# Stabilization of POD-ROMs

David Wells

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

Doctor of Philosophy
in
Mathematics

Traian Iliescu, Chair
Nathan Glatt-Holtz
Serkan Gugercin
Mark Paul

6 May 2015
Blacksburg, Virginia

Keywords: Reduced Order Modeling, Proper Orthogonal Decomposition, Large Eddy Simulation, Regularized Models, Streamline-Upwind Petrov-Galerkin, Scientific Computing

# Stabilization of POD-ROMs

David Wells

(ABSTRACT)

This thesis describes several approaches for stabilizing POD-ROMs (that is, reduced order models based on basis functions derived from the proper orthogonal decomposition) for both the CDR (convection-diffusion-reaction) equation and the NSEs (Navier-Stokes equations). Stabilization is necessary because standard POD-ROMs of convection-dominated problems usually display numerical instabilities.

The first stabilized ROM investigated is a streamline-upwind Petrov-Galerkin ROM (SUPG-ROM). I prove error estimates for the SUPG-ROM and derive optimal scalings for the stabilization parameter. I test the SUPG-ROM with the optimal parameter in the numerical simulation of a convection-dominated CDR problem. The SUPG-ROM yields more accurate results than the standard Galerkin ROM (G-ROM) by eliminating the inherent numerical artifacts (noise) in the data and dampening spurious oscillations.

I next propose two regularized ROMs (Reg-ROMs) based on ideas from large eddy simulation and turbulence theory: the Leray ROM (L-ROM) and the evolve-then-filter ROM (EF-ROM). Both Reg-ROMs use explicit POD spatial filtering to regularize (smooth) some of the terms in the standard G-ROM. I propose two different POD spatial filters: one based on the POD projection and a novel POD differential filter. These two new Reg-ROMs and the two spatial filters are investigated in the numerical simulation of the three-dimensional flow past a circular cylinder problem at $Re = 100$. The numerical results show that EF-ROM-DF is the most accurate Reg-ROM and filter combination and the differential filter generally yields better results than the projection filter. The Reg-ROMs perform significantly better than the standard G-ROM and decrease the CPU time (compared against the direct numerical simulation) by orders of magnitude (from about four days to four minutes).

*To my parents.*

# Acknowledgments

This thesis represents the culmination of nearly two years of work in model reduction (and two before that in finite elements). I would not have made it this far, in what seemed like a short period of time, without the help of both students and faculty at the Interdisciplinary Center for Applied Mathematics (ICAM), the Mathematics Department, Virginia Tech in general, and Los Alamos National Laboratory (LANL). Without people at ICAM like Dr. Zietsmann, Dr. Gugercin, Dr. Borggaard, and (of course) Dr. Iliescu, I would be at a loss for both guidance and direction. I would also like to thank my fellow students at ICAM both past and present (Weiwei Hu, Vitor Nunes, Hans-Werner van Wyk, Alan Lattimer, Boris Kramer, and Chris Jarvis, to name a few) for both their wisdom (particularly with regards to POD) and their commiseration. I would also like to thank Balu Nadiga at LANL for introducing me to (pseudo)spectral methods and for many long discussions on the nature of scientific computing and ocean modeling.

I could not have gotten this far without my academic family. As Erich said in his thesis, without the four of you (Dr. Iliescu, Dr. Wang, Dr. Foster, and Xuping Xie) "I would still be figuring things out". This also goes for Dr. John and Swetlana Giere, with whom I have worked closely during the last two years.

Perhaps the best feature of Virginia Tech is its large collection of disciplines in one place. I owe a great debt to the many teachers I have had during my time at Virginia Tech whose contributions may not be as obvious as those I have worked with directly. In particular, I would like to acknowledge Dr. John Walker for his valuable opinions on undergraduate education (as well as those more directly related to oboe playing).

This thesis depends on a large body of software. I use `matplotlib` [58] for all of the graphics in (5) and `VisIt` [30] for all of the graphics in Chapters (7) and (6). I generated all of the three-dimensional meshes with `gmsh` [46]. Finally, I owe a great "deal" to the authors of `deal.II` for their excellent finite element software, which underlies the work in the last three chapters.

# Contents

# List of Figures

# List of Tables

# List of Algorithms

# Chapter 1

# Introduction

## 1.1 Summary

In this chapter I discuss both the achievements and fundamental limits of POD-ROMs (that is, reduced order models based on proper orthogonal decomposition) as well as some of their most common applications. I show several examples of POD-ROMs that benefit from various stabilization techniques similar to the ones discussed in this thesis.

Unless otherwise noted, I only consider ROMs based on POD (e.g., I consider POD-ROM synonymous with ROM for most of this thesis).

## 1.2 Why Use Reduced Order Modeling?

The primary goal of this thesis is to develop two stabilization techniques for reduced order models and motivate their usage with examples of problems in need of stabilization. Reduced order modeling attempts to take a high-resolution data set obtained from either a physical experiment or numerical simulation and construct a cheap surrogate model that maintains the large-scale features of the original data. In this thesis I consider problems described by partial differential equations (PDEs), where the exact solution of the relevant PDE resides in a Hilbert space and has an infinite number of degrees of freedom. In this thesis I use either the pseudospectral or finite element method to discretize the relevant PDE and transform it into an $\mathcal{O}(100,000)$ degrees of freedom ordinary differential equation (ODE) for $3D$ problems, $\mathcal{O}(10,000)$ for $2D$ problems, and $\mathcal{O}(1,000)$ for $1D$ problems. It is possible to reduce even higher order systems, which are common, in practice, for three dimensional problems; I give numerical evidence that the method of snapshots scales well for large data sets in Section 4.6. In all dimensions I usually consider a ROM of $\mathcal{O}(10)$ degrees of freedom, which reduces simulation time from days to minutes. Therefore I consider the answer to the question posed in the section header to be

Reduced order modeling attempts to take a high rank model with tens or hundreds of thousands of degrees of freedom and replace it with a model with perhaps ten degrees of freedom. This is valuable whenever the same problem (or slight variations on it) must be solved repeatedly e.g., for control applications, parameter estimation, or as part of a larger problem.

## 1.3 ROM Achievements

ROMs can successfully provide surrogate models of complex, large-dimensional systems with prudent choices of snapshots. Indeed, the dependence on snapshots (rather than an underlying equation, which then requires a simulation) is frequently a strength of POD. Below are four examples of ROM achievements.

In [76] the authors used mesoscale (that is, length scales on the order of 30 to 300 kilometers) measurements of the Algerian current taken between October 1992 and December 1993 to construct an ODE model of the vortex shedding into the western Mediterranean. The authors used this ROM to calculate the apparent eddy viscosity present in the original data by utilizing the two-layer quasigeostrophic equation (QGE). The two-layer QGE is a simplified ocean model which contains two layers of fluids (a warmer surface layer and a colder, denser bottom layer) and assumes almost geostrophic balance. The principal source of instability in this model is the baroclinic instability. Indeed, the authors note "this is the minimal model accounting for these types of instabilities". The choice of eddy viscosity in the QGE is an open problem. Table 1.1 lists the wide range of eddy viscosity values used in ocean models throughout the years; as the available resolutions increase, the eddy viscosities generally decrease to more realistic values. The approach used in practice to determine the eddy viscosity value is an inverse problem: given the available mesh resolution, one determines the eddy viscosity value that yields flows matching the oceanic observations, which for the setting in [76] is a six month periodic traveling wave pattern with a Hopf bifurcation. Using a full order model for this inverse problem ([76]) would have posed untenable demands on the computational resources. Thus, the authors used a ROM to obtain an accurate eddy viscosity value in [76]. Solving such an inverse problem with a full rank PDE would be a far more computationally intensive procedure. The authors of [76] report that the value $\nu = 212$ seemed to yield flows resembling the correct, observed behavior, which for this problem is a six month periodic traveling wave pattern with a Hopf bifurcation.

Similar work was carried out in [10] where scientists used a high-fidelity heart model to construct a low rank surrogate for inverse problem applications. Inverse problems require repeated solution of nearly identical problems that are usually described by PDEs: this is an ideal application area for reduced order modeling.

One of the most popular applications of ROMs is open loop control: a cheap surrogate model is necessary for any sort of real time control of a complex system whose finite element representation may require a supercomputer to accurately represent [69, 80].

In [77] the authors successfully controlled a chemical reactor with multiphase, multicomponent flow. The governing equations for such a system account for conservation of mass,

| Study | Range of $\nu$ | Resolution |
|---|---|---|
| [20] | $500 - 10000$ | $40 \times 80$ |
| [42] | $6000 - 10000$ | $74 \times 50$ |
| [52] | 330 | $50 \times 50$ |
| [63] | 300 | $50 \times 100$ |
| [85] | 50 | $151 \times 151$ |
| [16] | $400 - 1600$ | $256 \times 256$ |
| [108] | 100 | $512 \times 256$ |
| [110] | 100 | $500 \times 500$ |

Table 1.1: Range of eddy viscosity parameters, as reported in [103].

momentum, and energy, and are too complex for a controller (usually a small embedded device) to calculate while the reactor is in operation. The authors generated snapshots with a standard fluid solver and applied POD to generate the ROM: the ROM was very successful in their simulations of open loop control. Many complex chemical engineering problems (such as the operation of distillation columns) with multiphase and multispecies flows across complex geometries are at the limit of computational science: a ROM based on empirical measurements (which may already be available) could generate a ROM for control purposes.

## 1.4 ROM Challenges

Despite its undeniable achievements (such as those listed above), ROM also has to face significant challenges. Two of these challenges are described in this section.

A fundamental challenge for the standard ROM formulation is that it cannot extrapolate well: that is, the POD functions usually cannot resolve phenomena outside of those observed in the snapshots. Put another way, standard POD is fundamentally limited in the scope of questions it may answer: the POD basis is limited by the snapshots from which it is computed.

To illustrate this fundamental challenge for standard ROMs, consider Burgers' equation [69] with the standard test problem initial conditions

$$u_0(x) = \begin{cases} 0, & x = 0 \\ 1, & 0 \leq x < 0.5 \\ 0, & 0.5 \leq x \leq 1. \end{cases} \tag{1.1}$$

I generated the snapshots with the following discretization choices: a uniform mesh with cell width 1/2048, piecewise linear finite element in the spatial discretization, and the trapezoid method with a time step of 1/2048 in the time discretization. As can be seen in Figure 1.1, as time evolves the initial condition (a step function with a sharp layer at $x = 0.5$) moves to the right while decreasing the left side of its plateau.

Although Burgers' equation is integrated over the entire time interval (i.e., $[0, 1]$), I only use snapshots from the first half of the time interval (i.e., $[0, 0.5]$) to generate the POD basis. I use the first 50 POD basis functions in the ROM. Thus, the snapshots do not capture

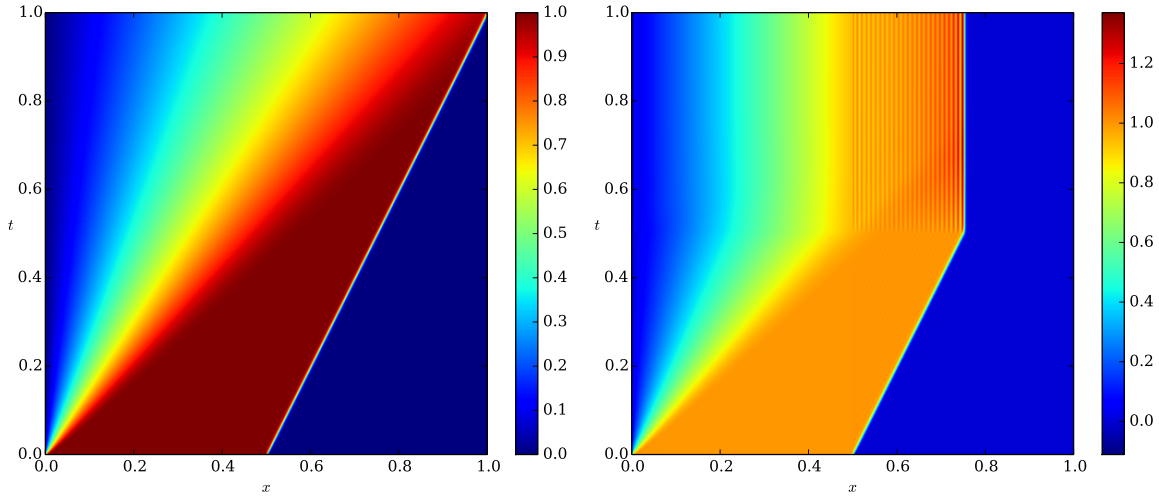


Figure 1.1: Solution of Burgers' equation for $\nu = 0.001$, 2048 linear finite element basis functions, $0 \leq x \leq 1$, and $0 \leq t \leq 1$. The DNS is on the left and the ROM solution is on the right.

the spatial structures of the solution beyond $t = 0.5$; e.g., the sharp internal layer that approaches the right end of the spatial domain. This is reflected in the ROM solution (see Figure 1.1) which accurately captures the evolution of the internal layer on the first half of the time interval but not on the second half.

When ROMs are used for convection-dominated problems (as in this thesis), in addition to the challenge discussed above, they also face the *closure problem* [117], which is briefly described next. Since standard ROMs usually employ a small (e.g., $r = \mathcal{O}(10)$) number of POD modes, the G-ROM provides an efficient surrogate model for simulating laminar flows. However, for complex flows, e.g., when the Reynolds number is high, the G-ROM does not represent a viable tool. Indeed, for example, for a three-dimensional (3D) flow past a cylinder at $Re = 10^3$, the G-ROM results were inaccurate: they over-predicted the coherent structures, kinetic energy content, and POD mode coefficient evolution [117]. Other examples of poor performance of the standard G-ROM can be found, e.g., in [6, 9, 11, 93, 117]. One of the main reasons for the poor performance of the G-ROM in these applications is that the POD modes that are not used in the G-ROM play an important role in the physical evolution of the system. Thus, to develop accurate ROMs for complex flows, the closure problem needs to be addressed, i.e., the ROM needs to model the effect of the discarded POD modes. It should be noted that the ROM closure problem is similar to the *large eddy simulation (LES)* closure problem [17, 94, 102].

**Remark 1.1** (Closure Problem)**.** *In LES, the closure problem is formally defined as follows [17, 94, 102]. A spatial filter is used to average the flow variables (i.e., velocity and pressure) and separate the large scales (i.e., the averaged variables) from the small scales (i.e., the fluctuations). This spatial filter is then applied to the NSEs to get equations for the averaged variables. Since the NSEs are nonlinear, the resulting filtered equations do not form a closed system of equations because they still involve the original unfiltered velocity. The closure problem in LES refers to modeling the unfiltered velocity in terms of the filtered velocity and,*

*thus, closing the filtered system of equations.*

*In ROM, I use the closure problem in its broader sense. Specifically, since the full rank model produces accurate results, it is clear that the main reason for the inaccurate G-ROM results is that the effect of the discarded POD modes is not modeled. In this thesis, closure modeling represents any modeling approach that aims at including the effect of the high index POD modes (i.e., the modes not included in the standard G-ROM).*

Over the last few decades, numerous ROM closure modeling strategies have been proposed to address the inaccuracy (and numerical instability) of the standard G-ROM. Although a survey of these approaches is beyond the scope of this thesis, I only mention a few: (i) calibration; (ii) finite time thermodynamics formalism [82]; (iii) special inner products to generate the POD modes [13, 65]. Some of the more recent developments can be found in, e.g., [2, 6, 9, 11, 13, 15, 82, 93, 117] and references therein.

## 1.5 ROM Stabilization

One of the most popular class of ROM closure models is the class of stabilized ROMs. These models are based on a simple idea: Since one of the most obvious drawbacks of the standard G-ROM in convection-dominated flows is instability, a ROM closure model should aim at diminishing or eliminating that. This is similar to the LES case, where the concepts of stabilization and closure are closely related.

Many stabilized ROMs have been developed over the years. To the best of my knowledge, the first stabilized ROM (and the first ROM closure model) was the mixing length model proposed by Lumley and his group [6]. In this investigation of turbulent pipe flow, before that start of the simulation, a characteristic length scale and a characteristic velocity scale were computed and used to define a turbulent viscosity. This turbulent viscosity was then added to the fluid's real viscosity to increase the ROM's stability. Many variations of the stabilized ROM in [6] were proposed over the years [88, 89, 90, 91]. One of the most successful improvements was the stabilized ROM that used a mode dependent artificial viscosity, in which more artificial viscosity was added to the higher index POD modes (i.e., those POD modes corresponding to the smaller scales) [25, 95, 96, 97]. Karniadakis and his group [105] developed a different type of stabilized ROM using Tadmor's spectral vanishing viscosity technique [109].

In the last five years, the similarity between the closure problems in LES and ROM was taken one step further. A central role in these ROM developments was played by the concept of *energy cascade*, which states that the energy in large scales is transferred to smaller and smaller scales, until is dissipated at the smallest (Kolmogorov length) scales [41, 92]. The energy cascade, combines Richardson's vision [98] and Kolmogorov's phenomenological insight [67], is a *universal* physical law that has precise scalings and is valid across a wide range of Reynolds numbers and flow settings [41, 92]. The energy cascade is the main tool used in devising LES closure models. Indeed, since the energy cascade states that the main effect of the small scales (which are discarded in the LES models) is to dissipate energy from the LES system, most LES closure models introduce some sort of numerical dissipation [17, 94, 102] and, therefore, increase the numerical stability of the resulting LES models. Since the energy cascade has been instrumental in developing closure models for

LES, one could wonder whether the same energy cascade concept could be used to develop closure models for ROM as well. First, however, one needs to address the following natural question:

(Question 1) Is (some of) the energy cascade framework also valid in a POD context?

A complete answer to Question 1 is, to my knowledge, not known. There are, however, two pieces of information that might give hope to those who aim at using the energy cascade concept to develop ROM closure models.

First, as shown on page 100 in Section 3.3.3 of [53], the POD modes satisfy the following important property:

(Property 1) For homogeneous systems, the POD modes are the Fourier modes.

A system is called homogeneous if the correlation satisfies $R(x, x') = R(x - x')$, i.e., it depends only on the difference between the two coordinates. Thus, a homogeneous system is translation invariant. Homogeneity occurs in systems that are unbounded or have periodic boundary conditions. Property 1 has been extensively exploited in practical applications of POD. For example, for turbulent channel or pipe flows, the POD modes are decomposed as the tensor product of POD components in the non-homogeneous directions and Fourier modes in the homogeneous directions (see, e.g., pages 100-101 in [53] or [6]). I believe, however, that Property 1 is also important for a completely different reason: it suggests that Question 1 might be answered in the affirmative.

Another reason (besides Property 1) that suggests a positive answer to Question 1 is the numerical investigation in [33], where Couplet, Sagaut and Basdevant study the energy transfer among the POD modes of a turbulent flow in a 3D backward-facing step at a Reynolds number $Re = 66,100$. I emphasize that the computational domain considered in [33] is homogeneous in one direction (in which Fourier components are used), but non-homogeneous in the other two directions (in which POD components are used). The authors report that "[i]t is observed that energy transfers are local in the POD basis, and that the Fourier-decomposition-based concepts of forward and backward energy cascades are also valid in the POD basis, the net effect being a forward energy cascade."

The third reason to expect answering Question 1 in the affirmative is that the vast majority of current ROM closure models are stabilizing, which is consistent with the concept of energy cascade in a POD setting.

The three reasons outlined above suggest a positive answer to Question 1. I emphasize, however, that there are several outstanding issues that need to be understood before being able to address the validity of the energy cascade in a POD setting, e.g., numerical investigations of computational settings and Reynolds number ranges that are different from those used in [33].

Notwithstanding these open questions, our group has used the concept of energy cascade to extend to POD state-of-the-art LES closure modeling strategies. Before giving a brief survey of these LES-ROMs, I mention that probably the first LES-ROM was the mixing length ROM (ML-ROM) proposed by Lumley and his group [6] (also see [25, 88, 89, 90,

91, 95, 96, 97]). The first LES-ROM proposed by our group was the Smagorinsky ROM closure model (Smag-ROM) [18] (see also [81]), which was investigated in [111, 116, 117]. The turbulent eddy viscosity coefficient in the Smag-ROM is computed dynamically, at each time step and at each point in space, and depends on the norm of the deformation tensor of the velocity. Thus, the Smag-ROM is an improvement over the ML-ROM, since the latter uses a constant turbulent eddy viscosity coefficient, computed only once, before the start of the simulation. The Smag-ROM was significantly more accurate than the ML-ROM in the numerical simulation of a 3D flow past a circular cylinder at $Re = 1000$ [117]. An improvement over the Smag-ROM was the variational multiscale ROM (VMS-ROM) proposed in [117] (see also [15, 59, 61] for different types of VMS-ROM), in which the eddy viscosity closure model acts only on the high index POD modes of the ROM. This LES-ROM is based on the variational multiscale LES model proposed by Hughes and his group [55, 56, 57], which centers around the idea of locality of energy transfer in the energy cascade. The final LES-ROM proposed in our group was the dynamic subgrid-scale ROM (DS-ROM), in which the constant in the Smagorinsky closure model is determined through a least squares approach. The DS-ROM is based on the model proposed in [45], which, together with the VMS model, is a state-of-the-art LES closure model. Both the DS-ROM and VMS-ROM were significantly more accurate than the Smag-ROM (and, of course, the ML-ROM) in the numerical investigation in [117]. These LES-ROMs are physically accurate: they approximate well both the energy spectrum (which is an average quantity) and the evolution of the POD coefficients (which is a time dependent quantity). The computational implementation of the LES-ROMs is, however, relatively challenging. Indeed, the closure models in the LES-ROMs involve nonlinearities that preclude the preassembling of some ROM matrices and tensors. Thus, a naïve implementation of the LES-ROMs would require the assembly of ROM matrices and tensors at each time step, which would render the LES-ROMs impractical. Indeed, the computational efficiency of the G-ROM is due to the preassembly of all the vectors, matrices, and tensors only once, before the start of the simulation. To avoid a prohibitive cost, more involved numerical discretizations of the LES-ROM closure models need to be adopted, e.g., the two level method used in [116, 117] (see also [5]), the empirical interpolation method [14], or the discrete empirical interpolation method [26, 27].

Regularized models [74] represent another class of closure modeling approaches for the numerical simulation of turbulent flows. The regularized models are different from the LES models since they do not add any closure model to the original set of equations. Instead, regularized models use spatial filtering to smooth some terms in the original equations. Thus, the regularized models possess an increased numerical stability and, by careful parameter tuning, yield accurate approximations of physically relevant quantities, e.g., the energy spectrum [74]. Therefore, although they use different approaches, the LES and regularized models aim at solving the same closure problem, i.e., representing the effect of the discarded scales.

In this thesis, I propose two new regularized ROMs (Reg-ROMs). To my knowledge, regularized modeling has not been used in a ROM setting. To develop these Reg-ROMs, I put forth a novel POD spatial filter, the POD differential filter. I also consider a POD projection as an explicit spatial filter. I use these two spatial filters to smooth some of the ROM terms and obtain more stable ROMs. Finally, I also propose a streamline-upwind Petrov-

Galerkin (SUPG) ROM closure model (SUPG-ROM), in which the stabilization parameters are determined through error analysis and scaling.

**Remark 1.2** (RBM Stabilization)**.** *In this thesis (and, in particular, in this section), I exclusively discuss stabilization methods for the ROMs. There is, however, an extensive body of literature on stabilization methods for the reduced basis methods (RBMs) (see, e.g., [101, 86, 11] and the references therein), which share some similarities with the approaches discussed in this thesis.*

# Chapter 2

# Notation & Acronyms

## 2.1 Overview

This section summarizes the notation I use in this thesis. While I may reintroduce notation later inline, everything I use is here. I also include a table of common acronyms at the end of this chapter.

## 2.2 Finite Element (FE) Notation

Much like in `deal.II` [12], where nearly all classes accept the template parameter `int dim`, I notate the dimensionality of the current problem as $d$. I (also like [12]) begin indexing all collections and sums at 0. In this thesis I consider examples in 1, 2, and 3 dimensions. This typically shows up in places like Equation (5.16), with terms

$$\sum_{i=0}^{d-1} \|\vec{u}_i - \vec{u}_{h,i}\|. \tag{2.1}$$

as the sum of the $L^2$ errors of the finite element approximation. I use $h$ to refer to the current cell diameter. I also use $v_h$ as a generic finite element basis function. Regardless of the choice of finite element or spatial dimension I always notate generic finite elements (each containing some number of basis functions) as $T_i$ and the set of finite elements (the triangulation) as $\mathcal{T}_h$. I always use the same finite element space for multicomponent problems (e.g., , for a three dimensional problem I use the same finite element basis functions to represent all three components). I usually express a finite element approximation of some PDE solution $\vec{u}$ as

$$\vec{u}_h[k](t, \vec{x}) = \sum_{i=0}^{N-1} c_{i,k}(t) v_{h,i}(\vec{x}) \tag{2.2}$$

where $\vec{u}_h[k]$ is the $k$th component of $\vec{u}_h$, $c_{i,k}(t)$ is the $i$th finite element coefficient of the $k$th finite element approximation, and $N$ is the number of finite element basis functions. Note that $N$, the number of finite element basis functions in dimension $k$, does not depend on $k$. I drop the $h$ subscript when it is clear I am writing about finite element snapshots of a solution.

I notate the finite element mass matrix as $M$ and the finite element stiffness (or Laplace) matrix as $S$, where for dimensionality argument $k$, $0 \leq k < d$,

$$M_k(i,j) = \int v_{hi} v_{hj} \, d\vec{x} \tag{2.3}$$

and

$$S_k(i,j) = \int \nabla v_{hi} : \nabla v_{hj} \, d\vec{x} = \int \sum_{n=0}^{d-1} v_{hi,n} u_{hj,n} \, d\vec{x} \tag{2.4}$$

where $v_{hi,d}$ is the $d$th spatial derivative of the $i$th basis function. I use the notation $M_k$ to emphasise that, in higher dimensions (such as Experiment 3.2), problems usually have a "block diagonal" mass matrix.

## 2.3 Proper Orthogonal Decomposition (POD) Notation

I use $\langle \cdot, \cdot \rangle$ with no subscript to denote the inner product to which the POD basis is orthogonal; in vector form, this inner product is

$$\langle x, y \rangle = x^T M y = x^T L L^T y \tag{2.5}$$

where the finite element mass matrix $M$ may be written as some Cholesky-like factorization $M = LL^T$. As $M$ is a sparse matrix, the usual approach for computing a Cholesky decomposition involves computing a permutation matrix $P$ such that $LP$ is lower triangular. This approach minimizes "fill-in" of the sparse matrix and may save large amounts of memory [28]. I do not rely on any properties of $L$ other than $LL^T = M$, so I ignore this detail. Unless otherwise noted, $M$ is always the finite element mass matrix (that is, the $L^2$ inner product) used to generate the POD basis. Put another way, the POD basis (unless otherwise noted) is orthogonal under the $L^2$ inner product. I use $\langle \cdot, \cdot \rangle_s$ to denote some other inner product relevant to the given problem.

Since POD is closely related to the SVD (singular value decomposition), I establish notation for SVD quantities as well. Let

$$Y = \begin{bmatrix} \vec{u}_0, & \vec{u}_1, & \cdots, & \vec{u}_{R-1} \end{bmatrix} \tag{2.6}$$

be the matrix of snapshots used to compute the POD basis, where $R$ is the number of snapshots. The snapshots, in this thesis, come from solutions of time-dependent finite element problems. Let $\nu_j$ and $\sigma_j$ notate the left singular vectors and singular values of $L^T Y$. Let $\lambda_j$ be the $j$th eigenvalue of the related system

$$Y^T M Y \tag{2.7}$$

so $\sigma_j^2 = \lambda_j$.

## 2.4 Reduced Order Modeling (ROM) Notation

I notate the ROM as $\vec{u}_r(t, \vec{x})$, given by

$$\vec{u}_r(t, \vec{x}) = \vec{u}_s + \sum_{i=0}^{r-1} \vec{a}_i(t)\vec{\varphi}_i(\vec{x}), \tag{2.8}$$

where $\vec{a}_i(t)$ is the $i$th POD coefficient, $\vec{\varphi}_i(\vec{x})$ is the $i$th POD basis function, and $\vec{u}_s$ is the centering trajectory (i.e., the mean of the snapshots)

$$\vec{u}_s = \frac{1}{R} \sum_{i=0}^{R-1} \vec{u}_i. \tag{2.9}$$

For the Navier-Stokes ROMs I notate the nonlinearity as

$$N[i, j, k] = \int \vec{\varphi}_i \cdot \vec{\varphi}_j \cdot \nabla\vec{\varphi}_k \, d\vec{x}. \tag{2.10}$$

I fully explain the usage of this tensor in Section 6.3. I notate the combination of linear terms (i.e., those resulting from use of the centering trajectory) as $\tilde{L}_r$, which is fully explained in Section 6.4. I notate the ROM analogues of the finite element matrices with the subscript $r$: for example, POD the mass matrix $M_r$ has entries

$$M_r(i, j) = \int \vec{\varphi}_i \cdot \vec{\varphi}_j \, d\vec{x}. \tag{2.11}$$

## 2.5 Table of Acronyms

| Acronym | Expansion |
|---|---|
| CDRE | Convection-Diffusion-Reaction Equation |
| DNS | Direct Numerical Simulation |
| DF | Differential Filter |
| EF-ROM-DF | Evolve then filter ROM with the POD Differential Filter |
| EF-ROM-Proj | Evolve then filter ROM with the POD Projection Filter |
| FE | Finite Element |
| L-ROM-DF | Leray ROM with the Differential Filter |
| L-ROM-Proj | Leray ROM with the Projection Filter |
| LES | Large Eddy Simulation |
| NSEs | Navier-Stokes Equations |
| POD | Proper Orthogonal Decomposition |
| QGEs | Quasi-Geostrophic Equations |
| Reg-ROM | Regularized Reduced Order Model |
| ROM | Reduced Order Model |
| SUPG | Streamline-Upwind Petrov-Galerkin |

Table 2.1: List of acronyms used in this thesis.

# Chapter 3

# Numerical Experiments

## 3.1 Overview

This thesis relies on a few numerical experiments that I cite in multiple places. Since the details do not quite fit in any particular section I describe the equations and discretizations in this chapter.

The principal goal of this thesis is to demonstrate practical algorithms for stabilizing ROMs describing complicated flows. Therefore I use several numerical experiments to demonstrate properties of the POD basis. Since the details of each equation and discretization are common to several chapters I describe them here instead of scattering this information throughout the rest of this thesis.

## 3.2 Flow Past a Cylinder in 3D

Flow past a cylinder is a common test problem for ROMs because the flow generates a vortex shedding pattern that has been observed in a variety of settings. To this end I model the flow with the unforced Navier-Stokes equations (NSEs):

$$\begin{aligned} \vec{u}_t &= \frac{1}{Re}\Delta\vec{u} - \vec{u}\cdot\nabla\vec{u} - \nabla p \\ \nabla\cdot\vec{u} &= 0, \end{aligned} \tag{3.1}$$

where $\vec{u}$ is the velocity field, $p$ is the pressure, and $Re$ is the Reynolds number. I use the $3D$ case because it is a more challenging problem for reduced order modeling; the authors of [62] and [78] noted that the ROM only needs about six basis functions to fully resolve the vortex shedding pattern in $2D$.

I consider a flow past a cylinder problem with the fluid described by the NSEs with simple inflow and outflow conditions. This setup is nearly identical to the setup used in the `step-35` example included in the `deal.II` library [12].

The channel is 4.5 units wide, 2.5 units tall, and 25 units tall. The cylinder has a diameter of 1 unit and is positioned in the center of the channel 2.5 units away from the inflow.

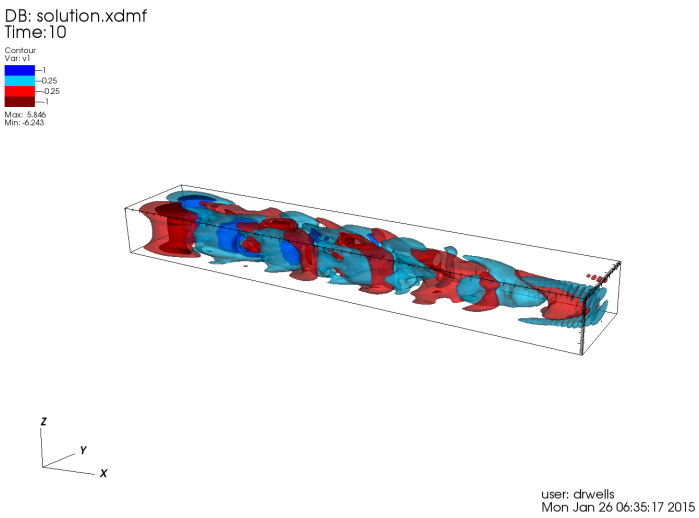


Figure 3.1: Snapshot of the flow past a cylinder problem. I plot isosurfaces of the $y$ velocity, with blue for positive values and red for negative values. Even though the Reynolds number is relatively small, the constrained geometry produced interesting effects. The outflow contains some numerical oscillations.

The inflow condition, in $2D$ or $3D$, is

$$\vec{u} = \begin{bmatrix} 6y(H-y)/H^2 \\ 0 \end{bmatrix} \text{ or } \vec{u} = \begin{bmatrix} 6z(H-z)/H^2 \\ 0 \\ 0 \end{bmatrix}, \text{ respectively,} \tag{3.2}$$

where $H$ is the length of the vertical (either $y$ or $z$ coordinate, depending on the dimension) inflow component. I use the no slip boundary conditions on the interior and exterior walls. I use

$$\vec{u} \times \vec{n} = \begin{bmatrix} 0 & w & -v \end{bmatrix} = \vec{0} \text{ and } p = 0 \tag{3.3}$$

for the outflow condition, where $\vec{n}$ is the unit outward normal. This corresponds to zero velocity in the $y$ and $z$ directions at the outflow in 3D and zero velocity in the $y$ direction in 2D. I discretize the domain into hexahedra with triquadratic basis functions for velocity and trilinear basis functions for pressure.

The time stepping scheme is a fractional step method based on the work of Guermond, Minev, Quartapelle, and Shen; see [50] and [49] for additional details. This method was proved to be unconditionally stable in [50]. Briefly, the method consists of an extrapolation step followed by diffusion and projection:

1. Extrapolation step: Define extrapolated variables $\vec{u}^*$ and $p^\#$ as

$$u^* = 2u^k - u^{k-1} \tag{3.4}$$

and

$$p^\# = p^k + \frac{4}{3}\phi^k - \frac{1}{3}\phi^{k-1}, \tag{3.5}$$

where $\phi^n$ is an auxiliary variable that roughly corresponds to the pressure update.

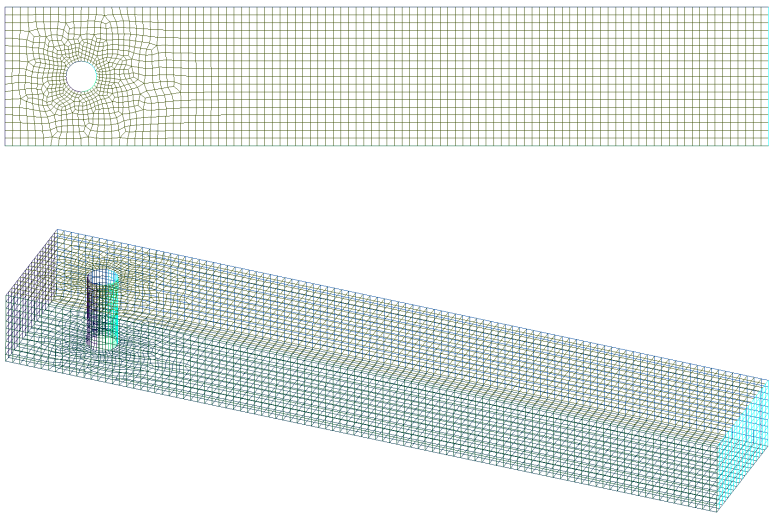

Figure 3.2: The computational mesh used for Experiment 3.2. I generated it by extruding a 2D mesh with `GMSH` [46]. The top picture is a vertical slice and the bottom picture shows the element boundaries along surfaces.

2. Diffusion step: Find $\vec{u}^{k+1}$ that solves the *linear* equation

$$\frac{1}{2\Delta t}(3u^{k+1} - 4u^k + u^{k-1}) + \vec{u}^* \cdot \nabla \vec{u}^{k+1} + \frac{1}{2}(\nabla \cdot \vec{u}^*)\vec{u}^{k+1} - \frac{1}{Re}\Delta u^{k+1} + \nabla p^{\#} = \vec{0} \quad (3.6)$$

   with appropriate enforcement of boundary conditions on $\vec{u}^{k+1}$.

3. Pressure correction: Calculate the pressure by the update formula

$$p^{k+1} = p^k + \phi^{k+1} - \frac{1}{Re}\nabla \cdot \vec{u}^{k+1}. \quad (3.7)$$

## 3.3 Gyres Forming & Dissipating

Finally, I occasionally refer to the quasigeostrophic equations (QGEs) on the rectangular domain $[0,1] \times [0,2]$

$$\begin{aligned} \omega_t &= \frac{1}{Re}\Delta\omega + \psi_x\omega_y - \psi_y\omega_x + \frac{1}{Ro}\psi_x + \frac{1}{Ro}\sin(\pi(y-1)) \\ \omega &= -\Delta\psi, \end{aligned} \quad (3.8)$$

where $\omega$ is the vorticity, $\psi$ is the streamfunction, and $Ro$ is the Rossby number, a dimensionless ratio between the inertial and rotational forces. I use $Re = 450$ and $Ro = 0.0036$, which corresponds to a moderate amount of rotational forces; lower Rossby numbers decrease the width of the western boundary layer (that is, a thin boundary layer appearing along $x = 0$) which makes the problem more difficult to solve. I discretize the domain with a double sine expansion (a pseudospectral method)

$$\omega(t,x,y) \approx= \sum_{i=1}^{257}\sum_{j=1}^{513} a_{ij}(t)\sin(i\pi x)\sin(j\pi y), \quad (3.9)$$

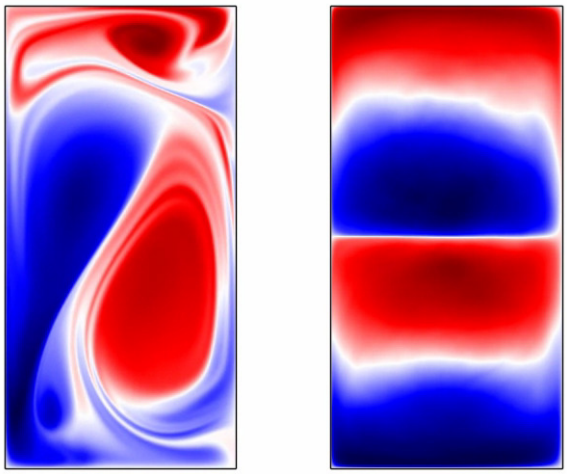

Figure 3.3: Instantaneous (left) and time-average (right) plot of a solution of the QGEs. Blue corresponds to negative vorticity (counterclockwise rotation) and red corresponds to positive vorticity (clockwise rotation).

where I chose the values 255 and 511 both for faster discrete sine algorithms (see [40] and the online documentation of `FFTW` for further information) and to ensure that I resolve all scales of motion for the given Reynolds and Rossby numbers.

An integral part of the behavior of the QGE is the formation of large gyres and their subsequent break up into small gyres: the ROM can capture the former but not necessarily the later.

# Chapter 4

# Proper Orthogonal Decomposition (POD)

## 4.1 Introduction

Ultimately, proper orthogonal decomposition is a data analysis tool for deriving the "most pertinent" features from a set of data. Dealing with inaccuracies, oscillations, or other errors in the data is an open problem that is partially addressed in Chapter 5; however, I do not consider these issues when computing the POD of a dataset, and leave these worries for the ROM itself. POD attempts to answer the question

> Given a set of data with a high dimensionality, what is the best (under some norm) approximation to the data for a given rank $r$?

In a finite element context, this is equivalent to the question

> Given a set of $d$ solutions to a finite element discretization at time instances $t_0, t_1, \cdots t_{d-1}$, each consisting of $N$ locally defined basis functions, what would have been the best choices for $r$ globally defined basis functions?

Put another way, given a set of solutions to some time-evolved finite element problem, we may calculate an *optimal* set of basis functions (to replace the standard piecewise polynomials, where each basis function has support on a small number of cells) that are each defined over *every* cell.

From this point onwards, I will refer to the solutions of the time-evolved finite element problem at time instances $t_0, t_1, \cdots, t_{d-1}$ as the *snapshots*.

## 4.2 Motivation

Simulating many complex problems requires the repeated solution of subproblems. This is particularly true in fields like optimization and control, where the same system may be solved over and over again for slightly different input data or parameters.

Reduced order modeling aims to ameliorate this issue. The goal of reduced order modeling is to replace such subproblems with cheap surrogates with an acceptable loss in accuracy.

This thesis exclusively considers reduced order models based on the proper orthogonal decomposition, or ROMs for short. This method is sometimes referred to as the *method of empirical basis functions* for exactly this reason: the basis functions are computed from measurements from either numerical or physical experiments.

There are many situations where Galerkin ROMs (henceforth abbreviated as G-ROMs) are sufficiently stable and accurate. Indeed, for the standard flow past a cylinder model for the NSEs in $2D$, the essential features of the flow may be captured by as few as six POD vectors. More examples include [53, 72, 80]. The G-ROM may produce inaccurate results in other situations (e.g., [7]): the main focus of this thesis is to investigate techniques for ameliorating the G-ROM's performance. One of the main reasons for these inaccurate results is that the underlying G-ROM may be numerically unstable, e.g., the G-ROM solution can blow up in a nonphysical way (see [2, 66, 13] for the compressible NSEs and [8, 105] for the incompressible NSEs).

The principal question behind stabilization of ROMs is simply

> Given a set of $r$ basis functions, what is the best numerical method to use for the ROM to provide both accuracy and stability?

The optimality of the POD basis comes from the SVD: the first $r$ basis functions, computed by the SVD as the first $r$ left singular vectors (modulo a coordinate change), form the optimal (with respect to the Frobenius norm) basis for approximating the given snapshots.

## 4.3 Mathematical Formulation

I recommend reading [21, 66, 69, 106, 114, 115] for more detailed accounts of the mathematical formulation of POD. Stated formally, the POD problem is to take a set of snapshots $Y$ that represent the essential behavior of a system and recombine them as a set of basis functions

$$\{\vec{\varphi}_0, \vec{\varphi}_1, \cdots, \vec{\varphi}_{d-1}\} \tag{4.1}$$

such that $(\vec{\varphi}_i, \vec{\varphi}_j)_X = \delta_{ij}$ and the POD vectors are ordered from most significant to least significant. I exclusively consider ROMs that are orthogonal under the $L^2$ inner product: this is the most common choice in the literature. Some results indicate that the $H^1$ norm may be a better choice for flow problems. This problem is equivalent to

$$\min \sum_{j=0}^{d-1} \left\| \vec{u}_j - \sum_{i=0}^{r-1} (\vec{u}_j, \vec{\varphi}_i)\vec{\varphi}_i \right\|_X^2. \tag{4.2}$$

In particular, this is equivalent to the eigenvalue problem

$$Cv_i = \sigma_i^2 v_i \tag{4.3}$$

where

$$\vec{\varphi}_i = \frac{1}{\sigma_i} \sum_{j=0}^{d-1} (v_i)_j \vec{u}_j, \tag{4.4}$$

$C(i,j) = (\vec{u}_i, \vec{u}_j)_X$ is the correlation matrix, which I discuss below in Section 4.5 in greater detail, and $\sigma_i^2$ is the $i$th eigenvalue. I use the notation $\sigma_i^2$ for consistency with the POD basis generated from the SVD algorithm (see Section 4.4). The notation $\sigma_i^2$ also emphasises the point that the eigenvalues of this system are both real and nonnegative.

## 4.4 The SVD Algorithm

I provide a sample implementation for this algorithm in Subsection A.2.1. The algorithm for scalar problems is, briefly:

1. Load the snapshots into memory. I notate the $i$th snapshot by $\vec{u}_i$, which (unless otherwise noted) is a vector of finite element basis function coefficients.
2. Center the snapshots by subtracting the component-wise average of the snapshots, $\vec{u}_s$, from each snapshot.
3. Factorize the mass matrix, $M$, as $LL^T$. As I mentioned above, the structure of $L$ does not matter.
4. Compute as many singular vectors $\nu_j$ and singular values $\sigma_j$ of $L^T Y$ as are desired, where $Y = \begin{bmatrix} \vec{u}_0 & \vec{u}_1 & \cdots & \vec{u}_{R-1} \end{bmatrix}$.
5. Compute the POD vectors as $L^T \vec{\varphi}_j = \nu_j$.

This algorithm is useful when only a few POD basis functions are needed: they may be quickly calculated with an iterative sparse SVD solver. This algorithm also emphasises the optimality of the POD basis: the optimality properties of the left singular vectors apply (after the coordinate transformation induced from changing orthogonality under the Euclidean inner product to the inner product induced by $M$) to the POD basis.

## 4.5 The Method of Snapshots

The SVD algorithm may not be appropriate for very large datasets due to the amount of memory required by the Cholesky factor. The most common approach to generating POD vectors is the *method of snapshots*, which was proposed by Sirovich in [106]. I provide a sample implementation of this algorithm in Subsection A.3.1. Unless otherwise noted, I use the method of snapshots to compute POD basis vectors in all examples as it is better suited for large data sets. The algorithm for vector-valued problems is, briefly:

1. Load the snapshots into memory. I use the same notation as in the SVD based algorithm. Each snapshot $\vec{u}_i$ is (in the notation of `deal.II`) a block vector, where $\vec{u}_{i0}$ is the $x$-component, $\vec{u}_{i1}$ is the $y$-component, and so on.
2. Center the snapshots by subtracting the component-wise average of the snapshots, $\vec{u}_s$, from each snapshot, as in Equation (2.9).

3. Form the correlation matrix
$$C(i,j) = \vec{u}_i^T \vec{M} \vec{u}_j \tag{4.5}$$
where
$$\vec{M} = \begin{pmatrix} M & 0 \\ 0 & M \end{pmatrix} \text{ or } \vec{M} = \begin{pmatrix} M & 0 & 0 \\ 0 & M & 0 \\ 0 & 0 & M \end{pmatrix} \tag{4.6}$$
depending on the dimensionality of the problem. This part can take a long time, particularly in three dimensions.

4. Calculate the eigenvectors and eigenvalues of $C$, $\nu_i$ and $\lambda_i$. As $C$ is symmetric, the eigenvalues are real, and thus may be sorted in descending order (so $\lambda_0 \geq \lambda_1 \geq \cdots$). There are many suitable algorithms available for doing this accurately; I use the LAPACK routine `DSYEVX` to compute the eigenvalues and eigenvectors, which uses a tridiagonalization algorithm followed by postprocessing [4].

5. The $i$th singular value, $\sigma_i$, is simply $\sqrt{\lambda_i}$. The $i$th singular vector $\varphi_i$ is
$$\varphi_i = \sum_{n=0}^{R-1} \vec{u}_n \nu_i(n). \tag{4.7}$$

As the POD vectors are simply linear combinations of the snapshots, all linear properties are preserved. Therefore the POD vectors automatically satisfy homogeneous boundary conditions and any incompressibility condition that the snapshots possess.

The principal advantage of the method of snapshots is that it provides a much easier linear algebra problem than the SVD algorithm. The system solved by `DSYEVX` is $R \times R$, while the system solved by the SVD algorithm is $R \times N$, where typically $N \gg R$. For example, for the NSEs data set used in this chapter, $R = 2001$ and $N = 600,000$, which means the array necessary to store the matrix for SVD calculations is about 9 gigabytes in size while the array used for the method of snapshots is only about 30 megabytes.

## 4.6 Accuracy of the Method of Snapshots

As the method of snapshots involves matrix-matrix products of the form
$$C = \begin{bmatrix} \vec{u}_0 \\ \cdots \\ \vec{u}_{R-1} \end{bmatrix}^T \vec{M} \begin{bmatrix} \vec{u}_0 \\ \cdots \\ \vec{u}_{R-1} \end{bmatrix} \tag{4.8}$$
one may expect that the system is ill-conditioned. Indeed, the method is derived from the SVD algorithm described in Section 4.4 by forming the equivalent eigenvalue problem
$$A^T A, \text{ where } A = L^T Y. \tag{4.9}$$
In particular, inaccurately calculated eigenvectors for this relatively small ($R \times R$) matrix may not recombine into orthogonal snapshots in the next step of the algorithm. I address the accuracy of the method of snapshots by answering two questions:

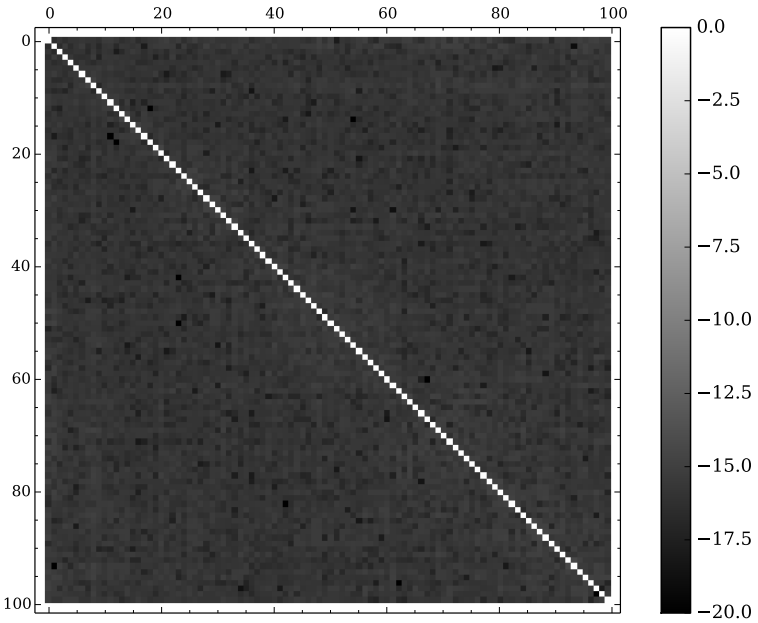


Figure 4.1: Plot of the first $100 \times 100$ block of the magnitudes of the entries of the POD mass matrix: that is, the values $\log_{10}(|M_r(i,j)|)$ for $0 \leq i \leq 99$ and $0 \leq j \leq 99$. The off-diagonal entries of $M_r$ have a maximum magnitude of $3.65 \times 10^{-15}$, which is represented on the plot as $-14.437$. Some of the off-diagonal entries are actually zero; to clarify the plot I colored these blocks black.

1. Does either the method of snapshots or the reduced order matrices suffer a loss of accuracy from inaccurate inner product calculations?

2. Do the POD vectors calculated by the method of snapshots recover the POD interpolation error equations such as Equation (4.33)?

To examine the accuracy of the method of snapshots, I use 2001 snapshots from Experiment 3.2 with $Re = 100$ and the centering trajectory. Each snapshot has approximately $600,000$ degrees of freedom. I show below that, with a careful choice for an inner product algorithm, accuracy is not a serious concern for the method of snapshots.

The first issue I investigate is the ability to form the reduced matrices (that is, the reduced mass and Laplacian matrix) from POD vectors derived from the method of snapshots. Fortunately, the problem of forming accurate vector-vector products, to which this issue is analogous, is fairly well understood: `deal.II` natively uses a blocked algorithm with blocks of size 2. In particular, the file `include/deal.II/lac/vector.templates.h` (approximately on lines 1156 through 1167) offers the following explanation in the documentation of a generic accumulation function [12]:

> "[T]his is the main working loop for all vector sums using the templated operation above. it accumulates the sums using a block-wise summation algorithm with post-update. this blocked algorithm has been proposed in a similar form by Castaldo, Whaley and Chronopoulos (SIAM J. Sci. Comput. 31, 1156-1174,

> 2008) and we use the smallest possible block size, 2. Sometimes it is referred to as pairwise summation. The worst case error made by this algorithm is on the order O(eps * log2(vec_size)), whereas a naive summation is O(eps * vec_size). Even though the Kahan summation is even more accurate with an error O(eps) by carrying along remainders not captured by the main sum, that involves additional costs which are not worthwhile. See the Wikipedia article on the Kahan summation algorithm."

Indeed, a numerical experiment shows that the Frobenius norm of the entries not along the main diagonal of the entire (that is, $R \times R$) mass matrix formed with the default (blockwise summation) algorithm is $2.10574 \times 10^{-12}$, while the Frobenius norm of the same entries in the mass matrix formed from the naïve inner product algorithm is $7.789 \times 10^{-12}$. While the errors are small in magnitude, I expect this difference to grow for engineering and geophysical datasets which may contain hundreds of millions (rather than hundreds of thousands) of entries.

A fortunate result of the accurate inner product algorithm (besides yielding nice orthogality properties without use of reorthogonalization) is that the entries of the correlation matrix $C$ are very accurate. This guarantees, when coupled with an accurate symmetric eigenvalue solver, that the singular values inherit this accuracy.

The second issue is the accuracy of the POD vectors themselves: they should recover the exact error result (4.33) up to some negligible loss in accuracy. To verify this, I check the interpolation result for a number of POD vectors $r_1$, where $1 \leq r_1 \leq 100$. I summarize the results in Table 4.1.

## 4.7 Effect of Centering the Snapshots

As I mention above in Section 4.4 and Section 4.5, usually in practice the POD basis is computed from the *fluctuations* of the snapshots, or the set

$$\{\vec{u}_0 - \vec{u}_s, \vec{u}_1 - \vec{u}_s, \cdots, \vec{u}_{R-1} - \vec{u}_s\}. \tag{4.10}$$

This approach tends to result in more accurate ROMs.

One benefit of this approach is the conversion between time-independent boundary conditions (either Dirichlet or Neumann) and simple homogeneous boundary conditions. For example, consider a flow with a specified inflow condition $g(\vec{x})$ on the boundary $\Gamma_0$; hence, for every snapshot (and therefore the mean) the following equality holds:

$$\vec{u}_i\Big|_{\Gamma_0} = \vec{u}_s\Big|_{\Gamma_0} = g(\vec{x}), \tag{4.11}$$

so trivially

$$(\vec{u}_i - \vec{u}_s)\Big|_{\Gamma_0} = \vec{0}. \tag{4.12}$$

| $r_1$ | $\sum_{i=r_1}^{R-1} \sigma_i^2$ | $\sum_{n=0}^{R-1} \left\| \vec{u}_n - \sum_{i=0}^{r_1-1} \langle \vec{u}_n, \vec{\varphi}_i \rangle \vec{\varphi}_i \right\|^2$ |
|---|---|---|
| 2 | 182753.567915 | 182753.570693 |
| 4 | 164311.705302 | 164311.712343 |
| 6 | 156296.758264 | 156296.757146 |
| 8 | 148780.806184 | 148780.808336 |
| 10 | 141653.502162 | 141653.507387 |
| 20 | 114794.313701 | 114794.326822 |
| 30 | 96979.8741540 | 96979.8850875 |
| 40 | 83565.3337824 | 83565.3313631 |
| 50 | 73562.9786325 | 73562.9771764 |
| 60 | 65667.1960201 | 65667.1963493 |
| 70 | 59289.0654157 | 59289.0645334 |
| 80 | 53841.1631402 | 53841.1635371 |
| 90 | 49103.5845715 | 49103.5868911 |
| 100 | 45045.8004678 | 45045.8035251 |

Table 4.1: Numerical evidence that the method of snapshots is sufficiently accurate for the problems I consider in this thesis. The sum of squares of singular values, where the singular values themselves are accurate by the previous result, have about seven or eight digits in agreement with the projection residue norm. I expect that, relative to the errors incurred by model reduction, the error in the calculation of the POD basis vectors themselves will always be negligible.

| index | uncentered data | centered data |
|---|---|---|
| 1 | 2827.71 | 199.655 |
| 2 | 199.655 | 199.1 |
| 3 | 199.1 | 81.502 |
| 4 | 81.502 | 81.3539 |
| 5 | 81.3538 | 62.5117 |
| 6 | 62.3241 | 53.9131 |
| 7 | 53.9131 | 53.8783 |
| 8 | 53.8783 | 52.8608 |
| 9 | 52.8608 | 52.7477 |
| 10 | 52.7476 | 46.4141 |

Table 4.2: The first ten singular values for Experiment 3.2, with the POD computed from both centered and uncentered data. The singular values in both columns, if one ignores the first uncentered singular value, line up surprisingly well.

This provides a very clean implementation of boundary conditions in the ROM, as for all POD vectors $\vec{\varphi}_i$ (for the above example)

$$\vec{\varphi}_i\Big|_{\Gamma_0} = \vec{0}. \tag{4.13}$$

Therefore, in a ROM that uses the centering trajectory, the boundary conditions are enforced by $\vec{u}_s$, meaning that (as $\vec{u}_s$ is not evolved in time) the centered ROM automatically satisfies all time-independent boundary conditions. To the best of my knowledge this is the simplest way to enforce boundary conditions in a ROM; see [48] for an alternative to the centering trajectory. The centering trajectory enforces boundary conditions exactly and requires no further effort. Another useful property of the centered ROM is that it tends, at least for simple examples, to yield significantly better results than the ROM with POD vectors calculated from uncentered data [62]. I use the centering trajectory on every ROM in this thesis because of these two reasons.

There is an interesting empirical relationship between $\vec{u}_s$ and the first POD vector, which is also observed in [62]. I expect that the "best" single-vector approximation to the uncentered data set is the average and in practice this is reasonably true. For this experiment I used 2001 snapshots between $t = 30$ and $t = 50$ and the method of snapshots to generate two POD bases: one from centered data and one from uncentered data.

Another surprising property of $\vec{u}_s$ is that it is "somewhat" orthogonal to the centered POD vectors. In particular, if one tries to project the centering trajectory of Experiment 3.2 onto the first ten POD vectors, the norm of the projection is

$$\left|\left|\sum_{j=9}^{9} a_j \vec{\varphi}_j\right|\right|_{L^2} \approx 1.0899, \tag{4.14}$$

where $a_j = (\vec{\varphi}_j, \vec{u}_s)$ while the norm of the centering trajectory is

$$||\vec{u}_s||_{L^2} \approx 63.21. \tag{4.15}$$

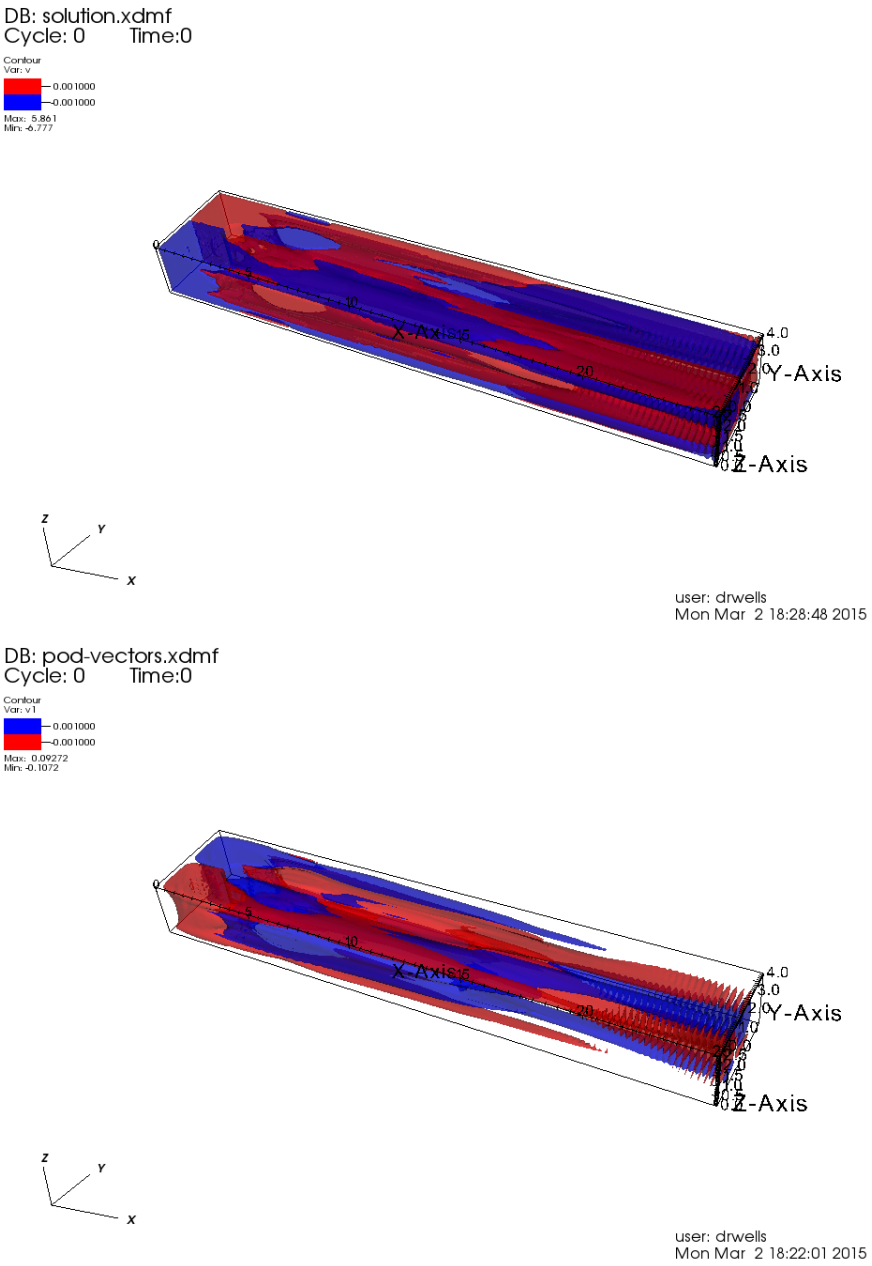


Figure 4.2: Plots of isosurfaces of the $y$-velocity of both the mean (top) and first POD vector from uncentered data (bottom) from Experiment 3.2. While the two pictures are not quite identical (partially due to scaling differences, as the POD vectors are normalized while the mean vector is not) the resemblance is clear.

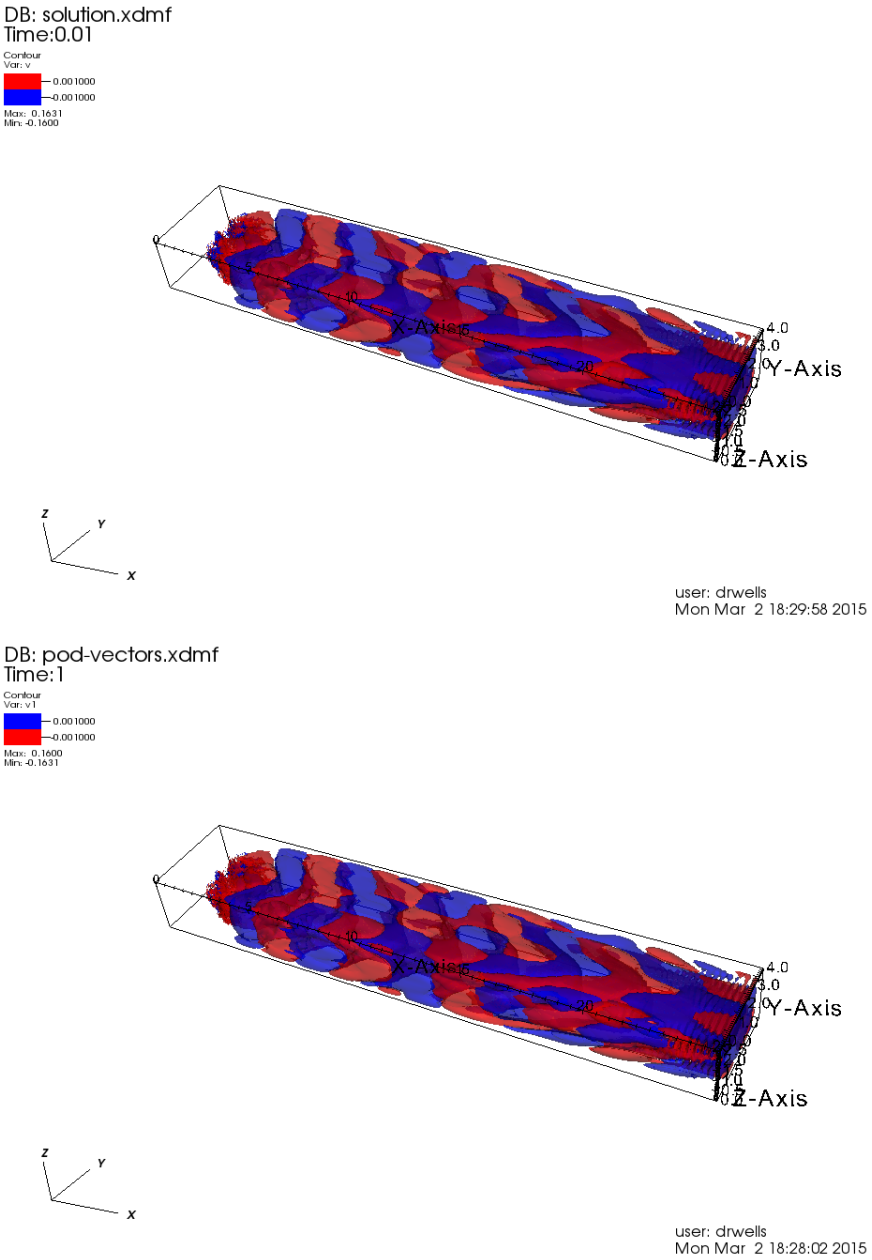


Figure 4.3: Plots of isosurfaces of the $y$-velocity of the first POD vector from centered data (top) and the *second* POD vector from uncentered data (bottom) for the same experiment as Figure 4.2. The resemblance is surprisingly strong.

The plot of the projection of the centering trajectory in Figure 4.4 onto the first ten POD vectors demonstrates why this is true.

## 4.8 Comparison with the Fourier Basis

One strategy for examining POD bases (which depend on the data set, and are therefore specific to each problem) is to examine the Fourier basis instead. This is useful because the POD basis is empirical and relies on the considered problem through the snapshots, while the Fourier basis is universal. This allows one to examine scalings in a more general context. In particular, in both [60] and [47] the authors considered the odd part of the basis functions, or $\vec{\varphi}_j = \sin(\pi j x)$, as a surrogate for the POD basis. This analogy works well for simple problems (in particular, the authors above considered Burgers equation and the convection-diffusion-reaction equation) but begins to break down for the NSEs. In particular, the $H^1$ seminorms of the odd Fourier basis in one dimension grow like

$$\sqrt{\int_0^1 \left(\frac{d}{dx}\sin\left((j+1)\pi x\right)\right)^2 dx} = \frac{\sqrt{2}(j+1)\pi}{2}. \tag{4.16}$$

I observe in Figure 4.5 that the growth is to $\mathcal{O}(j)$ with a constant offset.

## 4.9 Inverse Estimates

Kunisch and Volkwein, in [70], computed the POD basis from snapshots residing in an infinite dimensional Hilbert space (notated here as $X$). The authors proved that for a snapshot space $X^R$ with dimension $R$

$$\|\nabla v_R\|_0 \leq \sqrt{\|S_R\|_2 \|M_R^{-1}\|_2}\|v_d\|_0, \tag{4.17}$$

where $\|\cdot\|_2$ refers to the matrix 2 norm, $M_R$ is the POD mass matrix with *every* POD vector (that is, $M_R(i,j) = (\vec{\varphi}_i, \vec{\varphi}_j)$ for $i \leq R$ and $j \leq R$), and $S_R$ is the POD stiffness matrix with an analogous definition, for all $v_R \in X^R$. This result first appeared in [70].

I consider a finite-dimensional snapshot basis, where $X^r \subset X^h$. Therefore all of the usual finite element inverse estimates apply to the POD basis functions, or in the notation of this thesis

$$\vec{\varphi}_i \in \text{span}\left\{v_0, \cdots, v_{N-1}\right\}, \forall i \in 0, \cdots, r-1. \tag{4.18}$$

Let $H_r$ denote the POD Hessian matrix, so

$$(H_r)_{ij} = \sum_{T \in \mathcal{T}_h} (\Delta\vec{\varphi}_i, \Delta\vec{\varphi}_j)_T. \tag{4.19}$$

The summation is necessary because the Laplacian of individual finite element functions are only defined on individual mesh cells $T \in \mathcal{T}_h$.

**Remark 4.1.** *Note that the POD Hessian matrix $H_r$ can be only defined if $X^r \subset X^h$. If I were to use the construction used in [70], i.e., use a POD basis built from the snapshots coming directly from $X$, then the definition* (4.19) *would not be possible.*

I follow the approach used in [70] and derive a new inverse estimate involving $H_r$.

**Lemma 4.2** (POD Inverse Estimate)**.** *For all $\vec{v}_r \in X^r$, the following estimate holds:*

$$\|\Delta \vec{v}_r\|_0 \leq \sqrt{\|H_r\|_2 \, \|S_r^{-1}\|_2} \, \|\nabla \vec{v}_r\|_0. \tag{4.20}$$

*Proof.* As $\vec{v}_r \in X^r$, I may write it as a sum of POD basis functions, or

$$\vec{v}_r = \sum_{i=0}^{r-1} x_i \vec{\varphi}_i. \tag{4.21}$$

Notate the vector of POD coefficients by $\mathbf{x} = (x_0, \cdots, x_{r-1})^T$. From the definition of $H_r$, it follows that

$$\|\Delta \vec{v}_r\|_0^2 = \mathbf{x}^T H_r \mathbf{x}. \tag{4.22}$$

Since the Hessian matrix is symmetric, its matrix 2 norm is equal to its Rayleigh quotient [34]:

$$\|H_r\|_2 = \max_{\mathbf{x} \neq \mathbf{0}} \frac{\mathbf{x}^T \, H_r \, \mathbf{x}}{\mathbf{x}^T \, \mathbf{x}}. \tag{4.23}$$

Therefore, using Equations (4.22) and (4.23) I obtain

$$\|\Delta \vec{v}_r\|_0^2 = \mathbf{x}^T \, H_r \, \mathbf{x} \leq \|H_r\|_2 \, \mathbf{x}^T \, \mathbf{x} \, . \tag{4.24}$$

Furthermore, since $S_r^{-1}$ is also symmetric, I obtain

$$\mathbf{y}^T \, S_r^{-1} \, \mathbf{y} \leq \|S_r^{-1}\|_2 \, \mathbf{y}^T \, \mathbf{y} \tag{4.25}$$

for all vectors $\mathbf{y} \in \mathbb{R}^r$. I also note that, since $S_r$ is symmetric positive definite, I may use its Cholesky decomposition $S_r = L_r L_r^T$, where $L_r$ is a lower triangular nonsingular matrix. As $L_r$ is nonsingular, there exists some $\mathbf{y}$ such that

$$\mathbf{y} = L_r \, \mathbf{x}. \tag{4.26}$$

Therefore, using Equations (4.25) and (4.26), I obtain

$$\begin{aligned} \|S_r^{-1}\|_2 &\geq \frac{\mathbf{y}^T S_r^{-1} \mathbf{y}}{\mathbf{y}^T \mathbf{y}} \\ &= \frac{\mathbf{x}^T L_r^T (L_r^{-1})^T L_r^{-1} L_r \mathbf{x}}{\mathbf{x}^T L^T L \mathbf{x}} \\ &= \frac{\mathbf{x}^T \mathbf{x}}{\mathbf{x}^T S_r \mathbf{x}}. \end{aligned} \tag{4.27}$$

Hence, combining estimate (4.27) with (4.22), I obtain

$$\|\Delta v_r\|_0 = \sqrt{\mathbf{x}^T H_r \mathbf{x}} \leq \sqrt{\|H_r\|_2 \mathbf{x}^T \mathbf{x}} \tag{4.28}$$

$$\leq \sqrt{\|H_r\|_2 \|S_r^{-1}\|_2 \mathbf{x}^T S_r \mathbf{x}} \tag{4.29}$$

$$= \sqrt{\|H_r\|_2 \|S_r^{-1}\|_2} \|\nabla v_r\|_0. \tag{4.30}$$

□

**Remark 4.3.** *The POD inverse estimates* (4.17) *and* (4.20) *are global over the whole domain* $\Omega$.

### 4.9.1 Asymptotic Behavior

A simple way to examine the scalings in Lemma 4.2 is to assume that the POD basis is the Fourier basis with Dirichlet boundary conditions, i.e., $\vec{\varphi}_j(x) = \sin(\pi(j+1)x)$ on $[0,1]$. With this simplification $H_r$ is diagonal and has entries

$$\int_0^1 (\pi^2(j+1)^2)^2 \sin(\pi(j+1)x)^2\, dx = \mathcal{O}(j^4), \quad j = 0, \cdots, r-1. \tag{4.31}$$

Similarly, $S_r^{-1}$ is diagonal and has entries

$$\left(\int_0^1 (\pi(j+1))^2 \cos(\pi(j+1)x)^2\, dx\right)^{-1} = \mathcal{O}(j^{-2}), \quad j = 0, \cdots, r-1. \tag{4.32}$$

Hence $\|H_r\|_2 = \mathcal{O}(r^4)$ and $\|S_r\|_2 = \mathcal{O}(1)$, so the first factor on the right hand side of Equation (4.20) scales like $\mathcal{O}(r^2)$.

## 4.10 An Error Equation for the POD projection

Perhaps the best known property of the POD projection is, for a basis derived as orthogonal under the $L^2$ norm (see [69]),

$$\sum_{n=0}^{R-1} \left\| \vec{u}_n - \sum_{j=0}^{r-1} \langle u_n, \phi_j \rangle \, \vec{\varphi}_j \right\|^2 = \sum_{j=r}^{R-1} \sigma_j^2. \tag{4.33}$$

However, for my purposes, I need estimates in multiple norms. POD-based projections possess the fortunate property that exact error estimates are known, which I derive below.

This lemma is similar to theorems proven in [104] and [59].

**Theorem 4.4.** *The POD projection error in the s norm satisfies*

$$\sum_{n=0}^{R-1} \left| \vec{u}_n - \sum_{j=0}^{r-1} \langle u_n, \phi_j \rangle \, \vec{\varphi}_j \right|_s^2 = \sum_{j=r}^{R-1} |\vec{\varphi}_j|_s^2 \, \sigma_j^2. \tag{4.34}$$

*Proof.* Let $\nu_j$ be a left singular vector of $L^T Y$ with corresponding singular value $\sigma_j$. Therefore $\nu_j$ is an eigenvector of $L^T Y (L^T Y)^T$ with eigenvalue $\sigma_j^2$. The singular vector $\nu_j$ is related to the POD vector $\vec{\varphi}_j$ by $L^T \vec{\varphi}_j = \nu_j$. Hence

$$(L^T Y)(L^T Y)^T u_j = \sigma_j^2 \nu_j \Longrightarrow (L^T Y)(L^T Y)^T (L^T \vec{\varphi}_j) = \sigma_j^2 (L^T \vec{\varphi}_j).$$

Therefore, canceling out a $L^T$ from the left,

$$\begin{aligned} \sigma_j^2 \vec{\varphi}_j &= Y Y^T L L^T \vec{\varphi}_j \\ &= Y (Y^T M \vec{\varphi}_j) \\ &= Y \left[ \langle \vec{u}_0, \vec{\varphi}_j \rangle, \ \langle \vec{u}_1, \vec{\varphi}_j \rangle, \ \cdots, \ \langle \vec{u}_{R-1}, \vec{\varphi}_j \rangle \right]^T \\ &= \sum_{n=0}^{R-1} \langle \vec{u}_n, \vec{\varphi}_j \rangle \, \vec{u}_n. \end{aligned} \tag{4.35}$$

Therefore, using Equation 4.33, I obtain

$$\sum_{n=0}^{R-1}\left|\vec{u}_n-\sum_{j=0}^{r-1}\langle u_n,\phi_j\rangle\,\vec{\varphi}_j\right|_s^2=\sum_{n=0}^{R-1}\left|\sum_{j=r}^{R-1}\langle u_n,\phi_j\rangle\,\vec{\varphi}_j\right|_s^2 \tag{4.36}$$

$$=\sum_{n=0}^{R-1}\left\langle\sum_{j=r}^{R-1}\langle u_n,\phi_j\rangle\,\vec{\varphi}_j,\sum_{k=r}^{R-1}\langle u_n,\phi_k\rangle\,\phi_k\right\rangle_s \tag{4.37}$$

$$=\sum_{n=0}^{R-1}\sum_{j=r}^{R-1}\sum_{k=r}^{R-1}\langle u_n,\phi_j\rangle\,\langle u_n,\phi_k\rangle\,\langle\vec{\varphi}_j,\phi_k\rangle_s \tag{4.38}$$

$$=\sum_{j=r}^{R-1}\sum_{k=r}^{R-1}\left\langle\sum_{n=0}^{R-1}\langle u_n,\phi_j\rangle\,\vec{u}_n,\phi_k\right\rangle\langle\vec{\varphi}_j,\phi_k\rangle_s \tag{4.39}$$

$$=\sum_{j=r}^{R-1}\sum_{k=r}^{R-1}\left\langle\sigma_j^2\vec{\varphi}_j,\phi_k\right\rangle\langle\vec{\varphi}_j,\phi_k\rangle_s \tag{4.40}$$

$$=\sum_{j=r}^{R-1}\sigma_j^2\,\langle\vec{\varphi}_j,\vec{\varphi}_j\rangle_s\,. \tag{4.41}$$

□

**Lemma 4.5.** *A similar result to Theorem 4.4 holds in the case where the solution $\vec{u}_n$ is approximated with a centered trajectory, or*

$$\vec{u}_n\approx\vec{u}_s+\sum_{j=0}^{r-1}\langle\vec{\varphi}_j,\vec{u}_n\rangle\,\vec{\varphi}_j \tag{4.42}$$

*such that*

$$\sum_{n=0}^{R-1}\left|\vec{u}_n-\vec{u}_s-\sum_{j=0}^{r-1}\langle\vec{\varphi}_j,\vec{u}_n-\vec{u}_s\rangle\,\vec{\varphi}_j\right|_s^2=\sum_{j=r}^{R-1}|\varphi|_s^2\,\sigma_j^2. \tag{4.43}$$

*Proof.* This result holds by applying Theorem 4.4 to the snapshots $\vec{u}_n-\vec{u}_s$. □

**Remark 4.6.** *It is worth noting here that this is a practical formula: As I use the method of snapshots to calculate the POD basis, I always have every (that is, one vector per snapshot) POD vector available. Therefore the offline cost of calculating these estimates is a fairly modest ($\mathcal{O}(NR)$, where $R$ is the number of snapshots and $N$ is the number of finite element basis functions).*

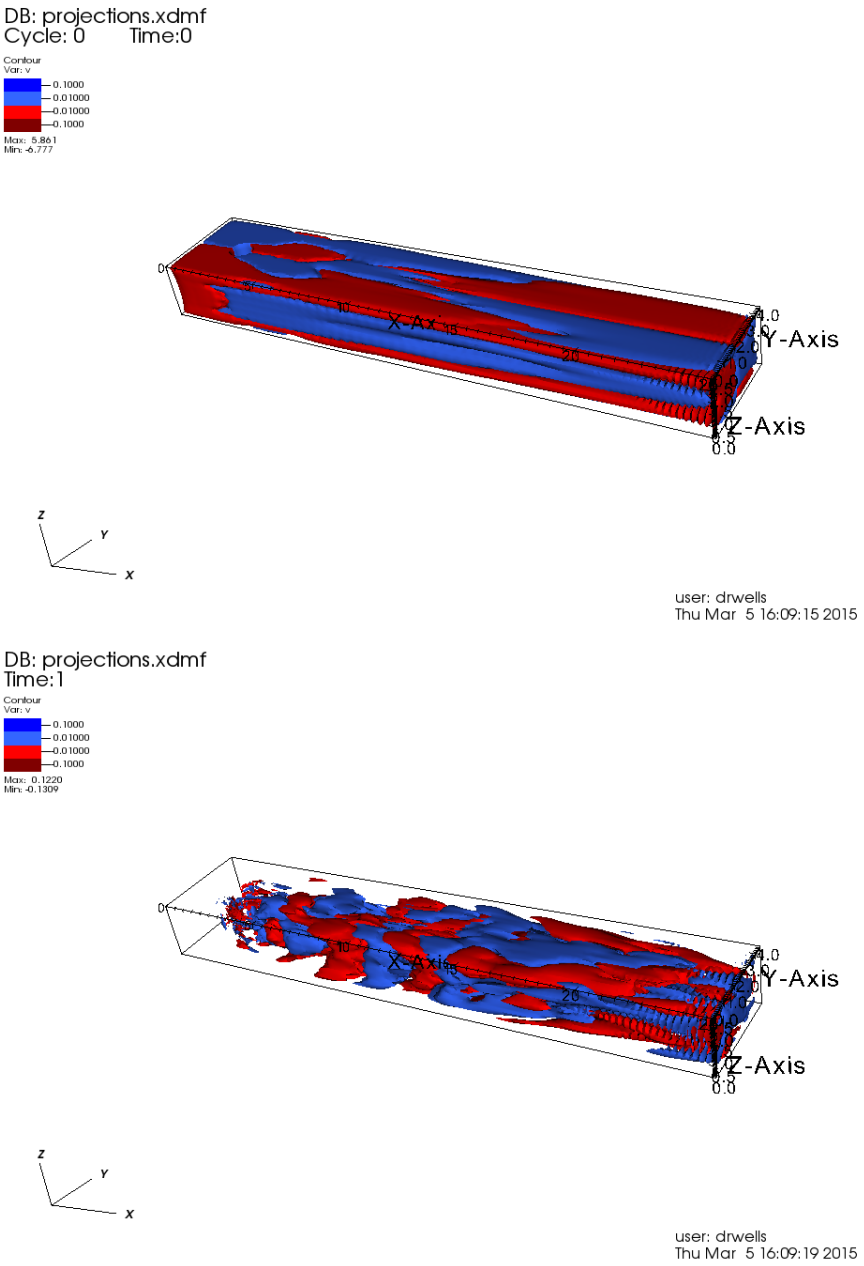


Figure 4.4: Plots of isosurfaces of the $y$-velocity of the mean vector (top) and its projection (bottom) onto the first ten POD vectors. The projection does not perform well. In particular, the inflow (and the flow immediately around the cylinder) is largely time-independent so the POD vectors calculated from centered data cannot resolve it well.

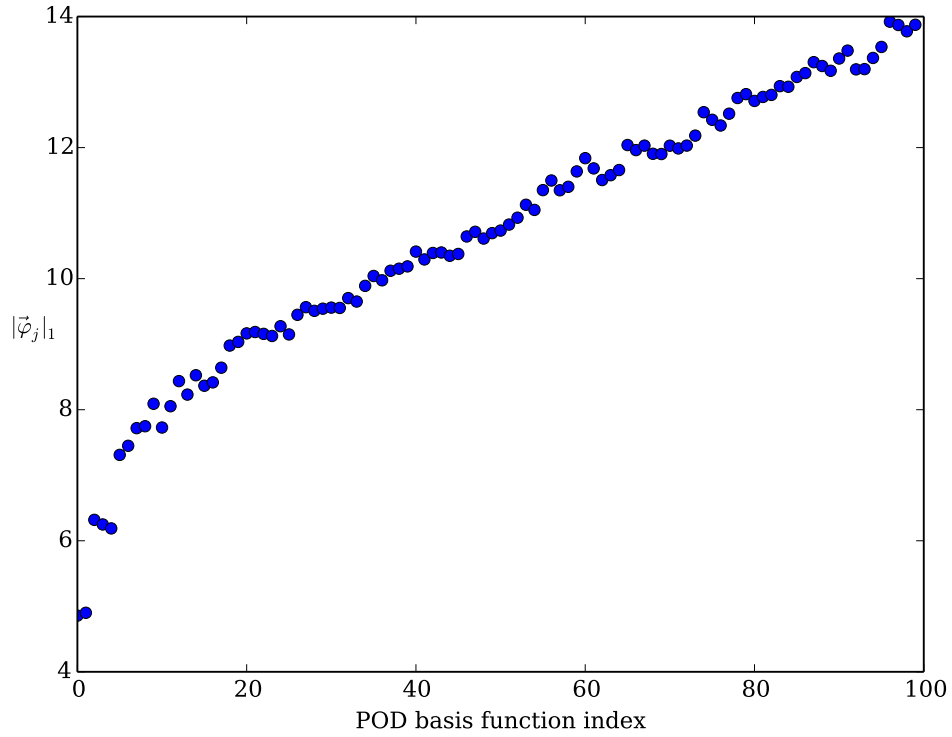


Figure 4.5: Values of the $H^1$ seminorm for the first hundred POD vectors for Experiment 3.2. The growth is linear, which matches the prediction in Equation (4.16).

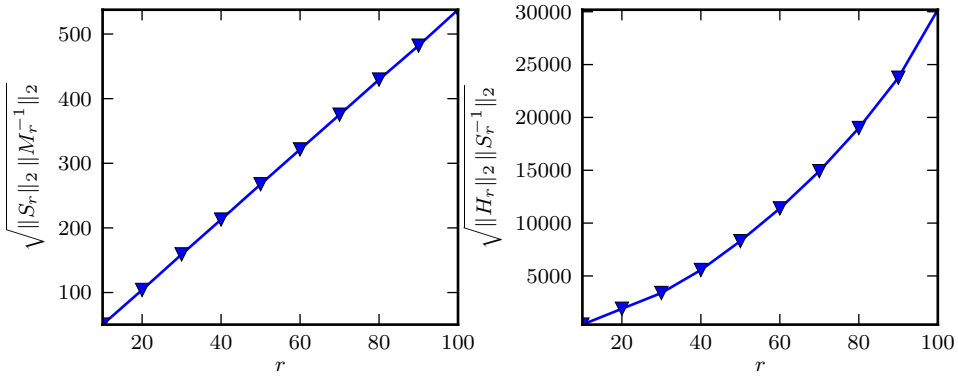


Figure 4.6: Illustration of the scaling result in Lemma 4.2 for the traveling wave problem described in [47]. The first inverse estimate scales like $\mathcal{O}(r)$ and the second scales like $\mathcal{O}(r^2)$.

# Chapter 5

# An SUPG-ROM

## 5.1 Overview

In this chapter I consider the convection-diffusion-reaction equation

$$\begin{aligned} u_t - \varepsilon \Delta u + \vec{b} \cdot \nabla u + cu &= f \text{ in } \Omega \times [0,T] \\ u(0) &= u_0 \\ u|_{\partial\Omega} &= 0, \end{aligned} \tag{5.1}$$

where $\Omega$ is the domain, $\partial\Omega$ is the boundary, $\varepsilon$ is the diffusion coefficient, $\vec{b}$ is the underlying convection field, and $c$ is the reaction coefficient. Typically, $\|\vec{b}\| \gg \varepsilon$ and $\|\vec{b}\| \gg c$. I propose a stabilization method to deal with the problem of numerical oscillations of the G-ROM and I prove error estimates with techniques similar to those used in [99].

The material in this chapter is based on [47]. The rest of this chapter is organized as follows: I provide an overview of the convection-diffusion-reaction problem, discuss some of its numerical stabilization schemes, derive two possible stabilization parameters for a new SUPG-ROM, and illustrate their efficacy with two numerical experiments.

## 5.2 The Convection-Diffusion-Reaction Problem

The principal numerical issue with convection-dominated problems is the presence of fine layers which cannot be resolved by the underlying mesh. These fine layers, typically of width $\sqrt{\varepsilon}$ for internal layers and $\varepsilon$ for boundary layers (see [99]), tend to cause spurious oscillations in standard Galerkin approximations. To the best of my knowledge this issue was perhaps first noticed in [54], where the authors stabilized a finite difference method with *upwinding*: that is, they added extra diffusion in the direction of the convection field and not in the orthogonal (crosswind) direction. Other authors have proposed many similar methods; see [31] for comparisons of various methods, [19] for an extension to the NSEs, and [99] for a recent summary of the field.

The question of which method is "best" (that is, comes close to resolving the boundary layers without excessive smearing or oscillations) is still open. The addition of an empirical

basis (i.e., the POD-ROM) poses significant challenges to several of the most common techniques: methods such as adaptive mesh refinement or Shishkin meshes no longer have clear implementations with global, rather than local, basis functions.

One of the most popular stabilized finite element methods is the Streamline-Upwind Petrov-Galerkin (SUPG) method proposed in [19, 54]. Solutions computed with this method usually possess steep layers, but also exhibit spurious oscillations near the layers. The SUPG method contains a stabilization parameter whose asymptotic value for steady-state problems is well-known from finite element error analysis (e.g., [99]); the parameter's value depends on the individual cell width. The situation is unclear for time-dependent problems: optimal error estimates may only be derived (to the best of my knowledge) for stabilization parameters dependent on the time step. Estimates may also be proven, in simplified situations, for parameters only dependent on the mesh width; see [64] for details. In practice, the stabilization parameter should not depend on the time step; driving the time step to zero does not reduce the oscillations present, but decreasing the cell width does.

This section examines SUPG stabilized ROMs, or SUPG-ROM for short. To the best of my knowledge, the SUPG-ROM was first used in [68] and later on in [15]; both papers utilized the the NSEs. I compare multiple SUPG-ROMs to the unstabilized G-ROM as well as the relevant problem's snapshots.

The principal concern (and the topic of much analysis) in the SUPG method is the question of what stabilization parameter to choose. I do not consider the time step when choosing the stabilization parameter for the reasons listed above. A ROM based on finite element data has two parts to its spatial resolution; it relies on both the spatial resolution of the finite element space and the number of POD basis functions used in the ROM. As the POD basis functions form a subspace of the finite element space, one recovers all scales of motion (at least in theory; in computations the higher index POD basis functions may be inaccurately calculated). However, the use of a small number of POD basis functions is analogous to a finite element model with a coarse mesh: only large scale features can be captured without a higher rank model. One of the principal questions studied in this section is

| Should the SUPG-ROM stabilization parameter depend on the spatial resolution of the underlying finite element space, or on the number of POD basis functions present? |
|---|

I address this question by means of numerical analysis. To the best of my knowledge, the use of numerical analysis to propose a SUPG-ROM stabilization parameter is new. The literature contains results where the stabilization parameter from the finite element situation was used, like in [68], or where an optimization problem was solved, as in [15]. The authors did not provide motivation for these approaches with numerical analysis, though the resulting ROMs were effective. In my opinion, it is important to motivate the choice of stabilization parameter with numerical analysis, as such advice will be valid in a wide range of settings, such as different values for the diffusion or convection, and not require problem or application specific knowledge.

The weak form of (5.1) (see Part 3, Chapter 4, Section 3 in [99]) is: find $u : [0, T] \to H_0^1(\Omega)$

such that

$$\left(\frac{\partial u}{\partial t}, v\right) + (\varepsilon \nabla u, \nabla v) + (\vec{b} \cdot \nabla u, v) + (cu, v) = (f, v) \tag{5.2}$$

for all $v \in H_0^1(\Omega)$. Nearly all literature (see the discussion in Part 3, Chapters 3 and 4, of [99], as well as citations therein) on numerical solutions to (5.2) relies on the assumption that there is a constant $c_0$ such that

$$0 < c_0 \leq \left(c - \frac{1}{2}\nabla \cdot \vec{b}\right)(t, \vec{x}) \tag{5.3}$$

for all $t \in [0, T]$ and $\vec{x} \in \Omega$. Another common assumption (that, in our case, applies to every considered example) is that $\vec{b}$, $c$, and $\nabla \cdot \vec{b}$ are in $L^2([0, T]; L^\infty)$.

The condition (5.3) may seem surprising given that the usual context of these problems involves $\|c\| \ll \|b\|$. However, for most applications, $c > 0$ as $c$ represents a reaction coefficient.

**Lemma 5.1.** *Any equation of the form* (5.1) *may be rewritten by a change of variable so that Inequality* (5.3) *holds.*

*Proof.* This result follows from the change of coordinates

$$\begin{aligned} \hat{x} &= \lambda x \\ \hat{y} &= \lambda y. \end{aligned} \tag{5.4}$$

If I rewrite all of the spatial derivatives in terms of transformed coordinates, then I obtain

$$\begin{aligned} \Delta u &= \frac{\partial^2}{\partial x^2} u + \frac{\partial^2}{\partial y^2} u \\ &= \frac{1}{\lambda^2} \frac{\partial^2}{\partial \hat{x}^2} u + \frac{1}{\lambda^2} \frac{\partial^2}{\partial \hat{y}^2} u \\ &= \frac{1}{\lambda^2} \hat{\Delta} u \end{aligned} \tag{5.5}$$

and, similarly,

$$\vec{b} \cdot \nabla u = \frac{1}{\lambda} \vec{b} \cdot \hat{\nabla} u. \tag{5.6}$$

Therefore I rewrite Equation (5.1) as

$$u_t - \frac{\varepsilon}{\lambda^2} \hat{\Delta} u + \frac{1}{\lambda} \vec{b} \cdot \hat{\nabla} u + cu = f. \tag{5.7}$$

Thus, I obtain a "new" convection field weighted by $1/\lambda$, so the coercivity constraint for the transformed equation is

$$0 < c_0 \leq c - \frac{1}{2\lambda} \hat{\nabla} \cdot \vec{b}. \tag{5.8}$$

Therefore, for $c > 0$ (which is the physical case) the coercivity condition may be enforced for any $\vec{b}$ by sufficiently large choice of $\lambda$. □

**Remark 5.2.** *The point of Lemma 5.1 is to prove that one may, for* any $\varepsilon$, $\vec{b}$, *and* $c$, *modify the problem so that it satisfies Inequality* (5.3). *All of the test problems I consider satisfy this constraint. Note, however, that the change of coordinates may make the resulting CDR problem much harder due to the substitution*

$$\varepsilon \to \frac{\varepsilon}{\lambda^2} \tag{5.9}$$

*when* $\lambda > 1$.

## 5.3 Finite Element Discretization

### 5.3.1 Spatial Discretization

As in [47], I only consider problems with sharp layers (either internal or boundary) that require stabilization in the finite element space. Rather than mesh refinement or other stabilization schemes, I use a Petrov-Galerkin stabilization scheme. "Petrov-Galerkin" refers to a choice of test functions (which I notate by $w$) that are not identical (though they may be in the same space) to the chosen basis functions (which I notate by $v$) for a spatial discretization. Here, as is common with convection-diffusion-reaction problems (e.g., see [99]), I choose

$$w = v + h\vec{b} \cdot \nabla v \tag{5.10}$$

where $v$ is a finite element basis function and $h$ is the average cell width. This choice corresponds to adding additional diffusion in areas where the gradient term is large in the direction of the convection field. The weak form of Equation (5.1) is

$$\int_\Omega w u_t + \varepsilon \nabla v \cdot \nabla u + w(\vec{b} \cdot \nabla u) + cwu \, d\vec{x} + \varepsilon \sum_{T \in \mathcal{T}_h} \int_T h \nabla(\vec{b} \cdot \nabla v) \cdot \nabla u \, d\vec{x} = \int_\Omega w f \, d\vec{x}. \tag{5.11}$$

Integration by parts results in the summation of second derivatives of test functions, so (as the test functions are not assumed to have continuous first derivatives) the integral must be computed over each cell $T$ in the triangulation $\mathcal{T}_h$.

### 5.3.2 Norms and Inner Products

The choice of weak form motivates the definition of both the SD (streamline diffusion) bilinear form

$$a(u,v)_{SD} = \varepsilon(\nabla u, \nabla v) + (\vec{b} \cdot \nabla u, v) + (cu, v) + \sum_{T \in \mathcal{T}_h} \delta(-\varepsilon \Delta u + \vec{b} \cdot \nabla u + cu, \vec{b} \cdot \nabla v)_T \tag{5.12}$$

and the $SD$ norm

$$|||u|||_{SD} = \sqrt{\varepsilon \|\nabla u\|^2 + \delta \|\vec{b} \cdot \nabla u\|^2 + c_0 \|u\|^2}. \tag{5.13}$$

### 5.3.3 Inverse Estimates

I assume that the underlying triangulation is quasi-uniform; put another way, all cells have a diameter of $\mathcal{O}(h)$. Therefore all finite element functions $v_h$ satisfy the per-cell inverse estimate

$$\|v_h\|_{m,K} \leq \mu_{inv} h^{l-m} \|v_j\|_{l,K} \tag{5.14}$$

for $0 \leq l \leq m$. The norm $\|\cdot\|_{m,K}$ refers to the norm over the $K$th cell. Values for the inverse constant, $\mu_{inv}$, may be found in [51]. These constants are typically $\mathcal{O}(1)$. For uniform triangulations, the summation of (5.14) over all cells yields

$$\left( \sum_{T \in \mathcal{T}_h} \|v_h\|_{m,K}^2 \right) \leq C h^{l-m} \sqrt{\sum_{T \in \mathcal{T}_h} \|v_h\|_{l,K}^2}. \tag{5.15}$$

### 5.3.4 Error Estimates

Most finite element error analysis for Equation (5.1) provides estimates in the energy norm or appropriately stabilized norms resulting from the SUPG discretization; see [64, Thm. 5.3, Thm. 5.4] for results of this form. Typical error estimates for this problem resemble

$$\sqrt{\Delta t} \left( \sum_{i=0}^{N-1} \|\vec{u}_i - (\vec{u}_h)_i\| + \sqrt{\varepsilon} \sum_{i=0}^{d-1} \|\nabla(\vec{u}_i - (\vec{u}_h)_i)\| + \sqrt{\delta_h} \sum_{i=0}^{d-1} \|\vec{b} \cdot \nabla(\vec{u}_i - (\vec{u}_h)_i)\| \right) \leq C h^{m+1/2} + (\Delta t)^k, \tag{5.16}$$

where $k$ is the order of the temporal discretization.

## 5.4 Numerical Analysis of the SUPG-ROM

### 5.4.1 Coercivity of the SUPG-ROM

As the SUPG-ROM is a Petrov-Galerkin method, I check that the bilinear form is coercive in $X_r$ to guarantee that the problem is well-posed. As in the finite element case, the coercivity requirement will impose some restrictions on the stabilization parameter $\delta$.

**Lemma 5.3** (Coercivity of the bilinear form in the SUPG-ROM)**.** *Let either*

$$0 \leq \delta \leq \frac{1}{2} \min \left( \left\{ \frac{c_0}{\|c\|_\infty^2}, \frac{h^2}{\varepsilon \mu_{\text{inv}}^2} \right\} \right) \tag{5.17}$$

*or*

$$0 \leq \delta \leq \frac{1}{2} \min \left( \left\{ \frac{c_0}{\|c\|_\infty^2}, \frac{1}{\varepsilon \|H_r\|_2 \|S_r^{-1}\|_2} \right\} \right). \tag{5.18}$$

*Then*

$$a_{\text{SUPG},r}(v_r, v_r) \geq \frac{1}{2} |||v_r|||_{\text{SUPG},r}^2, \; \forall v_r \in X_r, \tag{5.19}$$

*where*

$$|||v_r|||_{\text{SUPG},r} = \sqrt{\varepsilon |v_r|_1^2 + c_0 \|v_r\|^2 + \delta_r \|\vec{b} \cdot \nabla v_r\|^2}. \tag{5.20}$$

*Proof.* As for any $u \in X$

$$\begin{aligned}\left(u, \vec{b} \cdot \nabla u\right) &= \int_\Omega u \left(\vec{b} \cdot \nabla u\right) \, d\vec{x} \\ &= -\int_\Omega u \nabla \cdot \left(\vec{b} u\right) \, d\vec{x} \\ &= -\left(\left(\nabla \cdot \vec{b}\right) u, u\right) - \left(\vec{b} \cdot \nabla u, u\right)\end{aligned} \tag{5.21}$$

I obtain

$$\left(u, \vec{b} \cdot \nabla u\right) = -\frac{1}{2}\left(\left(\nabla \cdot \vec{b}\right) u, u\right). \tag{5.22}$$

Therefore

$$\left(\vec{b} \cdot \nabla u + cu, u\right) = \left(\left(c - \frac{1}{2}\nabla \cdot \vec{b}\right) u, u\right). \tag{5.23}$$

Therefore

$$\begin{aligned}a(u,u)_{\mathrm{SUPG},r} &= \varepsilon |u|_1^2 + \left(\vec{b} \cdot \nabla u + cu, u\right) + \delta_r \left(\vec{b} \cdot \nabla u + cu, \vec{b} \cdot \nabla u\right) - \delta_r \sum_{T \in \mathcal{T}_h} \left(\varepsilon \Delta u, u\right) \\ &= \varepsilon |u|_1^2 + \left(\left(c - \frac{1}{2}\nabla \cdot \vec{b}\right) u, u\right) + \delta_r \left(\vec{b} \cdot \nabla u + cu, \vec{b} \cdot \nabla u\right) - \delta_r \sum_{T \in \mathcal{T}_h} \left(\varepsilon \Delta u, u\right).\end{aligned} \tag{5.24}$$

By Inequality (5.3) I bound the second term below by

$$\left(\left(c - \frac{1}{2}\nabla \cdot \vec{b}\right) u, u\right) \geq c_0 \|u\|^2. \tag{5.25}$$

I bound the third term below by

$$\begin{aligned}\delta_r(\vec{b} \cdot \nabla u + cu, \vec{b} \cdot \nabla u) &= \delta_r \left(cu, \vec{b} \cdot \nabla u\right) + \delta_r \left\|\vec{b} \cdot \nabla u\right\|^2 \\ &\geq \delta_r \left\|\vec{b} \cdot \nabla u\right\|^2 - \delta_r \frac{k_1 \|c\|_\infty^2 \|u\|^2}{2} - \delta_r \frac{\left\|\vec{b} \cdot \nabla u\right\|^2}{2k_1}\end{aligned} \tag{5.26}$$

by Young's inequality, where $k_1 > 0$. Similarly, I bound the fourth term by

$$\begin{aligned}-\delta_r \sum_{T \in \mathcal{T}_h} \left(\varepsilon \Delta u, \vec{b} \cdot \nabla u\right) &= -\delta_r \left(\varepsilon \sum_{T \in \mathcal{T}_h} \Delta u_T, b \cdot \nabla u\right) \\ &\geq -\delta_r \frac{\varepsilon^2 k_2 \left\|\sum_{T \in \mathcal{T}_h} \Delta u_T\right\|^2}{2} - \delta_r \frac{\left\|\vec{b} \cdot \nabla u\right\|^2}{2k_2} \\ &= -\delta_r \frac{\varepsilon^2 k_2 \sum_{T \in \mathcal{T}_h} \|\Delta u\|_T^2}{2} - \delta_r \frac{\left\|\vec{b} \cdot \nabla u\right\|^2}{2k_2}\end{aligned} \tag{5.27}$$

by Young's inequality, where $k_2 > 0$ and $u_T$ is $u$ limited to the interior of the $T$th cell. Combining Equations (5.25), (5.26), and (5.27) with (5.24) yields

$$\begin{aligned}a(u,u)_{\mathrm{SUPG},r} \geq \varepsilon |u|_1^2 + c_0 \|u\|^2 + \delta_r \left\|\vec{b} \cdot \nabla u\right\|^2 - \delta_r \frac{k_1 \|c\|_\infty^2 \|u\|^2}{2} - \delta_r \frac{\left\|\vec{b} \cdot \nabla u\right\|}{2k_1} \\ - \delta_r \frac{\varepsilon^2 k_2 \sum_{T \in \mathcal{T}_h} \|\Delta u\|_T^2}{2} - \delta_r \frac{\left\|\vec{b} \cdot \nabla u\right\|^2}{2k_2}.\end{aligned} \tag{5.28}$$

Finally, I choose $k_1 = k_2 = 2$ in Estimate (5.28) and obtain

$$a(u,u)_{\text{SUPG},r} \geq \varepsilon \left|u\right|_1^2 + c_0 \left\|u\right\|^2 + \frac{\delta_r}{2}\left\|\vec{b}\cdot\nabla u\right\|^2 - \delta_r \left\|c\right\|_\infty^2 \left\|u\right\|^2 - \delta_r\varepsilon^2 \sum_{T\in\mathcal{T}_h} \left\|\Delta u\right\|_T^2 . \tag{5.29}$$

There are presently two options for estimating the Laplacian term: either the finite element inverse estimate (5.15) or the POD inverse estimate (4.20), specifically

$$-\delta_r\varepsilon^2 \sum_{T\in\mathcal{T}_h} \left\|\Delta u\right\|_T^2 \geq -\|H_r\|_2\|S_r^{-1}\|_2 \left|u\right|_1^2 \quad \text{or} \quad -\delta_r\varepsilon^2 \sum_{T\in\mathcal{T}_h} \left\|\Delta u\right\|_T^2 \geq -\mu_{inv}^2 h^{-2} \left|u\right|_1^2 . \tag{5.30}$$

These correspond to the second arguments in Inequalities (5.17) and (5.18). Applying either estimate yields one of the two constraints on the stabilization parameter $\delta_r$ for guaranteeing coercivity. In particular, for the first case, if

$$\delta_r \leq \frac{1}{2}\min\left(\left\{\frac{c_0}{\|c\|_\infty^2}, \frac{h^2}{\varepsilon\mu_{inv}^2}\right\}\right), \tag{5.31}$$

then

$$-\delta_r\|c\|_\infty^2\|u\|^2 \geq -\frac{1}{2}c_0\|u\|^2 \tag{5.32}$$

and, combining with Equation (5.14),

$$\begin{aligned} -\delta_r\varepsilon^2 \sum_{T\in\mathcal{T}_h} \left\|\Delta u\right\|_T^2 &\geq -\left(\frac{1}{2}\frac{h^2}{\varepsilon\mu_{inv}^2}\right)\varepsilon^2 \sum_{T\in\mathcal{T}_h} \left\|\Delta u\right\|_T^2 \\ &\geq -\left(\frac{1}{2}\frac{h^2}{\varepsilon\mu_{inv}^2}\right)\varepsilon^2\mu_{inv}^2 \left\|u\right\|_1^2 \\ &= -\frac{1}{2}\varepsilon|u|_1^2. \end{aligned} \tag{5.33}$$

Similarly, if

$$\delta_r \leq \frac{1}{2}\min\left(\left\{\frac{c_0}{\|c\|_\infty^2}, \frac{1}{\varepsilon\|H_r\|_2\|S_r^{-1}\|_2}\right\}\right), \tag{5.34}$$

then Inequality (5.32) still holds. I bound the Laplacian term by

$$\begin{aligned} -\delta_r\varepsilon^2 \sum_{T\in\mathcal{T}_h} \left\|\Delta u\right\|_T^2 &\geq -\left(\frac{1}{2}\frac{1}{\varepsilon\|H_r\|_2\|S_r\|_2}\right)\varepsilon^2 \sum_{T\in\mathcal{T}_h} \left\|\Delta u\right\|_T^2 \\ &\geq -\left(\frac{1}{2}\frac{1}{\varepsilon\|H_r\|_2\|S_r\|_2}\right)\varepsilon^2 \left(\|H_r\|_2\|S_r^{-1}\|_2|u|_1^2\right) \\ &= -\frac{1}{2}\varepsilon|u|_1^2. \end{aligned} \tag{5.35}$$

Therefore, in both cases,

$$\begin{aligned} a(u,u)_{\text{SUPG},r} &\geq \frac{1}{2}\varepsilon|u|_1^2 + \frac{1}{2}c_0\|u\|^2 + \frac{1}{2}\delta_r\|\vec{b}\cdot\nabla u\|^2 \\ &= \frac{1}{2}|||u|||_{\text{SUPG},r}^2. \end{aligned} \tag{5.36}$$

□

### 5.4.2 Error Bound for the SUPG-ROM

Let $u_r^n + u_s$ be the reduced order approximation at time step $n$ to the original snapshot $u^n$. Let $P_r(u)$ be the projection operator from $X^h$ to $X^r$, or

$$P_r(u) = \bar{u}_h + \sum_{i=0}^{r-1}(u - u_s, \varphi_i)_X \varphi_i \tag{5.37}$$

where $(\cdot,\cdot)_X$ is the inner product under which the POD basis is orthogonal. I use the decomposition

$$u_h^n - (\bar{u}_h + u_r^n) = (u_h^n - P_r(u_h^n)) + (P_r(u_h^n) - (\bar{u}_h + u_r^n)) \tag{5.38}$$

to derive a stabilization parameter for the ROM, where I notate the error terms by

$$\eta = u - P_r(u) \text{ and } \phi_r = P_r(u) - u_r, \tag{5.39}$$

where

$$\|\eta\| \leq \|u - u_h\| + \|P_r(u) - u_h\| \tag{5.40}$$

and

$$|||u - u_r|||_{SD} \leq |||\eta|||_{SD} + |||\phi_r|||_{SD} \tag{5.41}$$

by the triangle inequality. I bound the $|||\phi_r|||_{SD}$ term with the coercivity property as

$$\frac{1}{2}|||\phi_r|||_{SD}^2 \leq a(\phi_r, \phi_r)_{SD} = a(\eta, \phi_r)_{SD}, \tag{5.42}$$

which holds as $a(u, \phi_r)_{SD} = a(u_r, \phi_r)_{SD}$ and $\phi_r \in X$. The inequality

$$\varepsilon(\nabla\eta, \nabla\phi_r) \leq (\varepsilon\|\nabla\eta\|^2)^{1/2}(\varepsilon\|\nabla\phi_r\|^2)^{1/2} \quad \leq \sqrt{\varepsilon}\|\nabla\eta\|||\phi_r|||_{SD} \tag{5.43}$$

holds by the Cauchy-Schwarz inequality. I rewrite the convective and reactive terms as

$$(\vec{b}\cdot\nabla\eta + c\eta, \phi_r) = ((c - \nabla\cdot\vec{b})\eta, \phi_r) - (\eta, \vec{b}\cdot\nabla\phi_r) \tag{5.44}$$

via integration by parts. Applying the Cauchy-Schwarz inequality, I obtain

$$\begin{aligned}\Big((c - \nabla\cdot\vec{b})\eta, \phi_r\Big) &\leq \left(\frac{\Big(\|c\|_\infty + \|\nabla\cdot\vec{b}\|_\infty\Big)^2}{c_0}\|\eta\|^2\right)^{1/2}\Big(c_0\|\phi_r\|^2\Big)^{1/2} \\ &\leq \left(\frac{\Big(\|c\|_\infty + \|\nabla\cdot\vec{b}\|_\infty\Big)^2}{c_0}\|\eta\|^2\right)^{1/2}|||\phi_r|||_{SD}\end{aligned} \tag{5.45}$$

and

$$\begin{aligned}\Big(\eta, \vec{b}\cdot\nabla\phi_r\Big) &\leq (\delta^{-1}\|\eta\|^2)^{1/2}(\delta\|\vec{b}\cdot\nabla\phi_r\|^2)^{1/2} \\ &\leq (\delta^{-1}\|\eta\|^2)^{1/2}|||\phi_r|||_{SD}.\end{aligned} \tag{5.46}$$

I combine Estimates (5.44), (5.45), and (5.46) to bound the convection and reaction terms by

$$\left(\vec{b}\cdot\nabla\eta + c\,\eta, \phi_r\right) \leq \left[\sqrt{\frac{\left(\|c\|_\infty + \|\nabla\cdot\vec{b}\|_\infty\right)^2}{c_0}} + \frac{1}{\sqrt{\delta}}\right] \|\eta\|\|\|\phi_r\|\|_{SD}. \tag{5.47}$$

I rewrite the stabilization term as

$$\begin{aligned}
\delta\sum_{T\in\mathcal{T}_h}(-\varepsilon\Delta\eta, \vec{b}\cdot\nabla\phi_r)_T &= \delta\varepsilon\left(-\sum_{T\in\mathcal{T}_h}\Delta\eta, \vec{b}\cdot\nabla\phi_r\right)\\
&\leq \delta\varepsilon\left\|\sum_{T\in\mathcal{T}_h}\Delta\eta\right\|\left\|\vec{b}\cdot\nabla\phi_r\right\|\\
&= \delta\varepsilon\sqrt{\left(\sum_{T\in\mathcal{T}_h}\Delta\eta, \sum_{T\in\mathcal{T}_h}\Delta\eta\right)}\left\|\vec{b}\cdot\nabla\phi_r\right\|\\
&= \delta\varepsilon\sqrt{\sum_{T\in\mathcal{T}_h}\|\Delta\eta\|_T^2}\left\|\vec{b}\cdot\nabla\phi_r\right\|\\
&= \sqrt{\delta}\varepsilon\sqrt{\sum_{T\in\mathcal{T}_h}\|\Delta\eta\|_T^2}\sqrt{\delta\left\|\vec{b}\cdot\nabla\phi_r\right\|}\\
&\leq \sqrt{\delta}\varepsilon\sqrt{\sum_{T\in\mathcal{T}_h}\|\Delta\eta\|_T^2}|\|\phi_r\|\|_{SD}
\end{aligned} \tag{5.48}$$

and

$$\begin{aligned}
\delta\sum_{T\in\mathcal{T}_h}\left(\vec{b}\cdot\nabla\eta + c\eta, \vec{b}\cdot\nabla\phi_r\right)_T &= \delta\left(\vec{b}\cdot\nabla\eta + c\eta, \vec{b}\cdot\nabla\phi_r\right)\\
&\leq \sqrt{\delta}\left(K\|\nabla\eta\| + \|c\|_\infty\|\eta\|\right)\sqrt{\delta\|\vec{b}\cdot\nabla\eta_r\|}\\
&\leq \sqrt{\delta}\left(K\|\nabla\eta\| + \|c\|_\infty\|\eta\|\right)|\|\phi_r\|\|_{SD},
\end{aligned} \tag{5.49}$$

where $K = 2(\|b_1\|_\infty + \|b_2\|_\infty)$. Let

$$C = 2\max\left(\left\{\sqrt{\frac{\left(\|c\|_\infty + \|\nabla\cdot\vec{b}\|_\infty\right)^2}{c_0}}, 1, K, \|c\|_\infty\right\}\right). \tag{5.50}$$

Combining the pieces above yields

$$\begin{aligned}
|\|\phi_r\|\|_{SD} &\leq C\left[\sqrt{\varepsilon}\|\nabla\eta\| + \left(1+\frac{1}{\sqrt{\delta}}\right)\|\eta\| + \varepsilon\sqrt{\delta}\sqrt{\sum_{T\in\mathcal{T}_h}\|\Delta\eta\|_T^2} + \sqrt{\delta}\left(\|\nabla\eta\| + \|\eta\|\right)\right]\\
&= C\left[\sqrt{\varepsilon}\|\nabla\eta\| + \left(1+\frac{1}{\sqrt{\delta}}\right)\|\eta\| + \sqrt{\delta}\left(\varepsilon\sqrt{\sum_{T\in\mathcal{T}_h}\|\Delta\eta\|_T^2} + \|\nabla\eta\| + \|\eta\|\right)\right].
\end{aligned} \tag{5.51}$$

I combine Estimate (5.51) with estimates for various norms of $\eta$ in Sections 5.4.3 and 5.4.4 which may be combined with estimates for various norms of $\eta$ to calculate optimal values of $\delta$.

### 5.4.3 SUPG-ROM Parameter Scaling: The POD Option

This option uses the assumption that the error caused by truncation of POD basis vectors is much larger than the finite element error (hence the factor of 2). I use (4.10) to bound $\eta$ by:

$$\begin{aligned} \|\eta\| &\leq 2\sqrt{\sum_{j=r}^{R-1}\sigma_j^2} := 2\Lambda_0 \\ \|\nabla\eta\| &\leq 2\sqrt{\sum_{j=r}^{R-1}\sigma_j^2|\varphi_j|_1} := 2\Lambda_1 \\ \sqrt{\sum_{T\in\mathcal{T}_h}\|\delta\eta\|_T^2} &\leq 2\sqrt{\sum_{j=r}^{R-1}\sigma_j^2|\varphi_j|_2} := 2\Lambda_2. \end{aligned} \tag{5.52}$$

The inequalities in (5.52) use the property that the interpolation error at a single snapshot is bounded (not very sharply) by the sum of the interpolation errors described in Section 4.10. This choice leads to an optimization problem for $\delta$

$$F(\delta) = \sqrt{\varepsilon}\Lambda_1 + \left(1 + \frac{1}{\sqrt{\delta}}\right)\Lambda_0 + \sqrt{\delta}(\varepsilon\Lambda_2 + \Lambda_1 + \Lambda_0) \tag{5.53}$$

with solution (see Subsection A.1.1 for the `sage` [107] implementation)

$$\delta = \frac{\Lambda_0}{\Lambda_2\varepsilon + \Lambda_0 + \Lambda_1}. \tag{5.54}$$

### 5.4.4 SUPG-ROM Parameter Scaling: The FE Option

This option relies upon the existence of inverse estimates for bounding the $\|\nabla\eta\|$ and $\sum_{T\in\mathcal{T}_h}\|\Delta\eta\|_T^2$ terms. I use the standard finite element inverse estimate for stabilized flows

$$|\eta|_s \leq Ch^{m+1/2-s} \tag{5.55}$$

and

$$\sum_{n=0}^{R-1}|P_r(u_h) - u_h|_s \leq Ch^{-s}\Lambda_0, \tag{5.56}$$

where $s = 0, 1, 2$. In the case of second derivatives I implicitly use a sum over the mesh cells. I bound $\eta$ as

$$|\eta|_s \leq |u - u_h|_s + |P_r(u) - u_h|_s. \tag{5.57}$$

Using Equations (5.55)-(5.57) yields

$$\begin{aligned} \|\eta\| &\leq C(h^{m+1/2} + \Lambda_0) \\ \|\nabla\eta\| &\leq C\left(h^{m-1/2} + \frac{\Lambda_0}{h}\right) \\ \sum_{T\in\mathcal{T}_h}\|\Delta\eta\|_T &\leq C\left(h^{m-3/2} + \frac{\Lambda_0}{h^2}\right). \end{aligned} \tag{5.58}$$

I substitute Estimate (5.58) into Inequality (5.51) and obtain

$$\begin{aligned} F(\delta) =& \sqrt{\varepsilon}\left(h^{m-1/2}+\frac{\Lambda_0}{h}\right)+\left(1+\frac{1}{\sqrt{\delta}}\right)\left(h^{m+1/2}+\Lambda_0\right) \\ &+\sqrt{\delta}\left(\varepsilon\left(h^{m-3/2}+\frac{\Lambda_0}{h^2}\right)+\left(h^{m-1/2}+\frac{\Lambda_0}{h}\right)+\left(h^{m+1/2}+\Lambda_0\right)\right) \end{aligned} \tag{5.59}$$

with solution (see Subsection (A.1.2) for the `sage` implementation)

$$\delta = \frac{\Lambda_0 h^2 + h^{m+\frac{5}{2}}}{(h^2+\varepsilon+h)\Lambda_0 + \varepsilon h^{m+\frac{1}{2}} + h^{m+\frac{5}{2}} + h^{m+\frac{3}{2}}}. \tag{5.60}$$

As $m \geq 2$, I neglect higher powers of $h$ and assume $\varepsilon \ll h$ to obtain

$$\delta = \frac{\Lambda_0 h^2}{h\Lambda_0} = h, \tag{5.61}$$

which is the standard finite element result.

### 5.4.5 Implementation of the ROM

The implementation of the ROM (particularly the centering trajectory) requires care. To keep things simple I notate the reduced stabilized matrices (that is, the test functions are of the form given by Equation (5.10), where $v$ is a POD basis function, instead of simply the POD basis functions) with overlines (e.g., the stabilized POD mass matrix is $\bar{M}_r$). After integrating by parts the ROM may be written as

$$\bar{M}_r(u_r+u_s)_t = \varepsilon\bar{S}_r(u_r+u_s)+\bar{C}_r(u_r+u_s)+c\bar{M}_r(u_r+u_s)+\bar{f}_r, \tag{5.62}$$

where $\bar{C}_r$ is the matrix formed from the convective term and $\bar{f}_r$ is the forcing vector. As $(u_s)_t = 0$, I rearrange and obtain

$$\begin{aligned} \bar{M}_r(u_r)_t &= (\varepsilon\bar{S}_r+\bar{C}_r+c\bar{M}_r)u_r + \underbrace{(\varepsilon\bar{S}_r+\bar{C}_r+c\bar{M}_r)u_s+\bar{f}_r}_{\text{time-independent terms}} \\ &= (\varepsilon\bar{S}_r+\bar{C}_r+c\bar{M}_r)u_r + F_r, \end{aligned} \tag{5.63}$$

where $F_r$ is a combination of the forcing term $\bar{f}_r$, which may be time-dependent, and the contributions from $u_s$, which are time-independent. I typically use the trapezoid method for time integration of this system to guarantee stability; this choice yields a fully discrete system

$$\bar{M}_r u_r(t_1) = \bar{M}_r u_r(t_0) + \frac{\Delta t}{2}\left(\left(\varepsilon\bar{S}_r+\bar{C}_r+c\bar{M}_r\right)(u_r(t_0)+u_r(t_1))+F(t_0)+F(t_1)\right). \tag{5.64}$$

**Remark 5.4.** *Regardless of the time stepping method, evolving the ROM one time step is always $\mathcal{O}(r^2)$ operations for time-independent forcing functions. In this case, which generalizes to more implicit methods, $\left(\varepsilon\bar{S}_r+\bar{C}_r+c\bar{M}_r\right)$ may be factorized in advance (leading to $\mathcal{O}(r^2)$ evaluation). Any explicit method will require matrix-vector multiplication which is also $\mathcal{O}(r^2)$. Since $\mathcal{O}(r^2)$ is very cheap I do not spend much time worrying about time stepping aside from picking a method with nice stability properties.*

## 5.5 Numerical Investigation of the SUPG-ROM

### 5.5.1 Overview

I use numerical experiments to try and answer the following questions:

> *Question 1:* Does the SUPG-ROM yield more accurate results than the G-ROM?

and

> *Question 2:* Which of the two stabilization parameters of the SUPG-ROM (i.e., (5.54) and (5.61)) yields more accurate results?

I consider two two-dimensional tests to answer Question 1. For the sake of brevity I omit results where the finite element mesh was able to resolve all layers, since this is not common in realistic applications. When the mesh width $h \in [10^{-3}, 10^{-2}]$ and the layer width $\sqrt{\varepsilon}$ are on the same order of magnitude, then the SUPG-ROM and the G-ROM performed similarly in numerical tests.

I present results for three different ROMs. The first is the G-ROM, i.e., the ROM without any stabilization. It is well known that for convection-dominated problems the choice of basis may have a large impact on the stability of the discretization; see the proposals on bubble functions or exponentially fitted functions surveyed in [99].

The other two ROMs considered in this section are the FE-SUPG-ROM and the POD-SUPG-ROM, which use the stability parameters $\delta_r^{FE}$ and $\delta_r^{POD}$ derived in Equation (5.61) and Equation (5.54). Computing $\delta_r^{FE}$ is much easier than computing $\delta_r^{POD}$. However, since the stabilization parameter only needs to be computed *once* (that is, in the offline step), computing $\delta_r^{POD}$ is not nearly as expensive as running the full simulation.

Some of the numerical experiments have exact solutions; these solutions are enforced in the numerical tests by the forcing term $f$. This approach is usually called *the method of manufactured solutions*. Therefore, to compare the ROMs, I calculate a discrete analogue of the $L^1(0, T; L^2(\Omega))$ norm of the error by

$$\frac{T}{N} \sum_{n=0}^{N-1} \|u_n - (u_s + u_{r,n})\| . \tag{5.65}$$

### 5.5.2 Numerical Tests

The rest of this section is organized as follows: in Section 5.5.2 I describe a test case based on a benchmark proposed in [31] for problems with low diffusion, low reaction, and high convection. I examine a second test case in Section 5.5.2 which demonstrates the efficacy of the SUPG-ROM in reducing the noise present in the ROM solution. Finally, I present a discussion of the two test cases in Section 5.5.2.

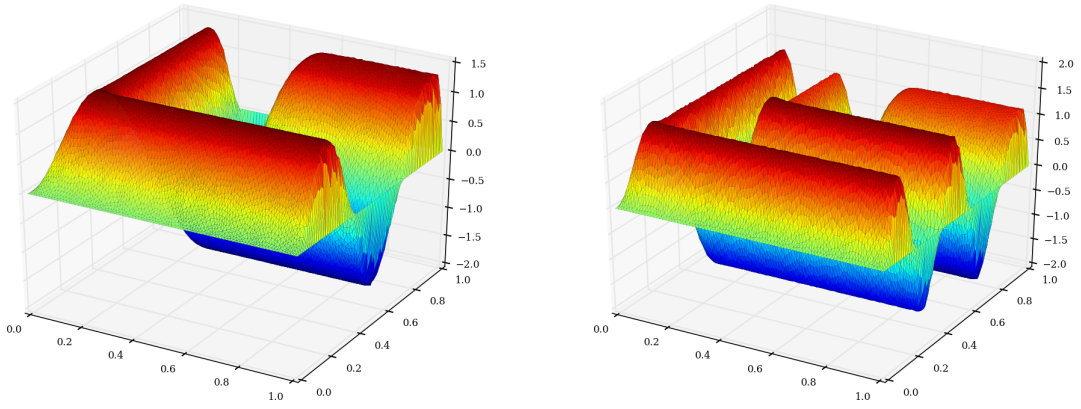

Figure 5.1: The third (left) and fifth (right) POD vectors corresponding to the rising wave problem. Note the similarity to the Fourier basis: the third POD vector has three humps and the fifth POD vector has five.

**A Rising Wave** I modify the first test case used in [31] to fit Equation (5.1) by adding a time derivative to the exact solution. I use $\varepsilon = 10^{-4}$, $c = 10^{-4}$, and $\vec{b} = (\cos(\pi/3), \sin(\pi/3))$.

This problem is convection-dominated, as $\varepsilon \ll \|\vec{b}\|$; without stabilization the solution is polluted by spurious noise caused by the sharp boundary layer along $x = 1$ and $y = 1$ (see Figure 5.2). I stabilize the solution with the Petrov-Galerkin method described in Equation (5.11). I use a quasi-uniform triangulation with triangular elements and piecewise quadratics. The finite element simulation uses $5,126$ elements with $10,137$ total degrees of freedom. I discretize in time with the trapezoid method and use $10,000$ time steps between $t = 0$ and $t = 1$ to generate the $1,001$ snapshots. The POD basis is generated by the SVD algorithm described in Section 4.4.

**A Stationary Hump** This example is nearly identical to the one used in [47]. I use $\varepsilon = 10^{-4}$, $b = (2, 3)$, and $c = 1$ combined with a forcing term derived, by the method of manufactured solutions, from the exact solution

$$u(t, x, y) = 16 \sin(\pi t) x(1 - x) y(1 - y) \left[ \frac{1}{2} + \frac{\operatorname{atan}(200(r^2 - (x - 2r)^2 - (y - 2r)^2))}{\pi} \right], \quad (5.66)$$

where $r = 0.25$. I use a quasi-uniform mesh with piecewise triangular quadratic finite elements. The mesh has $20,525$ degrees of freedom and a mean cell width of 0.0153. I integrate in time with the BDF-3 method and use 1001 time steps between 0 and 1, including the (zero) initial condition. As the exact solution $u$ may be written as the product of a time-dependent function and a space-dependent function, the solution should have exactly one POD vector with a time-dependent coefficient $C \sin(\pi t)$, where $C$ is some constant that corrects the magnitude of the POD vector. However, due to numerical instabilities caused by the very low diffusion and resulting boundary layer of width $\sqrt{200}$, this is not the case. I stabilize the finite element solution with the SUPG method and $\delta = 0.0075$, which is the "general" value specified in [31].

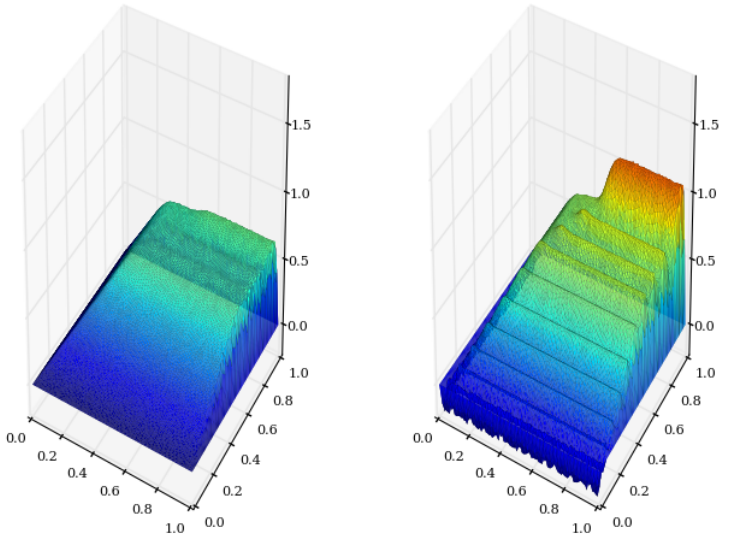


Figure 5.2: Stabilized (left) and unstabilized (right) solutions to (5.1) at $t = 0.6$ with $\delta = 0.007709$. The lack of stabilization causes the higher-frequency POD modes to contain an inaccurate amount of energy.

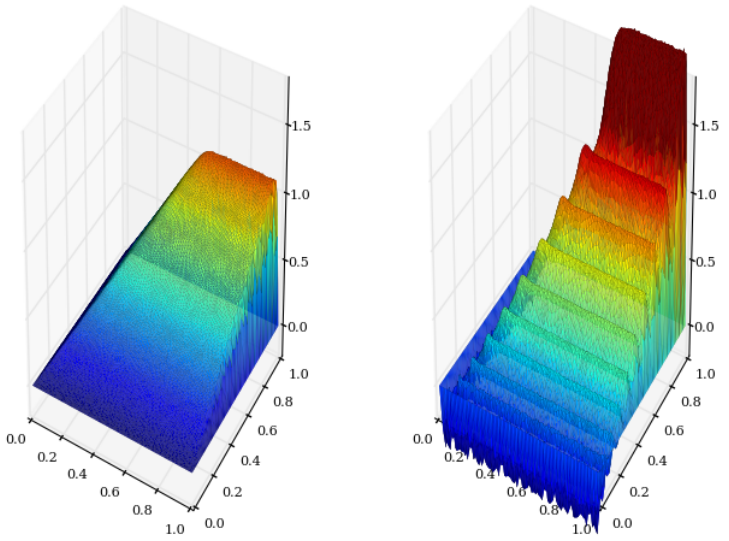


Figure 5.3: Stabilized (left) and unstabilized (right) POD-ROM solutions to (5.1) at $t = 1.0$ with $\delta = 0.007709$. The spurious oscillations have continued to grow in time and the error has increased substantially in the unstabilized G-ROM.

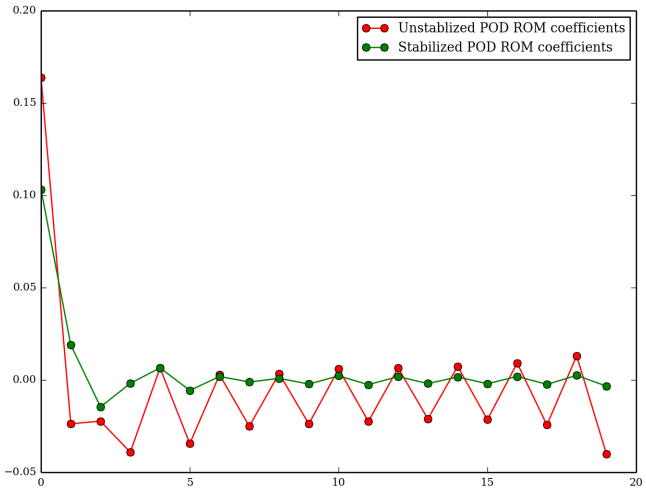


Figure 5.4: Coefficients of POD basis functions at $t = 0.6$.

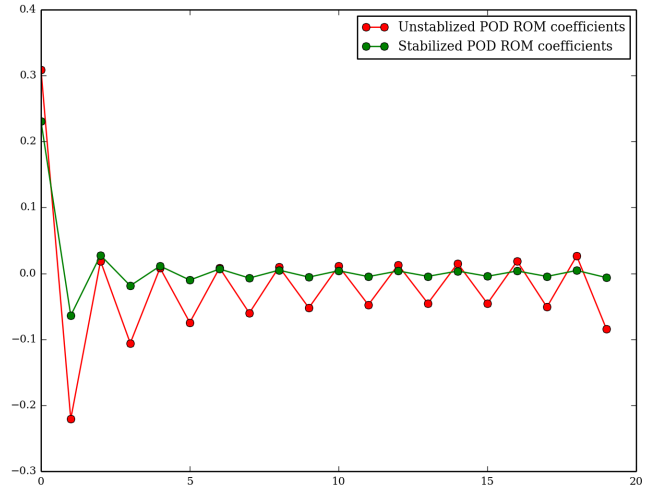


Figure 5.5: Coefficients of POD basis functions at $t = 1.0$.

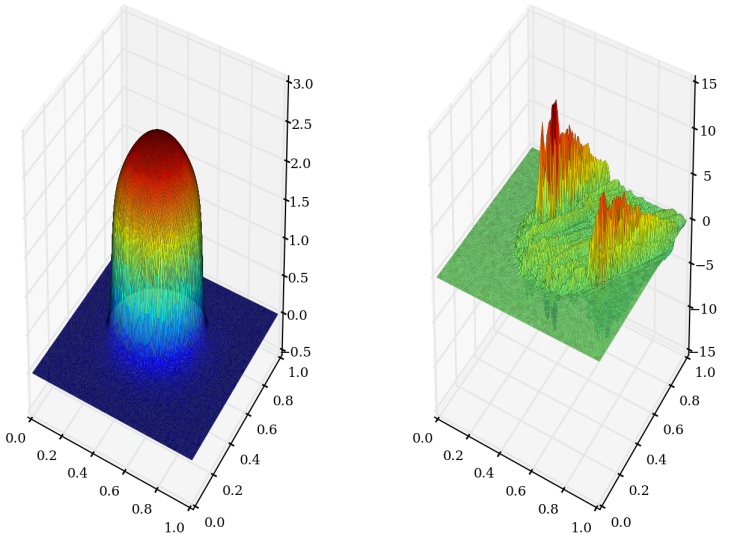


Figure 5.6: The first two POD basis vectors from example (5.66). The first POD vector should, in the absence of discretization errors, fully describe the solution. The associated singular values are $\sigma_0 = 3.543$ and $\sigma_1 = 0.002403$. The first POD vector captures 99.90% of the energy in the system, which implies that a ROM with one degree of freedom should capture the essential behavior of this problem.

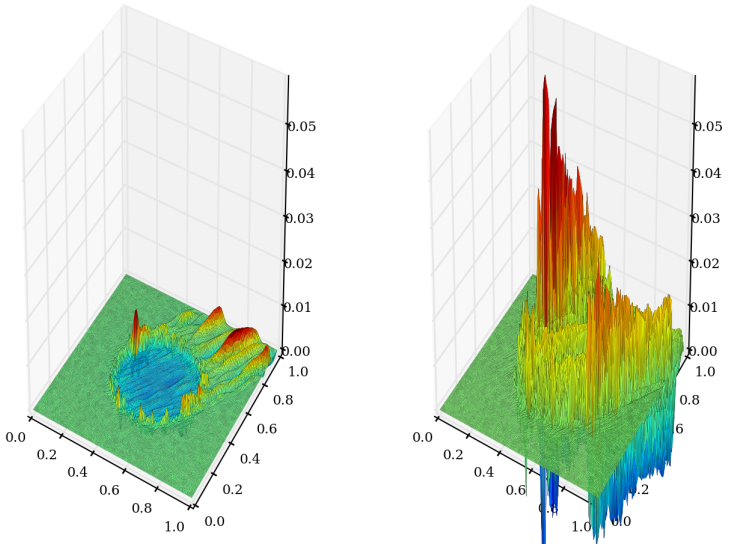


Figure 5.7: Errors in ROM solutions at $t = 0.5$, with the stabilized solution on the left (with $\delta = 0.004939$) and the unstabilized solution on the right. The stabilized model is nearly equal to the DNS solution, while the unstabilized one has picked up noise.

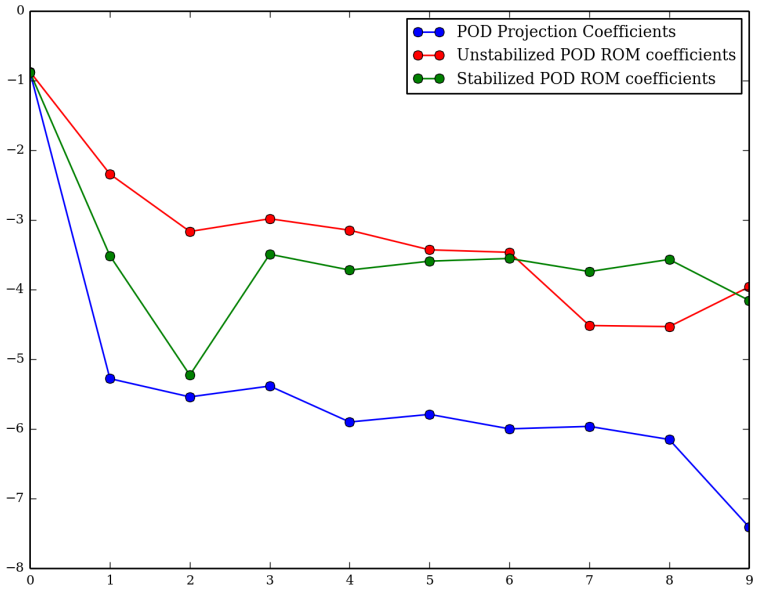


Figure 5.8: Coefficients of POD basis functions at $t = 0.5$.

**Discussion** This section presents a theoretical and numerical investigation of an SUPG-ROM for convection-dominated convection-diffusion-reaction problems. I use numerical analysis to suggest stabilization parameters for the SUPG-ROM. In particular, I propose two different values: one derived from the underlying finite element discretization and finite element inverse estimates (the FE-SUPG-ROM) and another based on the POD truncation error (the POD-SUPG-ROM). I use two representative test problems that contain sharp internal layers. The numerical experiments yield the following conclusions:

1. If the finite element discretization is fine enough to capture the internal layers (and thus the DNS requires no stabilization), i.e., $h \approx \mathcal{O}(\varepsilon)$, then the standard (unstabilized) G-ROM yields accurate results which the SUPG-ROM cannot improve upon.

2. On relatively coarse meshes (which are typical in practical applications) the finite element discretization requires stabilization. I use an SUPG finite element discretization in these circumstances to generate the snapshots. This approach yields POD basis functions with numerical artifacts. Therefore the considered ROMs use noisy POD data.

3. The standard G-ROM yields comparable results to the SUPG-ROMs with respect to the discrete $L^1(0,T;L^2(\Omega))$ norm (for problems with exact solutions) only if I use sufficiently few POD basis functions. Once I increase the number of basis functions past a certain limit the noise of these modes reflects strongly in the G-ROM results.

4. Both the FE-SUPG-ROM and the POD-SUPG-ROM yield significantly more accurate results than the G-ROM for larger numbers of POD basis functions. The SUPG-ROMs suppress noise present in the higher index POD basis functions much better than the G-ROM.

5. The exact meanings of "sufficiently few" and "large" numbers of POD basis functions in the previous points depend on both the equation (e.g., $\varepsilon$, $\vec{b}$, and $c$) and the mesh width $h$. In practice these values will not be known.

6. The studies suggest $\delta_r^{POD}$ is usually smaller than $\delta_r^{FE}$. Based on my experience, I recommend the FE-SUPG-ROM as it suppress oscillations better than the POD-SUPG-ROM and the stabilization coefficient is easy (free) to compute. Additionally, the FE-SUPG-ROM results were never significantly worse in any respect than the POD-SUPG-ROM.

7. The sensitivity study with respect to the mesh width shows that, although the ROMs yield different results, their qualitative behavior is unchanged.

# Chapter 6

# A Galerkin ROM of the Navier-Stokes Equations

## 6.1 Overview

The Galerkin, or unstabilized, NSE-ROM is the starting point for my work in Chapter 8. I discuss the procedure for reducing the NSEs down to an ODE system with $r$ degrees of freedom (and no references to the fine mesh) as well as practical considerations for time stepping.

## 6.2 Weak Form for Model Reduction Without Stabilization

POD basis functions calculated by the method of snapshots (see Section 4.5) are simply linear combinations of solution vectors. As such, the POD basis functions immediately satisfy both the homogeneous boundary conditions and (weakly) the incompressibility condition. The inflow condition is only represented by the centered trajectory (as it is not time dependent). As I show below, this has the fortuitous side effect of completely removing the pressure term from the reduced order model.

I notate the $j$th POD vector by $\vec{\varphi}_j$, which is simultaneously a basis function for all velocity components, and therefore has a number of components, notated as $\vec{\varphi}_{j0}$, $\vec{\varphi}_{j1}$, and $\vec{\varphi}_{j2}$ equal to the dimensionality of the problem.

Let $\vec{\varphi}_j$ be a POD basis function. By taking the inner product of Equation (3.1) and $\vec{\varphi}_j$, I obtain

$$\int_\Omega \vec{\varphi}_j \cdot \vec{u}_t \, d\vec{x} = \frac{1}{Re} \int_\Omega \vec{\varphi}_j \cdot \Delta \vec{u} \, d\vec{x} - \int_\Omega \vec{\varphi}_j \cdot (\vec{u} \cdot \nabla \vec{u}) \, d\vec{x} - \int_\Omega \vec{\varphi}_j \cdot \nabla p \, d\vec{x}. \tag{6.1}$$

Using the divergence theorem, I obtain

$$\begin{aligned} \frac{1}{Re} \int_\Omega \vec{\varphi}_j \cdot \Delta \vec{u} \, d\vec{x} &= -\frac{1}{Re} \int_\Omega \nabla \vec{\varphi}_j : \nabla \vec{u} \, d\vec{x} + \frac{1}{Re} \int_{\partial\Omega} \vec{\varphi}_j (\nabla \vec{u} : \vec{n}) \, d\vec{l} \\ &= -\frac{1}{Re} \int_\Omega \nabla \vec{\varphi}_j : \nabla \vec{u} \, d\vec{x} + \frac{1}{Re} \int_{\Gamma_2} \vec{\varphi}_j[0] \vec{u}_x[0] \, d\vec{l}, \end{aligned} \tag{6.2}$$

where $\vec{n}$ is the outward facing normal, $\vec{u}[0]$ is the $x$ component of $\vec{u}$, and $\Gamma_2$ is the outflow boundary. As $\vec{\varphi}_j = \vec{0}$ on all boundaries except the outflow (due to the homogeneous boundary conditions, which are the no-slip boundary conditions, and the centered trajectory) the other terms in the surface (or path) integral are zero. Similarly, I obtain

$$\begin{aligned} -\int_\Omega \vec{\varphi}_j \cdot \nabla p \, d\vec{x} &= \int_\Omega (\nabla \cdot \vec{\varphi}_j) p \, d\vec{x} - \int_{\partial\Omega} (\vec{\varphi}_j \cdot \vec{n}) p \, d\vec{l} \\ &= \int_\Omega (\nabla \cdot \vec{\varphi}_j) p \, d\vec{x} \\ &= 0 \end{aligned} \tag{6.3}$$

as $p = 0$ on the outflow, $\vec{\varphi}_j = \vec{0}$ on all other boundaries, and $\vec{\varphi}_j$ is assumed to be weakly (though perhaps not pointwise) divergence free. Therefore I may write the weak form of the ROM as

$$\int_\Omega \vec{\varphi}_j \cdot \vec{u}_t \, d\vec{x} = -\frac{1}{Re} \int_\Omega \nabla \vec{\varphi}_j : \nabla \vec{u} \, d\vec{x} + \frac{1}{Re} \int_{\Gamma_2} \vec{\varphi}_j[0] \vec{u}_x[0] \, d\vec{l} - \int_\Omega \vec{\varphi}_j \cdot (\vec{u} \cdot \nabla \vec{u}) \, d\vec{x}. \tag{6.4}$$

## 6.3 The Galerkin ROM

The boundary conditions (fortunately) allow for the complete elimination of the pressure from the ROM. Therefore, I may express the ROM as a small ($\mathcal{O}(10)$) system of nonlinear ODEs.

I notate the ROM by

$$\vec{u}(t, \vec{x}) \approx \vec{u}_s + \sum_{j=0}^{r-1} a_j(t) \vec{\varphi}_j(\vec{x}) \equiv \vec{u}_s + \vec{u}_r, \tag{6.5}$$

where $\vec{u}_s$ is the average snapshot defined by Equation (2.9) and $\vec{a}$ is the vector of POD coefficients with $j$th entry $a_j$. The choice of the centered trajectory relies on the ansatz that the flow may be divided into time-dependent fluctuations and a time-independent average flow and is similar in spirit to the Reynolds Averaged Navier-Stokes (RANS) strategy for turbulence. Reynolds himself used the idea in his initial forays into understanding turbulence.

The centering trajectory method is popular in ROM development [53]. This avoids a large number of challenges in applying boundary conditions to the ROM; in particular, all time-independent boundary conditions are exactly implemented by the ROM by $\vec{u}_s$ in $\vec{u} \approx \vec{u}_s + \vec{u}_r$. There are alternative approaches used in the literature; see [48] for more details.

I notate the reduced mass Laplacian, and boundary matrices by $M$, $K$, and $B$ so that

$$M(i,j) = \int_\Omega \vec{\varphi}_i \cdot \vec{\varphi}_j \, d\vec{x}, \; K(i,j) = \int_\Omega \nabla \vec{\varphi}_i : \nabla \vec{\varphi}_j \, d\vec{x}, \text{ and } B(i,j) = \int_{\Gamma_2} \vec{\varphi}_i[0] \vec{\varphi}_{j,x}[0] \, d\vec{l}. \tag{6.6}$$

The nonlinearity is more difficult to handle: in practice, due to the centering trajectory, I write the nonlinearity as a combination of four terms with discretizations

$$\begin{aligned} \int_\Omega \vec{\varphi}_i \cdot (\vec{u}_s + \vec{u}_r) \cdot \nabla (\vec{u}_s + \vec{u}_r) \, d\vec{x} = &\underbrace{\int_\Omega \vec{\varphi}_i \cdot \vec{u}_r \cdot \nabla \vec{u}_r \, d\vec{x}}_{\text{an } r \times r \times r \text{ tensor}} + \underbrace{\int_\Omega \vec{\varphi}_i \cdot \vec{u}_s \cdot \nabla \vec{u}_r \, d\vec{x}}_{\text{an } r \times r \text{ matrix}} \\ &+ \underbrace{\int_\Omega \vec{\varphi}_i \cdot \vec{u}_r \cdot \nabla \vec{u}_s \, d\vec{x}}_{\text{an } r \times r \text{ matrix}} + \underbrace{\int_\Omega \vec{\varphi}_i \cdot \vec{u}_s \cdot \nabla \vec{u}_s \, d\vec{x}}_{\text{an } r \times 1 \text{ vector}}. \end{aligned} \tag{6.7}$$

Note that as $\vec{\varphi}_i$ and $\vec{u}_s$ are known, the final term (the vector) is time-independent. The matrices are exactly the two possible linearizations of the NSEs. I notate the advective linearization of the nonlinearity about the $k$th POD vector as

$$N_k(i,j) = \int_\Omega \vec{\varphi}_i \cdot (\vec{\varphi}_k \cdot \nabla \vec{\varphi}_j)\, d\vec{x}. \tag{6.8}$$

Similarly, I notate the linearizations about the steady-state solution as

$$N_{s0}(i,j) = \int_\Omega \vec{\varphi}_i \cdot (\vec{u}_s \cdot \nabla \vec{\varphi}_j)\, d\vec{x} \text{ and } N_{s1}(i,j) = \int_\Omega \vec{\varphi}_i \cdot (\vec{\varphi}_j \cdot \nabla \vec{u}_s)\, d\vec{x}. \tag{6.9}$$

Therefore the discrete system is

$$M\vec{a}_t = -\frac{1}{Re}K\vec{a} + \frac{1}{Re}B\vec{a} - N_{s0}\vec{a} - N_{s1}\vec{a} - N_j(\vec{a})\vec{a} + \vec{f}, \tag{6.10}$$

where $N_j$ is a tensor representing the nonlinearity and

$$\vec{f}(j) = -\frac{1}{Re}\int_\Omega \nabla\vec{\varphi}_j \cdot \vec{u}_s\, d\vec{x} + \frac{1}{Re}\int_{\Gamma_2} \vec{\varphi}_j[0]\vec{u}_{s,x}[0]\, d\vec{l} - \int_\Omega \vec{\varphi}_j \cdot (\vec{u}_s \cdot \nabla \vec{u}_s)\, d\vec{x} \tag{6.11}$$

is the mean contribution.

## 6.4 The ROM Discretization

Before writing the ROM discretization, I simplify Equation (6.10) to combine all of the linear terms into one $r \times r$ matrix. Let

$$\tilde{L} = -\frac{1}{Re}K + \frac{1}{Re}B - N_{s0} - N_{s1}. \tag{6.12}$$

These choices in the discretization in (6.3) result in a system of ODEs which may be written as

$$M_r\vec{a}_t = \tilde{N}(\vec{a})\vec{a} + \tilde{L}\vec{a} + \vec{f}. \tag{6.13}$$

At this point the ODE system may be solved by any standard time integrator. I use the standard Runge Kutta 4 algorithm unless otherwise noted.

The nonlinearity's implementation is not straightforward and deserves special mention. There are a variety of ways to handle nonlinearities with POD; since the NSEs' nonlinearity is quadratic in $\vec{a}$, I have chosen the simplest method (that is, I implement the nonlinearity as an $r \times r \times r$ tensor). In practice, this means that I store $r$ $r \times r$ full matrices, where the $i$th such matrix (notated by $N[i]$) has entries

$$N[i](j,k) = \int_\Omega \vec{\varphi}_i \cdot \vec{\varphi}_j \cdot \nabla\vec{\varphi}_k\, d\vec{x}. \tag{6.14}$$

Therefore, in the language of `deal.II`, the tensor has the type `vector<FullMatrix<double>>`; that is, a vector (dynamic array) of dense matrices (each $r \times r$). This leads to the implementation (for explicit ODE solvers) of the action described in Algorithm 1 and Equation (6.13) shown in Algorithm 1.

The principal lesson from the implementation is the algorithmic complexity; evaluating each loop iteration requires $\mathcal{O}(r^2 + r)$ operations, which makes evaluation of the function $\mathcal{O}(r^3)$. This means that any reasonable online filtering scheme must also be $\mathcal{O}(r^3)$ or better for us to maintain the fundamental property of the ROM (its speed).

**Algorithm 1** Implementation of a method to evaluate the right-hand side of the ODE system described by Equation (6.13) and apply the mass matrix. In the source `linear_operator` refers to $\tilde{L}$ and `mean_contribution` refers to $\vec{f}$. Line 14 uses an $LU$ factorization of the mass matrix to apply the action of $M_r^{-1}$ to the result.

```
1  void PlainRHS::apply(Vector<double> &dst, const Vector<double> &src)
2  {
3    const unsigned int n_dofs = src.size();
4    linear_operator.vmult(dst, src);
5    dst += mean_contribution;
6
7    Vector<double> temp(n_dofs);
8    for (unsigned int pod_vector_n = 0; pod_vector_n < n_dofs; ++pod_vector_n)
9      {
10       nonlinear_operator[pod_vector_n].vmult(temp, src);
11       dst(pod_vector_n) -= temp * src;
12     }
13
14   factorized_mass_matrix.apply_lu_factorization(dst, false);
15 }
```

# Chapter 7

# POD Spatial Filtering

## 7.1 Overview

The new regularized ROMs (discussed in Chapter 8) use POD spatial filtering to smooth the flow variables and increase the numerical stability of the models. In standard reduced order modeling, however, spatial filtering is defined only implicitly, by truncating the POD basis used in the Galerkin approximation. Two types of *explicit* POD spatial filters are proposed in this Chapter: a POD projection filter (Section 7.2) and a POD differential filter (Section 7.3). The properties of both POD spatial filters are discussed in Section 7.4.

## 7.2 The POD Projection Filter

For a fixed $r_1 < r$ and a given $\vec{u}_r \in X^r$, the POD projection seeks $\mathcal{F}(\vec{u}_r) \in X^{r_1}$ such that

$$(\mathcal{F}(\vec{u}_r), \vec{\varphi}_j) = (\vec{u}_r, \vec{\varphi}_j), \quad \forall j = 0, \cdots, r_1 - 1. \tag{7.1}$$

The POD projection (7.1) has been used for theoretical purposes, e.g., in the error analysis of the G-ROM [70]. It was also used to determine the modeling constant in the dynamic subgrid-scale ROM (see Section 3.3.4 in [117]). Its use as a smoothing POD spatial filter is, to the best of my knowledge, new.

## 7.3 The POD Differential Filter

The POD differential filter is defined as follows: Let $\delta$ be the radius of the POD differential filter. For a given $\vec{u}_r \in X^r$, find $\mathcal{F}(u_r) \in X^r$ such that

$$\Big((I - \delta^2\Delta)\mathcal{F}(u^r), \vec{\varphi}_j\Big) = (\vec{u}_r, \vec{\varphi}_j), \quad \forall j = 0, \cdots, r - 1. \tag{7.2}$$

The differential filter has been used in LES of turbulent flows [44], [43]. Its use in a POD context is, again, to the best of my knowledge, new.

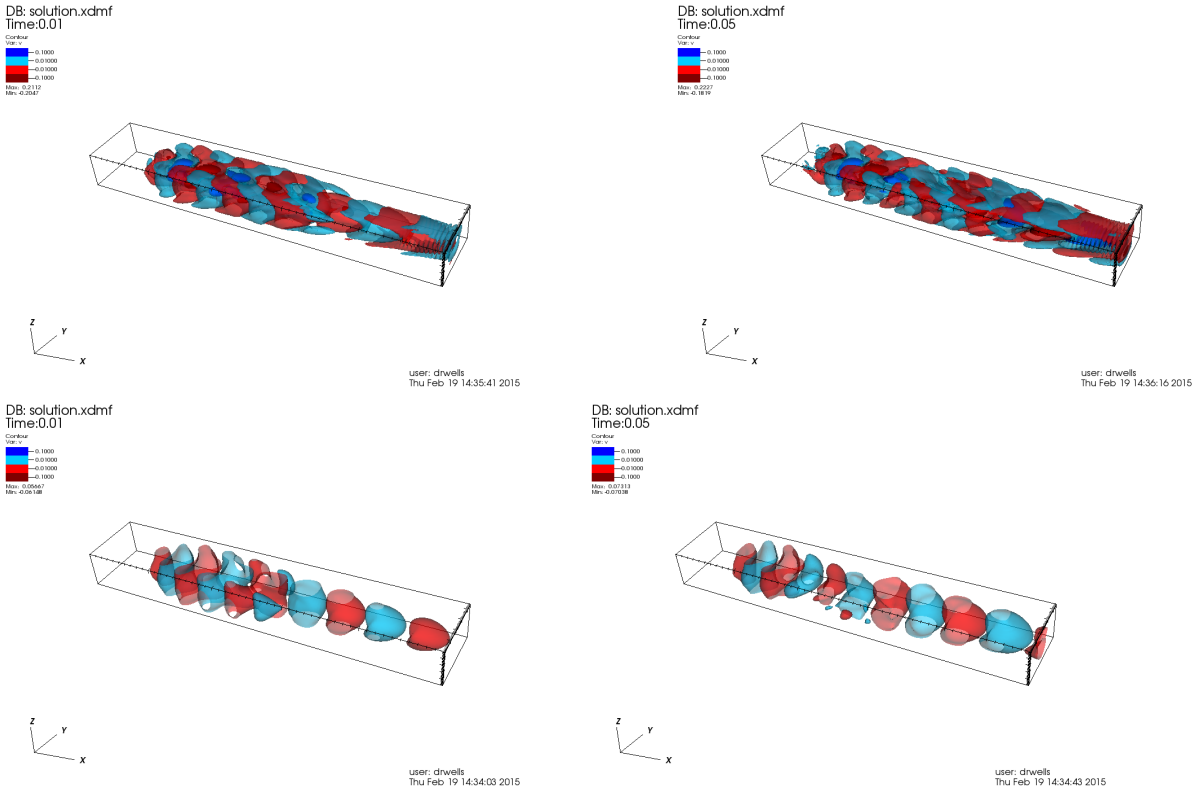


Figure 7.1: The first and fifth POD vectors from Experiment 3.2. I use the finite element option to filter with $\delta = 0.5$. The unfiltered POD vectors are on top and the filtered are on the bottom. The choice $\delta = 0.5$ is too large for practical purposes, but demonstrates that filtering both removes kinetic energy (the isosurfaces are smaller) and enlarges the scales of motion (e.g., the first POD vector goes from twelve structures to just nine.)

## 7.4 POD Filter Properties

The POD projection (7.1) and the POD differential filter (7.2) share several appealing properties [17]. They act as spatial filters, since they eliminate the small scales (i.e., high frequencies) from the input. Indeed, the POD differential filter (7.2) uses an elliptic operator to smooth the input variable. The effect of the POD projection on an input from the POD space $X^r$ is the elimination of the components corresponding to the POD basis functions $\{\vec{\varphi}_j\}_{j=r_1}^{r-1}$ which generally correspond to small scales.

Note that the POD projection is different from the truncation of the POD basis used in the G-ROM development: indeed, the effect of the POD basis truncation is that the G-ROM operates at low resolution and, therefore, produces inaccurate results. The POD projection, on the other hand, eliminates from the input the high frequency components, which are generally considered to play an important role in the numerical instability of the G-ROM.

Both POD spatial filters have a low computational overhead. As noted in Section 6.4, the desired algorithmic complexity of any additional filters should be dominated by the $\mathcal{O}(r^3)$ cost in evaluating the nonlinearity. The POD projection amounts to simply setting $a_j = 0$ for $j = r_1$ to $j = r$, which is a cost bounded by $\mathcal{O}(r)$. The POD differential filter is equivalent to solving an $r \times r$ linear system; since the matrix only depends on the POD basis, it may be factorized and repeatedly solved for a cost of $\mathcal{O}(r^2)$, which is also dominated by the cost of the nonlinearity.

A fortunate property of both filters is that they only modify $\vec{a}$, not the POD basis vectors themselves; therefore any filtered quantity will still be representable by the POD basis and therefore (still) weakly incompressible.

Although the two POD filters share important properties they vary in how they handle length scales in the ROM. The POD projection achieves spatial filtering *implicitly*, as the input is projected onto a smaller POD subspace which usually represents larger scales of motion. There is no explicit length scale in this operation. On the other hand, the POD differential filter uses an *explicit* length scale $\delta$ to filter the ROM solution vector.

## 7.5 POD Spatial Filter Implementation

### 7.5.1 Overview

There are several algorithmic decisions that need to be made in the implementation of the POD projection (7.1) and the POD differential filter (7.2). The most important decision is whether to regard the ROM variables as POD variables in the POD space $X^r$ (that is, the span of the POD basis functions) or as FE variables in the FE space $X^h$ (as expected, this is the span of the FE basis functions). I name these approaches "the POD option" and "the FE option". Both options are valid as $X^r \subset X^h$. Furthermore, regardless of the filter choice, one may need to filter quantities in $X^h \backslash X^r$ (that is, values in $X^h$ that are not in $X^r$, such as $\vec{u}_s$). Since the centering trajectory is commonly used in ROM applications (in particular, I use it for all of the experiments in this thesis) I provide a detailed implementation for both filters below.

Both filters take as input a variable $\vec{u}$ and return the filtered variable $\mathcal{F}(\vec{u})$. I represent

$\vec{u}$ as either

$$\vec{u} = \sum_{j=0}^{r-1} a_j \vec{\varphi}_j \tag{7.3}$$

in the POD space or

$$\vec{u}[k] = \sum_{j=0}^{N-1} c_{k,j} v_{h,j} \tag{7.4}$$

in the FE space, where $\vec{u}[k]$ refers to the $k$th component of $\vec{u}$.

### 7.5.2 The POD Projection

If the input variable $\vec{u}$ is in $X^r$ then the POD projection filter (7.1) is just a truncation. In particular, the output variable in this case is simply

$$\mathcal{F}(\vec{u}) = \sum_{j=0}^{r_1-1} a_j \vec{\varphi}_j. \tag{7.5}$$

If the input variable is in $X^h$ but not in $X^r$, then the POD projection requires careful treatment: for example, the projection of the centering trajectory onto the POD basis is disastrous (see Figure (4.4) for further discussion).

I do not cover the case where the input $\vec{u}$ *and* output $\mathcal{F}(\vec{u})$ are both in $X^h$ in this thesis. Note that a finite element representation of $\mathcal{F}(\vec{u})$ is sufficient for assembling the ROM tensors and matrices; recall the implementation details listed in Section 6.4. A possible approach is to replace the POD projection (7.1) with

$$(\mathcal{F}(\vec{u}), v_{h,i}) = (\vec{u}, v_{h,i}), \forall i = 0, \cdots, N_1 - 1. \tag{7.6}$$

where $N_1 < N$.

**Remark 7.1.** *The task of choosing the projection space in* (7.6) *may be straightforward or daunting depending on the spatial discretization implied by $X^h$. If a spectral (or spectral element) method is used then the projection space can simply be the span of the first few Fourier modes (or the low-order polynomials on each spectral element, see [35]). However, if $X^h$ consists of standard, low-order finite elements, then I do not believe there is an easy choice for the projection space in Equation* (7.6).

### 7.5.3 POD Differential Filter

In all cases the POD differential filter (7.2) reduces to the linear system

$$(M + \delta^2 S)\mathcal{F}(\vec{u}) = M\vec{u}, \tag{7.7}$$

where (as above) the dimensions and entries of the matrices $M$ and $S$ and vectors $\vec{u}$ and $\mathcal{F}(\vec{u})$ vary from case to case.

If $\vec{u}$ is in $X^r$, then I make the following choices in the linear system in Equation (7.7): $M$ and $S$ are the usual POD mass and POD stiffness matrices (see Section 2.4) and $\mathcal{F}(\vec{u})$ and $\vec{u}$ are vectors of POD coefficients.

If $\vec{u}$ is in $X^h$ but not in $X^r$, then I consider two approaches for implementation of the differential filter.

**The FE Option** In this case I seek the finite element representation of $\mathcal{F}(\vec{u})$. As I note above, the finite element representation of $\mathcal{F}(\vec{u})$ suffices in applications because I use it to assemble the components of the ROM before its time evolution. In this case all of the terms in (7.7) are the finite element quantities and representations.

**The POD Option** If I instead want $\mathcal{F}(\vec{u})$ in $X^r$, then I choose the quantities in Equation (7.7) to be the ROM matrices and the POD coefficient representations of the vectors.

# Chapter 8

# Two Regularized ROMs

## 8.1 Overview

I propose two *regularized ROMs* (Reg-ROMs) in this chapter. These new Reg-ROMs use spatial filtering to smooth, or regularize, various terms in the ROMs. Note that these Reg-ROMs are fundamentally different from the calibration approaches used in [1, 118, 32]. Indeed, the Reg-ROMs proposed in this thesis use equations different from those used in the G-ROM. The calibration models in [1, 118, 32], on the other hand, utilize the G-ROM equations and only calibrate their coefficients by utilizing a Tikhonov type regularization. I also emphasize that, although a spatial filtering is commonly used in developing regularized and LES models, it is not generally employed in reduced order modeling. In fact to the best of my knowledge, the only instance where spatial filtering was used to develop ROMs was [117], where the projection of the ROM variables on a subspace of the POD space $X^r$ was used to determine the modeling constant in the dynamic subgrid-scale ROM (see Section 3.3.4 in [117]). This projection, which is called the POD projection (Proj) in what follows, will be one of the two spatial filters used in this chapter. The other spatial filter is the POD differential filter described in Equation (7.2).

## 8.2 The Leray ROM

Jean Leray attempted to solve the NSEs in his landmark 1934 paper [75]. He was able to prove the existence of solutions for the altered problem

$$\begin{aligned} \vec{w}_t &= \frac{1}{Re}\Delta\vec{w} - \mathcal{F}(\vec{w})\cdot\nabla\vec{w} - \nabla p \\ \nabla\cdot\vec{w} &= 0, \end{aligned} \tag{8.1}$$

where $\mathcal{F}(\vec{w})$ is a convolution with a Gaussian distribution with filter radius $\delta$, or

$$\mathcal{F}(\vec{w}) = g_\delta \star w. \tag{8.2}$$

There is significant discussion of the choices between various filters in [74]. Unless otherwise noted, I always use the differential filter

$$\mathcal{F}(\vec{w}) = (\delta^2\Delta + 1)^{-1}\vec{w}, \tag{8.3}$$

which is provably within $\mathcal{O}(\delta^4)$ of the more theoretically sound Gaussian filter. For additional results and discussion on this type of filtering in the context of finite elements, see [102, 73].

In turbulence modeling, Leray's model is the basis for a class of stabilization methods called the Leray-$\alpha$ regularization models [74]. Leray's key observation was that the nonlinear term is the most problematic as it serves to transfer energy from resolved to unresolved scales. Therefore, I only filter the resulting nonlinear term (as $\vec{u}_s$ is known) and obtain the *Leray-ROM* (L-ROM)

$$\vec{u}_{ft} = \frac{1}{Re}\Delta\vec{u}_f + \frac{1}{Re}\Delta\vec{u}_s - \mathcal{F}(\vec{u}_f)\cdot\nabla\vec{u}_f - \vec{u}_s\cdot\nabla\vec{u}_f - \vec{u}_f\cdot\nabla\vec{u}_s - \vec{u}_s\cdot\nabla\vec{u}_s - \nabla p. \tag{8.4}$$

Therefore, following the same procedure as in Equation (6.3), I obtain

$$\vec{a}_t = \tilde{N}(\mathcal{F}(\vec{a}))\vec{a} + \tilde{L}\vec{a} + \vec{f}, \tag{8.5}$$

which is the same as Equation (6.13) up to the filtering of the advective term in the nonlinearity. There are several possible time discretizations for this model: see [38] for several time stepping schemes implemented for related finite element solvers. The Leray model has been recently used as a regularization model in the numerical simulation of turbulent flows [74]. Its use in a ROM context, however, is new.

There are two different ways to implement this modification to the nonlinearity in the L-ROM: one can either filter the POD coefficients when evaluating the right-hand side or filter the finite element representation of the POD basis vectors. I chose the second approach for consistency with the centering trajectory.

## 8.3 The Evolve-Then-Filter ROM

I also propose the evolve-then-filter regularized ROM (EF-ROM) model. As a first demonstration of the algorithm, consider the unforced Navier-Stokes equations with homogeneous boundary conditions. The evolve-then-filter procedure for a forward Euler time discretization is

$$\begin{aligned}\left(\frac{\vec{w}_r^{n+1} - \vec{u}_r^{n+1}}{\Delta t}\right) + \frac{1}{Re}\left(\nabla\vec{u}_s + \vec{u}_r^n, \nabla\vec{\varphi}_k\right) + \left(((\vec{u}_s + \vec{u}_r^n)\cdot\nabla)(\vec{u}_s + \vec{u}_r^n), \vec{\varphi}_k\right) = 0,\\ \vec{u}_r^{n+1} = \mathcal{F}(\vec{w}_r^{n+1}).\end{aligned} \tag{8.6}$$

That is, I *evolve* the POD coefficients by one time step to obtain temporary values $\vec{w}_r^{n+1}$ then *filter* the temporary value to obtain $\vec{u}_r^{n+1}$. I use either Equation (7.1) or Equation (7.7) to calculate $\mathcal{F}(\vec{w}_r^{n+1})$.

It is worth noting that the implementation of this ROM in the more general case is very simple and I discuss it in Figure 8.1.

In the first (evolve) step of the EF-ROM (8.6), the intermediate step $\vec{w}_r^{n+1}$ is just a linear combination of the fluctuations; put another way, the ROM only evolves the fluctuations, not $\vec{u}_s$, in time. It is worth mentioning that $\vec{w}_r^{n+1}$ *does* depend implicitly on $\vec{u}_s$ because of the forcing term and linearizations of the convective term present in the right hand side of the ROM in (6.13). In the second step of the EF-ROM (8.6), only the time dependent

```
1  void RungeKutta4PostFilter::step
2  (double time_step, const Vector<double> &src, Vector<double> &dst)
3  {
4    Vector<double> temp(src.size());
5    RungeKutta4::step(time_step, src, temp);
6    filter_function->apply(dst, temp);
7  }
8
9
10 void PostDifferentialFilter::apply(Vector<double> &dst,
11                                     const Vector<double> &src)
12 {
13   mass_matrix.vmult(dst, src);
14   factorized_post_filter_matrix.apply_lu_factorization(dst, false);
15 }
16
17
18 void PostL2ProjectionFilter::apply(Vector<double> &dst,
19                                     const Vector<double> &src)
20 {
21   dst = src;
22   for (unsigned int pod_vector_n = cutoff_n; pod_vector_n < dst.size();
23        ++pod_vector_n)
24     {
25       dst[pod_vector_n] = 0.0;
26     }
27 }
```

Figure 8.1: Implementation of three methods related to the evolve-then-filter ROM. Since I use `RungeKutta4` for time evolution the evolve-then-filter scheme amounts to calling the RK4 algorithm and then filtering the result, which I encapsulate in the method `RungeKutta4PostFilter`. The function call `filter_function->apply` on line 6 implements either Equation (7.7) (lines ten to eighteen) or Equation (7.1) (lines 18 to 27).

fluctuations are filtered; the time independent value $\vec{u}_s$ is *not* filtered. Furthermore, the approximation to $\vec{u}(t^{n+1})$ is $\vec{u}_s+\vec{u}_r(t_{n+1})$, which consists of $\mathcal{F}(\vec{w}_{n+1})$ and the time independent part of the approximation $\vec{u}_s$. This filtering strategy is consistent with the general philosophy of the EF-ROM, where only variables that are time dependent should be filtered. Put another way, since I never evolve $\vec{u}_s$, I never filter $\vec{u}_s$. In conclusion, there is no need for special treatment of $\vec{u}_s$ in the EF-ROM.

## 8.4 Numerical Results

### 8.4.1 Overview

This section has two goals:

1. to compare the L-ROM (8.5), the EF-ROM (8.6), and the standard G-ROM,
2. to perform a sensitivity study of the ROM results with respect to the number of POD basis functions retained in the ROM (i.e., $r$).

I use two filtering strategies for each of the two Reg-ROMs (the L-ROM and the EF-ROM): the POD differential filter (7.2) and the POD projection (7.1). Therefore I consider, in total, four ROM and filter combinations:

- L-ROM-DF (the Leray ROM with the Differential Filter)
- L-ROM-Proj (the Leray ROM with the Projection Filter)
- EF-ROM-DF (the evolve-then-filter ROM with the Differential Filter)
- EF-ROM-Proj (the evolve-then-filter ROM with the Projection Filter)

I assess the accuracy of the ROMs by comparing the time evolution of the $L^2$ norm of the fluctuation (i.e., comparing $\vec{u}-\vec{u}_s$ and $\vec{u}_r$) and the phase plots.

Each of the four ROM choices implies an optimization problem: since $\delta$ and $r_1$ are independent variables, I try to select the optimal values for each by a calibration approach. Put another way, I tried 101 values for $\delta$ and all integer values between $r/2$ and $r$ for $r_1$. By "optimal" I mean that the time-averaged $L^2$ norm of $\vec{u}_r$ is as close to the time-averaged $L^2$ norm of $\vec{u}-\vec{u}_s$ as possible. This optimization process captured the correct average $L^2$ norm when varying $\delta$, but did not for $r_1$ because of the limited number of possibilities (i.e., I can only truncate an integer number of POD vectors in the filter).

### 8.4.2 Test Case Description

All of the results in this section are based on Experiment 3.2. In particular, I generate $2,001$ snapshots between $t = 30$ and $t = 50$, which, by visual inspection, covers several vortex sheddings. I run every ROM from $t = 30$ to $t = 500$. I do not attempt to model the flow between $t = 0$ and $t = 30$ because that time interval corresponds to a variety of different flow conditions: the DNS varies from $\vec{u} = \vec{0}$ at $t = 0$ to quasiperiodic motion at $t = 30$.

Since the ROM can only capture essential structures in flow, I only use the ROM when such structures are present.

Each ROM takes about four minutes to run from $t = 30$ to $t = 500$, while the DNS took about four days to cover the same time interval. The L-ROM required about fifteen additional minutes to filter the POD vectors. This cost increased as $\delta$ increased as the resulting linear system is much worse conditioned for larger values of $\delta$. I suspect that better preconditioners could alleviate this cost, but since it is an offline cost (the POD basis vectors only need to be filtered *once* for a given value of $\delta$) I did not spend much time attempting to optimize this part of the code.

### 8.4.3 Implementation of the Differential Filter

I opted to implement the "FE-Option" for the POD differential filter (8.3): that is, the entries in the tensor (6.14) are replaced by

$$\int_\Omega \vec{\varphi}_i \cdot (\vec{u}_s + \vec{u}_r) \cdot \nabla(\vec{u}_s + \vec{u}_r)\, d\vec{x} \to \int_\Omega \vec{\varphi}_i \cdot \mathcal{F}(\vec{u}_s + \vec{u}_r) \cdot \nabla(\vec{u}_s + \vec{u}_r)\, d\vec{x}. \tag{8.7}$$

Put another way, when assembling the tensors (and the two linearization matrices $N_{s0}$ and $N_{s1}$, see Equation (6.9)), I replace the POD vectors that appear in the advective term with their filtered versions. I describe this procedure in Algorithm 2.

**Algorithm 2** Algorithm for assembling the reduced quantities with the FE option for stabilization.

**procedure** FILTER_POD_VECTORS
  **for** $j = 0, \cdots, r-1$ **do**
    $\mathcal{F}(\vec{\varphi}_j) \leftarrow (\vec{M} + \delta^2 \vec{S} - \delta^2 \vec{B})^{-1} \vec{\varphi}_j$

**procedure** ASSEMBLE_REDUCED_TENSOR
  **for** $l = 0, \cdots, r-1$ **do**
    **for** $i = 0, \cdots, r-1$ **do**
      **for** $j = 0, \cdots, r-1$ **do**

$$A_l[i,j] \leftarrow \int_\Omega \vec{\varphi}_i \cdot \mathcal{F}(\vec{\varphi}_j) \cdot \nabla \vec{\varphi}_j \, d\vec{x}$$

$$N[i](l,j) \leftarrow \vec{\varphi}_i^T \begin{pmatrix} A_l & 0 & 0 \\ 0 & A_l & 0 \\ 0 & 0 & A_l \end{pmatrix} \vec{\varphi}_j$$

**procedure** ASSEMBLE_REDUCED_ADVECTIVE_LINEARIZATION
  **for** $i = 0, \cdots, r-1$ **do**
    **for** $j = 0, \cdots, r-1$ **do**

$$N_{s0}(i,j) \leftarrow \int_\Omega \vec{\varphi}_i \cdot \mathcal{F}(\vec{u}_s) \cdot \nabla \vec{\varphi}_j \, d\vec{x}$$

**procedure** ASSEMBLE_REDUCED_GRADIENT_LINEARIZATION
  **for** $i = 0, \cdots, r-1$ **do**
    **for** $j = 0, \cdots, r-1$ **do**

$$N_{s1}(i,j) \leftarrow \int_\Omega \vec{\varphi}_i \cdot \mathcal{F}(\vec{\varphi}_j) \cdot \nabla \vec{u}_s \, d\vec{x}$$

**Remark 8.1.** *This approach has a few notable features:*

- *It modifies the resulting ROM system in an inconsistent way by changing the advective term. This looks a little like the SUPG approach described in Chapter 5, except in this case the resulting ROM no longer satisfies the original PDE.*
- *There is no need to filter the system during time evolution because all of the filtering is done when the ROM is assembled. The time evolution code for this stabilized ROM is identical to that of the G-ROM shown in Algorithm 1.*
- *I refer to this approach as* prefiltering *because there is no filtering in the time evolution.*

The alternative to this approach (which I do *not* use in this thesis) is to filter the POD coefficients themselves, which I describe in Algorithm 3. This is the same as Algorithm 1 except for lines $8-10$ and line 16.

**Algorithm 3** Algorithm for filtering the nonlinearity with the POD option for stabilization.

```
1  void PODDifferentialFilterRHS::apply
2  (Vector<double> &dst, const Vector<double> &src)
3  {
4    const unsigned int n_dofs = src.size();
5    linear_operator.vmult(dst, src);
6    dst += mean_contribution;
7
8    Vector<double> filtered_src(src.size());
9    mass_matrix.vmult(filtered_src, src);
10   factorized_filter_matrix.apply_lu_factorization(filtered_src, false);
11
12   Vector<double> temp(n_dofs);
13   for (unsigned int pod_vector_n = 0; pod_vector_n < n_dofs; ++pod_vector_n)
14     {
15       nonlinear_operator[pod_vector_n].vmult(temp, src);
16       dst(pod_vector_n) -= temp * filtered_src;
17     }
18
19   factorized_mass_matrix.apply_lu_factorization(dst, false);
20 }
```

**Remark 8.2.** *This prefiltering approach has some advantages and disadvantages:*

- *The implementation is very simple: an $r \times r$ linear system is solved every time the right-hand-side is evaluated. This is a negligible cost (see Section 6.4).*
- *This method is inconsistent with the original filtered PDE* (8.1) *because it does not filter the entire advective term $\vec{u}_s + \vec{u}_r$.*

I chose not to use this approach because of the open question of how to handle the centering trajectory: without use of the centering trajectory the implementation of this method is clear.

### 8.4.4 G-ROM Results

I present results for the standard G-ROM for comparison with the REG-ROMs. The main point of this example is to demonstrate that the G-ROM does very poorly for this test problem and can be improved with regularization.

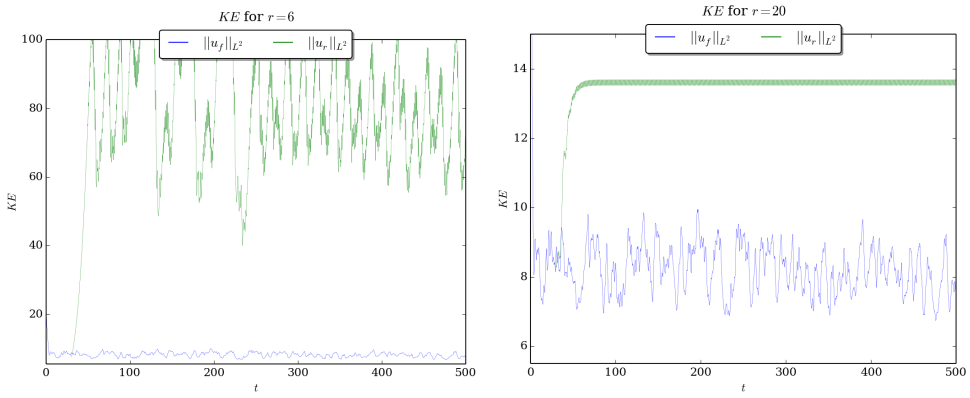


Figure 8.2: 3D flow past a cylinder, G-ROM (green) and DNS (blue). Time evolution of the $L^2$ norm; $r = 6$ (left) and $r = 20$ (right).

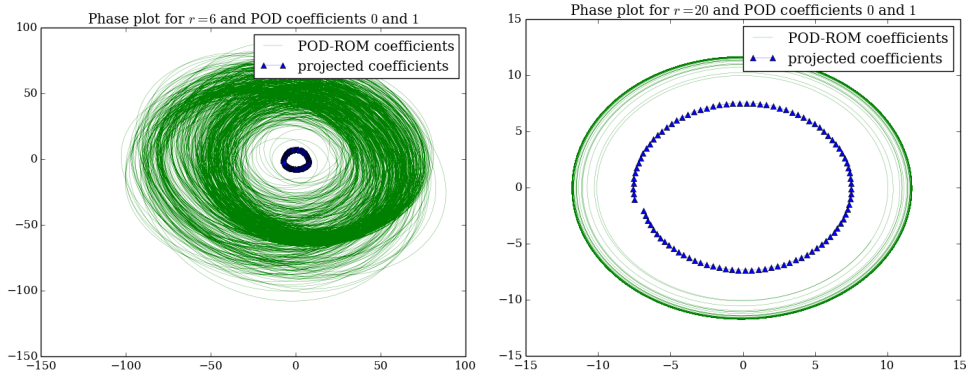


Figure 8.3: 3D flow past a cylinder, G-ROM (green) and POD projection of DNS data (blue). Phase portraits for $a_1$ and $a_2$; $r = 6$ (left) and $r = 20$ (right). Even for a relatively large number of POD vectors the flow contains too much kinetic energy (i.e., the $L^2$ norm is too large).

Figure 8.2 demonstrates the need for stabilization: the $r = 6$ G-ROM fails catastrophically while the $r = 20$ ROM consistently overshoots the correct amount of kinetic energy. Figure 8.3 displays the phase portraits for the first and second coefficients of the G-ROM and the POD projection of the centered DNS data for $r = 6$ and $r = 20$. Figure 8.2 shows that the G-ROM yields very inaccurate results for $r = 6$ and somewhat inaccurate results for $r = 20$.

### 8.4.5 L-ROM With the Differential Filter (L-ROM-DF)

Figure 8.4 displays the time evolution of the $L^2$ norm of the solutions of the L-ROM-DF and DNS for $r = 6$ and $r = 20$. Figure 8.4 shows that, for the optimal $\delta$ value, the L-ROM-DF accurately reproduces the average, but not the amplitude of the time evolution of the $L^2$ norm of the DNS results for both $r = 6$ and $r = 20$.

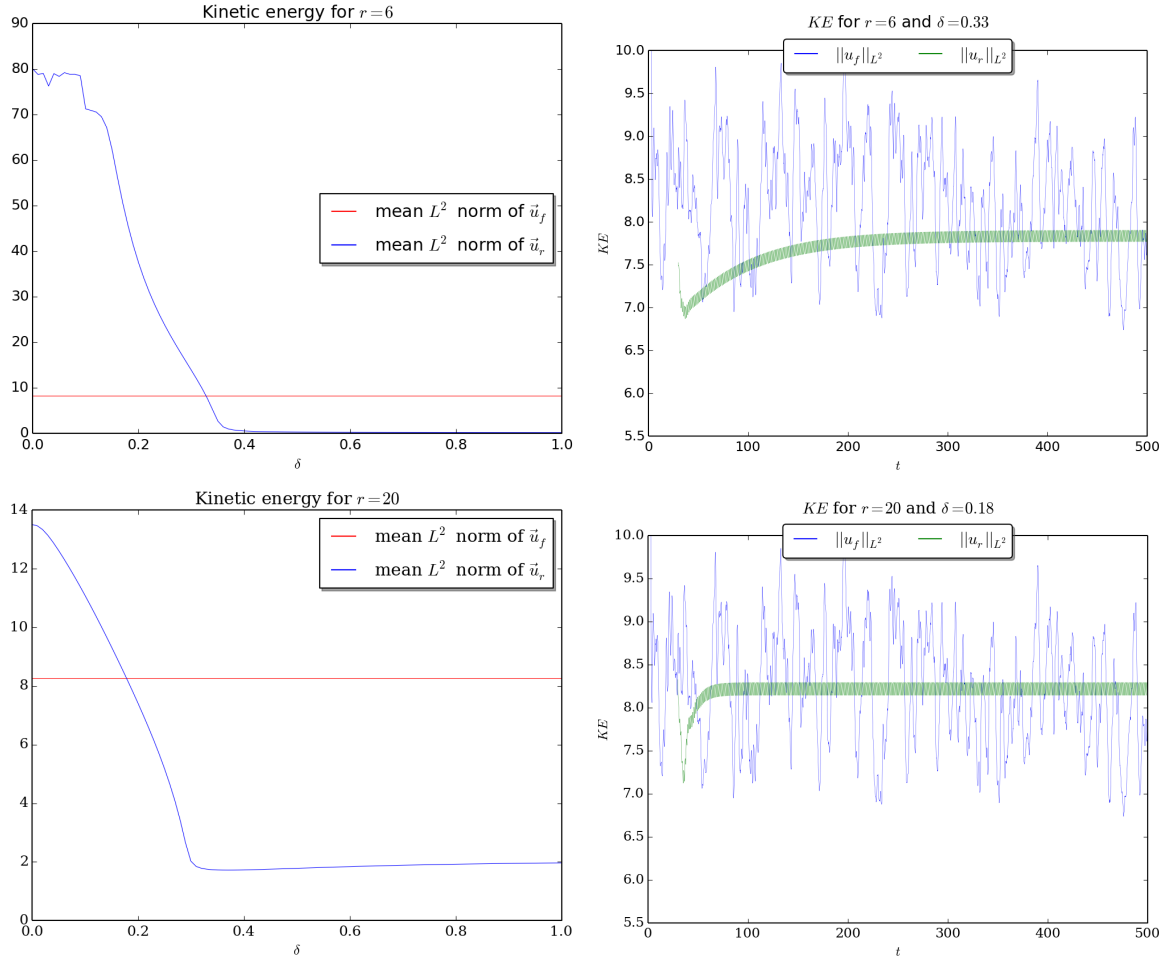


Figure 8.4: 3D flow past a cylinder, L-ROM-DF (green) and DNS (blue). Mean (left column) and time evolution (right column) of the $L^2$ norm of the solution; $r = 6$ (top row) and $r = 20$ (bottom row). The time evolution of the $L^2$ norm of the solution (right column) is plotted for the optimal mean $L^2$ norm of the solution (left column): $\delta = 0.33$ for $r = 6$ (top row) and $\delta = 0.18$ for $r = 20$ (bottom row).

Figure 8.5 displays the phase portraits for the first and second POD coefficients of the L-ROM-DF and POD projection of DNS data for $r = 6$ and $r = 20$. Figure 8.5 shows that, for the optimal $\delta$ value, the L-ROM-DF yields moderately accurate results for $r = 6$ and accurate results for $r = 20$.

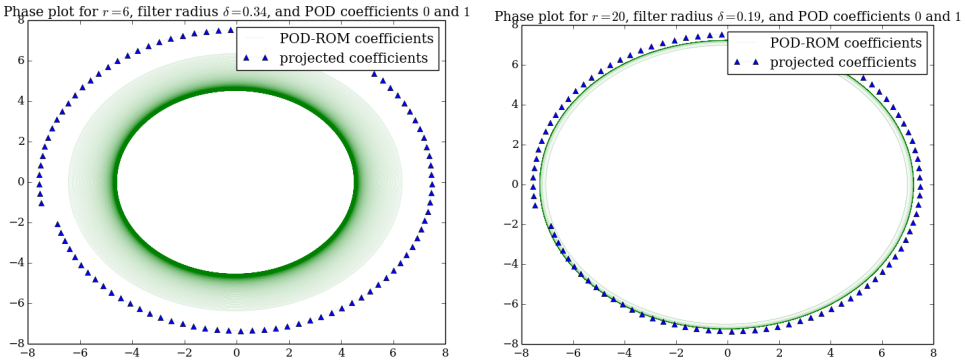


Figure 8.5: 3D flow past a cylinder, L-ROM-DF with optimal $\delta$ value (green) and POD projection of DNS data (blue). Phase portraits for $a_1$ and $a_2$; $r = 6$ (left) and $r = 20$ (right).

### 8.4.6 L-ROM with the POD Projection (L-ROM-Proj)

Figure 8.6 displays the time evolution of the $L^2$ norm of the solutions of the L-ROM-Proj and DNS for $r = 6$ and $r = 20$. Figure 8.6 shows that, for the optimal $r_1$ value, the L-ROM-Proj yields the inaccurate average and amplitude of the time evolution of the $L^2$ norm of the solution for both $r = 6$ and $r = 20$.

Figure 8.7 displays the phase portraits for the first and second POD coefficients of the L-ROM-Proj and POD projection of DNS data for $r = 6$ and $r = 20$. Figure 8.7 shows that, for the optimal $r_1$ value, the L-ROM-Proj yields extremely inaccurate results for $r = 6$ and moderately accurate results for $r = 20$.

### 8.4.7 The EF-ROM with Differential Filter (EF-ROM-DF)

Figure 8.8 displays the time evolution of the $L^2$ norm of the solutions of the L-ROM-DF and DNS for $r = 6$ and $r = 20$. Figure 8.8 shows that, for the optimal $\delta$ value, the L-ROM-DF accurately reproduces the average, but not the amplitude of the time evolution of the $L^2$ norm of the DNS results for both $r = 6$ and $r = 20$.

Figure 8.9 displays the phase portraits for the first and second POD coefficients of the EF-ROM-DF and POD projection of DNS data for $r = 6$ and $r = 20$. Figure 8.8 shows that, for the optimal $\delta$ value, the EF-ROM-DF yields accurate results for both $r = 6$ and $r = 20$.

### 8.4.8 EF-ROM with POD Projection (EF-ROM-Proj)

Figure 8.10 displays the time evolution of the $L^2$ norm of the solutions of the EF-ROM-Proj and DNS for $r = 6$ and $r = 20$. Figure 8.10 shows that, for the optimal $r_1$ value, the EF-ROM-Proj yields the inaccurate average and amplitude of the time evolution of the $L^2$ norm of the solution for both $r = 6$ and $r = 20$.

Figure 8.11 displays the phase portraits for the first and second POD coefficients of the EF-ROM-Proj and POD projection of DNS data for $r = 6$ and $r = 20$. Figure 8.10 shows that, for the optimal $r_1$ value, the EF-ROM-Proj yields extremely inaccurate results for $r = 6$ and accurate results for $r = 20$.

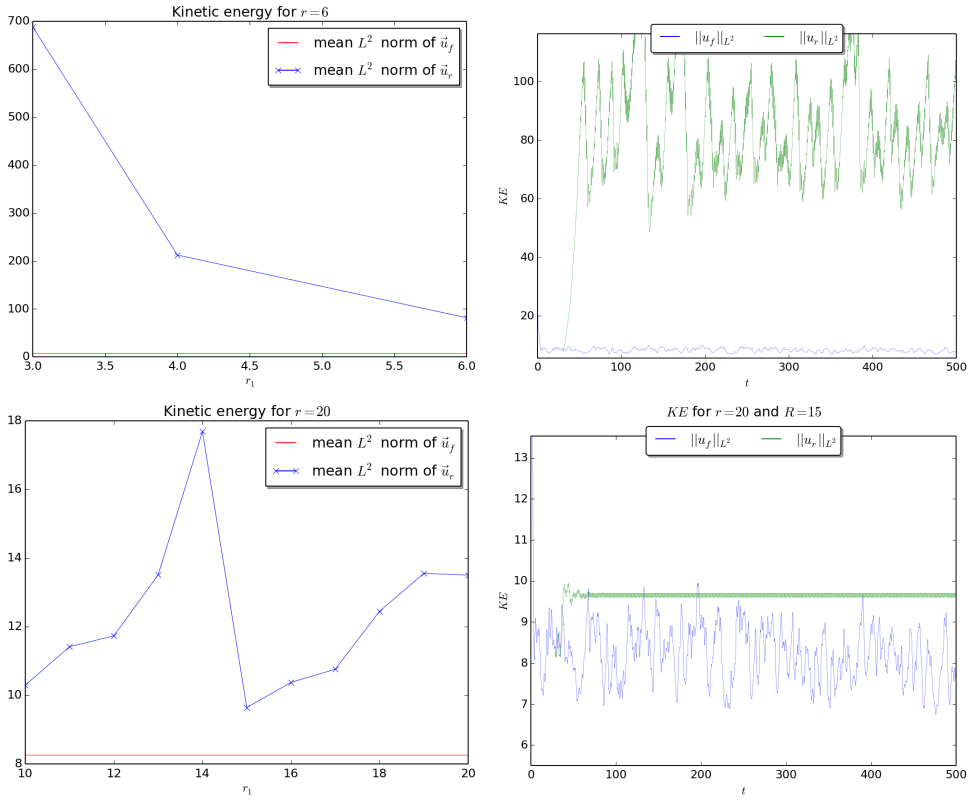


Figure 8.6: 3D flow past a cylinder, L-ROM-Proj (green) and DNS (blue). Mean (left column) and time evolution (right column) of the $L^2$ norm of the solution; $r = 6$ (top row) and $r = 20$ (bottom row). The time evolution of the $L^2$ norm of the solution (right column) is plotted for the optimal mean $L^2$ norm of the solution (left column): $r_1 = 6$ for $r = 6$ (top row) and $r_1 = 15$ for $r = 20$ (bottom row).

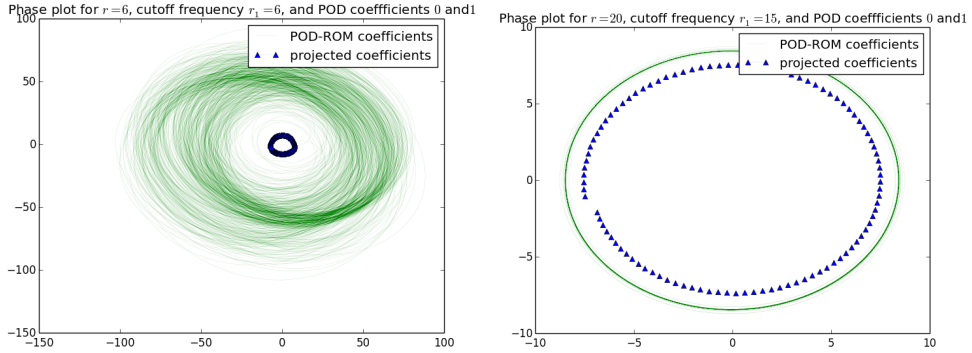


Figure 8.7: 3D flow past a cylinder, L-ROM-Proj with optimal $r_1$ value (green) and POD projection of DNS data (blue). Phase portraits for $a_1$ and $a_2$; $r = 6$ (left) and $r = 20$ (right).

### 8.4.9 Summary and Discussion

Table 8.1 displays the ranking of the four Reg-ROM/filter combinations (i.e., L-ROM-DF, L-ROM-Proj, EF-ROM-DF and EF-ROM-Proj) for the test problems' performance. The

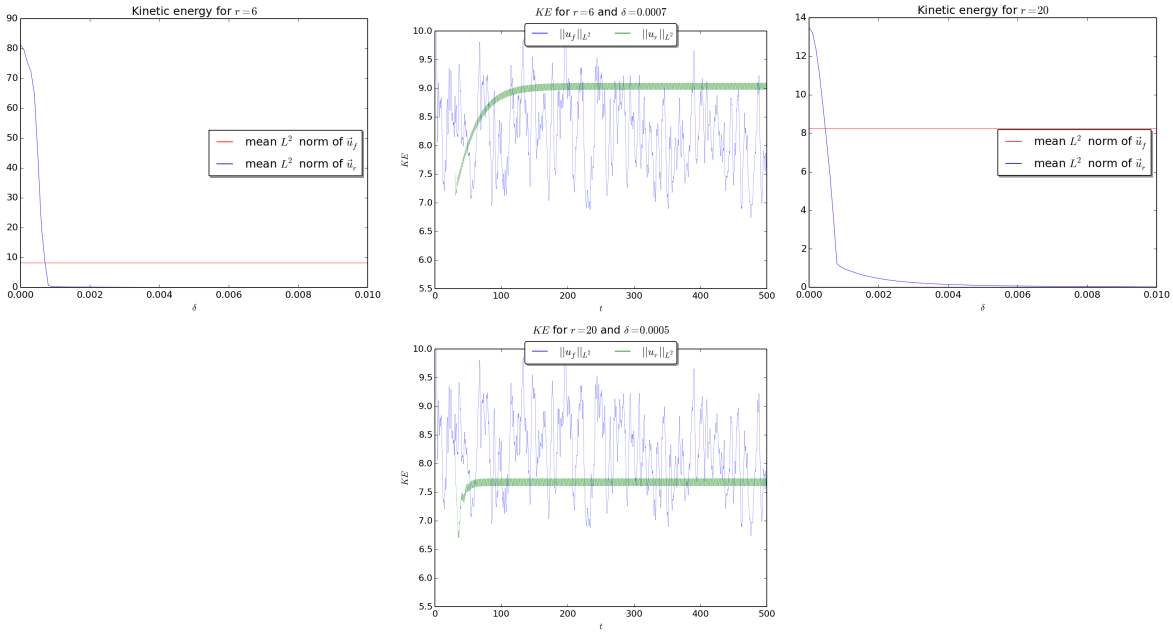


Figure 8.8: 3D flow past a cylinder, EF-ROM-DF (green) and DNS (blue). Mean (left column) and time evolution (right column) of the $L^2$ norm of the solution; $r = 6$ (top row) and $r = 20$ (bottom row). The time evolution of the $L^2$ norm of the solution (right column) is plotted for the optimal mean $L^2$ norm of the solution (left column): $\delta = 0.0005$ for $r = 6$ (top row) and $\delta = 0.0004$ for $r = 20$ (bottom row).

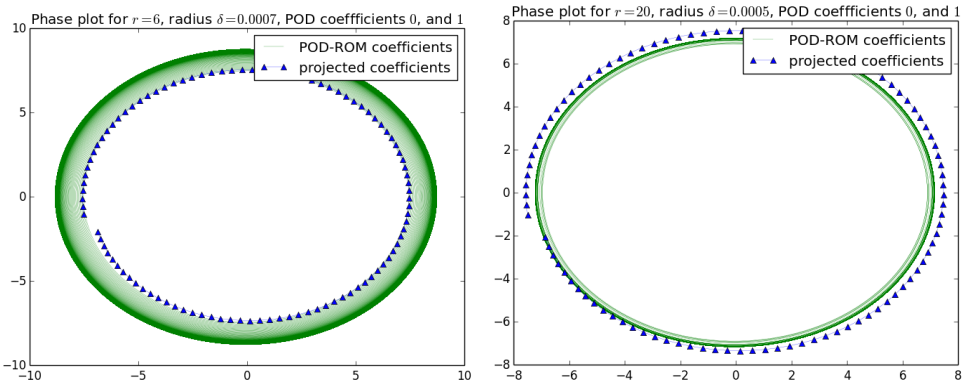


Figure 8.9: 3D flow past a cylinder, EF-ROM-DF with optimal $\delta$ value (green) and POD projection of DNS data (blue). Phase portraits for $a_1$ and $a_2$; $r = 6$ (left) and $r = 20$ (right).

numbers in Table 8.1 represent the rank of the Reg-ROM/filter combinations (with 1 the best and 4 the worst). It should be emphasized that the results in Table 8.1 represent a general evaluation of the Reg-ROM/filter combinations for all parameters (e.g., $r$, $\delta$ and $r_1$) and criteria (e.g., $L^2$ norm of the solution, $L^2$ norm of the error, and phase portraits) investigated.

For the 3D flow past a cylinder experiment (Section 3.2), the EF-ROM-DF clearly yields the most accurate approximation, both in terms of time evolution of the $L^2$ norm of the

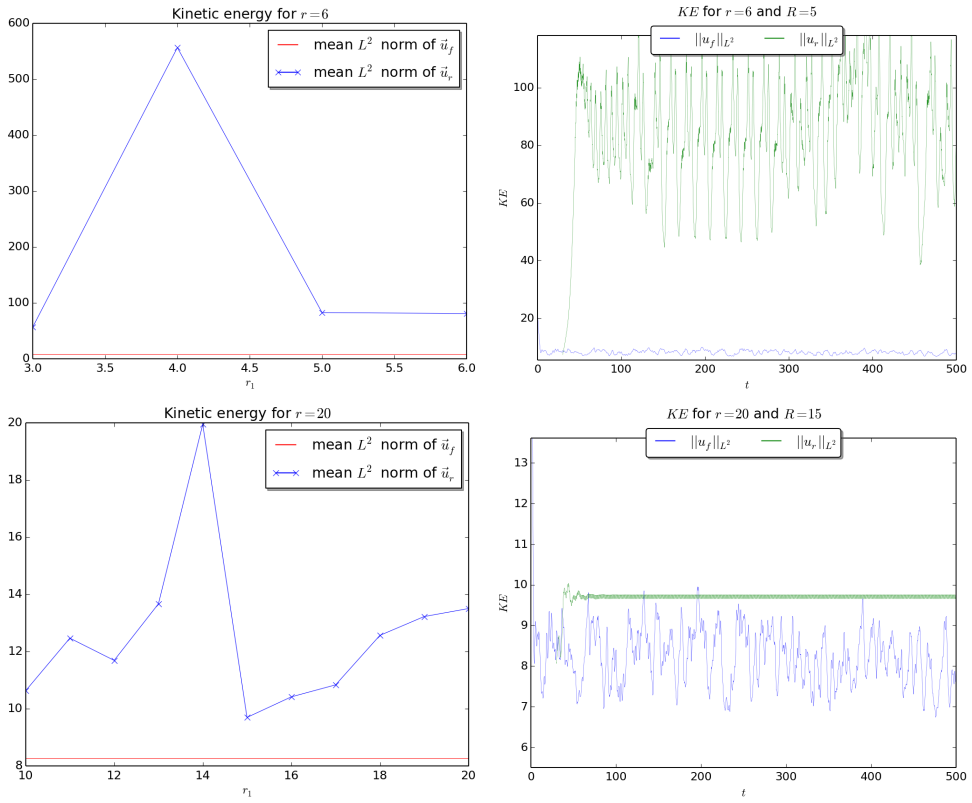


Figure 8.10: 3D flow past a cylinder, EF-ROM-Proj (green) and DNS (blue). Mean (left column) and time evolution (right column) of the $L^2$ norm of the solution; $r = 6$ (top row) and $r = 20$ (bottom row). The time evolution of the $L^2$ norm of the solution (right column) is plotted for the optimal mean $L^2$ norm of the solution (left column): $r_1 = 6$ for $r = 6$ (top row) and $r_1 = 15$ for $r = 20$ (bottom row).

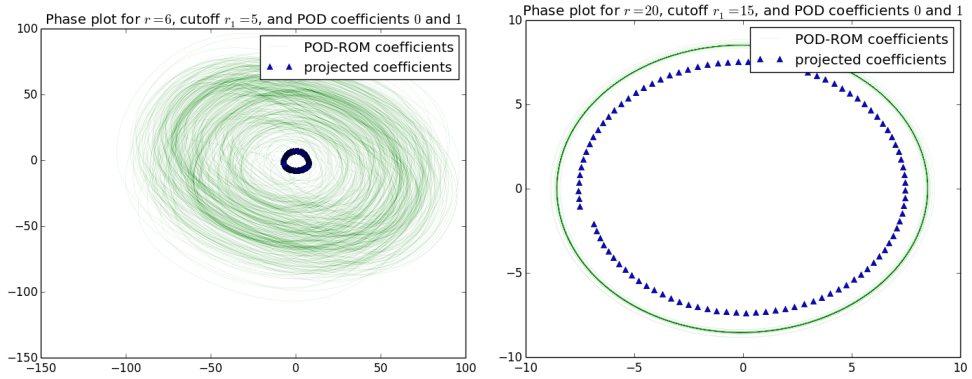


Figure 8.11: 3D flow past a cylinder, EF-ROM-Proj with optimal $r_1$ value (blue) and POD projection of DNS data (black). Phase portraits for $a_1$ and $a_2$; $r = 6$ (left) and $r = 20$ (right).

solution and phase portraits. The L-ROM-DF yields the second most accurate results. In terms of the $L^2$ norm of the solution, the L-ROM-DF and EF-ROM-DF results are

comparable. In terms of the phase portraits, however, the former is less accurate than the latter for $r = 6$. The L-ROM-Proj and the EF-ROM-Proj are the least accurate of the four Reg-ROM/filter combinations. Both the L-ROM-Proj and EF-ROM-Proj significantly over predict the time evolution of the $L^2$ norm of the solution, especially for $r = 6$. Furthermore, both the L-ROM-Proj and EF-ROM-Proj yield extremely inaccurate phase portraits for $r = 6$. With respect to the time evolution of the $L^2$ norm of the DNS solution, it should be noted that, although some Reg-ROM/filter combinations accurately approximate the mean, none can approximate the amplitude.

The EF-ROM-DF is consistently the most accurate with respect to all the criteria and parameters considered. The EF-ROM-Proj is consistently the least accurate. The L-ROM-DF and the L-ROM-Proj yield similar results, with a slight advantage for the former. Besides the rankings, the following general conclusions can be drawn from Table 8.1: First, the spatial filter seems to be more important than the actual Reg-ROM used: The Reg-ROMs using the DF are ranked first and second. Second, the above qualitative assessment is not sensitive with respect to the dimension of the POD space, $r$.

Finally, it should be emphasized that all four Reg-ROM combinations performed significantly better than the standard G-ROM.

Table 8.1: Ranking of the L-ROM-DF, L-ROM-Proj, EF-ROM-DF and EF-ROM-Proj based on the experiments in this chapter.

| | L-ROM-DF | L-ROM-Proj | EF-ROM-DF | EF-ROM-Proj |
|---|---|---|---|---|
| Ranking | **2** | **3** | **1** | **4** |

# Chapter 9

# Summary, Open Problems, & Future Work

## 9.1 Overview

This section has three goals. The first goal is to present a summary of the results in this thesis. The second goal is to outline several outstanding open problems in the research field of reduced order modeling of complex flows. The final goal is to briefly describe a few research directions and possible projects that I plan to pursue in the near future.

## 9.2 Summary

This thesis centered around two main directions: (i) the introduction of regularized models in the reduced order modeling field, and (ii) the use of numerical analysis to determine practical scalings for SUPG-ROMs. Next, these two developments are briefly summarized.

I introduced two REG-ROMs: the L-ROM and the EF-ROM. Both Reg-ROMs used explicit POD spatial filtering to regularize (smooth) some of the terms in the standard G-ROM. Two explicit POD spatial filters were proposed: a POD projection (Proj) on a subspace of the POD space and a POD differential filter (DF). To study the effect of the POD spatial filtering on the Reg-ROMs, four Reg-ROM/filter combinations were considered: L-ROM-Proj, L-ROM-DF, EF-ROM-Proj and EF-ROM-DF. These four Reg-ROM/filter combinations were assessed in the numerical simulation of the 3D flow past a circular cylinder at $Re = 100$. The Reg-ROM/filter combinations were tested with optimal values of $\delta$ (for the DF) and $r_1$ (for the Proj). These optimal values were determined by requiring that the DNS (benchmark) and Reg-ROM/filter combinations be as close as possible on the average. Two criteria were used in the numerical assessment of the Reg-ROM/filter combinations: the time evolution of the $L^2$ norm of the solution and the phase portraits. The numerical investigation of the four Reg-ROM/filter combinations yielded the following conclusions: The EF-ROM-DF was clearly the most accurate, the EF-ROM-Proj was the least accurate, and the L-ROM-DF and the L-ROM-Proj yielded similar results. The second conclusion was that the filter seemed to be more important than the actual Reg-ROM used: The Reg-ROMs using the DF were ranked first and second. The third conclusion was that the above

qualitative assessment was not sensitive with respect to the dimension of the POD space, $r$. Finally, it should be emphasized that all four Reg-ROM/filter combinations performed significantly better than the standard G-ROM. Furthermore, the CPU times of all four Reg-ROM/filter combinations were orders of magnitude lower than the CPU time of the DNS.

The second research development of this thesis is the use of numerical analysis to determine practical scalings for SUPG-ROMs. I used the SUPG finite element method on realistic meshes to compute snapshots; this lead to some noise in the POD data. I used numerical analysis to propose scalings for the stabilization parameter in the SUPG-ROM. I used two approaches for these scalings: one based on estimates from the underlying finite element discretization and another based on error equations for the POD truncation. I investigated both parameter scalings (as well as the G-ROM) on two convection-dominated CDR equation test problems. The SUPG-ROM generally yielded more accurate results than the standard G-ROM by eliminating some of the numerical artifacts in the data and dampening spurious oscillations. For realistic (coarse) finite element meshes, the finite element scaling for the SUPG-ROM produced more accurate results than the POD scaling.

## 9.3 Open Problems

The work presented in this thesis is part of our group's more general effort to investigate to what extent ROMs can be used for the numerical simulation of structure dominated turbulent flows. In this section, I present several open problems that, in my opinion, need to be answered to truly make ROMs practical tools for realistic flows. For each open problem, I also present my thoughts regarding possible answers (or lack thereof). Some of these questions have been circulating informally in the reduced order modeling community, but, to my knowledge, have not been stated explicitly. I hope that these open problems might serve as a guide to those who start investigating reduced order modeling of realistic flows.

**Open Problem 1**

| Can ROMs predict realistic structure dominated turbulent flows? |
|---|

This problem is the "Holy Grail" of our group. The other open problems are related to it; in particular, if the answer is "no", then Problems 3 and 4 will probably also be answered "no". In my opinion, a ROM that can predict structure dominated turbulent flows should function in challenging realistic settings such as closed loop control of flows at $Re = \mathcal{O}(10^6)$ (which is common in industrial settings) or data assimilation of large scale ocean flows. Although significant steps toward this goal have been made in the last few decades, several outstanding problems still remain.

I think that it is important to carefully choose the problems ROM is applied to. In particular, ROM should target turbulent flows in which organized structures play an important role, e.g., the boundary layer of turbulent pipe flows or channel flows (see [6, 53]) or the flow past a cylinder (see, e.g., [78, 117]). Homogeneous turbulent flows, on the other hand, would obviously not be an appropriate target for ROMs, since in this case the POD modes are nothing but the Fourier modes (see page 100 in [53]).

I also think that it is important to choose the features that ROM should predict. Qualitative features, such as the intermittency in the turbulent boundary layer studied in [6], could be realistic targets for ROMs. Quantitative features, such as the pointwise velocity field in a turbulent flow, would generally represent challenging goals for ROMs.

**Open Problem 2**

What are ROMs' limits?

This is an important question, since it determines what type of realistic settings ROMs are useful for. The answer to this question, however, is not straightforward, since it seems that it is a function of time.

For example, as mentioned in Section 1.4, the POD basis is limited by the snapshots from which it is computed (see Fig.1.1). While this has been for a long time one of the fundamental criticisms of ROMs, recent developments have shown that standard ROMs might be able to overcome this hurdle. Specifically, the *POD basis adaptivity*, i.e., adapting the POD basis to include features that were not present in the initial set of snapshots, was pursued in [3, 23, 36, 87].

Nonlinearities have also been considered for a long time to be a fundamental hurdle for ROMs. Specifically, nonlinearities that precluded the preassembly of ROM matrices and tensors (such as the nonlinearities in the REG-ROMs in Chapter 8) have been generally considered to be insurmountable for ROMs. In fact, I believe that one of the main reasons for using simplistic ROM closure models in [6] instead of more accurate ones (e.g., those in [117]) is that only simple nonlinearities could be treated at that time. The numerical discretization of more complex nonlinearities, however, has been made possible by recent developments, such as the empirical interpolation method (EIM) [14], the discrete interpolation method (DEIM) [26, 27] and the two-level method [116] (see also [5]).

**Open Problem 3**

Can ROMs be predictive?

That is, can the snapshots be generated on a time interval $[0, T]$ and the resulting ROMs be run on a much longer time interval ($[0, 100\,T]$, say)? I believe that the answer to this question is essential in developing ROMs for realistic turbulent flows (i.e., answering Open Problem 1 in the affirmative). Although several encouraging results [116, 117] have hinted at a positive answer to Open Problem 3, a complete answer is not known. I believe that the POD basis adaptivity [3, 23, 36, 87] will play an important role in answering this question.

**Open Problem 4**

To what extent is the energy cascade concept valid in POD?

As discussed in Section 1.5, there are a few encouraging results that suggest that some of the energy cascade concept is also valid in a POD context. The extent to which this is true, however, is not known. For example, it is well known [41, 92], that in a Fourier

setting the energy spectrum displays precise scaling laws that are universal, i.e., valid over a wide range of Reynolds numbers. A natural question is whether similar scaling laws exist in a POD setting, and if yes, whether these scaling laws are also valid over a range of Reynolds numbers. (Note that the numerical investigation in [33] considered only one Reynolds number.) Answering these questions is utterly important for the validity and range of applicability of the LES-ROMs and Reg-ROMs.

**Open Problem 5**

What is the effect of POD noisy data on the ROMs?

For the numerical simulation of realistic structure dominated turbulent flows, e.g., [84, 83], regularized and LES models are generally used to generate the snapshots employed in the POD basis construction [84, 83, 24]. Although the regularized and LES models yield relatively accurate data for these realistic applications, they inherently generate *noisy data*, i.e., data that displays numerical artifacts (noise). This, of course, is to be expected for relatively coarse meshes that are unavoidable in realistic applications. It was shown in [47], however, that running standard G-ROMs with noisy data yielded inaccurate results. Thus, addressing Open Problem 5 is essential for the development of ROMs that can eliminate the inherent noise induced by the POD modes, which, in turn, is needed in realistic settings.

**Open Problem 6**

What is the effect of the modeling consistency on the ROMs?

One of the challenges in coupling the data generation and the ROMs of realistic flows is the *modeling inconsistency*: the data generation uses regularized or LES models, whereas the ROMs usually do not. This modeling inconsistency was shown in [47] to yield inaccurate results for a convection-dominated problem. Specifically, a streamline-upwind/Petrov-Galerkin (SUPG) strategy was used to generate accurate data on a coarse mesh. Using this accurate data in a standard G-ROM, however, yielded extremely inaccurate results. I emphasize that the negative effect of the modeling inconsistency on the accuracy of the results was also observed for the reduced basis methods in [86]. The initial results in [47, 86] suggest that modeling consistency is beneficial to ROMs. A more thorough investigation is, however, needed.

**Open Problem 7**

What is the "right" spatial filter for LES-ROMs and Reg-ROMs?

Chapter 8 proposed two explicit POD spatial filters and described several natural approaches for their numerical discretization. Alternative POD spatial filters should, however, be investigated. Indeed, over the last few decades, several spatial filters have been developed for LES and regularized models [102, 17]. These and other spatial filters should be investigated in a POD context. Furthermore, different discretization methods for these POD spatial filters could also be considered.

**Open Problem 8**

Can rigorous mathematical support be provided for ROMs of realistic flows?

Error estimates for the standard G-ROM has been provided in several reports, starting with the pioneering work of Kunisch and Volkwein [69, 70, 71]. Similar results were provided for the RBM [112, 113, 93, 72, 79, 100, 101]. For ROM closure models, however, the mathematical support is scarce (see, e.g., [59, 61, 47] for a few exceptions). Error estimates for the new Reg-ROMs and LES-ROMs are needed. These results could yield useful scalings that can help guide the parameter choices in these ROMs (see, e.g., [59, 61, 47] for a few examples).

**Open Problem 9**

Are the answers to Open Problems 1-8 problem dependent?

I believe that some of the answers to Open Problems 1-8 might be problem dependent. For example, even if scaling laws for the energy spectrum in a POD setting were valid (see Open Problem 4), the scaling coefficients might be problem dependent. I do believe, however, that answering Open Problems 1-8 is still valuable, even though these answers might be problem dependent. Indeed, these answers could allow us to assemble a *ROM atlas* for various test problems that are representative for practical, realistic flows.

## 9.4 Future Work

There are several research directions that I would like to further pursue.

**SUPG** Future work in this direction includes studying the possibility of obtaining snapshots with fewer oscillations and to investigate if their availability is sufficient to obtain more accurate SUPG-ROM results than those obtained with this thesis' results.

**ROM Spatial Filter investigation** The radius of the DF (i.e., $\delta$) is calculated by calibration. Can one use phenomenology to determine the DF radius (just as in LES)? Several attempts have been made in [6, 117]. This investigation is related to Open Problems 4, 6, and 7. Alternatively, one could use error analysis (a more traditional option) to determine scalings. Error analysis for time-dependent, nonlinear POD-ROMs is a difficult area. Indeed, as evidenced by my work in Chapter 8, it may be more practical to investigate physical quantities, such as period and kinetic energy, instead of using more "traditional" numerical analysis.

**Approximate Deconvolution Leray ROM (AD-L-ROM)** A natural extension of the work in Section 8.2 is to apply a deconvolution operator to the filtered advective term. Dunca (see, e.g., [37]) used the model

$$\begin{aligned} \vec{w}_t + D(\bar{\vec{w}}) \cdot \nabla \vec{w} - \nu \Delta \vec{w} + \nabla q &= \bar{f} \\ \nabla \cdot w &= 0 \end{aligned} \tag{9.1}$$

and showed that it performs better than the standard Leray model. I will extend the model in Equation (9.1) to the ROM setting. I will investigate whether the resulting AD-L-ROM performs better than the L-ROM. As noted in [74], adding an extra step of deconvolution by the van Cittert deconvolution operator only requires an additional Poisson solve, which (in the case of the standard G-ROM with the nonlinearity handled by a tensor) has negligible computational cost.

**New Reg-ROMs** The regularized ROM presented in Section 8.2 is the simplest and oldest regularized modification of the NSEs. There are a wide variety of regularized models used in practice for finite elements, such as the NS-$\alpha$ model (see [74, 22, 29, 39] for more details)

$$\begin{aligned} \vec{w}_t - \mathcal{F}(\vec{w}) \times (\nabla \times \vec{w}) + \nabla Q - \frac{1}{Re}\Delta \vec{w} &= \vec{f}, \\ \nabla \cdot \mathcal{F}(\vec{w}) &= 0, \\ -\delta^2 \Delta \mathcal{F}(\vec{w}) + \mathcal{F}(\vec{w}) &= \vec{w}. \end{aligned} \tag{9.2}$$

Note that (unlike the L-ROM (8.5)) in this model the *filtered* value of the velocity field is incompressible (there was no such guarantee with the filtered POD vectors in the L-ROM). A related regularized model (without a time relaxation term) is the NS-$\omega$ model

$$\begin{aligned} \vec{w}_t + \vec{w} \times (\nabla \times D(\mathcal{F}(\vec{w}))) + \nabla Q - \frac{1}{Re}\Delta \vec{w} &= \vec{f}, \\ \nabla \cdot \vec{w} &= 0. \end{aligned} \tag{9.3}$$

This model (without viscosity and external forcing, and with periodic boundary conditions) preserves helicity.

I will extend both the NS-$\alpha$ and NS-$\omega$ models to the ROM setting. I will compare the two new Reg-ROMs with the L-ROM and EF-ROM in the numerical simulation of convection-dominated flows. OD-ROMs. One of the largest concerns with regularized models is the question of how to handle flows near walls, where the velocity has boundary layers that exhibit laminar behavior (see the discussion on this topic in [74]).

Another issue that needs investigation is the sensitivity of the EF-ROM to the time step. Since I filter after every time step, I expect that larger time steps would require lower filter radii to achieve the same result. The data shows that we need somewhat larger filter radii but that the scaling is not quite one-to-one (e.g., doubling the time step does not imply that one should double the filter radius).

**SUPG-ROM for the NSEs** I believe the natural extension of the work in Chapter 5 is applying a similar scheme to the NSEs. The numerical examples in Section 5.5 show promising results for resolving instabilities caused by fine layers and small scales of motion, which is certainly an issue with the NSEs (see the last part of [99]). The SUPG method has two strong advantages for ROMs:

1. As a consistent method, it does not rely on any physical or phenomenological arguments nor calibration for stabilization. As was discussed above, these are two open questions and sidestepping them would be helpful.

2. The method does not rely on complex nonlinear terms (such as the Smagorinsky model used in [116]) and does not require referencing the full-rank discretization in the ROM.

**Develop a `deal.II` ROM Library** Finally, as I alluded to in Algorithm 1, Figure 8.1, and Section A.3, I spent a large amount of time implementing ROMs on top of `deal.II`. I plan on making this effort publicly available; Dr. Heister (a principal author of `deal.II`) remarked to me that he only knew of one model reduction project based on `deal.II` and the implementation was very simple. `deal.II` is a sufficiently flexible library that it can be used for "unintended" work, such as model reduction, and I believe others would find such functionality useful.

# Bibliography


[1] A. K. Alekseev and I. M. Navon. The analysis of an ill-posed problem using multi-scale resolution and second-order adjoint techniques. *Comput. Meth. Appl. Mech. Eng.*, 190(15):1937–1953, 2001.

[2] D. Amsallem and C. Farhat. Stabilization of projection-based reduced-order models. *Int. J. Num. Meth. Eng.*, 91(4):358–377, 2012.

[3] D. Amsallem, M. J. Zahr, and C. Farhat. Nonlinear model order reduction based on local reduced-order bases. *Int. J. Num. Meth. Eng.*, 92(10):891–916, 2012.

[4] E. Anderson, Z. Bai, J. Dongarra, A. Greenbaum, A. McKenney, J. Du Croz, S. Hammerling, J. Demmel, C. Bischof, and D. Sorensen. Lapack: A portable linear algebra library for high-performance computers. In *Proceedings of the 1990 ACM/IEEE Conference on Supercomputing*, Supercomputing '90, pages 2–11, Los Alamitos, CA, USA, 1990. IEEE Computer Society Press.

[5] P. Astrid, S. Weiland, K. Willcox, and T. Backx. Missing point estimation in models described by proper orthogonal decomposition. *IEEE Trans. Autom. Control*, 53(10):2237–2251, 2008.

[6] N. Aubry, P. Holmes, J. L. Lumley, and E. Stone. The dynamics of coherent structures in the wall region of a turbulent boundary layer. *J. Fluid Mech.*, 192:115–173, 1988.

[7] N. Aubry, W. Y. Lian, and E. S. Titi. Preserving symmetries in the proper orthogonal decomposition. *SIAM J. Sci. Comput.*, 14:483–505, 1993.

[8] M. Balajewicz and E. H. Dowell. Stabilization of projection-based reduced order models of the Navier-Stokes. *Nonlinear Dynam.*, 70(2):1619–1632, 2012.

[9] M. J. Balajewicz, E. H. Dowell, and B. R. Noack. Low-dimensional modelling of high-Reynolds-number shear flows incorporating constraints from the Navier–Stokes equation. *J. Fluid Mech.*, 729:285–308, 2013.

[10] F. Ballarin. *Reduced-Order Models for Patient-Specific Haemodynamics of Coronary Bypass Grafts*. PhD thesis, Politecnico di Milano, 2015.

[11] F. Ballarin, A. Manzoni, A. Quarteroni, and G. Rozza. Supremizer stabilization of POD–Galerkin approximation of parametrized steady incompressible Navier–Stokes equations. *Int. J. Numer. Meth. Engng.*, 102:1136–1161, 2015.

[12] W. Bangerth, T. Heister, L. Heltai, G. Kanschat, M. Kronbichler, M. Maier, B. Turcksin, and T. D. Young. The `deal.ii` library, version 8.1. *arXiv preprint,* ***http://arxiv.org/abs/1312.2266v4***, 2013.

[13] M. F. Barone, I. Kalashnikova, D. J. Segalman, and H. K. Thornquist. Stable Galerkin reduced order models for linearized compressible flow. *J. Comput. Phys.*, 228(6):1932–1946, 2009.

[14] M. Barrault, Y. Maday, N. C. Nguyen, and A. T. Patera. An 'empirical interpolation' method: Application to efficient reduced-basis discretization of partial differential equations. *C. R. Acad. Sci. Paris, Ser. I*, 339:667–672, 2004.

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

[16] P. S. Berloff and J. C. McWilliams. Large-scale, low-frequency variability in wind-driven ocean gyres. *J. Phys. Oceanogr.*, 29:1925–1949, 1999.

[17] L. C. Berselli, T. Iliescu, and W. J. Layton. *Mathematics of large eddy simulation of turbulent flows*. Scientific Computation. Springer-Verlag, Berlin, 2006.

[18] J. Borggaard, A. Duggleby, A. Hay, T. Iliescu, and Z. Wang. Reduced-order modeling of turbulent flows. In *Proceedings of MTNS 2008*, 2008.

[19] A. Brooks and T. J. R. Hughes. Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. *Comput. Methods Appl. Mech. Engrg.*, 32(1-3):199–259, 1982. FENOMECH '81, Part I (Stuttgart, 1981).

[20] K. Bryan. A numerical investigation of a nonlinear model of a wind-driven ocean. *J. Atmos. Sci.*, 20:594–606, 1963.

[21] A. Caiazzo, T. Iliescu, V. John, and S. Schyschlowa. A numerical investigation of velocity-pressure reduced order models for incompressible flows. *J. Comput. Phys.*, 259:598–616, 2014.

[22] C. Cao, D. D. Holm, and E. S. Titi. On the Clark-$\alpha$ model of turbulence: global regularity and long-time dynamics. *J. Turbul.*, 6, 2005.

[23] K. Carlberg. Adaptive h-refinement for reduced-order models. *Int. J. Num. Meth. Eng.*, 102(5):1192–1210, 2015.

[24] K. Carlberg, M. Barone, and H. Antil. Galerkin v. discrete-optimal projection in nonlinear model reduction. *arXiv preprint arXiv:1504.03749*, 2015.

[25] W. Cazemier, R. Verstappen, and A.E.P. Veldman. Proper orthogonal decomposition and low-dimensional models for driven cavity flows. *Phys. Fluids*, 10:1685, 1998.

[26] S. Chaturantabut and D. C. Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM J. Sci. Comput.*, 32(5):2737–2764, 2010.

[27] S. Chaturantabut and D. C. Sorensen. A state space error estimate for POD-DEIM nonlinear model reduction. *SIAM J. Numer. Anal.*, 50:46–63, 2012.

[28] Y. Chen, T. Davis, W. Hager, and S. Rajamanickam. Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. *ACM Trans. Math. Softw.*, 35(3):22:1–22:14, October 2008.

[29] A. Cheskidov, E. Olson D. D. Holm, and E. S. Titi. On a Leray-$\alpha$ model of turbulence. *Proc. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci.*, 146:1–21, 2005.

[30] H. Childs, E. Brugger, K. Bonnell, J. Meredith, M. Miller, B. Whitlock, and N. Max. A contract-based system for large data visualization. In *Proceedings of IEEE Visualization 2005*, 2005.

[31] R Codina. Comparison of some finite element methods for solving the diffusion-convection-reaction equation. *Comput. Methods Appl. Mech. Engrg.*, 156:185–210, 1998.

[32] L. Cordier, B. Abou El Majd, and J. Favier. Calibration of POD reduced-order models using Tikhonov regularization. *Int. J. Num. Meth. Fluids*, 63(2):269–296, 2010.

[33] 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.

[34] J. W. Demmel. *Applied numerical linear algebra*. Society for Industrial and Applied Mathematics Philadelphia, 1997.

[35] M. O. Deville, P. F. Fischer, and E. H. Mund. *High-order methods for incompressible fluid flow*, volume 9 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, 2002.

[36] M. Drohmann, B. Haasdonk, and M. Ohlberger. Adaptive reduced basis methods for nonlinear convection–diffusion equations. In *Finite Volumes for Complex Applications VI Problems & Perspectives*, pages 369–377. Springer, 2011.

[37] A. Dunca and E. Epshteyn. On the Stolz-Adams deconvolution models for LES. Technical report, University of Pittsburgh, 2003.

[38] 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.

[39] C. Foiaş, D. D. Holm, and E. S. Titi. The Navier-Stokes-alpha model of fluid turbulence. *Phys. D*, 152/153:505–519, 2001. Advances in nonlinear mathematics and science.

[40] Matteo Frigo and Steven G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005. Special issue on "Program Generation, Optimization, and Platform Adaptation".

[41] U. Frisch. *Turbulence, The Legacy of A.N. Kolmogorov*. Cambridge University Press, Cambridge, 1995.

[42] W. L. Gates. A numerical study of transient Rossby waves in a wind-driven homogeneous ocean. *J. Atmos. Sci.*, 25, 1968.

[43] M. Germano. Differential filters for the large eddy numerical simulation of turbulent flows. *Phys. Fluids*, 29(6):1755–1757, 1986.

[44] M. Germano. Differential filters of elliptic type. *Phys. Fluids*, 29(6):1757–1758, 1986.

[45] M. Germano, U. Piomelli, P. Moin, and W.H. Cabot. A dynamic subgrid-scale eddy viscosity model. *Phys. Fluids A*, 3:1760–1765, 1991.

[46] C. Geuzaine and J.-F. Remacle. Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. *Int. J. Numer. Meth. Eng.*, 79:1309–1311, 2009.

[47] S. Giere, T. Iliescu, V. John, and D. Wells. SUPG reduced order models for convection-dominated convection-diffusion-reaction equations. *Comput. Methods Appl. Mech. Engrg.*, 289:454–474, 2015.

[48] W. R. Graham, J. Peraire, and K. Y. Tang. Optimal control of vortex shedding using low-order models. Part I – open-loop model development. *Int. J. Numer. Meth. Eng.*, 44(7):945–972, 1999.

[49] J.-L. Guermond, P. Minev, and J. Shen. Error analysis of pressure-correction schemes for the Navier-Stokes equations with open boundary conditions. *SIAM J. Numer. Anal.*, 43:239–258, 2005.

[50] J.-L. Guermond and L. Quartapelle. On the approximation of the unsteady Navier-Stokes equations by finite element projection methods. *Numer. Math.*, 80:207–238, 1998.

[51] I. Harari and T. J. R. Hughes. What are $C$ and $h$?: inequalities for the analysis and design of finite element methods. *Comput. Methods Appl. Mech. Engrg.*, 97(2):157–192, 1992.

[52] W. R. Holland and L. B. Lin. On the generation of mesoscale eddies and their contribution to the oceanic general circulation. i. a preliminary numerical experiment. *J. Phys. Oceanogr.*, 8, 1975.

[53] P. Holmes, J. L. Lumley, and G. Berkooz. *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*. Cambridge, 1996.

[54] T. J. R. Hughes and A. Brooks. A multidimensional upwind scheme with no crosswind diffusion. In *Finite element methods for convection dominated flows (Papers, Winter Ann. Meeting Amer. Soc. Mech. Engrs., New York, 1979)*, volume 34 of *AMD*, pages 19–35. Amer. Soc. Mech. Engrs. (ASME), New York, 1979.

[55] T. J. R. Hughes, L. Mazzei, and K. E. Jansen. Large eddy simulation and the variational multiscale method. *Comput. Vis. Sci.*, 3:47–59, 2000.

[56] T. J. R. Hughes, L. Mazzei, A. Oberai, and A. Wray. The multiscale formulation of large eddy simulation: Decay of homogeneous isotropic turbulence. *Phys. Fluids*, 13(2):505–512, 2001.

[57] T. J. R. Hughes, A. Oberai, and L. Mazzei. Large eddy simulation of turbulent channel flows by the variational multiscale method. *Phys. Fluids*, 13(6):1784–1799, 2001.

[58] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

[59] T. Iliescu and Z. Wang. Variational multiscale proper orthogonal decomposition: Convection-dominated convection-diffusion-reaction equations. *Math. Comput.*, 82(283):1357–1378, 2013.

[60] T. Iliescu and Z. Wang. Are the snapshot difference quotients needed in the proper orthogonal decomposition? *SIAM J. Sci. Comput.*, 36(3):A1221–A1250, 2014.

[61] T. Iliescu and Z. Wang. Variational multiscale proper orthogonal decomposition: Navier-Stokes equations. *Num. Meth. P.D.E.s*, 30(2):641–663, 2014.

[62] C. Jarvis. *Parameter Dependent Model Reduction for Complex Fluid Flows.* PhD thesis, Virginia Tech, 2014.

[63] S. Jiang and F. Jin. Multiple equilibria, periodic, and aperiodic solutions in a wind-driven, double-gyre, shallow-water model. *J. Phys. Oceanogr.*, 25, 1975.

[64] V. John and J. Novo. Error analysis of the SUPG finite element discretization of evolutionary convection-diffusion-reaction equations. *SIAM J. Numer. Anal.*, 49(3):1149–1176, 2011.

[65] I. Kalashnikova and M. F. Barone. On the stability and convergence of a Galerkin reduced order model (ROM) of compressible flow with solid wall and far-field boundary treatment. *Int. J. Num. Meth. Eng.*, 83(10):1345–1375, 2010.

[66] I. Kalashnikova, S. van Bloemen Waanders, B. Arunajatesan, and M. Barone. Stabilization of projection-based reduced order models for linear time-invariant systems via optimization-based eigenvalue reassignment. *Comput. Methods Appl. Mech. Engrg.*, 272(15):251–270, 2014.

[67] A.N. Kolmogorov. The local structure of turbulence in incompressible viscous fluids for very large Reynolds number. *Dokl. Akad. Nauk SSR*, 30:9–13, 1941.

[68] B. Kragel. *Streamline Diffusion POD Models in Optimization.* PhD thesis, University of Trier, 2005.

[69] K. Kunisch and S. Volkwein. Control of the Burgers equation by a reduced-order approach using proper orthogonal decomposition. *J. Optim. Theory Appl.*, 102(2):345–371, 1999.

[70] K. Kunisch and S. Volkwein. Galerkin proper orthogonal decomposition methods for parabolic problems. *Numer. Math.*, 90(1):117–148, 2001.

[71] K. Kunisch and S. Volkwein. Galerkin proper orthogonal decomposition methods for a general equation in fluid dynamics. *SIAM J. Numer. Anal.*, 40(2):492–515, 2002.

[72] T. Lassila, A. Manzoni, A. Quarteroni, and G. Rozza. Model order reduction in fluid dynamics: challenges and perspectives. In A. Quarteroni and G. Rozza, editors, *Reduced Order Methods for modeling and computational reduction*, volume 9 of *Modeling, Simulation and Applications*, pages 235–274. Springer, 2014.

[73] W. J. Layton. A Mathematical Introduction to Large Eddy Simulation. In H. Deconinck, editor, *Computational Fluid Dynamics-Multiscale Methods*. Von Karman Institute for Fluid Dynamics, Rhode-Saint-Genèse, Belgium, 2002.

[74] W. J. Layton and L. G. Rebholz. *Approximate Deconvolution Models of Turbulence*, volume 2042 of *Lecture Notes in Mathematics*. Springer, 2012.

[75] J. Leray. Sur le mouvement d'un liquide visqueux emplissant l'espace. *Acta Mathematica*, 63(1):193–248, 1934.

[76] C. López and E. Hernández-García. Low–dimensional dynamical system model for observed coherent structures in ocean satellite data. *Physical A: Statistical Mechanics and its Applications*, 328:233–250, 2008.

[77] H. V. Ly and H. T. Tran. Proper orthogonal decomposition for flow calculations and optimal control in a horizontal cvd reactor. Technical report, North Carolina State University, 1998.

[78] X. Ma and G. E. Karniadakis. A low-dimensional model for simulating three-dimensional cylinder flow. *J. Fluid Mech.*, 458:181–190, 2002.

[79] N. Nguyen, G. Rozza, and A. T. Patera. Reduced basis approximation and a posteriori error estimation for the time-dependent viscous Burgers' equation. *Calcolo*, 46(3):157–185, 2009.

[80] B. R. Noack, M. Morzynski, and G. Tadmor. *Reduced-Order Modelling for Flow Control*, volume 528. Springer Verlag, 2011.

[81] B. R. Noack, P. Papas, and P. A. Monkewitz. Low-dimensional Galerkin model of a laminar shear-layer. Technical Report 2002-01, École Polytechnique Fédérale de Lausanne, 2002.

[82] B. R. Noack, M. Schlegel, B. Ahlborn, G. Mutschke, M. Morzynski, P. Comte, and G. Tadmor. A finite-time thermodynamics of unsteady fluid flows. *J. Non-Equil. Thermody.*, 33(2):103–148, 2008.

[83] J. Östh. *Unsteady Numerical Simulations and Reduced-Order Modelling of Flows around Vehicles*. PhD thesis, Chalmers University of Technology, 2014.

[84] J. Östh, B. R Noack, S. Krajnović, D. Barros, and J. Borée. On the need for a nonlinear subscale turbulence term in POD models as exemplified for a high-Reynolds-number flow over an Ahmed body. *J. Fluid Mech.*, 747:518–544, 2014.

[85] T. M. Özgökmen and E. P. Chassignet. Emergence of inertial gyres in a two-layer quasigeostrophic ocean model. *J. Phys. Oceanogr.*, 28:461–484, 1998.

[86] P. Pacciarini and G. Rozza. Stabilized reduced basis method for parametrized advection–diffusion PDEs. *Comput. Meth. Appl. Mech. Eng.*, 274:1–18, 2014.

[87] B. Peherstorfer and K. Willcox. Online adaptive model reduction for nonlinear systems. Technical Report TR-14-1v2, MIT, 2014.

[88] B. Podvin. On the adequacy of the ten-dimensional model for the wall layer. *Phys. Fluids*, 13(1):210–224, 2001.

[89] B. Podvin. A proper-orthogonal-decomposition–based model for the wall layer of a turbulent channel flow. *Phys. Fluids*, 21:015111, 2009.

[90] B. Podvin and P. Le Quéré. Low-order models for the flow in a differentially heated cavity. *Phys. Fluids*, 13(11):3204–3214, 2001.

[91] B. Podvin and J. Lumley. A low-dimensional approach for the minimal flow unit. *J. Fluid Mech.*, 362:121–155, 1998.

[92] S.B. Pope. *Turbulent flows*. Cambridge University Press, Cambridge, 2000.

[93] A. Quarteroni, G. Rozza, and A. Manzoni. Certified reduced basis approximation for parametrized partial differential equations and applications. *Journal of Mathematics in Industry*, 1(1):1–49, 2011.

[94] T. Chacón Rebollo and R. Lewandowski. *Mathematical and numerical foundations of turbulence models and applications*. Springer, 2014.

[95] D. Rempfer. Investigations of boundary layer transition via galerkin projections on empirical eigenfunctions. *Phys. Fluids*, 8:175, 1996.

[96] D. Rempfer. On low-dimensional Galerkin models for fluid flow. *Theoretical and Computational Fluid Dynamics*, 14(2):75–88, 2000.

[97] D. Rempfer and H. F. Fasel. Dynamics of three-dimensional coherent structures in a flat-plate boundary layer. *J. Fluid Mech.*, 275:257–283, 1994.

[98] L.F. Richardson. *Weather Prediction by Numerical Process*. Cambridge University Press, Cambridge, 1922.

[99] H. G. Roos, M. Stynes, and L. Tobiska. *Robust Numerical Methods for Singularly Perturbed Differential Equations: Convection-Diffusion-Reaction and Flow Problems.*, volume 24 of *Springer Series in Computational Mathematics*. Springer, second edition, 2008.

[100] G. Rozza, D. B. P. Huynh, and A. T. Patera. Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations. *Arch. Comput. Method. E.*, 15(3):229–275, 2008.

[101] G. Rozza and K. Veroy. On the stability of the reduced basis method for Stokes equations in parametrized domains. *Comput. Methods Appl. Mech. Engrg.*, 196:1244–1260, 2007.

[102] P. Sagaut. *Large eddy simulation for incompressible flows*. Scientific Computation. Springer-Verlag, Berlin, third edition, 2006.

[103] O. San, A. E. Staples, Z. Wang, and T. Iliescu. Approximate deconvolution large eddy simulation of a barotropic ocean circulation model. *Ocean Modelling*, 40:120–132, 2011.

[104] J. R. Singler. New POD error expressions, error bounds, and asymptotic results for reduced order models of parabolic PDEs. *SIAM J. Numer. Anal.*, 52(2):852–876, 2014.

[105] S. Sirisup and G. E. Karniadakis. A spectral viscosity method for correcting the long-term behavior of POD models. *J. Comput. Phys.*, 194(1):92–116, 2004.

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

[107] W. A. Stein et al. *Sage Mathematics Software (Version 6.2)*. The Sage Development Team, 2014. http://www.sagemath.org.

[108] P. Sura, K. Fraedrich, and F. Lunkeit. Regime transitions in a stochastically forced double-gyre model. *J. Phys. Oceanogr.*, 31:411–426, 2009.

[109] E. Tadmor. Convergence of spectral methods for nonlinear conservation laws. *SIAM J. Numer. Anal.*, pages 30–44, 1989.

[110] Y. Tanaka and K Akitomo. Alternating zonal flows in a two-layer wind-driven ocean. *J. Oceanogr.*, 66:475–487, 2010.

[111] S. Ullmann and J. Lang. A POD-Galerkin reduced model with updated coefficients for Smagorinsky LES. In J. C. F. Pereira and A. Sequeira, editors, *V European Conference on Computational Fluid Dynamics, ECCOMAS CFD 2010*, Lisbon, Portugal, June 2010.

[112] K. Urban and A. T. Patera. A new error bound for reduced basis approximation of parabolic partial differential equations. *C. R. Acad. Sci. Paris Sér. I Math.*, 350(3):203–207, 2012.

[113] K. Veroy and A. T. Patera. Certified real-time solution of the parametrized steady incompressible Navier–Stokes equations: rigorous reduced-basis a posteriori error bounds. *Int. J. Numer. Meth. Fluids*, 47(8-9):773–788, 2005.

[114] S. Volkwein. Model reduction using proper orthogonal decomposition. *Lecture Notes, Faculty of Mathematics and Statistics, University of Konstanz*, 2011.

[115] S. Volkwein. Proper orthogonal decomposition: Theory and reduced-order modelling. *Lecture Notes, Department of Mathematics and Statistics, University of Konstanz*, 2013.

[116] Z. Wang, I. Akhtar, J. Borggaard, and T. Iliescu. Two-level discretizations of nonlinear closure models for proper orthogonal decomposition. *J. Comput. Phys.*, 230:126–146, 2011.

[117] 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.

[118] J. Weller, E. Lombardi, and A. Iollo. Robust model identification of actuated vortex wakes. *Phys. D*, 238(4):416–427, 2009.

# Appendix A

# Source Code

## A.1 Sage Code

### A.1.1 The POD Option

```
#!/usr/bin/env python
# -*- encoding: utf-8 -*-
"""
This script solves the optimization problem implied by 'The POD Option'
for minimizing the error in the SUPG-POD-ROM for a CDR problem.
"""
from __future__ import print_function
import sage.all as sg

def main():
    eps = sg.var('epsilon')
    l0, l1, l2 = sg.var(['Lambda_0', 'Lambda_1', 'Lambda_2'])
    delta = sg.var('delta')
    objective = (sg.sqrt(eps)*l1 + (1 + 1/sg.sqrt(delta))*l0
                 + sg.sqrt(delta)*(eps*l2 + l1 + l0))
    answer = sg.solve(objective.diff(delta) == 0, delta)[0]
    print(answer)
    print(sg.latex(answer))

if __name__ == '__main__':
    main()
```

### A.1.2 The FE Option

```
#!/usr/bin/env python
# -*- encoding: utf-8 -*-
"""
This script solves the optimization problem implied by 'The FE Option'
for minimizing the error in the SUPG-POD-ROM for a CDR problem.
"""
from __future__ import print_function
import sage.all as sg

```

```
def main():
    h = sg.var('h')
    eps = sg.var('epsilon')
    l0 = sg.var('Lambda_0')
    delta = sg.var('delta')
    m = sg.var('m')

    estimate_0 = h**(m + sg.Rational((1, 2))) + l0
    estimate_1 = h**(m - sg.Rational((1, 2))) + l0/h
    estimate_2 = h**(m - sg.Rational((3, 2))) + l0/h**2

    objective = (sg.sqrt(eps)*estimate_1
                 + (1 + 1/sg.sqrt(delta))*estimate_0
                 + sg.sqrt(delta)*(eps*estimate_2 + estimate_1 + estimate_0))
    answer = sg.simplify(sg.solve(objective.diff(delta) == 0, delta))[0]
    print(answer)
    print(sg.latex(answer))

if __name__ == '__main__':
    main()
```

## A.2 Python Code

### A.2.1 Computing the POD Basis

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import numpy as np
import scipy.sparse.linalg as splg
import scikits.sparse.cholmod as chol
"""
Functions for dealing with POD in Python.
"""


def basis(snapshots, mass, num_vectors=10):
    """
    Compute the POD basis from snapshots, where the POD vectors are
    orthogonal under the inner product induced by the matrix `mass`.

    For numerical stability, the POD basis is computed from the
    'centered trajectory'; that is, the mean of the data is subtracted
    off and returned separately.

    Required Arguments
    ------------------
    snapshots:
        The matrix of snapshots. Each column corresponds to one
        snapshot.

    mass:
        Matrix inducing an inner product; the POD basis will be
```

```
        orthogonal under said inner product.

    Optional Arguments
    ------------------
    num_vectors:
        Number of POD vectors to compute in the basis. Defaults to 10.
    """
    mean_vector = np.sum(snapshots, axis=1)/float(snapshots.shape[1])
    centered_data = np.zeros_like(snapshots)
    for index, column in enumerate(snapshots.T):
        centered_data[:, index] = column - mean
    _cholesky_decomposition = chol.cholesky(sp.csc_matrix(mass))
    _cholesky_factor = cholesky_decomposition.L()
    cholesky_factor = _cholesky_factor[np.argsort(cholesky_decomposition.P()), :]
    pod_data = (cholesky_factor.T).dot(centered_data)

    # use the dense solver if we need every vector.
    if pod_data.shape[0] == num_vectors:
        # don't compute extra columns in U and V.T (see svd documentation)
        use_full = (num_vectors >= pod_data.shape[1])
        s_vectors, s_values, _ = np.linalg.svd(pod_data, full_matrices=use_full)
        s_vectors = np.ascontiguousarray(s_vectors[:, :num_vectors])
        s_values = np.ascontiguousarray(s_values[:num_vectors])
    else:
        s_vectors, s_values, _ = splg.svds(pod_data, k=num_vectors)
        s_values = s_values[::-1]
        s_vectors = s_vectors[:, ::-1]

    pod_vectors = np.zeros(s_vectors.shape)
    for index, s_vector in enumerate(s_vectors.T):
        pod_vectors[:, index] = splg.spsolve(cholesky_factor.T, s_vector)

    return pod_vectors, mean_vector, s_values
```

## A.3 C++ Code

### A.3.1 The Method of Snapshots

```
using namespace dealii;
void method_of_snapshots(const SparseMatrix<double>     &mass_matrix,
                         const std::vector<std::string> &snapshot_file_names,
                         const unsigned int             n_pod_vectors,
                         const bool                     center_trajectory,
                         BlockPODBasis                  &pod_basis)
{
  BlockVector<double> block_vector;
  std::vector<BlockVector<double>> snapshots;

  const double mean_weight = 1.0/snapshot_file_names.size();
  bool pod_basis_initialized = false;
  const unsigned int n_snapshots = snapshot_file_names.size();
  unsigned int n_dofs_per_block = 0;
```

```
15    unsigned int n_blocks = 0;
16    unsigned int i = 0;
17
18    for (auto &snapshot_file_name : snapshot_file_names)
19      {
20        H5::load_block_vector(snapshot_file_name, block_vector);
21        if (!pod_basis_initialized)
22          {
23            n_blocks = block_vector.n_blocks();
24            Assert(n_blocks > 0, ExcInternalError());
25            n_dofs_per_block = block_vector.block(0).size();
26            pod_basis.reinit(n_blocks, n_dofs_per_block);
27            pod_basis_initialized = true;
28          }
29        Assert(block_vector.n_blocks() == n_blocks, ExcIO());
30        pod_basis.mean_vector.add(mean_weight, block_vector);
31        snapshots.push_back(std::move(block_vector));
32      }
33
34    if (center_trajectory)
35      {
36        for (auto &snapshot : snapshots)
37          {
38            snapshot.add(-1.0, pod_basis.mean_vector);
39          }
40      }
41    else
42      {
43        pod_basis.mean_vector = 0.0;
44      }
45
46    LAPACKFullMatrix<double> correlation_matrix(n_snapshots);
47    LAPACKFullMatrix<double> identity(n_snapshots);
48    identity = 0.0;
49    BlockVector<double> temp(n_blocks, n_dofs_per_block);
50    for (unsigned int row = 0; row < n_snapshots; ++row)
51      {
52        for (unsigned int block_n = 0; block_n < n_blocks; ++block_n)
53          {
54            mass_matrix.vmult(temp.block(block_n), snapshots.at(row).block(block_n));
55          }
56        for (unsigned int column = 0; column <= row; ++column)
57          {
58            double value = temp * snapshots.at(column);
59            correlation_matrix(row, column) = value;
60            correlation_matrix(column, row) = value;
61          }
62        identity(row, row) = 1.0;
63      }
64
65    std::vector<Vector<double>> eigenvectors(n_snapshots);
66    correlation_matrix.compute_generalized_eigenvalues_symmetric(identity,
67                                                                 eigenvectors);
68    pod_basis.singular_values.resize(n_snapshots);
```

```
69    for (i = 0; i < n_snapshots; ++i)
70      {
71        // As the matrix has provably positive real eigenvalues...
72        auto eigenvalue = correlation_matrix.eigenvalue(i);
73        Assert(eigenvalue.imag() == 0.0, ExcInternalError());
74        Assert(eigenvalue.real() > 0.0, ExcInternalError());
75        pod_basis.singular_values.at(i) = std::sqrt(eigenvalue.real());
76      }
77    std::reverse(eigenvectors.begin(), eigenvectors.end());
78    std::reverse(pod_basis.singular_values.begin(), pod_basis.singular_values.end());
79
80    const unsigned int n_actual_pod_vectors = std::min(n_snapshots, n_pod_vectors);
81    pod_basis.vectors.resize(n_actual_pod_vectors);
82  #pragma omp parallel for
83    for (unsigned int eigenvector_n = 0; eigenvector_n < n_actual_pod_vectors;
84         ++eigenvector_n)
85      {
86        BlockVector<double> pod_vector(n_blocks, n_dofs_per_block);
87        pod_vector.collect_sizes();
88        pod_vector = 0;
89        auto &eigenvector = eigenvectors.at(eigenvector_n);
90        auto singular_value = pod_basis.singular_values.at(eigenvector_n);
91
92        unsigned int snapshot_n = 0;
93        for (auto &snapshot : snapshots)
94          {
95            if (!std::isnan(eigenvector[snapshot_n]) && !std::isnan(singular_value))
96              {
97                pod_vector.add(eigenvector[snapshot_n], snapshot);
98              }
99            ++snapshot_n;
100         }
101       pod_basis.vectors.at(eigenvector_n) = std::move(pod_vector);
102     }
103
104   for (unsigned int pod_vector_n = 0; pod_vector_n < n_pod_vectors; ++pod_vector_n)
105     {
106       auto &singular_value = pod_basis.singular_values.at(pod_vector_n);
107       if (!std::isnan(singular_value))
108         {
109           pod_basis.vectors.at(pod_vector_n) *= 1.0/singular_value;
110         }
111     }
112 }
```