Trabajo Fin de Máster:

Desarrollo y análisis de un modelo de orden reducido POD con viscosidad artificial. Aplicaciones en mecánica de fluidos

Carlos Núñez Fernández
Máster Univesitario en Matemáticas

Tutorizado por:
Samuele Rubino
Febrero 2022

DPTO. ECUACIONES DIFERENCIALES Y ANÁLISIS NUMÉRICO - US

Resumen

La simulación de flujos de fluídos con altos números de Reynolds (régimen de convección dominante, turbulento) sigue siendo un área de investigación muy activa. La resolución precisa y eficiente al mismo tiempo de las ecuaciones en derivadas parciales subyacentes puede suponer un verdadero desafío. Por esta razón, la finalidad de este trabajo es proponer Modelos de Orden Reducido (MOR), de tipo Proper Orthogonal Descomposition (POD), para tratar este problema.

En particular, en este trabajo se expone un estudio teórico del método POD y se propone un modelo de orden reducido POD con viscosidad artifical. Este método se aplicará en primera instancia a la ecuación de Burgers con coeficientes de difusión muy pequeños. En este contexto, para el método propuesto (AV-POD-G-ROM) se derivan las estimaciones de error correspondientes y se realizan simulaciones numéricas. En segundo lugar, se extiende este método a las ecuaciones de Navier-Stokes con viscosidades pequeñas, dando lugar a un modelo de orden reducido POD de tipo Smagorinsky (S-POD-G-ROM). En este contexto, se realizan simulaciones numéricas del flujo no estacionario alrededor de un obstáculo.

Abstract

The numerical simulation of fluid flows at high Reynolds numbers (dominant convection, turbulent regime) is a very active research area. The precise and efficient resolution of the corresponding partial derivative equations can be a real challenge. For this reason, the goal of this work is to propose Reduced Order Models (ROMs), based on Proper Orthogonal Decomposition (POD), to treat this problem.

In particular, in this work we show a theoretical study of the POD method and propose a POD reduced order model with artificial viscosity. This method will be applied in first instance to the Burgers equation with small diffusion coefficients. In this context, for the proposed method (AV-POD-G-ROM) the corresponding error estimates are derived and numerical simulations will be carried out. Secondly, this method will extend to the Navier-Stokes equations with small viscosity, leading to a Smagorinsky POD reduced order model (S-POD-G-ROM). In this context, numerical simulations of unsteady flow around an obstacle will be carried out.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Nowadays, mathematics and numerical simulations have an important role on the design and optimization of industrial processes, specifically on the energy sector that involves fluid mechanics problems. Fluid mechanics problems are usually governed by the Navier-Stokes equations, which describe the fluid flow, possibly combined with other equations (e.g., energy equation to obtain the so-called Boussinesq equations). These types of problem can describe the combustion in cars engine, wind turbines and solar thermal receivers. However, the accurate resolution of these problems with Full Order Models (FOMs) such as the Finite Element Method (FEM) normally requires a high computational cost. To this purpose, Reduced Order Models (ROMs) appeared in the recent scientific literature with the main aim of reducing computational cost while maintaining a similar accuracy with respect to FOMs.

The main advantage of ROMs is that they drastically reduce the degrees of freedom of FOMs, with a consistent saving in computing time, and at the same time they guarantee similar error levels. Research on ROMs is growing faster since the beginning of this century and much faster algorithms appeared mainly based on RB-Reduced Basis [23] and POD-Proper Orthogonal Descomposition [30].

In this work, we will focus on ROMs based on the POD method, and specifically on POD-closure models to solve non-linear Partial Differential Equations (PDEs) in fluid mechanics. The POD method essentially provides a low-dimensional orthonormal basis for representing a given set of data in a certain least-squares optimal sense. Originally, the POD was introduced as a spectral analysis method, which was presented by K. Pearson [10] in 1901. Nowadays, the most common version of the POD method is the method of snapshots that was presented first by Sirovich [27] in 1987. This method has been succesfully applied to numerous fields such as pattern recognition, statics and geophysicial fluid dynamics. From that moment forth, numerical methods based on POD for PDEs underwent some rapid development. For instance, Kunisch and Volkwein applied a POD-Galerkin ROM

for the numerical solution of parabolic PDEs, presenting also the corresponding error estimates in [19]. Their work has been extended in several papers to more general PDEs, see e.g. [31] for a recent review paper.

In this work, we present the POD-ROM for Burgers and Navier-Stokes equations, by considering a specific closure model. In Chapter (2), we enumerate some notations and fundamental tools used in the rest of the work. Before applying the POD method to the previous equations, we first introduce it as a method in R^m in Chapter (3), and we consider its relation with the Singular Value Descomposition (SVD) and application to image compression. We also consider the POD method with weighted inner product, useful when applying it to PDEs. Moreover, we introduce several methods that allow the treatment of non-linear and non-affine terms (with respect to the parameter) within the POD method, such as the Discrete Empirical Interpolation Method (DEIM) or a method based on Radial Basis Functions (RBF). In particular, the former method will be used in this work to approximate the non-linear closure model. In Chapter (4), we introduce a POD-Galerkin ROM for the Burgers equation together with an artificial viscosity closure model. We perform its numerical analysis, by mainly deriving error estimates, and we consider its practical implementation and corresponding numerical results. In Chapter (5), we introduce a POD-Galerkin ROM for the Navier-Stokes equations together with the Smagorinsky modeling of the eddy viscosity. We perform numerical studies on a two dimensional unsteady flow around a cylinder. Finally, in Chapter (6), we state some conclusions and future research directions.

Chapter 2

Notation and preliminary results

In this chapter, we will briefly introduce some notation and results that we will be used through this work.

2.1 Notation

Let $\Omega \subset \mathbb{R}^d$, with $d \in \{2, 3\}$, be a open domain with Lipschitz boundary $\partial\Omega$. Hereafter, we are going to consider the following notation:

1. Partial derivatives:

$$\frac{\partial}{\partial x_k}(\cdot) = \partial_k(\cdot) \quad 1 \leq k \leq d.$$

$$\frac{\partial}{\partial t}(\cdot) = \partial_t(\cdot).$$

$$\frac{\partial^2}{\partial x_k \partial x_l}(\cdot) = \partial_{lk}(\cdot) \quad 1 \leq k, l \leq d.$$

2. Gradient of a scalar-valued function $v : \mathbb{R}^d \to \mathbb{R}$:

$$\nabla v = \begin{pmatrix} \partial_1 v \\ \vdots \\ \partial_d v \end{pmatrix}.$$

3. Gradient of a vector-valued function $v : \mathbb{R}^d \to \mathbb{R}^d$:

$$\nabla v = \begin{pmatrix} \partial_1 v_1 & \cdots & \partial_d v_1 \\ \vdots & \ddots & \vdots \\ \partial_1 v_d & \cdots & \partial_d v_d \end{pmatrix}.$$

4. Divergence of a vector function $v : \mathbb{R}^d \to \mathbb{R}^d$:

$$\nabla \cdot v = \sum_{k=1}^{d} \partial_k v_k.$$

5. Divergence of a tensor-valued function $\sigma : \mathbb{R}^d \to \mathbb{R}^{d\times d}$:

$$\nabla \cdot \sigma = \begin{pmatrix} \sum_{k=1}^{d} \partial_k \sigma_{1k} \\ \vdots \\ \sum_{k=1}^{d} \partial_k \sigma_{dk} \end{pmatrix}.$$

6. Laplace operator of a scalar-valued function $v : \mathbb{R}^d \to \mathbb{R}$:

$$\Delta v = \nabla \cdot \nabla v = \sum_{k=1}^{d} \partial_{kk} v.$$

7. Convection term of a vector-valued function $v : \mathbb{R}^d \to \mathbb{R}^d$, velocity of the Navier-Stokes equation:

$$(v \cdot \nabla)v = \begin{pmatrix} v \cdot \nabla v_1 \\ \vdots \\ v \cdot \nabla v_d \end{pmatrix} = \begin{pmatrix} \sum_{k=1}^{d} (\partial_k v_1) v_k \\ \vdots \\ \sum_{k=1}^{d} (\partial_k v_d) v_k \end{pmatrix}.$$

2.2 Preliminary results

In this dissertation, we work with several function spaces and we use its associated norms, so we are going to introduce them.

Let $1 \leq p \leq \infty$, the Lebesgue spaces are defined as:

$$L^p(\Omega) = \{f : \Omega \to \mathbb{R} / f \text{ measurable}, \|f\|_{L^p(\Omega)} < \infty\}.$$

The norm for a function $u : \Omega \to \mathbb{R}$ of $L^p(\Omega)$ is

$$\|u\|_{L^p(\Omega)} = (\int_{\Omega} |u(x)|^p dx)^{1/p} \qquad \text{with } 1 \leq p < \infty,$$

$$\|u\|_{L^\infty(\Omega)} = \text{ ess sup }_{x\in\Omega} |u(x)|.$$

Lebesgue spaces are Banach spaces and for $p = 2$ we get the Hilbert space $L^2(\Omega)$ with inner product:

$$(u,v)_{L^2(\Omega)} = \int_\Omega u(x)v(x)dx.$$

Normally, we will use the following notation for the L^2-inner product

$$(\cdot,\cdot) := (\cdot,\cdot)_{L^2(\Omega)}.$$

Sobolev spaces too play an important role in the analysis of PDEs. In this work, we use:

$$H^1(\Omega) = \{f \in L^2(\Omega)/\nabla f \in [L^2(\Omega)]^d\},$$

and when working with homogeneous Dirichlet boundary conditions, we consider:

$$H_0^1(\Omega) = \{f \in H^1(\Omega)/f|_{\partial\Omega} = 0\}.$$

This is a closed linear subspace of H^1 and thus a Hilbert space endowed with the H^1-norm ($\|u\|_{H^1}$). Thanks to Poincarè inequality, the H^1-norm is equivalent on H_0^1 to the norm $\|u\|_{H_0^1} = \|\nabla u\|_{L^2}$.

Finally, we introduce the Bochner spaces, needed for the analysis of time-dependent PDEs [9]. For $1 \leq p \leq \infty$, the Bochner space is defined as:

$$L^p((0,T),\Omega) = \{f : (0,T)\times\Omega \to \mathbb{R}/f \text{ Bochner measurable}, \|f\|_{L^p((0,T),\Omega)} < \infty\},$$

where for a function $u : (0,T)\times\Omega \to \mathbb{R}$, its associated norm is:

$$\|u\|_{L^p((0,T),\Omega)} = (\int_0^T \|u(t)\|^p_{L^p(\Omega)}dt)^{1/p} \text{ with } 1 \leq p < \infty,$$

$$\|u\|_{L^\infty((0,T),\Omega)} = \text{ ess sup }_{t\in(0,T)}\|u(t)\|_{L^\infty(\Omega)}.$$

Chapter 3

The POD method in $\mathbb{R}^m$ space

In this chapter, we introduce the Proper Orthogonal Decomposition (POD) method in $\mathbb{R}^m$. This method seeks a proper orthonormal basis called POD basis $\{\varphi_i\}_{i=1}^r$ of rank r, which preserve the essential information of n vectors $y_1, \ldots, y_n \in \mathbb{R}^m$ called snapshots, with $r \leq \min\{m, n\}$. Snapshots constitute a high dimensional data set that normally comes from a Direct Numerical Simulations(DNS) [10, 27]. The POD method is formulated as constrained optimization problem:

$$\max_{\varphi_1,\ldots,\varphi_r \in \mathbb{R}^m} \sum_{i=1}^{r}\sum_{j=1}^{n} (y_j, \varphi_i)_{\mathbb{R}^m} \quad \text{subject to} \quad (\varphi_i, \varphi_j)_{\mathbb{R}^m} = \delta_{ij} \quad 1 \leq i, j \leq r, \tag{3.1}$$

where δ_{ij} is the Kronecker delta and $(a, b)_{\mathbb{R}^m} = \sum_{i=1}^m a_i b_i$.
In this chapter, we are going to introduce the Singular Value Descomposition (SVD) and the connection with the POD, and some properties of the POD basis. This part follows [30]. Afterward, we consider a widely used application of the SVD method, such as image compression. We introduce the POD method with weighted inner product, useful when applying it to PDEs. Finally, we consider the treatment of non-linear and non-affine terms (with respect to the parameter) within the POD.

3.1 Singular Value Decomposition

Let $S = (y_1|\cdots|y_n) \in \mathbb{R}^{m\times n}$ be a real matrix of rank $d \leq \min(m, n)$, usually called the snapshots matrix. The Singular Value Decomposition (SVD) of S guarantees that there exists orthogonal matrices:

$$\Phi = [\varphi_1|\cdots|\varphi_m] \in \mathbb{R}^{m\times m} \quad \text{and} \quad V = [v_1|\ldots|v_n] \in \mathbb{R}^{n\times n} \tag{3.2}$$

such that

$$\Phi^T S V = \Sigma := \begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix} \in \mathbb{R}^{m\times n}, \tag{3.3}$$

where $D = \operatorname{diag}(\sigma_1, \ldots, \sigma_d) \in \mathbb{R}^{d\times d}$, with $\sigma_1 > \ldots > \sigma_d > 0$.(see [11]). Moreover the vectors $\{\varphi_i\}_{i=1}^d$ and $\{v_i\}_{i=1}^d$ satisfy

$$S\varphi_i = \sigma_i v_i \quad \text{and} \quad S^T v_i = \sigma_i \varphi_i \quad \text{for } i = 1, \ldots, d, \tag{3.4}$$

so

$$S^T S\varphi_i = \sigma_i S^T v_i = \sigma_i^2 \varphi_i \quad \forall i = 1, \ldots, d, \tag{3.5}$$

$$SS^T \varphi_i = \sigma_i S\varphi_i = \sigma_i^2 v_i \quad \forall i = 1, \ldots, d. \tag{3.6}$$

Therefore, we obtain that $\{\varphi_i\}_{i=1}^d$ and $\{v_i\}_{i=1}^d$ are the eigenvectors for SS^T and $S^T S$ and their eigenvalues are $\lambda_i = \sigma_i^2 > 0 \quad i = 1, \ldots, d$.

From (3.3) and using that Φ and V are orthogonal, we obtain that

$$S = \Phi \Sigma V^T. \tag{3.7}$$

Simplifying (3.7) we note

$$S = \Phi^d D (V^d)^T = \Phi^d B^d, \tag{3.8}$$

where $\Phi^d \in \mathbb{R}^{d\times n}$, $V^d \in \mathbb{R}^{d\times n}$ and $B^d = D(V^d)^T \in \mathbb{R}^{d\times n}$. The expression (3.8) says that the column space of S can be represented in terms of the d linearly independent columns of Φ^d and we have that

$$s_j = \sum_{i=1}^d B_{ij}^d \varphi_i^d = \sum_{i=1}^d (D(V^d)^T)_{ij} \varphi_i = \sum_{i=1}^d ((\Phi^d)^T \Phi^d D (V^d)^T)_{ij} \varphi_j \quad = \\ \overset{(3.8)}{=} \sum_{i=1}^d ((\Phi^d)^T S)_{ij} \varphi_j = \sum_{i=1}^d (\sum_{k=1}^m \varphi_{ki}^d s_{kj}) \varphi_j = \sum_{i=1}^d (\varphi_i, s_j)_{\mathbb{R}^m} \varphi_i.$$

Therefore, the column vector y_j of S can be expressed as linear combination of $\{\varphi_i\}_{i=1}^d$ as:

$$y_j = \sum_{i=1}^d (\varphi_i, y_j)_{\mathbb{R}^m} u_i \quad \forall j = 1, \ldots, n. \tag{3.9}$$

3.2 The relation between POD and SVD

Now, we are going to connect the POD with the SVD. Firstly, we want to compute iteratively an orthonormal basis, which approximates the data set $\{y_j\}_{j=1}^n$. From this computation, we obtain the vectors $\{\varphi_i\}_{i=1}^r$ that will be called the POD basis of rank r. The first POD mode solves the following maximization problem

$$\max_{\tilde{\varphi_1} \in \mathbb{R}^m} \sum_{j=1}^n |(y_j, \tilde{\varphi_1})_{\mathbb{R}^m}|^2 \text{ s.t. } \|\tilde{\varphi_1}\|_{\mathbb{R}^m}^2 = 1. \tag{3.10}$$

Thanks to Lagrange formalism, the first lest singular vector φ_1 solves (3.10). Following this procedure, the second POD mode is obtained by solving the following maximization problem

$$\max_{\tilde{\varphi_2}\in\mathbb{R}^m} \sum_{j=1}^{n} |(y_j, \tilde{\varphi_2})_{\mathbb{R}^m}|^2 \text{ s.t. } \|\tilde{\varphi_2}\|^2_{\mathbb{R}^m} = 1, (\varphi_1, \tilde{\varphi_2})_{\mathbb{R}^m} = 0, \tag{3.11}$$

and the solution is given by the left singular vector φ_2. According this inductive procedure, we get the following theorem:

Theorem 1. *Let $S = [y_1, \ldots, y_n] \in \mathbb{R}^{m\times n}$ with rank $d \leq \min(m, n)$. Moreover let $S = \Phi\Sigma V^T$ be the SVD described in (3.7). Then for $1 \leq r \leq d$ the optimization problem*

$$\max_{\tilde{\varphi_1},\ldots,\tilde{\varphi_r}\in\mathbb{R}^m} \sum_{i=1}^{r}\sum_{j=1}^{n} |(y_j, \tilde{\varphi}_i)_{\mathbb{R}^m}|^2 \text{ s.t. } (\tilde{\varphi}_i, \tilde{\varphi}_j)_{\mathbb{R}^m} = \delta_{ij} \forall 1 \leq i, j \leq r, \tag{3.12}$$

is being solved by the left singular vectors $\{\varphi_i\}_{i=1}^r$ and it holds that

$$\arg\max(3.12) = \sum_{i=1}^{r} \sigma_i^2 = \sum_{i=1}^{r} \lambda_i. \tag{3.13}$$

Proof. The proof of Theorem (1) can be found in [[30],theorem 1.1]. □

Having made the connection between the POD basis and the SVD through Theorem (1), we are going to show that the POD basis is optimal in the following sense:

Theorem 2. *(Optimality of the POD basis) Let all hypotheses of Theorem (1) be satisfied. Suppose that $\widehat{\Phi}^d \in \mathbb{R}^{m\times d}$ denotes a matrix with pairwise orthonormal vectors $\widehat{\varphi}_i$ and that the expansion of the columns of S in the basis $\{\widehat{\varphi}_i\}_{i=1}^d$ is given by*

$$S = \widehat{\Phi}^d C^d, \text{ where } C^d_{ij} = (\widehat{\varphi}_i, y_j)_{\mathbb{R}^m} \text{ for } 1 \leq i \leq d, 1 \leq j \leq n. \tag{3.14}$$

Then for every $r \in \{1, \ldots, d\}$ we have

$$\|S - \Phi^r B^r\|_F \leq \|S - \widehat{\Phi}^r C^r\|_F. \tag{3.15}$$

In (3.15), $\|\cdot\|_F$ denotes the Frobenius norm given by

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |A_{ij}^2|} = \sqrt{trace(A^t A)} \text{ for } A \in \mathbb{R}^{m\times n}, \tag{3.16}$$

the matrix Φ^r denotes the first $r \leq d$ columns of Φ, B^r the first r rows of B and similarly for $\widehat{\Phi}^r$ and C^r.

To prove Theorem (2), we need the following result:

Lemma 1. *Let $A \in \mathbb{R}^{m\times n}$ and Q an orthogonal matrix of order m. Then:*

$$\|QA\|_F = \|A\|_F. \tag{3.17}$$

Proof.

$$\|QA\|_F^2 = \text{trace}((QA)^T(QA)) = \text{trace}(A^TQ^TQA) = \text{trace}(A^TA) = \|A\|_F. \tag{3.18}$$

□

Now we are going to prove Theorem (2).

Proof. Using the Lemma (1)

$$\|Y - \widehat{\Phi}^r C^r\|_F^2 = \|\widehat{\Phi}^d(C^d - C_0^r)\|_F^2 = \|C^d - C_0^r\|_F^2 = \sum_{i=r+1}^{d}\sum_{j=1}^{n}|C_{ij}^r|^2, \tag{3.19}$$

where $C_0^r \in \mathbb{R}^{r\times n}$ is obtained by replacing the last $d-r$ rows of $C \in \mathbb{R}^{r\times n}$ by 0. Similarly we have

$$\begin{aligned}&\|Y - \Phi^r B^r\|_F^2 = \|\Phi^k(B^d - B_0^r)\|_F^2 = \|B^d - B_0^r\|_F^2 = \textstyle\sum_{i=r+1}^{d}\sum_{j=1}^{n}|B_{ij}^d|^2\\ &= \textstyle\sum_{i=r+1}^{d}\sum_{j=1}^{n}|(y_j,\varphi_i)|^= \sum_{i=r+1}^{d}\sum_{j=1}^{n}((y_j,\varphi_i)y_j,\varphi_i) =\\ &\textstyle\sum_{i=r+1}^{d}(SS^T\varphi_i,\varphi_i) = \sum_{i=r+1}^{d}\sigma_i^2.\end{aligned} \tag{3.20}$$

We know that, by Theorem (1), $\varphi_1,\ldots,\varphi_d$ solve problem (3.12). Now, using (3.20), we obtain:

$$\|S\|_F^2 = \|\widehat{\Phi}^d C^d\|_F^2 = \|C^d\|_F^2 = \sum_{i=1}^{d}\sum_{j=1}^{n}|C_{ij}^r|^2, \tag{3.21}$$

and

$$\|S\|_F^2 = \|\Phi^d B^d\|_F^2 = \|B^d\|_F^2 = \sum_{i=1}^{d}\sum_{j=1}^{n}|B_{ij}^r|^2 = \sum_{i=1}^{r}\sigma^2. \tag{3.22}$$

Therefore

$$\begin{aligned}\|S - \Phi^r B^r\|_F^2 &= \textstyle\sum_{i=r+1}^{d}\sigma_i^2 = \sum_{i=1}^{d}\sigma_i^2 - \sum_{i=1}^{r}\sigma_i^2 = \|S\|_F^2 - \sum_{i=1}^{r}\sum_{j=1}^{n}|(y_j,\varphi_i)|^2\\ &\le \|S\|_F^2 - \textstyle\sum_{i=1}^{r}\sum_{j=1}^{n}|(y_j,\widehat{\varphi}_i)|^2 = \sum_{i=1}^{d}\sum_{j=1}^{n}|C_{ij}^d|^2 - \sum_{i=1}^{r}\sum_{j=1}^{n}|C_{ij}^d|^2\\ &= \textstyle\sum_{i=r+1}^{d}\sum_{j=1}^{n}|C_{ij}^d|^2 = \|S - \widehat{\Phi}^r C^r\|_F^2.\end{aligned} \tag{3.23}$$

□

We note that

$$\|S - \widehat{\Phi}^r C^r\|_F^2 = \sum_{j=1}^{n}\sum_{i=1}^{m} |S_{ij} - \sum_{k=1}^{r}(\widehat{\varphi_k}, y_j)\widehat{\Phi}_{ik}^r|^2 = \sum_{j=1}^{n} \|y_j - \sum_{k=1}^{r}(y_j, \widehat{\varphi_k})\widehat{\varphi_k}\|^2, \quad (3.24)$$

and

$$\|S - \Phi^r B^r\|_F^2 = \sum_{j=1}^{n} \|y_j - \sum_{k=1}^{r}(y_j, \varphi_k)\varphi_k\|^2. \quad (3.25)$$

Hence, by Theorem (2)

$$\sum_{j=1}^{n} \|y_j - \sum_{k=1}^{r}(y_j, \varphi_k)\varphi_k\|^2 \leq \sum_{j=1}^{n} \|y_j - \sum_{k=1}^{r}(y_j, \widehat{\varphi_k})\widehat{\varphi_k}\|^2. \quad (3.26)$$

Then, we can obtain the POD basis of rank r using the Theorem (2) by solving the following optimization problem:

$$\min_{\tilde{\varphi_1},\ldots,\tilde{\varphi_r}\in\mathbb{R}^m} \sum_{j=1}^{n} \|y_j - \sum_{i=1}^{r}(y_j, \tilde{\varphi_i})\tilde{\varphi_i}\|^2 \text{ s.t. } (\tilde{\varphi_i}, \tilde{\varphi_j}) = \delta_{ij} \text{ for } 1 \leq i, j \leq r. \quad (3.27)$$

In practice, computing the SVD of S is computationally expensive, since $S \in \mathbb{R}^{m\times n}$ is a dense matrix and m or n are very large. However, we can transform the search of the POD basis into an eigenvalues problem of size $m \times m$ or $n \times n$, which is useful if $m << n$ or $m >> n$. Hence, we are going to distinguish two cases:

1. If $n < m$:
   We compute the eigenvectors $\phi_1, \ldots, \phi_r \in \mathbb{R}^n$ by solving the symmetric $n \times n$ eigenvalues problem:

   $$S^T S\phi_i = \lambda_i \phi_i \text{ for } i = 1, \ldots, r. \quad (3.28)$$

   and using (3.4),

   $$\varphi_i = \frac{1}{\sqrt{\lambda_i}} S\phi_i \text{ for } i = 1, \ldots, r. \quad (3.29)$$

2. If $n > m$:
   We can obtain the POD basis by solving the $m \times m$ eigenvalues problem:

   $$SS^T\varphi_i = \lambda_i \varphi_i \text{ for } i = 1, \ldots, r. \quad (3.30)$$

This method is usually called the *method of snapshots* for historical reasons [26].

Finally, we mention that for the choice of r in practical problems one could use the following formula:

$$\epsilon(r) := \frac{\sum_{i=1}^{r} \lambda_i}{\sum_{i=1}^{d} \lambda_i} \geq \delta_\epsilon, \tag{3.31}$$

where $\epsilon(d)$ is called the energy ratio and $\delta_\epsilon \in (0, 1]$ is a fixed threshold. Since we want to capture most of the energy system by the POD modes, usually we choose δ_ϵ close to 1.

3.3 Application to Image Compression

In this section, we show how helpful is the SVD in real life, showing one of its most used application that is image compression. Image compression [7] is based on: given an image $S \in \mathbb{R}^{m\times n}$ of rank d, we are looking for a matrix $\tilde{S} \in \mathbb{R}^{m\times n}$ with rank $r << d$ such that $\|S - \tilde{S}\|$ is minimal. Then by the Eckart-Young theorem [12], we have that:

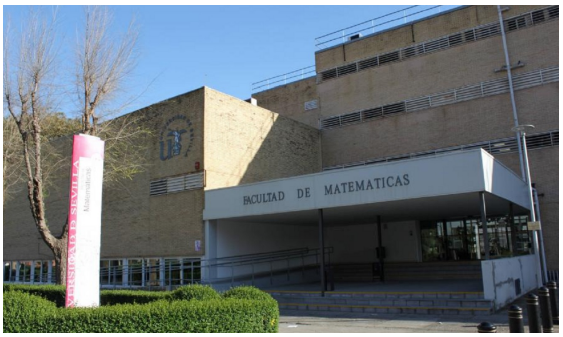
$$\min_{rank(\tilde{S})=r} \|S - \tilde{S}\| = \|S - S^{(r)}\| = \begin{cases} \sigma_{r+1} & \text{for the 2-norm.} \\ \sqrt{\sum_{i=r+1}^{d} \sigma_i^2} & \text{for the Frobenius-norm.} \end{cases} \tag{3.32}$$

The results (3.32) means that truncated matrix from the SVD:

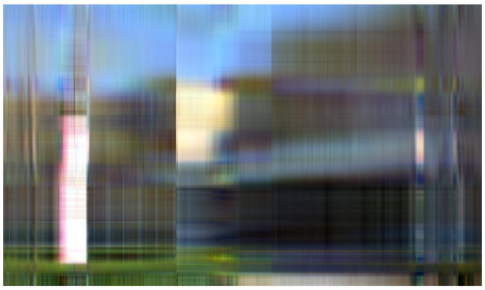
$$S^{(r)} := \Phi \begin{pmatrix} \Sigma^{(r)} & 0 \\ 0 & 0 \end{pmatrix} V^T = \sum_{i=1}^{r} \sigma_i \varphi_i v_i^T, \tag{3.33}$$

with $\Sigma^{(r)} := diag(\Sigma_{11}, \ldots, \Sigma_{rr})$ solves the minimization problem. Consequently, we only need to compute a truncated SVD with a good rank (high content of information) in order to compress an image.

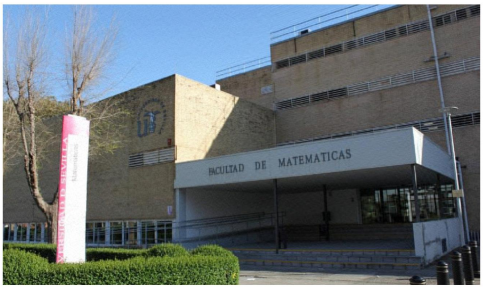
We use a colour image of the Faculty of Mathematics (University of Seville), which is 600 pixels wide and 900 pixels high. Each pixel is divided in three integer values corresponding to the intensity of red, blue and green of the colour. In Figure (3.1), we can see the original image and its SVD approximation of rank 5, 150 and 500.



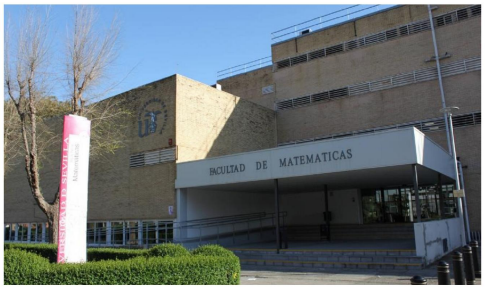
(a) Original image



(b) SVD Approximation of rank 5



(c) SVD Approximation of rank 150



(d) SVD Approximation of rank 500

Figure 3.1: Approximation of the original images with lower rank

The matrix representation of the image has full rank, i.e., it has rank 600. First of all, we have to divide the matrix of the original image into three matrices corresponding to each colour. Then, we have to perform an SVD approximation for each colour, therefore the decayment of its singular values are shown in Figure (3.2).

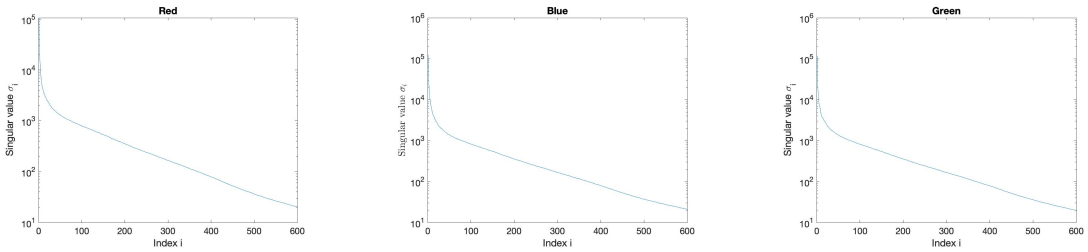


Figure 3.2: Singular values of red, blue and green.

From Figures (3.2), we observe a fast decay of the corresponding singular values, so that we expect a good approximation for relatively small values of r.

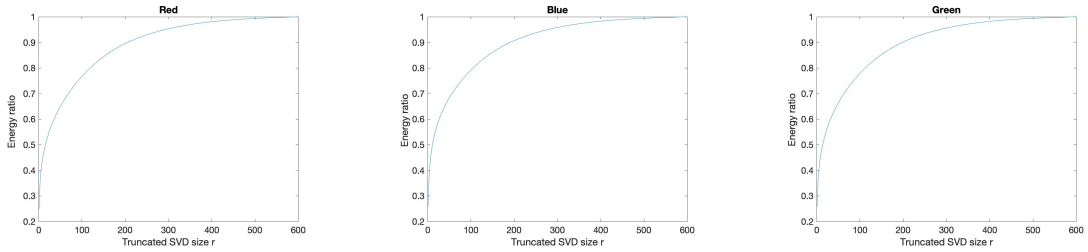


Figure 3.3: Energy ratio of red, blue and green.

In Figures (3.3), we show how the energy ratio $\varepsilon(r) := \frac{\sum_{i=1}^{r} \sigma_i}{\sum_{i=1}^{d} \sigma_i}$ depends on the size r of the truncated SVD and that a big portion of the energy ratio can be attributed to the first singular values, which motivates a truncated singular value descomposition for relatively small values of r.

3.4 The POD method with weighted inner product

In the next chapters, we are going to work with the L^2-inner product. Let consider two functions

$$u := \sum_{i=1}^{m} u_i^h \varphi_i^h,\ v := \sum_{j=1}^{m} v_j^h \varphi_j^h \in L^2(\Omega),$$

their L^2-inner product is

$$(u,v)_{L^2(\Omega)} = (\sum_{i=1}^{m} u_i^h \varphi_i^h, \sum_{j=1}^{m} v_j^h \varphi_j^h)_{L^2(\Omega)} = \sum_{i,j=1}^{m} u_i^h (\varphi_i^h, \varphi_j^h)_{L^2(\Omega)} v_j^h = (\mathbf{u}^h)^T M_h \mathbf{v}^h, \tag{3.34}$$

where $M_h \in \mathbb{R}^{m\times m}$ is the mass matrix, whose entries are $(M_h)_{ij} = (\varphi_i^h, \varphi_j^h)_{L^2(\Omega)}$ for $1 \le i,j \le m$. If we observe (3.34), we note that the L^2-inner product of two functions is a weighted inner product of their coefficient vectors. For this reason, we are going to extend the POD from Section 3.2 to weighted inner product.

Let us consider a weighted inner product:

$$(u,v)_W := u^T W v = (u, Wv)_{\mathbb{R}^m} = (Wu, v)_{\mathbb{R}^m} \text{ for } u,v \in \mathbb{R}^m, \tag{3.35}$$

where $W \in \mathbb{R}^{m\times m}$ is a symmetric, positive definite matrix. Moreover, $\|u\|_W = \sqrt{(u,u)_W}$ for $u \in \mathbb{R}^m$ is the associated induced norm. If we consider $W = I_m$, the identity matrix, the inner product (3.35) is the Euclidean inner product.

Now, we are going to repeat the same as in Section 3.2. The first POD mode solves the following maximization problem

$$\max_{\tilde{\varphi_1}\in\mathbb{R}^m} \sum_{j=1}^{n} |(y_j, \tilde{\varphi_1})_W|^2 \text{ s.t. } \|\tilde{\varphi_1}\|_W^2 = 1,$$

and then, the second POD mode is obtained by solving:

$$\max_{\tilde{\varphi_2}\in\mathbb{R}^m} \sum_{j=1}^{n} |(y_j, \tilde{\varphi_2})_W|^2 \text{ s.t. } \|\tilde{\varphi_2}\|_W^2 = 1.$$

Therefore, by induction, we get the following theorem:

Theorem 3. *(POD basis with weighted inner product) Let $S = [y_1, \ldots, y_n] \in \mathbb{R}^{m\times n}$ with rank $d \leq \min(m, n)$, $W \in \mathbb{R}^{m\times m}$ be a symmetric, positive definite matrix and $\bar{S} = W^{\frac{1}{2}}S$. Moreover, let $\bar{S} = \bar{\Phi}\Sigma\bar{V}^T$ be the singular value decomposition of $\bar{Y}$, where $\bar{\Phi} = [\bar{\varphi}_1, \ldots, \bar{\varphi}_m] \in \mathbb{R}^{m\times m}$ and $\bar{V} = [\bar{v}_1, \ldots, \bar{v}_n] \in \mathbb{R}^{n\times n}$ are the orthogonal matrices,*

$$\Sigma = \begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix} \in \mathbb{R}^{m\times n},$$

with $D = diag(\sigma_1, \ldots, \sigma_d) \in \mathbb{R}^{d\times d}$ and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_d > 0$. Then for $1 \leq r \leq d$ the optimization problem

$$\max_{\tilde{\varphi_1},\ldots,\tilde{\varphi_r}\in\mathbb{R}^m} \sum_{i=1}^{r}\sum_{j=1}^{n} |(y_j, \tilde{\varphi}_i)_W|^2 \text{ s.t. } (\tilde{\varphi}_i, \tilde{\varphi}_j)_W = \delta_{ij} \forall 1 \leq i, j \leq r. \tag{3.36}$$

is being solved by the vectors $\varphi_i = W^{-\frac{1}{2}}\bar{\varphi}_1$ for $1 \leq i \leq r$ and it holds that

$$\arg\max (3.36) = \sum_{i=1}^{r} \sigma_i^2 = \sum_{i=1}^{r} \lambda_i$$

Proof. The proof follows the same steps as Theorem (1). □

Remark 1. *Following again the same steps as in Section 3.2, we can show that*

$$\min_{\tilde{\varphi_1},\ldots,\tilde{\varphi_r}\in\mathbb{R}^m} \sum_{j=1}^{n} \|y_j - \sum_{i=1}^{d}(y_j, \tilde{\varphi}_i)_W\tilde{\varphi}_i\|_W^2 \text{ s.t. } (\tilde{\varphi}_i, \tilde{\varphi}_j)_W = \delta_{ij} \text{ for } 1 \leq i, j \leq r, \tag{3.37}$$

is an optimization problem equivalent to (3.27).

Remark 2. *We note that $\bar{S}^T\bar{S} = S^TWS$. Therefore, in order to solve the $n \times n$ eigenvalues problem we have*

$$\bar{S}^T\bar{S}\bar{\phi}_i = S^TWS\bar{\phi}_i = \lambda_i\phi_i \text{ for } i = 1, \dots, r.$$

and

$$\varphi_i = W^{-\frac{1}{2}}\bar{\varphi}_i = \frac{1}{\sqrt{\lambda_i}}W^{-\frac{1}{2}}(\bar{S}\bar{\varphi}_i) = \frac{1}{\sqrt{\lambda_i}}W^{-\frac{1}{2}}W^{\frac{1}{2}}S\bar{\varphi}_i = \frac{1}{\sqrt{\lambda_i}}S\bar{\varphi}_i, \text{ for } 1 \le i \le r.$$

Moreover,

$$(\varphi_i, \varphi_j)_W = \varphi_i^TW\varphi_j = \frac{\delta_{ij}\lambda_j}{\sqrt{\lambda_i\lambda_j}} = \delta_{ij} \quad \forall 1 \le i, j \le r,$$

so the matrix $W^{\frac{1}{2}}$ is not needed.

3.5 Treatment of non-linear and non-affine terms within the POD method

In this section, we introduce several techniques that allow to treat non-linear and non-affine terms (with respect to parameter) within the POD-ROM framework. Those methods help to reduce the complexity of these terms in such a way it is proportional to the number of the POD modes, which is crucial for reduced order modeling with POD. Their strategy consists in approximating a non-linear term $g : \mu \in \mathcal{P} \subset \mathbb{R}^p \to g(\mu) \in \mathbb{R}^N$ by projecting onto a POD basis $\mathbb{Q}$:

$$g(\mu) \approx g_{\tilde{r}}(\mu) = \mathbb{Q}\alpha(\mu) = \sum_{j=1}^{\tilde{r}} \alpha_j\tilde{\varphi}_j, \tag{3.38}$$

where μ is the parameter, $\mathbb{Q} = [\tilde{\varphi}_1, \dots, \tilde{\varphi}_{\tilde{r}}] \in \mathbb{R}^{N\times\tilde{r}}$ and $\alpha(\mu) \in \mathbb{R}^{\tilde{d}}$ is the coefficients vector with $\tilde{d} << N$. The POD basis $\mathbb{Q}$ is constructed applying the POD method described in section (3.2) on a set of snapshots:

$$S = [g(\mu^1)| \dots |g(\mu^{n_s})], \quad n_s > \tilde{r}.$$

Once the basis is obtained, we have to determine the vector $\alpha(\mu) \in \mathbb{R}^{\tilde{r}}$. This can be done by the following methods.

3.5.1 Discrete Empirical Interpolation Method

The first method we present to compute the coefficients vector α is the Discrete Empirical Interpolation Method (DEIM). To do so, we follow [3, 6, 23]. The DEIM procedure requires the following:

1. Construct a set of snapshots $S = [g(\mu^1)|\dots|g(\mu^{n_s})]$ and with this create a POD base $\mathbb{Q}$ as mentioned before.

2. Select iteratively $\tilde{r}$ indices $I \subset \{1,\dots,N\}$ from the POD basis $\mathbb{Q}$ by using a greedy procedure, which minimizes at each step the interpolation error over the snapshots set measured in the maximum norm.

3. Given a new parameter μ, compute the coefficients vector $\alpha(\mu)$ by solving the following liner system:

$$\mathbb{Q}_I \alpha(\mu) = g_I(\mu), \tag{3.39}$$

where $\mathbb{Q}_I \in \mathbb{R}^{\tilde{r}\times\tilde{r}}$ is the matrix formed by the I rows of $\mathbb{Q}$ and $g_I(\mu)$ is the evaluation of $g(\mu)$ on the I points. Therefore, we obtain:

$$g_{\tilde{r}}(\mu) = \mathbb{Q}\mathbb{Q}_I^{-1} g_I(\mu). \tag{3.40}$$

The detailed Algorithm with the online and offline phases completely separated is the following:

- OFFLINE PHASE

  1. The offline phase consists first in constructing the spatial basis $\mathbb{Q} = [\tilde{\varphi}_1|\cdots|\tilde{\varphi}_{\tilde{r}}]$ obtained by operating a POD over a set of snapshots $S = [g(\mu^1)|\cdots|g(\mu^{n_s})]$.
  2. The second step of the offline phase consists in selecting iteratively $\tilde{r}$ indices $I \subset \{1,\dots,N\}$, where N is usually the number of FOM degrees of freedom for $g(\mu)$, from the basis $\mathbb{Q}$ using the following greedy procedure
     - Initialization: $i_1 = \arg\ \max_{i=1,\dots,N} |(\tilde{\varphi}_1)_i|$; $\mathbb{Q} = \tilde{\varphi}_1$; $I = \{i_1\}$.
     - Iterations:

$$\begin{cases} \text{for } m = 2 : \tilde{r} \\ \quad \text{res} = \tilde{\varphi_m} - \mathbb{Q}\mathbb{Q}_I(\tilde{\varphi_m})_I; \\ \quad i_m = \ \arg\ \max_{\ i=1,\dots,N} |(\text{res})_i| \\ \quad \mathbb{Q} \leftarrow [\mathbb{Q}|\tilde{\varphi_m}]; I \leftarrow I \cup \{i_m\} \end{cases}$$

     where Q_I is the matrix formed by the I rows of $\mathbb{Q}$ and $(\tilde{\varphi_m})_I$ is formed by the I component of $\tilde{\varphi_m}$.

- ONLINE PHASE

1. In order to compute online the coefficients vector $\alpha(\mu) = [\alpha_1(\mu), \ldots, \alpha_{\tilde{r}}(\mu))]^T$, interpolation constraints are imposed at the points corresponding to the selected indices.
   First of all, we have to evaluate g on the interpolation points $\{x^{i_1}, \ldots, x^{i_{\tilde{r}}}\}$ in order to construct $g_I(\mu)$.
2. The second step of the online phase requires the solution of the following linear system:
$$\mathbb{Q}_{\mathcal{I}}\alpha = g_I(\mu).$$

Note that the complexity of solving the linear system (3.40) is $O(\tilde{r}^3)$ and we have an error bound

$$\|g(\mu) - g_{\tilde{r}}(\mu)\|_2 \leq \|\mathbb{Q}_{\mathcal{I}}^{-1}\|_2 \|(\mathbb{I} - \mathbb{Q}\mathbb{Q}^T)g(\mu)\|_2, \tag{3.41}$$

with

$$\|(\mathbb{I} - \mathbb{Q}\mathbb{Q}^T)g(\mu)\|_2 \approx \sigma_{r+1}, \tag{3.42}$$

being σ_{r+1} the first discarded singular values.

Alternatively to the DEIM, the Empirical Interpolation Method([4]) is also used for the approximation of non-linear terms. The philosophy of EIM is the same as DEIM, but the construction of the basis $\mathbb{Q}$ is done by a Reduced Basis procedure with greedy algorithm ([8]) instead of the POD.

3.5.2 Radial Basis Functions

The second method we introduce to compute the coefficient vector α in (3.38) is the Radial Basis Functions (RBF). To decribe it, we follow [13]. First, we note that in this method the non-linear term can depend on different parameters. For simplicity, we are going to consider two parameters, μ and t (e.g., time):

$$g(\mu, t) \approx g_{\tilde{r}}(\mu, t) = \mathbb{Q}\alpha(\mu, t) = \sum_{j=1}^{\tilde{r}} \alpha_j(\mu, t)\tilde{\varphi}_j.$$

The POD basis $\mathbb{Q}$ is now obtained by a POD method on

$$S = \{g(x, t_i; \mu_1), \ldots, g(x, t_{n_s}; \mu_M)\}.$$

Before explaining the method, we are going to fix some notations:

- $P_M = \{\mu_1, \ldots, \mu_M\}$ is the set of the M parameters.

- $X_{\mu,t} = P_M \times \{t_1, \dots, t_N\}$ is the Cartesian product of the discretized parameter set and the set of times instants at which snapshots were taken.
- $x^i_{\mu,t}$ is the i-th member of $X_{\mu,t}$.
- t^* is the time instant at which the ROM solution is sought.
- μ^* is the new parameter considered in the online stage of the ROM (within the range $[\mu_1, \mu_M]$, but different from the parameters used in the offline stage).
- $z^* = (t^*, \mu^*)$ is combination of the online parameter and the time instant at which the ROM solution is desired.
- $S = \{g(x, t_i; \mu_1), \dots, g(x, t_{n_s}; \mu_M)\} \in \mathbb{R}^{N \times \tilde{M}}$ is the set of snapshots, where $\tilde{M} = n_s \cdot M$.
- $\tilde{\alpha_{r,l}} = (S^r, \tilde{\varphi}_l)$ is the projection of the r-th column of S on the l-th POD mode $(\tilde{\varphi}_l)$ of $\mathbb{Q}$.

The interpolation statement will be the following

- Given $X_{\mu,t}, [S^i]_{i=1}^{\tilde{M}}$ and $[\alpha_{r,l}]_{r=1,l=1}^{\tilde{M},\tilde{d}}$, predict the value of α in (3.38) for z^*.
- The goal can be split to each of the scalar coefficient $[\alpha_i(t^*, \mu^*)]_{i=1}^{M}$.

Note that the interpolation procedure will be carried out for each mode separately, so we can fix one mode $\tilde{\varphi}_L$ and obtain $Y_L = [\alpha_{r,L}]_{r=1}^{\tilde{M}} \in \mathbb{R}^N$ set of observation. The procedure is the following:

1. Consider the pair $X_{\mu,t}, Y_L$.
2. The interpolation using RBF functions is based on the following formula:

$$G_L(z) = \sum_{j=1}^{N} w_{L,j}, \xi_{L,j}(\|z - x^j_{\mu,t}\|_{L^2(\mathbb{R}^{q+1})}) \quad L = 1, 2, \dots, M, \tag{3.43}$$

   where $z = (t, \mu), w_{L,j}$ are some appropriate weights and $\xi_{L,j}$ are the RBF functions which are chosen to be Gaussian functions centered in $x^i_{\mu,t}$.

3. Computations of the weights.
   (a) We know $G_L(x^i_{\mu,t}, t) = \alpha_{i,L}$.
   (b) $\sum_{j=1}^{N} w_{L,j}, \xi_{L,j}(\|x^i_{\mu,t} - x^j_{\mu,t}\|_{L^2(\mathbb{R}^{q+1})}) = g_{i,L}$.

(c) This corresponds to the following linear system:

$$A_L^{\xi} w_L = Y_L, \tag{3.44}$$

where

$$(A_L^{\xi})_{ij} = \xi_{L,j}(\|x_{\mu,t}^i - x_{\mu,t}^j\|_{L^2}). \tag{3.45}$$

The weights w_L can be thus computed in the offline phase, and then stored to be used in the online phase.

(d) In the online stage, we have z^* and the goal is to compute $\alpha(z^*) = [\alpha_i(z^*)]_{i=1}^{\tilde{r}}$ which it is done by

$$\alpha(z^*) \approx G_i(z^*) = \sum_{j=1}^{\tilde{M}} w_{i,j}\xi_{i,j}(\|z^* - x_{\mu,t}^j\|_{L^2(\mathbb{R}^{q+1})}) \quad i = 1, 2, \ldots, \tilde{r}. \tag{3.46}$$

This technique has been used in Appendix (A) to approximate the eddy viscosity.

Chapter 4

The POD for the Burgers equation

The Burgers equations with initial and boundary conditions is given by:

$$\begin{cases} u_t - \nu u_{xx} + u u_x = f & \text{in } \Omega \times (0,T), \\ u(x,0) = u_0 & \text{in } \Omega, \\ u(x,t) = g & \text{on } \partial\Omega \times (0,T). \end{cases} \tag{4.1}$$

where ν is the diffusion parameter, f is the forcing term, Ω is the 1D computational domain and T is the final time. We are going to consider the function space $X = H_0^1(\Omega)$ and assume that, without loss of generality, $g \equiv 0$ and $f \equiv 0$. Then, the weak formulation of the Burgers equation (4.1) reads:

$$(u_t, v)_{L^2} + \nu(u_x, v_x)_{L^2} + (uu_x, v)_{L^2} = 0 \quad \forall v \in X. \tag{4.2}$$

To derive the standard POD-Galerkin ROM applied to the Burgers equation, one has to solve first the fully discretized version of (4.2) for a certain set of time instances. In this way, we collect the snapshots, and we perform the POD method over the set of snapshots to compute the POD basis in L^2 (see Section 3.4). The Galerkin projection-based POD-ROM uses both Galerkin truncation and Galerkin projection. The former yields an approximation of the solution by a linear combination of the truncated POD basis:

$$u(x,t) \approx u_r(x,t) = \sum_{j=1}^{r} a_j(t)\varphi_j(x) \in X^r, \tag{4.3}$$

where $X^r = \text{span}\{\varphi_1, \ldots, \varphi_r\}$ is the space for the POD setting and $\{a_i(t)\}_{i=1}^r$ are the time coefficients.

Replacing the solution u with u_r in the Burgers equations (4.2), using the Galerkin method, and projecting the resulted equations onto the space X^r, one

obtains the standard POD-Galerkin ROM (POD-G-ROM) for the Burgers equations. To obtain its time discretization, we apply the backward Euler method and denoting $\bar{\partial} u_r^k = \dfrac{u_r^k - u_r^{k-1}}{\Delta t}$, the problem reads:

$$(\bar{\partial} u_r^k, \varphi)_{L^2} + \nu((u_r^k)_x, (\varphi)_x)_{L^2} + (u_r^k (u_r^k)_x, \varphi)_{L^2} = 0 \quad \forall \varphi \in X^r. \tag{4.4}$$

The POD-G-ROM (4.4) leads to the following autonomous system for the coefficients $a(t)$:

$$\dot{a}\,(t) = Ba + a^T Ca, \tag{4.5}$$

where B and C correspond to the linear and quadratic terms in the numerical discretization of the Burgers equation (4.4), respectively. The initial conditions are obtained by the projection:

$$\dot{a}_i\,(0) = (u(\cdot, 0), \varphi_i)_{L^2} \quad i = 1, \ldots, r.$$

The finite dimensional system (4.5) can be written componentwise as follow:

$$\dot{a}_i\,(t) = \sum_{m=1}^{r} B_{im} a_m(t) + \sum_{n=1}^{r} \sum_{m=1}^{r} C_{imn} a_m(t) a_n(t).$$

where

$$B_{im} = -((\varphi_m)_x, (\varphi_i)_x)_{L^2},$$

$$C_{imn} = -(\varphi_m (\varphi_n)_x, \varphi_i)_{L^2}.$$

4.1 AV-POD-G-ROM

In this section we are going to introduce an artificial viscosity POD-ROM (AV-POD-G-ROM). The reason to introduce this model is that for convection-dominated non-linear problems, the standard POD-G-ROM is not accurate. Thus, we introduce a new term of artificial viscosity (AV), which try to replicate in this framework the effect of the Smagorinsky turbulence model for the Navier-Stokes equations [32]. We emphasize that, although the Burgers equations is considered as simplification of NSE and small diffusion parameters ν are corresponding to high Reynolds numbers in NSE, in this context we are not representing the real turbulence. Therefore, the role of closure methods changes correspondingly. Instead of improving physical accuracy (dissipating energy according to the well known concept of energy cascades), they increase the numerical stability (in testing problems

showing shock-like behavior, for instance). Of course, once we fully understand the behavior of this kind of POD closure model in this simplified setting, we will analyze in the NSE setting (see (6)). Therefore, the new weak formulation of (4.2) is:

$$(u_t, v)_{L^2} + \nu(u_x, v_x)_{L^2} + (uu_x, v)_{L^2} + c(|u_x|u_x, v_x) = 0 \quad \forall v \in X, \tag{4.6}$$

where c is a positive coefficient.

Now, we repeat the same process as in the previous section, so we replace (4.3) into (4.6) and obtain the AV-POD-G-ROM:

$$(\bar{\partial} u_r^k, \varphi)_{L^2} + \nu((u_r^k)_x, (\varphi)_x)_{L^2} + (u_r^k (u_r^k)_x, \varphi)_{L^2} + c(|u_r^k|u_r^k, \phi_x)_{L^2} = 0 \quad \forall \varphi \in X^r. \tag{4.7}$$

In order to obtain an autonomous system with respect to the time coefficients $a(t)$, we have to treat the non-linear term $(|(u_d)_x|(u_d)_x, \varphi_x)$. In this case we are going to do an approximation by the DEIM (see Section (3.5.1)). Up to our knowledge, it is the first time that the DEIM is applied in such kind of model to reduce its computational complexity, as we will show in Section (4.3). Then, if we apply the DEIM method on $|u_x|$ we obtain:

$$|u_x| = \sum_{j=1}^{\tilde{r}} \alpha_j(t)\tilde{\varphi}_j,$$

where $\tilde{X}^{\tilde{r}} = span\{\tilde{\varphi}_1, \ldots, \tilde{\varphi}_{\tilde{r}}\}$ and $\alpha_j(t)$ are the time coefficients.
Consequently we obtain the following autonomous system for the coefficients $a(t)$:

$$\dot{a}\,(t) = Ba + a^T Ca + \alpha^T Da. \tag{4.8}$$

where B, C and D correspond to the linear and the two quadratic terms in the numerical discretization of the AV Burgers equation (4.7), respectively. The initial conditions are obtained by the projection:

$$\dot{a}_i\,(0) = (u(\cdot, 0), \varphi_i)_{L^2} \quad i = 1, \ldots, r.$$

The finite dimensional system (4.8) can be written componentwise as follow:

$$\dot{a}_i\,(t) = \sum_{m=1}^{r} B_{im} a_m(t) + \sum_{n=1}^{r}\sum_{m=1}^{r} C_{imn} a_m(t) a_n(t) + \sum_{n=1}^{\tilde{r}}\sum_{m=1}^{r} D_{imn} a_m(t)\alpha_n(t),$$

where

$$B_{im} = -((\varphi_m)_x, (\varphi_i)_x)_{L^2},$$

$$C_{imn} = -(\varphi_m(\varphi_n)_x, \varphi_i)_{L^2},$$

$$D_{imn} = -(c(\tilde{\varphi_m})_x(\varphi_n)_x, (\varphi_i)_x)_{L^2}.$$

4.2 Error estimates

In this section, our aim is to prove estimates for the time discretization error $\frac{1}{M}\sum_{k=1}^{M} \|u_r^k - u(t_k)\|_{L^2}^2$, where u_r^k is the solution of (4.7) and $u(t_k)$ is the solution of (4.6). For it, we follow [19] and [32]. However, in this section we only focus on time discretization and POD truncation, for finite element spatial discretization the reader is referred to [20] and [15].

First of all, we are going to introduce some notations and results that we need to prove the main result. For clarity and simplicity, let $L^2, H_0^1, W^{1,3}, W_0^{1,3}$ denote the usual Sobolev spaces on Ω and $(\cdot,\cdot) = (\cdot,\cdot)_{L^2}$.
For all $\varphi, \psi \in H_0^1$, we define

$$a(\varphi, \psi) := \nu(\varphi_x, \psi_x) \quad \forall \varphi, \psi \in H_0^1, \tag{4.9}$$

$$(F(\varphi), \psi) := (\varphi\varphi_x, \psi). \tag{4.10}$$

For $\varphi, \psi \in W^{1,3}$ and $c > 0$,

$$(G(\varphi), \psi) := c(|\varphi_x|\varphi_x, \psi_x). \tag{4.11}$$

We will denote by C_k, $k \in \mathbb{N}$ the different constants that will appear independent of the number of the POD modes (r) and the number of snapshots (M).

Now, we show several results, that will be used throughout this section. The first one:

Lemma 2 (Poincarè inequality)**.** $\exists \alpha > 0$ *constant such that*

$$\|\varphi\|_{L^2} \leq \alpha |\varphi|_{H^1} \quad \forall \varphi \in H_0^1, \tag{4.12}$$

where $|\cdot|_{H^1}$ *is the* H^1-*seminorm.*

Lemma 3. *The bilinear form* $a(\cdot,\cdot)$ *is continuous and coercive, this means that* $\exists \beta, \kappa$ *such that*

$$|a(\varphi, \psi)| \leq \beta \|\varphi\|_{H^1} \|\psi\|_{H^1} \quad \forall \varphi, \psi \in H_0^1, \tag{4.13}$$

$$|a(\varphi, \varphi)| \leq \kappa \|\varphi\|_{H^1}^2 \quad \forall \varphi \in H_0^1. \tag{4.14}$$

Lemma 4 (Strong monotonicity and Lipschitz continuity of G)**.** *For all $\varphi_1, \varphi_2, \psi \in W^{1,3}, \exists C$ constant, such that*

$$(G(\varphi_1) - G(\varphi_2), \varphi_1 - \varphi_2) \geq C\|(\varphi_1 - \varphi_2)_x\|_{L^3}^3, \tag{4.15}$$

$$(G(\varphi_1) - G(\varphi_2), \psi) \leq C\tilde{M}\|(\varphi_1 - \varphi_2)_x\|_{L^3}\|\psi_x\|_{L^3}, \tag{4.16}$$

where $\tilde{M} := \max\{\|(\varphi_1)_x\|_{L^3}, \|(\varphi_2)_x\|_{L^3}\}$.

Definition 1. *The projection $P_R : H_0^1 \to X^r$ is defined as follows: for all $\psi \in X^r$*

$$a(P_R u, \psi) = a(u, \psi) \quad \forall u \in H_0^1. \tag{4.17}$$

Lemma 5. *If Δt is small enough, we have the following approximation for P_R([19]):*

$$\frac{1}{M}\sum_{k=1}^{M}\|u(_k) - P_R u(t_k)\|_{H^1}^2 \leq C_1 \sum_{j=r+1}^{d}\lambda_j, \tag{4.18}$$

$$\frac{1}{M}\sum_{k=1}^{M}\|\bar{\partial}u(t_k) - P_R\partial\bar{u}(t_k))\|_{H_1}^2 \leq C_2 \sum_{j=r+1}^{d}\lambda_j. \tag{4.19}$$

We also need the following results:

Lemma 6. *$\exists C_3, C_4$ such that for all $v \in X$ it is verified*

$$\|v\|_{L^3} \leq C_3\|v\|_{L^2} \qquad \|v\|_{L^2} \leq C_4\|v\|_{L^3}. \tag{4.20}$$

Proof. We note that $\{\varphi_1, \ldots, \varphi_d\}$ is an orthonormal basis of X, so if $v \in X$, then $\exists v_i$ scalars such that $v_i = (v, \varphi_i)$ and $v = \sum_{i=1}^{d} v_i\varphi_i$. Consequently

$$\|v\|_{L^3} = \|\sum_{i=1}^{d} v_i\varphi_i\|_{L^3} \leq \sum_{i=1}^{d}|v_i|\|\varphi_i\|_{L^3} \leq (\max_{i=1,\ldots,r}\|\varphi_i\|_{L^3})\sum_{i=1}^{d}|v_i|. \tag{4.21}$$

Now we need the following auxiliar Lemma (7) whose proof is in ([5]):

Lemma 7 (auxiliar)**.** *Let $\{x_1, \ldots, x_n\}$ be a linearly independent set of vectors in a normed space X (of any dimension). Then there is a number $C > 0$ such that for every choice of scalars $\alpha_1, \ldots, \alpha_n$ we have*

$$\|\alpha_1 x_1 + \ldots + \alpha_n x_n\| \geq C(|\alpha_1| + \cdots + |\alpha_n|). \tag{4.22}$$

Applying the auxiliar lemma (7) into (4.21) we obtain the first inequality. The other one has the same proof as the first. □

Lemma 8. *Assume that the solution of* (4.6) *satisfies* $u_x \in H_0^1$. *There exists constants* C_5, C_6 *and* C_7, *such that, for all* $k = 1, \ldots, m$, *the following inequalities hold*

$$\|u_x(t_k)\|_{L^2} \leq C_5, \qquad \|u_x(t_k)\|_{L^3} \leq C_6, \qquad \|P_R u_x(t_k)\|_{L^2} \leq C_7. \tag{4.23}$$

Proof. The first two inequalities are easily proved by letting $\varphi = u$ in (4.6), using $(F(u), u) = 0$, Cauchy-Schwarz inequality, the strong monotonicity of G (4.15) and adjusting constants. Therefore, we want to prove the last inequality, so first of all, we remind the definition of the projection (4.17):

$$a(P_R u_x, \psi) = a(u_x, \psi) \quad \forall \psi \in X^r. \tag{4.24}$$

If we let $\psi = P_R u_x$, we obtain:

$$a(P_R u_x, P_R u_x) = a(u_x, P_R u_x). \tag{4.25}$$

Next, we are going to bound separately both terms in (4.25).
Firstly, using the continuity of $a(\cdot, \cdot)$:

$$a(P_R u_x, u_x) \leq \beta \|P_R u_x\|_{L^2} \|u_x\|_{L^2}. \tag{4.26}$$

Secondly, using the coercivity of $a(\cdot, \cdot)$:

$$a(P_R u_x, P_R u_x) \geq \kappa \|P_R u_x\|_{L^2}^2. \tag{4.27}$$

Then, if we join (4.27),(4.26) and (4.25):

$$\kappa \|P_R u_x\|_{L^2}^2 \leq a(P_R u_x, P_R u_x) = a(u_x, P_R u_x) \leq \beta \|P_R u_x\|_{L^2} \|u_x\|_{L^2}, \tag{4.28}$$

$$\|P_R u_x\|_{L^2} \leq \frac{\beta}{\kappa} \|u_x\|_{L^2} \leq C_7. \tag{4.29}$$

□

We are now ready to prove the main theorem:

Theorem 4. *Let* u *be the solution of* (4.6) *and* $\{u_r^k\}_{k=1}^M$ *be the solution of* (4.7). *Assume that* $u_x \in H_0^1$ *and* $u_{tt} \in L^2(0, T; L^2)$. *If* Δt *is small enough, then there exists a constant* $C > 0$, *such that*

$$\frac{1}{M} \sum_{k=1}^{M} \|u_r^k - u(t_k)\|_{L^2} \leq C(\|u_0 - P_R u_0\|^2 + \sum_{j=r+1}^{d} \lambda_j + \Delta t^2). \tag{4.30}$$

Proof. The proof follows the same ideas and procedure as the Theorem 7 and 11 in ([19]) and Theorem 1 in section 4 of [32]. For that reason, we only show the main differences and a small part of the proof.
First of all, we decompose the error:

$$u_r^k - u(t_k) = u_r^k - P_R u(t_k) + P_R u(t_k) - u(t_k). \tag{4.31}$$

Now, we define $\vartheta_k := u_r^k - P_R u(t_k)$ and $\varrho_k := P_R u(t_k) - u(t_k)$, so:

$$u_r^k - u(t_k) = \vartheta_k + \varrho_k. \tag{4.32}$$

If we use the triangle inequality, we have

$$\frac{1}{M}\|u_r^k - u(t_k)\|^2 \leq \frac{2}{M}\sum_{k=1}^{M}\|\vartheta_k\|^2 + \frac{2}{M}\sum_{k=1}^{M}\|\varrho_k\|^2. \tag{4.33}$$

The second term on the right hand side of (4.33) can be bounded by the properties of the projection (4.18)-(4.19) and the Poincarè's inequality (4.12):

$$\frac{1}{M}\sum_{k=1}^{M}\|\varrho_k\|^2 \leq C_8 \sum_{j=r+1}^{d}\lambda_j. \tag{4.34}$$

For the first term, we need to introduce some notation: for $k = 1, \ldots, m$ $\bar{\partial}\vartheta_k := \dfrac{\vartheta_k - \vartheta_k - 1}{\Delta t}$, $v_k := u(t_k) - \bar{\partial}P_R u(t_k) = w_k + z_k$, $w_k := u_t(t_k) - \bar{\partial}u(t_k)$, $z_k := \bar{\partial}u(t_k) - \bar{\partial}P_R u(t_k)$.

For all $\psi \in X^r$, we have

$$\begin{array}{rcl} (\bar{\partial}\vartheta_k, \psi) + a(\vartheta_k, \psi) = (\bar{\partial}u_r^k, \psi) + a(u_r^k, \psi) - (\bar{\partial}P_R u(t_k), \psi) - a(P_R u(t_k), \psi) \\ = (f(t_k), \psi) - (F(u_r^k), \psi) - (G(u_r^k), \psi) - (\bar{\partial}P_R u(t_k), \psi) - a(u(t_k), \psi) \\ = (v_k, \psi) + (F(u(t_k)) - F(u_r^k), \psi) + (G(u(t_k)) - G(u_r^k), \psi). \end{array} \tag{4.35}$$

Letting $\psi := \vartheta_k$, we get

$$\begin{array}{rcl} \|\vartheta_k\|^2 - (\vartheta_k, \vartheta_{k-1}) + \Delta t\nu\|(\vartheta_k)_x\|^2 = \Delta t(v_k, \vartheta_k) \\ + \Delta t(F(u(t_k)) - F(u_r^k), \vartheta_k) + \Delta t(G(u(t_k)) - G(u_r^k), \vartheta_k). \end{array} \tag{4.36}$$

We now have to estimate the r.h.s. of (4.36). The first one, if we use the Cauchy-Schwarz inequality, we obtain

$$(v_k, \vartheta_k) \leq \|v_k\|\|\vartheta_k\|. \tag{4.37}$$

The second one, if we do the same as in the page 139 of [19] we obtain the following estimate:

$$|(F(u(t_k)) - F(u_r^k), \vartheta_k)| \leq \nu \|\vartheta_k\|_{H^1}^2 + C_9 \|\vartheta_k\|_{L^2}^2 + C_{10} \|\varrho\|_{H^1}^2. \tag{4.38}$$

For the last term, since it is not included in [19], we are going to use the same ideas as in [32]. We start by:

$$\begin{array}{rcl} (G(u(t_k)) - G(u_r^k), \vartheta_k) & = & (G(u(t_k)) - G(P_R u(t_k)), \vartheta_k) \\ & + & (G(P_R u(t_k)) - G(u_r^k), \vartheta_k). \end{array} \tag{4.39}$$

Now, if we move the last term on the right hand side of (4.39) to the left hand side of (4.36) and apply the strong monotonicity property of G (4.15):

$$(G(u_r^k) - G(P_R u(t_k)), \vartheta_k) \geq C_{11} \|(\vartheta_k)_x\|_{L^3}^3. \tag{4.40}$$

If we apply the Lipschitz continuity of G (4.16) to the first term on the right hand side of (4.36):

$$(G(u(t_k)) - G(P_R u(t_k)), \vartheta_k) \leq C\tilde{M} \|(\varrho_k)_x\|_{L^3} \|(\vartheta_k)_x\|_{L^3}, \tag{4.41}$$

where $\tilde{M} = \max\{\|(u(t_k))_x\|_{L^3}, \|P_R(u(t_k))_x\|_{L^3}\}$ and we have used the Lemma (8).

Using the first estimate in Lemma (6) and Young's inequality:

$$\begin{array}{c} \|(\varrho_k)_x\|_{L^3} \|(\vartheta_k)_x\|_{L^3} \leq C_3^2 \|(\varrho_k)_x\| \|(\vartheta_k)_x\|_{L^2} \leq \\ \leq \dfrac{\nu \|(\vartheta_k)_x\|_{L^2}^2}{2} + \dfrac{1}{2\nu} C_3^4 \|(\varrho_k)_x\|_{L^2}^2. \end{array} \tag{4.42}$$

Using (4.36)-(4.42), we obtain the following approximation

$$\begin{array}{c} \|\vartheta_k\|_{L^2}^2 - (\vartheta_k, \vartheta_{k-1}) + \Delta t \nu \|(\vartheta_k)_x\|_{L^3}^3 \leq \\ \leq \|v_k\|_{L^2} \|\vartheta_k\|_{L^2} + (\nu + C_{13} + \dfrac{\nu}{2}) \|(\vartheta_k)_x\|_{L^2}^2 + (C_{10} + \dfrac{C_3^4}{2\nu}) \|(\varrho_k)_x\|_{L^2}^2. \end{array} \tag{4.43}$$

If we follow the rest of the proof in the page 140 of [19], choosing a small Δt and summing over k, we obtain

$$\|(\vartheta_k)_x\|_{L^2}^2 \leq e^{C_{14} T} (\|(v_0)_x\|_{L^2}^2 + \Delta t \sum_{j=1}^{k} (\|\vartheta\|_{L^2}^2 + C_{15} \|(\varrho)_x\|_{L^2}^2)). \tag{4.44}$$

Finally, using the estimate of v_k as done in [19] and (4.34) we obtain the main result (4.30). □

4.3 Practical Implementation and Numerical Results

The main goal of this section is to show the improvements of the AV-POD-G-ROM over the POD-G-ROM in terms of accuracy. We take the same initial condition and similar parameters used in [32, 21, 17].

In order to perform the computations, we consider the initial condition of (4.1) as :

$$u_0(x) = \begin{cases} 1 & \text{if } x \in (0, \frac{1}{2}], \\ 0 & \text{if } x \in (\frac{1}{2}, 1). \end{cases} \tag{4.45}$$

The following numerical discretization has been used in all our numerical tests. The computational domain is $\Omega = (0, 1)$ and the final time is $T = 1$. The parameters that we are going to use are: mesh size $h = \frac{1}{1024}$ for the Finite Element (FE) full order discretization; time step $\Delta t = 10^{-3}$; diffusion parameter $\nu = 10^{-5}$; number of snapshots $M = 1001$. Note that the used diffusion parameter is rather sharp with respect to the one used in [32, 21, 17], for which $\nu \geq 10^{-3}$.

Firstly, we run a Direct Numerical Simulation (DNS). In addition, given the spatial and temporal resolution used, we can consider the resolution of the problem by finite elements as our benchmark solution, see Figure (4.2). The 1001 snapshots are generated by piecewise linear finite elements in space and backward Euler method in time, and they are used to generate the POD modes (we can see some of them in Figure (4.1)). The first 30 modes are used to generate the POD-G-ROM (4.4) and the AV-POD-G-ROM (4.7) with $c = 10^{-4}$.

In order to employ the DEIM method to the non-linear term, we consider 1001 snapshots too of it. We have to take a determinated number of mode in order to get more than the 99% of energy for it. In this case, if we consider $\tilde{r} = 60$ we get it, but to show the efficiency of the DEIM algorithm we considered smaller values of $\tilde{r}$ too.

We show the space-time evolution of the DNS in Figure (4.2), POD-G-ROM model in Figure (4.3) and AV-POD-G-ROM models with different values of $\tilde{r}$ in Figures 4.4-4.5-4.6.

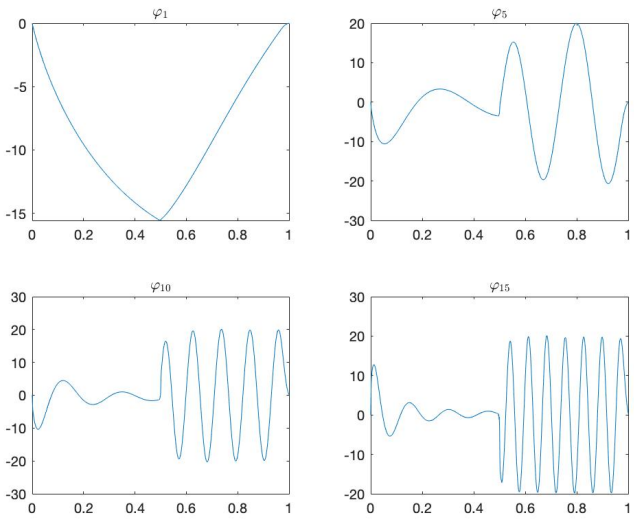


Figure 4.1: POD modes

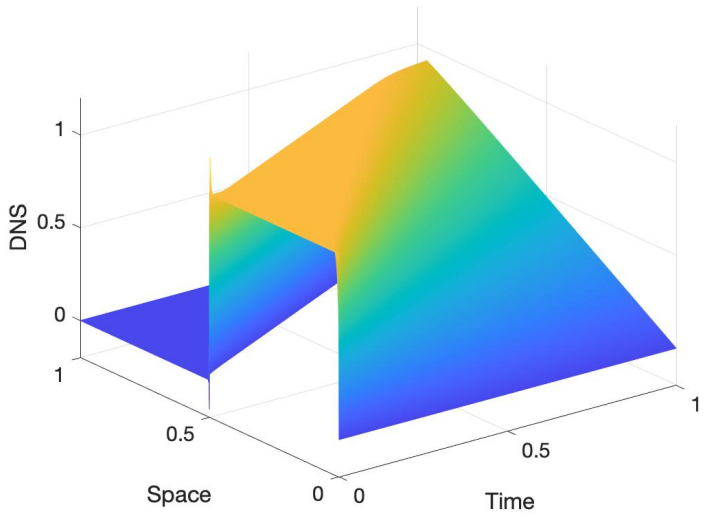


Figure 4.2: DNS solution.

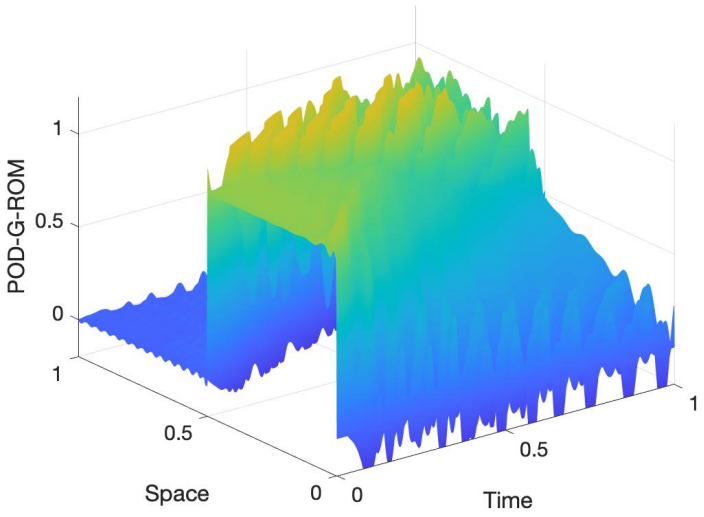


Figure 4.3: POD-G-ROM solution.

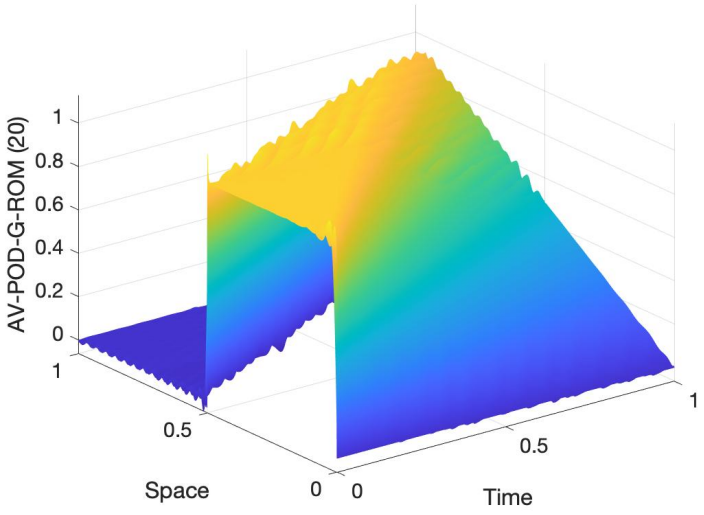


Figure 4.4: AV-POD-G-ROM solution in space-time with $\tilde{r} = 20$

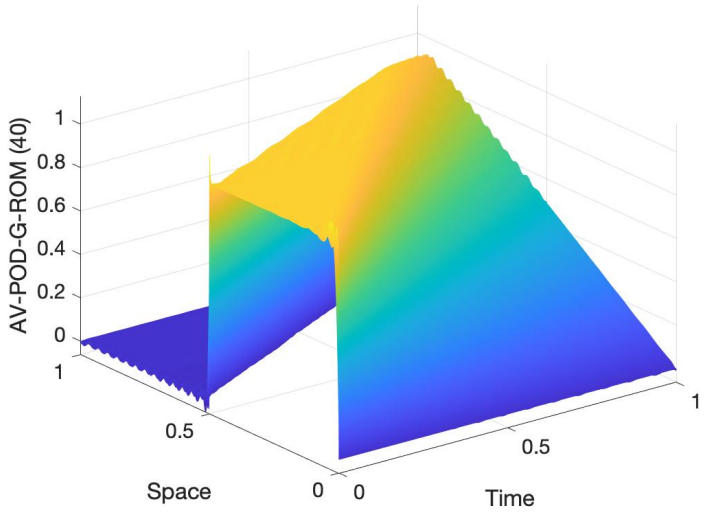


Figure 4.5: AV-POD-G-ROM solution in space-time with $\tilde{r} = 40$

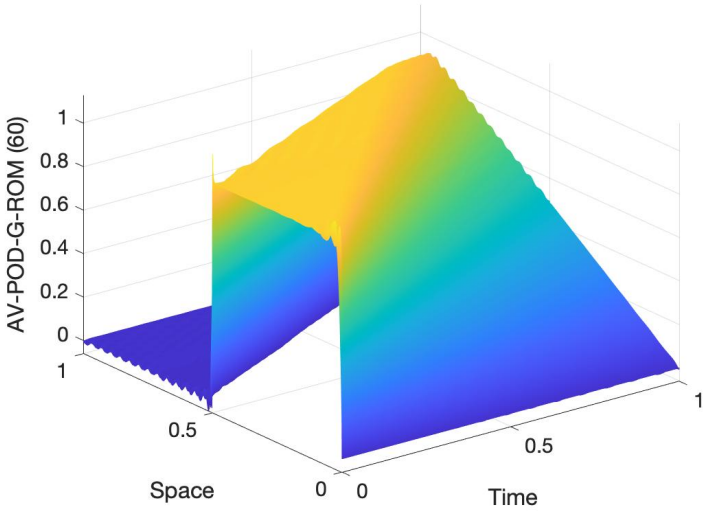


Figure 4.6: AV-POD-G-ROM solution in space-time with $\tilde{r} = 60$

We present the $l^2(L^2)$ errors for both models POD-G-ROM (4.4) and AV-POD-G-ROM (4.7) with different values of r, and the Normalized residual energy

r	Normalized residual energy $\frac{\sum_{j=r+1}^{1001}\lambda_j}{\sum_{j=1}^{1001}\lambda_j}$	POD	AV-POD(20)	AV-POD(40)	AV-POD(60)
6	1.1e-02	7.7e-04	4.0e-04	4.0e-04	4.0e-04
12	2.2e-03	7.0e-04	5.7e-05	5.5e-05	5.5e-05
21	1.7e-04	4.7e-04	6.0e-06	4.7e-06	4.9e-06
30	1.0e-05	4.2e-04	2.2e-06	9.3e-07	1.1e-06
39	3.0e-06	3.7e-04	1.9e-06	8.8e-07	9.2e-07
48	6.8e-07	3.5e-04	1.8e-06	8.5e-07	8.1e-07

Table 4.1: $l^2(L^2)$ errors between both ROMS and DNS and the Normalized residual energy.

If we observe Table 4.1, we can apreciate that the POD-G-ROM is always worse than the AV-POD-G-ROM and it has always, with different values of r, the same error order.

We also present the DNS, POD-G-ROM and AV-POD-G-ROM with different values of $\tilde{r}$ at the final time $T = 1$, in order to contrast the oscillations between the different models and the improvements that the AV-POD-G-ROM has if we increase the number of $\tilde{r}$.

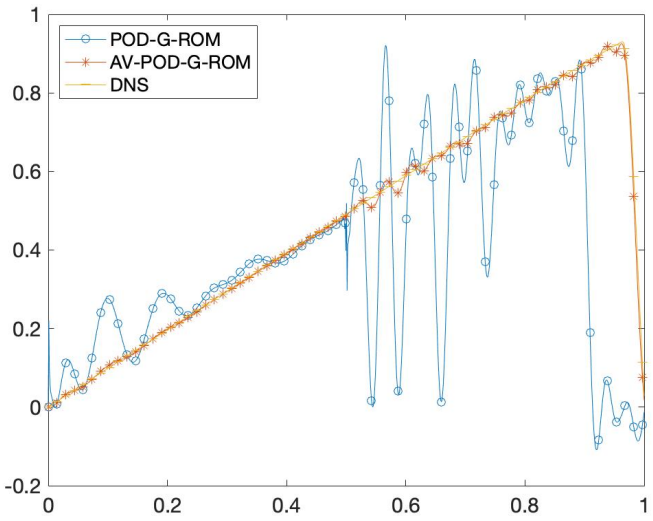


Figure 4.7: Solution curves of DNS, POD-G-ROM and AV-POD-G-ROM ($\tilde{r} = 20$) at final time $T = 1$.

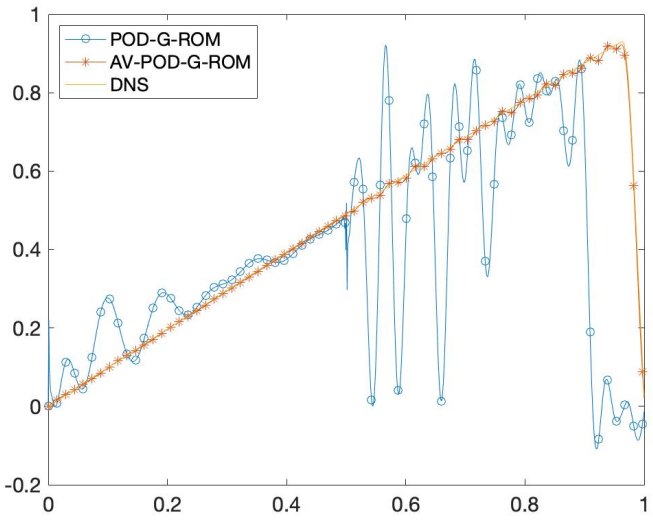


Figure 4.8: Solution curves of DNS, POD-G-ROM and AV-POD-G-ROM ($\tilde{r} = 40$) at final time $T = 1$.

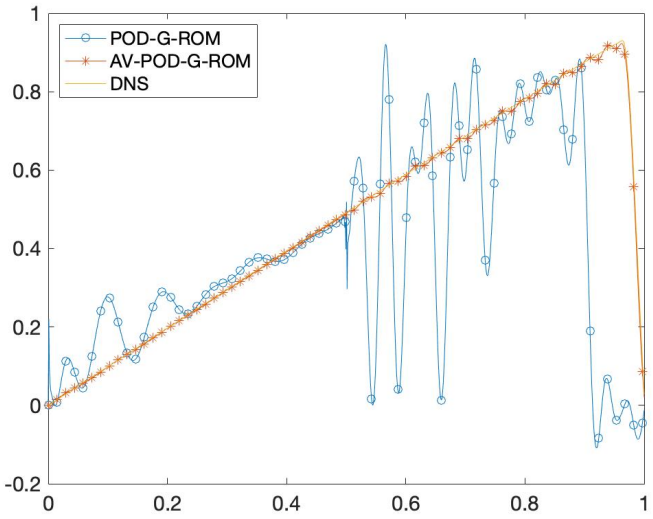


Figure 4.9: Solution curves of DNS, POD-G-ROM and AV-POD-G-ROM ($\tilde{r} = 60$) at final time $T = 1$.

To better compare both models we include the L^2 errors at final time $T = 1$ too.

r	Normalized residual energy $\frac{\sum_{j=r+1}^{1001}\lambda_j}{\sum_{j=1}^{1001}\lambda_j}$	POD	AV-POD(20)	AV-POD(40)	AV-POD(60)
6	1.1e-02	2.8e-03	1.0e-03	1.0e-03	1.0e-03
12	2.2e-03	1.6e-03	8.1e-05	8.0e-05	8.0e-05
21	1.7e-04	8.5e-04	7.4e-06	6.7e-06	6.6e-06
30	1.0e-05	8.3e-04	1.2e-06	4.6e-07	4.9e-07
39	3.0e-06	7.6e-04	1.1e-06	3.8e-07	3.9e-07
48	6.8e-07	7.4e-04	1.1e-06	3.6e-07	3.1e-07

Table 4.2: L^2 errors between ROMs and DNS at final time $T = 1$.

Now, if we observe Table 4.2, we can appreciate that the POD-G-ROM error has a slow decay in its order when we increase the number of modes. On the other hand, the AV-POD-G-ROM rapidly reaches errors of 10^{-7} order.

To study the efficiency of the proposed ROMs, we have included the CPU time of the DNS and the online phase of the POD-G-ROM (4.4) and AV-POD-G-ROM (4.7).

CPU TIME	
	Time
DNS	6.014s
POD-G-ROM	0.257s
AV-POD-G-ROM($\tilde{r} = 20$)	0.272s
AV-POD-G-ROM($\tilde{r} = 40$)	0.302s
AV-POD-G-ROM($\tilde{r} = 60$)	0.358s

Table 4.3: CPU time of DNS, POD-G-ROM and AV-POD-G-ROM with different values of $\tilde{r}$

From Table 4.3, we observe a saving in CPU time of more than one order of magnitude for this simple setting in 1D, which holds for all ROMs. Note that in more complex settings (see next Section), we can arrive at three orders of magnitude of saving in CPU time for ROMs with respect to FOMs. We emphasize that with the AV-POD-G-ROM with $\tilde{r} = 60$ we can obtain an L^2 error of 10^{-7} order, in contrast to the error of POD-G-ROM of 10^{-4} order, although the time difference is insignificant.

Chapter 5

The POD for the Navier-Stokes equations

In this chapter, we are going to study the numerical approximation of incompressible flows with the POD-ROMs. For this purpose, we consider the Navier-Stokes equations:

$$\begin{cases} u_t - \nu\Delta u + (u\cdot\nabla)u + \nabla p = f & \text{in } \Omega\times(0,T), \\ \nabla\cdot u = 0 & \text{in } \Omega\times(0,T), \end{cases} \tag{5.1}$$

in a boundary domain $\Omega\subset\mathbb{R}^d, d\in\{2,3\}$, and with initial condition $u(0)=u^0$. In the above equation, u is the velocity field, p the pressure, $\nu>0$ the kinematic viscosity coefficient and f the forcing term. It is necessary to complete (5.1) with boundary conditions, so for simplicity we are going to consider Dirichlet homogeneous condition.

In order to give a variational formulation of problem (5.1), let us consider the velocity space:

$$X=(H_0^1)^d=\{v\in(H^1(\Omega))^d : v=0 \text{ on } \partial\Omega\},$$

and the pressure space:

$$Q=L_0^2(\Omega)=\{q\in L^2(\Omega) : \int_\Omega q dx=0\}.$$

The weak formulation of (5.1) reads:

Given $f \in L^2(H^{-1})$, find $u : (0,T) \to X, p : (0,T) \to Q$ such that:

$$\begin{cases} \dfrac{d}{dt}(u,v) + (u \cdot \nabla u, v) + \nu(\nabla u, \nabla v) - (p, \nabla \cdot v) = \langle f, v \rangle & \forall v \in X \text{ in } \mathcal{D}'(0,T) \\ (\nabla \cdot u, q) = 0 & \forall q \in Q \\ u(0) = u_0, \end{cases} \tag{5.2}$$

where $\langle \cdot, \cdot \rangle$ it is the duality between X and its dual X' and $\mathcal{D}'(0,T)$ is the space of distributions in $(0,T)$.

Once obtained the weak formulation, we are going to give a FE approximation of (5.2). Let $\{T_h\}_{h>0}$ be a family of regular triangulations of $\bar{\Omega}$. For any mesh cell $k \in T_h$, the diameter will be called h_k and $h = \max_{k \in T_h} h_k$. The FE space for velocity and pressure are $X_h \subset X$ and $Q_h \subset Q$ respectively. Therefore the FE approximation of (5.2) is:

Find $(u_h, p_h) : (0,T) \to X_h \times Q_h$ such that:

$$\begin{cases} \dfrac{d}{dt}(u_h, v_h) + (u_h \cdot \nabla u_h, v_h) + \nu(\nabla u_h, \nabla v_h) - (p_h, \nabla \cdot v_h) = \langle f, v_h \rangle & \text{in } \mathcal{D}'(0,T) \\ (\nabla \cdot u_h, q_h) = 0 & \text{a.e. in } (0,T) \\ u_h(0) = u_{0h}, \end{cases} \tag{5.3}$$

for any $(v_h, q_h) \in X_h \times Q_h$, and u_{oh} is some stable approximation to u_0 into X_h.

Now, if we collect snapshots of the velocity field at certain time instances and we apply a standard POD-Galerkin ROM, we obtain an approximation of the solution of problem (5.1):

$$u(x,t) \approx u_r(x,t) = \sum_{j=1}^{r} a_j(t)\varphi_j(x),$$

where $X^r = span\{\varphi_1, \ldots, \varphi_r\}$ is the space for the POD setting and $\{a_i(t)\}_{i=1}^r$ are the time coefficients. Then, if we replace u with u_r in (5.1), using the Galerkin method and projecting the resulted equations onto X^r, we obtain the POD-G-ROM for the Navier-Stokes equation (5.1):

$$(\frac{\partial u_r}{\partial t}, \varphi) + (u_r \cdot \nabla u_r, \varphi) + \nu(\nabla u_r, \nabla \varphi) = 0 \quad \forall \varphi \in X^r. \tag{5.4}$$

We note that, the POD modes are linear combinations of the snapshots, and thus belong to the space of discrete divergence-free functions. Also, without loss of generality, $f \equiv 0$.

Remark 3. *Since the POD velocity modes are linear combinations of the velocity snapshots, the POD velocity modes satisfy the boundary conditions in* (5.1)*. This is because of the particular choice we have made at the beginning to work with homogeneous Dirichlet boundary conditions. In general, one has to manipulate the velocity snapshots set. This is the case, for instance, of steady-state non-homogeneous Dirichlet boundary conditions, for which is preferable to consider a proper lift in order to generate POD velocity modes for the lifted velocity snapshots, satisfying homogeneous Dirichlet boundary conditions. This would lead to work with centered-trajectory method in the POD-ROM setting [16].*

Remark 4. *In* (5.4)*, the pressure term vanishes due to the fact that the POD modes are weakly divergence-free. However, there exists different methods to recover the pressure, two of the most common methods are explained in [28]. These methods are used to solve the stability issues too, due to the fulfillment of the inf-sup condition in the reduced order framework. The first one enriches the velocity space adding more modes, which is called supremizer enrichment. The second one consists of adding a Possion equation for pressure and new boundary conditions for pressure too. We also consider another method, the Local Projection Stabilization(LPS) [24, 22]. This method leads to a coupled velocity-pressure POD-ROM that uses pressure modes as well to compute the reduced order pressure by adding a LPS term to the reduced equations (see Section 6.2.2).*

From the POD-G-ROM (5.4), we obtain an autonomous system for the coefficient $a(t)$:

$$\dot{a}\,(t) = Aa + aCa^T, \tag{5.5}$$

where A,C correspond to the linear and quadratic term in the numerical discretization of the Navier-Stokes equations, respectively.

The finite dimensional system (5.5) can be written componentwise as follow:

$$\dot{a}_i\,(t) = \sum_{j=1}^{r} A_{ij}a_j(t) + \sum_{j=1}^{r}\sum_{m=1}^{r} C_{ijm}a_m(t)a_j(t), \tag{5.6}$$

where

$$A_{ij} = \nu(\nabla\varphi_i, \nabla\varphi_j), \tag{5.7}$$

$$C_{ijm} = -(\varphi_j \cdot \nabla\varphi_m, \varphi_i). \tag{5.8}$$

5.1 S-POD-G-ROM

The POD-G-ROM is an efficient tool for many applications, for example it works well on diffusion-dominated and laminar fluid flows. However, the POD-G-ROM for convection-dominated or turbulent flows yields inaccurate result, due to the effect of the discarded POD modes $\{\varphi_{r+1}, \dots, \varphi_d\}$ on the retained ones, which is not taken into account. Indeed, although the disregarded modes do not contain a significant amount of the system's kinetic energy, they have a significant role in the dynamics of the reduced-order system. For this reason, we are going to introduce a closure model to address this issue, the Smagorinsky POD-G-ROM (S-POD-G-ROM), see [25].

This closure model consists in adding, both to the offline and online stage, an eddy viscosity:

$$\nu_S := (C_s\delta)^2\|\nabla u\|_F, \tag{5.9}$$

where C_s is the Smagorinsky constant, δ is the lengthscale and $\|\nabla u\|_F$ is the Frobenius norm of ∇u. Adding this into (5.3), we obtain the following problem:

$$\begin{cases} \text{Find } (u_h, p_h) : (0,T) \to X_h \times Q_h \text{ such that} & \\ \dfrac{d}{dt}(u_h, v_h) + (u_h \cdot \nabla u_h, v_h) + \nu(\nabla u_h, \nabla v_h) & \\ +(C_s\delta)^2(\|\nabla u_h\|_F \nabla u_h, \nabla v_h) - (p_h, \nabla \cdot v_h) = \langle f, v_h\rangle & \text{in } \mathcal{D}'(0,T) \\ (\nabla \cdot u_h, q_h) = 0 & \text{a.e. in } (0,T) \\ u_h(0) = u_{0h}. & \end{cases} \tag{5.10}$$

Now if we apply the same process for the online stage as in the previous section, we obtain the new autonomous system (S-POD-G-ROM):

$$\dot{a}\,(t) = (A + \tilde{A})a + aCa^T. \tag{5.11}$$

This finite dimensional system (5.11) can be written componentwise as follows:

$$\dot{a}_i\,(t) = \sum_{j=1}^{r}(A_{ij} + \tilde{A}_{ij})a_j(t) + \sum_{j=1}^{r}\sum_{m=1}^{r} C_{ijm}a_m(t)a_j(t) \quad \forall i = 1, \dots, r, \tag{5.12}$$

where

$$\tilde{A}_{ij} = -(C_s\delta)^2(\|\nabla u_r\|_F \nabla\varphi_j, \varphi_i). \tag{5.13}$$

As presented in [32], the S-POD-G-ROM has a disadvantage, which is to compute the matrix $\tilde{A}$ (depending on $\|\nabla u_r\|_F$) at every time step. Therefore, we

propose to employ the DEIM algorithm (3.5.1) to reduce the computational complexity. So, we obtain:

$$\|\nabla u\|_F = \sum_{j=1}^{\tilde{r}} \alpha_j(t)\tilde{\varphi}_j, \tag{5.14}$$

where $\tilde{X}^{\tilde{r}} = \text{span}\{\tilde{\varphi}_1, \dots \tilde{\varphi}_{\tilde{r}}\}$ and $\alpha_j(t)$ are the time coefficients. Consequently, if we replace (5.14) into our autonomous system (5.11), we obtain:

$$\dot{a} = Aa + \alpha^T Da + a^T Ca. \tag{5.15}$$

This finite dimensional system (5.15) can be written componentwise as follow:

$$\dot{a}_i\ (t) = \sum_{j=1}^{r} A_{ij}a_j(t) + \sum_{j=1}^{\tilde{r}}\sum_{m=1}^{r} D_{ijm}\alpha_j(t)a_m(t) + \sum_{j=1}^{r}\sum_{m=1}^{r} C_{ijm}a_j(t)a_m(t) \quad \forall i = 1, \dots, r, \tag{5.16}$$

where

$$D_{ijm} = -(C_s\delta)^2(\tilde{\varphi}_j\nabla\varphi_m, \nabla\varphi_i). \tag{5.17}$$

5.2 Numerical Studies

In this section, we present numerical results for the S-POD-G-ROM we introduced in the previous section. The numerical experiments are performed on the benchmark problem of the 2D unsteady flow around a cylinder with circular cross-section [14], and we take the same parameters as in [2] and [22].

5.2.1 Setup for numerical simulations

Following [29], the computational domain is given by a rectangular channel with a circular hole (see Figure (5.1)):

$$\Omega = \{(0, 2.2) \times (0, 0.41)\} \setminus \{x : (x - (0.2, 0.2))^2 \leq 0.05^2\}.$$

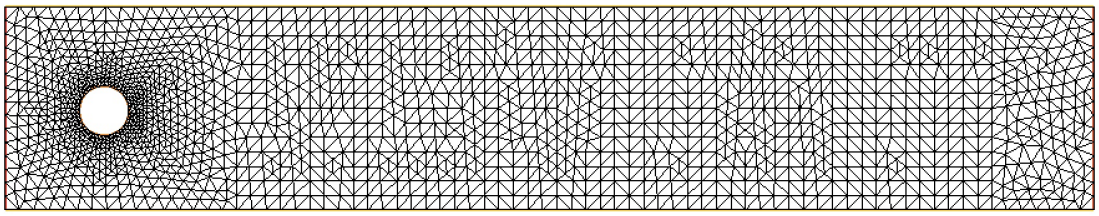


Figure 5.1: Computational grid

No slip boundary conditions are prescribed on the horizontal walls and on the cylinder, and a parabolic inflow profile is provided at the inlet

$$u(0, y, t) = (4U_m y(A - y)/A^2, 0)^T,$$

with $U_m = u(0, A/2, t) = 1.5\ m/s$ and $A = 0.41\ m$ the channel height. At the outlet, we impose outflow (do nothing) boundary conditions $(\nu\nabla u - pId)n = 0$ with n the outward normal to the domain. The chosen Reynolds number is $Re = 100$, depending on the inflow velocity $\overline{U} = 2U_m/3 = 1m/s$, the cylinder diameter $D = 0.1m$, and the kinematic viscosity $\nu = 10^{-3}\ m^2/s$ and there is no external forcing, $f = 0\ m/s^2$. In the fully developed periodic regime, a vortex shedding can be observed behind the obstacle, resulting in the well-known von Kármán vortex street, see Figure 5.2.

For the evaluation of computational results, we are interested in studying the temporal evolution of the following quantities of interest. The first one is the kinetic energy of the flow, given by:

$$E_{Kin} = \frac{1}{2}\|u\|^2_{L^2}.$$

The others quantities of interest are the drag and lift coefficients. In the present work, they are computed as volume integrals [18]:

$$c_D = -\frac{2}{D\overline{U}}[(\partial_t u, v_D) + (u \cdot \nabla u, v_D) + \nu(\nabla u, \nabla v_D) - (p, \nabla \cdot v_D)],$$

$$c_L = -\frac{2}{D\overline{U}}[(\partial_t u, v_L) + (u \cdot \nabla u, v_L) + \nu(\nabla u, \nabla v_L) - (p, \nabla \cdot v_L)],$$

for arbitrary test functions $v_D, v_L \in H^1$ such that $v_D = (1, 0)^T$ on the boundary of the cylinder and vanishes on the other boundaries, $v_L = (0, 1)^T$ on the boundary of the cylinder and vanishes on the other boundaries. In the actual computations we have used the approach in ([33]), where the pressure term is not necessary to compute c_D, c_L, since we used discretely divergence-free test functions v_D, v_L obtained by Stokes projection.

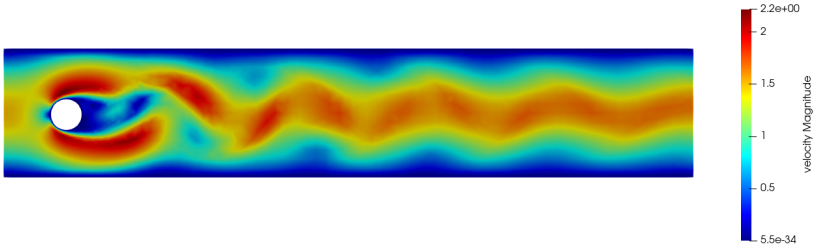


Figure 5.2: Final velocity FOM.

5.2.2 FOM and POD modes

Our aim is to compare the performance of the S-POD-G-ROM with the POD-G-ROM.

The numerical method to get the snapshots is the Smagorinsky method (5.1), with a spatial discretization using $\mathbb{P}^2 - P^1$ FE for the pair velocity-pressure on a relatively coarse computational grid (see Figure (5.1), $h = 6,6 \times 10^{-2}$), resulting in 6175 d.o.f for velocities and in 1735 d.o.f for pressure. For the time discretization, a semi-implicit Backward Differentation Formula of order 2 (BDF2) has been applied, which guarantees a good balance between numerical accuracy and computational complexity [1]. In particular, the discrete time derivative has been approximated by the operator D_t^2:

$$D_t^2 u_h^{n+1} = \frac{3u_h^{n+1} - 4u_h^n + u_h^{n-1}}{2\Delta t} \qquad n \geq 1, \tag{5.18}$$

and we have considered an extrapolation for the convection velocity: $\tilde{u}_h^n = 2u_h^n - u_h^{n-1}, n \geq 1$, in order to guarantee a second order accuracy in time. At the inizialization ($n = 0$), we have to consider $u_h^{-1} = u_h^0 = u_{0h}$, being u_{0h} the discrete initial condition, so the time scheme for the first iteration reduces from BDF2 to the semi-implicit Euler method for the first time step $(\Delta t)^0 = (2/3)\Delta t$.

In the FOM simulation, the initial condition is a zero velocity field and the time step is $\Delta t = 2 \cdot 10^{-3} s$. Time integration is performed till a final time $T = 7s$. For $Re = 100$, in the time period $[0, 5]s$ the flow is expected to develop to full extent, including a subsequent relaxation time. We have plotted the time evolution of drag and lift coefficients in Figure (5.3) and kinetic energy in Figure (5.4) to show this.

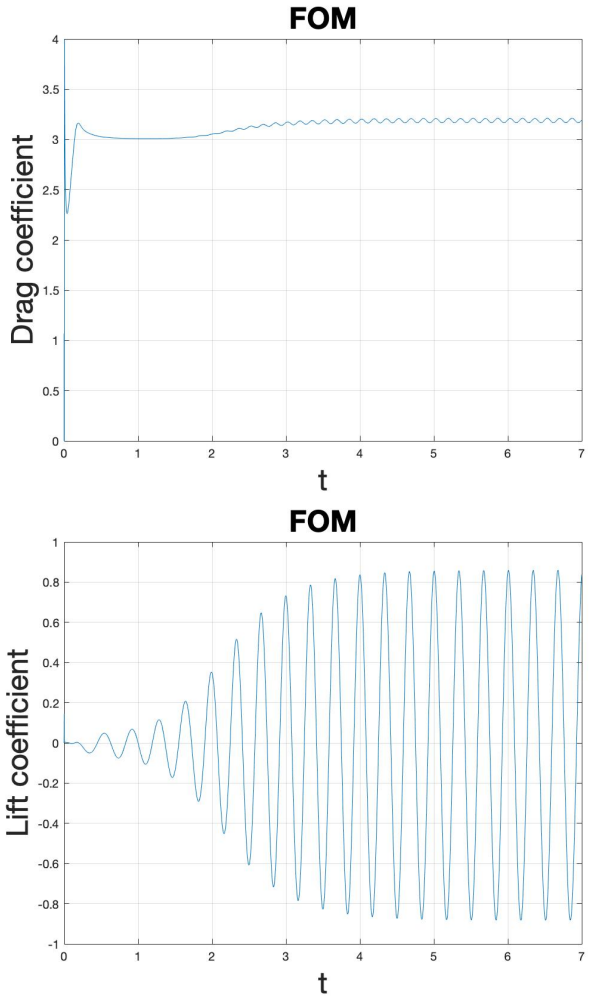


Figure 5.3: Temporal evolution of drag and lift coefficients.

The POD modes are generated in L^2 by the method of snapshots with centered-trajectories by storing every FOM solution from $T = 5s$ to one period. The full period length of the statistically steady state is $0.332s$ for $Re = 100$, so we collect 167 snapshots. In Figure (5.5) we can see the decay of the eigenvalues associated to the correlation matrix of the velocity, and for the DEIM algorithm applied to

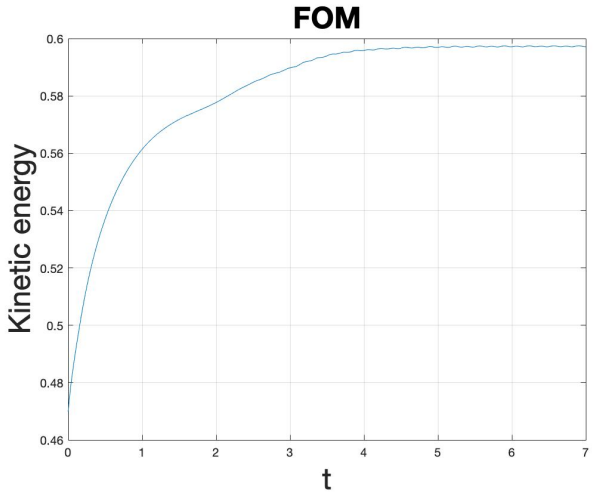


Figure 5.4: Temporal evolution of kinetic energy.

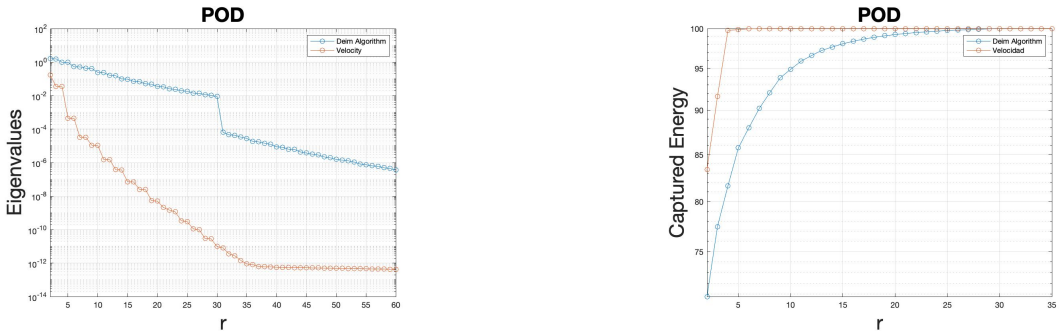


Figure 5.5: Decay of the eigenvalues and captured energy.

the eddy viscosity, and the captured energy calculated by $100\sum_{k=1}^{r}\lambda_k/\sum_{k=1}^{d}\lambda_k$ where λ_k are the corresponding eigenvalues. Note that with 5 POD velocity modes, we already capture more than 99% of the energy, while for the DEIM algorithm we need around 20 modes.

Once the POD modes are generated, the S-POD-G-ROM and POD-G-ROM are constructed as it is explained in section 5.1, using the same time discretization as the FOM and run in $[5, 7]s$ for $Re = 100$ using only up to 7 POD velocity modes and 30 DEIM modes in order to get good results. We also decided to employ an adaptive in time algorithm for the Smagorinsky coefficient C_s, by adjusting its dissipation with the FOM energy, see next section. This caused a considerable improvement in the results.

5.2.3 Adaptive in time algorithm for the Smagorinsky coefficient C_s

In this section, we are going to describe the adaptive in time algorithm proposed, following ([22]). This is the first time that it is applied in the S-POD-G-ROM. The strategy that this algorithm entails is adjusting the Smagorinsky constant in order to remove dissipation if the ROM energy is too small and add disipation if it is large with respect to FOM energy. The algorithm starts with an initial value $\bar{C}_s$ that we obtained by minimizing the L^∞ error in time with respect to the snapshots energy computed. As we only have one period, we have to repeat for the rest of periods. The detailed algorithm is as follow:
Algorithm (ROM with adaptive in time C_s)

1. Initialize the online $C_s = \bar{C}_s$.
2. Set $C_{min} > 0$ the minimum value that C_s can reach in the algorithm.
3. Set F to be the frequency in the number of time steps to adapt C_s.
4. Set δ to be the adjustment size to change C_s.
5. Set *tol* to be the tolerance chosen for making a change to C_s.
6. For time step $n = 1, 2, \ldots$
   if mod $(n, F) == 0$
   - Compute $E_{kin}^{diff} = \frac{1}{2}(\|u_d^n\|_{L^2}^2 - \|u_h{}^{\text{mod }(n,M)}\|_{L^2}^2)$, where M is the number of snapshots collected.
   - if $E_{kin}^{diff} > tol$ set $C_s = \max\{C_{min}, C_s + \delta\}$,
     Else if $E_{kin}^{diff} < -tol$, set $C_s = \max\{C_{min}, C_s - \delta\}$.

   end
   Recompute u_r^n

5.2.4 Numerical Results

All the ROMs models(POD-G-ROM, S-POD-G-ROM, Adapt-S-POD-G-ROM) are tested in the interval $[5, 7]s$, in order to show the ability of ROMs to predict in time six period of lift coefficient. Note that the fact that the snapshots are collected just for the first period increases the difficulty of simulations. With this,

we want to show the improvements of the S-POD-G-ROM and Adapt-S-POD-G-ROM over the general POD-G-ROM. Thus, we decided to test the temporal evolution of the drag and lift coefficients and kinetic energy.

Before showing the numerical results of the quantities of interest, we show the first four POD modes of velocity in Figure (5.6). For S-POD-G-ROM the value of C_s is 10^{-2} and for Adapt-S-POD-G-ROM the values of the parameters of the adaptive method are $\bar{C}_s = 10^{-2}$, $C_{\min} = 10^{-4}$, $\delta = 10^{-3}$, $F = 3$ and $tol = 10^{-5}$.

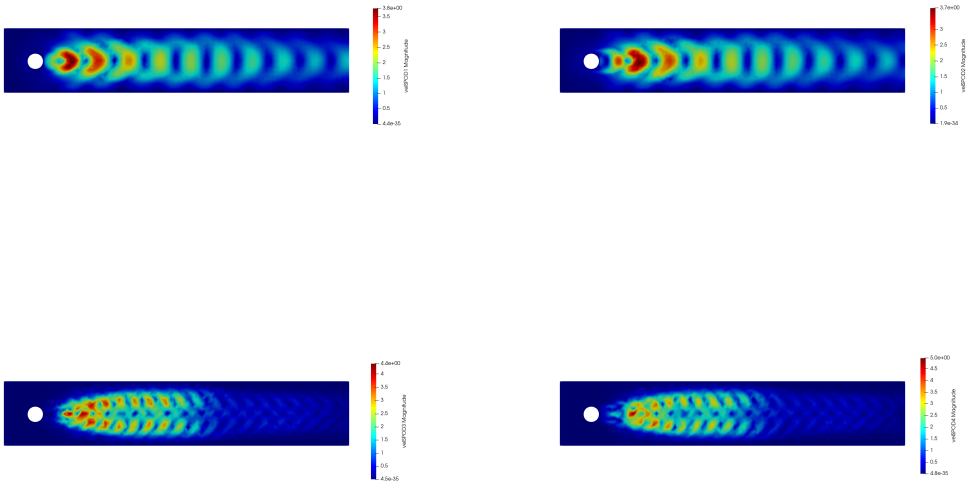


Figure 5.6: First four POD modes of velocity.

First, we are going to show the numerical results for kinetic energy using $r = 7$ modes are shown in Figure (5.7), where we display a comparison of the the three ROMs models.

In order to complete these results, we also show the temporal evolution of the absolute error in kinetic energy and of the adapted Smagorinsky coefficient C_s in Figure (5.8) .

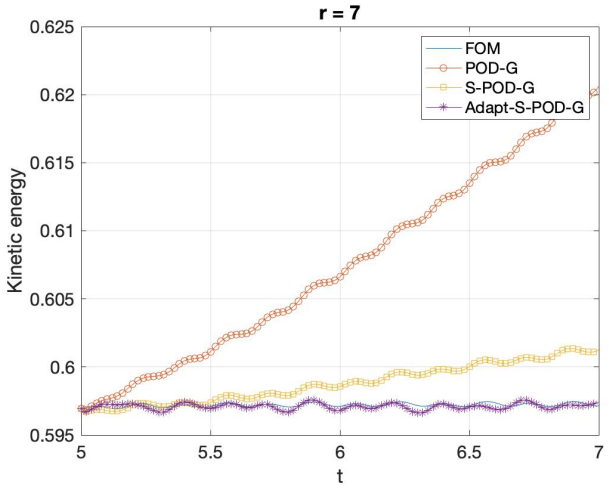


Figure 5.7: Time evolution of the kinetic energy for FOM, POD-G-ROM, S-POD-G-ROM and Adapt-S-POD-G-ROM for $r = 7$.

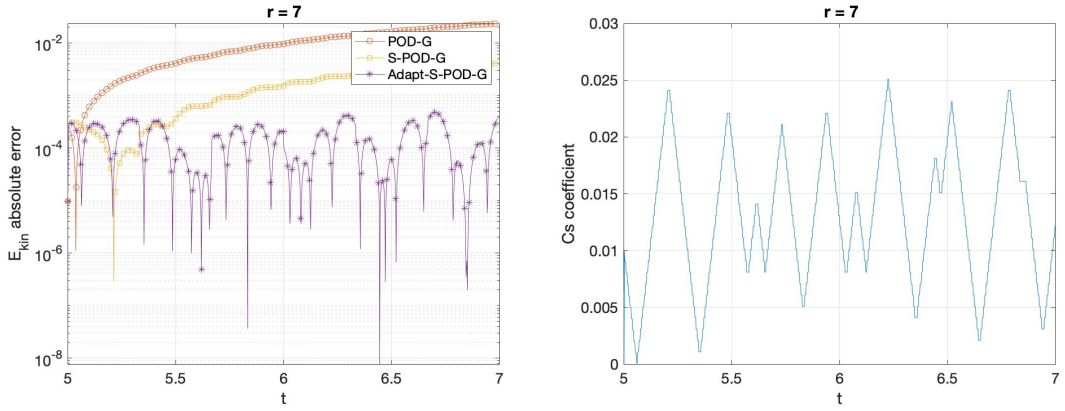


Figure 5.8: Temporal evolution of absolute error in kinetic energy of POD-G-ROM, S-POD-G-ROM and Adapt-S-POD-G-ROM with respect to FOM on the left and the value of C_s coefficient in the adaptive method on the right using $r = 7$.

From Figure (5.8), we observe that error levels are maintained below 10^{-3} for Adapt-S-POD-G-ROM, while for S-POD-G-ROM this holds mainly for the first period. The POD-G-ROM already loses this property even in the first period. This means that the Adapt-S-POD-G-ROM is the only that guarantees good results with a few modes in a predictive regime, which is not the case for the other two, being the S-POD-G-ROM at least accurate in the reconstrution regime.

A similar conclusion holds for the numerical results for drag and lift coefficients using $r = 7$ modes shown in Figures (5.9), (5.10).

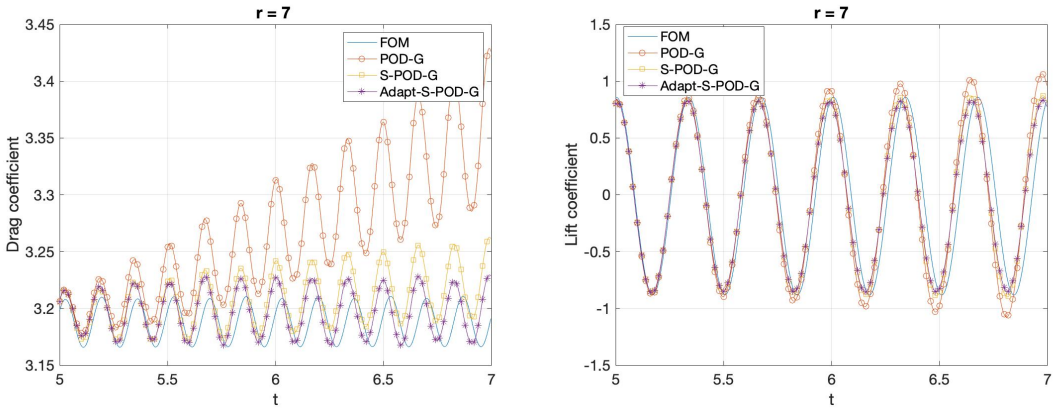


Figure 5.9: Temporal evolution of drag coefficient (left) and lift coefficient (right) for FOM, POD-G-ROM, S-POD-G-ROM and Adapt-S-POD-G-ROM for $r = 7$.

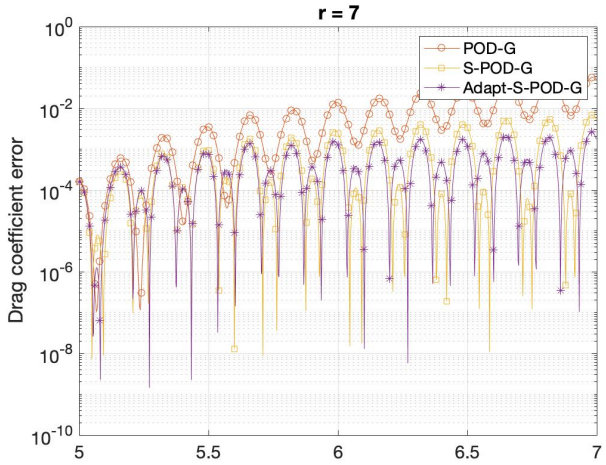


Figure 5.10: Temporal evolution of drag coefficients error for FOM, POD-G-ROM, S-POD-G-ROM and Adapt-S-POD-G-ROM for $r = 7$.

Chapter 6

Conclusion and outlook

6.1 Conclusion

In this work, we have considered POD-ROMs for closure modeling. We started with the AV(Artificial Viscosity)-POD-G-ROM for the Burgers equation, and then moved to the S(Smagorinsky)-POD-G-ROM for the Navier-Stokes equations. In both cases, we applied an hyperreduction technique such as the DEIM to reduce the computational complexity for the arising nonlinear tems.

To sum up, the obtained results show how these models outperform the standard POD-G-ROM in terms of accuracy, and also their efficiency in computing convection-dominated flows. Indeed, with few basis functions we are able to compute accurate results, achieving a significant speed-up over the FEM (high-fidelity) simulations.

6.2 Outlook

In this section, we present some future research directions based on this work that could be pursued:

6.2.1 NSE error estimates

In Chapter 4, we performed the numerical analysis of the AV-POD-G-ROM for the Burgers equation, by mainly deriving its error estimates. One could think to extend this analysis to the S-POD-G-ROM for the Navier-Stokes equations, which is not being performed yet in the literature up to our knowledge. In this case, a careful treatment of the pressure should be taken into account, as outlined in the next subsection too.

6.2.2 Pressure recovery and turbulent flows

In chapter 5, we already mentioned several methods to recover the pressure in fluid dynamics reduced order simulations. A recent proposal has been to apply a Local Projection Stabilization (LPS) technique in the POD-ROM framework [24, 22].

This method is a velocity-pressure ROM that uses pressure modes as well to compute the reduced order pressure needed for instance in the computation of relevant engineering quantities. It circumvents the standard discrete inf-sup condition for the POD velocity-pressure spaces, whose fulfillment can be rather expensive in realistic applications in Computational Fluid Dynamics (CFD). Also, the velocity modes do not have to be either strongly or weakly divergence-free, which allows the use of snapshots generated, for instance, with penalty or projection-based stabilized methods.

In particular, in my ongoing research (see Appendix A), I obtained some new results combining the LPS for pressure stability and the RBF technique (see section 3.5.2) to approximate the eddy viscosity, to tackle the increase of Reynolds number in multi-parametric POD-ROMs [13].

We believe this is a first step towards the challenging simulation of realistic turbulent flows by ROMs, which would be the goal of future research.

Appendix A

Ongoing research

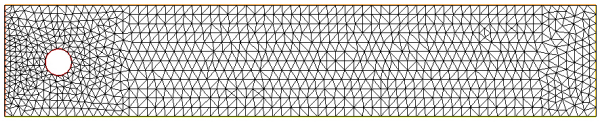


Figure A.1: Mesh of the problem (Flow past cylinder).

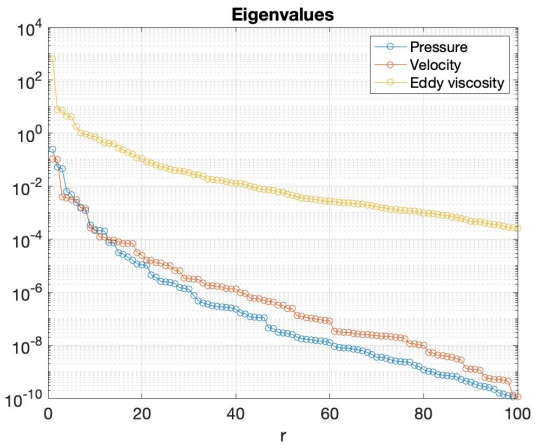


Figure A.2: Decay of POD eigenvalues of velocity, pressure and eddy viscosity.

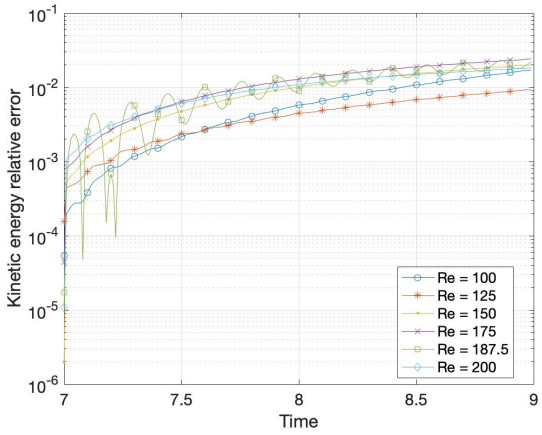


Figure A.3: Kinetic energy relative error for different Reynolds values including $Re = 187.5$ (not considered in the snapshots sample).

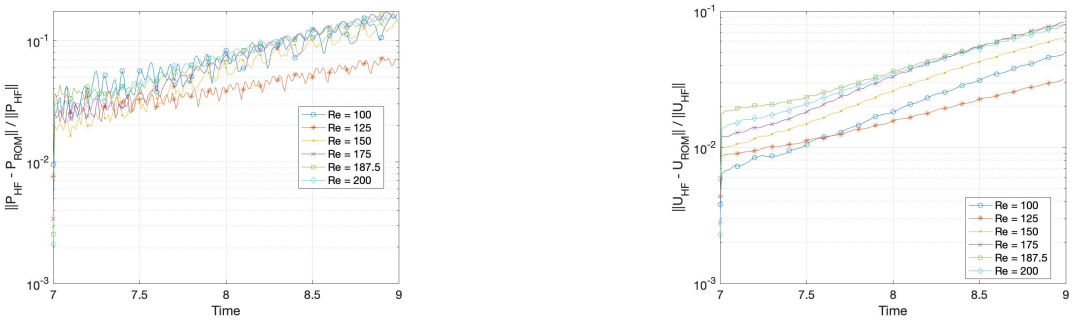


Figure A.4: Temporal evolution of L^2 norm of relative error of velocity (right) and pressure (left).

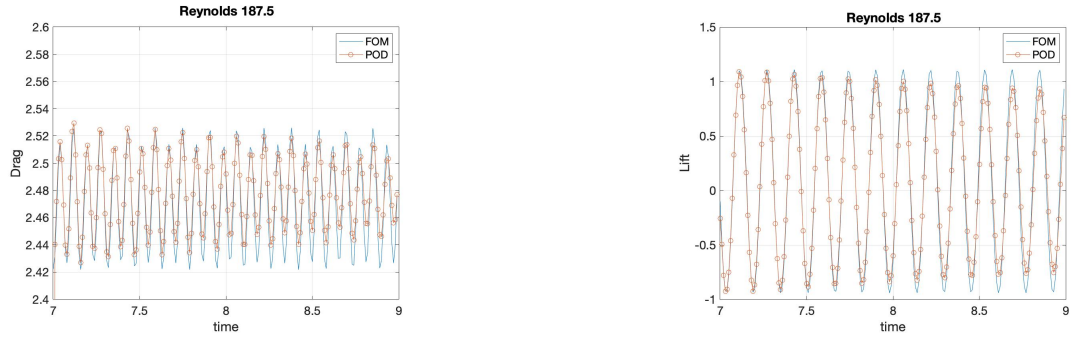


Figure A.5: Drag and Lift coefficient for $Re = 187.5$.

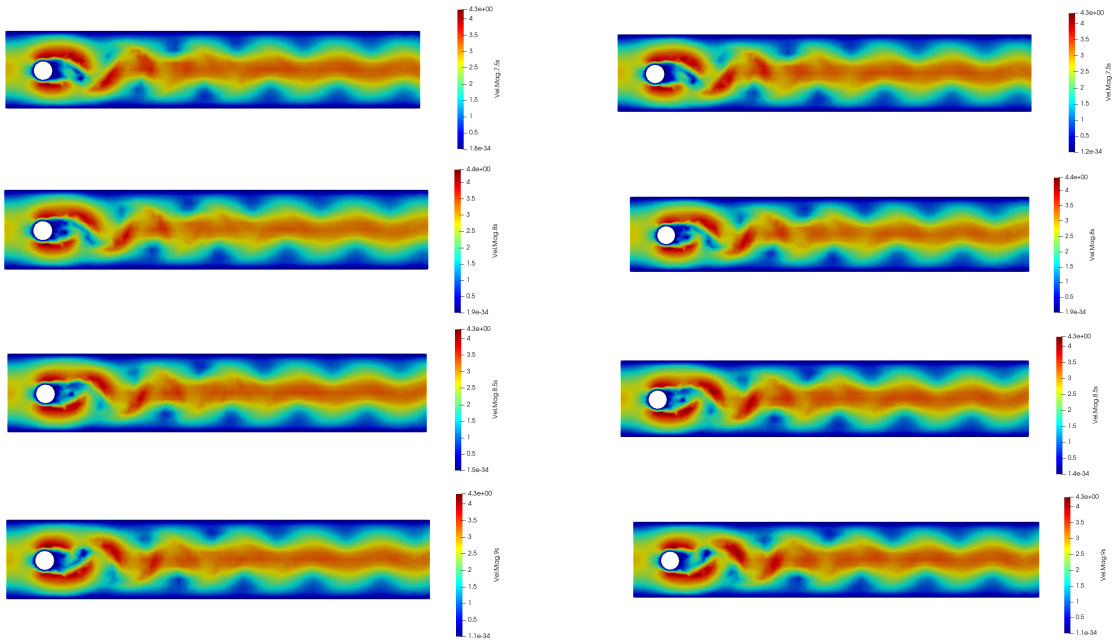


Figure A.6: Comparison of the velocity field for FOM (left) and ROM (right) for $Re = 187.5$. The fields are depicted for different time instant equal to t = 7.5 s , 8 s , 8.5 s and 9s and increasing from top to bottom. The ROM solutions are obtained with 8 modes for velocity and pressure and 60 modes for eddy viscosity.

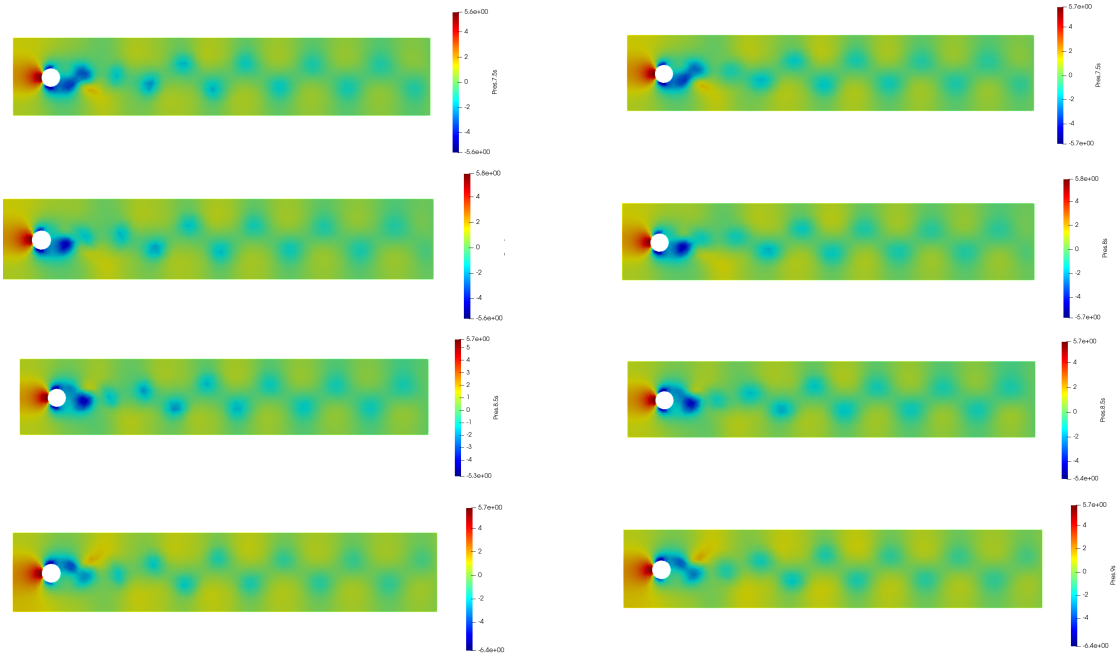


Figure A.7: Comparison of the pressure field for FOM (left) and ROM (rigth) for $Re = 187.5$. The fields are depicted for different time instant equal to t = 7.5 s , 8 s , 8.5 s and 9s and increasing from top to bottom. The ROM solutions are obtained with 8 modes for velocity and pressure and 60 modes for eddy viscosity.

Bibliography

[1] N. Ahmed and S. Rubino, *Numerical comparisons of finite element stabilized methods for a 2d vortex dynamics simulation at high reynolds number*, Computer Methods in Applied Mechanics and Engineering, (2019), pp. 191–212.

[2] B. Archilla, J. Novo, and S. Rubino, *Error analysis of proper orthogonal decomposition data assimilation schemes for the navier-stokes equations*, (2020).

[3] M. Azaïez, T. C. Rebollo, and S. Rubino, *Streamline derivative projection-based pod-rom for convection-dominated flows. part i : Numerical analysis*, 2017.

[4] M. Barrault, Y. Maday, N. Nguyen, and A. Patera, *An 'empirical interpolation' method: Application to efficient reduced-basis discretization of partial differential equations*, Comptes Rendus Mathematique, 339 (2004), p. 667–672.

[5] M. C. Bramwell, *Introductory functional analysis with applications*, The Mathematical Gazette, 63 (1979), p. 137–138.

[6] S. Chaturantabut and D. C. Sorensen, *Discrete empirical interpolation for nonlinear model reduction*, in Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference, 2009, pp. 4316–4321.

[7] L. Cordier, *Réduction de dynamique par décomposition orthogonale aux valeurs propres (pod 1)*, 2006.

[8] J. Edmonds, *Matroids and the greedy algorithm*, Mathematical programming, 1 (1971), pp. 127–136.

[9] L. Evans, *Partial Differential Equations*, Graduate studies in mathematics, American Mathematical Society, 2010.

[10] K. P. F.R.S., *Liii. on lines and planes of closest fit to systems of points in space*, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 2 (1901), pp. 559–572.

[11] G. Golub and C. Van Loan, *Matrix Computations*, Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, 2013.

[12] N. J. Higham, *Matrix nearness problems and applications*, 1989.

[13] S. Hijazi, G. Stabile, A. Mola, and G. Rozza, *Data-driven pod-galerkin reduced order model for turbulent flows*, Journal of Computational Physics, 416 (2020), p. 109513.

[14] E. H. Hirschel, *Flow simulation with high-performance computers ii*, AIAA Journal, 35 (1996), pp. 1560–1560.

[15] C. Homescu, L. R. Petzold, and R. Serban, *Error estimation for reduced-order models of dynamical systems*, SIAM Rev., 49 (2007), p. 277–299.

[16] T. Iliescu, V. John, S. Schyschlowa, and D. Wells, *Supg reduced order models for convection-dominated convection-diffusion-reaction equations*, 09 2014.

[17] T. Iliescu and Z. Wang, *Are the snapshot difference quotients needed in the proper orthogonal decomposition?*, 2013.

[18] V. John, *Reference value for drag and lift of a two-dimensional time dependent flow around cylinder*, International Journal for Numerical Methods in Fluids, 44 (2004), pp. 777 – 788.

[19] K. Kunisch and S. Volkwein, *Galerkin proper orthogonal decomposition methods for parabolic problems*, Numerische Mathematik, 90 (2001), pp. 117–148.

[20] Z. Luo, J. Chen, I. M. Navon, and X. Yang, *Mixed finite element formulation and error estimates based on proper orthogonal decomposition for the nonstationary navier–stokes equations.*

[21] C. Mou, B. Koc, O. San, L. G. Rebholz, and T. Iliescu, *Data-driven variational multiscale reduced order models*, Computer Methods in Applied Mechanics and Engineering, 373 (2021), p. 113470.

[22] J. Novo and S. Rubino, *Error analysis of proper orthogonal decomposition stabilized methods for incompressible flows*, SIAM Journal on Numerical Analysis, 59 (2020), pp. 334–369.

[23] A. Quarteroni, A. Manzoni, and F. Negri, *Reduced basis methods for partial differential equations: An introduction*, 01 2015.

[24] S. Rubino, *Numerical analysis of a projection-based stabilized pod-rom for incompressible flows*, 2019.

[25] P. Sagaut and Y.-T. Lee, *Large eddy simulation for incompressible flows: An introduction. scientific computation series*, Applied Mechanics Reviews, 55 (2002), pp. 115–.

[26] L. Sirovich, *Turbulence and the dynamics of coherent structures. i - coherent structures. ii - symmetries and transformations. iii - dynamics and scaling*, Quarterly of Applied Mathematics - QUART APPL MATH, 45 (1987).

[27] L. SIROVICH, *Turbulence and the dynamics of coherent structures part iii: Dynamics and scaling*, Quarterly of Applied Mathematics, 45 (1987), pp. 583–590.

[28] G. Stabile and G. Rozza, *Finite volume pod-galerkin stabilised reduced order methods for the parametrised incompressible navier–stokes equations*, Computers and Fluids, 173 (2018), pp. 273–284.

[29] S. Turek, *Recent benchmark computations of laminar flow around a cylinder*, (2000).

[30] S. Volkwein, *Proper orthogonal decomposition: Theory and reduced-order modelling*, Lecture Notes, University of Konstanz, (2012).

[31] ——, *Proper orthogonal decomposition methods for partial differential equations*, SIAM Review, 63 (2021), pp. 242–244.

[32] Z. Wang, *Reduced-Order Modeling of Complex Engineering and Geophysical Flows: Analysis and Computations*, PhD thesis, Faculty of the Virginia Polytechnic Institute and State University, 4 2012.

[33] C. Zerfas, L. Rebholz, M. Schneier, and T. Iliescu, *Continuous data assimilation reduced order models of fluid flow*, Computer Methods in Applied Mechanics and Engineering, 357 (2019), p. 112596.