FLORIDA STATE UNIVERSITY

COLLEGE OF ARTS AND SCIENCES

THE SOLUTION OF A BURGERS' EQUATION INVERSE PROBLEM WITH REDUCED-ORDER MODELING PROPER ORTHOGONAL DECOMPOSITION

By

JEFF STEWARD

A Thesis submitted to the
Department of Scientific Computing
in partial fulfillment of the
requirements for the degree of
Master of Science

Degree Awarded:
Summer Semester, 2009

The members of the committee approve the dissertation of Jeff Steward defended on May 20, 2009.

Ionel M. Navon
Professor Directing Thesis

Max Gunzburger
Committee Member

Gordon Erlebacher
Committee Member

Approved:

Max Gunzburger, Chair, Department of Scientific Computing

Joseph Travis, Dean, College of Arts and Sciences

The Graduate School has verified and approved the above-named committee members.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

This thesis presents and evaluates methods for solving the 1D viscous Burgers' partial differential equation with finite difference, finite element, and proper orthogonal decomposition (POD) methods in the context of an optimal control inverse problem. Based on downstream observations, the initial conditions that optimize a lack-of-fit cost functional are reconstructed for a variety of different Reynolds numbers. For moderate Reynolds numbers, our POD method proves to be not only fast and accurate, it also demonstrates a regularizing effect on the inverse problem.

CHAPTER 1

INTRODUCTION

Consider an in general nonlinear dynamical partial differential equation (PDE) system for $u(\mathbf{x}, t)$

$$\begin{array}{rcll} \mathcal{L}u & = & f(\mathbf{x}, t), & \mathbf{x} \in \Omega, \quad t \in (t_0, t_f] \\ \mathcal{R}u & = & g(\mathbf{x}, t), & \mathbf{x} \in \partial\Omega, \quad t \in (t_0, t_f] \\ u & = & u_0(\mathbf{x}), & \mathbf{x} \in \bar{\Omega}, \quad t = t_0 \end{array} \tag{1.1}$$

posed over a space H appropriate for the operator. Reduced-Order Modeling (ROM) aims to accurately approximate u using as few degrees of freedom as possible. In this process, there is a trade off between speed, accuracy and degrees of freedom.

In solving (1.1) via finite difference, finite volume, finite element or spectral computational methods, both experience and numerical analysis theory show that as we reduce the mesh size we will get increasingly accurate solutions (up to round off error) at the cost of additional computing time. In situations such as optimal control problems we may wish to compute a model solution many times. At first glance such a problem would seem to require either: a) an inordinate amount of time to solve, or b) the sacrifice of accuracy to reduce the computational load. However, using proper orthogonal decomposition (POD) model reduction suited for nonlinear PDEs, we can reuse information about previous model runs to find a low-order basis that best describes the sampled model dynamics. The model can thus be executed many times quickly without an excessive loss of accuracy.

In this work we will consider an optimal control inverse problem to reconstruct an initial condition $u(\mathbf{x}, t_0) = u_0(\mathbf{x})$ given downstream observations u_{obs} at grid points $\mathbf{x}_{obs}$. Optimal control is an extension of the calculus of variations first formulated by Pontryagin [1] in 1962, Bellman [2] in 1957, and Lions [3] in 1968, translated in 1971 [4]. As we will see, a good and fast approximation is very useful in this context.

As a simple test case, we take $\mathbf{x}_{obs}$ to be N spatial locations at the final time t_f. We will seek to minimize the cost functional $J(u_0)$ subject to the constraint (1.1)

$$J(u_0) = \sum_{i=1}^{N} (u_{obs}^i - u_f^i)^2 \quad \text{s.t } u(\mathbf{x}, t|u_0) \text{ satisfies (1.1)} \tag{1.2}$$

where

- $u_0 = u_0(\mathbf{x})$ is the unknown initial condition
- u_{obs}^i is the i^{th} observation at time t_f at $\mathbf{x}_i$
- $u(\mathbf{x}, t_f|u_0)$ is the model solution of (1.1) at time t_f with u_0 initial condition
- $u_f^i = u(\mathbf{x}_i, t_f|u_0)$ is the model solution at $\mathbf{x}_i$ at time t_f

This cost functional measures the lack-of-fit between our model solution at time t_f and our observations. We will assume our model to be perfect, i.e. we have our model as a strong constraint [5]. Furthermore, our observations are available at the space and time mesh locations, obviating the need for an observation interpolation operator. We will discretize our model and, using an adjoint of J, seek the optimal u_0 to describe the observations. This will require many model integrations, both forward and backward. Each model integration may take an extremely large amount of computational effort. We will therefore first compare the numerical efficiency of a variety of finite difference and finite element approaches in chapter 2. Since each of these methods is computationally intensive, accurate reduced order modeling will prove advantageous. In this study we will use Proper Orthogonal Decomposition, one of the most popular reduced order modeling techniques, to solve such an inverse problem for the 1D viscous Burgers' equation. In chapter 3 we will present the numerical results of these various methods with an eye to comparing them to POD. Finally, in chapter 4 we will consider three different techniques for minimizing (1.2), comparing and contrasting these methods. We will then study the effect of the Reynolds number on the numerical accuracy of our method.

1.1 Proper Orthogonal Decomposition

There are many techniques for model reduction. Two such methods are balanced truncation and global eigenmodes [6]. Balanced truncation is only available for linear systems [7], while the global eigenmodes method is prohibitively expensive for large systems [8]. Among the various competitors for model reduction, Proper Orthogonal Decomposition (POD) has emerged as the leading contender for a low-dimensional description of nonlinear partial differential equations (PDEs) [6].

The POD technique is also known in other contexts as Principal Component Analysis, the discrete (Kosambi-)Karhunen-Loève transform, and the Hotelling transform. When time series data as well as spatial data is used, the procedure is called the method of Empirical Orthogonal Functions. One imagines that the prevalence of different names in different application areas is an indicator of the quality of the original idea.

The basic POD technique, credited first to Pearson in 1901 [9], essentially decomposes a set of zero mean data into an orthogonal coordinate system such that the maximum amount of variance between data points is projected onto the first coordinate, the second maximum amount of variance is projected onto the second coordinate, and so on [10]. One can then truncate the number of dimensions used to describe the problem while still maintaining an optimal amount of information about the full system. This optimal property will be shown in more detail in section 1.1.1.

In the context of reduced-order modeling, finding these coordinates becomes equivalent to finding the discrete (or empirical) eigenfunctions of the system which will then be used as a low-dimensional basis within a Galerkin projection. This approach stands in opposition to finite difference, finite element, and spectral methods where the basis functions used are not related to the PDE in any way.

Before we can find our basis functions, however, we first require accurate solutions of the full model sampled around various parameters. As first described by Sirovich in 1987 [11], we collect model solutions at various times for various parameters into a matrix of *snapshots*. The redundant information is extracted from our snapshot matrix using an eigenvalue or singular value decomposition and the eigenvectors found become our discrete approximations to the most energetic eigenfunctions of the system.

As we will describe in further detail in the next section, the POD basis captures more energy than any other linear basis. This allows us to describe the dynamics of our system with as few degrees of freedom as possible. POD is thus a good technique for solving systems such as (1.1) quickly and accurately. Studies show POD reduces model integration time by 60%-97% while maintaining a large degree of accuracy [12], [13].

However, POD is not without its weaknesses. Depending on how many modes are captured, small-order details can be lost. These small order details may lead to changes in global behavior, especially over a long period of time. Furthermore, the accuracy of the POD method is highly dependent upon on the quality of the snapshots, and if the model behavior is not present in the snapshots, the POD system will be unable to accurately reproduce this behavior [14]. While the quality of the snapshots depends upon the sampling of the parameter space, POD sampling methods are currently "a primitive art" [15]. Better methods are needed to address this issue. Finally, another issue for the POD method is that while a posteriori error bounds are available [16], no a priori error bounds currently exist. Despite these short-comings, POD remains a method of choice for reduced order modeling due to its (as a rule) highly accurate and efficient numerical solutions.

1.1.1 POD Optimality

In this section we will prove the optimality of POD as a linear basis. This analysis closely follows the work of Volkwein [17].

Suppose we are given snapshots $y_1, \dots, y_n \in \mathbb{R}^m$. Set $V = \text{span}\{y_1, \dots, y_n\}$. It is now our goal to find $\ell \leq \dim V$ orthonormal vectors $\Psi = \{\psi_i\}_{i=1}^{\ell}$ in $\mathbb{R}^m$

$$\arg\min_{\Psi} f(\Psi) = \sum_{j=1}^{n} ||y_j - \sum_{i=1}^{\ell} (y_j, \psi_i)\psi_i||^2 \text{ s.t. } (\psi_i, \psi_j) = \delta_i^j$$

For convenience, we will assume a Euclidean norm and a classical (dot) inner product.

In terms of a Lagrangian with multipliers $\Lambda = \{\lambda_{ij}\}_{i,j=1}^{\ell}$, this system is

$$\mathcal{L}(\Psi, \Lambda) = f(\Psi) + \sum_{i,j=1}^{\ell} \lambda_{ij} \left((\psi_i, \psi_j) - \delta_i^j \right)$$

Taking the partial derivative of the Lagrangian and analyzing the KKT conditions [18], we see that the optimality conditions for this system are

$$\begin{array}{ll} \frac{\partial \mathcal{L}}{\partial \psi_i}(\psi_i, \dots, \psi_j, \lambda_{11}, \dots, \lambda_{\ell\ell}) = 0 & \text{for } i = 1, \dots, \ell \\ \frac{\partial \mathcal{L}}{\partial \lambda_{ij}}(\psi_i, \dots, \psi_j, \lambda_{11}, \dots, \lambda_{\ell\ell}) = 0 & \text{for } i, j = 1, \dots, \ell \end{array}$$

From this, we have

$$\frac{\partial \mathcal{L}}{\partial \psi_i} = 0 \Leftrightarrow \sum_{j=1}^{n} (y_j, \psi_i) y_j = \lambda_{ii} \psi_i$$

$$\frac{\partial \mathcal{L}}{\partial \lambda_{ij}} = 0 \Leftrightarrow (\psi_i, \psi_j) = \delta_i^j$$

Collecting $\lambda_{ii} = \lambda_i$, $Y = [y_1, \ldots, y_n] \in \mathbb{R}^{m \times n}$ gives

$$YY^T \psi_i = \lambda_i \psi_i$$

In other words, solving the $m \times m$ [11] eigenvalue problem gives the necessary optimality conditions for Ψ that minimize $\mathcal{L}$ over all linear basis functions. By KKT, these conditions are also sufficient [17].

1.1.2 Properties of POD

To compute Ψ, we can solve the rank d truncated singular value decomposition (SVD) for $Y \in \mathbb{R}^{m \times n}$ for $1 \leq d \leq \text{rank } Y$. Here $\sigma_1 \geq \ldots \geq \sigma_d > 0$. For the orthogonal matrices $U = [u_1, \ldots, u_m] \in \mathbb{R}^{m \times m}$ and $V = [v_1, \ldots, v_n] \in \mathbb{R}^{n \times n}$, we have

$$U^T Y V = \begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix} = S \in \mathbb{R}^{m \times n}$$

where $D = \text{diag}(\sigma_1, \ldots, \sigma_d) \in \mathbb{R}^{d \times d}$. This gives, equivalently, all of the following:

$$Yv_i = \sigma_i u_i, Y^T u_i = \sigma_i v_i, YY^T u_i = \sigma_i^2 u_i, Y^T Y v_i = \sigma_i^2 v_i$$

The i^{th} POD basis vector $\psi_i = u_i$ is given by the i^{th} left singular vector of Y.

We note that some studies such as [19] and [20] prefer to compute the POD basis on the covariance matrix

$$C_{ij} = \frac{1}{n} \int_\Omega y_i y_j \, dx$$

This covariance method was tested in this study and found to be significantly slower while offering very little or no increase in accuracy. Therefore the approach described above, advocated in studies such as [21], is chosen.

In a generalized orthonormal Fourier series, a function g square integrable over the domain Ω can be written as

$$g(x,t) = \sum_{i=1}^{\infty} c_i(t) \phi_i(x)$$

where $c_i(t) = \int_\Omega g(x,t)\phi_i(x) \, dx$ and $\{\phi_i(x)\}_{i=1,2,\ldots}$ form an orthogonal set of basis functions.

Note that if we take our $\{\phi_i(x)\}$ to be orthonormal eigenfunctions of the infinite dimensional system (1.1), we have the infinite dimensional Kosambi [22]-Karhunen [23]-

Loève [24] transform. This transform is useful for theoretical analysis on stochastic processes such as the Wiener process [24]

If, however, we use only a finite number ℓ of basis functions computed from the eigenvectors of discrete model realizations as a finite dimensional basis, we have the POD method which is also called the *empirical* Kosambi-Karhunen-Loève transform due to its dependence on the snapshots.

We add that due to this close association between POD and the generalized orthonormal Fourier series, it is no surprise that a variant of the Parseval identity [25] holds

$$f(\psi_1, \ldots, \psi_\ell) = \sum_{j=1}^{n} ||y_j - \sum_{i=1}^{\ell} (y_j, \psi_i)\psi_i||^2 = \sum_{i=\ell+1}^{d} \lambda_i \tag{1.3}$$

In other words, the amount of misfit between the snapshots and the basis functions is the magnitude of the neglected singular values. Note again that this only refers to values within the set of snapshots: other model dynamics outside of the snapshots Y need not be approximated accurately by POD. This again underscores the importance of choosing our snapshots carefully.

Using (1.3), we can choose to keep $1 \leq \ell \leq D$ basis vectors based on a prescribed error tolerance γ such that

$$\frac{\sum_{j=1}^{\ell} \sigma_j^2}{\sum_{j=1}^{D} \sigma_j^2} \geq 1 - \gamma$$

where $D = \operatorname{rank} Y$. This ℓ rank basis will capture $100(1-\gamma)\%$ of the energy of the system [26].

1.2 Burgers' Equation

In this work we will consider the viscous one dimensional Burgers' equation

$$\begin{array}{rcllr} \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} & = & \mu\frac{\partial^2 u}{\partial x^2}, & x \in \Omega, & t \in (t_0, t_f] \\ \mathcal{R}u & = & g, & x \in \partial\Omega, & t \in (t_0, t_f] \\ u & = & u_0(x), & x \in \bar{\Omega}, & t = t_0 \end{array} \tag{1.4}$$

This equation is very useful in the context of fluids because, like Navier-Stokes, Burgers' has [27]:

- u_t - an unsteady term
- uu_x - a nonlinear convective term

- μu_{xx} - a viscous term

In fact, we can arrive at Burgers' equation from the Navier-Stokes equation by:

- Considering only the x component,
- Neglecting the pressure, and
- Neglecting the forcing term of Navier-Stokes

Much research and effort has gone into solving and analyzing this simple PDE. Beginning with the seminal work of Burgers [28], many analytic and numerical methods have been developed to work with this equation. Fletcher has an excellent, if by now somewhat outdated, survey of the various methods and applications of Burgers' equation [29]. More recent surveys of numerical methods include [27] and [30].

1.2.1 Cole-Hopf transform

Cole [31] and Hopf [32] discovered a substitution for Burgers' equation that transforms (1.4) to the linear heat equation. The Cole-Hopf transformation is given by

$$u(x,t) = -2\mu \frac{\theta_x(x,t)}{\theta(x,t)} \tag{1.5}$$

for $\theta(x,t) \neq 0$, and by this transformation, Burgers' equation becomes

$$\frac{\partial \theta}{\partial t} = \mu \frac{\partial^2 \theta}{\partial x^2}$$

However, this complicates the boundary conditions and initial conditions as we shall see shortly. Nonetheless, this transformation allows us to derive exact solutions for the 1-D Burgers' equation for many general cases

1.2.2 Exact solution

Consider in particular homogeneous Dirichlet boundary conditions

$$\begin{array}{rcll} \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} & = & \mu\frac{\partial^2 u}{\partial x^2}, & x \in \Omega, \quad t \in (t_0, t_f] \\ u & = & 0, & x \in \partial\Omega, \quad t \in (t_0, t_f] \\ u & = & u_0(x), & x \in \bar{\Omega}, \quad t = t_0 \end{array} \tag{1.6}$$

By the Cole-Hopf transformation, we have $u(x,t) = 0 = -2\mu\frac{\theta_x(x,t)}{\theta(x,t)}$, $x \in \partial\Omega$. Therefore, our boundary conditions become the Neumann conditions $\theta_x(x,t) = 0$ for $x \in \partial\Omega$. Solving a separable ODE for our initial conditions, we find $\theta(x,0) = \theta_0(x) = \exp\left(-\frac{1}{2\mu}\int_\alpha^x u_0(\xi)\ d\xi\right)$

Together, this gives

$$\begin{array}{rcll} \frac{\partial\theta}{\partial t} &=& \mu\frac{\partial^2\theta}{\partial x^2}, & x \in\ \Omega,\quad t \in (t_0, t_f] \\ \frac{\partial\theta}{\partial x} &=& 0, & x \in\partial\Omega,\quad t \in (t_0, t_f] \\ \theta &=& \exp\left(-\frac{1}{2\mu}\int_\alpha^x u_0(\xi)\ d\xi\right), & x \in\ \bar{\Omega},\quad t = t_0 \end{array} \tag{1.7}$$

where α is the left boundary of $\partial\Omega$.

Following Wood [33], we can now find an exact solution to test our numerical methods against. We choose $\Omega = (0,1)$. Since we have $\theta_x(0,t) = \theta_x(1,t) = 0$, we take $\theta(x,t) = a + \exp(-\pi^2\mu t)\cos(\pi x)$ for $a > 1$. This choice satisfies the boundary condition and leads to a closed-form initial condition for u_0.

By the Cole-Hopf transform, we have

$$u(x,t) = \frac{2\mu\pi\exp(-\pi^2\mu t)\sin(\pi x)}{a + \exp(-\pi^2\mu t)\cos(\pi x)} \tag{1.8}$$

and the corresponding initial condition is

$$u_0(x) = \frac{2\mu\pi\sin(\pi x)}{a + \cos(\pi x)} \tag{1.9}$$

We will primarily use the exact solution (1.8) with initial condition (1.9) for testing the methods that follow.

CHAPTER 2

NUMERICAL METHODS

In this chapter we will describe various methods for solving Burgers' equation in order to test the efficiency of POD. The methods tested include:

- Finite differences:
  - First-order implicit Rusanov (Burstein-Murin) method
  - The Roe scheme
  - MacCormack method
- Finite elements:
  - Fast Linear Discrete-time FEM
  - Arbitrary polynomial degree Discrete-time FEM
  - Arbitrary polynomial degree Continuous-time FEM
- POD:
  - Discrete-time POD based on FEM
  - Continuous-time POD based on FEM

Here, "discrete-time" and "continuous-time" approaches refer to how our equation is discretized in time. The details are given below.

After describing these methods, we will then compare each of them to the exact solution (1.8) in order to compute the L^2, H^1, and RMSE errors of each method versus the total amount of time necessary.

More explicative emphasis will be given to the finite element methods due to their similarities with the POD method.

2.1 Finite Difference Methods

For all of the finite difference methods tested, we assume a uniformly spaced grid of $N \times M$ points with a spacing of Δx and Δt in the x and t directions respectively. For $i = 0, \dots, N$, $j = 0, \dots, M$, let $x_i = i\Delta x$ and $t_j = j\Delta t$. Our approximation at the grid point (i, j) is denoted by $u_i^j = u^h(x_i, t_j) \approx u(x_i, t_j)$. We utilize the boundary conditions by setting $u_0^j = u(0, t_j) = 0$ and $u_N^j = u(1, t_j) = 0$ and the initial condition by $u_i^0 = u_0(x_i)$

We now discuss each individual method tested.

2.1.1 Rusanov method

The Rusanov method, first implemented by Taylor in 1972 [34], is first order accurate in space and time [29]. It is an implicit method. At time step n, we solve for $u^{n+1} =$ with the following

$$Au^{n+1} = b$$

where

$$A_{ij} = 1 + 2r$$

for $i = j$ (that is, on the diagonal), and

$$A_{ij} = -r$$

for $|i - j| = 1$ (that is, on the sub- and super-diagonals), where $r = \mu\frac{\Delta t}{\Delta x^2}$, and

$$b_i = u_i^n - 0.5\frac{\Delta t}{\Delta x}(F_{i+1}^n - F_i^n) + 0.5w\frac{\Delta t}{\Delta x}(\phi_{j+1}^n - \phi_{j-1}^n)$$

where $F_i^n = 0.5(u_i^n)^2$, $\phi_j^n = 0.5(u_i^n + u_{i-1}^n)(u_i^n - u_{i-1}^n)$ and w is an artificial viscosity parameter set to 0.5 for our tests.

The Rusanov method is stable for $\mu \leq 1$ and Courant numbers less than 1 [27].

The results of this method for various stable values of Δx and Δt are shown in table 3.1.

2.1.2 Roe Method

The Roe Method, developed in 1980 [35] and 1981 [36], is second order accurate in space and first order accurate in time. It is an explicit method given by:

$$u_j^{n+1} = u_j^n - 0.5\frac{\Delta t}{\Delta x}(F_{j+1}^n - F_j^n - \phi_{j+1}^n + \phi_j^n) + r(u_{j+1}^n - 2u_j^n + u_{j-1}^n)$$

where $F_i^n = 0.5(u_i^n)^2$, $\phi_j^n = 0.5|u_i^n + u_{i-1}^n|(u_i^n - u_{i-1}^n)$ and $r = \mu\frac{\Delta t}{\Delta x^2}$

This method is stable as long as the Courant number is less than 1 [27].

The results of this method are shown in table 3.2.

2.1.3 MacCormack Method

The MacCormack method, developed by MacCormack in 1969 [37] is second order accurate in space and time. It is a predictor-corrector method. The predictor is:

$$u_j^{\overline{n+1}} = u_j^n - \frac{\Delta t}{\Delta x}(F_{j+1}^n - F_j^n) + r(u_{j+1}^n - 2u_j^n + u_{j-1}^n)$$

where $F_i^n = 0.5(u_i^n)^2$ and $r = \mu\frac{\Delta t}{\Delta x^2}$, and the corrector is

$$u_j^{n+1} = \frac{1}{2}\left[u_j^n + u_j^{\overline{n+1}} - \frac{\Delta t}{\Delta x}(F_j^{\overline{n+1}} - F_{j-1}^{\overline{n+1}}) + r(u_{j+1}^{\overline{n+1}} - 2u_j^{\overline{n+1}} + u_{j-1}^{\overline{n+1}})\right]$$

As with our other methods, the MacCormack method is stable as long as the Courant number is less than 1 [27].

The results of this method are shown in table 3.3.

2.2 Finite Element Methods

Starting from the 2003 approach by Öziş [38], we compute the finite element method on our linearized equation (1.7). This approach is relatively easy to implement and avoids the necessity of other linearization techniques for (1.6). However, since after solving for our approximation $\theta^h(x,t)$ we will use the Cole-Hopf transform (1.5) to find $u^h(x,t)$, this approach will in general lose one degree of accuracy due to the derivative term in (1.5). It is thus important to use higher order FEM methods as discussed in this study. Using higher order methods to solve (1.7) and then applying (1.5) is, to the author's best knowledge, a new approach for solving (1.6).

2.2.1 Weak form

We begin multiplying (1.7) by an arbitrary test function $v(x) \in H(0,1)$, where $H(0,1)$ is the complete space of piecewise continuous functions that have at least one piecewise continuous derivative over the $(0,1)$ domain. Since we are working with homogeneous Neumann conditions, our boundary conditions are natural rather than essential.

This gives

$$\int_0^1 v\frac{\partial\theta}{\partial t} - v\mu\frac{\partial^2\theta}{\partial x^2}\ dx = 0$$

Integrating by parts, we have

$$\int_0^1 v\frac{\partial\theta}{\partial t} + \mu\frac{\partial\theta}{\partial x}\frac{dv}{dx}\ dx = \mu\left(v(1)\frac{\partial\theta}{\partial x}(1,t) - v(0)\frac{\partial\theta}{\partial x}(0,t)\right)$$

By the boundary condition $\frac{\partial\theta}{\partial x}(1,t) = 0$, $\frac{\partial\theta}{\partial x}(0,t) = 0$, we have

$$\int_0^1 v\frac{\partial\theta}{\partial t} + \mu\frac{\partial\theta}{\partial x}\frac{dv}{dx}\ dx = 0 \tag{2.1}$$

A solution of (2.1) is a *weak* solution to (1.7). In other words, a classical solution of (1.7) will also satisfy (2.1), but a solution of (2.1) will only satisfy (1.7) if it is continuous.

2.2.2 Discretization

We discretize our domain with N evenly spaced intervals. We then add $p-1$ evenly spaced points to the interior of each interval, giving the points x_i, $i = 0, \ldots, Np$. Here, p is the degree of the polynomial space we will use for our finite element approximation.

We now take a finite dimensional subset of $S^h \subset H(0,1)$ and choose a basis of piecewise polynomials of degree p for S^h. We denote our basis functions by $\{\phi_i\}_{i=0,\ldots,Np}$. Each ϕ_i is associated with the point x_i and has support over one or two intervals. In particular, basis functions corresponding to the endpoints of the original N intervals have support over both adjacent intervals (with the exception of ϕ_0 and ϕ_{Np}, which have no support outside the domain $(0,1)$), while each basis function ϕ_i corresponding to an interior point has support only over the one original interval that x_i is contained within.

Furthermore, we stipulate that our basis is *nodal* - in other words $\phi_j(x_i) = 1$ for $i = j$ and $\phi_j(x_i) = 0$ for $i \neq j$ for x_i in the support of ϕ_j. We can thus write $\theta^h(x,t) = \sum_{i=0}^{Np} c_i(t)\phi_i(x)$. One immediately apparent benefit of this formulation is that each coefficient $c_i(t)$ corresponds to our model's approximation at x_i at time t, *i.e.*

$$c_i(t) = u^h(x_i, t) \tag{2.2}$$

2.2.3 Matrix form

Our equation (2.1) now becomes

$$\int_0^1 \left(v \sum_{i=0}^{Np} c_i'(t)\phi_i(x) + \mu \sum_{i=0}^{Np} c_i(t)\phi_i'(x)v'(x) \right) \, dx = 0$$

Rewriting, we have

$$\sum_{i=0}^{Np} c_i'(t) \int_0^1 v(x)\phi_i(x) \, dx + \mu \sum_{i=0}^{Np} c_i(t) \int_0^1 \phi_i'(x)\frac{dv}{dx} \, dx = 0$$

If we now take v to be a basis function $\phi_j(x)$, we have

$$\sum_{i=0}^{Np} c_i'(t) \int_0^1 \phi_i(x)\phi_j(x) \, dx + \mu \sum_{i=0}^{Np} c_i(t) \int_0^1 \phi_i'(x)\phi_j'(x) \, dx = 0$$

Furthermore, if we take $v = \phi_j(x)$ for $j = 0, \ldots, Np$, we have the system of equations

$$Mc'(t) = -Kc(t), \quad M_{ij} = \int_a^b \phi_i(x)\phi_j(x) \, dx, \quad K_{ij} = \int_a^b \phi_i'(x)\phi_j'(x) \, dx$$

Thus, our solution to problem (2.1) is found by solving an ordinary differential equation.

Here, M is our *mass* matrix and K is our *stiffness* matrix. By our choice of basis, it can be shown that both our mass and stiffness matrices are sparse, positive definite and symmetric [38]. We solve these integrals by using Gaussian quadrature of order $2p-1$ in order to compute these matrices "exactly." [39].

2.2.4 Discrete in time method

We consider two methods for solving this ODE. The first is to discretize the problem in time as well as space.

For this method, we approximate $c(t)$ with $c^j \approx c(j\Delta t)$ and $c'(t)$ with the finite difference $c'(t) \approx \frac{1}{\Delta t}(c^{j+1} - c^j)$. This gives

$$\frac{1}{\Delta t} M(c^{j+1} - c^j) + \mu K(\gamma c^{j+1} + (1-\gamma)c^j) = 0$$

where $0 \leq \gamma \leq 1$ is a weight factor corresponding to

- $\gamma = 0$: the explicit first-order in time conditionally stable forward-Euler method

- $\gamma = 1$: the implicit first-order in time unconditionally stable but dissipative backward-Euler method
- $\gamma = 0.5$: the implicit second-order in time unconditionally stable Crank-Nicolson method

In this study we use the Crank-Nicolson method exclusively since we are seeking high degrees of accuracy.

We can now rewrite our equation

$$Dc^{j+1} = Ec^j$$

where

$$D = M + \mu\gamma\Delta t K$$

and

$$E = M - \mu(1-\gamma)\Delta t K$$

Thus, if we compute D and E once, we can advance our system in time simply by solving $Ac^{j+1} = b$, where $A = D$ and $b = Ec^j$. Since D is sparse, symmetric and positive definite, we can use a sparse Cholesky solver to efficiently compute this solution. MATLAB® provides an efficient solver for this purpose in the form of the `chol(A,'lower')` command.

The results of this method are shown in tables 3.5 through 3.7.

2.2.5 Finite Element - Continuous in time method

Our second method is to consider our ODE as a continuous problem. Rather than discretize our problem, we can solve the ODE

$$Mc'(t) = -Kc(t)$$

for $c(t)$ using an ODE solver. While of necessity our solver will also view the ODE in a discrete way, in contrast to the finite difference in time approach we will allow adaptive step sizes and thus we will not directly create a grid in time.

For this we utilize the ODE solvers available in MATLAB® :

- Explicit Runge-Kutta (4,5) Dormand-Prince, non-stiff `ode45`

- Explicit Runge-Kutta (2,3) Bogacki and Shampine, non-stiff `ode23`
- Adams-Bashforth-Moulton, non-stiff `ode113`
- Numerical Differentiation Formula, stiff `ode15s`
- Modified Rosenbrock formula of order 2, stiff `ode23s`

Our numerical studies show that `ode15s` works best for this problem.

The results of this method are shown in tables 3.8 through 3.10.

2.2.6 Methods for Cole-Hopf transform

Once we have found $\theta^h(x,t)$ by solving (1.7) we must apply the inverse Cole-Hopf substitution to find $u^h(x,t)$ by the formula

$$u^h = -2\mu \frac{\theta_x}{\theta}$$

Öziş [38] approximates θ_x with the second-order accurate central difference

$$\theta_x(x_i, t_j) \approx \frac{\theta_{i+1}^j - \theta_{i-1}^j}{2\Delta x}$$

For $p = 1$, *i.e.* piecewise linears, this is an excellent choice. For $p > 1$, however, this choice limits our accuracy to second order in the spatial domain, thus negating any advantage of using higher order polynomials. Instead, in this study for $p > 1$ we approximate $\theta_x(x,t)$ with the finite element derivative

$$\theta_x(x_i, t_j) = \sum_{i=1}^{Np} c_i(t)\phi_i'(x)$$

This approximation is accurate up to order p. Therefore we expect to have total accuracy of order p in the L^2 norm and $p-1$ in the H^1 norm.

2.3 Proper Orthogonal Decomposition

Using one of the above approaches as a solver for our model, we compute N_s column vectors y_i at various times and parameters and place them in

$$Y = \begin{pmatrix} y_1 & \dots & y_{N_s} \end{pmatrix} \in \mathbb{R}^{Np+1 \times N_s}$$

Here each column vector y_i represents one snapshot in time of the system evolution around a particular parameter at each of our $Np+1$ grid points. We now compute the mean along the rows, *i.e.* the time-averaged mean along a particular grid point x_i, by

$$u_i^m = \frac{1}{N_s} \sum_{j=1}^{N_s} Y(i,j)$$

From this, we can compute the mean-subtracted ensemble Y^M by $Y_{ij}^M = Y_{ij} - u_i^m$ and take the singular value decomposition $Y^M = [U \Sigma V]$. Our $1 \le j \le \ell \le \text{rank}(Y)$ basis functions $\psi_j(x)$ are computed by the left singular vector columns U^i. In a FEM setting, these singular vectors become the coefficients of the FEM nodal polynomial basis, *i.e.*

$$\psi_j(x) = \sum_{i=0}^{Np} U_j^i \phi_i(x)$$

For future reference, we also define the projection from the full space to the reduced space by

$$\mathbb{P}(y) = \mathcal{A}^{-1} \vec{c}(y - u^m) \tag{2.3}$$

Where $\mathcal{A}_{ij} = \int_0^1 \psi_i \psi_j \, dx$ for $1 \le i, j \le \ell$ and $\vec{c}_i(y(x,t)) = \int_0^1 y(x,t)\psi_i \, dx$, $1 \le i \le \ell$, and our inverse projection from the reduced space to the full space

$$\mathbb{Q}(c^r) = \Psi^T c^r + u^m \tag{2.4}$$

where $\Psi = \left(\psi_1, \ldots, \psi_\ell\right)$ and c^r is our reduced space approximation. By this formulation and the properties of POD, [40] the error

$$\int_0^{t_f} y(x,t) - \mathbb{Q}(\mathbb{P}(y(x,t))) \, dt$$

is minimized over all rank ℓ projections.

In a finite difference setting these singular values can be transformed to continuous basis functions through interpolation, although in this study only finite element-based POD methods were considered.

2.3.1 Matrix form

We now can express our reduced-order solution as

$$u^r(x,t) = u^m(x) + \sum_{i=1}^{\ell} c_i^r(t)\psi_i(x)$$

Putting our approximation $u^r(x,t)$ into our weak form (2.1) gives

$$\int_0^1 \left(v \sum_{i=1}^{\ell} (c_i^r)'\psi_i + \mu \left((u^m)' + \sum_{i=1}^{\ell} c_i^r \psi_i' \right) v' \right) \, dx = 0$$

As with the finite element method, if we take $v = \psi_j$ we have

$$\sum_{i=1}^{\ell} (c_i^r)' \int_0^1 \psi_i \psi_j \, dx + \mu \left(\int_0^1 (u^m)'\psi_j' \, dx + \sum_{i=1}^{\ell} c_i^r \int_0^1 \psi_i'\psi_j' \, dx \right) = 0$$

In matrix form for $v = \psi_j$, $j = 1, \ldots, \ell$, we have

$$M^r(c^r)' = -\mu(K^r c^r + F), \quad \begin{aligned} M_{ij}^r &= \textstyle\int_0^1 \psi_i(x)\psi_j(x) \, dx, \\ K_{ij}^r &= \textstyle\int_0^1 \psi_i'(x)\psi_j'(x) \, dx \\ F_j &= \textstyle\int_0^1 (u^m)'\psi_j' \, dx \end{aligned}$$

M^r and K^r are no longer sparse (the orthonormality of our basis functions holds only in the classical inner product), but these matrices are still symmetric and positive definite. Symmetry is clear from our formulation while positive definiteness follows from the linear independence of the eigenvectors.

In order to compute $(u^m)'$, needed to compute F_j, we use the FEM formulation $f(x) = \sum_{i=0}^{Np} u_i^m \phi_i(x)$ to give $f'(x) = \sum_{i=0}^{Np} u_i^m \phi_i'(x)$. We now have

$$F_j = \int_0^1 f'\psi_j' \, dx$$

2.3.2 Discrete in time method

Once again, we reuse our FEM derivation to find the discrete-time method

$$D^r(c^r)^{j+1} = E^r(c^r)^j - \mu\Delta t F$$

where

$$D^r = M^r + \mu\gamma\Delta t K^r$$

and

$$E^r = M^r - \mu(1-\gamma)\Delta t K^r$$

and γ gives the same methods as the FEM case.

Once again we can advance the system in time by solving the equation $A(c^r)^{j+1} = b$ for $A = D^r$ and $b = E^r(c^r)^j - F$ using a Cholesky based solver, although in this case no sparsity can be exploited. The main advantage here, however, is that $\dim(D^r) << \dim(D)$, where D is the equivalent finite element matrix.

2.3.3 Continuous in time method

As with FEM, we can directly solve the ODE

$$M^r(c^r)' = -\mu(K^r c^r - F)$$

for $c^r(t)$, and again as with FEM, the stiff MATLAB® solver `ode15s` proved to work best in numerical tests.

CHAPTER 3

NUMERICAL RESULTS

We now present results that summarize our comparison of these different numerical methods. We present these results at $\mu = .1$ (i.e. a Reynolds number of 10), a potentially more difficult numerical setting than $\mu = 1$.

The exact solution for (1.8) is shown in figure 3.1. An approximate solution for $p = 8$, $\Delta x = 1/4$ and $\Delta t = 10^{-13}$ for the CFEM method is shown in figure 3.2. Since the two are visually indistinguishable, the error is shown in figure 3.3. At some point for each of the methods, a Δx or Δt that is too large leads to less accuracy due to numerical round-off. This situation is shown in figure 3.4, which represents the same situation as figure 3.3 except a $\Delta x = 1/8$ is now used. Note how the numerical noise begins to dominate.

The H^1, L^2 and RMSE errors of the various methods versus the time necessary to find the solution are shown in figures 3.5-3.7. The H^1 error is given by

$$||u^h - u||_1 = \int_0^1 u^h + \frac{du^h}{dx} - u - \frac{du}{dx}\; dx$$

Our L^2 error is given by

$$||u^h - u||_0 = \int_0^1 u^h - u\; dx$$

and our RMSE error is given by

$$\sqrt{\frac{1}{n-2}\sum_{i=0}^{Np} \left(u^h(x_i) - u(x_i)\right)^2}$$

where u^h and u are our approximate and exact solutions at time t_f and x_i are our grid points. The H^1 and L^2 errors are computed by using Gaussian quadrature of degree $2p$ over each interval to ensure accurate results.

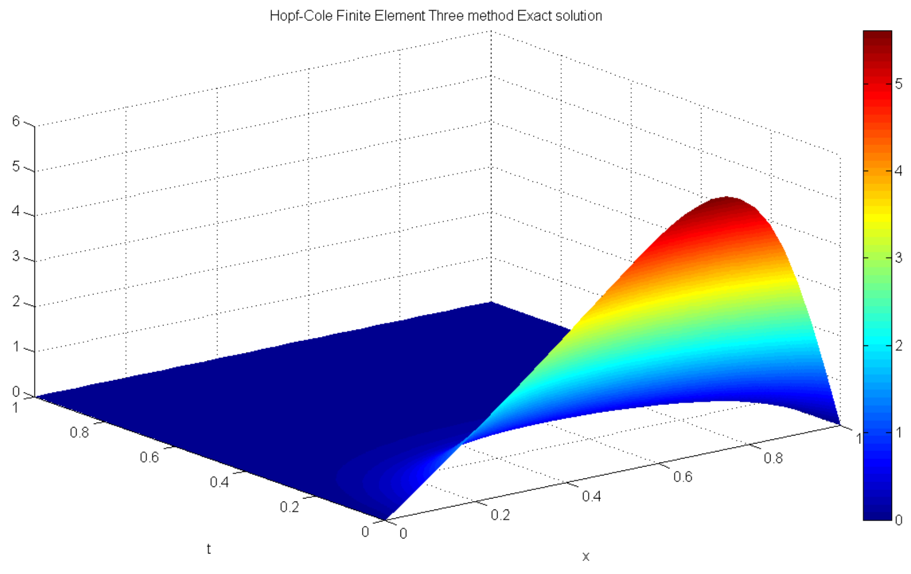

Figure 3.1: Exact solution of (1.8)

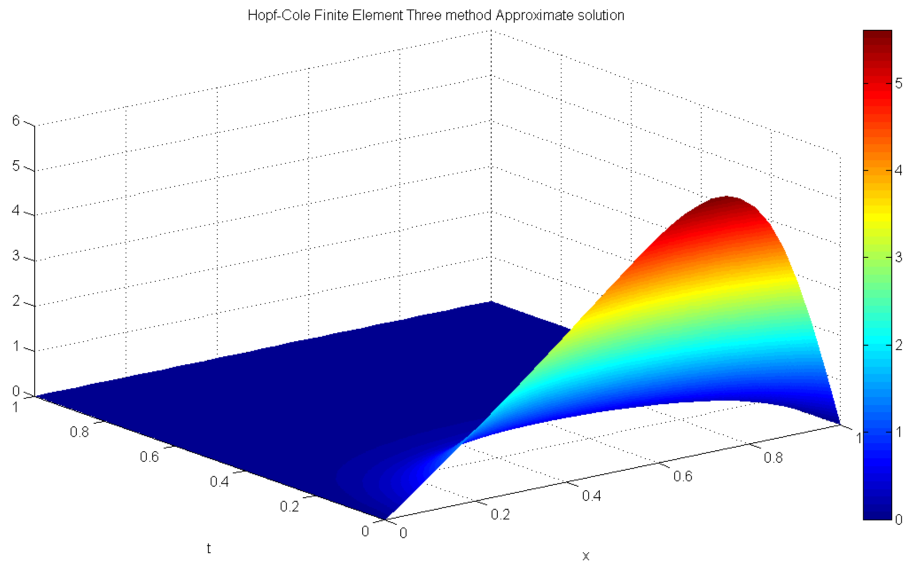

Figure 3.2: Approximate solution of (1.8), CFEM method, $\mu = .1$, $p = 8$, $\Delta x = 1/4$, $\Delta t = 10^{-13}$

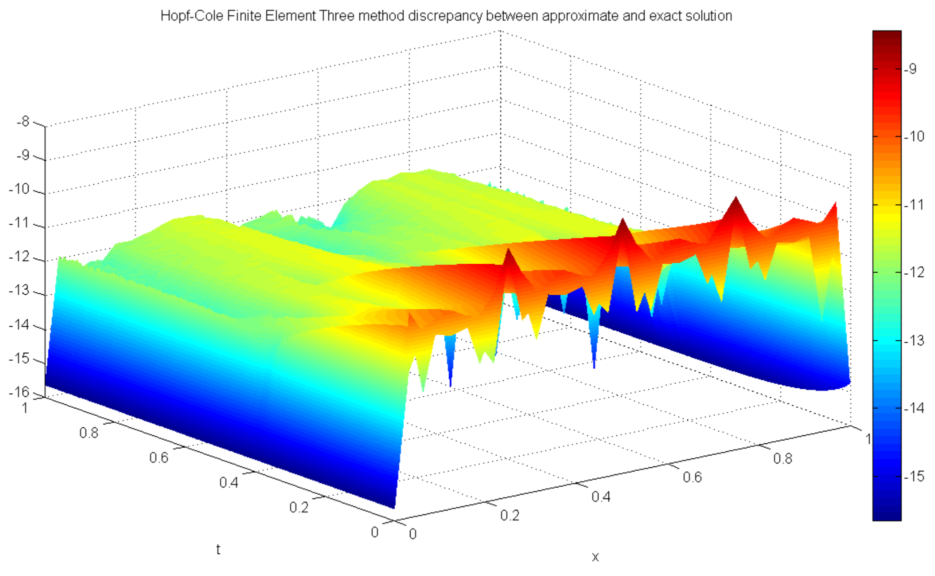

Figure 3.3: Log error of approximation versus exact solution for (1.8), CFEM method, $\mu = .1$, $p = 8$, $\Delta x = 1/4$, $\Delta t = 10^{-13}$

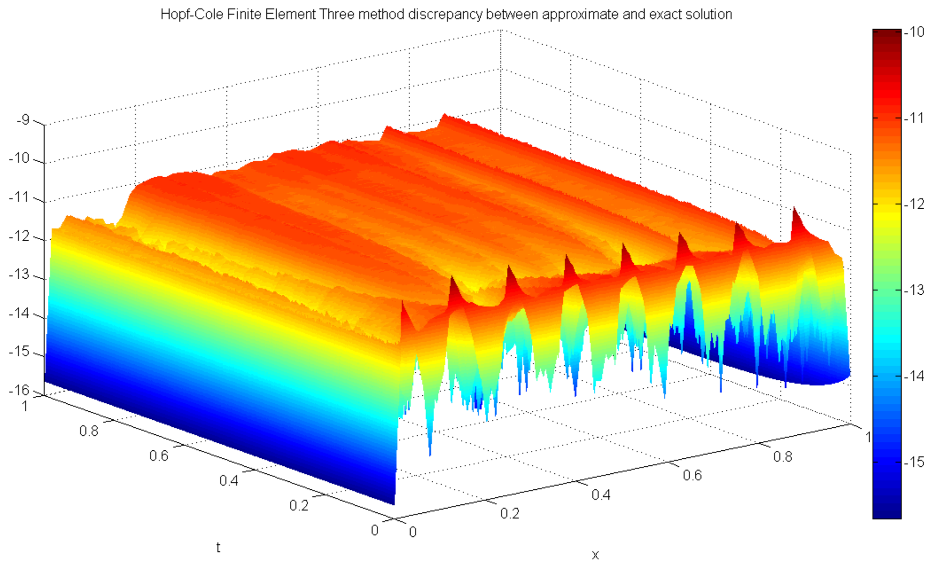

Figure 3.4: Log error of approximation versus exact solution for (1.8), CFEM method, $\mu = .1$, $p = 8$, $\Delta x = 1/8$, $\Delta t = 10^{-13}$

Note that only the approximately and qualitatively "best" solution as a function of time and numerical accuracy are shown for each method. The results in figures 3.5-3.7 should be qualified with the caveat that these graphed values represent only a single point within the parameter space $(\Delta x, \Delta t)$. A more complete picture is presented in tables 3.1 through 3.16. Here we can see a more accurate trade-off between accuracy and time needed to solve the model. In addition, we can see that we are getting at or near the expected convergence rates detailed for each of the above methods.

Of the finite difference methods, both our Rusanov and Roe methods are achieving the expected first order of accuracy (as measured in the L^2 norm), while the MacCormack method is getting a rate of convergence of 2 in the L^2 norm. In the H^1 norm all finite difference methods only achieve a rate of convergence of 1.

As for the finite element formulations, we see that the DFEM linear method is achieving rates of $p+1$ in the L^2 and p in the H^1 norms, respectively. This is due to the second-order accurate approximation of θ_x employed in the inverse Cole-Hopf transform. On the other hand, the FEM methods are achieving a rate at or near p in the L^2 norm and $p-1$ in the H^1 norm. Our RMSE error seems to be consistently in-between our L^2 and H^1 error rates, *i.e.* at $p - 0.5$.

For the four methods of DFEM, CFEM, Discrete POD based on CFEM and Continuous POD based on CFEM, only tables corresponding to $p = \{2, 5, 8\}$ are shown due to space considerations. The other tables show similar results.

The times listed are in ms and are computed on a Dell Latitude 531 2Ghz processor with 1 gigabyte of RAM. Some amount of variance is to be expected among the times needed to complete a model run. This said, it is clear from these graphs that POD is achieving two to three degrees of accuracy more than the fastest method, Rusanov, in half the time. Furthermore, it is achieving approximately the same degree of accuracy as methods that require two orders of magnitude more time to complete. We thus conclude that our POD method, based on using snapshots from every other model realization and retaining here a single basis function that describes all but $1e-12$ of the energy of the system, is indeed a formidable model for solving (1.6) quickly and accurately.

We should mention that the time needed to compute the full model, the averaged ensemble, and computation of the SVD are not included in the ROM time. The reason for this is that once a method for solving the full model has been chosen, all of these tasks can be completed offline just once and stored for easy access. Not including this time is therefore fully justified.

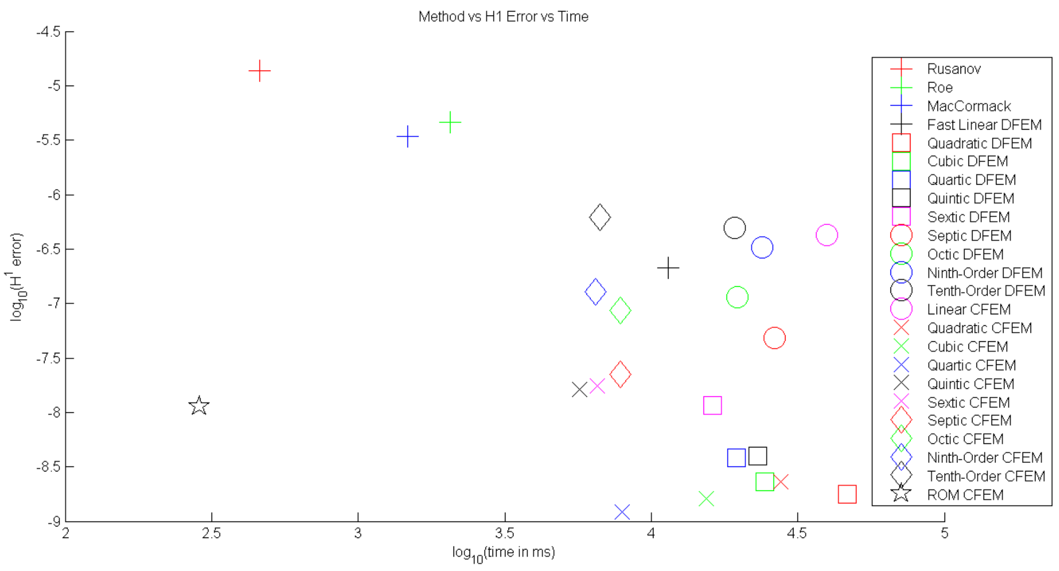

Figure 3.5: H^1 error versus time, per method

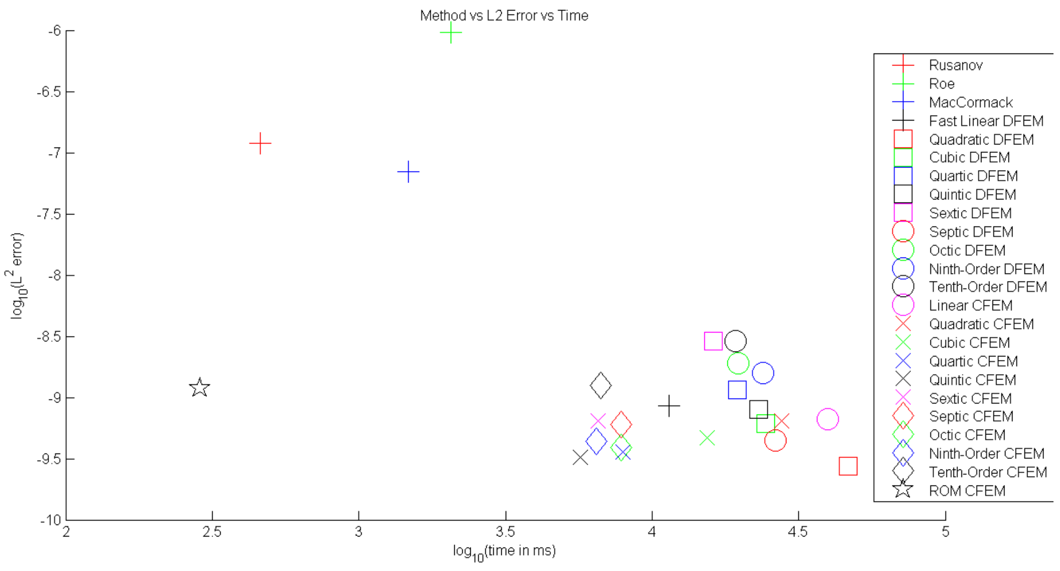

Figure 3.6: L^2 error versus time, per method

Table 3.1: L^2, H^1 and RMSE error and convergence rates for the Rusanov method

Δx	Δt	L^2 error	L^2 conv	H^1 error	H^1 conv	RMSE	R conv	time
6.25E-02	6.25E-02	1.41E-03	—	2.51E-02	—	5.25E-03	—	0.8
3.12E-02	3.12E-02	7.49E-04	0.919	1.26E-02	0.994	3.84E-03	0.450	1.1
1.56E-02	1.56E-02	3.85E-04	0.960	6.31E-03	0.998	2.75E-03	0.479	3.2
7.81E-03	7.81E-03	1.95E-04	0.981	3.15E-03	0.999	1.96E-03	0.489	9.6
3.90E-03	3.90E-03	9.82E-05	0.991	1.57E-03	0.999	1.39E-03	0.495	63.1
1.95E-03	1.95E-03	4.92E-05	0.995	7.89E-04	0.999	9.87E-04	0.498	155.3
9.76E-04	9.76E-04	2.46E-05	0.998	3.95E-04	1.000	6.98E-04	0.499	586.3
4.88E-04	4.88E-04	1.23E-05	0.999	1.97E-04	1.000	4.94E-04	0.499	2384.3
2.44E-04	2.44E-04	6.17E-06	0.999	9.87E-05	1.000	3.49E-04	0.499	9644.9

Table 3.2: L^2, H^1 and RMSE error and convergence rates for the Roe method

Δx	Δt	L^2 error	L^2 conv	H^1 error	H^1 conv	RMSE	R conv	time
6.25E-02	3.90E-03	3.94E-03	—	2.71E-02	—	1.14E-02	—	1.5
3.12E-02	1.95E-03	2.00E-03	0.977	1.39E-02	0.968	8.57E-03	0.423	3.8
1.56E-02	4.88E-04	9.72E-04	1.043	6.95E-03	0.999	5.96E-03	0.523	18.9
7.81E-03	1.22E-04	4.78E-04	1.022	3.47E-03	0.999	4.18E-03	0.511	130.9
3.90E-03	3.05E-05	2.37E-04	1.011	1.73E-03	0.999	2.94E-03	0.505	914.5
1.95E-03	7.62E-06	1.18E-04	1.005	8.69E-04	0.999	2.08E-03	0.503	6756.3
9.76E-04	1.90E-06	5.90E-05	1.003	4.34E-04	1.000	1.47E-03	0.501	50881.8

Table 3.3: L^2, H^1 and RMSE error and convergence rates for the MacCormack method

Δx	Δt	L^2 error	L^2 conv	H^1 error	H^1 conv	RMSE	R conv	time
1.25E-01	3.12E-02	8.65E-04	—	4.53E-02	—	4.59E-03	—	0.7
6.25E-02	7.81E-03	2.14E-04	2.014	2.25E-02	1.005	1.53E-03	1.579	1.4
3.12E-02	1.95E-03	5.36E-05	1.998	1.12E-02	1.001	5.27E-04	1.543	5.2
1.56E-02	4.88E-04	1.34E-05	1.997	5.64E-03	1.000	1.83E-04	1.522	33.8
7.81E-03	1.22E-04	3.36E-06	1.998	2.82E-03	1.000	6.44E-05	1.511	211.3
3.90E-03	3.05E-05	8.42E-07	1.999	1.41E-03	1.000	2.26E-05	1.506	1579.4
1.95E-03	7.62E-06	2.10E-07	1.999	7.05E-04	0.999	8.00E-06	1.503	12150.8
9.76E-04	1.90E-06	5.27E-08	1.999	3.52E-04	1.000	2.82E-06	1.501	96299.2

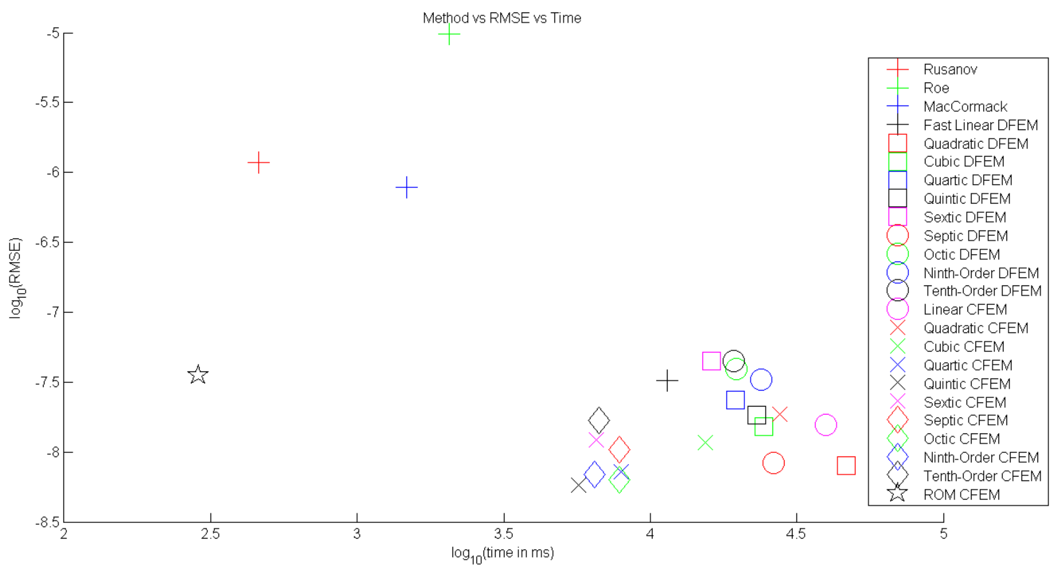

Figure 3.7: RMSE error versus time, per method

Table 3.4: L^2, H^1 and RMSE error and convergence rates for the linear DFEM method

Δx	Δt	L^2 error	L^2 conv	H^1 error	H^1 conv	RMSE	R conv	time
1.25E-01	3.12E-02	5.89E-03	—	4.74E-02	—	1.16E-02	—	15.2
6.25E-02	1.56E-02	1.50E-03	1.975	2.28E-02	1.053	4.06E-03	1.519	26.7
3.12E-02	7.81E-03	3.76E-04	1.994	1.13E-02	1.015	1.42E-03	1.516	53.6
1.56E-02	3.90E-03	9.42E-05	1.998	5.64E-03	1.004	4.98E-04	1.509	129.2
7.81E-03	1.95E-03	2.35E-05	1.999	2.82E-03	1.001	1.75E-04	1.505	236.1
3.90E-03	9.76E-04	5.89E-06	1.999	1.41E-03	1.000	6.20E-05	1.503	542.3
1.95E-03	4.88E-04	1.47E-06	1.999	7.05E-04	1.000	2.19E-05	1.501	1147
9.76E-04	2.44E-04	3.68E-07	2.000	3.52E-04	1.000	7.74E-06	1.501	2703.5
4.88E-04	1.22E-04	9.21E-08	2.000	1.76E-04	1.000	2.73E-06	1.500	7527
2.44E-04	6.10E-05	2.30E-08	2.001	8.81E-05	1.000	9.66E-07	1.501	24206.4
1.22E-04	3.05E-05	5.82E-09	1.983	4.40E-05	1.000	3.47E-07	1.476	84658.1

Table 3.5: L^2, H^1 and RMSE error and convergence rates for the quadratic DFEM method

Δx	Δt	L^2 error	L^2 conv	H^1 error	H^1 conv	RMSE	R conv	time
1.25E-01	1.25E-01	6.66E-04	—	3.90E-02	—	3.92E-03	—	308.6
6.25E-02	6.25E-02	1.66E-04	2.004	1.98E-02	0.974	1.38E-03	1.498	507.9
3.12E-02	3.12E-02	4.15E-05	1.999	1.00E-02	0.989	4.90E-04	1.502	897.7
1.56E-02	1.56E-02	1.04E-05	1.999	5.01E-03	0.996	1.73E-04	1.502	1777.7
7.81E-03	7.81E-03	2.60E-06	1.999	2.51E-03	0.998	6.11E-05	1.501	3748
3.90E-03	3.90E-03	6.50E-07	1.999	1.25E-03	0.999	2.16E-05	1.501	8191.1
1.95E-03	1.95E-03	1.62E-07	1.999	6.28E-04	0.999	7.64E-06	1.500	23711.6
9.76E-04	9.76E-04	4.06E-08	1.999	3.14E-04	0.999	2.70E-06	1.500	109360.9

Table 3.6: L^2, H^1 and RMSE error and convergence rates for the quintic DFEM method

Δx	Δt	L^2 error	L^2 conv	H^1 error	H^1 conv	RMSE	R conv	time
5.00E-01	3.E-02	2.54E-05	—	7.80E-04	—	4.18E-05	—	303.1
2.50E-01	3.E-03	6.56E-07	5.278	3.74E-05	4.380	1.65E-06	4.658	482.7
1.25E-01	9.E-04	2.05E-08	4.998	1.82E-06	4.360	9.44E-08	4.132	927.5
6.25E-02	1.E-04	5.64E-10	5.186	1.10E-07	4.044	3.61E-09	4.707	2648
3.12E-02	3.E-05	1.96E-11	4.847	6.93E-09	3.997	1.94E-10	4.216	19683.9

Table 3.7: L^2, H^1 and RMSE error and convergence rates for the octic DFEM method

Δx	Δt	L^2 error	L^2 conv	H^1 error	H^1 conv	RMSE	R conv	time
1.00E+00	3.12E-02	2.10E-05	—	5.55E-04	—	2.47E-05	—	489.1
5.00E-01	1.95E-03	1.82E-07	6.846	1.06E-05	5.707	1.36E-07	7.505	777.8
2.50E-01	1.22E-04	2.25E-10	9.660	1.92E-08	9.107	7.40E-10	7.522	1405.8
1.25E-01	1.52E-05	9.34E-12	4.594	2.78E-10	6.116	5.77E-11	3.679	3854.5
6.25E-02	6.10E-05	3.65E-11	-1.968	1.58E-10	0.816	3.66E-10	-2.668	10065.6

Table 3.8: L^2, H^1 and RMSE error and convergence rates for the quadratic CFEM method

Δx	Δt	L^2 error	L^2 conv	H^1 error	H^1 conv	RMSE	R conv	time
1.25E-01	1.00E-04	6.44E-04	—	3.89E-02	—	3.90E-03	—	198.8
6.25E-02	1.00E-04	1.61E-04	1.999	1.98E-02	0.973	1.38E-03	1.495	325.1
3.12E-02	1.00E-05	4.04E-05	1.995	1.00E-02	0.989	4.90E-04	1.501	582.9
1.56E-02	1.00E-05	1.04E-05	1.956	5.01E-03	0.995	1.73E-04	1.501	1131.3
7.81E-03	1.00E-06	2.57E-06	2.017	2.51E-03	0.998	6.11E-05	1.501	2261.3
3.90E-03	1.00E-07	6.36E-07	2.016	1.25E-03	0.999	2.16E-05	1.500	5180.5
1.95E-03	1.00E-07	1.69E-07	1.909	6.28E-04	0.999	7.64E-06	1.500	13948.4
9.76E-04	1.00E-08	3.97E-08	2.094	3.14E-04	0.999	2.70E-06	1.500	65447.1

Table 3.9: L^2, H^1 and RMSE error and convergence rates for the quintic CFEM method

Δx	Δt	L^2 error	L^2 conv	H^1 error	H^1 conv	RMSE	R conv	time
1.00E+00	1.00E-04	4.13E-04	—	6.98E-03	—	6.44E-05	—	177.2
5.00E-01	1.00E-05	2.26E-05	4.189	7.79E-04	3.164	3.18E-05	1.014	234
2.50E-01	1.00E-07	6.26E-07	5.177	3.74E-05	4.379	1.52E-06	4.389	377.6
1.25E-01	1.00E-08	1.74E-08	5.166	1.82E-06	4.360	7.43E-08	4.354	692.8
6.25E-02	1.00E-10	5.36E-10	5.024	1.10E-07	4.044	3.48E-09	4.415	1271.4
3.12E-02	1.00E-11	3.77E-11	3.830	6.93E-09	3.996	4.21E-10	3.048	2434.3
1.56E-02	1.00E-12	3.92E-12	3.265	4.41E-10	3.973	6.29E-11	2.743	5273.7
7.81E-03	1.00E-13	2.10E-12	0.901	2.03E-10	1.119	4.78E-11	0.395	18951.3

Table 3.10: L^2, H^1 and RMSE error and convergence rates for the octic CFEM method

Δx	Δt	L^2 error	L^2 conv	H^1 error	H^1 conv	RMSE	R conv	time
1.00E+00	1.00E-05	1.88E-05	—	5.54E-04	—	6.43E-06	—	364.5
5.00E-01	1.00E-07	1.85E-07	6.662	1.06E-05	5.704	1.73E-07	5.217	556.5
2.50E-01	1.00E-10	1.88E-10	9.946	1.92E-08	9.108	6.30E-10	8.101	967.5
1.25E-01	1.00E-13	1.08E-12	7.443	1.79E-10	6.750	5.53E-12	6.832	1862.4
6.25E-02	1.00E-13	7.23E-13	0.583	8.50E-11	1.074	6.83E-12	-0.305	3657.1
3.12E-02	1.00E-13	4.85E-12	-2.746	3.17E-10	-1.902	6.49E-11	-3.247	7668.8

Table 3.11: L^2, H^1 and RMSE error and convergence rates for the Discrete-time POD on quadratic CFEM method

Δx	Δt	L^2 error	L^2 conv	H^1 error	H^1 conv	RMSE	R conv	time
5.00E-01	1.56E-02	1.00E-02	3.718	1.22E-01	2.726	2.79E-02	3.307	1.8
2.50E-01	1.56E-02	2.58E-03	1.959	7.40E-02	0.725	1.08E-02	1.372	2.5
1.25E-01	1.56E-02	6.42E-04	2.008	3.89E-02	0.925	3.90E-03	1.467	4.4
6.25E-02	1.56E-02	1.61E-04	1.996	1.98E-02	0.973	1.38E-03	1.495	8.5
3.12E-02	1.56E-02	4.05E-05	1.992	1.00E-02	0.989	4.90E-04	1.501	19.8
1.56E-02	1.56E-02	1.04E-05	1.961	5.01E-03	0.995	1.73E-04	1.501	48.8
7.81E-03	3.90E-03	2.53E-06	2.035	2.51E-03	0.998	6.11E-05	1.501	299.5
3.90E-03	3.90E-03	6.50E-07	1.964	1.25E-03	0.999	2.16E-05	1.500	973.2
1.95E-03	3.90E-03	2.15E-07	1.593	6.28E-04	0.999	7.64E-06	1.500	3418.1

Table 3.12: L^2, H^1 and RMSE error and convergence rates for the Discrete-time POD on quintic CFEM method

Δx	Δt	L^2 error	L^2 conv	H^1 error	H^1 conv	RMSE	R conv	time
1.00E+00	3.90E-03	4.29E-04	—	7.15E-03	—	1.16E-04	—	3.2
5.00E-01	3.90E-03	2.24E-05	4.261	7.68E-04	3.217	3.05E-05	1.936	6.1
2.50E-01	3.90E-03	6.57E-07	5.093	3.73E-05	4.365	1.64E-06	4.214	10.8
1.25E-01	2.44E-04	1.75E-08	5.226	1.82E-06	4.355	7.20E-08	4.513	155
6.25E-02	6.10E-05	5.21E-10	5.072	1.08E-07	4.072	3.12E-09	4.528	1807.3
3.12E-02	3.05E-05	1.89E-11	4.783	6.79E-09	3.996	1.86E-10	4.065	10965.6

Table 3.13: L^2, H^1 and RMSE error and convergence rates for the Discrete-time POD on octic CFEM method

Δx	Δt	L^2 error	L^2 conv	H^1 error	H^1 conv	RMSE	R conv	time
1.00E+00	3.12E-02	2.11E-05	—	5.57E-04	—	2.49E-05	—	2.9
5.00E-01	3.90E-03	2.40E-07	6.458	1.06E-05	5.711	5.36E-07	5.540	9.4
2.50E-01	4.88E-04	2.31E-09	6.699	2.10E-08	8.983	1.16E-08	5.526	73.7
1.25E-01	3.05E-05	8.28E-12	8.124	1.85E-10	6.825	5.99E-11	7.601	2812.7

Table 3.14: L^2, H^1 and RMSE error and convergence rates for the Continuous-time POD on quadratic CFEM method

Δx	Δt	L^2 error	L^2 conv	H^1 error	H^1 conv	RMSE	R conv	time
1.00E+00	1.00E-03	5.53E-02	—	1.98E-01	—	7.57E-02	—	10
5.00E-01	1.00E-03	1.00E-02	2.463	1.22E-01	0.697	2.79E-02	1.438	11.1
2.50E-01	1.00E-03	2.57E-03	1.962	7.40E-02	0.724	1.08E-02	1.372	11.5
1.25E-01	1.00E-03	6.41E-04	2.006	3.89E-02	0.926	3.90E-03	1.467	14.7

Table 3.15: L^2, H^1 and RMSE error and convergence rates for the Continuous-time POD on quintic CFEM method

Δx	Δt	L^2 error	L^2 conv	H^1 error	H^1 conv	RMSE	R conv	time
1.00E+00	1.00E-04	4.29E-04	—	7.14E-03	—	1.15E-04	—	27.6
5.00E-01	1.00E-04	2.27E-05	4.239	7.69E-04	3.214	3.04E-05	1.929	16.3
2.50E-01	1.00E-07	6.24E-07	5.185	3.73E-05	4.366	1.50E-06	4.337	34.6
1.25E-01	1.00E-07	1.74E-08	5.163	1.82E-06	4.356	7.38E-08	4.349	32.8
6.25E-02	1.00E-09	5.39E-10	5.016	1.08E-07	4.075	3.56E-09	4.371	64.5
3.12E-02	1.00E-11	1.72E-11	4.968	6.79E-09	3.992	1.70E-10	4.386	152.4
1.56E-02	1.00E-13	6.44E-13	4.739	4.36E-10	3.962	9.00E-12	4.244	455.5

Table 3.16: L^2, H^1 and RMSE error and convergence rates for the Continuous-time POD on octic CFEM method

Δx	Δt	L^2 error	L^2 conv	H^1 error	H^1 conv	RMSE	R conv	time
1.00E+00	1.00E-05	1.88E-05	—	5.56E-04	—	4.65E-06	—	17.2
5.00E-01	1.00E-07	1.77E-07	6.732	1.06E-05	5.709	5.95E-08	6.289	26.4
2.50E-01	1.00E-10	1.57E-10	10.137	1.93E-08	9.105	2.18E-10	8.092	53.2
1.25E-01	1.00E-12	2.07E-12	6.244	1.84E-10	6.714	1.30E-11	4.058	106.9

CHAPTER 4

OPTIMAL CONTROL INVERSE PROBLEM

We now test ROM versus our full model on an inverse problem using our cost function (1.2).

Our problem is to minimize J directly through the use of a minimization algorithm. We restate our problem as

$$\min_{u_0^h} J(u_0^h) = \sum_{i=1}^{N_{obs}} \left(u_{obs}^i - u_f^i(u_0^h) \right)^2 \tag{4.1}$$

where u_0^h is a finite dimensional vector in our finite element space and $u_f^i(u_0^h)$ is our model solution at time t_f at x_i, written to emphasize the model's dependence on u_0^h. The rest of the notation follows (1.2).

We define our reduced-order problem for the reduced-order cost function J^r as

$$\min_{u_0^r} J^r(u_0^r) = \sum_{i=1}^{N_{obs}} \left(u_{obs}^i - (u_f^r(u_0^r))^i \right)^2 \tag{4.2}$$

where

- $u_0^r = u_0^r(x)$ is the unknown reduced-order initial condition
- u_{obs}^i is the i^{th} observation at time t_f at x_i
- $u^r(x, t_f | u_0^r) = u_m(x) + \sum_{i=0}^{Np} c_i^r(t)\psi_i(x)$ is the model solution of (1.6) at time t_f with u_0^r initial condition
- $(u_f^r(u_0^r))^i = u^r(x_i, t_f | u_0^r)$ is the model solution at x_i at time t_f, written to emphasize the dependence on u_0^r

We begin by generating solutions u_i^j from our full model (1.6). Based on our results in the previous chapter, we choose the 5th order continuous-time finite element method to compute our solution due to its excellent error rate as well as stability. We choose $\Delta x = 1/32$ and $\Delta t = 10^{-13}$ for our full solution. We then use our final-time solution as observations, *i.e.* $u_{obs}^i = u^h(x_i, t_f)$.

After computing our forward model solution, we then "forget" our initial condition and only retain our final solution u_{obs}, from which we try to find what initial conditions could have generated this data. This approach, while artificial, is an excellent way to test the methods themselves.

We will require an initial guess for our initial conditions. One realistic choice is to begin with our observations, *i.e.* $u_0^h = u_{obs}$. Similarly for the ROM model, after computing our forward model solution and computing our snapshots from this run, we then try to reconstruct the projected initial conditions. We compute our initial guess u_0^r by projecting the observations down to the reduced-order space.

For our reduced order problem, as with the ROM method listed above, we compute our snapshots by taking every second time step of our full model run.

We first test this problem on our exact solution (1.8). Figure 4.1 shows the magnitude of singular values for this problem in the linear heat equation space for θ^h. Figure 4.2 shows the shows the magnitude of the singular values in the full non-linear Burgers space for u^h. As we can see from these graphs and Parseval's identity (1.3), the Cole-Hopf regularizes this problem to a large extent so that many fewer singular values are needed to describe the same amount of variance. In fact, in the linear space a single POD basis function only leaves 1e-12 of the energy of the system unaccounted for, while 20 basis functions in the full space are needed to account for this much variance. This is probably due to the closed expression for (1.8). We recall that the Fourier series expansion of the function $f(x) = \sin(\pi x)$ on $L^2(a, b)$ only has one component in its expansion. In the same way, our function $\theta(x, t) = a + e^{-\pi^2 vt}\cos(\pi x)$ can be represented using just one POD eigenvector, shown in figure 4.3. Therefore, our problem is one of the most favorable situations for POD. Another problem will therefore be considered in section 4.2 to test our method more rigorously.

4.1 Minimization algorithms and results

Several minimization approaches are possible for solving our minimization problems (4.1) and (4.2). As our code is implemented in MATLAB® we try three methods readily available

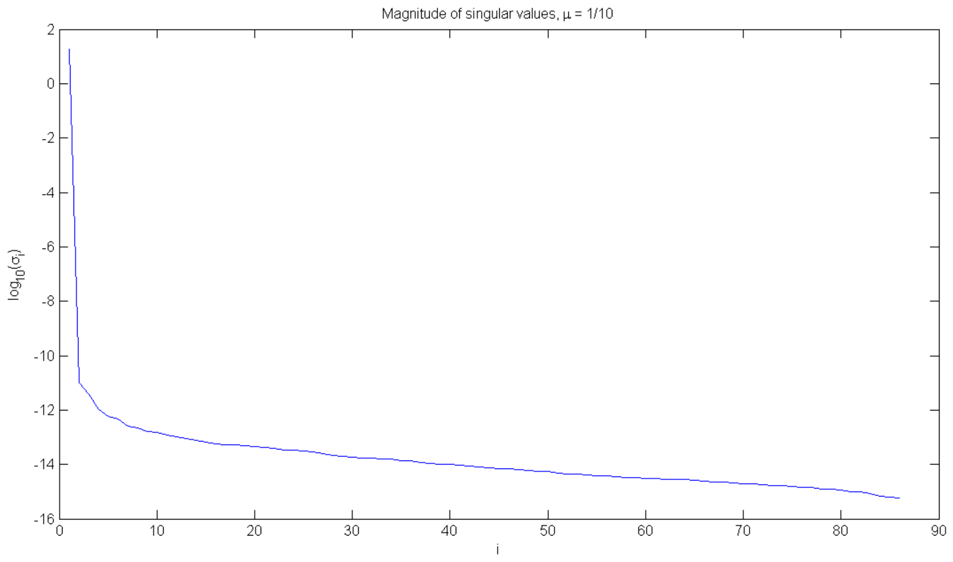

Figure 4.1: The magnitude of the singular values for a realization of (1.8) in the linear space (1.7)

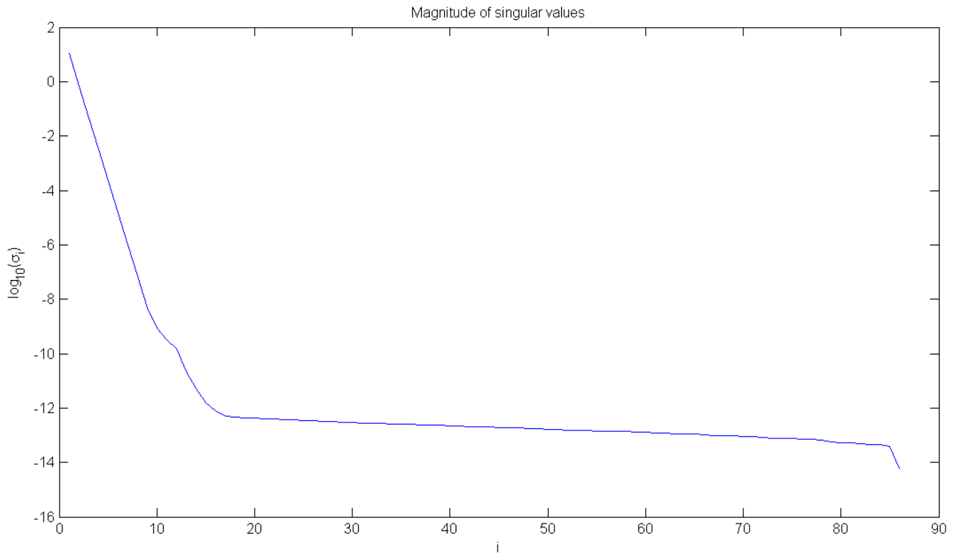

Figure 4.2: The magnitude of the singular values for a realization of (1.8) in the nonlinear space (1.6)

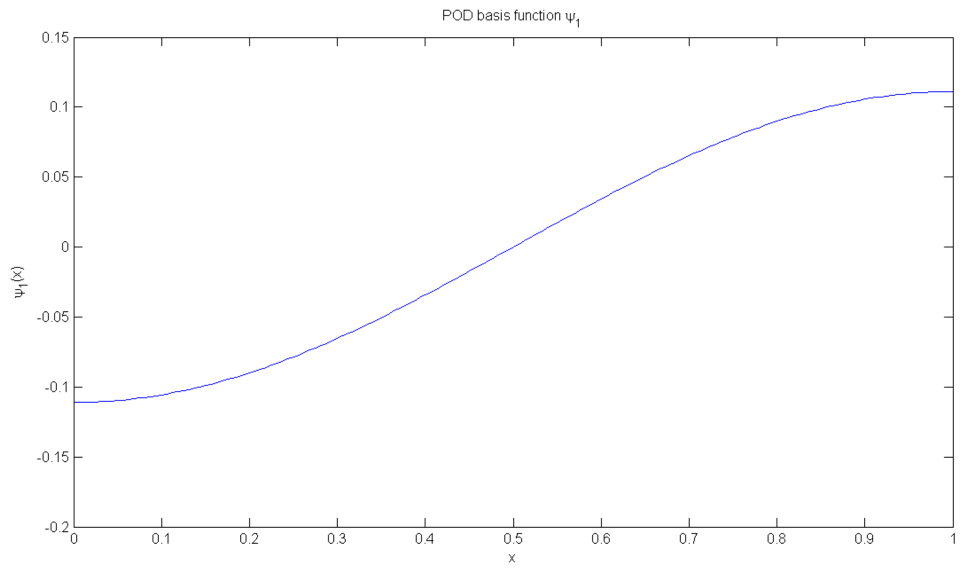

Figure 4.3: The first empirical POD basis for a solution of (1.7) in the linear space

in the Optimization Toolbox [41]:

1. The `fminsearch` derivative-free Nelder-Mead amoeba/simplex algorithm [42]
2. The `fminunc` gradient-based unconstrained minimization method based on the BFGS quasi-Newton algorithm [18]
3. The `fmincon` gradient-based constrained minimization method based on an active set sequential quadratic programming [43]

One might naively expect that we can use a Nelder-Mead derivative free optimization method to solve our problem without the need for an adjoint of the cost function [18]. The Nelder-Mead method, developed by Nelder and Mead at the National Vegetable Research Station in Warwick, United Kingdom, is also known as the amoeba or derivative-free simplex method. It creates a polytope of $N + 1$ vertices in N dimensions which it then expands, contracts, reflects, and reduces according to the function values encountered in order to undulate its way towards an optimal value (hence the protozoan moniker). One would expect that for high dimensional spaces this approach will, in general, be unreasonably slow due to the number of function evaluations required [44]. As expected, the Nelder-Mead method is completely unusable for the full model optimization due to the high dimension of the problem as shown in figures 4.4, 4.6 and 4.8. Note that 71 minutes of processing time with Nelder-Mead yielded a far inferior result to 5 seconds with a gradient-based method. However, we see that, perhaps somewhat unexpectedly, Nelder-Mead is an option for our reduced model as shown in figures 4.5, 4.7 and 4.9. At just 5 to 10 times the computational cost, Nelder-Mead obtained a results virtually identical to the gradient based methods in this low-dimensional space. This reduced-order inverse modeling approach is attractive as it eliminates the need for an adjoint while still giving excellent results. The extension of this method to two and three dimensional POD models seems dubious, however, since the performance of Nelder-Mead is highly dependent upon the dimension of the space it is minimizing over.

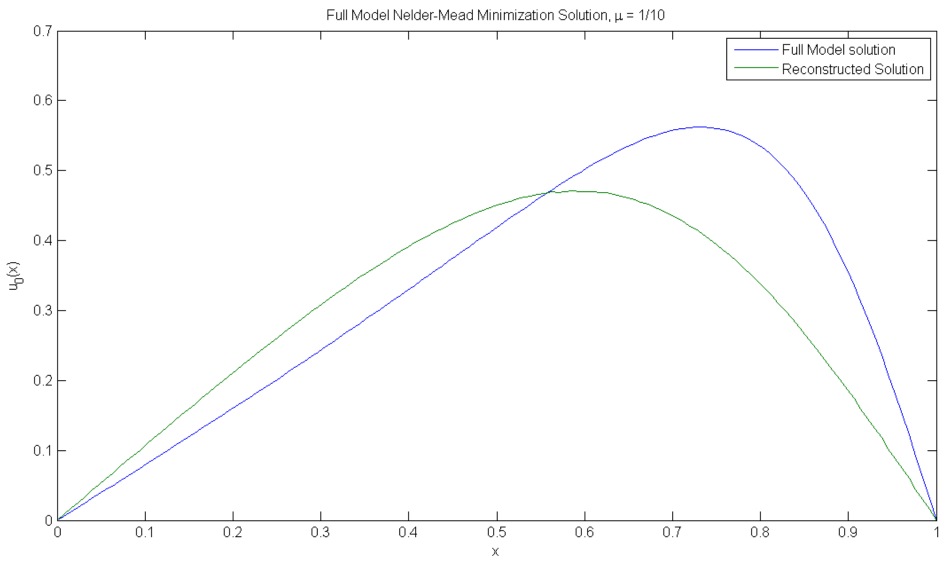

Figure 4.4: Full model solution versus exact solution, `fminsearch` method

Of our two gradient based minimization algorithms, the `fminunc` method is more straight- forward. This function uses a BFGS quasi-Newton line search approach with pre-conditioned conjugate gradients [18] as detailed in [41]. The constrained minimization method `fmincon` adds the ability to enforce linear or non-linear constraints using the Lagrangian or Augmented-Lagrangian formulation and then repeatedly solving a quadratic problem [43]. The algorithm is presented in detail in [41]. Here we use a constraint to keep

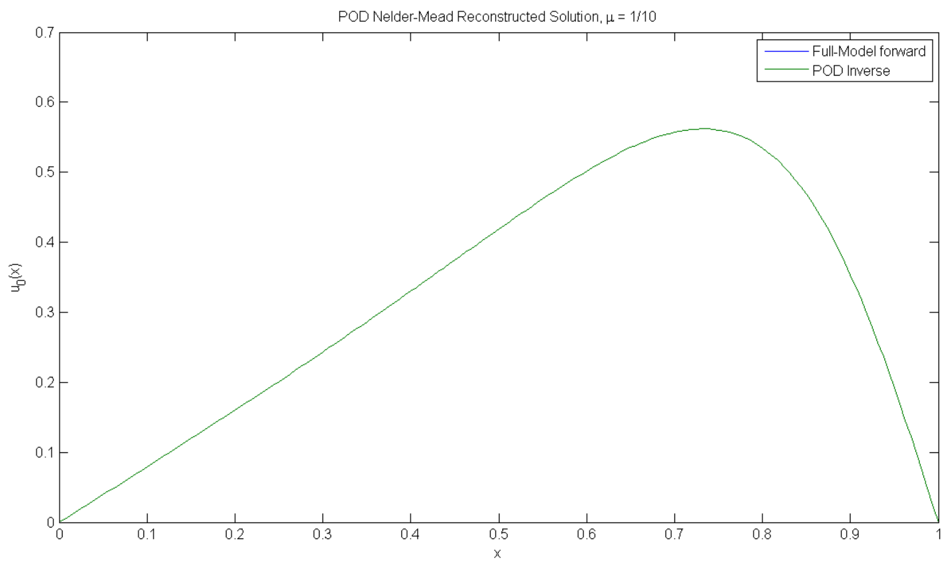

Figure 4.5: ROM model solution versus exact solution, `fminsearch` method

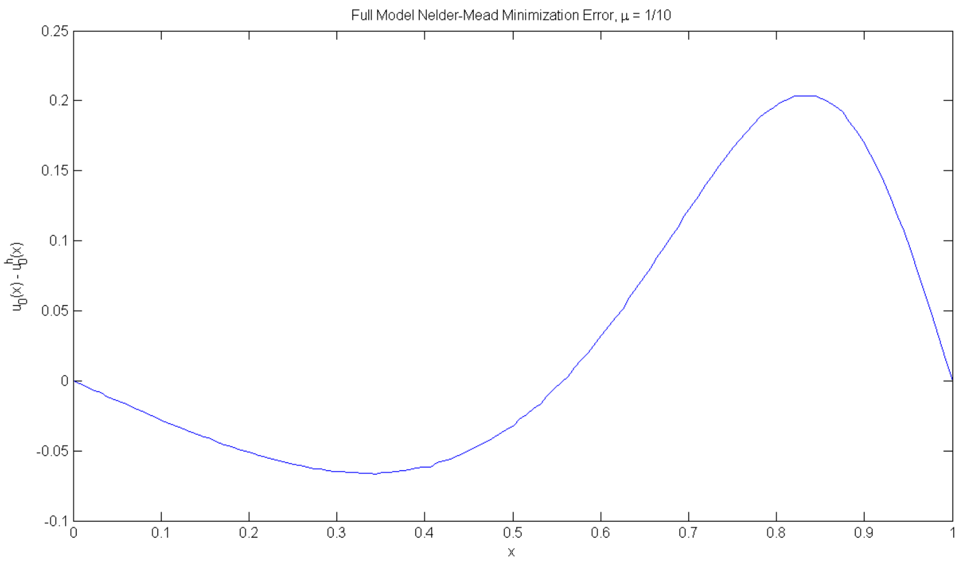

Figure 4.6: Full model error versus exact solution, `fminsearch` method. $||u_0 - u_0^h|| = 1.1066$

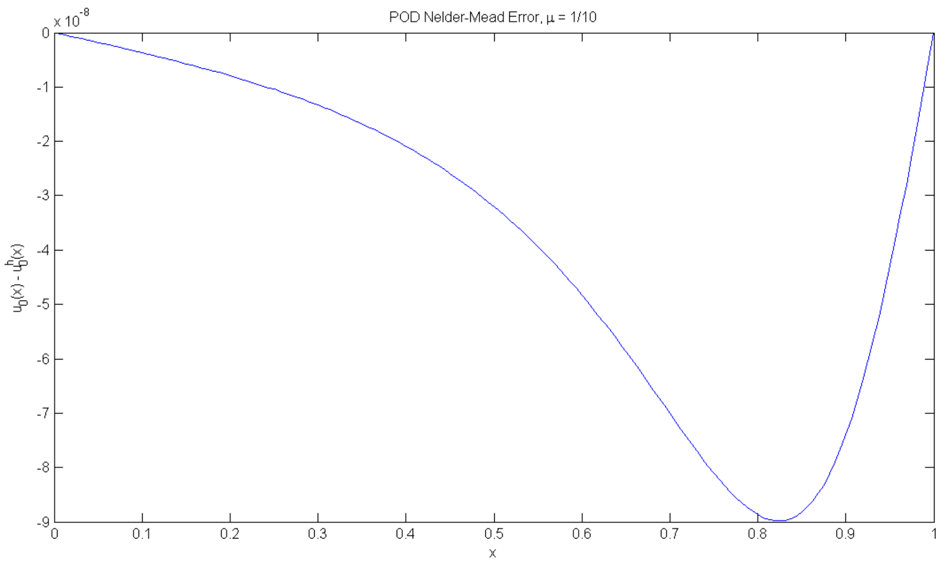

Figure 4.7: ROM model error versus exact solution, **fminsearch** method. $||u_0 - u_0^h|| = 5.9703 \times 10^{-7}$

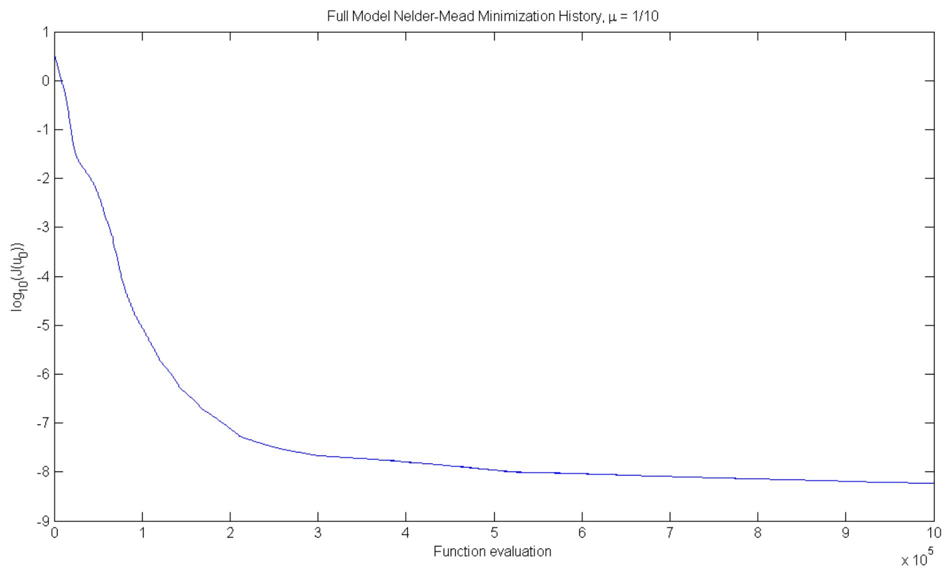

Figure 4.8: Full cost function minimization history versus function evaluations, **fminsearch** method. Total time: 4318618 ms

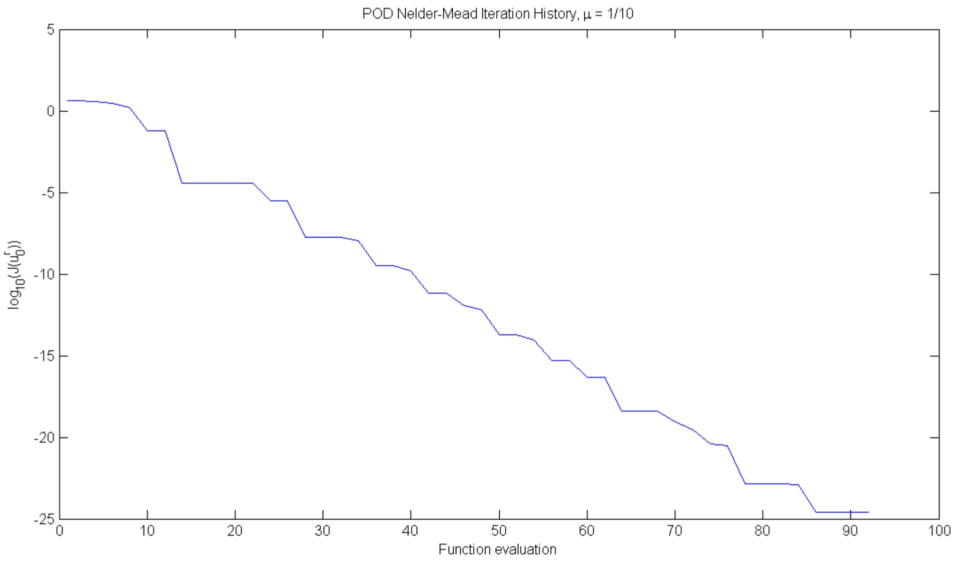

Figure 4.9: ROM cost function minimization history versus function evaluations, `fminsearch` method. Total time: 780 ms

our linearized $\theta(x,t)$ sufficiently far from zero to avoid any discontinuities. This improves the stability of the inverse solver at high Reynolds numbers where undesirable shock solutions (with $\theta(x,t)=0$) can describe the observations reasonably well. The added constraint keeps the solver from moving towards these discontinuous solutions.

4.1.1 Full model adjoint

Both of the previous methods require the use of $\nabla J = \frac{\partial J}{\partial u_0}$. This can be found through a continuous or discrete adjoint. In this study we use a discrete (discretize-then-differentiate) adjoint computed by hand through a careful application of the chain rule. Rather than computing our adjoint by hand, we could have alternately used an automatic differentiation tool.

Consider the discrete-time finite element and POD methods considered above. If we rewrite our cost function as

$$J(u(u_0)) = \sum_{i=1}^{N_{obs}} (u_{obs}^i - u_i(u_0))^2$$

by the chain rule, we can write

$$\frac{\partial J}{\partial u_0^i} = \sum_{j=1}^{N_{obs}} -2(u_{obs}^i - u_f^i(u_0))\frac{\partial u_f^i}{\partial u_0^i}$$

where again u_0^i is the i^{th} component of our initial condition vector (originally set to our observations), and $u_f^i(u_0)$ is the model solution at time t_f at x_i written to show the explicit dependence on the initial condition.

If we consider our discrete-time finite element method, we have

$$Dc^{j+1} = Ec^j$$

By the nodal property (2.2) of our finite element basis, $u_i^j = c_i^j$. Therefore, by recurrence, we have

$$u_f(u_0) = (D^{-1}E)^{t_f/\Delta t}u_0 \tag{4.3}$$

Taking the Jacobian with respect to u_0, we have

$$\nabla u_f = (D^{-1}E)^{t_f/\Delta t}$$

We can thus write

$$\frac{\partial J}{\partial u_0^i} = -2u_{err}^T \nabla u_f^i \tag{4.4}$$

where $u_{err} = u_{obs} - u_f$ and ∇u_f^i is the i^{th} column of the Jacobian ∇u_f.

We test the correctness of this adjoint with the `'DerivativeCheck'` set to `'on'` in the MATLAB® Optimization Toolkit [41]. For example, this reports the following for our full model unconstrained minimization method:

```
Maximum relative discrepancy between derivatives = 7.90199e-005
Caution: user-supplied and finite-difference derivatives do
 not match within 1e-006 relative tolerance.
Maximum difference occurs in element (106,1):
  User-supplied derivative, @(x)costfull(x,obs,iCDm,C,D,dx,dt,tf):     0.0386494
  Finite-difference derivative:     0.0385704
```

Due to the inverse and matrix power in the expression for ∇u_f, we expect the loss of some precision. Indeed, one disadvantage of the discretize-then-differentiate adjoint approach is the potential loss of accuracy [14]. However, because our derivatives are correct to within 7.90199e-005 of a finite difference test, and because our overall results are excellent, we are lead to believe the gradient being “accurate enough” is sufficient for our purposes.

4.1.2 ROM gradient

We now proceed to compute the gradient of the ROM.

Our POD discrete-in-time method described in section 2.3.2 is written as

$$D^r(c^r)^{j+1} = E^r(c^r)^j - \mu\Delta t F$$

where D^r, E^r, and F are defined as in section 2.3.2.

Computing the first few terms in the recurrence, we have

$$\begin{gathered}(c^r)^1 = ((D^r)^{-1}E^r)c_0^r - \mu\Delta t F) \\ (c^r)^2 = ((D^r)^{-1}E^r)^2 c_0^r - (D^r)^{-1}(E^r(D^r)^{-1} + I)(\mu\Delta t F) \\ (c^r)^3 = ((D^r)^{-1}E^r)^3 c_0^r - (D^r)^{-1}((E^r(D^r)^{-1})^2 + E^r(D^r)^{-1} + I)(\mu\Delta t F)\end{gathered}$$

From which we see the relationship

$$(c^r)^j = ((D^r)^{-1}E^r)^j c_0^r - G^{(j)}$$

where

$$G^{(j)} = \mu\Delta t(D^r)^{-1}\left(\sum_{i=1}^{j}(ED)^{j-1}\right)F$$

In order to compute our initial set of reduced-order coefficients, we project u_{obs} to our reduced order space via (2.4), i.e. we solve the system $M^r c_0^r = u_{obs}^r - u^m$ where $M^r = \mathcal{A}$ is defined as in section 2.3.2.

Furthermore, once we have found $(c^r)^f$, we project it back to the full space with by (2.3) $u_f^r = \Psi c_f^r + u^m$, where $\Psi = \begin{pmatrix}\psi_1 & \dots & \psi_\ell\end{pmatrix} \in \mathbb{R}^{Np\times\ell}$ is the column-wise matrix of POD basis functions.

Therefore, we have that

$$u_f^r(u_0^r) = \Psi\left(((D^r)^{-1}E^r)^{t_f/\Delta t}(M^r)^{-1}u_0^r) - G^{(f)}\right) \tag{4.5}$$

where $G^{(f)}$ is $G^{(j)}$ for t_f.

We have from (4.5) and (4.2) that

$$\frac{\partial J^r}{\partial (u_0^r)^i} = -2u_{err}^T \nabla (u_f^r)^i \tag{4.6}$$

where u_{err} is $u_{obs} - u_f^r$ and $\nabla u_f^r = \Psi(((D^r)^{-1}E^r)^{t_f/\Delta t}(M^r)^{-1}$, and $\nabla (u_f^r)^i$ is the i^{th} column of ∇u_f^r.

4.1.3 Recurrence Discussion

We now have our full model and ROM gradients with respect to the full and reduced cost functions, respectively. In our implementation we compute these gradients as matrix multiplications in order to increase our model speed. This will introduce inaccuracies in the gradient due to the computation of the model inverse and matrix power as opposed to solving linear systems for each time step. However, as we have noted above, having some inaccuracies in the gradient is an acceptable trade-off for increases in performance.

We comment that at first glance both (4.3) and (4.5) seem attractive from a numerical perspective. If we store $A = (D^{-1}E)^{t_f/\Delta t}$, only a single matrix multiplication is necessary to compute the final time step for (4.3). For (4.5) by storing the matrix $A^r = ((D^r)^{-1}E^r)^{t_f/\Delta t}(M^r)^{-1}$ and $G^{(f)}$ we can compute our final state u_f^r with just two matrix multiplications and a vector subtraction. However, the loss of accuracy here is more troublesome since it is of our model solution itself. Nonetheless, this method might be interesting, for example, in the context of computer graphics, where low accuracy would be an acceptable trade-off for capturing high-level details while maintaining fast performance. Unfortunately such investigation is beyond the scope of this thesis.

4.1.4 Gradient-Based Method Results

We use the gradients (4.4) and (4.6) as inputs to `fminunc` and `fmincon` for our full and reduced cost functions to give the results shown in figures 4.10 through 4.15 for the unconstrained method and figures 4.16 through 4.21 for the constrained method. In both of these methods we are forced to use a very large time step $\Delta t = 0.5$ for the full model to prevent numerical rank deficiencies from forming; see section 4.1.5 below for more details. This limits the amount of accuracy that can be achieved with the full model inverse problem. The ROM method displays no such deficiencies here, however.

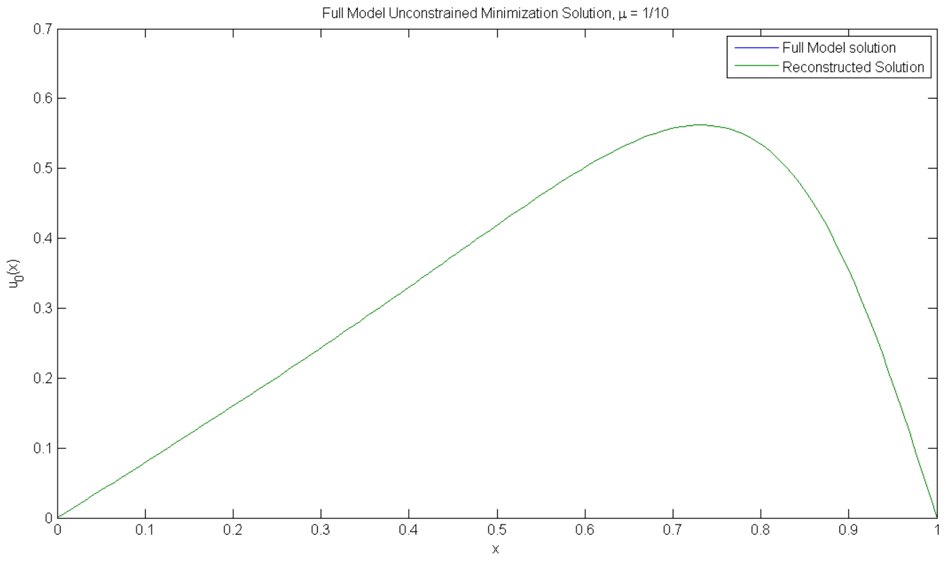

Figure 4.10: Full model solution versus exact solution, `fminunc` method

4.1.5 Optimal Control Inverse Results Discussion

From these results we see that both the constrained and unconstrained methods are efficient and effective choices for solving our inverse problem. They are 5 to 10 times more efficient than Nelder-Mead in the reduced formulation, and in the full formulation are still practical methods while the Nelder-Mead method simply is not. The expense of the success of these methods is the complexity involved in deriving the gradients of the cost functional.

The unconstrained minimization method `fminunc` is slightly slower than `fmincon` for our full model due to inner conjugate-gradient loops that are taken within `fminunc`; however, the reward is higher accuracy. For POD, our constrained method is slower than the unconstrained method due to requiring the inverse ROM projection to test whether $\theta^r(x,t)$ is less than zero. Therefore, we recommend the unconstrained method unless shock solutions are developing which are giving unphysical results.

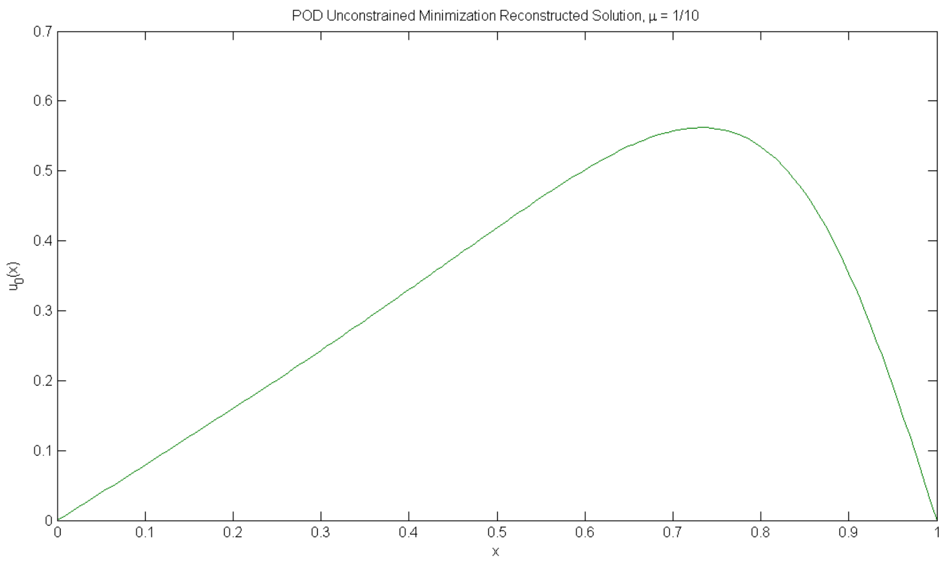

Figure 4.11: ROM model solution versus exact solution, `fminunc` method

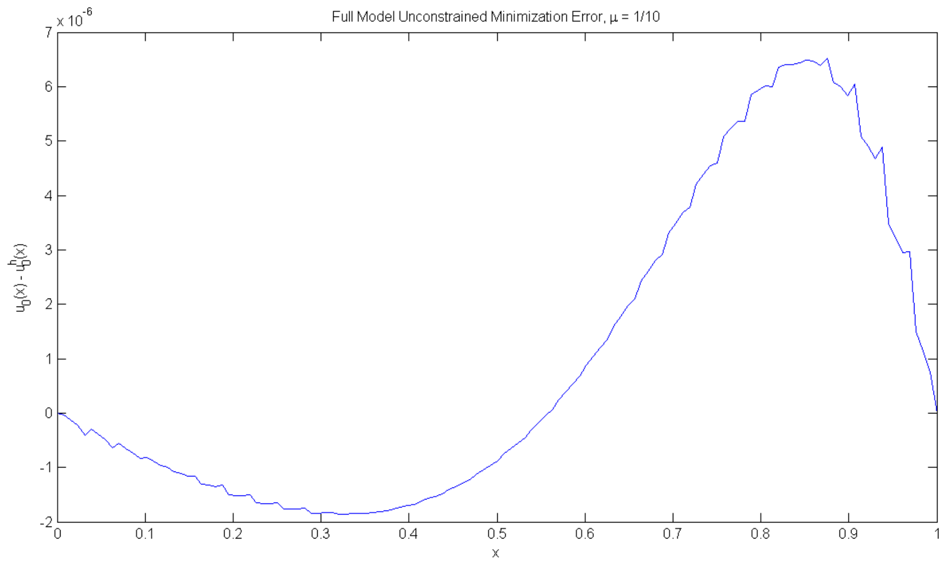

Figure 4.12: Full model error versus exact solution, `fminunc` method. $||u_0 - u_0^h|| = 3.4217 \times 10^{-5}$

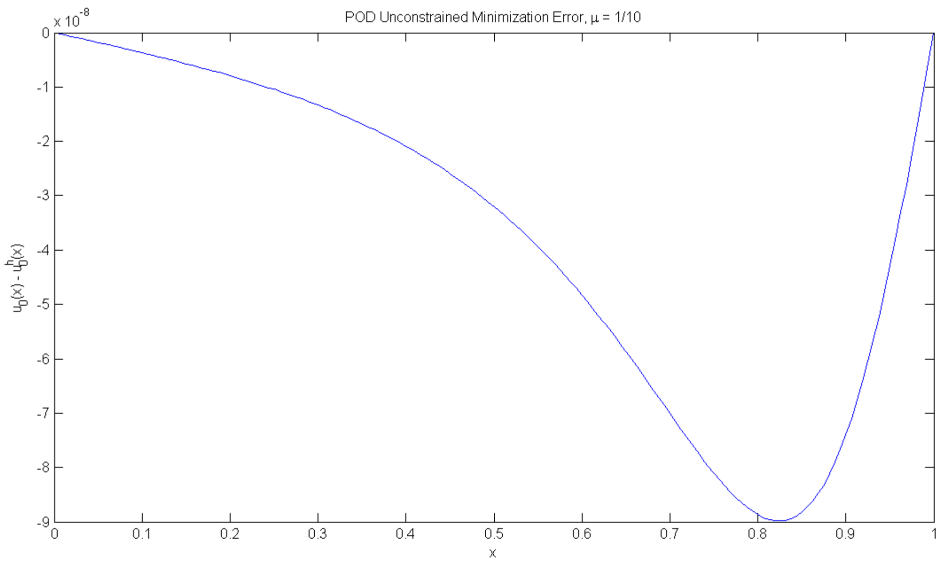

Figure 4.13: ROM model error versus exact solution, `fminunc` method. $||u_0 - u_0^h|| = 5.9703 \times 10^{-7}$

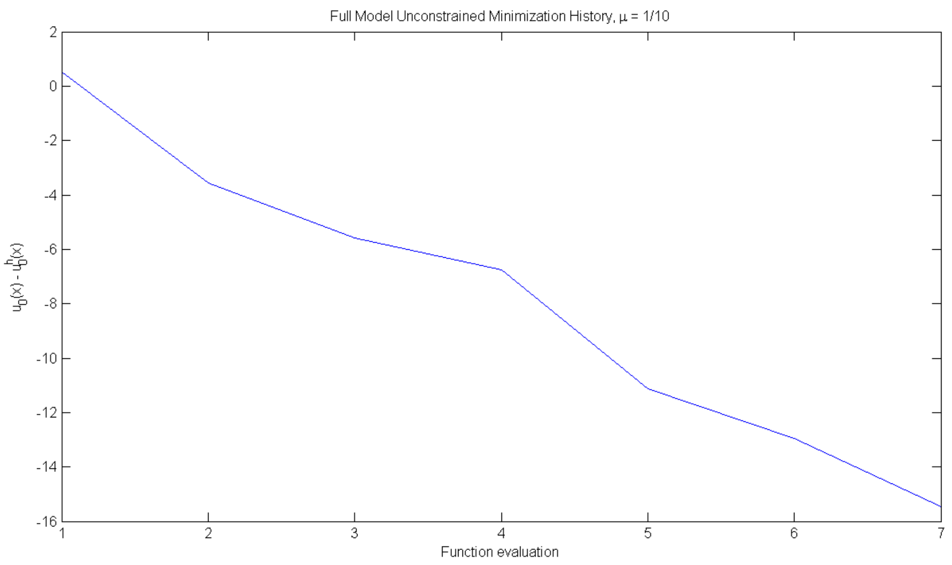

Figure 4.14: Full cost function minimization history versus function evaluations, `fminunc` method. Total time: 4088 ms

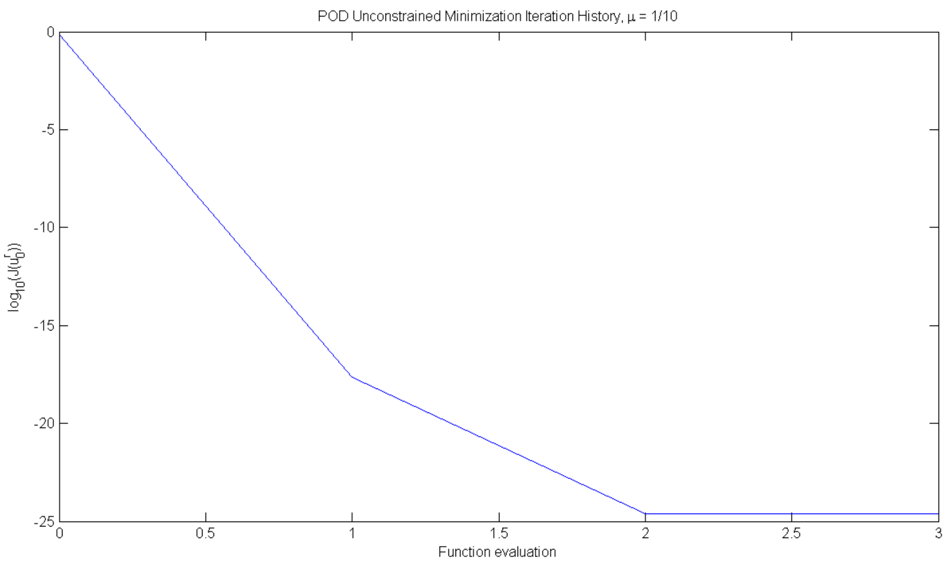

Figure 4.15: ROM cost function minimization history versus function evaluations, `fminunc` method. Total time: 82 ms

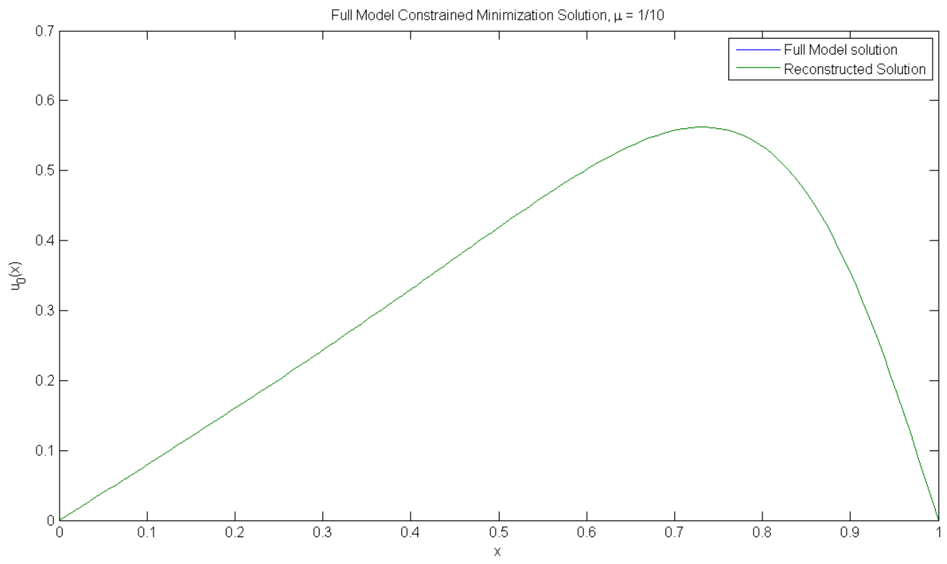

Figure 4.16: Full model solution versus exact solution, `fmincon` method

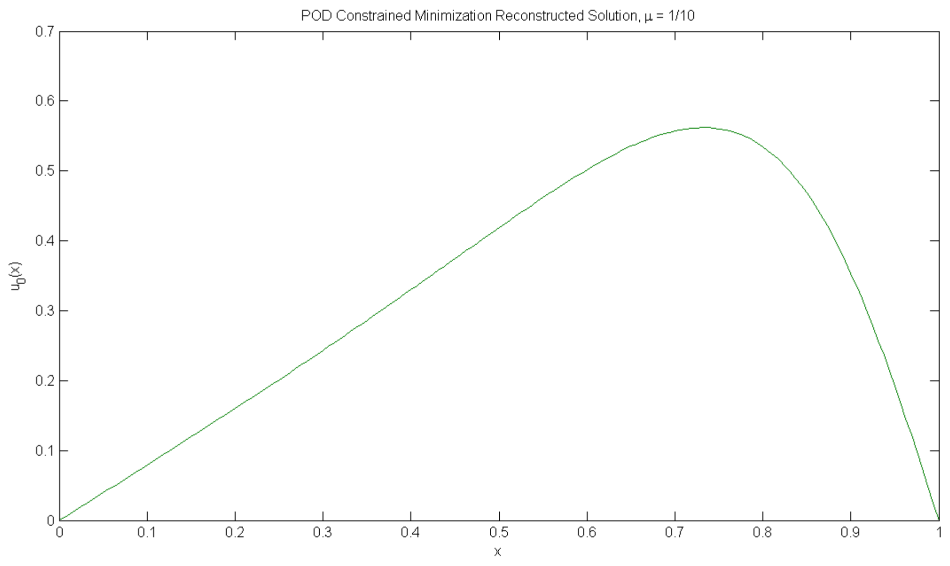

Figure 4.17: ROM model solution versus exact solution, `fmincon` method

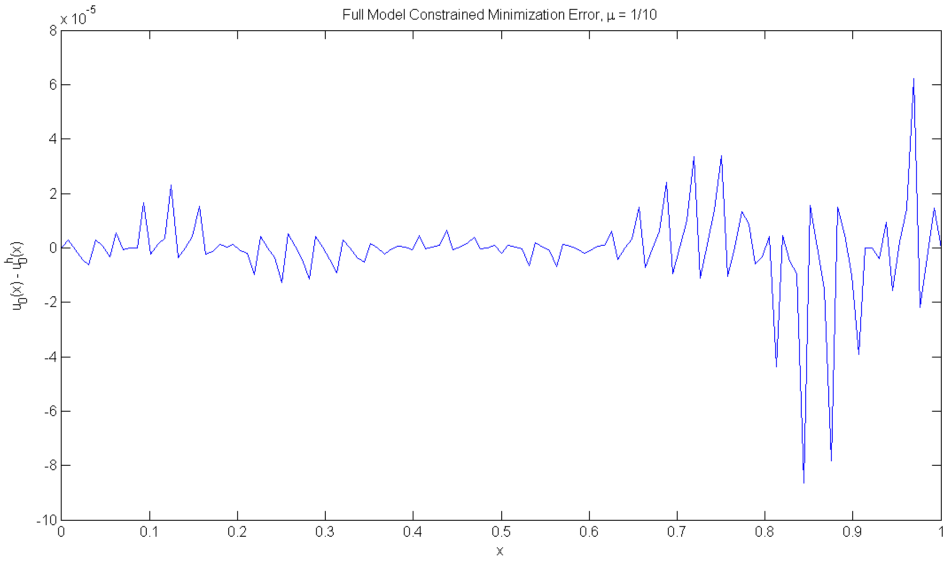

Figure 4.18: Full model error versus exact solution, `fmincon` method. $||u_0 - u_0^h|| = 1.7133 \times 10^{-4}$

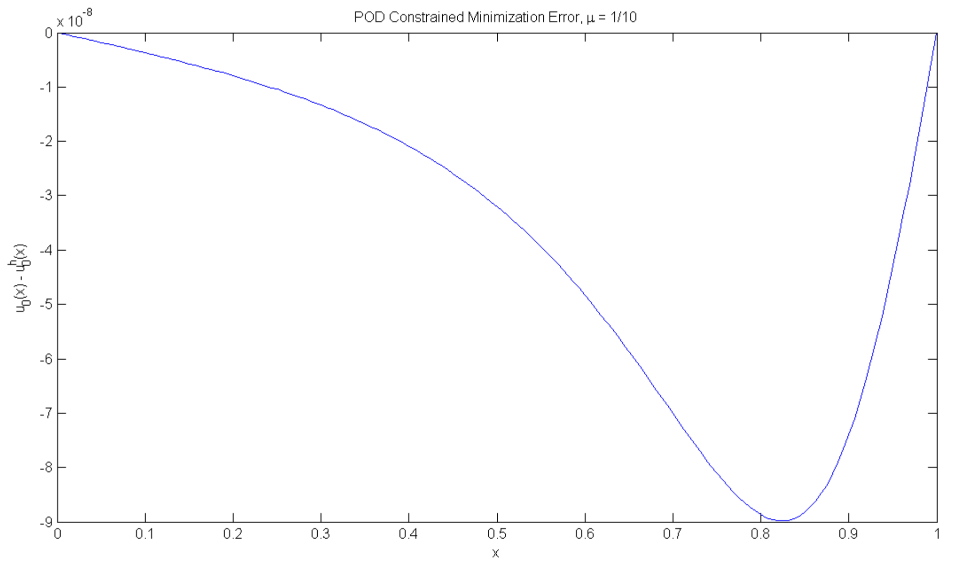

Figure 4.19: ROM model error versus exact solution, `fmincon` method. $||u_0 - u_0^h|| = 5.9703 \times 10^{-7}$

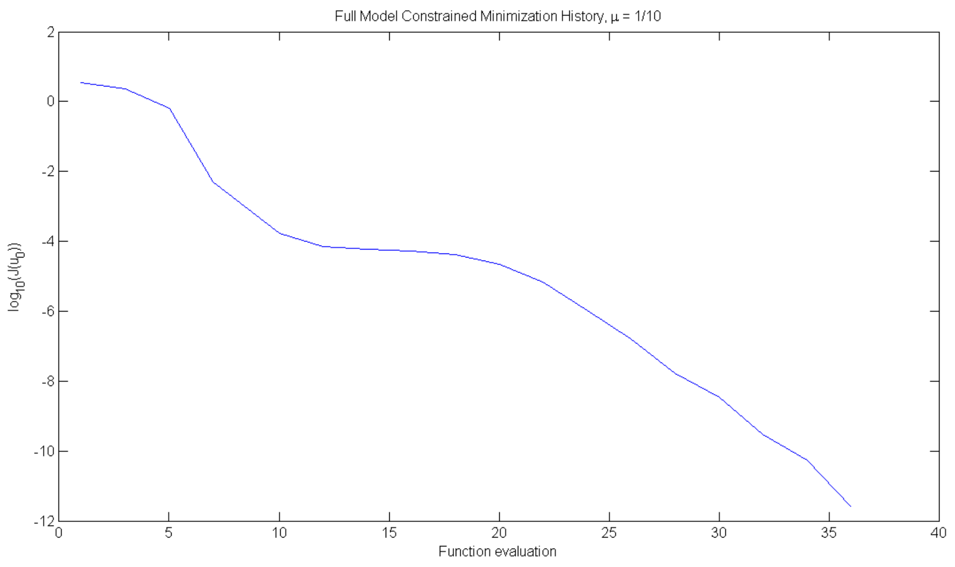

Figure 4.20: Full cost function minimization history versus function evaluations, `fmincon` method. Total time: 1937 ms

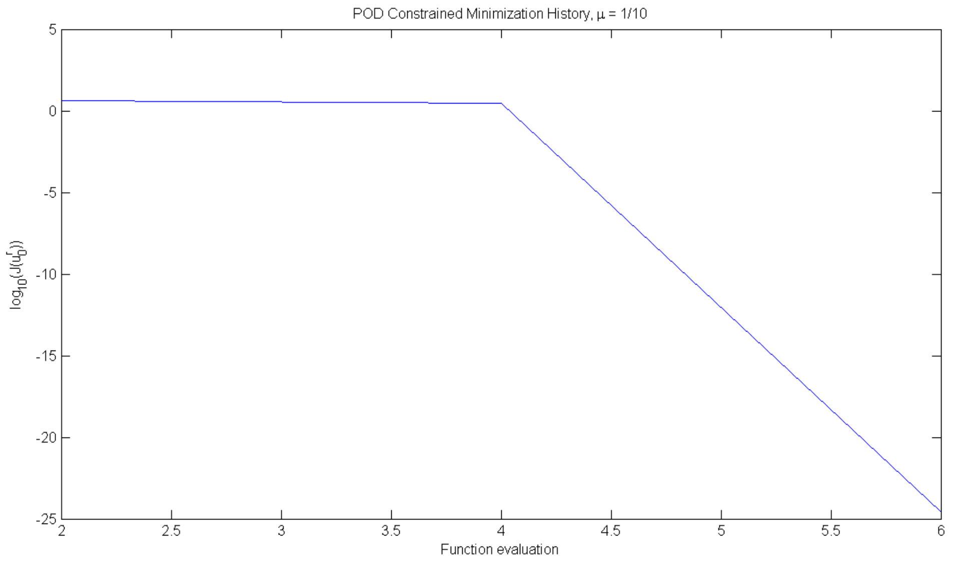

Figure 4.21: ROM cost function minimization history versus function evaluations, `fmincon` method. Total time: 188 ms

We also note that the use of ROM cut 98% and 90% off of the total times for the unconstrained and constrained methods, respectively, even while delivering superior accuracy. This is due to the preconditioning effect of POD described in Daescu and Navon [45].

We can see in comparing the full model to our ROM model (see, *e.g.*, figures 4.16 and 4.17) that the full model solution to the inverse problem is not only many times slower than the ROM solution, it is also less accurate. In fact, as our time tolerance Δt becomes smaller, our full model becomes increasingly worse at solving the inverse problem while our POD performance continues to increase. This situation is shown more dramatically in figures 4.22 through 4.25. How can it be that:

1. The full model is solving this problem with less accuracy than a reduced one?

2. As our time step decreases our full model inverse solution accuracy decreases?

To answer these questions, let us consider our discrete-in-time finite element model in 2.2.4. From (4.3), we have $u^{j+1} = (M^{-1}K)^j)^{t_f/\Delta t}u_0$ As $j \to \infty$, $(M^{-1}K)^{t_f/\Delta t} \in \mathbb{R}^{Np+1 \times Np+1}$ quickly becomes increasingly numerically deficient, as shown in table 4.1.

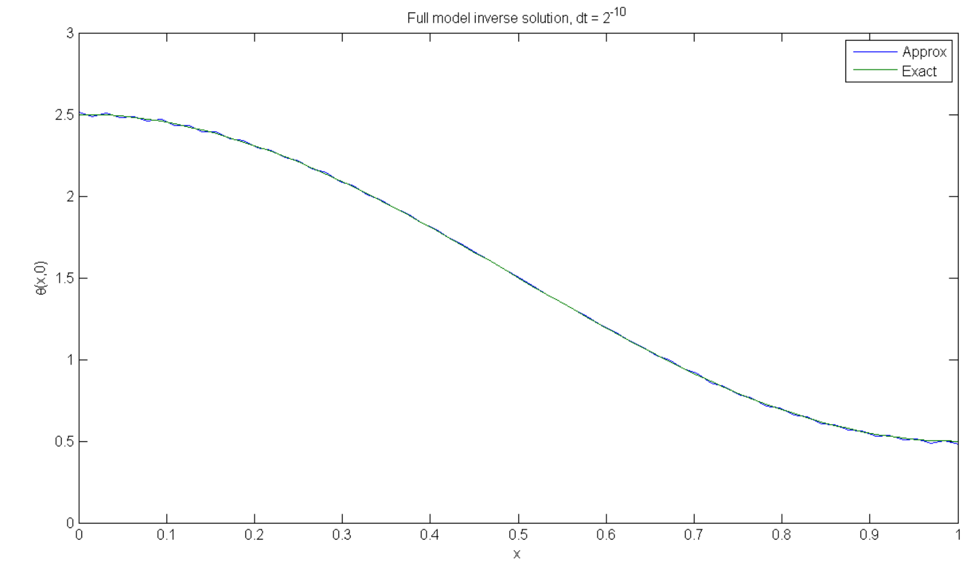

Figure 4.22: Initial condition - full model vs exact, $\Delta t = 2^{-10}$

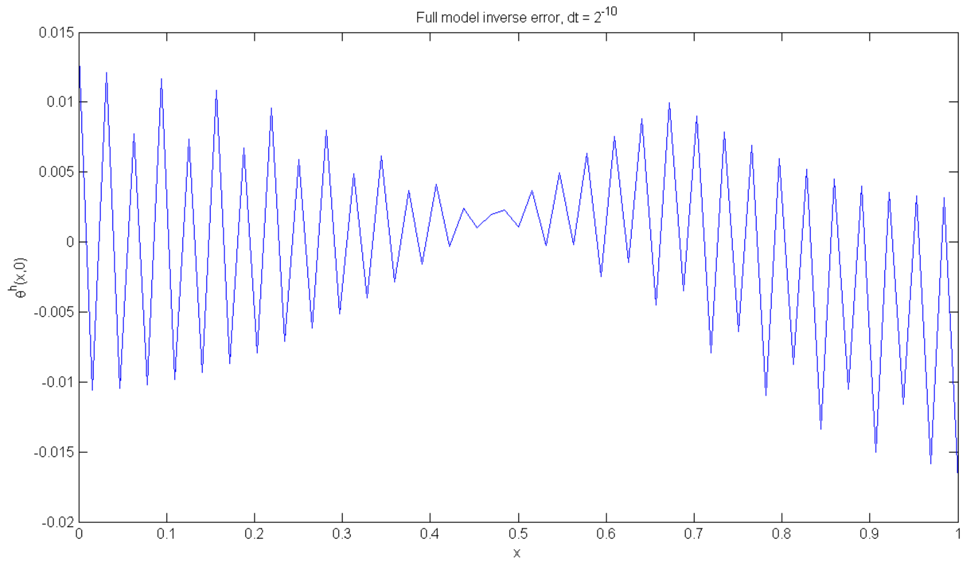

Figure 4.23: Full solution error vs exact initial conditions, $\Delta t = 2^{-10}$

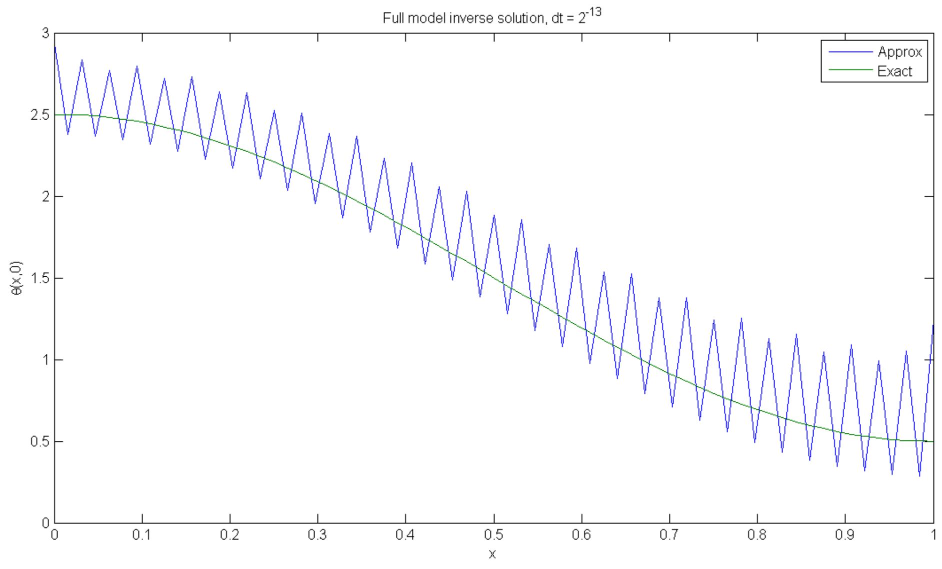

Figure 4.24: Initial condition - full model vs exact, $\Delta t = 2^{-13}$

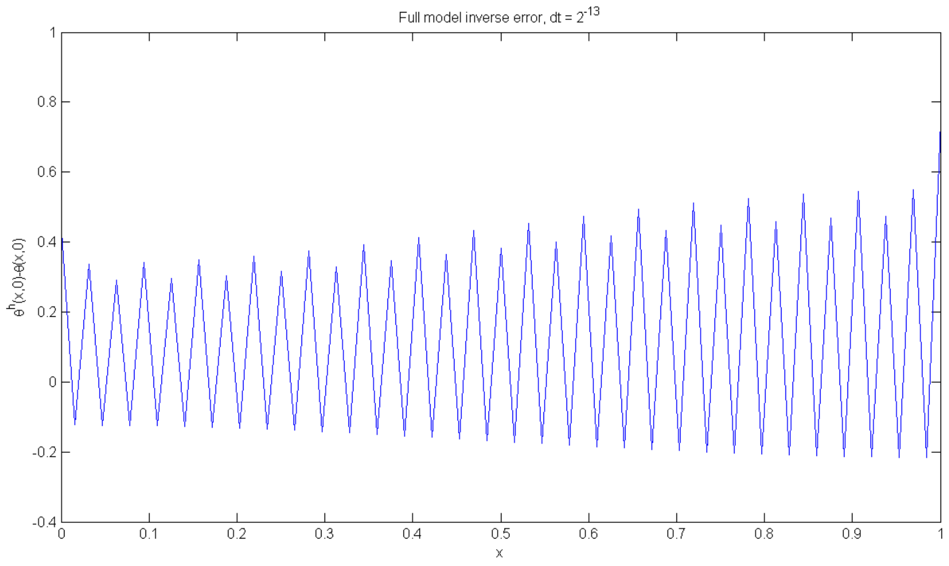

Figure 4.25: Full solution error vs exact initial conditions, $\Delta t = 2^{-13}$

Table 4.1: Rank of $(M^{-1}K)^{t_f/\Delta T} \Leftrightarrow$ # of singular values greater than 2.220e-016 for solving (1.8), $\Delta x = 1/16$, $p = 4$

Δt	rank$(M^{-1}K)$
2^{-3}	65
2^{-4}	64
2^{-5}	63
2^{-6}	60
2^{-7}	53
2^{-8}	39
2^{-9}	19
2^{-10}	2

When a matrix A is rank deficient, there is no longer a unique solution to the problem $Ax = b$ [46]. Thus, while a system may give results that agree well with the exact solution solving the forward problem, if the matrix is rank deficient it is very difficult to find the exact initial condition that generated the observations.

On the other hand, the POD matrix does not become rank deficient until much smaller Δt values are encountered. This is because the condition number of the matrix $(M^r)^{-1}K^r$ shows excellent numerical properties. For example, in the problem (4.7) from the next section, if four POD basis functions are kept (explaining all but 7.8970×10^{-5} of the energy), we see the numerical behavior shown in table 4.2. Here, losing half the rank of the POD matrix does not occur for a step-size 16 to 32 times smaller than the full model. This numerical performance can be explained by considering the condition number of $(M^r)^{-1}K^r$, which is 1.016428 in this example. Compare this to the condition number of $M^{-1}K$ from table 4.1, which is 6.2157×10^{16}!

Furthermore, the POD matrix contains only the most relevant information about the matrix due to the properties of the singular value decomposition and has less opportunity to be corrupted by numerical noise. In other words, POD is serving to *regularize* this problem. Indeed, we see that POD is very similar to the method of "zeroth-order" Tikhonov regularization described in Press et al. 19.4.1 [39].

Another way to regularize the problem is to add an additional term to our cost function (1.2) that provides a measure of smoothness or stability as described in Press et al. 19.4.1 [39]. However, this either adds an in general non-linear constraint to the problem or requires

Table 4.2: Rank of $((M^r)^{-1}K^r)^{1/\Delta T}$ $\Leftrightarrow$ # of singular values greater than 2.220e-016 for $\gamma = 1e-5$

Δt	$\text{rank}((M^r)^{-1}K^r)$
2^{-6}	4
2^{-7}	4
2^{-8}	4
2^{-9}	4
2^{-10}	4
2^{-11}	4
2^{-12}	3
2^{-13}	2
2^{-14}	1

minimization of multiple objective functions which in general will require more function evaluations. By contrast, in this situation using POD is both accurate, fast, and relatively simple to implement. Further connections between POD and inverse problem regularization is left to a further study.

4.2 Reynolds number tests

We will conclude with tests of the properties of our POD inverse algorithm in the face of increasing Reynolds numbers. We know that fluids become increasingly turbulent and difficult to predict as the Reynolds number, here equal to $1/\mu$, becomes large. This is made more difficult because the problem we are trying to solve is known to be ill-posed [47]. At sufficiently high Reynolds numbers it becomes difficult to accurately simulate such a flow, let alone solve an inverse problem based on it.

Here we will use our reduced-order minimization problem (4.2) with our unconstrained solver for the true initial condition $u_0(x) = \sin(\pi x)$ which satisfies our boundary conditions. By the Cole-Hopf transform (1.5), we have

$$\theta_0(x,t) = \exp\left(-\frac{1}{2\mu}\int_0^x \sin(\pi\xi)\, d\xi\right)$$

or, integrating,

$$\theta_0(x,t) = \exp\left(\frac{1}{2\pi\mu}\cos(\pi x)\right) \tag{4.7}$$

This often tested initial condition [38] represents the opposite case of Woods [33] exact problem (1.8) for our method - here the nonlinear function is simple while the linear transformed initial condition is more complex. This situation is demonstrated by the empirical singular value distribution of the two methods shown in figure 4.26. We can see that while the situation is not as pronounced as in 4.1, the linear transform still shows regularization properties over the non-linear space due to the singular values being more concentrated towards the top of the spectrum.

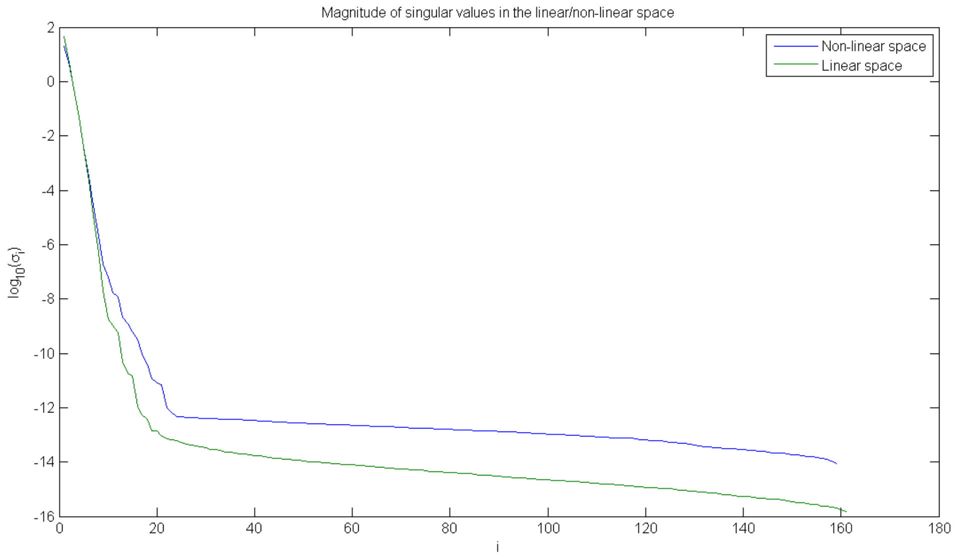

Figure 4.26: Comparison of singular values for the snapshot matrix of u and θ for (4.7) for $\mu = 1/10$

Running our inverse problem on the full model for $\mu = 1/10$ and $t_f = 0.25$, we see the results in figures 4.27 through 4.29. On the other hand, ROM gives the inverse results shown in figures 4.30 through 4.32. We can see once again that POD is regularizing our problem and achieving much more accuracy than the full model.

We run our full model again with $\Delta x = 1/32$ and $p = 5$ and varying values of μ. This time, however, we only run our model for $t_f < 0.25$ because for this test function the energy dissipates very quickly. The accuracy of our solution versus the Reynolds number at different t_f values is shown in figures 4.33 through 4.37.

Unfortunately as we can see the accuracy of our inverse problem solution decreases from

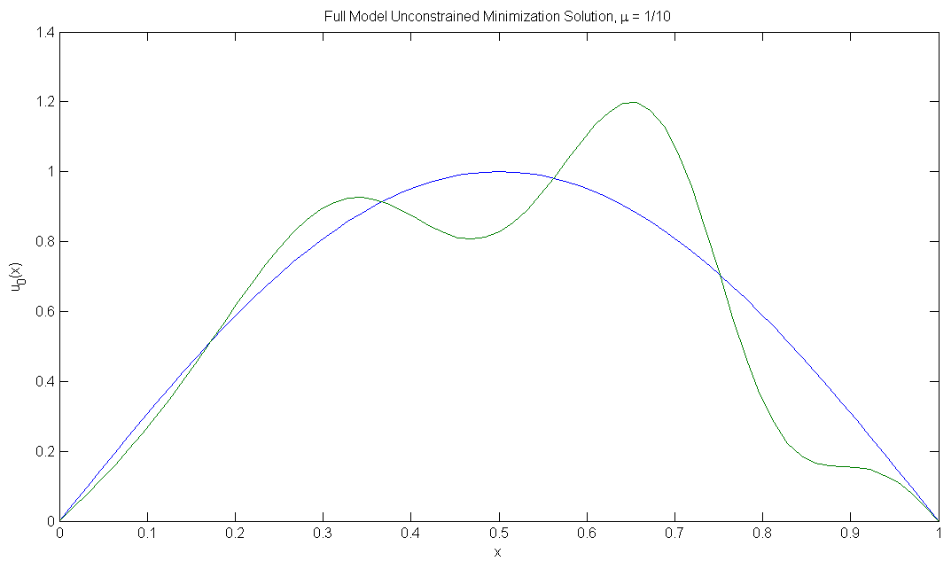

Figure 4.27: Full model inverse solution for (4.7), $\mu = 1/10$

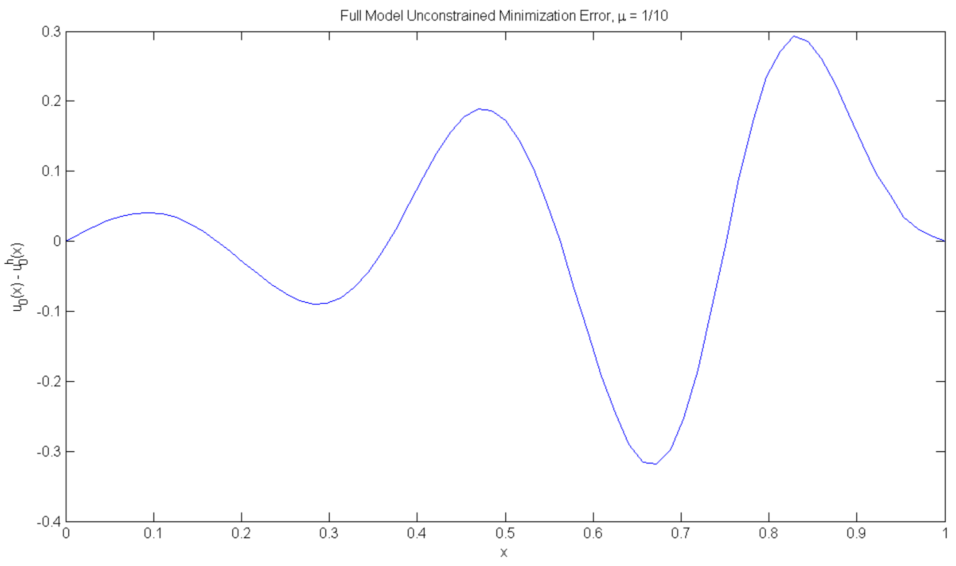

Figure 4.28: Full model inverse error for (4.7), $\mu = 1/10$

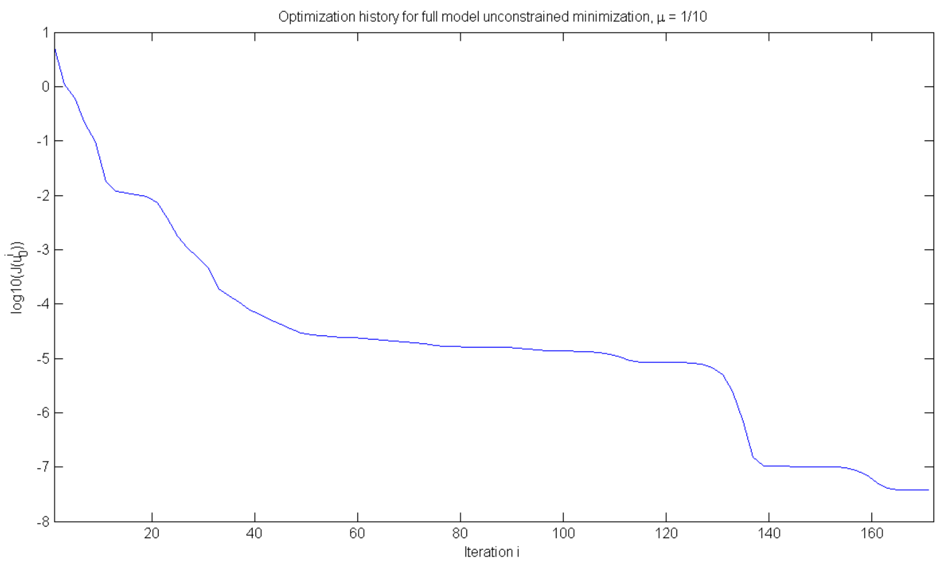

Figure 4.29: Full model minimization history for (4.7), $\mu = 1/10$

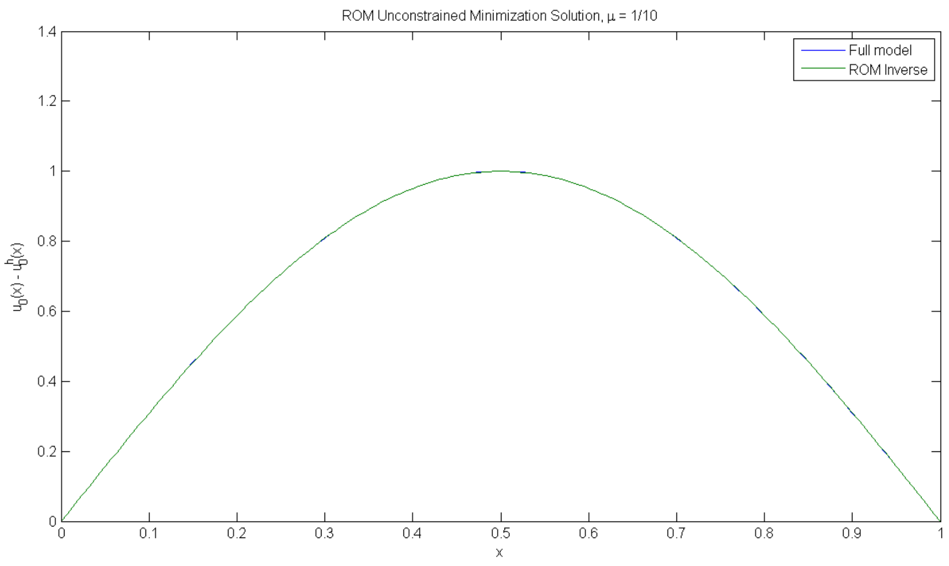

Figure 4.30: ROM inverse solution for (4.7), $\mu = 1/10$

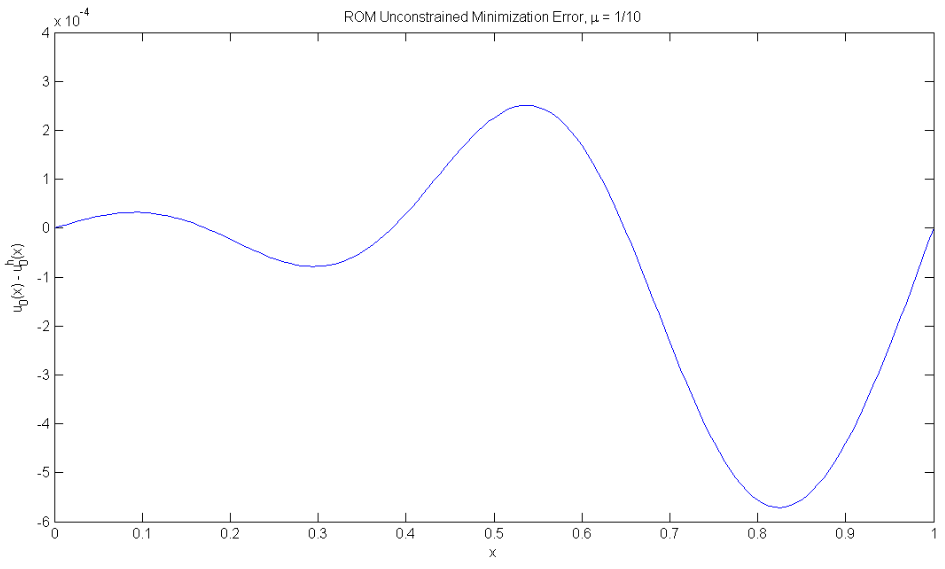

Figure 4.31: ROM inverse error for (4.7), $\mu = 1/10$

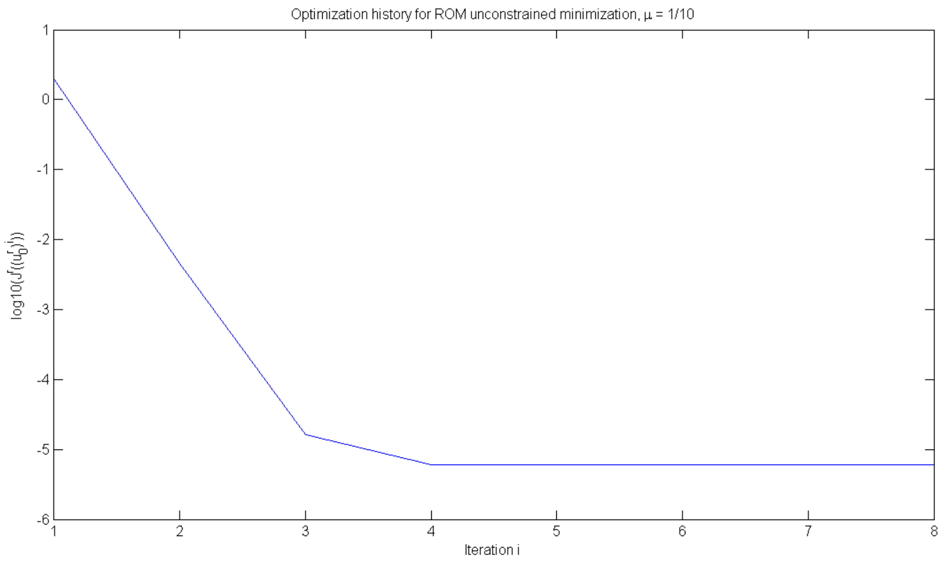

Figure 4.32: ROM minimization history for (4.7), $\mu = 1/10$

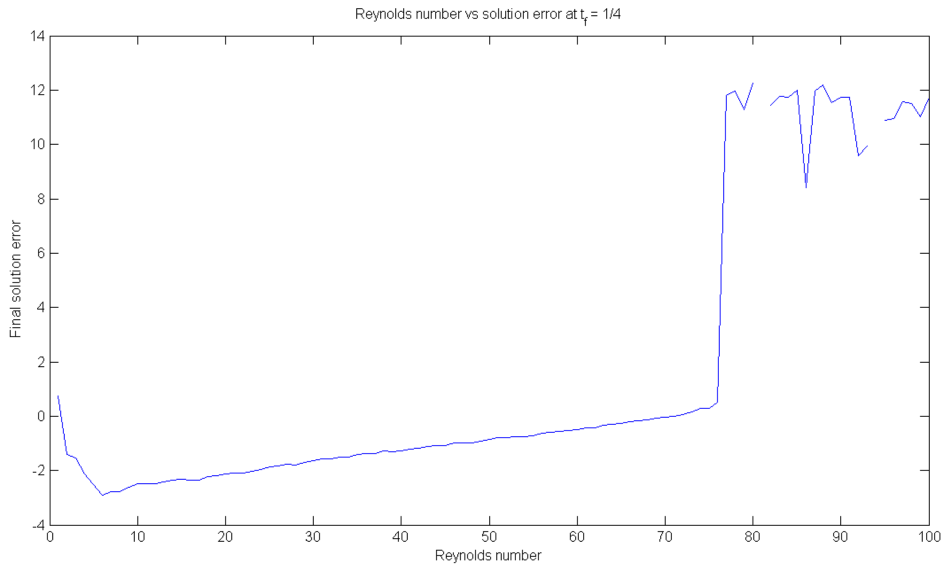

Figure 4.33: Reynolds number versus solution error for $t_f = 0.25$

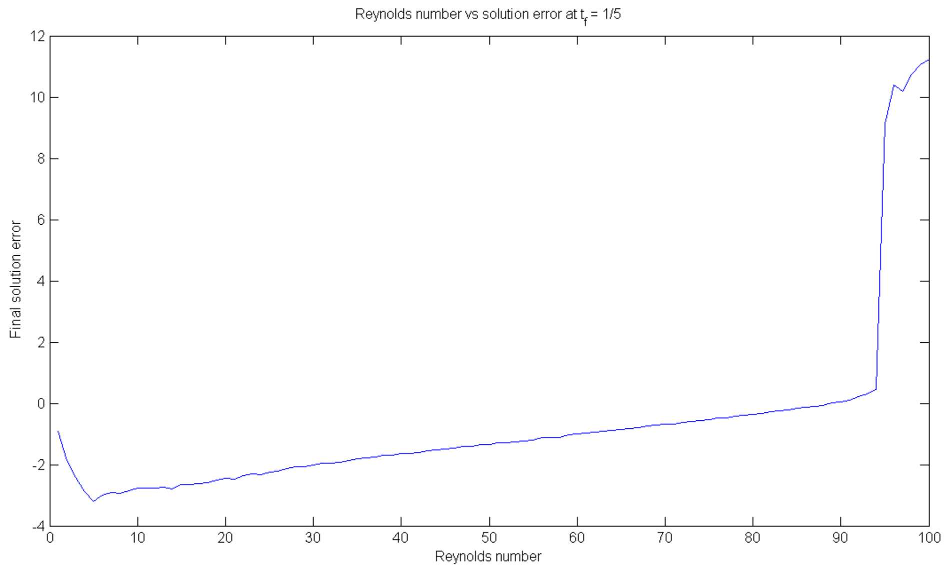

Figure 4.34: Reynolds number versus solution error for $t_f = 0.20$

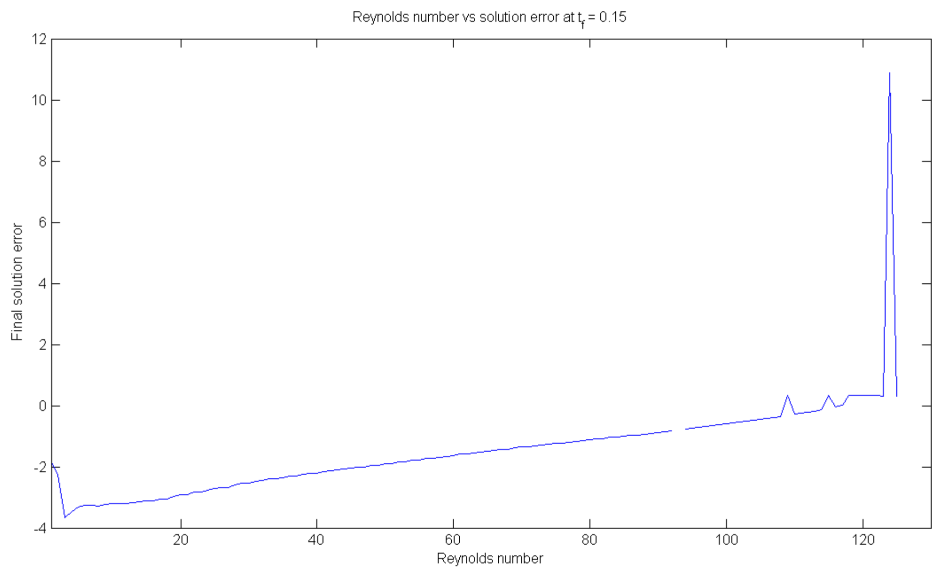

Figure 4.35: Reynolds number versus solution error for $t_f = 0.15$

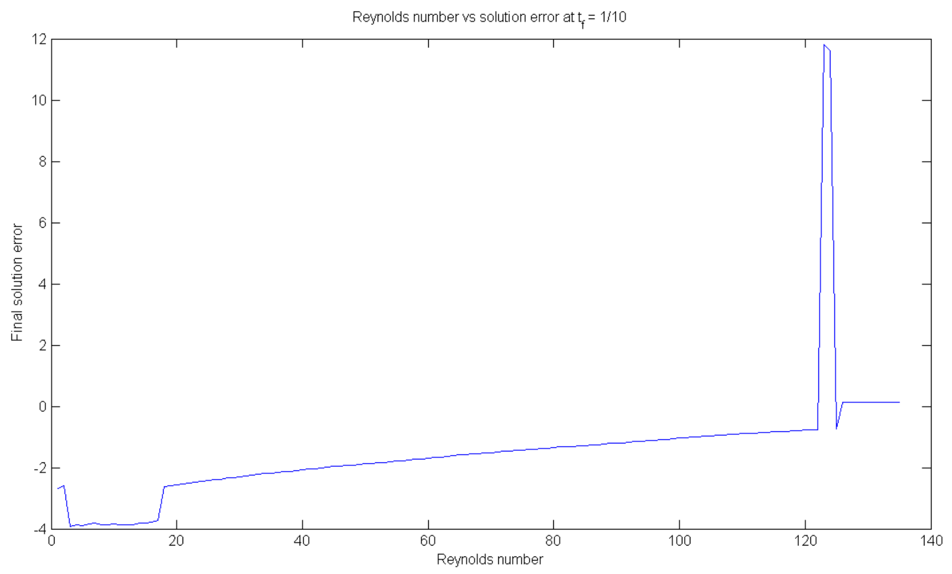

Figure 4.36: Reynolds number versus solution error for $t_f = 0.10$

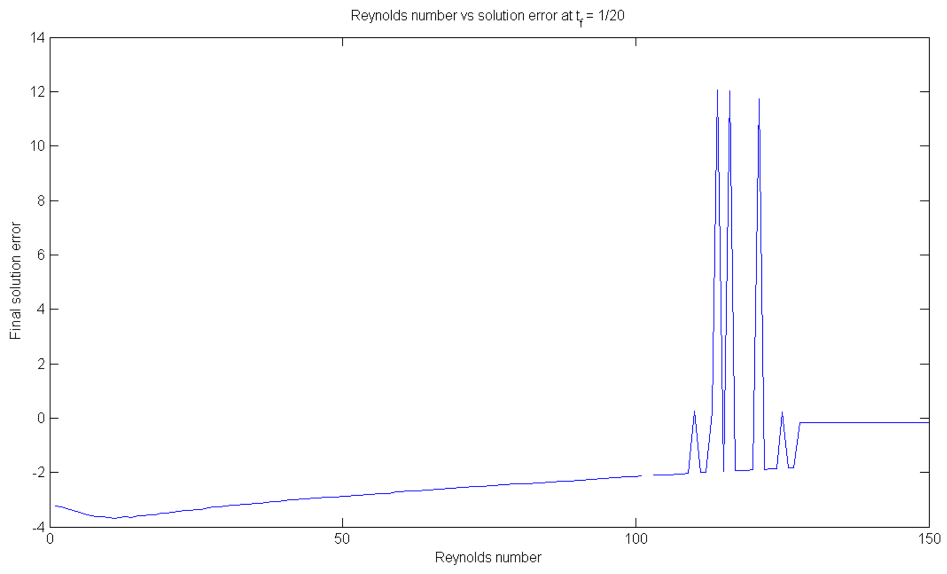

Figure 4.37: Reynolds number versus solution error for $t_f = 0.05$

Re = 1 until approximately Re = 100 where the method breaks down. This means that our method is only applicable for moderate Reynolds flows. However, as we have seen in the previous section, for flows where this method can be applied, we achieve surprisingly good results, especially when compared to the minimization against the full model. An example of this for $\mu = 1/40$ at $t_f = 0.25$ is shown in figure 4.38 through 4.40 for the problem (4.7).

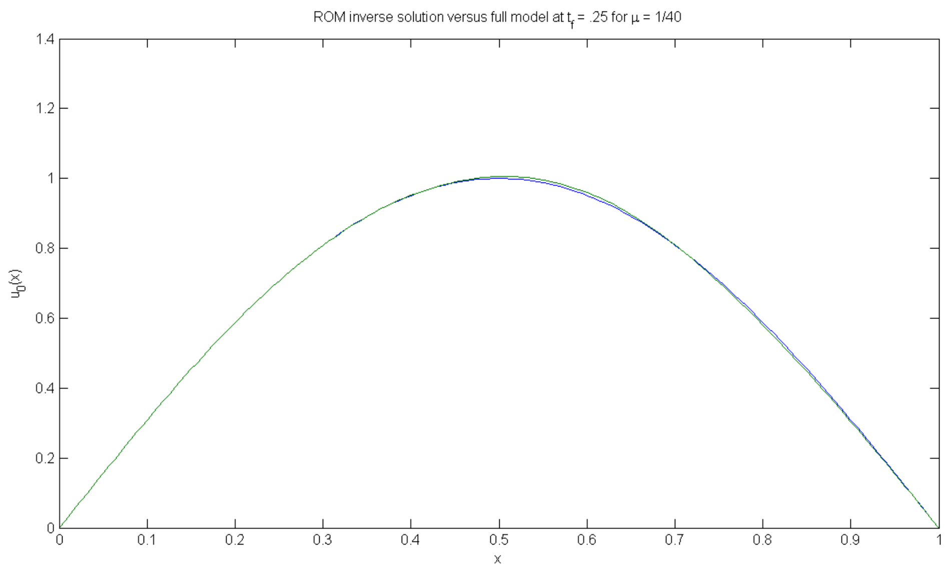

Figure 4.38: ROM model solution versus full model solution solution, $t_f = 1/4$, $\mu = 1/40$

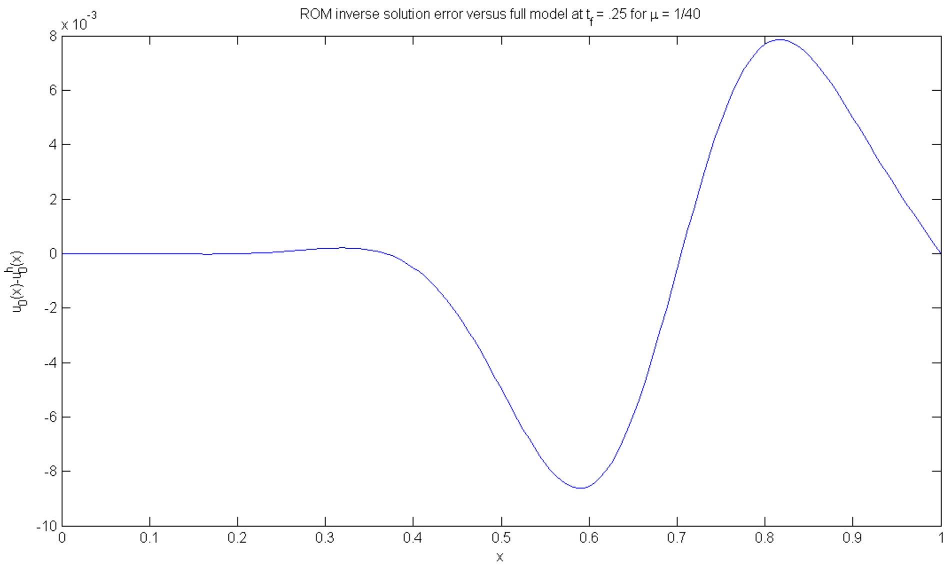

Figure 4.39: ROM model error versus full model solution, $t_f = 1/4$, $\mu = 1/40$

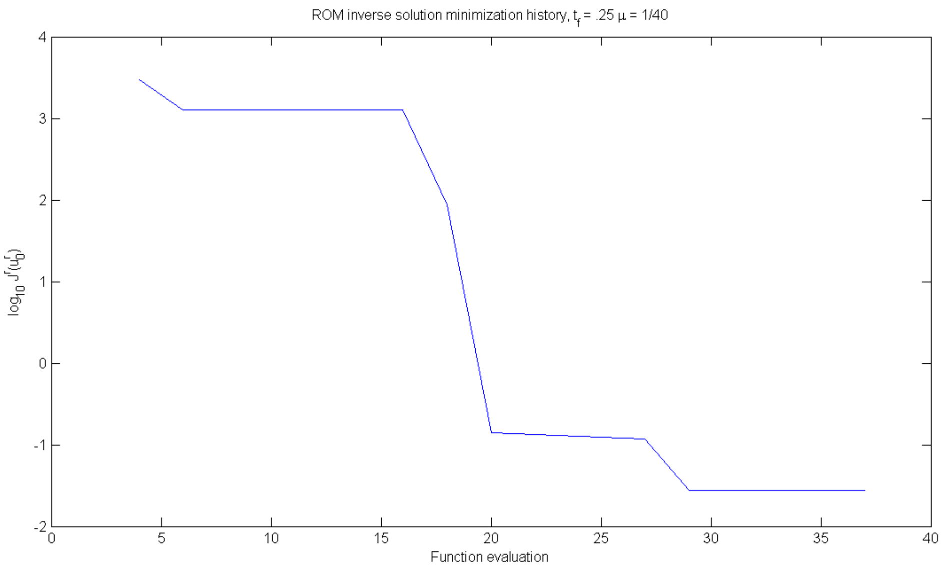

Figure 4.40: ROM model error versus full model minimization history, $t_f = 1/4$, $\mu = 1/40$

CHAPTER 5

CONCLUSION

In this thesis we have used the simple nonlinear 1D Burgers' equation to study optimal control of a viscous fluid flow. We have developed, compared and contrasted finite difference, finite element, and proper orthogonal decomposition methods. Our finite element methods were based on using a high-order polynomial space to solve the linearized Burgers' equation (1.7) before applying the inverse Cole-Hopf transform (1.5). From our results, we see that our finite element methods, especially CFEM-based methods, are able to achieve an extremely high degree of accuracy in a relatively short amount of time.

Basing our POD formulation on these highly accurate snapshots enables our POD method to also enjoy high numerical accuracy. Even so, the POD formulation remains very computationally efficient. Our high-order polynomial finite element formulation on the Cole-Hopf transformed equation proved to be a successful approach, reducing the model computation time by 97%. The two downsides to our approach are the loss of one degree of accuracy - not much of an issue for very high-degree polynomials - and the complication of the boundary/initial conditions - also not too much of an issue given the ability to either analytically or numerically compute the integral in (1.8).

Having derived an efficient solver, we tested our models on our simple cost functional-based optimal control problem. In this context, our POD basis – constructed from snapshots of every other timestep of the full model – proved to regularize the problem (1.8) with just one basis function and provide results far superior to the full model without the use of any an additional smoothing term. POD is thus successful on two levels in this problem: reducing the amount of time necessary to reach a solution as well as improving the numerical stability of our result. It is thus worthwhile to dedicate future study to the relationship between POD, Tikhonov, and other regularization techniques for inverse problems. The use of such methods for optimal control, especially on the full Navier-Stokes equations model, could easily form the basis of a full Ph. D. thesis.

Finally, we demonstrated that the model performance breaks down around a Reynolds number of 100, and therefore conclude that our method is only applicable for moderate Reynolds number flows. Given that this problem is an inverse ill-posed problem, this limitation is somewhat understandable. Improving the Reynolds number ranges for which this method can be applied is another area worthy further investigation.

REFERENCES

[1] L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, and E. Mischenko. *The Mathematical Theory of Optimal Processes.* Wiley Interscience Publishers, New York, 1962.

[2] R. Bellman. *Dynamic Programming.* Dover Publications, Mineola, N.Y., 1957.

[3] J. L. Lions. *Contrôle optimal de systèmes gouvernés par des équations aux dérivées partielles.* Dunod, Paris, 1968.

[4] J. L. Lions. *Optimal Control of Systems Governed by Partial Differential Equations.* Springer, Berlin, June 1971.

[5] E. Di Lorenzo, A. M. Moore, H. G. Arango, B. D. Cornuelle, A. J. Miller, B. Powell, B. S. Chua, and A. F. Bennett. Weak and strong constraint data assimilation in the inverse regional ocean modeling system (ROMS): development and application for a baroclinic coastal upwelling system. *Ocean Modelling*, 16(3-4):160–187, 2007.

[6] M. Bergmann, C.H. Bruneau, and A. Iollo. Enablers for robust POD models. *Journal of Computational Physics*, 228(2):516–538, February 2009.

[7] C. W. Rowley. Links between POD and balanced truncation. Presentation at CIMMS seminar, 2003.

[8] K. Willcox and J. Peraire. Balanced model reduction via the proper orthogonal decomposition. *American Institute of Aeronautics and Astronautics Journal*, 40(11):2323–2330, 2002.

[9] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2:559–572, 1901.

[10] I. T. Jolliffe. *Principal Component Analysis.* Springer, 2nd edition, October 2002.

[11] L. Sirovich. Turbulence and the dynamics of coherent structures. *Quarterly of Applied Mathematics*, 45(3):561–590, 1987.

[12] J. Favier, L. Cordier, and A. Kourta. Accurate POD Reduced-Order models of separated flows. *Physics of Fluids*, 8(3):259–265, December 2007.

[13] K. Kunisch and S. Volkwein. Control of the burgers equation by a Reduced-Order approach using proper orthogonal decomposition. *Journal of Optimization Theory and Applications*, 102(2):345–371, 1999.

[14] M. D. Gunzburger. Adjoint Equation-Based methods for control problems in incompressible, viscous flows. *Flow, Turbulence and Combustion*, 65(3):249–272, December 2000.

[15] M. D. Gunzburger. Reduced-Order modeling: Data compression and the design of experiments. Second DOE workshop on multiscale Mathematics, 2004.

[16] K. Veroy, C. Prud'homme, and A. T. Patera. Reduced-basis approximation of the viscous burgers equation: rigorous a posteriori error bounds. *Comptes Rendus Mathematique*, 337(9):619–624, November 2003.

[17] S. Volkwein. Proper orthogonal decomposition (POD) for nonlinear dynamical systems. DISC Summerschool 2005, 2005.

[18] J. Nocedal and S. Wright. *Numerical Optimization.* Springer, 2nd edition, July 2006.

[19] J. A. Atwell and B. B. King. Proper orthogonal decomposition for reduced basis feedback controllers for parabolic equations. *Mathematical and computer modelling*, 33(1-3):1–19, 2001.

[20] S. S. Ravindran. A reduced-order approach for optimal control of fluids using proper orthogonal decomposition. *International Journal for Numerical Methods in Fluids*, 34(5):425–448, 2000.

[21] F. Fang, C. C. Pain, I. M. Navon, G. J. Gorman, M. D. Piggott, P. A. Allison, P. Farrell, and A. J. H. Goddard. A POD reduced order unstructured mesh ocean modelling method for moderate reynolds number flows. *Ocean Modelling*, In press, 2009.

[22] D Kosambi. Statistics in function space. *Journal of Indian Mathematical Society*, 7:76–88, 1943.

[23] K Karhunen. Über lineare methoden in der wahrscheinlichkeitsrechnung. *Annales Academiae Scientiarum Fennicae Mathematica*, 37:1–79, 1947.

[24] M Loève. *Probability Theory*, volume 1. Springer, New York, 1978.

[25] K. Rektorys. *Variational Methods in Mathematics, Science and Engineering.* Springer, 2nd edition, 2001.

[26] S. S. Ravindran. Proper orthogonal decomposition in optimal control of fluids. *International Journal for Numerical Methods in Fluids*, 34:425–448, 1999.

[27] J. C. Tannehill, D. A. Anderson, and R. H. Pletcher. *Computational Fluid Mechanics and Heat Transfer.* Taylor & Francis, Washington, DC, 2nd edition, 1997.

[28] J.M. Burgers. *A Mathematical Model Illustrating the Theory of Turbulence*, volume 1, pages 171–196. Academic Press, New York, 1948.

[29] C. A. J. Fletcher. *Burgers' equation: A model for all reasons*, pages 139–225. North-Holland, New York, 1982.

[30] I. A. Hassanien, A. A. Salama, and H. A. Hosham. Fourth-order finite difference method for solving burgers' equation. *Applied Mathematics and Computation*, 170(2):781–800, November 2005.

[31] J.D. Cole. On a quasi linear parabolic equation occurring in aerodynamics. *Quarterly of Applied Mathematics*, 9:225–236, 1951.

[32] E. Hopf. The partial differential equation $u_t + uu_x + \mu u_{xx}$. *Communications on Pure and Applied Mathematics*, 3:201–230, 1950.

[33] W. L. Wood. An exact solution for burger's equation. *Communications In Numerical Methods in Engineering*, 22(7):797–798, 2006.

[34] T. D. Taylor, E. Ndefo, and B. S. Masson. A study of numerical methods for solving viscous and inviscid flow problems. *Journal of Computational Physics*, 9:99–119, 1972.

[35] P. L. Roe. The use of the riemann problem in Finite-Difference schemes. *Lecture Notes in Physics*, 141:354–359, 1980.

[36] P. L. Roe. Approximate riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, 43:357–372, 1981.

[37] R. W. MacCormack. The effect of viscosity on hypervelocity impact cratering. *AAIA Paper*, pages 69–354, 1969.

[38] T. Öziş, E. N. Aksan, and A Özdeş. A finite element approach for solution of burgers' equation. *Applied Mathematics and Computation*, 139(2-3):417–428, July 2003.

[39] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing.* Cambridge University Press, 3 edition, September 2007.

[40] I. M. Navon. POD model reduction of large scale geophysical models. Workshop on Industrial Applications of Low Order Models Based on Proper Orthogonal Decomposition (POD), 2008.

[41] The MathWorks. MATLAB® optimization toolbox v4.1 - documentation. http://www.mathworks.com/access/helpdesk/help/toolbox/optim/, 2008.

[42] J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965.

[43] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization.* Academic Press, San Diego, 1982.

[44] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright. Convergence properties of the Nelder-Mead simplex algorithm in low dimensions. *SIAM Journal of Optimization*, 9:112—147, 1996.

[45] D. N. Daescu and I. M. Navon. Efficiency of a pod-based reduced second order adjoint model in 4-D VAR data assimilation. *International Journal for Numerical Methods in Fluids*, 53:985–1004, 2007.

[46] A. G. Akritas, G. I. Malaschonok, and P. S. Vigklas. The SVD-Fundamental theorem of linear algebra. *Nonlinear Analysis: Modelling and Control*, 11(2):123–136, May 2006.

[47] A. K. Alekseev, I. M. Navon, and J. L. Steward. Comparison of advanced large-scale minimization algorithms for the solution of inverse ill-posed problems. *Optimization Methods and Software*, 24(1):63–87, 2009.

BIOGRAPHICAL SKETCH

Jeff Steward

Jeff Steward was born on September 9th, 1980. This date is the ninth day of the ninth month of a year that is a multiple of nine. For as long as he can remember, he always found this interesting. When he was young, Jeff often thought that his 19th birthday on 9/9/1999 would have been one of the best days in his life. Unfortunately, on that day he locked his keys in his office at IBM (where he was a software engineering intern) and had to walk nine miles home in the dark. From this, Jeff learned that numbers, by themselves, don't mean much.

Jeff's research interests include optimal control of partial differential equations, scientific visualization, fluid dynamics, numerical weather prediction, computational mycology and translating ancient Chinese texts.