Filter Based Stabilization Methods for Reduced Order Models of Convection-Dominated Systems

Ian R. Moore

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Mathematics

Traian Iliescu, Chair

Lizette Zietsman

Honghu Liu

April 28, 2023

Blacksburg, Virginia

Keywords: Differential Filter, Reduced Order Model, Leray Model, Evolve-Filter-Relax, Approximate Deconvolution

Filter Based Stabilization Methods for Reduced Order Models of Convection-Dominated Systems

Ian R. Moore

(ABSTRACT)

In this thesis, I examine filtering based stabilization methods to design new regularized reduced order models (ROMs) for under-resolved simulations of unsteady, nonlinear, convection-dominated systems. The new ROMs proposed are variable delta filtering applied to the evolve-filter-relax ROM (V-EFR ROM), variable delta filtering applied to the Leray ROM, and approximate deconvolution Leray ROM (ADL-ROM). They are tested in the numerical setting of Burgers equation, a nonlinear, time dependent problem with one spatial dimension. Regularization is considered for the low viscosity setting.

Filter Based Stabilization Methods for Reduced Order Models of Convection-Dominated Systems

Ian R. Moore

(GENERAL AUDIENCE ABSTRACT)

Numerical solutions of partial differential equations may not be able to be efficiently computed in a way that fully captures the true behavior of the underlying model, especially if significant changes in the solution occur over a very small spatial area. In this case, non-physical numerical artifacts may appear in the computed solution. We discuss methods of treating these calculations with a goal of improving the fidelity of numerical solutions with respect to the original model.

Dedication

To Daniel K. Moore

Acknowledgments

This material is partially based upon work supported by the National Science Foundation through grant DMS-2012253.

Material from this thesis is included in the paper [40] "Approximate Deconvolution Leray Reduced Order Model" by Anna Sanfilippo, Ian R. Moore, Francesco Ballarin, and Traian Iliescu.

Contents

List of Figures

Chapter 1

Introduction

In this thesis, we will consider a variety of techniques with the goal of creating new models for the stabilization of under-resolved complex flow in a context of reduced order modeling. Any numerical method for solving turbulent, multiscale flows will eventually run into the problem of resolution. That is, small scales which cannot be fully captured by a mesh that you can numerically afford have a significant impact on the quality of your solution. Reduced order modeling, and in particular the Galerkin reduced order model (G-ROM) lean into this issue by sacrificing the precise spatial resolution offered by the finite element method (FEM) for computational speed. The speedup factor is desirable in order to solve complicated flow patterns in real time, and may be necessary depending on the hardware available. For example, methods of solving models of human blood flow [3, 4, 23] may require the model to be operable on hospital available hardware due to time and privacy concerns, not state of the art super-computers.

The techniques which will be considered in this thesis fall into the category of regularization [32]. Regularized ROMs (Reg-ROMs) [47] are a type of ROM stabilization [5, 15]. ROM stabilizations are methods which seek to treat the ill effects of under-resolved ROMs, where the small scales (think of different sized vortices left behind by a paddle moving through water) in the multiscale simulation are not fully captured by the model reduction process. The loss of these small scales can have compounding effects on the accuracy of the computed ROM solution, which often present as non-physical oscillations in the computed solution.

Regularization is a symptom-based process which seeks to smooth away these oscillations which are often obscuring a reasonably accurate ROM solution. A common alternative to ROM regularization, which still falls into the stabilization framework are ROM closures. ROM closures seek to directly model the effects of what was left behind in creating a fast, low-dimensional model by adding a correction term to account for the under-resolved nature of the ROM. For more on closures, particularly for fluid flow, please see, e.g., [28, 42] and the review in [2]. Additionally, a closure based approach using the same setting of Burgers equation may be found in the following paper [1]. The advantages of regularization are that they offer generally simple implementations into pre-existing programs and require limited to no additional computation in the active solution stage.

Regardless of whether a regularization or closure based approach is taken, a typical goal of ROM stabilizations is to add the right amount of dissipation into the system to match the kinetic energy of the original system. In physical fluid flows, energy tends to dissipate at the smallest scales, with frictional forces dissipating tiny eddies. If an under-resolved reduced order model cannot resolve these small scales, then a standard G-ROM approach will tend to hang on to excess energy. This phenomenon has been shown in the following paper for the Navier-Stokes equations as justification for a filtering based regularization approach [47]. For closure approaches, it has been shown that the correction term mentioned above should dissipate energy from the system [11]. Following this, in closure models such as eddy viscosity ROMs [1, 46], an explicit dissipation term is added to account for the extra energy in the G-ROM. Alternatively, physical constraints might be added to a closure ROM, as was done in the following paper [34], which also have the effect of dissipating energy from the system. This thesis will follow a filtering based regularization approach, and takes considerable inspiration from large eddy simulations.

1.1 Why Galerkin Reduced Order Models?

In a computational landscape of ever-increasing capacity and big data growing ever bigger, it is fair to ask why we go back to these old models. Neural networks and other machine learning models have shown some promise in capturing complex dynamics [36], but there is also evidence of weaknesses and inflexibility [13, 16]. The methods which we are using are a clear evolution of numerical methods which date to the mid-20th century, and have been used in their current form for decades. The reason for going back to this is two-fold: firstly, in our ROM, data is used to formulate the model, not to run it. Once the model is formulated, we rely on physics and tested numerical methods to generate our results. Furthermore, there are things that can be proven about Galerkin ROMs that have yet to be shown for machine learning ROMs. Consistency, stability, and convergence are all desirable properties which can sometimes be observed in machine learning models, but without much proof. Combining data with traditional numerical methods offers some hope that these desirable properties might be proven.

This hybrid of data and physics is the core of the long-term future we hope to see with Galerkin type ROMs. Recent work in the field has seen improvements to the standard "black box" machine learning models by augmenting them with physics [39], and by hybridizing traditional ROM methods based on fundamental equations with machine learning on mathematically challenging nonlinear relationships [36]. The hope is that data-driven, physics informed models such as the Galerkin ROM can meet a compromise between the flexibility of fully data-driven machine learning models and the robustness of provably convergent methods of solving partial differential equations for complicated flow patterns.

1.2 The Numerical Setting

Our numerical experiments will be carried out in the setting of Burgers equation. This equation gives rise to a nonlinear, time dependent problem given in 1D as follows:

$$u_t - \nu u_{xx} + u \cdot u_x = f(x), \tag{1.1}$$

subject to boundary conditions and initial conditions. We will consider the unforced form such that $(x,t) \in [0,1] \times (0,1]$, $u(x,0) = u_0$, and $u(0,t) = u(1,t) = 0$. Burgers equation can display sharp boundary layers when ν is very small, and behaves similarly to a hyperbolic partial differential equation. Because of these reasons, and because of its nonlinearity, Burgers equation is a useful test case for stabilization of numerical methods, and is a relatively common test case for new methods [1, 29, 30].

The problem also shares some key characteristics with the Navier-Stokes equations, with a combination of convective terms ($u \cdot u_x$) and viscous terms (νu_{xx}), where in particular we think of the viscous terms as small. This setting is similar to the case of turbulent fluid flows. In a numerical simulation of fluid flows, simulations are often tested in settings of high Reynolds numbers, where the Reynolds number represents the ratio of inertial to viscous forces present in a particular flow. Burgers equation is not properly simulating a physical flow, so we do not have a Reynolds number for this problem, but the low viscosity setting we are working with (small values of ν) aims to establish a correspondence with the turbulent fluid flow environment we ultimately want to apply these models to.

Chapter 2

Overview of Methods

2.1 The Finite Element Method

A reduced order model (ROM) calculated through a Galerkin procedure is a data driven model which takes high dimension full order model (FOM) data and represents it using a low dimensional basis. This FOM can be data generated through finite elements, finite volumes, finite differences, observational data, or other methods, but in this thesis it is important that FOM refers to the finite element method (FEM). Sometimes the act of reduction can under-resolve the ROM and cause oscillations. However, ROMs are also frequently used when calculation of the underlying FEM is inherently difficult, and the underlying FEM solution may not always be fully resolved. This can lead to spurious oscillations in the FEM, particularly around sharp layers, which propagate into the ROM.

Returning to Burgers equation, we first use the finite element method to obtain a numerical solution. First, define a finite element trial space for u: $\mathcal{U}_h \subset H_0^1(0,1)$. Our objective now is to describe a prospective solution as a linear combination of functions in this space $\mathcal{U}_h$. In particular, for all numerical experiments, we choose $\mathcal{U}_h$ to be the standard N-dimensional Lagrangian linear basis with elements ϕ_i, $i = 1, \ldots, N$. Then, the finite element solution may be written,

$$u_h(t,x) = \sum_{i=1}^{N} c_i(t)\phi_i(x).$$

Choose the test space to be the same as the trial space, multiply by a test function v in the test space, and integrate by parts to obtain the weak form, where $(\cdot,\cdot)$ refers to the L^2 inner product,

$$(u_t, v) - \nu(u_x, v_x) + (u \cdot u_x, v) = (f, v) \qquad \forall v \in \mathcal{U}_h. \tag{2.1}$$

The first and second terms correspond to the FE mass and stiffness matrices. The third, tri-linear term, comprises the nonlinearity in the system. Then, replacing u with our finite element solution u_h and choosing our test functions to also be members of the linear basis, i.e., our test functions are of the form ϕ_j, $j = 1, \ldots, N$, the convective, tri-linear term $(u \cdot u_x, \phi_j)$ takes the form,

$$(u \cdot u_x, \phi_j) = \sum_{i=1}^{N} c_i(t) \sum_{k=1}^{N} c_k(t)(\phi_i \cdot \phi_{k_x}, \phi_j), \quad j = 1, \ldots N.$$

This form results in a tensor, where each entry in the tensor is a matrix, with one matrix for each particular basis function ϕ_j. The full assembly may be written as,

$$\sum_{i=1}^{N} c_i'(t)(\phi_i, \phi_j) + \nu \sum_{i=1}^{N} c_i(t)(\phi_i', \phi_j') + \sum_{i=1}^{N} c_i(t) \sum_{k=1}^{N} c_k(t)(\phi_i \cdot \phi_{k_x}, \phi_j) = (f, \phi_j),\ j = 1, \ldots N,$$

which may be written as a dynamical system in the following manner:

$$\mathbf{M_h}\dot{\mathbf{c}}(t) + \nu \mathbf{S_h}\mathbf{c}(t) + \mathbf{c}(\mathrm{t})^{\mathbf{T}}\mathbf{B^h}\mathbf{c}(t) = \mathbf{F}. \tag{2.2}$$

In equation (2.2), the term $\mathbf{M_h}$ refers to the finite element mass matrix, $\mathbf{S_h}$ the stiffness matrix, and $\mathbf{B^h}$ the finite element tensor, while $\mathbf{c}$ is the vector of spatially discretized coefficients which vary in time. The tensor product produces the required vector in the following manner: the tensor $\mathbf{B^h}$ contains N matrices, which we may label as $\mathbf{B^h}_i$, for $i = 1, \ldots, N$. To represent $\mathbf{c^T B^h c}$ as a vector $\mathbf{b}$ with N components, define $\mathbf{b}_i = \mathbf{c^T B^h}_i \mathbf{c}$ for $i = 1, \ldots, N$.

The resulting system (2.2) tends to display stiff behavior, so it is useful to solve the system implicitly. All subsequent numerical solutions are solved with a first order backward Euler method to improve stability concerns with low viscosity and coarse discretization. Newton's method is used to solve the resulting nonlinear system.

2.2 The Galerkin ROM

2.2.1 G-ROM Overview

The Galerkin reduced order model (G-ROM) is a projection based method that attempts to find the best low order representation of high-order input data. In particular, if the finite element method generates data from an N-dimensional space, the G-ROM constructs the best representation (see Section 2.2.2 for more context as to "best") of this data in an R-dimensional space, where $R < N$, ideally much less than N. This process will transform the sparse $N \times N$-dimensional FEM operators into dense $R \times R$-dimensional ROM operators, so R must be minimized if this is to be of benefit. In particular, we consider our ROM solution $u_r(x,t)$ to be a linear combination of R ROM basis functions:

$$u_r(x,t) = \sum_{i=1}^{R} a_i(t)\varphi_i^r(x).$$

The basis functions which are generated are global in nature, instead of local as in the finite element method. Furthermore, if the snapshot data is generated from a FEM simulation, then we can express the ROM basis functions as a linear combination of the FEM basis functions, that is,

$$\varphi_i^r = \sum_{j=1}^{N} c_{ij}\phi_j^h, \quad i = 1, \ldots, R,$$

where we define $\mathbf{c_i} = (c_{ij})_{j=1}^{N}$ as the i^{th} vector of ROM coefficients with N components. This implies that our G-ROM solution is also a member of the original finite element space $\mathcal{U}_h$.

One of the most important concepts in modeling and particularly in ROM for large parameter spaces or lengthy time spans is the "offline" versus the "online" stage. The main interest of this thesis is unsteady (time-dependent) problems, which requires two wholly separate stages of (i) offline spatial discretization, which converts the PDE into an ODE, and (ii) online time integration, which solves the ODE. In the offline stage, data is used to generate a basis which is ideally flexible enough to be used outside of the exact conditions of the data source, as well as to assemble operators that can be used in the online stage. The majority of the ROM time savings are found in the online stage, as time integrating a low order model is much faster than integrating a high order model.

2.2.2 Proper Orthogonal Decomposition

Proper orthogonal decomposition (POD) is a common method of obtaining such set of basis functions $\varphi_i^r(x)$ for $i = 1, \ldots, R$, which we briefly discuss here. POD is a data-driven method that uses snapshots $\mathbf{Y} = [\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_m}]$ of spatial data of dimension N at m distinct times to construct a low-dimensional orthogonal basis that best represents the data with respect to a norm. There are several algorithms capable of generating a POD basis. For more detail, consult [45]. We will compute the basis by solving the eigenvalue problem,

$$\mathbf{Y}^{\mathrm{T}}\mathbf{Y}v_i = \lambda_i v_i. \tag{2.3}$$

Then, ROM basis coefficients chosen such that the $\mathbf{c}_i$, the i^{th} vector of ROM basis coefficients satisfies,

$$\mathbf{c}_i = \lambda_i^{-1/2}\mathbf{Y}v_i, \quad i = 1, \ldots, R,$$

can be shown to satisfy the minimization and orthogonality constraints we require to best represent the data contain in our snapshot matrix [45]. In particular, the vectors $\mathbf{c}_i$ satisfy the minimization problem,

$$\min_{\psi_1^r, \psi_2^r, \ldots, \psi_R^r,} \sum_{j=1}^{m} \|y_j - \sum_{i=1}^{R} \langle y_j, \psi_i^r \rangle \psi_i^r \|_{\mathbb{R}^N}^2,$$

subject to $\langle \psi_i^r, \psi_j^r \rangle_{\mathbb{R}^N} = \delta_{ij}$ for every R dimensional set of mutually orthogonal vectors in $\mathbb{R}^N$, where N is the FE dimension [45].

The eigenvalues of this system (2.3) are guaranteed to be non-negative because $\mathbf{Y}^{\mathrm{T}}\mathbf{Y}$ is symmetric semi-positive-definite, and can be ordered in decreasing order. In particular, this method (which is called the method of snapshots in literature [41]) has a clear connection to the SVD of the snapshot matrix $\mathbf{Y}$, as singular values of $\mathbf{Y}$ are the square roots of the eigenvalues of $\mathbf{Y}^{\mathrm{T}}\mathbf{Y}$. The SVD gives an ordered list of right singular vectors vectors which represent the directions of maximum action of the data matrix $\mathbf{Y}$.

The ROM basis functions associated with the larger eigenvalues are associated with the more dominant structures contained in the underlying data, and in particular, the ROM is constructed when we choose $R \ll N$, such that we use only the modes corresponding to the highest energy share of the system.

2.2.3 ROM Operators

Once the basis is assembled through POD, the G-ROM is solved at the same weak level as the FEM solution, merely with different basis functions and a different dimension. That is,

$$\sum_{i=1}^{N} a_i'(t)(\varphi_i, \varphi_j) + \nu \sum_{i=1}^{N} a_i(t)(\varphi_i', \varphi_j') + \sum_{i=1}^{N} a_i(t) \sum_{k=1}^{N} a_k(t)(\varphi_i \cdot \varphi_{k_x}, \varphi_j) = (f, \varphi_j),\ j = 1, \dots R, \tag{2.4}$$

which assembles into a system as,

$$\mathbf{M_r}\dot{\mathbf{a}}(t) + \nu \mathbf{S_r}\mathbf{a}(t) + \mathbf{a}(\mathrm{t})^{\mathrm{T}}\mathbf{B_r}\mathbf{a}(t) = \mathbf{F_r}. \tag{2.5}$$

We note that because POD yields an orthonormal basis, $\mathbf{M_r}$ reduces to the $(R \times R)$ dimension identity matrix. There is no such convenient property for the ROM stiffness matrix and tensor, both of which will be dense, but the low dimension of these operators is enough to reduce computation time significantly in the time-integrating 'online' stage of the ROM.

2.3 Differential Filter and Approximate Deconvolution

Several recent papers have considered filtering methods applied to reduced order modeling. As the technique is very general, it can be applied in a wide variety of settings, such as evolve-filter-relax methods [35, 43, 47], or Leray-type methods applied to a variety of scenarios such as Galerkin ROMs in a convection context [26, 44], as we will be further considering in this thesis, or other methods such as closure modeling [38, 42, 49] or the finite volume method [18]. The flexibility and adaptability of filtering based methods make them an attractive topic of research for stabilizing issues that arise when modeling fluid flows. In general, the goal with

differential filtering is to smooth out numerical oscillations and noise that is either present in the data or generated by the model. Current research has investigated filter-based Leray methods with constant radius at a ROM level [18, 47, 48], but to our knowledge not using variable delta filtering which will be introduced later.

2.3.1 Constant Radius Filter

The constant radius delta filter that we will be using is a simple differential filter, given for generic functions $u, \overline{u} \in L^2(0,1)$ and $\delta \in \mathbb{R}$ as,

$$(I - \delta^2 \Delta)\overline{u}(x) = u(x), \tag{2.6}$$

where I is the identity operator and Δ is the Laplacian operator. This is a very common filtering option for numerical methods, including simulation of fluids [37] as well as in a ROM context for numerical analysis of EFR methods [12] and applied to the Navier-Stokes equations [44, 47]. This filter enforces our requirement that our filtered function, $\overline{u}$, is a spatially smoother version of the unfiltered function u. Here, δ is a parameter which regulates the strength of the filter [35]. If $\delta = 0$, the filtered and unfiltered functions are the same. As δ grows larger, the filtered function $\overline{u}$ becomes more and more smooth as compared to the unfiltered function u. In our case, the particular filter described in Equation (2.6) may be problematic, as we require that $u(x,t) = \sum_{i=1}^{N} c_i(t)\phi_i(x)$, where $\phi_i(x)$ may be a piecewise linear function, and in fact is for our numerical experiments. Instead of satisfying this relationship, we satisfy a weak form.

Here, we stress that the filter is a *general* method, not a specific one restricted to one type of problem or setting [14]. First, consider the case where $\overline{u}_h, v_h \in \mathcal{U}_h$ to be the finite

element trial and test functions originating from the same space (this is valid in the case of homogeneous boundary conditions, as we are considering in this paper). Then, multiplying our filter by a test function and integrating by parts, we look for a solution $\overline{u}_h$ that satisfies,

$$(\overline{u}_h, v_h) + \delta^2(\overline{u}'_h, v'_h) = (u, v_h) \quad \forall v_h \in \mathcal{U}_h, \overline{u}_h(0) = \overline{u}_h(1) = 0. \tag{2.7}$$

This is a typical 2-point BVP, where the left side can be shown to be coercive and continuous, and the right side can be shown to be continuous, hence the weak form has a unique solution in $\mathcal{U}_h$ by the Lax-Milgram theorem. See [31] pages 192-193 for a similar application of the Lax-Milgram theorem. Additionally, this exact filter operator (2.7) with zero boundary conditions has been shown to be self-adjoint in [49].

If we choose a standard Lagrangian space composed of functions $\phi_j^h(x),\ j = 1, \ldots, N$, then we are seeking a solution $\overline{u}_h$ such that,

$$\overline{u}_h(t, x) = \sum_{i=1}^{N} \overline{a_i(t)} \phi_i^h(x).$$

This is similar to the approach used by [47]. In keeping with our spatial discretization and ODE solver in time, the constant is allowed to vary in time but the basis functions are not. Returning to equation (2.7),

$$(\overline{u}_h, v_h) + \delta^2(\overline{u}'_h, v'_h) = (u, v_h) \quad \forall v_h \in \mathcal{U}_h,$$

$$(\sum_{i=1}^{N} \overline{a_i(t)} \phi_i^h(x), \phi_j^h(x)) + \delta^2(\sum_{i=1}^{N} \overline{a_i(t)} \phi_i'^h(x), \phi_j'^h(x)) = (\sum_{i=1}^{N} c_i \phi_i^h(x), \phi_j^h(x)) \quad \text{for } j = 1, 2, \ldots, N$$

$$(M_h + \delta^2 S_h)\overline{\mathbf{a}(t)} = M_h \cdot \mathbf{c}(t).$$

That is, given a known solution, we can compute the filtered coefficients at any particular

time by a linear solve with guaranteed existence and uniqueness.

This works in an identical manner at a ROM level through the secondary Galerkin projection onto the low dimensional space. In this case, the standard ROM solution is of the form,

$$u_r(x,t) = \sum_{i=1}^{R} a_i(t)\varphi_i^r(x),$$

and we seek a filtered solution of the form,

$$\overline{u}_r = \sum_{i=1}^{R} \overline{a_i(t)}\varphi_i^r(x).$$

This is again similar to the formulation in [47]. The formulation for the numerical solve is exactly the same as for the finite elements solve on a lower dimension:

$$(M_r + \delta^2 S_r)\overline{\mathbf{a}(t)} = M_r \cdot \mathbf{c}(t). \tag{2.8}$$

Even though this is a paper about ROM methods, not FEM methods, we have spent the time establishing this filter at a FEM level because being able to use the filter at either level is convenient depending on the application. For EFR methods, where the solution is filtered potentially after each ODE time integration step, it is much more efficient to do this at a low dimensional ROM level. For the Leray or ADL model, having access to this filter at a finite element level allows one to pre-compute the filtered operators, shifting the cost into the offline stage. Furthermore, the FEM filter allows us access to local spatial information that is not present at a ROM level, which allows us to use the variable delta filter we will introduce later.

2.3.2 Pre-Assembly of Filtered Operators

We pause for a moment to give an example useful in computation and in understanding of the filter behavior for complicated systems. The Burgers Leray model with 0 source term is given by,

$$u_t - \nu u_{xx} + \overline{u} \cdot u_x = 0. \tag{2.9}$$

At a ROM level, one implementation choice is to compute the low-dimensional, nonlinear convective operator at each time step. An alternative is to pre-assemble by using the fact that the ROM basis functions are themselves members of the finite element trial space, and can thus be filtered themselves. (This requires more care when boundary conditions are non-homogeneous.)

Let us consider only the computation of the nonlinear filtered term at a ROM level. Then, multiplying by a test functions $\varphi_k^r(x)$ and integrating over the domain,

$$\begin{aligned}
(\overline{u} \cdot u_x, \varphi_k^r) &\to \left(\overline{\sum_{i=1}^{r} c_i(t)\varphi_i^r(x)} \cdot \sum_{j=1}^{r} c_j(t)\varphi_j^r(x)_x, \varphi_k^r(x) \right) k = 1, \ldots, r \\
&= \left(\sum_{i=1}^{r} c_i(t)\overline{\varphi_i^r(x)} \cdot \sum_{j=1}^{r} c_j(t)\varphi_j^r(x)_x, \varphi_k^r(x) \right) k = 1, \ldots, r \qquad (2.10) \\
&= \sum_{i=1}^{r} c_i(t) \sum_{j=1}^{r} c_j(t) \left(\overline{\varphi_i^r(x)} \cdot \varphi_j^r(x)_x, \varphi_k^r(x) \right) k = 1, \ldots, r \qquad (2.11) \\
&= \mathbf{c}^r(t)^T \cdot \overline{\mathbf{B_r}} \cdot \mathbf{c}^r(t).
\end{aligned}$$

The most important thing to keep in mind when following these calculations is that the filter we are using is a linear, *spatial* filter, not time dependent. If you can separate parts of the function you seek to filter as depending exlusively on t, you may safely pull them out of the filter as you would any coefficient to a linear solve, as was done in equation (2.10).

Equation (2.11) is the most instructive line for actually accomplishing the pre-assembly. The matrix $\overline{\mathbf{B_r}}$ should be thought of as the representation of the full dimensional nonlinear tensor $\mathbf{B_h}$ in the low dimensional ROM space using a combination of filtered and unfiltered basis functions. This hybrid approach should ideally curb spurious oscillations in the full system, while limiting the excessive smoothing that could easily happen if we computed the tensor with only filtered basis functions.

2.3.3 Lavrentiev Regularized Approximate Deconvolution

Approximate deconvolution is a method which attempts to avoid the often ill-posed problem of signal inversion. Our desired application here is to give a secondary lever of control over the filtering process. It can be easy to over-smooth with differential filtering, and the energy that we are looking to capture may be excessively diffused or spatially delayed by the filtering process. Approximate deconvolution is a compromise between non-physically noisy solutions and overly smoothed solutions, where approximate deconvolution is used in our case to avoid the ill-conditioning of exact deconvolution. We will be solving linear systems for our filtering, which are well-posed, but well-posed problems need not be well-conditioned. Exact deconvolution is typically not well conditioned, being highly sensitive to small perturbations [8]. Approximate deconvolution is a tested technique for general modeling purposes [17], in particular LES modeling [6, 9, 24], and in a ROM context once to our knowledge [49] for NSE flow past a cylinder with Re = 1,000. In that paper, the authors filtered both terms of the convective part of the equation (in the context of Burgers equation, filtering the term $u \cdot u_x$). We will apply this approximate deconvolution idea to a Leray type model where only one component of the convective term is filtered.

We will now apply a particular form of approximate deconvolution based on the constant

radius filter discussed in the previous sections.

Given some filtered signal,

$$\overline{u} = Gu,$$

for known operator G, a standard image processing technique [8] that has been used in ROMs previously [49] is to use approximate deconvolution to recover the original signal:

$$u^{AD} = (G + \mu I)^{-1}\overline{u}. \tag{2.12}$$

This avoids the typically ill-conditioned problem of exact deconvolution. Lavrentiev regularization is not the only form of approximate deconvolution, as Tikhonov or Van Cittert methods may also be used at this stage.

First, recall that we are using filter described in Equation (2.6):

$$\begin{aligned}(I - \delta^2\Delta)\overline{u}(x) &= u(x)\\ \overline{u}(x) &= (I - \delta^2\Delta)^{-1}u(x).\end{aligned}$$

Then, using the Lavrentiev method (Equation (2.12)), we can identify that G is the operator $(I - \delta^2\Delta)^{-1}$ and determine how to solve for $u^{AD}(x)$:

$$\begin{aligned}u^{AD}(x) &= ((I - \delta^2\Delta)^{-1} + \mu I)^{-1}\overline{u}(x)\\ ((I - \delta^2\Delta)^{-1} + \mu I)u^{AD}(x) &= \overline{u}(x)\\ (I + \mu(I - \delta^2\Delta))u^{AD}(x) &= (I - \delta^2\Delta)\overline{u}(x)\\ (I + \mu I)u^{AD}(x) - \mu \cdot \delta^2\Delta u^{AD}(x) &= (I - \delta^2\Delta)\overline{u}(x).\end{aligned} \tag{2.13}$$

At a numerical level, we take the inner product defining the finite element or Galerkin ROM projection, multiply by a test function, integrate by parts, and satisfy the resulting weak form of Equation (2.13) for every function in our test space.

As an example, suppose that we seek to calculate the filtered and AD versions of a particular ROM basis function $\varphi_i^r(x)$. These ROM basis functions live in the FE space, and so do the solutions we seek. Then, write,

$$\overline{\varphi_i^r}(x) = \sum_{j=1}^{N} \overline{c}_{ij}\phi_j(x), \qquad AD(\varphi_i^r) = \sum_{j=1}^{N} c_{ij}^{AD}\phi_j(x). \tag{2.14}$$

Substituting in $\overline{\varphi_i^r}(x)$ for $\overline{u}(x)$ and $AD(\varphi_i^r)$ for $u^{AD}(x)$ in Equation (2.13), we multiply by each of our FE test functions and integrate by parts to obtain the following system:

$$\left((1+\mu)\mathbf{M}_h + \mu\cdot\delta^2\mathbf{S}_h\right)\mathbf{c}^{AD} = \left(\mathbf{M}_h + \delta^2\mathbf{S}_h\right)\overline{\mathbf{c}}, \tag{2.15}$$

where $\overline{\mathbf{c}}$ is a known vector of filtered basis coefficients. The entries are determined by $\overline{c}_{ij}$ from Equation (2.14), and we solve for the vector of unknown AD coefficients.

This works in the same manner at a ROM level by replacing the FE matrices with ROM matrices, though in this case the coefficients you find would be those determining the linear combination of ROM basis functions, that is,

$$AD\left(u_r(x,t)\right) = \sum_{i=1}^{R} a_i^{AD}(t)\varphi_i^r(x).$$

Because most of our Leray type models will have pre-assembled operators that depend on filtering of the ROM basis functions, let us compare the effects of filtering on a representative basis function.

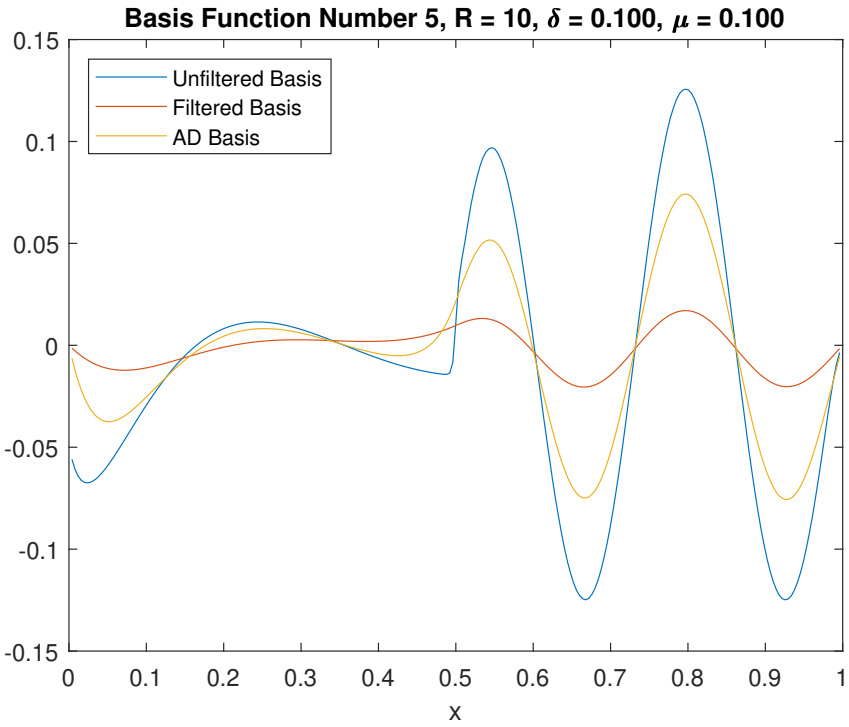


Figure 2.1: Effects of filtering and approximate deconvolution.

With the above parameters, the constant radius filter strongly diffuses the basis functions, while approximate deconvolution dials back this filtering to give us a smoother version of the original basis function.

2.3.4 Variable Radius Differential Filter

One drawback of the filtering and pre-assembly methods which have so far been described is that they filter global functions, such as the ROM basis functions or actual solution functions in an equal manner across space. However, it may not be the case that these functions need to be filtered everywhere. If our solution or some basis function is performing well in a certain spatial area, it is detrimental to ROM performance to filter either of them. We will accomplish spatial filtering by making an explicit connection between the FE model and

the ROM model using the assumption that the ROM basis functions, and hence the ROM solution, are themselves linear combination of FE basis functions. This is often a useful assumption to make in the POD process, but a typical ROM does not have much use for the FE data after the process of building the ROM.

First, let us consider a natural extension to the differential filter described in Equation (2.6) to accept a continuous delta function:

$$(I - \delta^2(x)\Delta)\overline{u}(x) = u(x). \tag{2.16}$$

Unfortunately, there are reasons to believe this operator is not self-adjoint without significant assumptions on the function $\delta(x)$. Following the procedure in [49] for the self-adjointness of the constant radius filter described in Equation (2.6), which was also proved in [27], let $u, v \in L^2(0,1)$. We seek a filtered solution $\overline{u} \in L^2(0,1)$ satisfying Equation (2.16) such that $\overline{u}(0) = \overline{u}(1) = 0$. Multiplying Equation (2.16) by $\overline{v}$ such that $\overline{v}(0) = \overline{v}(1) = 0$ and integrating by parts, where $\langle \cdot, \cdot \rangle$ here refers to the $L^2(0,1)$ inner product for parenthetical clarity,

$$\begin{gathered}\langle \overline{u}(x), \overline{v}(x) \rangle + \langle \overline{u}'(x), 2\delta(x)\delta'(x)\overline{v}(x) + \delta^2(x)\overline{v}'(x) \rangle = \langle u(x), \overline{v}(x) \rangle \\ \langle \overline{u}(x), \overline{v}(x) \rangle + \underline{\langle \overline{u}'(x), 2\delta(x)\delta'(x)\overline{v}(x) \rangle} + \langle \overline{u}'(x), \delta^2(x)\overline{v}'(x) \rangle = \langle u(x), \overline{v}(x) \rangle.\end{gathered} \tag{2.17}$$

In contrast, multiplying the filter

$$(I - \delta^2(x)\Delta)\overline{v}(x) = v(x), \quad \overline{v}(0) = \overline{v}(1) = 0,$$

by $\overline{u}$ and integrating by parts yields a similar form:

$$\langle \overline{v}(x), \overline{u}(x) \rangle + \underline{\langle \overline{v}'(x), 2\delta(x)\delta'(x)\overline{u}(x) \rangle} + \langle \overline{v}'(x), \delta^2(x)\overline{u}'(x) \rangle = \langle v(x), \overline{u}(x) \rangle. \tag{2.18}$$

In the context of the $L^2(0,1)$ inner product, the problematic terms are the underlined pair in equations (2.17) and (2.18), where the derivative we are taking is swapped between functions $\overline{v}$ and $\overline{u}$ in the adjoint Equation (2.18) vs. the original filter in Equation (2.17). The simplest way to alleviate this issue is to consider $\delta(x)$ to be a piecewise constant function with a finite number of discontinuities. If $\delta(x)$ is such a function, then we can break up the integral over the domain. That is, for p discontinuities $0 < x_1, \ldots, x_p < 1$ in the piecewise constant $\delta(x)$,

$$\underline{\langle \overline{v}'(x), 2\delta(x)\delta'(x)\overline{u}(x)\rangle} = \int_0^1 \overline{v}'(x) \cdot 2\delta(x)\delta'(x)\overline{u}(x)dx$$
$$= \int_0^{x_1} \overline{v}'(x) \cdot 2\delta(x)\delta'(x)\overline{u}(x)dx + \cdots + \int_{x_p}^1 \overline{v}'(x) \cdot 2\delta(x)\delta'(x)\overline{u}(x)dx,$$

where $\delta'(x) = 0$ on each sub-interval, eliminating this term. The remaining terms in equations (2.17) and (2.18) are the same by standard integral properties. This piecewise constant requirement on $\delta(x)$ provides an idea of how to proceed for the particular problem of filtering FE functions. However, note the above discussion is for motivational purposes, and applying the variable filter from Equation (2.16) with suitably chosen $\delta(x)$ is an alternative method to the approach used in this thesis.

To filter our FE functions, recall that we consider the ROM solution $u_r(x,t)$ to be a linear combination of ROM basis functions:

$$u(x,t) = \sum_{i=1}^{R} a_i(t)\varphi_i^r(x).$$

The ROM basis functions (φ_i^r) are constructed as a linear combination of the FE basis functions, (ϕ_i^h), through POD:

$$\varphi_i^r(x) = \sum_{j=1}^{N} c_{ij}\phi_j^h(x),$$

and we obtain the filtered ROM basis functions, $(\overline{\varphi}_i^r(x))$, as a linear combination of filtered

FE basis functions *using the same coefficients* c_{ij}, because the filter affects spatial functions only:

$$\overline{\varphi}_i^r(x) = \sum_{j=1}^{N} c_{ij}\overline{\phi}_j^h(x).$$

The filtered FE basis functions are also a linear combination of the unfiltered FE basis functions, i.e.,

$$\overline{\phi}_i^h(x) = \sum_{k=1}^{N} \overline{b}_{ki}\phi_k^h(x). \tag{2.19}$$

It is at this point that we look approximate a variable level of piecewise constant filtering as suggested in filter of Equation (2.16) by assigning each FE function an individual level of filtering such that,

$$(I - \delta_i^2\Delta)\overline{\phi}_i^h(x) = \phi_i^h(x), \quad i = 1, \dots N. \tag{2.20}$$

Now, δ_i is a constant which is allowed to discretely vary based on the index i. For any particular i, the weak form of this operator is exactly the same as the weak form of the constant radius filter in Equation (2.7), which implies that it is both self-adjoint [27, 49] and has a unique solution in the FE space by the Lax-Milgram theorem as with the constant radius filter.

Filtering at a FE level gives us access to the local FE functions, which allows us to specifically target portions of our domain which are behaving poorly. Going from operator in Equation (2.20) to the matrix form of the weak equation, we solve the system,

$$\left(\mathbf{M}_h + \delta_i^2\mathbf{S}_h\right)\overline{\mathbf{b}}_i = \mathbf{M}_h \cdot \mathbf{e}_i, \quad i = 1, \dots, N. \tag{2.21}$$

Above, e_i refers to the i^{th} standard basis vector for $\mathbb{R}^N$, and $\overline{\mathbf{b}}_i$ is a vector of FE coefficients satisfying Equation (2.19). Algorithm 1 provides a way to compute a filtered variable delta ROM basis for the purpose of pre-assembling filtered operators, as discussed in Section 2.3.2.

First, this algorithm assembles the finite dimensional representation of the filtered FEM basis functions on a discrete level as $\overline{\boldsymbol{\phi}^h}$, an $N \times N$ matrix. Secondly, we assemble the finite dimensional representation of the filtered ROM basis functions in the $N \times R$ matrix $\overline{\boldsymbol{\varphi}}^r$. Recall that $\mathbf{c}_j$ refers to the j^{th} column of the $N \times R$ matrix that contains the coefficients to assemble the ROM basis functions. These filtered and unfiltered matrices allow us to move between the ROM and FEM spaces. See Figure 3.2 for the effect this has on the ROM basis.

Note that $\overline{\mathbf{b}}_i$ is a list of coefficients, not a function. If some calculation requires the values of the filtered functions in between node points, then it is necessary to return to the proper interpretation of $\overline{\mathbf{b}}_i$ as FE coefficients.

Algorithm 1 could be made computationally more efficient by setting a tolerance ϵ for δ_i, such that if $\delta_i < \epsilon$, no filtering is performed. The motivation for this choice is that, for very small values of δ_i, the filtering is negligible, such that $\overline{\phi_i^h} \approx \phi_i^h$. This formulation could add up to significant time savings for large values of N. However, for our test problems, we chose problems that were simple enough to not require this additional efficiency. In particular, this process could be automated by means of an appropriate indicator function that determines whether or not filtering needs to be done [7, 32]. The most robust form of this variable delta could use such an indicator function to change the filtered basis coefficients efficiently in real time, though this would only be useful in a context where the filtered operators are not pre-assembled, such as EFR. Some research has been done on indicator functions with applications to ROMs [19], particularly in an EFR context, but this is not used here.

When doing computations for a variable delta filter, note particularly that we have converted the strictly local FE basis functions into global basis functions. Care must be taken when doing further computations which are usually done in an element by element way, such as comparing error of the reconstructed ROM solution in an L^2 sense.

Algorithm 1 Computing ROM basis with Variable Delta Filtering

Define $\overline{\boldsymbol{\phi}^h}$ to be the matrix whose columns contain $N \times 1$ vectors representing the filtered FE basis functions at FE nodes. Let $\mathbf{c}_j$ be the $N \times 1$ vector with coefficients such that: $\varphi_j^r(x) = \sum_{i=1}^n c_{ij}\phi_i^h(x)$.

$i \leftarrow 1$

while $i \leq N$ **do**

$\delta_i \leftarrow \delta(x_i)$

Solve $(\mathbf{M_h} + \delta_i^2\mathbf{S_h})\overline{\mathbf{b}}_i = \mathbf{M_h}e_i$ ▷ e_i is i^{th} identity vector

$\overline{\boldsymbol{\phi}^h} \leftarrow [\overline{\boldsymbol{\phi}^h},\ \overline{\mathbf{b}}_i]$

$i \leftarrow i + 1$

end while

$j \leftarrow 1$

while $j \leq R$ **do**

$\overline{\varphi}_j^r \leftarrow \overline{\boldsymbol{\phi}^h} \cdot \mathbf{c}_j$

$\overline{\boldsymbol{\varphi}^r} \leftarrow [\overline{\boldsymbol{\varphi}^r},\ \overline{\varphi}_j^r]$

$j \leftarrow j + 1$

end while

2.4 Evolve-Filter-Relax ROM

The Evolve-Filter-Relax ROM (EFR ROM) is a three step process which may be applied to either a constant filter ROM, which we will refer to as the constant radius EFR ROM or, as introduced in this paper, a variable radius filter ROM, which we will refer to as the V-EFR ROM. The process for constant radius filter is as follows: first, the problem is set up as in the G-ROM in Section 2.2. The solution $u^{\text{ROM}}(x,t)$ is evolved to a prospective solution

$w^{\text{ROM}}(x,\tau) = \sum_{i=1}^{R} w_i^r(\tau)\varphi_i^r(x)$ for some time τ by the numerical method of choice. The filtered quantity of interest is solved for in the ROM dimension as the solution to Equation (2.8) but for constant δ, not discretely variable δ_i. This discretizes into the following system:

$$(\mathbf{M_r} + \delta^2 \mathbf{S_r})\overline{w}_r = \mathbf{M_r} w_r. \tag{2.22}$$

This filtered solution can then be relaxed onto the original solution, such that,

$$u^{\text{ROM}}(x,\tau) = (1-\chi)\overline{w}^{\text{ROM}}(x,\tau) + \chi w^{\text{ROM}}(x,\tau). \tag{2.23}$$

For more details on the constant radius EFR ROM, we direct the reader to [48].

For the V-EFR, evolve as normal, but determine the filtered $\overline{w}^{\text{ROM}}(x,\tau)$ from the pre-calculated filtering at the FEM level controlled by δ_i if Algorithm 1 was used. We then reconstruct the solutions onto the FEM space, because the spatial component of the differential filter is more meaningful on the higher dimension space, and relax our solution as in the constant radius EFR. Finally, we project our reconstructed solution back onto the ROM space in order to advance in time on the low-dimensional space.

This process repeats for all future evolutions of the model in time. The filtering reduces oscillations in our solution, while the relaxation prevents excessively diffusing our solution. In practice, the parameters $\delta_i, i = 1, \ldots, N$ and χ are highly dependent upon each other, such that similar results can be generated from greatly different choices of δ_i and χ.

Algorithm 2 Evolve-Filter-Relax With Variable Filter

$i \leftarrow 0$

while $i \leq NT$ **do** ▷ NT is number of time steps

Evolve u_t^r to w_{t+1}^r ▷ Method does not depend on time integrator

$w_{t+1}^h = \sum_{i=1}^R w_{i,t+1}^r \varphi_i^r$ ▷ Reconstruction on finite element space

$\overline{w}_{t+1}^h = \sum_{i=1}^R w_{i,t+1}^r \overline{\varphi}_i^r$ ▷ Filtered Reconstruction, filtered basis computed in Alg. 1

$u_{t+1}^h = (1-\chi)\overline{w}_{t+1}^h + \chi \cdot w_{t+1}^h$ ▷ Relaxation

$u_{t+1}^r = \mathbf{proj_{ROM}} u_{t+1}^h$ ▷ Projection of reconstructed sol. onto ROM space

$i \leftarrow i+1$

end while

2.5 Approximate Deconvolution Leray ROM

In this section, we outline the construction of the approximate deconvolution Leray ROM (ADL-ROM), which was recently introduced in [40]. The ADL-ROM applied to Burgers equation finds a weak solution for the following problem in the ROM space:

$$u_t - \nu u_{xx} + AD(u) \cdot u_x = 0,$$

where $AD(\cdot)$ denotes applying the approximate deconvolution operator as described in Equation (2.13) to the function u. Because the filter and approximate decovolution are solely spatial, this ROM can be entirely pre-assembled as in the discussion on pre-assembly of filtered operators in Section 2.3.2. Instead of using the filtered basis $\overline{\varphi_i^r}$, we require the AD basis as computed in Equation (2.15) in order to assemble the filtered operator. Then, at

every time step, we advance the system,

$$\mathbf{M_r}\dot{\mathbf{a}}(t) + \nu\mathbf{S_r}\mathbf{a}(t) + \mathbf{a}(\mathrm{t})^{\mathrm{T}}\mathbf{B_r^{AD}}\mathbf{a}(t) = \mathbf{0}.$$

Algorithm 3 Approximate Deconvolution Leray Algorithm (ADL-ROM)

Define $\mathbf{C} \in \mathbb{R}^{N\times R}$ to be the matrix whose i^{th} column $\mathbf{c}_i$ is the $N \times 1$ vector containing the coefficients c_{ij} such that $\varphi_i^r(x) = \sum_{j=1}^n c_{ij}\phi_j^h(x)$. $\mathbf{C}$ is obtained through POD (Section 2.2.2). $\langle\cdot,\cdot\rangle$ is the inner product defining the ROM projection.

$i \leftarrow 1$

while $i \leq R$ **do** ▷ R is ROM dimension

Solve $(\mathbf{M}_h + \delta^2\mathbf{S}_h)\overline{\mathbf{c}}_i = \mathbf{M}_h\mathbf{c}_i$ ▷ Filtering pass

Solve $((1+\mu)\mathbf{M}_h + \mu\cdot\delta^2\mathbf{S}_h)\,\mathbf{c}_i^{AD} = (\mathbf{M}_h + \delta^2\mathbf{S}_h)\,\overline{\mathbf{c}}_i$ ▷ AD step

$i \leftarrow i + 1$

end while

Define $AD(\varphi_i^r(x)) = \sum_{j=1}^n c_{ij}^{AD}\phi_j^h(x)$, $i = 1,\ldots,R$

Compute $\mathbf{B_r^{AD}} = \langle AD(\varphi_i^r(x))\cdot(\varphi_j^r(x))', \varphi_k^r(x))\rangle \forall i,j,k \leq R$ ▷ Pre-Assembly Sec. 2.3.2

Evolve $\mathbf{M_r}\dot{\mathbf{a}}(t) + \nu\mathbf{S_r}\mathbf{a}(t) + \mathbf{a}(\mathrm{t})^{\mathrm{T}}\mathbf{B_r^{AD}}\mathbf{a}(t) = \mathbf{0}$.

2.6 Variable Delta Leray ROM

The variable delta Leray ROM (VL-ROM) is very similar to the ADL-ROM case. Instead of computing the convective operator with the AD basis, we compute the operator with a variably filtered basis, which can be generated by Algorithm 1. Then, since the variable delta filter is still a spatial filter, we again use the techniques described in Section 2.3.2 to generate the variable delta filtered convective operator and advance the resulting system in

time.

Algorithm 4 Variable Delta Delta Leray (VL-ROM)

Define $\mathbf{C} \in \mathbb{R}^{N \times R}$ to be the matrix whose i^{th} column $\mathbf{c}_i$ is the $N \times 1$ vector containing the coefficients c_{ij} such that $\varphi_i^r(x) = \sum_{j=1}^{n} c_{ij}\phi_j^h(x)$. $\mathbf{C}$ is obtained through POD (Section 2.2.2). $\langle \cdot, \cdot \rangle$ is the inner product defining the ROM projection.

Send $\mathbf{C}$ to Algorithm 1 to obtain a $\overline{\mathbf{C}}$ variably filtered ROM basis $\overline{\varphi_i^r}, i = 1, \dots R$.

Compute $\mathbf{B_r^{V\delta}} = \langle \overline{\varphi^r}_i(x)) \cdot (\varphi_j^r(x))', \varphi_k^r(x)) \rangle \forall i, j, k \leq R$ ▷ Pre-Assembly Sec. 2.3.2

Evolve $\mathbf{M_r}\dot{\mathbf{a}}(t) + \nu \mathbf{S_r}\mathbf{a}(t) + \mathbf{a}(\mathrm{t})^{\mathrm{T}}\mathbf{B_r^{V\delta}}\mathbf{a}(t) = \mathbf{0}$.

Chapter 3

Results

In this section, we present some results of applying the differential filter in a ROM setting. We have several ROMs to work with, so we will present them in sequence, beginning with the variable delta evolve-filter-relax ROM (V-EFR-ROM, presented in Section 2.4 with algorithms 1 and 2), then the approximate deconvolution ROM (ADL-ROM, presented in Section 2.5 with Algorithm 3), then the variable delta Leray ROM (VL-ROM, presented in Section 2.6 with Algorithm 4). Each section will focus on the performance of the specific ROM with respect to the G-ROM or immediately relevant comparisons. We will conclude with a section on what comparisons can be made numerically between these three alternative ROMs.

3.1 Variable Delta Evolve-Filter-Relax

3.1.1 FEM Computational Setting

The initial solution to the equation is solved at a finite element level using 100 elements, linear basis functions, and 75 time steps. This is a case of under-resolved FEM data. In

particular, the equation and boundary conditions that we solve are,

$$u_t - \nu u_{xx} + u \cdot u_x = 0, \quad (x,t) \in [0,1] \times (0,1],$$
$$u(0,t) = u(1,t) = 0 \quad t \in [0,1]$$
$$u(x,0) = \begin{cases} 0, & x < 0.5 \\ 1, & 0.5 \leq x < 1. \end{cases}$$

For the purposes of a continuous Galerkin method, we assume the initial condition is continuous, but that it is too steep to be captured outside of its extreme values by any mesh. The full solution is plotted in Figure 3.1, and the black lines near $x = 1$ indicate the cascading oscillations of the solution near the boundary layer due to an insufficient number of elements, i.e., an under-resolved FEM simulation.

3.1.2 ROM Computational Setting

All versions of the ROM were also solved with a full spatial resolution of 100 mesh points, 75 time steps, and $\nu = 0.001$. In principle, neither the number of time steps nor ν need to remain the same between the FEM and ROM solutions, though the full spatial resolution is fixed by the FEM. However, changing ν in particular leads to significant questions of fair comparisons, so we have chosen to leave it the same. For the EFR algorithm, $\chi = 0.4$ for all simulations, while values of δ will be labeled when they vary. Furthermore, all ROM solutions we show have been computed for dimension $R = 5$, that is five global ROM basis functions, down from 101 theoretical FEM basis functions. Note that only 99 FEM basis functions are used in practice due to the homogeneous Dirichlet boundary conditions.

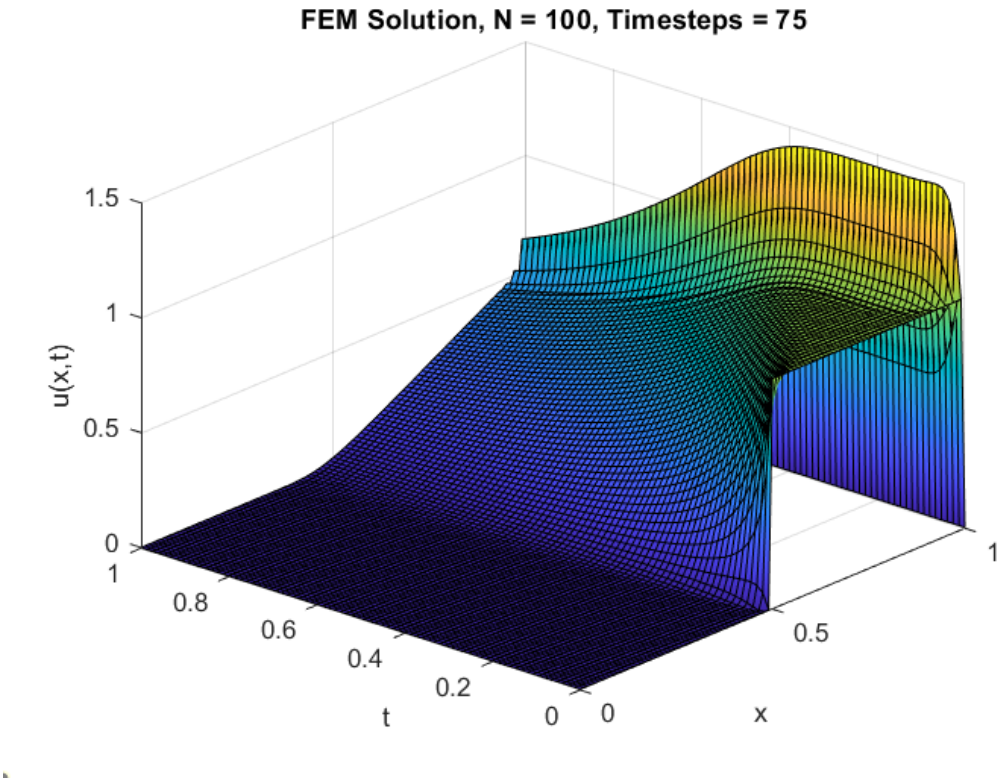


Figure 3.1: Under-resolved FE solution with 100 elements, 75 time steps

In Section 2.4, we discussed the computation of the filtered ROM basis functions for variable δ through the filtered FEM basis functions. For this example, we use a priori information about the boundary layer to tune δ to be large near the boundary and small away from it. In particular, we defined $\delta(x)$ by centering the mean of a normal distribution at $x = 99/100$ with standard deviation $\sigma = 0.05$, and scaled the values of the distribution such that the maximum value of $\delta = \delta_M$. Further figures, for the V-EFR have been generated with $\delta_M = 0.01$, following Algorithm 1. A value of $\delta_M = 0.01$ was also used to generate Figure 3.2, which demonstrates the effect of this variable filtering on the first ROM basis function.

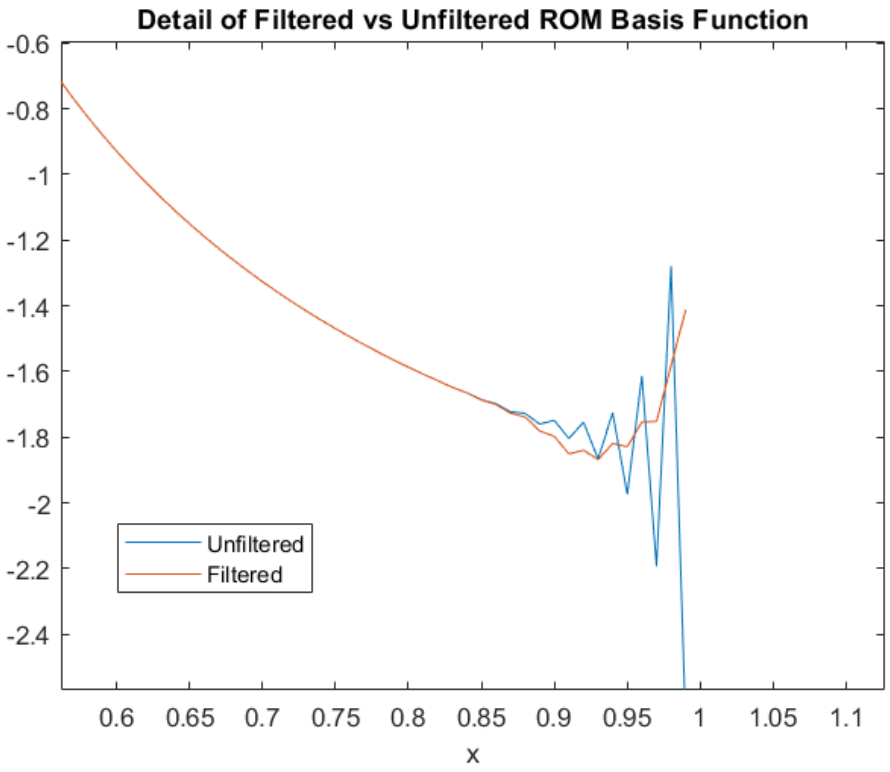


Figure 3.2: Variable delta filter diffuses ROM basis functions only near $x = 1$.

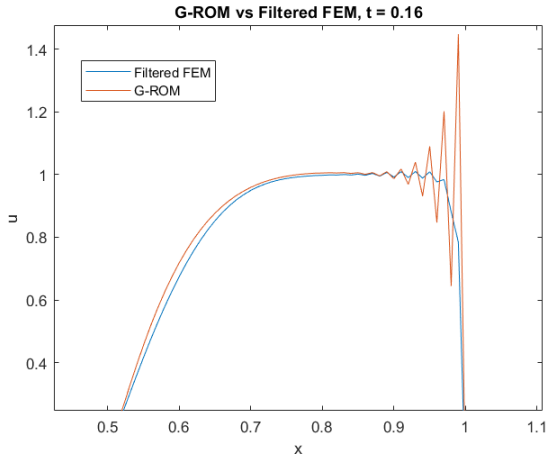
3.1.3 Comparing Filter Performance

In this section, we will compare the standard G-ROM with the constant radius EFR and the V-EFR. For details on generating the constant radius EFR using filtering at a ROM level, we refer you to [47, 48]. The question of fair comparisons is always relevant in numerical experiments, and we have decided to compare all of our ROMs against the filtered FEM, that is,

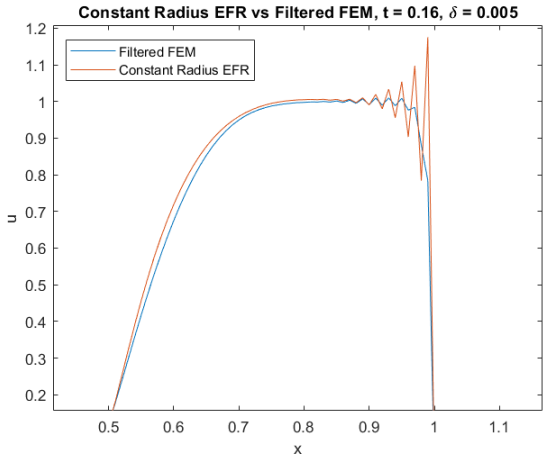
$$\overline{u}_h(x,t) = \sum_{i=1}^{N} c_i^h(t)\overline{\phi}_i^h(x), \tag{3.1}$$

which is the FEM solution constructed from the original FEM coefficients but the filtered FEM basis functions discussed in Section 2.4. Note that all of the ROMs under consideration

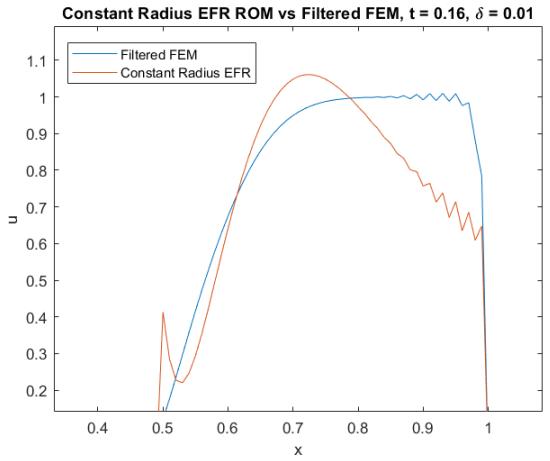
are generated via the POD process on the original, unfiltered FEM solution, and in all cases, the relaxation parameter $\chi = 0.4$.



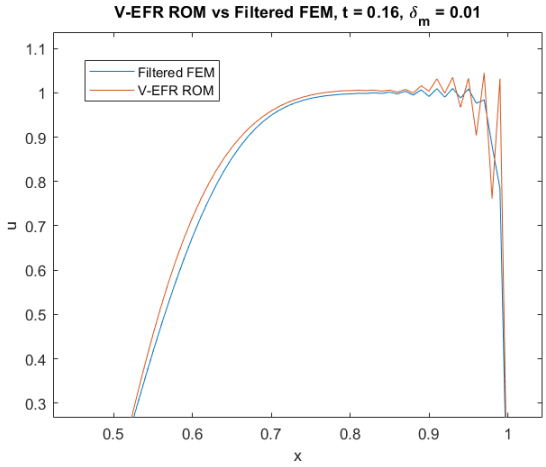
(a) G-ROM



(b) Constant radius filter, $\delta = 0.005$



(c) Constant radius filter, $\delta = 0.01$



(d) V-EFR with max filtering $\delta_M = 0.01$

Figure 3.3: Comparing ROMs with filtered FEM at $t = 0.16$.

For this problem, we demonstrate the 'best case' scenario for the constant filter in the comparison between Figure 3.3(b) and (d). In Figure 3.3(b), the constant $\delta = 0.005$ is half of the maximum filtering of the V-EFR in Figure 3.3(d). Even in this best case scenario for the constant radius filter in 3.3(b), the solution still oscillates much more significantly away from the filtered FEM solution than does the V-EFR solution. In Figure 3.3(c), twice

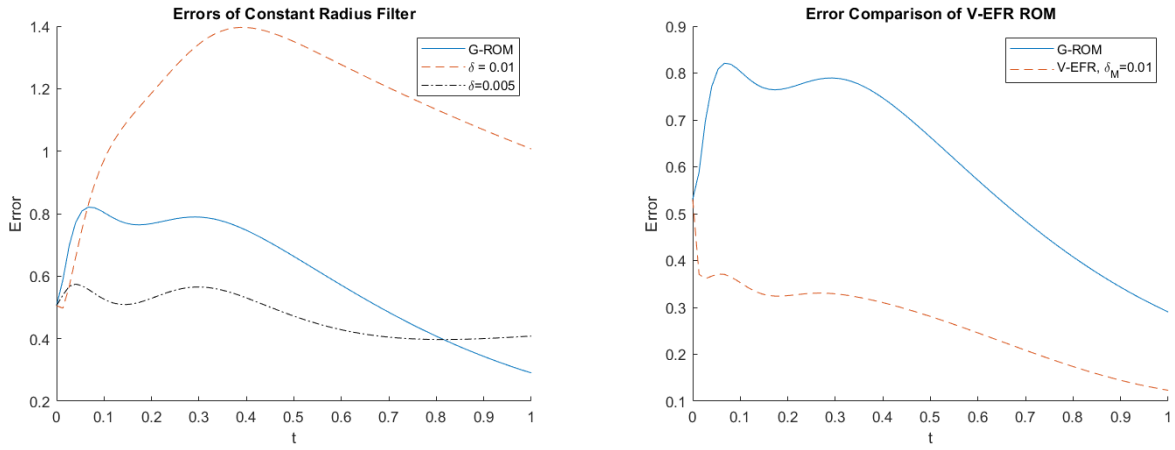
as much diffusion in the constant radius EFR ROM as opposed to Figure 3.3 does quell the oscillations, but it completely destroys the solution. Only the variable radius filter, Figure 3.3(d), is able to bring the oscillations down to a level where they approximate the filtered FEM solution.

3.1.4 Error

We will end our dedicated discussion on the V-EFR-ROM with a short comparison of errors. We will use the Euclidean norm to define our error as,

$$||\vec{u} - \vec{v}||_2 = \sqrt{\sum_{i=1}^{N}(u_i - v_i)^2}. \tag{3.2}$$

Below, all figures will present error of the form $||\overline{u^h} - u^{\text{ROM}}||_2$, that is the filtered FEM solution compared with either the unfiltered G-ROM, constant filter EFR, or V-EFR as indicated. The vector inputs to these norms are the coefficients of the indicated solution at the discretized spatial mesh points.



(a) Constant Radius Error (b) V-EFR Error

Figure 3.4: Errors compared against Filtered FEM solution.

The error at $t = 0$ is non-zero because the filtered FEM initial condition is not the same as the unfiltered initial condition used to construct the ROMs.

In Figure 3.4, we plot the errors of the constant and variable radius EFR against the G-ROM solution.

In this Euclidean definition of error, the V-EFR ROM is the only method that is consistently better than the G-ROM at every time step, and it is also yields less error than even the best constant radius ROM at all time steps. In our experience, if the the constant radius ROM is computed with a $\delta < 0.005$, it stops having much impact on the solution at all, and converges to the G-ROM as $\delta \to 0$. On the other hand, the constant radius filter cannot be as aggressive as the V-EFR, as we can see by the enormous error for $\delta = 0.01$ in Figure 3.4(a), corresponding to excessive diffusion of the solution. For this problem setting, with the very sharp boundary layer, the constant spatial radius ROM is not able to handle the contradictory behavior wherein the G-ROM solution is working well for 90% or more of the spatial domain, and extremely poorly for the last 10%.

3.1.5 Updating the Numerical Setting

The numerical setting as presented in Figure 3.1 displays the most extreme case of a boundary layer, in which the computed FE solution behaves perfectly well at all timesteps except for the very edge of our domain near $x = 1$. While this is the best case we were able to find for the variable EFR model, there are some issues with this design. First, the case of the immediate, static boundary layer strongly limits both the convective and unsteady portions of Burgers equation.

More importantly, this immediate boundary layer eliminates the under-resolved nature of

the ROM. Referring again to Figure 3.1, note that the solution has a very similar profile throughout time. That is, very little spatial movement occurs across the time domain. This means that a small number of ROM basis functions are needed to capture the behavior of the FE solution. The problem with our ROM is not that the ROM is under-resolved, but that the FE solution is under-resolved, and that under-resolved FE solution propagates into the ROM solution. Recall also that we compared performance of these filtered ROMs in Section 3.1.3 against the filtered FE solution, which we were able to do because the essential problem was in the FE solution. The main result here should be interpreted as evidence that Galerkin ROMs can, under certain conditions, be useful even when the input FE data is known to be poor, which is often the case in practical computations (e.g., turbulent flow simulations).

However, the main goal for these models is in application to the under-resolved ROM. For this reason, future sections will test models in a setting where both convection and time dependency are in full effect. We will also see the V-EFR model in this setting in Section 3.4. We will also take the opportunity to update our computation of error into a slightly more standard form, which will be discussed at the beginning of Section 3.2 immediately following this comment.

3.2 Approximate Deconvolution Leray ROM

Here, we present results for the Approximate Deconvolution Leray ROM (ADL-ROM) described in Section 2.5. As a reminder, the PDE we seek to solve is of the form,

$$u_t - \nu u_{xx} + AD(u) \cdot u_x = 0,$$

with homogeneous boundary conditions and a variety of initial conditions. We compute the approximate deconvolution filtered operator and integrate in time as in a standard G-ROM. We will present three cases for this model, starting with a completely resolved input of FE data.

Comment on Error

Previously, Section 3.1 used a pointwise vector formulation of the error. Moving forward, error refers to the L^2 error across the spatial domain, that is,

$$L2(\tau) = \sqrt{\int_0^1 e(x,\tau)^2 dx} \tag{3.3}$$

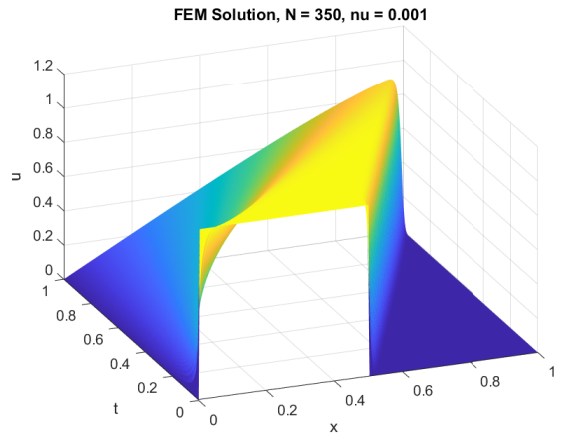
where $e(x,\tau) = u_h(x,\tau) - u_r(x,\tau)$, the difference between the FE solution and the ROM solution at some particular time τ. Calculations were carried out using a 2 point Gaussian quadrature across each finite element node, as we are using piecewise linear polynomials as our FE basis.

3.2.1 ADL For Well Resolved FEM Data

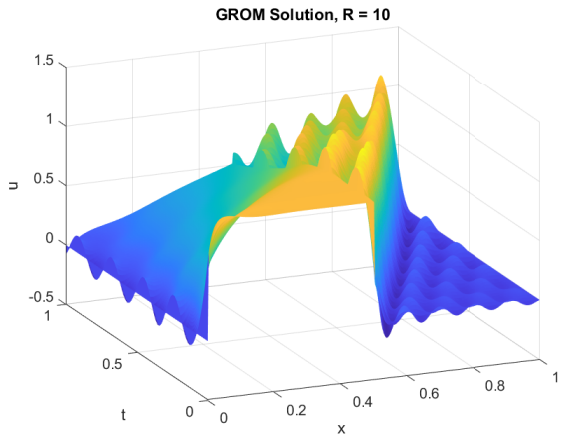
Using 350 elements, $\nu = 10^{-3}$, and an initial condition given as,

$$u_0(x,0) = \begin{cases} 1, & 0 < x < 0.5, \\ 0, & x = 0, 0.5 \leq x \leq 1, \end{cases}$$

we are able to completely resolve the FEM data. However, this is a thesis on under-resolved ROMs for a challenging problem. Even if the input data is excellent, if we limit the number of ROM basis functions then we will still have poor results using a standard ROM:

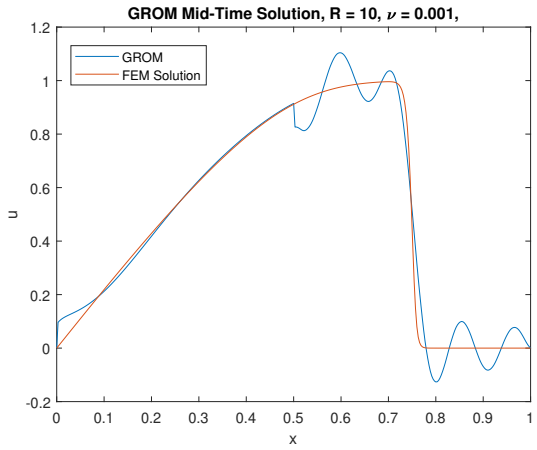


(a) FEM Solution

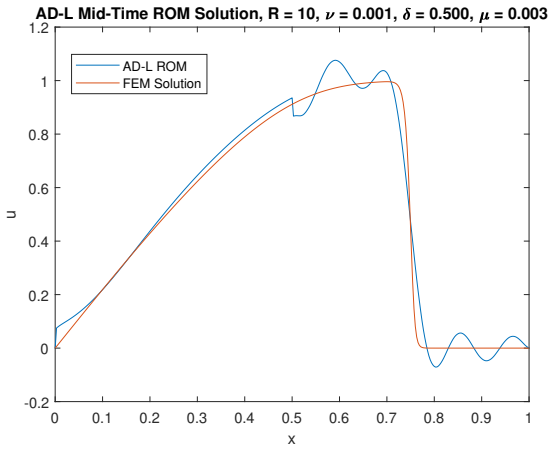


(b) G-ROM, 10 basis functions

Recalling Figure 2.1, which was pulled from this simulation, the reason is clear. The basis functions themselves are oscillatory after the shock at $x = 0.5$ when using this initial condition, and these oscillations propagate through the ROM. We have investigated a variety of AD parameters μ and δ, and present optimized ones below.



(a) G-ROM

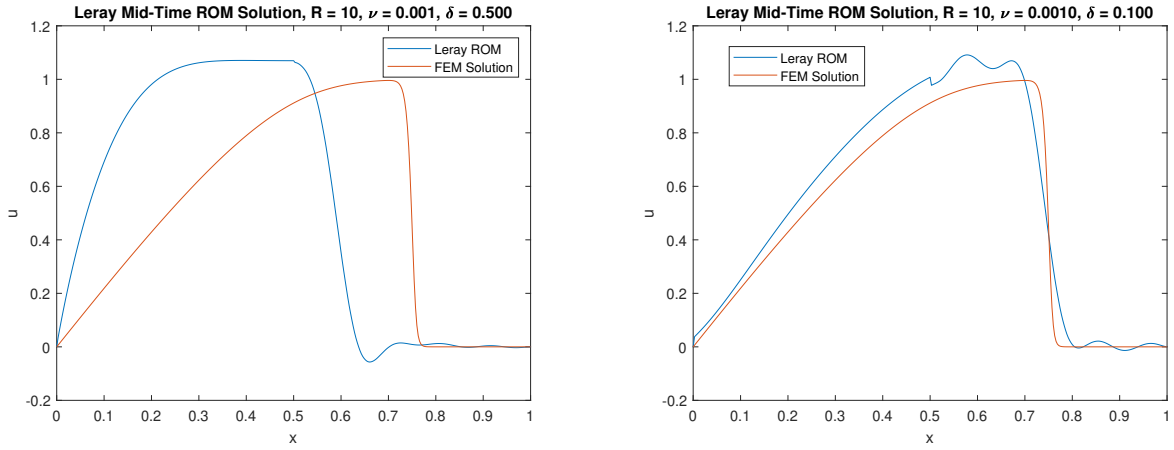


(b) ADL Solution

Figure 3.6: ADL/G-ROM solutions at half-way time.

The above parameters ($\delta = 0.5, \mu = 0.003$) were found using MATLAB's fmincon function carried out over both δ and μ simultaneously to find minimized errors as compared against

the input FEM data, averaged over all time steps. This is not a claim that these values are the ideal parameters for every setting, but they do demonstrate a key difference with the EFR case, that the ADL model is very robust to large values of δ. Note that fmincon likes to minimize one parameter and maximize the other when there is any lack of identifiability between two parameters in constrained optimization.



(a) Leray with $\delta = 0.5$ (b) Leray with $\delta = 0.1$

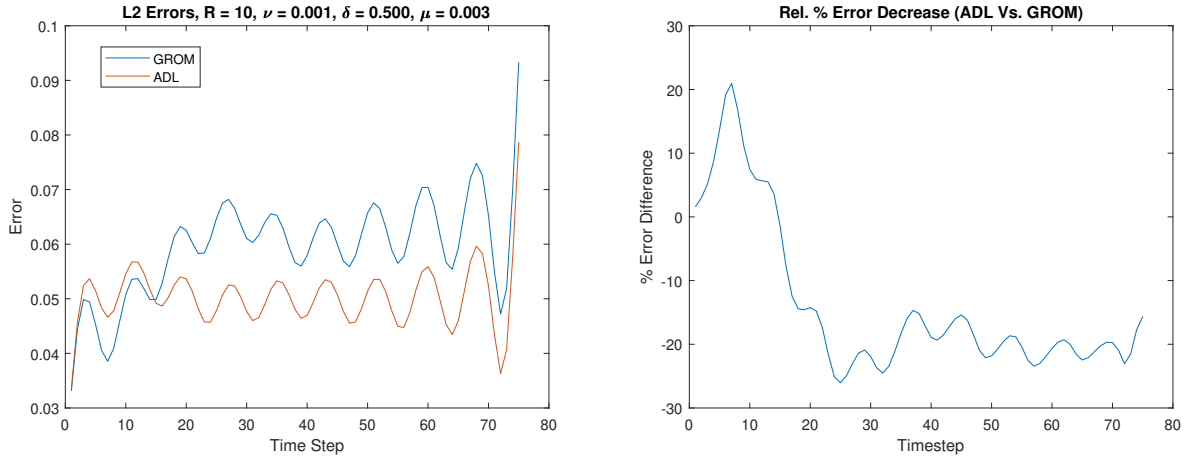
Figure 3.7: Leray solutions at half-way time.

When we compare against the Leray model, the same value of δ as used in the ADL causes an extremely poor solution in Figure 3.7(a). However, the Leray model is naturally less robust to changes in δ than the ADL model. For that reason, we compare against a more reasonable Leray delta value in Figure 3.7(b). Even in this case, excessive diffusion is still added into the solution for this choice of δ. For this reason, even though the oscillations are more smoothed in the Leray model, the model in Figure 3.6 will perform better in overall L^2 error.

Note that no diffusion is added into the solution using the Leray or ADL model, in contrast to what happened with the EFR as shown in Figure 3.3. In this case, the convective movement of the solution across the domain is severely delayed, which is the primary source of error in

both models.

Errors for Well Resolved FE data



(a) L^2 Errors Per Time Step (b) Relative Error Change

Figure 3.8: ADL-ROM errors for well resolved FE data

Comparing the ADL-ROM and G-ROM performance in Figure 3.8, the ADL-ROM stabilizes at around 20% reduction in relative error. This is a reasonable reduction for a process which took exactly 10 extra N dimensional linear solves on the offline stage only, and is identical in cost to the G-ROM in the online stage.

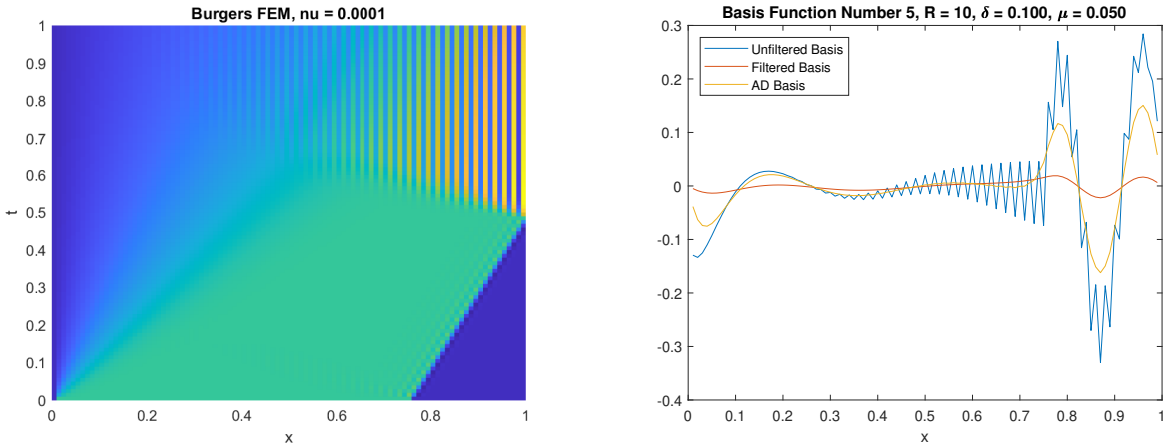
3.2.2 ADL for Poorly Resolved FEM Data

Since ADL is designed to produce a smoother basis than was present in the original G-ROM, it seems reasonable to consider whether we could see much improvement over the modest gains of Section 3.2.1 by making the original G-ROM produce a worse basis. To accomplish this, we lowered ν by a factor of to 10^{-4}, lowered the number of elements in the original FE

solve to 150, and shifted the initial condition such that,

$$u_0(x,0) = \begin{cases} 1, & 0 < x < 0.75, \\ 0, & x = 0, 0.75 \leq x \leq 1. \end{cases}$$

These effects combine to create a strong boundary layer at $x = 1$ in the FE solution after $t = 0.5$. The shock of the poorly resolved boundary condition propagates backwards in space after the right edge of the wave profile reaches the poorly resolved boundary.



(a) FEM Solution

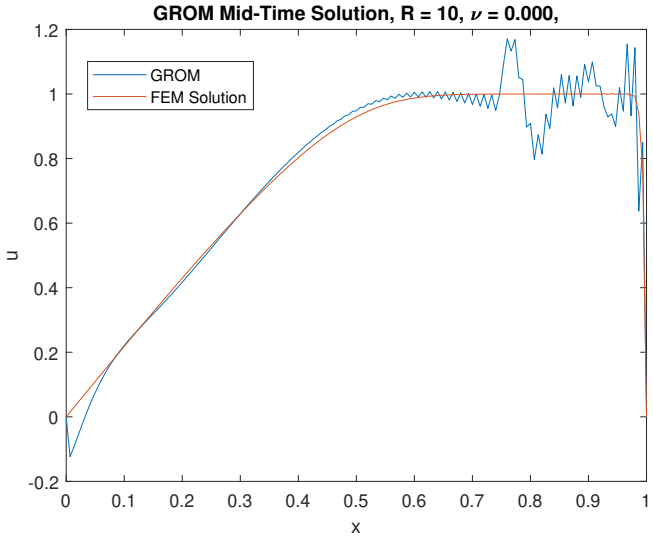
(b) G-ROM, 10 Basis Functions

Figure 3.9: Poorly resolved FE solution: $\nu = 10^{-4}$, 150 elements and filtering basis.

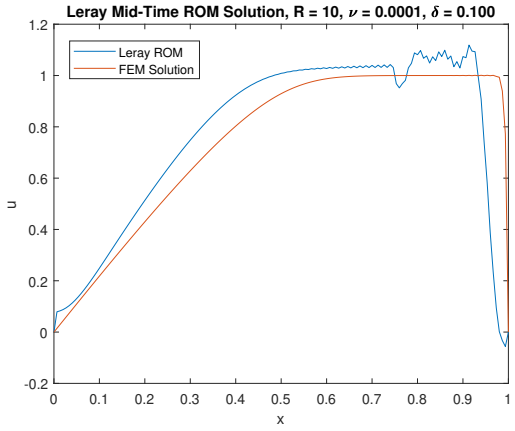
Additionally, the number of time steps between $t = 0$ and $t = 1$ was increased to 100 to prevent serious stability concerns in the Newton solver.

When given this noisy data, the POD algorithm at the heart of the G-ROM produces an extremely poor basis as expected. In Figure 3.9b(a), we can see the oscillations in the FE data increasing as time progresses past $t = 0.5$ as indicated by the pattern of lines. In particular, the G-ROM basis produced from this partially noisy data as shown in Figure 3.9b(b) demonstrates multiple scales of oscillations. A Leray filter chosen too large will wipe

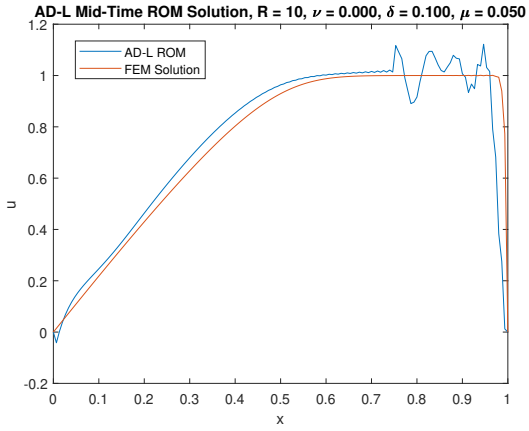
out most of the necessary information contained in the G-ROM basis, while AD is able to easily recapture the important mean behavior in the G-ROM basis without the secondary oscillations.



(a) G-ROM



(b) Lery, $\delta = 0.1$



(c) ADL, $\delta = 0.1$, $\mu = 0.05$

Figure 3.10: Three ROM mid-time solutions, $\nu = 10^{-4}$

Similar things are happening here as occurred in the case with well-resolved FE data. The Leray solution has smoothed much of the oscillations at the cost of delaying the convective motion. This is especially impactful in an L^2 error sense in the places where the G-ROM

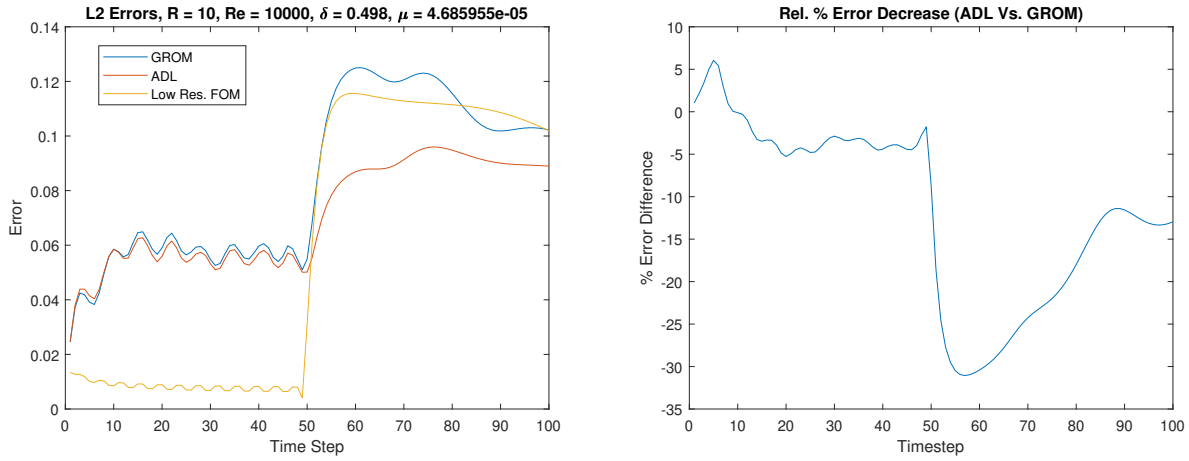
was already doing well, before $x = 0.7$. In contrast, the ADL solution is able to mimic the performance of the G-ROM where it was performing well, while still smoothing the oscillations.

The ADL also displays a capacity for improved multiscale filtering. The secondary smoothing parameter $\mu = 0.05$ in the ADL solution does a better job of filtering out the very small scale oscillations seen in the Leray solution in Figure 3.6(b).

Errors in Poorly Resolved Case

For this problem, it does not make much sense to compare our final solutions against the FEM input data, because one of the major problems we are trying to address is the corruption of the basis by the under-resolved nature of the FE data. A large number of FE nodes is needed in order to compute a well-resolved solution, but the 101 nodes that define the low-resolution FE solution is perfectly adequate for actually describing a well-resolved solution, if it has already been computed. For this reason, we use the same format of L^2 errors as in Equation (3.3), but compared against a much higher resolution FE solution. The high resolution FE solution in question had 900 elements and took significantly longer to compute than any other FE or ROM solution used in this thesis.

This jump in computation time speaks to a very important topic in ROM, but one which is hard to make judgment on in a one-dimensional spatial setting, which is the fact that actually computing a completely resolved solution either at a FE or ROM level may be prohibitively expensive. If ROM performance can be trusted when run on poorly or even moderately well resolved FEM data, a significant amount of time and energy (in a very literal sense) has the potential to be saved on very large scale projects.



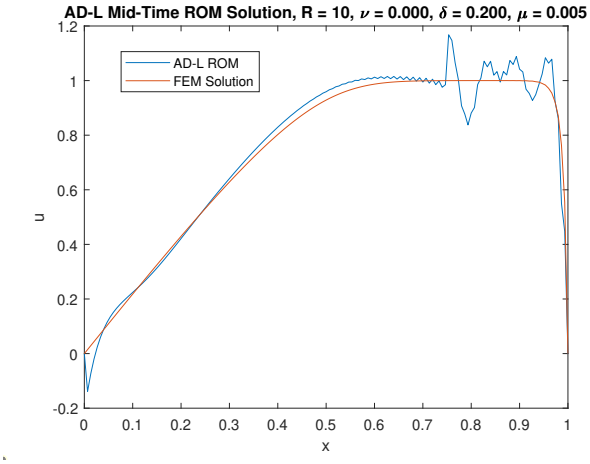
(a) L^2 Errors Per Time Step (b) Relative Error Change

Figure 3.11: Errors for poorly resolved FE Data

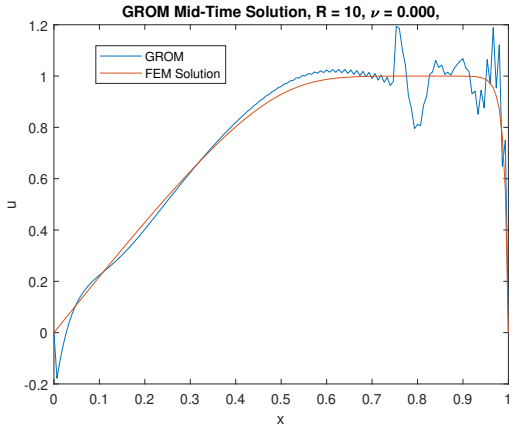
In Figure 3.11, we are again using some optimized parameters that minimize the overall error across all time steps at once. The line "Low Res. FOM" corresponds to calculating the error in the FE input data we feed into the ROM vs the high resolution FE solution we are comparing against. The FE solution performs well for the first 50 time steps, and poorly thereafter. Going back to Figure 3.9b(a), time step 50 (at $t = 0.5$ for 100 time steps across the domain $[0, 1]$) is the point at which the FE solution begins to present a sharp boundary layer caused by the convection of the solution across the spatial domain towards the right endpoint of $x = 1$. The G-ROM performs well in the time steps where the FE solution is performing well, but quite poorly afterwards. In contrast, the ADL-ROM is able to match the G-ROM for the time steps where it is performing well, and improve upon it significantly for time steps the G-ROM is doing poorly. However, the results are no more impressive than in the case of the well-resolved FE data.

3.2.3 ADL for EFR-Stabilized FEM Data

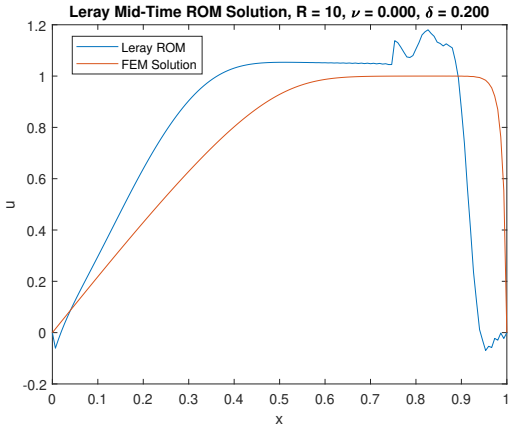
Now, we will consider ADL as applied to an already stabilized FE solution. For this stabilization, we will use the easy to implement stabilization EFR method as described previously at a ROM level in Section 2.4, though it works the same at a FE level. In particular, using a constant radius filter of $\delta = 2 \cdot h$ (where $h = 1/150$) and relaxation parameter $\chi = 0.5$ corresponding to mixing half of the filtered updated solution with half of the unfiltered updated solution, we are able to significantly improve our FE input data.



(a) ADL, $\delta = 0.2, \mu = 0.005$



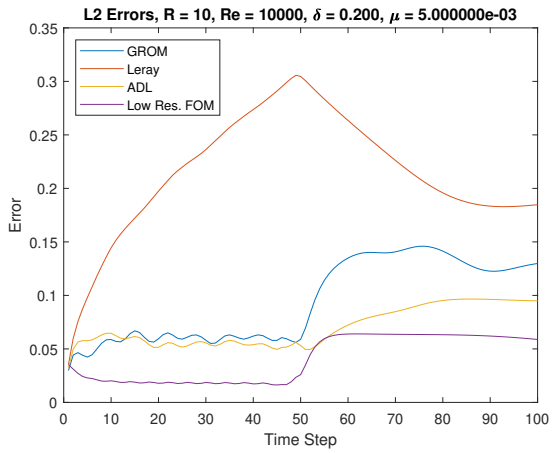
(b) G-ROM



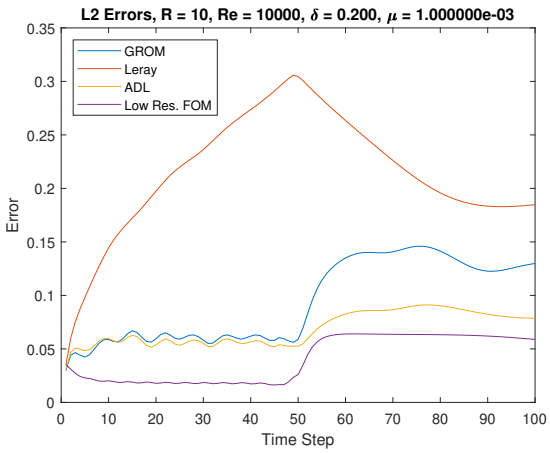
(c) Leray, $\delta = 0.2$

Figure 3.12: 3 ROM mid-time solutions with stabilized FE, $\nu = 10^{-4}$

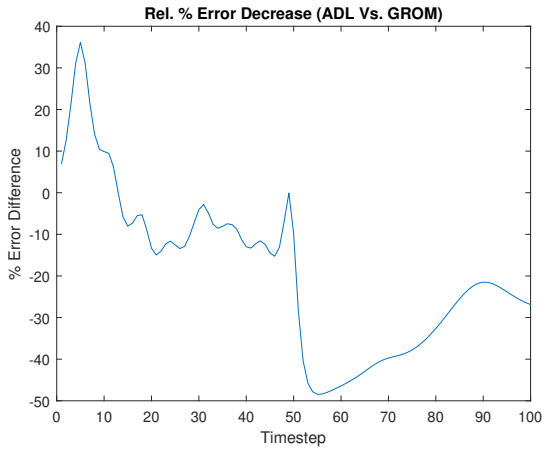
Our error computation will again be against the high resolution FE solution. The EFR FE data was generated using the same conditions as Figure 3.11(a). In that case, the error stabilized at an absolute value near 0.12, while in Figure 3.13(a), we can see that the low resolution, EFR stabilized FE solution stabilizes at an absolute value of 0.06, which is around a 50% decrease in error.



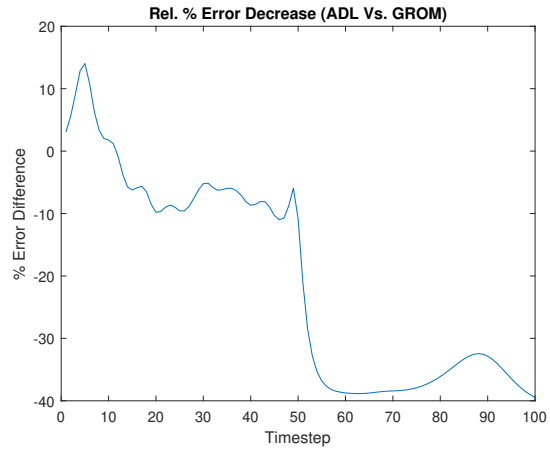
(a) Error with larger μ



(b) Error with smaller μ



(c) Rel. % Decrease with larger μ



(d) Rel. % Decrease with smaller μ

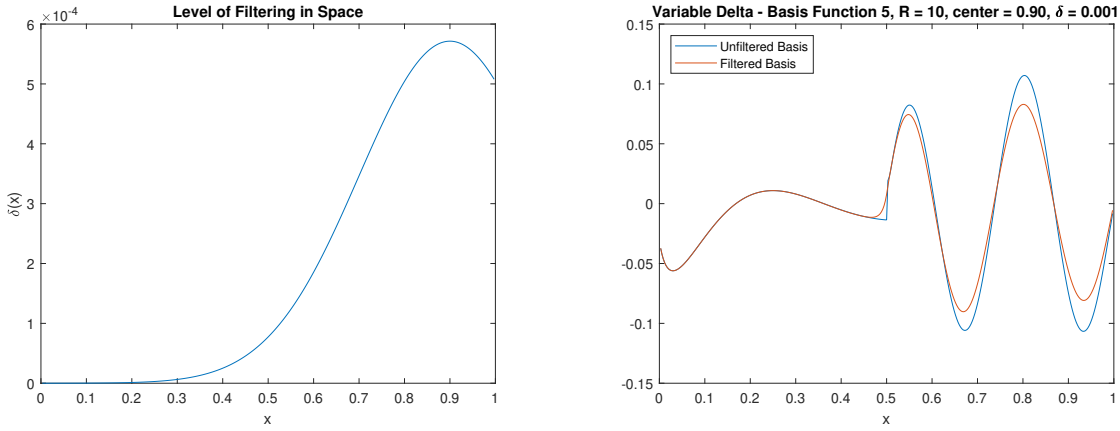
Figure 3.13: Errors for EFR stabilized FE data

While the ADL ROM is no longer out-performing the stabilized FEM solution, it is a significant improvement over both the G-ROM and Leray models for both parameters. In

particular, the smaller μ value (less secondary smoothing) stabilizes at more than 30% relative decrease in error compared to the G-ROM. These are the best results we have seen, and suggest that there may be something of a sweet spot to hit when looking to stabilize with ADL. There is some evidence here that a basis which performs poorly, but not too poorly, is the best fit for this kind of smoothing.

3.3 Variable Delta Leray

Recall that the VL-ROM was assembled by constructing a discretely variable ROM basis as in Algorithm 1. In this section, as with the V-EFR, we define the filtering values δ_i to be drawn from a normal distribution which has been normalized to have a specified maximum frequency. For each FE basis function $\phi_i^h(x)$ (which is 1 at a specific node x_i), we specify that it is filtered by an amount δ_i.



(a) $\delta(x)$ (b) Effect on Basis

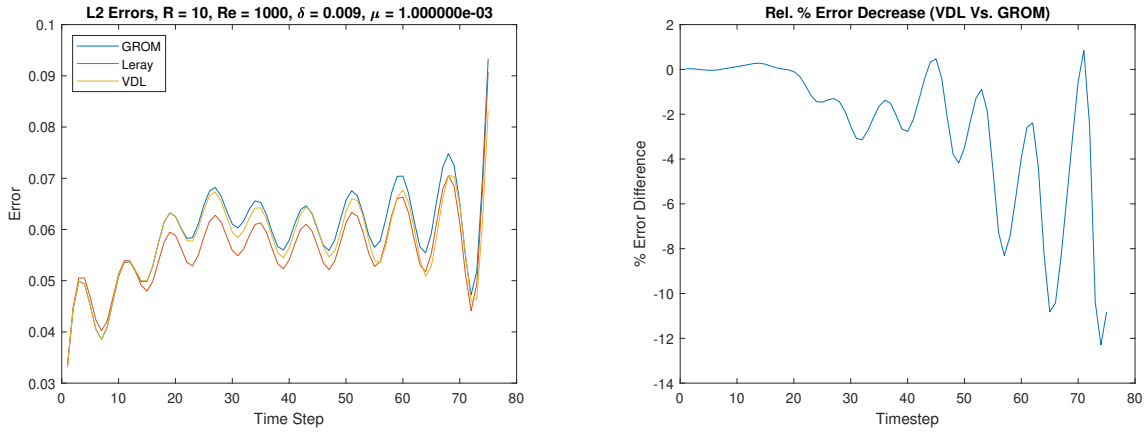
Figure 3.14: Delta function and effect on ROM basis

We can see that the ROM basis function is an exact match for the unfiltered basis function where low amounts of filtering are prescribed. It is important that neighboring values of δ_i

are similar. If they are not, then filtering neighboring FE functions by significantly different amount will introduce discontinuities into the solution.

3.3.1 VL-ROM For Well Resolved FE Data

This section corresponds to the similarly named section in the results for ADL-ROM in Section 3.2 in terms of the numerical setting and initial conditions. For well-resolved FE data, we will be using 350 elements, $\nu = 10^{-3}$, and 75 time steps between 0 and 1. In the well-resolved case, the errors were generated by comparing the reconstructed ROM solution against the original FE input data. No solution is plotted, as the VL-ROM in this case does not differ much from the G-ROM.



(a) Errors (b) Percentage Change

Figure 3.15: VL-ROM errors

Tthe value of $\delta = 0.009$ corresponds to $3 \cdot h$, where h is the constant mesh size. It is applied evenly in the case of the Leray model, and is set as the maximum height of the $\delta(x)$ function in the VL-ROM. The results are inconclusive. It is difficult to optimize the result because of the variety of parameters involved in a VL-ROM. Choosing δ_i depends on choosing a continuous function, which in our case is the result of a 3-parameter distribution, described

by a mean, standard deviation, and maximum value, each of which have an immense effect on the resulting filtered basis independently. Choosing higher levels of δ_i seemed to excessively delay the movement of the solution across the domain; choosing much lower levels of δ_i seemed to have little to no effect on the solution.

The VL-ROM was also looked at for poorly resolved FE data, but similarly uninspiring results were found. We will see the model again for a comparison in the stabilized FE case in Section 3.4. The VL-ROM is challenging to optimize, because it depends on a parameterized distribution in addition to our filtering parameters.

3.4 A Numerical Comparison

This section will be a test of all of the models so far presented on a single test case. The case which seems the most representative of conditions to which we would like to apply these models to is the poorly resolved FE data (with a boundary layer) in which the FE solution has undergone semi-successful stabilization efforts. This is the same test considered in Section 3.2.3, and the input FEM data looks similar to Figure 3.9a.

For parameters, we will be using the "best guess" approach. That is, there should be a reason for choosing parameters as they are, but the goal is to test how these models perform without a posteriori knowledge or optimized parameters.

For the constant radius Leray model, we choose $\delta = 3 \cdot h = 0.02$. For the Variable Delta Leray model, we choose the maximum delta to be the same $3 \cdot h$, but it is only achieved at the mean of a normal distribution centered on $x = 0.9$ with standard deviation $\sigma = 0.2$. Small changes of nearby multiples of h were also tried, but presented no significant changes

to overall error. A filtering radius dependent on h was chosen because the finite element mesh size is the smallest scale we can hope to resolve.

For the Variable Delta EFR (VEFR), we choose the same normal distribution as for the Variable Delta Leray model. This information was presented in Figure 3.14(a). Various small multiples of h were tested but seemed to have no significant effect on the error. In the following image, a maximum δ of $h = 0.0067$ was specified.

For the Approximate Deconvolution Leray (ADL) model, we choose $\delta = 0.2$ and the secondary smooth parameter $\mu = 0.001$. ADL is able to tolerate much larger values of δ, and the oscillations away from the mean velocity profile of $u = 1$ deviate by a maximum distance of 0.2 in the G-ROM, which is why this particular value of δ was chosen.

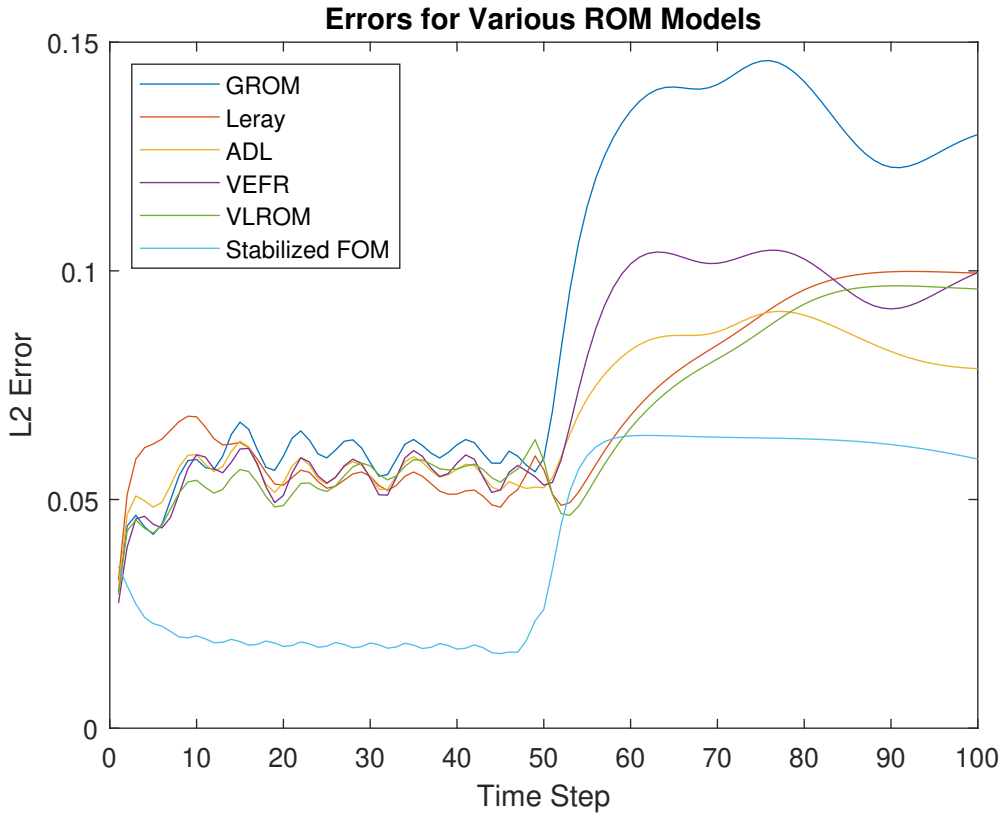


Figure 3.16: Comparison of L^2 errors against high resolution FE solution

Each of the models are at least capable of surpassing G-ROM in the solution region where the boundary layer exists (for time steps past 50). As expected, the constant radius Leray model over-filters at first (time steps 0-15), but the Variable Radius Leray (VLROM) does not seem to offer significant improvements over Leray elsewhere.

The Approximate Deconvolution ROM (ADL) and the Variable Radius EFR ROM (VEFR) seem to stabilize their performance more so than either of the Leray ROMs. The ADL ROM seems to offer more robust performance for non-optimally chosen parameters. There is a certain lack of unique identifiability in the ADL parameters δ and μ, which is sometimes useful when trying to choose parameters without prior knowledge.

Chapter 4

Discussion

4.1 Comments on Delta

The filtering parameter δ is of great importance to all of these models, and each of them use this parameter in a slightly different way. For all of the non-ADL models (EFR, V-EFR, Leray, V-Leray), we have found that a value of $\delta \approx h$, where h is the uniform FE mesh size, is a good starting point. This is in keeping with standard practice for filter based models [6]. A posteriori optimization may prefer more unusual parameter choices depending on the setting, but that information is by its nature not available when making the initial choice.

For ADL, there is considerable flexibility in the choice of δ because of the secondary smoothing parameter μ. In this setting, the choice of δ is largely dependent on the quality of the ROM basis functions. If an arbitrary ROM basis function, $\varphi_i^r(x)$, is highly oscillatory, then a large value of δ may be chosen. In this case, the approximate deconvolution will be applied to an extremely smooth function, so very small amounts of secondary smoothing will be needed, meaning μ may be chosen small.

In general, our results suggest that filtering should be chosen too light rather than too heavy if the decision is made based on error compared to a more accurate solution. For each of the models presented, it would have been simple to choose parameters resulting in a very

smooth solution, but when using regularization it is important to remember that we are treating symptoms, not causes, and we should not exchange one non-physical phenomenon for another by trading excessive diffusion for oscillations.

4.2 Qualitative Comparison of Models

We have presented several models in this thesis. Some have been seen before in a ROM context (Constant EFR, Leray, ADL) and some are new (Variable EFR and Variable Leray). Here, we take stock of what has been presented, and make suggestions of the path to take when deciding when to implement one model over another. The models we presented in order were the EFR-ROM (with non-variable and variable filtering), the ADL-ROM, and the VDL-ROM. This order tracks well the ease of implementation. A constant radius EFR can be implemented in a handful of lines of code and in some cases may provide significant improvements to stabilization, as we saw in this thesis going from Figure 3.11 to Figure 3.13, where the error in the low resolution FE solution dropped by 50%. All of this requires only one additional linear solve at each time integration step with size corresponding to your model (N for FEM, R for ROM).

Transitioning to a variable delta EFR in a ROM context requires the filtering of potentially every single FE basis function. This could be significantly improved by indicator functions [10, 19] that determine where filtering is and is not needed, but this is still potentially up to N linear solves in the offline stage. Even though a ROM is usually optimized for the online stage, the cost of the offline stage cannot be ignored.

One strong advantage of EFR methods is that filtering can be turned on or off at each time step, given an appropriate indicator function. With a constant radius filter, the level

of filtering may even be customized at each time step simply by changing the parameter δ. For a variable radius filter, this is more challenging, however, all that is needed are the coefficients associated with the filtered ROM basis function $\overline{\varphi_i^r(x)} = \sum_{k=1}^{N} \overline{b_{ik}}\phi_i^h(x)$. If only a few of the constituent FE basis functions need to be adjusted, an adaptive V-EFR model could be accomplished with numerical efficiency. This is something that cannot be said for any Leray type method - the requirement of constructing a nonlinear operator makes a time step by time step recalibration of filtering levels impractical.

There are a few drawbacks to EFR: first, every EFR method inherently preserves numerical artifacts in its solution by the nature of the relaxation process. How serious those issues are determines the viability of the EFR method. Secondly, the diffusion added by the EFR method requires a very light touch to get right. Because we are inserting this diffusion directly into the solution, we have to be extremely careful in choosing our filtering levels.

For the ADL method, all of the work is offline. For the pre-assembly method of Section 2.3.2, we compute R linear solves to construct the filtered basis and R linear solves to construct the AD basis. Once that is done and the operators are assembled, the online stage has identical cost and operation compared to the G-ROM. Assembling nonlinear operators can be complicated and numerically expensive, however, so an efficient program to assemble these operators is necessary.

One of the main advantages of Leray type models is that no diffusion is added directly into the solution. All of the filtering is concentrated in the typically problematic convective term, so the movement of the solution across the spatial domain is slowed, but there is no additional diffusion of the solution as occurs in EFR. This means that things like the kinetic energy of the G-ROM are more likely to be preserved in a Leray or ADL type solution as opposed to an EFR solution. If you have a velocity profile of a certain magnitude, it does

not matter where that profile is located to recover the G-ROM's kinetic energy. In fairness to the EFR, it has been shown [48] that the G-ROM tends to hang onto excess energy in Navier-Stokes fluid simulations, so siphoning some of this excess energy is not unreasonable.

The variable delta Leray model is most similar to the ADL model, with the additional requirement of up to N linear solves in the offline stage to construct the variably filtered basis. No additional work is required in the online stage. However, as seen in the results, the behavior of the constructed nonlinear operator is not necessarily obvious. Additionally, as with V-EFR, determining a suitable continuous distribution or function to draw values of δ_i is challenging.

4.2.1 Tabular Comparison

ROM Model:	Constant EFR	Variable EFR	Leray	V-Leray	ADL
Ease of Implementation	✓✓✓	✓	✓✓	✓	✓✓
Spatial Flexibility	✗	✓✓	✗	✓✓	✓
Temporal Flexibility	✓✓	✓	✗✗	✗✗	✗✗
Parameter Flexibility	✗✗	✗✗	✗✗	✗✗	✓
Low Online Cost	✓✓	✓	✓✓✓	✓✓✓	✓✓✓
Low Offline Cost	✓✓✓	✗✗	✓✓	✗✗	✓
Accuracy	✓	✓	✓	✓	✓✓

The meaning of Spatial Flexibility is to what degree our filtering methods can permit different effects on the solution based on what part of the domain we are considering. For constant delta Leray and EFR, the single pass of filtering gives very little flexibility across the domain, whereas the two stage filter for ADL does have somewhat different effects across the domain. That is, if the original G-ROM basis functions are performing well in a certain location, ADL

does not alter them much, but if the basis functions are performing poorly elsewhere in the domain, the same AD parameters will provide significant smoothing. See Figure 3.9b for an example of this behavior.

Temporal Flexibility refers to whether or not filtering can be controlled in the online time integration stage. For the models in which we pre-assemble an operator, time alterations are not feasible. For the Constant EFR model, filtering levels can be prescribed at will in the online stage, though a suitable indicator function is needed. For Variable EFR, more work would be needed, but since no operator is pre-computed such filtering could be feasible. This approach was not considered in this thesis.

Parameter Flexibility refers mainly to the ability to choose δ and μ freely. For the models in which we rely on the single smoothing parameter δ (all models except ADL), all of the tests in the results section required a dependency on h, the FE mesh size, in order to optimize errors. Only with ADL were we able to escape this requirement.

The Accuracy row refers mainly to Section 3.4, the numerical comparison between all examined models. In that comparison with non-optimized parameters, ADL performed the most consistently across all time steps. Additionally, ADL was the most amenable to optimization and demonstrated the best error reduction we were able to present in Section 3.2.3. It must be stressed that this is a test in a particular shock solution setting for a particular problem with one spatial dimension. This may not be the case for all problems, but in this under-resolved ROM setting it proved effective.

Chapter 5

Conclusions and Future Work

This thesis has presented an overview of older, recent, and new models for differential filtering applied to Galerkin ROMs in an under-resolved setting, with attempts to be partly a user guide for which and why a certain ROM might be chosen and how to go about implementing it. However, this has primarily been an investigation into models with limited testing at a ROM level, and as such much work remains to be done.

An example of where these techniques might be used could be in stochastic processes. In particular, because approximate deconvolution is an image processing technique, it is naturally suited to handling noisy data. This could take the form of random input data, e.g., [20, 21, 22, 25, 33], where the concern is that certain inputs may produce poorly behaving solutions, or perhaps in a control style problem driven by sensors subject to random noise.

On an implementation level, incorporation of indicator functions [10, 19] could significantly help the performance of variable delta models. Filtering exactly the problematic basis functions while leaving the rest untouched could lead to significant cost savings, which is one of the main problems with a naive implementation of a variable delta level which filters every FE basis function. For EFR models which filter at each time step, indicator functions could be used to determine whether filtering is needed at all at a given time step, and if so, at what level.

An additional level of spatial flexibility is available in the ADL and constant Leray models in that different ROM basis functions can easily be filtered by differing amounts. This has not been done in this thesis, but the reason for mentioning this is that a Galerkin ROM procedure generates modes which resolve smaller and smaller scale information. This has a tendency to create oscillations in basis functions as the index of the ROM basis function increases. Higher levels of filtering increasing with the index of the ROM basis function may help this, especially if many ROM basis functions are used.

An important part of ROM utilization is scanning through multiple parameter or temporal values without the costs associated with the full order model. That is, the predictive regime, where for example a ROM might be built from data of certain pre-determined parameter values (in the setting considered in this thesis, that would be changing the viscosity parameter ν) but used to scan through many parameter values in order to solve an optimization problem. Alternatively, given data available up to a certain time, we may wish to predict into the future. For the ROMs built in this thesis, we aimed to replicate a FE solution given data using true parameter value ν, but with 50% of the time steps removed. However, every other time step was removed from the data, so none of these ROMs have been tested in a truly predictive regime.

In the big picture, and as mentioned in the introduction, a large part of the appeal of Galerkin ROMs is their physics and PDE derived nature which at least suggests the feasibility of proving things about these models, such as, stability, consistency, and convergence. To our knowledge, these topics have received limited publication as of yet, with some investigation as to FOM-ROM consistency [43] and the stability and *a priori* error bounds for the constant delta Leray ROM [50]. This has not been accomplished in this thesis, and is probably the most important next step. Provably reliable models is something that is challenging to

provide when using AI learning techniques, but is something that Galerkin ROMs have a natural framework for.

Bibliography

[1] M. Ahmed and O. San. Stabilized principal interval decomposition method for model reduction of nonlinear convective systems with moving shocks. *Comp. Appl. Math.*, 37 (5):6870–6902, 2018.

[2] S. E. Ahmed, S. Pawar, O. San, A. Rasheed, T. Iliescu, and B. R. Noack. On closures for reduced order models − a spectrum of first-principle to machine-learned avenues. *Phys. Fluids*, 33(9):091301, 2021.

[3] F. Ballarin, E. Faggiano, S. Ippolito, A. Manzoni, A. Quarteroni, G. Rozza, and R. Scrofani. Fast simulations of patient-specific haemodynamics of coronary artery bypass grafts based on a POD–Galerkin method and a vascular shape parametrization. *J. Comput. Phys.*, 315:609–628, 2016.

[4] A. Barone, M. G. Carlino, A. Gizzi, S. Perotto, and A. Veneziani. Efficient estimation of cardiac conductivities: A proper generalized decomposition approach. *J. Comput. Phys.*, 423:109810, 2020.

[5] M. Bergmann, C. H. Bruneau, and A. Iollo. Enablers for robust POD models. *J. Comput. Phys.*, 228(2):516–538, 2009.

[6] Luigi Berselli, Traian Iliescu, and William Layton. *Mathematics of Large Eddy Simulation of Turbulent Flows*. Springer, 01 2006. ISBN 3-540-26316-0. doi: 10.1007/b137408.

[7] L. Bertagna, A. Quaini, and A. Veneziani. Deconvolution-based nonlinear filtering for incompressible flows at moderately large Reynolds numbers. *Int. J. Num. Meth. Fluids*, 81(8):463–488, 2016.

[8] M Bertero and P Boccacci. *Introduction to inverse problems in imaging*. CRC Press, August 2020. doi: 10.1201/9780367806941.

[9] J.T. Borggaard and T. Iliescu. Approximate deconvolution boundary conditions for LES. *Appl. Math. Letters*, 19(8):735–740, 2006.

[10] Abigail L. Bowers and Leo G. Rebholz. Numerical study of a regularization model for incompressible flow with deconvolution-based adaptive nonlinear filtering. *Computer Methods in Applied Mechanics and Engineering*, 258:1–12, 2013. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2013.02.003. URL https://www.sciencedirect.com/science/article/pii/S0045782513000352.

[11] M. Couplet, P. Sagaut, and C. Basdevant. Intermodal energy transfers in a proper orthogonal decomposition–Galerkin representation of a turbulent separated flow. *J. Fluid Mech.*, 491:275–284, 2003. ISSN 0022-1120.

[12] V. J. Ervin, W. J. Layton, and M. Neda. Numerical analysis of filter-based stabilization for evolution equations. *SIAM J. Numer. Anal.*, 50(5):2307–2335, 2012.

[13] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1625–1634, 2018. doi: 10.1109/CVPR.2018.00175.

[14] I. Farcas, R. Munipalli, and K. E. Willcox. On filtering in non-intrusive data-driven reduced-order modeling. In *AIAA AVIATION 2022 Forum*, page 3487, 2022.

[15] L. Fick, Y. Maday, A. T. Patera, and T. Taddei. A stabilized POD model for turbulent flows over a range of Reynolds numbers: Optimal parameter sampling and constrained projection. *J. Comp. Phys.*, 371:214–243, 2018.

[16] Samuel G. Finlayson, John D. Bowers, Joichi Ito, Jonathan L. Zittrain, Andrew L. Beam, and Isaac S. Kohane. Adversarial attacks on medical machine learning. *Science*, 363(6433):1287–1289, 2019. doi: 10.1126/science.aaw4399. URL https://www.science.org/doi/abs/10.1126/science.aaw4399.

[17] K. J. Galvin, L. G. Rebholz, and C. Trenchea. Efficient, unconditionally stable, and optimally accurate FE algorithms for approximate deconvolution models. *SIAM J. Numer. Anal.*, 52(2):678–707, 2014.

[18] Michele Girfoglio, Annalisa Quaini, and Gianluigi Rozza. A finite volume approximation of the navier-stokes equations with nonlinear filtering stabilization. *Computers & Fluids*, 187:27–45, 2019. doi: 10.1016/j.compfluid.2019.05.001.

[19] Michele Girfoglio, Annalisa Quaini, and Gianluigi Rozza. Fluid-structure interaction simulations with a les filtering approach in solids4foam. *Communications in Applied and Industrial Mathematics*, 12:13–28, 08 2021. doi: 10.2478/caim-2021-0002.

[20] M. Gunzburger, T. Iliescu, M. Mohebujjaman, and M. Schneier. An evolve-filter-relax stabilized reduced order stochastic collocation method for the time-dependent Navier-Stokes equations. *SIAM-ASA J. Uncertain.*, 7(4):1162–1184, 2019.

[21] M. Gunzburger, T. Iliescu, and M. Schneier. A Leray regularized ensemble-proper orthogonal decomposition method for parameterized convection-dominated flows. *IMA J. Numer. Anal.*, 40(2):886–913, 2020.

[22] M. D. Gunzburger, C. G. Webster, and G. Zhang. Stochastic finite element methods for partial differential equations with random input data. *Acta Numerica*, 23:521–650, 2014.

[23] Sofia Guzzetti, Simona Perotto, and Alessandro Veneziani. Hierarchical model reduction for incompressible fluids in pipes. *International Journal for Numerical Methods in Engineering*, 114(5):469–500, 2018. doi: https://doi.org/10.1002/nme.5726. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.5726.

[24] M. A. Habisreutinger, R. Bouffanais, E. Leriche, and M. O. Deville. A coupled approximate deconvolution and dynamic mixed scale model for large-eddy simulation. *J. Comput. Phys.*, 224(1):241–266, 2007.

[25] T. Iliescu, H. Liu, and X. Xie. Regularized reduced order models for a stochastic Burgers equation. *Int. J. Numer. Anal. Mod.*, 15(4–5):594–607, 2018.

[26] Kento Kaneko and Paul Fischer. Augmented reduced order models for turbulence. *Frontiers in Physics*, 10, 2022. ISSN 2296-424X. doi: 10.3389/fphy.2022.905392. URL https://www.frontiersin.org/articles/10.3389/fphy.2022.905392.

[27] S. Kaya and C. C. Manica. Convergence analysis of the finite element method for a fundamental model in turbulence. *Mathematical Models and Methods in Applied Sciences*, 22(11):1250033, 2012. doi: 10.1142/S0218202512500339. URL https://doi.org/10.1142/S0218202512500339.

[28] Birgul Koc. *Numerical Analysis for Data-Driven Reduced Order Model Closures*. PhD thesis, Virginia Tech, 2021. URL http://hdl.handle.net/10919/103202.

[29] Birgul Koc, Tomás Chacón Rebollo, and Samuele Rubino. Uniform bounds with difference quotients for proper orthogonal decomposition reduced order models of the burgers equation. *Journal of Scientific Computing*, 95(2):43, Mar 2023. ISSN 1573-7691. doi: 10.1007/s10915-023-02160-2. URL https://doi.org/10.1007/s10915-023-02160-2.

[30] 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, Aug 1999. ISSN 1573-2878. doi: 10.1023/A:1021732508059. URL https://doi.org/10.1023/A:1021732508059.

[31] Mats G. Larson and Fredrik. Bengzon. *The finite element method : theory, implementation, and applications*. Springer, Berlin ;, 2013. doi: 10.1007/978-3-642-33287-6. URL https://doi.org/10.1007/978-3-642-33287-6.

[32] W. J. Layton and L. G. Rebholz. *Approximate deconvolution models of turbulence : analysis, phenomenology and numerical analysis*. Springer, Berlin ;, 2012. doi: 10.1007/978-3-642-24409-4. URL http://books.scholarsportal.info/viewdoc.html?id=/ebooks/ebooks2/springer/2012-05-29/2/9783642244094.

[33] F. Lu, C. Mou, H. Liu, and T. Iliescu. Stochastic data-driven variational multiscale reduced order models. *arXiv preprint, http://arxiv.org/abs/2209.02739*, 2022.

[34] M. Mohebujjaman, L. G. Rebholz, and T. Iliescu. Physically-constrained data-driven correction for reduced order modeling of fluid flows. *Int. J. Num. Meth. Fluids*, 89(3):103–122, 2019.

[35] C. Mou, E. Merzari, O. San, and T. Iliescu. An energy-based lengthscale for reduced order models of turbulent flows. *arXiv preprint, http://arxiv.org/abs/2108.02254*, 2022.

[36] Changhong Mou, Leslie M. Smith, and Nan Chen. Combining stochastic parameterized reduced-order models with machine learning for data assimilation and uncertainty quantification with partial observations, 2022.

[37] Julie S. Mullen and Paul F. Fischer. Filtering techniques for complex geometry fluid

flows. *Communications in Numerical Methods in Engineering*, 15(1):9–18, 1999. doi: https://doi.org/10.1002/(SICI)1099-0887(199901)15:1<9::AID-CNM219>3.0.CO;2-Y.

[38] Dylan Park, Changhong Mou, Honghu Liu, Adrian Sandu, and Traian Iliescu. A two-level Galerkin reduced order model for the steady Navier-Stokes equations, 2022.

[39] Haakon Robinson, Suraj Pawar, Adil Rasheed, and Omer San. Physics guided neural networks for modelling of non-linear dynamics. *Neural Networks*, 154:333–345, 2022. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2022.07.023. URL https://www.sciencedirect.com/science/article/pii/S0893608022002854.

[40] A. Sanfilippo, I. R. Moore, F. Ballarin, and T. Iliescu. Approximate deconvolution Leray reduced order model. 2023. in preparation.

[41] L. Sirovich. Turbulence and the dynamics of coherent structures. Parts I–III. *Quart. Appl. Math.*, 45(3):561–590, 1987.

[42] William Snyder, Changhong Mou, Honghu Liu, Omer San, Raffaella DeVita, and Traian Iliescu. *Reduced Order Model Closures: A Brief Tutorial*, pages 167–193. Springer International Publishing, Cham, 2022. ISBN 978-3-031-14324-3. doi: 10.1007/978-3-031-14324-3_8. URL https://doi.org/10.1007/978-3-031-14324-3_8.

[43] M. Strazzullo, M. Girfoglio, F. Ballarin, T. Iliescu, and G. Rozza. Consistency of the full and reduced order models for evolve-filter-relax regularization of convection-dominated, marginally-resolved flows. *Int. J. Num. Meth. Eng.*, 123(14):3148–3178, 2022.

[44] Ping-Hsuan Tsai and Paul Fischer. Parametric model-order-reduction development for unsteady convection. *Frontiers in Physics*, 10, 2022. ISSN 2296-424X. doi: 10.3389/fphy.2022.903169. URL https://www.frontiersin.org/articles/10.3389/fphy.2022.903169.

[45] Stefan Volkwein. Proper orthogonal decomposition: Theory and reduced-order modelling. *Lecture Notes, University of Konstanz*, 01 2012.

[46] Z. Wang, I. Akhtar, J. Borggaard, and T. Iliescu. Proper orthogonal decomposition closure models for turbulent flows: A numerical comparison. *Comput. Meth. Appl. Mech. Eng.*, 237-240:10–26, 2012.

[47] D. Wells, Z. Wang, X. Xie, and T. Iliescu. An evolve-then-filter regularized reduced order model for convection-dominated flows. *International Journal for Numerical Methods in Fluids*, 84(10):598–615, 2017. doi: https://doi.org/10.1002/fld.4363. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/fld.4363.

[48] David Reese Wells. *Stabilization of POD-ROMs*. PhD thesis, Virginia Tech, 2015. URL http://hdl.handle.net/10919/52960.

[49] Xie X., Iliescu T., Wells D., and Wang Z. Approximate deconvolution reduced order modeling. *Computer Methods in Applied Mechanics and Engineering*, 313:512–534, 2017. ISSN 0045-7825. doi: 10.1016/j.cma.2016.10.005. URL https://doi.org/10.1016/j.cma.2016.10.005. 512.

[50] X. Xie, D. Wells, Z. Wang, and T. Iliescu. Numerical analysis of the Leray reduced order model. *J. Comput. Appl. Math.*, 328:12–29, 2018.