w(x,t)|_{x \in S} \approx v_{w,h}(x,t) = \sum_{i=1}^{n_v} v_{w,ij}(t) \chi_{\sigma_{ij}}(x), \quad v(x,t)|_{x \in S} \approx v_{h,h}(x,t) = \sum_{i=1}^{n_v} v_{ij}(t) \chi_{\sigma_{ij}}(x)
\]

3. approximate the flux terms \( v_w(x,t) \cdot n \) and \( v(x,t) \cdot n \) on the volume boundaries \( \sigma_{ij} \) for neighboring elements \( e_i \) and \( e_j \) via the finite difference approximation,

\[
v(x,t) \cdot n|_{\sigma_{ij}} = K(x) \lambda(s_w(x,t)) \nabla p(x,t) \cdot n|_{\sigma_{ij}} \\
\approx v_{ij} := \{K(x)\}_{ij} \frac{\lambda(s_i(t)) + \lambda(s_j(t))}{2} \frac{p_j(t) - p_i(t)}{d_{ij}},
\]

\[
v_w(x,t) \cdot n|_{\sigma_{ij}} = \{K(x)\}_{ij} \lambda_w(s_w(x,t)) \nabla p(x,t) \cdot n|_{\sigma_{ij}} \\
\approx v_{w,ij} = \{K(x)\}_{ij} \lambda_{w,ij}(s(t)) \frac{p_j(t) - p_i(t)}{d_{ij}},
\]
where \( \{K(x)\}_{ij} \) denotes the harmonic average of \( K(x) \) on the interface \( \sigma_{ij} \) so that
\[
\{K(x)\}_{ij} = \left[ \{\kappa_x(x)\}_{ij} \right], \quad \{\kappa_x(x)\}_{ij} = \frac{2}{\frac{1}{\kappa_{x_i}(x_i)} + \frac{1}{\kappa_{x_j}(x_j)}},
\]
and \( \hat{\lambda}_{w,ij}(s) \) denotes the upwinded fractional flow on edge \( \sigma_{ij} \),
\[
\hat{\lambda}_{w,ij}(s) = \begin{cases} \lambda_w(s_i(t)), & v_{ij} \cdot n \geq 0, \\ \lambda_w(s_j(t)), & v_{ij} \cdot n < 0. \end{cases}
\]

**Semi-Discretized Model Problem Formulation**

For modeling incompressible flows, the net flow rates for the reservoir must be zero. That is, given vector of well rates \( q(t) \) satisfying the condition that \( \sum_{i=1}^{n_w} q_i(t) = 0 \) for all \( t > 0 \), find saturation and pressure
\[
s_h(x,t) = \sum_{i=1}^{n_w} s_i(t) \chi_{e_i}(x), \quad p_h(x,t) = \sum_{i=1}^{n_w} p_i(t) \chi_{e_i}(x)
\]
that satisfy the following mixed formulation on every volume \( e_i \),
\[
m(e_i) \frac{\partial s_i(t)}{\partial t} + \sum_{j \in N(e_i)} m(\sigma_{ij}) v_{w,ij} = q_w(s_i(t)),
\]
\[
\sum_{j \in N(e_i)} m(\sigma_{ij}) v_{ij} = q_i,
\]
\[
v_{w,ij} := \{K(x)\}_{ij} \hat{\lambda}_{w,ij}(s(t)) \left( \frac{p_j(t) - p_i(t)}{d_{ij}} \right),
\]
\[
v_{ij} := \{K(x)\}_{ij} \left( \frac{\lambda(s_i(t)) + \lambda(s_j(t))}{2} \right) \left( \frac{p_j(t) - p_i(t)}{d_{ij}} \right),
\]
where \( \phi_i = \frac{1}{m(e_i)} \int_{e_i} \phi(x) dx \) denotes the average porosity over the volume \( e_i \).

The above semi-discretized formulation can be written in a consolidated form for \( s(t), p(t), q(t) \in \mathbb{R}^{n_w} \) and vector-valued functions, \( G : \mathbb{R}^{n_w} \times \mathbb{R}^{n_w} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_w} \), \( H : \mathbb{R}^{n_w} \times \mathbb{R}^{n_w} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_w} \),
\[
Ms'(t) + G(s(t), p(t), q(t)) = 0, \quad (2.7.2a)
\]
\[
H(s(t), p(t), q(t)) = 0. \quad (2.7.2b)
\]
where \( H, \ G \) and \( M \) are defined,

\[
[H(s(t), p(t), q(t))]_i := \sum_{j \in N(e_i)} \left[ m(\sigma_{ij})v_{ij} \right] - q_i,
\]

\[
[G(s(t), p(t), q(t))]_i := \sum_{j \in N(e_i)} \left[ m(\sigma_{ij})v_{w,ij} \right] - q_w(s_i(t)),
\]

\[
M := \begin{bmatrix} m(e_1) & \cdots & m(e_{n_v}) \end{bmatrix}.
\]

One problem with this formulation is that the pressure solution is not unique. The discretized saturation and pressure equations are translation invariant in the pressure argument as they require the gradient of pressure only. Therefore, if \( p_h(t) \) is a solution to the pressure equation, then so is are constant translations of the solution, \( p_h(t) + \alpha \) for arbitrary \( \alpha \in \mathbb{R} \). To force uniqueness of the solution, some authors impose an additional constraint,

\[
\sum_{i=1}^{n_y} p_i(t) = 0.
\]

However, implementation of this approach in the reservoir simulator is nontrivial without a least squares formulation. In a practical approach, the discretization of the pressure equation \( H \) can be redefined with an additional regularization term,

\[
[H(s(t), p(t), q(t))]_i := \alpha e_1 e_1^T + \sum_{j \in N(e_i)} \left[ m(\sigma_{ij})v_{ij} \right] - q_i.
\]

This approach, while mathematically inexact, has a unique solution and is trivial to implement. A more detailed explanation of this modification can be found in Aarnes et al. (2007) [1]. For the reservoir simulations in this document I choose the coefficient \( \alpha \) to match Wiegand (2010) [56], scaling \( \alpha \) with permeability,

\[
\alpha := \kappa_{x_1}(x) + \kappa_{x_2}(x),
\]

for some arbitrary \( x \in \Omega \).
Temporal Discretization  A fully implicit temporal discretization on a uniform time stepping scheme, \( t_k = k\Delta t = \frac{kT}{n_t} \), \( k = 0, \ldots, n_t \), is used where saturation, pressure and well rates are approximated,

\[
\begin{align*}
    s_k & \approx s(t_k), \\
    p_k & \approx p(t_k), \\
    q_k & \approx q(t_k).
\end{align*}
\]

Then given well rates \( \{q_k\}^{n_t}_{k=1} \), the saturation and pressure are computed simultaneously,

\[
\begin{align*}
    M(s_{k+1} - s_k) + \Delta t G(s_{k+1}, p_{k+1}, q_{k+1}) &= 0, \\
    H(s_{k+1}, p_{k+1}, q_{k+1}) &= 0.
\end{align*}
\]

Note here that as \( G \) is nonlinear in the first and second arguments (saturation and pressure), and likewise, \( H \) is nonlinear in the first argument (saturation), we need to solve each update via Newton’s Method, detailed in Algorithm 7.

Scaling of Newton Solver  Two issues arise in the numerical solution of the reservoir equations above via Algorithm 7:

1. The diagonal mass matrix \( M \) is potentially ill-conditioned as the porosity field can vary by more than six orders of magnitude in practice. This can create scaling issues in Newton’s Method’s residual computation for the saturation equation, \( r_s(s_{k+1}, p_{k+1}, q_{k+1}) \). The results in the following section alleviate this concern by modifying the saturation equation by pre-multiplying by \( M^{-1} \), making the mass matrix for the saturation equation effectively the identity. The inverse \( M^{-1} \) is trivial to compute as \( M \) is diagonal and positive definite.

2. Pressure and saturation are not scaled well as the pressure field is many orders of magnitude larger than saturation. I implement a rescaling of the pressure by six orders of magnitude, dramatically improving the convergence rate and behavior of Newton’s method.

Integrating these numerical considerations into the Newton solver, we can summarize the Newton’s Method for the reservoir equations in Algorithm 7.
Algorithm 2.7: Newton’s Method for the Fully Implicit Reservoir Equations

1: Given the initial saturation \( s_0 \), solve for the initial pressure \( p_0 \) via the linear pressure equation: \( H(s_0, p_0, q_0) = 0 \).

2: for \( k = 0, \ldots, n_t - 1 \) do

3: Warm start Newton’s Method with previous time step:

\[
\begin{bmatrix}
  s_{k+1} \\
  p_{k+1}
\end{bmatrix}
\leftarrow
\begin{bmatrix}
  s_k \\
  p_k
\end{bmatrix}
\]

4: Compute the residual:

\[
\begin{bmatrix}
  r_s(s_{k+1}, p_{k+1}, q_{k+1}) \\
  r_p(s_{k+1}, p_{k+1}, q_{k+1})
\end{bmatrix}
\leftarrow
\begin{bmatrix}
  (s_{k+1} - s_k) + \Delta t M^{-1} G(s_{k+1}, p_{k+1}, q_{k+1}) \\
  H(s_{k+1}, p_{k+1}, q_{k+1})
\end{bmatrix}
\]

5: while \( \| r_s(s_{k+1}, p_{k+1}, q_{k+1}) \| > \text{tol} \) or \( \| r_p(s_{k+1}, p_{k+1}, q_{k+1}) \| > \text{tol} \) do

6: Solve for Newton updates \( \delta s_{k+1} \) and \( \delta p_{k+1} \):

\[
\begin{bmatrix}
  I + \Delta t M^{-1} G_s(s_{k+1}, p_{k+1}, q_{k+1}) & I + \Delta t M^{-1} G_p(s_{k+1}, p_{k+1}, q_{k+1}) \\
  H_s(s_{k+1}, p_{k+1}, q_{k+1}) & H_p(s_{k+1}, p_{k+1}, q_{k+1})
\end{bmatrix}
\begin{bmatrix}
  \delta s_{k+1} \\
  \delta p_{k+1}
\end{bmatrix}
= -
\begin{bmatrix}
  r_s(s_{k+1}, p_{k+1}, q_{k+1}) \\
  r_p(s_{k+1}, p_{k+1}, q_{k+1})
\end{bmatrix}
\]

7: Update saturation and pressure with \( \alpha \) chosen via line search:

\[
\begin{bmatrix}
  s_{k+1} \\
  p_{k+1}
\end{bmatrix}
\leftarrow
\begin{bmatrix}
  s_{k+1} \\
  p_{k+1}
\end{bmatrix}
+ \alpha
\begin{bmatrix}
  \delta s_{k+1} \\
  \delta p_{k+1}
\end{bmatrix}
\]

8: Update the residual:

\[
\begin{bmatrix}
  r_s(s_{k+1}, p_{k+1}, q_{k+1}) \\
  r_p(s_{k+1}, p_{k+1}, q_{k+1})
\end{bmatrix}
\leftarrow
\begin{bmatrix}
  (s_{k+1} - s_k) + \Delta t M^{-1} G(s_{k+1}, p_{k+1}, q_{k+1}) \\
  H(s_{k+1}, p_{k+1}, q_{k+1})
\end{bmatrix}
\]

9: end while

10: end for
2.8 Example Problem - SPE10-2

The SPE10-2 model is a benchmark in reservoir simulation literature. The dimensions of the model are 1200 (ft) x 2200 (ft) x 170 (ft), discretized with uniform hexahedral volumes 20 (ft) x 10 (ft) x 2 (ft) each.

I have extracted porosity and permeability from the top layer of the data set. Both terms are visualized in Figure 2.1. To make the simulation more tractable, the input data has been chosen on the following way:

1. porosity as been floored to not be less than $10^{-3}$,

2. injected saturation is pure water, $s_{in} = 1.0$,

3. viscosity of water and oil are 0.3 (cP) and 3 (cP) respectively,

4. simulation is run for 4000 days with time steps of 200 days.
Figure 2.5: Reservoir simulation pressure and saturation at $t = 2000$ and $t = 4000$ days. Single injector and four producers are denoted as a circle and as squares respectively.
Chapter 3

Motivation: Reservoir Simulation

This chapter studies the application of model reduction for reservoir simulation and reservoir management problems in the context of two-phase immiscible incompressible flows. I study this basic task: generate a reduced order model from a large scale finite volume discretization and then approximate this simulation using the reduced order model.

While a discretization scheme may perform well for the full order model simulation, it can become unstable when applied to the reduced order model. In Section 3.3 I study the issue of sampling. Like all projection based reduced order models, reduced order models generate the ‘important dynamics’ by sampling, i.e., by simulating the full order model at select inputs, and extract a low dimensional subspace from these samples.

I will show that sampling can impact stability of the reduced order model. If a reduced order model is applied at a set of well rates for which the corresponding saturation and pressure dynamics are not represented in the reduced basis, then the reduced order model can become unstable, while otherwise it produces an excellent approximation of the full model.

Optimization algorithms explore a wide range of inputs (such as a wide range of
well rate settings). Based on observations it is not clear that reduced order models can be constructed that reliably replace the full order model at a cost that provides substantial computational savings. Therefore, I propose to use reduced order model not to replace reservoir simulations, but to use them to reduce computational tasks within the optimization: Hessian computations. This is similar to the use of preconditioners in iterative linear system solvers: in my approach reduced order models accelerate the optimization, but will not impact the accuracy of the simulation or optimization.

3.1 Projection-Based Model Order Reduction for Reservoir Simulation

In this section I will introduce an abbreviated projection-based model order reduction scheme for reservoir engineering. My goal in this section is to briefly demonstrate some of the challenges of model order reduction in reservoir simulation.

I restate the semi-discretized reservoir PDEs (2.7.2) from subsection 2.7.3 for convenience. Given well rates \( \{ q_k \}_{k=1}^{n_t} \), the saturation \( \{ s_k \} \subset \mathbb{R}^{n_y} \) and pressure \( \{ p_k \} \subset \mathbb{R}^{n_y} \) are computed simultaneously,

\[
M(s_{k+1} - s_k) + \Delta t \ G(s_{k+1}, p_{k+1}, q_{k+1}) = 0, \quad \text{(3.1.1)}
\]

\[
H(s_{k+1}, p_{k+1}, q_{k+1}) = 0. \quad \text{(3.1.2)}
\]

Projection-based reduced order models compute low dimensional subspaces for the saturations and pressures and project the equations on these subspaces. More specifically, they compute orthogonal matrices \( U_s \in \mathbb{R}^{n_y \times r_s}, \ U_p \in \mathbb{R}^{n_y \times r_p} \) with \( r_s, r_p \ll n_y \), such that the columns of these matrices are bases for the saturation subspace and pressure subspace, respectively. The approach seeks saturation and pressure approximations of the form

\[
s_k \approx U_s \hat{s}_k, \quad p_k \approx U_p \hat{p}_k,
\]
which are obtained by inserting these approximations into (3.1.1) and forcing the resulting error to be perpendicular to the same subspaces. Thus, the reduced order model equations are

\begin{align}
U_s^T MU_s(\hat{s}_{k+1} - \hat{s}_k) + \Delta t \ U_s^T G(U_s \hat{s}_{k+1}, U_p \hat{p}_{k+1}, q_{k+1}) &= 0, \quad (3.1.3a) \\
U_p^T H(U_s \hat{s}_{k+1}, U_p \hat{p}_{k+1}, q_{k+1}) &= 0. \quad (3.1.3b)
\end{align}

Since \( \hat{s}_k \in \mathbb{R}^{r_s} \) and \( \hat{p}_k \in \mathbb{R}^{r_p} \) with \( r_s, r_p \ll n_v \), the degrees of freedom in system (3.1.3) are substantially fewer than that of system (3.1.1). Note that since \( H \) or \( G \) is are nonlinear, the cost of the evaluation

\begin{align}
(\hat{s}_{k+1}, \hat{p}_{k+1}) \mapsto U_p^T H(U_s \hat{s}_{k+1}, U_p \hat{p}_{k+1}, q_{k+1}), \\
(\hat{s}_{k+1}, \hat{p}_{k+1}) \mapsto U_s^T G(U_s \hat{s}_{k+1}, U_p \hat{p}_{k+1}, q_{k+1}),
\end{align}

still depends on the size \( n_y \) of the full order model. Thus, to arrive at a reduced order system in which the cost of evaluation of all terms depends only on reduced dimensions \( r_s, r_p \ll n_y \), but not on the original dimension \( n_y \), further approximations such as the Discrete Empirical Interpolation Method (DEIM) (see, e.g., [12, 20]) need to be applied. The main observations I want to make already arise when (3.1.3) is used; therefore, I omit a further approximation by DEIM for now.

The full and reduced order systems (3.1.1) and (3.1.3) are solved using a fully implicit update to both saturation and pressure variables, outlined in Algorithm 7.

### 3.2 Results

I use a 2D reservoir model obtain from the top layer of the SPE 10 data set (http://www.spe.org/web/csp/datasets/set02.htm). See Figure 3.1 to observe heterogeneities in permeability and porosity of the top layer.
Figure 3.1: Top layer of the SPE10 model with significant heterogeneities in porosity and permeability.

The 2D reservoir is of size $1200 \times 2200$ feet, and discretized into $60 \times 220 = 13200$ finite volume cells. The discretization follows the discussion in subsection 2.7.3. Thus discrete saturations and pressures in the finite volume discretization are of size $N = 13200$. Simulation time span is 6000 days, split into $n_t = 60$ subintervals of 100 days each.

The reduced order model is computed as follows:

1. Run the simulator to generate snapshots $s_0, \ldots, s_{n_t}$ and $p_0, \ldots, p_{n_t}$ for saturations and pressures.

2. Given a tolerance $\text{tol}_{\text{POD}} > 0$ apply Algorithm 2.4 to $s_0, \ldots, s_{n_t}$ and to $p_0, \ldots, p_{n_t}$ to compute ROM matrices $U_s \in \mathbb{R}^{n_y \times r_s}$ and $U_p \in \mathbb{R}^{n_y \times r_p}$.

Figure 3.2 shows the singular values for the saturation snapshots $s_0, \ldots, s_{n_t}$ and the pressure snapshots $p_0, \ldots, p_{n_t}$ (recall that in this example $n_t = 60$) together
with dashed line representing the POD tolerance where $\text{tol}_{\text{POD}} = 10^{-4}$. This results in reduced saturations and reduced pressures of size $r_s = 29$ and $r_p = 15$, respectively. The next plot shows the first 5 columns of the POD bases $U_s \in \mathbb{R}^{13200 \times 29}$ and $U_p \in \mathbb{R}^{13200 \times 15}$.

Figure 3.2: Singular values of the saturation and pressure snapshots generated using Algorithm 7. POD tolerance is set to $\text{tol}_{\text{POD}} = 10^{-4}$ resulting in reduced order models of sizes $r_s = 29$ and $r_p = 15$ for saturation and pressure.
Figure 3.3: (Top) First 5 POD modes (i.e. leading columns of \( U_s \)) for the saturation trajectory \( \{s_k\} \). (Bottom) First 5 POD modes (i.e. leading columns of \( U_p \)) for the saturation trajectory \( \{p_k\} \). Single injector and four producers are denoted as circular and square marker respectively.

Now I use the POD reduced order model to approximate the full order simulations. Observe that the reduced order model simulation using is in excellent agreement with the full order model simulation solving (3.1.3) in Figure 3.4.
Figure 3.4: Saturation for Full vs Reduced Order Simulations. (Top) Saturation at $t = 3000$ days. (Bottom) Saturation at $t = 6000$ days. The ROM simulation approximates the FOM simulation with high fidelity when simulations are run for the same well schedule as the POD training.

Table 3.1 compares the relative error between saturations generated by the full order model and the reduced order model for both sequential schemes over a range of POD tolerances $\text{tol}_{\text{POD}}$. The following shows that the error between reduced order
model simulation and full order model simulation decreases as the POD tolerance \( \text{tol}_{\text{POD}} \) decreases, which is what one expects to see. Since the spatial and temporal discretization are uniform and finite volume saturations \( s_w \) are piecewise constant, then

\[
\|s_w\|_{L^\infty(0,T;L^\infty(\Omega))} = \max_k \|s_k\|_\infty.
\]

Therefore, the error in saturation can be computed as

\[
\text{Sat._Error} = \|s_w - \tilde{s}_w\|_{L^\infty(L^2(\Omega);[0,T])} = \max_k \|s_k - U_k \tilde{s}_k\|_\infty.
\]

<table>
<thead>
<tr>
<th>tol_{\text{POD}}</th>
<th>r_s</th>
<th>r_p</th>
<th>Sat._Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1e-3</td>
<td>16</td>
<td>8</td>
<td>1.54e-2</td>
</tr>
<tr>
<td>1e-4</td>
<td>29</td>
<td>15</td>
<td>2.32e-3</td>
</tr>
<tr>
<td>1e-5</td>
<td>42</td>
<td>26</td>
<td>3.99e-4</td>
</tr>
</tbody>
</table>

Table 3.1: Relative errors in saturation for various POD tolerances. Errors in saturation are decreasing as the degrees of freedom in the ROMs grow.

### 3.3 Snapshot Generation for Reservoir Simulation

#### Reduced Order Models

The results presented in the previous section demonstrate that ROMs can generate high fidelity reservoir simulations when the full- and reduced-order systems are run on identical scenarios (model parameters, injection rates, well placement, etc.). However, a more interesting question is: how effective are ROMs for scenarios that differ from their “training”? This section investigates what happens if a reduced order model is applied to different well rates.

I will demonstrate why the training set matters with an example. I train the system on a constant well schedule, labeled Schedule #1 in Figure 3.5. In other
words, the POD basis is constructed from the dominant left singular vectors of the snapshot matrices populated with the full order snapshots from Schedule #1. I did this on the top layer of the SPE10 model found earlier in this chapter (see Figure 3.1).

Next, I will perturb the well rate schedule mildly by decreasing the flow for producers when breakthrough occurs. In this simulation breakthrough occurs for production wells #1 and #2 around $t = 1000$ days; therefore, I decrease their flow rates by 10% and increase well #3 and #4 by 10%. This process is repeated later when production well #3 reaches breakthrough at $t = 3000$ days; wells #1, #2, and #3 are decreased by 10% and well #4 is increased by 30%. Both well schedules can be seen in Figure 3.5.

![Well Schedule #1](image1.png)

![Well Schedule #2](image2.png)

Figure 3.5: Well Schedules. Schedule #1 used in training the ROM. Schedule #2 used to generate ROM results in Figure 3.6
Note that in practice, the engineer may want to completely shut a well off in this case. However, for this example 10% perturbations in the well rates was sufficient to demonstrate how sensitive ROM simulations are to their training.

Figure 3.6: Saturation for Full vs Reduced Order Simulations. FOM and ROM simulations run for same well schedule as the POD training. (Top) Saturation at $t = 3000$ days. (Bottom) Saturation at $t = 6000$ days.
3.4 Project Motivation

In existing approaches reduced order models are use to replace the full order models in simulation and in optimization. This only seems possible when the relevant reservoir dynamics are captured at all inputs quantities of interest (i.e. well rates controls in my examples). This will require extensive snapshot generations, i.e., expensive number of simulations at different well rates. The expense associated with this task seems to outweigh possible gains from using the reduced order model.

Instead, I propose to use reduced order models not to replace reservoir simulations, but to reduce computational tasks within the optimization, such as Hessian computations. As mentioned earlier, this is similar to the use of preconditioners in iterative linear system solvers: in my approach reduced order models accelerate the optimization, but will not impact the accuracy of the simulation or optimization.

For example, reduced order models can be used within implicit time stepping methods as preconditioners (or predictors). Furthermore, they can be used in the linearized state equation and 2nd order adjoint equation solves that arise in Hessian-times-vector computations. This will lead to inexact gradients and Hessian; the impact of these sources of inexactness can be analyzed in the context of optimization algorithms. Moreover, approximate derivative information, especially second order derivative information can be computed much faster, thus accelerating Newton’s Method.
Chapter 4

Model Reduction for
Hessian-Vector Computation

In this chapter, I introduce the core concept of this dissertation: approximating Hessian-Vector computation via subspace-based model reduction. The goal is to implement model reduction to accelerate Hessian-vector computations by replacing the two PDE solves in the Hessian-vector multiplication with their surrogates constructed via subspace-based model reduction as the Hessian-vector multiplications are the dominant cost in the Newton-CG algorithm introduced in Algorithm 2.1. To begin, first consider the "traditional approach" used in optimization and model reduction.

Traditional Approach  Recall the optimization program,

$$\min_u \mathcal{J}(y(u), u)$$  \hspace{1cm} (4.0.1)

where $y(u)$ satisfies the implicit constraint, $c(y(u), u) = 0$, in a setting where the mapping from control to state $u \mapsto y(u)$ is expensive to evaluate. Conventional approaches to model reduction in optimization seek to approximate this mapping with a surrogate of lesser cost, $u \mapsto \hat{y}(u)$, so that $V\hat{y}(u) \approx y(u)$. This approach
leads to the surrogate problem,

$$\min_u J(V\hat{y}^u, u)$$  \hspace{1cm} (4.0.2)$$

where $\hat{y}(u)$ solves $V^T c(V\hat{y}^u, u) = 0$ and where $\hat{y}(u)$ is represented with many fewer degrees of freedom than the expensive solution $y(u)$.

Typically, the ROM is not valid for all $u$ and therefore (4.0.2) does not replace the original problem (4.0.2) globally, but only in a neighborhood of a current iterate $u_c$. An approximate minimization of $J(V\hat{y}^u, u)$ in that neighborhood gives the new iterate $u_+$. The reduced order model $V$ is then updated if needed, and the process is repeated. For an introduction to these approaches and their strategies for ROM updates, see Sachs and Volkwein (2010) [46]. For these approaches to be effective, the neighborhood around the current iterate $u_c$ on which the ROM surrogate function $J(V\hat{y}^u, u)$ approximates the original function $J(y(u), u)$ needs to be sufficiently large. Otherwise relative expensive ROM calculations have to be done too often.

However, for the reservoir management problem, I have demonstrated in the previous chapter that ROMs are unreliable with modest perturbations away from their training. In this framework, the subspace $V$ would need to be recomputed at nearly every iteration of the optimization at the cost of full order solves of the state and adjoint, $y(u)$ and $\lambda(u)$. In doing so, the true gradient computation $\nabla J(u)$ becomes negligible in cost, and as such, we may as well use the exact gradient, state and adjoint information instead of their approximations. This realization inspires the approach I propose in this chapter.

### 4.1 Model Problem and Derivative Computation

I consider the following model problem. Given functions $\ell : \mathbb{R}^{n_y} \to \mathbb{R}$, $\sigma : \mathbb{R}^{n_u} \to \mathbb{R}$, $F : \mathbb{R}^{n_y} \to \mathbb{R}^{n_y}$, and $G : \mathbb{R}^{n_u} \to \mathbb{R}^{n_y}$, and given $y_0 \in \mathbb{R}^{n_y}$ consider the minimization
problem in variables $\mathbf{u} = (u^T_1, \ldots, u^T_{nt})^T \in \mathbb{R}^{nu}, \mathbf{y} = (y^T_1, \ldots, y^T_{nt})^T \in \mathbb{R}^{ny}$.

$$\text{Minimize } \sum_{k=1}^{nt} \ell(y_k) + \sigma(u_k) \quad (4.1.1a)$$

where $\mathbf{y}$ and $\mathbf{u}$ satisfy an implicit constraint,

$$M \left( \frac{y_{k+1} - y_k}{\Delta t} \right) + F(y_{k+1}) = G(u_{k+1}), \quad y_0 \text{ given.} \quad (4.1.1b)$$

Note here that the mixed derivatives of the objective function and implicit constraint vanish. This assumption accommodates the semilinear parabolic analysis in Chapter 5 and the semilinear least squares optimal control problem in Chapter 6). However, it does not accommodate the reservoir management problem as the source term for the saturation equation depends on the state (saturation). Likewise, the objective function for the well-rate optimal control problem also has a coupling of the state (saturation) and control variables (well rates). However, in both cases, the coupling is weak in the sense that the only changes needed are in the objective function and source term, $\ell(y_k, u_k) : \mathbb{R}^{ny} \to \mathbb{R}$ and $G(y_k, u_k) : \mathbb{R}^{ny} \to \mathbb{R}^{ny}$. To simplify notation, I will ignore this for now as the next chapters do not need this generality and revisit for the chapter on reservoir management.

Adapting the assumptions in Section 2.1 for the notation in (4.1.1), I restate the following assumptions in new notation.

(A1) There exists an open set $\mathcal{U} \subset \mathbb{R}^{nu}$ such that for every $\mathbf{u} = (u^T_1, \ldots, u^T_{nt})^T \in \mathcal{U}$ there exists a unique solution $\mathbf{y}(\mathbf{u}) = (y^T_1(v), \ldots, y^T_{nt}(v))^T$ of the state equation (4.1.1b).

(A2) There exists an open set $\mathcal{Y} \subset \mathbb{R}^{ny}$ such that $\{\mathbf{y}(\mathbf{u}) : \mathbf{u} \in \mathcal{U}\} \subset \mathcal{Y}$ and the functions $\ell : \mathcal{Y} \to \mathbb{R}, \sigma : \mathcal{U} \to \mathbb{R}, F : \mathcal{Y} \to \mathbb{R}^{ny}$, and $G : \mathcal{U} \to \mathbb{R}^{ny}$ are twice continuously differentiable.

(A3) For every $\mathbf{y} = (v^T_1, \ldots, v^T_{nt})^T \in \{\mathbf{y}(\mathbf{u}) : \mathbf{u} \in \mathcal{U}\}$ and every $r_1, \ldots, r_{nt} \in \mathbb{R}^{ny}$
there exists a unique solution \( w_1, \ldots, w_{n_t} \in \mathbb{R}^{n_y} \) of the equations

\[
\begin{align*}
  w_0 &= 0 \in \mathbb{R}^{n_y} \\
  w_{k+1} - w_k &= \Delta t F_y(y_{k+1})w_{k+1} + r_{k+1}, \quad k = 0, \ldots, n_t - 1.
\end{align*}
\]

Under the assumptions (A1)-(A3) the function \( J : \mathcal{U} \to \mathbb{R} \),

\[
J(\vec{u}) = \sum_{k=1}^{K} \ell(y_k(\vec{u}), u_k),
\]

and defined model problem (4.1.1) can be written as an unconstrained minimization problem

\[
\min_{\vec{u}} J(\vec{u}).
\]

Moreover the function \( J \) is twice continuously differentiable on \( \mathcal{U} \).

**Algorithm 4.1: Gradient Computation**

1: Solve the state equations

\[
M(y_{k+1} - y_k) + \Delta t F(y_{k+1}) = \Delta t G(u_{k+1}), \quad k = 0, \ldots, n_t - 1.
\]

for \( y_1, \ldots, y_{n_t} \).

2: Solve the adjoint equations

\[
\begin{align*}
  [M + \Delta t F_y(y_{n_t})]^T \lambda_{n_t} &= -\nabla_y \ell(y_{n_t}), \\
  [M + \Delta t F_y(y_k)]^T \lambda_k &= M\lambda_{k+1} + \nabla_y \ell(y_k), \quad k = n_t - 1, \ldots, 1,
\end{align*}
\]

for \( \lambda_{n_t}, \ldots, \lambda_1 \).

3: Compute the gradient

\[
\nabla J(\vec{u}) = \begin{pmatrix}
  \nabla_u \sigma(u_1) - \Delta t \, G_u(u_1)^T \lambda_1 \\
  \vdots \\
  \nabla_u \sigma(u_{n_t}) - \Delta t \, G_u(u_{n_t})^T \lambda_{n_t}
\end{pmatrix}.
\]
Algorithm 4.2: Hessian-times-vector computation – $\nabla^2 J(\mathbf{u})\mathbf{v}$

1: Solve the state equations

$$
M(y_{k+1} - y_k) + \Delta t \mathbf{F}(y_{k+1}) = \Delta t \mathbf{G}(u_{k+1}), \quad k = 0, \ldots, n_t - 1.
$$

for $y_1, \ldots, y_{n_t}$.

2: Solve the adjoint equations

$$
[M + \Delta t \mathbf{F}_y(y_{n_t})]^T \lambda_{n_t} = -\nabla_y \ell(y_{n_t}),
$$

$$
[M + \Delta t \mathbf{F}_y(y_k)]^T \lambda_k = M\lambda_{k+1} - \nabla_y \ell(y_k), \quad k = n_t - 1, \ldots, 1,
$$

for $\lambda_{n_t}, \ldots, \lambda_1$.

3: Solve the linearization,

$$
\mathbf{w}_0 = 0, \quad (4.1.2)
$$

$$
[M + \Delta t \mathbf{F}_y(y_{k+1})]w_{k+1} = Mw_k + \Delta t \mathbf{G}_u(u_{k+1})v_{k+1}, \quad k = 0, \ldots, n_t - 1,
$$

for $w_1, \ldots, w_{n_t}$.

4: Solve the 2nd order adjoint equations

$$
[M + \Delta t \mathbf{F}_y(y_{n_t-1})]^T \mathbf{p}_{n_t} = -[\Delta t (\lambda_{n_t}^T \mathbf{F}(y_{n_t}))_{yy} + \ell_{yy}(y_{n_t})]w_{n_t}, \quad (4.1.3)
$$

$$
[M + \Delta t \mathbf{F}_y(y_k)]^T \mathbf{p}_k = M\mathbf{p}_{k+1} - [\Delta t (\lambda_k^T \mathbf{F}(y_k))_{yy} + \ell_{yy}(y_k)]w_k,
$$

for $\mathbf{p}_{n_t}, \ldots, \mathbf{p}_1$.

5: Compute the action of the Hessian

$$
\nabla^2 J(\mathbf{u})\mathbf{v} = \left( \begin{array}{c}
-\Delta t \mathbf{G}_u(u_1)^T \mathbf{p}_1 + [\sigma_{uu}(u_1) - \Delta t (\lambda_1^T \mathbf{G}(u_1))_{uu}]v_1 \\
\vdots \\
-\Delta t \mathbf{G}_u(u_{n_t})^T \mathbf{p}_{n_t} + [\sigma_{uu}(u_{n_t}) - \Delta t (\lambda_{n_t}^T \mathbf{G}(u_{n_t}))_{uu}]v_{n_t}
\end{array} \right).
$$
4.2 Hessian Approximation by Model Order Reduction

Instead of computing the Hessian of the original problem $\nabla^2 J(u)$ or the Hessian of the reduced problem in the traditional framework $\nabla^2 \tilde{J}(u)$, I introduce a third Hessian, $\tilde{\nabla}^2 \tilde{J}(u)$. As equations (4.1.2) and (4.1.3) require solving linear PDEs they can be replaced with subspace-based model reductions:

1. Substitute $w_k \approx V \hat{w}_k, p_k \approx V \hat{p}_k$ into (4.1.2) and (4.1.3).

2. Force the residual error at each time step to be perpendicular to the subspace spanned by $V$,

$$[I + \Delta t V^T F_y(y_{k+1})V] \hat{w}_{k+1} = \hat{w}_k + \Delta t V^T G_u(u_{k+1})v_{k+1},$$

$$[I + \Delta t V^T F_y(y_{n_t})V]^T \hat{p}_{n_t} = -V^T [\Delta t (\lambda_{n_t}^T F(y_{n_t}))_{yy} + \ell_{yy}(y_{n_t})]V \hat{w}_{n_t},$$

$$[I + \Delta t V^T F_y(y_{k})V]^T \hat{p}_k = \hat{p}_{k+1} - V^T [\Delta t (\lambda_k^T F(y_{k}))_{yy} + \ell_{yy}(y_{k})]V \hat{w}_k.$$

Replacing the full PDE solves in Algorithm 4.2 with their POD approximations results in Algorithm 4.4. As mentioned in previous chapters, we construct $V \in \mathbb{R}^{n_y \times n_r}$ satisfying $V^T M V = I$ by running POD on the state $\tilde{y}(\tilde{u})$ and adjoint $\tilde{\lambda}(\tilde{u})$ solutions. See Algorithm 4.3.

**Algorithm 4.3:** Construction of POD Subspace

1. Given the solution to the state and adjoint equations, collate $y_1, \ldots, y_{n_t}$ and $\lambda_1, \ldots, \lambda_{n_t}$ into snapshot matrices,

$$Y = [y_1, \ldots, y_{n_t}], \quad \Lambda = [\lambda_1, \ldots, \lambda_{n_t}].$$

2. Construct by truncated SVD matrices $V_y \in \mathbb{R}^{n_y \times r_y}$ and $V_\lambda \in \mathbb{R}^{n_y \times r_\lambda}$ so that

$$\frac{\| (I - V_y V_y^T) Y \|}{\| Y \|} < tol \quad \text{and} \quad \frac{\| (I - V_\lambda V_\lambda^T) \Lambda \|}{\| \Lambda \|} < tol.$$

3. Orthogonalize, orth($[V_y \ V_\lambda]$) $\mapsto V$.

We are now ready to write the modified Hessian-time-vector computation in Algorithm 4.4. The subspace matrix $V \in \mathbb{R}^{n_y \times n_r}$ used in steps (3) and (4) is computed via Algorithm 4.3.
Algorithm 4.4: Hessian Times Vector Computation w/ MOR – $\nabla^2 J(\hat{\mathbf{u}}) \hat{\mathbf{v}}$

1: Solve the state equations

\[ \mathbf{M}(y_{k+1} - y_k) + \Delta t \mathbf{F}(y_{k+1}) = \Delta t \mathbf{G}(u_{k+1}), \quad k = 0, \ldots, n_t - 1. \]

for $y_1, \ldots, y_{n_t}$.

2: Solve the adjoint equations

\[
\begin{align*}
\left[ \mathbf{M} + \Delta t \mathbf{F}_y(y_{n_t}) \right]^T \lambda_{n_t} &= -\nabla y \ell (y_{n_t}), \\
\left[ \mathbf{M} + \Delta t \mathbf{F}_y(y_k) \right]^T \lambda_k &= \mathbf{M} \lambda_{k+1} - \nabla y \ell (y_k), \quad k = n_t - 1, \ldots, 1,
\end{align*}
\]

for $\lambda_{n_t}, \ldots, \lambda_1$.

3: Solve the POD linearization for $\hat{\mathbf{w}}_k$,

\[
\begin{align*}
\left[ \mathbf{I} + \Delta t \mathbf{V}^T \mathbf{F}_y(y_{k+1}) \mathbf{V} \right] \hat{\mathbf{w}}_{k+1} &= \hat{\mathbf{w}}_k + \Delta t \mathbf{V}^T \mathbf{G}_u(u_{k+1}) \mathbf{v}_{k+1}, \\
&= 0, \ldots, n_t - 1
\end{align*}
\]

where $\hat{\mathbf{w}}_0 = \mathbf{V}^T \mathbf{w}_0 = 0$.

4: Solve the POD adjoint equation for $\hat{\mathbf{p}}_k$,

\[
\begin{align*}
\left[ \mathbf{I} + \Delta t \mathbf{V}^T \mathbf{F}_y(y_{n_t}) \mathbf{V} \right]^T \hat{\mathbf{p}}_{n_t} &= -\mathbf{V}^T \left[ \Delta t \left( \lambda_{n_t}^T \mathbf{F}(y_{n_t}) \right) y + \ell_{yy}(y_{n_t}) \right] \mathbf{V} \hat{\mathbf{w}}_{n_t}, \\
\left[ \mathbf{I} + \Delta t \mathbf{V}^T \mathbf{F}_y(y_k) \mathbf{V} \right]^T \hat{\mathbf{p}}_k &= \hat{\mathbf{p}}_{k+1} - \mathbf{V}^T \left[ \Delta t \left( \lambda_k^T \mathbf{F}(y_k) \right) y + \ell_{yy}(y_k) \right] \mathbf{V} \hat{\mathbf{w}}_k, \\
&= 0, \ldots, n_t - 1, 1.
\end{align*}
\]

5: Compute the approximation to the action of the Hessian,

\[
\nabla^2 J(\hat{\mathbf{u}}) \hat{\mathbf{v}} = \left( \begin{array}{c}
-\Delta t \mathbf{G}_u(u_1)^T \mathbf{V} \hat{\mathbf{p}}_1 + [\sigma_{uu}(u_1) - \Delta t (\lambda_1^T \mathbf{G}(u_1))_{uu}] \mathbf{v}_1 \\
\vdots \\
-\Delta t \mathbf{G}_u(u_{n_t-1})^T \mathbf{V} \hat{\mathbf{p}}_{n_t} + [\sigma_{uu}(u_{n_t-1}) - \Delta t (\lambda_{n_t}^T \mathbf{G}(u_1))_{uu}] \mathbf{v}_{n_t}
\end{array} \right)
\]

Note that steps (1) and (2) and $\mathbf{V}$ need only be computed once per control $\hat{\mathbf{u}}$. 
The Newton subproblem is solved while using the full order gradient and approximate Hessian to get a descent direction $\vec{d}$,
\[ \nabla^2 J(\bar{u}) \vec{d} = -\nabla J(\bar{u}). \]  
(4.2.1)

The modified Newton-CG algorithm is outlined in Algorithm 4.5.

**Algorithm 4.5: Modified Newton-CG**

1: loop
2: Given control, $\bar{u}$, compute the gradient $\nabla J(\bar{u})$.
3: Solve the modified Newton subproblem for $\vec{d}$ via the Conjugate Gradient method:
   \[ \nabla^2 J(\bar{u}) \vec{d} = -\nabla J(\bar{u}) \]
4: Choose $\alpha > 0$ so that $\bar{u} + \alpha \vec{d}$ satisfies the Armijo line search condition,
   \[ J(\bar{u} + \alpha \vec{d}) \leq J(\bar{u}) + c_1 \alpha \nabla J(\bar{u})^T \vec{d} \]
5: Update $\bar{u} + \alpha \vec{d} \rightarrow \bar{u}$.
6: end loop

I will demonstrate that this approach dramatically speeds up the Newton subproblem. Before doing so, I will explain how I propose constructing the approximated Hessian via POD on the state and adjoint solutions. The proposed quasi-Newton method can be found in Algorithm 4.5.

**Symmetry of the ROM Hessian** While one would never construct the Hessian in practice, writing the Hessian in closed form is helpful in understanding its structure. The Hessian in a consolidated, closed form by writing the parabolic implicit constraint,
\[ c(\bar{y}(\bar{u}), \bar{u}) := \begin{bmatrix} 
M(y_1 - y_0) + \Delta t F(y_1) - \Delta t G(u_1) \\
\vdots \\
M(y_{n_t} - y_{n_t-1}) + \Delta t F(y_{n_t}) - \Delta t G(u_{n_t}) 
\end{bmatrix}, \]
and Lagrangian with Lagrange multiplier \( \mathcal{X}(\bar{u}) = (\lambda_1^T, \ldots, \lambda_{n_t}^T)^T \),

\[
L(\bar{y}(\bar{u}), \bar{u}, \mathcal{X}(\bar{u})) := J(\bar{u}) + \lambda^T c(\bar{y}(\bar{u}), \bar{u}).
\]

Then the Hessian can be written in closed form,

\[
\nabla^2 J(\bar{u}) = c_u(\bar{y}(\bar{u}), \bar{u})^T c_y(\bar{y}(\bar{u}), \bar{u})^{-1} c_u(\bar{y}(\bar{u}), \bar{u}) + \nabla^2_u L(\bar{y}(\bar{u}), \bar{u}, \mathcal{X}(\bar{u})).
\]

From this formulation, the symmetry of the Hessian becomes apparent. Note here that derivatives of the implicit constraint \( c_y(\bar{y}(\bar{u}), \bar{u}) \in \mathbb{R}^{n_y n_t \times n_y n_t} \) forms a lower block diagonal matrix representing the spatial-temporal differential operator. As this matrix is enormous in size, linear systems are solved via block-forward substitution in time which is in fact equivalent to the linearization in the Hessian-time-vector computation for a particular choice of right-hand side. Since the subspace-based model reduction is applied at every time step, then ROM linearization and adjoint can be written with block diagonal matrix,

\[
\mathbf{\hat{V}} = \begin{bmatrix}
V \\
\vdots \\
V
\end{bmatrix} \in \mathbb{R}^{n_y n_t \times n_r n_t},
\]

and the reduced spatial temporal operator can be written as

\[
\mathbf{\hat{V}}^T c_y(\bar{y}(\bar{u}), \bar{u}) \mathbf{\hat{V}} \in \mathbb{R}^{n_r n_t \times n_r n_t}.
\]

Then quasi-Hessian can be written in the same concise form. However, to consolidate notation, let \( \mathbf{\hat{c}}_c(\bar{y}(\bar{u}), \bar{u}) := \mathbf{\hat{V}}^T c_y(\bar{y}(\bar{u}), \bar{u}) \mathbf{\hat{V}} \) and \( \mathbf{\hat{c}}_u(\bar{y}(\bar{u}), \bar{u}) := \mathbf{\hat{V}}^T c_u(\bar{y}(\bar{u}), \bar{u}) \). Then

\[
\nabla^2 \mathcal{X}(\bar{u}) = \mathbf{\hat{c}}_u(\bar{y}(\bar{u}), \bar{u})^T \mathbf{\hat{c}}_y(\bar{y}(\bar{u}), \bar{u})^{-1} \mathbf{\hat{c}}_u(\bar{y}(\bar{u}), \bar{u}) + \nabla^2_u L(\bar{y}(\bar{u}), \bar{u}, \mathcal{X}(\bar{u})),
\]

where \( \bar{y} := \bar{y}(\bar{u}) \) and \( \mathcal{X} := \mathcal{X}(\bar{u}) \).
Therefore, like the full Hessian $\nabla^2 J(\bar{u})$, the ROM Hessian is also symmetric,

$$[\tilde{\nabla}^2 J(\bar{u})]^T = \tilde{\nabla}^2 J(\bar{u}).$$

Additionally, to justify why we anticipate $\tilde{\nabla}^2 J(\bar{u}) \approx \nabla^2 \tilde{J}(\bar{u})$, note that their expressions are equivalent save the evaluations of state and adjoint: specifically, $\bar{y}(\bar{u})$ and $\bar{\lambda}(\bar{u})$ are replaced with the subspace-based surrogates,

$$\tilde{y}(\bar{u}) := [(V\hat{y}_1(\bar{u}))^T, \ldots, (V\hat{y}_{nt}(\bar{u}))^T]^T,$$

$$\tilde{\lambda}(\bar{u}) := [(V\hat{\lambda}_1(\bar{u}))^T, \ldots, (V\hat{\lambda}_{nt}(\bar{u}))^T]^T,$$

in the expression for $\nabla^2 \tilde{J}(\bar{u})$. That is,

$$\tilde{\nabla}^2 \tilde{J}(\bar{u}) = \tilde{c}_u(\tilde{y}, \bar{u})^T \tilde{c}_y(\tilde{y}, \bar{u})^{-T} [\tilde{V}^T \nabla^2 \tilde{L}(\tilde{y}, \bar{u}, \bar{\lambda}) \tilde{V}] \tilde{c}_y(\tilde{y}, \bar{u})^{-1} \tilde{c}_u(\tilde{y}, \bar{u})$$

$$+ \nabla^2 \tilde{L}(\tilde{y}, \bar{u}, \bar{\lambda}),$$

where $\tilde{y} := \tilde{y}(\bar{u})$ and $\tilde{\lambda} := \tilde{\lambda}(\bar{u})$.

### 4.3 Inexact Newton and Convergence

Replacing the full order Newton subproblem, $\nabla^2 J(\bar{u}) \bar{d} = -\nabla J(\bar{u})$, with the approximated one, (4.2.1) generates natural questions as to whether the approach converges to a minimizer $\bar{u}^\star$. I analyze the convergence by following the theory for Inexact Newton methods.

#### 4.3.1 Local Convergence

To prove local convergence I restate the following theorem from Dembo et al. (1982), [17] on the convergence of inexact Newton.

**Theorem 4.1.** Suppose that $\nabla^2 J(\bar{u})$ exists and is continuous in a neighborhood of a minimizer $\bar{u}^\star$ with $\nabla^2 J(\bar{u}^\star)$ positive definite. Consider the iterations
\( \mathbf{u}^{(j+1)} = \mathbf{u}^{(j)} + \mathbf{p}^{(j)} \) where \( \mathbf{p}^{(j)} \) satisfies

\[
\| \nabla^2 J(\mathbf{u}^{(j)}) \mathbf{p}^{(j)} + \nabla J(\mathbf{u}^{(j)}) \| \leq \eta^{(j)} \| \nabla J(\mathbf{u}^{(j)}) \|. \quad (4.3.1)
\]

Assume that for all \( j \), \( \eta^{(j)} \leq \eta \) for some constant \( \eta \in [0,1) \). Then, if the starting point \( \mathbf{u}^{(0)} \) is sufficiently near \( \mathbf{u}^* \), the sequence \( \{\mathbf{u}^{(j)}\} \) converges to \( \mathbf{u}^* \) such that

\[
\| \nabla^2 J(\mathbf{u}^*)(\mathbf{u}^{(j+1)} - \mathbf{u}^*) \| \leq \hat{\eta} \| \nabla^2 J(\mathbf{u}^*)(\mathbf{u}^{(j)} - \mathbf{u}^*) \|
\]

for some constant \( \eta < \hat{\eta} < 1 \).

The authors Dembo et al. demonstrate that if the \( \eta^{(j)} \to 0 \), then the convergence \( \mathbf{u}^{(j)} \to \mathbf{u}^* \) is superlinear instead of linear [17]. However, for the proposed method, we do not expect \( \eta^{(j)} \) to decrease arbitrarily without decreasing the POD and DEIM tolerances (i.e., increasing the size of the reduced order model). Hence, I omit that detail of their theorem when modifying it to apply to my method.

**Theorem 4.2.** Assume that \( \nabla^2 J(\mathbf{u}) \) is symmetric positive definite in a neighborhood of \( \mathbf{u}^* \) and that \( \nabla^2 J(\mathbf{u}^{(j)}) \) are symmetric positive definite for \( k \geq 0 \). Let \( \{\mathbf{u}^{(j)}\} \) denote the sequence of quasi-Newton updates where

\[ \mathbf{u}_{k+1} = \mathbf{u}^{(j)} - [\nabla^2 J(\mathbf{u}^{(j)})]^{-1} \nabla J(\mathbf{u}^{(j)}). \]

Then if the starting point \( \mathbf{u}^{(0)} \) is sufficiently near \( \mathbf{u}^* \) and \( \| \nabla^2 J(\mathbf{u}^{(j)})[\nabla^2 J(\mathbf{u}^{(j)})]^{-1} - \mathbf{I} \| \leq \eta \) for all \( j \geq 0 \) and for some \( \eta < 1 \) then the sequence \( \{\mathbf{u}^{(j)}\} \) converges to \( \mathbf{u}^* \) \( q \)-linearly in the weighted norm,

\[
\| \nabla^2 J(\mathbf{u}^*)(\mathbf{u}^{(j+1)} - \mathbf{u}^*) \| \leq \hat{\eta} \| \nabla^2 J(\mathbf{u}^*)(\mathbf{u}^{(j)} - \mathbf{u}^*) \|
\]

for some \( \eta < \hat{\eta} < 1 \).

**Proof.** Let \( \mathbf{p}^{(j)} = -[\nabla^2 J(\mathbf{u}^{(j)})]^{-1} \nabla J(\mathbf{u}^{(j)}) \) from the quasi-Newton subproblem and denote the residual \( \mathbf{r}^{(j)} = \nabla^2 J(\mathbf{u}^{(j)}) \mathbf{p}^{(j)} + \nabla J(\mathbf{u}^{(j)}) \). Then

\[
\| \mathbf{r}^{(j)} \| = \| - \nabla^2 J(\mathbf{u}^{(j)})[\nabla^2 J(\mathbf{u}^{(j)})]^{-1} \nabla J(\mathbf{u}^{(j)}) + \nabla J(\mathbf{u}^{(j)}) \|
\leq \| \nabla^2 J(\mathbf{u}^{(j)})[\nabla^2 J(\mathbf{u}^{(j)})]^{-1} - \mathbf{I} \| \| \nabla J(\mathbf{u}^{(j)}) \|
\leq \eta \| \nabla J(\mathbf{u}^{(j)}) \|.\]
Therefore the hypotheses of Theorem 4.1 are satisfied and

$$\|\nabla^2 J(\tilde{u}^*) (\tilde{u}^{(j+1)} - \tilde{u}^*)\| \leq \tilde{\eta} \|\nabla^2 J(\tilde{u}^*) (\tilde{u}^{(j)} - \tilde{u}^*)\|$$

for $\tilde{u}^{(0)}$ sufficiently near $\tilde{u}^*$. Since $\nabla^2 J(\tilde{u}^*)$ is symmetric positive definite, it follows that $\|\tilde{u}^{(j)} - \tilde{u}^*\| \to 0$.

We note that $\|\nabla^2 J(\tilde{u}^{(j)}) - \nabla^2 J(\tilde{u}^{(j)}(\tilde{u}^{(j)}))\| < \eta \sigma_n \left( \nabla^2 J(\tilde{u}^{(j)}) \right)$ where $\sigma_n(A)$ is the smallest singular value of $A$, is also sufficient as a hypothesis to the previous theorem. However, this condition can be much more restrictive than the one I chose.

While local convergence to the minimizer is guaranteed, the quadratic local convergence of Newton’s method is lost however. Newton’s method exhibits quadratic convergence because for each iteration, the method builds a quadratic approximation to the full optimization problem via a Taylor expansion. The Newton subproblem, (4.2.1), then solves for the minimizer of this quadratic approximation.

The approach presented does not generate the true quadratic expansion in Newton’s method but approximates it instead. Hence, the convergence rate is no longer quadratic, but linear as the control $\tilde{u}_k$ approaches the optimizer $\tilde{u}^*$ when $\tilde{u}^{(0)}$ is sufficiently close to $\tilde{u}^*$. However, in practice optimization methods run until the norm of the gradient, $\|\nabla J(\tilde{u})\|$ drops below a specified tolerance. Therefore if POD and DEIM tolerances are chosen sufficiently small, then the quasi-Hessian $\nabla^2 \tilde{J}(\tilde{u})$ is a good approximation of the Hessian $\nabla^2 J(\tilde{u})$ and consequently, the convergence behavior appears quadratic early in the optimization and stalls as the inaccuracy of the Hessian.

The observation here is that if the algorithm is terminated early, the linear convergence of the quasi-Newton method will not be realized and the convergence history will appear quadratic. Hence, in large-scale optimization scenarios where few iterations of Newton’s Method or Gradient Descent are feasible, the acceleration of this approach is dramatic.
4.4 Spectral Characteristics of the Reduced-Order Hessian for an Example Problem

Ideally, one would like to say something about the positive definiteness of the reduced-order Hessian. However, even when the full-order Hessian is symmetric positive definite, it does not necessarily follow that the ROM Hessian is as well save a few special cases (e.g. linear implicit constraint with quadratic objective function). Despite this, it is still useful to analyze the Hessian for a specific case to appreciate the effect the regularization as on the convexity of the quadratic approximations generated by Newton’s Method.

Convergence guarantees for quasi-Newton methods depend on the positive definiteness of the reduced-order Hessian $\tilde{\nabla}^2 J(\mathbf{u})$ evaluated at the optimizer $\mathbf{u} = \mathbf{u}^*$. The model reduction scheme proposed decreases the convexity of objective function as is demonstrated in Figure 4.1.

Example Problem The optimal control problem used is the following time-dependent semilinear parabolic control problem with cubic nonlinearity.

Minimize $\frac{1}{2} \int_0^1 \int_\Omega (y(x,t;u) - y_d(x,t))^2 dx dt + \frac{\alpha}{2} \int_0^1 \int_\Omega u(x,t)^2 dx dt$, \hspace{1cm} (4.4.1a)

where for given function $u$ the function $y(\cdot, \cdot; u)$ is the solution of

\begin{align*}
y_t - \Delta y + y^3 &= u \quad (x,t) \in \Omega \times (0,T), \hspace{1cm} (4.4.1b) \\
y(x,t) &= 0 \quad (x,t) \in \partial\Omega \times (0,T), \hspace{1cm} (4.4.1c) \\
y(x,0) &= y_0(x) \quad x \in \Omega.
\end{align*}

For this problem, mixed derivative terms of control and state ($u$ and $y$) do not appear in either the objective function or the constraint. Therefore, both mixed derivative terms of the Lagrangian vanish

$$\nabla_{\tilde{y}u} L(\tilde{y}(\tilde{u}), \tilde{u}, \tilde{\lambda}(\tilde{u}))^T = \nabla_{\tilde{y}y} L(\tilde{y}(\tilde{u}), \tilde{u}, \tilde{\lambda}(\tilde{u})) = 0.$$
Note that both Hessians $\nabla^2 J(\tilde{u})$ and $\nabla^2 \tilde{J}(\tilde{u})$ in equations (4.2.2) and (4.2.3) share a common regularization term, $\nabla^2_\Omega L(\tilde{y}(\tilde{u}), \tilde{u}, \tilde{\lambda}(\tilde{u}))$. In the following figure we analyze the spectral characteristics of both Hessians in Figure 4.1.

Figure 4.1: The regularization term $\nabla^2_\Omega L(\tilde{y}(\tilde{u}), \tilde{u}, \tilde{\lambda}(\tilde{u}))$ is a discretization of the $L^2(\Omega)$ regularization penalty. Results for $\Omega = (0,1)^2$, $T = 1$, $\alpha = 10^{-6}$, $y_0(x) = \sin(2\pi x_1)$ and $y_d(x,t) = 2e^t + 2(x_1(x_1 - 1) + x_2(x_2 - 1))$. Solved by piecewise linear finite element method with triangular uniform mesh, $n_x = n_y = 10$ and $n_t = 20$.

Note that as the leading term of the ROM Hessian $\nabla^2 \tilde{J}(\tilde{u})$ becomes rank-deficient, the regularization parameter keeps the ROM Hessian symmetric positive definite.

Note that for this particular example, the assumption that the reduced-order Hessian is symmetric positive definite at the optimizer is satisfied; however, this is not expected to hold in general. An important observation is that the symmetric positive definiteness of the Hessian is lost to some degree as the first term in the Hessian becomes low-rank in the quasi-Hessian approximation. Hence, some of the convexity of the problem is lost. However, the regularization term saves the day, making the reduced order Hessian positive definite.
4.5 ROM Hessian Error Bounds

First, to simplify notation in the following section, let us redefine the notation in the gradient and Hessian vector notation acknowledging that for a given control \( \tilde{u} \), the corresponding state and adjoint solutions, \( \tilde{\mathbf{y}}(\tilde{u}) \) and \( \tilde{\lambda}(\tilde{u}) \) are fixed during the Conjugate Gradient iterations in the Newton subproblem, \( \nabla^2 J(\tilde{u}) = -\nabla J(\tilde{u}) \). Define

\[
A_k := F_y(y_k) \in \mathbb{R}^{n_y \times n_y},
B_k := G_u(u_k) \in \mathbb{R}^{n_y \times n_u},
c_k := \nabla_y \ell(y_k) \in \mathbb{R}^{n_y},
d_k := \nabla_u \sigma(u_k) \in \mathbb{R}^{n_u},
H_k := \ell_{yy}(y_k) + \Delta t (\lambda_k^T F(y_k))_{yy} \in \mathbb{R}^{n_y \times n_y},
R_k := \sigma_{uu}(u_k) - \Delta t (\lambda_k^T G(u_k))_{uu} \in \mathbb{R}^{n_u \times n_u}.
\]

The following relation between the Newton subproblem \( \nabla^2 J(\tilde{u}) \tilde{v} = -\nabla J(\tilde{u}) \) and a linear quadratic program is show, (e.g. Heinkenschloss (2008) [29]). If \( \nabla^2 J(\tilde{u}) \) is symmetric positive definite, then \( \tilde{v} = (v_1^T, \ldots, v_{n_t}^T) \in \mathbb{R}^{n_u n_t} \) solves the Newton system,

\[
\nabla^2 J(\tilde{u}) \tilde{v} = -\nabla J(\tilde{u}),
\] (4.5.1)

if and only if \( \tilde{v} = (v_1^T, \ldots, v_{n_t}^T) \in \mathbb{R}^{n_u n_t}, \tilde{w} = (w_1^T, \ldots, w_{n_t}^T) \in \mathbb{R}^{n_y n_t} \) solve

Minimize \( \sum_{k=1}^{K} c_k^T w_k + d_k^T v_k + \frac{1}{2} w_k^T H_k w_k + \frac{1}{2} v_k^T R_k v_k \) \hspace{1cm} (4.5.2a)

subject to \( M(w_{k+1} - w_k) + \Delta t A_{k+1} w_{k+1} = \Delta t B_{k+1} v_{k+1}, \quad k = 0, \ldots, n_t - 1, \) \hspace{1cm} (4.5.2b)

\( w_0 = 0. \) \hspace{1cm} (4.5.2c)

The program (4.5.2) is equivalent to minimizing

\[
\min \frac{1}{2} \tilde{v}^T [\nabla^2 J(\tilde{u})] \tilde{v} + \tilde{v}^T \nabla J(\tilde{u})
\]
where

\[
\vec{v}^T [\nabla^2 J(\vec{u})] \vec{v} = \sum_{k=1}^{K} w_k^T H_k w_k + v_k^T R_k v_k, \\
\vec{v}^T [\nabla J(\vec{u})] = \sum_{k=1}^{K} v_k^T d_k + w_k^T c_k.
\]

Introduce a new objective function for the linear / quadratic problem formulated in (4.5.2) not to be confused with the original minimization problem,

\[
\text{Minimize } Q(\vec{v}) := \sum_{k=1}^{K} c_k^T w_k + d_k^T v_k + \frac{1}{2} w_k^T H_k w_k + \frac{1}{2} v_k^T R_k v_k
\]

where \(\vec{w}\) and \(\vec{v}\) satisfy the implicit constraint

\[
M (w_{k+1} - w_k) + \Delta t A_{k+1} w_k = \Delta t B_{k+1} v_{k+1}.
\]

The gradient \(\nabla Q(\vec{v})\) can be computed,

\[
\nabla Q(\vec{v}) = \begin{pmatrix}
-\Delta t B_1^T q_1 + d_1 + R_1 v_1 \\
\vdots \\
-\Delta t B_n^T q_n + d_n + R_n v_n
\end{pmatrix},
\]

(4.5.3)

where \(\vec{q} = (q_1^T, \ldots, q_n^T)^T \in \mathbb{R}^{n_y n_t}\) solves the adjoint solution

\[
[M + \Delta t A_k]^T q_n = -c_{n_t} - H_{n_t} w_{n_t}, \\
[M + \Delta t A_k]^T q_k = M q_{k+1} - c_k - H_k w_k,
\]

for \(k = n_t - 1, \ldots, 1\).

Note here that by linearity \(q_k = \lambda_k + p_k\) where \(\lambda_k\) solves the adjoint equation in the gradient computation, step 2 in Algorithm 4.1 and \(p_k\) solves the 2nd order adjoint equation in the Hessian-vector equation, step 4 in Algorithm 4.2.

### 4.5.1 Modified Newton Subproblem

Consider the modified Newton subproblem where we solve

\[
[\nabla^2 J(\vec{u})] \vec{v} = -\nabla J(\vec{u}),
\]

(4.5.4)
with
\[ \nabla^2 \tilde{J}(\vec{u}) \tilde{\vec{v}} = \sum_{k=1}^{n_t} \hat{w}_k^T V_k^T H_k V \tilde{\hat{w}}_k + \vec{v}_k^T R_k \vec{v}_k, \]
\[ \nabla J(\vec{u}) \tilde{\vec{v}} = \sum_{k=1}^{n_t} \vec{v}_k^T \hat{d}_k + \hat{w}_k^T \hat{c}_k. \]

Define a new functional
\[ \tilde{Q}(\vec{v}) := \frac{1}{2} \nabla^2 \tilde{J}(\vec{u}) \tilde{\vec{v}} + \nabla J(\vec{u}) \tilde{\vec{v}}. \]

and quadratic program,
\[ \text{Minimize } \tilde{Q}(\vec{v}) = \sum_{k=1}^{K} c_k^T \vec{w}_k + \vec{d}_k^T \vec{v}_k + \frac{1}{2} \hat{w}_k^T V_k^T H_k V \hat{\hat{w}}_k + \frac{1}{2} \vec{v}_k^T R_k \vec{v}_k, \]
where \(\vec{w}, \hat{w}\) and \(\vec{v}\) satisfy the implicit constraints
\[ M(\vec{w}_{k+1} - \vec{w}_k) = \Delta t A_{k+1} \vec{w}_{k+1} + \Delta t B_{k+1} \vec{v}_{k+1}, \tag{4.5.5a} \]
\[ \hat{w}_{k+1} - \hat{w}_k = \Delta t V^T A_{k+1} V \hat{\hat{w}}_{k+1} + \Delta t V^T B_{k+1} \vec{v}_{k+1}. \tag{4.5.5b} \]

Then the gradient for this problem can be computed
\[ \nabla \tilde{Q}(\vec{v}) = \left( \begin{array}{c} d_1 + R_1 \vec{v}_1 - \Delta t B_1^T \lambda_1 - \Delta t V^T B_1^T \hat{\hat{p}}_1 \\ \vdots \\ d_{n_t} + R_{n_t} \vec{v}_{n_t} - \Delta t B_{n_t}^T \lambda_{n_t} - \Delta t V^T B_{n_t}^T \hat{\hat{p}}_{n_t} \end{array} \right) \tag{4.5.6} \]
where \(\lambda = (\lambda_1^T, \ldots, \lambda_{n_t}^T)^T \in \mathbb{R}^{n_t \times n_t}\) and \(\hat{\hat{p}} = (\hat{\hat{p}}_1^T, \ldots, \hat{\hat{p}}_{n_t}^T)^T \in \mathbb{R}^{n_t \times n_t}\) satisfy
\[ [M + \Delta t A_k]^T \lambda_k = \lambda_{k+1} - c_k, \tag{4.5.7a} \]
\[ [I + \Delta t V^T A_k V]^T \hat{\hat{p}}_k = \hat{\hat{p}}_{k+1} - V^T H_k V \hat{\hat{w}}_k. \tag{4.5.7b} \]

Notice that \(\{\lambda_1, \ldots, \lambda_{n_t}\}\) is from the adjoint solve in the gradient computation of \(\nabla J(\vec{u})\) and \(\{\hat{\hat{p}}_1, \ldots, \hat{\hat{p}}_{n_t}\}\) is from the 2nd order adjoint solve in the Hessian-vector computation of \(\nabla^2 J(\vec{u})) \tilde{\vec{v}}.\)
Lemma 4.3. Assume that $\nabla^2 J(\bar{u})$ and $\nabla^2 J(\bar{u})$ are symmetric positive definite and let $\bar{v}^* \in \mathbb{R}^{n_{nu}}$ satisfy the modified Newton subproblem (4.5.4). Then the residual can be bounded by the errors in the linearization and 2nd-order adjoint equations,

$$\|\nabla^2 J(\bar{u})\bar{v}^* + \nabla J(\bar{u})\|^2 = \sum_{k=1}^{n_t} \|\Delta t B^T_k (p_k - \hat{V}_k)\|^2,$$

where $p_k$ solves the adjoint equation in the Hessian-vector equation, step 3 in Algorithm 4.2 and $\hat{p}_k$ solves (4.5.7b) when $v = \bar{v}^*$.

Proof. First we recall that $\nabla^2 J(\bar{u})\bar{v}^* + \nabla J(\bar{u}) = 0$. Therefore we can substitute $\nabla J(\bar{u})$,

$$\|\nabla^2 J(\bar{u})\bar{v}^* + \nabla J(\bar{u})\| = \|\nabla^2 J(\bar{u})\bar{v}^* - \nabla^2 J(\bar{u})\bar{v}^*\|

= \|\nabla Q(\bar{v}^*) - \nabla \tilde{Q}(\bar{v}^*)\|,$$

since

$$Q(\bar{v}) := \frac{1}{2} \bar{v}^T [\nabla^2 J(\bar{u})] \bar{v} + \bar{v}^T [\nabla J(\bar{u})],$$

$$\tilde{Q}(\bar{v}) := \frac{1}{2} \bar{v}^T [\nabla^2 J(\bar{u})] \bar{v} + \bar{v}^T [\nabla J(\bar{u})].$$

Therefore using (4.5.3) and (4.5.6)

$$\|\nabla Q(\bar{v}^*) - \nabla \tilde{Q}(\bar{v}^*)\|^2 = \sum_{k=1}^{n_t} \|\Delta t B^T_k (p_k - \hat{V}_k)\|^2.$$

\[\square\]

Perturbation Analysis  The following Lemma is a perturbation analysis of a linear, implicit discrete time system with a perturbation in the forcing term. First, recall the definition of the logarithmic Lipschitz constant defined in (2.2.5) in section 2.2,

$$\mathcal{M}_M[F] = \sup_{u \neq v} \frac{\langle u - v, F(u) - F(v) \rangle_M}{\|u - v\|^2_M}.$$
For a linear mapping $A \in \mathbb{R}^{n \times n}$, the logarithmic Lipschitz constant is equivalent to the logarithmic norm,

$$
M_{\text{log}}[A] = \sup_{v \neq 0} \frac{\langle v, Av \rangle_M}{\|v\|_M^2},
$$

which is used in the following lemma. For a discussion of the logarithmic norm and its generalization, the logarithmic Lipschitz constant, see Soderlind (2006) [48].

**Lemma 4.4.** Let \( \{y_k\} \) solve the implicit discretization for a given initial condition \( y_0 \),

$$
\mathbf{M} \left( \frac{y_{k+1} - y_k}{\Delta t} \right) + A_{k+1}y_{k+1} = g_{k+1},
$$

and let \( \{\tilde{y}_k\} \) solve a perturbed equation with the same initial condition \( y_0 \),

$$
\mathbf{M} \left( \frac{\tilde{y}_{k+1} - \tilde{y}_k}{\Delta t} \right) + A_{k+1}\tilde{y}_{k+1} = g_{k+1} + \epsilon_{k+1}.
$$

Assume that \( \Delta t \ M_{\text{log}}[-A_k] < 1 \) for \( k = 1, \ldots, n_t \). Then we can bound the error \( y_k - \tilde{y}_k \) so that

$$
\sum_{k=1}^{n_t} \|y_k - \tilde{y}_k\|_M^2 \leq \Delta t \ T q \left( \sum_{k=1}^{n_t} \|\epsilon_k\|_M^2 \right),
$$

where

$$q := \sum_{\ell=1}^{n_t} \zeta^{2\ell} = \zeta^2 \left( \frac{1 - \zeta^{2n_t}}{1 - \zeta^2} \right), \quad \zeta := \max_k \frac{1}{1 - \Delta t \ M_{\text{log}}[-A_k]}.
$$

**Proof.** Define the error term,

$$e_k := y_k - \tilde{y}_k.$$

Then by the Cauchy-Schwarz inequality,

$$
\frac{\|e_{k+1}\|_M - \|e_k\|_M}{\Delta t} \leq \frac{1}{\|e_{k+1}\|_M} \left\langle e_{k+1}, \frac{e_{k+1} - e_k}{\Delta t} \right\rangle_M = \frac{1}{\|e_{k+1}\|_M} \left\langle e_{k+1}, A_{k+1}(y_{k+1} - \tilde{y}_{k+1}) - \epsilon_{k+1} \right\rangle_M
$$

$$
\leq M_{\text{log}}[-A_{k+1}] \|e_{k+1}\|_M + \|\epsilon_{k+1}\|_M.
$$
Using this inequality, we can then bound the error at a given time step with $\zeta$ as it is defined in the hypothesis,

$$\|e_{k+1}\|_M \leq \zeta \|e_k\|_M + \Delta t \zeta \|e_{k+1}\|_M \leq \zeta^{k+1} \|e_0\|_M + \Delta t \sum_{\ell=1}^{k+1} \zeta^\ell \|e_{k+\ell+2}\|_M$$

Since $e_0 = y_0 - \tilde{y}_0 = 0$, then via Cauchy-Schwarz,

$$\|e_{k+1}\|_M \leq \Delta t \left( \sum_{\ell=1}^{k+1} \zeta^{2\ell} \right)^{1/2} \left( \sum_{\ell=1}^{k+1} \|e_\ell\|_M^2 \right)^{1/2} \leq \Delta t q^{1/2} \left( \sum_{\ell=1}^{n_t} \|e_\ell\|_M^2 \right)^{1/2},$$

where $q$ is defined in (4.5.8). Then finally the error bound can be written,

$$\sum_{k=1}^{n_t} \|e_k\|_M^2 \leq (\Delta t)^2 q \sum_{k=1}^{n_t} \left( \sum_{\ell=1}^{n_t} \|e_\ell\|_M^2 \right) = \Delta t q \left( \sum_{k=1}^{n_t} \|e_k\|_M^2 \right).$$

We will use this Lemma in the following theorem that bounds the error of the Hessian approximation.

**Theorem 4.5.** Given a control $\bar{\mathbf{u}}$ and associated state $\bar{\mathbf{y}} := \mathbf{y}(\bar{\mathbf{u}})$, assume that the Hessian $\nabla^2 J(\bar{\mathbf{u}})$ and ROM Hessian $\tilde{\nabla}^2 J(\bar{\mathbf{u}})$ are symmetric positive definite and let $\tilde{\mathbf{v}}^*$ denote the solution to the reduced Newton subproblem,

$$\tilde{\nabla}^2 J(\bar{\mathbf{u}}) \tilde{\mathbf{v}}^* = -\nabla J(\bar{\mathbf{u}}).$$

If the logarithmic norm of the linear mappings $A_k := \mathbf{F}_y(y_k) \in \mathbb{R}^{n_y \times n_y}$ satisfy $\Delta t \mathcal{M}_M[A_k] < 1$ and $\Delta t \mathcal{M}_I[-V^T A_k V] < 1$ for $k = 1, \ldots, n_t$, then the residual of the Newton subproblem can be bounded,

$$\|\nabla^2 J(\bar{\mathbf{u}}) \tilde{\mathbf{v}}^* + \nabla J(\bar{\mathbf{u}})\|^2 \leq (\max_{\ell} \|B_\ell\|_{M^{-1}}^2) \left[ 4C \left( \sum_{k=1}^{n_t} \|(I - VV^T M)P_k\|_M^2 \right) \right] + (2\Delta t T q + 4\Delta t T q K C^2) \left( \sum_{k=1}^{n_t} \|(I - VV^T M)w_k\|_M^2 \right).$$
where \( \{w_k\}, \{p_k\} \subset \mathbb{R}^n \) satisfy the linearization (4.1.2) and 2nd order adjoint (4.1.3) equations in the Hessian-time-vector computation for \( \tilde{\nu} = \tilde{\nu}^* \) and where

\[
C := 1 + \Delta t \, TL^2 q, \\
L := \max_k \|M^{-1/2}A_kM^{-1/2}\|, \\
q := \sum_{\ell=1}^n \zeta^{2\ell} = \zeta^2 \left( \frac{1 - \zeta^{2nt}}{1 - \zeta^2} \right), \\
\zeta = \max_k \left\{ \frac{1}{1 - \Delta t \, M\left[-A_k\right]} \right\}^{1/2}, \\
K := \max \|H_k\|^2_M. 
\]

**Proof.** For the entirety of the proof, set \( \tilde{\nu} = \tilde{\nu}^* = (\tilde{\nu}_1^T, \ldots, \tilde{\nu}_{nt}^T)^T \). Recall the linearization,

\[
[M + \Delta t \, A_{k+1}] w_{k+1} = Mw_k + \Delta t B_{k+1} \tilde{\nu}_{k+1}, \tag{4.5.9}
\]

and the second order adjoint equation,

\[
[M + \Delta t \, A_{nt}]^T p_{nt} = -H_k w_{nt}, \tag{4.5.10a}
\]

\[
[M + \Delta t \, A_k]^T p_k = M p_{k+1} - H_k w_k. \tag{4.5.10b}
\]

Since \( A_k \) is linear, the Lipschitz constant in Theorem 2.1 is equivalent to the maximum of the norms \( \{M^{-1/2}A_kM^{-1/2}\} \),

\[
L := \max_k \|M^{-1/2}A_kM^{-1/2}\|. 
\]

Let \( \{\tilde{w}_k\} \) solve the reduced-order linearization equation,

\[
[I_r + \Delta t \, V^T A_{k+1} V] \tilde{w}_{k+1} = \tilde{w}_{nt} + \Delta t \, V^T B_{k+1} \tilde{\nu}_{k+1}, \tag{4.5.11}
\]

and \( \{\tilde{p}_k\} \) solve the reduced-order second order adjoint equation,

\[
[I_r + \Delta t \, V^T A_{nt} V]^T \tilde{p}_{nt} = V^T H_k V \tilde{w}_{nt}, \tag{4.5.12a}
\]

\[
[I_r + \Delta t V^T A_k V]^T \tilde{p}_k = \tilde{p}_{k+1} + V^T H_k V \tilde{w}_k. \tag{4.5.12b}
\]
From Theorem 2.1 the error of the linearization (4.5.9) and the ROM linearization (4.5.11) can be bounded,
\[
\sum_{k=1}^{n_t} \| w_k - V\hat{w}_k \|_M^2 \leq (1 + \Delta t TL^2 q) \left[ \sum_{k=1}^{n_t} \| (I - VV^T M)w_k \|_M^2 \right].
\] (4.5.13)

To bound the quantity \( \sum_{k=1}^{n_t} \| p_k - V\hat{p}_k \|_M^2 \), Theorem 2.1 is not immediately applicable as the two systems differ in their right hand sides. Instead, I use the inequality
\[
\sum_{k=1}^{n_t} \| p_k - V\hat{p}_k \|_M^2 \leq \sum_{k=1}^{n_t} 2\| p_k - V\hat{p}_k \|_M^2 + 2\| p_k - \hat{p}_k \|_M^2,
\]
where \( \{\hat{p}_k\} \subset \mathbb{R}^{n_t} \) solves a modified linear system that is a perturbed version of (4.5.10) in the Hessian-vector computation where \( w_k \) is replaced with \( V\hat{w}_k \),
\[
[M + \Delta t A_{k+1}]^T \hat{p}_k = M\hat{p}_{k+1} + H_k V\hat{w}_k.
\] (4.5.14)

Using the same coefficient in (4.5.13) and Theorem 2.1, we can bound the perturbed adjoint equation (4.5.14) and the ROM adjoint equation, (4.5.12),
\[
\sum_{k=1}^{n_t} \| \hat{p}_k - V\hat{p}_k \|_M^2
\]
\[
\leq (1 + \Delta t TL^2 q) \left[ \sum_{k=1}^{n_t} \| (I - VV^T M)\hat{p}_k \|_M^2 \right]
\] (4.5.15a)
\[
\leq (1 + \Delta t TL^2 q) \left[ \sum_{k=1}^{n_t} 2\| (I - VV^T M)p_k \|_M^2 + 2\| (I - VV^T M)(p_k - \hat{p}_k) \|_M^2 \right]
\] (4.5.15b)
\[
\leq (1 + \Delta t TL^2 q) \left[ \sum_{k=1}^{n_t} 2\| (I - VV^T M)p_k \|_M^2 + 2\| p_k - \hat{p}_k \|_M^2 \right]
\] (4.5.15c)
\[
\leq (1 + \Delta t TL^2 q) \left[ \sum_{k=1}^{n_t} 2\| (I - VV^T M)p_k \|_M^2 + 2\| p_k - \hat{p}_k \|_M^2 \right].
\] (4.5.15d)

Applying Lemma 4.4 to the adjoint equation, the error \( \sum_{k=1}^{n_t} \| p_k - \hat{p}_k \|_M^2 \) can be bounded,
\[
\sum_{k=1}^{n_t} \| p_k - \hat{p}_k \|_M^2 \leq \Delta t T q \left( \sum_{k=1}^{n_t} \| H_k (w_k - V\hat{w}_k) \|_M^2 \right)
\]
\[
\leq \Delta t T q K \left( \sum_{k=1}^{n_t} \| w_k - V\hat{w}_k \|_M^2 \right),
\] (4.5.16)
where $K := \max \|H_k\|_M^2$. Define the coefficient $C := 1 + \Delta t TL^2 q$ and combining (4.5.15) and (4.5.16) we get,

$$
\sum_{k=1}^{n_t} \|\tilde{p}_k - V\hat{p}_k\|_M^2 \leq 2C \left[ \sum_{k=1}^{n_t} \|(I - VV^TM)p_k\|_M^2 \\
+ \Delta t TqKC \left( \sum_{k=1}^{n_t} \|(I - VV^TM)w_k\|_M^2 \right) \right].
$$

From Lemma 4.3,

$$
\|\nabla^2 J(\tilde{u})\tilde{v}^* + \nabla J(\tilde{u})\|^2 = \sum_{k=1}^{n_t} \|\Delta t B_k^T(p_k - V\hat{p}_k)\|^2,
$$

\[
\leq (\max_{\ell} \|B_{\ell}\|_{M^{-1}}^2) \left( \sum_{k=1}^{n_t} \|p_k - V\hat{p}_k\|_M^2 \right)
\leq (\max_{\ell} \|B_{\ell}\|_{M^{-1}}^2) \left( \sum_{k=1}^{n_t} 2\|\tilde{p}_k - V\hat{p}_k\|_M^2 + 2\|p_k - \tilde{p}_k\|_M^2 \right)
\leq (\max_{\ell} \|B_{\ell}\|_{M^{-1}}^2) \left[ 4C \left( \sum_{k=1}^{n_t} \|(I - VV^TM)p_k\|_M^2 \right) \\
+ (2\Delta t Tq + 4\Delta t TqKC^2) \left( \sum_{k=1}^{n_t} \|(I - VV^TM)w_k\|_M^2 \right) \right].
\]
Chapter 5

Semilinear Parabolic Model Order Reduction

Efficient implementation of reduced-order modeling depends on the ability to efficiently generate ROMs whose simulation cost is independent of the discretization. Chaturantabut and Sorensen (2010) [12] demonstrate with naive treatment of the nonlinearities, the ROMs can become more costly than the original it seeks to replace. Hence, the interpolation frameworks in EIM and DEIM introduced in Section 2.4 and Section 2.5 respectively become a critical step in construction of efficient reduced order models. This chapter explores a particular implementation of DEIM for finite elements, originally introduced as the Finite Element Method with Interpolated Coefficients (FEIC) in Wang (2015) [55].

This chapter differs from Wang (2015) in two significant ways. First, the FEIC method has an non-intrusive implementation that can be comfortably built on pre-existing finite element software. The ability to construct ROMs non-intrusively is the dominant advantage of this approach of its competition. Further, the approach generalizes to derivatives of the nonlinearity, which are needed in Hessian-vector multiplication in subsequent chapters of this document. Second, I develop an error analysis that accommodates a broader set of discretizations to include the piecewise linear
5.1 Model Problem

Consider a semilinear parabolic PDE with a nonlinear reaction term where for given function \( u : \Omega \times (0, T) \rightarrow \mathbb{R} \) the function \( y : \Omega \times [0, T] \rightarrow \mathbb{R} \) is the solution of

\[
\begin{align*}
y_t(x, t) - \mu \Delta y(x, t) + f(y(x, t)) &= u(x, t) \quad (x, t) \in \Omega \times (0, T), \\
y(x, t) &= 0 \quad (x, t) \in \partial \Omega \times (0, T), \\
y(x, 0) &= y_0(x) \quad x \in \Omega.
\end{align*}
\]

(5.1.1a)

for rectangular domain \( \Omega \subset \mathbb{R}^d, d \in \{1, 2, 3\} \), initial condition \( y_0 : \Omega \rightarrow \mathbb{R} \), and diffusion coefficient \( \mu > 0 \). Assume Lipschitz continuity of the nonlinearity,

\[
|f(y) - f(z)| \leq L|y - z|, \quad \forall y, z \in \mathbb{R}. \tag{5.1.2}
\]

5.1.1 Variational Form

The variational formulation seeks to find \( y(t) \) for all \( \varphi \in H^1_0(\Omega) \) such that

\[
\begin{align*}
\langle y_t(t), \varphi \rangle_{H^{-1}(\Omega), H^1_0(\Omega)} + a(y(t), \varphi) + \langle f(y(t)), \varphi \rangle_{H^{-1}(\Omega), H^1_0(\Omega)} &= \langle u(t), \varphi \rangle_{H^{-1}(\Omega), H^1_0(\Omega)}, \\
\langle y(0), \varphi \rangle_{L^2(\Omega)} &= \langle y_0, \varphi \rangle_{L^2(\Omega)}. \tag{5.1.3a}
\end{align*}
\]

where \( a(\cdot, \cdot) : H^1_0(\Omega) \times H^1_0(\Omega) \rightarrow \mathbb{R} \) is a bounded coercive bilinear operator defined

\[
a(v, w) = \mu \int_{\Omega} \nabla v(x) \cdot \nabla w(x) dx, \quad \forall v, w \in H^1_0(\Omega),
\]

and for fixed \( t \), the mapping \( x \mapsto y(x, t) \) is an element of \( H^1_0(\Omega) \). We look for a mapping \( t \mapsto y(t) \in H^1_0(\Omega) \) that satisfies the weak form of (5.1.3). Denote this element as \( y(t) \in H^1_0(\Omega) \) such that

\[
[y(t)](x) := y(x, t).
\]
Following Zeidler, the solution $y$ exists in the Sobolev space defined [58, Ch. 23],

$$W_2^1(0, T; H^1_0(\Omega), L^2(\Omega))$$

$$= \left\{ y : [0, T] \rightarrow H^1_0(\Omega) \middle| \int_0^T \|y(t)\|_{H^1_0(\Omega)}^2 + \|y_t(t)\|_{H^{-1}(\Omega)}^2 dt < \infty \right\}.$$ 

The weak derivative $y_t(t) : [0, T] \rightarrow H^{-1}(\Omega)$ is defined so that

$$\langle y_t(t), v \rangle_{H^{-1}(\Omega), H^1_0(\Omega)} = \frac{d}{dt} \langle u(t), v \rangle_{L^2(\Omega)} \quad \forall v \in H^1_0(\Omega),$$

where $\frac{d}{dt}$ denotes a weak derivative. Then we can rewrite (5.1.3),

$$\frac{d}{dt} \langle y(t), \varphi \rangle_{L^2(\Omega)} + a(y(t), \varphi) + \langle f(y(t)), \varphi \rangle_{H^{-1}(\Omega), H^1_0(\Omega)} = \langle u(t), \varphi \rangle_{H^{-1}(\Omega), H^1_0(\Omega)},$$

(5.1.4a)

$$\langle y(0), \varphi \rangle_{L^2(\Omega)} = \langle y_0, \varphi \rangle_{L^2(\Omega)}. \quad (5.1.4b)$$

Since $\frac{d}{dt}$ is a weak derivative, the variational form (5.1.4) needs to be satisfied almost everywhere in $(0, T)$. Using the assumption of Lipschitz continuity, (5.1.2), the following theorem proves existence and uniqueness of the variational form. The following theorem can be found in Evans (1998), Theorem 2, p. 500, [23].

**Theorem 5.1.** There exists a unique solution to (5.1.4).

### 5.2 Finite Element Method Discretization

Let $V^h$ denote the finite dimensional space of piecewise polynomials on the mesh $\mathcal{T}$,

$$V^h = \{ v \in H^1_0(\Omega) : \text{v}|_K \in \mathcal{P}_m(K) \quad \forall K \in \mathcal{T} \},$$

where $\mathcal{P}_m(\Omega)$ denotes the space of polynomial functions on $\Omega \subset \mathbb{R}^d$ at most degree $m$. Define the basis functions $\{ \phi_i \}_{i=1}^{n_v}$ to be piecewise polynomial Lagrange basis functions $\phi_i$ so that

$$V^h = \text{span}\{ \phi_1, \ldots, \phi_{n_v} \}, \quad \phi_i(x_j) = \delta_{ij}.$$
where \( \{x_j\} \) are the nodes of the mesh. We make the assumption (Thomée, Eq. (1.10), [50]) that the family of finite dimensional subspaces \( \{V^h\} \) satisfies for all \( v \in H^{m+1}(\Omega) \cap H^1_0(\Omega) \),

\[
\inf_{\varphi \in V^h} \{ \|v - \varphi\|_{L^2(\Omega)} + h\|\nabla(v - \varphi)\|_{L^2(\Omega)} \} \leq C h^{m+1} \|v\|_{m+1},
\]

(5.2.1)

where \( h \) denotes mesh diameter.

**Finite Element Variational Form**  
The finite element method is the following: find \( y^h \in H^1(0,T;V^h) \) such that for every \( \varphi \in V^h \) and \( t \in (0,T) \),

\[
\frac{d}{dt} \langle y^h(\cdot,t), \varphi \rangle_{L^2(\Omega)} - \mu \langle \nabla y^h(\cdot,t), \nabla \varphi \rangle_{L^2(\Omega)} + \langle f(y^h(\cdot,t)), \varphi \rangle_{L^2(\Omega)} = \langle u(t), \varphi \rangle_{L^2(\Omega)},
\]

(5.2.2a)

\[
\langle y^h(\cdot,0), \varphi \rangle_{L^2(\Omega)} = \langle y_0, \varphi \rangle_{L^2(\Omega)}.
\]

(5.2.2b)

### 5.2.1 Finite Element Error Estimates

Let \( X \) denote a real Banach space with norm \( \|\cdot\|_X \). The Sobolev space \( H^1(0,T;X) \) consists of all functions \( y \in L^2(0,T;X) \) such that \( y' \) exists in the weak sense and belongs to \( L^2(0,T;X) \). Furthermore,

\[
\|y\|_{H^1(0,T;X)} := \left( \int_0^T \|y(t)\|^2_X + \|y'(t)\|^2_X \, dt \right)^{1/2}.
\]

The following result from calculus is extended to the function space setting in Evans (1998), Theorem 2, p. 286 [23] where a proof is provided.

**Lemma 5.2.** Let \( y \in H^1(0,T;X) \). Then

\[
y(t) = y(0) + \int_0^t y'(s) \, ds, \quad \text{for all } 0 \leq t \leq T.
\]

Define the **elliptic projection** onto \( V^h \) with respect to \( \langle \nabla v, \nabla w \rangle \) so that \( R^h : H^1_0(\Omega) \to V^h \) satisfies,

\[
\langle \nabla R^h v, \nabla \varphi \rangle = \langle \nabla v, \nabla \varphi \rangle, \quad \text{for all } \varphi \in V^h \text{ and } v \in H^1_0(\Omega).
\]

(5.2.3)
The following lemma can be found in Thomée (2006) [50].

**Lemma 5.3.** Assume that (5.2.1) holds. Then with $R^h$ defined by (5.2.3) we have for $1 \leq s \leq m + 1$,

$$
\|R^h v - v\|_{L^2(\Omega)} + h \|\nabla (R^h v - v)\|_{L^2(\Omega)} \leq Ch^s \|v\|_{H^s(\Omega)}, \; \forall v \in H^s \cap H^1_0.
$$

**Linear Parabolic Case** The following theorem and proof follows Theorem 1.2 in Thomée [50].

**Theorem 5.4.** Assume a vanishing reaction term, $f \equiv 0$, and that the initial condition $y_0$ vanishes on the boundary $\partial \Omega$. Let $y$ and $y_h$ denote the solutions to (5.1.1) and (5.2.2) respectively. If $y$ satisfies the regularity assumption, $y \in H^1(0,T;H^{m+1}_0(\Omega))$, then the error can be bounded for $t \in (0,T)$,

$$
\|y_h(t) - y(t)\|_{L^2(\Omega)} \leq \|y_h(0) - y_0\|_{L^2(\Omega)} + Ch^{m+1} \left( \|y_0\|_{H^{m+1}(\Omega)} + \int_0^t \|y_t(s)\|_{H^{m+1}(\Omega)} ds \right).
$$

**Proof.** We apply the triangle inequality and bound each term separately,

$$
\|y_h(t) - y(t)\|_{L^2(\Omega)} \leq \|y_h(t) - R^h y(t)\|_{L^2(\Omega)} + \|R^h y(t) - y(t)\|_{L^2(\Omega)}.
$$

To bound the 2nd term, use Lemma 5.3,

$$
\|R^h y(t) - y(t)\|_{L^2(\Omega)} \leq Ch^{m+1} \|y(t)\|_{H^{m+1}(\Omega)}.
$$

Then via Lemma 5.2 and the triangle inequality,

$$
\|y(t)\|_{H^{m+1}(\Omega)} \leq \|y_0\|_{H^{m+1}(\Omega)} + \left\| \int_0^t y_t(s) ds \right\|_{H^{m+1}(\Omega)} \\
\leq \|y_0\|_{H^{m+1}(\Omega)} + \int_0^t \|y_t(s)\|_{H^{m+1}(\Omega)} ds.
$$

Then combining the inequalities we can bound the 2nd term in (5.2.4),

$$
\|R^h y(t) - y(t)\|_{L^2(\Omega)} \leq Ch^{m+1} \left( \|y_0\|_{H^{m+1}(\Omega)} + \int_0^t \|y_t\|_{H^{m+1}(\Omega)} ds \right).
$$

(5.2.5)
The rest of the proof seeks to bound the term, $\|y^h - R^h y\|_{L^2(\Omega)}$, in (5.2.4). From (5.2.2) and (5.2.3),

$$
\langle y_{h,t}(t) - R^h y_t(t), \phi \rangle_{L^2(\Omega)} + \langle \nabla y^h(t) - \nabla R^h y(t), \nabla \phi \rangle_{L^2(\Omega)}
$$

$$
= \langle y_{h,t}(t), \phi \rangle_{L^2(\Omega)} + \langle \nabla y^h(t), \nabla \phi \rangle_{L^2(\Omega)} - \langle R^h y_t(t), \phi \rangle_{L^2(\Omega)} - \langle \nabla R^h y(t), \nabla \phi \rangle_{L^2(\Omega)}
$$

$$
= \langle u, \phi \rangle_{L^2(\Omega)} - \langle R^h y_t(t), \phi \rangle_{L^2(\Omega)} - \langle \nabla y(t), \nabla \phi \rangle_{L^2(\Omega)}
$$

$$
= \langle y_t(t) - R^h y_t(t), \phi \rangle_{L^2(\Omega)}.
$$

By substituting $\phi = y^h(t) - R^h y(t)$, using $\|\nabla y^h(t) - \nabla R^h y(t)\|_{L^2(\Omega)} \geq 0$ and applying Cauchy-Schwarz to the right hand side,

$$
\langle y_{h,t}(t) - R^h y_t(t), y^h(t) - R^h y(t) \rangle_{L^2(\Omega)} \leq \|y_t - R^h y_t\|_{L^2(\Omega)} \|y^h(t) - R^h y(t)\|_{L^2(\Omega)}.
$$

Note that the left side of this inequality satisfies

$$
\langle y_{h,t}(t) - R^h y_t(t), y^h(t) - R^h y(t) \rangle_{L^2(\Omega)}
$$

$$
= \frac{1}{2} \frac{d}{dt} \|y^h(t) - R^h y(t)\|_{L^2(\Omega)}^2
$$

$$
= \|y^h(t) - R^h y(t)\|_{L^2(\Omega)} \frac{d}{dt} \|y^h(t) - R^h y(t)\|_{L^2(\Omega)}.
$$

Therefore,

$$
\frac{d}{dt} \|y^h(t) - R^h y(t)\|_{L^2(\Omega)} \leq \|y_t(t) - R^h y_t(t)\|_{L^2(\Omega)}.
$$

Then we can bound the 2nd term in (5.2.4),

$$
\|y^h(t) - R^h y(t)\|_{L^2(\Omega)} \leq \|y^h(0) - R^h y(0)\|_{L^2(\Omega)} + \int_0^t \frac{d}{ds} \|y_h(s) - R^h y(s)\|_{L^2(\Omega)} ds
$$

$$
\leq \|y^h(0) - R^h y(0)\|_{L^2(\Omega)} + \int_0^t \|y(s) - R^h y_t(s)\|_{L^2(\Omega)} ds
$$

We apply Lemma 5.3 to both terms in the right side of the inequality to get

$$
\|y^h(0) - R^h y_0\|_{L^2(\Omega)} \leq \|y^h(0) - y_0\|_{L^2(\Omega)} + \|y_0 - R^h y_0\|_{L^2(\Omega)}
$$

$$
\leq \|y^h(0) - y_0\|_{L^2(\Omega)} + Ch^{m+1} \|y_0\|_{H^{m+1}(\Omega)},
$$

$$
\|y_t(t) - R^h y_t(t)\|_{L^2(\Omega)} \leq Ch^{m+1} \|y_t(t)\|_{H^{m+1}(\Omega)}.
$$
Therefore the following inequality holds,

\[
\|y^h(t) - R^h y(t)\|_{L^2(\Omega)} \leq \|y^h(0) - y_0\|_{L^2(\Omega)} + C h^{m+1} \left( \|y_0\|_{H^{m+1}(\Omega)} + \int_0^t \|y_t(s)\|_{H^{m+1}(\Omega)} \right) ds.
\]

The inequalities (5.2.4), (5.2.5), and (5.2.6) give the desired result.

Semilinear Parabolic Case  The finite element solution error can be bounded by the following theorem from Thomée, [50, Theorem 14.1].

**Theorem 5.5.** Assume that (5.1.2) and (5.2.1) hold, and let \( y^h \) and \( y \) be the solutions of (5.2.2) and (5.1.1) respectively. Then if the initial condition \( y_0 \) is appropriately chosen and \( y \) sufficiently smooth, we have, with \( C = C(y,T) \),

\[
\|y^h(t) - y(t)\|_{L^2(\Omega)} + h \|y^h(t) - y(t)\|_{H^1(\Omega)} \leq C h^{m+1}, \quad \text{for } t \in [0,T].
\]

Note that choosing initial condition \( y_{h,0} \) to satisfy

\[
\|y_{h,0} - y(0)\| + h \|y_{h,0} - y(0)\|_{H^1(\Omega)} \leq C h^{m+1},
\]

is an "appropriately chosen" initial condition is sufficient to satisfy the previous lemma. Additionally, by Assumption (5.2.1), we know there exists such an initial condition whenever \( y(0) \in H^{m+1}(\Omega) \cap H_0^1(\Omega) \).

5.2.2 Matrix Representation of the Finite Element Method

Let \( \{\phi_1, \ldots, \phi_{n_y}\} \) be a basis of \( V^h \). The finite element approximation \( y^h \) in (5.2.2) can be written as

\[
y^h(x,t) = \sum_{i=1}^{n_y} y_i(t) \phi_i(x).
\]

Furthermore we let \( \{\phi_1, \ldots, \phi_{n_y}, \ldots, \phi_{n_u}\} \) be a basis of the discretized control space \( U^h \subset L^2(\Omega) \). Note that states in \( V^h \subset H_0^1(\Omega) \) are zero on the boundary.
Therefore \( \{\phi_1, \ldots, \phi_{ny}\} \subset \{\phi_1, \ldots, \phi_{ny}, \ldots, \phi_{nu}\} \). Controls \( u = u^h \) in (5.2.2) can be written as
\[
   u^h(x, t) = \sum_{i=1}^{nu} u_i(t) \phi_i(x). \tag{5.2.9}
\]

Inserting (5.2.8) and (5.2.9) into (5.2.2) and requiring that (5.2.2) holds for \( \phi = \phi_1, \ldots, \phi_{ny} \) leads to the matrix form
\[
   M \frac{dy(t)}{dt} + Ay(t) + N^{\text{int}}(y(t)) = Bu(t), \tag{5.2.10a}
\]
\[
   y(0) = y_0 \tag{5.2.10b}
\]
of (5.2.2), where
\[
   M_{ij} = \int_{\Omega} \phi_i(x) \phi_j(x) dx, \quad i, j = 1, \ldots, ny, \tag{5.2.11a}
\]
\[
   A_{ij} = \mu \int_{\Omega} \nabla \phi_i(x) \cdot \nabla \phi_j(x) dx, \quad i, j = 1, \ldots, ny, \tag{5.2.11b}
\]
\[
   B_{ij} = \int_{\Omega} \phi_i(x) \phi_j(x) dx, \quad i = 1, \ldots, ny, j = 1, \ldots, nu, \tag{5.2.11c}
\]
\[
   N^{\text{int}}_i(y) = \int_{\Omega} f \left( \sum_{j=1}^{ny} y_j \phi_j(x) \right) \phi_i(x) dx, \quad i = 1, \ldots, ny, \tag{5.2.11d}
\]
and
\[
   My_0 = \left( \int_{\Omega} y_0(x) \phi_1(x) dx, \ldots, \int_{\Omega} y_0(x) \phi_{ny}(x) dx \right)^T.
\]
Note that the \( L^2(\Omega) \) inner product leads to a weighted inner product on \( \mathbb{R}^{ny} \),
\[
   \langle v^h, u^h \rangle_{L^2(\Omega)} = \langle v, w \rangle_M.
\]

**Approximating Integrals via Quadrature**  In practice, we compute integrals via quadrature,
\[
   Q_\Omega[v] := \sum_{i=1}^{ny} w_i v(\bar{x}_i) \approx \int_{\Omega} v(x) dx.
\]
We choose the quadrature weights and nodes, \( w_i \) and \( \bar{x}_i \) respectively, so that \( Q_K[v] \) is exact for polynomials of some degree \( m \) on the triangle \( K \),
\[
   Q_K[v] := \int_{K} v(x) dx, \quad \forall v \in P_m(K).
\]
For example, we use a Gaussian quadrature scheme defined on a reference triangle that is exact for polynomials of degree 4. See Figure 5.1.

![Figure 5.1: Example of Gaussian quadrature points (grey dots) for exact quadrature of functions in $P_4(K)$](image)

Collating the quadrature points for all triangles $K \in \mathcal{T}$, we can define a quadrature scheme for the entire domain,

$$Q_\Omega[v] := \sum_{K \in \mathcal{T}} Q_K[v] = \sum_{j=1}^{n_q} w_j v(\tilde{x}_j).$$  \hfill (5.2.12)

Let $\{\phi_i\}_{i=1}^{n_y}$ form a piecewise polynomial Lagrange bases for the nodes $\{x_j\}_{j=1}^{n_y}$ for the finite element space $V^h = \text{span}\{\phi_1, \ldots, \phi_{n_y}\}$, satisfying $\phi_i(x_j) = \delta_{ij}$. We compute the stiffness and mass matrices via quadature,

$$A_{ij} := Q_\Omega[\mu \nabla \phi_i \cdot \nabla \phi_j] = \mu \sum_{k=1}^{n_q} w_k \nabla \phi_i(\tilde{x}_k) \cdot \nabla \phi_j(\tilde{x}_k),$$  \hfill (5.2.13a)

$$M_{ij} := Q_\Omega[\phi_i \phi_j] = \sum_{k=1}^{n_q} w_k \phi_i(\tilde{x}_k) \phi_j(\tilde{x}_k),$$  \hfill (5.2.13b)

$$B_{ij} := Q_\Omega[\phi_i \phi_j] = \sum_{k=1}^{n_q} w_k \phi_i(\tilde{x}_k) \phi_j(\tilde{x}_k).$$  \hfill (5.2.13c)

Note that for bases of degree $m$, if the quadrature scheme is exact for polynomials of degree $2m$, then the matrices (5.2.11a-5.2.11c) and (5.2.13) are identical. In the numerical examples of this thesis this is the case and therefore, we use the same notation in (5.2.11a-5.2.11c) and (5.2.13).
Applying quadrature to the integral in (5.2.11d) gives

\[ \mathbf{N}_i(y) = \mathcal{Q}_{\Omega}[f \circ y^h(\cdot)\phi_i] = \sum_{j=1}^{n_q} w_j f(y^h(\tilde{x}_j))\phi_i(\tilde{x}_j) = e_i^T \Phi \mathbf{W} f^q(y), \quad (5.2.14) \]

where \( e_i \) is the \( i \)-th unit vector,

\[ \Phi = \begin{bmatrix} \phi_1(\tilde{x}_1) & \cdots & \phi_1(\tilde{x}_{n_q}) \\ \vdots & \ddots & \vdots \\ \phi_{n_q}(\tilde{x}_1) & \cdots & \phi_{n_q}(\tilde{x}_{n_q}) \end{bmatrix}, \quad \mathbf{W} = \begin{bmatrix} w_1 \\ \vdots \\ w_{n_q} \end{bmatrix}, \]

and \( f^q(y) \) is a vector of the nonlinear function \( f \circ y^h \) evaluated at the quadrature points, \( \{\tilde{x}_i\}_{i=1}^{n_q} \).

Since \( x \mapsto f(y^h(x)) \) is not a piecewise polynomial, in general, the quadrature used in (5.2.14) is in general not exact, and therefore

\[ \mathbf{N}^{\text{int}}(y) \neq \mathbf{N}(y). \]

In practice we solve

\[ \mathbf{M} \frac{d}{dt} \mathbf{y}(t) + \mathbf{A} \mathbf{y}(t) + \mathbf{N}(\mathbf{y}(t)) = \mathbf{B} \mathbf{u}(t), \quad (5.2.15a) \]

\[ \mathbf{y}(0) = \mathbf{y}_0, \quad (5.2.15b) \]

instead of (5.2.10).

5.3 POD-ROM Approximation

5.3.1 Proper Orthogonal Decomposition

Proper Orthogonal Decomposition (POD) has already been described in Section 2.3 for the fully discrete problem. It is briefly restated for the finite element formulation (5.2.2).
Given the solution \( y^h \) of (5.2.2), POD computes a subspace \( V_r^h \) of \( V^h \) with basis \( \{ \psi_j \}_{j=1}^r \), \( r \ll n_y \), given as the solution of the following minimization problem,

\[
\min \int_0^T \left\| y^h(\cdot, t) - \sum_{j=1}^r \left( y^h(\cdot, t), \psi_j(\cdot) \right) \psi_j(\cdot) \right\|_{L^2(\Omega)}^2 \, dt, \tag{5.3.1}
\]

s.t. \( \{ \psi_j \}_{j=1}^r \subset V^h \),

\( (\psi_j, \psi_i) = \delta_{ij} \).

Using the basis \( V_r^h = \text{span} \{ \psi_j \}_{j=1}^r \subset V^h = \text{span} \{ \phi_i \}_{i=1}^{n_y} \), the representation (5.2.8) of the finite element solution, and associating \( \psi_j \) with a vector

\[
v_j = (v_{1j}, \ldots, v_{n_yj})^T \in \mathbb{R}^{n_y}
\]

defined through the representation \( \psi_j \) via the basis \( \{ \phi_i \}_{i=1}^{n_y} \),

\[
\psi_j = \sum_{i=1}^{n_y} v_{ij} \phi_i, \quad j = 1, \ldots, r,
\]

the minimization problem (5.3.1) can be written equivalently as

\[
\min \int_0^T \left\| y(t) - \sum_{j=1}^r y(t)^T M v_j \right\|_M^2 \, dt, \tag{5.3.3}
\]

s.t. \( v_j^T M v_i = \delta_{ij} \).

Once the POD subspace

\[
V_r^h := \text{span} \{ \psi_j \}_{j=1}^r \subset V^h
\]
is computed, the finite element problem (5.2.2) is approximated by

\[
\frac{d}{dt} (y_{\text{POD}}^h(t), v_r^h) + a(y_{\text{POD}}^h(t), v_r^h) + (f(y_{\text{POD}}^h(t)), v_r^h) = (u, v_r^h), \quad \forall v_r^h \in V_r^h, \tag{5.3.4a}
\]

\[
(y_{\text{POD}}^h(0), v_r^h) = (y_0, v_r^h), \quad \forall v_r^h \in V_r^h. \tag{5.3.4b}
\]

To relate the POD approximation

\[
y_{\text{POD}}^h(x, t) = \sum_{j=1}^r \hat{y}_j(t) \psi_j(x) \tag{5.3.5}
\]
to the finite element solution (5.2.8) we express the POD basis vectors $\psi_j$ in terms of
the basis vectors $\phi_i$. Since

$$V_h^r = \text{span} \{\psi_j\}_{j=1}^r \subset V^h = \text{span} \{\phi_i\}_{i=1}^{ny}$$

there exists scalars $v_{ij}$, $i = 1, \ldots, ny$, $j = 1, \ldots, r$, such that

$$\psi_j = \sum_{i=1}^{ny} v_{ij} \phi_i, \quad j = 1, \ldots, r. \tag{5.3.6}$$

Note that $v_j = (v_{1j}, \ldots, v_{n_yj})^T \in \mathbb{R}^{n_y}$ are the vectors computed from the matrix
representation (5.3.3) of (5.3.1) (see (5.3.2)). With (5.3.6) we can write

$$y_{POD}^h(x,t) = \sum_{j=1}^r \hat{y}_j(t) \psi_j(x) = \sum_{i=1}^{ny} \left( \sum_{j=1}^r v_{ij} \hat{y}_j(t) \right) \phi_i(x). \tag{5.3.7}$$

Requiring (5.3.4) is equivalent to requiring the identify in (5.3.4) to hold for
$v_r^h = \psi_1, \ldots, \psi_r$. Inserting (5.3.6) and (5.3.7) leads to following matrix representa-
tion of (5.3.4).

$$M \frac{d}{dt} \hat{y}(t) + V^T A V \hat{y}(t) + V^T N^{\text{int}}(V \hat{y}(t)) = V^T B u(t), \tag{5.3.8a}$$

$$\hat{y}(0) = V^T M y_0 \tag{5.3.8b}$$

with $A, B, N^{\text{int}}$ as in (5.2.11) and with matrix

$$V = (v_{ij}) \in \mathbb{R}^{n_y \times r}.$$  

Note that $(\psi_j, \psi_i) = \delta_{ij}$ implies

$$V^T MV = I.$$  

### 5.3.2 Proper Orthogonal Decomposition Error

The solution of (5.3.1) requires the operator

$$V^h \ni \psi \mapsto \int_0^T y^h(t) \langle y^h(t), \psi \rangle dt \in V^h.$$
Expressing functions in $V^h$ via the basis $\{\phi_i\}_{i=1}^{n_y}$ and using (5.2.8) reveals that the matrix representation of the above operator is

$$\int_0^T y(t) y(t)^T M dt \in \mathbb{R}^{n_y \times n_y}$$

which is the operator associated with the matrix representation (5.3.3) of the problem (5.3.1). The eigenvalues $\lambda$ and corresponding eigenvectors $z$ of the matrix are computed from

$$\int_0^T M y(t) y(t)^T M dt \cdot z = \lambda M z$$

(see also (2.3.2)). Since $\int_0^T M y(t) y(t)^T M dt$ is symmetric positive semidefinite and $M$ is symmetric positive definite the eigenvalues are real and non-negative. These eigenvalues are ordered

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq 0.$$ 

These eigenvalues describe the approximation properties of POD, see, e.g., Hinze and Volkwein (2005) [31], and Section 2.3.

The next result is taken from Singler (2014) [47]. The first estimate in the $L^2$ norm is standard, see, e.g., Hinze and Volkwein (2005) [31].

**Lemma 5.6.** Let $w^h_r$ be the projection of $y^h \in V^h$ onto $V^h_r$ so that

$$(y^h - w^h_r, v^h_r) = 0 \quad \forall v^h_r \in V^h_r.$$ 

Then $w^h_r$ satisfies the following error:

$$\int_0^T \|y^h - w^h_r\|_{L^2(\Omega)}^2 dt = \sum_{j>r} \lambda_j,$$

$$\int_0^T \|y^h - w^h_r\|_{H^1(\Omega)}^2 dt = \sum_{j>r} \|\psi_j\|_{H^1}^2 \lambda_j.$$ 

I use Lemma 5.6 to bound the POD error from the FEM solution $y^h - y^h_{\text{POD}}$. The following theorem is an adaptation of the error analysis found in Wang (2015) [55] assuming Lipschitz continuity instead of local Lipschitz continuity. A similar analysis with Lipschitz continuous nonlinearities can be found in Kunisch and Volkwein (2001) [36].
Theorem 5.7. Assume the nonlinear reaction $f$ is Lipschitz continuous. Then the error between the finite element solution $y^h$ to (5.2.2) and the POD solution $y_{\text{POD}}^h$ to (5.3.4) obeys

$$
\int_0^T \| y^h(t) - y_{\text{POD}}^h(t) \|^2 dt \leq C \left( \sum_{j>r} \lambda_j + \| \psi_j \|^2_{H^1(\Omega)} \lambda_j \right).
$$

Proof. Let $y^h$ and $y_{\text{POD}}^h$ denote the solutions to (5.2.2) and (5.3.4) respectively, and let $w_r^h$ be the projection of $y^h \in V^h$ onto $V_r^h$ (see Lemma (5.6)). Define the error between the solutions,

$$
e = y^h - y_{\text{POD}}^h = y^h - w_r^h + w_r^h - y_{\text{POD}}^h.
$$

Taking the difference of the two equations, (5.2.2) and (5.3.4) we get

$$
\left( \frac{\partial e}{\partial t}, v_r^h \right) + a(e, v_r^h) + (f(y^h) - f(y_{\text{POD}}^h), v_r^h) = 0.
$$

Since $w_r^h$ is the projection of $y^h$ onto $V_r^h$, then it follows that $\left( \frac{\partial}{\partial t} [y^h - w_r^h], v_r^h \right) = 0$ for all $v_r^h \in V_r^h$. Therefore

$$
\left( \frac{\partial}{\partial t} [w_r^h - y_{\text{POD}}^h], v_r^h \right) + a(w_r^h - y_{\text{POD}}^h, v_r^h) = a(w_r^h - y^h, v_r^h) + (f(y_{\text{POD}}^h) - f(y^h), v_r^h).
$$

Define $\alpha > 0$ and $\beta > 0$ to be the boundedness and coercivity constants of the bilinear form respectively,

$$
\beta \| \nabla v \|^2 \leq a(v, v), \quad a(v, w) \leq \alpha \| \nabla v \| \| \nabla w \|, \quad \forall v, w \in H^1_0(\Omega).
$$

Set $v_r^h = w_r^h - y_{\text{POD}}^h$. Then

$$
\frac{1}{2} \frac{d}{dt} \| w_r^h - y_{\text{POD}}^h \|^2 + \beta \| \nabla [w_r^h - y_{\text{POD}}^h] \|^2 \leq \alpha \| \nabla [w_r^h - y^h] \| \| \nabla [w_r^h - y_{\text{POD}}^h] \| + \| f(y_{\text{POD}}^h) - f(y^h) \| \| w_r^h - y_{\text{POD}}^h \|.
$$

By Young’s inequality $\left( \epsilon = \frac{\alpha}{2\beta} \right)$,

$$
\alpha \| \nabla [w_r^h - y^h] \| \| \nabla [w_r^h - y_{\text{POD}}^h] \| \leq \beta \| \nabla [w_r^h - y_{\text{POD}}^h] \|^2 + \frac{\alpha^2}{4\beta} \| \nabla [w_r^h - y^h] \|^2.
$$
Then utilizing the Lipschitz continuity, \( \| f(y^h_{\text{POD}}) - f(y^h) \| \leq L \| y^h_{\text{POD}} - y^h \| \), we get
\[
\frac{d}{dt} \| w^h - y^h_{\text{POD}} \|^2 \leq \frac{\alpha^2}{2\beta} \| \nabla [w^h - y^h] \|^2 + 2L \| y^h_{\text{POD}} - y^h \| \| w^h - y^h_{\text{POD}} \|
\]
\[
\leq \frac{\alpha^2}{2\beta} \| \nabla [w^h - y^h] \|^2 + 2L \| w^h - y^h \| \| w^h - y^h_{\text{POD}} \| + 2L \| w^h - y^h_{\text{POD}} \|^2
\]
\[
\leq \frac{\alpha^2}{2\beta} \| \nabla [w^h - y^h] \|^2 + L \| w^h - y^h \|^2 + 3L \| w^h - y^h_{\text{POD}} \|^2.
\]

By Gronwall’s inequality and Lemma 5.6,
\[
\| w^h(t) - y^h_{\text{POD}(t)} \|^2 \leq C \int_0^T \| \nabla [w^h(s) - y^h(s)] \|^2 + \| w^h(s) - y^h(s) \|^2 ds
\]
\[
\leq C \left( \sum_{j>r} \lambda_j \| \psi_j \|^2_{H^1(\Omega)} \lambda_j \right).
\]

The previous inequality and Lemma 5.6 give the desired result.

\[ \square \]

### 5.4 The DEIM for Finite Elements

While the reduced order model (5.3.8) is of size \( r \) and significantly smaller than the finite element problem (5.2.11), which is of size \( n_y \), the evaluation of the nonlinearity
\[ \hat{y} \mapsto V^T N^{\text{int}}(V\hat{y}) \]

is expensive and an additional approximation is needed.

To describe what is done in the literature and to introduce my approach, I replace \( N^{\text{int}} \) by the nonlinear term,
\[ N(y) = \Phi W f^q(\Phi^T y) = \sum_{j=1}^{n_q} w_j f(y^h(\bar{x}_j)) \phi_i(\bar{x}_j). \quad (5.4.1) \]

See the discussion in Section 5.2.2. Although \( N^{\text{int}} \neq N \) in general, my approach will relate back to (5.3.8) and (5.3.4).

Again, the issue is that the evaluation of the nonlinearity
\[ \hat{y} \mapsto V^T N(V\hat{y}) \]
is expensive as it requires evaluation of all the quadrature points over the domain to evaluate. Where $V^TAV, V^TB$ can be computed once, for time-dependent nonlinear problems, the number of evaluations of $\tilde{y} \mapsto V^T(\nabla \tilde{y})$ can be enormous as the nonlinearity needs to be computed repeatedly throughout the simulation as $\tilde{y}(t)$ changes. Hence, we desire an interpolation method EIM / DEIM to approximate the nonlinearity in (5.3.8) or, equivalently, (5.3.4).

### 5.4.1 DEIM Applied Directly to Nonlinearity

As $N(y)$ is vector-valued, one could apply the DEIM oblique projection to the vector directly,

$$N(y) \mapsto U(P^TU)^{-1}P^TN(y). \quad (5.4.2)$$

Chaturantabut and Sorensen [12, 14] develop this for a finite difference discretization where the nonlinearity evaluated element-wise ($N_i(y)$ depends on $y_i$ only). However for the finite element method, $N_i(y)$ depends not only on $y_i$ but also the elements nearby to evaluate the FEM solution $y^h$ on support of $\phi_i$. See Figure 5.2.

Application of DEIM directly to the nonlinearity is inefficient as it requires many evaluations at quadrature points for each component of the nonlinearity evaluated.

### 5.4.2 Unassembled DEIM

Tiso et al. (2012) [51] propose disassembling the mesh. This approach is also discussed in Antil et al. [4]. The nonlinearity expressed in its unassembled form, evaluates integration of each basis element $\phi_j$ and element $e_i$ individually and combines them with a large matrix, $Q \in \mathbb{R}^{n_y \times n_e n_p}$,

$$N(y) = Qf^e(y), \quad f^e(y) = [f^{e1}(y)^T, \ldots, f^{en_e}(y)^T]^T,$$

where $f^{e_i}(y) \in \mathbb{R}^{n_p}$ denotes the nonlinearity evaluated on the nodes of each element independently. Using the quadrature rule in equation (5.2.12), we can approximate
Figure 5.2: Piecewise linear nodes in black+red, quadrature points in grey (exact for $\mathcal{P}_4(\mathcal{K})$). Evaluation of the red dot $N_i(y)$ in this case requires 7 terms of $y$ (neighboring black dots) and 36 function evaluation of $f \circ y^h$.

the nonlinear inner product,

\[ f^e_k(y) = Q_{e_i} \left[ f \left( \sum_{j=1}^{n_y} y_j \phi_j \right) \phi_k \right], \quad k = 1, \ldots, n_p. \]

where $Q_{e_i}$ denotes the quadrature scheme on the triangular element $e_i$. Tiso et al. apply DEIM to the unassembled nonlinearity,

\[ QU(P^T U)^{-1} P^T f^e(y). \]

This approach is more efficient than the former method of applying the interpolation to the nonlinearity itself in equation (5.4.2) as measured in accuracy of the nonlinearity for computational cost. However, this approach is an intrusive method to implement for preexisting FEM code bases such at PETSc, Trilinos, etc. where the quadrature schemes used may not be immediately available to the user, nor would modifying the code to evaluate the quadrature at only a few points be convenient. Instead, we propose a nonintrusive version of DEIM based on a forming piecewise polynomial approximations of $f \circ y^h$ on the triangles $\{K\}_{K \in \mathcal{T}}$. 
5.4.3 A nonintrusive EIM+FEM precomputation

Recall that $V^h = \text{span}\{\phi_j\}_{j=1}^{n_y}$ forms a Lagrange basis so that $\phi_i(x_j) = \delta_{ij}$. Define the piecewise polynomial Lagrange interpolant $I^h[v] : \mathcal{C} \to V^h$ so that

$$I^h[v](x) = \sum_{\ell=1}^{n_y} v(x_\ell) \phi_\ell(x), \quad I^h[v](x_i) = v(x_i). \quad (5.4.3)$$

Recall the definition of the nonlinearity, (5.4.1), where

$$N_i(y) = Q_\Omega[\phi_i(f \circ y^h)] = \sum_{j=1}^{n_q} w_j \phi_i(\tilde{x}_j) f(y^h(\tilde{x}_j)),\]$$

which requires knowledge of the quadrature scheme $\{\tilde{x}_j\}_{j=1}^{n_q}$.

We introduce an approximation by replacing $f \circ y^h$ with its piecewise polynomial Lagrange interpolant $I^h[f \circ y^h],$

$$\tilde{N}_i(y) := Q_\Omega[I^h[f \circ y^h]] \quad (5.4.4)$$

$$= \sum_{j=1}^{n_q} \phi_i(\tilde{x}_j) I^h[f \circ y^h](\tilde{x}_j)$$

$$= \sum_{j=1}^{n_q} \phi_i(\tilde{x}_j) \sum_{\ell=1}^{n_y} f(y^h(x_\ell)) \phi_\ell(\tilde{x}_j)$$

$$= \sum_{\ell=1}^{n_y} f(y^h(x_\ell)) \sum_{j=1}^{n_q} \phi_i(\tilde{x}_j) \phi_\ell(\tilde{x}_j)$$

$$= \sum_{\ell=1}^{n_y} f(y_\ell)[\Phi W \Phi^T]_{i\ell}. \quad (5.4.5)$$

Note that by (5.2.13b),

$$M = \Phi W \Phi^T.$$

Thus, in matrix form, we can write the approximated nonlinearity with the mass matrix (5.2.13) defined above,

$$\tilde{N}(y) := M f^n(y) = M \begin{bmatrix} f(y^h(x_1)) \\ \vdots \\ f(y^h(x_{n_y})) \end{bmatrix} = M \begin{bmatrix} f(y_1) \\ \vdots \\ f(y_{n_y}) \end{bmatrix}. \quad (5.4.5)$$
Replacing the nonlinear function \( f \circ y^h \) with its piecewise linear approximation defeats the purpose of the evaluating the nonlinearity at the quadrature points. That is, the high degree of accuracy in the quadrature scheme is wasted. The tradeoff of this approach is that the approximation can be done non-intrusively without prior knowledge of quadrature scheme or even the discretization used. Ultimately, this approach can be understood as an special implementation of EIM, where the interpolation points search is restricted to the nodes of the discretization.

**EIM Approximation.** Evaluation of the nonlinearity \( \tilde{N}(y) \) still requires \( n_y \) evaluations of the nonlinear function \( f \circ y^h \). EIM allows us to evaluate a few and interpolate the rest. We introduce EIM on the vector-valued function evaluation,

\[
\tilde{f}^n(y) := U(P^T U)^{-1} P^T f^n(y) \approx f^n(y).
\]

Then we introduce a further approximation to the nonlinearity \( N(y) \),

\[
\hat{N}(y) := M \tilde{f}^n(y) = MU(P^T U)^{-1} P^T f^n(y)
\]

The construction of the DEIM subspaces \( U \) and \( BP \) are handled via POD on the nonlinear history: \( U = POD[f(y_1), \ldots, f(y_{n_t})] \) and \( P = DEIM(U) \). This is not a significant additional cost in the ROM construction as the state history \( \{y_1, \ldots, y_{n_t}\} \) is already needed for the POD computation and the nonlinearities \( f(y) \) are cheap to evaluate as they can be done element-wise on the mesh nodes.

**Variational Form of \( \tilde{N} \).** The approximation \( \tilde{N}(y) \) (5.4.5) is identical to (5.4.4). If the quadrature is exact for polynomials of degree \( 2m \), then because \( \mathcal{I}^h[f \circ y^h] \) is a polynomials of degree \( m \) and \( \phi_i \) is a polynomial of degree \( m \) on each triangle,

\[
\tilde{N}_i(y) = Q[\phi_i \mathcal{I}^h[f \circ y^h]] = \int_{\Omega} \phi_i(x) \mathcal{I}^h[f \circ y^h](x) \, dx.
\] (5.4.6)
Thus, if we replace $V^T N_{\text{int}}(V\tilde{y}(t))$ in (5.3.8) by $V^T \tilde{N}(V\tilde{y}(t))$ we arrive at

$$M\frac{d}{dt}\tilde{y}(t) + V^T AV\tilde{y}(t) + V^T \tilde{N}(V\tilde{y}(t)) = V^T Bu(t),$$  \hspace{1cm} (5.4.7a)

$$\tilde{y}(0) = V^T My_0$$  \hspace{1cm} (5.4.7b)

The system (5.3.8) is the matrix representation of the variational form (5.3.4). Equation (5.4.6) shows that the system (5.4.7) is the matrix representation of the variational form

$$\left(\frac{\partial y^h_r}{\partial t}, v^h_r\right) + a(y^h_r, v^h_r) + (I^h f(y^h_r), v^h_r) = (u, v^h_r) \quad \forall v^h_r \in V^h_r;$$  \hspace{1cm} (5.4.8a)

$$(y^h_r(0), v^h_r) = (y_0, v^h_r), \quad \forall v^h_r \in V^h_r.$$  \hspace{1cm} (5.4.8b)

As mentioned before, (5.4.7) is not yet efficient, since the cost of $\tilde{y} \mapsto V^T \tilde{N}(V\tilde{y})$ still depends on the size of the full order model. However, (5.4.7) is now in a form to which the DEIM approximation can be applied efficiently and non-intrusively.

The next section analyses the error due to replacing (5.3.4) by (5.4.8).

### 5.5 Error Due to Piecewise Polynomial Interpolation of Nonlinearity

This section analyzes the error due to replacing the nonlinearity in the POD model (5.3.4) by its piecewise polynomial interpolation which leads to

$$\left(\frac{\partial y^h_r}{\partial t}, v^h_r\right) + a(y^h_r, v^h_r) + (I^h f(y^h_r), v^h_r) = (u, v^h_r) \quad \forall v^h_r \in V^h_r;$$  \hspace{1cm} (5.5.1a)

$$(y^h_r(0), v^h_r) = (y_0, v^h_r), \quad \forall v^h_r \in V^h_r.$$  \hspace{1cm} (5.5.1b)

To perform this error analysis a few technical results are needed, which are stated next.

Recall the Lagrange interpolant $I^h : L^2(\Omega) \to V^h$ defined in (5.4.3),

$$I^h[v](x) = \sum_{i=1}^{n_\varphi} v(x_i) \varphi_i(x).$$
Theorem 5.8. Let $\{\hat{K}, \hat{P}, \hat{\Sigma}\}$ be a finite element with associated normed vector space $V(\hat{K})$. Let $1 \leq p \leq \infty$ and assume that there exists an integer $m$ such that

$$\mathbb{P}_m \subset \hat{P} \subset W^{m+1,p}(\hat{K}) \subset V(\hat{K}).$$

Let $T_K : \hat{K} \rightarrow K$ be an affine bijective mapping and let $I^m_K$ be the interpolation operator on $K$. Let $l$ be such that $0 \leq l \leq m$ and $W^{l+1,p}(\hat{K}) \subset V(\hat{K})$ with continuous embedding. Then setting $\sigma_K = \frac{h_K}{p_K}$, there exists $c > 0$ such that, for all $k \in \{0, \ldots, l+1\}$,

$$\forall K, \forall v \in W^{l+1,p}(K), \quad |v - I^m_K v|_{k,p,K} \leq c h^{l+1-k} \sigma^m_K |v|_{l+1,p,K}.$$  

The next result is from Ern and Guermond (2004) (Cor. 1.109), p.61, [22].

Theorem 5.9. Let $p$, $m$ and $l$ satisfy the assumptions of Theorem 5.8. Let $\Omega$ be a polyhedron and let $\{T^h\}_{h>0}$ be a shape-regular family of affine meshes of $\Omega$. Denote by $V^h$ the approximation space based on $T^h$ and $\{\hat{K}, \hat{P}, \hat{\Sigma}\}$. Let $I^h$ be the corresponding global interpolation operator. Then, there exists $c$ such that, for all $h$ and $v \in W^{l+1,p}(\Omega)$,

$$\|v - I^h v\|_{L^p(\Omega)} + \sum_{k=1}^{l+1} h^k \left( \sum_{K \in T^h} |v - I^h v|_{k,p,K}^p \right)^{1/p} \leq c h^{l+1} |v|_{l+1,p,\Omega},$$

for $p < \infty$, and for $p = \infty$,

$$\|v - I^h v\|_{L^\infty(\Omega)} + \sum_{k=1}^{l+1} h^k \max_{K \in T^h} |v - I^h v|_{k,\infty,K} \leq c h^{l+1} |v|_{l+1,\infty,\Omega}.$$  

Furthermore, for $p < \infty$ and $v \in L^p(\Omega)$, the following density result holds:

$$\lim_{h \rightarrow 0} \left( \inf_{v^h \in V^h} \|v - v^h\|_{L^p(\Omega)} \right) = 0.$$  

Wang introduces a new norm called the auxiliary ”Euclidean” norm defined on $C(\Omega)$ [55]:

$$\|\chi\|_h = \left[ \sum_{i=1}^{n_{dof}} |\chi(x_i)|^2 \right]^{1/2}.$$  

The following result can be found in Wang (2015) [55, L. 4.3]
Lemma 5.10. There exist two strictly positive constants $c_1$ and $c_2$ independent of $h$ such that

$$c_1 h^{d/2} \|\chi\| \leq \|\chi\| \leq c_2 h^{d/2} \|\chi\|_h,$$

for all $\chi \in V^h$.

Theorem 5.11. Let $\Omega$ be a Lipschitz domain, let the integer $m \geq 1$ and let the solution $y$ of (5.1.3) and the nonlinearity $f$ satisfy

(A1) $y \in C(0, T; H^{m+1}(\Omega) \cap H^1_0(\Omega))$,

(A2) $f(y) \in C(0, T; H^{m+1}(\Omega))$,

(A3) $|f(\xi) - f(\eta)| \leq L|\xi - \eta|$ for all $\xi, \eta \in \mathbb{R}$.

Then there exists a constant $C > 0$ such that the error between the solution $y^h_{\text{POD}}$ of the POD model (5.3.4) and the solution $y^h_r$ of (5.5.1) satisfies

$$\|y^h_{\text{POD}}(t) - y^h_r(t)\|^2 + \int_0^T \|\nabla[y^h_{\text{POD}}(t) - y^h_r(t)]\|^2 dt$$

$$\leq c_4 \int_0^T \|y^h_{\text{POD}}(t) - y(t)\|^2 dt + c_4 h^{2m+2} \int_0^T |f(y(t))|_{H^{m+1}(\Omega)}^2 dt. \quad (5.5.2)$$

Proof. Define $\alpha > 0$ and $\beta > 0$ to be the boundedness and coercivity constants of the bilinear form respectively,

$$\beta \|\nabla v\|^2 \leq a(v, v), \quad a(v, w) \leq \alpha \|\nabla v\| \|\nabla w\|, \quad \forall v, w \in H^1_0(\Omega).$$

Subtracting (5.3.4) and (5.5.1) gives

$$\left(\frac{\partial}{\partial t}(y^h_{\text{POD}}(t) - y^h_r(t), v^h_r) + a(y^h_{\text{POD}}(t) - y^h_r(t), v^h_r)$$

$$+ (f(y^h_{\text{POD}}(t)) - I^h f(y^h_r(t)), v^h_r) = 0. \quad (5.5.3)$$

Setting $v^h_r = y^h_{\text{POD}}(t) - y^h_r(t)$ in (5.5.3) gives

$$\frac{1}{2} \frac{d}{dt} \|y^h_{\text{POD}}(t) - y^h_r(t)\|^2 + \beta \|\nabla[y^h_{\text{POD}}(t) - y^h_r(t)]\|^2$$

$$\leq \|f(y^h_{\text{POD}}(t)) - I^h f(y^h_r(t))\| \|y^h_{\text{POD}}(t) - y^h_r(t)\|$$

$$\leq \frac{1}{2} \|f(y^h_{\text{POD}}(t)) - I^h f(y^h_r(t))\|^2 + \frac{1}{2} \|y^h_{\text{POD}}(t) - y^h_r(t)\|^2. \quad (5.5.4)$$
Finally, by Lipschitz continuity

\[ \| f(y^h_{\text{POD}}(t)) - I^h f(y^h_{r}(t)) \| \]
\[ \leq \| I^h f(y^h_{\text{POD}}(t)) - I^h f(y^h_{r}(t)) \| + \| I^h f(y(t)) - f(y^h_{\text{POD}}(t)) \| + \| f(y(t)) - f(y^h_{r}(t)) \|. \]  \tag{5.5.5}

Utilizing the Lipschitz continuity and norm equivalency (Lemma 5.10),

\[ \| I^h f(y^h_{\text{POD}}(t)) - I^h f(y^h_{r}(t)) \| \leq c_2 h^{d/2} \| f(y^h_{\text{POD}}(t)) - f(y^h_{r}(t)) \|_h \]
\[ \leq L c_2 h^{d/2} \| y^h_{\text{POD}}(t) - y^h_{r}(t) \|_h \]
\[ \leq L c_2 c_1^{-1} \| y^h_{\text{POD}}(t) - y^h_{r}(t) \|. \]  \tag{5.5.6}

Similarly (note that \( y(t) \in H^{m+1}(\Omega) \subset C(\overline{\Omega}) \) by (A1)),

\[ \| I^h f(y^h_{\text{POD}}(t)) - I^h f(y(t)) \| \leq L c_2 c_1^{-1} \| y^h_{\text{POD}}(t) - y(t) \|. \]  \tag{5.5.7}

Using Theorem 5.9 with \( p = 2 \) and \( l = m \) we can bound

\[ \| I^h f(y(t)) - f(y(t)) \| \leq C h^{m+1} | f(y(t)) |_{H^{m+1}(\Omega)}. \]  \tag{5.5.8}

Finally, by Lipschitz continuity

\[ \| f(y(t)) - f(y^h_{\text{POD}}(t)) \| \leq L \| y^h_{\text{POD}}(t) - y(t) \|. \]  \tag{5.5.9}

Inserting (5.5.5)-(5.5.9) into (5.5.4) gives the existence of a constant \( c_3 > 0 \) such that

\[ \frac{d}{dt} \| y^h_{\text{POD}}(t) - y^h_{r}(t) \|^2 + 2 \beta \| \nabla [ y^h_{\text{POD}}(t) - y^h_{r}(t) ] \|^2 \]
\[ \leq c_3 \| y^h_{\text{POD}}(t) - y^h_{r}(t) \|^2 + c_3 \| y^h_{\text{POD}}(t) - y(t) \|^2 + c_3 h^{2m+2} | f(y(t)) |_{H^{m+1}(\Omega)}^2. \]  \tag{5.5.10}

Application of Gronwall's inequality to (5.5.10) gives the existence of a constant \( c_4 > 0 \) such that

\[ \| y^h_{\text{POD}}(t) - y^h_{r}(t) \|^2 + \int_0^T \| \nabla [ y^h_{\text{POD}}(t) - y^h_{r}(t) ] \|^2 \, dt \]
\[ \leq c_4 \int_0^T \| y^h_{\text{POD}}(t) - y(t) \|^2 \, dt + c_4 h^{2m+2} \int_0^T | f(y(t)) |_{H^{m+1}(\Omega)}^2 \, dt, \]

which is the desired inequality. \( \square \)
**Remark 5.12.** The error bound (5.5.2) depends on \( \int_0^T \| y_{\text{POD}}(t) - y(t) \|^2 dt \). However, this quantity can be estimated using Theorems 5.5 and 5.7 as follows

\[
\int_0^T \| y_{\text{POD}}(t) - y(t) \|^2 dt \leq C h^{2m+2} + C \left( \sum_{j>r} \lambda_j + \| \psi_j \|^2_{H^1(\Omega)} \right).
\]

**5.6 Example Problem - Solid Fuel Ignition Model**

I consider the diffusion-nonlinear reaction problem with linear diffusion and nonlinear exponential reaction from Ito et al. (2002) and Borzi et al. (2000) [9, 32].

\[
y_t(x, t) - \mu \Delta y(x, t) - \delta e^{y(x,t)} = u(x, t) \quad (x, t) \in \Omega \times (0, T), \tag{5.6.1a}
y(x, t) = 0 \quad (x, t) \in \partial \Omega \times (0, T), \tag{5.6.1b}
y(x, 0) = y_0(x) \quad x \in \Omega.
\]

where the domain is rectangular \( \Omega = (0, 1)^2 \) with initial state \( y_0(x, t) \equiv 0 \), diffusion coefficient \( \mu = 1 \) and reaction term coefficient \( \delta = 5 \) are given.

Results in this section are run on a uniform triangular mesh for two discretization, \( n_x = n_y = 20 \) or \( n_x = n_y = 40 \), a consistent temporal discretization for both, \( n_t = 50 \), and piecewise linear finite elements bases resulting in 1681 and 1521 degrees of freedom in the control and state respectively due to the Dirichlet boundary conditions. In the following results, I compare the error introduced by five model reduction frameworks from the full-order, expensive computation as follows

0. **FOM (Full Order Model)**

\[
M \frac{Y_{k+1} - Y_k}{\Delta t} + Ay_{k+1} + N(y_{k+1}) = Bu_{k+1}, \quad y_0 \text{ given} \tag{5.6.2}
\]

1. **POD Only (Proper Orthogonal Decomposition in State Only)**

\[
\frac{\tilde{Y}_{k+1} - \tilde{Y}_k}{\Delta t} + V^T AV \tilde{Y}_{k+1} + V^T N(V \tilde{Y}_{k+1}) = V^T Bu_{k+1}
\]

\[
V^T MV = I_r, \quad y_k \approx V \tilde{Y}_k, \quad \tilde{y}_0 = V^T My_0
\]
2. **POD+DEIM** (Proper Orthogonal Decomposition in State, Discrete Empirical Interpolation Method in Nonlinearity)

\[
\frac{\tilde{y}_{k+1} - \tilde{y}_k}{\Delta t} + V^T A V \tilde{y}_{k+1} + V^T \bar{U} (\bar{P}^T \bar{U})^{-1} \bar{P}^T N(V \tilde{y}_{k+1}) = V^T B u_{k+1}
\]

\[V^T M V = I_r, \quad y_k \approx V \tilde{y}_k, \quad \tilde{y}_0 = V^T M y_0\]

3. **POD+DEIM+PWL** (Proper Orthogonal Decomposition in State, Discrete Empirical Interpolation Method in Nonlinearity, Piecewise Linear Lagrange Interpolation)

\[
\frac{\tilde{y}_{k+1} - \tilde{y}_k}{\Delta t} + V^T A V \tilde{y}_{k+1} + V^T M U (P^T U)^{-1} P^T f^n(y_{k+1}) = V^T B u_{k+1}
\]

\[V^T M V = I_r, \quad y_k \approx V \tilde{y}_k, \quad \tilde{y}_0 = V^T M y_0\]

4. **PWL Only** (Piecewise Linear Lagrange Interpolation)

\[
M \frac{y_{k+1} - y_k}{\Delta t} + A y_{k+1} + M f^n(y_{k+1}) = B u_{k+1}, \quad y_0 \text{ given}
\]

Error calculations use one of two metrics for distance,

\[
\| \tilde{y} \|_{L^2(0,T;\mathbb{R}^n(M))} := \sqrt{\sum_{k=1}^{n_t} \Delta t \| y_k \|_M^2}, \quad (5.6.3)
\]

\[
\| N(\tilde{y}) \|_{L^2(0,T;\mathbb{R}^n(M^{-1}))} := \sqrt{\sum_{k=1}^{n_t} \Delta t \| N(y_k) \|_{M^{-1}}^2}. \quad (5.6.4)
\]

The first norm (5.6.3) matches the \(L^2(0, T; L^2(\Omega))\) when \(y_h(t)\) is piecewise constant in time and a subset of the Finite Element Space. The second norm (5.6.4), however, requires more explanation as the \(M^{-1}\) weighting is the appropriate choice to measure distance in the nonlinearity \(N(y)\). To see this, consider the change of basis, \(\tilde{y}_k := M^{1/2} y_k\) for \(k = 0, \ldots, n_t\). Under this transformation, the Euclidean 2-norm of \(\tilde{y}_k\) is the correct measure of distance that corresponds to the \(L^2(0,T;\Omega)\) norm after the FEM change of basis. Then the full order model (5.6.2) after substitution is,

\[
\frac{\tilde{y}_{k+1} - \tilde{y}_k}{\Delta t} + M^{-1/2} A M^{-1/2} \tilde{y}_{k+1} + M^{-1/2} N(M^{-1/2} \tilde{y}_{k+1}) = M^{-1/2} B u_{k+1}.
\]
While the simulation is not run in this change of basis, it shows that the $M^{-1}$-weighted norm is a reasonable choice to measure distance between the different methods of calculating the nonlinearities (e.g. PWL, DEIM, POD, etc.).

![Error in State, Coarse Mesh](image)

![Error in Nonlinearity, $N(y)$, Coarse Mesh](image)

Figure 5.3: Comparison of ROM treatment of nonlinearity $N(y)$ as measured by error from the FOM simulation (top figure) and error in the nonlinearity separately (bottom figure). While the PWL approximation is inferior in terms of accuracy, its ease of implementation on pre-existing FEM software is its advantage. Results are generated on a coarse mesh: $n_x = n_y = 20, n_t = 50$. 

<table>
<thead>
<tr>
<th>POD Tolerance</th>
<th>POD Only</th>
<th>POD+DEIM(1e-3)</th>
<th>POD+DEIM(1e-4)</th>
<th>POD+DEIM(1e-4)+PWL</th>
<th>PWL Only</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-8}$</td>
<td>$10^{-3}$</td>
<td>$10^{-4}$</td>
<td>$10^{-5}$</td>
<td>$10^{-6}$</td>
<td></td>
</tr>
<tr>
<td>$10^{-7}$</td>
<td>$10^{-3}$</td>
<td>$10^{-4}$</td>
<td>$10^{-5}$</td>
<td>$10^{-6}$</td>
<td></td>
</tr>
<tr>
<td>$10^{-6}$</td>
<td>$10^{-3}$</td>
<td>$10^{-4}$</td>
<td>$10^{-5}$</td>
<td>$10^{-6}$</td>
<td></td>
</tr>
<tr>
<td>$10^{-5}$</td>
<td>$10^{-3}$</td>
<td>$10^{-4}$</td>
<td>$10^{-5}$</td>
<td>$10^{-6}$</td>
<td></td>
</tr>
<tr>
<td>$10^{-4}$</td>
<td>$10^{-3}$</td>
<td>$10^{-4}$</td>
<td>$10^{-5}$</td>
<td>$10^{-6}$</td>
<td></td>
</tr>
<tr>
<td>$10^{-3}$</td>
<td>$10^{-3}$</td>
<td>$10^{-4}$</td>
<td>$10^{-5}$</td>
<td>$10^{-6}$</td>
<td></td>
</tr>
<tr>
<td>$10^{-2}$</td>
<td>$10^{-3}$</td>
<td>$10^{-4}$</td>
<td>$10^{-5}$</td>
<td>$10^{-6}$</td>
<td></td>
</tr>
<tr>
<td>$10^{-1}$</td>
<td>$10^{-3}$</td>
<td>$10^{-4}$</td>
<td>$10^{-5}$</td>
<td>$10^{-6}$</td>
<td></td>
</tr>
<tr>
<td>$10^{-0}$</td>
<td>$10^{-3}$</td>
<td>$10^{-4}$</td>
<td>$10^{-5}$</td>
<td>$10^{-6}$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>POD Tolerance</th>
<th>POD Only</th>
<th>POD+DEIM(1e-3)</th>
<th>POD+DEIM(1e-4)</th>
<th>POD+DEIM(1e-4)+PWL</th>
<th>PWL Only</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-18}$</td>
<td>$10^{-3}$</td>
<td>$10^{-4}$</td>
<td>$10^{-5}$</td>
<td>$10^{-6}$</td>
<td></td>
</tr>
<tr>
<td>$10^{-14}$</td>
<td>$10^{-3}$</td>
<td>$10^{-4}$</td>
<td>$10^{-5}$</td>
<td>$10^{-6}$</td>
<td></td>
</tr>
<tr>
<td>$10^{-10}$</td>
<td>$10^{-3}$</td>
<td>$10^{-4}$</td>
<td>$10^{-5}$</td>
<td>$10^{-6}$</td>
<td></td>
</tr>
<tr>
<td>$10^{-6}$</td>
<td>$10^{-3}$</td>
<td>$10^{-4}$</td>
<td>$10^{-5}$</td>
<td>$10^{-6}$</td>
<td></td>
</tr>
<tr>
<td>$10^{-2}$</td>
<td>$10^{-3}$</td>
<td>$10^{-4}$</td>
<td>$10^{-5}$</td>
<td>$10^{-6}$</td>
<td></td>
</tr>
<tr>
<td>$10^{0}$</td>
<td>$10^{-3}$</td>
<td>$10^{-4}$</td>
<td>$10^{-5}$</td>
<td>$10^{-6}$</td>
<td></td>
</tr>
</tbody>
</table>
Figure 5.4: Comparison of ROM treatment of nonlinearity $N(y)$ as measured by error from the FOM simulation (top figure) and error in the nonlinearity separately (bottom figure). While the PWL approximation is inferior in terms of accuracy, its ease of implementation on pre-existing FEM software is its advantage. Results are generated on a coarse mesh: $n_x = n_y = 40$, $n_t = 50$. 
Figures 5.3 and 5.4 show that the Lagrange interpolation in the nonlinearity, $N(y) \mapsto Mf^{n}(y)$, introduces a floor in the ROM fidelity, labeled PWL in the figures. However, this floor may be below desired accuracy tolerance depending on the POD tolerance chosen. Further, the floor depends on the discretization as it decreases with refinement of the mesh. To see this, I compare the error of the PWL approximation in the nonlinearity for various mesh sizes in Figure 5.5.

![PWL Only Error in Nonlinearity, N(y)](image)

Figure 5.5: Error in the Piecewise Linear Lagrange Interpolation as it varies with mesh discretization.
Chapter 6

Semilinear Parabolic Optimal Control

This chapter combines the methods detailed in the two preceding it: model reduction to accelerate Hessian-vector computation in Chapter 4 and piecewise polynomial approximation of the nonlinear term in semilinear parabolic problems in Chapter 5. Specifically, I will demonstrate how piecewise polynomial approximation can simplify the precomputation for the ROM Hessian-times-vector computation.

The dominating cost of the Hessian-Vector computation $\nabla^2 J(\vec{u}) \vec{v}$ in Algorithm 4.2 are the two linear solves of the linearization and 2nd order adjoint and construction of $[\lambda^T_k F(y_k)]_{yy}$. Consequently, the Discrete Empirical Interpolation Method (DEIM) is a critical step in accelerating the Hessian-vector computation in Newton-CG. Without the DEIM approximations, the subspace-based model reduction of the nonlinearities $V^T F_y(y_k) V$ and $V^T [\lambda^T_k F(y_k)]_{yy} V$ have costs that depend on the quadrature scheme and full order degrees of freedom in the online construction for every step in time. Thus, subspace-based model reduction without DEIM reduces the number of unknowns without necessarily decreasing the computational complexity.

I propose a model reduction precomputation scheme in this chapter that dramatically accelerates the Hessian-vector computation. In practice, the quadrature scheme
may be unavailable to the user when using pre-existing Finite Element code. Therefore, I propose a non-intrusive method for constructing the DEIM subspaces that can be done without modifying existing code for the nonlinear terms $F(y), \text{F}_y(y)$, and $[\lambda^T F(y)]_{yy}$. I will demonstrate the acceleration of Newton’s method with this approach for two cases of nonlinear reaction: (1) a cubic reaction term and (2) an exponential reaction term in a solid fuel ignition model and show a dramatic acceleration of Newton’s Method.

6.1 Problem Formulation

Consider an optimal control problem for the diffusion-advection-reaction equation with a nonlinear reaction term. Minimize

$$\frac{1}{2} \int_0^1 \int_\Omega (y(x,t;u) - y_d(x,t))^2dxdt + \frac{\alpha}{2} \int_0^1 \int_\Omega u(x,t)^2dxdt,$$

(6.1.1a)

where for given function $u$ the function $y = y(\cdot; \cdot; u)$ is the solution of

$$y_t(x,t) - \mu \Delta y(x,t) + f(y(x,t)) = u(t) \quad (x,t) \in \Omega \times (0,T),$$

(6.1.1b)

$$\nabla y(x,t) \cdot n = 0 \quad (x,t) \in \partial \Omega \times (0,T),$$

(6.1.1c)

$$y(x,0) = y_0(x) \quad x \in \Omega.$$

To simplify notation, let $\Omega \subset \mathbb{R}^d$, $d \in \{1,2,3\}$ and denote its boundary, $\partial \Omega$. Here we denote the desired state $y_d : \Omega \times (0,T) \rightarrow \mathbb{R}$, control $u : \Omega \times (0,T) \rightarrow \mathbb{R}$, and initial condition $y_0 : \Omega \rightarrow \mathbb{R}$ are given functions and $\mu > 0$ is diffusion coefficient. Further, assume $f : \mathbb{R} \rightarrow \mathbb{R}$ is twice continuously differentiable which is needed for 2nd differentiability of the objective function.

6.1.1 Finite Element Discretization

The following finite element discretization is developed with more care in Chapter 5. I abbreviate the spatial and temporal discretizations here for convenience.
Construct a finite element discretization of the semilinear equation, (6.1.1b) write the implicit constraint in its semi-discretized form (5.2.15a) which I restate here for convenience,

\[ M \frac{d}{dt} y(t) + Ay(t) + N(y(t)) = Bu(t), \]  

(6.1.2a)
\[ y(0) = y_0, \]  

(6.1.2b)

where \( A, B, M \in \mathbb{R}^{n \times n} \) are defined element-wise,

\[ A_{ij} = \mu \int_\Omega \nabla \phi_i(x) \cdot \nabla \phi_j(x) dx, \quad M_{ij} = B_{ij} = \int_\Omega \phi_i(x) \phi_j(x) dx. \]

The nonlinearity \( N: \mathbb{R}^n \to \mathbb{R}^n \) is defined via quadrature,

\[ N(y; f) = \Phi W f^g(y), \]

\[ \Phi = \begin{bmatrix} \phi_1(\tilde{x}_1) & \cdots & \phi_1(\tilde{x}_{n_q}) \\ \vdots & \ddots & \vdots \\ \phi_{n_y}(\tilde{x}_1) & \cdots & \phi_{n_y}(\tilde{x}_{n_q}) \end{bmatrix}, \quad W = \begin{bmatrix} w_1 \\ \vdots \\ w_{n_q} \end{bmatrix}, \]

and \( f^g(y): \mathbb{R}^{n_y} \to \mathbb{R}^{n_q} \) is a vector of the nonlinear function \( f \circ y_h \) evaluated at the quadrature points \( \{\tilde{x}_i\}_{i=1}^{n_q}, \)

\[ f^g(y(t)) = \begin{bmatrix} f(y_h(\tilde{x}_1), t) \\ \vdots \\ f(y_h(\tilde{x}_{n_q}, t)) \end{bmatrix}, \quad y_h(x, t) = \sum_{i=1}^{n_y} y_i \phi_i(x). \]

The finite element discretization of the objective function (6.1.1a) is written,

\[ \frac{1}{2} \int_0^1 (y(t) - y_d(t_k))^T Q(y(t) - y_d(t_k)) dt + \frac{\alpha}{2} \int_0^1 u(t)^T R u(t) \]  

(6.1.3)

where \( Q, R \in \mathbb{R}^{n_y \times n_y} \) are defined element-wise,

\[ Q_{ij} = R_{ij} = \int_\Omega \phi_i(x) \phi_j(x) dx. \]

The semidiscretized objective function (6.1.3) and implicit constraint (6.1.2a) are discretized in time via Backward Euler.
**Backward Euler Discretization**  Define a uniform time stepping scheme where

\[ 0 = t_0 < t_1 < \ldots < t_{n_t} = T \]

and \( \Delta t = t_{k+1} - t_k = \frac{T}{n_t} \). Then the fully discretized problem becomes

\[
\min_{\vec{y}, \vec{u}} \sum_{k=1}^{n_t} \Delta t \left[ \frac{1}{2} (\vec{y}_k - \vec{y}_d(t_k))^T Q (\vec{y}_k - \vec{y}_d(t_k)) + \frac{\alpha}{2} \vec{u}_k^T \vec{R} \vec{u}_k \right]
\]

s.t. \((M + \Delta t A)\vec{y}_{k+1} + \Delta t N(\vec{y}_{k+1}) = M\vec{y}_k + \Delta t \vec{B} \vec{u}_{k+1}, \quad k = 0, \ldots, n_t - 1 \)

where \( \vec{y}_0 \) is given and the optimization variables are defined

\[
\vec{u} = ((u_1)^T, \ldots, (u_{n_t})^T)^T \in \mathbb{R}^{n_y n_t}, \quad \vec{y} = ((y_1)^T, \ldots, (y_{n_t})^T)^T \in \mathbb{R}^{n_y n_t}.
\]

and \( n_y \) and \( n_t \) are the number of elements and time steps respectively.

I will restate the ROM Hessian-vector found in Algorithm 4.4 in Chapter 4 for this specific case in (6.1.4). First, note that \( F : \mathbb{R}^{n_y} \rightarrow \mathbb{R}^{n_y} \) in Algorithm 4.4 is equivalent in this context to the stiffness matrix and the nonlinearity,

\[
F(\vec{y}_k) = A\vec{y}_k + N(\vec{y}_k), \quad F_y(\vec{y}_k) = A + N_y(\vec{y}_k).
\]

Second, many of the quantities in Algorithm 4.4 reduce in the following ways:

\[
G(\vec{u}_k) = \vec{B}\vec{u}_k, \quad G_u(\vec{u}_k)\vec{v}_k = \vec{B}\vec{v}_k, \quad \ell_y(\vec{y}_k) = \Delta t Q (\vec{y}_k - \vec{y}_d(t_k)), \quad \ell_{yy}(\vec{y}_k) = \Delta t Q, \quad \ell_{uu}(\vec{u}_k) = \Delta t \vec{R}, \quad (\lambda_k^T F(\vec{y}_k))_{yy} = (\lambda_k^T N(\vec{y}_k))_{yy}, \quad (\lambda_k^T G(\vec{u}_k))_{uu} = 0.
\]

With the former substitutions made, I restate the Hessian-vector computation for this specific case to aide in the discussion of what is approximated by piecewise
polynomial approximation and where precomputation is needed fast ROM Hessian-vector computation in Algorithm 6.1.

Recall that the Hessian-vector computation depends on solutions to the state and adjoint solutions. For convenience, I restate both here in the notation introduced above. Given a control $u_1, \ldots, u_{n_t}$, solve the state equation forward in time,

$$ M(y_{k+1} - y_k) + \Delta t \, Ay_{k+1} + \Delta t \, N(y_{k+1}) = \Delta t \, Bu_{k+1}, $$

$k = 0, \ldots, n_t - 1,$

for $y_1, \ldots, y_{n_t}$. Then the adjoint equation is solved backwards in time for $\lambda_{n_t}, \ldots, \lambda_1$,

$$ [M + \Delta t \, A + \Delta t \, N_y(y_{n_t})]^T \lambda_{n_t} = - \Delta t \, Q(y_{n_t} - y_d(t_{n_t})), $$

$$ [M + \Delta t \, A + \Delta t \, N_y(y_k)]^T \lambda_k = M \lambda_{k+1} - \Delta t \, Q(y_k - y_d(t_k)), $$

$k = n_t - 1, \ldots, 1$. 

As both quantities, the state and adjoint solutions, are computed in the gradient computation, they are omitted from the Hessian-vector computation in Algorithm 6.1. Both solutions are recycled to construct a low-dimensional subspace spanned by the columns of $V$ in Algorithm 4.3.

Ultimately, to accelerate the ROM Hessian-vector computation I will introduce approximations to the nonlinear $N(y)$ and its derivatives. However, it is important to note that the state and adjoint equations (6.1.5) and (6.1.6a) are solved without POD or DEIM approximations to compute the true gradient of the discretized problem (6.1.4).

The terms $V^T N_y(y_k)V$ and $V^T [A^T N(y_k)]_{yy} V$ vary with $y_k$ and therefore change at every time step. To make the ROM Hessian-vector computation efficient, further approximations of these terms is needed. The ROM Hessian-vector computation with a naive implementation of these terms is described in Algorithm 6.1. The following section describes how the DEIM approach of subsection 5.4.3 is applied to efficiently approximate these matrices.
**Algorithm 6.1: Hessian Times Vector Computation w/ Naive MOR – $\nabla^2 J(\mathbf{u}) \mathbf{v}$**

1: Solve the POD linearization,

$$[\mathbf{I} + \Delta t \mathbf{V}^T \mathbf{A} \mathbf{V} + \mathbf{V}^T \mathbf{N}_y(y_{k+1}) \mathbf{V}] \hat{\mathbf{w}}_{k+1} = \hat{\mathbf{w}}_k + \Delta t \mathbf{V}^T \mathbf{B} \mathbf{v}_{k+1},$$

\[ k = 0, \ldots, n_t - 1. \]

with $\hat{\mathbf{w}}_0 = \mathbf{V}^T \mathbf{w}_0 = 0$ for $\hat{\mathbf{w}}_1, \ldots, \hat{\mathbf{w}}_{n_t}$.

2: Solve the POD adjoint equation for $\hat{\mathbf{p}}_k$,

$$[\mathbf{I} + \Delta t \mathbf{V}^T \mathbf{A} \mathbf{V} + \mathbf{V}^T \mathbf{N}_y(y_{n_t}) \mathbf{V}]^T \hat{\mathbf{p}}_{n_t} = -\Delta t \mathbf{V}^T [(\lambda_{n_t}^T \mathbf{N}(y_{n_t}))_{yy} + \mathbf{Q}] \mathbf{V} \hat{\mathbf{w}}_{n_t},$$

$$[\mathbf{I} + \Delta t \mathbf{V}^T \mathbf{A} \mathbf{V} + \mathbf{V}^T \mathbf{N}_y(y_k) \mathbf{V}]^T \hat{\mathbf{p}}_k = \hat{\mathbf{p}}_{k+1} - \Delta t \mathbf{V}^T [(\lambda_k^T \mathbf{N}(y_k))_{yy} + \mathbf{Q}] \mathbf{V} \hat{\mathbf{w}}_k,$$

\[ k = n_t - 1, \ldots, 1, \]

for $\hat{\mathbf{p}}_{n_t}, \ldots, \hat{\mathbf{p}}_1$.

3: Compute the approximation to the action of the Hessian,

$$\nabla^2 J(\mathbf{u}) \mathbf{v} = \begin{pmatrix}
-\Delta t \mathbf{B}^T \mathbf{V} \hat{\mathbf{p}}_1 + \mathbf{R} \mathbf{v}_1 \\
\vdots \\
-\Delta t \mathbf{B}^T \mathbf{V} \hat{\mathbf{p}}_{n_t} + \mathbf{R} \mathbf{v}_{n_t}
\end{pmatrix}.$$

To make these simulations fast, several matrices need to be precomputed:

$$\mathbf{A}_r := \mathbf{V}^T \mathbf{A} \mathbf{V},$$

$$\mathbf{M}_r := \mathbf{V}^T \mathbf{M} \mathbf{V},$$

$$\mathbf{B}_r := \mathbf{V}^T \mathbf{B},$$

$$\mathbf{Q}_r := \mathbf{V}^T \mathbf{Q} \mathbf{V}.$$ 

In the next section I will discuss the piecewise polynomial approximation introduced in Chapter 5 and demonstrate how the DEIM can be applied to the approximation.
6.2 Piecewise Polynomial Approximation of the Nonlinear Reaction Term and DEIM

There are various approaches available for the implementation of the DEIM on the nonlinear term $N(y)$. For a summary, see Section 5.4. In this section, we consider the piecewise polynomial approximation only as it allows for a nonintrusive method for precomputation of the nonlinearities $V^T N_y(y) V \in \mathbb{R}^{n_r \times n_r}$ and $V^T [\lambda^T N(y)]_{yy} V \in \mathbb{R}^{n_r \times n_r}$.

Recall that we can approximate the nonlinearity $N : \mathbb{R}^{n_y} \rightarrow \mathbb{R}^{n_y}$ with an approximation using the mass matrix $M$ and an element-wise nonlinear vector-valued function $f_n : \mathbb{R}^{n_y} \rightarrow \mathbb{R}^{n_y}$,

$$N(y) \approx Mf_n(y), \quad f_n(y) := \begin{bmatrix} f(y_1) \\ \vdots \\ f(y_n) \end{bmatrix},$$

whose derivatives are considerably easier to evaluate than the original $N(y)$:

$$[Mf(y)]_y = M \begin{bmatrix} f'(y_1) \\ \vdots \\ f'(y_n) \end{bmatrix},$$

$$[\lambda^T Mf^n(y)]_{yy} = \begin{bmatrix} \lambda^T M e_1 f''(y_1) \\ \vdots \\ \lambda^T M e_n f''(y_n) \end{bmatrix}.$$

This approximation to $N(y)$ is amenable to the DEIM by approximation of the element-wise nonlinearity $f_n(y) : \mathbb{R}^{n_y} \rightarrow \mathbb{R}^{n_y}$,

$$f_n(y) \approx U(P^T U)^{-1} P^T f(y),$$

where $U \in \mathbb{R}^{n_y \times m}$ is chosen via POD on the snapshot matrix, $[f_n(y_1), \ldots, f_n(y_m)]$ and $P \in \mathbb{R}^{n_y \times m}$ are columns of the identity chosen via DEIM. Define this new reaction
term with the piecewise polynomial approximation and the DEIM as \( \tilde{N} : \mathbb{R}^{n_y} \to \mathbb{R}^{n_y}, \)

\[
\tilde{N}(y) := MU(P^T U)^{-1}P^T f^n(y).
\]

Derivatives of \( \tilde{N}(y) \) are considerably easier to evaluate than the original nonlinearity \( N(y) \) as \( f^n \) is evaluated element-wise.

Note that it is possible to construct the product \( MU(P^T U)^{-1} \in \mathbb{R}^{n_y \times m} \) column-wise without access to the mass matrix \( M \) nor the quadrature scheme used by calling the nonlinearity with \( f \) temporarily replaced by the identity function \( \text{id} \),

\[
[MU(P^T U)^{-1}]_{:,i} = N([U(P^T U)^{-1}]_{:,i}; f \equiv \text{id}), \quad i = 1, \ldots, m.
\]

This trick will be useful later in this chapter when we want to precompute the approximations to ROM nonlinear terms \( N_y(y) \) and \( (\lambda^T N(y))_{yy} \).

### 6.2.1 ROM Precomputation of First Derivative of Nonlinearity

Exploiting the element-wise dependence of \( f^n : \mathbb{R}^{n_y} \to \mathbb{R}^{n_y} \) (i.e. \( f^n_i(y) = f(y_i) \) for \( i = 1, \ldots, n_y \) ) the derivative of \( \tilde{N}(y) \) can be written,

\[
\tilde{N}_y(y) := MU(P^T U)^{-1}P^T \begin{bmatrix}
  f'(y_1) \\
  \vdots \\
  f'(y_{n_y})
\end{bmatrix}

= MU(P^T U)^{-1} \left( \sum_{i=1}^m f'(y_{\varphi_i}) \tilde{e}_i e^T_{\varphi_i} \right)

= \sum_{i=1}^m f'(y_{\varphi_i}) M g_i e^T_{\varphi_i}
\]

This approximation to the nonlinearity can be written as a sum of \( m \) matrices \( M g_i e^T_{\varphi_i} \in \mathbb{R}^{n_y \times n_y} \) where \( g_i \in \mathbb{R}^{n_y} \) the \( i \)th column of the product \( U(P^T U)^{-1} \in \mathbb{R}^{n_y \times m} \). Then the subspace-based approximation \( V^T \tilde{N}_y(y) V \in \mathbb{R}^{n_r \times n_r} \) can be written,

\[
V^T \tilde{N}_y(y) V := \sum_{i=1}^m f'(y_{\varphi_i}) V^T M g_i e^T_{\varphi_i} V,
\]
where $V^T M g_i e_{\theta_i} V \in \mathbb{R}^{n_r \times n_r}$ is precomputed for $i = 1, \ldots, m$.

### 6.2.2 ROM Precomputation of the Second Derivative of Nonlinearity

In a similar fashion, the second derivative in the right-hand side of the second order adjoint equation can be written,

$$\lambda^T \tilde{N}(y)|_{yy} = \lambda^T M f(y_{\theta_i})|_{yy},$$

$$= \sum_{i=1}^{m} \lambda^T M g_i f(y_{\theta_i})|_{yy},$$

$$= f''(y_{\theta_i}) \sum_{i=1}^{m} \lambda^T M g_i e_{\theta_i} e_{\theta_i}^T,$$

where $g_i \in \mathbb{R}^{n_y}$ is the $i$th column of the product $U(P^T U)^{-1} \in \mathbb{R}^{n_y \times m}$. However, the subspace-based projection of $\lambda^T \tilde{N}(y)$,

$$V^T [\lambda^T \tilde{N}(y)|_{yy} V = \sum_{i=1}^{m} f''(y_{\theta_i}) \lambda^T M g_i (V^T e_{\theta_i} e_{\theta_i}^T V),$$

cannot be precomputed as the coefficient $\lambda^T M g_i \in \mathbb{R}$ changes at every time step as $\lambda$ depends on time and the cost of the inner product depends on the degrees of freedom of the full model. As an alternative, one can do a second DEIM approximation on the adjoint variable $\lambda$. Here, I differentiate the two DEIM oblique projections on $f^n$ and $\lambda$ as $U_f (P_f^T U_f)^{-1} P_f^T$ and $U_{\lambda} (P_{\lambda}^T U_{\lambda})^{-1} P_{\lambda}^T$ respectively. With this approach, $[\lambda^T \tilde{N}(y)|_{yy} \in \mathbb{R}^{n_y \times n_y}$ can be approximated,

$$[\lambda^T \tilde{N}(y)|_{yy} = [\lambda^T M f^n(y)|_{yy} \approx [\lambda^T P_{\lambda} (U_{\lambda}^T P_{\lambda})^{-1} U_{\lambda}^T M U_f (P_f^T U_f)^{-1} P_f^T f^n(y)|_{yy}.$$

To simplify notation, define $H := (U_{\lambda}^T P_{\lambda})^{-1} U_{\lambda}^T M U_f (P_f^T U_f)^{-1} \in \mathbb{R}^{m_{\lambda} \times m_f}$ with
columns $h_i := H_{:,i} \in \mathbb{R}^{m \lambda}$ and the new approximation,

$$[\lambda^T \tilde{N}(y)]_{yy} = [\lambda^T P \lambda H P_f T f^n(y)]_{yy}$$

$$= [\lambda^T P \lambda \sum_{i=1}^m h_i f(y_{\theta_i})]_{yy}$$

$$= \sum_{i=1}^m f''(y_{\theta_i})(\lambda^T P \lambda h_i)(V^T e_{\theta_i} e_{\theta_i}^T V).$$

Unlike (6.2.1), $V^T [\lambda^T \tilde{N}(y)]_{yy} V$ can be conveniently precomputed,

$$V^T [\lambda^T \tilde{N}(y)]_{yy} V = \sum_{i=1}^m f''(y_{\theta_i})(\lambda^T P \lambda h_i)(V^T e_{\theta_i} e_{\theta_i}^T V)$$

where $V^T e_{\theta_i} e_{\theta_i}^T V \in \mathbb{R}^{n_r \times n_r}$ is precomputed and the inner product $\lambda^T P \lambda h_i$ can be evaluated online with $O(n_r)$ complexity. Note also that the cost of computing POD on the trajectory \{\lambda_1, \ldots, \lambda_n\} is already done in the process of computing the subspace $V$ in Algorithm 4.3.

### 6.2.3 Computation of the DEIM Subspaces

Throughout this section, I have demonstrated the piecewise polynomial approximation with DEIM for a single oblique projection, $U(P^T U)^{-1} P^T$. However, there is no reason why the same projection needs to be used for both derivatives $\tilde{N}_y(y)$ and $(\lambda^T \tilde{N}(y))_{yy}$. In fact, it would be preferable to run DEIM twice, once for each case,

$$\tilde{N}_y(y) := MU_1(P_1^T U_1)^{-1} P_1^T \begin{bmatrix} f'(y_1) \\
\ddots \\
f'(y_{n_y}) \end{bmatrix}$$

$$[\lambda^T \tilde{N}(y)]_{yy} := [\lambda^T M U_2(P_2^T U_2)^{-1} P_2^T f^n(y)]_{yy}.$$
Each projection is computed from the first and second derivative snapshots separately in Algorithm 6.2.

**Algorithm 6.2: Construction of DEIM Subspaces**

1: Given solutions to the state equation, $\vec{y}(\vec{u})$, compute the element-wise reaction terms,

$$
X_1 = \begin{bmatrix}
    f'(\vec{y}_1) & \cdots & f'(\vec{y}_{n_1}) \\
    \vdots & \ddots & \vdots \\
    f'(\vec{y}_{n_y}) & \cdots & f'(\vec{y}_{n_y})
\end{bmatrix}, \quad X_2 = \begin{bmatrix}
    f''(\vec{y}_1) & \cdots & f''(\vec{y}_{n_1}) \\
    \vdots & \ddots & \vdots \\
    f''(\vec{y}_{n_y}) & \cdots & f''(\vec{y}_{n_y})
\end{bmatrix}.
$$

2: Construct by POD the matrices $U_1 \in \mathbb{R}^{n_y \times m_1}$ and $U_2 \in \mathbb{R}^{n_y \times m_2}$,

$$
\frac{\|M^{1/2}(I - U_1 U_1^T M)X_1\|_F}{\|M^{1/2}X_1\|_F} < \text{tol}, \quad \text{and} \quad \frac{\|M^{1/2}(I - U_2 U_2^T M)X_2\|_F}{\|M^{1/2}X_2\|_F} < \text{tol}.
$$

3: Run the DEIM to construct associated matrices,

$$
P_1 = \text{deim}(U_1) \quad \text{and} \quad P_2 = \text{deim}(U_2).
$$

Before presenting my numerical results, I will restate the ROM Hessian-vector computation in Algorithm 6.1 with the nonintrusive DEIM approximations in Algorithm 6.3. Recall that for the nonintrusive approximations,

$$
V^T \tilde{N}_y(y_k)V := \sum_{i=1}^{m} f'(\vec{y}_{k\rho_i}) V^T M g_i e_{\rho_i}^T V,
$$

$$
V^T \tilde{\Lambda}_k^{T} \tilde{N}(y_k)_{yy} V = \sum_{i=1}^{m} f''(\vec{y}_{k\rho_i}) \tilde{\Lambda}_k^{T} M g_i (V^T e_{\rho_i} e_{\rho_i}^T V),
$$

the quantities $V^T M g_i e_{\rho_i}^T V \in \mathbb{R}^{n_r \times n_r}$ and $V^T e_{\rho_i} e_{\rho_i}^T V \in \mathbb{R}^{n_r \times n_r}$ are precomputed. Therefore the cost of constructing either term after precomputation is that of adding only $m$ matrices of small dimension $n_r \times n_r$ and the cost of of product $V^T \tilde{\Lambda}_k^{T} \tilde{N}(y_k)_{yy} V \tilde{w}_k$ is $O(n_r^2)$. 
Algorithm 6.3: Hessian-Vector w/ Nonintrusive DEIM – $\nabla^2 J(\bar{u})\bar{v}$

1: Solve the POD linearization,

$$[I + \Delta t V^T AV + V^T \tilde{N}_y(y_{k+1})V]\hat{w}_{k+1} = \hat{w}_k + \Delta t V^T B v_{k+1},$$

$k = 0, \ldots, n_t - 1.$

with $\hat{w}_0 = V^T w_0 = 0$ for $\hat{w}_1, \ldots, \hat{w}_{n_t}.$

2: Solve the POD adjoint equation for $\hat{p}_k,$

$$[I + \Delta t V^T AV + V^T \tilde{N}_y(y_{nt})V]^T \hat{p}_{nt} = -\Delta t V^T [(\lambda_{nt}^T \tilde{N}(y_{nt}))yy + Q] V \hat{w}_{nt},$$

$$[I + \Delta t V^T AV + V^T \tilde{N}_y(y_k)V]^T \hat{p}_k = \hat{p}_{k+1} - \Delta t V^T [(\lambda_k^T \tilde{N}(y_k))yy + Q] V \hat{w}_k,$$

$k = n_t - 1, \ldots, 1,$

for $\hat{p}_{nt}, \ldots, \hat{p}_1.$

3: Compute the approximation to the action of the Hessian,

$$\nabla^2 J(\bar{u})\bar{v} = \begin{pmatrix} -\Delta t B^T V \hat{p}_1 + R v_1 \\ \vdots \\ -\Delta t B^T V \hat{p}_{nt} + R v_{nt} \end{pmatrix}.$$

Numerical results in the following section use the latter ROM Hessian-vector computation in Algorithm 6.3 with the nonintrusive DEIM implemented on the nonlinearity and its derivatives.

### 6.3 Results - Cubic Reaction

I consider the diffusion-advection-reaction optimal control problem with linear diffusion and advection and nonlinear (cubic) reaction. For convenience, I restate the
optimal control problem,
\[
\min u \frac{1}{2} \int_0^1 \int_{\Omega} (y(x,t;u) - y_d(x,t))^2 \, dx \, dt + \frac{\alpha}{2} \int_0^1 \int_{\Omega} u(x,t)^2 \, dx \, dt,
\]
where for given function \( u \) the function \( y(\cdot, \cdot; u) \) is the solution of
\[
y_t(x,t) - \mu \Delta y(x,t) + f(y(x,t)) = u(x,t) \quad (x, t) \in \Omega \times (0, T), \tag{6.3.1b}
\]
\[
y(x,t) = 0 \quad (x, t) \in \partial \Omega \times (0, T), \tag{6.3.1c}
\]
\[
y(x,0) = y_0(x) \quad x \in \Omega.
\]
where the nonlinear reaction term is cubic, \( f(y) = y^3 \), the domain is rectangular \( \Omega = [0, 1]^2 \). Desired state, \( y_d(x,t) = 2e^t + 2x_1(x_1 - 1) + 2x_2(x_2 - 1) \), initial state \( y_0(x,t) = \sin(2\pi x_1) \), diffusion coefficient \( \mu = 1 \) and control penalty \( \alpha = 10^{-4} \) are given.

Results in this section are run on a uniform triangular mesh with \( n_x = n_y = 40 \) and \( n_t = 100 \) and piecewise linear finite elements resulting in 1681 and 1521 degrees of freedom in the control and state respectively. Both optimizations with FOM Hessian-vector and ROM Hessian-vector converged to the same solution. However, the latter converged much faster on MATLAB running on a MacBook Pro (13-inch, Retina, Mid 2014).

<table>
<thead>
<tr>
<th></th>
<th>FOM</th>
<th>ROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>State Solves</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>Adjoint Solves</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>Hess-Vec Mult</td>
<td>113</td>
<td>175</td>
</tr>
<tr>
<td>Total Time (s)</td>
<td>432.61</td>
<td>50.28</td>
</tr>
</tbody>
</table>

Table 6.1: Optimization FOM vs ROM. Overall speedup is 8.5×
Table 6.2: Optimization via FOM Hessian-Vector.

<table>
<thead>
<tr>
<th>iter</th>
<th>$|\nabla J(\tilde{u})|$</th>
<th>$\alpha$</th>
<th>CG</th>
<th>$|\nabla^2 J(\tilde{u})\tilde{v} + \nabla J(\tilde{u})|$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$2.78 \cdot 10^{-4}$</td>
<td>1.00</td>
<td>18</td>
<td>$4.69 \cdot 10^{-8}$</td>
</tr>
<tr>
<td>1</td>
<td>$2.50 \cdot 10^{-5}$</td>
<td>1.00</td>
<td>20</td>
<td>$6.20 \cdot 10^{-10}$</td>
</tr>
<tr>
<td>2</td>
<td>$4.73 \cdot 10^{-6}$</td>
<td>1.00</td>
<td>26</td>
<td>$1.72 \cdot 10^{-11}$</td>
</tr>
<tr>
<td>3</td>
<td>$2.93 \cdot 10^{-7}$</td>
<td>1.00</td>
<td>35</td>
<td>$6.71 \cdot 10^{-14}$</td>
</tr>
<tr>
<td>4</td>
<td>$1.49 \cdot 10^{-9}$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6.3: Optimization via ROM Hessian-Vector. $tol_{POD} = 10^{-3}$, $tol_{DEIM} = 10^{-3}$.

Stopping criteria triggered in both optimizations: $\|\nabla J(\tilde{u})\| < 10^{-7}$.
Figure 6.1: Optimal control, $u(x,t)$ at $t = \{0, 0.1, 0.5, 1\}$. 
Figure 6.2: State at optimal solution, $y(x, t)$ at $t = \{0, 0.1, 0.5, 1\}$ (below). Desired state $y_d(x, t)$ (above).
Figure 6.3: Convergence history for Newton-CG with and without POD+DEIM approximations in the Hessian-vector computation
6.4 Results - Solid Fuel Ignition Model

I consider the diffusion-reaction reaction optimal control problem with linear diffusion and nonlinear exponential reaction from Ito et al. (2002) and Borzi et al. (2000) [9, 32].

\[
\min_u \frac{1}{2} \int_0^1 \int_\Omega (y(x, t; u) - y_d(x, t))^2 dx dt + \frac{\alpha}{2} \int_0^1 \int_\Omega u(x, t)^2 dx dt, \quad (6.4.1a)
\]

where for given function \(u\) the function \(y(\cdot, \cdot; u)\) is the solution of

\[
y_t(x, t) - \mu \Delta y(x, t) - \delta e^{y(x,t)} = u(x, t) \quad (x, t) \in \Omega \times (0, T), \quad (6.4.1b)
\]

\[
y(x, t) = 0 \quad (x, t) \in \partial\Omega \times (0, T), \quad (6.4.1c)
\]

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

where the domain is rectangular \(\Omega = [0,1]^2\), control penalty \(\alpha = 5^{-3}\), desired state \(y_d(x, t) = \frac{1}{\pi^2} \sin(\pi x_1) \sin(\pi x_2)\), initial state \(y_0(x, t) \equiv 0\), diffusion coefficient \(\mu = 1\), and reaction term coefficient \(\delta = 5\) are given.

Results in this section are run on a uniform triangular mesh with \(n_x = n_y = 40\) and \(n_t = 100\) and piecewise linear finite elements resulting in 1681 and 1521 degrees of freedom in the control and state respectively. Both optimizations with FOM Hessian-vector and ROM Hessian-vector converged to the same solution. However, the latter converged much faster on MATLAB running on a MacBook Pro (13-inch, Retina, Mid 2014).

<table>
<thead>
<tr>
<th></th>
<th>FOM</th>
<th>ROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>State Solves</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Adjoint Solves</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Hess-Vec Mult</td>
<td>50</td>
<td>51</td>
</tr>
<tr>
<td>Total Time (s)</td>
<td>180.42</td>
<td>21.26</td>
</tr>
</tbody>
</table>

Table 6.4: Optimization FOM vs ROM. Overall speedup is 8.6×
Table 6.5: Optimization via FOM Hessian-Vector.

<table>
<thead>
<tr>
<th>iter</th>
<th>$|\nabla J(\bar{u})|$</th>
<th>$\alpha$</th>
<th>CG</th>
<th>$|\nabla^2 J(\bar{u})\tilde{v} + \nabla J(\bar{u})|$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$4.56 \cdot 10^{-5}$</td>
<td>1.00</td>
<td>9</td>
<td>$8.55 \cdot 10^{-10}$</td>
</tr>
<tr>
<td>1</td>
<td>$3.87 \cdot 10^{-6}$</td>
<td>1.00</td>
<td>12</td>
<td>$9.35 \cdot 10^{-12}$</td>
</tr>
<tr>
<td>2</td>
<td>$2.62 \cdot 10^{-8}$</td>
<td>1.00</td>
<td>27</td>
<td>$6.48 \cdot 10^{-16}$</td>
</tr>
<tr>
<td>3</td>
<td>$1.27 \cdot 10^{-12}$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6.6: Optimization via ROM Hessian-Vector. $tol_{POD} = 10^{-3}$, $tol_{DEIM} = 10^{-3}$. Stopping criteria triggered in both optimizations: $\|\nabla J(\bar{u})\| < 10^{-8}$.

Figure 6.4: No Control State at $t = \{0.5, 1\}$
Figure 6.5: Optimal control, $u(x, t)$ at $t = \{0, 0.1, 0.5, 1\}$. 
Figure 6.6: Desired state, $y_d(x)$ (top left). State at optimal solution, $y(x, t)$ at $t = \{0.1, 0.5, 1\}$ (below).
Figure 6.7: Convergence history for Newton-CG with and without POD+DEIM approximations in the Hessian-vector computation
6.5 Discussion of Results

The results in this chapter demonstrate a significant acceleration of Newton’s Method for the optimal control problem with semilinear parabolic constraints for two nonlinear reaction terms: cubic and exponential. While these results are encouraging, further investigation is needed.

First, results in this chapter trigger stopping criterion of a small gradient norm quickly after only a few iterations. Newton-CG using the FOM Hessian can observe quadratic convergence with appropriately chosen stopping criteria for CG. The results in this chapter use the following stopping condition for CG:

\[ \|\nabla^2 J(\tilde{u})\tilde{d} + \nabla J(\tilde{u})\| \leq \min\{\|\nabla J(\tilde{u})\|^2, 0.01\|\nabla J(\tilde{u})\|\}. \]

However, when using the ROM Hessian \( \nabla^2 J(\tilde{u}) \) to compute the descent direction \( \tilde{d} \), the Newton subproblem residual \( \|\nabla^2 J(\tilde{u})\tilde{d} + \nabla J(\tilde{u})\| \) cannot be solved to arbitrary precision without refinement of the POD and DEIM tolerances in the ROM Hessian-vector computation. A scheme for refining the ROM Hessian-vector computation would be helpful for this reason.

Second, the Newton-CG algorithm is sensitive to fidelity of the gradient computation which in turn depends on the fidelity of the nonlinear state solver. As nonlinear state equation is not solved exactly but instead to a desired tolerance, this creates errors in derivative computation that can impact the convergence history of Newton-CG. Unfortunately, this is a feature of nonlinear optimization regardless of the optimization procedure used. The state equation stopping criteria used in these simulations is \( 8.8645 \times 10^{-16} \); however, this may not be realistic in costly large-scale simulations or where state variable scaling is a concern. Therefore, simulation results depend on many tolerances: stopping criteria in the state equation, CG, Newton-CG, and potentially POD and DEIM tolerances if using a ROM Hessian.
Chapter 7

Well Rate Optimization

This chapter explores the efficacy of my approach on the well rate optimization problem from oil and gas engineering. I will run an optimization with respect to well rates of the two-phase incompressible flows in porous media found in Section 2.6. Two optimization challenges are introduced for the optimization problem that were not present in the optimal control of semilinear parabolic PDEs in Chapter 6:

1. the incompressible fluid model dictates that the net flows in and out of the wells, injection and production, respectively, must sum to zero, and

2. well engineering specifications place maximum flow rates for each well.

Therefore, our control variables must maximize the objective function and satisfy both equality and box inequality constraints. I accommodate the inequality constraints by modifying the objective function with a barrier penalty and the equality constraints via projection.

7.1 Problem Formulation

First, I assume that wells are either injection wells or production wells. That is, \([q_k]_i\), the flow rate of the \(i\)th well at time \(t_k\), is nonnegative if the \(i\)th well is an
injector ([q_k]_i ≥ 0 for all i ∈ I_{inj}) and nonpositive if a producer ([q_k]_i ≤ 0 for all i ∈ I_{prod}). The sets I_{prod} and I_{inj} are chosen beforehand and do not change during the optimization.

I discretize the two-phase incompressible reservoir equations neglecting capillary pressure in (2.6.7) into a coupled system of equations that form an implicit constraint,

\[ M \frac{s_{k+1} - s_k}{\Delta t} + G(s_{k+1}, p_{k+1}, q_{k+1}) = 0, \]  
\[ H(s_{k+1}, p_{k+1}, q_{k+1}) = 0. \]

where \( \{s_k\}_{k=1}^{n_t}, \{p_k\}_{k=1}^{n_t} \) denote saturation and pressure solutions in time. For a description of the finite volume discretization used, see subsection 2.7.3.

The objective function has three terms:

(a) a reward for production revenue, \( r_{oil} \sum_{i \in I_{prod}} |[q_k]_i| f_0([s_k]_i), \)

(b) a penalty for the cost of injected water, \( r_{inj} \sum_{i \in I_{inj}} [q_k]_i, \)

(c) a penalty for the cost of water treatment, \( r_{oper} \sum_{i \in I_{prod}} |[q_k]_i| f_w([s_k]_i). \)

All three terms are discounted via the daily discount rate \( r_{disc} \) to compute the net present value of all revenues less costs which we write as the reduced objective function,

\[ J(\bar{q}) := -\sum_{k=1}^{n_t} \Delta t (1 + r_{disc})^{-k} \Delta t \left[ r_{oil} \sum_{i \in I_{prod}} |[q_k]_i| f_0([s_k]_i) - r_{inj} \sum_{i \in I_{inj}} [q_k]_i \right. \]
\[ \left. - r_{oper} \sum_{i \in I_{prod}} |[q_k]_i| f_w([s_k]_i) \right] \]

where given control \( \bar{q} \in \mathbb{R}^{n_{inj}} \), saturation \( \{s_k\} \) and pressure \( \{p_k\} \) satisfy the implicit constraint (7.1.1a). Note that as the goal is to maximize the net present value, I add a leading negative sign to (7.1.2) to convert the problem to a minimization program. Therefore, given a well schedule \( \bar{q} \), the net present value is computed as \(-J(\bar{q})\).
As previously mentioned, the net flow rate must sum to zero,

\[ \mathbf{q}_k^T \mathbf{1} = 0, \quad k = 1, \ldots, n_t \]  

(7.1.3)

and additionally, each well must obey box inequality constraints,

\[ c_l \leq \mathbf{q}_k \leq c_u, \quad k = 1, \ldots, n_t. \]  

(7.1.4)

Finally, we can write the optimization program with mixed constraints as follows:

\[
\begin{align*}
\min_{\mathbf{q}} \quad & J(\mathbf{q}) \\
\text{s.t.} \quad & \mathbf{q}_k^T \mathbf{1} = 0, \quad k = 1, \ldots, n_t \\
& c_l \leq \mathbf{q}_k \leq c_u, \quad k = 1, \ldots, n_t.
\end{align*}
\]  

(7.1.5a, 7.1.5b, 7.1.5c)

Notice that as I am solving a maximization problem, the optimization algorithms in this chapter compute ascent directions instead of descent directions.

### 7.2 Projected Gradient Method

As updates to the control \( \bar{\mathbf{q}} - \alpha \nabla J(\bar{\mathbf{q}}) \) are not necessarily feasible, (i.e. satisfy (7.1.3) and (7.1.4)) the projected gradient method projects the update to the feasible cone by a projection \( P : \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_u} \) defined,

\[
P\mathbf{q} := \arg\min_{\bar{\mathbf{q}} \in \mathbb{R}^{n_u}} \|\bar{\mathbf{q}} - \mathbf{q}\|^2, \quad (7.2.1a)
\]

\[
\text{s.t.} \quad \bar{\mathbf{q}}^T \mathbf{1} = 0, \quad (7.2.1b)
\]

\[
c_l \leq \bar{\mathbf{q}} \leq c_u. \quad (7.2.1c)
\]

The projection (7.2.1) is solved via the Sequential Least Squares Quadratic Program in the SciPy optimization toolbox. The projection is relatively inexpensive as it depends on the number of control variables \( n_u \) which is considerably less than the degrees of freedom in discretized implicit constraint. Finally, as the projection \( P \) must
be applied at every time step $k = 1, \ldots, n_t$, for convenience I introduce an aggregated projection $\bar{P} : \mathbb{R}^{n_u n_t} \to \mathbb{R}^{n_u n_t}$ defined,

$$\bar{P} \bar{q} := \begin{pmatrix} P_{q_1} \\ \vdots \\ P_{q_{n_t}} \end{pmatrix}.$$ 

The projected gradient method is written in Algorithm 7.1.

Algorithm 7.1: Projected Gradient Method

1: while $\| \nabla J(\bar{q}) \| > \text{tol}$ do
2: Compute the gradient $\nabla J(\bar{q})$
3: Define the projection of update $\bar{q} - \alpha \nabla J(\bar{q})$ to the feasible cone,
4: Find $\alpha > 0$ that satisfies the generalized sufficient decrease condition,
5: Update control, $\bar{q} \leftarrow \bar{q} - \alpha \nabla J(\bar{q})$
6: end while

The generalized sufficient decrease condition is called the Generalized Armijo Step-Size Rule in Bertsekas (1976) [8]. To recognize it as the generalization of the Armijo condition, assume $P$ to be the identity mapping. Then substituting $\bar{q}_+(\alpha) - \bar{q} = -\alpha \nabla J(\bar{q})$ into (7.2.2) gives us the Armijo condition when the descent direction is chosen to be the gradient,

$$f(\bar{q}_+ - \alpha \nabla J(\bar{q})) - f(\bar{q}) \leq -c_1 \alpha \| \nabla J(\bar{q}) \|^2.$$ 

Unfortunately, naive extension of the projected gradient method to Newton’s method doesn’t guarantee a descent direction. That is, while the Newton direction $-\nabla^2 J(\bar{q})^{-1} \nabla J(\bar{q})$ is guaranteed to be an descent direction provided the Hessian $\nabla^2 J(\bar{q})$ is symmetric positive definite, the projection $\bar{P}(\bar{q} - \alpha \nabla^2 J(\bar{q})^{-1} \nabla J(\bar{q}))$ can
be an ascent direction. Thus, incorporating second derivative information into the optimization necessitates another approach to the mixed constraints.

### 7.3 An Implementation Newton’s Method with Mixed Constraints

I will treat the inequality and equality constraints separately. First, I will handle the inequality box constraints with a barrier method that penalizes control values near the perimeter of the box constraint. Second, I will accommodate the equality constraints with new projection operation whose range is perpendicular to the ones vector $\mathbf{1}$.

To remove the inequality constraints from the optimization program (7.1.5), I add to the objective function a logarithmic penalty $g : \mathbb{R}^{n_u n_t} \to \mathbb{R}$ defined,

$$g(\bar{q}) := -\sum_{k=1}^{n_t} \log(q_k - c_k)^T \mathbf{1} + \log(c_u - q_k)^T \mathbf{1}.$$  

Then the objective function is modified with some penalty parameter $\gamma > 0$,

$$\min_{\bar{q}} J(\bar{q}) + \gamma g(\bar{q}) \quad (7.3.1a)$$

s.t. $q_k^T \mathbf{1} = 0. \quad (7.3.1b)$

Next, to accommodate the equality constraint, I construct an orthogonal projection $P_E : \mathbb{R}^{n_u} \to \mathbb{R}^{n_u}$ that projects onto the $(n_u - 1)$-dimensional subspace that is perpendicular to the ones vector $\mathbf{1}$,

$$P_E q := \left[ I - \frac{1}{n_u} \mathbf{1} \mathbf{1}^T \right] q.$$  

Since the equality constraint must be satisfied at every time step $k = 1, \ldots, n_t$, I introduce an aggregate projection operator $\tilde{P} : \mathbb{R}^{n_u n_t} \to \mathbb{R}^{n_u n_t}$ defined,

$$\tilde{P}_{E} \bar{q} := \begin{pmatrix} P_E q_1 \\ \vdots \\ P_E q_{n_t} \end{pmatrix}.$$
Then I formulate an unconstrained optimization problem,

$$\min_{\vec{q}} \hat{J}(\vec{q}) := J(\vec{P}_E\vec{q}) + \gamma g(\vec{P}_E\vec{q}).$$  \hspace{1cm} (7.3.2)

The two formulations (7.3.1) and (7.3.2) are not equivalent. In particular, the solution to (7.3.2) is not unique; that is, if $\vec{q}^*$ is a solution to (7.3.1), then $\vec{q}^* + \mathcal{R}(1)$ are solutions in the unconstrained optimization (7.3.2) as well. However, this is not an issue for Newton-CG. To see why, recall that $P_E$ is orthogonal ($P_E = P_E^T$) implies that,

$$\nabla \hat{J}(\vec{q}) = P_E \nabla \left[ J(P_E\vec{q}) + \gamma g(P_E\vec{q}) \right],$$

$$\nabla^2 \hat{J}(\vec{q}) = P_E \nabla^2 \left[ J(P_E\vec{q}) + \gamma g(P_E\vec{q}) \right] P_E.$$

Therefore, gradient and Hessian computations for the unconstrained problem can be computed by projecting gradient and Hessian computations onto the range of $P_E$. Further, since $\nabla \hat{J}(\vec{q}) \in \text{Ran}(\nabla^2 \hat{J}(\vec{q}))$ the system is consistent. Then the Newton subproblem,

$$\nabla^2 \hat{J}(\vec{q})\vec{v} = -\nabla \hat{J}(\vec{q}),$$

for (7.3.2) can be solved via CG as repeated action of the Hessian in the Krylov subspace construction will remain in the range of the projection $\vec{P}_E$. The algorithm I use to compute results in following section is detailed in Algorithm 7.2.
Algorithm 7.2: Newton’s Method with Mixed Constraints

1: while $\|\hat{\nabla} J(\bar{q})\| > \text{tol}$ do
2: Compute the gradient $\nabla J(P_E \bar{q})$.
3: Solve the Newton subproblem via the Conjugate Gradient Method,

$$P_E \nabla^2 [J(P_E \bar{q}) + \gamma g(P_E \bar{q})] P_E \bar{v} = -P_E \nabla J(P_E \bar{q}).$$

Terminate CG when the residual satisfies,

$$\|P_E \nabla^2 [J(P_E \bar{q}) + \gamma g(P_E \bar{q})] P_E \bar{v} + P_E \nabla J(P_E \bar{q})\| < \min\{\|\nabla J(P_E \bar{q})\|^2, 0.01\|\nabla J(P_E \bar{q})\|\}.$$  

4: if $\bar{q} + \bar{v}$ violates inequality box constraints then

$$\alpha \leftarrow 0.95 \min_i \left| \frac{\bar{P}_I[\bar{q} + \bar{v}] - \bar{q}_i}{\bar{v}_i} \right|,$$

$$\bar{P}_I \bar{q} := \begin{pmatrix} \max(c_l, \min(c_u, q_1)) \\ \vdots \\ \max(c_l, \min(c_u, q_n)) \end{pmatrix}$$

5: else

6: $\alpha \leftarrow 1$

7: end if

8: Determine the smallest integer $m \geq 0$ so that $\alpha = (0.5)^m$ satisfies the sufficient decrease condition,

$$J(\bar{q} + \alpha \bar{v}) - J(\bar{q}) \leq c_1 \alpha \nabla \hat{J}(\bar{q})^T \bar{v}.$$

9: Update control, $\bar{q} \leftarrow \bar{q} + \alpha \bar{v}$.

10: end while
7.4 Results

I solve the well rate optimization program with one injector and four producers. The well configuration can be found in Figure 7.2. The optimization is run using the parameters found in Table 7.1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>5000 days</td>
</tr>
<tr>
<td>$\Delta t$</td>
<td>200 days</td>
</tr>
<tr>
<td>$c_l$</td>
<td>$(0, -20, -20, -20, -20)^T$</td>
</tr>
<tr>
<td>$c_u$</td>
<td>$(40, 0, 0, 0, 0)^T$</td>
</tr>
<tr>
<td>$r_{inj}$</td>
<td>$5$ per barrel</td>
</tr>
<tr>
<td>$r_{oil}$</td>
<td>$80$ per barrel</td>
</tr>
<tr>
<td>$r_{oper}$</td>
<td>$5$ per barrel</td>
</tr>
<tr>
<td>$r_{disc}$</td>
<td>$2e-4$</td>
</tr>
</tbody>
</table>

Figure 7.1: Optimization Parameter Values

Figure 7.2: Well configuration: injector denoted with circle, producers denoted with squares.
To compute $\nabla^2 \tilde{J}(\vec{q})\vec{v}$, the Hessian-vector of the unconstrained optimization (7.3.2), the Hessian-vector of the original formulation $\nabla^2 J(\vec{P}_E \vec{q})\vec{v}$ must be computed as an intermediate step. In the results, I use both the FOM Hessian-vector $\nabla^2 J(\vec{P}_E \vec{q})\vec{v}$ utilizing full-order PDE solves, and the ROM Hessian-vector $\nabla^2 \tilde{J}(\vec{P}_E \vec{q})\vec{v}$ in Figure 7.3. Results are shown for a fixed POD tolerance of $10^{-5}$.

The results in Figure 7.3 compute the action of the ROM Hessian using the subspace-based model reduction in Algorithm 4.4. The ROM Hessian calculation I use in these results do not include the DEIM interpolation necessary to fully accelerate the algorithms. Therefore I use the number of PDE solves as a surrogate to cost as can be seen in the third subplot in Figure 7.3. Finally, the two algorithms converge to the same well rate schedule shown in Figure 7.4.

I initialize the optimization with an initial condition at the center of the control space with uniform production across the four producers. That is, $\vec{q}_k^{(0)} = (20, -5, -5, -5, -5)^T$ for all $k = 1, \ldots, n_t$. The Newton-CG algorithm terminates when the norm of the gradient drops four orders of magnitude. Specifically, optimization its terminated when,

$$\|\nabla J(\vec{q})\| < 10^{-4}\|\nabla J(\vec{q}^{(0)})\|,$$

where $\nabla J(\vec{q}^{(0)})$ is the gradient at the initial condition $\vec{q}^{(0)}$. 
Figure 7.3: FOM vs ROM Newton-CG Optimization. Barrier penalty, $\gamma = 10^2$. Convergence rates with FOM and ROM Hessians are similar and converge to the same schedule with 165 and 48 full order PDE solves respectively.
The results above were chosen to explore the efficacy of my approach for a considerably more challenging problem than the semilinear parabolic optimal control problem from the previous chapter. The Hessian-vector computations in these results do not include a DEIM interpolation which is needed to make the ROM Hessian-vector computations independent of degrees of freedom. Therefore, wall clock times are not included and full-order PDE solves are used to approximate relative cost of the two
optimizations. As demonstrated in Figure 7.3, Newton-CG with the FOM and ROM Hessians take 165 and 48 full order PDE solves respectively.

Additionally, we do not observe quadratic convergence when using either the FOM or ROM Hessians. I attribute this to a relative error in the gradient computation of $10^{-8}$ observed via finite difference calculations. There are multiple explanations for this error in the gradient computation. First, the gradient computation depends on the solutions to the nonlinear state equation which is not solved exactly, but up to a desired tolerance. In the results in this chapter, I have set the tolerance in the state solver to $10^{-10}$ in Algorithm 7. As the saturation and pressure equations are solved simultaneously, the relative size of the pressure and saturation variables differ dramatically from each other and also vary significantly throughout the forward simulation. This makes a higher fidelity state solver challenging. Second, upwinding in the saturation equation creates discontinuities in the derivative approximation in finite difference estimates of the gradient.

Figures 7.3 and 7.4 are run for a relatively large barrier penalty, $\gamma = 10^2$ as evidenced by the distance from the box constraints $[-20, 0]$ on the production wells the schedule in Figure 7.4. Ideally, the barrier penalty would be lessened throughout the optimization diminishing the effect of the penalty and allowing the solution to the modified problem (7.3.2) to approach the corners of the box constraints. These preliminary results suggest that a 2nd-order interior-point method may be a promising direction for future research.
Chapter 8

Conclusion

This thesis explores a novel use for model reduction to accelerate Newton-type methods for optimal control problems governed by PDE constraints. My approach differs from conventional approaches that seek to significantly lessen the cost of solving the PDE by replacing it with a surrogate, generating a surrogate problem whose solution is considerably less expensive to compute. However, as I have demonstrated in Chapter 3, generating reduced order models that are consistently representative of their original full order models throughout the optimization is expensive, if not impossible, depending on the application. Further, as optimization algorithms explore a wide range of inputs (such as a wide range of well rate settings), based on my observations it is not clear that reliable reduced order models can replace the full order models in a cost saving way when both online and offline computational costs are considered.

Instead, I propose approximating the PDE solves within Hessian-vector computation via reduced order modeling. Rather than generating a reduced order model to replace the original implicit constraint, I use the structure of Hessian and subspace-based ROMs to compute approximate reduced order Hessian information by recycling data from the full order state and adjoint solves in the gradient computation. While full order state and adjoint solves are costly in large-scale settings, Hessian-vector computations dominate the computational cost of solving the Newton subproblem.
via Conjugate Gradient. The results presented in the previous chapters demonstrate that my approach dramatically accelerates computation times for optimal control problems in two settings: semilinear parabolic implicit constraint and the well rate optimization problem. This overall acceleration includes the training and precomputation of the ROMs that is conventionally discounted as "offline" costs.

**Future Work** As mentioned in the introduction, Trehan and Durlofsky (2016) [53] and Jansen and Durlofsky (2017) [33] combine Proper Orthogonal Decomposition and Trajectory Piecewise Linear (TPWL) approximations to accelerate the well rate optimization. TPWL requires expensive offline precomputations to "train" the approximation to handle various operating scenarios whereas my approach requires expensive online computations of the state and adjoint equations for the gradient computation. It would be interesting to compare the two approaches when both online and offline costs are considered.

Benner et al. (2016) [6] develop a model reduction scheme for quadratic-bilinear control systems. A class of smooth nonlinear systems can be reformulated in this form and become eligible for the $H_2$-based model reduction Benner et al. develop. A future area of research would be to investigate this framework in the context of optimal control or for reservoir simulation.

Additionally, I would like to investigate the well rate optimization problem with smaller barrier penalties to obtain well schedules closer to the boundary. This can introduce numerical challenges as the Hessians can be become ill-conditioned from the large derivatives near the boundary. Potentially, an interior-point method using my ROM Hessian-vector computations may be a future direction for research for well rate optimization and other large-scale optimal problems.

Model reduction in large-scale optimization an exciting area of research with an immediate and major impact in engineering and science. I look forward to continuing this vein of research as it applies to more complex and nuanced challenges in large-scale settings.
Bibliography


