Master of Science Thesis

A semi-continuous approach to reduced-order modelling

S. Mattei

August 19, 2010

Aerodynamics

Faculty of Aerospace Engineering

TU Delft

Delft University of Technology

A semi-continuous approach to reduced-order modelling

Master of Science Thesis

For obtaining the degree of Master of Science in Aerospace Engineering at Delft University of Technology

S. Mattei

August 19, 2010

Faculty of Aerospace Engineering · Delft University of Technology

Delft University of Technology

DELFT UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF AERODYNAMICS

The undersigned hereby certify that they have read and recommend to the Faculty of Aerospace Engineering for acceptance the thesis entitled **"A semi-continuous approach to reduced-order modelling"** by **S. Mattei** in fulfillment of the requirements for the degree of **Master of Science**.

Dated: August 19, 2010

Supervisors:

Dr. S.J. Hulshoff

Prof. dr. ir. drs. H. Bijl

Dr. R.P. Dwight

Dr. ir. A.S.J. Suiker

ir. P.W. Fick

Table of Contents

Chapter 1

Introduction

The vast majority of problems encountered in physics and engineering are described by partial differential equations. Since solutions in closed form are available only for a very small subset of problems, many strategies have been developed to obtain numerical approximations to describe the problem in an efficient and reliable way. Great advances in the field of computer engineering, together with the development of refined numerical techniques, have allowed in recent years high fidelity simulations of complex systems. However, due to the enormous computational cost required by such calculations, the time required for high-fidelity simulations remains extremely high.

In this framework, the aim of reduced-order modelling is to find techniques to simulate very complicated systems in a more compact way. The main idea is to construct simpler models which capture the main features of the original complex system, avoiding the complete characterisation of all the states in the problem. In the literature several methods are available to construct Reduced-Order Models (ROM) either from numerical simulations, or experimental data sets. Often these methods rely upon projection frameworks, in which the governing equations are projected onto suitably chosen basis functions.

In this context a method that has been successfully used over the last decades to construct an optimal set of basis functions is the Proper Orthogonal Decomposition (POD). The POD, also known in the literature as Karhunen-Loéve decomposition, Principal Component Analysis (PCA) or Singular Value Decomposition (SVD), is a procedure for extracting a basis for a modal decomposition from an ensemble of reference data. Its mathematical properties illustrate that the error introduced by projecting a set of reference data in a reduced-space is minimised upon choosing the POD basis functions. This makes the POD

particularly attractive for the construction of reduced-order models, and has been successfully applied in many fields, e.g fluid dynamics and turbulence (1; 2; 3), structural vibrations (4), biology (5), meteorology (6), just to name a few. Its limitations, however, arise in the fact that the POD is purely data-driven and no explicit reference is made to the underlying governing equations. In other words, being the solution of a reduced-order model a different procedure than the projection of reference data in a reduced-space, we have no guarantee that the POD optimality holds in such circumstances. Furthermore, often we might not be interested in the full description of a system, but rather we would only like to know how our system responds to a specific input change, a different initial/boundary condition and so on. Targeting different outputs requires a re-definition of optimality and there is therefore a need to seek basis functions specifically built for accurate ROM-solutions.

In particular, a relevant approach is to seek reduced-order models to optimally represent a specific output functional, a class that falls under the name of Goal-Oriented Reduced-Order Model (GO-ROM). These models provide an accurate description of the quantity of interest using a minimum number of degrees of freedom, making them attractive for use in the design of control systems or as a part of a larger simulation.

Recently, Bui-Thanh et al. (7) proposed an optimisation method to generate a set of goal-oriented ROM-targeted basis functions, in which the satisfaction of the reduced-order model is imposed as an explicit constraint of the problem. This improves on a data-driven approach by bringing additional knowledge in the construction of the basis, allowing the modes to be specifically constructed to best represent the goal-oriented outputs. Their results showed a clear improvement upon the POD. Due to its fully discrete nature, we will refer to the approach employed by reference (7) as the Fully-Discrete Formulation (FDF).

Several issues, however, limit its applicability. First of all, the method proposed has a fully-discrete character and its implementation requires as input matrices with the dimension of the reference data, which represent in fact a discretisation of the governing equations. If the reference data comes from CFD simulations one might be able to construct these matrices from the CFD method employed to generate the data, but this is not always possible since in some circumstances, numerical schemes do not have the required form. Even when possible, the choice of the matrices used by the original CFD method are not necessarily optimal, as their characteristics influence the final behaviour of the ROM

which might have to operate under quite different requirements. In case that the reference data comes from experiments, no obvious choice exists, yet the implications for the find performance of the ROM remain. Furthermore, the derivation proposed is limited to linear PDE's and linear functionals, restricting considerably the span of applicability.

To overcome these drawbacks, we consider the definition of the reduced-order model, as well as the optimisation technique, in a continuous setting. The reference data, on the other hand, being generated by numerical simulations or experiments, is always defined in a discrete space. In addition, the most straight-forward option is to consider the ROM basis functions to be defined in the same discrete space as that of the reference data. We will therefore refer to the combined approach considered in this work as the Semi-Continuous Formulation (SCF). Using the SCF we can accommodate any differential equation and we also avoid the ambiguities associated with choosing the input matrices arbitrarily at the beginning of the optimisation process. Furthermore, we clarify the treatment of arbitrary functionals in the optimisation process and are no longer limited to be in a linear form, but can now be any function.

The structure of this report is organised as follows: In chapter 2 we give a definition of the POD, highlight its optimality conditions and give a recipe to construct reduced-order model based on the POD. Chapter 3 is devoted to the presentation of discrete method of ref. (7) and the optimisation algorithm. We will describe how the goal-orientation can lead to improved ROM for specific functionals and derive a procedure to determine a set of optimal basis functions. Chapter 4 exemplifies the benefits of moving to the semi-continuous setting, describing its implementation and giving several results. Finally, we give our conclusions and recommendations for future work.

Chapter 2

Reduced-Order Modelling using the POD

2.1 Motivation

One strategy to construct reduced-order models is to rely upon projection frameworks, in which the underlying governing equations are projected onto suitably-chosen basis functions. When dealing with fluid dynamics, we are often required to solve problems in a 4-dimensional world, accounting for three spatial coordinates and a temporal one. In such cases it is handy to expand the solution, here referred as y, as a finite series in the separated-variables form:

$$y(x,t) \approx \sum_{i=1}^{M} a_i(t)\phi_i(x), \tag{2.1}$$

with the hypothesis that as M approaches infinity our solution becomes exact. While in equation (2.1) there is no fundamental difference between t and x, we refer x as a space coordinate, usually a vector, and t as time.

The representation of equation (2.1) is not unique, in fact $\phi_i(x)$ can be chosen for instance as a Fourier series, Legendre polynomials, Chebyshev polynomials and so on. Recalling that we are interested in a finite-series expansion, which rapidly approaches the exact solution using a small number of terms, we may like to determine $\phi_i(x)$ such that the approximation for each M is optimal in a L^2-norm sense. That means that we will try to find a sequence of $\phi_i(x)$ in which the first three terms of the expansion are the best possible three-term approximation to the exact solution, the first seven terms the best seven-term approximation and so on. In addition, to simplify the calculations, it would be convenient to have an orthonormal basis since the formula for a vector space projection is

much simpler.

To summarise, one approach to construct a ROM is to use a set of orthonormal basis functions which can best represent the solution in a L^2-norm measure. Such basis functions can be found using a POD.

In this chapter we describe the procedure to determine the POD modes and developing a ROM based on them. We will explain in what terms the POD modes are optimal and give a recipe to construct POD based ROM.

2.2 Discrete POD

As mentioned in the preamble of this chapter, the idea underlying the POD is to determine $\phi_i(x)$ such that the approximation for each M is as good as possible in a L^2-norm sense. In this context, we can think to the POD as an optimisation problem for determining the basis functions such that the projected equations are as close as possible to the original system. For convenience we will omit indication of the space and time dependence of y, ϕ and a throughout this section. However keep in mind that a is only time-dependent, and ϕ is only space-dependent. Furthermore we will use the notation $\tilde{y}$ for the reduced-order solution.

Our goal is to express the solution as an approximation in the form:

$$y = \sum_{i=1}^{M} a_i \phi_i, \tag{2.2}$$

with $y \in \mathbb{R}^N$.
We require ϕ_i to be a set of orthonormal basis functions, (see appendix A.4), such that $\phi_i \cdot \phi_j = \delta_{ij}$ or equivalently in the finite dimensional case $\phi_j^T \phi_i = \delta_{ij}$, where δ_{ij} represents the Kronecker delta. In this case, we notice the following: if we pre-multiply each side of the equation (2.2) by ϕ_j^T we have:

$$\phi_j^T y = \phi_j^T \sum_{i=1}^{M} a_i \phi_i \tag{2.3}$$

since a_i is simply a coefficient, we can plug in ϕ_j^T, obtaining:

$$\phi_j^T y = \sum_{i=1}^{M} a_i \phi_j^T \phi_i \tag{2.4}$$

by orthogonality all terms with $i \neq j$ vanish, leading to:

$$\phi_j^T y = a_j \tag{2.5}$$

which shows the important result that for orthonormal basis functions, the determination of the coefficient function a_j depends only on ϕ_j and not on the other ϕ's.
Substituting this result into our expansion, the approximation in the basis representation becomes:

$$\tilde{y} = \sum_{i=1}^{M} (y^T \phi_i)\phi_i, \tag{2.6}$$

in which we have rewritten $\phi_j^T y$ as $y^T \phi_j$.

When dealing with data from numerical simulations, or experiments, we usually do not have a single vector, but rather a set of vectors y_j containing the state of the system at each time step. In this case our series expansion can be written as:

$$\sum_{j=1}^{L} \tilde{y_j} = \sum_{j=1}^{L} \sum_{i=1}^{M} (y_j^T \phi_i)\phi_i \tag{2.7}$$

Our goal is to minimise the difference between the original system and the projected equations in the L^2-norm sense. Therefore we seek a set of orthonormal vectors $\Phi \in \mathbb{R}^{N \times M}$ such that:

$$\Phi = \arg\min_{\Phi} E(\Phi) \tag{2.8}$$

with the error measure:

$$E(\Phi) = \sum_{j=1}^{L} \left\| y_j - \sum_{i=1}^{M} (y_j^T \phi_i)\phi_i \right\|^2 \tag{2.9}$$

where $\|y\| = \sqrt{y^T y}$.
This is a so-called constrained minimisation problem, in which the error E must be minimised subject to constraints on the possible values of the independent variables. In our case this means minimising E with the constraint that the ϕ's are orthonormal.

$$\min_{\Phi} E(\phi_1, ..., \phi_M) \quad \text{subject to:} \quad \phi_j^T \phi_i = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{2.10}$$

The most frequently used method for constrained problems is to employ Lagrange multipliers (8). This entails the construction of a Lagrange functional containing the original functional plus the constraints expressed in the form $g(\cdot,\cdot) = 0$, each multiplied by a Lagrange multiplier λ_{ij}.
For our specific case this results in:

$$L(\phi_1, ..., \phi_M, \lambda_{11}, ..., \lambda_{MM}) = E(\phi_1, ..., \phi_M) + \sum_{i,j=1}^{M} \lambda_{ij}(\phi_j^T \phi_i - \delta_{ij}) \tag{2.11}$$

According to the classical theory of minimisation, a necessary condition for the existence of an extrema is that the first partial derivatives of the function to be optimised are zero at the extrema.
Therefore a minimum can be found by differentiating L with respect to the independent variables ϕ_i and λ_{ij} and setting their derivatives equal to zero.

$$\frac{\partial L(\phi_1, ..., \phi_M, \lambda_{11}, ..., \lambda_{MM})}{\partial \phi_i} = 0 \qquad \text{for } i = 1..M \tag{2.12}$$

$$\frac{\partial L(\phi_1, ..., \phi_M, \lambda_{11}, ..., \lambda_{MM})}{\partial \lambda_{ij}} = 0 \qquad \text{for } i, j = 1..M \tag{2.13}$$

From this it follows that:

$$\frac{\partial L}{\partial \phi_i} = \sum_{j=1}^{L} y_j(y_j^T \phi_i) = \lambda_{ii}\phi_i \quad \text{and } \lambda_{ij} = 0 \quad \text{for } i \neq j \tag{2.14}$$

$$\frac{\partial L}{\partial \lambda_{ij}} = \phi_i^T \phi_j = \delta_{ij} \tag{2.15}$$

Arranging y in matrix form, such that each y_j is a column of the matrix Y:

$$Y = \begin{pmatrix} \vdots & \vdots & & \vdots \\ y_1 & y_2 & \cdots & y_L \\ \vdots & \vdots & & \vdots \end{pmatrix}$$

and setting $\lambda_{ii} = \lambda_i$, we have:

$$YY^T \phi_i = \lambda_i \phi_i \tag{2.16}$$

in which the matrix $YY^T \in \mathbb{R}^{N\times N}$ implying that the minimisation conditions can be expressed by the $N \times N$ eigenvalue problem.

Solving equation (2.16) for ϕ_i and λ_i is equivalent to performing a singular value decomposition. As shown in appendix B we can decompose the matrix Y such that

$$U^T Y V = \Sigma \tag{2.17}$$

Furthermore, since the matrices V and U are orthogonal, their transposes are equivalent to their respective inverses. This results in the following two equations:

$$YV = U\Sigma \tag{2.18}$$
$$Y^T U = V\Sigma \tag{2.19}$$

For any $0 < i < M$ this can be rewritten as;

$$Yv_i = \sigma_i u_i \tag{2.20}$$
$$Y^T u_i = \sigma_i v_i \tag{2.21}$$

Multiplying equation (2.21) by Y and then using (2.20) we have:

$$YY^T u_i = Y\sigma_i v_i = \sigma_i^2 u_i \tag{2.22}$$

Comparing (2.22) with (2.16) we see that $\phi_i = u_i$ and $\lambda_i = \sigma_i^2$. This means that, in a *L^2-norm measure*, the modes derived with the SVD procedure represent an optimal set of basis functions, implying that any reduced order approximation is the best possible approximation of that rank in a L^2-norm sense.

2.3 POD-ROM

In order to generate the reference data on which the ROM will be based, we solve the model equations using a finite difference method on a relatively fine mesh. This results in a N-by-L matrix Y in which the columns represent the state of the solution at a given time step. There are a number of ways to construct ROM. Perhaps the most natural way is by

variational projection. However, to facilitate comparison with results in the next chapter, we will generate the ROM with a method similar to that used by ref. (7).

The procedure for the construction of the reduced-order models can be summarised in the following lines:

1. Construct the POD basis by taking the SVD of the data matrix Y. The decomposition leads to $Y = U\Sigma V^T$ in which U is the set of basis functions.

2. Set the dimension of the reduced-order model by choosing the first M basis vectors of the matrix U. This results in the matrix $\Phi \in \mathbb{R}^{N\times M}$.

3. Substitute $u = \sum_{j=1}^{M} \alpha_j \phi_j = \Phi\alpha$ into the governing equation, and project in a reduced space by premultiplying the equation by Φ^T. This leads to the a differential equation in the α unknowns. The solution $\alpha = \alpha(t)$ represents the modal amplitudes as function of time and $\sum_{j=1}^{M} \alpha_j \phi_j$ is the solution of the reduced-order model.

In order to assess the described procedures, we carried out a preliminary analysis using three linear one-dimensional equations: the heat equation, the pure convection equation and the convection-diffusion equation.

2.3.1 Heat Equation

The heat equation, also known as diffusion equation, is a linear parabolic partial differential equation that describes the distribution of heat (or variation in temperature) in a given region over time. We investigate the temperature distribution in a constant area rod of unitary length. The heat equation reads as:

$$\frac{\partial u}{\partial t} - k\frac{\partial^2 u}{\partial x^2} = 0 \tag{2.23}$$

where u is the temperature in the rod, $k > 0$ is the diffusion coefficient and where we have neglected any source or sinks. To solve this equation a boundary condition is needed at each end of the rod, plus an initial condition describing the temperature distribution in the rod at $t = 0$. We have analysed the prescribed boundary temperature case (Dirichlet boundary conditions), in which we have set $u(0) = 1$ and $u(1) = 0$. The heat equation allows steady state solutions, and in our particular case it is represented by a straight line connecting the boundary conditions, namely $u = 1 - x$ for $0 \leq x \leq 1$. As initial condition we chose a Fourier sine series of order 3 added to the steady state solution (Fig. 2.1).

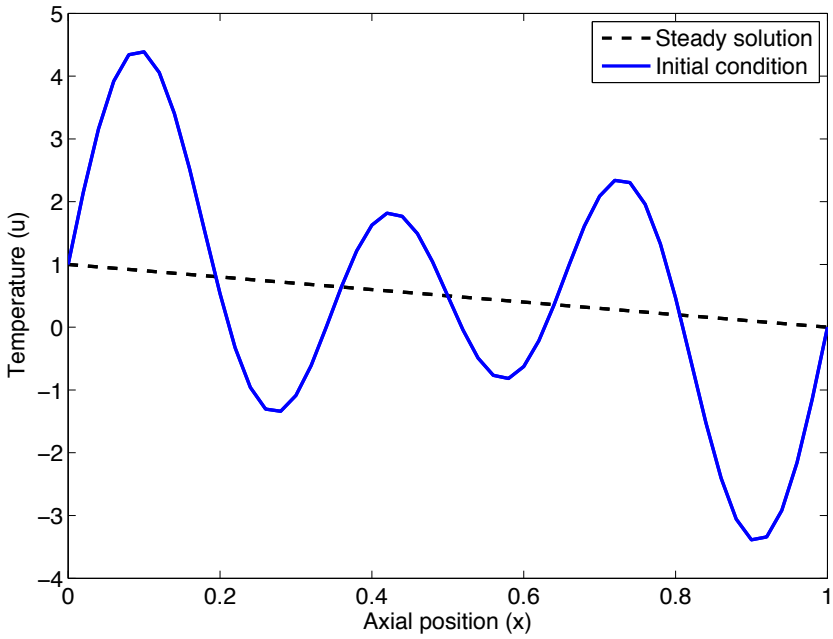


Figure 2.1: Initial temperature distribution

Reference data

We provide the reference data used to construct the ROM by solving the partial differential equation (2.23) numerically, using a finite difference technique. We use a uniform grid with N subintervals ($N+1$ nodes). The spatial derivative is central in space, and we integrate in time using a third-order Runge-Kutta explicit multistage method. The discretised equation can be written as:

$$\frac{du^n}{dt} - k\frac{u^n_{i-1} - 2u^n_i + u^n_{i+1}}{\Delta x^2} = 0 \tag{2.24}$$

where the superscript indicates the time step, and the subscript the spatial position.

The time integration is performed using a third order Runge-Kutta explicit method, where the new time level (u^{n+1}) is computed as a linear combination of solutions at intermediate stages. By writing $\dot{u} = F(u(t))$ the stages have the form:

$$
\begin{aligned}
u^{(1)} &= u^n, \\
u^{(2)} &= u^n + \Delta t F(u^{(1)}), \\
u^{(3)} &= u^n + \Delta t \left[\tfrac{1}{4}F(u^{(1)}) + \tfrac{1}{4}F(u^{(2)})\right], \\
u^{n+1} &= u^n + \Delta t \left[\tfrac{1}{6}F(u^{(1)}) + \tfrac{1}{6}F(u^{(2)}) + \tfrac{2}{3}F(u^{(3)})\right]
\end{aligned} \tag{2.25}
$$

POD-ROM results

The POD basis is constructed upon decomposing the matrix A by the SVD procedure. As we have seen in section 2.2 the vectors u_i represent the basis functions when the matrix A is organised taking as columns the state vectors for each time step.

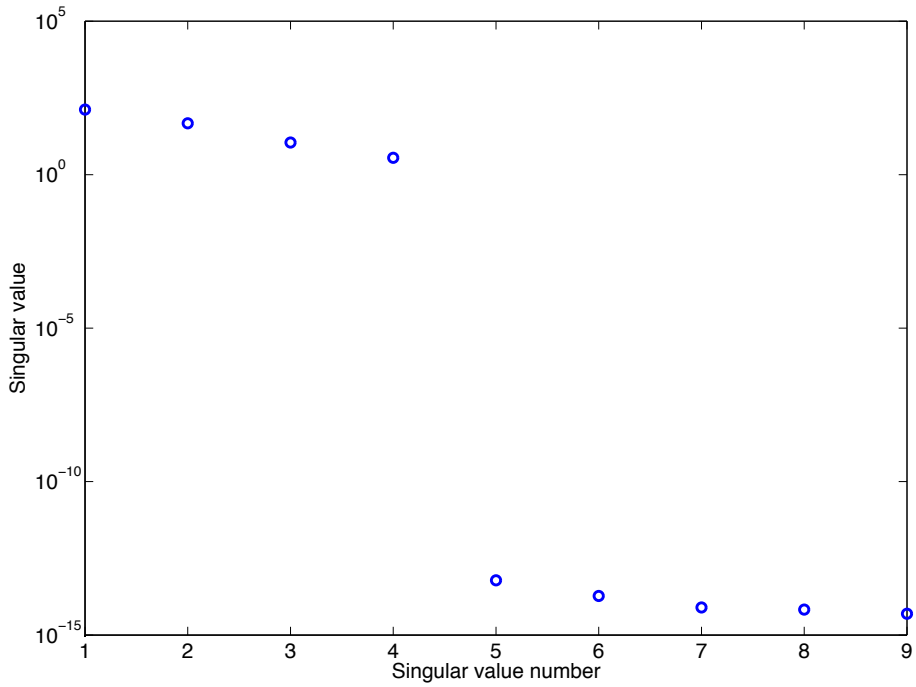


Figure 2.2: First nine singular values

The singular values in the matrix Σ represent the energy carried by each mode and in this simple case, over 99% of the energy was contained in the first four modes (Fig. 2.2). The associated modes presented the shape illustrated in Fig. 2.3:

To investigate the performance of the reduced-order model, we compare POD-ROM and reference data at a chosen instant of time (t=0.005 in our case) and at steady state.

The ROM equations are advanced in time using the Runge-Kutta time march described by (2.25). The reduced-order model is very close to the numerical solution, with the difference being practically unrecognisable when four modes are used. For this particular case, we are able to reproduce the solution while reducing the size of the problem from 50 (our chosen N) to 4. For the final solution (Fig. 2.5) we see a fairly good agreement with 2 and 3 modes, and a perfect match with 4 modes. The final solution time is taken as the time required by the numerical solution to converge to a given tolerance (t=0.0843).

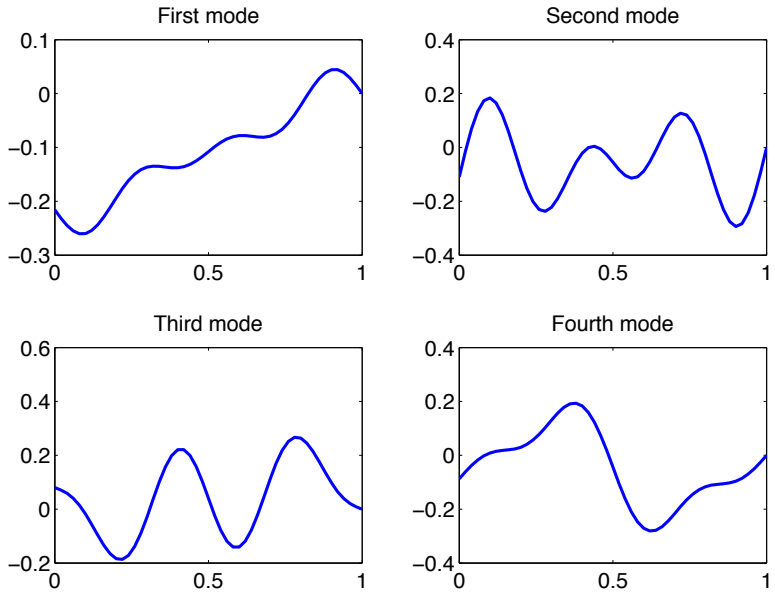


Figure 2.3: Modes shape

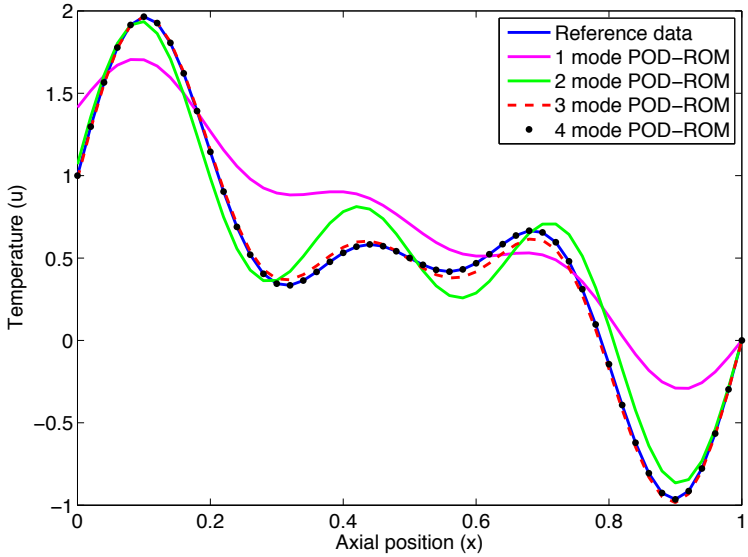


Figure 2.4: Comparison of reference data with ROM at t=0.005

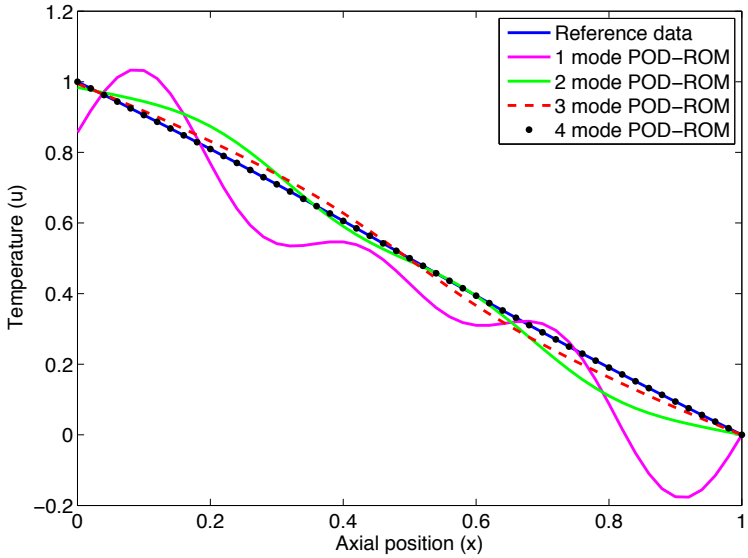


Figure 2.5: Comparison of reference data with ROM at the final solution time (t=0.0843)

2.3.2 Pure convection

The second example analysed is the linear convection case. The linear convection equation, a hyperbolic partial differential equation describing wave propagation in a medium, can be written as:

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0 \tag{2.26}$$

where u represents the transported quantity and a is the convection speed. We consider the transport of a wave as shown in Fig. 2.6 with speed $a = 1$ (right running wave) and periodic boundary conditions, running in time for a full cycle.

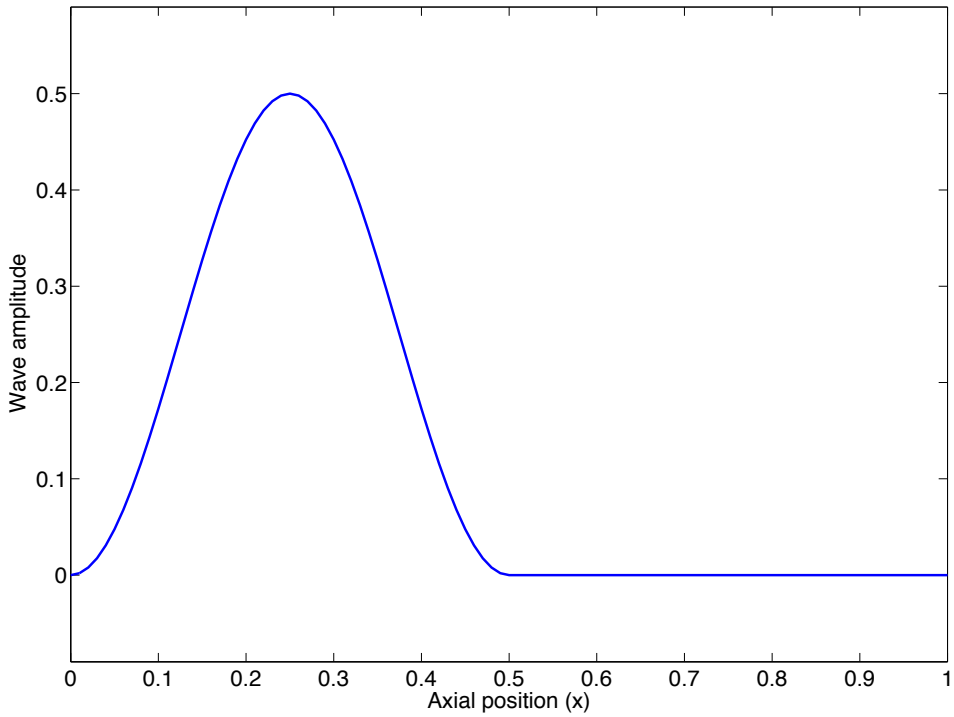


Figure 2.6: Wave form (initial condition)

Reference data

We follow the same procedure as for the heat equation, namely discretising the partial differential equation central in space and integrating in time with a third order Runge-

Kutta explicit method. The semi-discretised equation is:

$$\frac{du^n}{dt} + a\frac{u^n_{i+1} - u^n_{i-1}}{2\Delta x} = 0 \tag{2.27}$$

POD-ROM results

In the case of the heat equation, we have seen that four modes contained over 99% of the total energy. Here things are a bit more complicated. Physically now we are trying to track the movement of a wave only using space dependent modes, which is impossible in the case of a single mode and not trivial with very few modes. As we can see in the plot of the singular values (Fig. 2.7), in this case the energy is distributed over several more modes.

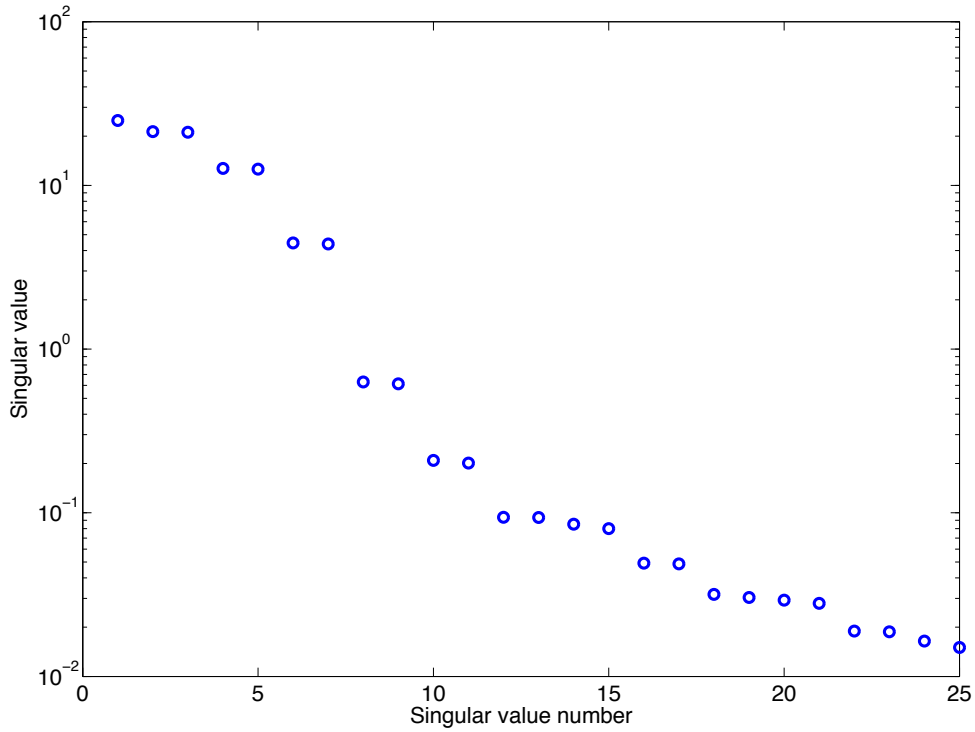


Figure 2.7: First twenty-five singular values

The singular values seem to be disposed in pairs (2-3, 4-5, ...) and their respective shapes reflect this observation. As can be seen in Fig. 2.8 mode 2 has the same shape as mode 3 but with a different phase. This can be ascribed to the energy transport phenomena. Since the modes are fixed in space, one mode alone is not able to carry the energy to a different location. Therefore the energy transport takes place using pairs of modes. The

exception is the first mode, which is constant and is responsible for representing the average of the wave in the domain. If the wave shape has average zero, this first mode disappears and the pairs are formed 1-2, 3-4 and so on. It is worth mentioning that the construction of the POD modes could have been performed by using a centred reference data, in which the average of the solution is subtracted from the snapshots. The expansion would then be written as: $u = \bar{u} + \sum_{j=1}^{M} a_j \phi_j$, with $\bar{u}$ the average solution. This would allow the elimination of one mode, in case the POD-ROM would be used to reproduce the reference data.

We compare the reference data with different number of modes in a chosen instant of time (t=0.5). As anticipated, the solution improves when a pair of modes is added to the reduced-order model and it is practically unnoticeable when a single one is added. The difference between three and four modes is basically unnoticeable, but when the fifth mode is added to the solution (forming the pair with number four), the solution rapidly improves. The same is noted for the other pairs

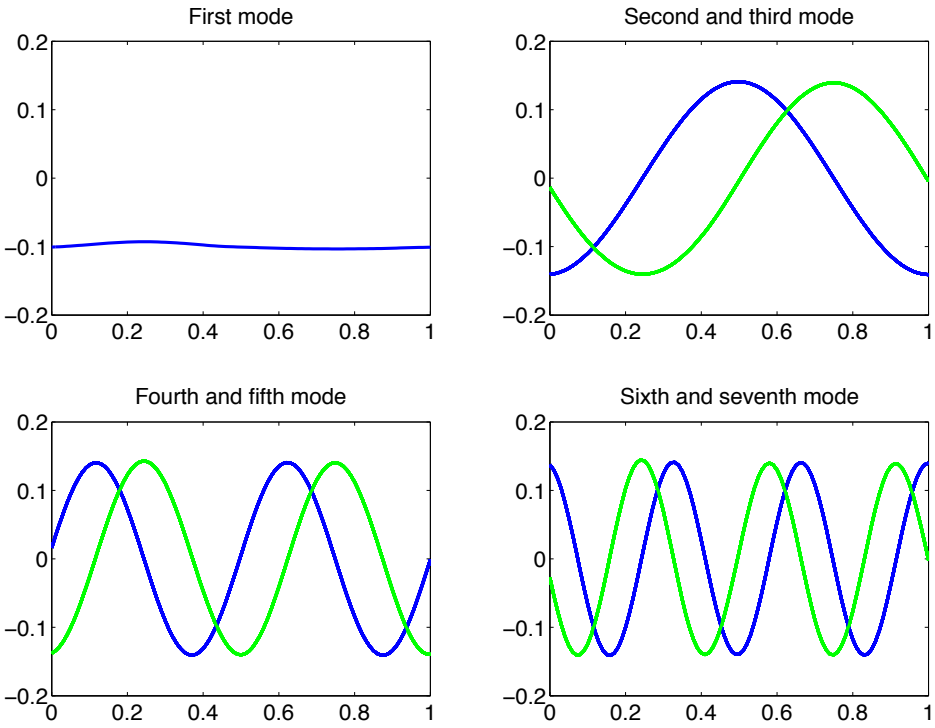


Figure 2.8: Pure convection modes shape

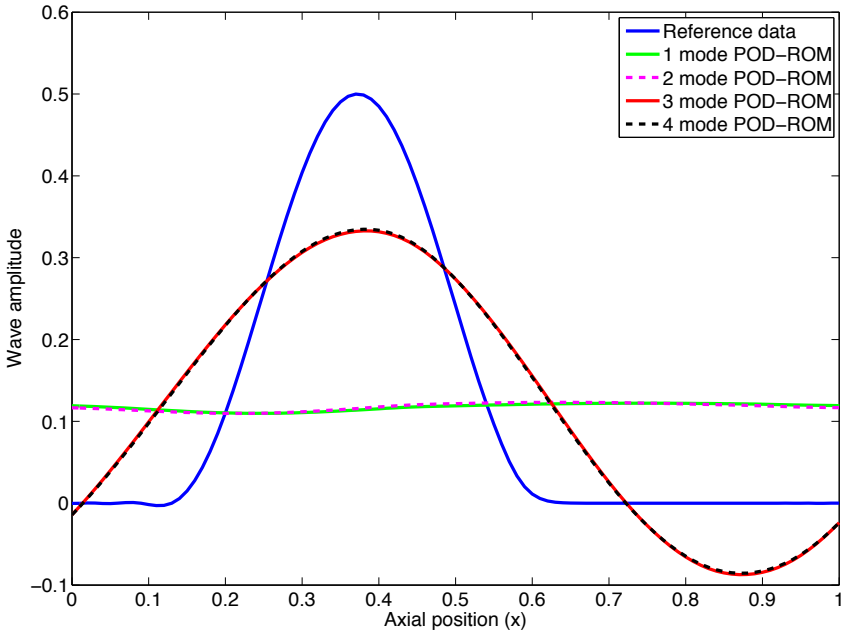


Figure 2.9: Comparison of the reference data with POD-ROM results for lower number of modes

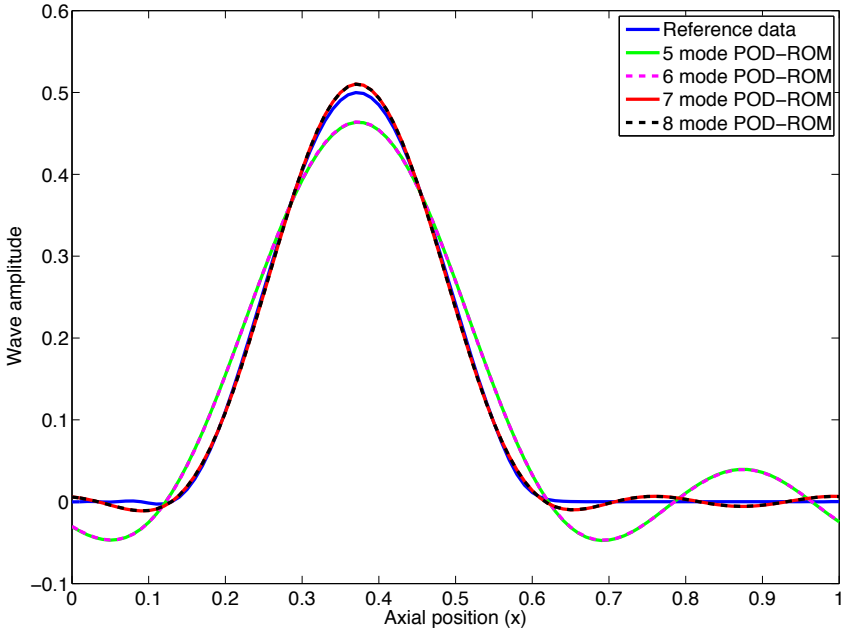


Figure 2.10: Comparison of the reference data with POD-ROM results for higher number of modes

2.3.3 Convection Diffusion

The convection-diffusion equation is often used as a model for the incompressible Navier-Stokes equations, since both admit boundary layer development at the wall. The convection-diffusion equation reads as:

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = k\frac{\partial^2 u}{\partial x^2} \tag{2.28}$$

where u represents the flow velocity, a is the convection speed and k the diffusion constant. We investigated the Dirichlet boundary condition case, where we take $u(0) = 1$ and $u(1) = 0$. These conditions are physically representative in the case of wall flow, where the "left" boundary condition $u(0) = 1$ is the asymptotic velocity and the "right" boundary condition $u(1) = 0$ is the no-slip condition. As initial condition we take a linear velocity distribution, as in Fig. 2.11. The final solution depends on the convection and diffusion constant values. If $a >> k$ we will have a steep boundary layer development and if $k >> a$ the solution will tend towards a straight line connecting the boundary conditions. In Fig. 2.11 an intermediate case is shown ($a = 2$ and $k = 0.1$) with dominant convection. The development of the boundary layer at the wall is clearly visible.

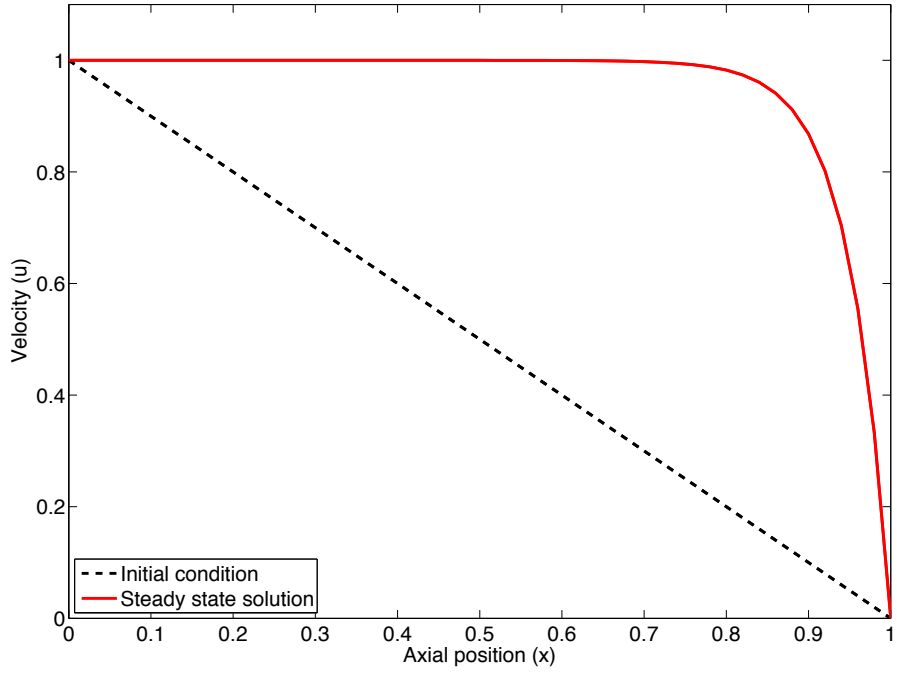


Figure 2.11: Initial condition and steady state solution

Reference data

Once again we use central finite differences in space, and integrate in time with a Runge-Kutta explicit method. The semi-discretised equation reads as:

$$\frac{du^n}{dt} + a\frac{u^n_{i+1} - u^n_{i-1}}{2\Delta x} = k\frac{u^n_{i+1} - 2u^n_i + u^n_{i-1}}{\Delta x^2} \tag{2.29}$$

POD-ROM results

We want to investigate the ability of the POD to represent such a solution. As before we analyse the distribution of energy on the singular values and the shape of the dominant modes. In this case the energy is regularly distributed over several modes, as Fig. 2.12 indicates.

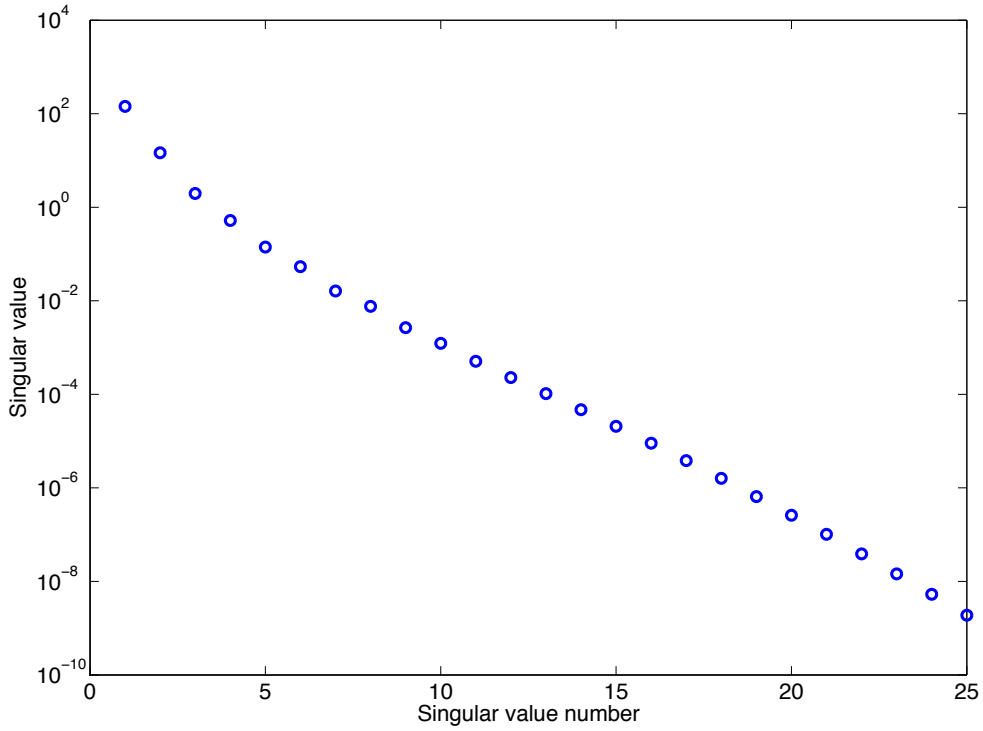


Figure 2.12: First twenty-five singular values

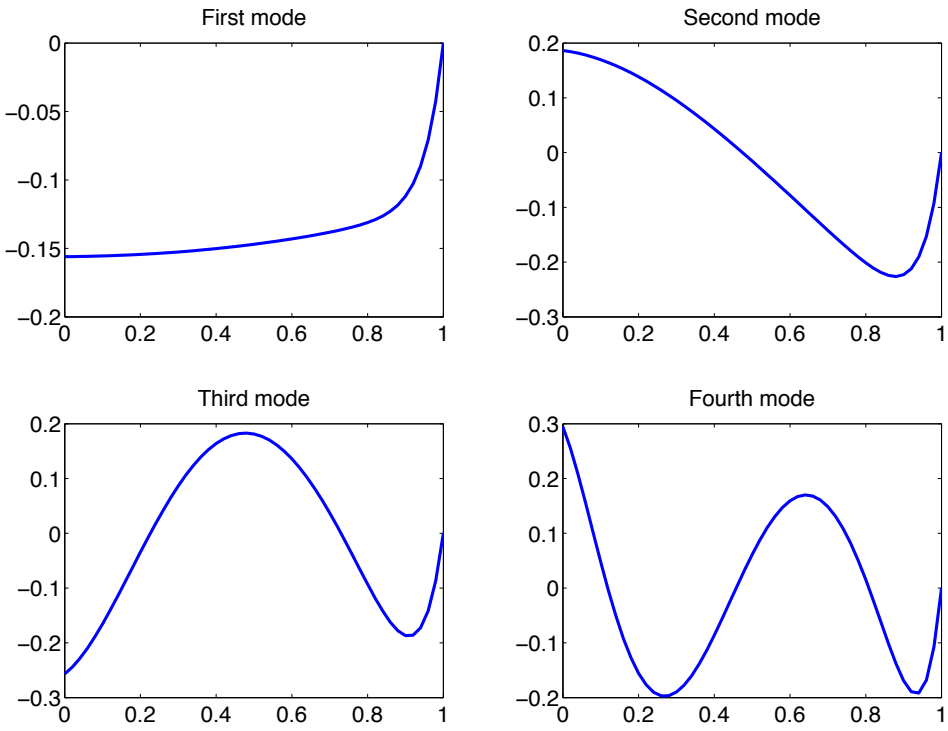


Figure 2.13: Convection-diffusion modes shape

In the convection-diffusion case we are mainly interested in the steady-state solution, since the boundary layer is then fully developed and it represents the real wall flow. In Fig. 2.14 we compare the steady state solution with reduced-order models of different order. Although the total energy is distributed over several modes, a very good representation can be obtained with only 3 modes, and with 4 the difference is almost unnoticeable.

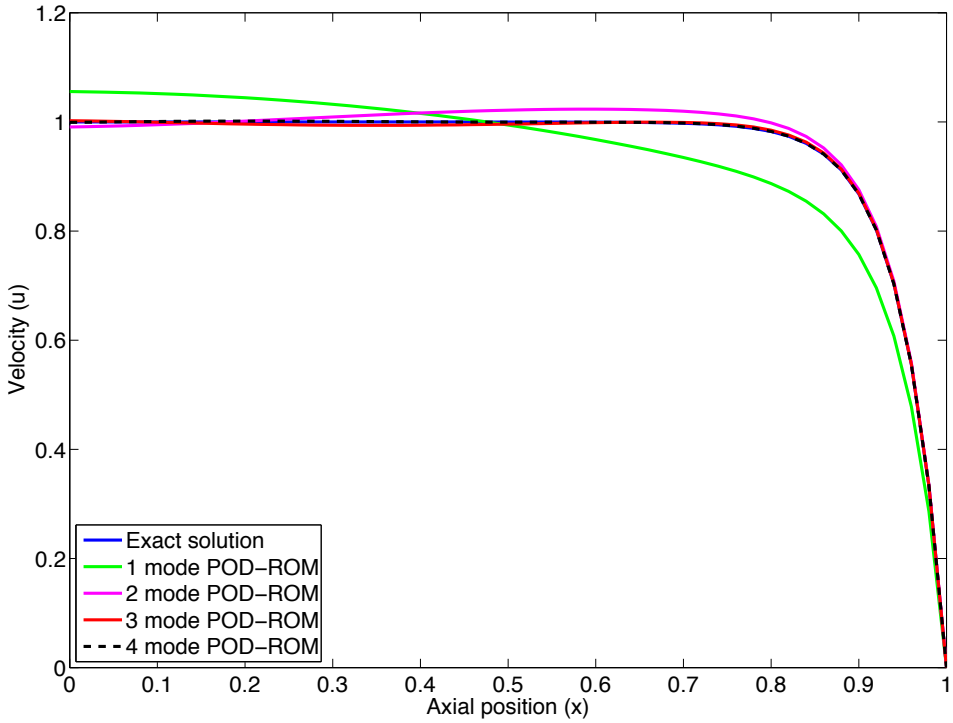


Figure 2.14: Steady state solution

Chapter 3

Goal-Oriented Model-Order Reduction

In the previous chapter we have seen how the POD provides a procedure for the reconstruction of the system states which can be considered optimal in a certain sense, since the error between the reduced space solution and the reference data (see equation 2.9) is minimised in the L^2-norm by choosing the POD basis functions. However, it is important to note that the POD is a purely data-driven procedure and no reference is made to the underlying governing equations. Therefore we have no guarantee that the same optimality holds for the output of reduced-order models constructed using the POD modes. In the following sections we will present a different method to derive basis functions for reduced-order models, rigorously tied to the ROM solution and in particular by targeting a specific output functional. This Goal-Oriented approach has been recently proposed by T. Bui-Thanh, K. Willcox, O. Ghattas, B. van Bloemen Waanders (7), We will refer to it as the Fully Discrete-Approach (FDF).

3.1 Overview of the method

The procedure of Bui-Thanh et al. starts from a discrete representation of the governing equations. For a general Linear Time-Invariant (LTI) dynamical system:

$$M\dot{u} + Ku = f \tag{3.1}$$

$$g = Cu \tag{3.2}$$

with initial condition:

$$u(0) = u_0$$

where $u(t) \in \mathbb{R}^N$ is the system state, $\dot{u}(t)$ is the time derivative of u. $f(t)$ defines the input to the system and the matrix $C \in \mathbb{R}^{Q \times N}$ defines the Q output of interest, which are contained in the output vector $g(t)$. $M \in \mathbb{R}^{N \times N}$ and $K \in \mathbb{R}^{N \times N}$ are often referred in the literature as *Mass matrix* and *Stiffness matrix* respectively. These can be derived by considering a particular discretisation scheme for the original partial differential equations. The choice of this discretisation scheme is not inconsequential, as we will see that it will ultimately be incorporated in the definition of the reduced-order model. Furthermore, it is not clear how one would choose M and K if one wanted to use reference data from experiments or unknown numerical methods. We will ignore these issues for the moment, however, and assume that a suitable discretisation (i.e finite difference or finite elements) can be defined.

A reduced-order model can be constructed by approximating the states u as:

$$\hat{u} = \Phi\alpha$$

in which we recognise $\Phi \in \mathbb{R}^{N \times M}$ to be the projection matrix containing as columns the basis vectors and where $M << N$. $\hat{u}$ is the reduced model approximation of the state u, and the vector $\alpha(t)$ contains the corresponding modal amplitudes. Upon substituting this expression into the governing equations, and taking the Galerkin projection by premultiplying by Φ^T we have:

$$\hat{M}\dot{\alpha} + \hat{K}\alpha = \hat{f} \tag{3.3}$$

$$\hat{g} = \hat{C}\alpha \tag{3.4}$$

$$\hat{M}\alpha_0 = \Phi^T M u_0 \tag{3.5}$$

where $\hat{M} = \Phi^T M \Phi, \hat{K} = \Phi^T K \Phi, \hat{f} = \Phi^T f, \hat{C} = C\Phi$ and $\alpha_0 = \alpha(0)$.
Expressions (3.3-3.5) represent the reduced-order model in the α unknowns of the LTI system (3.1).

We now pose the problem of determining the set of basis functions Φ, such that the goal functional predicted by the reduced-order model $\hat{g}$ is as close as possible to the reference one. Before proceeding with the derivation, it is important to note that this procedure

differs from the derivation for the POD since we are now considering errors in the solution of the ROM, $\hat{u}$, rather than those of the state projection in the reduced space ($\tilde{u}$). In other words $\hat{u} \neq \tilde{u}$, since $\hat{u}$ is the solution of the reduced-order model (3.3-3.5), while $\tilde{u}$ is the representation of a known state in the reduced space, namely $\tilde{u} = \Phi\Phi^T u$, or equivalently $\tilde{u} = \sum_{i=1}^{M}(u^T\phi_i)\phi_i$. Furthermore, we will consider more general definitions for the error in $\hat{u}$, namely $\hat{g}$. This brings additional knowledge into the construction of the basis, and therefore ensures reliability of the reduction technique.

Once again we can formulate the problem in an optimisation framework in which we seek an optimal set of orthonormal basis functions, but now with a more general error measure:

$$\Phi = \arg\min_{\Phi} E(\Phi) \tag{3.6}$$

$$E(\Phi) = \int_0^{t_f} (g-\hat{g})^T(g-\hat{g})dt \tag{3.7}$$

$$\tag{3.8}$$

which is the L^2-norm error of the output functional with respect to the reference data. By simultaneously imposing orthonormality in the basis functions and the satisfaction of the reduced-order model, the constrained error minimisation problem can be written:

$$\min_{\Phi,\alpha} \mathscr{G} = \frac{1}{2}\int_0^{t_f} (g-\hat{g})^T(g-\hat{g})dt + \frac{\beta}{2}\sum_{j=1}^{m}(1-\phi_j^T\phi_j)^2 + \frac{\beta}{2}\sum_{\substack{i,j=1 \\ i\neq j}}^{m}(\phi_i^T\phi_j)^2 \tag{3.9}$$

$$\text{subject to: } \Phi^T M\Phi\dot{\alpha} + \Phi^T K\Phi\alpha = \Phi^T f \tag{3.10}$$

$$\Phi^T M\Phi\alpha_0 = \Phi^T M u_0 \tag{3.11}$$

$$\hat{g} = C\Phi\alpha \tag{3.12}$$

In case of linear relationship between outputs and states the objective function, this can be rewritten as:

$$\min_{\Phi,\alpha} \mathscr{G} = \frac{1}{2}\int_0^{t_f} (u-\hat{u})^T H(u-\hat{u})dt + \frac{\beta}{2}\sum_{j=1}^{m}(1-\phi_j^T\phi_j)^2 + \frac{\beta}{2}\sum_{\substack{i,j=1 \\ i\neq j}}^{m}(\phi_i^T\phi_j)^2 \tag{3.13}$$

where $H = C^T C$ can be interpreted as a weighting matrix that defines the states relevant to the specified output. In equation (3.13) we notice the following: the first term is similar to that encountered in the minimisation process we used for determining the POD. However the matrix H, now focuses the reduction of the error on the specific functional. If $H = 1$ or equivalently if H is the identity matrix I, this procedure would tend to optimise the same error as the POD, although the results is still not equivalent since $(\hat{u} \neq \tilde{u})$.

Before describing the details of the optimisation algorithm, it is important to notice that the optimisation problem (3.9-3.12) is non-linear and nonconvex. This has several implications. First of all, we have no guarantees that a purely local optimisation will converge to the global optimum, since algorithms usually tend to be "trapped" at local minima. The choice of the initial guess will play a crucial role in the behaviour of the algorithm. Furthermore, the size and complexity of the problem indicate that general search routines are likely to fail without specifically-targeted modifications. It is therefore crucial to employ a robust and reliable optimisation algorithm, as described in the next section.

3.2 Optimisation algorithm

This section is devoted to the description of the algorithm we used to solve the optimisation problem for the determination of the reduced-order model basis function. We will present the methodology for an unconstrained optimisation problem, illustrating latterly ways to handle the additional constraints of the problem.

3.2.1 Line search and trust-region

Most optimisation algorithms are iterative in nature. This means that beginning at an initial guess x_0, algorithms try to generate a sequence of iterates x_k that terminate when either no more progress can be made or when it seems that a solution point has been approximated with sufficient accuracy. Although algorithms that do not require monotone descent have been successfully implemented (see for instance (9)), usually one requires that the value of the function at the next iteration has to be lower than at the current one, hence $f(x_{k+1}) < f(x_k)$. In deciding how to move efficiently from one iterate to the next, one generally applies a so-called globalisation strategy. This can be viewed as an

enhancement which increases the set of initial guesses for which the method will converge. There are mainly two strategies we can follow, *line search* and *trust region* algorithms. Generally, both ones approximate the objective function around the current iterate by a quadratic model m_k:

$$m_k(x_k + p) = f_k + p^T \nabla f_k + \frac{1}{2} p^T \nabla^2 f_k p \tag{3.14}$$

where $f_k = f(x_k)$, p is a candidate step, ∇f_k is the gradient of the objective function at the current iterate x_k and $\nabla^2 f_k$ is the Hessian matrix (or when not readily available, some approximation of it). As we will see however, line search and trust region algorithms use this information in different ways to generate the next step.

In the line search method, the algorithm chooses a direction d_k and searches along this direction for a new iterate with a lower function value. Given a direction, the step length is calculated such that we (approximately) minimise the function along this direction. In other words, given d_k we choose α such that:

$$\min_{\alpha > 0} \quad m_k(x_k + \alpha d_k) \tag{3.15}$$

If we could solve (3.15) exactly, we would derive the maximum benefit from the direction d_k. The exact procedure is usually expensive, however, and most of the time it is enough to move closer to the minimum. Therefore line search algorithms generate a limited number of trial step lengths until one that loosely approximates the solution of the minimisation subproblem (3.15) is found. At the new point, a new search direction and step length are computed, and the process is repeated.

In contrast, trust region algorithms define a region around the current iterate in which it is understood that the model function is a good approximation of the objective function. Since the model might only be accurate in a close neighbourhood of the current iterate, we restrict the search for a minimiser of m_k to a limited region around x_k, the so-called trust region. In other words, we find the candidate step p by approximately solving the following subproblem:

$$\min_p m_k(x_k + p) \quad \text{where } x_k + p \text{ lies inside the trust region.} \tag{3.16}$$

Contrary to the line search method, here we impose a maximum step length and then minimise the model function subject to this constraint, which in turn can imply a different

search direction. If the candidate solution does not produce a sufficient decrease in f, we conclude that the trust region is too large, and therefore we shrink it and re-solve (3.16). Usually, the trust region is a ball defined by $||p||_2 \leq \Delta$, with Δ the trust region radius.

Although both methods can be efficiently applied to complex optimisation problems like (3.9-3.12), we choose to implement a trust region algorithm since it has been shown to outperform the line-search algorithm for large-scale ill-conditioned problems. When the Hessian matrix $\nabla^2 f_k$ is nearly singular, for example, the line search algorithm requires many iterations and gives only a small reduction in the objective function. The trust region method deals more effectively with this situation and is therefore preferable. In particular we will implement a trust-region inexact-Newton-conjugate-gradient method, as described in the coming section.

3.2.2 Trust-region inexact-Newton method

Trust region methods compute the next iterate by solving a much easier handled model function, which represents with good approximation the objective function at the current iterate. Since the necessary condition for stationarity of a function is that the first derivative vanishes at the extrema, we have:

$$m'_k(x_k + p_k) = \nabla f_k + \nabla^2 f_k p_k = 0 \tag{3.17}$$

where ∇f_k and p are vectors and $\nabla^2 f_k$ a matrix. The basic Newton step p_k is chosen by solving the symmetric linear system:

$$\nabla^2 f_k p_k = -\nabla f_k \tag{3.18}$$

and a new iterate x_{k+1} can be computed since the step $p = x_{k+1} - x_k$. Newton methods have the advantage of being rapidly convergent to the solution, provided that we start with a "close enough" initial guess x_0. However, in large scale algorithms, the direct factorisation of the Hessian matrix might be intractable. We are therefore forced to find approximations of p_k that are inexpensive to calculate but that still represent good steps. This can be done by solving equation (3.18) iteratively. Since the Newton step is not directly computed by inverting $\nabla^2 f_k$, this family of methods are generally referred to in the literature as *inexact-Newton methods*. In particular, we will use the conjugate-gradient method with modifications to handle negative curvature in the Hessian $\nabla^2 f_k$ to solve

eq.(3.18) and we will globalise the algorithm by a trust-region scheme. These two aspects are described in detail in the next sections.

The trust-region main loop

A key part of trust-region algorithms is the strategy used to choose the trust-region radius Δ_k at each iteration. Since we want to define Δ_k as a region in which the model function is a good approximation of the objective function, we base our choice on the agreement between the two functions in previous iterations. For the remainder of this section we will consider the model function to be centred around the current iterate, expressing dependence only with respect to p:

$$m_k(p) = f_k + p^T \nabla f_k + \frac{1}{2} p^T \nabla^2 f_k p \tag{3.19}$$

Given the step p_k we define the *gain factor* ρ_k to be:

$$\rho_k = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)} \tag{3.20}$$

where the numerator is called the *actual reduction* and the denominator the *predicted reduction*. The latter represents the reduction in f predicted by the model function m_k, being $m_k(0) = f(x_k)$. Since p_k is obtained by minimising the model m_k over a region that includes $p = 0$, the denominator will always be nonnegative. Hence, if ρ happens to be negative, it follows that the objective function at x_{k+1} has a higher value than at x_k, implying that the step must be rejected. On the other hand, if ρ is nonnegative we can distinguish the following cases: if ρ is close to 1, we infer that the model function adequately approximates the objective function and therefore we can allow the algorithm to take larger steps in the next iteration by expanding the trust region. Contrarily, if ρ is close to 0, the model is behaving poorly and therefore the region must be shrunk. Lastly, when ρ is "safely" between 0 and 1 we do not alter the trust region. In the literature it is suggested to reduce Δ for $\rho < \frac{1}{4}$, not alter it for $\frac{1}{4} \leq \rho \leq \frac{3}{4}$ and expand it for $\rho > \frac{3}{4}$. Denoting the maximum bound for the step lengths by $\hat{\Delta}$, the algorithm can be summarised as:

Algorithm 1 - Trust Region

Given: $\hat{\Delta} > 0, \Delta_0 \in (0, \hat{\Delta})$, and $\eta \in [0, \frac{1}{4})$:

for k = 0, 1, 2, ... **do**

Obtain the step p_k by approximately solving (3.19)

Evaluate the gain factor ρ_k from Eq. (3.20)

if $\rho_k < \frac{1}{4}$ **then**

$\Delta_{k+1} = \frac{1}{4}\Delta_k$

else if $\rho_k > \frac{3}{4}$ and $||p_k|| = \Delta_k$ **then**

$\Delta_{k+1} = min(2\Delta_k, \hat{\Delta})$

else

$\Delta_{k+1} = \Delta_k$

end if

if $\rho_k > \eta$ **then**

$x_{k+1} = x_k + p_k$

else

$x_{k+1} = x_k$

end if

end for

It is important to notice that the trust region is only expanded when two conditions are satisfied: first the gain factor needs to be larger than $\frac{3}{4}$, and second the actual step has to reach the trust region bound. If p_k stays strictly within the region, we infer that the region bound is not interfering with the progress of the algorithm, so we leave it unchanged. Furthermore with the parameter η we only accept the step when the reduction is at least higher than a minimum threshold.

To practically implement algorithm 1 we have to focus our attention in solving (3.19), which is called the trust-region subproblem. The trust region subproblem can be regarded as a constrained optimisation problem in which we attempt to minimise m_k by imposing that the step length does not exit the trust region.

$$\min_p f_k + p^T \nabla f_k + \frac{1}{2} p^T \nabla^2 f_k p \tag{3.21}$$

$$\text{such that: } ||p||_2 \leq \Delta_k \tag{3.22}$$

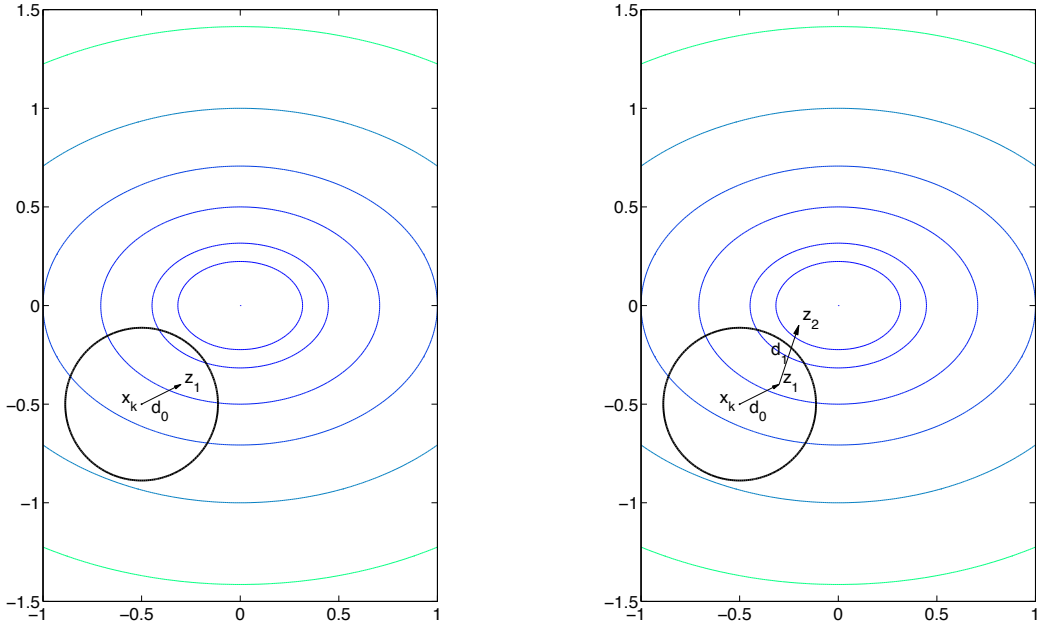
As we have seen, by Newton's method, the optimality condition is given by imposing stationarity of the model function and by solving the linear system (3.18) for p_k. A very efficient way to solve large linear systems iteratively is to use the conjugate gradient (CG) method, described in appendix D. This is used to generate the results in this report.

The complete algorithm

Now that all the machinery is in place we can construct the complete algorithm, globalising the conjugate gradient method by trust region scheme. The idea is as follows: at each step x_k we approximate the objective function by the corresponding quadratic form $m_k(p)$. By imposing stationarity of $m_k(p_k)$ and by applying the trust-region constraint on the step lengths, we generate the Newton's equations which give the problem in the form $\nabla^2 f_k p_k + \nabla f_k = 0$ such that $||p_k|| \leq \Delta_k$. This set of equations can be solved by the conjugate-gradient method provided that $\nabla^2 f_k$ is symmetric and positive definite. Symmetry is not an issue since the Hessian matrix is symmetric by definition, but unfortunately the positive definiteness is not guaranteed. Therefore we need to modify the CG algorithm (Alg. 3 in the appendix) to be able to handle this situation. Furthermore we have the constraint of the trust region bound, which limits us on the step length to be taken. As was previously explained, the conjugate-gradient method constructs the step iteratively as linear combination of search directions. In the algorithm we identify this sequence as z_j. Starting with $z_0 = 0$, we construct the step p_k by CG until one of the following conditions is encountered:

- The residual at z_j is lower than a given tolerance, meaning that the algorithm has reached the minima of the model function.
- The sequence exits the trust region bound, in which case we determine the minima along d_j such that $||p_k|| = \Delta_k$
- We find a direction of negative curvature, namely $d_j^T \nabla^2 f_k d_j \leq 0$. In this case we move to the boundary since, being the function upwardly convex, a minima cannot be found.

This is more clearly explained with the aid of Fig. 3.1. Fig. 3.1(a) shows the first iteration: starting at x_k we construct the step p_k (initially $p_k = 0$) by first moving in the direction $d_0 = -r_0$. This leads to $p_k = z_1$. At this point, if the residual of $m_k(z_1)$ is lower than the tolerance ϵ_k the step is complete and we return $p_k = z_1$, otherwise we continue iterating.



(a) Case 1 (b) Case 2

Figure 3.1: Trust-region conjugate gradient

In Fig. 3.1(b) this leads to z_2 which lies outside the trust-region. Since the constraint is violated we calculate the sub-step length τ such that we reach the trust bound along d_2, hence $p_k = z_1 + \tau d_2$ with $||p_k|| = \Delta_k$ and we return the step p_k. Yet a last possibility would be to encounter a negative curvature. In this case, similarly, we move to the boundary and calculate $p_k = z_j + \tau d_j$ such that $||p_k|| = \Delta_k$ and it minimises m_k along d_j.

Once the step is constructed we update $x_{k+1} = x_k + p_k$, we calculate the gain factor, verify that we had a descent in the objective function and continue with the update of the trust region.

The algorithm just described is due to Steihaug (10). In practice we will use Algorithm 2 in combination with algorithm 1, in which CG-Steihaug solves the step "Obtain the step p_k by approximately solving (3.19)".

When we terminate by exiting the trust-region bound or by encountering negative curvature, the evaluation of τ is required. This can be computed by taking the positive root of the quadratic equation:

$$||z_j + \tau d_j|| = \Delta_k \tag{3.23}$$

$$(z_j + \tau d_j)^T (z_j + \tau d_j) = \Delta_k^2 \tag{3.24}$$

$$\tau^2 d_j^T d_j + 2\tau z_j d_j = \Delta_k^2 - z_j^T z_j \tag{3.25}$$

For further details and proofs of convergence of the method, the reader can consult Steihaug (10). For further information regarding trust-region methods and PDE-constrained optimisation, the reader is referred to (11; 12; 13; 14)

Algorithm 2 - CG-Steihaug

Given tolerance $\epsilon_k > 0$;
Set $z_0 = 0$, $r_0 = \nabla f_k$, $d_0 = -r_0$;
if $||r_0|| < \epsilon_k$ **then**
  return $p_k = z_0 = 0$
end if

for k = 0, 1, 2, ... **do**
  if $\delta_k^T H_k d_k \leq 0$ **then**
    Find τ such that $p_k = z_j + \tau d_j$ minimizes $m_k(p_k)$
      and satisfies $||p_k|| = \Delta_k$;
    return p_k;
  end if

  Set $\alpha_j = r_j^T r_j / d_j^T H d_j$
  Set $z_{j+1} = z_j + \alpha_j d_j$
  if $||z_{j+1}|| \geq \Delta_k$ **then**
    Find τ such that $p_k = z_j + \tau d_j$ satisfies $||p_k|| = \Delta_k$;
    return p_k;
  end if

  Set $r_{j+1} = r_j + \alpha_j H d_j$
  if $||r_{j+1}|| < \epsilon_k$ **then**
    return $p_k = z_{j+1}$
  end if

  Set $\beta_{j+1} = r_{j+1}^T r_{j+1} / r_j^T r_j$
  Set $d_{j+1} = -r_{j+1} + \beta_{j+1} d_j$
end for

3.3 Solution of the Optimisation Problem

Now that we have defined the algorithm we will implement, we can focus our attention on the actual solution of the optimisation problem (3.9-3.12). This represents a so-called PDE-constrained optimisation problem. The term indicates that the objective function (3.9) needs to be minimised subject to constraints on the possible values of the independent variables, and these constraints can in general have the form of Partial-Differential Equations (PDE). In our particular case this differential equation has the form of an Ordinary-Differential Equation (ODE) since the only dependence is time. Hereinafter we will refer to the ODE constraint as the *state equation*. In the previous section we have derived an algorithm that can efficiently solve unconstrained optimisation problems. We will now show that it is possible to convert a constrained optimisation problem to an unconstrained one by segregation.

As we have previously seen, a classical way to solve constrained optimisation problems is to introduce Lagrange multipliers and form a Lagrangian functional $\mathscr{L}$ that incorporates the constraints in the form of inner product with the multipliers. In our case this leads to:

$$\begin{aligned}\mathscr{L}(\Phi,\alpha,\lambda,\mu) = \frac{1}{2}\int_0^{t_f}(u-\hat{u})^T H(u-\hat{u})dt + \frac{\beta}{2}\sum_{j=1}^{m}(1-\phi_j^T\phi_j)^2 \\ + \frac{\beta}{2}\sum_{\substack{i,j=1\\ i\neq j}}^{m}(\phi_i^T\phi_j)^2 + \int_0^{t_f}\lambda^T(\Phi^T M\Phi\dot{\alpha} \\ + \Phi^T K\Phi\alpha - \Phi^T f)dt + \mu^T(\Phi^T M\Phi\alpha_0 - \Phi^T M u_0)\end{aligned} \tag{3.26}$$

where $\lambda = \lambda(t) \in \mathbb{R}^M$ and $\mu \in \mathbb{R}^M$ are the Lagrange multipliers, that respectively enforce the state equation and initial conditions. The optimality conditions can be derived by imposing stationarity of the Lagrangian with respect to $\Phi, \alpha, \lambda, \mu$.
Setting the first variation of the Lagrangian with respect to λ to zero and arguing that the variation of λ is arbitrary in $[0, t_f]$, and setting the derivative of the Lagrangian with respect to μ to zero, simply recovers the state equation and initial conditions (3.10-3.11). Setting the first variation of the Lagrangian with respect to α to zero, and arguing that the variation of α is arbitrary in $[0, t_f]$, at $t = 0$, and at $t = t_f$, yields the so called *adjoint equation*, final condition and definition of μ:

$$-\Phi^T M\Phi\dot{\lambda} + \Phi^T K^T\Phi\lambda = \Phi^T H(u - \Phi\alpha), \tag{3.27}$$

$$\lambda(t_f) = 0, \tag{3.28}$$

$$\mu = \lambda(0) \tag{3.29}$$

where, without loss of generality, M is assumed to be symmetric. Finally, taking the derivative of the Lagrangian with respect to the basis vector Φ yields the following matrix equation:

$$\begin{aligned}\delta\mathscr{L}_\Phi = &\int_0^{t_f} H(\Phi\alpha - u)\alpha^T dt + 2\beta\Phi(\Phi^T\Phi - I)\\ &+ \int_0^{t_f} [M\Phi(\lambda\dot{\alpha}^T + \dot{\alpha}\lambda^T) + K^T\Phi\lambda\alpha^T + K\Phi\alpha\lambda^T - f\lambda^T]dt\\ &+ M\Phi\mu\alpha_0^T + M(\Phi\alpha_0 - u_0)\mu^T = 0.\end{aligned} \tag{3.30}$$

The trick to re-express the constrained optimisation (3.9-3.12) to an unconstrained one is by eliminating the dependence on α from the objective function (3.9). This can be done by segregating the solution of the state equation from the optimisation problem, solving them sequentially. More specifically we first solve (3.10-3.11) to determine α, and substitute this result into the objective function (3.9). Secondly, since α does not represent a dependent variable in the optimisation process anymore, the optimisation problem is converted to an unconstrained one, which can be solved for only the Φ variables. The original problem is thus approximated by the segregated approach where we require that for small steps in Φ, and particularly at the limit $\Delta\Phi \to 0$, the two approaches converge to the same result.

Having rewritten the optimisation problem to an equivalent unconstrained one, it is now possible to apply the algorithm previously developed in the determination of the optimal Φ. To construct the Newton set of equations we need the evaluation of the gradient and of the Hessian matrix at the current iterate. The gradient of the unconstrained function $\hat{\mathscr{G}}$ is given by $\delta\mathscr{L}_\Phi$ in equation (3.30) when α satisfies the state equation and (λ, μ) satisfy the adjoint equation. Hence the procedure to calculate the gradient can be summarised by the following steps: first we solve the state equation to determine α, this allows the solution of the adjoint equation, leading to the determination of λ and μ. With the computed (α, λ, μ) we have all the information to evaluate the gradient at the given iterate Φ_k.

In the conjugate gradient method used in algorithm 2, we notice that the Hessian matrix is always multiplied by the search direction d_j. Therefore it is actually not required to compute the exact Hessian matrix, but we can approximate the Hessian-vector product on-the-fly. Consider a Taylor expansion of the gradient $g(x)$ of a given function:

$$g(x + \Delta x) \approx g(x) + H(x)\Delta x \tag{3.31}$$

with $H(x)$ the Hessian matrix. Now consider the Hessian vector we want to compute. For small h we have:

$$g(x + hd_j) \approx g(x) + hH(x)d_j \tag{3.32}$$

And hence:

$$H(x)d_j = \frac{g(x + hd_j) - g(x)}{h} \tag{3.33}$$

The Hessian-vector product is then approximated by the directional derivative of the gradient with respect to the search direction. In our case this means calculating another set of state and adjoint equations to define $g(x+hd_j)$, where x for us represents Φ. This because modifying Φ leads to a different solution of the state equation, thus different α's, which in turn changes the definition of λ and therefore of the gradient. The complete algorithm requires consequently the solution of a pair of state and adjoint equations at each iteration.

As a last note, we mentioned in section (3.1) that as the problem is nonconvex, we are not guaranteed to reach a global minima. Accordingly, the choice of the initial guess plays an important role, since being "close enough" at the beginning of the algorithm assures better convergence properties of the Newton's method. For this reason we choose Φ derived by the POD procedure to be our candidate initial guess. Firstly, the POD is understood to perform relatively well in the determination of an optimal set of basis functions, which will also be shown in the preliminary results section. And secondly the determination of the POD basis is relatively cheap.

3.4 Results

3.4.1 Heat Equation

To write this system in the LTI formulation (3.1), we have to define mass matrix, stiffness matrix and input vector. For the finite difference case the mass matrix corresponds to the identity matrix ($M = I$); the stiffness matrix K, as we can see in the second term of Eq. (2.24) is a tridiagonal matrix containing 2 on the main diagonal and -1 on the upper and lower diagonals, multiplied by $\frac{k}{\Delta x^2}$. This except first and last line which contain the equations corresponding to the boundary conditions. The vector f contains the input to the system (forcing terms, sources, sinks, etc.), which in our case are zero. The only terms will then be the known boundary values moved to the right hand side. In this way we re-express the problem in the LTI system:

$$M\dot{u} + Ku = f \tag{3.34}$$

where $M, K \in \mathbb{R}^{N+1\times N+1}$ and $u, f \in \mathbb{R}^{N+1}$

The first functional we will consider using the FDF-ROM approach is the average temperature in the region towards the end of the rod $0.8 \leq x \leq 1$ (hereinafter referred as FDF-ROM$_{Ave}$). The matrix C specifying the functional has to be constructed such that, once applied to the solution vector u, it returns the average over the targeted domain. To illustrate this, imagine for a moment that our rod is discretised in 5 subintervals. In this case starting from the left end, each node corresponds to the position 0, 0.2, 0.4,...,1 respectively. The region we are interested in is spanned by the last 2 nodes and therefore to obtain the average over this region ($g = Cu$), we have to construct C as:

$$g = \begin{bmatrix} 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{bmatrix} = \frac{u_4 + u_5}{2}$$

which is the average of the last two nodes.

Fig. (3.2) presents a comparison between POD-ROM and FDF-ROM for both the

average solution case just described (FDF-ROM$_{Ave}$) and for the full domain reconstruction (FDF-ROM$_{Full}$). Although small, we notice an improvement for every dimension of the reduced-order model when the goal-oriented approach for the rod-end average is used. For the complete domain the difference is basically unrecognisable, being in the order of 10^{-4} for the first mode and lower than 10^{-7} for the other cases. Furthermore, since over 99% of the energy is contained in the first 4 modes, the reduced-order model of order 4 resolves the temperature distribution very accurately and almost no improvement can be achieved by adding extra modes. The error is evaluated by the error measure (3.13).

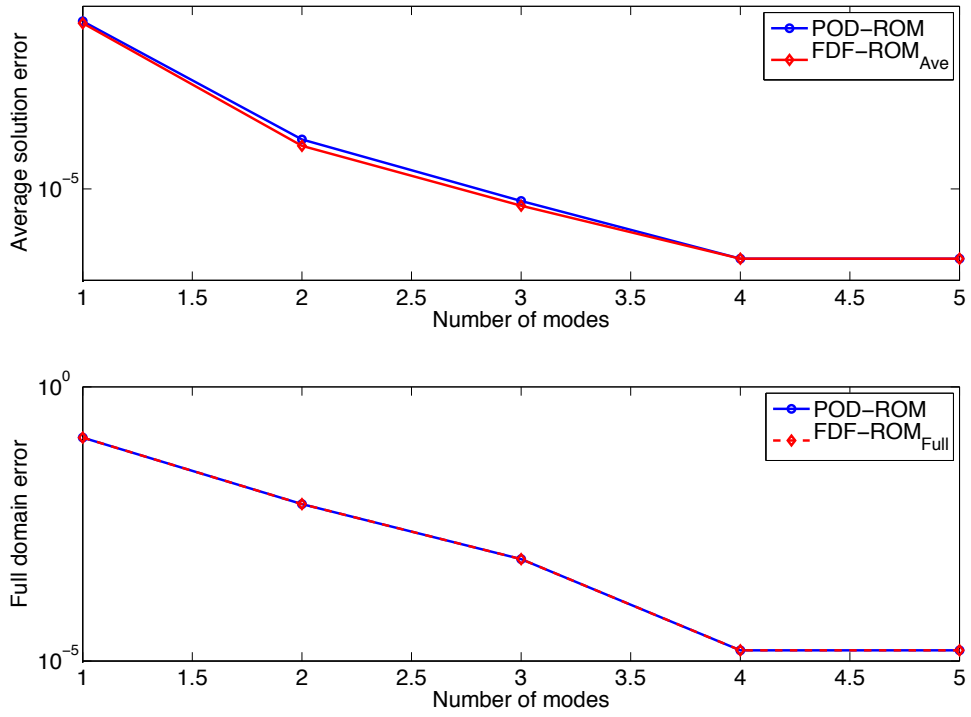


Figure 3.2: Objective function POD and goal-oriented errors

For completeness we present the difference brought by the optimisation procedure in the mode shape. Only the order 1 case is presented since in the other cases the difference is not recognisable in the plot. Since in the reconstruction of the full domain the POD basis performs well, the difference brought by the optimised basis is very limited. On the other hand choosing the rod-end average as a functional results in a higher modification of the mode shape.

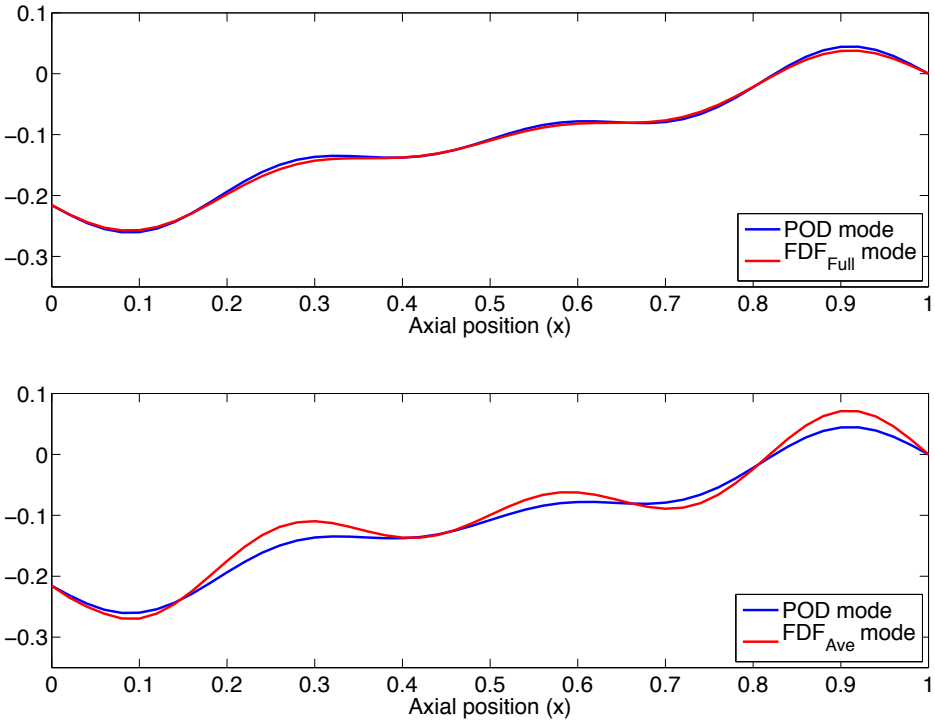


Figure 3.3: Comparison of differently optimised modes

3.4.2 Pure convection

In this case the stiffness matrix will be bi-diagonal with entries only on the upper and lower diagonal, plus entries in the north-east and south-west corners accounting for the periodic boundary conditions. f is the zero vector and M the identity matrix once again.

As indicated, the pure convection case has a specific physical constraint in the determination of the mode shapes since the energy transport is only possible by using shifted modes which span the complete domain. As shown in Fig. 3.4 (bottom) the new approach, in the case of full domain goal, only leads to a very small improvement over the POD. On the other hand Fig. 3.4 (top) shows a comparison between POD and the new method in case we target only half of the solution domain (FDF-ROM$_{Part}$). In particular we have chosen the solution in $0.5 \leq x \leq 1$.

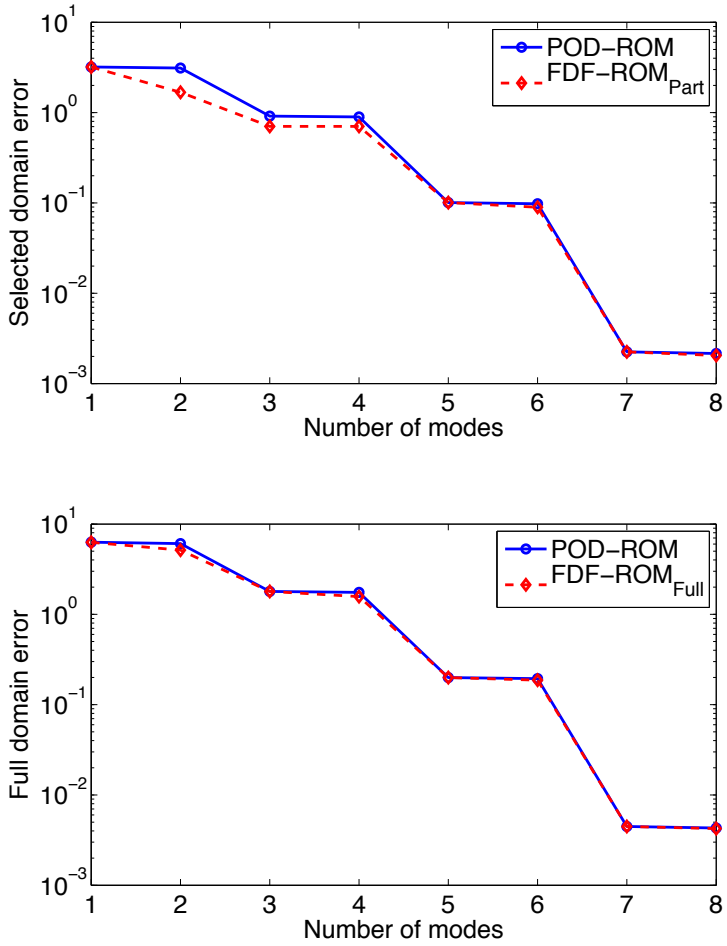


Figure 3.4: Objective function POD and goal-oriented errors

3.4.3 Convection-Diffusion

For the convection-diffusion equation the stiffness matrix will be tri-diagonal, with the following entries:

- On the main diagonal ($\frac{2k}{\Delta x^2}$)
- On the upper diagonal ($\frac{a}{2\Delta x} - \frac{k}{\Delta x^2}$)
- On the lower diagonal ($-\frac{a}{2\Delta x} - \frac{k}{\Delta x^2}$)

M the identity matrix once again and f the zero vector with only entries accounting for the boundary conditions.

We investigated the behaviour of the POD-ROM and FDF-ROM in both the full state reconstruction and with a specific functional. The targeted functional is the solution in half of the domain, $0.5 \leq x \leq 1$. Fig.(3.5) presents a comparison between the POD-ROM and the FDF-ROM approach for the full domain representation and for the second functional. The advantages of the new methodology are much clearer in this case, both for the full domain and for a local target.

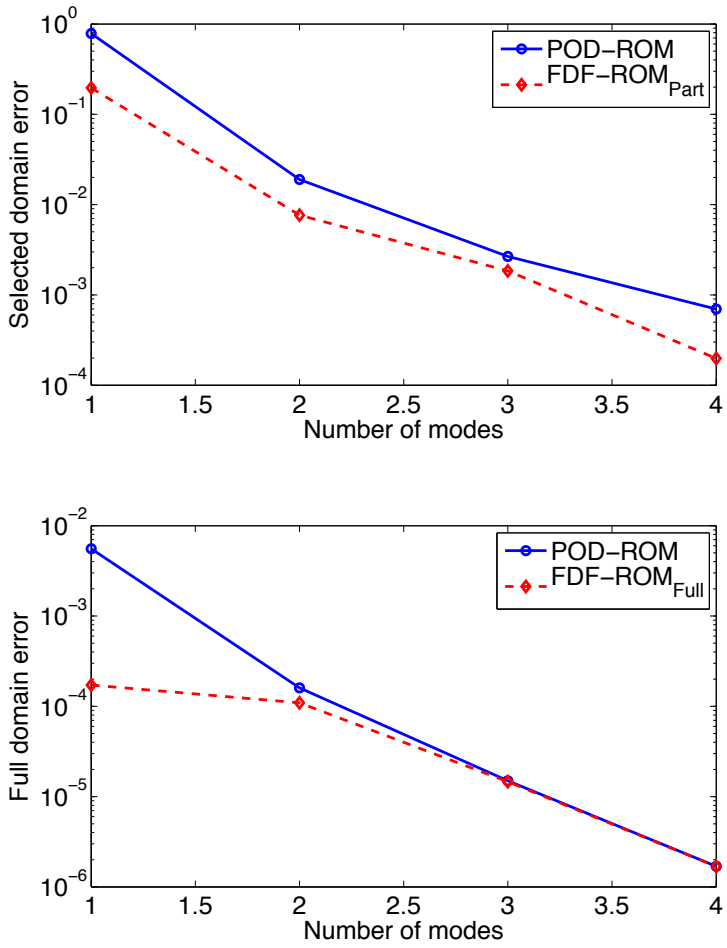


Figure 3.5: POD and goal oriented objective function errors

Chapter 4

Semi-Continuous formulation

The method presented in the preceding section however, poses several limitations in terms of applicability. First of all the discrete nature of the equations implies that the reduced-order model can only be constructed when mass and stiffness matrices are readily available. How these are to be constructed is not always obvious. If the reference data comes from a detailed computational fluid dynamics simulation a natural choice might be the corresponding mass and stiffness matrices of the discretisation used to generate the data. If the reference data comes from experiments, then no obvious choice for M and K exists. In any case, the choice of the matrices affects the speed and performance of the find ROM in a hard-to-predict way. A second limitation is the strictly linear character of the derivation proposed, limiting the span of governing equations considered and the choice of the goal-functional. Lastly, an important issue in the construction of the reduced-order model is the treatment of the boundary conditions. In the derivation it is implicitly understood that the boundary conditions are handled by direct substitution, but other possibilities are available and clarifications must be addressed.

To overcome these drawbacks, we have generalised the approach of ref. (7) by considering the generation of the reduced-order model, and the optimisation process, in a continuous setting. This implies coupling the discrete nature of the modes (derived from numerical simulations or experiments), with the continuous setting of the governing equations. The reduced-order model is then constructed by projecting the governing equation into each mode according to Galerkin method. Note, however, that since the reference data are either generated by numerical simulations or experiments, they always have a finite character. It is most-straight forward to define the modes in the same discrete space

and thus the generalisation we consider here couples the discrete data with a continuous representation of the governing equations, functionals, and optimisation process. We refer to this as a semi-continuous formulation (SCF).

Using the SCF, we are no longer limited to the LTI formalism, but we can accommodate any differential equation and we also avoid the ambiguities associated with choosing mass and stiffness matrices arbitrarily at the beginning of the optimisation process. Furthermore, we clarify the treatment of arbitrary functionals in the optimisation process and are no longer limited to be $g = Cu$. g can now take any form, e.g. $sin(u), u^2$. For clarity of exposition we will refer to the LTI algebraic approach propose by Bui-Thanh as the fully-discrete formulation FDF, and the semi-continuous formulation as SCF.

4.1 Derivation

To exemplify the SCF, we first consider the construction of the reduced-order model for two test cases, the 1D heat equation with homogeneous Dirichlet boundary conditions and the convection-diffusion equation with non-homogeneous Dirichlet boundary conditions.

Heat Equation

We will consider the following equation:

$$\frac{\partial u}{\partial t} = k\frac{\partial^2 u}{\partial x^2} \tag{4.1}$$

$$u(0,t) = u(1,t) = 0; \tag{4.2}$$

To construct a reduced-order model, we approximate u by $\sum_{j=1}^{n} \alpha_j \phi_j$, where ϕ's are now continuous functions of x, and we employ Galerkin projection. This leads to:

$$\int_0^1 \sum_{j=1}^{n} \dot{\alpha}_j \phi_j \phi_k dx - k \int_0^1 \sum_{j=1}^{n} \alpha_j \phi_j'' \phi_k dx = 0 \tag{4.3}$$

where $\dot{}$ indicates a time derivative, and $'$ a spatial derivative. Rearranging and integrating the second integral by parts we obtain:

$$\sum_{j=1}^{n} \dot{\alpha}_j \int_0^1 \phi_j \phi_k dx - k \sum_{j=1}^{n} \alpha_j [\phi_j' \phi_k \big|_0^1] + k \sum_{j=1}^{n} \alpha_j \int_0^1 \phi_j' \phi_k' dx \tag{4.4}$$

For homogeneous BC ϕ must be 0 at $x = 0$ and $x = 1$. The second term in equation (4.4) vanishes and we are left with:

$$\sum_{j=1}^{n} \dot{\alpha_j} \int_0^1 \phi_j \phi_k dx + k \sum_{j=1}^{n} \alpha_j \int_0^1 \phi_j' \phi_k' dx = 0 \tag{4.5}$$

which represents the weak form of the heat equation.

Remark: it is important to notice that the assumption of continuity in the interval of definition plays a crucial role. If a discontinuity is present, the derivative of the mode is not defined at the jump and therefore specific adaptations must be addressed, details of which are beyond the scope of this work.

Convection-diffusion equation

We now consider the convection-diffusion equation for the case of non-homogeneous Dirichlet boundary conditions (NH-BC). When treating NH-BC, special attention must be paid to evaluating the boundary terms. Different formulations are possible. We will present two different cases, namely direct substitution and the penalty method.

The convection-diffusion equation and boundary conditions to be considered are:

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = k\frac{\partial^2 u}{\partial x^2} \tag{4.6}$$

$$u(0,t) = 1 \tag{4.7}$$

$$u(1,t) = 0; \tag{4.8}$$

As before, we approximate the solution by $u = \sum_{j=1}^{n} \alpha_j \phi_j$, and employ Galerkin projection:

$$\int_0^1 \sum_{j=1}^{n} \dot{\alpha_j} \phi_j \phi_k dx + a \int_0^1 \sum_{j=1}^{n} \alpha_j \phi_j' \phi_k dx = k \int_0^1 \sum_{j=1}^{n} \alpha_j \phi_j'' \phi_k dx \tag{4.9}$$

which can be rearranged as:

$$\sum_{j=1}^{n} \dot{\alpha_j} \int_0^1 \phi_j \phi_k dx + a \sum_{j=1}^{n} \alpha_j \int_0^1 \phi_j' \phi_k dx = k \sum_{j=1}^{n} \alpha_j \int_0^1 \phi_j'' \phi_k dx \tag{4.10}$$

To reduce the order of the derivatives we integrate by parts, leading to:

$$\sum_{j=1}^{n} \dot{\alpha_j} \int_0^1 \phi_j \phi_k dx - a \sum_{j=1}^{n} \alpha_j \int_0^1 \phi_j \phi_k' dx + k \sum_{j=1}^{n} \alpha_j \int_0^1 \phi_j' \phi_k' dx + a \sum_{j=1}^{n} \alpha_j [\phi_j \phi_k \big|_0^1] - k \sum_{j=1}^{n} \alpha_j [\phi_j' \phi_k \big|_0^1] = 0 \tag{4.11}$$

which represents the weak form of the convection diffusion equation.

The first method to enforce the boundary conditions, is to substitute the exact value at the boundary when required. Specifically, we know that at the boundary $u(0) = 1$ and hence $\sum_{j=1}^{n} \alpha_j \phi_j(0) = 1$, while $u(1) = 0$ meaning that $\sum_{j=1}^{n} \alpha_j \phi_j(1) = 0$. Furthermore $\phi = 0$ at $x = 1$ being the boundary condition homogeneous at $x = 1$. This leads to:

$$\sum_{j=1}^{n} \dot{\alpha_j} \int_0^1 \phi_j \phi_k dx - a \sum_{j=1}^{n} \alpha_j \int_0^1 \phi_j \phi_k' dx + k \sum_{j=1}^{n} \alpha_j \int_0^1 \phi_j' \phi_k' dx - a u(0) \phi_k(0) + k \sum_{j=1}^{n} \alpha_j [\phi_j'(0) \phi_k(0)] = 0 \tag{4.12}$$

where the final flux term therefore remains unknown.

The second method we consider for treating boundary conditions is a so-called penalty method. The key idea of the method is to leave undetermined the boundary fluxes arising from the weak formulation, and enforce the satisfaction of the boundary conditions by adding a penalty term in the form $\sigma(u - \gamma)\big|_0^1$, with γ the boundary value. As σ goes to infinity the deviation from the exact solution is increasingly penalised, and the solution converges to the solution of the original problem. Furthermore, particularly for diffusive-dominated problems, a second term should be added to the formulation to increase stability. This is the so-called adjoint-consistency term and has the form $k[\phi'(u - \gamma)]\big|_0^1$ (see (15)) . Notice that the addition of these extra terms does not influence the consistency of the system, since upon substitution of the exact solution, penalty and adjoint consistency terms vanish and we are left with the original problem.

$$\sum_{j=1}^{n} \dot{\alpha_j} \int_0^1 \phi_j \phi_k dx - a \sum_{j=1}^{n} \alpha_j \int_0^1 \phi_j \phi_k' dx + k \sum_{j=1}^{n} \alpha_j \int_0^1 \phi_j' \phi_k' dx$$
$$+ a \sum_{j=1}^{n} \alpha_j [\phi_j \phi_k \big|_0^1] - k \sum_{j=1}^{n} \alpha_j [\phi_j' \phi_k \big|_0^1]$$
$$+ [\sigma(u-\gamma) + k\phi'(u-\gamma)]\big|_0^1 = 0 \tag{4.13}$$

The penalty parameter must be chosen depending on the specific need of the application. If we are interested in having a high fidelity of the solution close to the boundary we would increase the penalty term to bring the weak formulation the closest possible to the exact solution. On the other hand, allowing jumps in the boundary terms, allows for a better representation of the interior solution, as will be illustrated by the numerical tests.

4.2 Optimisation

Having seen the derivation of the SCF-ROM for two examples of interest, we are now ready to move one step forward and set up the optimisation procedure in the continuous setting. As for the discrete setting, we now seek a set of model-constrained optimised modes in the continuous framework such that the L_2-norm is minimised. However, in the current formulation we no longer wish to be limited in choosing g to be linear, $g = C\Phi$, but desire to be free to pick whichever function suits our needs. We will indicate this as $g = f(u)$. For ease of presentation we will derive the method for a specific example, the convection-diffusion equation analysed in Sec. 4.3 with direct substitution. The optimisation problem can be stated as, find $\Phi \in \mathbb{R}^N$ where:

$$\Phi = \arg\min_{\Phi} E(\Phi) \tag{4.14}$$

$$E(\Phi) = \int_0^{t_f} \int_0^1 (g - \hat{g})^2 dx dt \tag{4.15}$$

Applying the model constraint and orthonormality, the problem becomes find $\Phi \in \mathbb{R}^N$ where:

$$\min_{\Phi,\alpha} \mathscr{G} = \frac{1}{2}\int_0^{t_f}\int_0^1 (g-\hat{g})^2 dxdt + \frac{\beta}{2}\sum_{j=1}^{m}\left(1-\int_0^1 \phi_j^2 dx\right)^2 + \frac{\beta}{2}\sum_{\substack{i,j=1\\ i\neq j}}^{m}\left(\int_0^1 (\phi_i\phi_j)dx\right)^2 \tag{4.16}$$

$$\begin{aligned}\text{subject to: } & \sum_{j=1}^{n}\dot{\alpha_j}\int_0^1 \phi_j\phi_k dx - a\sum_{j=1}^{n}\alpha_j\int_0^1 \phi_j\phi_k' dx + k\sum_{j=1}^{n}\alpha_j\int_0^1 \phi_j'\phi_k' dx \\ & - au(0)\phi_k(0) + k\sum_{j=1}^{n}\alpha_j[\phi_j'(0)\phi_k(0)] = 0 \end{aligned} \tag{4.17}$$

$$\sum_{j=1}^{n}\alpha_j(0)\int_0^1 \phi_j\phi_k dx = \int_0^1 u_0\phi_k dx \tag{4.18}$$

$$\hat{g} = f(u) \tag{4.19}$$

We again construct the gradient using an adjoint method. The first step is to define the Lagrangian by taking the inner product of each constraint with its respective Lagrange multiplier. Since each projected state equation has to be satisfied, there will be λ_k for $k = 1..n$ for which:

$$\begin{aligned}\mathscr{L}(\Phi,\alpha,\lambda,\mu) = & \frac{1}{2}\int_0^{t_f}\int_0^1 (g-\hat{g})^2 dxdt + \frac{\beta}{2}\sum_{j=1}^{m}\left(1-\int_0^1 \phi_j^2 dx\right)^2 \\ & + \frac{\beta}{2}\sum_{\substack{i,j=1\\ i\neq j}}^{m}\left(\int_0^1 (\phi_i\phi_j)dx\right)^2 + \int_0^{t_f}\sum_{k=1}^{n}\lambda_k\Bigg[\sum_{j=1}^{n}\dot{\alpha_j}\int_0^1 \phi_j\phi_k dx \\ & - a\sum_{j=1}^{n}\alpha_j\int_0^1 \phi_j\phi_k' dx + k\sum_{j=1}^{n}\alpha_j\int_0^1 \phi_j'\phi_k' dx \\ & - au(0)\phi_k(0) + k\sum_{j=1}^{n}\alpha_j\phi_j'(0)\phi_k(0)\Bigg]dxdt \\ & + \sum_{k=1}^{n}\mu_k\left(\sum_{j=1}^{n}\alpha_j(0)\int_0^1 \phi_j\phi_k dx - \int_0^1 u_0\phi_k dx\right) = 0 \end{aligned} \tag{4.20}$$

We now impose stationarity of the Lagrangian to define the optimality conditions. This implies taking a set of Fréchet derivatives with respect to each defining function. Setting the variation of the Lagrangian with respect to each λ_k and μ_k to zero simply recovers

the respective projected state equation and the initial condition, respectively. Setting the variation with respect to each α (which we will denote α_q) to zero yields the corresponding adjoint equation, final condition and definition of μ.

$$-\sum_{j=1}^{n}\dot{\lambda}_j\int_0^1\phi_q\phi_j dx+\sum_{j=1}^{n}\lambda_j\left[k\int_0^1\phi_q'\phi_j' dx-a\int_0^1\phi_q\phi_j' dx+k\phi_j(0)\phi_q'(0)\right]$$
$$=\int_0^1\left[f(u)-f\Big(\sum_{j=1}^{n}\alpha_j\phi_j\Big)\right]f'\Big(\sum_{j=1}^{n}\alpha_j\phi_j\Big)\phi_q dx \tag{4.21}$$

$$\lambda_q(t_f)=0, \tag{4.22}$$

$$\mu_q=\lambda_q(0) \tag{4.23}$$

The last step is to differentiate the Lagrangian with respect to the each basis vector ϕ_q. For clarity, we split the derivation for each term, denoted with F_i.

1^{st} term

$$\frac{\delta F_1[\phi_q]}{\delta\phi_q(x)}=\frac{\delta}{\delta\phi_q(x)}\left\{\frac{1}{2}\int_0^{t_f}\int_0^1\left[f(u)-f\Big(\sum_{j=1}^{n}\alpha_j\phi_j\Big)\right]^2 dxdt\right\}=$$
$$=\int_0^{t_f}\left[f\Big(\sum_{j=1}^{n}\alpha_j\phi_j\Big)-f(u)\right]f'\Big(\sum_{j=1}^{n}\alpha_j\phi_j\Big)\alpha_q dt \tag{4.24}$$

2^{nd} term

$$\frac{\delta F_2[\phi_q]}{\delta\phi_q(x)}=\frac{\delta}{\delta\phi_q(x)}\left\{\frac{\beta}{2}\sum_{j=1}^{m}\left(1-\int_0^1\phi_j^2 dx\right)^2\right\}=$$
$$=2\beta\phi_q\left(\int_0^1(\phi_q)^2dx-1\right) \tag{4.25}$$

3^{rd} term

$$\frac{\delta F_3[\phi_q]}{\delta\phi_q(x)}=\frac{\delta}{\delta\phi_q(x)}\left\{\frac{\beta}{2}\sum_{\substack{i,j=1\\ i\neq j}}^{m}\left(\int_0^1(\phi_i\phi_j)dx\right)^2\right\}=$$
$$=2\beta\sum_{\substack{j=1\\ j\neq q}}^{m}\phi_j\left(\int_0^1(\phi_q\phi_j)dx\right) \tag{4.26}$$

4^{th} term

$$\frac{\delta F_4[\phi_q]}{\delta\phi_q(x)} = \frac{\delta}{\delta\phi_q(x)}\left\{ \int_0^{t_f} \sum_{k=1}^{n} \lambda_k \left[\sum_{j=1}^{n} \dot{\alpha}_j \int_0^1 (\phi_j\phi_k)dx \right] dt \right\} =$$
$$= \int_0^{t_f} \left(2\lambda_q\dot{\alpha}_q\phi_q + \lambda_q \sum_{\substack{j=1 \\ j\neq q}}^{n} \dot{\alpha}_j\phi_j + \dot{\alpha}_q \sum_{\substack{j=1 \\ j\neq q}}^{n} \lambda_j\phi_j \right) dt \tag{4.27}$$

5^{th} term

$$\frac{\delta F_5[\phi_q]}{\delta\phi_q(x)} = \frac{\delta}{\delta\phi_q(x)}\left\{ \int_0^{t_f} -a \sum_{j=1}^{n} \alpha_j \int_0^1 \phi_j\phi_k' dx \right] dt \right\} =$$
$$= -a \int_0^{t_f} \left(\lambda_q \sum_{\substack{j=1 \\ j\neq q}}^{n} \alpha_j\phi_j' - \alpha_q \sum_{\substack{j=1 \\ j\neq q}}^{n} \lambda_j\phi_j' \right) dt \tag{4.28}$$

6^{th} term

$$\frac{\delta F_6[\phi_q]}{\delta\phi_q(x)} = \frac{\delta}{\delta\phi_q(x)}\left\{ \int_0^{t_f} \sum_{k=1}^{n} \lambda_k \left[k \sum_{j=1}^{n} \alpha_j \int_0^1 \phi_j'\phi_k' dx \right] dt \right\} =$$
$$= -k \int_0^{t_f} \left(2\lambda_q\alpha_q\phi_q'' + \lambda_q \sum_{\substack{j=1 \\ j\neq q}}^{n} \alpha_j\phi_j'' + \alpha_q \sum_{\substack{j=1 \\ j\neq q}}^{n} \lambda_j\phi_j'' \right) dt \tag{4.29}$$

7^{th} term

$$\frac{\delta F_7[\phi_q]}{\delta\phi_q(x)} = \frac{\delta}{\delta\phi_q(x)}\left\{ \int_0^{t_f} \sum_{k=1}^{n} \lambda_k \left[k \sum_{j=1}^{n} \alpha_j \int_0^1 \phi_j'\phi_k' dx \right] dt \right\} =$$
$$= -\int_0^{t_f} \left(2\lambda_q\alpha_q\phi_q'' + \lambda_q \sum_{\substack{j=1 \\ j\neq q}}^{n} \alpha_j\phi_j'' + \alpha_q \sum_{\substack{j=1 \\ j\neq q}}^{n} \lambda_j\phi_j'' \right) dt \tag{4.30}$$

8^{th} term

$$\frac{\delta}{\delta\phi_q(x)}\left\{ -au(0)\phi_k(0) \right\} = -au(0) \tag{4.31}$$

9^{th} term

$$\frac{\delta}{\delta\phi_q(x)}\left\{ k \sum_{j=1}^{n} \alpha_j\phi_j'(0)\phi_k(0) \right\} = k \sum_{j=1}^{n} \alpha_j\phi_j'(0) \tag{4.32}$$

10^{th} term

$$\frac{\delta F_{10}[\phi_q]}{\delta\phi_q(x)} = \frac{\delta}{\delta\phi_q(x)}\left\{\sum_{k=1}^{n}\mu_k\left(\sum_{j=1}^{n}\alpha_j(0)\int_0^1\phi_j\phi_k dx\right)\right\} = \\ = 2\mu_q\alpha_q\phi_q + \mu_q\sum_{\substack{j=1\\j\neq q}}^{n}\alpha_j\phi_j + \alpha_q\sum_{\substack{j=1\\j\neq q}}^{n}\mu_j\phi_j \tag{4.33}$$

11^{th} term

$$\frac{\delta F_{11}[\phi_q]}{\delta\phi_q(x)} = \frac{\delta}{\delta\phi_q(x)}\left\{\sum_{k=1}^{n}\mu_k\left(-\int_0^1 u_0\phi_k dx\right)\right\} = \\ = -\mu_q u_0 \tag{4.34}$$

Putting all together we have the expression for the gradient:

$$\begin{aligned}\delta\mathscr{L}_{\Phi_q} = &\int_0^{t_f}\left[f\left(\sum_{j=1}^{n}\alpha_j\phi_j\right) - f(u)\right]f'\left(\sum_{j=1}^{n}\alpha_j\phi_j\right)\alpha_q dt + 2\beta\phi_q\left(\int_0^1(\phi_q)^2dx - 1\right)\\
&+ 2\beta\sum_{\substack{j=1\\j\neq q}}^{m}\phi_j\left(\int_0^1(\phi_q\phi_j)dx\right) + \int_0^{t_f}\left\{\left(2\lambda_q\dot{\alpha}_q\phi_q + \lambda_q\sum_{\substack{j=1\\j\neq q}}^{n}\dot{\alpha}_j\phi_j + \dot{\alpha}_q\sum_{\substack{j=1\\j\neq q}}^{n}\lambda_j\phi_j\right)\right.\\
&\left. - a\left(\lambda_q\sum_{\substack{j=1\\j\neq q}}^{n}\alpha_j\phi_j' - \alpha_q\sum_{\substack{j=1\\j\neq q}}^{n}\lambda_j\phi_j'\right) - k\left(2\lambda_q\alpha_q\phi_q'' + \lambda_q\sum_{\substack{j=1\\j\neq q}}^{n}\alpha_j\phi_j'' + \alpha_q\sum_{\substack{j=1\\j\neq q}}^{n}\lambda_j\phi_j''\right)\right\}dt\\
&- au(0)\lambda_q + k\sum_{j=1}^{n}\alpha_j\phi_j'(0)\lambda_q + 2\mu_q\alpha_q\phi_q + \mu_q\sum_{\substack{j=1\\j\neq q}}^{n}\alpha_j\phi_j + \alpha_q\sum_{\substack{j=1\\j\neq q}}^{n}\mu_j\phi_j - \mu_q u_0\end{aligned} \tag{4.35}$$

All the ingredients are now in place to implement the procedure computationally. Although the base formulation is continuous, the reference data is discrete. We are therefore forced to evaluate operators such as $g-\hat{g}$ in a discrete way. Correspondingly, the solutions to the equation (4.35) will live in the same discrete space as the reference data. Specifically, the basis vectors Φ will have the dimension of the reference data in space, while the mode amplitudes, α, have the dimension of the reference data in time. As a consequence, in the solution of (4.35) we require discrete operators for integration in space and time, and for differentiation in space the choice of these operators will be discussed in the results section which follows. In the next sections we will give results computed with the SCF for several different cases.

4.3 SCF Results

As mentioned in the previous section, the evaluation of integrals and derivatives in equation (4.35) must be carried out discretely. We choose to approximate the spatial derivatives by central differences and integrate over the domain by the trapezoidal rule as follows:

$$\int_0^1 f(x)g(x)dx \approx \sum_{i=1}^{N} \bar{f}_i \bar{g}_i h$$

where $\bar{f}_i$ and $\bar{g}_i$ represent the average values of the function f and g on each interval, $\bar{f} = \frac{f_i + f_{i+1}}{2}$, and h is the length of the subinterval of integration. This can be organised in matrix form, by taking $\mathbf{f} = [\bar{f}_1 \cdots \bar{f}_N]^T$, $\mathbf{g} = [\bar{g}_1 \cdots \bar{g}_N]^T$, and $\mathbf{\Delta}$ as a matrix containing only h on the diagonal. The integration therefore takes the form: $\int_0^1 f(x)g(x)dx \approx \bar{\mathbf{f}}^T \mathbf{\Delta} \bar{\mathbf{g}}$.

We are now in a position to compare the final discretisation arising from FDF and SCF. The n equations arising in (4.5) can also be expressed as:

$$\mathcal{M}\dot{\alpha} + \mathcal{K}\alpha = 0; \tag{4.36}$$

Where $\mathcal{M} = \bar{\Phi}^T \mathbf{\Delta} \bar{\Phi}$ and $\mathcal{K} = \bar{\Phi}'^T \mathbf{\Delta} \bar{\Phi}'$

In FDF, one might have constructed M and K using an arbitrary discretisation method for the governing equations. This might have been of high order, and included complex stabilisation operators, for example. In the case of experimental data, it might not have been possible to choose K and M at all. In contrast, in the SCF $\mathcal{M}$ and $\mathcal{K}$ are clearly associated with the process of integration and differentiation required for the reduced-order model. They can be chosen based on the final operating requirements of the ROM.

Looking more carefully at the M-matrices, we see that in case of SCF-ROM the matrix $\mathcal{M}$ is equal to the identity matrix. This because, by definition of orthonormal basis, the inner product $\int_0^1 \phi_i \phi_k dx = \delta_{ik}$ and therefore only terms on the diagonal ($i = k$) contribute. For BT-ROM the situation is slightly different, since $\hat{M}$ is dependent on the discretisation scheme chosen. If the numerical simulation is constructed by finite differences, the matrix M of the LTI formulation is also the identity matrix and therefore $\mathcal{M} = \hat{M} = I$. On the other hand, if a finite element approach is used, M is no longer identical and leads to a different result.

The following results will be generated by similar finite difference schemes for FDF and SCF, which results in a similar but not identical structure for the matrices M and

K. To solve equation (4.36) we need to supplement the equation by an appropriate initial condition. In our case we choose a triangular distribution as illustrated in Fig. 4.1. The reduced order model is integrated in time by a third-order explicit Runge-Kutta method. In Fig. 4.4 we compare the two formulations at a chosen instant of time t=0.025. As expected, the results from the FDF-ROM and SCF-ROM are similar to each other. For comparison, we consider the relative error as the number of modes increases, calculated as:

$$\epsilon = \frac{||U - U_r||_\infty}{||U_r||}$$

where U and U_r are the matrices containing the set of snapshots of the numerical solution and of the reduced-order model respectively. As we can see the difference is practically unnoticeable, although the continuous one is slightly lower.

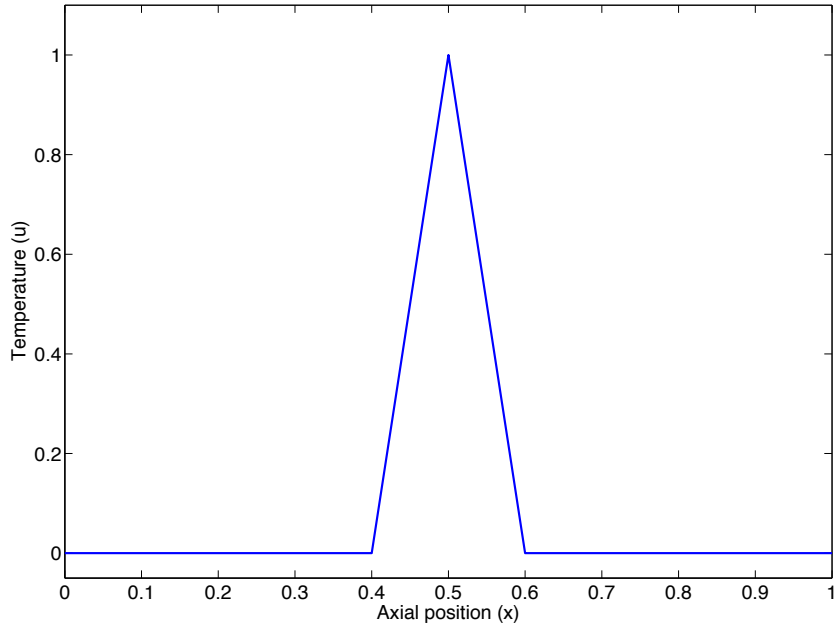


Figure 4.1: Initial condition for the Continuous POD

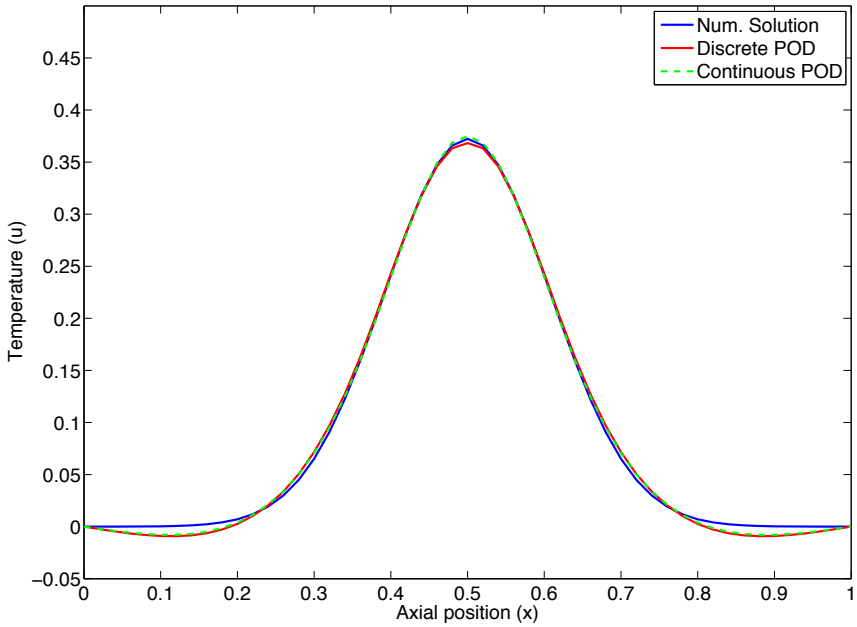


Figure 4.2: Comparison Discrete and Continuous POD of order 3 in a randomly chosen instant of time (t=0.0025)

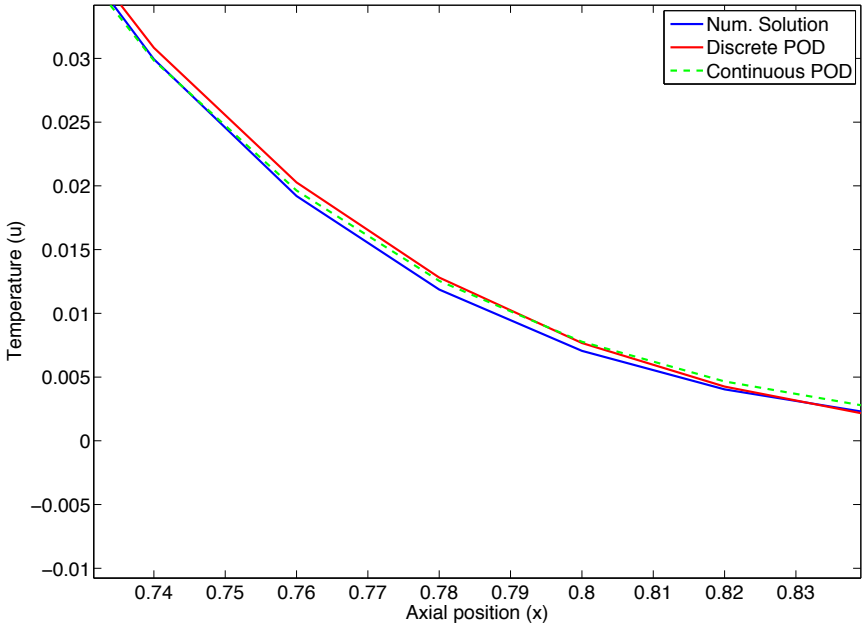


Figure 4.3: Zoom Fig. 4.4

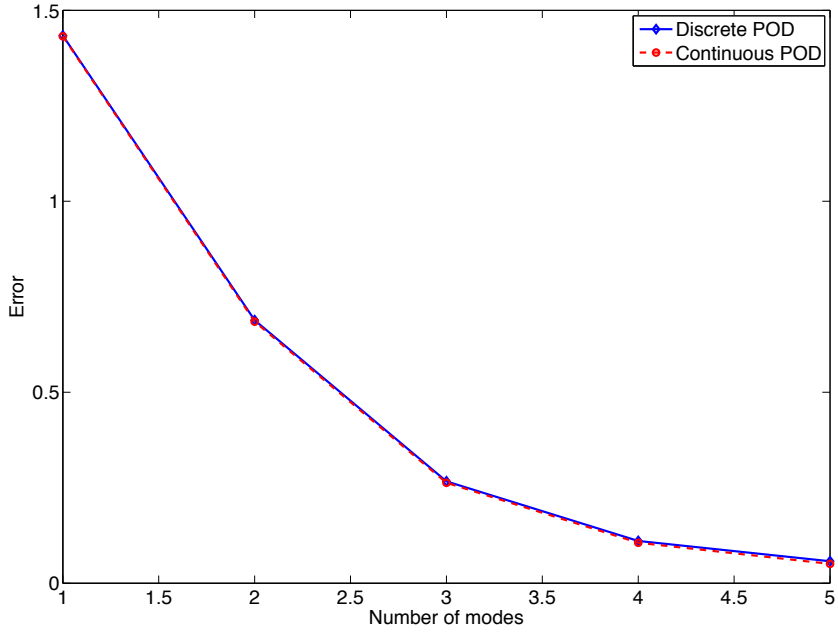


Figure 4.4: Comparison error continuous and discrete formulations

Convection Diffusion with NH-BC

In a similar fashion as the the heat equation, to implement the convection-diffusion equation we discretise the governing equations to allow numerical implementation. By applying the same differentiation and integration techniques we arrive at a similar result. Particularly, by defining the vector $\Phi_0 = [\phi_1(0), ..., \phi_n(0)]$, as well as $\Phi_0' = [\phi_1'(0), ..., \phi_n'(0)]$, also in this case we can set up the numerical counterpart, gathering the equations in matrix form for compactness. This gives:

$$\mathcal{M}\dot{\alpha} + (\mathcal{K}_a + \mathcal{K}_p)\alpha = \mathcal{b} \tag{4.37}$$

Where $\mathcal{M} = \bar{\Phi}^T \mathbf{\Delta} \bar{\Phi}$, $\mathcal{K}_a = -a(\bar{\Phi}'^T \mathbf{\Delta} \bar{\Phi}) + k(\bar{\Phi}'^T \mathbf{\Delta} \bar{\Phi}')$, $\mathcal{K}_p = k\Phi_0^T \Phi_0'$ and $\mathcal{b} = k\Phi_0^T u_0$.

To solve numerically (4.37) we employ a third order explicit Runge-Kutta method. At steady state the performance of the reduced-order model is shown in Fig. 4.5 for order 1 to 4 in case of the direct substitution method. Since the penalty method is strongly dependent of the σ value, it is worthwhile comparing how this affects the accuracy of the solution. As can be seen in Fig. 4.6 for the one mode case, the adoption of a high penalty term enforces the boundary conditions in a stronger manner, but the price to pay is a lack of optimality in the interior domain. On the other hand reducing σ to lower values, thus allowing bigger jumps at the boundary, shows a better representation of the boundary layer. Depending on one needs, the choice of σ plays an important role and should be ultimately be incorporated as an additional optimisation variable.

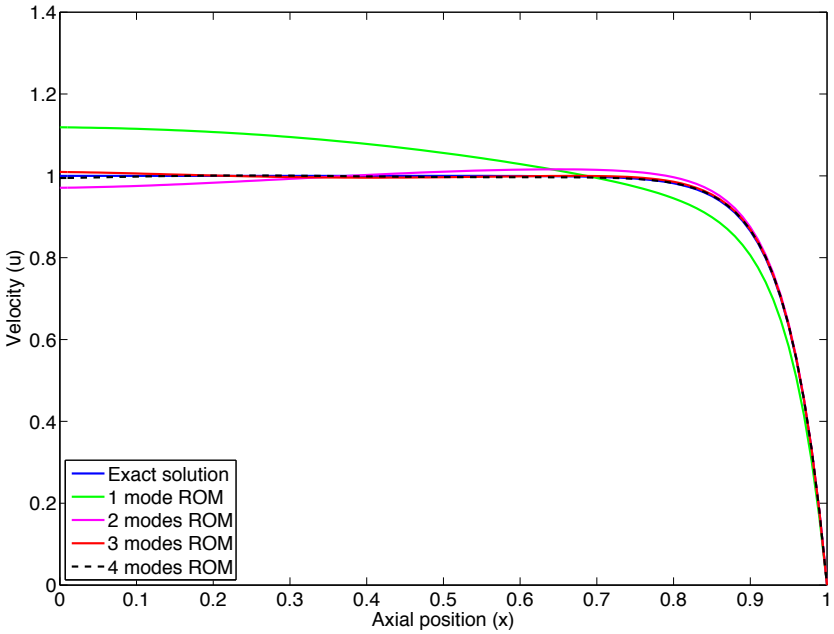


Figure 4.5: Steady state solution of convection diffusion equation. Comparison between different order of ROM

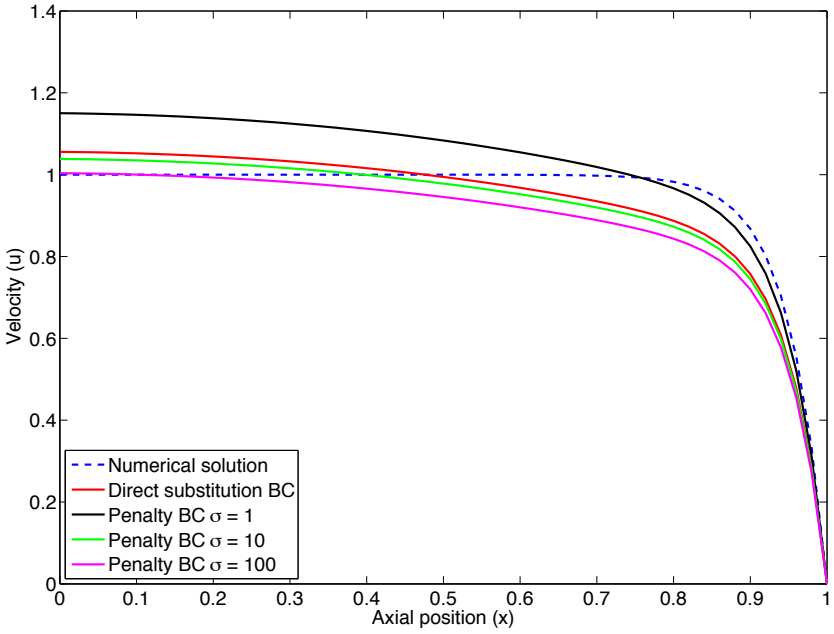


Figure 4.6: Comparison of different boundary condition implementation

4.4 Optimisation - Non linear functional

In a similar way as we did for the state equations, to implement the optimisation for the SCF derived in Sec. 4.2, adjoint equation and gradient expression need to be brought in matrix form. Since the goal-oriented term depends on the function we choose to optimise, for sake of clarity we take as example the identity function $f(u) = u$. The adjoint equation can be rewritten as:

$$-\mathcal{M}^T\dot{\lambda} + (\mathcal{K}_a{}^T + \mathcal{K}_p{}^T)\lambda = \mathcal{b} \tag{4.38}$$

With $\mathcal{M}, \mathcal{K}_a, \mathcal{K}_p$ as previously defined and $\mathcal{b} = \bar{\Phi}^T\mathbf{\Delta}\overline{(u - \Phi\alpha)}$

And similarly, in matrix form the gradient reads as:

$$\begin{aligned}\delta\mathscr{L}_\Phi = &\int_0^{t_f}(\Phi\alpha - u)\alpha^T dt + 2\beta\Phi(\bar{\Phi}^T\Delta\bar{\Phi} - I) + \int_0^{t_f}\big[\Phi(\lambda\dot{\alpha}^T + \dot{\alpha}\lambda^T)\\ &- a\Phi'(\lambda\alpha^T - \alpha\lambda^T) - k\Phi''(\lambda\alpha^T + \alpha\lambda^T) - au_0\lambda^T + k\Phi'\alpha\lambda^T\big]dt\\ &+ \Phi(\mu\alpha_0^T + \alpha_0\mu^T) - u_0\mu^T = 0\end{aligned} \tag{4.39}$$

It is worth contrasting (4.39) with expression (3.30) here recalled:

$$\begin{aligned}\delta\mathscr{L}_\Phi = &\int_0^{t_f} H(\Phi\alpha - u)\alpha^T dt + 2\beta\Phi(\Phi^T\Phi - I)\\ &+ \int_0^{t_f}[M\Phi(\lambda\dot{\alpha}^T + \dot{\alpha}\lambda^T) + K^T\Phi\lambda\alpha^T + K\Phi\alpha\lambda^T - f\lambda^T]dt\\ &+ M\Phi\mu\alpha_0^T + M(\Phi\alpha_0 - u_0)\mu^T = 0.\end{aligned} \tag{4.40}$$

The matrices M and K no longer appear. These have been replaced by expressions which correspond to the Galerkin projection of the continuous governing equations, and the goal-functional has assumed a general functional $g = f(u)$.

To validate the formulation we carried out a numerical test for the convection-diffusion equation in which we target the functional $g = ||u||^2$. As we can see in Fig. 4.7 for each mode we obtain a substantial improvement with respect to the POD-ROM. In addition, especially for the 1-mode case, we notice a considerable change in the mode shape, reflecting the adaptation of the system to the goal-orientation.

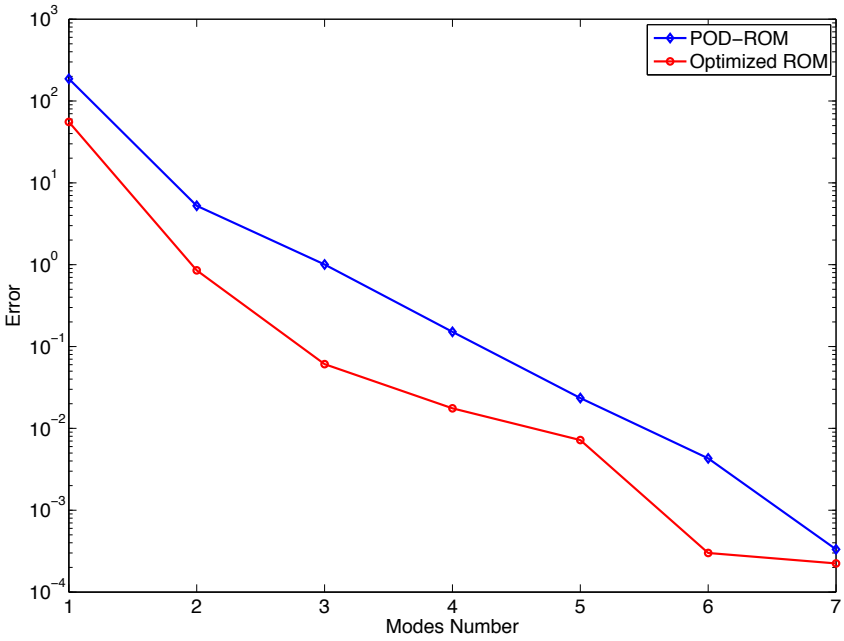


Figure 4.7: Comparison accuracy POD-ROM with optimised basis for $g = ||u||^2$

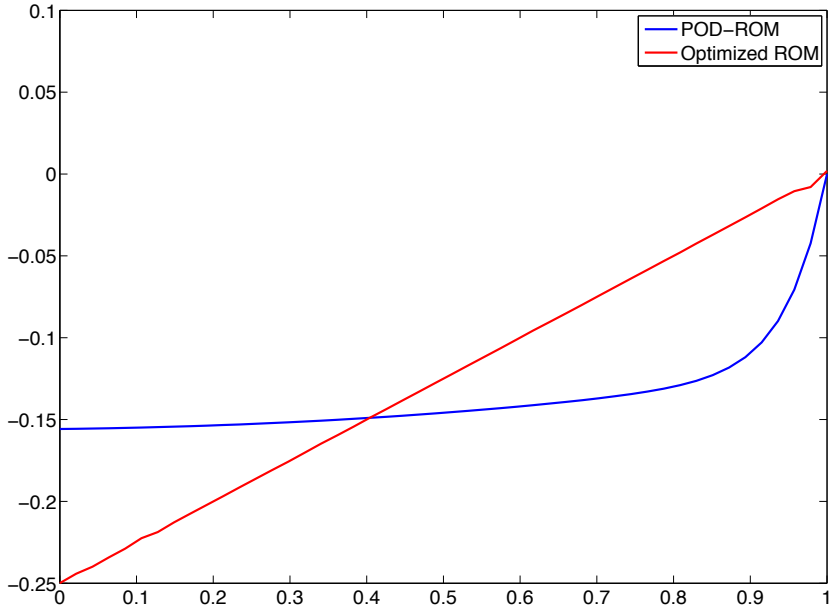


Figure 4.8: POD-mode and optimised-Mode for the 1-mode ROM with $g = ||u||^2$

Chapter 5

Conclusions

We have considered a number of approaches to the development of reduced order models. Firstly we described the commonly-applied POD approach and then reviewed the much more useful fully-discrete approach introduced by (7) which allows for model constraint and goal orientation. The FDF, however, is limited to LTI systems and linear functionals. To address these limitations, we have introduced a semi-continuous approach, which is completely independent of the method used to generate the reference data, avoiding coupling of sub-optimal defining matrices and being applicable in case of experimental data as well. Furthermore the approach is no longer limited to linear PDE and linear functionals, but any governing equation or goal-orientation can be considered.

We have designed the method to be applicable to large-scale simulations by using an inexact-globalised-Newton-method. The construction of the gradient by the adjoint method remains tractable, requiring exclusively state and adjoint solution of the ROM and not of the full system. However, depending on the size of the problem, the computational cost of the complete procedure might still represent an issue and further investigations are required. Furthermore, improvements could be achieved by including a goal-oriented adaptive ROM penalty term in the optimisation process.

Results have shown important improvements compared to the POD. We have considered the convection-diffusion equation with non-linear goal functional. The mode adaptation brought by the SCF showed improved ROM solutions, allowing a much better representation of the output of interest and clearly outperforming the POD-ROM.

The general framework provided by the method, together with encouraging results obtained for the linear convection-diffusion equation, make the SCF attractive for future

works, such as applications to nonlinear governing equations and implementation in control systems.

Bibliography

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

[2] K. Kunisch and S. Volkwein, "Control of the burgers equation by a reduced-order approach using proper orthogonal decomposition," *J. Optim. Theory Appl.*, vol. 102, no. 2, pp. 345–371, 1999.

[3] K. Kunisch and S. Volkwein, "Galerkin proper orthogonal decomposition methods for a general equation in fluid dynamics," *SIAM Journal on Numerical Analysis*, vol. 40, no. 2, pp. 492–515, 2003.

[4] B. F. Feeny and R. Kappagantu, "On the physical interpretation of proper orthogonal modes in vibrations," *Journal of Sound and Vibration*, vol. 211, no. 4, pp. 607 – 616, 1998.

[5] M. Bozkurttas and P. Madden, "Fish Pectoral Fin Hydrodynamics; Part III: Low Dimensional Models via POD Analysis," *APS Meeting Abstracts*, pp. A3+, Nov. 2005.

[6] R. W. Preisendorfer and C. D. Mobley, *Principal component analysis in meteorology and oceanography.* Amsterdam: Elsevier, 1988.

[7] T. Bui-Thanh, K. Willcox, O. Ghattas, and B. van Bloemen Waanders, "Goal-oriented, model-constrained optimization for reduction of large-scale systems," *Journal of Computational Physics*, vol. 224, no. 2, pp. 880 – 896, 2007.

[8] M. J. Forray, *Variational calculus in science and engineering.* McGraw-Hill New York, 1968.

[9] N. I. M. Gould, S. Lucidi, M. Roma, and T. P. L., "Solving the trust-region subproblem using the lanczos method," *SIAM Journal on Optimization*, vol. 9, no. 2, pp. 504–525, 1999.

[10] T. Steihaug, "The conjugate gradient method and trust regions in large scale optimization," *SIAM Journal on Numerical Analysis*, vol. 20, no. 3, pp. 626–637, 1983.

[11] A. R. Conn, N. I. M. Gould, and P. L. Toint, *Trust-region methods*. MPS-SIAM Series on Optimization 1, 2000.

[12] V. Akcelik, G. Biros, O. Ghattas, J. Hill, D. Keyes, and B. van Bloemen Waanders, "Parallel algorithms for pde-constrained optimization and application to optimal control of viscous flows," tech. rep., 2000.

[13] V. Akcelik, G. Biros, O. Ghattas, J. Hill, D. Keyes, and B. van Bloemen Waanders, "Parallel algorithms for pde-constrained optimization," *Frontiers of Parallel Computing*, 2006.

[14] J. Nocedal and S. J. Wright, *Numerical optimization, 2nd edition*. New York, Springer, 2006.

[15] D. N. Arnold, F. Brezzi, B. Cockburn, and D. L. Marini, "Unified analysis of discontinuous galerkin methods for elliptic problems," *SIAM J. Numer. Anal.*, vol. 39, no. 5, pp. 1749–1779, 2001.

[16] G. H. Golub and C. F. Van Loan, *Matrix computations (3rd ed.)*. Baltimore, MD, USA: Johns Hopkins University Press, 1996.

[17] R. G. Parr and W. Yang, *Density-Functional Theory of Atoms and Molecules*. Oxford University Press, 1989.

[18] M. Schechter, *An introduction to Nonlinear Analysis*. Cambridge University Press, 2004.

Appendix A

Mathematical definitions

A.1 Notions on vector spaces

- **Linear Independence:** A set of vectors $\{a_1, ..., a_n\}$ in $\mathbb{R}^m$ is *linearly independent* if the only solution of the problem $\sum_{j=1}^{n} \alpha_j a_j = 0$ is the zero solution $\alpha(1:n) = 0$. In other words the n vectors are linearly independent if the only linear combination equal to the 0-vector is the one with all coefficients equal to 0.

- **Subspace:** A *subspace* of $\mathbb{R}^m$ is a subset that is also a vector space.

- **Span:** Given a set of vectors $\{a_1, ..., a_n\}$ the set of all linear combinations of these vectors is a subspace referred to as the *span* of $\{a_1, ..., a_n\}$.

$$\text{span}\{a_1, ..., a_n\} = \left\{ \sum_{j=1}^{n} \beta_j a_j : \beta_j \in \mathbb{R} \right\}$$

- **Basis:** The subset $\{a_{i_1}, ..., a_{i_k}\}$ is called a *maximal linearly independent subset* of $\{a_1, ..., a_n\}$ if it is linearly independent and is not properly contained in any linearly independent subset of $\{a_1, ..., a_n\}$. Furthermore if $\{a_1, ..., a_n\}$ is maximal, the $\text{span}\{a_1, ..., a_n\} = \text{span}\{a_{i_1}, ..., a_{i_k}\}$ and $\{a_{i_1}, ..., a_{i_k}\}$ is called a *basis* for $\text{span}\{a_1, ..., a_n\}$.

- **Dimension:** If $S \subseteq \mathbb{R}^m$ is a subspace, then it is possible to find independent basic vectors $\{a_{i_1}, ..., a_{i_k}\}$ such that $S = \text{span}\{a_1, ..., a_k\}$. All bases for a subspace S have the same number of elements and this number is called the *dimension* of S, denoted by $\dim(S)$.

A.2 Range, Null Space and Rank

- **Range:** the *range* of a matrix A is defined by:

$$\text{range}(A) = \{y \in \mathbb{R}^m : y = Ax \text{ for some } x \in \mathbb{R}^n\}$$

  In other words the range of A is the set of all vectors v for which the equation $Ax = y$ has a solution. Still another equivalent definition: the range of A is the span of columns of A.

- **Ker or Null Space:** the *ker or null space* of a matrix A is defined by:

$$\ker(A) = \{x \in \mathbb{R}^n : Ax = 0\}$$

  In other words the ker of A is the set of all the solutions to the equation $Ax = 0$.

- **Rank:** the *rank* of a matrix A is defined by:

$$\text{rank}(A) = \dim(\text{range}(A))$$

A.3 Norms

Norms provide measures of distance. They are indicated with a double bar notation and subscripts to distinguish between various norms, the most common of which is the 2-norm.

- **Vector Norm:** a useful class of vector norms are defined by the p-norm:

$$||x||_p = (|x_1|^p + ...|x_n|^p)^{1/p} \quad p \geq 1$$

  In particular we distinguish:

$$\begin{aligned} \text{1-norm} \quad ||x||_1 &= |x_1| + ...|x_n|, \\ \text{2-norm} \quad ||x||_2 &= (|x_1|^2 + ...|x_n|^2)^{1/2} = (x^T x)^{1/2}, \\ \infty\text{-norm} \quad ||x||_\infty &= \max_{1 \leq i \leq n} |x_i|. \end{aligned}$$

- **Function Norm:** similarly to vector norms, for a function $f(x)$ we can define:

$$||f||_p = \left[\int_0^1 |f(x)|^p dx\right]^{1/p}$$

In particular we distinguish:

$$\begin{aligned} L_1\text{-norm} \quad ||f||_1 &= \int_0^1 |f(x)dx \\ L_2\text{-norm} \quad ||f||_2 &= \left[\int_0^1 |f(x)|^2 dx\right]^{1/2} \\ L_\infty\text{-norm} \quad ||f||_\infty &= \max_{0\leq x\leq 1} |f(x)|. \end{aligned}$$

- **Matrix Norm:** are defined using the vector norm above defined. The matrix p-norm reads as:

$$||A||_p = \sup_{x\neq 0} \frac{||Ax||_p}{||x||_p}$$

In other words the Matrix p-norm is the p-norm of the largest vector obtained by applying A to a unit (p-norm) vector.

$$||A||_p = \sup_{x\neq 0} \frac{||Ax||_p}{||x||_p} = \max_{||x||_p=1} ||Ax||_p$$

A.4 Orthogonality

- **Vectors:** a set of vectors $\{x_1, ..., x_p\}$ in $\mathbb{R}^m$ is *orthogonal* if $x_i^T x_j = 0$ whenever $i \neq j$ and *orthonormal* if $x_i^T x_j = \delta_{ij}$. So in particular orthonormality requires $||x_i^T x_j|| = 1$ whenever $i = j$.

- **Matrices:** a matrix $Q \in \mathbb{R}^m$ is said to be *orthogonal* if $Q^T Q = I$, where I is the identity matrix. Consequently for an orthogonal matrix $Q^T = Q^{-1}$. If Q is orthogonal, then the q_i form an orthonormal basis for $\mathbb{R}^m$.

Appendix B

Singular Value decomposition (SVD)

In 1965 G. Golub and W. Kahan introduced the Singular Value Decomposition (SVD) as a technique for calculating the singular values, pseudo-inverse and rank of a matrix. We begin by stating, without proof, the SVD theorem (interested readers can refer to (16)):

Theorem 1:
If A is a real m-by-n matrix, then there exist orthogonal matrices:

$$U = (u_1, ..., u_m) \in \mathbb{R}^{m \times m} \quad and \quad V = [v_1, ..., v_n] \in \mathbb{R}^{n \times n}$$

such that:

$$U^T AV = diag[\sigma_1, ..., \sigma_p] \in \mathbb{R}^{m \times n} \quad p = min\{m, n\} \tag{B.1}$$

where $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_p \geq 0$.

The σ_i are denoted as the singular values of A and the vectors u_i and v_i are the ith *left singular vector* and the ith *right singular vector*. There is an important relationship between the SVD of the the matrix A and the matrix itself:

- U is a matrix whose columns are the eigenvectors of the AA^T matrix, termed the left eigenvectors.
- V is a matrix whose columns are the eigenvectors of the $A^T A$ matrix, termed the right eigenvectors.

- the squares of the singular values are the $p = min(m, n)$ largest eigenvalues of AA^T or A^TA, since the p right and left eigenvalues are equivalent.

It is convenient to introduce the following notation to refer to singular values:

$$\begin{aligned} \sigma_i(A) &= \text{the } i\text{th largest singular value of A,} \\ \sigma_{max}(A) &= \text{the largest singular value of A,} \\ \sigma_{min}(A) &= \text{the smallest singular value of A.} \end{aligned}$$

If the SVD of A is given by Th. 1 and we define r as the index of the last positive singular value:

$$\sigma_1 \geq ... \geq \sigma_r \geq \sigma_{r+1} = ... = \sigma_p = 0,$$

then:

$$\text{rank}(A) = r \tag{B.2}$$

$$\ker(A) = \text{span}\{v_{r+1}, ..., v_n\} \tag{B.3}$$

$$\text{range}(A) = \text{span}\{u_1, ..., u_r\} \tag{B.4}$$

and we have the SVD expansion:

$$A = \sum_{i=1}^{r} \sigma_i u_i v_i^T$$

and in particular the matrix 2-norm is given by:

$$||A||_2 = \sigma_1$$

B.0.1 Geometrical interpretation

The SVD of a matrix has a nice geometric interpretation. An $n \times m$ matrix A is a linear operator that maps vectors from an m-dimensional space to an n-dimensional space. When applying a matrix to a vector, the latter can undergo rotation and scaling. As an example we describe the effect of a matrix on a two-dimensional vector.

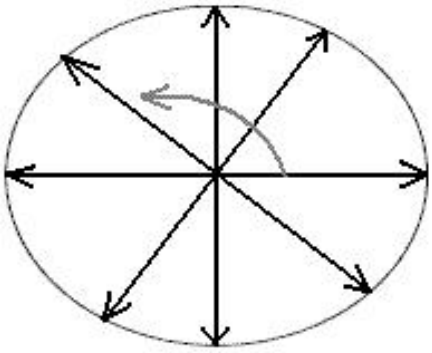


Figure B.1: Rotating Vector

Take a rotating vector, which describes a circle. Now suppose that when rotating the vector we stretch and squeeze it in a variable manner and up to a maximum and minimum length (application of the matrix). Instead of a circle the arrow now describes a two-dimensional ellipsoid. As said the multiplication of a vector x with a matrix A results in a new vector $Ax = y$ and the matrix performs two operations on the vector: rotation (the vector changes coordinates) and scaling (the length of the vector changes). In terms of SVD, the maximum stretching and squeezing are determined by the singular values of the matrix. In Fig.(B.2) the effect of the two largest singular values, s_1 and s_2 has been labeled. The important thing to notice is that singular values describe the extent by which the matrix modifies the original vector and, thus, can be used to highlight which dimension(s) is/are affected the most by the matrix. Evidently, the largest singular value

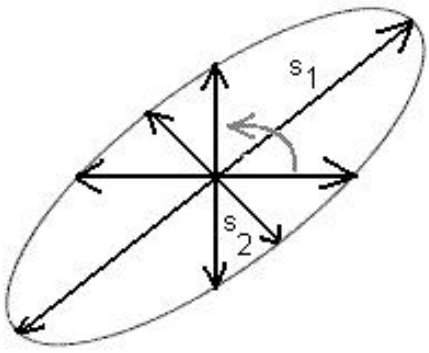


Figure B.2: Application of a matrix to a vector

has the greatest influence on the overall dimensionality of the ellipsoid.
This idea can be easily extended to an M-dimensional vector space. As we found, the greatest influence on the system is represented by the largest singular values. In fluid mechanics, by taking velocity measurements, the square of the singular values represent the kinetic energy of the associated modes, and therefore keeping the first singular values corresponds to keep the most energetic modes in the flow.

B.0.2 Why the SVD?

Recalling from the introduction, we are interested in constructing an approximation of the form $y(x,t) = \sum_{i=1}^{M} a_i(t)\phi_i(x)$ which approximates our data in the best possible way. This raises two questions: what is the relation between the SVD and the approximation? And why did we introduce the SVD?
Let's first answer the first of these questions. From Th. 1 we know that it is possible to decompose A such that $U^T AV = diag[\sigma_1, ..., \sigma_p]$. Furthermore we know that U and V are orthogonal matrices, implying that their inverses exist and are equal to their respective transposes. We can therefore decompose the matrix A in the following way:

$$A = U\Sigma V^T \tag{B.5}$$

where Σ is the diagonal matrix containing the singular values σ_i.
Now let $U\Sigma = Q$ in Eq.(B.5). The matrix Q is then $N \times M$ and $A = QV^T$. Taking q_i as the i-th column of Q and v_i the i-th column of V, we can write

$$A = QV^T = \sum_{i=1}^{M} q_i v_i^T. \tag{B.6}$$

In this form Eq.(B.6) can be referred as the discrete form of Eq.(2.1) where we recognize $y(x,t)$ to be the matrix A, $a_i(t)$ the column matrix q_i and $\phi_i(x)$ the row matrix v_i^T.

The answer to the second question requires the relationship between SVD and POD, which will be treated in the next section. However, we can have a first idea by looking more in detail at the matrix Σ containing the singular values. Σ is constructed in such a way that the singular values are placed in the main diagonal in descending order. The singular values are an indication of the energy carried by the associated mode. Therefore it is very easy to construct a lower-rank approximation of the matrix A, by choosing only

the first k values of σ_i, meaning to select the most energetic features in the system. For any $k < r$ the matrix Σ_k is obtained by setting $\sigma_{k+1} = \sigma_{k+2} = ... = \sigma_p = 0$.

Since the singular values σ_i are ordered in decreasing order along the diagonal of Σ and this ordering is preserved when constructing U and V^T, keeping the first k singular values is equivalent to keeping the first k rows of V^T, the first k columns of U and replacing Σ by the submatrix containing its first k rows and k columns. It follows that the the lower-rank approximation of A is

$$A_k = U_k \Sigma_k V_k^T \tag{B.7}$$

This process is termed dimensionality reduction, and A_k is referred to as the *Rank k* approximation of A or the "Reduced SVD" of A. Why is this dimensionality reduction useful will become more clear in the next section, where the POD and the concept of optimality will be introduced.

Appendix C

SVD vs. Eigenvalue decompostion (ED)

The SVD of a matrix has several advantages with respect to the Eigenvalue decomposition. First of all the SVD can be computed for non-square matrices, while the ED cannot since it requires the evaluation of the inverse of a non-orthogonal, non-square matrix, which is not defined. Furthermore the SVD remains real when A is real, while eigenvalues and eigenvectors of unsymmetric real matrices can be complex valued. Last but not least, the left and right singular vectors (columns of U and of V respectively) are each orthogonal, while eigenvectors of unsymmetric matrices need not to be orthogonal. The SVD allows therefore the choice of an optimal orthonormal basis (rows of V_i^T), while the ED does not. However a strong connection between the two methods exist. As described in appendix B we know that U is a matrix whose columns are the eigenvectors of the AA^T matrix, V is a matrix whose columns are the eigenvectors of the A^TA matrix and the squares of the singular values are the $r = min(n, m)$ largest eigenvalues of AA^T. If A is symmetric and positive definite, then its eigenvalues are also its singular values, and $U = V$. If A is symmetric with some negative eigenvalues (they will be real, because A is symmetric), then the singular values are the magnitudes of the eigenvalues, and U and V are the same up to multiplication by minus one for the columns corresponding to negative eigenvalues.

Appendix D

The conjugate gradient method

The conjugate gradient method was introduced by Hestenes and Stiefel in the 1950's as a technique to solve large sparse systems of linear equations in the form $Ax = b$ with A a n-by-n positive definite coefficient matrix. The problem can also be seen equivalently as the minimization problem:

$$\min \phi(x) = \frac{1}{2}x^T Ax - b^T x + c \tag{D.1}$$

In fact solution of Eq.(D.1) is obtained by setting $\phi'(x) = 0$ and therefore retrieving:

$$\phi'(x) = \frac{1}{2}A^T x + \frac{1}{2}Ax - b = Ax - b = 0$$

since A is symmetric. This allows us to see the conjugate-gradient method either as an algorithm for solving linear systems or as a technique for minimizing convex quadratic functions.

Since it is more easily understood in terms of optimization, we pose the problem as the determination of the minima of the quadratic function (D.1). As previously said, starting at an initial point, we choose a search direction and we require that the new iterate satisfies $f(x_{k+1}) < f(x_k)$. The search direction can be chosen in several different ways, the most obvious would be the direction of steepest descent, corresponding to the negative of the gradient at x_k. This choice, however, is generally inefficient since the algorithm finds itself taking steps in the same direction as earlier steps. The conjugate-gradient method overcomes this drawback by using a set of search directions with the property of *conjugacy*. Two vectors are said to be conjugate (or A-orthogonal), if

$$d_i^T A d_j = 0 \quad \text{for all } i \neq j$$

Before illustrating the method, we introduce the following notation

- Iterate: we choose the new point to be $x_{k+1} = x_k + \alpha_k d_k$, with α_k the step length and d_k the search direction.
- Error: the difference between the value of the current iterate x_k and the exact solution x, namely $e_k = x_k - x$.
- Residual: indicates how well the current iterate approximates the solution, namely $r_k = Ax_k - b$. It is important to notice that the residual is also equal to $r_k = \phi'(x_k)$, which represents the gradient at the current iterate. Furthermore with a simple substitution we can also write the residual as $r_k = Ae_k$.

Consider now the contours of a given function (Fig. D.1), in which x_0 represents the initial guess, d_0 the first search direction, x_1 the new iterate and e_1 the error after the first step.

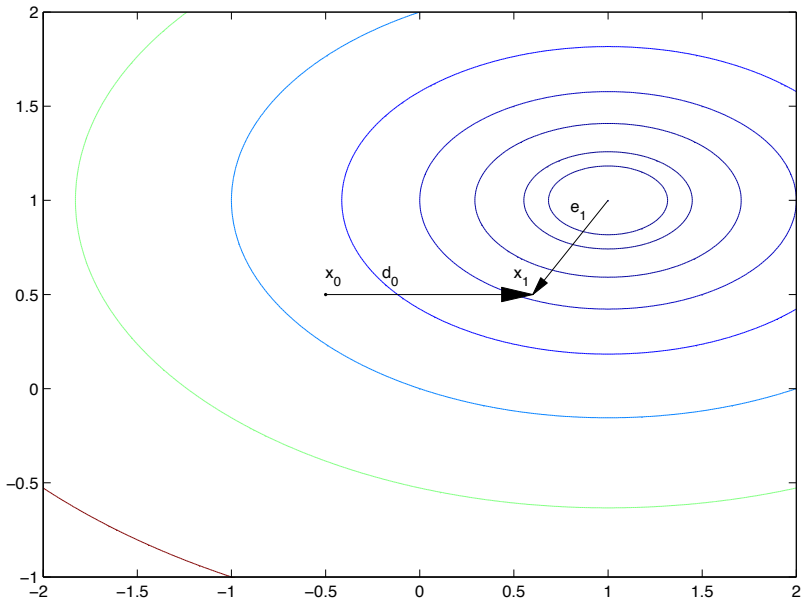


Figure D.1: CG iteration

What we require is that the error e_{k+1} has to be A-orthogonal to the previous search direction, hence $d_k^T A e_{k+1} = 0$. This is equivalent to finding the minimum point along the search direction d_k. In fact, setting the directional derivative of $\phi(x_{k+1})$ with respect to α_k to zero we have:

$$\frac{d}{d\alpha}\phi(x_{k+1}) = 0 \tag{D.2}$$

$$\phi'(x_{k+1})^T \frac{d}{d\alpha} x_{k+1} = 0 \tag{D.3}$$

$$r_{k+1}^T d_k = 0 \tag{D.4}$$

$$d_k^T A e_{k+1} = 0 \tag{D.5}$$

with this condition α_k can be calculated as:

$$d_k^T A e_{k+1} = 0 \tag{D.6}$$

$$d_k^T A(e_k + \alpha_k d_k) = 0 \tag{D.7}$$

$$\alpha_k = -\frac{d_k^T A e_k}{d_k^T A d_k} = -\frac{d_k^T r_k}{d_k^T A d_k} \tag{D.8}$$

To demonstrate that this procedure actually works, we can express the error e_0 as a linear combination of search directions, and show that after n iterations all components of the error term have been cut away. Writing the error term as:

$$e_0 = \sum_{k=0}^{n-1} \delta_k d_k \tag{D.9}$$

we can find the values of δ_k by exploiting the conjugacy property of the search directions. Upon premultiplying Eq.(D.9) by $d_j^T A$ we get:

$$d_j^T A e_0 = \sum_{k=0}^{n-1} \delta_k d_j^T A d_k \tag{D.10}$$

$$= \delta_j d_j^T A d_j \tag{D.11}$$

since $d_j^T A d_k = 0$ for all $j \neq k$. Each δ_j can then be rearranged as:

$$\delta_j = \frac{d_j^T A e_0}{d_j^T A d_j} \tag{D.12}$$

$$= \frac{d_j^T A(e_0 + \sum_{i=0}^{j-1} \alpha_i d_i)}{d_j^T A d_j} \tag{D.13}$$

$$= \frac{d_j^T A e_j}{d_j^T A d_j} \tag{D.14}$$

where we are allowed to add $\sum_{i=0}^{j-1} \alpha_i d_i$ since by conjugacy this is equal to zero. From the result in Eq.(D.14) we notice that $\alpha_j = -\delta_j$. Since the error term at a given iteration can be seen as the initial error e_0 plus the progress already made by the algorithm (Fig. D.2), we can rearrange the expression as:

$$e_i = e_0 + \sum_{k=0}^{i-1} \alpha_k d_k \tag{D.15}$$

$$= \sum_{k=0}^{n-1} \delta_k d_k - \sum_{k=0}^{i-1} \delta_k d_k \tag{D.16}$$

$$= \sum_{k=i}^{n-1} \delta_k d_k \tag{D.17}$$

$$\tag{D.18}$$

Therefore after n iterations every component of the error vanishes and $e_n = 0$, which proves the effectiveness of the procedure.

The only thing now missing is a procedure for computing the next search direction. This turns out to be a very special feature of the conjugate-gradient method, because it can generate a new search direction only with information regarding the previous search direction d_k and residual r_k, avoiding storage of the whole history of data. Other methods, such as the conjugate Gram-Schmidt process require all the old search vectors to be kept in memory, leading them to be computationally expensive.

To construct a new direction, d_{k+1}, we take:

$$d_{k+1} = -r_{k+1} + \beta_{k+1} d_k \tag{D.19}$$

where the scalar β_{k+1} is determined by imposing d_{k+1} and d_k to be conjugate with respect to A. By premultiplying (D.19) by $d_k^T A$ and imposing the condition $d_k^T A d_{k+1} = 0$ we obtain:

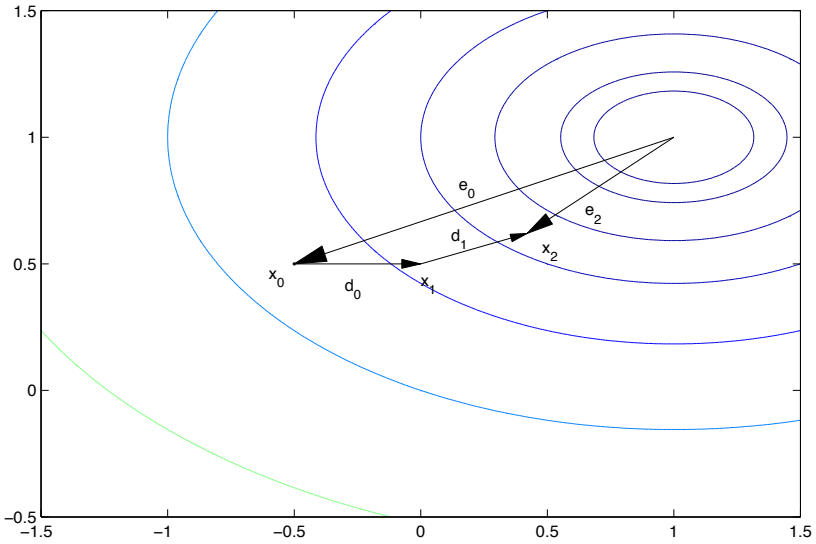


Figure D.2: Error at different iterations: $e_2 = e_0 + \alpha_0 d_0 + \alpha_1 d_1 = e_0 + \sum_{k=0}^{1} \alpha_k d_k$

$$d_k^T A r_{k+1} = \beta_{k+1} d_k^T A d_k \tag{D.20}$$

$$\beta_{k+1} = \frac{d_k^T A r_{k+1}}{d_k^T A d_k} \tag{D.21}$$

The method would be complete as defined above, but with a little modification we can simplify the computations a bit. If we premultiply (D.17) by $d_j^T A$ we have:

$$d_j^T A e_i = \sum_{k=i}^{n-1} \delta_k d_i^T A d_k \tag{D.22}$$

$$d_j^T r_i = 0 \quad \text{for all} j < i \tag{D.23}$$

Which shows that the residual is orthogonal to every previous search direction. Now by premultiplying Eq.(D.19) by r_{k+1}^T, and using the orthogonality property of (D.23) we have:

$$r_{k+1}^T d_{k+1} = -r_{k+1}^T r_{k+1} + \beta_{k+1} r_{k+1}^T d_k \tag{D.24}$$

$$r_{k+1}^T d_{k+1} = -r_{k+1}^T r_{k+1} \tag{D.25}$$

Hence at each iteration we have that $-r_k^T d_k = r_k^T r_k$ and therefore α_k in Eq.(D.8) can be rewritten as:

$$\alpha_k = \frac{r_k^T r_k}{d_k^T A d_k} \tag{D.26}$$

As a last step we can simplify the evaluation of β by, first noticing that:

$$r_{k+1} = A e_{k+1} \tag{D.27}$$
$$= A(e_i + \alpha_k d_k) \tag{D.28}$$
$$= r_k + \alpha_k A d_k \tag{D.29}$$

and secondly, in the same fashion as we did for α, we premultiply (D.29) by r_{k+1}^T and we use orthogonality condition (D.23). This leads to:

$$r_{k+1}^T r_{k+1} = r_{k+1}^T r_k + \alpha_k r_{k+1}^T A d_k \tag{D.30}$$
$$r_{k+1}^T r_{k+1} = r_{k+1}^T r_{k+1} \tag{D.31}$$

And therefore the expression for β can be simplified as:

$$\beta_{k+1} = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k} \tag{D.32}$$

which completes the required calculations. Setting the first direction to be the negative of the residual and putting all together, the Conjugate gradient method can be summarized as:

Algorithm 3 - Conjugate Gradient

Set initial residual and search direction: $r_0 = Ax_0 - b$, $d_0 = -r_0$

while $||r_k|| > \epsilon$ **do**

- Calculate $\alpha_k = r_k^T r_k / d_k^T A d_k$
- Update x: $x_{k+1} = x_k + \alpha_k d_k$
- Update r: $r_{k+1} = r_k + \alpha_k A d_k$
- Calculate $\beta_{k+1} = r_{k+1}^T r_{k+1} / r_k^T r_k$
- Update d: $d_{k+1} = -r_{k+1} + \beta_{k+1} d_k$

end while

Appendix E

Functional derivative

The derivation of the gradient by means of the adjoint-method requires the use of functional derivatives, i.e. the differentiation of a functional with respect to its argument. In this section we give a brief overview of the concept, stating the main properties. For a detailed analysis see (17; 18)

Functional: A functional $F[\phi]$ is defined as a mapping from a normed linear space of functions (a Banach space) $M = \{\phi(x) : x \in \mathbb{R}\}$ to the field of real or complex numbers, $F : M \to \mathbb{R}$ or $\mathbb{C}$.

Differential: The differential of a functional is the part of the difference $F[f + \delta f] - F[f]$ that depends of δf linearly. Each $\delta f(x)$ may contribute to this difference, and for very small δf we write:

$$\delta f = \int \frac{\delta F[\phi]}{\delta f(x)} \delta f(x) dx$$

where the quantity $\delta F[\phi]/\delta f(x)$ is the functional derivative of F with respect to f at the point x.

Functional derivative: The functional derivative (also known as Fréchet derivative) is the quantity $\delta F[\phi]/\delta\phi(x)$ measuring how the value of the functional changes if the function $\phi(x)$ is changed at the point x. In other words, $\delta F[\phi]/\delta\phi(x)$ is explicitly defined by the process:

$$\lim_{\epsilon \to 0} \left[\frac{F[f + \epsilon\phi] - F[f]}{\epsilon} \right] = \left\{ \frac{d}{d\epsilon} (F[f + \epsilon\phi]) \right\}_{\epsilon=0} = \int \frac{\delta F}{\delta f(x)} \phi(x) dx$$

where $\phi(x)$ is arbitrary.

Most of the rules of ordinary differential calculus also apply to functional derivatives.

- Product of two functionals $F[\phi] = G[\phi]H[\phi]$

$$\frac{\delta F[\phi]}{\delta \phi(x)} = \frac{\delta G[\phi]}{\delta \phi(x)} H[\phi] = G[\phi] \frac{\delta H[\phi]}{\delta \phi(x)}$$

- Functional of a functional $F[G[\phi]]$

$$\frac{\delta F[G[\phi]]}{\delta \phi(y)} = \int \frac{\delta F[G]}{\delta G(x)} \frac{\delta G[\phi]}{\delta \phi(y)} dx$$

  In the special case $g(\phi)$ is an ordinary function, we have a similar expression but the integral disappears. Hence we have:

$$\frac{\delta F[g(\phi)]}{\delta \phi(y)} = \frac{\delta F[g]}{\delta g(\phi(y))} \frac{\delta g(\phi)}{\delta \phi(y)}$$

Acknowledgements

I offer my sincerest gratitude to my supervisor Dr. Steven Hulshoff for his support and guidance throughout the realisation of this project. His kindness and patience have always accompanied me, particularly in the tough moments of this work. Moreover, I am really grateful to him for always showing interest in my future, helping me choosing the right path and encouraging me in my career.

I am thankful to ir. Peter W. Fick. His knowledge and suggestions have been of great help for the understanding of the subject. I also would like to thank the committee members, Prof. dr. ir. drs. H. Bijl, Dr. R.P. Dwight and Dr. ir. A.S.J. Suiker for accepting to be part of the exam commission and for their interest in my work.

I would like to show my gratitude to my closest friends. Particularly many thanks to my dear friend Federico, with whom I have shared some of the best moments in my life and however far away he has always supported me closely. Domenico, because I know that you are always with me. Sergio for his enthusiasm that has encouraged me many times. Enrico for always making me dream. Remziye for her understanding and the fantastic friendship we share.

I am heartily thankful to my lovely girlfriend Ines. Her love and tenderness have been vital for me in the last year, encouraging me and making me smile even in the most difficult moments. A special thanks goes to my uncle Silvano, who has always treated me like a son, giving me the strength to give my best at any time. I am grateful to Manuel and Kenneth, for the immense love they always showed me.

I owe sincere and earnest thankfulness to my family for all their love and encouragement, Mariuccia, Guido, Gina and my father Aldo for believing in me during the whole time of my studies.

My deepest gratitude goes to my mother Iris, for allowing me to study and because she made everything possible. Without her help and love I could never have achieved this important result.

Finally, I would like to remember my uncle Tarcisio and my grandfather Riccardo, that will always be in my heart and with whom I would have loved to share this moment.