Freie Universität Berlin

Fachbereich Mathematik, Informatik und Physik

SELBSTSTÄNDIGKEITSERKLÄRUNG

Name: Merten	(BITTE nur Block- oder Maschinenschrift verwenden.)
Vorname(n): Clara Daisy Emma	
Studiengang: Mathematik, M. Sc.	
Matr. Nr.: 5297282	

Ich erkläre gegenüber der Freien Universität Berlin, dass ich die vorliegende Masterarbeit selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe.

Die vorliegende Arbeit ist frei von Plagiaten. Alle Ausführungen, die wörtlich oder inhaltlich aus anderen Schriften entnommen sind, habe ich als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch bei keiner anderen Universität als Prüfungsleistung eingereicht.

Datum: 28.03.2025 Unterschrift: C. Merten

Master Thesis

Reduced Order Modeling with a Variational Multiscale Method

submitted in fulfillment of the requirements for the degree
M.Sc Mathematics

Submitted by:	Clara Daisy Emma Merten
Born on:	November 25, 1992
Matriculation number:	5297282
Supervisor:	Univ.-Prof. Dr. Volker John
Second Reviewer:	PD Dr. Alfonso Caiazzo

Department of Mathematics and Computer Science
Freie Universität Berlin

March 28, 2025

Contents

1 Introduction **5**
1.1 Convection-Diffusion-Reaction Equation 5
1.2 Variational Formulation and Discretization 7

2 Proper Orthogonal Decomposition **9**
2.1 Mathematical Motivation 10
2.2 Calculation of the POD Basis (Modes) and the Method of Snapshots 12
2.3 Error Representation 15

3 Variational Multiscale (VMS) Method **16**
3.1 Introduction to VMS 16
3.2 VMS for our Application 16

4 VMS-POD Model **18**

5 Critical Review of an Error Analysis **19**
5.1 Preliminaries 19
5.2 Definitions and Conditions 20
5.3 Error Analysis 21

6 Summary of the VMS-POD-Model to Generate the Algorithm Step by Step **26**

7 Implementation **28**
7.1 Finite Element Approximation and Equations 28
7.2 Example and Numerical Studies 33
7.3 Conclusion and Other Approaches 41

Appendix **42**

References **45**

1 Introduction

This thesis is about a *reduced-order model* (ROM) for solving convection dominated *convection-diffusion-reaction equations*. The method was proposed in [16]. The aim is to apply the method to the *Navier-Stokes equations*. One of the most successful techniques for generating ROMs is the *proper orthogonal decomposition* (POD). The idea is to construct an ansatz space of few global basis functions instead of a large number of local *finite element* (FE) basis functions.

If, in the convection-diffusion-reaction equations, the convection term dominates over the diffusion term, stability problems arise with standard *finite element* (FE) methods (small perturbations in the input sometimes cause large changes in the output). As results, non-physical oscillations are encountered, unless the grid size h is very small, or at least locally small enough with sufficient prior knowledge about the solution. This means a very high computational effort. The problems are due to the fact that the solution has different scales. The smallest, the boundary layers, cannot generally be resolved by the given grid. . Therefore a classical POD Galerkin (POD-G) method performs poorly. There are methods to cope with this. One of these approaches is the *variational multiscale* (VMS) method, which takes the different scales into account. This leads to a VMS-POD model which is presented here. Another frequently used method is for example the *streamline-upwind Petrov–Galerkin* (SUPG) stabilization [12]. For the VMS-POD model we use the VMS method which is presented in [18]. It introduces an artificial viscosity which acts only on the 'fine' scales.

1.1 Convection-Diffusion-Reaction Equation

First, we define the *Bochner space* in order to then describe the *convection-diffusion-reaction equation*.

Definition 1.1.1 (Bochner space)**.** *Let X be a Banach space, $\|\cdot\|_X$ the corresponding norm, $[t_1, t_2]$ a time interval and $1 \leq p \leq \infty$. A Bochner space $L^p(t_1, t_2; X)$ is the space of all functions $f : [t_1, t_2] \longrightarrow X$ such that:*

$$\|f\|_{L^p(t_1,t_2;X)} < \infty\,,$$

with

$$\|f\|_{L^p(t_1,t_2;X)} = \begin{cases} \left(\int_{t_1}^{t_2} \|f(t)\|_X^p dt\right)^{\frac{1}{p}} & ,p < \infty, \\ \underset{t_1 \leq t \leq t_2}{esssup} \|f(t)\|_X & ,p = \infty. \end{cases}$$

The *convection-diffusion-reaction* equation is a partial differential equation. In physics, it mainly describes the transport of any scalar quantity. Let $\Omega \subset \mathbb{R}^n$ for $n \in \{1,2,3\}$, be an open set with Lipschitz boundary and $[0,T] \subset \mathbb{R}$ be a time interval. Then, one searches for $u : [0,T] \times \Omega \longrightarrow \mathbb{R}$ such that

$$\begin{aligned} &\partial_t u - \varepsilon \Delta u + \mathbf{b} \cdot \nabla u + cu = f && \text{, in } (0,T] \times \Omega, \\ &u(0,x) = u_0(x) && \text{, in } \Omega, \\ &u(t,x) = 0 && \text{, on } (0,T] \times \partial\Omega. \end{aligned} \tag{1.1}$$

We have the diffusion coefficient $\varepsilon \ll 1$, the convective field $\mathbf{b} \in (L^\infty(0,T;L^\infty(\Omega)))^n$ with $\| \mathbf{b} \|_{(L^\infty(0,T;L^\infty(\Omega)))^n} \gg \varepsilon$, the reaction coefficient $c \in L^\infty(0,T;L^\infty(\Omega))$ and a forcing term $f \in L^2(0,T;L^2(\Omega))$.

See [29] for a good explanation of why you need to be careful with the dominant convection term. This is a so-called multiscale problem where scales of different sizes are involved. What exactly this means can be seen in the following example where we consider the time-independent problem in one dimension with $\Omega = (0,1)$, $c \equiv 0$, $f \equiv 1$ and $\mathbf{b} \equiv 1$ and :

$$\begin{aligned} -\varepsilon u''(x) + u'(x) &= 1\,, \ \forall x \in (0,1)\,, \\ u(0) = u(1) &= 0\,. \end{aligned} \tag{1.2}$$

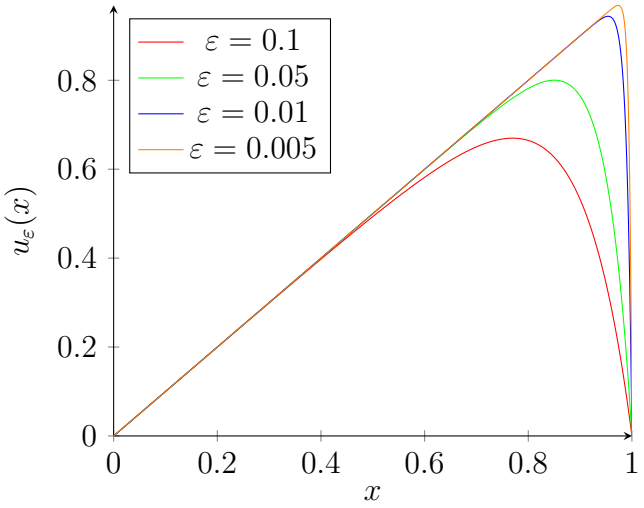

Figure 1.1: solution of (1.2) for different ε

For $\varepsilon = 0$ there is generally no solution for the problem, but for $\varepsilon > 0$ there is. It becomes difficult to handle numerically when $\varepsilon \ll 1$. Such problems with very small parameters that cannot be approximated by 0 are called *singularly perturbed*.

The solution (Figure 1.1) depending on ε is:

$$u_\varepsilon(x) = x - \frac{e^{\frac{1}{\varepsilon}x} - 1}{e^{\frac{1}{\varepsilon}} - 1}\,.$$

It is noticeable that $u_\varepsilon(x)$ assumes a maximum. Let $x_\varepsilon^E \in (0,1)$ be the x value for the maximum. For $\varepsilon \longrightarrow 0$ both x_ε^E and $u_\varepsilon\left(x_\varepsilon^E\right)$ converge to 1. Thus, in an approximation of u_ε, one must do justice to the behavior on the rather large interval $\left[0, x_\varepsilon^E\right)$ and that on the increasingly small interval $\left(x_\varepsilon^E, 1\right]$ for $x_\varepsilon^E = \varepsilon\left[\ln\varepsilon + \ln\left(e^{\frac{1}{\varepsilon}} - 1\right)\right]$. Here one can clearly see the different scales.

1.2 Variational Formulation and Discretization

First, the variational problem for (1.1) is formulated. Let $X = H_0^1(\Omega)$ and $(\cdot,\cdot)$ be the standard $L^2(\Omega)$ inner product. One searches for $u\colon (0,T] \longrightarrow X$ with $u(0,x) = u_0(x) \in X$ such that

$$(\partial_t u, v) + a(u,v) = (f,v) \quad \forall\, v \in X, \tag{1.3}$$

where $a(u,v) = \varepsilon(\nabla u, \nabla v) + (\mathbf{b}\cdot\nabla u, v) + (cu, v)$. (A solution of (1.1) solves (1.3) as well. In general this does not hold for the other direction, without assuming further conditions.)

One approach to solve the problem numerically consists in constructing a suitable finite-dimensional subspace $X^h = span\left\{\beta^1, ..., \beta^M\right\} \subset X$. For this purpose we use FE method.

We need a grid on the domain Ω. The grid cells $T_i \in \overline{\Omega}$, $i = 1, ..., m$, are usually polyhedrons. As standard the functions $\beta_j|_{T_i}$ are polynomials for all $i = 1, ..., M$ and $j = 1, ..., m$. Additionally so called nodes $\{x_1, ..., x_M\} \in \overline{\Omega}$ are defined on the grid cells and it holds $\beta_j(x_i) = \delta_{j,i}$. All this can be found in detail e.g. in [4].

Now we put X^h instead of X in (1.3). Thus, we find $u^h \in X^h$, with

$$\left(\partial_t u^h, v^h\right) + a\left(u^h, v^h\right) = \left(f, v^h\right)\ ,\ \forall\, v^h \in X^h. \tag{1.4}$$

Since the functions β_i, for $i = 1, ..., M$, form a basis of V^h and all $v^h \in X^h$ have a representation of the form $v^h = \sum_{i=1}^{M} \hat{v}_i \beta_i$ with suitable scalars $\hat{v}_i \in \mathbb{R}$, it suffices to solve (1.4) for $v^h = \beta_i$ with $i = 1, ...M$.

There are no major difficulties with the discretization in time. In terms of t, it is only a first-order differential equation. Divide the time interval $[0,T]$ into N equidistant sections (t_{i-1}, t_i) for $i = 1, ..., N$ such that $t_0 = 0$ and $t_N = T$, the step size is $\Delta t = t_i - t_{i-1}$.

Now one can choose between different methods. In [16] the backward Euler method was selected, but here the decision was made in favor of the Crank–Nicolson method because it leads to better solutions in the numerical tests.

This is an implicit procedure in which the initial value problem is

$$y'(t) = f(t, y(t))\ ,\ \ y(t_0) = y_0\,.$$

For $t_i = t_0 + i\Delta t$ the iteration is as follows:

$$\begin{aligned} y_0 &= y_0 \\ y_i &= y_{i-1} + \frac{\Delta t}{2}\left(f(t_{i-1}, y_{i-1}) + f(t_i, y_i)\right)\ ,\ \ i = 1, ..., N\,. \end{aligned}$$

An initial value u_0 is given in (1.1). Now one applies this to (1.4) and looks for $\left\{u_k^h\right\}_{k=1}^N \subset X^h$:

$$\frac{1}{\Delta t}\left(u_k^h - u_{k-1}^h, v^h\right) + \frac{1}{2}a\left(u_{k-1}^h + u_k^h, v^h\right) = \frac{1}{2}\left(f_{k-1} + f_k, v^h\right)\ ,\ \forall\, v^h \in X^h\,. \quad (1.5)$$

In the following chapters, the VMS-POD method from [16] is derived. Reduced order modeling, in particular the POD and the VMS method, will be explained. This will be followed by a section where these two methods gets connected. Chapter six deals with the error estimation of [16], as it turns out, it has some gaps. Afterwards one can find a summary of the method with the intention to give a step by step explanation of it. Then follows a chapter about the implementation, which included numerical tests for an example problem. The VMS-POD and the POD-G methods are compared. The results are interpreted in conjunction with the results of [16]. They verify each other.

2 Proper Orthogonal Decomposition

The idea of *reduced order modeling* (ROM), as the name suggests, is to reduce the order (dimension) of a problem. In the case of a linear system, this is done by constructing matrices of lower dimensions that approximate the original system. This approach is also applicable to many nonlinear problems. A solution is not longer searched in the ansatz space X^h, but in a lower dimensional subspace $X^l \subset X^h$.

For example, if we consider the FE discretization, we quickly get a very large number of basis functions. Depending on the structure of the solutions, we often need much fewer functions to represent the essential properties of the solution sufficiently well. (An approach for this would be for example adaptive grids.)

However, it should be noted that the set of required basis functions depends on the choice of the basis itself. An ideal basis should represent the entire expected solution space with as few functions as possible. To construct such a basis, one needs information about the expected solutions in advance. Depending on the system to be solved, there are different possibilities to get such information (e.g. derivative properties).

For reducing the order with *proper orthogonal decomposition* (POD), a solution is first determined as accurately as possible. This solution is then used to obtain further solutions with lower complexity, where we hope that the characteristic properties are transferable.

In practice, it is very rare to find such a basis (with few functions) that perfectly covers the solution space. However, it is possible to reduce the number of basis functions significantly and still keep the most important characteristics. This is a very strong reduction of the basis functions, where a basis has rather a one to two digit number of basis elements instead of many thousands. This simplification is inevitably at the expense of the expected accuracy. In general the ROM basis consists of global functions, in contrast to the FE basis functions.

One aim is to solve similar problems with the reduced basis. This is already been used in some applications, for example in the *parametric global mode reduction*.

In a sense, the POD method provides an 'optimal' basis $\{\phi_1, ..., \phi_r\}$ that is both orthonormal and best describes a given function $u : [0, T] \longrightarrow X \subset L^2(\Omega)$ by the approximation $u \approx u^r := \sum_{i=1}^{r} (u, \phi_i)_X \phi_i$.

The procedure is equivalent to the singular value decomposition in the discrete case, it was also described in this way in [5]. Here the derivation from [13] or [7] was chosen, because this has the optimality claim as starting point. The connections are obvious, but can be read again very clearly in [23]. In other contexts POD is also known under the names *principal component analysis* or *Karhunen-Loève transformation*.

2.1 Mathematical Motivation

We choose $X \subset L^2(\Omega)$ as a Hilbert space, with the inner product $(\cdot,\cdot)_X$ and let $u : [0,T] \longrightarrow X$ be a solution of (1.3). Search inductively for a basis $\{\phi_1, ..., \phi_r\}$, starting with $r = 1$.
Assuming the existence of such a minimum, we seek the solution of:

$$\min_{\phi \in X} \left\langle \|u - (u,\phi)_X \phi\|_X^2 \right\rangle \quad \text{, such that} \quad \|\phi\|_X^2 = 1, \tag{2.1}$$

where $\langle\cdot\rangle$ describes the time average. (We will use the average of a discrete number of time points, but in theory it is also possible to use the continuous mean value of the function.) Since

$$\begin{aligned} 0 \leq \|u - (u,\phi)_X \phi\|_X^2 &= (u - (u,\phi)_X \phi, u - (u,\phi)_X \phi)_X \\ &= \|u\|_X^2 - 2|(u,\phi)_X|^2 + |(u,\phi)_X|^2 \underbrace{\|\phi\|_X^2}_{=1} = \|u\|_X^2 - |(u,\phi)_X|^2, \end{aligned}$$

(2.1) is equivalent to:

$$\max_{\phi \in X} \left\langle |(u,\phi)_X|^2 \right\rangle \quad \text{, such that} \quad \|\phi\|_X^2 = 1\,. \tag{2.2}$$

By incorporating the constraints via the Lagrange multiplier λ, we obtain the corresponding functional:

$$J[\phi,\lambda] = \left\langle |(u,\phi)_X|^2 \right\rangle - \lambda \left(\|\phi\|_X^2 - 1 \right).$$

Hereby we obtain a necessary criterion for a solution of (2.2):

$$\frac{d}{d\delta} J[\phi + \delta\psi, \lambda]\,|_{\delta=0} = 0,$$

for all ψ, so that $\phi + \delta\psi \in L^2(\Omega)$ with arbitrary $\delta \in \mathbb{R}$.

It follows that:

$$\begin{aligned} 0 =& \frac{d}{d\delta} J[\phi + \delta\psi, \lambda]\,|_{\delta=0} \\ =& \frac{d}{d\delta} \Big[\left\langle (u, \phi + \delta\psi)_X (u, \phi + \delta\psi)_X \right\rangle - \lambda (\phi + \delta\psi, \phi + \delta\psi)_X \Big] \Big|_{\delta=0} \\ =& \frac{d}{d\delta} \Big[\left\langle (u,\phi)_X^2 + 2\delta (u,\phi)_X (u,\psi)_X + \delta^2 (u,\psi)_X \right\rangle - \\ & \lambda[(\phi,\phi)_X + 2\delta (\phi,\psi)_X + \delta^2 (\psi,\psi)_X] \Big] \Big|_{\delta=0} \\ =& 2 \left[\left\langle (u,\phi)_X (u,\psi)_X \right\rangle - \lambda (\phi,\psi)_X \right]. \end{aligned}$$

Since only $u : [0,T] \longrightarrow L^2(\Omega)$ is a time-dependent function and both $(\cdot,\cdot)_X$ and $\langle\cdot\rangle$ are commutative, we have:

$$\begin{aligned} 0 &= \left\langle (u,\phi)_X (u,\psi)_X \right\rangle - \lambda (\phi,\psi)_X \\ &= \left\langle ((u,\phi)_X u, \psi)_X \right\rangle - \lambda (\phi,\psi)_X \\ &= (\langle (u,\phi)_X u \rangle - \lambda\phi, \psi)_X\,. \end{aligned}$$

The function ψ can be chosen arbitrarily, this results in:

$$\langle (u,\phi)_X \, u(x) \rangle = \lambda \phi(x) \ .$$

If we now write the left-hand side as an operator $\mathcal{R}$ so that

$$\mathcal{R}\phi(x) = \langle (u,\phi)_X \, u(x) \rangle \ ,$$

we get the eigenvalue problem:

$$\mathcal{R}\phi(x) = \lambda \phi(x) \ . \tag{2.3}$$

A detour into spectral theory follows, to be read for example in [30] or in the appendix of [13].

Theorem 2.1.1 (A Spectral Theorem for Operators on Hilbert Spaces)**.** *For a compact and selfadjoint (in $\mathbb{R}$) or normal (in $\mathbb{C}$) operator $T: H \longrightarrow H$, exists an orthonormal system $\{e_1, e_2, ...\}$ and a null sequence $(\lambda_k)_{k\in\mathbb{N}} \in \mathbb{R}\setminus\{0\}$, such that*

$$H = ker(T) \oplus \overline{span\{e_1, e_2, ...\}}$$

and $\forall x \in H$

$$Tx = \sum_k \lambda_k \langle x, e_k \rangle_H \, e_k \ ,$$

where λ_k are the eigenvalues $\neq 0$ and e_k the corresponding eigenvectors. Moreover we have:

$$\|T\| = \sup_{\|x\|=1} \langle Tx, x \rangle = \sup_k |\lambda_k|.$$

(For a compact and self-adjoint operator T, either $\|T\|$ or $-\|T\|$ are eigenvalues of T, this implies $\sup_k |\lambda_k|$ is a maximum.)

For proof, see [30, p.294 f].

A generalization of this theorem for non-selfadjoint operators is the theorem on *singular value decomposition*. If an operator is also positive, then it has no negative eigenvalues.

So we now consider the operator $\mathcal{R}$ to apply Theorem 2.1.1:

$$\begin{aligned}(\mathcal{R}\phi,\phi)_X &= (\langle (u,\phi)_X \, u \rangle, \phi)_X \\ &= \langle (u,\phi)_X \, (u,\phi)_X \rangle = \left\langle |(u,\phi)_X|^2 \right\rangle \geq 0 \quad \text{(positive)},\end{aligned} \tag{2.4}$$

$$\begin{aligned}(\mathcal{R}\phi,\psi)_X &= (\langle (u,\phi)_X \, u \rangle, \psi)_X = \langle (u,\phi)_X \, (u,\psi)_X \rangle \\ &= \langle (u,\psi)_X \, (\phi,u)_X \rangle = (\phi, \langle (u,\psi)_X \, u \rangle)_X = (\phi, \mathcal{R}\psi)_X \quad \text{(selfadjoint)}.\end{aligned} \tag{2.5}$$

The compactness of $\mathcal{R}: X \longrightarrow L^2(\Omega)$ depends in general on X, $(\cdot,\cdot)_X$ and $\langle\cdot\rangle$. However, we only consider finite-dimensional X (finite-dimensional after *FE discretization*). It is easy to see that $\mathcal{R}$ is linear and continuous. In [30, p.72] it can be read that compactness follows with these conditions.

In connection with (2.4), (2.5) and Theorem 2.1.1 it follows that the maximum in (2.2) exists and is equal to the largest eigenvalue λ_{max} of $\mathcal{R}$. Therefore, the corresponding eigenfunction e_{max} is a solution of (2.2). W. l. o. g. let $(\lambda_n)_{n\in\mathbb{N}}$ be such that $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq ...$, it follows $\lambda_{max} = \lambda_1$ and $e_{max} = e_1 =: \phi_1$. To detect the basis $\{\phi_1, ..., \phi_r\}$, one continues wit $r = 2$.

After finding a solution of (2.2), ϕ_2 is searched in $span\{\phi_1\}^{\perp}$.

$$\max_{\phi\in L^2(\Omega)} \left\langle |(u,\phi)_X|^2 \right\rangle \text{ , such that } \quad \|\phi\|_X^2 = 1 \text{ and } (\phi,\phi_1)_X = 0\,. \tag{2.6}$$

Condition (2.3) is still necessary. Moreover $(\mathcal{R}\phi,\phi)_X = \langle |(u,\phi)_X|^2 \rangle$ holds because of (2.4). So we can see that $e_2 =: \phi_2$ solves (2.6). Thus we continue in $span\{\phi_1,\phi_2\}^{\perp}$ and further in $span\{\phi_1,\phi_2,...,\phi_r\}^{\perp}$. Note that for all $r \leq \dim(X)$ the basis $\{\phi_1,\phi_2,...,\phi_r\}$ is an optimal basis with r elements, in the sense of the criteria set out at the beginning.

Remark: Since it holds for all e_j that $\left\langle |(u,e_j)_X|^2 \right\rangle = \lambda_j$, the eigenvalues are a measure of the contribution of $\phi_j = e_j$ to the characterization of u.

2.2 Calculation of the POD Basis (Modes) and the Method of Snapshots

The first step is to calculate a solution $u : [0,T] \longrightarrow X \subset H_0^1(\Omega)$ of (1.3) as accurately as possible, where we do not determine u on the whole time interval, but only for $N+1$ time points $\{t_0,...,t_N\} \subset [0,T]$.

For $i = 0,...,N$, the functions $w_i(x) := u(t_i,x)$ are called *snapshots*. (For the sake of clarifying the presentation , these functions are called w_i instead of u_i.) The time average is: $\langle w(x)\rangle := \frac{1}{N+1}\sum_{i=0}^{N} w_i(x)$. Now we are looking for $r \leq N+1$ functions $\{\phi_j\}_{j=1,...,r} \subset \text{span}\{w_0,...,w_N\}$ such that ϕ_1 solves problem (2.2), ϕ_2 solves problem (2.6), etc.

Let us have a look at:

$$\mathcal{R}\phi(x) = \langle (w,\phi)_X\, w(x)\rangle = \frac{1}{N+1}\sum_{i=0}^{N} (w_i,\phi)_X\, w_i(x)\,.$$

The solution ϕ is in span $\{w_0,...,w_N\}$. So we are looking for a function $\tilde{\phi} = \sum_{i=0}^{N} a^j w_j$, with $a^j \in \mathbb{R}$, $j = 0,...,N$. This leads to:

$$\mathcal{R}\tilde{\phi}(x) = \frac{1}{N+1}\sum_{i=0}^{N}\sum_{j=0}^{N} a^j\,(w_i,w_j)_X\, w_i(x)\,,$$

which implies:

$$\sum_{i=0}^{N}\left[\frac{1}{N+1}\sum_{j=0}^{N}(w_i,w_j)_X\, a^j\right] w_i(x) = \mathcal{R}\tilde{\phi}(x) = \lambda\tilde{\phi}(x) = \sum_{i=0}^{N}\lambda a^i w_i(x)\,,$$

respectively

$$\frac{1}{N+1}\sum_{j=0}^{N}\left(w_i, w_j\right)_X a^j = \lambda a^i\,,\ \ \forall i = 0, ..., N.$$

With $a = \left(a^0, ..., a^N\right)^T \in \mathbb{R}^{N+1}$ and $W \in \mathbb{R}^{(N+1)\times(N+1)}$ where $W_{ij} = \frac{1}{N+1}\left(w_i, w_j\right)_X$, it turns to the form:

$$Wa = \lambda a\,. \tag{2.7}$$

The matrix W is symmetric and positive semi definite, therefore it has only real eigenvalues ≥ 0, and all the eigenvectors corresponding to different positive eigenvalues are orthogonal.

Let $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_d > 0$, $d \leq N+1$, be the nonzero eigenvalues of (2.7), $a_k \in \mathbb{R}^{N+1}$ with $1 \leq k \leq d$ the corresponding normalized eigenvectors, where these are orthonormal in $\mathbb{R}^{N+1}$. However, we cannot conclude from $\left(a_k\right)^T a_l = \delta_{k,l}$ that the functions $\tilde{\phi_k} := \sum_{i=0}^{N} a_k^i w_i$ and $\tilde{\phi_l} := \sum_{j=0}^{N} a_l^j w_j$ are orthonormal in X for all $k, l = 1, ..., d$. So let us consider:

$$\begin{aligned}\left(\tilde{\phi_k}, \tilde{\phi_l}\right)_X &= \left(\sum_{i=0}^{N} a_k^i w_i, \sum_{j=0}^{N} a_l^j w_j\right)_X = \sum_{i=0}^{N}\sum_{j=0}^{N} a_k^i a_l^j \left(w_i, w_j\right)_X \\ &= \sum_{i=0}^{N}\sum_{j=0}^{N} a_k^i a_l^j \left(\left(N+1\right) W_{ij}\right) = \left(N+1\right)\left(a_k\right)^T W a_l \\ &= \left(N+1\right)\left(a_k\right)^T \lambda_l\, a_l = \left(N+1\right)\,\lambda_l\,\delta_{kl}.\end{aligned}$$

Therefore, we chose the following POD basis functions (POD modes):

$$\phi_k = \frac{1}{\sqrt{\left(N+1\right)\,\lambda_k}}\sum_{i=0}^{N} a_k^i w_i. \tag{2.8}$$

We do not need to take all ϕ_k, $k = 1, ..., d$, into the basis, but only $r \leq d$. Let $X^r = \text{span}\left\{\phi_1, ..., \phi_r\right\}$ be our new ansatz space.

If we select $W_{i,j} = \left(w_i, w_j\right)_X$ and $\phi_k = \frac{1}{\sqrt{\lambda_k}}\sum_{i=0}^{N} a_k^i w_i$ we get the same solution. This might be a better choice for the calculation, and is used for the implementation in Chapter 7.

Note that in [16] and [24] they use $W_{i,j} = \frac{1}{N+1}\left(w_i, w_j\right)_X$ *and* $\phi_k = \frac{1}{\sqrt{\lambda_k}}\sum_{i=0}^{N} a_k^i w_i$. *This is correct, because the chosen eigenvectors are not specified. For the implementation it is useful to normalize them.*

We link this with (1.3) ore (1.4) and get the POD-Galerkin method. Find $u^r : [0, T] \longrightarrow X^r$:

$$\left(\partial_t u^r, v^r\right) + a\left(u^r, v^r\right) = \left(f, v^r\right)\ ,\ \forall\, v^r \in X^r. \tag{2.9}$$

With the choice of $X^r \subset H_0^1(\Omega)$ one considers the boundary condition $u(t,x) = 0$ on $(0,T] \times \partial\Omega$. After the time discretization (1.5) we search for $\{u_l^r\}_{l=1}^N \subset X^r$, such that for all $v^r \in X^r$, $l = 1, ..., N$ holds:

$$\frac{1}{\Delta t}\left(u_l^r - u_{l-1}^r, v^r\right) + \frac{1}{2}a\left(u_{l-1}^r + u_l^r, v^r\right) = \frac{1}{2}\left(f_{l-1} + f_l, v^r\right).$$

As initial function for $t = 0$ one could use $u_0^r := \sum_{i=1}^{r} \left(u_0, \phi_i\right)_X \phi_i$.

In practice there are several opportunities to choose the snapshots. A common way, which allows to cope with nonhomogeneous boundary conditions, is to calculate the mean value $\langle w(x)\rangle$ and subtract it from each snapshot. Then calculate the modes with $\{\hat{w}_0, ..., \hat{w}_N\}$, where $\hat{w}_i = w_i - \langle w(x)\rangle$ and solve the adapted problem to find a solution $\hat{u}^r := u^r - \langle w(x)\rangle$ such that:

$$\left(\partial_t \hat{u}^r, \hat{v}^r\right) + a\left(\hat{u}^r, \hat{v}^r\right) = \left(f, \hat{v}^r\right) - \underbrace{\left(\partial_t \langle w(x)\rangle, \hat{v}^r\right)}_{=0} - a\left(\langle w(x)\rangle, \hat{v}^r\right), \tag{2.10}$$

for all $\hat{v}^r \in \hat{X}^r$. This also makes sense from a geometric perspective as described in [5, Section 3.5].

Another much discussed method is to include the difference quotients (DQs) $\overline{\partial} w_j = \frac{w_j - w_{j-1}}{\Delta t}$ $, \forall\ j = 1, ..., N$ into the set of snapshots. This is also done in [16], where they follow the argumentation of [24], which expects better error estimates. In particular very good pointwise error bounds in time are proven in this case as one can read in [22] or [8, 26]. On the other hand, it is not certain that this method will lead to better results in practice. In [19] the results were worse when the DQs are included. In [17] they try to prove better results with the DQs, but in the numerical tests they use much more POD basis functions in the DQ case as in the non-DQ case.

A very recent article [11] may provide an explanation for this. A similar error bound as in [22] is proven for the non-DQ case, where the solution has to be sufficiently smooth, as the examples in [19] and [17] are.

Additionally, one has to take care of the physical dimensions, since a function and its derivative describe different physical quantities. This can be taken into account by adding a time scalar as a factor in front of the DQ. This is done in [8] and described for example in [10].

Including the DQs might be a good choice for the general case, but this work does not include them.

2.3 Error Representation

The following error representation is valid:

$$\begin{aligned}
\left\langle \left\| w - \sum_{i=1}^{r} (w,\phi_i)_X \, \phi_i \right\|_X^2 \right\rangle &= \left\langle \left\| \sum_{i=1}^{d} (w,\phi_i)_X \, \phi_i - \sum_{i=1}^{r} (w,\phi_i)_X \, \phi_i \right\|_X^2 \right\rangle \\
&= \left\langle \left\| \sum_{i=r+1}^{d} (w,\phi_i)_X \, \phi_i \right\|_X^2 \right\rangle \\
&= \left\langle \left(\sum_{i=r+1}^{d} (w,\phi_i)_X \, \phi_i, \sum_{l=r+1}^{d} (w,\phi_l)_X \, \phi_l \right)_X \right\rangle \\
&= \left\langle \sum_{i=r+1}^{d} \sum_{l=r+1}^{d} (w,\phi_i)_X \, (w,\phi_l)_X \underbrace{(\phi_i,\phi_l)_X}_{\delta_{il}} \right\rangle \\
&= \sum_{i=r+1}^{d} \left\langle |(w,\phi_i)_X|^2 \right\rangle \\
&\overset{(2.4)}{=} \sum_{i=r+1}^{d} (\mathcal{R}\phi_i,\phi_i)_X = \sum_{i=r+1}^{d} \lambda_i . \qquad (2.11)
\end{aligned}$$

One can choose r in dependence of this representation.

In the case of including the difference quotients $w_{N+j} := \overline{\partial} w_j = \frac{w_j - w_{j-1}}{\Delta t}$ for all $j = 1, ..., N$ in the set of snapshots, it leads to:

$$\begin{aligned}
\left\langle \left\| w - \sum_{i=1}^{r} (w,\phi_i)_X \, \phi_i \right\|_X^2 \right\rangle &= \frac{1}{2N+1} \sum_{j=0}^{2N} \left\| w_j - \sum_{i=1}^{r} (w_j,\phi_i)_X \, \phi_i \right\|_X^2 \\
&= \frac{1}{2N+1} \sum_{j=0}^{N} \left\| w_j - \sum_{i=1}^{r} (w_j,\phi_i)_X \, \phi_i \right\|_X^2 \\
&\quad + \frac{1}{2N+1} \sum_{j=1}^{N} \left\| \overline{\partial} w_j - \sum_{i=1}^{r} \left(\overline{\partial} w_j,\phi_i\right)_X \phi_i \right\|_X^2 \\
&= \sum_{j=r+1}^{d} \lambda_j \, . \qquad (2.12)
\end{aligned}$$

Thereby, with $2N+1 \leq 3N$, it follows:

$$\begin{aligned}
\frac{2N+1}{N\,(2N+1)} \sum_{j=1}^{N} \left\| \overline{\partial} w_j - \sum_{i=1}^{r} \left(\overline{\partial} w_j,\phi_i\right)_X \phi_i \right\|_X^2 &\leq \frac{2N+1}{N} \sum_{j=r+1}^{d} \lambda_j \, , \\
\frac{1}{N} \sum_{j=1}^{N} \left\| \overline{\partial} w_j - \sum_{i=1}^{r} \left(\overline{\partial} w_j,\phi_i\right)_X \phi_i \right\|_X^2 &\leq 3 \sum_{j=r+1}^{d} \lambda_j \, . \qquad (2.13)
\end{aligned}$$

Without the additional snapshots or other conditions there would be a factor $\frac{C}{(\Delta t)^2}$ on the RHS of (2.13), which is a significantly worse estimate.

3 Variational Multiscale (VMS) Method

So-called multiscale problems are problems in which scales of different sizes are present. Grids which are manageable can often not represent the smallest scales. But to get a physically consistent solution the small scales are important. In order to cope with these, suitable multiscale methods are used. A detailed explanation can be found in [14, 15] and also in [2].

In the references [6, 21] and [18, 25], VMS methods for the *convection-diffusion-reaction equation* are differently motivated and derived. These are so-called two-scale methods.

3.1 Introduction to VMS

The derivation in [6, 21] is very 'classical' for VMS and is done by decomposing the solution space V such that $V = V^h \oplus \tilde{V}$, where V^h mostly represents the resolved scales and $\tilde{V}$ the unresolved ones. The solution is written in the form $u = u^h + \tilde{u}$ with $u^h \colon [0,T] \longrightarrow X^h$ and $\tilde{u} \colon [0,T] \longrightarrow \tilde{X}$. Now this is inserted into the variational formulation (1.3):

$$\left(\partial_t u^h, v^h\right) + \left(\partial_t \tilde{u}, v^h\right) + a\left(u^h, v^h\right) + a\left(\tilde{u}, v^h\right) = \left(f, v^h\right) \ , \ \forall\, v^h \in X^h, \tag{3.1}$$

$$\left(\partial_t u^h, \tilde{v}\right) + \left(\partial_t \tilde{u}, \tilde{v}\right) + a\left(u^h, \tilde{v}\right) + a\left(\tilde{u}, \tilde{v}\right) = \left(f, \tilde{v}\right) \ , \ \forall\, \tilde{v} \in \tilde{X}. \tag{3.2}$$

The hope is that it is sufficient to solve (3.1). Equation (3.2) can be used here to characterize $\left(\partial_t \tilde{u}, v^h\right) + a\left(\tilde{u}, v^h\right)$, as well as properties of $\tilde{X}$ such as orthogonality.

So it comes to a method where we search $\left(u^h, \tilde{u}\right) \in X^h \times \tilde{X}$ with:

$$\begin{aligned} \left(\partial_t u^h, v^h\right) + a\left(u^h, v^h\right) + A\left(\tilde{u}, v^h\right) &= \left(f, v^h\right) \ , \ \forall\, v^h \in X^h, \\ K\left(\tilde{u}\right) &= Q. \end{aligned} \tag{3.3}$$

Here $A\left(\tilde{u}, v^h\right)$ is a characterization of $\left(\partial_t \tilde{u}, v^h\right) + a\left(\tilde{u}, v^h\right)$ and $K\left(\tilde{u}\right) = Q$ describes conditions on $\tilde{u}$, usually in the form of a suitable projection operator.

3.2 VMS for our Application

In [18, 25] the second scale is introduced rather indirectly. Since this is the same method as used in [16], a detailed explanation follows.

We begin with the term that causes the instability, in this case $\varepsilon \Delta u$ or in the variational formulation $\varepsilon\left(\nabla u, \nabla v\right)$. After adding and subtracting $\varepsilon_+\left(\nabla u^h, \nabla v^h\right)$ in the equation (1.4), for all $v^h \in X^h$, we have:

$$\left(\partial_t u^h, v^h\right) + a\left(u^h, v^h\right) + \varepsilon_+\left(\nabla u^h, \nabla v^h\right) - \varepsilon_+\left(\nabla u^h, \nabla v^h\right) = \left(f, v^h\right).$$

Now let $\nabla u^H = g^H \colon (0,T] \longrightarrow L^H \subseteq L^2(\Omega)$ (L^H specified later), where u^H stands for the large scales of the FE solution u^h. Now replace $-\varepsilon_+ \left(\nabla u^h, \nabla v^h\right)$ to subtracts only the resolved part $\varepsilon_+ \left(g^H, \nabla v^h\right)$. Therefore the artificial viscosity $\varepsilon_+ \left(\nabla u^h, \nabla v^h\right) - \varepsilon_+ \left(g^H, \nabla v^h\right)$ only applies to the fine scales. It comes to the formulation:

$$\left(\partial_t u^h, v^h\right) + a\left(u^h, v^h\right) + \varepsilon_+ \left(\nabla u^h - g^H, \nabla v^h\right) = \left(f, v^h\right) \ , \ \forall\, v^h \in X^h, \tag{3.4}$$

$$\left(\nabla u^h - g^H, w^H\right) = 0 \, , \ \forall\, w^H \in L^H. \tag{3.5}$$

Then we have a method of the form (3.3).

We now choose ε_+ as a nonnegative constant, which will be considered in more detail later. Let $g^H = P_H \nabla u^h$ and P_H be the L^2-orthogonal projection into the space L^H. Putting this into (3.4), we get:

$$\left(\partial_t u^h, v^h\right) + a\left(u^h, v^h\right) + \varepsilon_+ \left(\left(\mathbb{I} - P_H\right) \nabla u^h, \nabla v^h\right) = \left(f, v^h\right) \ , \ \forall\, v^h \in X^h.$$

Due to (3.5) and $P_H \nabla v^h \in L^H$ it holds $\left(\left(\mathbb{I} - P_H\right) \nabla u^h, P_H \nabla v^h\right) = 0$. Thus, it follows:

$$\begin{aligned} \left(\partial_t u^h, v^h\right) + a\left(u^h, v^h\right) + \varepsilon_+ \left(\left(\mathbb{I} - P_H\right) \nabla u^h, \left(\mathbb{I} - P_H\right) \nabla v^h\right) \\ = \left(f, v^h\right) \ , \ \forall\, v^h \in X^h. \end{aligned}$$

With $P'_H := \mathbb{I} - P_H$, we get:

$$\left(\partial_t u^h, v^h\right) + a\left(u^h, v^h\right) + \varepsilon_+ \left(P'_H \nabla u^h, P'_H \nabla v^h\right) = \left(f, v^h\right) \ , \ \forall\, v^h \in X^h. \tag{3.6}$$

The choice of the space L^H is crucial. To choose a useful L^H we take a look at (3.4), more precisely:

$$\varepsilon_+ \left(\nabla u^h, \nabla v^h\right) - \varepsilon_+ \left(g^H, \nabla v^h\right). \tag{3.7}$$

We have $\nabla u^h \in \nabla X^h := \mathit{span}\left\{\nabla \beta^1, ..., \nabla \beta^M\right\}$. If one chooses g^H from ∇X^h, then (3.7) is equal to 0.

In [18] they use a coarser grid, with width $H > h$ to construct X^H. More precisely, the triangulation they use for X^h is a refinement of the triangulation of X^H. It is also possible to choose independent grids, as in [25]. Then take g^H from $L^H = \nabla X^H$. In this case (3.7) acts only on the unresolved scales, because we subtract the resolved part. This is exactly what we need for stabilization.

4 VMS-POD Model

Now we want to use the described method to stabilize (2.9), (2.10). Thus be $\{w_i\}_{i=0}^{N}$ the set of all snapshots and $X^r = \operatorname{span}\{\phi_1, ..., \phi_r\}$ the corresponding POD ansatz space. $E(v^r)$ describes the RHS of (2.9, 2.10) depending on the chosen snapshots, where $E : X^r \longrightarrow \mathbb{R}$ is a linear functional. Applying (3.6) one searches for $u^r : [0,T] \longrightarrow X^r$ such that:

$$(\partial_t u^r, v^r) + a(u^r, v^r) + \varepsilon_+ (P'_R \nabla u^r, P'_R \nabla v^r) = E(v^r)\ ,\ \forall\, v^r \in X^r. \tag{4.1}$$

Here P'_R is defined equivalently to P'_H with $P'_R = \mathbb{I} - P_R$ and P_R is the L^2- orthogonal projection onto a space $L^R \subseteq L^2(\Omega)$, which is similar to L^H. (For simplification the h and H is used for the FE ansatz space and all corresponding subspace and functions, where r and R replace them after the *proper orthogonal decomposition.*) As in (3.7) one can see that a reasonable choice for L^R is a subspace of ∇X^r. Let $R < r$ and $X^R := \operatorname{span}\{\phi_1, ..., \phi_R\}$, we choose $L^R = \nabla X^R \subset \nabla X^r$.

With the Crank–Nicolson time discretization (1.5) one now obtains the following procedure. For $k = 1, ..., N$ and with $u_0^r = \sum_{i=1}^{r} \left(w_0^h, \phi_i\right) \phi_i \in X^r$, find $u_k^r \in X^r$ satisfying:

$$\begin{aligned}\frac{1}{\Delta t}\left(u_k^r - u_{k-1}^r, v^r\right) + \frac{1}{2} a\left(u_{k-1}^r + u_k^r, v^r\right) + \varepsilon_+ \frac{1}{2}\left(P'_R \nabla\left(u_{k-1}^r + u_k^r\right), P'_R \nabla v^r\right) \\ = \frac{1}{2} E_{k-1}(v^r) + \frac{1}{2} E_k(v^r)\ ,\ \forall\, v^r \in X^r\,.\end{aligned} \tag{4.2}$$

We take a closer look at the artificial viscosity term $\varepsilon_+ (P'_R \nabla u_k^r, P'_R \nabla v^r)$. How this term looks in detail depends on the inner product $(\cdot,\cdot)_X$ one chose. As we are acting in $H_0^1(\Omega)$ it can be chosen from the following three (there might be more possibilities but the following are the most common ones):

$$\begin{aligned}(u,v)_X &= (u,v)_{L^2(\Omega)}\,, \\ (u,v)_X &= (u,v)_{H^1(\Omega)} = (u,v)_{L^2(\Omega)} + (\nabla u, \nabla v)_{L^2(\Omega)}\,, \\ (u,v)_X &= (u,v)_{H_0^1(\Omega)} = (\nabla u, \nabla v)_{L^2(\Omega)}\,.\end{aligned} \tag{4.3}$$

In most literature, as well as in [16], one finds $(u,v)_X = (u,v)_{H^1(\Omega)}$. It may be a good choice, because it best covers all characteristics, as it includes both the snapshot functions and their gradients. This can be an advantage in error estimation. A benefit of $(u,v)_X = (u,v)_{H_0^1(\Omega)}$ is the calculation of $P_R \nabla v^r = \sum_{i=1}^{R} (\nabla v^r, \nabla \phi_i)_{L^2(\Omega)} \nabla \phi_i$, since $\{\nabla\phi_1, \ldots, \nabla\phi_R\}$ then forms an L^2-orthonormal basis of ∇X^R. In the other cases, an orthonormal basis must first be determined, for example with *Gram–Schmidt* process. I could not find any advantages of the $L^2(\Omega)$ inner product for this application.

Still to be determined are the coefficients r, R and ε_+. These should be chosen in dependence of the desired accuracy.

5 Critical Review of an Error Analysis

This section contains a critical review of the error estimation of [16]. Unfortunately, there are some gaps and lacks of clarity that are pointed out. Chapter 3 of [16] is reproduced below, and corrected where necessary, up to the point where the claims are no longer conclusive, which is *Lemma 3.4* in [16].

5.1 Preliminaries

Our aim is to prove estimates for the average error

$$\frac{1}{N+1}\sum_{n=0}^{N}\|u_n - u_n^r\|.$$

The function $u_n = u(t_n, \cdot) \in X = H_0^1(\Omega)$, where u is the solution of (1.3):

$$(u_t, v) + \varepsilon(\nabla u, \nabla v) + (\mathbf{b}\cdot\nabla u, v) + (cu, v) = (f, v)\ ,\ \ \forall v \in X. \tag{5.1}$$

And $u_n^r \in X^r \subset X^d = span\left\{u_0^h, ..., u_N^h, u_{N+1}^h, ..., u_{2N}^h\right\} \subset X$ is the solution of the VMS-POD model at time t_n:

$$\begin{aligned}\frac{1}{\Delta t}\left(u_n^r - u_{n-1}^r, v^r\right) + \varepsilon(\nabla u_n^r, \nabla v^r) + (\mathbf{b}\cdot\nabla u_n^r, v^r) + \\ (cu_n^r, v^r) + \varepsilon_+(P_R'\nabla u_n^r, P_R'\nabla v^r) = (f_n, v^r)\ ,\ \ \forall v^r \in X^r.\end{aligned} \tag{5.2}$$

For $n = 0, 1, ...N$ the snapshots u_n^h are the full order FE solutions[1] at time t_n and for $j = 1, ..., N$ the difference quotients $u_{N+j}^h = \overline{\partial} u_j^h = \frac{u_j^h - u_{j-1}^h}{\Delta t}$.

In [16] the backward Euler method is used for the time discretization and the difference quotients are included in the set of snapshots. Denote C as a generic constant, independent of $d, N, r, R, h, \varepsilon, \varepsilon_+$, where d is the dimension of X^d and h is the mesh size in the FE discretization. The following representations are bilinear forms:

$$\begin{aligned}b(u, v) &:= (\mathbf{b}\cdot\nabla u, v) + (cu, v),\\ a(u, v) &:= \varepsilon(\nabla u, \nabla v) + b(u, v),\\ A(u, v) &:= a(u, v) + \varepsilon_+(P_R'\nabla u, P_R'\nabla v).\end{aligned}$$

Therefore (5.2) becomes:

$$\frac{1}{\Delta t}\left(u_n^r - u_{n-1}^r, v^r\right) + A(u_n^r, v^r) = (f_n, v^r)\ ,\ \forall\, v^r \in X^r. \tag{5.3}$$

[1] In [16] no details about the full order model are stated. It is not clear which time discretization is used and how it is stabilized.

5.2 Definitions and Conditions

We consider the weighted norm:

$$\|u\|_{d,e,\alpha} := \sqrt{d\|u\|^2 + e\|\nabla u\|^2 + \alpha\|P_R'\nabla u\|^2}. \tag{5.4}$$

Assumption 1. *(Coercivity and Continuity) For $\beta, \gamma > 0$, the following holds:*

$$c - \frac{1}{2}\nabla \cdot \boldsymbol{b} \geq \beta, \tag{5.5}$$

$$\max\{\|c\|_c, \|\boldsymbol{b}\|_b\} = \gamma. \tag{5.6}$$

(Here $\|\cdot\|_c = \|\cdot\|_{(L^\infty(0,T;L^\infty(\Omega)))}$, $\|\cdot\|_b = \|\cdot\|_{(L^\infty(0,T;L^\infty(\Omega)))^n}$.)[2]

Now $A(\cdot,\cdot)$ is coercive and continuous in $H_0^1(\Omega)$ with respect to the norm $\|\cdot\|_{\beta,\varepsilon,\varepsilon_+}$. From (5.5) follows $\|u\|^2_{\beta,\varepsilon,\varepsilon_+} \leq A(u,u)$, which means coercivity. The continuity follows with (5.6) and the Cauchy-Schwarz inequality:

$$\begin{aligned}
A(u,v) &= \varepsilon(\nabla u, \nabla v) + (\mathbf{b}\cdot\nabla u, v) + (cu, v) + \varepsilon_+(P_R'\nabla u, P_R'\nabla v) \\
&\leq \varepsilon\|\nabla u\|\|\nabla v\| + \|\mathbf{b}\|_b\|\nabla u\|\|v\| + \|c\|_c\|u\|\|v\| + \varepsilon_+\|P_R'\nabla u\|\|P_R'\nabla v\| \\
&\leq \varepsilon\|\nabla u\|\|\nabla v\| + \gamma\|\nabla u\|\|v\| + \gamma\|u\|\|v\| + \varepsilon_+\|P_R'\nabla u\|\|P_R'\nabla v\| \\
&\leq C_1\left(\sqrt{\varepsilon}\|\nabla u\| + \sqrt{\beta}\|u\| + \sqrt{\varepsilon_+}\|P_R'\nabla u\|\right) \\
&\quad \left(\sqrt{\varepsilon}\|\nabla v\| + \sqrt{\beta}\|v\| + \sqrt{\varepsilon_+}\|P_R'\nabla v\|\right) \\
&\leq 3C_1\|u\|_{\beta,\varepsilon,\varepsilon_+}\|v\|_{\beta,\varepsilon,\varepsilon_+}.
\end{aligned} \tag{5.7}$$

We need $\gamma\|\nabla u\|\|v\| \leq C_1\sqrt{\gamma}\sqrt{\varepsilon}\|\nabla u\|\|v\|$ and $\gamma\|u\|\|v\| \leq C_1\beta\|u\|\|v\|$, therefore $C_1 := \max\left\{1, \frac{\sqrt{\gamma}}{\sqrt{\varepsilon}}, \frac{\gamma}{\beta}\right\}$.

Assumption 2. *(Approximability) Let $1 \leq m \leq k$ and k be the order of accuracy of the FE ansatz space $X^h \subset H_0^1(\Omega)$. For all $v \in H^{m+1} \cap H_0^1(\Omega)$ the following applies:*

$$\inf_{v^h \in X^h}\left\{\|v - v^h\| + h\|\nabla v - \nabla v^h\|\right\} \leq Ch^{m+1}\|v\|_{m+1}. \tag{5.8}$$

Assumption 3. *(FE Inverse Estimate) There exists a constant C so that it holds:*

$$\|\nabla v^h\| \leq Ch^{-1}\|v^h\|, \ \ \forall v^h \in X^h. \tag{5.9}$$

Lemma 5.2.1. *(POD Inverse Estimate) Let $M_r, H_r, S_r \in \mathbb{R}^{r\times r}$ where $M_{i,j} := (\phi_i, \phi_j)$ is the POD mass matrix, $H_{i,j} := (\nabla\phi_i, \nabla\phi_j)$ is the POD stiffness matrix and $S_{i,j} := (\phi_i, \phi_j)_{H^1(\Omega)}$ is the H^1 POD mass matrix. With the matrix 2-norm $\|\cdot\|_2$ holds for all $v^r \in X^r$:*

$$\|v^r\|_{L^2(\Omega)} \leq \sqrt{\|M_r\|_2\|S_r^{-1}\|_2}\|v^r\|_{H^1(\Omega)}, \tag{5.10}$$

$$\|v^r\|_{H^1(\Omega)} \leq \sqrt{\|S_r\|_2\|M_r^{-1}\|_2}\|v^r\|_{L^2(\Omega)}, \tag{5.11}$$

$$\|\nabla v^r\|_{L^2(\Omega)} \leq \sqrt{\|H_r\|_2\|M_r^{-1}\|_2}\|v^r\|_{L^2(\Omega)}. \tag{5.12}$$

[2] In [16] there is no characterization of $\|\cdot\|_c$ and $\|\cdot\|_b$.

The proof of (5.10) and (5.11) is given in [24], (5.12) follows in the same way.

Remark: Since $X = H_0^1(\Omega)$, either H_r or S_r is the identity matrix[3]. In both cases follows

$$\|\nabla v^r\|_{L^2(\Omega)} \leq \sqrt{\|M_r^{-1}\|_2}\|v^r\|_{L^2(\Omega)}. \tag{5.13}$$

5.3 Error Analysis

Theorem 5.3.1. *For $n = 1, ..., N$ the solution u_n^r of (5.3) satisfies the following bound[4]:*

$$\|u_n^r\| \leq \|u_0^r\| + \Delta t \sum_{i=1}^{n} \|f_i\|. \tag{5.14}$$

Proof: For $n = 1, ..., N$ choose $v^r = u_n^r$ in (5.3) and get:

$$\frac{1}{\Delta t}\left(u_n^r - u_{n-1}^r, u_n^r\right) + A\left(u_n^r, u_n^r\right) = \left(f_n, u_n^r\right).$$

From coercivity follows $A\left(u_n^r, u_n^r\right) \geq 0$, this leads to:

$$\left(u_n^r - u_{n-1}^r, u_n^r\right) \leq \Delta t \left(f_n, u_n^r\right).$$

By the Cauchy-Schwarz inequality it follows:

$$\|u_n^r\| \leq \|u_{n-1}^r\| + \Delta t \|f_n\|.$$

If one starts with $n = 1$, (5.14) follows by induction. □

One now considers the *Ritz projection* $\omega^r \in X^r$ of $u \in X = H_0^1(\Omega)$:

$$A\left(u - \omega^r, v^r\right) = 0\,, \ \forall v^r \in X^r.$$

The Lax-Milgram lemma (to find, e.g. in [3]) implies the existence and uniqueness of the ω^r.

The following proposition is *Lemma 3.4* from [16]: *"The Ritz projection ω_n^r of the solution u_n satisfies the following error estimate:*

$$\begin{aligned}\frac{1}{N}\sum_{n=1}^{N}\|u_n - \omega_n^r\| \leq & C\left[\left(1 + \sqrt{\|M_r^{-1}\|_2} + \varepsilon_+^{-1}\right)^{\frac{1}{2}}\left(h^{m+1}\frac{1}{N}\sum_{n=1}^{N}\|u_n\|_{m+1} + \sqrt{\sum_{j=r+1}^{d}\lambda_j}\right)\right. \\ & \left. + \sqrt{\varepsilon + \varepsilon_+}\left(h^m\frac{1}{N}\sum_{n=1}^{N}\|u_n\|_{m+1} + \sqrt{\sum_{j=r+1}^{d}\lambda_j}\right)\right]."\end{aligned} \tag{5.15}$$

[3]This depends on the chosen inner product (the candidates are stated in (4.3)). In *Remark* 3.2 of [16] one can see that they use the $H^1(\Omega)$ inner product such that S_r is the identity matrix. Obviously the same results follows also in the case of the $H_0^1(\Omega)$ inner product.

[4]In [16] is proved $\|u_n^r\| \leq \|u_0^r\| + \Delta t \sum_{n=0}^{N-1}\|f_{n+1}\|$. But the stated version is slightly stronger.

Next, we critically review the reasoning presented in [16] for deriving this statement.

First, apply the *Ritz projection* on u_n and get:

$$A\left(u_n - \omega_n^r, v^r\right) = 0\,,\ \forall v^r \in X^r.$$

Now we decompose the error $u_n - \omega_n^r$ into:

$$u_n - \omega_n^r = \left(u_n - I_{h,r}\left(u_n\right)\right) - \left(\omega_n^r - I_{h,r}\left(u_n\right)\right) =: \eta_n - \psi_n^r,$$

where $I_{h,r}\left(u_n\right)$ is an interpolant of u_n in X^r, to be defined below. By the triangle inequality, we have:

$$\frac{1}{N}\sum_{n=1}^{N}\|u_n - \omega_n^r\| \leq \frac{1}{N}\sum_{n=1}^{N}\|\eta_n\| + \frac{1}{N}\sum_{n=1}^{N}\|\psi_n^r\|. \tag{5.16}$$

Let us start by estimating $\|\eta_n\|$. Consider u_n^h, the solution of (1.4) at time t_n, which yields the ensemble of snapshots:

$$\begin{aligned}\frac{1}{N}\sum_{n=1}^{N}\|\eta_n\| &= \frac{1}{N}\sum_{n=1}^{N}\|u_n - I_{h,r}\left(u_n\right)\| \\ &\leq \frac{1}{N}\sum_{n=1}^{N}\|u_n - u_n^h\| + \frac{1}{N}\sum_{n=1}^{N}\|u_n^h - I_{h,r}\left(u_n\right)\|.\end{aligned} \tag{5.17}$$

Subsequently the ψ_n^r is estimated as follows[5]:

$$\frac{1}{N}\sum_{n=1}^{N}\|\psi_n^r\|_{1,\varepsilon,\varepsilon_+}^2 \leq C\left(1 + \|M_r^{-1}\|_2 + \varepsilon_+^{-1}\right)\frac{1}{N}\sum_{n=1}^{N}\|\eta_n\|^2 + \left(\varepsilon + \varepsilon_+\right)\frac{1}{N}\sum_{n=1}^{N}\|\nabla\eta_n\|^2. \tag{5.18}$$

In order to find upper bounds for (5.17), the following claims are made in [16]. In the opinion of the author, these are not sufficiently substantiated.

"With Assumption 2 and [27] one can easily show that:

$$\frac{1}{N}\sum_{n=1}^{N}\|u_n - u_n^h\| \leq Ch^{m+1}\frac{1}{N}\sum_{n=1}^{N}\|u_n\|_{m+1}. \tag{5.19}$$

For $I_{h,r}\left(u_n\right) := \sum_{j=1}^{r}\left(u_n^h, \phi_j\right)_X \phi_j$, we can use the error representation of (2.12) to get:[6]

$$\frac{1}{N}\sum_{n=1}^{N}\|u_n^h - I_{h,r}\left(u_n\right)\| \leq \sqrt{\sum_{j=r+1}^{d}\lambda_j}. \tag{5.20}$$

[5]This is derived from [16, (3.38)], which appears to be correct.

[6]In order to apply (2.12) to get these results, we need $X = L^2(\Omega)$.

If (5.19) were true, we would have:

$$\frac{1}{N}\sum_{n=1}^{N}\|\eta_n\| \leq Ch^{m+1}\frac{1}{N}\sum_{n=1}^{N}\|u_n\|_{m+1} + \sqrt{\sum_{j=r+1}^{d}\lambda_j}. \tag{5.21}$$

Similarly, using the inverse estimate, Assumption 3, in (5.19) and $X = H_0^1(\Omega)$ *in (2.12), we get:*[7]

$$\frac{1}{N}\sum_{n=1}^{N}\|\nabla\eta_n\| \leq Ch^{m}\frac{1}{N}\sum_{n=1}^{N}\|u_n\|_{m+1} + \sqrt{\sum_{j=r+1}^{d}\lambda_j}.\,''\tag{5.22}$$

The main problem here is that (5.19) does not hold in general. We look at the FE error of a time-dependent unstable problem. It is very unlikely to find an error estimate that is independent of the coefficients of the problem, any stabilization measures used and the temporal discretization error. Since the full order method is not characterized, the only thing we can do is to define an error bound E_{snap} as a placeholder, which estimates the error of the snapshots, such that:

$$\frac{1}{N}\sum_{n=1}^{N}\|u_n - u_n^h\| \leq E_{snap}. \tag{5.23}$$

One could use the error estimation of [9, Theorem 3.3] which is from [20]:

$$E_{snap} = M_u\left(h^{m+\frac{1}{2}} + (\Delta t)^p\right).$$

Here p is the convergence order of the timestep method and M_u is a positive constant that depends on $\|u\|_{L^\infty(H^{m+1})}$, $\|\partial_t u(0)\|_{L^\infty(H^{m+1})}$, $\|\partial_{tt}u\|_{L^2(H^{m+1})}$, $\|u_0\|_{m+1}$ and the coefficients of the problem but not on ε^{-1}. This result holds only for $\varepsilon \leq h\|\mathbf{b}\|_b$, in the other case one should get the factor h^m. Applying the error analysis from [9, 20] and taking into account that there the term $\varepsilon^{1/2}\|\nabla(u_n - u_n^h)\|$ appears on the left-hand side, which comes from $\varepsilon(\nabla u, \nabla v)$ in (5.1), then one gets:

$$\frac{1}{N}\sum_{n=1}^{N}\|\nabla\left(u_n - u_n^h\right)\| \leq \varepsilon^{-\frac{1}{2}}M_u\left(h^{m+\frac{1}{2}} + (\Delta t)^p\right). \tag{5.24}$$

The second inaccuracy concerns (5.20). As shown in (2.13), it should read:

$$\frac{1}{N}\sum_{n=1}^{N}\|u_n^h - I_{h,r}(u_n)\| \leq \sqrt{3\sum_{j=r+1}^{d}\lambda_j}, \tag{5.25}$$

as well as:

$$\frac{1}{N}\sum_{n=1}^{N}\|\nabla\left(u_n^h - I_{h,r}\right)(u_n)\| \leq \sqrt{3\sum_{j=r+1}^{d}\lambda_j}. \tag{5.26}$$

[7] This conflicts with the condition for (5.20) in footnote 6.

This does not play a role for the estimation. One can compensate $\sqrt{3}$ with the factor C. Now we could either calculate a new estimate with (5.23), (5.24) and (5.18), ore use the continuity and coercivity of A for an alternative estimate. With the assumption $\varepsilon \leq h\|\mathbf{b}\|_b$ we can write the following lemma:

Lemma 5.3.2. *The Ritz projection ω_n^r of the solution u_n satisfies the following error estimates:*

$$\frac{1}{N}\sum_{n=1}^{N}\|u_n-\omega_n^r\| \leq \quad C\Bigg(\left(1+\|M_r^{-1}\|_2+\varepsilon_+^{-1}+\varepsilon_+\varepsilon^{-1}\right)^{\frac{1}{2}} M_u\left(h^{m+\frac{1}{2}}+(\Delta t)^p\right) + \left(1+\|M_r^{-1}\|_2+\varepsilon_+^{-1}+\varepsilon_++\varepsilon\right)^{\frac{1}{2}}\sqrt{\sum_{j=r+1}^{d}\lambda_j}\Bigg), \tag{5.27}$$

$$\frac{1}{N}\sum_{n=1}^{N}\|u_n-\omega_n^r\| \leq \qquad \tilde{C}\left(\left(1+\varepsilon^{-\frac{1}{2}}\right)M_u\left(h^{m+\frac{1}{2}}+(\Delta t)^p\right)+\sqrt{\sum_{j=r+1}^{d}\lambda_j}\right). \tag{5.28}$$

Here, M_u is a positive constant which depends on several norms of u and its time derivatives and the coefficients of (5.1), but not on ε^{-1}. The coefficient C is defined as before [8] *and $\tilde{C}$ is a product of* $\max\left\{1, \frac{\sqrt{\gamma}}{\sqrt{\varepsilon}}, \frac{\gamma}{\beta}\right\}$ *and an additional scalar*[9].

Proof: For the first inequality, from (5.16) and (5.18), follows:

$$\frac{1}{N}\sum_{n=1}^{N}\|u_n-\omega_n^r\| \leq \frac{1}{N}\sum_{n=1}^{N}\|\eta_n\|+\frac{1}{N}\sum_{n=1}^{N}\|\psi_n^r\| \leq$$
$$\frac{1}{N}\sum_{n=1}^{N}\|\eta_n\|+C\left(1+\|M_r^{-1}\|_2+\varepsilon_+^{-1}\right)^{\frac{1}{2}}\frac{1}{N}\sum_{n=1}^{N}\|\eta_n\|+C\left(\varepsilon+\varepsilon_+\right)^{\frac{1}{2}}\frac{1}{N}\sum_{n=1}^{N}\|\nabla\eta_n\|.$$

With (5.17), (5.23), (5.24), (5.25) and (5.26), we get:

$$\frac{1}{N}\sum_{n=1}^{N}\|u_n-\omega_n^r\| \leq$$
$$C\left(1+\left(1+\|M_r^{-1}\|_2+\varepsilon_+^{-1}\right)^{\frac{1}{2}}+\left(1+\varepsilon_+\varepsilon^{-1}\right)^{\frac{1}{2}}\right)M_u\left(h^{m+\frac{1}{2}}+(\Delta t)^p\right)+$$
$$C\left(1+\left(1+\|M_r^{-1}\|_2+\varepsilon_+^{-1}\right)^{\frac{1}{2}}+\left(\varepsilon_++\varepsilon\right)^{\frac{1}{2}}\right)\sqrt{\sum_{j=r+1}^{d}\lambda_j},$$

which implies (5.27).

[8] Note that C depends on β^{-1}. A very small β in Assumption 1 is theoretically possible.

[9] The scalar either does not depend on the coefficients of the problem, ore it depends on $\frac{\sqrt{\varepsilon+\varepsilon_+}}{\sqrt{\beta}}$ iff $\sqrt{\beta} < \sqrt{\varepsilon+\varepsilon_+}$.

The second estimate results from Assumption 1, especially with (5.7):

$$\begin{aligned}\|u_n-\omega_n^r\|_{\beta,\varepsilon,\varepsilon_+}^2 \leq & A\left(u_n-\omega_n^r, u_n-\omega_n^r\right)\\ = & A\left(u_n-\omega_n^r, u_n-u_n^h\right)+A\left(u_n-\omega_n^r, u_n^h-I_{h,r}\left(u_n\right)\right)\\ & +\underbrace{A\left(u_n-\omega_n^r, I_{h,r}\left(u_n\right)-\omega_n^r\right)}_{=0}\\ \leq & 3C_1\|u_n-\omega_n^r\|_{\beta,\varepsilon,\varepsilon_+}\left(\|u_n-u_n^h\|_{\beta,\varepsilon,\varepsilon_+}+\|u_n^h-I_{h,r}\left(u_n\right)\|_{\beta,\varepsilon,\varepsilon_+}\right).\end{aligned}$$

This implies:

$$\|u_n-\omega_n^r\|_{\beta,\varepsilon,\varepsilon_+} \leq 3C_1\left(\|u_n-u_n^h\|_{\beta,\varepsilon,\varepsilon_+}+\|u_n^h-I_{h,r}\left(u_n\right)\|_{\beta,\varepsilon,\varepsilon_+}\right).$$

Since P_R is an orthogonal projection, $\|P_R\|_{op}=1=\|\mathbb{I}-P_R\|_{op}=\|P_R'\|_{op}$, with the operator norm $\|P_R\|_{op}=\sup\limits_{u\in\nabla X}\frac{\|P_Ru\|}{\|u\|}$. We also have $\|P_Ru\|\leq\|P_R\|_{op}\|u\|=\|u\|$. With $\sqrt{\beta}\|u_n-\omega_n^r\|\leq\|u_n-\omega_n^r\|_{\beta,\varepsilon,\varepsilon_+}$ follows:

$$\|u_n-w_n^r\|\leq\tilde{C}\left(\|u_n-u_n^h\|_{H^1(\Omega)}+\|u_n^h-I_{h,r}\left(u_n\right)\|_{H^1(\Omega)}\right).$$

Therefore (5.28) follows with (5.23), (5.24) (5.25) and (5.26). □

In the further course of [16, Chapter 3], it can be observed that the work is still not done properly, for example in *Corollary 3.1* which states an estimate for $\|\left(u_n-\omega_n^r\right)_t\|$. The proof should be done in the same way as the proof of (5.15), but the *Ritz projection* $\omega_n^r\in X^r$ is not time dependent. One has to guess how $\left(\omega_n^r\right)_t$ is defined. Another example can be found in [16, (3.61)], where it is assumed that the Assumption 2 also applies in X^R. Since this assumption can be made because $X^h\longrightarrow X$ for $h\longrightarrow 0$, this can generally not be transferred to X^R which spans only functions with the characteristics of the snapshots.

As one can see there are gaps in the proof of the error estimate from [16].

6 Summary of the VMS-POD-Model to Generate the Algorithm Step by Step

This chapter contains a summary of the VMS-POD-Method which we use for the numerical studies. We chose the Crank-Nicolson method for time discretization, and the difference quotients are not included in the snapshot, but the mean value is subtracted.

1 Given Problem: Let $\Omega \subset \mathbb{R}^n$ for $n \in \{1,2,3\}$ be a bounded domain with Lipschitz boundary and $[0,T] \subset \mathbb{R}$ be a time interval.

One searches $u : [0,T] \times \Omega \longrightarrow \mathbb{R}$ such that

$$\begin{aligned} &\partial_t u - \varepsilon \Delta u + \mathbf{b} \cdot \nabla u + cu = f && \text{, in } (0,T] \times \Omega, \\ &u(0,x) = u_0(x) && \text{, in } \Omega, \\ &u(t,x) = 0 && \text{.on } (0,T] \times \partial\Omega, \end{aligned}$$

where we have $\varepsilon \ll 1$, $\mathbf{b} \in (L^\infty(0,T;L^\infty(\Omega)))^n$ with $\|\mathbf{b}\|_{(L^\infty(0,T;L^\infty(\Omega)))^n} \gg \varepsilon$, $c \in L^\infty(0,T;L^\infty(\Omega))$ and $f \in L^2(0,T;L^2(\Omega))$.

For the variational form find $u : (0,T] \longrightarrow X = H_0^1(\Omega)$, such that

$$(\partial_t u, v) + a(u,v) = (f,v) \ , \ \forall\, v \in X,$$

$a(u,v) = \varepsilon(\nabla u, \nabla v) + (\mathbf{b} \cdot \nabla u, v) + (cu, v)$, with $(\cdot,\cdot)$ as the standard $L^2(\Omega)$ inner product.

2 Determining the snapshots by solving a *full order model* (FOM) in $X^h \subset H_0^1(\Omega)$. Thus one has to find $u^h : (0,T] \longrightarrow X^h$ with

$$\left(\partial_t u^h, v^h\right) + a\left(u^h, v^h\right) = \left(f, v^h\right) \ , \ \forall\, v^h \in X^h.$$

Therefore N time points $\{t_1, ...t_N\} \in (0,T]$ are chosen, such that $t_N = T$, $t_0 = 0$ and $t_n - t_{n-1} = \Delta t \ \forall i \in \{1,...,N\}$. Solve the time discretised problem to get the solutions $w_n(x) := u^h(t_n, x)$ for $x \in \Omega$. Here w_0 is the orthogonal projection of u_0 onto X^h.

$$\{w_0, ..., w_N\} \in H_0^1(\Omega)$$

3 Calculate the mean value of the snapshots:

$$\langle w \rangle := \frac{1}{N+1} \sum_{i=0}^{N} w_i.$$

Subtract them from each w_i to get the new set of snapshots $\{\hat{w}_0, ..., \hat{w}_N\}$ with $\hat{w}_i = w_i - \langle w \rangle$ for all $i = 0, ..., N$. Adapt the problem accordingly.

4 Generate the matrix $W \in \mathbb{R}^{N+1\times N+1}$, with $W_{i,j} := (\hat{w}_i, \hat{w}_j)_{H^1(\Omega)}$ and calculate its nonzero eigenvalues λ_i, for $i = 1, ...d$ and the corresponding normalized eigenvectors $a_i = \left(a_i^1, ..., a_i^{N+1}\right)^T$. Choose the r largest eigenvalues:

$$\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_r > 0.$$

5 Create the POD-Basis $\{\phi_1, ..., \phi_r\}$, with

$$\phi_i(x) = \frac{1}{\sqrt{\lambda_i}} \sum_{j=0}^{2N} a_i^j \hat{w}_j(x) \ , \ x \in \Omega,$$

$((\phi_i, \phi_j)_{H^1(\Omega)} = \delta_{i,j})$.

6 P_R is the L^2-orthogonal projection onto $L^R := \nabla X^R = span\{\nabla\phi_1, .., \nabla\phi_R\}$ with $R < r$ and $P_R' = \mathbb{I} - P_R$.

Calculate an L^2-orthonormal basis of ∇X^R (which is equivalent to an H_0^1-orthonormal basis of X^R). One can do this with Gram–Schmidt process. Let $\left\{\nabla\tilde{\phi}_1, ..., \nabla\tilde{\phi}_R\right\}$ be this basis. Then follows for all $v \in X$:

$$P_R'\nabla v = v - \sum_{j=1}^{R} \left(\nabla\tilde{\phi}_j, v\right) \nabla\tilde{\phi}_j.$$

7 Solve the VMS-POD scheme with the Crank–Nicolson method. One searches $u_k^r \in X^r := span\{\phi_1, ..., \phi_r\}$:

$$\begin{aligned} \frac{1}{\Delta t}\left(u_k^r - u_{k-1}^r, \phi_j\right) + \frac{1}{2}a\left(u_{k-1}^r + u_k^r, \phi_j\right) + \frac{1}{2}\varepsilon_+\left(P_R'\nabla u_{k-1}^r + P_R'\nabla u_k^r, P_R'\nabla\phi_j\right) \\ = \frac{1}{2}\left(E_{k-1}(\phi_j) + E_k(\phi_j)\right) \ , \ \forall j = 1, ..., r, \end{aligned} \tag{6.1}$$

where $u_0^r = \sum_{i=1}^r (w_0, \phi_i)\phi_i$ and $E_k(\phi_j) = (f_k, \phi_j) - a(\langle w\rangle, \phi_j)$.

7 Implementation

In [16, Chapter 4] numerical studies are presented. They used an example for $\Omega \in \mathbb{R}^2$ and demonstrate that the POD-G model shows non-physical oscillations that can be reduced considerably with the VMS-POD model [16, Figure 1]. For the POD-G method it holds that the error reduces minimal where r increases [16, Table 1]. The average error $\frac{1}{N+1}\sum_{n=0}^{N}\|w_n - u_n^r\|$ between the snapshot w_n and the reduced order solution u_n^r for the VMS-POD method is 10^{-1} up to 10^{-2} times the average error of the POD-G method [16, Table 4]. The results show that $r = 40$ and $R = 20$ are a good choice [16, Table 3]. For the factor ε_+ they used $0.01\alpha^*$, α^* and $100\alpha^*$, with $\alpha^* = \max\left\{\tilde{\alpha}, \frac{h}{2}\right\}$ and

$$\tilde{\alpha} = \frac{h^{m+1} + \sqrt{\sum_{j=r+1}^{d}\lambda_j}}{2h^m + \sqrt{\sum_{j=r+1}^{d}\lambda_j} + \sqrt{\sum_{j=R+1}^{d}\lambda_j}},$$

which comes from the error estimation of [16]. In [16, Table 4 and Table 5] one can observe a low sensitivity of the VMS-POD method with respect to changes in ε.

The following part describes the implementation for the one dimensional case. First, the architecture used to code is outlined. Then results of numerical test are presented. In contrast to [16], we use different scalings of h as ε_+. The second difference lies in the time discretization.

7.1 Finite Element Approximation and Equations

We use piecewise quadratic functions as in [16]. With the Crank-Nicolson method we have the same order of convergence for the spacial and temporal discretization. The difference between VMS-POD and POD-G method is bigger with linear FEs, therefore these are also included for the visualization.

This leads to the following procedure (which is actually a *finite difference method*). The interval (a, b) is divided in $2M$ equidistant parts of length $\frac{h}{2}$, $x_0 = a$, $x_{2M} = b$ and $\forall\, k \in \{1, 2, ..., 2M-1\}$. One constructs the ansatz space $V^h = span\,\{\beta_1, ..., \beta_{2M}\}$.

For $k, l \in \{1, 2, ..., 2M-1\}$ with k is odd and l is even we have the following basis functions:

$$\beta_k(x) = \begin{cases} \frac{4(x-x_{k-1})(x_{k+1}-x)}{h^2} & , x \in (x_{k-1}, x_{k+1}), \\ 0 & , else, \end{cases}$$

$$\beta_l(x) = \begin{cases} \frac{2(x-x_{l-2})(x-x_{l-1})}{h^2} & , x \in (x_{l-2}, x_l), \\ \frac{2(x-x_{l+2})(x-x_{l+1})}{h^2} & , x \in [x_l, x_{l+2}), \\ 0 & , else. \end{cases}$$

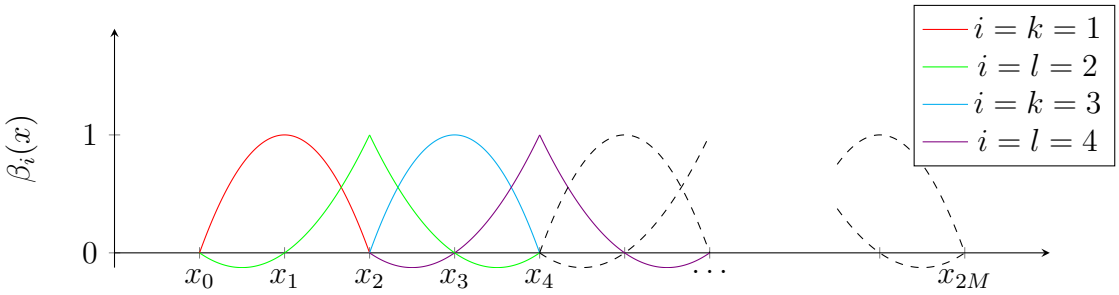

Figure 7.1: piecewise quadratic finite element functions

It applies almost everywhere:

$$\beta_k'(x) = \begin{cases} \frac{4(-2x+x_{k+1}+x_{k-1})}{h^2} & , x \in (x_{k-1}, x_{k+1}), \\ 0 & , else, \end{cases}$$

$$\beta_l'(x) = \begin{cases} \frac{2(2x-x_{l-1}-x_{l-2})}{h^2} & , x \in (x_{l-2}, x_l), \\ \frac{2(2x-x_{l+1}-x_{l+2})}{h^2} & , x \in (x_l, x_{l+2}), \\ 0 & , else. \end{cases}$$

We define:

$$\beta_{l,1} := \beta_l|_{(x_{l-2}, x_l)},$$
$$\beta_{l,2} := \beta_l|_{(x_l, x_{l+2})}.$$

In the reference cell (Figure 7.2) we have the following basis functions:

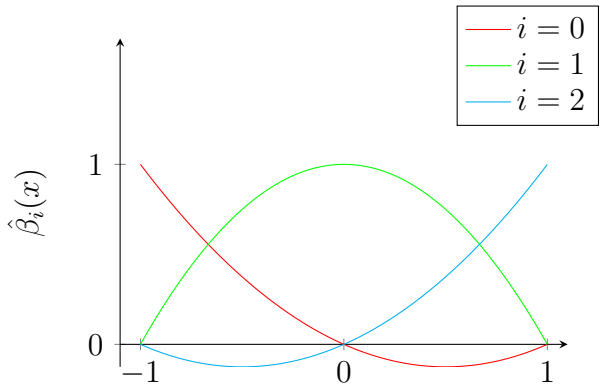

Figure 7.2: reference cell

$$\hat{\beta}_0 = \frac{x^2 - x}{2}, \qquad \hat{\beta}_0' = \frac{2x - 1}{2},$$
$$\hat{\beta}_1 = -x^2 + 1, \qquad \hat{\beta}_1' = -2x,$$
$$\hat{\beta}_2 = \frac{x^2 - x}{2}, \qquad \hat{\beta}_2' = \frac{2x + 1}{2}.$$

For the most $i,j \in \{1,...,2M-1\}$ holds $\int_{x_0}^{x_{2M}} \beta_i \beta_j dx = 0$. Only the following integrals are $\neq 0$ (equivalently for $\int_{x_0}^{x_{2M}} \beta_i' \beta_j' dx$):

$$\begin{aligned}
&\int_{x_0}^{x_{2M}} \beta_k^2 dx = \int_{x_{k-1}}^{x_{k+1}} \beta_k^2, \\
&\int_{x_0}^{x_{2M}} \beta_l^2 dx = \int_{x_0}^{x_M} \beta_{l,1}^2 dx + \int_{x_0}^{x_M} \beta_{l,2}^2 dx = \int_{x_{l-2}}^{x_l} \beta_{l,1}^2 dx + \int_{x_l}^{x_{l+2}} \beta_{l,2}^2 dx, \\
&\int_{x_0}^{x_{2M}} \beta_k \beta_{k+1} dx = \int_{x_0}^{x_M} \beta_k \beta_{k+1,1} dx = \int_{x_{k-1}}^{x_{k+1}} \beta_k \beta_{k+1,1} dx, \\
&\int_{x_0}^{x_{2M}} \beta_k \beta_{k-1} dx = \int_{x_0}^{x_M} \beta_k \beta_{k-1,2} dx = \int_{x_{k-1}}^{x_{k+1}} \beta_k \beta_{k-1,2} dx, \\
&\int_{x_0}^{x_{2M}} \beta_l \beta_{l+2} dx = \int_{x_0}^{x_M} \beta_{l,2} \beta_{l+2,1} dx = \int_{x_l}^{x_{l+2}} \beta_{l,2} \beta_{l+2,1} dx.
\end{aligned}$$

With $\varphi_j(y) = \frac{h}{2} y + x_j$, for $j = \{1,...,2M-1\}$, we have $\beta_k(\varphi_k(y)) = \hat{\beta}_1(y)$, $\beta_{l,1}(\varphi_l(y)) = \hat{\beta}_2(y)$ and $\beta_{l,2}(\varphi_l(y)) = \hat{\beta}_0(y)$ for the derivatives $\beta_k'(\varphi_k(y)) = \frac{2}{h}\hat{\beta}_1'(y)$, $\beta_{l,1}'(\varphi_l(y)) = \frac{2}{h}\hat{\beta}_2'(y)$ and $\beta_{l,2}'(\varphi_l(y)) = \frac{2}{h}\hat{\beta}_0'(y)$. By the substitution formula

$$\int_a^b f(x)\, dx = \int_{\varphi^{-1}(a)}^{\varphi^{-1}(b)} f(\varphi(y))\, \varphi'(y)\, dy$$

follows, for all $k,l \in \{1,2,...,2M-1\}$ with k is odd and l is even:

$$\begin{aligned}
&\int_{x_{k-1}}^{x_{k+1}} \beta_k^2 = \frac{h}{2} \int_{-1}^{1} \hat{\beta}_1^2 = \frac{16h}{30}, \\
&\int_{x_{l-2}}^{x_l} \beta_{l,1}^2 dx + \int_{x_l}^{x_{l+2}} \beta_{l,2}^2 dx = \frac{h}{2} \int_{-1}^{1} \hat{\beta}_0^2 + \frac{h}{2} \int_{-1}^{1} \hat{\beta}_2^2 = \frac{8h}{30}, \\
&\int_{x_{k-1}}^{x_{k+1}} \beta_k \beta_{k+1,1} dx = \frac{h}{2} \int_{-1}^{1} \hat{\beta}_1 \hat{\beta}_2 = \frac{2h}{30}, \\
&\int_{x_{k-1}}^{x_{k+1}} \beta_k \beta_{k-1,2} dx = \frac{h}{2} \int_{-1}^{1} \hat{\beta}_1 \hat{\beta}_0 = \frac{2h}{30}, \\
&\int_{x_l}^{x_{l+2}} \beta_{l,2} \beta_{l+2,1} dx = \frac{h}{2} \int_{-1}^{1} \hat{\beta}_0 \hat{\beta}_2 = -\frac{h}{30},
\end{aligned}$$

as well as:

$$\int_{x_{k-1}}^{x_{k+1}} (\beta_k')^2 = \frac{2}{h}\int_{-1}^{1} \left(\hat{\beta}_1'\right)^2 = \frac{16}{3h},$$
$$\int_{x_{l-2}}^{x_l} \left(\beta_{l,1}'\right)^2 dx + \int_{x_l}^{x_{l+2}} \left(\beta_{l,2}'\right)^2 dx = \frac{2}{h}\int_{-1}^{1} \left(\hat{\beta}_0'\right)^2 + \frac{2}{h}\int_{-1}^{1} \left(\hat{\beta}_2'\right)^2 = \frac{14}{3h},$$
$$\int_{x_{k-1}}^{x_{k+1}} \beta_k'\beta_{k+1,1}' dx = \frac{2}{h}\int_{-1}^{1} \hat{\beta}_1'\hat{\beta}_2' = -\frac{8}{3h},$$
$$\int_{x_{k-1}}^{x_{k+1}} \beta_k'\beta_{k-1,2}' dx = \frac{2}{h}\int_{-1}^{1} \hat{\beta}_1'\hat{\beta}_0' = -\frac{8}{3h},$$
$$\int_{x_l}^{x_{l+2}} \beta_{l,2}'\beta_{l+2,1}' dx = \frac{2}{h}\int_{-1}^{1} \hat{\beta}_0'\hat{\beta}_2' = \frac{1}{3h}.$$

The subspace $V^h \subset H_0^1((a,b))$ is isomorphic to $\mathbb{R}^{2M-1}$ and each function

$$v^h \;=\; \sum_{i=1}^{M-1} \left(v^h\right)^i \beta_i \;\cong\; \left(\left(v^h\right)^1, \ldots, \left(v^h\right)^{2M-1}\right)^T \;=:\; \overline{v} \in \mathbb{R}^{2M-1}.$$

We generate the matrices $B_0, B_1, B_2 \in \mathbb{R}^{2M-1\times 2M-1}$, where $(B_0)_{i,j} = (\beta_i, \beta_j)_{H_0^1((a,b))}$, $(B_1)_{i,j} = (\beta_i, \beta_j)_{H^1((a,b))}$, $(B_2)_{i,j} = (\beta_i, \beta_j)_{L^2((a,b))}$. For all $u, v \in V^h$ and the corresponding $\overline{u}, \overline{v} \in \mathbb{R}^{2M-1}$ it holds:

$$(u,v)_{H_0^1((a,b))} = \overline{u}^T B_0 \overline{v},$$
$$(u,v)_{H^1((a,b))} = \overline{u}^T B_1 \overline{v},$$
$$(u,v)_{L^2((a,b))} = \overline{u}^T B_2 \overline{v}.$$

The matrices are invertible and symmetric, therefore $(\overline{u}, \overline{v})_{B_l} := \overline{u}^T B_l \overline{v}$, for $l = 0, 1, 2$ define scalar products in $\mathbb{R}^{2M-1}$.

$$B_0 = \frac{1}{3h}\begin{pmatrix} 16 & -8 & 0 & 0 & 0 & 0 & \cdots & 0 \\ -8 & 14 & -8 & 1 & 0 & 0 & \cdots & 0 \\ 0 & -8 & 16 & -8 & 0 & 0 & \cdots & 0 \\ 0 & 1 & -8 & 14 & -8 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & -8 & 16 \end{pmatrix},$$
$$B_2 = \frac{h}{30}\begin{pmatrix} 16 & 2 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 2 & 8 & 2 & -1 & 0 & 0 & \cdots & 0 \\ 0 & 2 & 16 & 2 & 0 & 0 & \cdots & 0 \\ 0 & -1 & 2 & 8 & 2 & -1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & 2 & 16 \end{pmatrix},$$
$$B_1 = B_2 + B_0.$$

For linear FEs we get the $\mathbb{R}^{M-1\times M-1}$ matrices[1]:

$$B_0^{lin} = \frac{1}{h}\begin{pmatrix} 2 & -1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & -1 & 2 \end{pmatrix},$$

$$B_2^{lin} = \frac{h}{6}\begin{pmatrix} 4 & 1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 1 & 4 & 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 4 & 1 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 4 \end{pmatrix},$$

$$B_1^{lin} = B_2^{lin} + B_0^{lin}.$$

Let $(w_0, ..., w_n)$ be the snapshots, $\langle w\rangle = \frac{1}{N+1}\sum_{i=0}^{N} w_i$ their mean value and $\hat{w}_i = w_i - \langle w\rangle$, for $i = 0, ...N$. With the calculations above, one can treat $\langle w\rangle$ and the new snapshots $\{\hat{w}_0, \ldots, \hat{w}_N\}$ as vectors $\langle\overline{w}\rangle, \overline{w}_i \in \mathbb{R}^{2M-1}$ for $i = 0, ..., N$. We define the matrix $U \in \mathbb{R}^{(2M-1)\times(N+1)}$ with $\overline{w}_i$ as column i. This leads to the following relation for the matrix W, $W_{i,j} = (\hat{w}_i, \hat{w}_j)_X$:

$$W = U^T B_l U.$$

Also the POD-Basis functions $\{\phi_1, \ldots, \phi_r\}$ have representations in $\mathbb{R}^{2M-1}$ of the form:

$$\overline{\phi_i} = \frac{1}{\sqrt{\lambda_i}} U a_i,$$

with the r largest eigenvalues of W, $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r \geq 0$ and the corresponding eigenvectors a_i.

Let $\Phi \in \mathbb{R}^{(2M-1)\times r}$ be the matrix with the POD-Basis vectors $\overline{\phi_i}$ as columns, $N_k, C_k, P_R' \in \mathbb{R}^{(2M-1)\times(2M-1)}$ such that $(N_k)_{i,j} = (\mathbf{b}_k \cdot \nabla\beta_i, \beta_j)_{L^2}$, $(C_k)_{i,j} = (c_k\beta_i, \beta_j)_{L^2}$ and P_R' the corresponding Matrix for the projection $P_R'(\cdot)$, $\overline{f}_k \in \mathbb{R}^{2M-1}$ with $\left(\overline{f}_k\right)_i = f(t_k, x_i)$, and $E_k = \Phi^T B_2 \overline{f}_k - \varepsilon\Phi^T B_0 \langle\overline{w}\rangle - \Phi^T C_k \langle\overline{w}\rangle - \frac{1}{2}\Phi^T N_k^T \langle\overline{w}\rangle$.

One can transfer the equation (6.1) into a linear system to find $\overline{b}_k \in \mathbb{R}^r$ such that:

$$\begin{aligned}&\left[\frac{1}{\Delta t}\Phi^T B_2\Phi + \frac{\varepsilon}{2}\Phi^T B_0\Phi + \frac{1}{2}\Phi^T N_k^T\Phi + \frac{1}{2}\Phi^T C_k\Phi + \frac{\varepsilon_+}{2}\left(P_R'\right)^T B_0 P_R'\right]\overline{b}_k = \\ &\left[\frac{1}{\Delta t}\Phi^T B_2\Phi - \frac{\varepsilon}{2}\Phi^T B_0\Phi - \frac{1}{2}\Phi^T N_{k-1}^T\Phi - \frac{1}{2}\Phi^T C_{k-1}\Phi - \frac{\varepsilon_+}{2}\left(P_R'\right)^T B_0 P_R'\right]\overline{b}_{k-1} \\ &+ \frac{1}{2}\left(E_{k-1} + E_k\right),\end{aligned} \tag{7.1}$$

where $\overline{b}_0 = \Phi^T B_1 \overline{w}_0$.

[1] In the linear case one divides the interval (a, b) into M equidistant parts of length h and has also just $M - 1$ basis functions.

Let $\gamma_k \in \mathbb{R}^r$ be the right-hand side and $\Gamma_k \in \mathbb{R}^{r\times r}$ the matrix on the left-hand side of (7.1) at step k. Since $\Gamma_k \bar{b}_k = \gamma_k$, one can see the following:

$$\gamma_{k+1} = \frac{1}{2}\left(E_k + E_{k+1}\right) + \frac{2}{\Delta t}\Phi^T B_2 \Phi \bar{b}_k - \gamma_k.$$

For the L^2-orthogonal projection onto ∇X^R one needs a basis of $span\left\{\overline{\phi_i}\right\}_{i=1}^R$ which is orthonormal with respect to $(\cdot,\cdot)_{B_0}$, lets call $P_R^* \in \mathbb{R}^{(2M-1)\times R}$ the matrix of these ON-basis vectors. If $(\cdot,\cdot)_X = (\cdot,\cdot)_{H_0^1}$ nothing is to do for it and one can use the first R columns of Φ, but if $(\cdot,\cdot)_X = (\cdot,\cdot)_{H^1}$ one has to calculate P_R^* for example with Gram–Schmidt process. Then P_R' can be calculated as follows:

$$P_R' = \Phi - P_R^*\left(P_R^*\right)^T B_0 \Phi. \tag{7.2}$$

In the case of $(\cdot,\cdot)_X = (\cdot,\cdot)_{H_0^1}$ one obviously does not have to calculate $\Phi^T B_0 \Phi$ and $\left(P_R'\right)^T B_0 P_R'$ in (7.1) in this way. They are either the identity matrix or the identity matrix with the first R diagonal entries replaced by zero.

To interpret the results one can retransform the solutions $\bar{b}_k$ into $\mathbb{R}^{2M-1}$ by calculating $\Phi\bar{b}_k \in \mathbb{R}^{2M-1}$.

7.2 Example and Numerical Studies

In this part numerical studies are presented. The following example is examined:

$$u_t - \varepsilon u_{xx} + u_x + u = f, \tag{7.3}$$

where $\Omega = (0,1)$, $u(t,0) = u(t,1) = 0$, $\forall t \in (0,1]$ and f is chosen to satisfy the solution

$$u(t,x) = \exp\left(1 - t - 100\,(x-t)^2\right)\left(x - \frac{e^{\frac{x}{z}} - 1}{e^{\frac{1}{z}} - 1}\right), \tag{7.4}$$

with $z = 0.01$.

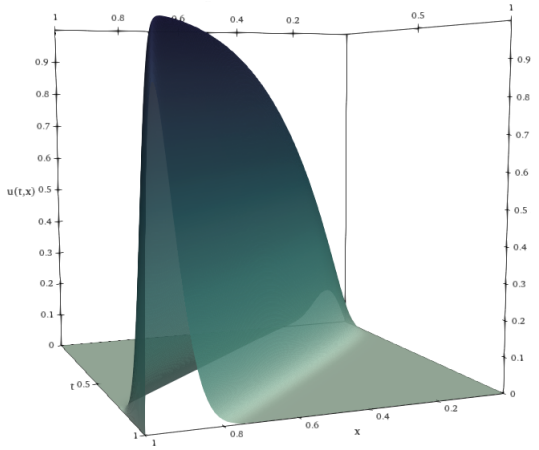

Figure 7.3: exact solution (7.4)

For all $k = 0, ..., N$ holds: $C_k = B_2$ and

$$N_k = \frac{1}{6}\begin{pmatrix} 0 & -4 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 4 & 0 & -4 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 4 & 0 & -4 & 0 & 0 & \cdots & 0 \\ 0 & -1 & 4 & 0 & -4 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & 4 & 0 \end{pmatrix}.$$

With linear FEs we have:

$$N_k^{lin} = \frac{1}{2}\begin{pmatrix} 0 & -1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 0 \end{pmatrix}.$$

The code is implemented in C++. The library Eigen [1] is used, especially its integrated solver for linear equations, the eigenvalue solver for selfadjoint matrices and the matrix and vector representation. Some of the source code and a link to the full implementation can be found in the *Appendix*. All computations are carried out on a PC with 2.80 GHz Intel core i7-1165G7 processor.

For the chosen example it turns out that the results with $(\cdot,\cdot)_X = (\cdot,\cdot)_{H^1(\Omega)}$ or $(\cdot,\cdot)_X = (\cdot,\cdot)_{H_0^1(\Omega)}$ are very similar. Therefore only the POD method with $(\cdot,\cdot)_X = (\cdot,\cdot)_{H^1(\Omega)}$ is shown here.

In the following, $h = \frac{1}{M}$ and $\Delta t = \frac{1}{N}$. For linear FEs we use $M = N = 500$, for the quadratic case we use $M = 250$ and $N = 500$ to get the same number of *nodes*. Results for $\varepsilon = 10^{-10}$ and $\varepsilon = 10^{-4}$ are presented, in particular, the average error and some graphs of the solutions. For the snapshots we do not solve a FOM. Instead, we project the exact solution (7.4) at times t_i, $i = 0, ..., N$ onto the FE ansatz space.

	linear FE ($M = N = 500$)		quadratic FE ($2M = N = 500$)	
	$\varepsilon = 10^{-10}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-10}$	$\varepsilon = 10^{-4}$
r	e^*	e^*	e^*	e^*
20	0.00059517	0.00059517	0.000596332	0.000596332
25	$1.20173e-05$	$1.20173e-05$	$1.20462e-05$	$1.20462e-05$
30	$7.51773e-08$	$7.51773e-08$	$7.61374e-08$	$7.61374e-08$
35	$4.99111e-08$	$4.99111e-08$	$5.35439e-08$	$5.35439e-08$
40	$4.61657e-08$	$4.61657e-08$	$4.78322e-08$	$4.78322e-08$
45	$4.69651e-08$	$4.69651e-08$	$4.87872e-08$	$4.87872e-08$

Table 7.1: error $e^* = \frac{1}{N+1}\sum_{i=0}^{N} \|w_i - \sum_{j=1}^{r} (w_i, \phi_j)_{H^1}\, \phi_j\|_X$ of the POD basis

		linear FE ($M = N = 500$)		quadratic FE ($2M = N = 500$)	
$r = 20$		$\varepsilon = 10^{-10}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-10}$	$\varepsilon = 10^{-4}$
R	ε_+	e	e	e	e
0 $(POD - G)$		**0.00051805**	**0.000508049**	**0.000254201**	**0.000250956**
5	$\frac{h}{2}$	0.00704167	0.0069049	0.0120214	0.0118269
5	h	0.0120371	0.0118418	0.0192404	0.0189977
5	$2h$	0.0192315	0.0189888	0.0287432	0.0284693
5	$4h$	0.0287161	0.0284427	0.0407502	0.0404592
5	$8h$	0.0407091	0.04041864	0.0559327	0.0556287
10	$\frac{h}{2}$	0.00204509	0.00198173	0.00310549	0.00301802
10	h	0.00317917	0.00309048	0.00456577	0.00445816
10	$2h$	0.00459659	0.00448839	0.0060935	0.00597603
10	$4h$	0.00609179	0.00597429	0.0075832	0.00745822
10	$8h$	0.00756255	0.00743767	0.00899383	0.00885755
15	$\frac{h}{2}$	0.000532824	0.000520476	0.00034653	0.000336712
15	h	0.000554989	0.000540928	0.000412372	0.000399509
15	$2h$	0.000582461	0.000566706	0.000484588	0.000469666
15	$4h$	0.000611104	0.000594471	0.000549381	0.000534172
15	$8h$	0.000647748	0.000631246	0.000605631	0.000590718
$r = 25$		$\varepsilon = 10^{-10}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-10}$	$\varepsilon = 10^{-4}$
R	ε_+	e	e	e	e
0 $(POD - G)$		0.000533348	0.000511758	0.000163984	0.000159731
10	$\frac{h}{2}$	0.00209539	0.002023	0.00307866	0.00299195
10	h	0.00314503	0.00305573	0.00446032	0.00435958
10	$2h$	0.0044733	0.00437208	0.00598765	0.0058749
10	$4h$	0.00597869	0.005866	0.00756152	0.00743643
10	$8h$	0.00754113	0.00741608	0.00907035	0.00893199
15	$\frac{h}{2}$	0.000492047	0.000472475	0.000288704	0.000277824
15	h	0.000479655	0.000461918	0.000355841	0.000343721
15	$2h$	0.00047012	0.000455025	0.000427869	0.000414593
15	$4h$	0.000494123	0.000480014	0.00049791	0.000483624
15	$8h$	0.000561276	0.000546189	0.000554994	0.000540458
20	$\frac{h}{2}$	0.000431507	0.000415878	0.000135143	0.000132312
20	h	0.000362918	0.000351117	0.000121558	0.000119298
20	$2h$	0.000284273	0.00027669	0.000111106	0.000109207
20	$4h$	**0.000234063**	**0.000229099**	**0.000107871**	**0.000106049**
20	$8h$	0.000237994	0.000233106	0.000110159	0.000108279

Table 7.2: average error $e = \frac{1}{N+1} \sum_{i=0}^{N} \|w_i - u_i^r\|_{L^2}$ of the ROM solution

		linear FE ($M = N = 500$)		quadratic FE ($2M = N = 500$)	
$r = 30$		$\varepsilon = 10^{-10}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-10}$	$\varepsilon = 10^{-4}$
R	ε_+	e	e	e	e
0 ($POD - G$)		0.000612551	0.000581997	0.000195589	0.00018924
15	$\frac{h}{2}$	0.000504372	0.000481686	0.000291252	0.000280692
15	h	0.000464171	0.000445891	0.000354111	0.000342204
15	$2h$	0.00044657	0.000431992	0.000426424	0.000413247
15	$4h$	0.000483291	0.000469449	0.000496235	0.000482076
15	$8h$	0.000557955	0.00054299	0.000553434	0.000538953
20	$\frac{h}{2}$	0.00045707	0.000437057	0.000145091	0.000141546
20	h	0.000360395	0.000346749	0.000124016	0.000121548
20	$2h$	0.0002639	0.000256478	0.000110554	0.000108656
20	$4h$	**0.000218089**	**0.000213595**	**0.00010724**	**0.000105436**
20	$8h$	0.000229356	0.000224731	0.0001097	0.000107833
25	$\frac{h}{2}$	0.000506101	0.000482779	0.00015591	0.000151765
25	h	0.000430673	0.000412452	0.000135555	0.000132531
25	$2h$	0.000341493	0.000329366	0.000120434	0.000118202
25	$4h$	0.000283807	0.000275728	0.000118439	0.00011632
25	$8h$	0.000290571	0.000282455	0.000126959	0.000124467
$r = 35$		$\varepsilon = 10^{-10}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-10}$	$\varepsilon = 10^{-4}$
R	ε_+	e	e	e	e
0 ($POD - G$)		0.000709584	0.000664541	0.00026741	0.000253678
15	$\frac{h}{2}$	0.000511666	0.000486017	0.000304615	0.000293085
15	h	0.000448433	0.000429729	0.000356582	0.000344748
15	$2h$	0.000427312	0.000413437	0.000426133	0.00041304
15	$4h$	0.000476101	0.000462521	0.000496164	0.000482003
15	$8h$	0.000556806	0.000541852	0.000553397	0.000538915
20	$\frac{h}{2}$	0.000463907	0.000440771	0.00015486	0.000150633
20	h	0.000338455	0.000324738	0.000125243	0.000122736
20	$2h$	0.000235054	0.000228825	0.000109513	0.000107661
20	$4h$	**0.000204815**	**0.000200878**	**0.000106524**	**0.000104741**
20	$8h$	0.000226081	0.000221561	0.000109446	0.000107584
25	$\frac{h}{2}$	0.000501157	0.000475167	0.000159381	0.000154845
25	h	0.000386622	0.0003694	0.000129143	0.00012646
25	$2h$	0.000282076	0.000272794	0.000114257	0.000112278
25	$4h$	0.000248358	0.000241948	0.00011571	0.000113659
25	$8h$	0.000278528	0.000270832	0.000125869	0.000123395

Table 7.3: average error $e = \frac{1}{N+1} \sum_{i=0}^{N} \|w_i - u_i^r\|_{L^2}$ of the ROM solution

		linear FE ($M = N = 500$)		quadratic FE ($2M = N = 500$)	
$r = 35$		$\varepsilon = 10^{-10}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-10}$	$\varepsilon = 10^{-4}$
R	ε_+	e	e	e	e
30	$\frac{h}{2}$	0.000564115	0.000533707	0.000169539	0.000164314
30	h	0.00048499	0.000460593	0.000140992	0.000137769
30	$2h$	0.00040446	0.000386	0.000134123	0.000131101
30	$4h$	0.000373458	0.000357623	0.000146586	0.000142734
30	$8h$	0.000415833	0.000397421	0.000163841	0.000159058
$r = 40$		$\varepsilon = 10^{-10}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-10}$	$\varepsilon = 10^{-4}$
R	ε_+	e	e	e	e
0 ($POD - G$)		0.000866393	0.000781988	0.000334636	0.000310134
15	$\frac{h}{2}$	0.000472264	0.000441799	0.000298292	0.000286668
15	h	0.000393035	0.000376074	0.000352422	0.000340661
15	$2h$	0.000402984	0.000390335	0.000425225	0.000412134
15	$4h$	0.000472103	0.000458643	0.000496031	0.000481868
15	$8h$	0.00055631	0.00054136	0.000553374	0.000538891
20	$\frac{h}{2}$	0.00042411	0.000395613	0.000150104	0.000145433
20	h	0.000277623	0.000265833	0.000119522	0.000117179
20	$2h$	0.000205034	0.00020047	0.000107545	0.000105749
20	$4h$	**0.000199108**	**0.000195407**	**0.000106175**	**0.000104397**
20	$8h$	0.000225279	0.000220778	0.000109395	0.000107534
25	$\frac{h}{2}$	0.000457538	0.000425173	0.000154934	0.000149792
25	h	0.000309712	0.000294955	0.000122358	0.000119866
25	$2h$	0.000235854	0.000229336	0.000111742	0.000109847
25	$4h$	0.000238134	0.000232226	0.000115348	0.000113305
25	$8h$	0.000277238	0.000269579	0.000125872	0.000123396
30	$\frac{h}{2}$	0.000538717	0.000496573	0.000175837	0.000168619
30	h	0.000393775	0.000369912	0.000139306	0.00013553
30	$2h$	0.000310719	0.000297543	0.000131801	0.000128699
30	$4h$	0.000326864	0.000313874	0.000145299	0.000141462
30	$8h$	0.0003996	0.000382074	0.000163246	0.000158475

Table 7.4: average error $e = \frac{1}{N+1} \sum_{i=0}^{N} \|w_i - u_i^r\|_{L^2}$ of the ROM solution

		linear FE ($M = N = 500$)		quadratic FE ($2M = N = 500$)	
$r = 45$		$\varepsilon = 10^{-10}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-10}$	$\varepsilon = 10^{-4}$
R	ε_+	e	e	e	e
0 $(POD - G)$		0.000864447	0.000779656	0.000337693	0.000312948
15	$\frac{h}{2}$	0.00047059	0.000440287	0.000298821	0.000287157
15	h	0.000392357	0.000375466	0.000352589	0.000340824
15	$2h$	0.000402807	0.000390168	0.000425286	0.000412195
15	$4h$	0.00047205	0.00045859	0.000496048	0.000481885
15	$8h$	0.000556298	0.000541348	0.000553378	0.000538896
20	$\frac{h}{2}$	0.000422292	0.000393967	0.000150736	0.000146014
20	h	0.000276819	0.000265112	0.000119762	0.000117411
20	$2h$	0.000204788	0.000200238	0.000107666	0.000105869
20	$4h$	**0.000199035**	**0.000195336**	**0.000106214**	**0.000104436**
20	$8h$	0.000225263	0.000220763	0.000109403	0.000107542
25	$\frac{h}{2}$	0.000455347	0.000423198	0.000155636	0.000150433
25	h	0.000308578	0.000293943	0.000122592	0.00012009
25	$2h$	0.000235503	0.000229009	0.000111835	0.000109939
25	$4h$	0.000238056	0.000232152	0.000115374	0.000113331
25	$8h$	0.000277226	0.000269567	0.000125877	0.000123402
30	$\frac{h}{2}$	0.000535269	0.000493452	0.000176952	0.000169626
30	h	0.000391117	0.000367535	0.000139696	0.000135889
30	$2h$	0.000309356	0.00029631	0.00013187	0.000128765
30	$4h$	0.00032644	0.000313483	0.000145292	0.000141456
30	$8h$	0.000399509	0.000381988	0.000163237	0.000158467

Table 7.5: average error $e = \frac{1}{N+1} \sum_{i=0}^{N} \|w_i - u_i^r\|_{L^2}$ of the ROM solution

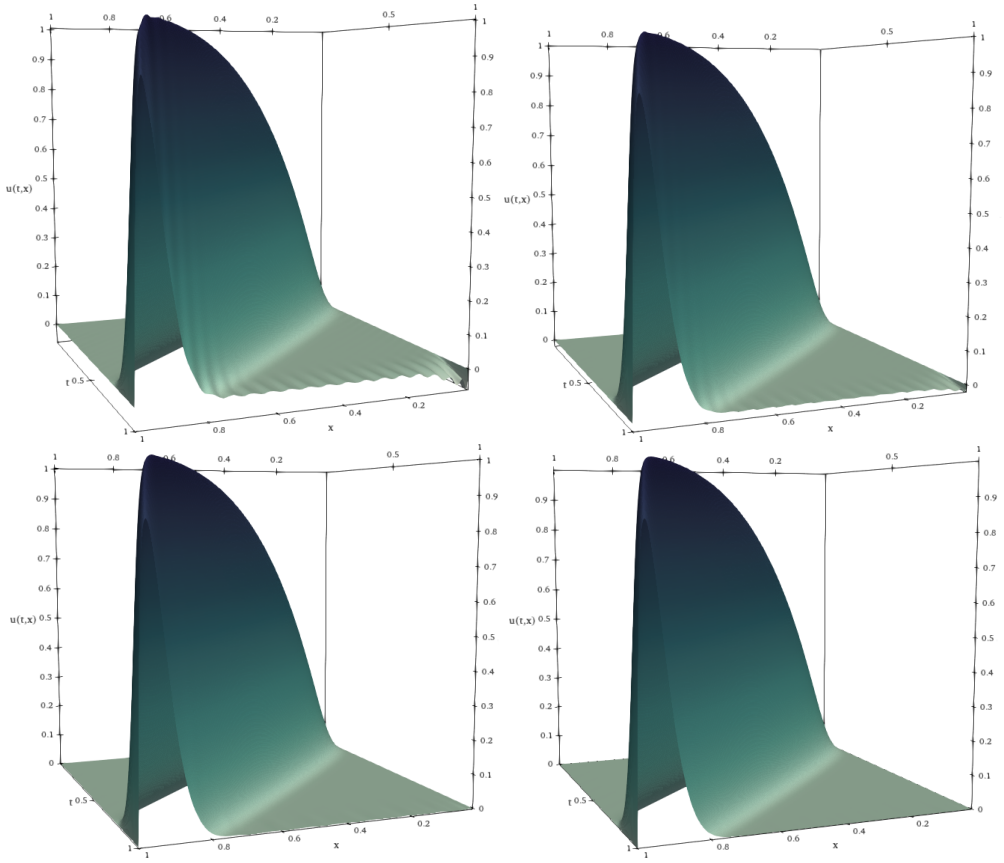

Figure 7.4: reduced order solutions of (7.3) for $\varepsilon = 10^{-10}$, with $r = 40$ (from top left to bottom right: POD-G with linear FE, POD-G with quadratic FE, VMS-POD with linear FE and $R = 20$, VMS-POD with quadratic FE and $R = 20$)

Between $\varepsilon = 10^{-10}$ and $\varepsilon = 10^{-4}$ only a small difference in the results is visible. This supports the observations of [16] about the low sensitivity with respect to changes of the diffusion coefficient.

In Table 7.2-7.5 one can observe an advantage of the VMS-POD method compared to the POD-G method. For all $r > 20$ tested, the average error for the VMS-POD method, with $R = 20$ and $\varepsilon_+ = 4h$, is better than the average error of the POD-G solution. With linear FEs the difference is significantly larger. One can also see that the POD-G method for $r = 25$ with quadratic FEs leads to better results than any other test for the linear case. Considering the effort, the POD-G model with $r = 25$ delivers quite good results.

The results for the linear FE method show that the average error for the POD-G increases if r increases. The error for the VMS-POD method with $R = 20$ and $\varepsilon_+ = 4h$ decreases until $r = 40$. The best results came from the VMS-POD method with $r = 40$, $R = 20$ and $\varepsilon_+ = 4h$. The outcome is similar for quadratic FEs but the POD-G method with $r = 25$ seems to be better than the method with $r = 20$.

For $r = 45$ we do not get better results. One can probably see the main reason in Table 7.1. The error $e^* = \frac{1}{N+1} \sum_{i=0}^{N} \|w_i - \sum_{j=1}^{r} (w_i, \phi_j)_X \phi_j\|_{H^1}$ has increased from $r = 40$ to $r = 45$. But from (2.11) follows $e^* = \sum_{i=r+1}^{d} \lambda_i$ and since all λ_i should be ≥ 0 this indicates an error in the calculation of the smallest eigenvalues and it

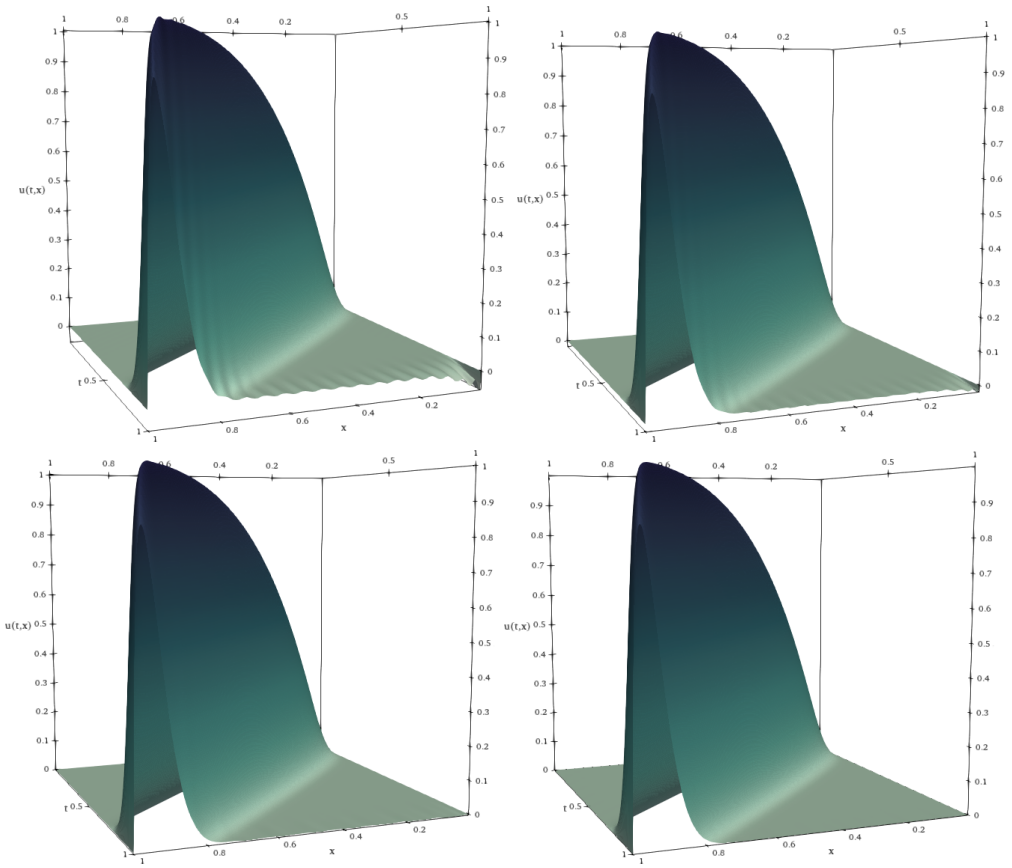

Figure 7.5: reduced order solutions of (7.3) for $\varepsilon = 10^{-4}$, with $r = 40$ (from top left to bottom right: POD-G with linear FE, POD-G with quadratic FE, VMS-POD with linear FE and $R = 20$, VMS-POD with quadratic FE and $R = 20$)

makes no sense to use them. The outcome of $r = 40$ and $R = 20$ as best choice matches the results in [16].

In Figure 7.4 and Figure 7.5 we see that the VMS-POD method does exactly what we expect. It levels the non-physical oscillations that occur with the POD-G method. The difference between the exact solution in Figure 7.3 and the VMS-POD solutions is barely visible.

The chosen example may not be a strong challenge for the method, but an advantage is visible. The effort of solving linear equations in $\mathbb{R}^{40}$ is much smaller than in $\mathbb{R}^{499}$. We compare the CPU time of a VMS-POD method with $r = 40$ and a full order method. The VMS-POD method runs around 45 seconds with different solvers, while the full order method needs nearly 3 minutes with the faster solver[2].

[2]The solvers of 'fullPivHouseholderQr' and 'partialPivLu' are used. The first one is a very accurate but slow method and the second is faster but in general less accurate.
(see [1, `https://eigen.tuxfamily.org/dox/group__TutorialLinearAlgebra.html`])

7.3 Conclusion and Other Approaches

The VMS method successfully stabilized the POD model and reduced non-physical oscillations. The numerical tests here and the tests in [16] correspond. It was possible to reduce the dimension of the equations significantly. The method shows a low sensitivity with respect to the diffusion coefficient. The error estimation in [16] is not appropriate for an a priori error bound.

In [12], a ROM method for convection-dominant *convection-diffusion-reaction equation* is also described. This is a SUPG-ROM method. In both [12] and [16] the same example was used for numerical analysis. It is visible that the results are quite comparable, but it would be necessary to collect the exact error data for the same r again to make any conclusive statements on this. In [19] one finds an error estimate for a SUPG-ROM method, which is much more coherent than the error estimate in [16]. In [28], a ROM method with VMS approach is also described, but without error estimates. The approach is more general and not related to the VMS method in [18].

Appendix

This is the C++ code for the VMS-POD model with quadratic FEs with $N = 500$, $M = 250$, $r = 40$, $R = 20$, $\varepsilon = 10^{-10}$ and $\varepsilon_+ = 4h = 8M$. If necessary, one can change them in the code. It outputs the average error and a csv file with the result matrix. The hadder files contain functions for the main, which do not directly concern the VMS-POD procedure. The full implementation can be found on: `https://github.com/ClaraDaisyEmma/VMS-POD-Master-thesis`.

```
#include "save-csv.h" //includes save(MatrixXd matrix,const std::string& name)
#include "example-problem-P2.h" //includes Snapshots(int numbNotes, int timesteps) and forcing_term(int dim_space, int dim_time, double epsylon)
#include "FE-P2-matrices.h" //includes B_0(int dim) and B_2(int dim)
#include "orthogonalisation.h" //includes resolved_ON(MatrixXd PODbasis,int R, MatrixXd B)

using namespace Eigen;
//This function does the proper orthogonal decomposition.
MatrixXd POD(MatrixXd Snapshots, int r, MatrixXd X){
    MatrixXd snapshotproduct=(Snapshots.transpose()*X*Snapshots), result_POD=MatrixXd::Zero(Snapshots.rows(),r);
    int dim_time= Snapshots.cols();
    double lamda;
    //Eigenvalue solver for selfadjoint matrices, provides eigenvalues in ascending order of size;
    SelfAdjointEigenSolver<MatrixXd> es(snapshotproduct);
    //creation of the modes:
    for(int k=0; k<r ; k++){
        lamda=(es.eigenvalues()(dim_time-1-k));
        if (lamda < 1e-12) { // Avoid division by zero
            std::cerr << "Warning: Small eigenvalue encountered for the POD modes!" << std::endl;
            continue;
        }
        result_POD(all,k)= 1/sqrt(lamda)*Snapshots*es.eigenvectors().col(dim_time-1-k);
    }
    return result_POD;
}

int main(){
    //Implementation:
    constexpr int M=250, N=500, r=40, R=20;
    int nodes=2*M-1;
    constexpr double Eps=1e-10, DeltaT=1.0/N ,Eps_plus=4.0/M;
    double error=0 ;
    MatrixXd VMS_POD_solution=MatrixXd::Zero(r, N+1), snapshots=MatrixXd::Zero(nodes,N+1),
             B0, B2, LHS, F, Nk, POD_modes, Rev_transformation, unresolved_scales, resolved_scales;
    VectorXd RHS, AV=VectorXd::Zero(nodes), AVRHS;

    if(R==0 || R>r){
```

```
37          std::cout << "Chose $r>R>0$"<< std::endl;
38          return EXIT_FAILURE;
39      }
40      B0=B_0(nodes); //matrix with H^1_0 inner product of the FE basis funktions
41      B2=B_2(nodes); //matrix with L^2 inner product of the FE basis funktions
42      snapshots=Snapshots(nodes, N); //matrix with snapshots as columns
43      Nk=N_k(nodes); //matrix with (b_i', b_j)_L^2, whith the FE basis funktions b_i, b_j
44      F=forcing_term(nodes,N, Eps); //matrix with the forcing term solutions f(t,x)
45      //Computing the mean value and subtract from the snapshots:
46      for(int k=0; k<N+1; k++){
47          AV=AV+snapshots(all,k);
48      }
49      AV=AV/((double)N +1); //averrage mean value of the full order solution
50      for(int k=0; k<N+1; k++){
51          snapshots(all, k)=snapshots(all,k)-AV;
52      }
53      POD_modes=POD(snapshots,r,(B0+B2)); //matrix with POD modes as columns
54      resolved_scales=resolved_ON(POD_modes,R,B0); //matrix of the ON basis of L^R
55      //Computing the artificial viscosity matrix:
56      unresolved_scales=POD_modes- (resolved_scales*(resolved_scales.transpose()*(B0)*
            POD_modes));
57      //Computing the startingpoint u^r_0
58      VMS_POD_solution(all,0)=POD_modes.transpose()*(B0+B2)*snapshots(all,0);
59
60      AVRHS=DeltaT*POD_modes.transpose()*(B2+Nk.transpose()+Eps*B0)*AV; //term on
            the RHS that is influenced by the AV
61      LHS = ((DeltaT/2)+1)*POD_modes.transpose()*B2*POD_modes + (DeltaT/2)*
            POD_modes.transpose()*Nk.transpose()*POD_modes+
62          (DeltaT/2)*Eps*POD_modes.transpose()*B0*POD_modes +(DeltaT/2)*Eps_plus*
                unresolved_scales.transpose()*B0*unresolved_scales;
63      RHS=LHS*VMS_POD_solution(all,0);
64      //Loop with linear systems to finde the solution:
65      for(int k=1; k<=N; k++){
66          RHS=POD_modes.transpose()*B2*((DeltaT/2)*(F(all,k-1)+F(all,k))+2*POD_modes
                *VMS_POD_solution(all,k-1))-AVRHS-RHS;
67          VMS_POD_solution(all,k)= LHS.fullPivHouseholderQr().solve(RHS);
68      }
69      //Transform back to finite element space:
70      Rev_transformation=POD_modes*VMS_POD_solution;
71      //Averrage L^2 error:
72      for(int k=0; k<=N; k++){
73          error=error+sqrt((snapshots(all,k)-Rev_transformation(all,k)).transpose()*B2*(
                snapshots(all,k)-Rev_transformation(all,k)));
74      }
75      error=error/((double)N+1);
76      std::cout<< "Average error: " << error << std::endl;
77      //Adding the meanvalue:
78      for(int k=0; k<N+1; k++){
79          Rev_transformation(all,k)=Rev_transformation(all,k)+AV;
80      }
81      const std::string dataName_results = "VMS-POD-P2-results.csv";
82      save(Rev_transformation, dataName_results); //save the results in a csv file
83
84      return EXIT_SUCCESS;
85  }
```

References

[1] Eigen 3.4.0. *https://eigen.tuxfamily.org*, 18.08.2021.

[2] N. Ahmed, T. Chacon Rebollo, V. John, and S. Rubino. A review of variational multiscale methods for the simulation of turbulent incompressible flows. *Archives of Computational Methods in Engineering*, 24(1):115–164, 2017.

[3] H. W. Alt. *Linear functional analysis*. Springer, 2016.

[4] D. Braess. *Finite Elemente: Theorie, schnelle Löser und Anwendungen in der Elastizitätstheorie*. Springer-Verlag, 2013.

[5] A. Chatterjee. An introduction to the proper orthogonal decomposition. *Current science*, pages 808–817, 2000.

[6] R. Codina. On stabilized finite element methods for linear systems of convection-diffusion-reaction equations. *Computer Methods in Applied Mechanics and Engineering*, 188(1-3):61–82, 2000.

[7] L. Cordier and M. Bergmann. Post-Processing of experimental and numerical data. *Proper Orthogonal Decomposition: An Overview, von Karman Institute for Fluid Dynamics, Lecture Series*, 3:2003, 2003.

[8] S. L. Eskew and J. R. Singler. A new approach to proper orthogonal decomposition with difference quotients. *Advances in Computational Mathematics*, 49(2):13, 2023.

[9] B. García-Archilla, V. John, and J. Novo. On the convergence order of the finite element error in the kinetic energy for high Reynolds number incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 385:114032, 2021.

[10] B. García-Archilla, V. John, and J. Novo. POD-ROMs for incompressible flows including snapshots of the temporal derivative of the full order solution. *SIAM Journal on Numerical Analysis*, 61(3):1340–1368, 2023.

[11] B. García-Archilla and J. Novo. Pointwise error bounds in POD methods without difference quotients. *arXiv preprint arXiv:2407.17159*, 2024.

[12] S. Giere, T. Iliescu, V. John, and D. Wells. SUPG reduced order models for convection-dominated convection-diffusion-reaction equations. *Computer Methods in Applied Mechanics and Engineering*, 289:454–474, 2015.

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

[14] T. J. Hughes. Multiscale phenomena: Green's functions, the Dirichlet-to-Neumann formulation, subgrid scale models, bubbles and the origins of stabilized methods. *Computer methods in applied mechanics and engineering*, 127(1-4):387–401, 1995.

[15] T. J. Hughes and J. R. Stewart. A space-time formulation for multiscale phenomena. *Journal of Computational and Applied Mathematics*, 74(1-2):217–229, 1996.

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

[17] T. Iliescu and Z. Wang. Are the snapshot difference quotients needed in the proper orthogonal decomposition? *SIAM Journal on Scientific Computing*, 36(3):A1221–A1250, 2014.

[18] V. John, S. Kaya, and W. Layton. A two-level variational multiscale method for convection-dominated convection-diffusion equations. *Computer Methods in Applied Mechanics and Engineering*, 195(33-36):4594–4603, 2006.

[19] V. John, B. Moreau, and J. Novo. Error analysis of a SUPG-stabilized POD-ROM method for convection-diffusion-reaction equations. *Computers & Mathematics with Applications*, 122:48–60, 2022.

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

[21] R. Juanes and T. W. Patzek. Multiscale-stabilized finite element methods for miscible and immiscible flow in porous media. *Journal of Hydraulic Research*, 42(S1):131–140, 2004.

[22] B. Koc, S. Rubino, M. Schneier, J. Singler, and T. Iliescu. On optimal pointwise in time error bounds and difference quotients for the proper orthogonal. *SIAM journal on numerical analysis, 59 (4), 2163-2196.*, 2021.

[23] K. Kunisch and S. Volkwein. Control of the Burgers equation by a reduced-order approach using proper orthogonal decomposition. *Journal of optimization theory and applications*, 102(2):345–371, 1999.

[24] K. Kunisch and S. Volkwein. Galerkin proper orthogonal decomposition methods for parabolic problems. *Numerische mathematik*, 90:117–148, 2001.

[25] W. Layton. A connection between subgrid scale eddy viscosity and mixed methods. *Applied Mathematics and Computation*, 133(1):147–157, 2002.

[26] S. K. Locke and J. R. Singler. A new approach to proper orthogonal decomposition with difference quotients. *arXiv preprint arXiv:2106.10224*, 2021.

[27] A. Quateroni and A. Valli. *Numerical Approximation of Partial Differential Equations*. Springer, 2008.

[28] R. Reyes and R. Codina. Projection-based reduced order models for flow problems: A variational multiscale approach. *Computer Methods in Applied Mechanics and Engineering*, 363:112844, 2020.

[29] H.-G. Roos, M. Stynes, and L. Tobiska. *Robust numerical methods for singularly perturbed differential equations: convection-diffusion-reaction and flow problems*. Springer, 2008.

[30] D. Werner. *Funktionalanalysis*. Springer, 2018.