A semi-continuous formulation for goal-oriented reduced-order models

Cheng, Lei

DOI
10.4233/uuid:861b26cf-499d-4038-af2e-a29641f84ada

Publication date
2017

Document Version
Final published version

Citation (APA)
Cheng, L. (2017). *A semi-continuous formulation for goal-oriented reduced-order models*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:861b26cf-499d-4038-af2e-a29641f84ada

A Semi-Continuous Formulation for Goal-Oriented Reduced-Order Models

A Semi-Continuous Formulation for Goal-Oriented Reduced-Order Models

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. ir. K.C.A.M. Luyben,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op
vrijdag 15 december 2017 om 10:00 uur

door

Lei Cheng

Master of Engineering,
Northwestern Polytechnical University, Xi'an, China
geboren te Hebei province, China.

Dit proefschrift is goedgekeurd door de

Promotor: Prof. dr. ir. drs. H. Bijl
Copromotor: Dr. S. J. Hulshoff

Samenstelling promotiecommissie:

Rector Magnificus,	voorzitter
Prof. dr. ir. drs. H. Bijl,	Technische Universiteit Delft, promotor
Dr. S. J. Hulshoff,	Technische Universiteit Delft, copromotor

Onafhankelijke leden:

Prof. dr. ir. E. H. van Brummelen,	Technische Universiteit Eindhoven
Prof. dr. K. Morgan,	Swansea University
Prof. dr. S. Hickel,	Technische Universiteit Delft
Dr. M. Bergmann,	Université Bordeaux
Dr. ir. F. Fang,	Imperial College
Prof. dr. ir. L. J. Sluys,	Technische Universiteit Delft, reservelid

Printed by: Rijna Repro

Front & Back: Lei Cheng

ISBN 978-94-6186-876-3

An electronic version of this dissertation is available at
http://repository.tudelft.nl/.

SUMMARY

Modern computational and experimental techniques can represent the detailed dynamics of complex systems using large numbers of degrees of freedom. To facilitate human interpretation or the optimal design of control systems, however, reduced-order models (ROMs) are required. Conventional reduced-order modeling techniques, such as those based on proper orthogonal decomposition (POD), balanced proper orthogonal decomposition (BPOD), and dynamic mode decomposition (DMD), are purely data-driven. That is, the governing equations are not taken into account when determining the solution basis of the ROM. The resulting ROMs are thus sub-optimal, particularly when low numbers of degrees of freedom are used. Bui-Thanh *et al.* addressed this problem by determining ROM solution bases using a goal-oriented optimization procedure that seeks to minimize the error between the full and reduced-order goal functionals with the reduced-order model as a constraint. However, several issues limit the application of this approach. First, it requires explicit input matrices with the dimension of the reference data that result from spatially discretizing the governing equations. In addition, its derivation is restricted to linear governing equations and goal functionals. To overcome these limitations, our research group has proposed an alternative, a semi-continuous formulation (SCF), in which the ROM constraint and the optimization process are defined in a continuous setting. In this thesis, the mathematical framework of the SCF is illustrated, as is the algorithm used to solve the optimization problem.

The SCF is first demonstrated using the one-dimensional linear advection-diffusion equation. We then apply the SCF to one-dimensional Burgers equation to clarify the treatment of nonlinear governing equations. In these two applications, we investigate various goal functionals including linear and nonlinear functions. The results have shown that substantial improvements in absolute accuracy over the conventional POD ROM are achieved. We also demonstrate that it is possible to have multiple extrema in the optimization problem and discuss how the resulting ambiguities can be avoided. Finally, we demonstrate that when applying the SCF to a partial differential equation that only approximates the dynamics of the reference dataset, the beneficial effect of the model constraint in determining an optimal projection basis is large.

For applications of the SCF to two-dimensional problems, we consider the Stokes equations. In this application, we use a vector-valued projection basis where the projection basis functions for each variable (the velocity and pressure) are taken together as a single vector. The results clearly demonstrate the viability of the SCF in determining an optimal projection basis for a specific goal functional in multi-dimensional multivariable problems.

SAMENVATTING

Moderne rekenkundige en experimentele methodes kunnen, door gebruik te maken van een groot aantal vrijheidsgraden, een gedetailleerde representatie geven van de dynamica van complexe systemen. Echter, om menselijke interpretatie of optimaal ontwerp van een controle systeem te bevorderen, zijn modellen met een gereduceerde orde (ROMs) noodzakelijk. Conventionele methodes voor het construeren van een gereduceerd orde model, zoals methodes gebaseerd op een passend loodrechte decompositie (POD), een gebalanceerde, passend loodrechte decompositie (BPOD) of een dynamische modus decompositie (DMD), zijn puur gebaseerd op verkregen data. Dit betekent dat de beschrijvende vergelijkingen niet in acht worden genomen bij het bepalen van basis oplossingen van de ROM. De resulterende ROM is dus sub-optimaal, vooral wanneer slechts een laag aantal vrijheidsgraden wordt toegestaan. Bui-Thanh *et al.* hebben dit nadeel aangepakt door de basis oplossingen te bepalen met behulp van een doelgerichte optimalisatie procedure, die het verschil in de doelfunctie van het volledige en gereduceerde model minimaliseert, waarbij het gereduceerde orde model als restrictie geldt. Echter, een aantal zaken beperkt nog de toepassing van deze methode. Om te beginnen heeft deze methode een expliciete input matrix nodig met een dimensie zo groot als de referentie data die volgt uit de ruimtelijke discretisatie van de beschrijvende vergelijkingen. Verder is de afleiding van de methode slechts geldig voor lineaire beschrijvende vergelijkingen en doelfuncties. Om deze beperkingen het hoofd te bieden stelt onze onderzoeksgroep een alternatieve, semi-continue formulering (SCF) voor, welke de restrictie van de ROM en de optimalisatie opstelt in een continue formulering. In dit proefschrift wordt zowel het mathematische kader van de SCF geschetst, als het algoritme toegelicht om het optimalisatie probleem op te lossen.

De SCF wordt eerst gedemonstreerd op een één-dimensionale, lineaire, advectie-diffusie vergelijking. Om de behandeling van niet-lineaire beschrijvende vergelijking toe te lichten wordt de SCF vervolgens op een één-dimensionale Burgers vergelijking toegepast. Bij deze twee toepassingen worden verscheidene doelfuncties, zowel lineaire als niet-lineaire varianten, onderzocht. De resultaten tonen aan dat aanzienlijke verbetering in absolute nauwkeurigheid wordt verkregen in vergelijk met conventionele POD ROM. Tevens wordt getoond dat in de optimalisatie meerdere extremen mogelijk zijn en wordt beargumenteerd op welke manier deze ambivalenties kunnen worden vermeden. Tot slot wordt gedemonstreerd dat de SCF, bij toepassing met een partieel differentiaal vergelijking die de dynamica dan de referentie dataset slechts benadert, het gebruik van het model als restrictie een groot voordeel oplevert bij het bepalen van de optimale, geprojecteerde basis.

Als toepassing van SCF op een twee-dimensionaal probleem is gekozen voor de Stokes vergelijkingen. In deze toepassing wordt een vector-waarde projectie basis gebruikt, waarbij de basis functies van de projectie voor elke variabele (snelheid en druk) worden samengevoegd tot één enkele vector. De toepasbaarheid van SCF voor het bepalen van

de optimale projectie basis voor een specifieke doelfunctie in een multi-dimensionaal probleem wordt overtuigend ondersteund door de resultaten.

CONTENTS

1

INTRODUCTION

Everything should be made as simple as possible, but not simpler.

A. Einstein

1

1.1. BACKGROUND

Many physical systems encountered in science and engineering can be described by partial differential equations. The development of discretization methods for partial differential equations, combined with advances in computer hardware, have made the simulation of nonlinear physical behaviors in three space dimensions relatively commonplace. Problems with millions of numbers of degrees of freedom can be routinely simulated, thereby allowing researchers to study very complex phenomena.

However, in general, the numerical simulation of partial differential equations is insufficient in itself, due to the following two aspects:

1. While numerical simulation can provide high-fidelity data (i.e. detailed time histories of refined discretized solutions), researchers may not necessarily obtain an increased level of understanding concerning the physics essential to a given phenomenon from this data. As is true of experiment, careful analysis of the data is required to develop simpler models that can be used to predict the key characteristics of system behavior. This process can be hampered by the enormous size of the data.

2. The increasing need for improved accuracy requires the inclusion of more detail in the modeling stage, leading inevitably to larger-scale, more complex partial differential equations of systems. In some problems, such as design optimization or uncertainty quantification, we need to simulate many different possible realizations. Performing multiple simulations in such large-scale settings often leads to unmanageably large demands on computational resources. Thus, there is a fundamental gap between the analysis fidelity available to simulate an individual case and that practical for multi-disciplinary analysis.

Both of these aspects have motivated the development of low-order models of complex systems that can serve as the basis for additional analysis. A low-order model is a characterization of the physical processes of the original system, such that the essential behaviors of the system are captured with a relatively small number of degrees of freedom. Reduced-Order Models (ROMs) are low-order models derived from an appropriate projection of the original full system to a much smaller basis set consisting of basis functions that can encapsulate most, if not all, of the system's fundamental dynamics. Therefore, by constructing ROMs, we can

- Provide the means by which system dynamics can be readily interpreted.
- Provide quantitative accurate descriptions of the dynamics of systems at a computational cost much lower than the original numerical model.

The accuracy of a reduced-order model typically depends on the number of retained degrees of freedom and the convergence properties of the ROM. In general, reductions in computational cost needed of the ROM must be traded off with potential losses in accuracy and model robustness.

A number of reduced-order modeling techniques are reviewed here (see also [1–7]). The most popular reduced-order modeling technique is proper orthogonal decomposition (POD). It was introduced independently by several scientists [8], including Karhunen,

Loéve, Kosambi, Obukhov and Pougachev. The POD is also known as Karhunen-Loève decomposition (KLD), principal component analysis (PCA), and singular value decomposition (SVD). For a detail discussion about the equivalence of the POD, KLD, PCA, and SVD, see references [9, 10]. For the connection between POD and SVD, see the reference [11]. The POD yields an orthogonal basis for a known ensemble of experimental data or numerical solutions that are referred to as the reference data. The orthogonal basis is optimal in the sense that it maximizes the projection of the ensemble onto the basis, in other words, the error introduced by projecting the ensemble onto a subspace spanned by the truncated set of POD basis functions is minimized in a least-squares sense. This optimality makes POD very attractive and effective when performing data analysis and compression.

The POD has successfully been applied in various fields. In fluid mechanics, the POD was first introduced in the context of turbulence by Lumley [12] in 1967 as an objective definition of coherent structures. In the same year, Backewell and Lumley [13] applied the POD to the analysis of turbulent flow data obtained from experiments. Since then, the POD has been widely employed to characterize coherent structures of wall-bounded flows and free shear flows using experimental data and detailed numerical solutions [14–22]. For early applications of the POD in other disciplines, including random variables, image processing, signal analysis, data compression, process identification and control (see the textbook [23] written by Berkooz *et al.*).

Combined with a projection framework, the POD provides a method for the generation of low-dimensional models of complex dynamic systems. Reduced-order modeling by POD is based on projecting the governing partial differential equations onto a subspace spanned by the POD basis functions (using e.g. Galerkin projection) yielding to low-dimensional sets of ordinary differential equations. The models resulting from this process will be referred to here as the proper orthogonal decomposition reduced-order models (POD ROMs). In fluid dynamics, Aubry *et al.* [24] studied the near-wall evolution of the flow within a turbulent boundary layer using the POD ROM, where the neglected POD basis functions were modeled using a Smagoringsky-type sub-grid-scale model. Their POD ROM exhibited intermittent features reminiscent of those found in experimental flows. References [8, 23, 25] provide reviews of similar POD ROMs for turbulence. A tutorial on using the proper orthogonal decomposition to construct low-dimensional models for turbulent flows is given by Smith *et al.* [26], where the key steps of the analysis are explicitly described. A detailed illustration of how the POD is employed to derive ROMs for fluid flows through Galerkin projection can be found in the reference [27]. As the POD ROM allows for efficient approximate simulations of high-dimensional dynamic systems, it provides a computationally tractable method for optimization and control problems, e.g. see [28–36]. For applications of POD in other fields, for example, structural dynamics, fluid-structure interaction and aeroelastic systems, the interested reader is referred to [37–39].

In the last decades, scientists have developed alternatives to POD ROM, which have advantages in some aspects. One of these is balanced proper orthogonal decomposition (BPOD), introduced in 2005 by Rowley [40]. The BPOD is a modal decomposition technique that extracts two sets of modes for specified inputs and outputs, and forms a combination between balanced truncation and POD. Balanced truncation is a well-

known method in the control theory community. It was introduced by Moore [41] for model reduction of linear input-output systems, and Lall *et al.* [42, 43] who extended it to nonlinear systems. Most notably, balanced truncation has error bounds that are close to the lowest error possible from any reduced-order model. However, balanced truncation becomes computationally intractable for very large dimensional systems. This is why BPOD comes in. BPOD uses an output projection to reduce the number of necessary adjoint simulations, which results in a computationally feasible approximation to balanced truncation. BPOD has been successfully applied to various problems, e.g., see [44–49].

Another alternative is dynamic mode decomposition (DMD) that was first introduced to fluids community by Schmid and Sesterhenn [50] in 2008, subsequently followed up with a journal paper by Schmid [51] which presented a detailed description. DMD is a decomposition technique that is able to extract dynamic information from flow fields without relying on the availability of a model equation. It is instead based on time-resolved experimental or numerical data. This technique is essentially similar to Koopman analysis of nonlinear dynamical systems promoted by Mezić [52], as explained in Rowley *et al.* [53] (also see the reference [54]). Compared to POD, which seeks basis functions ranked in terms of energy content, DMD computes a set of modes each of which is associated with a fixed oscillation frequency and decay/growth rate that are defined by their corresponding eigenvalues. DMD modes represent spatial-temporal dominant structures within the reference data and the eigenvalues contain information about stability of their corresponding eigenfunctions. In contrast, the POD basis functions do not contain temporal evolution of the underlying process and the eigenvalues obtained by POD represent the energy content of the corresponding eigenfunctions. Since its inception in 2008, DMD has quickly gained popularity in fluids community and many variants of the DMD algorithm have been developed, including optimized DMD [55], and optimal mode decomposition [56], extended DMD [57]. The DMD and its variants are most often used to analyze flow fields using data from experiments or numerical simulations [51, 54, 58–64]. Recently, the use of DMD for control and construction of a reduced-order model has been studied by Proctor *et al.* [65, 66] and Annoni *et al.* [67, 68].

When it comes to constructing a reduced-order model, the POD, BPOD and DMD are all entirely data-driven. This means that the governing equations are not introduced until the generation of a reduced-order model, so after the basis functions (modes) have already been determined. The POD does not require any a priori knowledge of the underlying dynamics, that is, the POD is completely data dependent and does not take into account the governing equations in the POD process. This is inconsequential when applying the POD to data analysis and compression since only the dominant structures of the reference data need to be captured by the POD basis functions. However, when a reduced-order model is generated by projecting the governing equations onto the subspace spanned by the POD basis functions, we can't guarantee that the optimality of the POD basis functions still holds (i.e. whether the solution of POD ROM is a good approximation of the reference data). In addition, although the truncated set of POD basis functions can represent the most energetic processes within the reference data, these might not be the processes of interest. This can be the case, for example, when considering problems in acoustics, where the perturbation of interest are much smaller in energy

than the main flow. Moreover, the POD basis does not account for outputs of the system, although this can be improved upon using the BPOD method which uses adjoint information. Because no information regarding the governing equations is considered in the POD process, the POD basis functions do not properly reflect the fact that the data snapshots used are associated with different parametric instances of a dynamic system.

As an alternative to the data-driven based methods, in 2007, Bui-Thanh *et al.* proposed a goal-oriented model-constrained optimization method for identification of the basis functions [69]. In this method, the set of basis functions is optimized for a specific output functional of interest with a reduced-order model for the governing equations as an explicit constraint. Since the reduced-order model used as constraints is in discrete form, we refer to it as the fully-discrete formulation (FDF). It brings additional knowledge of the reduced-order governing equations into the determination of the basis functions. The ROM which is built through these optimal basis functions then should in principle have a better quality than that built through basis functions which are obtained from the data-driven methods. The results in [69] have shown that the FDF provides significant advantages over the POD.

1.2. DATA-DRIVEN REDUCED-ORDER MODELING TECHNIQUES

In this section, we will focus on the mathematical formulations of ROMs based on the proper orthogonal decomposition, balanced proper orthogonal decomposition, and dynamic mode decomposition.

1.2.1. PROPER ORTHOGONAL DECOMPOSITION

The proper orthogonal decomposition is a procedure for extracting an orthogonal basis of spatial functions from an ensemble of reference data obtained from experiments or high-dimension numerical simulations. The elements of the orthogonal basis set are referred to by many names [23], including empirical eigenfunctions, empirical basis functions, empirical orthogonal functions, or simply, modes. We will refer to them as POD modes. The most attractive feature of the POD is its optimality: it provides the most efficient way to interpolate the dominant components of a high-dimensional reference dataset using only a limited number of modes, as is described below.

The fundamental idea is straightforward. Suppose that we have an ensemble $\{u^n\}$ of reference data, where u^n represents the reference data at $t = t_n$, i.e. $u^n = u(x, t_n)$. We seek an orthogonal basis $\{\phi_j(x)\}$ for the Hilbert space $L^2(x)$ which is optimal for the reference dataset in the sense that a finite series in the separated-variables form

$$u_{\mathrm{nM}}(x,t) = \sum_{j=1}^{\mathrm{nM}} \alpha_j(t)\phi_j(x) \tag{1.1}$$

approximates the ensemble better than representations of the same dimensions in terms of any other bases. The ensemble may contain scalar-valued or vector-valued data $u(x,t)$. For simplicity, we consider an ensemble of scalar-valued, real-valued data $u(x,t)$ in the following, i.e. $u(x,t) \in \mathbb{R}$.

In mathematical terms, a normalized basis function ϕ is optimal if the averaged pro-

jection of u onto ϕ is maximized, that is, we seek a set of functions ϕ such that

$$\frac{\langle |(u,\phi)|^2 \rangle}{\|\phi\|^2} \tag{1.2}$$

is maximized. Here $(\cdot,\cdot)$ and $\|\cdot\|$ denote an appropriate inner product and norm for the space $L^2(x)$ of square integrable functions, e.g.,

$$(u,\,\phi) = \int_x u\phi dx \quad \text{and} \quad \|\phi\|^2 = (\phi,\,\phi) = \int_x \phi\phi dx. \tag{1.3}$$

In (1.2), $\langle\cdot\rangle$ denotes the ensemble average. Solution of the maximization problem (1.2) would yield only the best approximation by a single normalized function, but the other critical points of this problem are also physically significant, for they correspond to a set of functions which are taken together to provide the desired basis.

Thus, we now have a problem

$$\arg_\phi \max \langle |(u,\,\phi)|^2 \rangle, \qquad \text{subject to} \quad \|\phi\|^2 = 1. \tag{1.4}$$

Note that the problem (1.4) is a constrained optimization problem that can be solved by considering first-order optimality conditions. The optimality conditions can be derived through the Lagrangian functional, which is defined as

$$\mathscr{L}(\phi) = \langle |(u,\phi)|^2 \rangle + \lambda(1 - \|\phi\|^2), \tag{1.5}$$

where λ is the Lagrange multiplier and enforces the orthonormal constraint for ϕ. The optimality condition is defined by requiring that the first derivative of the Lagrangian functional vanishes for all variations $\phi + \delta\psi \in L^2(x)$ and $\delta \in \mathbb{R}$, that is,

$$\begin{aligned} \frac{d}{d\delta}\mathscr{L}(\phi+\delta\psi)|_{\delta=0} &= \frac{d}{d\delta}\left[\langle |(u,\phi+\delta\psi)|^2\rangle - \lambda(\phi+\delta\psi,\phi+\delta\psi)\right]\Big|_{\delta=0} \\ &= 2\left[\langle (u,\phi)(\psi,u)\rangle - \lambda(\phi,\psi)\right] \\ &= 0. \end{aligned} \tag{1.6}$$

The expression in the brackets can be written as

$$\begin{aligned} &\langle \int_x u(x)\phi(x)dx \int_{x'} u(x')\psi(x')dx' \rangle - \lambda \int_x \phi(x)\psi(x)dx \\ &= \langle \int_x \left[\int_{x'} u(x)u(x')\phi(x')dx' \right] \phi(x)dx \rangle - \lambda \int_x \phi(x)\psi(x)dx \\ &= \int_x \left[\int_{x'} \langle u(x)u(x')\rangle \phi(x')dx' - \lambda\phi(x) \right] \psi(x)dx, \end{aligned}$$

where we have brought the average "inside" using the commutativity of the ensemble average ($\langle\cdot\rangle$) and the space integral ($\int \cdot\, dx$) and rearranged the integrals. Since $\psi(x)$ is an arbitrary function, the necessary optimality condition reduces to

$$\int_{x'} \langle u(x)u(x')\rangle \phi(x')dx' = \lambda\phi(x). \tag{1.7}$$

This is a Fredholm integral equation whose kernel is the two-point space correlation tensor averaged over the data ensemble and is defined as

$$\mathbf{R}(x,x') \overset{\text{def}}{=\!=} \langle u(x)u(x')\rangle\,. \tag{1.8}$$

Thus, the optimal basis of POD modes is given by the set of eigenfunctions $\{\phi_j\}$ of the integral equation (1.7) [23, 70]. Each new eigenfunction ϕ_j is sought as the solution of the maximization problem (1.2) subject to the constraint of being orthogonal to all previously found eigenfunctions. Furthermore, the eigenvalues represent the energy content of the corresponding modes, then the eigenvalues are ordered so that $\lambda_j \geq \lambda_{j+1}$ with $j = 1, 2, \cdots$. Thus, using the subspace spanned by the first nM POD modes (ϕ_1, ϕ_2, $\cdots$, ϕ_{nM}), the representation (1.1) is the best approximation of the reference data with respect to other linear representations (e.g. Fourier series). In practice, to solve the eigenvalue problem (1.7), there are two different methods: the direct method and the snapshot POD, both of which are described in Appendix A.

1.2.2. BALANCED PROPER ORTHOGONAL DECOMPOSITION

The balanced proper orthogonal decomposition is an approximation of balanced truncation that balances the properties of controllability and observability: the most controllable states correspond to those that are most easily excited by the inputs, and the most observable states correspond to those that excite large future outputs. Balancing includes determining a coordinate system in which the most controllable directions in state space are also the most observable directions. One then truncates the least controllable/observable states. We first introduce the main ideas behind the balanced truncation below, and then proceed to describe the BPOD.

BALANCED TRUNCATION

Balanced truncation is a model reduction method used for stable linear input-output systems in the following form

$$\begin{aligned} \dot{x} &= Ax + Bu, \\ y &= Cx, \end{aligned} \tag{1.9}$$

where $u(t) \in \mathbb{R}^p$ is a vector containing p external inputs, $y(t) \in \mathbb{R}^q$ is a vector of outputs, and $x(t) \in \mathbb{R}^n$ is the state vector, and the matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times p}$, $C \in \mathbb{R}^{q\times n}$ (although generally u, y, x, A, B, and C can be complex as well).

Starting from defining controllability and observability Gramians, they are symmetric positive semidefinite matrices defined by

$$\begin{aligned} W_c &= \int_0^\infty e^{At} BB^* e^{A^* t}\, dt, \\ W_o &= \int_0^\infty e^{A^* t} C^* C e^{At}\, dt. \end{aligned} \tag{1.10}$$

The matrices W_c and W_o describe the controllable and observable subspaces of the system (1.9). The controllable subspace is the set of states which can be obtained with zero

initial state (i.e. $x(0) = 0$) and a specific input $u(t)$, while the observable subspace consists of those states which as initial conditions could produce a nonzero output $y(t)$ with no external input. The dominant eigenvectors (those correspond to the largest eigenvalues) of W_c and W_o describe the most controllable and observable states in the system [71], respectively. Normally the controllability and observability Gramians are evaluated by solving the Lyapunov equations [72]

$$\begin{aligned} AW_c + W_c A^* + BB^* = 0, \\ A^* W_o + W_o A + C^* C = 0. \end{aligned} \tag{1.11}$$

Here A^*, B^*, and C^* are Hermitian matrices of A, B, and C (for real matrices the Hermitian is equivalent to the transpose), respectively.

The Gramians depend on the coordinate system. When using a change of state coordinates $x = Tz$, they transform as:

$$\begin{cases} W_c \mapsto T^{-1} W_c (T^{-1})^*, \\ W_o \mapsto T^* W_o T. \end{cases}$$

To obtain a balanced realization of the linear system (1.9), the state transformational coordinate T is chosen so that the transformed controllability and observability Gramians are equal and diagonal, that is,

$$T^{-1} W_c (T^{-1})^* = T^* W_o T = \Sigma = \mathrm{diag}(\sigma_1, \cdots, \sigma_n), \tag{1.12}$$

where $\sigma_1 \geq \cdots \geq \sigma_n \geq 0$. Hence, the balancing transformation T can be found by computing the eigenvectors of the product $W_c W_o$, i.e.

$$W_c W_o T = \Sigma^2 T. \tag{1.13}$$

The eigenvectors of $W_c W_o$ correspond to states through which the input is transmitted to the output. The diagonal elements $\{\sigma_j\}_{j=1}^n$ in (1.12) are known as the Hanker singular values of the system. The Hanker singular values indicate the importance of the corresponding state for transmitting input to output and are independent of the particular state coordinate system T. In the balanced truncation, only those states which correspond to the largest Hanker singular values, in other words which have more effect on the input-output behavior, are retained.

With the balancing transformation T defined, the reduced-order model for the linear system (1.9) is formulated as

$$\begin{aligned} \dot{x}_r &= T^{-1} A T x_r + T^{-1} B u, \\ y_r &= C T x_r, \end{aligned} \tag{1.14}$$

where $x_r \in \mathbb{R}^{n_r}$ is the reduced-order state vectors and $n_r << n$ is the number of retained states. An error criterion for model reduction is derived [73], in which the Hankel singular values of the neglected states give an error bound on the output. The error bound is given by

$$\|y - y_r\| \leq 2 \sum_{i=n_r+1}^{n} \sigma_i^2 \|u\|, \tag{1.15}$$

where $\|\cdot\|$ denotes the L_2 norm.

BALANCED PROPER ORTHOGONAL DECOMPOSITION

The computation of W_c and W_o by solving the Lyapunov equations is not possible for very large systems, as the Gramians are $n \times n$ matrices that are not sparse. BPOD provides an approximation to balanced truncation using an algorithm that is computationally tractable for high-dimensional systems. It is based on defining the empirical observability and observability Gramians, which approximate W_c and W_o from the Lyapunov equations using data from numerical simulations (the empirical observability and observability Gramians are used by Lall *et al.* [42, 43] to extend balanced truncation to nonlinear systems). Then the balancing transformation is computed using an SVD.

To compute the empirical controllability Gramian W_c for a system with p inputs, we first calculate solutions of

$$\begin{aligned} \frac{d}{dt}x_i(t) &= Ax_i(t), \\ x_i(0) &= b_i, \end{aligned} \tag{1.16}$$

where $i = 1, 2, \cdots, p$ and b_i is the ith column of the matrix B in equation (1.9). Then the empirical controllability Gramian is given by

$$W_c = \int_0^\infty \left(x_1(t)x_1^*(t) + \cdots + x_p(t)x_p^*(t) \right) dt. \tag{1.17}$$

In practice, the snapshots of the vector $x_i(t)$ from numerical simulations are usually given at discrete times $t_1, \cdots, t_m$, and we form a matrix X in the following form

$$X = \left[x_1(t_1)\sqrt{\omega_1}, \cdots, x_1(t_m)\sqrt{\omega_m}, \cdots, x_p(t_1)\sqrt{\omega_1}, \cdots, x_p(t_m)\sqrt{\omega_m} \right], \tag{1.18}$$

where ω_j are quadrature coefficients, and thus the empirical controllability Gramian is rewriten as

$$W_c = XX^*. \tag{1.19}$$

To make the calculation of the empirical observability Gramian tractable when the number of outputs is very large, the BPOD uses an output projection method, in which the output is projected onto an appropriate subspace in such a way that the new input-output system is the same order as the original system but with a much smaller dimensional output. That is, instead of the system (1.9) used for the balanced truncation method, BPOD considers a new related system

$$\begin{aligned} \dot{x} &= Ax + Bu, \\ y &= P_r Cx, \end{aligned} \tag{1.20}$$

where P_r is an orthogonal projection with rank r (for the determination of r, the interested reader is referred to [40]). This projection allows us to compute the empirical observability Gramian using only r simulations of the adjoint system

$$\begin{aligned} \frac{d}{dt}z(t) &= A^* z(t), \\ z(0) &= C^* \Phi_r, \end{aligned} \tag{1.21}$$

where Φ_r is a $q \times r$ matrix with $\Phi_r^* \Phi_r = I_r$. Furthermore, the matrix Φ_r can be chosen as a matrix that consists of the first r POD modes of the dataset $\{Cx_1(t), \cdots, Cx_p(t)\}$. Using solutions of the adjoint system $z(t)$ at discrete times, we form a data matrix Z as in (1.18), and calculate the empirical observability Gramian by

$$W_o = ZZ^*. \tag{1.22}$$

Once the matrices X and Z are determined, the balancing transformation is computed by forming the SVD of the matrix Z^*X, i.e.

$$\begin{aligned} Z^*X &= U\Sigma V^* \\ &= [U_1 \quad U_2] \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1^* \\ V_2^* \end{bmatrix} \\ &= U_1 \Sigma_1 V_1^*, \end{aligned} \tag{1.23}$$

where $\Sigma_1 \in R^{r \times r}$ is invertible, r is the rank of the matrix Z^*X, and $U_1^* U_1 = V_1^* V_1 = I_r$. Denote the matrices $T_1 \in R^{n \times r}$ and $S_1 \in R^{r \times n}$ as

$$T_1 = XV_1\Sigma_1^{-1/2}, \quad S_1 = \Sigma_1^{-1/2} U_1^* Z^*. \tag{1.24}$$

If $r = n$, then the matrix Σ_1 contains the Hankel singular values, T_1 determines the balancing transformation, and S_1 is its inverse. If $r < n$, then the columns of T_1 form the first r columns of the balancing transformation, and the rows of S_1 form the first r rows of the inverse transformation. Finally, the BPOD ROM of order r for the linear system (1.9) is written as

$$\begin{aligned} \dot{x}_r &= S_1 A T_1 x_r + S_1 B u, \\ y_r &= C T_1 x_r. \end{aligned} \tag{1.25}$$

1.2.3. DYNAMIC MODE DECOMPOSITION

In this section, we describe the dynamic mode decomposition technique for extracting dynamical features from experimental or numerical flow field data. It is assumed that the data is presented in the form of a snapshot sequence, given by a matrix V_i^N that is defined as

$$V_1^N = [v_1, v_2, \cdots, v_N], \tag{1.26}$$

where $v_i \in \mathbb{R}^{N_g}$ is the ith snapshot of the flow field. In (1.26), the subscript 1 and superscript N denote the first and last columns of the matrix $V_1^N \in \mathbb{R}^{N_g \times N}$, respectively. These snapshots are assumed to be separated by a constant sampling time Δt. It is assumed that a linear mapping D connects the flow field v_i at time $t = t_i$ to v_{i+1} at the next time step, that is,

$$v_{i+1} = D v_i, \tag{1.27}$$

and that this mapping remains approximately the same over the full sampling interval. Hence, we can formulate the snapshot sequence as a Krylov sequence, i.e.

$$V_1^N = \left[v_1, Dv_1, D^2 v_1, \cdots, D^{N-1} v_1\right].$$

The goal of the DMD is to determine the eigenvalues and eigenvectors of D based on the snapshot sequence V_1^N. The eigenvalues and eigenvectors of D are referred to the DMD eigenvalues and DMD modes, respectively.

As the number of snapshots increases, we can assume that the Nth snapshot v_N is described as a linear combination of the previous and linearly independent snapshots v_i, $i = 1, \cdots, N-1$, that is,

$$v_N = \sum_{i=1}^{N-1} c_i v_i + \boldsymbol{r} = V_1^{N-1}\boldsymbol{c} + \boldsymbol{r}, \tag{1.28}$$

where $\boldsymbol{c} = [c_1, c_2, \cdots, c_{N-1}]^T$ is a vector of unknown coefficients and $\boldsymbol{r}$ is the residual vector. By following Ruhe [74], the flow field can be written as

$$D[v_1, v_2, \cdots, v_{N-1}] = [v_2, v_3, \cdots, v_N] = [v_2, v_3, \cdots, V_1^{N-1}\boldsymbol{c}] + \boldsymbol{r}\boldsymbol{e}_{N-1}^T, \tag{1.29}$$

or in a matrix form

$$DV_1^{N-1} = V_2^N = V_1^{N-1}S + \boldsymbol{r}\boldsymbol{e}_{N-1}^T, \tag{1.30}$$

where $\boldsymbol{e}_{N-1} = [0, 0, \cdots, 1]^T \in \mathbb{R}^{N-1}$, and S is the companion matrix

$$S = \begin{bmatrix} 0 & 0 & \cdots & 0 & c_1 \\ 1 & 0 & \cdots & 0 & c_2 \\ 0 & 1 & \cdots & 0 & c_3 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & c_{N-1} \end{bmatrix}. \tag{1.31}$$

The unknown matrix S is determined by solving a least-squares problem

$$S = \arg\min_S \| V_2^N - V_1^{N-1}S \|, \tag{1.32}$$

which minimizes the residual $\boldsymbol{r}$. In general, we use the QR decomposition of $V_1^{N-1} = QR$ to solve this least squares problem (1.32). The vector $\boldsymbol{c}$ then is given by

$$\boldsymbol{c} = R^{-1}Q^T v_N. \tag{1.33}$$

Once S has been determined, its eigenvalues and eigenvectors can be computed. The eigenvalues of S are approximations of the eigenvalues of D. Furthermore, if ψ_j is an eigenvector of S, then $(U_1^{N-1}\psi_j)$ is an approximate eigenvector of D.

The implementation of the above decomposition based on the companion matrix S can yield an ill-conditioned algorithm that is often not capable of extracting more than the first or first two dominant dynamic modes. Thus, in practice, the SVD-based approach presented in [51], also named as Standard DMD by Tu *et al.* [75], is used to compute the DMD eigenvalues and DMD modes. The algorithm proceeds as follows:

1. Split the sequence of snapshots V_1^N into two sequences

$$V_1^{N-1} = \big(v_1, v_2, \cdots, v_{N-1}\big), \quad V_2^N = \big(v_2, v_3, \cdots, v_N\big). \tag{1.34}$$

2. Perform the singular value decomposition of V_1^{N-1}

$$V_1^{N-1} = U\Sigma W^T, \tag{1.35}$$

where the left singular vectors U are recognized as POD modes of the data sequence V_1^{N-1}.

3. Define the matrix $\tilde{S}$, which amounts to a projection of the linear operator D onto a POD basis

$$\begin{aligned} D &= V_2^N \left(V_1^{N-1}\right)^{-1} = V_2^N W \Sigma^{-1} U^T \\ \Longrightarrow \tilde{S} &= U^T D U = U^T V_2^N W \Sigma^{-1}. \end{aligned} \tag{1.36}$$

4. Compute the eigenvalues and eigenvectors of $\tilde{S}$

$$\tilde{S}\psi = \lambda\psi. \tag{1.37}$$

5. Define the DMD eigenvalues as the nonzero eigenvalues of $\tilde{S}$, and the DMD mode corresponding to λ_i by

$$\phi_i = U\psi_i. \tag{1.38}$$

Note that the DMD eigenvalues produce stability information about DMD modes when mapped onto the complex plane as defined in [64]

$$\mu_i = \frac{\log(\lambda_i)}{\Delta t}. \tag{1.39}$$

Here the real and imaginary components of μ_i represent the growth/decay rate and frequency of the corresponding DMD mode, respectively.

1.3. FULLY-DISCRETE FORMULATION

The FDF provides a general framework for identifying bases that optimally represent an output functional of interest. To obtain optimal bases, Bui-Thanh *et al.* considered a goal-oriented, model-constrained optimization problem that seeks to minimize the error between the full-space output functional and reduced-order output functional over a time interval $(0,\ t_f)$, subject to satisfying a ROM for the linear-time invariant (LTI) system.

1.3.1. REDUCED ORDER MODEL FOR THE LINEAR TIME INVARIANT SYSTEM

Considering the LTI system

$$M\dot{u} + Ku = F \tag{1.40}$$

$$g = Cu \tag{1.41}$$

with the initial condition

$$u(0) = u_0 \tag{1.42}$$

where $u(t)$ is the system state, $\dot{u}(t)$ is the derivative of $u(t)$ with respect to time, and the vector u_0 contains the specified initial state. Generally, we are interested in systems of the form(1.40) that result from discretizing partial differential equations in space. The matrices $M \in \mathbb{R}^{N\times N}$ and $K \in \mathbb{R}^{N\times N}$ are dependent on the chosen spatial discretization method. N is the dimension of the system in space, while the vector $F \in \mathbb{R}^N$ defines the input to the system. The matrix $C \in \mathbb{R}^{Q\times N}$ defines the Q outputs of interest, which are contained in the output functional (g). g is referred to the *goal functional*.

To construct a reduced-order model for the LTI system, it is assumed that the state $u(t)$ is represented as a linear combination of nM projection basis functions (Φ), i.e.

$$\hat{u} = \Phi\alpha, \tag{1.43}$$

where $\hat{u}$ is the reduced order approximation of the state $u(t)$ ($\mathrm{nM} \ll N$) and α is the state variable. The matrix $\Phi \in \mathbb{R}^{N\times \mathrm{nM}}$ contains as columns the projection basis functions ϕ_j, i.e., $\Phi = [\phi_1, \phi_2, \cdots, \phi_{\mathrm{nM}}]$, and $\alpha(t) \in \mathbb{R}^{\mathrm{nM}}$ is a vector containing the corresponding modal amplitudes as a function of time. Inserting (1.43) into (1.40)-(1.42) and projecting them onto reduced space by pre-multiplying with Φ^T yields the reduced-order model for the LTI system (1.40)-(1.42) with state variables $\alpha(t)$ and goal functional $\hat{g}$

$$\hat{M}\dot{\alpha} + \hat{K}\alpha = \hat{F} \tag{1.44}$$

$$\hat{g} = \hat{C}\alpha \tag{1.45}$$

$$\hat{M}\alpha_0 = \Phi^T M u_0 \tag{1.46}$$

where $\hat{M} = \Phi^T M\Phi \in \mathbb{R}^{\mathrm{nM}\times\mathrm{nM}}$, $\hat{K} = \Phi^T K\Phi \in \mathbb{R}^{\mathrm{nM}\times\mathrm{nM}}$, $\hat{F} = \Phi^T F \in \mathbb{R}^{\mathrm{nM}}$, $\hat{C} = C\Phi \in \mathbb{R}^{Q\times\mathrm{nM}}$, and $\alpha_0 = \alpha(0) \in \mathbb{R}^{\mathrm{nM}}$.

1.3.2. CONSTRAINED OPTIMIZATION PROBLEM

As stated at the beginning of this section, using the ROM for the LTI system (1.44)-(1.46), the constrained optimization problem used to find optimal bases for the goal functional is expressed mathematically as

$$\arg_{\Phi,\alpha}\min \mathscr{G} = \frac{1}{2}\int_0^{t_f} (g-\hat{g})^T(g-\hat{g})dt + \frac{\beta}{2}\sum_{i,j=1}^{\mathrm{nM}} \left(\delta_{ij} - \phi_i^T\phi_j\right)^2 \tag{1.47}$$

subject to

$$\Phi^T M\Phi\dot{\alpha} + \Phi^T K\Phi\alpha = \Phi^T F \tag{1.48}$$

$$\Phi^T M\Phi\alpha_0 = \Phi^T M u_0 \tag{1.49}$$

$$\hat{g} = C\Phi\alpha \tag{1.50}$$

where δ_{ij} is the Kronecker delta function, i.e.

$$\delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

In the case of a linear relationship between the goal functional g and the state u as in (1.41), the *objective functional* $\mathscr{G}$ can be written as

$$\mathscr{G} = \frac{1}{2}\int_0^{t_f} (u-\hat{u})^T H(u-\hat{u})\,dt + \frac{\beta}{2}\sum_{i,j=1}^{\mathrm{nM}} \left(\delta_{ij} - \phi_i^T \phi_j\right)^2, \tag{1.51}$$

where $H = C^T C$ can be interpreted as a weighting matrix that defines the states relevant to the specified goal functional. The first term in the objective functional $\mathscr{G}$ (1.51) expresses the error for a particular goal functional rather than for the general state vector, as in the POD approach. Through the constraints (1.48)-(1.50), the FDF requires satisfaction of the reduced-order governing equations to determine $\hat{u}$. Thus, the error minimized by the FDF is tied rigorously to the reduced-order model, in contrast to the POD which is based purely on the reference data. The second term in (1.51) is a regularization term that penalizes the deviation of the projection basis from an orthonormal set, with β as a regularization parameter.

1.3.3. OPTIMALITY SYSTEM FOR THE CONSTRAINED OPTIMIZATION PROBLEM

The problem (1.47)-(1.50) is a constrained minimization problem, for which optimality conditions can be derived through the Lagrangian functional $\mathscr{L}$, defined as the objective functional (1.51) plus the constraints (1.48) and (1.49) multiplied by Lagrange multipliers, i.e.

$$\begin{aligned}\mathscr{L}(\Phi,\,\alpha,\,\lambda,\,\mu) = &\frac{1}{2}\int_0^{t_f} (u-\hat{u})^T H(u-\hat{u})\,dt + \frac{\beta}{2}\sum_{i,j=1}^{\mathrm{nM}} \left(\delta_{ij} - \phi_i^T \phi_j\right)^2 \\ &+ \int_0^{t_f} \lambda^T(\Phi^T M\Phi\dot{\alpha} + \Phi^T K\Phi\alpha - \Phi^T F)\,dt \\ &+ \mu^T(\Phi^T M\Phi\alpha_0 - \Phi^T M u_0),\end{aligned} \tag{1.52}$$

where $\lambda = \lambda(t) \in \mathbb{R}^{\mathrm{nM}}$ and $\mu \in \mathbb{R}^{\mathrm{nM}}$ are Lagrange multipliers (which can also be named adjoint variables,) that enforce the reduced-order governing equations and initial conditions, respectively. The optimality system can be derived by taking variations of the Lagrangian functional with respect to state variables $\alpha(t)$, adjoint variables $\lambda(t)$ and μ, and the projection basis functions ϕ.

Setting the first variation of the Lagrangian functional with respect to μ and $\lambda(t)$ to zero and arguing that the variation of $\lambda(t)$ is arbitrary in $(0,\,t_f)$, simply recovers initial condition constraint (1.49) and the ROM (1.48). From now on, (1.48) and (1.49) are referred to as the **State Equations**.

Setting the first variation of the Lagrangian functional with respect to $\alpha(t)$ to zero, and arguing the variation of $\alpha(t)$ is arbitrary in $(0,\,t_f)$, at $t = 0$ and at $t = t_f$, yields the so-called adjoint equation, final condition for λ, and definition of μ

$$-\Phi^T M\Phi\dot{\lambda} + \Phi^T K^T\Phi\lambda = \Phi^T H(u-\Phi\alpha) \tag{1.53}$$

$$\lambda(t_f) = 0 \tag{1.54}$$

$$\mu = \lambda(0) \tag{1.55}$$

Note that $H = C^T C$ is a symmetric matrix and it was assumed that M is a symmetric matrix as well, otherwise M^T would appear. Equations (1.53)-(1.55) are referred to as **Adjoint Equations**.

Setting the first variation of the Lagrangian functional with respect to the projection basis Φ to zero yields the following matrix equation, which is referred to as the **Gradient** of the Lagrangian functional,

$$\begin{aligned}\delta\mathscr{L}_\Phi = &\int_0^{t_f} H(\Phi\alpha - u)\alpha^T dt + 2\beta\Phi(\Phi^T\Phi - I) + \int_0^{t_f} \left[M\Phi(\lambda\dot{\alpha}^T + \dot{\alpha}\lambda^T)\right. \\ &\left. + K^T\Phi\lambda\alpha^T + K\Phi\alpha\lambda^T - F\lambda^T\right]dt + \left[M\Phi\mu\alpha_0^T + M(\Phi\alpha_0 - u_0)\mu^T\right] \\ =&0\end{aligned} \tag{1.56}$$

where I is the identity matrix. The combined system, (1.48)-(1.49), (1.53)-(1.55), and (1.56), represents the first-order Karush-Kuhn-Tucker (KKT) optimality conditions for the optimization problem (1.47)-(1.50).

1.4. RESEARCH OBJECTIVES

It has been previously mentioned that the FDF has advantages over the POD, BPOD and DMD since it takes into account the reduced-order governing equations in the process of determining an optimal basis set for a specific goal functional. However, the FDF has several limitations in terms of its applicability:

1. To implement the FDF, we need to explicitly define the matrices M and K. How these are to be constructed is not always obvious. If the reference data (u) is generated from a detailed numerical simulation of a linear partial differential equation, it is natural to use the corresponding matrices M and K which result from the spatial discretization of the governing equation. If the reference data comes from

   - experiments, or
   - commercial software

   the definition of M and K is not readily available. In these cases, use of the FDF requires the construction of the LTI system capable of reproducing the reference data, which is a complex task. The problem is exacerbated if the reference dataset is sparse, unstructured in space or time, or comes from a system which behaves non-linearly, e.g. for which $K = K(u)$.

2. Even if the problem of constructing M and K can be resolved, the FDF is unnecessarily restrictive. In the interests of creating a stable and robust reduced-order model, for example, it can be advantageous to construct the ROM based on partial differential equations different from these used to generate the reference data. This would be the case when considering coarse reduced-order models for highly multiscale problems, such as turbulent flow, for which additional unresolved scale models [76] are required.

3. In a similar vein, one may wish to use different boundary condition formulations for the reduced-order model. For example, one may wish to employ weak formulations for Dirichlet boundary conditions, which are known to reduce interpolation errors when coarse solution interpolations are used [77].

4. Within the current formulation in the FDF, the treatment of nonlinear reduced-order governing equations and goal functionals has not been addressed.

To address the above mentioned limitations, our research group including Steven J. Hulshoff, Stefano Mattei [78], and Lei Cheng and so on has proposed an alternative, the semi-continuous formulation (SCF), where the reduced-order model used as constraints and optimization techniques are defined in the continuous setting. Thus, the objectives of this thesis are:

1. Investigate the mathematical formulation of the SCF in which continuous reduced-order models for general partial differential equations are used as constraints and the goal functionals are defined as linear or nonlinear functionals of the ROM solution.

2. Examine the behavior and expense of the optimization algorithm used to solve the optimization problem.

3. Demonstrate the viability of the SCF through its applications to one-dimensional and two-dimensional problems in which the governing equations are linear or nonlinear and the goal functionals are linear or nonlinear functions of the constrained ROM's solution.

4. Demonstrate that when the optimal basis obtained through the SCF provides improvements relative to the standard POD-based method.

1.5. THESIS OUTLINE

The content of this thesis is organized as follows:

1. **Goal-Oriented ROMs:** In Chapter 2, we will first illustrate the SCF proposed by our research group where the reduced-order governing equations are in the continuous setting. Next, the algorithm used to solve the constrained optimization problem, a Trust-Region Inexact-Newton Conjugate-Gradient algorithm, is introduced. Lastly, the structure of the basis to be optimized is described.

2. **Applications of the SCF approach:** In Chapter 3, we investigate applications of the SCF approach to one-dimensional problems: (1) the linear advection-diffusion equation, (2) the Burgers equation, (3) a problem in which an approximate governing equation is used. The application of SCF to two-dimensional Stokes equations is investigated in Chapter 4. The goal functionals to be investigated in both one-dimensional and two-dimentional applications include linear and nonlinear functions of the ROM's solutions.

3. **Discussion and conclusions:** In the last chapter, the results are summarized and the potential of the SCF is evaluated.

2

THE SEMI-CONTINUOUS FORMULATION FRAMEWORK

To overcome the limitations of the FDF described in Chapter 1, an alternative, the semi-continuous formulation (SCF), has been proposed by our research group. In this chapter, we will illustrate the mathematical expressions of the SCF. Then, the optimization techniques and the optimization algorithm used to solve the constrained optimization problem are presented. Lastly, the structure of projection basis is defined.

2.1. SEMI-CONTINUOUS FORMULATION

In this section, we will describe the mathematical formulations in the SCF. We name our method as SCF since the generation of the reduced-order model and the optimization process are defined in the continuous setting, while the projection basis functions are discrete.

2.1.1. A CONTINUOUS REDUCED-ORDER MODEL

Considering the general partial differential equation with initial condition

$$\mathfrak{L}(u) = f, \tag{2.1}$$

$$u(0) = u_0, \tag{2.2}$$

where $\mathfrak{L}$ represents a combination of differential operators in time and space. Defining the goal functional as

$$g = g(u), \tag{2.3}$$

which can be a linear or nonlinear function of the solution u of equation (2.1) with the initial condition (2.2).

A continuous reduced-order model for (2.1)-(2.3) can be derived by assuming that the state $u(t)$ is represented by a linear combination of nM projection basis functions:

$$\hat{u} = \sum_{j=1}^{\mathrm{nM}} \alpha_j(t)\phi_j \tag{2.4}$$

Inserting (2.4) into (2.1)-(2.3) and making use of Galerkin projection framework yields the continuous reduced-order model with the reduced-order goal functional $\hat{g}$:

$$\int_\Omega \phi_i \left[\mathfrak{L}(\sum_{j=1}^{\mathrm{nM}} \alpha_j\phi_j) - f \right] d\Omega = 0 \tag{2.5}$$

$$\int_\Omega \phi_i \left(\sum_{j=1}^{\mathrm{nM}} \alpha_j^0\phi_j - u_0 \right) d\Omega = 0 \tag{2.6}$$

$$\hat{g} = g\left(\sum_{j=1}^{\mathrm{nM}} \alpha_j\phi_j \right) \tag{2.7}$$

where Ω represents the space of interest, $i = 1, 2, \cdots, \mathrm{nM}$ and $\alpha_j^0 = \alpha_j(0)$.

2.1.2. CONSTRAINED OPTIMIZATION PROBLEM AND OPTIMALITY SYSTEM

The goal-oriented optimization problem solved in the SCF is analogous to that solved in the FDF, that is, the problem is to minimize the error between the full-space goal functional (g) and reduced-order goal functional ($\hat{g}$) over a space domain Ω and a time interval $(0,\ t_f)$, subject to satisfying the underlying continuous reduced-order governing equations (2.5)-(2.7). The problem of determining an optimal projection basis, Φ, can

be described as:

$$\arg_{\Phi,\alpha}\min \mathscr{G} = \frac{1}{2}\int_0^{t_f}\int_\Omega (g-\hat{g})^2 d\Omega dt + \frac{\beta}{2}\sum_{i,j=1}^{\mathrm{nM}}\left(\delta_{ij} - \int_\Omega \phi_i\phi_j d\Omega\right)^2 \tag{2.8}$$

subject to

$$\int_\Omega \phi_i\left[\mathfrak{L}(\sum_{j=1}^{\mathrm{nM}}\alpha_j\phi_j) - f\right]d\Omega = 0 \tag{2.9}$$

$$\int_\Omega \phi_i\left(\sum_{j=1}^{\mathrm{nM}}\alpha_j^0\phi_j - u_0\right)d\Omega = 0 \tag{2.10}$$

$$\hat{g} = g\left(\sum_{j=1}^{\mathrm{nM}}\alpha_j\phi_j\right) \tag{2.11}$$

Note that the formulation (2.8)-(2.11) is continuous with the exception of the evaluation of the first term in (2.8), which must be evaluated discretely when the reference data is provided in discrete form.

The problem (2.8)-(2.11) is a constrained minimization problem, with optimality conditions derived by defining the Lagrangian functional $\mathscr{L}$. To simplify the expression of $\mathscr{L}$, we define

$$E(\Phi,\alpha) = (g-\hat{g})^2, \qquad G_1(\Phi) = \sum_{i,j=1}^{\mathrm{nM}}\left(\delta_{ij} - \int_\Omega \phi_i\phi_j d\Omega\right)^2,$$
$$G_2^i(\Phi,\Phi_x,\alpha,\dot{\alpha}) = \phi_i(\mathfrak{L}(\hat{u}) - f), \qquad G_3^i(\Phi,\alpha^0) = \phi_i(\hat{u}_0 - u_0).$$

In the definitions presented above, we have made some assumptions:

- The goal functional is not a function of u's space or time derivative, for example, u_x, $\dot{u}$, etc.
- The governing equation only includes the first-order time derivative of u and up to the second-order space derivative of u, i.e. $\dot{u}$, u_x and u_{xx}. This assumption could be replaced if necessary. Note, however if the governing equation includes the second-order space derivative, we could always reduce it into the first-order space derivative when constructing $\mathscr{L}$, i.e. $G_2^i(\Phi,\Phi_x,\alpha,\dot{\alpha})$.

The Lagrangian functional then is written as:

$$\begin{aligned}\mathscr{L} = &\int_0^{t_f}\int_\Omega \frac{1}{2}E(\Phi,\alpha)d\Omega dt + \frac{\beta}{2}G_1(\Phi) + \int_0^{t_f}\sum_{i=1}^{\mathrm{nM}}\lambda_i\int_\Omega G_2^i(\Phi,\Phi_x,\alpha,\dot{\alpha})d\Omega dt \\ &+ \sum_{i=1}^{\mathrm{nM}}\mu_i\int_\Omega G_3^i(\Phi,\alpha^0)d\Omega\end{aligned} \tag{2.12}$$

Optimality conditions for problem (2.8)-(2.11) are derived by taking the first variations of the Lagrangian functional (2.12) with respect to adjoint variables λ_i and μ_i, state variable α_j, and projection basis functions ϕ_q ($q = 1, 2, \cdots, \mathrm{nM}$).

Setting the first variation of the Lagrangian functional (2.12) with respect to λ_i to zero, and arguing that the variation of λ_i is arbitrary in $(0,\ t_f)$, recovers the ROM (2.9). Setting the first variation of the Lagrangian functional with respect to μ_i to zero naturally recovers the initial condition (2.10). Thus, equations (2.9) and (2.10) are the **State Equations**.

Setting the first variation of the Lagrangian functional (2.12) with respect to α_j to zero yields

$$\begin{aligned}\delta\mathscr{L}_{\alpha_j} = &\int_0^{t_f}\left[\int_\Omega \frac{1}{2}\frac{\partial E}{\partial \alpha_j} + \sum_{i=1}^{\mathrm{nM}} \lambda_i \frac{\partial G_2^i}{\partial \alpha_j} - \frac{d}{dt}\left(\sum_{i=1}^{\mathrm{nM}} \lambda_i \frac{\partial G_2^i}{\partial \dot{\alpha}_j}\right) d\Omega\right]\delta\alpha_j dt \\ &+ \int_\Omega \sum_{i=1}^{\mathrm{nM}} \lambda_i \frac{\partial G_2^i}{\partial \dot{\alpha}_j}\delta\alpha_j\Big|_0^{t_f} d\Omega + \int_\Omega \sum_{i=1}^{\mathrm{nM}} \mu_i \frac{\partial G_3^i}{\partial \alpha_j}\delta\alpha_j\Big|^0 d\Omega \\ =&0.\end{aligned} \tag{2.13}$$

Arguing that the variations of α_j, i.e. $\delta\alpha_j$, are arbitrary in the time interval $(0,\ t_f)$, then leads to

$$\int_\Omega \left[\frac{1}{2}\frac{\partial E}{\partial \alpha_j} + \sum_{i=1}^{\mathrm{nM}} \lambda_i \frac{\partial G_2^i}{\partial \alpha_j} - \frac{d}{dt}\left(\sum_{i=1}^{\mathrm{nM}} \lambda_i \frac{\partial G_2^i}{\partial \dot{\alpha}_j}\right)\right] d\Omega = 0. \tag{2.14}$$

Arguing that the variations of α_j at the boundary points, i.e. $\delta\alpha_j(0)$ and $\delta\alpha_j(t_f)$, are arbitrary, implies that the following equations must be satisfied:

$$\begin{aligned}&\int_\Omega \sum_{i=1}^{\mathrm{nM}} \lambda_i \frac{\partial G_2^i}{\partial \dot{\alpha}_j}\Big|^{t_f} d\Omega = 0 \\ &\int_\Omega \sum_{i=1}^{\mathrm{nM}} \left(\mu_i \frac{\partial G_3^i}{\partial \alpha_j} - \lambda_i \frac{\partial G_2^i}{\partial \dot{\alpha}_j}\right)\Big|^0 d\Omega = 0\end{aligned}$$

These lead to the so-called final condition and definition of μ_i:

$$\lambda_i(t_f) = 0 \tag{2.15}$$

$$\int_\Omega \sum_{i=1}^{\mathrm{nM}} \left(\mu_i \frac{\partial G_3^i}{\partial \alpha_j} - \lambda_i \frac{\partial G_2^i}{\partial \dot{\alpha}_j}\right)\Big|^0 d\Omega = 0 \tag{2.16}$$

The **Adjoint Equations** then are equations (2.14), (2.15), and (2.16).

Setting the first variation of Lagrangian functional (2.12) with respect to ϕ_q to zero yields

$$\begin{aligned}\delta\mathscr{L}_{\phi_q} = &\int_\Omega \left[\int_0^{t_f} \frac{1}{2}\frac{\partial E}{\partial \phi_q} + \sum_{i=1}^{\mathrm{nM}} \lambda_i \frac{\partial G_2^i}{\partial \phi_q} - \frac{d}{dx}\left(\sum_{i=1}^{\mathrm{nM}} \lambda_i \frac{\partial G_2^i}{\partial \phi_{qx}}\right) dt + \sum_{i=1}^{\mathrm{nM}} \mu_i \frac{\partial G_3^i}{\partial \phi_q}\right]\delta\phi_a d\Omega \\ &+ \frac{\beta}{2}\frac{\delta G_1}{\delta \phi_q} + \int_0^{t_f} \sum_{i=1}^{\mathrm{nM}} \lambda_i \frac{\partial G_2^i}{\partial \phi_{qx}}\delta\phi_q\Big|_{\partial\Omega} dt \\ =&0.\end{aligned} \tag{2.17}$$

Arguing that the variations of ϕ_q are zero at boundaries of the space domain, i.e. $\delta\phi_q\big|_{\partial\Omega} = 0$, leads to the **Gradient**:

$$\begin{aligned}\delta\mathscr{L}_{\phi_q} = &\int_\Omega \left\{ \int_0^{t_f} \left[\frac{1}{2}\frac{\partial E}{\partial \phi_q} + \sum_{i=1}^{\mathrm{nM}} \lambda_i \frac{\partial G_2^i}{\partial \phi_q} - \frac{d}{dx}\left(\sum_{i=1}^{\mathrm{nM}} \lambda_i \frac{\partial G_2^i}{\partial \phi_{qx}} \right) \right] dt + \sum_{i=1}^{\mathrm{nM}} \mu_i \frac{\partial G_3^i}{\partial \phi_q} \right\} \delta\phi_q d\Omega \\ &+ 2\beta \int_\Omega \sum_{i=1}^{\mathrm{nM}} \left(\int_\Omega \phi_i \phi_q d\Omega - \delta_{iq} \right) \phi_i \delta\phi_q d\Omega \\ =&0\end{aligned} \tag{2.18}$$

We continue to argue that the variations of ϕ_q are arbitrary in the space interval Ω. The **Gradient** thus is rewritten as:

$$\begin{aligned}\delta\mathscr{L}_{\phi_q} = &\int_0^{t_f} \left[\frac{1}{2}\frac{\partial E}{\partial \phi_q} + \sum_{i=1}^{\mathrm{nM}} \lambda_i \frac{\partial G_2^i}{\partial \phi_q} - \frac{d}{dx}\left(\sum_{i=1}^{\mathrm{nM}} \lambda_i \frac{\partial G_2^i}{\partial \phi_{qx}} \right) \right] dt + \sum_{i=1}^{\mathrm{nM}} \mu_i \frac{\partial G_3^i}{\partial \phi_q} \\ &+ 2\beta \sum_{i=1}^{\mathrm{nM}} \left(\int_\Omega \phi_i \phi_q d\Omega - \delta_{iq} \right) \phi_i \\ =&0\end{aligned} \tag{2.19}$$

2.1.3. COMMENTS

The SCF is a generalization of the FDF. It still has the same advantages as the FDF, but it overcomes some drawbacks of the FDF. Using the SCF, we avoid the ambiguities in the definition of matrices M and K at the beginning of the optimization process, and the reference data can come from any source, such as experiments, commercial software, or a combination. In addition,

- The SCF is not limited to the LTI system and clarifies the treatment of fully nonlinear governing equations.
- The SCF disconnects the definition of the reduced-order governing equations from the governing equations used to generate the reference data.
- The SCF generalizes the treatment of the goal functional. It thus allows the goal functional to be defined as any linear or nonlinear function of the reference data u. That is, the goal functional g can be taken any form, like $g = u^2$, $g = \sin(u)$.
- The SCF allows us to freely define the boundary conditions that are used for the reduced-order model.

2.2. OPTIMIZATION ALGORITHM

This section is devoted to the description of the optimization algorithm used to solve the optimization problem for the determination of the projection basis functions ϕ. First, the methodology for an unconstrained optimization problem will be presented. The translation of our constrained optimization problem to an unconstrained one will be illustrated afterwards.

2.2.1. OPTIMIZATION TECHNIQUES

A general formulation for an unconstrained minimization problem is

$$\min_x f(x), \tag{2.20}$$

where $x \in \mathbb{R}^n$ is a real vector with $n \geq 1$ components and the objective function f: $\mathbb{R}^n \to \mathbb{R}$ is a function of x. Textbooks such as [79–81] provide an excellent introduction into the algorithms which can be used to solve (2.20). Most are iterative in nature. That is, beginning with an initial guess x_0, they generate a sequence of iterates x_k which terminate when either no more progress can be made or when it seems that a solution point has been approximated with sufficient accuracy. The algorithms use information about the objective function f at x_k, and possibly also information from earlier iterates x_0, x_1, $\cdots$, x_{k-1} to decide how to move from current iterate x_k to the next. Usually, this information is used to find a new iterate x_{k+1} with a smaller objective function value than x_k, i.e. $f(x_{k+1}) < f(x_k)$. However, non-monotone algorithms that don't insist on a decrease in f at every step (see [82, 83]) also exist, and they require $f(x_{k+m}) < f(x_k)$ where m is a prescribed number of iterations.

There are two fundamental strategies for moving from the current iterate x_k to the next iterate x_{k+1}, the *line search* and *trust region* strategies. In the *line search* strategy, the algorithm firstly chooses a direction p_k, and searches along this direction from the current iterate x_k for a new iterate x_{k+1} with a smaller objective function value. Then, the step length, α, to move along p_k can be found by approximately solving the the following minimization problem:

$$\min_{\alpha>0} f(x_k + \alpha p_k) \tag{2.21}$$

If we can solve (2.21) exactly, we will derive the maximum benefit from the direction p_k. However, obtaining an exact minimization solution is usually expensive and not necessary. The line search algorithm generates a limited number of trial step lengths until it finds one that loosely approximates the minimum of (2.21). At the new point, a new search direction and step length need to be computed, then the process is repeated.

In the second strategy, *trust region*, the information gathered about f is used to construct a model function m_k whose behavior near the current point x_k is similar to that of the actual objective function f. We restrict the search for a minimizer of m_k to a limited region, the called trust region, around x_k so that the model function m_k is a good approximation of f. In other words, we find the candidate step p by approximately solving the following subproblem:

$$\min_p m_k(x_k + p) \qquad \text{where } x_k + p \text{ lies inside the trust region} \tag{2.22}$$

The trust region is usually a ball defined by $\|p\|_2 \leq \Delta$, where the scalar $\Delta > 0$ is the trust region radius. The size of the trust region is of great importance. If it is too small, the algorithm is inefficient since it does not get the chance to make substantial steps towards the minimum. On the other hand, if it is too large, the model function m_k will deviate excessively from the objective function f.

Both *line search* and *trust region* methods can generate steps which minimize a quadratic model function of the objective function:

$$m_k(x_k + p_k) = f_k + p_k^T \nabla f_k + \frac{1}{2} p_k^T B_k p_k \tag{2.23}$$

where f_k, ∇f_k and B_k are a scalar, vector and matrix, respectively. f_k, ∇f_k are chosen to be the objective function and its gradient at the iterate x_k, so that m_k and f are in agreement to the first order at the current iterate x_k. The matrix B_k can be either the Hessian matrix $\nabla^2 f_k$ or one approximation of $\nabla^2 f_k$. *Line search* methods use (2.23) to generate a search direction and focus their efforts on finding a suitable step length α along this direction [79, 81]. *Trust region* methods choose the direction and length of the step simultaneously by minimizing the model function (2.23) subject to the trust region constraint: $\|p_k\|_2 \leq \Delta$.

Both *line search* and *trust region* methods can be effectively applied to complex optimization problems. But when the matrix B_k is nearly singular, *line search* algorithms require many iterations and give only a small reduction in the objective function. While in this condition, it can be very effective to use *trust region* methods. Therefore *trust region* methods are more preferable.

To solve the optimization problem (1.47)-(1.50), Bui-Thanh *et al.* [69] adopted a *Trust-Region Inexact-Newton Conjugate-Gradient* method, which combines the rapid locally-quadratic convergence rate properties of Newton's method, the effectiveness of trust region globalization for treating ill-conditioned problems, and the Eisentat-Walker idea of preventing oversolving [84]. Therefore, to solve the optimization problem (2.8)-(2.11), we also employ the same optimization algorithm that will be illustrated in the following section.

2.2.2. TRUST-REGION INEXACT-NEWTON CONJUGATE-GRADIENT

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. The objective function is approximated by a quadratic model m_k, (2.23). The candidate step p_k is found by approximately solving the following subproblem:

$$\min_{p_k} m_k(p_k) = f_k + p_k^T \nabla f_k + \frac{1}{2} p_k^T \nabla^2 f_k p_k \quad \text{subject to } \|p_k\|_2 \leq \Delta \tag{2.24}$$

The necessary condition for stationarity of a function is that its first derivative vanishes at the extrema, that is,

$$\begin{aligned} \frac{\partial m_k}{\partial p_k} = \nabla f_k + \nabla^2 f_k p_k = 0 \implies \\ \nabla^2 f_k p_k = -\nabla f_k \end{aligned} \tag{2.25}$$

which is a symmetric linear system due to the definition of Hessian matrix $\nabla^2 f_k$. These are known as "Newton's equations", and can be solved directly, i.e.

$$p_k = -(\nabla^2 f_k)^{-1} \nabla f_k. \tag{2.26}$$

The Newton method has the advantage of being rapidly convergent to the solution provided that we start with a sufficiently good initial guess x_0. However, it can be expensive to compute (2.26) directly when the system is large and the direct factorization of the Hessian matrix $\nabla^2 f_k$ is intractable. Therefore we use an iterative method and solve (2.25) only approximately, that is to say, we don't directly compute the inverse of $\nabla^2 f_k$. This

class of methods are generally referred to in the literature, e.g.[85], as *Inexact-Newton methods*, which compute an approximate solution (p_k) to the Newton's equations (2.25) in some manner such that

$$\|r_k\| < \epsilon_k,$$

where ϵ_k is a specified tolerance, and r_k is the residual, defined as

$$r_k = -\nabla f_k - \nabla^2 f_k p_k.$$

Here, we will solve (2.25) using the Conjugate-Gradient (CG) method with modifications to handle negative curvature in the Hessian matrix $\nabla^2 f_k$ and globalize the algorithm using a trust-region scheme.

Trust-region algorithm

A key part of trust-region algorithms is the strategy for choosing the trust-region radius Δ_k at each iteration. Since we want to make the model function to be a good approximation of the objective function in the trust region, we determine Δ_k based on the agreement between the model function and the objective function at previous iterations. Given a step p_k, we define a ratio

$$\rho_k = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)}, \tag{2.27}$$

where the numerator is called the actual reduction and the denominator is called the predicted reduction. The predicted reduction is always non-negative since $m_k(p_k)$ is the minimum of m_k over a trust region that includes $p_k = 0$. Thus, if ρ_k is negative, the new value of the objective function is larger than the current value, i.e. $f(x_k + p_k) > f(x_k)$, then the step p_k must be rejected. On the other hand, if ρ_k is non-negative, there are three possible responses:

- If ρ_k is close to 1, there is good agreement between the model function and the objective function over this step. Thus we can take a larger step for the next iteration by expanding the trust region.
- If ρ_k is close to 0, the size of the trust region should be reduced.
- If ρ_k is between 1 and 0, the size of the trust region is not changed.

In the literature [79], the trust region algorithm is described as:

Trust Region Algorithm

Given $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, $\eta \in \left[0, \frac{1}{4}\right)$
for $k = 0, 1, 2, \cdots$

Obtain p_k by approximately solving (2.25)
Evaluate ρ_k from (2.27)

if $\rho_k < \frac{1}{4}$
$\Delta_{k+1} = \frac{1}{4}\Delta_k$ %reduce trust region radius
else
if $\rho_k > \frac{3}{4}$ and $\|p_k\|_2 \geq \Delta_k$
$\Delta_{k+1} = \min\left(2\Delta_k, \hat{\Delta}\right)$ %enlarge trust region radius
else
$\Delta_{k+1} = \Delta_k$ %keep trust region radius
end if
end if

if $\rho_k > \eta$
$x_{k+1} = x_k + p_k$ %enough reduction made, advance step
else
$x_{k+1} = x_k$ %do not advance
end if

end for

Here, $\hat{\Delta}$ is the maximum bound on the step lengths. Note that the trust region radius is increased only if $\|p_k\|_2$ reaches the boundary of the trust region. If the step stays inside the trust region, the current trust region radius Δ_k is apparently not interfering with the progress of the algorithm, so we keep it for the next iteration. Furthermore we accept the step p_k only if ρ_k is at least larger than a minimum threshold η.

To turn the trust region algorithm described above into a practical algorithm, we need to focus on solving the linear system (2.25). When $\nabla^2 f_k$ is symmetric and positive definite, a very efficient way to solve large linear systems iteratively is to use the CG method, described in Appendix B. Symmetry is not an issue since $\nabla^2 f_k$ is symmetric by definition, but unfortunately the positive definiteness can not be guaranteed. Therefore as mentioned earlier, we need to modify the conventional CG algorithm (see Appendix B), which constructs the step iteratively as linear combination of search directions, to be able to handle this situation.

Conjugate gradient algorithm with modifications

The modified conjugate gradient algorithm used is due to Steihaug [86]. This algorithm is terminated when one of the following conditions is encountered:

- The residual at the current iteration is lower than a given tolerance, meaning that the algorithm has reached the minimum of the the model function.
- A direction of negative curvature is found, i.e. the Hessian matrix $\nabla^2 f_k$ is not positive-definite. In this case we move to the boundary since the minimum of

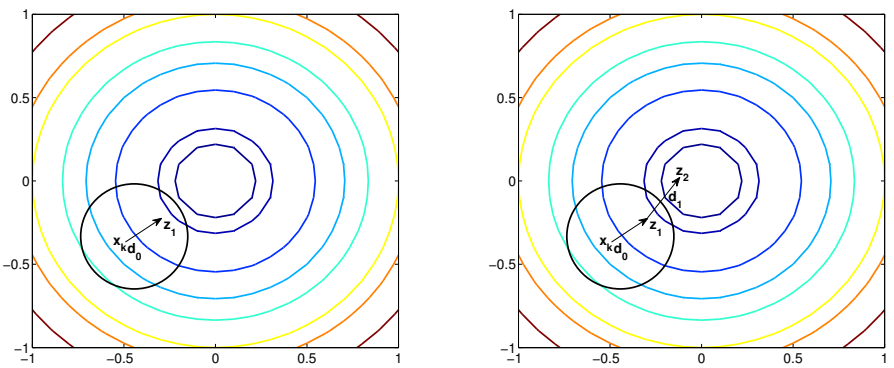
the model function can not be found for such a upwardly convex function.

- The next iteration violates the trust region bound, $\|p_k\|_2 \geq \Delta_k$, in which case we determine the minimum of the model function along the search direction such that $\|p_k\|_2 = \Delta_k$.

2

The last two termination conditions are what differentiates this modified conjugate gradient algorithm from the conventional CG algorithm described in Appendix B.

With the assistance of Fig.2.1, it is easy to understand the three termination conditions. We denote the search directions by d_j and the sequence of iterates produced by the CG algorithm by z_j. Fig.2.1(a) shows the first iteration, starting from 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 a given tolerance ϵ_k, then the CG procedure is stopped and the step $p_k = z_1$ is returned; otherwise, the iteration is continued.



(a) Case 1 (b) Case 2

Figure 2.1: The contour lines of the objective function and the trust region of the model function together with a sequence of CG iterates

In Fig.2.1(b), z_2 lies outside the trust region. Since the trust region constraint is violated we compute the sub-step length $\tau > 0$ such that we reach the trust region bound along the direction d_1, hence $p_k = z_1 + \tau d_1$ satisfies $\|p_k\|_2 = \Delta_k$, and then we return p_k. The last possibility is a negative curvature is encountered. In this situation, similarly, we move to the boundary of the trust region along d_j and calculate $p_k = z_j + \tau d_j$ such that $\|p_k\|_2 = \Delta_k$, and $m_k(p_k)$ is the minimum of the model function along d_j.

When encountering negative curvature or violating the trust region bound, i.e. encountering one of the last two termination conditions presented above, the evaluation of τ is required. It can be calculated by taking the positive root of the following quadratic equation:

$$\begin{aligned} \|p_k\|_2^2 = \Delta_k^2 &\Leftrightarrow (z_j + \tau d_j)^T (z_j + \tau d_j) = \Delta_k^2 \\ &\Leftrightarrow \tau^2 d_j^T d_j + 2\tau d_j^T z_j = \Delta_k^2 - z_j^T z_j \end{aligned} \tag{2.28}$$

Here, we use trust region algorithm in combination of the Steihaug CG algorithm, where Steihaug CG method solves the step p_k used in the trust region algorithm by approximately solving (2.25). The Steihaug CG algorithm is given by

Steihaug CG Algorithm

Given tolerance $\epsilon_k > 0$
Set $z_0 = 0$, $r_0 = -\nabla f_k - \nabla^2 f_k p_k = -\nabla f_k - \nabla^2 f_k z_0 = -\nabla f_k$, $d_0 = r_0$

if $\|r_0\|_2 < \epsilon_k$
  return $p_k = z_0 = 0$
end if

for $j = 0, 1, 2, \cdots$
  if $d_j^T \nabla^2 f_k d_j \leq 0$
    Compute $\tau > 0$ such that $\|z_j + \tau d_j\|_2 = \Delta_k$
    return $p_k = z_j + \tau d_j$
  end if
  Compute $\alpha_j = r_j^T r_j / d_j^T \nabla^2 f_k d_j$
  Compute $z_{j+1} = z_j + \alpha_j d_j$
  if $\|z_{j+1}\|_2 \geq \Delta_k$
    Compute $\tau > 0$ such that $\|z_j + \tau d_j\|_2 = \Delta_k$
    return $p_k = z_j + \tau d_j$
  end if
  Compute $r_{j+1} = r_j - \alpha_j \nabla^2 f_k d_j$
  if $\|r_{j+1}\|_2 < \epsilon_k$
    return $p_k = z_{j+1}$
  end if
  Compute $\beta_{j+1} = r_{j+1}^T r_{j+1} / r_j^T r_j$
  Compute $d_{j+1} = r_{j+1} + \beta_{j+1} d_j$
end for

2.3. SOLVING THE CONSTRAINED OPTIMIZATION PROBLEM

The FDF and SCF lead to constrained optimization problems, while the algorithm introduced in the previous section is for unconstrained optimization problems. The FDF and SCF constrained optimization problems can be converged into equivalent unconstrained optimization problems in the projection basis Φ variables by eliminating the state variables α and the **State Equations**. That is, we replace $\min_{\Phi,\alpha} \mathcal{G}(\Phi, \alpha)$ with $\min_{\Phi} \tilde{\mathcal{G}}(\Phi, \alpha(\Phi))$, where the dependence of α on Φ is implicit through the **State Equations**.

The gradient of the unconstrained function $\tilde{\mathcal{G}}$ with respect to Φ, $\nabla\tilde{\mathcal{G}}(\Phi)$, as required by Inexact-Newton methods, can be computed efficiently by an adjoint method. The gradient is given by the first variation of Lagrangian functional with respect to Φ, $\delta\mathcal{L}_\Phi$, when the α satisfy the **State Equations** and (λ, μ) satisfy the **Adjoint Equations**. The procedure to compute the gradient is shown in Fig.2.2. First, determine $\alpha(t)$ by solving the **State Equations** with known Φ. Second, determine $\lambda(t)$ and μ by solving the **Ad-**

joint Equations with known Φ and $\alpha(t)$ obtained in the first step. Finally, calculate the gradient through the computed variables $(\alpha(t), \lambda(t), \mu)$ and known projection basis Φ.

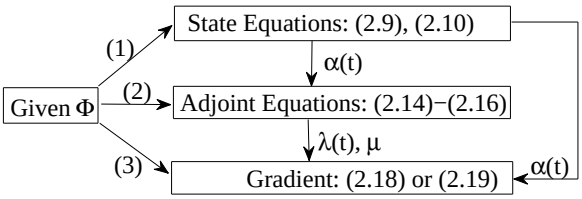


Figure 2.2: Procedure to compute the gradient, taking SCF as example

To practically implement the optimization algorithm presented in the above section, one thing that needs to be addressed is the calculation of the (approximate) Hessian matrix $\nabla^2 f_k$. By looking into the Steihaug CG algorithm, we see that each iteration requires us to supply the Hessian-vector product $\nabla^2 f_k d_j$ instead of only the Hessian matrix $\nabla^2 f_k$. That is, we can compute an approximation of the Hessian-vector product $\nabla^2 f_k d_j$ rather than evaluate the exact Hessian matrix $\nabla^2 f_k$. Considering a Taylor expansion of the gradient ∇f_k:

$$\nabla f(x_k + \theta d_j) = \nabla f(x_k) + \theta \nabla^2 f(x_k) d_j + \mathcal{O}(\theta^2) \Rightarrow$$
$$\nabla^2 f(x_k) d_j \approx \frac{\nabla f(x_k + \theta d_j) - \nabla f(x_k)}{\theta}$$

The latter is a $\mathcal{O}(\theta)$ accurate forward differencing formula, where θ is a small positive number. The Hessian-vector product is then approximated by the directional derivative of the gradient with respect to the search direction d_j. In our case, the Hessian-vector product is:

$$\nabla^2 f(x_k) d_j = \nabla^2 \tilde{\mathcal{G}}(\Phi) d_j \approx \frac{\nabla \tilde{\mathcal{G}}(\Phi + \theta d_j) - \nabla \tilde{\mathcal{G}}(\Phi)}{\theta} \tag{2.29}$$

As mentioned earlier in this section, to obtain the gradient $\nabla \tilde{\mathcal{G}}(\Phi)$, we need to solve the **State Equations** and **Adjoint Equations** where Φ is given or calculated in a previous iteration. Thus, to obtain $\nabla \tilde{\mathcal{G}}(\Phi + \theta d_j)$, we need to solve another set of **State Equations** and **Adjoint Equations**, where $(\alpha(t), \lambda(t), \mu)$ are calculated using $\Phi + \theta d_j$ instead of Φ. Therefore, the optimization algorithm requires solutions of a pair of **State Equations** and **Adjoint Equations** at each CG iteration.

One last thing to be noticed is that the optimization problem used to identify an optimal projection basis has no guarantee of convexity, implying that there is also no guarantee that a purely local optimum will converge to the global optimum since such algorithms usually tend to be trapped at local minima/maxima. Accordingly, the choice of initial guess for the projection basis Φ plays a crucial role in the final output of the optimization algorithm.

2.4. STRUCTURE OF PROJECTION BASIS Φ

The formulation defined by the constrained minimization problem (2.8)-(2.11) provides a continuous mathematical definition of the desired optimal projection basis. However, during implementation, the projection basis functions are defined in discrete forms. Therefore, we will illustrate this in the following part.

Since the reference data are either produced by numerical simulations or experiments, they always have a finite character. That is, the reference data is given in a discrete form. It can thus be straightforward to define the projection basis functions in the same discrete space. Thus, each projection basis function can be defined as a vector of which the dimension is N (the number of discrete points in space), see Fig.2.3.

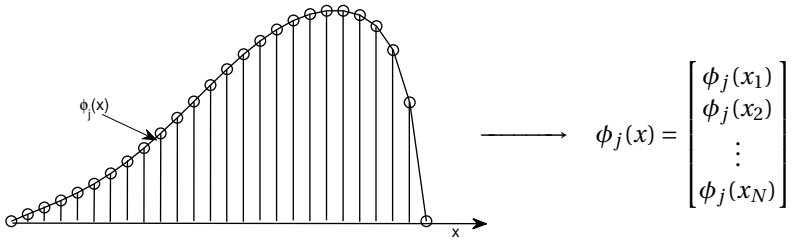


Figure 2.3: First definition of each projection basis function

Using the vector ϕ, the number of optimization variables is equal to $N \times \mathrm{nM}$ —— the number of discrete points in space multiplied by the desired number of projection basis functions. However, in large-scale problems, N becomes very large, the optimization problem then can not be tractable. Therefore, it is better to assume that each projection basis function can be represented as a linear combination of pre-specified functions (see Fig.2.4):

$$\phi_j = \sum_{l=1}^{\mathrm{nB}} \mathrm{Coe}_{jl} \psi_l \tag{2.30}$$

where nB is the desired number of pre-specified functions and the coefficients Coe_{jl} are the optimization variables in the modified optimization problem. This representation reduces the number of optimization variables from $N \times \mathrm{nM}$ to $\mathrm{nB} \times \mathrm{nM}$ since nB is usually much smaller than N (i.e. $\mathrm{nB} \ll N$) when using appropriate pre-specified functions. As a consequence, neither the gradient computation nor the optimization step computation scale with the full system size N.

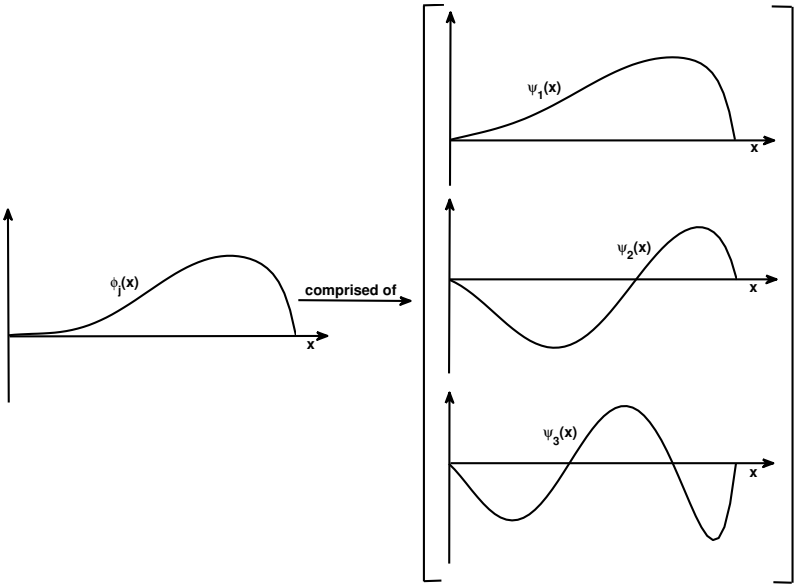


Figure 2.4: Second definition of each projection basis function

For compactness, the following nomenclature is used from now on,

primary basis function-ϕ: projection basis function

primary basis: projection basis

secondary basis function-ψ: pre-specified function

secondary basis: basis consisting of pre-specified functions

P-type primary basis: projection basis where ϕ_j is defined as a vector (Fig.2.3)

F-type primary basis: projection basis where ϕ_j is represented as a linear combination of pre-specified functions ψ_l (Fig.2.4)

Note that the **P-type primary basis functions** are similar to **F-type basis functions** when choosing the delta functions as the pre-specified functions. That is, the **P-type primary basis** functions can also be seen as linear combinations of delta functions, where the coefficients for the delta function are the elements in vectors, i.e.

$$\phi_j(x) = \sum_{l=1}^{N} \phi_j(x_l)\delta(x) \qquad \text{and} \qquad \delta(x) = \begin{cases} 1 & x = x_l \\ 0 & x \neq x_l \end{cases}$$

In addition, considering the above equation, the pre-specified basis functions used to construct the **F-type primary basis** can also be chosen to be local functions rather than global functions that are shown in Fig.2.4. For example, the pre-specified basis functions

can be the local shape functions that are used in Finite Element Method. Thus, parallel implementation of solving the optimization problem could be employed.

2.5. SUMMARY

Like the FDF, the SCF is a method that has advantages by targeting the primary basis to a particular goal functional, and by incorporating the reduced-order governing equations as constraints in the primary basis derivation. Moreover, the SCF addresses the limitations of the FDF in terms of applicability. Through the derivation for the SCF, it is seen that the SCF clarified the treatments of nonlinear governing equations and nonlinear goal functionals; the SCF disconnects the reduced-order governing equations from the governing equations used to generate the reference data; more possibilities of treatments of the boundary conditions are clarified using the SCF.

The constrained minimization problem is first transformed into an equivalent unconstrained problem. Then a Trust-Region Inexact-Newton Conjugate-Gradient algorithm is used to solve the unconstrained minimization problem. As there is no guarantee of convexity, the choice of initial guess for the primary basis can be very important.

In terms of the number of the optimization variables, we introduced two types of primary basis: **P-type primary basis** and **F-type primary basis**. In some sense, the **P-type primary basis** can be seen as a special case of the **F-type primary basis**. When the length of each primary basis function (i.e. the number of discrete points in space) is large, the use of **F-type primary basis** makes the implementation of solving the optimization problem tractable.

3

APPLICATION I: 1D PROBLEMS

In this chapter, we will investigate the behavior of the SCF using a set of one-dimensional problems. These include ones where (1) The governing equation is a linear partial differential equation—the linear advection-diffusion equation. (2) The governing equation is a nonlinear partial differential equation—the Burgers equation. (3) A model equation is used which only approximates governing equation and the dynamics of the reference dataset. We also demonstrate how considering purely local goal functional in problems with limited domains of dependence can introduce multiple extrema in the optimization problem, and discuss how optimal basis functions can be found in such circumstances. In the meantime, the computational costs of the POD and SCF are compared.

Results of this chapter has been published in *Int. J. Numer. Meth. Engng* **105**, 464-480 (2016) [87].

In this chapter, for the three applications of the SCF, a **P-type primary basis** is used unless mentioned otherwise. For clarity, in this chapter, we denote

- $\Phi \Longleftrightarrow$ the set of primary basis functions
- POD modes $\Longleftrightarrow$ primary basis functions obtained using the POD
- SCF modes $\Longleftrightarrow$ primary basis functions obtained using the SCF

3.1. DEFINITION OF INITIAL GUESS FOR Φ

In Chapter 2, we have mentioned that the choice of the initial guess for Φ is very important since the optimization problem might be non-convex. As in [69], two strategies are considered,

- (S1): Choosing the first nM POD modes as an initial guess.
- (S2): The initial guess is chosen to be the solution of the optimization problem for nM $-$ 1 primary basis functions plus the nMth POD mode.

In this chapter, we adopt strategy 'S1' except where noted.

3.2. LINEAR PARTIAL DIFFERENTIAL EQUATION EXAMPLE

We first consider the one-dimensional linear advection-diffusion equation. The initial-boundary value problem is given by

$$u_t + a u_x - k u_{xx} = f \quad \text{in } x \in [0,\, 1], \tag{3.1a}$$

$$u = 0 \quad \text{at } x = 0,\ x = 1, \tag{3.1b}$$

$$u = u_0 \quad \text{in } x \in [0,\, 1] \text{ for } t = 0, \tag{3.1c}$$

where a is the advection speed, k is the diffusion coefficient, f represents the force term, and u_0 represents the initial condition for u. For the case to be investigated,

- $a = 2,\ k = 0.1,\ f = 1.$
- $u_0 = \sin(2\pi x).$

3.2.1. REFERENCE DATA

To implement the SCF approach, we need a reference dataset to calculate the goal functional in a full space (i.e. g) and the POD modes.

We provide the reference data by solving the equation (3.1) numerically, using the Finite Difference Method. We use a uniform grid with N_x nodes and here $N_x = 51$. The spatial derivative is approximated by central finite difference formulas, and then the semi-discretised equation of (3.1a) can be written as

$$\frac{du_i^n}{dt} + a\frac{u_{i+1}^n - u_{i-1}^n}{2\Delta x} - k\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2} = f, \tag{3.2}$$

where the superscript–n indicates the time step, the subscript–i indicates the spatial position and $i = 2, 3, \cdots, N_x - 1$, and Δx is the length of the subinterval of the grid.

A third order Runge-Kutta explicit multistage method is used to advance the solutions in time with $\Delta t = 0.001$, where the new time level–u^{n+1} is computed as a linear combination of solutions at intermediate stages. By writing $u_t = F(u(t))$ the stages have the form

$$\begin{aligned} u^{(1)} &= u^n, \\ u^{(2)} &= u^n + \Delta t F(u^{(1)}), \\ u^{(3)} &= u^n + \Delta t \left[\frac{1}{4} F(u^{(1)}) + \frac{1}{4} F(u^{(2)}) \right], \\ u^{n+1} &= u^n + \Delta t \left[\frac{1}{6} F(u^{(1)}) + \frac{1}{6} F(u^{(2)}) + \frac{2}{3} F(u^{(3)}) \right]. \end{aligned} \tag{3.3}$$

Fig.3.1 shows solution profiles at different times. From $t = 0.75$ to $t = 1.125$ the solution slowly changes, then we stop the Runge-Kutta procedure at $t = 1.125$.

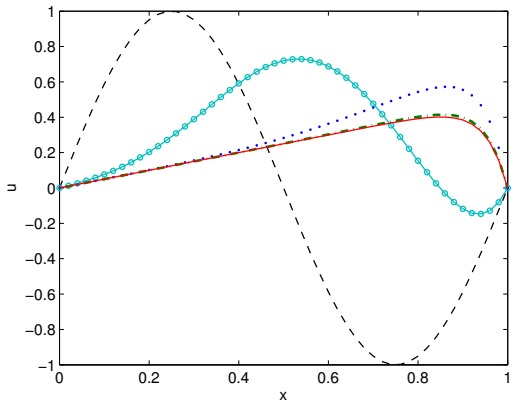


Figure 3.1: Solution of advection-diffusion equation at $t = 0$, 0.15, 0.45, 0.75 and 1.125s

3.2.2. POD ROM

The POD modes are extracted from $N_s = 1126$ snapshots using eigenvalue decomposition. The percentage of energy captured by each of the POD modes appears in Fig.3.2 (left), over 99% of the energy is contained in the first six POD modes. Fig.3.2 (right) shows the fist six POD modes.

We can construct a reduced-order model using a Galerkin projection framework. Projecting the advection-diffusion equation (3.1a) into the subspace spanned by POD modes (ϕ_i) results in

$$\int_0^1 \phi_i \left(\hat{u}_t + a\hat{u}_x - k\hat{u}_{xx} \right) dx = \int_0^1 \phi_i f \, dx, \qquad i = 1, 2, \cdots, \tag{3.4}$$

where $\hat{u}$ is expanded in a series of POD modes as shown in (2.4). Keeping the first nM

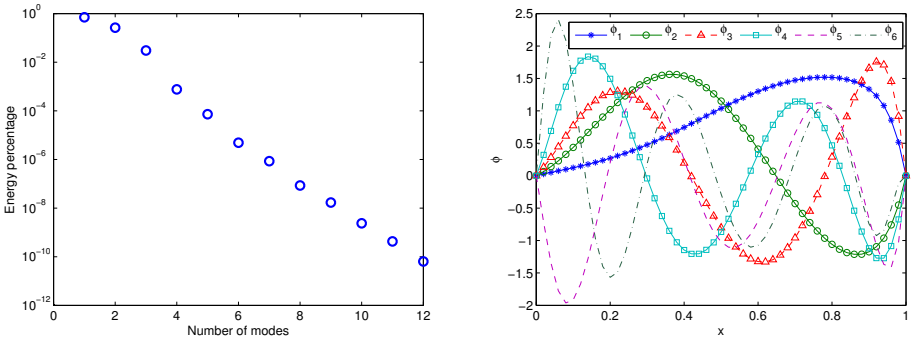


Figure 3.2: Energy percentage and the first six POD modes

POD modes leads to a POD ROM

$$\sum_{j=1}^{\mathrm{nM}} \dot{\alpha}_j \int_0^1 \phi_i \phi_j \, dx + \sum_{j=1}^{\mathrm{nM}} \alpha_j \int_0^1 \left(a\phi_i \phi_{jx} - k\phi_i \phi_{jxx} \right) dx = \int_0^1 \phi_i f \, dx, \quad i \in [1, \mathrm{nM}]. \tag{3.5}$$

This system determines the evolution of the amplitudes of POD modes, α_i. The initial value of α (i.e. the value of α at $t = 0$), α^0, is calculated by solving the following system:

$$\sum_{j=1}^{\mathrm{nM}} \alpha_j^0 \int_0^1 \phi_i \phi_j \, dx = \int_0^1 \phi_i u_0 \, dx \tag{3.6}$$

3.2.3. SEMI-CONTINUOUS FORMULATION

CONSTRAINED MINIMIZATION PROBLEM

For this application of the SCF, the linear advection-diffusion equation is used in (2.8)-(2.11) to arrive at the constrained minimization problem:

$$\arg_{\Phi,\alpha} \min \mathcal{G} = \frac{1}{2} \int_0^{t_f} \int_0^1 (g - \hat{g})^2 \, dx \, dt + \frac{\beta}{2} \sum_{i,j=1}^{\mathrm{nM}} \left(\delta_{ij} - \int_0^1 \phi_i \phi_j \, dx \right)^2 \tag{3.7a}$$

subject to

$$\sum_{j=1}^{\mathrm{nM}} \dot{\alpha}_j \int_0^1 \phi_i \phi_j \, dx + \sum_{j=1}^{\mathrm{nM}} \alpha_j \int_0^1 \left(a\phi_i \phi_{jx} - k\phi_i \phi_{jxx} \right) dx = \int_0^1 \phi_i f \, dx \tag{3.7b}$$

$$\sum_{j=1}^{\mathrm{nM}} \alpha_j^0 \int_0^1 \phi_i \phi_j \, dx = \int_0^1 \phi_i u_0 \, dx \tag{3.7c}$$

$$\hat{g} = g\left(\sum_{j=1}^{\mathrm{nM}} \alpha_j \phi_j \right) \tag{3.7d}$$

The primary basis is assumed to satisfy $\Phi(0) = \Phi(1) = 0$ due to the homogeneous Dirichlet boundary condition for u, (3.1b).

OPTIMALITY SYSTEM

The optimality conditions of problem (3.7) can be obtained by defining the Lagrangian functional. For this particular application, the Lagrangian functional is written as:

$$\begin{aligned}\mathscr{L} =& \frac{1}{2}\int_0^{t_f}\int_0^1 (g-\hat{g})^2 dx dt + \frac{\beta}{2}\sum_{i,j=1}^{\mathrm{nM}}\left(\delta_{ij} - \int_0^1 \phi_i\phi_j dx\right)^2 \\ &+ \int_0^{t_f}\sum_{i=1}^{\mathrm{nM}}\lambda_i\left[\sum_{j=1}^{\mathrm{nM}}\dot{\alpha}_j\int_0^1\phi_i\phi_j dx + \sum_{j=1}^{\mathrm{nM}}\alpha_j\int_0^1\left(a\phi_i\phi_{jx} - k\phi_i\phi_{jxx}\right)dx - \int_0^1\phi_i f dx\right]dt \\ &+ \sum_{i=1}^{\mathrm{nM}}\mu_i\left(\sum_{j=1}^{\mathrm{nM}}\alpha_j^0\int_0^1\phi_i\phi_j dx - \int_0^1\phi_i u_0 dx\right)\end{aligned} \tag{3.8}$$

The optimality system is then derived by taking the first variation of the Lagrangian functional (3.8) with respect to λ_i, μ_i, α_j, and ϕ_q.

State Equations As discussed in Section 2.1, the **State Equations** are the ROM (3.7b) with initial condition (3.7c).

Adjoint Equations Section 2.1 shows that the **Adjoint Equations** are defined by (2.14)-(2.16). To complete the **Adjoint Equations** for this problem, we need to derive these terms: $\frac{\partial E}{\partial \alpha_j}$, $\frac{\partial G_2^i}{\partial \alpha_j}$, $\frac{\partial G_2^i}{\partial \dot{\alpha}_j}$ and $\frac{\partial G_3^i}{\partial \alpha_j}$.

Term 1:

$$\begin{aligned}\frac{\partial E}{\partial \alpha_j} &= \frac{\partial}{\partial \alpha_j}\left\{(g-\hat{g})^2\right\} \\ &= -2(g-\hat{g})\frac{\partial \hat{g}}{\partial \alpha_j}\end{aligned}$$

Term 2[1]*:*

$$\begin{aligned}\frac{\partial G_2^i}{\partial \alpha_j} &= \frac{\partial}{\partial \alpha_j}\left\{-\sum_{j=1}^{\mathrm{nM}}\alpha_j\left(a\phi_{ix}\phi_j - k\phi_{ix}\phi_{jx}\right)\right\} \\ &= -a\phi_{ix}\phi_j + k\phi_{ix}\phi_{jx}\end{aligned}$$

Term 3:

$$\begin{aligned}\frac{\partial G_2^i}{\partial \dot{\alpha}_j} &= \frac{\partial}{\partial \dot{\alpha}_j}\left\{\sum_{j=1}^{\mathrm{nM}}\dot{\alpha}_j\phi_i\phi_j\right\} \\ &= \phi_i\phi_j\end{aligned}$$

Term 4:

$$\begin{aligned}\frac{\partial G_3^i}{\partial \alpha_j} &= \frac{\partial}{\partial \alpha_j}\left\{\sum_{j=1}^{\mathrm{nM}}\alpha_j^0\phi_i\phi_j - \phi_i u_0\right\} \\ &= \phi_i\phi_j\end{aligned}$$

[1] This derivation makes use of integration by parts, terms associated with boundary vanish because $\Phi|_0^1 = 0$.

Inserting the above equations into (2.14)-(2.16) leads to the **Adjoint Equations**:

$$-\sum_{i=1}^{\mathrm{nM}} \dot{\lambda}_i \int_0^1 \phi_j \phi_i \, dx - \sum_{i=1}^{\mathrm{nM}} \lambda_i \int_0^1 \left(a\phi_j \phi_{ix} - k\phi_{jx}\phi_{ix} \right) dx = \int_0^1 (g - \hat{g}) \frac{\partial \hat{g}}{\partial \alpha_j} dx \tag{3.9a}$$

$$\lambda_i(t_f) = 0 \tag{3.9b}$$

$$\mu_i = \lambda_i(0) \tag{3.9c}$$

Gradient According to (2.19), the individual terms of $\frac{\partial E}{\partial \phi_q}$, $\frac{\partial G_2^i}{\partial \phi_q}$, $\frac{\partial G_2^i}{\partial \phi_{qx}}$, and $\frac{\partial G_3^i}{\partial \phi_q}$ for this application are presented below,

Term 1:

$$\begin{aligned} \frac{\partial E}{\partial \phi_q} &= \frac{\partial}{\partial \phi_q} \left\{ (g - \hat{g})^2 \right\} \\ &= 2(\hat{g} - g) \frac{\partial \hat{g}}{\partial \phi_q} \end{aligned}$$

Term 2:

$$\begin{aligned} \frac{\partial G_2^i}{\partial \phi_q} &= \frac{\partial}{\partial \phi_q} \left\{ \sum_{j=1}^{\mathrm{nM}} \dot{\alpha}_j \phi_i \phi_j + a \sum_{j=1}^{\mathrm{nM}} \alpha_j \phi_i \phi_{jx} - \phi_i f \right\} \\ &= \sum_{j=1}^{\mathrm{nM}} \dot{\alpha}_j \phi_j + \dot{\alpha}_q \phi_i + a \sum_{j=1}^{\mathrm{nM}} \alpha_j \phi_{jx} - f \end{aligned}$$

Term 3[2]*:*

$$\begin{aligned} \frac{\partial G_2^i}{\partial \phi_{qx}} &= \frac{\partial}{\partial \phi_{qx}} \left\{ \sum_{j=1}^{\mathrm{nM}} \alpha_j \left(a\phi_i \phi_{jx} + k\phi_{ix}\phi_{jx} \right) \right\} \\ &= a\alpha_q \phi_i + k \sum_{j=1}^{\mathrm{nM}} \alpha_j \phi_{jx} + k\alpha_q \phi_{ix} \end{aligned}$$

Term 4:

$$\begin{aligned} \frac{\partial G_3^i}{\partial \phi_q} &= \frac{\partial}{\partial \phi_q} \left\{ \sum_{j=1}^{\mathrm{nM}} \alpha_j^0 \phi_i \phi_j - \phi_i u_0 \right\} \\ &= \sum_{j=1}^{\mathrm{nM}} \alpha_j^0 \phi_j + \alpha_q^0 \phi_i - u_0 \end{aligned}$$

[2] Note that to avoid higher-order (higher than second-order) spatial derivatives appearing in the gradient, firstly we reduce ϕ_{jxx} to ϕ_{jx} through integration by parts, and terms associated with boundary vanish because $\Phi|_0^1 = 0$.

Putting all the above into (2.19) we have the formulation for the **Gradient**:

$$\begin{aligned}\delta\mathscr{L}_{\phi_q} = &\int_0^{t_f}(\hat{g}-g)\frac{\partial\hat{g}}{\partial\phi_q}dt + 2\beta\sum_{j=1}^{\mathrm{nM}}\phi_j\left(\int_0^1\phi_q\phi_j dx-\delta_{qj}\right)\\ &+\int_0^{t_f}\left(\lambda_q\sum_{j=1}^{\mathrm{nM}}\dot{\alpha}_j\phi_j+\dot{\alpha}_q\sum_{i=1}^{\mathrm{nM}}\lambda_i\phi_i+a\lambda_q\sum_{j=1}^{\mathrm{nM}}\alpha_j\phi_{jx}-a\alpha_q\sum_{i=1}^{\mathrm{nM}}\lambda_i\phi_{ix}\right.\\ &\left.-k\lambda_q\sum_{j=1}^{\mathrm{nM}}\alpha_j\phi_{jxx}-k\alpha_q\sum_{i=1}^{\mathrm{nM}}\lambda_i\phi_{ixx}-\lambda_q f\right)dt\\ &+\mu_q\sum_{j=1}^{\mathrm{nM}}\alpha_j^0\phi_j+\alpha_q^0\sum_{i=1}^{\mathrm{nM}}\mu_i\phi_i-\mu_q u_0\\ =&0\end{aligned} \tag{3.10}$$

DIRECT COMPARISON WITH FULLY-DISCRETE FORMULATION

In this application, the governing equation is the linear advection-diffusion equation, implying that the **State Equations** and the **Adjoint Equations** can be written in a matrix form similar to that in the FDF approach. In this case, the SCF can then be directly compared with the FDF.

To rewrite (3.7b) and (3.9a) in a matrix form, we need to use matrices to represent the evaluation of the spatial integral. As an example, we employ the trapezoidal rule. Given two continuous functions $c(x)$ and $e(x)$ defined in $[0, 1]$, using $c(x)e(x)$ as the integrand, the spatial integral can be approximated as:

$$\int_0^1 c(x)e(x)dx \approx \sum_{i=1}^{N_p-1}\bar{c}_i(x)\bar{e}_i(x)h$$

where N_p is the number of integration points, h is the length of the subinterval of integration, $\bar{c}_i(x)$ and $\bar{e}_i(x)$ represent the average values of $c(x)$ and $e(x)$ on each subinterval, i.e. $\bar{c}_i = \frac{c(x_i)+c(x_{i+1})}{2}$, $\bar{e}_i = \frac{e(x_i)+e(x_{i+1})}{2}$. By defining $\bar{\mathbf{c}} = [\bar{c}_1, \bar{c}_2, \cdots, \bar{c}_{N_p-1}]^T$, $\bar{\mathbf{e}} = [\bar{e}_1, \bar{e}_2, \cdots, \bar{e}_{N_p-1}]^T$ and $\mathbf{\Delta}$ as a diagonal matrix with h on the diagonal, the spatial integral can be written in the following form:

$$\int_0^1 c(x)e(x)dx \approx \bar{\mathbf{c}}^T\mathbf{\Delta}\bar{\mathbf{e}}$$

This allows (3.7b) and (3.9a) to be expressed as:

$$\mathcal{M}\dot{\alpha}+\mathcal{K}\alpha = \mathcal{b} \tag{3.11}$$

$$-\mathcal{M}^T\dot{\lambda}+\mathcal{K}_\lambda\lambda = \mathcal{b}_\lambda \tag{3.12}$$

where $\mathcal{M} = \bar{\Phi}^T\mathbf{\Delta}\bar{\Phi}$, $\mathcal{K} = a\bar{\Phi}^T\mathbf{\Delta}\bar{\Phi}_x - k\bar{\Phi}^T\mathbf{\Delta}\bar{\Phi}_{xx}$, $\mathcal{b} = \bar{\Phi}^T\mathbf{\Delta}\bar{f}$, $\mathcal{K}_\lambda = -a\bar{\Phi}^T\mathbf{\Delta}\bar{\Phi}_x + k\bar{\Phi}_x^T\mathbf{\Delta}\bar{\Phi}_x$, and $\mathcal{b}_\lambda = \bar{\Phi}^T\mathbf{\Delta}\overline{(g-\hat{g})\hat{g}_\alpha}$. Here the matrices $\mathcal{M}$, $\mathcal{K}$ and $\mathcal{K}_\lambda$ are only associated with the physical constants of the partial differential equation and the process of integration and differentiation. For a **P-type primary basis**, these have dimensions that are associated

with the number of nodes used to represent the discrete POD modes but are not necessarily associated with the dimension of the reference data. In contrast, the matrices $\mathcal{M}$ and $\mathcal{K}$ in the FDF are defined using a discrete form of the partial differential equation which reproduces the reference data. Specifically, in FDF $\mathcal{M} = \Phi^T M \Phi$ and $\mathcal{K} = \Phi^T K \Phi$, equation (1.48), where the M and K matrices represent components of a discrete partial differential equation approximation with the dimension of the reference data.

3.2.4. COMPARING SCF ROM WITH POD ROM

In our **P-type** optimization code, we evaluate the integrals in space using the trapezoidal rule mentioned earlier, and adopt central finite difference to represent spatial derivatives. The **State Eqautions** are integrated in time with the third order Runge-Kutta explicit multistage method used to produce reference data. The **Adjoint Equations** are integrated in time with backward Euler method[3].

When the goal functional is set to be the solution over the entire domain, i.e. $g = u\,(0 \leqslant x \leqslant 1)$, the SCF seeks to minimize the same error norm as POD interpolation. In this case, however, the error is measured using the solution of the ROM ($\hat{u}$). Typical results are shown in Fig.3.3. The SCF modes perform slightly better than POD modes, due to the enforcement of the model constraint. As we add more and more modes, the error for POD and SCF modes converge (Fig.3.3). This is due to the reduction in the approximation error of the ROM as we use more modes.

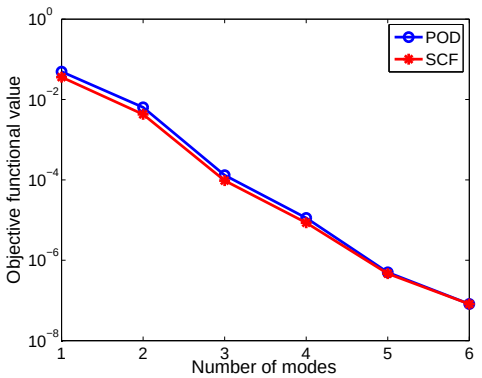


Figure 3.3: Error vs number of modes for $g = u\,(x \in [0, 1])$

Other goal functionals are more illustrative of the benefits of the SCF approach. In Fig.3.4, the error for POD and SCF modes are shown when a goal functional, $g = u$, is only considered in the region $0 \leqslant x \leqslant 0.5$. Here the SCF modes provide a clear improvement over the POD modes. For small nM, the error is reduced by almost an order of magnitude.

[3] Backward Euler method:

$$\frac{u^{n+1} - u^n}{\Delta t} = F(u^n, u^{n+1}) \tag{3.13}$$

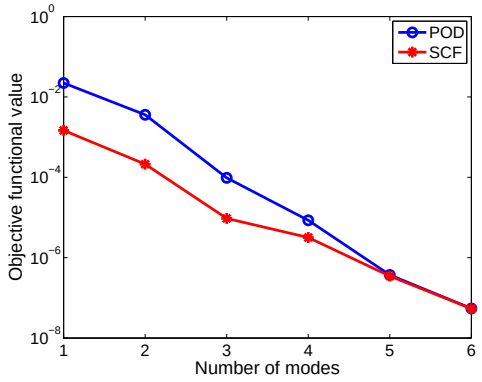


Figure 3.4: Error vs number of modes for $g = u\,(x \in [0, 0.5])$

Another functional of interest might be $g = u_x|_{x=0}$. Fig. 3.5 gives the corresponding results based on 'S1' and 'S2' initial guesses for Φ. In both cases, the SCF modes give a substantial improvement over the POD modes. As mentioned previously, however, the optimization algorithm used finds only local minima. Because of the 'S1' inappropriate initial guess (nM = 2) we only found a local, not global, minimum. The appearance of multiple solutions for the optimization problem is discussed further in the following section.

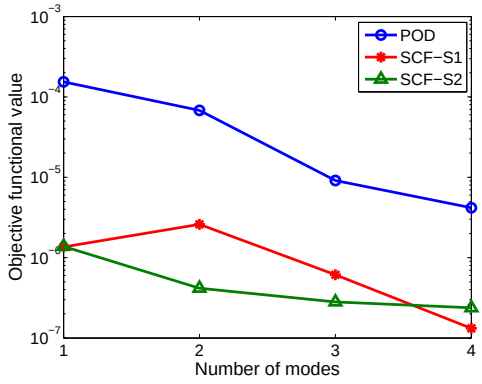


Figure 3.5: Error vs number of modes for $g = u_x\,(x = 0)$

For the nonlinear goal functional, we set $g = u^2$, where the region of interest is $0 \leqslant x \leqslant 1$ and $0 \leqslant t \leqslant 0.249$. In this case, the SCF modes are only modestly better than POD modes (Fig.3.6), as could be anticipated from the result for $g = u\,(0 \leqslant x \leqslant 1)$.

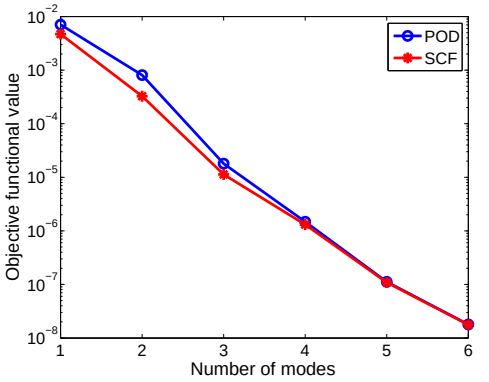


Figure 3.6: Error vs number of modes for $g = u^2$

3.3. NONLINEAR PARTIAL DIFFERENTIAL EQUATION EXAMPLE

As an example of the application of the SCF to a nonlinear partial differential equation, we choose one-dimensional Burgers equation with a forcing term as the governing equation. The nondimensionalized one-dimensional Burgers equation with initial-boundary conditions is given by

$$u_t + uu_x - \frac{1}{Re}u_{xx} = f, \tag{3.14a}$$

$$u(0,t) = u(1,t) = 0, \tag{3.14b}$$

$$u_0 = \sin(2\pi x), \tag{3.14c}$$

where f represents a constant force. For the results shown, $f = 1$.

3.3.1. REFERENCE DATA AND POD ROM

Reference data

To provide the reference data, the Finite Element Method (FEM) with linear basis functions [88, 89] is used to solve the Burgers equation (3.14a), where the weighting functions, $w(x)$, are chosen to be the same as the trial functions used to approximate the solution. Due to the homogeneous Dirichlet boundary condition, we require the trial functions to satisfy $w(0) = w(1) = 0$. We discretize the Burgers equation using a fine uniform grid, and the solution is advanced in time by a generalized-trapezoidal time-integration scheme which is up to second-order accurate [90, 91]. The semi-discrete variational formulation of the Burgers equation is then given by

$$\frac{1}{\Delta t}\big(w,\,[u]\big)_x - \frac{1}{2}\Big(w_x,\,\langle u\rangle^2_{a_v}\Big)_x + \frac{1}{Re}\big(w_x,\,\langle u_x\rangle_{a_v}\big)_x = (w,\,f)_x, \tag{3.15}$$

where

$$\begin{aligned}(.\,,\,.)_x &= \int_x (.)\cdot(.)dx,\\ \Delta t &= t^{n+1}-t^n,\\ [u] &= u^{n+1}-u^n,\\ \langle u\rangle_{a_v} &= a_v u^{n+1}+(1-a_v)u^n,\end{aligned} \tag{3.16}$$

i.e. $[\cdot]$ is a jump operator, $\langle\cdot\rangle_{a_v}$ is a weighted average, and $u(x,t)$ is the numerical solution. The superscript n represents the time level. We assume

$$\begin{aligned}w &= \sum_{A=1}^{N_{el}} N_A(x),\\ u &= \sum_{B=1}^{N_{el}} N_B(x)d_B(t),\end{aligned} \tag{3.17}$$

where N_A and N_B are the standard Finite Element linear basis functions, N_{el} is the number of elements, d_B are the unknown nodal amplitudes. We adopt the same definitions used in [90, 91], i.e.

$$\begin{aligned}K^{\Delta t}_{AB} &= \frac{1}{\Delta t}(N_A,\, N_B)_x,\\ K^{Re}_{AB} &= \frac{1}{Re}(N_{Ax},\, N_{Bx})_x,\\ K^{u}_{AB} &= -(N_{Ax},\, uN_B)_x,\\ F^{f}_{A} &= (N_A,\, f)_x.\end{aligned} \tag{3.18}$$

Then, after inserting Eq.(3.17) into Eq.(3.15) and grouping the linear, nonlinear and known right-hand side terms, we obtain

$$\left(A^{LIN}+A^{NL}\right)_{AB} d_B^{n+1} = F_A, \tag{3.19}$$

where

$$\begin{aligned}A^{LIN}_{AB} &= \left\{K^{\Delta t}+a_v K^{Re}+a_v(1-a_v)K^{u^n}\right\}_{AB},\\ A^{NL}_{AB} &= \frac{a_v^2}{2}K^{u^{n+1}}_{AB},\\ F_A &= \left\{K^{\Delta t}-(1-a_v)K^{Re}-\frac{1}{2}(1-a_v)^2K^{u^n}\right\}_{AB} d_B^n + F_A^f.\end{aligned} \tag{3.20}$$

The time discretization is second order accurate when $a_v = 1/2$ and the unconditionally stable range of values is $a_v \in (1/2, 1]$. From [91], we know that if the numerical simulations performed are always run at $a_v = 1/2 + \Delta t$ then second order accuracy is still preserved and stability is achieved. Therefore, to keep second order accuracy in our numerical simulations, we set $a_v = 1/2 + \Delta t$.

The assembly of (3.19) leads to a $N_{el} \times N_{el}$ nonlinear system of equations and therefore it cannot be solved by means of standard linear algebra techniques. Instead we adopt a Newton method, in which the residual is linearized through a Taylor expansion.

Let D^i denote the i^{th} iterative approximation of $D^{n+1} = \left[d_1^{n+1}\ d_2^{n+1}\ \cdots\ d_{N_{el}}^{n+1}\right]^T$ and define the i^{th} iterative residual as

$$G^i = \left(A^{LIN} + A^{NL}\right) D^i - F, \tag{3.21}$$

where A^{LIN} and A^{NL} are the $N_{el} \times N_{el}$ matrices resulting from the assembly of A_{AB}^{LIN} and A_{AB}^{NL}, $F \in \mathbb{R}^{N_{el}}$ is the vector resulting from the assembly of F_A. When we get the exact D^{n+1}, i.e. $D^{i+1} = D^{n+1}$, the residual $G^{i+1} = 0$, and according to the Taylor expansion, we can write

$$\begin{aligned} G^{i+1} = G^i + \frac{\partial G^i}{\partial D^i}\Delta D^i = 0 \iff \\ \frac{\partial G^i}{\partial D^i}\Delta D^i = -G^i. \end{aligned} \tag{3.22}$$

By examining (3.19) and (3.20), the $N_{el} \times N_{el}$ Jacobian matrix $\frac{\partial G^i}{\partial D^i}$ can be found through the assembly of

$$M = \frac{\partial G^i}{\partial D^i} = \left(A^{LIN} + a_v^2 K^{u^i}\right)_{AB}. \tag{3.23}$$

Now all the definitions are in place to formulate the Newton iteration algorithm, which is expressed as,

Newton Iteration

$D^0 = D^n$
$i = 1$
$Err = 1$
while $Err > 1e-12$
    $\Delta D^i = -M^{-1} G^i$
    $D^{i+1} = D^i + \Delta D^i$
    update M and G^i
    $i = i + 1$
    $Err = \sqrt{(D^i - D^{i-1})^T (D^i - D^{i-1})}$
end while
$D^{n+1} = D^{i+1}$

Three solutions of Burgers equation for $Re = 10$, 100 and 1000 are compared here. Uniform meshes with 128 elements for $Re = 10$, 100 and 2048 elements for $Re = 1000$ were used. We adopted the above numerical method to solve the Burgers equation with a constant $\Delta t = 0.001$ for all cases. By terminating the time advance loop when the solution became almost unchanged, we created an ensemble of 3085, 1816 and 1605 snapshots for $Re = 10$, 100 and 1000 respectively by sampling the solution every Δt from $t = 0$.

Fig.3.7 shows initial and nearly steady solutions for the three Reynolds numbers, at $t_f = 3.084$, 1.185, 1.604 for $Re = 10$, 100 and 1000, respectively. From Fig.3.7, it is seen that the gradient of the solution becomes steeper with increasing Reynolds number.

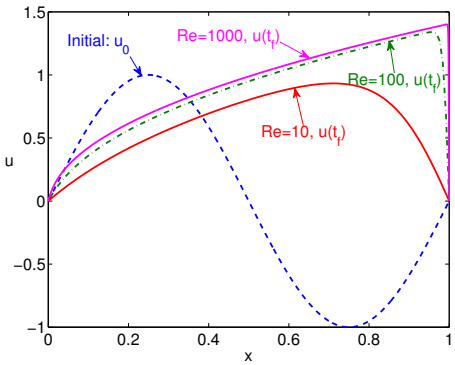


Figure 3.7: Initial and nearly steady solutions

POD ROM

We extract POD modes from all the snapshots, that is, 3085, 1816 and 1605 snapshots are used to calculate POD modes for $Re = 10$, 100 and 1000, respectively. Due to homogeneous Dirichlet boundary condition ($u(0,t) = u(1,t) = 0$) POD modes are zero at the boundaries (i.e. $\Phi(0) = \Phi(1) = 0$).

Using Galerkin projection, we construct the POD ROM for the Burgers equation. Projecting (3.14a) onto the subspace spanned by POD modes yields

$$\int_0^1 \phi_i(\hat{u}_t + \hat{u}\hat{u}_x - \frac{1}{Re}\hat{u}_{xx})dx = \int_0^1 \phi_i f dx, \qquad i = 1, 2, \cdots,$$

where $\hat{u}$ is expanded in a series of POD modes as shown in (2.4). Keeping the first nM POD modes, the POD ROM is derived[4],

$$\sum_{j=1}^{\mathrm{nM}} \dot{\alpha}_j \int_0^1 \phi_i\phi_j dx - \frac{1}{2}\int_0^1 \phi_{ix}\left(\sum_{j=1}^{\mathrm{nM}} \alpha_j\phi_j\right)^2 dx - \frac{1}{Re}\sum_{j=1}^{\mathrm{nM}} \alpha_j \int_0^1 \phi_i\phi_{jxx}dx = \int_0^1 \phi_i f dx, \tag{3.24}$$

where $i = 1, 2, \cdots,$ nM. And α^0 is calculated from

$$\sum_{j=1}^{\mathrm{nM}} \alpha_j^0 \int_0^1 \phi_i\phi_j dx = \int_0^1 \phi_i u_0 dx. \tag{3.25}$$

[4] $\int_0^1 \phi_i \hat{u}\hat{u}_x dx = \int_0^1 \phi_i \frac{1}{2}\frac{\partial(\hat{u}^2)}{\partial x} dx = \frac{1}{2}\int_0^1 \phi_{ix}(\hat{u}^2)dx$

3.3.2. SEMI-CONTINUOUS FORMULATION

For this application of the SCF, the governing equation is the Burgers equation. Thus, according to (2.8)-(2.11), the constrained minimization problem is expressed as:

$$\arg_{\Phi,\alpha}\min \mathscr{G} = \frac{1}{2}\int_0^{t_f}\int_0^1 (g-\hat{g})^2 dx dt + \frac{\beta}{2}\sum_{i,j=1}^{\mathrm{nM}}\left(\delta_{ij} - \int_0^1 \phi_i\phi_j dx\right)^2 \tag{3.26a}$$

subject to

$$\sum_{j=1}^{\mathrm{nM}}\dot{\alpha}_j\int_0^1\phi_i\phi_j dx - \frac{1}{2}\int_0^1\phi_{ix}\left(\sum_{j=1}^{\mathrm{nM}}\alpha_j\phi_j\right)^2 dx - \frac{1}{Re}\sum_{j=1}^{\mathrm{nM}}\alpha_j\int_0^1\phi_i\phi_{jxx}dx = \int_0^1\phi_i f dx \tag{3.26b}$$

$$\sum_{j=1}^{\mathrm{nM}}\alpha_j^0\int_0^1\phi_i\phi_j dx = \int_0^1\phi_i u_0 dx \tag{3.26c}$$

The primary basis functions are assumed to satisfy the boundary condition of u, i.e. $\Phi(0) = \Phi(1) = 0$. The Lagrangian functional $\mathscr{L}$ is written as:

$$\begin{aligned}\mathscr{L} =& \frac{1}{2}\int_0^{t_f}\int_0^1 (g-\hat{g})^2 dx dt + \frac{\beta}{2}\sum_{i,j=1}^{\mathrm{nM}}\left(\delta_{ij} - \int_0^1\phi_i\phi_j dx\right)^2 \\ &+ \int_0^{t_f}\sum_{i=1}^{\mathrm{nM}}\lambda_i\left[\sum_{j=1}^{\mathrm{nM}}\dot{\alpha}_j\int_0^1\phi_i\phi_j dx - \frac{1}{2}\int_0^1\phi_{ix}\left(\sum_{j=1}^{\mathrm{nM}}\alpha_j\phi_j\right)^2 dx - \frac{1}{Re}\sum_{j=1}^{\mathrm{nM}}\alpha_j\int_0^1\phi_i\phi_{jxx}dx\right. \\ &\left. - \int_0^1\phi_i f dx\right]dt + \sum_{i=1}^{\mathrm{nM}}\mu_i\left(\sum_{j=1}^{\mathrm{nM}}\alpha_j^0\int_0^1\phi_i\phi_j dx - \int_0^1\phi_i u_0 dx\right)\end{aligned} \tag{3.27}$$

The optimality conditions are then derived by taking the first variation of the Lagrangian functional (3.27) with respect to λ_i, μ_i, α_j, and ϕ_q.

State Equations Setting the first variation of $\mathscr{L}$ with respect to λ_i to zero and arguing $\delta\lambda_i$ is arbitrary at $t \in (0,\, t_f)$, and setting the first variation of $\mathscr{L}$ with respect to μ_i to zero yields the **State Equations** (3.26b)-(3.26c).

Adjoint Equations Setting the first variation of $\mathscr{L}$ with respect to α_j to zero and arguing $\delta\alpha_j$ is arbitrary in the time interval $(0,\, t_f)$, $t = 0$ and $t = t_f$, yields the **Adjoint Equations**:

$$-\sum_{i=1}^{\mathrm{nM}}\dot{\lambda}_i\int_0^1\phi_i\phi_j dx - \sum_{i=1}^{\mathrm{nM}}\lambda_i\int_0^1\phi_{ix}\phi_j\left(\sum_{j=1}^{\mathrm{nM}}\alpha_j\phi_j\right)dx + \frac{1}{Re}\sum_{i=1}^{\mathrm{nM}}\lambda_i\int_0^1\phi_{ix}\phi_{jx}dx = \int_0^1(g-\hat{g})\frac{\partial\hat{g}}{\partial\alpha_j}dx \tag{3.28a}$$

$$\lambda_i(t_f) = 0 \tag{3.28b}$$

$$\mu_i = \lambda_i(0) \tag{3.28c}$$

Gradient Setting the first variation of $\mathscr{L}$ with respect to ϕ_q to zero, and arguing $\delta\phi_q$ is zero at $x = 0$, $x = 1$ and arbitrary in the space interval $(0, 1)$, yields the **Gradient**:

$$\begin{aligned}\nabla\mathscr{L} = &\int_0^{t_f} (\hat{g} - g)\frac{\partial \hat{g}}{\partial \phi_q} dt + 2\beta \sum_{j=1}^{\mathrm{nM}} \phi_j \left(\int_0^1 \phi_q \phi_j dx - \delta_{jq} \right) \\ &+ \int_0^{t_f} \left[\lambda_q \sum_{j=1}^{\mathrm{nM}} \dot{\alpha}_j \phi_j + \dot{\alpha}_q \sum_{i=1}^{\mathrm{nM}} \lambda_i \phi_i - \alpha_q \left(\sum_{i=1}^{\mathrm{nM}} \lambda_i \phi_{ix} \right) \left(\sum_{j=1}^{\mathrm{nM}} \alpha_j \phi_j \right) \right. \\ &\left. + \lambda_q \left(\sum_{j=1}^{\mathrm{nM}} \alpha_j \phi_{jx} \right) \left(\sum_{j=1}^{\mathrm{nM}} \alpha_j \phi_j \right) - \frac{1}{Re} \lambda_q \sum_{j=1}^{\mathrm{nM}} \alpha_j \phi_{jxx} - \frac{1}{Re} \alpha_q \sum_{i=1}^{\mathrm{nM}} \lambda_i \phi_{ixx} - \lambda_q f \right] dt \\ &+ \mu_q \sum_{j=1}^{\mathrm{nM}} \alpha_j^0 \phi_j + \alpha_q^0 \sum_{i=1}^{\mathrm{nM}} \mu_i \phi_i - \mu_q u_0 \\ =&0 \end{aligned} \tag{3.29}$$

For the derivation of **Adjoint Equations** and **Gradient** in detail, you can look the derivation presented in Section 3.2.3, the only difference between them is the advection term.[5]

3.3.3. COMPARING SCF ROM WITH POD ROM

In the optimization code, the integrals in space are evaluated using the trapezoidal rule, and central finite differences are used to approximate the spatial derivatives. To solve the **State Equations**, a generalized-trapezoidal time-integration scheme (mentioned in Section 3.3.1) is used. The backward Euler method (3.13) is used to advance the **Adjoint Equations** in time.

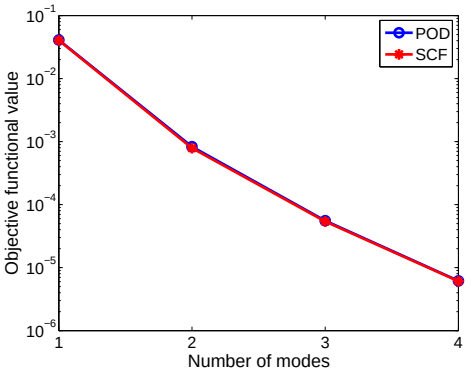
Fig.3.8 shows the results for $g = u\,(0 \leqslant x \leqslant 1)$. When $Re = 10$, the improvements over POD modes are small. With increasing Reynolds number, the solution gradients become sharper and the approximation error of the ROM is greater. The SCF ROM then provides substantial benefits relative to the POD, except when nM = 1. These benefits increase as the Reynolds number gets larger.

To clarify the lack of improvement for nM = 1, we focus on the results for $Re = 10$ (The analysis for the other two Reynolds numbers is similar.). Fig.3.9 shows the evolution of u in time and the first POD mode. The first POD mode is similar to the shape of the steady-state solution, which a significant part of the reference data resembles. This allows for little improvement via the SCF if only the first mode is used.

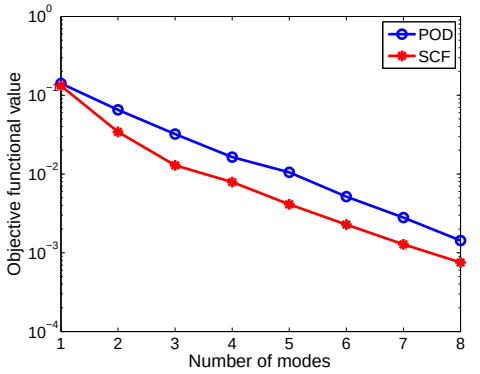
5

$$-\frac{1}{2}\frac{\partial}{\partial \alpha_j}\left[\phi_{ix}\left(\sum_{j=1}^{\mathrm{nM}} \alpha_j \phi_j\right)^2\right] = -\phi_{ix}\phi_j\left(\sum_{j=1}^{\mathrm{nM}} \alpha_j \phi_j\right)$$

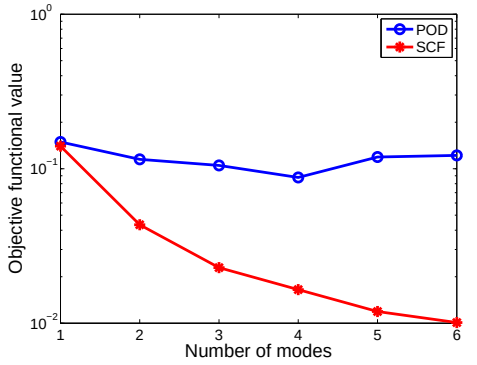
$$-\frac{1}{2}\frac{\partial}{\partial \phi_q}\left[\phi_{ix}\left(\sum_{j=1}^{\mathrm{nM}} \alpha_j \phi_j\right)^2\right] = -\alpha_q \phi_{ix}\left(\sum_{j=1}^{\mathrm{nM}} \alpha_j \phi_j\right), \quad -\frac{1}{2}\frac{\partial}{\partial \phi_{qx}}\left[\phi_{ix}\left(\sum_{j=1}^{\mathrm{nM}} \alpha_j \phi_j\right)^2\right] = -\left(\sum_{j=1}^{\mathrm{nM}} \alpha_j \phi_j\right)^2$$



(a) $Re = 10$



(b) $Re = 100$



(c) $Re = 1000$

Figure 3.8: Error when $g = u$ ($x \in [0, 1]$)

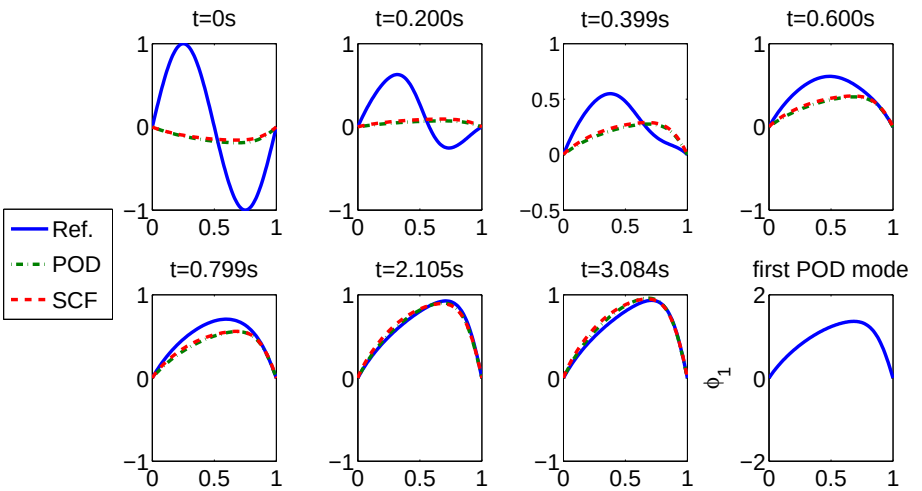
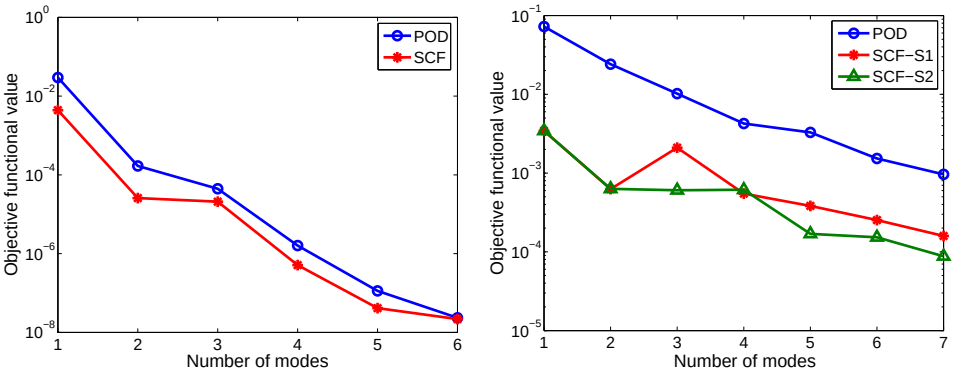


Figure 3.9: Reference data compared to the solutions from the POD ROM and SCF ROM for $g = u$ when $Re = 10$, and nM = 1

Fig.3.10 shows the results for $g = u$ $(0 \leqslant x \leqslant 0.5)$. In this case, the error reduction is substantial for $Re = 10$ and $Re = 100$. The improvement is smaller for low Re than that for high Re. Multiple solutions occur once again in this case.



(a) $Re = 10$ (b) $Re = 100$

Figure 3.10: Error when $g = u$ $(x \in [0, 0.5])$

Fig.3.11 shows results for the nonlinear goal functional $g = u^2$ $(x \in [0, 1])$ when $Re = 100$. Trends similar to the $g = u$ $(x \in [0, 1])$ case are observed.

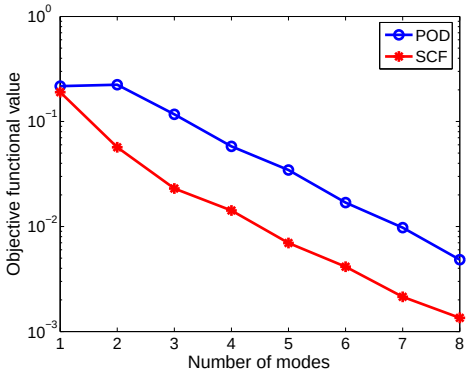


Figure 3.11: Error when $g = u^2$ and $Re = 100$

3.3.4. MULTIPLE EXTREMA IN THE OPTIMIZATION PROBLEM

In both the linear and nonlinear applications, multiple solutions were observed for some goal functional (Fig.3.5 and Fig.3.10(b)). Here, we demonstrate that for certain goal functional, multiple extrema can appear in this optimization problem. For clarity, we illustrate this problem using **F-type primary basis** defined by

$$\mathrm{nM} = 2, \ \phi_1 = \psi_1 + a_1\psi_2, \ \phi_2 = \psi_2 + b_1\psi_3,$$

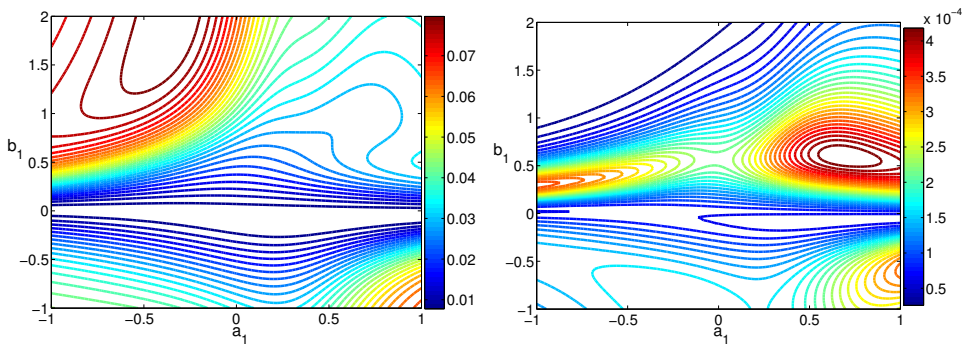
where ψ_{1-3} are the first three POD modes and define response surfaces using

$$f_{err} = \frac{1}{2}\int_0^{t_f}\int_0^1 (g-\hat{g})^2 dxdt.$$

In Fig.3.12 and Fig.3.13, contours of f_{err} versus a_1 and b_1 are shown for the advection-diffusion and Burgers problems with global and local functionals of u. For both the advection-diffusion and Burgers problem, only one minimum appears for reasonable values of a_1 and b_1 when $g = u\,(0 \leqslant x \leqslant 1)$ (Fig.3.12(a) and Fig.3.13(a)). Choosing a goal functional such as $g = u$ on $0 \leqslant x \leqslant 0.5$ or $g = u_x(0)$, however, results in ambiguity in the definition of the minima. This might be anticipated when considering the hyperbolic nature of both partial differential equations, for which information propagates from left to right. One might deal with such a situation using algorithms which treat multiple minima, but this does not guarantee that the part of the ROM solution which does not influence the functional will resemble the physical solution. In situations where the domain of dependence is as limited as the one considered in these problems, a better approach would be to define a weighted global functional rather than a purely local one. In Fig.3.14, contours of f_{err} versus a_1 and b_1 are shown for the advection-diffusion problem with $u_x W(x)$ as the goal functional, where

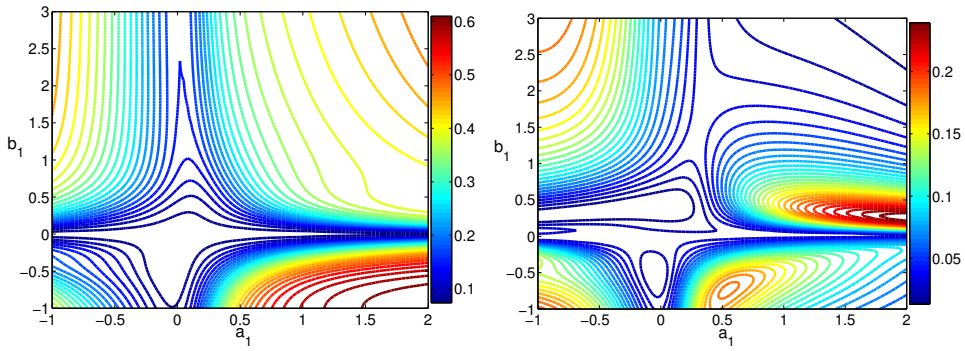
$$W(x) = \begin{cases} 0.6 + 0.4\cos(4\pi x) & x \leq 0.25 \\ 0.2 & x > 0.25 \end{cases}$$

which heavily weighs the part of the domain near $x = 0$. Comparing Fig.3.12(b) with Fig.3.14(a), the extraneous minima at $a_1 = -1$, $b_1 = 2$ is reduced significantly in magnitude, while not significantly affecting the desired minima near $a_1 = 0.3$, $b_1 = -0.1$.



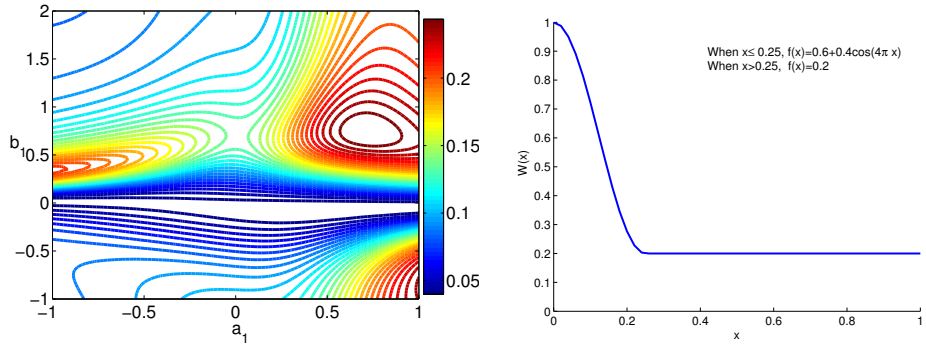
(a) $g = u(0 \leqslant x \leqslant 1)$ (b) $g = u_x(x = 0)$

Figure 3.12: Contours of f_{err} for the advection-diffusion problem



(a) $g = u(0 \leqslant x \leqslant 1)$ (b) $g = u(0 \leqslant x \leqslant 0.5)$

Figure 3.13: Contours of f_{err} for the Burgers problem



(a) $g = u_x W(x)$ (b) $W(x)$

Figure 3.14: Contours of f_{err} for advection-diffusion problem (a), and the function $W(x)$(b)

3.4. COST OF DETERMINING GOAL-ORIENTED MODES

In this section, the computational cost of SCF and its scaling are described. For the small problems considered here, SVD algorithms are efficient, so the costs of determining POD modes are generally two to three orders of magnitude less than required for determining goal-oriented modes. This ratio depends on several factors, including such as initial trust region radius, initial guesses of Φ, the criterion coefficients for updating the modes and radius, and the chosen goal functional. In general, the CPU required increases with the number of modes, as shown in Fig.3.15. In general, the increase in cost is justified if the ROM is to be used repeatedly after identification, or the modes are being found with the objective of identifying the processes of most importance for a specific goal functional. It is worth noting, however, for problems which require high spatial resolution the costs of SVD rise rapidly, as shown in Fig.3.16(a). The costs of a single sweep in the

SCF procedure, incorporating the adjoint, ROM (state) and gradient solves, scales close to linearly (Fig.3.16(b)). Furthermore, each of these steps can be parallelised using standard techniques. This provides the possibility of computing very large problems with SCF, provided reasonable initial conditions can be defined.

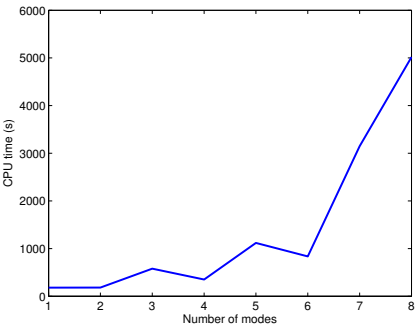
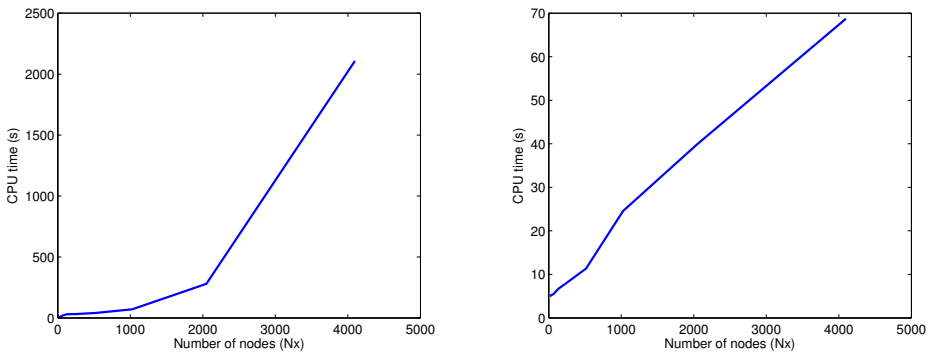


Figure 3.15: CPU time of SCF versus number of modes when $g = u(0 \leqslant x \leqslant 1)$



(a) SVD

(b) SCF sweep with nM = 6

Figure 3.16: CPU time of SVD and a single SCF sweep versus the number of points used to represent the modes (Nx)

3.5. AN APPROXIMATE PARTIAL DIFFERENTIAL EQUATION EXAMPLE

Often it is desired to approximate experimental data with a relatively simple model partial differential equation, introducing substantial model errors. The SCF can be effective in such applications, for obtaining modes which improve the performance of the ROM constructed by the simplified partial differential equation. As an example, we will consider solutions to the one-dimensional Burgers equation as reference data, while con-

structing a ROM using a simplified partial differential equation, the linear advection-diffusion equation. Homogeneous boundary conditions and a constant source $f = 1$ are used in both the generation of the reference data and the ROM. The viscosity factor k and the advection speed a in the advection-diffusion equation are defined as

$$k = \frac{1}{Re}, \quad a = \frac{1}{N_{el}+1}\frac{1}{N_s}\sum_{j=1}^{N_{el}+1}\sum_{i=1}^{N_s} u_{ij}.$$

where $Re = 10$ and u is the corresponding solution of the Burgers equation.

Fig.3.17 shows results when $g = u$ over whole space domain. The optimal modes are substantially more accurate. In this case, the effect of the model constraint is large, allowing the optimal modes to partially compensate for the inaccuracy of the partial differential equation approximation.

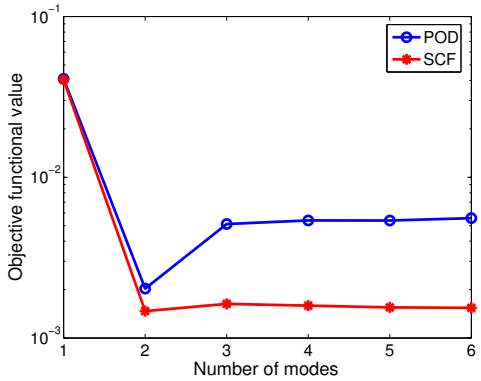
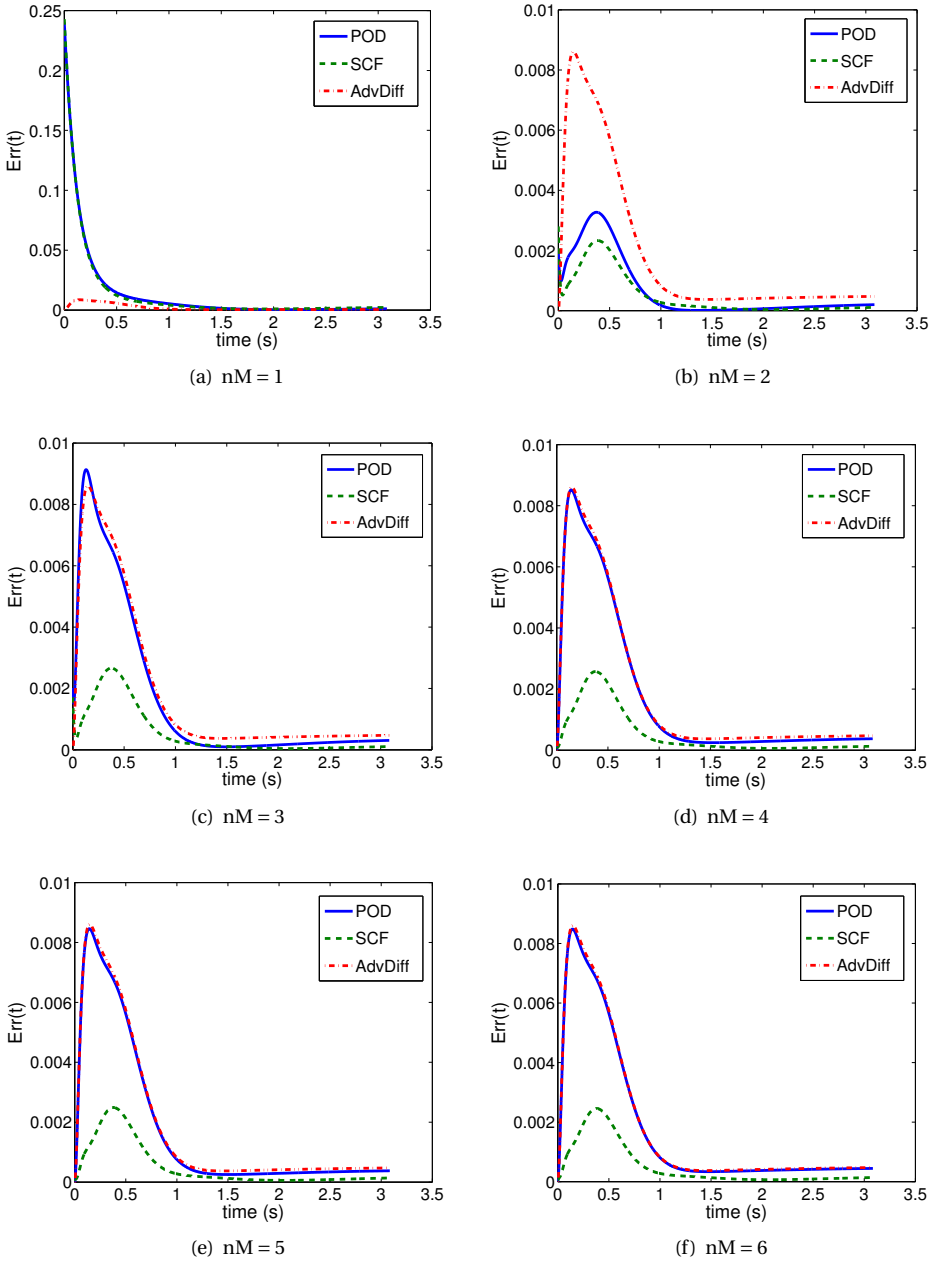


Figure 3.17: Error vs number of modes for the approximate partial differential equation when $g = u(0 \leqslant x \leqslant 1)$

To clarify the behavior in Fig.3.17, we define a function

$$\mathrm{Err}(t) = \frac{1}{2}\int_0^1 (u_{burgers} - u_{other})^2 dx, \tag{3.30}$$

where $u_{burgers}$ is the Burgers' solution. In Fig.3.18, the label 'POD' refers to the case where u_{other} is the solution of ROM constructed by advection-diffusion equation using POD modes as projection basis functions; 'SCF' is the case where u_{other} is the solution of ROM constructed by advection-diffusion equation using optimal modes as projection basis functions; 'AdvDiff' is the case where u_{other} is the solution of advection-diffusion equation and obtained from a fully refined finite-difference method. When only one mode is used, the SCF and POD ROMs are equally unable to reproduce the Burgers' solution because of the convective nature of the problem. Adding a second mode allows both the SCF and POD ROMs to improve. Adding further modes, however, results in the POD ROM converging to the advection-diffusion solution, while the SCF solution has a lower error, because the output of the approximate partial differential equation is taken into account during optimization.



(a) nM = 1 (b) nM = 2

(c) nM = 3 (d) nM = 4

(e) nM = 5 (f) nM = 6

Figure 3.18: Error in approximating the Burgers solution using a refined advection-diffusion solution (AdvDiff), POD ROM and SCF ROM approximations

3.6. CONCLUSIONS

Results from both linear and nonlinear model problems confirm that the benefits of using a SCF goal-oriented approach for mode identification can be substantial, particularly when only part of the domain is of interest. In all cases, substantial improvements in absolute accuracy over POD modes were observed, except for global goal functional related to the error norm already minimized by POD interpolation. For such cases the benefits were only significant when the governing equation used to derive the ROM roughly approximates the dynamics of the reference data.

Finally, it was shown that the choice of goal functional can influence the number of extrema appearing in the optimization problem. The resulting ambiguities can be avoided by choosing weighted global rather than purely local goal functional.

4

APPLICATION II: 2D STOKES PROBLEMS

In this chapter, we apply the SCF to the two-dimensional unsteady incompressible Stokes equations. Optimal primary basis functions are found for both the velocity and pressure. To verify our optimization code, we investigate a case where the exact primary basis function is one of the secondary basis functions. Then, a more general case is investigated where the secondary basis does not include the exact primary basis function.

4.1. GOVERNING EQUATION

For this application, we use the unsteady incompressible Stokes equations in the following form:

$$\frac{\partial \boldsymbol{V}}{\partial t} - \nabla \cdot (2\nu \nabla^s \boldsymbol{V}) + \nabla p = \boldsymbol{f} \tag{4.1a}$$

$$\nabla \cdot \boldsymbol{V} = 0 \tag{4.1b}$$

where $\boldsymbol{V} = [u,\ v,\ w]^T$ represents the velocity, u, v and w are the $x-$, $y-$ and $z-$direction velocities, p represents the pressure, $\boldsymbol{f}$ represents the body force, and ν is the kinematic coefficient of viscosity which is assumed to be constant. Furthermore, the symmetric velocity gradient is defined as

$$\nabla^s \boldsymbol{V} = \frac{1}{2}(\nabla \boldsymbol{V} + \nabla \boldsymbol{V}^T). \tag{4.2}$$

To arrive at a proper problem statement, a homogeneous Dirichlet boundary condition for the velocity is used, i.e.

$$\boldsymbol{V} = 0 \qquad \text{on} \quad \Gamma, \tag{4.3}$$

where $\Gamma = \partial\Omega$ is the boundary of the space domain Ω of interest, which is an open, bounded subset of $\mathbb{R}^3$.

4.2. REDUCED-ORDER MODELS

In an incompressible flow, the important dynamical flow variable is the velocity, while the pressure acts only to enforce the incompressibility constraint. Thus, we might ask if we need to take into account the pressure when constructing ROMs for incompressible flows? When the primary basis functions used for the velocity are divergence free, the pressure can be eliminated completely from the construction of the ROM for incompressible flows with periodic boundary conditions, which is done by integrating the pressure gradient term in the corresponding Galerkin formulation by parts. For incompressible flows, the POD modes extracted from only the velocity field can be chosen as the divergence free primary basis functions since they are linear combinations of velocity snapshots which are themselves divergence free; besides, there will be no pressure term in the POD ROM when imposing homogeneous boundary conditions for the velocity or periodic boundary conditions for both the velocity and pressure [23, 92–95]. That is, to construct the ROM for incompressible flows, the primary basis functions for the pressure are not necessary in these cases.

However, in many cases, the pressure integral does not vanish on the boundaries, and the primary basis functions used for the velocity are not always divergence free. The pressure then must be modeled in these cases. Noack *et al.* [96] made use of the pressure-Poisson equation to model the pressure gradient term when constructing a POD ROM for incompressible flows. While, in this chapter, we use a different method to model the pressure, where the primary basis functions used for not only the velocity but also the pressure need to be determined previously. Thus, a ROM for the Stokes equations (4.1) is generated by directly projecting the Stokes equations onto the subspace spanned by the given primary basis functions.

Throughout this chapter, we use the following notation:

$\boldsymbol{x}$: the three coordinates in the space, and $\boldsymbol{x} = (x, y, z)$;

j: primary basis functions index and $j = 1, 2, \cdots,$ nM;

$\phi_j^u(\boldsymbol{x})$: the j^{th} primary basis function for the x-direction velocity u;

$\phi_j^v(\boldsymbol{x})$: the j^{th} primary basis function for the y-direction velocity v;

$\phi_j^w(\boldsymbol{x})$: the j^{th} primary basis function for the z-direction velocity w;

$\phi_j^p(\boldsymbol{x})$: the j^{th} primary basis function for the pressure p;

$\boldsymbol{\phi}_j(\boldsymbol{x})$: the j^{th} primary basis function for the velocity $\boldsymbol{V}$, and

$$\boldsymbol{\phi}_j(\boldsymbol{x}) = \begin{bmatrix} \phi_j^u(\boldsymbol{x}) \\ \phi_j^v(\boldsymbol{x}) \\ \phi_j^w(\boldsymbol{x}) \end{bmatrix};$$

$\boldsymbol{\Phi}_j(\boldsymbol{x})$: the j^{th} primary basis function for the velocity $\boldsymbol{V}$ and pressure p, and

$$\boldsymbol{\Phi}_j(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{\phi}_j(\boldsymbol{x}) \\ \phi_j^p(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} \phi_j^u(\boldsymbol{x}) \\ \phi_j^v(\boldsymbol{x}) \\ \phi_j^w(\boldsymbol{x}) \\ \phi_j^p(\boldsymbol{x}) \end{bmatrix}.$$

The primary basis functions for the velocity and pressure are chosen to satisfy the following requirements:

1. The primary basis functions should satisfy the boundary conditions of the problem. That is, in our particular case, with $\boldsymbol{V}_\Gamma = 0$, the primary basis function for the velocity, $\boldsymbol{\phi}_j$, on the boundary is zero as well,

$$\boldsymbol{\phi}_j = 0 \qquad \text{on} \quad \Gamma. \tag{4.4}$$

2. For the solution to be unique, the primary basis functions must be linearly independent.

As the pressure is taken into account, there are four flow variables to be determined, three-directional velocities and pressure (u, v, w, p). For the model expansion of them, there are three options. First, they could all be expanded using their own bases, implying that the velocities and pressure are modeled separately. Alternatively, the velocities could be expanded using a single vector basis ($\boldsymbol{\phi}$) and the pressure is expanded using its own basis (ϕ^p). Finally, both the velocities and pressure could be expanded using a single vector basis ($\boldsymbol{\Phi}$), implying that the velocities and pressure are taken as a single vector-valued variable (e.g.,$\boldsymbol{U} = [u, v, w, p]^T$). We will consider the first and last of these options, and refer to them as the scalar- and vector-valued primary basis function options, respectively.

4.2.1. SCALAR-VALUED PRIMARY BASIS FUNCTIONS

We first consider expanding the $x-$, $y-$ and $z-$components of velocity as well as the pressure in their own primary bases

$$\hat{u} = \sum_{j=1}^{\mathrm{nM}} \alpha_j^u(t)\phi_j^u(\boldsymbol{x}), \tag{4.5a}$$

$$\hat{v} = \sum_{j=1}^{\mathrm{nM}} \alpha_j^v(t)\phi_j^v(\boldsymbol{x}), \tag{4.5b}$$

$$\hat{w} = \sum_{j=1}^{\mathrm{nM}} \alpha_j^w(t)\phi_j^w(\boldsymbol{x}), \tag{4.5c}$$

$$\hat{p} = \sum_{j=1}^{\mathrm{nM}} \alpha_j^p(t)\phi_j^p(\boldsymbol{x}), \tag{4.5d}$$

where, e.g. $\alpha_j^u(t)$ is the amplitude for the primary basis function ϕ_j^u. For real scalar-valued functions, we define

$$(\phi_i^u, \phi_j^u)_\Omega = \int_\Omega \phi_i^u \phi_j^u d\Omega.$$

The amplitudes are calculated using the Galerkin projection of the Stokes equations (4.1), in which the momentum equations (4.1a) are rewritten as

$$u_t - \nu(2u_{xx} + u_{yy} + u_{zz} + v_{xy} + w_{xz}) + p_x = f^x, \tag{4.6a}$$

$$v_t - \nu(2v_{xx} + v_{yy} + v_{zz} + u_{yx} + w_{yz}) + p_y = f^y, \tag{4.6b}$$

$$w_t - \nu(2w_{xx} + w_{yy} + w_{zz} + u_{zx} + v_{zy}) + p_z = f^z, \tag{4.6c}$$

where f^x, f^y and f^z are the body force in $x-$, $y-$ and $z-$direction, respectively. Inserting (4.5) into (4.1b) and (4.6) leads to

$$\sum_{j=1}^{\mathrm{nM}} \dot{\alpha}_j^u \phi_j^u - \sum_{j=1}^{\mathrm{nM}} \left[\nu\alpha_j^u \left(2\phi_{jxx}^u + \phi_{jyy}^u + \phi_{jzz}^u\right) + \nu\alpha_j^v \phi_{jxy}^v + \nu\alpha_j^w \phi_{jxz}^w - \alpha_j^p \phi_{jx}^p \right] = f^x, \tag{4.7a}$$

$$\sum_{j=1}^{\mathrm{nM}} \dot{\alpha}_j^v \phi_j^v - \sum_{j=1}^{\mathrm{nM}} \left[\nu\alpha_j^v \left(2\phi_{jyy}^v + \phi_{jxx}^v + \phi_{jzz}^v\right) + \nu\alpha_j^u \phi_{jyx}^u + \nu\alpha_j^w \phi_{jyz}^w - \alpha_j^p \phi_{jy}^p \right] = f^y, \tag{4.7b}$$

$$\sum_{j=1}^{\mathrm{nM}} \dot{\alpha}_j^w \phi_j^w - \sum_{j=1}^{\mathrm{nM}} \left[\nu\alpha_j^w \left(2\phi_{jzz}^w + \phi_{jyy}^w + \phi_{jxx}^w\right) + \nu\alpha_j^u \phi_{jzx}^u + \nu\alpha_j^v \phi_{jzy}^v - \alpha_j^p \phi_{jz}^p \right] = f^z, \tag{4.7c}$$

$$\sum_{j=1}^{\mathrm{nM}} \left(\alpha_j^u \phi_{jx}^u + \alpha_j^v \phi_{jy}^v + \alpha_j^w \phi_{jz}^w \right) = 0. \tag{4.7d}$$

Using Galerkin projection of (4.7a), (4.7b), (4.7c) and (4.7d) onto the subspace spanned by the primary basis functions ϕ_j^u, ϕ_j^v, ϕ_j^w and ϕ_j^p yields a system of ordinary differential

equations

$$\sum_{j=1}^{\mathrm{nM}} m_{ij}^u \dot{\alpha}_j^u = \sum_{j=1}^{\mathrm{nM}} \left(c_{ij}^u \alpha_j^u + d_{ij}^u \alpha_j^v + e_{ij}^u \alpha_j^w + h_{ij}^u \alpha_j^p \right) + b_i^u, \tag{4.8a}$$

$$\sum_{j=1}^{\mathrm{nM}} m_{ij}^v \dot{\alpha}_j^v = \sum_{j=1}^{\mathrm{nM}} \left(c_{ij}^v \alpha_j^u + d_{ij}^v \alpha_j^v + e_{ij}^v \alpha_j^w + h_{ij}^v \alpha_j^p \right) + b_i^v, \tag{4.8b}$$

$$\sum_{j=1}^{\mathrm{nM}} m_{ij}^w \dot{\alpha}_j^w = \sum_{j=1}^{\mathrm{nM}} \left(c_{ij}^w \alpha_j^u + d_{ij}^w \alpha_j^v + e_{ij}^w \alpha_j^w + h_{ij}^w \alpha_j^p \right) + b_i^w, \tag{4.8c}$$

$$0 = \sum_{j=1}^{\mathrm{nM}} \left(c_{ij}^p \alpha_j^u + d_{ij}^p \alpha_j^v + e_{ij}^p \alpha_j^w \right), \tag{4.8d}$$

where the coefficients are given by:

$$m_{ij}^u = (\phi_i^u, \phi_j^u)_\Omega \qquad m_{ij}^v = (\phi_i^v, \phi_j^v)_\Omega \qquad m_{ij}^w = (\phi_i^w, \phi_j^w)_\Omega$$

$$\begin{aligned}
c_{ij}^u &= -\nu \left[2(\phi_{ix}^u, \phi_{jx}^u)_\Omega + (\phi_{iy}^u, \phi_{jy}^u)_\Omega + (\phi_{iz}^u, \phi_{jz}^u)_\Omega \right] \\
c_{ij}^v &= -\nu (\phi_{ix}^v, \phi_{jy}^u)_\Omega \\
c_{ij}^w &= -\nu (\phi_{ix}^w, \phi_{jz}^u)_\Omega \\
c_{ij}^p &= (\phi_i^p, \phi_{jx}^u)_\Omega
\end{aligned}$$

$$\begin{aligned}
d_{ij}^u &= -\nu (\phi_{iy}^u, \phi_{jx}^v)_\Omega \\
d_{ij}^v &= -\nu \left[2(\phi_{ix}^v, \phi_{jx}^v)_\Omega + (\phi_{iy}^v, \phi_{jy}^v)_\Omega + (\phi_{iz}^v, \phi_{jz}^v)_\Omega \right] \\
d_{ij}^w &= -\nu (\phi_{iy}^w, \phi_{jz}^v)_\Omega \\
d_{ij}^p &= (\phi_i^p, \phi_{jy}^v)_\Omega
\end{aligned}$$

$$\begin{aligned}
e_{ij}^u &= -\nu (\phi_{iz}^u, \phi_{jx}^w)_\Omega \\
e_{ij}^v &= -\nu (\phi_{iz}^v, \phi_{jy}^w)_\Omega \\
e_{ij}^w &= -\nu \left[2(\phi_{ix}^w, \phi_{jx}^w)_\Omega + (\phi_{iy}^w, \phi_{jy}^w)_\Omega + (\phi_{iz}^w, \phi_{jz}^w)_\Omega \right] \\
e_{ij}^p &= (\phi_i^p, \phi_{jz}^w)_\Omega
\end{aligned}$$

$$h_{ij}^u = (\phi_{ix}^u, \phi_j^p)_\Omega \qquad h_{ij}^v = (\phi_{iy}^v, \phi_j^p)_\Omega \qquad h_{ij}^w = (\phi_{iz}^w, \phi_j^p)_\Omega$$

$$b_i^u = (\phi_i^u, f^x)_\Omega \qquad b_i^v = (\phi_i^v, f^y)_\Omega \qquad b_i^w = (\phi_i^w, f^z)_\Omega$$

The first terms on the right hand of (4.8a)-(4.8c) are obtained by integrating by parts while employing $\boldsymbol{\phi}_j|_\Gamma = 0$. The equations (4.8) represent a reduced-order model for the Stokes equations (4.1). The original Stokes equations which consist of four coupled partial differential equations valid at an infinite number of points in the domain have been transformed into a system of $4 \times \mathrm{nM}$ coupled ordinary differential equations where nM is relatively small.

Note that for the scalar-valued primary basis we determine the primary basis functions ϕ^u, ϕ^v, ϕ^w and ϕ^p separately and independently. Thus, ϕ^u, ϕ^v, ϕ^w and ϕ^p can be the same, such as in the case where the Finite Element Method is used to solve fluid mechanical problems; they also can be different from each other. Therefore, when we use the ROM (4.8) as constraints in the SCF, there is much more freedom to choose the primary basis for **P-type primary basis** (the secondary basis for **F-type primary basis**).

4.2.2. VECTOR-VALUED PRIMARY BASIS FUNCTIONS

The velocity and pressure can also be expanded using a vector-valued primary basis functions

$$\hat{\boldsymbol{U}} = \begin{bmatrix} \hat{u} \\ \hat{v} \\ \hat{w} \\ \hat{p} \end{bmatrix} = \sum_{j=1}^{\mathrm{nM}} \alpha_j(t)\boldsymbol{\Phi}_j(\boldsymbol{x}) = \sum_{j=1}^{\mathrm{nM}} \alpha_j(t) \begin{bmatrix} \phi_j^u(\boldsymbol{x}) \\ \phi_j^v(\boldsymbol{x}) \\ \phi_j^w(\boldsymbol{x}) \\ \phi_j^p(\boldsymbol{x}) \end{bmatrix}. \tag{4.9}$$

For real vector-valued functions, we define

$$(\boldsymbol{U}, \boldsymbol{U})_\Omega = \int_\Omega \left(u^2 + v^2 + w^2 + p^2\right) d\Omega.$$

Then, inserting (4.9) into the Stokes equations (4.1) and projecting (4.1) onto the primary basis $\boldsymbol{\Phi}$ yields

$$\left(\boldsymbol{\phi}_i, \sum_{j=1}^{\mathrm{nM}} \dot{\alpha}_j \boldsymbol{\phi}_i - \sum_{j=1}^{\mathrm{nM}} \alpha_j \nabla\cdot(2\nu\nabla^s\boldsymbol{\phi}_j) + \sum_{j=1}^{\mathrm{nM}} \alpha_j \phi_j^p - \boldsymbol{f}\right)_\Omega + \left(\phi_i^p, \sum_{j=1}^{\mathrm{nM}} \alpha_j \nabla\cdot\boldsymbol{\phi}_j\right)_\Omega = 0. \tag{4.10}$$

Considering that $\boldsymbol{\phi}_j|_\Gamma = 0$, and integrating the second and third terms in the left hand of (4.10) by parts, the reduced-order model is then

$$\sum_{j=1}^{\mathrm{nM}} L_{ij}\dot{\alpha}_j = \sum_{j=1}^{\mathrm{nM}} (B_{ij}^v + B_{ij}^p + B_{ij}^c)\alpha_j + b_i^m, \tag{4.11}$$

where the coefficients L_{ij}, B_{ij}^v, B_{ij}^p, B_{ij}^c and b_i^m are given by

$$\begin{aligned} L_{ij} &= \left(\boldsymbol{\phi}_i, \boldsymbol{\phi}_j\right)_\Omega, \\ B_{ij}^v &= -\left(\nabla\boldsymbol{\phi}_i, 2\nu\nabla^s\boldsymbol{\phi}_j\right)_\Omega, \\ B_{ij}^p &= \left(\nabla\cdot\boldsymbol{\phi}_i, \phi_j^p\right)_\Omega, \\ B_{ij}^c &= -\left(\phi_i^p, \nabla\cdot\boldsymbol{\phi}_j\right)_\Omega, \\ b_i^m &= \left(\boldsymbol{\phi}_i, \boldsymbol{f}\right)_\Omega. \end{aligned}$$

The equations (4.11) represents the reduced-order model for the Stokes equations (4.1) when vector-valued primary basis functions are used. The ROM (4.11) is a system of nM ordinary differential equations, while (4.8) is a system of $4 \times$ nM ordinary differential equations.

Note that when we use this ROM the determination of the primary basis functions for velocity and pressure must be done with care. We choose to follow ideas from applications of the POD to compressible flows [97–101], where POD modes are extracted from velocity field combined with other flow variables fields (e.g. pressure, entropy) and then a reduced-order model is constructed by projecting the governing equations onto a single vector-valued set of these POD modes.

4.3. CONSTRUCTION OF PRIMARY BASIS FUNCTIONS

For applications of the SCF into two/three-dimensional problems, the **P-type primary basis** defined in Chapter 2, which is used for applications of the SCF into 1D problems (Chapter 3), can lead to a very large number of optimized variables associated with a given primary basis function. Since we are considering smooth solutions, one can expect that a F-type basis will provide good approximations for primary basis functions with a relatively small number of optimized variables (Coe). Therefore, for the application of the SCF into 2D Stokes equations, we adopt a **F-type primary basis**. In other words, the primary basis, $\mathbf{\Phi}$, is a linear combination of linearly independent pre-specified basis functions ψ_l. As the symbol $\mathbf{\Phi}$ is used for the primary basis, we denote the secondary basis functions as:

$$\mathbf{\Psi}_l = \begin{bmatrix} \psi_l^u \\ \psi_l^v \\ \psi_l^w \\ \psi_l^p \end{bmatrix} \qquad l = 1, 2, \cdots, \text{nB}$$

where, e.g. ψ_l^u represents the l^{th} secondary basis function for velocity u. Now a question is introduced – How can we construct the primary basis functions $\mathbf{\Phi}_j$ with these pre-specified secondary basis functions $\mathbf{\Psi}_l$? Following ideas from the two definitions of primary basis functions presented above, we consider two options as described below.

1. Scalar-valued secondary basis functions In this option, we consider each secondary basis function separately and expand each primary basis function in its own set of secondary basis functions

$$\phi_j^u = \sum_{l=1}^{\text{nB}} \text{Coe}_{jl}^u \psi_l^u, \tag{4.12a}$$

$$\phi_j^v = \sum_{l=1}^{\text{nB}} \text{Coe}_{jl}^v \psi_l^v, \tag{4.12b}$$

$$\phi_j^w = \sum_{l=1}^{\text{nB}} \text{Coe}_{jl}^w \psi_l^w, \tag{4.12c}$$

$$\phi_j^p = \sum_{l=1}^{\text{nB}} \text{Coe}_{jl}^p \psi_l^p, \tag{4.12d}$$

where the coefficients Coe_{jl}^u, Coe_{jl}^v, Coe_{jl}^w and Coe_{jl}^p are the optimized variables. The number of optimized variables is thus $4 \times \text{nM} \times \text{nB}$ for the construction of nM optimal primary basis functions.

2. Vector-valued secondary basis functions In this option, the secondary basis functions for each variable (e.g. u, v, w and p) are considered as a big single vector, and then the primary basis functions are constructed as:

$$\left.\begin{matrix}\phi_j^u\\ \phi_j^v\\ \phi_j^w\\ \phi_j^p\end{matrix}\right\} = \sum_{l=1}^{\mathrm{nB}} \mathrm{Coe}_{jl}\boldsymbol{\Psi}_l = \sum_{l=1}^{\mathrm{nB}} \mathrm{Coe}_{jl}\begin{bmatrix}\psi_l^u\\ \psi_l^v\\ \psi_l^w\\ \psi_l^p\end{bmatrix} \tag{4.13}$$

where the coefficients Coe_{jl} are the optimized variables. Using (4.13) to construct primary basis functions results in nM × nB optimized variables.

4

4.4. SEMI-CONTINUOUS FORMULATIONS

In this section, we will present mathematical formulations of the optimization problem for our particular case – the Stokes equations and their corresponding optimality system.

4.4.1. CONSTRAINED OPTIMIZATION PROBLEM

In section 4.2, we have introduced two forms of the reduced-order model for the Stokes equations, (4.8) and (4.11); and in section 4.3, two alternatives were used to construct primary basis functions from secondary basis functions, (4.12) and (4.13). Thus, in theory, there will be four alternatives for the constrained optimization problem:

(a) Using ROM (4.8) as constraints and constructing the primary basis functions using (4.12), i.e. employing scalar-valued primary basis functions which are constructed of scalar-valued secondary basis functions.

(b) Using ROM (4.8) as constraints and constructing the primary basis functions using (4.13), i.e. employing scalar-valued primary basis functions which are constructed of vector-valued secondary basis functions.

(c) Using ROM (4.11) as constraints and constructing the primary basis functions using (4.12), i.e. employing vector-valued primary basis functions which are constructed of scalar-valued secondary basis functions.

(d) Using ROM (4.11) as constraints and constructing the primary basis functions using (4.13), i.e. employing vector-valued primary basis functions which are constructed of vector-valued secondary basis functions.

Note that with alternative (a), there is no limitation on the secondary basis functions, that is, even though the secondary basis functions do not provide compatible values of velocity and pressure, we might still obtain good primary basis functions for the goal functional; with alternative (b), it is not likely for a ROM with separate primary basis functions expanded in terms of a combined secondary basis to provide a good approximation of the original system (4.1); with alternatives (c) and (d), because we construct the ROM in terms of vector-valued primary basis functions, we might need to carefully

choose the secondary basis functions to ensure that an optimal primary basis is finally obtained through the SCF.

Thus there are three practical alternatives, (a), (c), and (d), for describing the constrained optimization problem. The dimensions of the ROM (the number of variables to be solved) and the number of optimized variables, are shown for each alternative in table 4.1. The dimension of the **Adjoint Equations** is the same as that of the ROM, and the size of the **Gradient** equals to the number of optimized variables. From table 4.1, we can see that it is reasonable to expect that a smaller size of constrained optimization problem will be possible with (d) than with (a) and (c). For applications of the SCF into the Stokes equations, we will thus investigate the constrained optimization problem described by (d).

Alternative	Dimension of ROM	Number of optimized variables
(a)	$4 \times \mathrm{nM}$	$4 \times \mathrm{nM} \times \mathrm{nB}$
(c)	nM	$4 \times \mathrm{nM} \times \mathrm{nB}$
(d)	nM	$\mathrm{nM} \times \mathrm{nB}$

Table 4.1: Dimensions for the three alternatives of the constrained optimization problem

To make the optimal primary basis an orthonormal basis, we need to include a regularization term. Corresponding to the two alternatives for primary basis functions, scalar-valued and vector-valued, we can define the regularization term in two forms.

(I) A regularization term suitable for the scalar-valued primary basis functions is:

$$\frac{\beta}{2}\sum_{i,j=1}^{\mathrm{nM}}\left\{\left[\delta_{ij}-\left(\phi_i^u,\phi_j^u\right)_\Omega\right]^2+\left[\delta_{ij}-\left(\phi_i^v,\phi_j^v\right)_\Omega\right]^2+\left[\delta_{ij}-\left(\phi_i^w,\phi_j^w\right)_\Omega\right]^2 + \left[\delta_{ij}-\left(\phi_i^p,\phi_j^p\right)_\Omega\right]^2\right\} \tag{4.14}$$

where the first three terms make the matrices (m^u, m^v, m^p) appearing in (4.8) identity matrices, and the last term is added to make the primary basis functions for the pressure (ϕ^p) to be orthonormal to each other.

(II) A regularization term suitable for the vector-valued primary basis functions is:

$$\frac{\beta}{2}\sum_{i,j=1}^{\mathrm{nM}}\left[\delta_{ij}-\left(\mathbf{\Phi}_i,\mathbf{\Phi}_j\right)_\Omega\right]^2 \tag{4.15}$$

To give an explicit expression for the constrained optimization problem, we still need to address the definition of the goal functional. There are potentially many goal functionals of interest, as an example, we consider the following class of goal functional defined by:

$$\boldsymbol{g}=\Big[f(u),\,f(v),\,f(w),\,f(p)\Big]^T \tag{4.16}$$

where, e.g. $f(u)$ represents a continuous or discrete function of only u.

In summary, in case of an unsteady incompressible Stokes problem, the constrained optimization problem which is investigated here can be expressed as:

$$\arg_{\mathrm{Coe},\alpha}\min \mathscr{G} = \frac{1}{2}\int_0^{t_f} \left(\boldsymbol{g}-\hat{\boldsymbol{g}},\, \boldsymbol{g}-\hat{\boldsymbol{g}}\right)_\Omega dt + \frac{\beta}{2}\sum_{i,j=1}^{\mathrm{nM}} \left[\delta_{ij} - \left(\boldsymbol{\Phi}_i,\, \boldsymbol{\Phi}_j\right)_\Omega\right]^2 \tag{4.17a}$$

subject to

$$\sum_{j=1}^{\mathrm{nM}} L_{ij}\dot{\alpha}_j = \sum_{j=1}^{\mathrm{nM}} \left(B^{\nu}_{ij} + B^{p}_{ij} + B^{c}_{ij}\right)\alpha_j + b^{m}_{i} \tag{4.17b}$$

$$\sum_{j=1}^{\mathrm{nM}} \alpha^0_j \left(\boldsymbol{\Phi}_i,\, \boldsymbol{\Phi}_j\right)_\Omega = \left(\boldsymbol{\Phi}_i,\, \boldsymbol{U}_0\right)_\Omega \tag{4.17c}$$

$$\hat{\boldsymbol{g}} = \left[f\Big(\sum_{j=1}^{\mathrm{nM}} \alpha_j \phi^u_j\Big),\, f\Big(\sum_{j=1}^{\mathrm{nM}} \alpha_j \phi^v_j\Big),\, f\Big(\sum_{j=1}^{\mathrm{nM}} \alpha_j \phi^w_j\Big),\, f\Big(\sum_{j=1}^{\mathrm{nM}} \alpha_j \phi^p_j\Big)\right]^T \tag{4.17d}$$

and

$$\boldsymbol{\Phi}_j = \sum_{l=1}^{\mathrm{nB}} \mathrm{Coe}_{jl}\boldsymbol{\Psi}_l \tag{4.17e}$$

where the definition of L_{ij}, B^{ν}_{ij}, B^{p}_{ij}, B^{c}_{ij}, and b^{m}_{i} are presented in Section 4.2.2, and $\boldsymbol{U}_0$ represents the initial conditions for the velocity and pressure, i.e. $\boldsymbol{U}_0 = \left[u_0,\, v_0,\, w_0,\, p_0\right]^T$.

4.4.2. OPTIMALITY SYSTEM

To obtain the optimality conditions for the problem (4.17), we will define the Lagrangian functional as:

$$\begin{aligned}\mathscr{L} =& \frac{1}{2}\int_0^{t_f} \left(\boldsymbol{g}-\hat{\boldsymbol{g}},\, \boldsymbol{g}-\hat{\boldsymbol{g}}\right)_\Omega dt + \frac{\beta}{2}\sum_{i,j=1}^{\mathrm{nM}} \left[\delta_{ij} - \left(\boldsymbol{\Phi}_i,\, \boldsymbol{\Phi}_j\right)_\Omega\right]^2 \\ &+ \int_0^{t_f} \sum_{i=1}^{\mathrm{nM}} \lambda_i \left[\sum_{j=1}^{\mathrm{nM}} L_{ij}\dot{\alpha}_j - \sum_{j=1}^{\mathrm{nM}} \left(B^{\nu}_{ij} + B^{p}_{ij} + B^{c}_{ij}\right)\alpha_j - b^{m}_{i}\right] dt \\ &+ \sum_{i=1}^{\mathrm{nM}} \mu_i \left[\sum_{j=1}^{\mathrm{nM}} \alpha^0_j \left(\boldsymbol{\Phi}_i,\, \boldsymbol{\Phi}_j\right)_\Omega - \left(\boldsymbol{\Phi}_i,\, \boldsymbol{U}_0\right)_\Omega\right]\end{aligned} \tag{4.18}$$

where $\lambda_i = \lambda_i(t)$ is a function of time, which enforces the reduced-order governing equations, and μ enforces the initial conditions. The optimality system can be obtained by taking first variation of the Lagrangian functional with respect to the state variables α, adjoint variables λ and μ, and optimization variables Coe.

Setting the first variation of the Lagrangian $\mathscr{L}$ with respect to λ_i to zero and arguing that the variations $\delta\lambda_i$ is arbitrary in $(0,\, t_f)$, and setting the first variation of $\mathscr{L}$ with respect to μ_i to zero, simply recovers the **State Equations**, i.e. the ROM (4.17b) with initial conditions (4.17c).

Setting the first variation of the Lagrangian $\mathscr{L}$ with respect to α_j to zero, the following

equation is obtained

$$\begin{aligned}\delta\mathscr{L}_{\alpha_j} = &\int_0^{t_f}\left\{\frac{1}{2}\frac{\partial}{\partial\alpha_j}\left[\left(\boldsymbol{g}-\hat{\boldsymbol{g}},\,\boldsymbol{g}-\hat{\boldsymbol{g}}\right)_\Omega\right]-\sum_{i=1}^{\mathrm{nM}}\lambda_i\left(B_{ij}^v+B_{ij}^p+B_{ij}^c\right)-\frac{d}{dt}\left(\sum_{i=1}^{\mathrm{nM}}\lambda_i L_{ij}\right)\right\}\delta\alpha_j dt\\ &+\sum_{i=1}^{\mathrm{nM}}\lambda_i L_{ij}\delta\alpha_j\Big|_0^{t_f}+\sum_{i=1}^{\mathrm{nM}}\mu_i\left(\boldsymbol{\Phi}_i,\,\boldsymbol{\Phi}_j\right)_\Omega\alpha_j\Big|^0\\ =&0.\end{aligned}$$

Then, arguing that the variations $\delta\alpha_j$ are arbitrary in $(0,\ t_f)$, at $t=0$, and at $t=t_f$, leads to the **Adjoint Equations**:

$$-\sum_{i=1}^{\mathrm{nM}}L_{ij}\dot{\lambda}_i-\sum_{i=1}^{\mathrm{nM}}\left(B_{ij}^v+B_{ij}^p+B_{ij}^c\right)\lambda_i=\left(\boldsymbol{g}-\hat{\boldsymbol{g}},\,\frac{\partial\hat{\boldsymbol{g}}}{\partial\alpha_j}\right)_\Omega \tag{4.19a}$$

$$\lambda_i(t_f)=0 \tag{4.19b}$$

$$\sum_{i=1}^{\mathrm{nM}}\left(\boldsymbol{\Phi}_i,\,\boldsymbol{\Phi}_j\right)_\Omega\mu_i=\sum_{i=1}^{M}L_{ij}\lambda_i(0) \tag{4.19c}$$

For clarity, we split the Lagrangian $\mathscr{L}$, where each term is denoted as $\mathscr{L}_i$. Setting the first variation of the Lagrangian $\mathscr{L}$ with respect to optimization variables Coe_{qs} to zero yields the **Gradient**:

$$\begin{aligned}\delta\mathscr{L}_{\mathrm{Coe}_{qs}}=&\frac{\delta\mathscr{L}_1}{\delta\mathrm{Coe}_{qs}}+\frac{\delta\mathscr{L}_2}{\delta\mathrm{Coe}_{qs}}+\frac{\delta\mathscr{L}_3}{\delta\mathrm{Coe}_{qs}}+\frac{\delta\mathscr{L}_4}{\delta\mathrm{Coe}_{qs}}+\frac{\delta\mathscr{L}_5}{\delta\mathrm{Coe}_{qs}}+\frac{\delta\mathscr{L}_6}{\delta\mathrm{Coe}_{qs}}+\frac{\delta\mathscr{L}_7}{\delta\mathrm{Coe}_{qs}}\\ =&0\end{aligned} \tag{4.20}$$

where each term is given by

$$\begin{aligned}\frac{\delta\mathscr{L}_1}{\delta\mathrm{Coe}_{qs}}&=\frac{\delta}{\delta\mathrm{Coe}_{qs}}\left\{\frac{1}{2}\int_0^{t_f}\left(\boldsymbol{g}-\hat{\boldsymbol{g}},\,\boldsymbol{g}-\hat{\boldsymbol{g}}\right)_\Omega dt\right\}\\ &=\int_0^{t_f}\left(\boldsymbol{g}-\hat{\boldsymbol{g}},\,\frac{\partial\hat{\boldsymbol{g}}}{\partial\mathrm{Coe}_{qs}}\right)_\Omega dt,\end{aligned} \tag{4.21a}$$

$$\begin{aligned}\frac{\delta\mathscr{L}_2}{\delta\mathrm{Coe}_{qs}}&=\frac{\delta}{\delta\mathrm{Coe}_{qs}}\left\{\frac{\beta}{2}\sum_{i,j=1}^{\mathrm{nM}}\left[\delta_{ij}-\left(\boldsymbol{\Phi}_i,\,\boldsymbol{\Phi}_j\right)_\Omega\right]^2\right\}\\ &=2\beta\sum_{j=1}^{\mathrm{nM}}\left[\left(\boldsymbol{\Phi}_q,\,\boldsymbol{\Phi}_j\right)_\Omega-\delta_{qj}\right]\left(\boldsymbol{\Phi}_j,\,\boldsymbol{\Psi}_s\right)_\Omega,\end{aligned} \tag{4.21b}$$

$$\begin{aligned}\frac{\delta\mathscr{L}_3}{\delta\mathrm{Coe}_{qs}}&=\frac{\delta}{\delta\mathrm{Coe}_{qs}}\left\{\int_0^{t_f}\sum_{i=1}^{\mathrm{nM}}\lambda_i\sum_{j=1}^{\mathrm{nM}}L_{ij}\dot{\alpha}_j dt\right\}\\ &=\int_0^{t_f}\left[\lambda_q\sum_{j=1}^{\mathrm{nM}}\dot{\alpha}_j\left(\boldsymbol{\psi}_s,\,\boldsymbol{\phi}_j\right)_\Omega+\dot{\alpha}_q\sum_{i=1}^{\mathrm{nM}}\lambda_i\left(\boldsymbol{\phi}_i,\,\boldsymbol{\psi}_s\right)_\Omega\right]dt,\end{aligned} \tag{4.21c}$$

$$
\begin{aligned}
\frac{\delta \mathscr{L}_4}{\delta \mathrm{Coe}_{qs}} &= \frac{\delta}{\delta \mathrm{Coe}_{qs}} \left\{ -\int_0^{t_f} \sum_{i=1}^{\mathrm{nM}} \lambda_i \sum_{j=1}^{\mathrm{nM}} \left(B_{ij}^{v} + B_{ij}^{p} + B_{ij}^{c} \right) \alpha_j \, dt \right\} \\
&= \int_0^{t_f} \Big\{ 2\nu \Big[\lambda_q \sum_{j=1}^{\mathrm{nM}} \alpha_j \left(\nabla \boldsymbol{\psi}_s, \nabla^s \boldsymbol{\phi}_j \right)_\Omega + \alpha_q \sum_{i=1}^{\mathrm{nM}} \lambda_i \left(\nabla \boldsymbol{\phi}_i, \nabla^s \boldsymbol{\psi}_s \right)_\Omega \Big] \\
&\quad - \Big[\lambda_q \sum_{j=1}^{\mathrm{nM}} \alpha_j \left(\nabla \cdot \boldsymbol{\psi}_s, \phi_j^p \right)_\Omega + \alpha_q \sum_{i=1}^{\mathrm{nM}} \lambda_i \left(\nabla \cdot \boldsymbol{\phi}_i, \psi_s^p \right)_\Omega \Big] \\
&\quad + \Big[\lambda_q \sum_{j=1}^{\mathrm{nM}} \alpha_j \left(\psi_s^p, \nabla \cdot \boldsymbol{\phi}_j \right)_\Omega + \alpha_q \sum_{i=1}^{\mathrm{nM}} \lambda_i \left(\phi_i^p, \nabla \cdot \boldsymbol{\psi}_s \right)_\Omega \Big] \Big\} dt,
\end{aligned} \tag{4.21d}
$$

$$
\begin{aligned}
\frac{\delta \mathscr{L}_5}{\delta \mathrm{Coe}_{qs}} &= \frac{\delta}{\delta \mathrm{Coe}_{qs}} \left\{ -\int_0^{t_f} \sum_{i=1}^{\mathrm{nM}} \lambda_i b_i^m \, dt \right\} \\
&= -\int_0^{t_f} \lambda_q \left(\boldsymbol{\psi}_s, \boldsymbol{f} \right)_\Omega dt,
\end{aligned} \tag{4.21e}
$$

$$
\begin{aligned}
\frac{\delta \mathscr{L}_6}{\delta \mathrm{Coe}_{qs}} &= \frac{\delta}{\delta \mathrm{Coe}_{qs}} \left\{ \sum_{i=1}^{\mathrm{nM}} \mu_i \sum_{j=1}^{\mathrm{nM}} \alpha_j^0 \left(\boldsymbol{\Phi}_i, \boldsymbol{\Phi}_j \right)_\Omega \right\} \\
&= \mu_q \sum_{j=1}^{\mathrm{nM}} \alpha_j^0 \left(\boldsymbol{\Psi}_s, \boldsymbol{\Phi}_j \right) \Omega + \alpha_q^0 \sum_{i=1}^{\mathrm{nM}} \mu_i \left(\boldsymbol{\Phi}_i, \boldsymbol{\Psi}_s \right)_\Omega,
\end{aligned} \tag{4.21f}
$$

$$
\begin{aligned}
\frac{\delta \mathscr{L}_7}{\delta \mathrm{Coe}_{qs}} &= \frac{\delta}{\delta \mathrm{Coe}_{qs}} \left\{ -\sum_{i=1}^{\mathrm{nM}} \mu_i \left(\boldsymbol{\Phi}_i, \boldsymbol{U}_0 \right)_\Omega \right\} \\
&= -\mu_q \left(\boldsymbol{\Psi}_s, \boldsymbol{U}_0 \right)_\Omega,
\end{aligned} \tag{4.21g}
$$

Here, $\boldsymbol{\psi}$ is used to represent the secondary basis functions for the velocity, and

$$
\boldsymbol{\psi} = [\psi^u, \psi^v, \psi^w]^T.
$$

The combined system (4.17b)-(4.17d), (4.19), and (4.20) are the first-order KKT optimality conditions for the optimization problem (4.17).

4.5. PRACTICAL NUMERICAL TECHNIQUES

Definitions of some useful parameters

One important free parameter in the Steihaug CG algorithm is the tolerance ϵ_k, which is a critical condition for the residual and has an influence on the accuracy and convergence rate of the algorithm. In the Inexact-Newton method, the forcing sequence η_k is defined as the residual over its relative gradient, i.e.

$$
\eta_k = \frac{\| r_k \|}{\| \nabla f(x_k) \|}.
$$

There are many references which give guidelines for how to choose the forcing sequence η_k, see [79, 84, 85, 102]. A good option is to set

$$
\epsilon_k = \eta_k \| \nabla f(x_k) \|. \tag{4.22}
$$

In this study the tolerance ϵ_k is determined by

$$\epsilon_k = \min\left(0.5,\ \|\nabla\mathscr{L}(\mathrm{Coe}^k)\|_2\right)\|\nabla\mathscr{L}(\mathrm{Coe}^k)\|_2. \tag{4.23}$$

A finite difference formula, (2.29), is used to represent the Hessian-vector product $(\nabla^2 f(x)d)$. For this θ needs to be determined. According to [102], we define the finite difference stepsize (θ) as

$$\theta = \frac{10^{-8}}{\|d\|_2}. \tag{4.24}$$

Integration in space and time

To calculate the **Gradient** (4.20) and the objective functional $(\mathscr{G})$ we use discrete operators for integration in space and time. For the calculation of the integration in space, a Gaussian quadrature rule is applied; for the calculation of the integration in time, the trapezoidal rule is used.

Temporal integration method

Both the **State Equations** (4.17b)-(4.17c) and the **Adjoint Equations** (4.19) are systems of ordinary differential equations, for which we use the same temporal integration method – a generalized-α method.

The generalized-α method was first developed for computational solid dynamics as chronicled by Chung and Hulbert [103]. Then Jansen *el at.*[104] extended it to the application of the Navier-Stokes equations within the context of the finite element method. The method is formulated to obtain a second-order accurate family of time integrators whose high frequency amplification factor is the sole free parameter. Such an approach allows the replication of midpoint rule (zero damping), Gear's method (maximal damping), or anything in between.

Applying the finite element method in space to the Navier-Stokes equations yields a system of nonlinear ordinary differential equations which can be written as

$$\boldsymbol{M}\dot{\boldsymbol{Y}} = \boldsymbol{N}(\boldsymbol{Y}), \tag{4.25}$$

where $\boldsymbol{Y}$ is the vector of solution at discrete points (spatially interpolated with the finite element shape functions), $\dot{\boldsymbol{Y}}$ represents the time derivative of $\boldsymbol{Y}$.

Integrating (4.25) from t_n to t_{n+1} by the generalized-α method yields

$$\boldsymbol{G}(\dot{\boldsymbol{Y}}_{n+\alpha_m}, \boldsymbol{Y}_{n+\alpha_f}) = \boldsymbol{M}_{n+\alpha_m}\dot{\boldsymbol{Y}}_{n+\alpha_m} - \boldsymbol{N}(\boldsymbol{Y}_{n+\alpha_f}) = 0, \tag{4.26a}$$

$$\boldsymbol{Y}_{n+1} = \boldsymbol{Y}_n + \Delta t\dot{\boldsymbol{Y}}_n + \gamma\Delta t(\dot{\boldsymbol{Y}}_{n+1} - \dot{\boldsymbol{Y}}_n), \tag{4.26b}$$

$$\dot{\boldsymbol{Y}}_{n+\alpha_m} = \dot{\boldsymbol{Y}}_n + \alpha_m(\dot{\boldsymbol{Y}}_{n+1} - \dot{\boldsymbol{Y}}_n), \tag{4.26c}$$

$$\boldsymbol{Y}_{n+\alpha_f} = \boldsymbol{Y}_n + \alpha_f(\boldsymbol{Y}_{n+1} - \boldsymbol{Y}_n), \tag{4.26d}$$

where $\boldsymbol{G}$ is the vector of nodal values of the nonlinear residual. A predictor-multicorrector algorithm is used to treat the nonlinearities. We start the algorithm by making a prediction of the solution and its time derivative at time t_{n+1}. Because multiple corrections are

used in the algorithm, a superscript (inside parentheses) is introduced to represent the iteration number. The predictor is given an iteration count of zero and is given by

$$\boldsymbol{Y}_{n+1}^{(0)} = \boldsymbol{Y}_n, \tag{4.27a}$$

$$\dot{\boldsymbol{Y}}_{n+1}^{(0)} = \frac{\gamma-1}{\gamma}\dot{\boldsymbol{Y}}_n, \tag{4.27b}$$

where (4.27a) predicts that the solution will be the same as it was at the previous time step and (4.27b) is the time derivative at t_{n+1} that is consistent with (4.26b) (i.e. the predictor that preserves second order accuracy).

After making the prediction, the algorithm enters a loop of multi-corrector passes with i initialized to zero. The first step within the loop is to calculate the solution at $t_{n+\alpha_f}$ and the time derivative of the solution at $t_{n+\alpha_m}$

$$\boldsymbol{Y}_{n+\alpha_f}^{(i)} = \boldsymbol{Y}_n + \alpha_f(\boldsymbol{Y}_{n+1}^{(i)} - \boldsymbol{Y}_n), \tag{4.28a}$$

$$\dot{\boldsymbol{Y}}_{n+\alpha_m}^{(i)} = \dot{\boldsymbol{Y}}_n + \alpha_m(\dot{\boldsymbol{Y}}_{n+1}^{(i)} - \dot{\boldsymbol{Y}}_n). \tag{4.28b}$$

These quantities enable the evaluation of $\boldsymbol{G}^{(i)}(\dot{\boldsymbol{Y}}_{n+\alpha_m}^{(i)}, \boldsymbol{Y}_{n+\alpha_f}^{(i)})$ which, for small i can be expected to be far from its desired value of $\mathbf{0}$. To find an improvement to the current values of (4.28a) and (4.28b), a Newton's linearization of $\boldsymbol{G}^{(i)}$ with respect to the solution variable is utilised, viz.,

$$\boldsymbol{G}^{(i)}\left(\dot{\boldsymbol{Y}}_{n+\alpha_m}^{(i)}, \boldsymbol{Y}_{n+\alpha_f}^{(i)}\right) + \frac{\partial \boldsymbol{G}^{(i)}\left(\dot{\boldsymbol{Y}}_{n+\alpha_m}^{(i)}, \boldsymbol{Y}_{n+\alpha_f}^{(i)}\right)}{\partial \boldsymbol{Y}_{n+\alpha_f}^{(i)}} \Delta \boldsymbol{Y}_{n+\alpha_f}^{(i)} = \mathbf{0}. \tag{4.29}$$

The first term of (4.29) is the nonlinear residual, the second term of (4.29) is the tangent matrix multiplying the increment of the solution. This yields a linear matrix problem to be solved for each corrector step

$$\boldsymbol{K}^{(i)} \Delta \boldsymbol{Y}_{n+\alpha_f}^{(i)} = -\boldsymbol{G}^{(i)}, \tag{4.30}$$

where $\boldsymbol{K}^{(i)}$ is the tangent matrix. Once this system is solved for $\Delta \boldsymbol{Y}_{n+\alpha_f}^{(i)}$, the solution is updated

$$\boldsymbol{Y}_{n+\alpha_f}^{(i+1)} = \boldsymbol{Y}_{n+\alpha_f}^{(i)} + \Delta \boldsymbol{Y}_{n+\alpha_f}^{(i)}, \tag{4.31}$$

$$\dot{\boldsymbol{Y}}_{n+\alpha_m}^{(i+1)} = \left(1 - \frac{\alpha_m}{\gamma}\right)\dot{\boldsymbol{Y}}_n + \frac{\alpha_m}{\gamma \Delta t \alpha_f}\left(\boldsymbol{Y}_{n+\alpha_f}^{(i+1)} - \boldsymbol{Y}_n\right), \tag{4.32}$$

and i is incremented. If $i < i_{max}$, the algorithm returns to solve (4.29), then initiating the next corrector pass. Otherwise, the solution and its time derivative at the time t_{n+1} is determined as follows

$$\boldsymbol{Y}_{n+1} = \boldsymbol{Y}_n + \frac{\boldsymbol{Y}_{n+\alpha_f}^{(i_{max})} - \boldsymbol{Y}_n}{\alpha_f}, \tag{4.33}$$

$$\dot{\boldsymbol{Y}}_{n+1} = \dot{\boldsymbol{Y}}_n + \frac{\dot{\boldsymbol{Y}}_{n+\alpha_m}^{(i_{max})} - \dot{\boldsymbol{Y}}_n}{\alpha_m}. \tag{4.34}$$

This completes the step from $t_n \longrightarrow t_{n+1}$.

To define a second-order accurate family of methods with a specified high-frequency damping, the parameters α_m, α_f, and γ are defined as

$$\alpha_m = \frac{1}{2}\left(\frac{3-\rho_\infty}{1+\rho_\infty}\right), \tag{4.35}$$

$$\alpha_f = \frac{1}{1+\rho_\infty}, \tag{4.36}$$

$$\gamma = \frac{1}{2} + \alpha_m - \alpha_f. \tag{4.37}$$

In this generalize-α algorithm, we set $\rho_\infty = \frac{1}{2}$.

In summary, the Generalized-α Algorithm is given by

Generalized-α Algorithm

Given ϵ_{GA} (a very small number)

Set $\rho_\infty = \frac{1}{2}$

Compute α_m, α_f, and γ through (4.35), (4.36) and (4.37)

% Predictor step

Initialize $\boldsymbol{Y}_{n+1}^{(0)}$ and $\dot{\boldsymbol{Y}}_{n+1}^{(0)}$ by (4.27a), (4.27b)

Compute $\boldsymbol{Y}_{n+\alpha_f}^{(0)}$ and $\dot{\boldsymbol{Y}}_{n+\alpha_m}^{(0)}$ by (4.28a), (4.28b)

% Multi-corrector loop

Initialize $i = 0$, $\|\Delta \boldsymbol{Y}\|_2 = 1$

while $i < i_{max}$ and $\|\Delta \boldsymbol{Y}\|_2 > \epsilon_{GA}$

Compute $\boldsymbol{G}^{(i)}(\dot{\boldsymbol{Y}}_{n+\alpha_m}^{(i)}, \boldsymbol{Y}_{n+\alpha_f}^{(i)})$ according with (4.26a);

Compute the tangent matrix $\boldsymbol{K}^{(i)}$;

Compute $\Delta \boldsymbol{Y}_{n+\alpha_f}^{(i)}$ according with (4.30);

Compute $\boldsymbol{Y}_{n+\alpha_f}^{(i+1)}$ and $\dot{\boldsymbol{Y}}_{n+\alpha_m}^{(i+1)}$ according with (4.31), (4.32);

i++;

do

Update $\boldsymbol{Y}_{n+1}$ and $\dot{\boldsymbol{Y}}_{n+1}$ according with (4.33) and (4.34)

4.6. VERIFICATION STUDY

Our optimization code is verified through studying a very simple case, where the velocity and pressure are assumed to be the following sinusoidal function of space (x and y) and time (t),

Exact velocity and pressure:

$$u,\ v,\ p = \sin(\pi x)\sin(\pi y)\sin^2(\pi t) \tag{4.38}$$

where $x,\ y \in [0, 1]$ and $t \in [0, 0.5]$ (a half period for the velocity and pressure).

From (4.38), we can see that only one primary basis function in space is needed to approximate the velocity and pressure exactly:

$$\phi^{u,v,p}_{\text{exact}} = c\sin(\pi x)\sin(\pi y) \tag{4.39}$$

where c is a nonzero constant. We will orthonormalize the exact primary basis function (4.39) so that we can compare it with the optimal primary basis function obtained through the SCF. Considering the orthonormal constraint (see Section 4.4), c is chosen to satisfy

$$\int_0^1\int_0^1 \left(\phi^u_{\text{exact}}\phi^u_{\text{exact}} + \phi^v_{\text{exact}}\phi^v_{\text{exact}} + \phi^p_{\text{exact}}\phi^p_{\text{exact}}\right) dxdy = 1$$

to obtain the exact orthonormal primary basis function:

$$\phi^{u,v,p}_{\text{orth}} = \frac{2}{\sqrt{3}}\sin(\pi x)\sin(\pi y) \tag{4.40}$$

In order to unambiguously verify the method, the secondary basis functions $\boldsymbol{\Psi}_l$ are chosen to be sine functions of space with same frequency in $x-$ and $y-$direction,

Secondary basis functions:

$$\boldsymbol{\Psi}_l = \begin{bmatrix} \psi^u_l \\ \psi^v_l \\ \psi^p_l \end{bmatrix} = \sin(l\pi x)\sin(l\pi y) \qquad l = 1, 2, \cdots, \text{nB} \tag{4.41}$$

As (4.13), the primary basis functions are constructed of the secondary basis functions, so that for this case, there is only one primary basis function which needs to be optimized (that is nM = 1). The optimized primary basis function is

Optimized primary basis function:

$$\boldsymbol{\Phi} = \begin{bmatrix} \phi^u_1 \\ \phi^v_1 \\ \phi^p_1 \end{bmatrix} = \sum_{l=1}^{\text{nB}} \text{Coe}_{1l}\boldsymbol{\psi}_l = \sum_{l=1}^{\text{nB}} \text{Coe}_{1l}\sin(l\pi x)\sin(l\pi y) \tag{4.42}$$

Comparing the optimized primary basis function (4.42) with the exact primary basis function (4.40), we see that in theory the optimal coefficients Coe_{1l} should be

Theoretical optimal coefficients:

$$\mathrm{Coe}_{1l} = \begin{cases} \frac{2}{\sqrt{3}} & l = 1 \\ 0 & l = 2, \cdots, \mathrm{nB} \end{cases} \tag{4.43}$$

That is, the optimization coefficients (Coe_{1l}) are expected to satisfy (4.43) to verify the code.

Now we will discuss the choice of Δ_0 – initial trust region radius. Since the choice of Δ_0 can affect the efficiency of the trust region method [105–108], Δ_0 selection may be considered as important. In [105], Sartenaer provided a new strategy for determining Δ_0 which prevented the algorithm from decreasing the efficiency of the trust region method since in Sartenaer's strategy Δ_0 is set to be dependent on the information from gradient and Hessian-matrix. We will investigate the following choices of Δ_0

In this case, choices of Δ_0: (referring to [86, 105])

(a) $\Delta_0 = \|\nabla \mathscr{L}(\mathrm{Coe}^0)\|_2$;

(b) $\Delta_0 = 0.1\|\nabla \mathscr{L}(\mathrm{Coe}^0)\|_2$;

(c) $\Delta_0 = \frac{\|\nabla \mathscr{L}(\mathrm{Coe}^0)\|_2^2}{\nabla \mathscr{L}^T(\mathrm{Coe}^0)\nabla^2 \mathscr{L}(\mathrm{Coe}^0)\nabla \mathscr{L}(\mathrm{Coe}^0)} \|\nabla \mathscr{L}(\mathrm{Coe}^0)\|_2$.

The remaining parameters which should be determined for our optimization problem are the regularization parameter β and the kinematic coefficient of viscosity ν. Here we use

$$\beta = 2, \qquad \nu = 0.1.$$

4.6.1. RESULTS FOR TWO SECONDARY BASIS FUNCTIONS

In this section, the primary basis function which needs to be found is constructed of two secondary basis functions, i.e.

$$\mathbf{\Phi} = \begin{bmatrix} \phi_1^u \\ \phi_1^v \\ \phi_1^p \end{bmatrix} = \mathrm{Coe}_{11} \sin(\pi x)\sin(\pi y) + \mathrm{Coe}_{12} \sin(2\pi x)\sin(2\pi y),$$

and three goal functionals are chosen to be tested. Besides, to see the influence of initial coefficients on the results, we have tested three initial guesses for the coefficients (Coe_{1l}) which are:

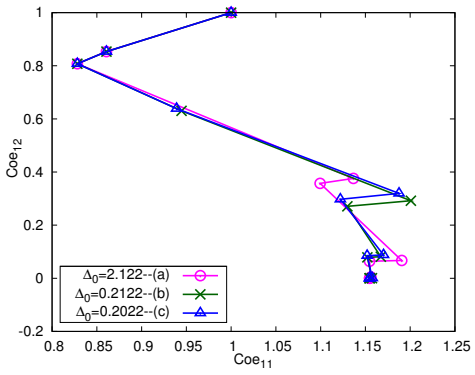
- $\mathrm{Coe}^0 = (1,\ 1)$
- $\mathrm{Coe}^0 = (0.5,\ 0.5)$
- $\mathrm{Coe}^0 = (1,\ 0.5)$

where $\mathrm{Coe}^0 = \left(\mathrm{Coe}_{11}^0,\ \mathrm{Coe}_{12}^0\right)$.

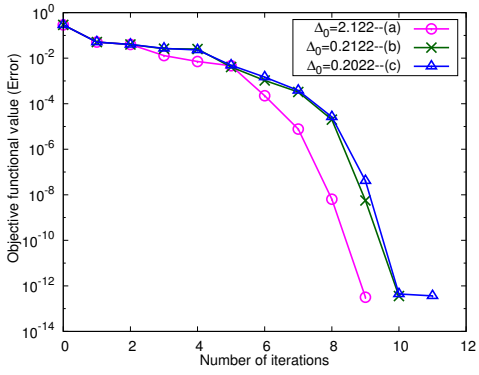
(1) Global linear: $\boldsymbol{g} = [u,\ v,\ p]^T\ \ (x, y \in [0, 1])$
This goal functional is chosen since we want to investigate the effect of the reduced-order underlying governing equations on the determination of primary basis functions. Fig.4.1(a), Fig.4.1(c) and Fig.4.1(e) show the trajectory of the two coefficients, Coe_{11} and Coe_{12}, during the optimization procedure. Starting from different initial coefficients and different initial trust region radius, in all tests, the first coefficient Coe_{11} converges to 1.1547, and the find value of the second coefficient $|\text{Coe}_{12}|$ is usually smaller than $1e-6$. That is, the theoretical optimal coefficients, $\text{Coe}_{11} = \frac{2}{\sqrt{3}} \approx 1.154700538$ and $\text{Coe}_{12} = 0$, are obtained by implementing our optimization code in the allowable range of computational error. The objective functional value $\mathscr{G} < 4e-13$ (see Fig.4.1(b), Fig.4.1(d), Fig.4.1(f)), which means that we can reproduce the exact goal functional with the optimal primary basis function obtained by the SCF. We also see that the chosen initial trust region radius has a small influence on the results. One reason might be the initial coefficients are all close to the theoretically optimal ones and that all of the three Δ_0 are in a reasonable range.

(2) Local linear: $\boldsymbol{g} = [u,\ v,\ p]^T\ \ (x, y \in [0, 0.5])$
Sometimes we may be interested in the velocity and pressure in part of the space domain. Here the velocity and pressure in $x, y \in [0, 0.5]$ are of interest. From Fig.4.2, we can see that in all tests the theoretically optimal coefficients are still obtained within the specified iteration tolerance ($\text{Coe}_{11} \rightarrow 1.15470051$ and $|\text{Coe}_{12}| < 6e-7$). The difference between the exact goal functional ($\boldsymbol{g}$) and the reduced-order goal functional ($\hat{\boldsymbol{g}}$) is negligible. For this goal functional, we still could not see an obvious influence of the chosen initial trust region radius on the results, since the exact solution was in the secondary basis $\boldsymbol{\Psi}$.

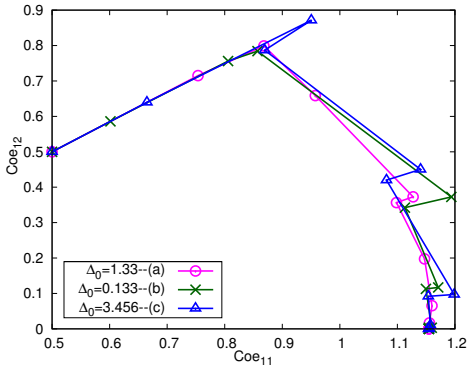
(3) Global nonlinear: $\boldsymbol{g} = [u^2,\ v^2,\ p^2]^T\ \ (x, y \in [0, 1])$
This nonlinear goal functional: $\boldsymbol{g} = [u^2,\ v^2,\ p^2]^T\ \ (x, y \in [0, 1])$ is tested to continue verifying our optimization code. As in the last two test cases, from Fig.4.3, it is seen that the theoretical optimal coefficients are still obtained within its specified iteration tolerance (the SCF coefficients are $\text{Coe}_{11} \approx 1.1547003$ and $|\text{Coe}_{12}| < 2e-6$) and that the initial radius (Δ_0) has a small influence on the convergence behavior of the optimized coefficients.



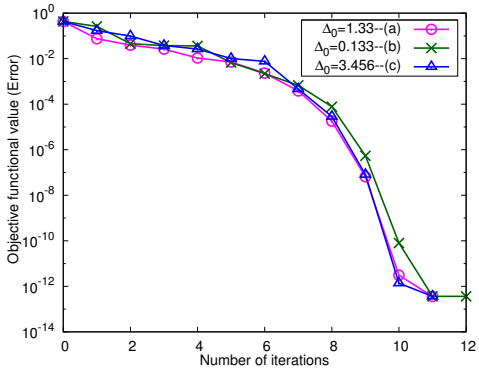
(a) $\mathrm{Coe}^0 = (1, 1)$



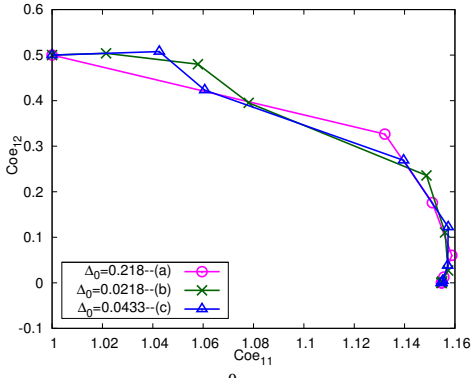
(b) $\mathrm{Coe}^0 = (1, 1)$



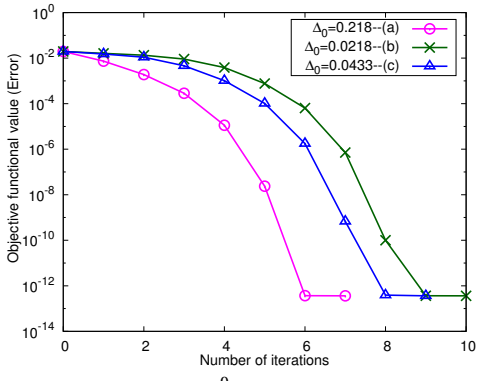
(c) $\mathrm{Coe}^0 = (0.5, 0.5)$



(d) $\mathrm{Coe}^0 = (0.5, 0.5)$

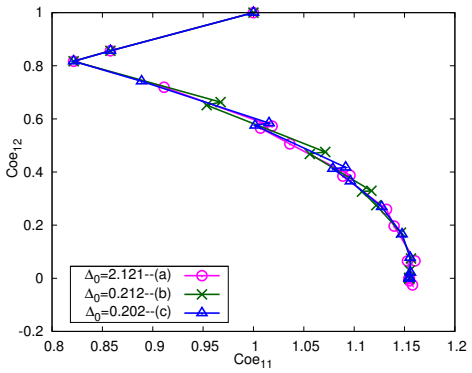


(e) $\mathrm{Coe}^0 = (1, 0.5)$

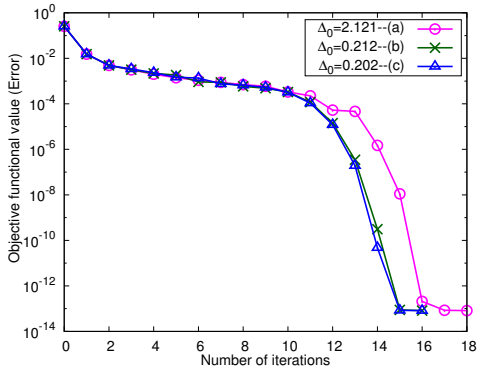


(f) $\mathrm{Coe}^0 = (1, 0.5)$

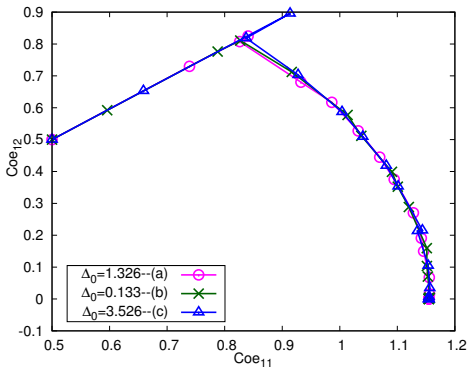
Figure 4.1: Trajectory of Coe and error versus iterations when $g = [u,\ v,\ p]^T$ $(x, y \in [0, 1])$ and nB = 2



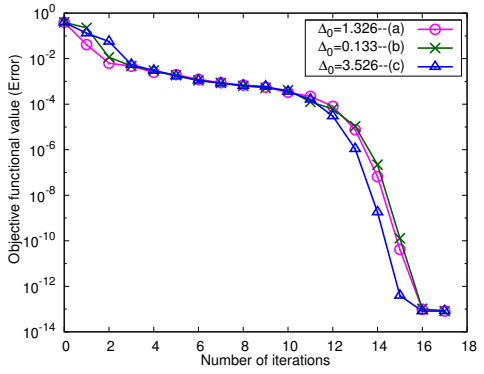
(a) $\mathrm{Coe}^0 = (1, 1)$



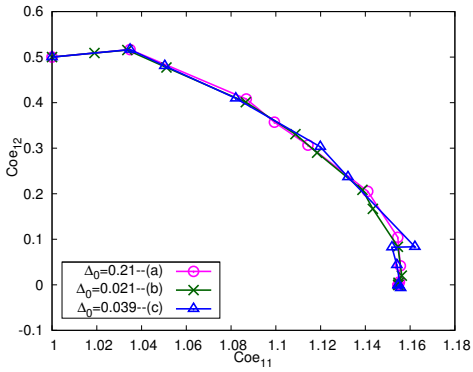
(b) $\mathrm{Coe}^0 = (1, 1)$



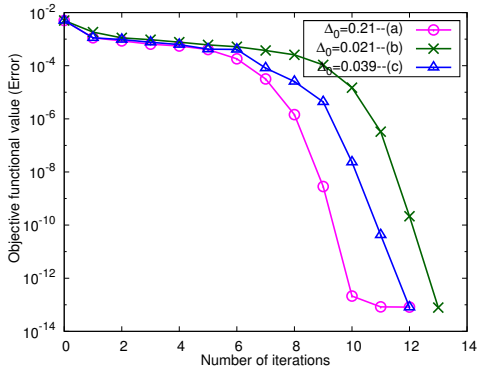
(c) $\mathrm{Coe}^0 = (0.5, 0.5)$



(d) $\mathrm{Coe}^0 = (0.5, 0.5)$

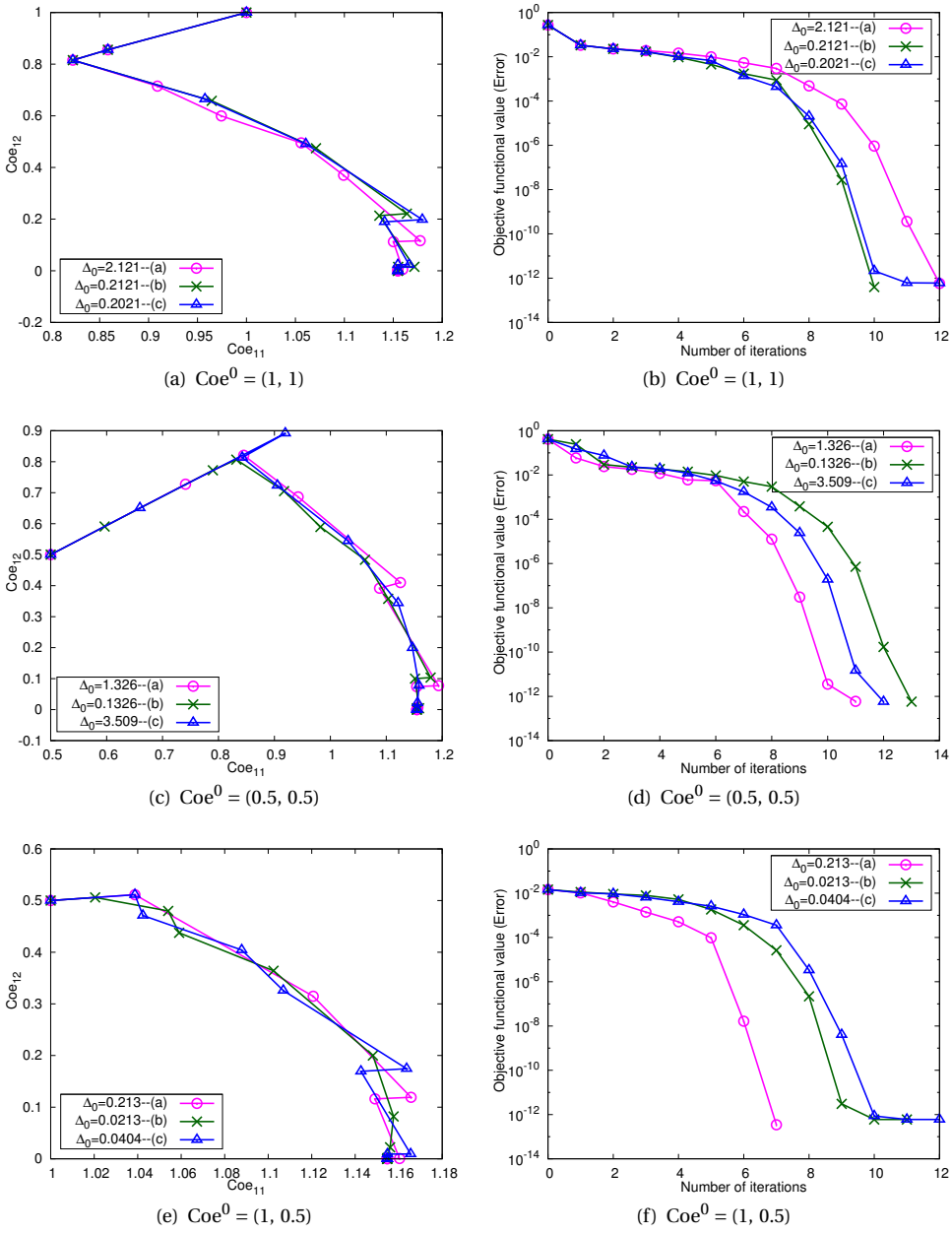


(e) $\mathrm{Coe}^0 = (1, 0.5)$



(f) $\mathrm{Coe}^0 = (1, 0.5)$

Figure 4.2: Trajectory of Coe and error versus iterations when $\boldsymbol{g} = [u,\ v,\ p]^T$ $(x, y \in [0, 0.5])$ and nB = 2



(a) $\mathrm{Coe}^0 = (1, 1)$ (b) $\mathrm{Coe}^0 = (1, 1)$

(c) $\mathrm{Coe}^0 = (0.5, 0.5)$ (d) $\mathrm{Coe}^0 = (0.5, 0.5)$

(e) $\mathrm{Coe}^0 = (1, 0.5)$ (f) $\mathrm{Coe}^0 = (1, 0.5)$

Figure 4.3: Trajectory of Coe and error versus iterations when $\boldsymbol{g} = [u^2,\ v^2,\ p^2]^T$ $(x, y \in [0, 1])$ and nB = 2

4.6.2. RESULTS FOR FIVE SECONDARY BASIS FUNCTIONS

If more secondary basis functions are used to construct the primary basis function, are the theoretical coefficients still be obtained through our optimization code? Hence, we increase the number of secondary basis functions to five, i.e. nB = 5. Here results are presented when we set the initial guess for the coefficients to be:

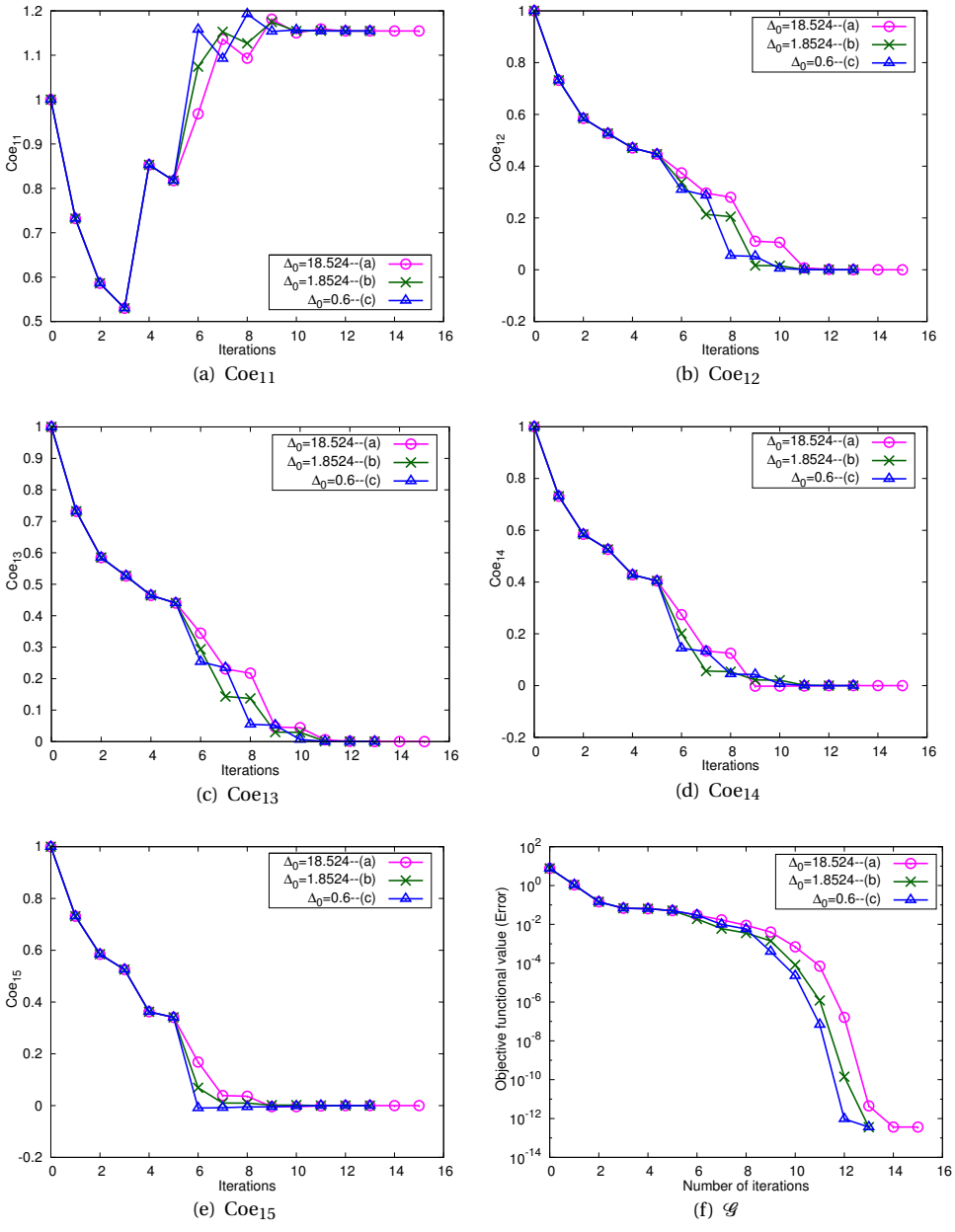
$$\mathrm{Coe}^0 = \left(\mathrm{Coe}^0_{11}, \mathrm{Coe}^0_{12}, \mathrm{Coe}^0_{13}, \mathrm{Coe}^0_{14}, \mathrm{Coe}^0_{15}\right) = (1,\ 1,\ 1,\ 1,\ 1)$$

From Fig.4.4-4.6, we see that starting from the different initial trust region radius, the SCF coefficients are:

- $\mathrm{Coe} = \left(1.15470042, |\mathrm{Coe}_{1l}| < 1e-6\right)$ when $\boldsymbol{g} = [u,\ v,\ p]^T$ $(x, y \in [0, 1])$.
- $\mathrm{Coe} = \left(1.15470051, |\mathrm{Coe}_{1l}| < 1e-6\right)$ when $\boldsymbol{g} = [u,\ v,\ p]^T$ $(x, y \in [0, 0.5])$.
- $\mathrm{Coe} = \left(1.15470043, |\mathrm{Coe}_{1l}| < 2e-6\right)$ when $\boldsymbol{g} = [u^2,\ v^2,\ p^2]^T$ $(x, y \in [0, 1])$.

Here, $l = 2, \cdots, 5$. That is, if only the secondary basis includes the exact solution, whatever the goal functional is, with a specified iteration tolerance, we can obtain the theoretical optimal coefficients (4.43) using the SCF, and then the exact solution is reproduced.

Seeing Fig.4.4 and Fig.4.5, the initial trust region radius still has little influence on the convergence behavior. While, in Fig.4.6, we see that when $\Delta_0 = 18.524$ the convergence behavior is different from that when $\Delta_0 = 1.852$ and $\Delta_0 = 0.6$ and the convergence rate of the coefficients is slower.

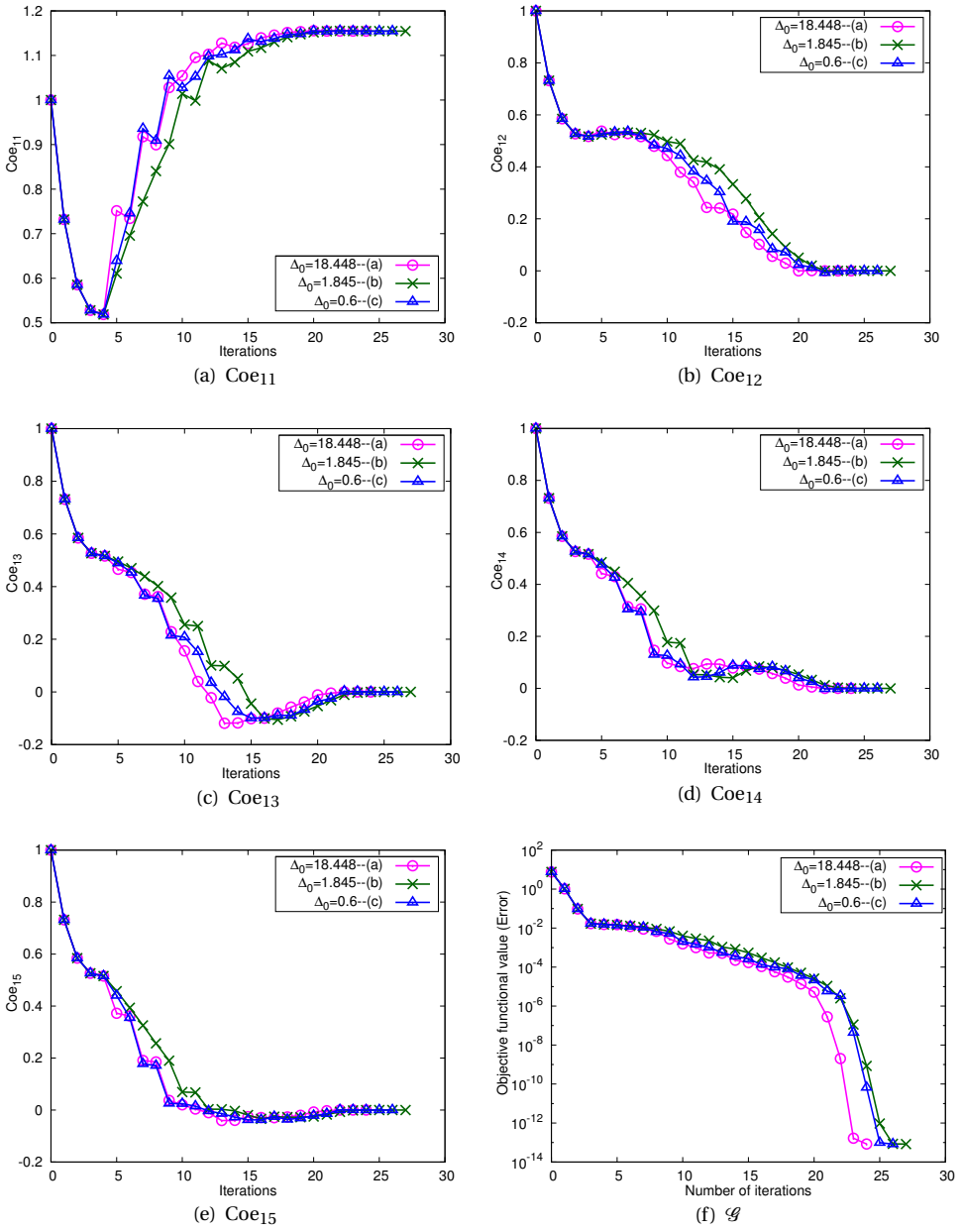


(a) Coe_{11} (b) Coe_{12}

(c) Coe_{13} (d) Coe_{14}

(e) Coe_{15} (f) $\mathscr{G}$

Figure 4.4: Coe and error versus iterations when $g = [u,\ v,\ p]^T$ $(x, y \in [0, 1])$ and nB = 5

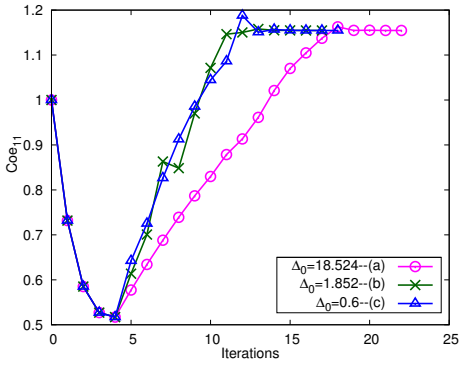


(a) Coe_{11} (b) Coe_{12}

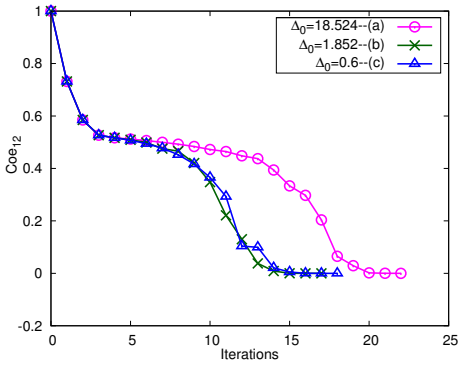
(c) Coe_{13} (d) Coe_{14}

(e) Coe_{15} (f) $\mathscr{G}$

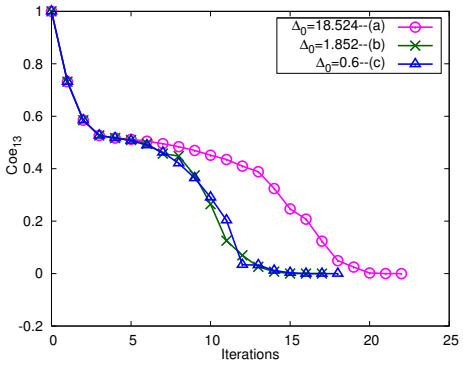
Figure 4.5: Coe and error versus iterations when $g = [u,\ v,\ p]^T$ $(x, y \in [0, 0.5])$ and nB = 5



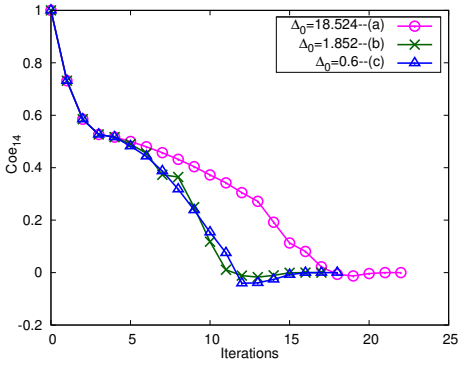
(a) Coe_{11}



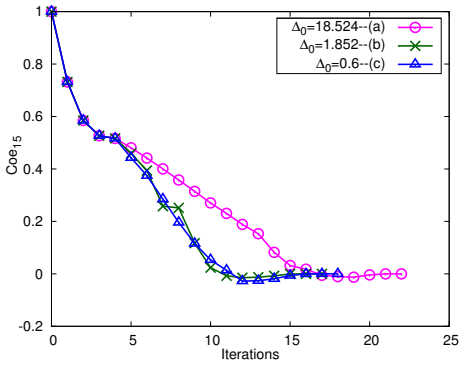
(b) Coe_{12}



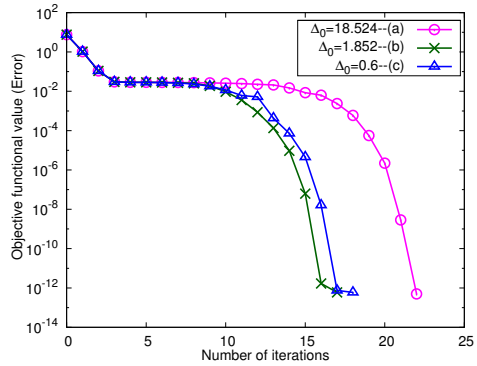
(c) Coe_{13}



(d) Coe_{14}



(e) Coe_{15}



(f) $\mathscr{G}$

Figure 4.6: Coe and error versus iterations when $g = [u^2,\ v^2,\ p^2]^T$ $(x, y \in [0, 1])$ and nB = 5

4.7. APPROXIMATE CASE STUDY

With the algorithm verified, we now consider an approximate case more representative of realistic applications. Thus, we assume that the secondary basis does not include functions present in the reference velocity and pressure.

4.7.1. PROBLEM DESCRIPTION

As in the verification case, it is assumed that the velocity and pressure can be modeled adequately using only one primary basis function. Here the exact velocity and pressure are chosen to be:

Exact velocity and pressure:

$$u,\ v,\ p = 9000x^3(1-x)^3(1-2x)^2y^2(1-y)^2\sin^2(\pi t) \tag{4.44}$$

where $x,\ y \in [0, 1]$ and $t \in [0, 0.5]$ (a half period for the velocity and pressure).

The magnitude of the exact velocity and pressure increases in time. Fig.4.7 shows the exact velocity and pressure at $t = 0.5$.

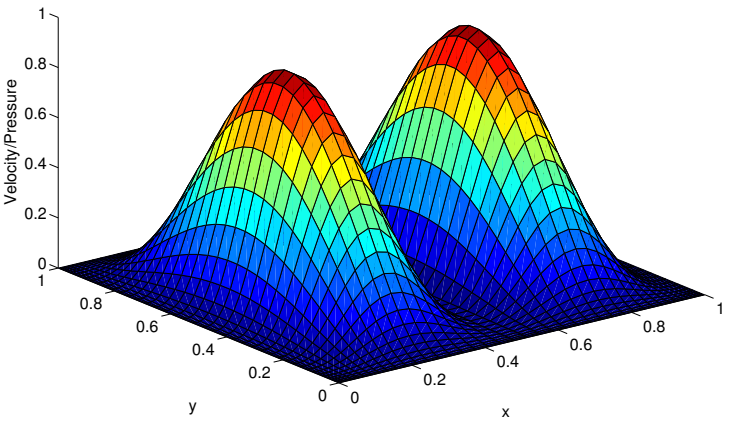


Figure 4.7: The velocity/pressure at $t = 0.5$

In this case, we will use a sinusoidal secondary basis, meaning that the exact solution can not be reproduced exactly using a finite number of sine/cosine functions. Taking into account the homogeneous Dirichlet boundary conditions, we define the secondary basis functions as:

Secondary basis functions:

$$\mathbf{\Psi}_l = \begin{bmatrix} \psi_l^u \\ \psi_l^v \\ \psi_l^p \end{bmatrix} = \sin(k_x \pi x)\sin(k_y \pi y) \qquad l = 1, 2, \cdots, \text{nB} \tag{4.45}$$

where

$$k_x,\ k_y \in \left[1,\ \sqrt{\text{nB}}\right]$$

and

$$l = \sqrt{\text{nB}}(k_x - 1) + k_y \tag{4.46}$$

which defines the index of the secondary basis functions.

For example, when nB = 4, the secondary basis functions are:

$$\begin{bmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \end{bmatrix}^{u,\,v,\,p} = \begin{bmatrix} \sin(\pi x)\sin(\pi y) \\ \sin(\pi x)\sin(2\pi y) \\ \sin(2\pi x)\sin(\pi y) \\ \sin(2\pi x)\sin(2\pi y) \end{bmatrix}$$

From (4.44), it is seen that the velocity and pressure could be modeled by using only one primary basis function. Moreover, the primary basis functions are constructed of vector-valued secondary basis functions, (4.13). Hence, in this test case, the primary basis function to be optimized is defined as:

Primary basis function to be optimized:

$$\mathbf{\Phi} = \begin{bmatrix} \phi_1^u \\ \phi_1^v \\ \phi_1^p \end{bmatrix} = \sum_{l=1}^{\text{nB}} \text{Coe}_{1l}\mathbf{\Psi}_l = \sum_{l=1}^{\text{nB}} \text{Coe}_{1l}\sin(k_x \pi x)\sin(k_y \pi y) \tag{4.47}$$

As before, we will investigate three different goal functionals – two linear and one nonlinear:

$$\boldsymbol{g} = \begin{cases} [u,\ v,\ p]^T & \text{where } x, y \in [0, 1]) \\ [u,\ v,\ p]^T & \text{where } x, y \in [0, 0.5] \\ [u^2,\ v^2,\ p^2]^T & \text{where } x, y \in [0, 1] \end{cases}$$

and use

$$\beta = 2, \quad \nu = 0.1.$$

4.7.2. RESULTS

For all of the three goal functionals, the optimization procedure is started with the same initial guess for the coefficients:

$$\text{Coe}_{1l}^0 = \begin{cases} 1 & l = 1 \\ 0 & l = 2, \cdots, \text{nB} \end{cases}$$

(1) Global linear: $\boldsymbol{g} = [u,\ v,\ p]^T$ $(x, y \in [0, 1])$
Here, we choose the solution in the whole space domain to be the goal functional. Fig.4.8 shows changes of the objective functional value with the number of secondary basis functions. It is seen that the value of the objective functional, i.e. the error between the exact solution and the solution obtained by solving the ROM, is reduced with the increasing number of secondary basis functions. When increasing the number of secondary basis functions from 1, 9, 25 to 4, 16, 36, respectively, the difference of the error has an order of 10^{-6}. Thus when at least one of k_x and k_y is an even number (e.g. $k_x,\ k_y = 2,\ 4,\ \cdots$), the contribution of the additional basis functions is negligible.

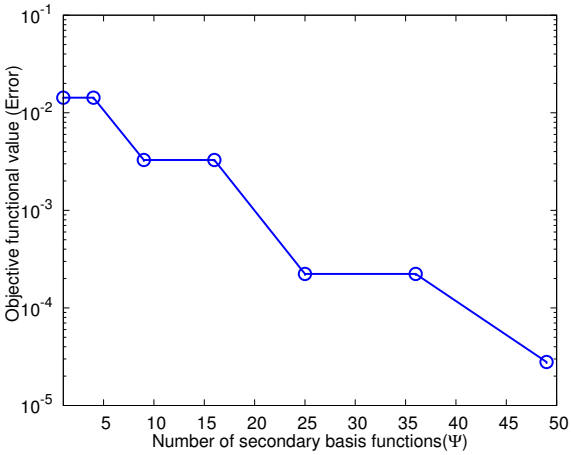
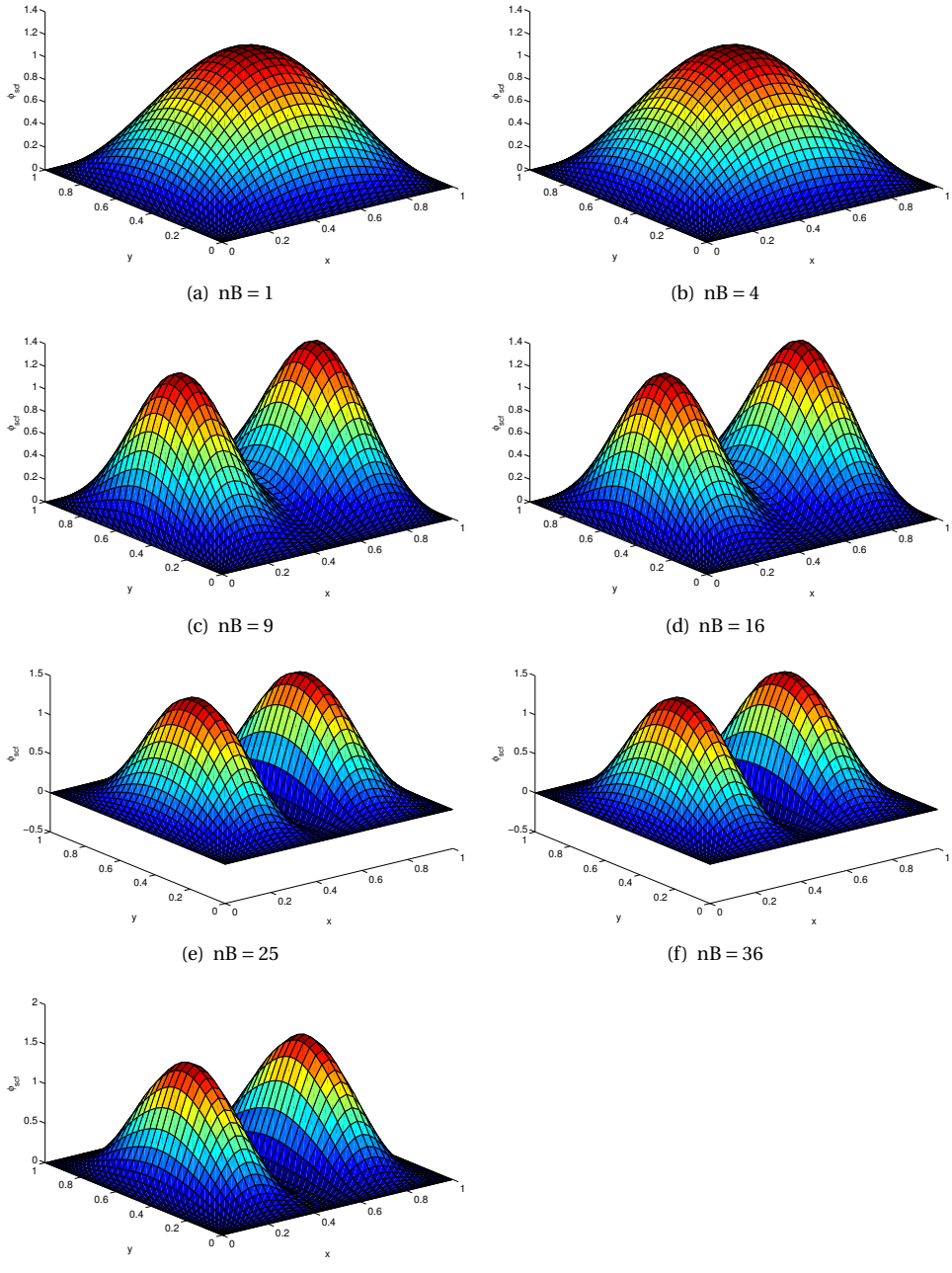


Figure 4.8: Error versus the number of secondary basis functions when $\boldsymbol{g} = [u,\ v,\ p]^T$ $(x, y \in [0, 1])$

From Fig.4.9, we see that the optimal primary basis function consisting of 4, 16 and 36 secondary basis functions does not deviate from that consisting of 1, 9 and 25 secondary basis functions, respectively. It is to be expected due to the symmetry existing in the exact solution and explains why there is no improvement when the number of secondary basis functions is increased from 1 to 4, from 9 to 16, and from 25 to 36. Therefore, if you are aware of symmetries that exist in the reference data, then you should exploit it in your choice of secondary basis to reduce the size of the optimization problem.



(a) nB = 1 (b) nB = 4

(c) nB = 9 (d) nB = 16

(e) nB = 25 (f) nB = 36

(g) nB = 49

Figure 4.9: Optimal primary basis function constructed of a different number of secondary basis functions when $\boldsymbol{g} = [u,\ v,\ p]^T (x, y \in [0, 1])$

(2) Local linear: $\boldsymbol{g} = [u,\ v,\ p]^T$ $(x, y \in [0, 0.5])$
When the velocity and pressure in a quarter of space domain are of interest, Fig.4.10 shows that the error is gradually reduced from 10^{-3} to 10^{-6} when we increase the number of secondary basis functions from nB = 1 to nB = 49. Whenever increasing the number of second basis functions from an odd number to an even number or from an even number to an odd number, such as from nB = 1 to nB = 4 and from nB = 4 to nB = 9, the error always decreases.

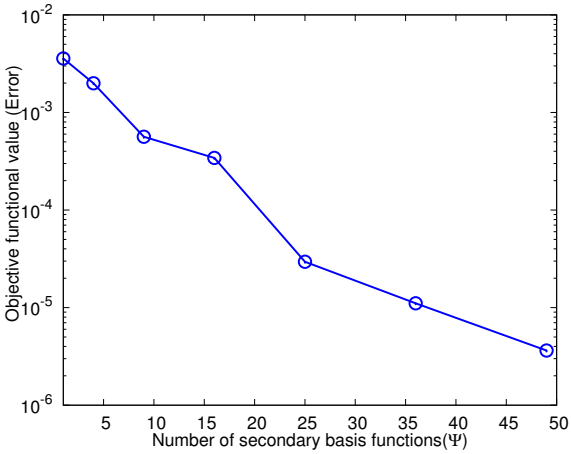
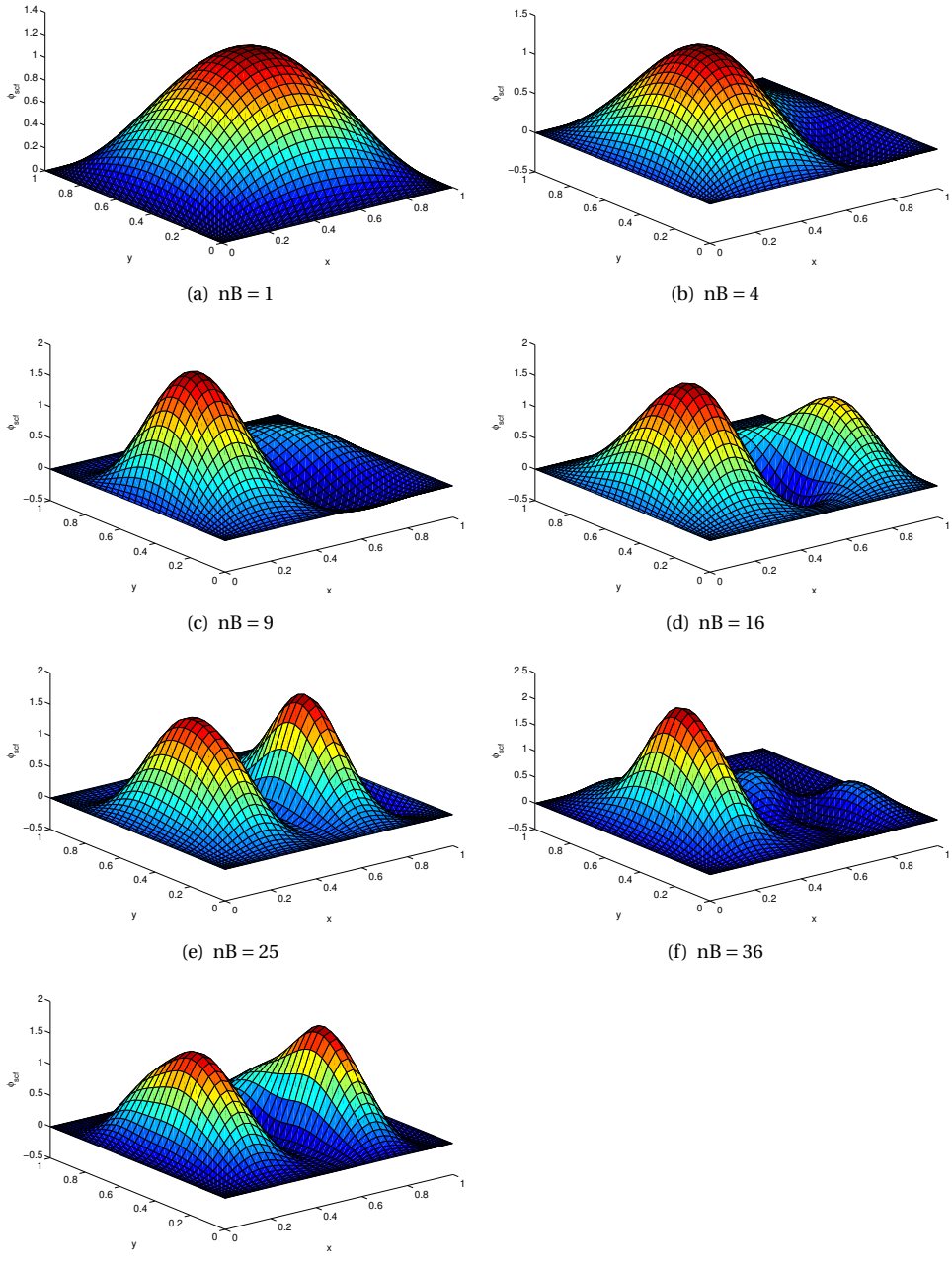


Figure 4.10: Error versus the number of secondary basis functions when $\boldsymbol{g} = [u,\ v,\ p]^T$ $(x, y \in [0, 0.5])$

Fig.4.11 shows the development of the optimal primary basis function with the increasing number of secondary basis functions. The optimal primary basis function in $x, y \in [0, 0.5]$ becomes close to the exact solution as the number of secondary basis functions is increased; however, outside of this region (i.e. $x \geq 0.5$ or $y \geq 0.5$), it sometimes is very different from the exact solution. This result indicates the effectiveness of the goal-oriented approach, but also implies that the weighted functional approach introduced in Chapter 3 should be used if a realistic solution outside of the region of interest is desired as well.



(a) nB = 1

(b) nB = 4

(c) nB = 9

(d) nB = 16

(e) nB = 25

(f) nB = 36

(g) nB = 49

Figure 4.11: Optimal primary basis function constructed of a different number of secondary basis functions when $\boldsymbol{g} = [u,\ v,\ p]^T\ (x, y \in [0, 0.5])$

(3) Global nonlinear: $\boldsymbol{g} = [u^2,\ v^2,\ p^2]^T$ $(x, y \in [0, 1])$
When square of the solution is of interest — $\boldsymbol{g} = [u^2,\ v^2,\ p^2]^T$ $(x, y \in [0, 1])$, trends similar to $\boldsymbol{g} = [u,\ v,\ p]^T$ $(x, y \in [0, 1])$ are observed (Fig.4.12 versus Fig.4.8 and Fig.4.13 versus Fig.4.9), e.g.,

- Fig.4.12 shows that the error is reduced with the increasing number of secondary basis functions, and it almost does not change when we increase the number of secondary basis functions from nB = 1, 9, 25 to nB = 4, 16, 36, respectively;
- Fig.4.13 shows that the difference between the optimal primary basis function consisting of 1 (9 or 25) secondary basis functions and that consisting of 4 (16 or 36) secondary basis functions is negligible, and the optimal primary basis function becomes close to the exact primary basis function when increasing the number of secondary basis functions (comparing Fig.4.12 with Fig.4.7).

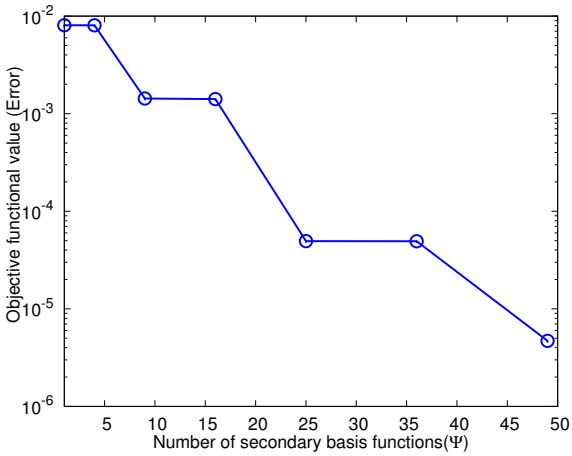
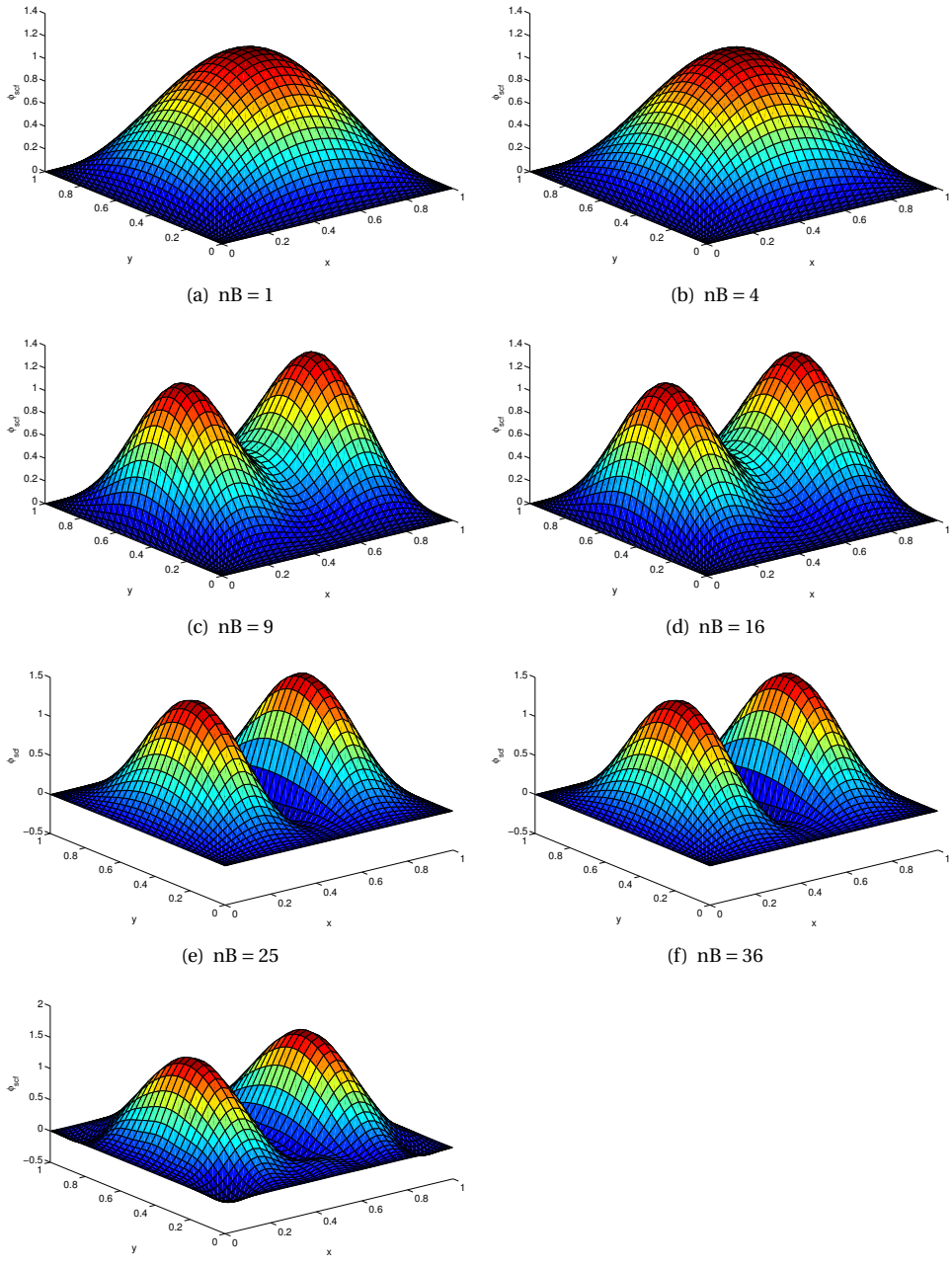


Figure 4.12: Error versus the number of secondary basis functions when $\boldsymbol{g} = [u^2,\ v^2,\ p^2]^T (x, y \in [0, 1])$



(a) nB = 1

(b) nB = 4

(c) nB = 9

(d) nB = 16

(e) nB = 25

(f) nB = 36

(g) nB = 49

Figure 4.13: Optimal primary basis function constructed of a different number of secondary basis functions when $\boldsymbol{g} = [u^2,\ v^2,\ p^2]^T (x, y \in [0, 1])$

4.8. SUMMARY

In this chapter, the construction of ROMs was illustrated for the two-dimensional Stokes equations which include multiple partial differential equations and multiple variables (i.e. u, v, p), as was the construction of the primary/secondary basis functions for the multiple variables.

The optimization algorithm was first verified. The results confirm that if the space function of the exact velocity and pressure is one of the secondary basis functions, then the exact orthonormal primary basis function will be obtained, independent of the choice of goal functional, the number of the secondary basis functions, initial trust region radius and initial guesses for the coefficients. As mentioned previously, the choice of the initial trust region radius will affect the convergence behavior.

Then, the SCF was applied to an approximate case. The results show that when increasing the number of the secondary basis functions the estimation of the goal functional is rapidly improved. This demonstrates the viability of the SCF for determining an optimal primary basis for a particular goal functional in two-dimensional Stokes problems. It is noted that if symmetries are present in the reference data, we should choose an appropriate secondary basis in order to minimize the size of the optimization problem. However, even if this is not done, then the correct optimum will still be obtained by the SCF algorithm at an increased cost.

5

Conclusions & outlook

In this chapter, we give a summary for this thesis. It also presents the outlook based on our current researches.

5.1. CONCLUSIONS

This thesis presents a semi-continuous formulation (SCF) to determine bases for reduced-order models which optimally represent a specific goal functional. In the SCF, we pose the problem of determining an optimal primary basis as a goal-oriented minimization problem that seeks to minimize the difference between the full-space goal functional and reduced-order goal functional over a time interval, with the reduced-order model for the underlying governing equations imposed as a constraint. In contrast with the previously developed FDF (introduced in [69]), the SCF separates the governing equation used for the constrained ROM and the discrete reference dataset. It thus allows a straightforward application to problems where the algebraic system which can reproduce the reference data is undefined or unknown, such as when the data comes from experiments, or from simulations with commercial software. The SCF also allows the choice of governing equations and boundary conditions for the constrained ROM which are different from those used to generate the reference data. Furthermore, the SCF makes the treatment of nonlinear partial differential equations and nonlinear goal functionals straightforward. For the implementation of the SCF, this thesis considers two kinds of structure for the primary basis functions: **P-type primary basis functions** and **F-type primary basis functions**. They are used in applications of the SCF to one-dimensional problems and the Stokes problems, respectively.

We applied the SCF to one-dimensional problems, and the results confirm that the primary basis functions obtained with the SCF are superior to POD modes. In particular, the improvements are substantial when the goal functional is a local function in space. For the nonlinear example, i.e. the application of the SCF to the Burgers equation, we have observed that the improvements become significant as the solution becomes less smooth. In applications of the SCF to the linear advection-diffusion equation and the Burgers equation, we have observed that the choice of goal functional can influence the number of extrema appearing in the optimization problem. The use of a weighted global goal functional rather than a purely local goal functional is suggested to avoid ambiguities. Results from the application of the SCF using an approximate partial diffusion equation show that an optimal primary basis can partially compensate for the model error of an approximate partial diffusion equation.

In the application of the SCF to two-dimensional Stokes equations, we take into account the pressure when constructing the ROM. We describe two options for the structure of the primary basis. The first is a scalar-valued basis where the primary basis functions for the three-directional velocities and pressure are independent. The second is a vector-valued basis where the primary basis functions for the velocities and pressure are seen as a whole. Similar choices are made for the secondary basis used in **F-type primary bases**. We argue that the second structure is superior for both bases because the size of the optimization problem can be reduced. We then provide numerical results for the cases where both primary and secondary basis are a vector-valued basis. The procedure is found to be robust and insensitive to the inclusion of superfluous secondary basis functions. It is noted, however, that if symmetries are present in the reference data, then we should choose appropriate secondary basis functions in order to minimize the size of the optimization problem. In general, the results demonstrate the viability of the SCF for seeking an optimal primary basis for a specific goal functional of interest in two-

dimensional, multi-variable problems.

5.2. OUTLOOK

In the optimization code used for the application of the SCF to the 2D Stokes equations, the integration of the primary and secondary basis functions over the space domain of interest is expensive and is one of the largest factors in the computational cost. Consequently, our group is developing an approach where spatial integrations of the secondary basis functions are pre-computed. Thus, the linear terms in the **State Equations**, **Adjoint Equations**, **Gradient** and the goal functional are evaluated as a matrix product of these precomputed terms. In this condition, the cost to determine an optimal primary basis for a specific goal functional then can be significantly reduced.

This thesis has applied the SCF to 2D Stokes equations. In this application, vector-valued secondary basis functions are used to construct the primary basis functions (i.e., the optimized basis functions) and vector-valued primary basis functions are used to construct the constrained ROM. In such case, we must be careful to choose the secondary basis to obtain a ROM which best represents a specific goal functional. Hence, other optional choices mentioned in Section 4.4.1, such as the combination of scalar-valued secondary basis functions and scalar-valued primary basis functions, need to be investigated.

We have seen that the SCF is an effective method that can find an optimal primary basis for a specific goal functional. Thus, the SCF can be used in problems of optimal design and optimal control. The next step then can be that applying the SCF to the three-dimensional incompressible Navier-Stokes equations.

A

DISCRETE PROPER ORTHOGONAL DECOMPOSITION

In this section, we will follow the view in [26, 70] to introduce the two different practical ways to calculate POD modes. Which method requires less computational effort relies on the number of grid points (n_x) and the number of ensemble members (N_t). Usually, ensemble members are referred to as snapshots in the reference data ensemble.

A.1. DIRECT METHOD

Here, the ensemble average is defined as a time average of N_t snapshots, i.e.

$$\langle u(x)u(x')\rangle = \frac{1}{N_t}\sum_{n=1}^{N_t} u^n(x)u^n(x'), \tag{A.1}$$

where $u^n(x) = u(x,t_n)$. Interchanging the sum and integral, we can rewrite the eigenvalue problem (1.7) as

$$\frac{1}{N_t}\sum_{n=1}^{N_t} u^n(x)\int_{x'} u^n(x')\phi(x')dx' = \lambda\phi(x). \tag{A.2}$$

Now we can approximate the integral over x' using a quadrature method, for example, the trapezoidal rule. Then, the integral can be expressed as

$$\int_{x'} u^n(x')\phi(x')dx' = \sum_{i=1}^{n_x}\omega_i u^n(x_i)\phi(x_i) = (\hat{\boldsymbol{u}}^n)^T\hat{\boldsymbol{\phi}}, \tag{A.3}$$

where

$$\hat{\boldsymbol{u}}^n = \begin{bmatrix} \sqrt{\omega_1}u^n(x_1) \\ \sqrt{\omega_2}u^n(x_2) \\ \vdots \\ \sqrt{\omega_{n_x-1}}u^n(x_{n_x-1}) \\ \sqrt{\omega_{n_x}}u^n(x_{n_x}) \end{bmatrix}, \qquad \hat{\boldsymbol{\phi}} = \begin{bmatrix} \sqrt{\omega_1}\phi(x_1) \\ \sqrt{\omega_2}\phi(x_2) \\ \vdots \\ \sqrt{\omega_{n_x-1}}\phi(x_{n_x-1}) \\ \sqrt{\omega_{n_x}}\phi(x_{n_x}) \end{bmatrix}, \tag{A.4}$$

and ω_i are the weight functions for the chosen quadrature method. With these definitions we could write (A.2) as

$$\frac{1}{N_t}\sum_{n=1}^{N_t} u^n(x)(\hat{\boldsymbol{u}}^n)^T\hat{\boldsymbol{\phi}} = \lambda\phi(x). \tag{A.5}$$

At each of the n_x grid points x_j, the equation (A.5) is satisfied, i.e.

$$\frac{1}{N_t}\sum_{n=1}^{N_t} u^n(x_j)(\hat{\boldsymbol{u}}^n)^T\hat{\boldsymbol{\phi}} = \lambda\phi(x_j) \quad \text{for } j = 1,\cdots,n_x. \tag{A.6}$$

Multiplying (A.6) by ω_j for every $j = 1,\cdots,n_x$, we write the resulting set of equations as a single matrix-vector equation

$$\frac{1}{N_t}\sum_{n=1}^{N_t} \hat{\boldsymbol{u}}^n(\hat{\boldsymbol{u}}^n)^T\hat{\boldsymbol{\phi}} \overset{\text{def}}{=} \boldsymbol{A}\hat{\boldsymbol{\phi}} = \lambda\hat{\boldsymbol{\phi}}, \tag{A.7}$$

implying that the integral equation (1.7) becomes a symmetric eigenvalue problem for the $n_x \times n_x$ matrix $\boldsymbol{A}$. To ensure that the POD modes are orthonormal, remembering to multiply the components of the eigenfunctions for the eigenvalue problem (A.7) by $\frac{1}{\sqrt{\omega_j}}$ to get the POD modes ϕ, that is, the POD modes are

$$\phi(x_j) = \frac{1}{\sqrt{\omega_j}}\hat{\boldsymbol{\phi}}(x_j) \quad \text{for } j = 1,\cdots,n_x. \tag{A.8}$$

A.2. SNAPSHOT POD

The Snapshot POD method was proposed by Sirovich [109]. This method can change an eigenvalue problem from the size of $n_x \times n_x$ into $N_t \times N_t$. When $n_x \gg N_t$, it can save time in solving the eigenvalue problem .

Suppose that ϕ has a special form in terms of the reference data:

$$\phi(x) = \sum_{k=1}^{N_t} \alpha(t_k)u(x,t_k) \tag{A.9}$$

where the coefficients $\alpha(t_k)$ remain to be determined so that ϕ given by (A.9) provides a maximum for (1.4). The eigenvalue problem (1.7) can be written as

$$\int_{x'} \mathbf{R}(x,x')\sum_{k=1}^{N_t}\alpha(t_k)u(x',t_k)dx' = \lambda\sum_{k=1}^{N_t}\alpha(t_k)u(x,t_k), \tag{A.10}$$

where the two-point space correlation tensor $\mathbf{R}(x,x')$ is estimated by

$$\mathbf{R}(x,x') = \frac{1}{T}\int_T u(x,t)u(x',t)dt = \frac{1}{N_t}\sum_{n=1}^{N_t} u(x,t_n)u(x',t_n). \tag{A.11}$$

Substituting (A.11) into the equation (A.10) yields

$$\sum_{n=1}^{N_t}\left[\sum_{k=1}^{N_t}\frac{1}{N_t}\left(\int_{x'} u(x',t_n)u(x',t_k)dx'\right)\alpha(t_k)\right]u(x,t_n) = \lambda\sum_{k=1}^{N_t}\alpha(t_k)u(x,t_k),$$

and we conclude that a sufficient condition for the solution of (A.10) is to find coefficients $\alpha(t_k)$ such that

$$\sum_{k=1}^{N_t} \frac{1}{N_t} \left(\int_{x'} u(x', t_n) u(x', t_k) dx' \right) \alpha(t_k) = \lambda \alpha(t_n) \quad n = 1, \cdots, N_t. \tag{A.12}$$

This is now an $N_t \times N_t$ eigenvalue problem

$$C\alpha = \lambda \alpha, \tag{A.13}$$

where

$$C_{nk} = \frac{1}{N_t} \int_{x'} u(x', t_n) u(x', t_k) dx' \quad \text{and} \quad \alpha = \begin{bmatrix} \alpha(t_1) \\ \alpha(t_2) \\ \vdots \\ \alpha(t_{N_t}) \end{bmatrix}. \tag{A.14}$$

Note that in order for (A.12) to be a necessary condition, one needs to assume that the observations $u(x, t_k)$, $k = 1, \cdots, N_t$ are linearly independent.

It is easy to verify that if the POD modes are not evaluated through (A.9) but as

$$\phi_i(x) = \frac{1}{\sqrt{N_t \lambda_i}} \sum_{k=1}^{N_t} \alpha_i(t_k) u(x, t_k), \tag{A.15}$$

then the POD modes are orthonormal.

B

THE CONJUGATE GRADIENT METHOD

The Conjugate Gradient Method (CG) is the most popular iterative method to solve large systems of linear equations in the form

$$Ax = b, \tag{B.1}$$

where x is an unknown vector, b is a known vector, and A is a known, square, positive definite matrix ($x^T Ax > 0$ for every nonzero vector x). If A is symmetric, the problem (B.1) can also be seen equivalently as the following minimization problem

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

where c is a scalar constant. The solution of (B.2) is given by

$$f'(x) = \frac{1}{2}\left(A^T + A\right)x - b = Ax - b = 0 \;\Leftrightarrow Ax = b, \tag{B.3}$$

since A is symmetric. This allows us to see the CG method either as an algorithm to solve linear systems or as a technique to minimize convex quadratic functions. It is more easily understood in terms of minimization, we pose the problem as the determination of the minima of the quadratic function $f(x)$, (B.2). Before illustrating the CG method, we introduce some notations

1. The new point: $x_{k+1} = x_k + \alpha_k d_k$ with α_k the step length and d_k the search direction for the current iterate.
2. Error: $e_k = x_k - x$, a vector that indicates how far we are from the solution x.
3. Residual: $r_k = b - Ax_k$, which describes how well the current iterate approximates the solution. Note that the residual can also be written as $r_k = -f'(x_k) = -Ae_k$.

Further, two vectors d_i and d_j are *A-orthogonal* or *conjugate*, if

$$d_i^T Ad_j = 0 \quad \text{for } i \neq j. \tag{B.4}$$

B.1. THE METHOD OF CONJUGATE DIRECTIONS

Methods such as the method of Steepest Decent often take steps in the same direction as earlier iterates and can be slow to converge [110]. The idea behind the method of conjugate directions (and conjugate gradient) is to take a set of *A-orthogonal* search directions $\{d_0, d_1, \cdots, d_{n-1}\}$, and to take one step in each direction with just the right step length such that after n steps we line up with the solution x. In order to achieve this, the error e_{k+1} must be *A-orthogonal* to the previous search direction d_k

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

which is equivalent to finding a minimum of $f(x)$ along the current search direction d_k. To see this, setting the directional derivative of $f(x_k + 1)$ with respect to α_k to zero:

$$\begin{aligned} \frac{d}{d\alpha_k} f(x_{k+1}) = 0 \Rightarrow f'^T(x_{k+1}) \frac{d}{d\alpha_k} x_{k+1} = 0 \Rightarrow -r_{k+1}^T d_k = 0 \\ \Rightarrow d_k^T A e_{k+1} = 0 \end{aligned} \tag{B.6}$$

Using (B.5) we can calculate the required step length α_k

$$\begin{aligned} d_k^T A e_{k+1} = 0 \Leftrightarrow d_k^T A (e_k + \alpha_k d_k) = 0 \\ \Rightarrow \alpha_k = -\frac{d_k^T A e_k}{d_k^T A d_k} = \frac{d_k^T r_k}{d_k^T A d_k} \end{aligned} \tag{B.7}$$

where the relation $e_{k+1} = e_k + \alpha_k d_k$ was found by subtracting e_{k+1} from e_k. To prove that this procedure really does compute the solution x in n steps, express the initial error e_0 as a linear combination of search directions (Fig.B.1), i.e.

$$e_0 = \sum_{j=0}^{n-1} \delta_j d_j. \tag{B.8}$$

In view of (B.4), we can eliminate all the δ_j values but one from (B.8) by premultiplying this expression with $d_k^T A$

$$\begin{aligned} d_k^T A e_0 = \sum_{j=0}^{n-1} \delta_j d_k^T A d_j \overset{(B.4)}{\Longrightarrow} d_k^T A e_0 = \delta_k d_k^T A d_k \\ \Rightarrow \delta_k = \frac{d_k^T A e_0}{d_k^T A d_k} \overset{(B.4)}{=\!=} \frac{d_k^T A \left(e_0 + \sum_{i=0}^{k-1} \alpha_i d_i\right)}{d_k^T A d_k} \\ \Rightarrow \delta_k = \frac{d_k^T A e_k}{d_k^T A d_k}. \end{aligned} \tag{B.9}$$

Comparing (B.9) with (B.7), we see that $\delta_k = -\alpha_k$, so

$$\begin{aligned} e_i = e_0 + \sum_{j=0}^{i-1} \alpha_j d_j = \sum_{j=0}^{n-1} \delta_j d_j - \sum_{j=0}^{i-1} \delta_j d_j \\ = \sum_{j=i}^{n-1} \delta_j d_j. \end{aligned} \tag{B.10}$$

That is, after n steps every component of the error has been cut away and $e_n = 0$, which proves the effectiveness of this procedure.

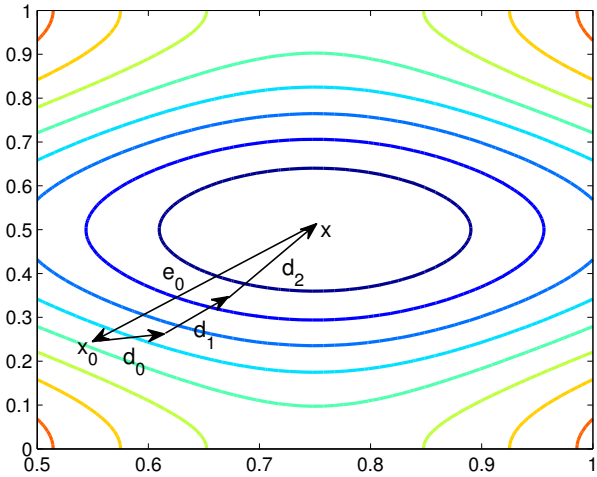


Figure B.1: The initial error e_0 can be expressed as a sum of *A-orthogonal* components (d_0, d_1 and d_2)

All that is now needed is a set of *A-orthogonal* search directions, which we can construct with the conjugate Gram-Schmidt process from a set of linearly independent vectors $\{u_0, \cdots, u_{n-1}\}$. To construct d_i, take u_i and subtract out any components that are not *A-orthogonal* to the previous search directions $d_0, d_1, \cdots, d_{i-1}$. The procedure can be illustrated by Fig.B.2, given two linearly independent vectors u_0 and u_1, set $d_0 = u_0$, the vector u_1 is composed of two components: u^* that is *A-orthogonal* to d_0, u^+ that is parallel (not *A-orthogonal*) to d_0, then $d_1 = u_1 - u^+ = u_1 + d_0 = u^*$ which only includes the *A-orthogonal* portion. Generally, for $i > 0$, the i^{th} search direction is given by

$$d_i = u_i + \sum_{k=0}^{i-1} \beta_{ik} d_k, \tag{B.11}$$

where β_{ik} are defined for $i > k$, and we always set $d_0 = u_0$. To find the values of β_{ik}, we take the transpose of (B.11) and multiply by Ad_j $(i > j)$

$$\begin{aligned} d_i^T A d_j &= u_i^T A d_j + \sum_{k=0}^{i-1} \beta_{ik} d_k^T A d_j \\ &\xrightarrow{(B.4)} 0 = u_i^T A d_j + \beta_{ij} d_j^T A d_j, \\ &\Longrightarrow \beta_{ij} = -\frac{u_i^T A d_j}{d_j^T A d_j}. \end{aligned} \tag{B.12}$$

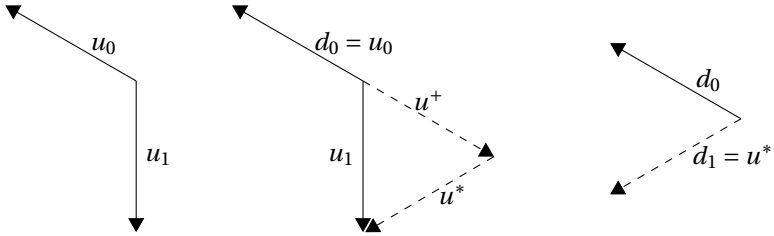


Figure B.2: Gram-Schmidt conjugation of two vectors

Next, some important properties of this method will be presented.

1. The new residual r_i is orthogonal to all previous search directions d_k $(i > k)$. To show this, premultiply (B.10) by $-d_k^T A$,

$$-d_k^T A e_i = -\sum_{j=i}^{n-1} \delta_j d_k^T A d_j$$

$$\xrightarrow[(B.4)]{k<i\leq j} d_k^T r_i = 0. \tag{B.13}$$

2. The new residual r_j is also orthogonal to all previous u_i $(j > i)$. This is proven by taking the inner product of (B.11) and r_j,

$$d_i^T r_j = u_i^T r_j + \sum_{k=0}^{i-1} \beta_{ik} d_k^T r_j \tag{B.14}$$

$$\xrightarrow[(B.13)]{k<i<j} u_i^T r_j = 0. \tag{B.15}$$

There is one more relation between the current residual r_i and search direction d_i. In (B.14), setting $i = j$, since $d_k^T r_j = 0$ when $k < j$, then we get

$$d_i^T r_i = u_i^T r_i. \tag{B.16}$$

B.2. CONJUGATE GRADIENT METHOD

The Conjugate Gradient Method can be seen as the method of Conjugate Directions where the search directions are constructed by conjugation of the residuals r_i, that is, setting $u_i = r_i$. Therefore, according to (B.11), we can calculate the search directions through

$$d_i = r_i + \sum_{k=1}^{i-1} \beta_{ik} d_k. \tag{B.17}$$

And (B.15) becomes

$$r_i^T r_j = 0 \quad i < j \tag{B.18}$$

which means that the new residual is orthogonal to all previous residuals, and it is also orthogonal to all previous search directions (B.13).

In the method of conjugate directions the Gram-Schmidt constants β_{ij} was given by

$$\beta_{ij} = -\frac{u_i^T A d_j}{d_j^T A d_j},$$

and now that we use the residual r_i instead of u_i, it becomes

$$\beta_{ij} = -\frac{r_i^T A d_j}{d_j^T A d_j}.$$

A big advantage of the CG method over the method of conjugate directions is that it is no longer necessary to store old search directions to ensure the *A-orthogonality* of the new search directions. To achieve this, the above expression of β_{ij} should be simplified, which can be realized through relations existing among the residuals. Firstly, the residual r_{j+1} is given by

$$r_{j+1} = -Ae_{j+1} = -A(e_j + \alpha_j d_j) = r_j - \alpha_j A d_j. \tag{B.19}$$

Then, taking the inner product of (B.19) and r_i

$$\begin{aligned}
& r_i^T r_{j+1} = r_i^T r_j - \alpha_j r_i^T A d_j \\
\Rightarrow\ & r_i^T A d_j = \frac{1}{\alpha_j}\left(r_i^T r_j - r_i^T r_{j+1}\right) \\
\Rightarrow\ & r_i^T A d_j = \begin{cases} \frac{1}{\alpha_i} r_i^T r_i & j = i \\ -\frac{1}{\alpha_{i-1}} r_i^T r_i & j = i-1 \\ 0 & \text{otherwise} \end{cases} \qquad \text{By (B.18)}
\end{aligned}$$

Lastly, β_{ij} is calculated as

$$\beta_{ij} = \begin{cases} \frac{r_i^T r_i}{\alpha_{i-1} d_{i-1}^T A d_{i-1}} & j = i-1 \\ 0 & j < i-1 \end{cases} \tag{B.20}$$

As we expect that most of the β_{ij} terms have disappeared except one. Using $\beta_i = \beta_{i,i-1}$ and $\alpha_{i-1} = \frac{d_{i-1}^T r_{i-1}}{d_{i-1}^T A d_{i-1}}$, we get

$$\beta_i = \frac{r_i^T r_i}{d_{i-1}^T r_{i-1}}. \tag{B.21}$$

We can simplify this further by using relation (B.16) and noting that u_i is now replaced by r_i

$$\beta_i = \frac{r_i^T r_i}{r_{i-1}^T r_{i-1}}. \tag{B.22}$$

Now, let's put it all together into one piece. The Conjugate Gradient Algorithm is

Conjugate Gradient Algorithm

Given x_0
Set initial residual and search direction: $r_0 = b - Ax_0$, $d_0 = r_0$
$i = 0$
while $\|r_i\| > \epsilon$ **do**
$\alpha_i = \frac{r_i^T r_i}{d_i^T A d_i}$
$x_{i+1} = x_i + \alpha_i d_i$
$r_{i+1} = r_i - \alpha_i A d_i$
$\beta_{i+1} = \frac{r_{i+1}^T r_{i+1}}{r_i^T r_i}$
$d_{i+1} = r_{i+1} + \beta_{i+1} d_i$
i = i+1
end while

BIBLIOGRAPHY

[1] David J. Lucia, Philip S. Beran, and Walter A. Silva. Reduced-order modeling: new approaches for computational physics. *Progress in Aerospace Sciences*, 40(1–2):51 – 117, 2004.

[2] Peter Benner, Volker Mehrmann, and Danny C. Sorensen. *Dimension reduction of large-scale systems*, volume 45 of *Lecture Notes in Computational Science and Engineering*. Springer-Verlag Berlin Heidelberg, 2005.

[3] A. C. Antoulas. An overview of approximation methods for large-scale dynamical systems. *Annual Reviews in Control*, 29(2):181 – 190, 2005.

[4] Wilhelmus H. A. Schilders, Henk A. van der Vorst, and Joost Rommes. *Model Order Reduction: Theory, Research Aspects and Applications*, volume 13 of *Mathematics in Industry*. Springer Berlin Heidelberg, 2008.

[5] Alfio Quarteroni and Gianluigi Rozza. *Reduced Order Methods for Modeling and Computational Reduction*, volume 9 of *MS&A - Modeling, Simulation and Applications*. Springer International Publishing, 2014.

[6] Ulrike Baur, Peter Benner, and Lihong Feng. Model order reduction for linear and nonlinear systems: A system-theoretic perspective. *Archives of Computational Methods in Engineering*, 21(4):331–358, 2014.

[7] Peter Benner, Serkan Gugercin, and Karen Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM Review*, 57(4):483–531, 2015.

[8] G. Berkooz, P. Holmes, and J. L. Lumley. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual Review of Fluid Mechanics*, 25(1):539–575, 1993.

[9] Y. C. Liang, H. P. Lee, S. P. Lim, W. Z. Lin, K. H. Lee, and C. G. Wu. Proper orthogonal decomposition and its applications—Part I: Theory. *Journal of Sound and Vibration*, 252(3):527 – 544, 2002.

[10] C. G. Wu, Y. C. Liang, W. Z. Lin, H. P. Lee, and S. P. Lim. A note on equivalence of proper orthogonal decomposition methods. *Journal of Sound and Vibration*, 265(5):1103 – 1110, 2003.

[11] Stefan Volkwein. Model reduction using proper orthogonal decomposition. *Lecture Notes, Institute of Mathematics and Scientific Computing, University of Graz. see http://www. uni-graz. at/imawww/volkwein/POD. pdf*, 2011.

[12] J. L. Lumley. The Structure of Inhomogeneous Turbulent Flows. In A. M. Yaglom and V. I. Tatarski, editors, *Atmospheric turbulence and radio propagation*, pages 166–178. Nauka, Moscow, 1967.

[13] Henry P. Bakewell Jr. and John L. Lumley. Viscous sublayer and adjacent wall region in turbulent pipe flow. *The Physics of Fluids*, 10(9):1880–1889, 1967.

[14] Parviz Moin and Robert D. Moser. Characteristic-eddy decomposition of turbulence in a channel. *Journal of Fluid Mechanics*, 200:471–509, 1989.

[15] L. Sirovich, K. S. Ball, and L. R. Keefe. Plane waves and structures in turbulent channel flow. *Physics of Fluids A: Fluid Dynamics*, 2(12):2217–2226, 1990.

[16] L. Sirovich, K. S. Ball, and R. A. Handler. Propagating structures in wall-bounded turbulent flows. *Theoretical and Computational Fluid Dynamics*, 2(5):307–317, 1991.

[17] K. S. Ball, L. Sirovich, and L. R. Keefe. Dynamical eigenfunction decomposition of turbulent channel flow. *International Journal for Numerical Methods in Fluids*, 12(6):585–604, 1991.

[18] Dietmar Rempfer and Hermann F. Fasel. Evolution of three-dimensional coherent structures in a flat-plate boundary layer. *Journal of Fluid Mechanics*, 260:351–375, 1994.

[19] M. Sen, Kiran Bhaganagar, and V. Juttijudata. Application of proper orthogonal decomposition (POD) to investigate a turbulent boundary layer in a channel with rough walls. *Journal of Turbulence*, 8(41), 2007.

[20] Mark N. Glauser, Stewart J. Leib, and William K. George. Coherent structures in the axisymmetric turbulent jet mixing layer. In Franz Durst, Brian E. Launder, John L. Lumley, Frank W. Schmidt, and James H. Whitelaw, editors, *Turbulent Shear Flows 5: Selected Papers from the Fifth International Symposium on Turbulent Shear Flows, Cornell University, Ithaca, New York, USA, August 7–9, 1985*, pages 134–145. Springer Berlin Heidelberg, 1987.

[21] M. Kirby, J. Boris, and L. Sirovich. An eigenfunction analysis of axisymmetric jet flow. *Journal of Computational Physics*, 90(1):98 – 122, 1990.

[22] S. V. Gordeyev and F. O. Thomas. Coherent structure in the turbulent planar jet. Part 1. Extraction of proper orthogonal decomposition eigenmodes and their self-similarity. *Journal of Fluid Mechanics*, 414:145–194, 2000.

[23] Philip Holmes, John L. Lumley, and Gal Berkooz. *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*. Cambridge Monogr. Mech. Cambridge University Press, 1996.

[24] Nadine Aubry, Philip Holmes, John L. Lumley, and Emily Stone. The dynamics of coherent structures in the wall region of a turbulent boundary layer. *Journal of Fluid Mechanics*, 192:115–173, 7 1988.

[25] Philip J. Holmes, John L. Lumley, Gal Berkooz, Jonathan C. Mattingly, and Ralf W. Wittenberg. Low-dimensional models of coherent structures in turbulence. *Physics Reports*, 287(4):337 – 384, 1997.

[26] Troy R. Smith, Jeff Moehlis, and Philip Holmes. Low-Dimensional Modelling of Turbulence Using the Proper Orthogonal Decomposition: A Tutorial. *Nonlinear Dynamics*, 41:275–307, 2005.

[27] D. M. Luchtenburg, B. R. Noack, and M. Schlegel. An introduction to the POD Galerkin method for fluid flows with analytical examples and MATLAB source codes. Technical report, Berlin Institute of Technology, 2009.

[28] K. Kunisch and S. Volkwein. Control of the Burgers Equation by a Reduced-Order Approach Using Proper Orthogonal Decomposition. *Journal of Optimization Theory and Applications*, 102:345–371, 1999.

[29] S. S. Ravindran. A reduced-order approach for optimal control of fluids using proper orthogonal decomposition. *International Journal for Numerical Methods in Fluids*, 34(5):425–448, 2000.

[30] Marco Fahl. *Trust-region Methods for Flow Control based on Reduced Order Modelling*. PhD thesis, Universität Trier, 2000.

[31] Karl Kunisch and Stefan Volkwein. Proper orthogonal decomposition for optimality systems. *ESAIM: M2AN*, 42(1):1–23, 2008.

[32] Bernd R. Noack, Marek Morzynski, and Gilead Tadmor. *Reduced-order modelling for flow control*, volume 528. Springer Science & Business Media, 2011.

[33] Michel Bergmann, Laurent Cordier, and Jean-Pierre Brancher. Optimal rotary control of the cylinder wake using proper orthogonal decomposition reduced-order model. *Physics of Fluids*, 17(9):097101, 2005.

[34] Michel Bergmann, Laurent Cordier, and Jean-Pierre Brancher. Drag Minimization of the Cylinder Wake by Trust-Region Proper Orthogonal Decomposition. *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, 95:309–324, 2007.

[35] Matthew J. Zahr, David Amsallem, and Charbel Farhat. Construction of parametrically-robust CFD-based reduced-order models for PDE-constrained optimization. In *21st AIAA Computational Fluid Dynamics Conference*, page 2845, San Diego, 2013.

[36] Matthew J. Zahr and Charbel Farhat. Progressive construction of a parametric reduced-order model for pde-constrained optimization. *International Journal for Numerical Methods in Engineering*, 102(5):1111–1135, 2015.

[37] Gaetan Kerschen, Jean-Claude Golinval, Alexander F. Vakakis, and Lawrence A. Bergman. The Method of Proper Orthogonal Decomposition for Dynamical Characterization and Order Reduction of Mechanical Systems: An Overview. *Nonlinear Dynamics*, 41(1):147–169, 2005.

[38] A. Placzek, D.-M. Tran, and R. Ohayon. Hybrid proper orthogonal decomposition formulation for linear structural dynamics. *Journal of Sound and Vibration*, 318(4–5):943 – 964, 2008.

[39] Francesco Ballarin and Gianluigi Rozza. POD–Galerkin monolithic reduced order models for parametrized fluid-structure interaction problems. *International Journal for Numerical Methods in Fluids*, 82(12):1010–1034, 2016.

[40] C. W. Rowley. Model reduction for fluids, using balanced proper orthogonal decomposition. *International Journal of Bifurcation and Chaos*, 15(03):997–1013, 2005.

[41] Bruce C. Moore. Principal Component Analysis in Linear Systems: Controllability, Observability, and Model Reduction. *IEEE Transactions on Automatic Control*, 26(1):17–32, February 1981.

[42] Sanjay Lall and Jerrold E. Marsden. Empirical model reduction of controlled nonlinear systems. In *Proceedings of the IFAC World Congress*, pages 473–478, 1999.

[43] Sanjay Lall, Jerrold E. Marsden, and Sonja Glavaški. A subspace approach to balanced truncation for model reduction of nonlinear control systems. *International Journal of Robust and Nonlinear Control*, 12(6):519–535, 2002.

[44] M. Ilak and C. W. Rowley. Reduced-Order Modeling of Channel Flow Using Traveling POD and Balanced POD. In *3rd AIAA Flow Control Conference*, page 3194, 2006.

[45] Miloš Ilak and Clarence W. Rowley. Modeling of Transitional Channel Flow Using Balanced Proper Orthogonal Decomposition. *Physics of Fluids*, 20(3):034103, 2008.

[46] Shervin Bagheri, Luca Brandt, and Dan S. Henningson. Input-output analysis, model reduction and control of the flat-plate boundary layer. *Journal of Fluid Mechanics*, 620:263–298, 2009.

[47] Alexandre Barbagallo, Denis Sipp, and Peter J. Schmid. Closed-loop control of an open cavity flow using reduced-order models. *Journal of Fluid Mechanics*, 641:1–50, 2009.

[48] Sunil Ahuja and Clarence W. Rowley. Feedback control of unstable steady states of flow past a flat plate using reduced-order estimators. *Journal of fluid mechanics*, 645:447–478, 2010.

[49] Onofrio Semeraro, Shervin Bagheri, Luca Brandt, and Dan S. Henningson. Feedback control of three-dimensional optimal disturbances using reduced-order models. *Journal of Fluid Mechanics*, 677:63–102, 2011.

[50] Peter Schmid and Joern Sesterhenn. Dynamic mode decomposition of numerical and experimental data. In *Sixty-First Annual Meeting of the APS Division of Fluid Dynamics*, San Antonio, 2008.

[51] Peter J. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28, 2010.

[52] Igor Mezić. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics*, 41(1):309–325, 2005.

[53] Clarence W. Rowley, Igor Mezić, Shervin Bagheri, Philipp Schlatter, and Dan S. Henningson. Spectral analysis of nonlinear flows. *Journal of Fluid Mechanics*, 641:115–127, 2009.

[54] Santosh Tirunagari, Ville Vuorinen, Ossi Kaario, and Martti Larmi. Analysis of Proper Orthogonal Decomposition and Dynamic Mode Decomposition on LES of Subsonic Jets. *CSI Journal of Computing*, 1(3):20–26, 2012.

[55] Kevin K. Chen, Jonathan H. Tu, and Clarence W. Rowley. Variants of dynamic mode decomposition: boundary condition, Koopman, and Fourier analyses. *Journal of Nonlinear Science*, 22(6):887–915, 2012.

[56] A. Wynn, D. S. Pearson, B. Ganapathisubramani, and P. J. Goulart. Optimal mode decomposition for unsteady flows. *Journal of Fluid Mechanics*, 733:473–503, 2013.

[57] Matthew O. Williams, Ioannis G. Kevrekidis, and Clarence W. Rowley. A Data–Driven Approximation of the Koopman Operator: Extending Dynamic Mode Decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, 2015.

[58] P. J. Schmid, K. E. Meyer, and O. Pust. Dynamic Mode Decomposition and Proper Orthogonal Decomposition of flow in a lid-driven cylindrical cavity. In *8th international symposium on particle image velocimetry-PIV09*, number 3, pages 1–4, 2009.

[59] Abu Seena and Hyung Jin Sung. Dynamic mode decomposition of turbulent cavity flows for self-sustained oscillations. *International Journal of Heat and Fluid Flow*, 32(6):1098 – 1110, 2011.

[60] P. J. Schmid, L. Li, M. P. Juniper, and O. Pust. Applications of the dynamic mode decomposition. *Theoretical and Computational Fluid Dynamics*, 25(1):249–259, 2011.

[61] Peter J. Schmid. Application of the dynamic mode decomposition to experimental data. *Experiments in Fluids*, 50(4):1123–1130, 2011.

[62] Peter J. Schmid, Daniele Violato, and Fulvio Scarano. Decomposition of time-resolved tomographic PIV. *Experiments in Fluids*, 52(6):1567–1579, 2012.

[63] Onofrio Semeraro, Gabriele Bellani, and Fredrik Lundell. Analysis of time-resolved PIV measurements of a confined turbulent jet using POD and Koopman modes. *Experiments in Fluids*, 53(5):1203–1220, 2012.

[64] Qingshan Zhang, Yingzheng Liu, and Shaofei Wang. The identification of coherent structures using proper orthogonal decomposition and dynamic mode decomposition. *Journal of Fluids and Structures*, 49:53 – 72, 2014.

[65] Joshua L. Proctor, Steven L. Brunton, and J. Nathan Kutz. Dynamic mode decomposition with control. *arXiv preprint arXiv:1409.6358*, 2014.

[66] Joshua L. Proctor, Steven L. Brunton, and J. Nathan Kutz. Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems*, 15(1):142–161, 2016.

[67] Jennifer Annoni, Joseph Nichols, and Peter Seiler. Wind Farm Modeling and Control Using Dynamic Mode Decomposition. In *34th Wind Energy Symposium*, page 2201, 2016.

[68] Jennifer Annoni, Pieter Gebraad, and Peter Seiler. Wind Farm Flow Modeling Using an Input-Output Reduced-Order Model. In *American Control Conference (ACC), 2016*, pages 506–512. IEEE, 2016.

[69] T. Bui-Thanh, K. Willcox, O. Ghattas, and B. van Bloemen Waanders. Goal-oriented, model-constrained optimization for reduction of large-scale systems. *J. Comput. Phys.*, 224(2):880–896, June 2007.

[70] L. Cordier and M. Bergmann. Proper Orthogonal Decomposition: An Overview. In *Lecture series 2002-04, 2003-03 and 2008-01 on post-processing of experimental and numerical data, Von Karman Institute for Fluid Dynamics, 2008*, page 46 pages. VKI, 2008.

[71] T. Bui-Thanh and K. Willcox. Model Reduction for Large-Scale CFD Applications Using the Balanced Proper Orthogonal Decomposition. In *17th AIAA Computational Fluid Dynamics Conference*, page 4617, 2005.

[72] Alan J. Laub, Michael T. Heath, Chris C. Paige, and Robert C. Ward. Computation of System Balancing Transformations and Other Applications of Simultaneous Diagonalization Algorithms. *IEEE Transactions on Automatic Control*, 32(2):115–122, 1987.

[73] D. F. Enns. Model reduction with balanced realizations: An error bound and a frequency weighted generalization. In *The 23rd IEEE Conference on Decision and Control*, pages 127–132, Dec 1984.

[74] Axel Ruhe. Rational Krylov Sequence Methods for Eigenvalue Computation. *Linear Algebra and its Applications*, 58:391 – 405, 1984.

[75] Jonathan H. Tu, Clarence W. Rowley, Dirk M. Luchtenburg, Steven L. Brunton, and J. Nathan Kutz. On dynamic mode decomposition: Theory and applications. *Journal of Computational Dynamics*, 1(2):391–421, 2014.

[76] M. Bergmann, C.-H. Bruneau, and A. Iollo. Enablers for robust POD models. *Journal of Computational Physics*, 228(2):516 – 538, 2009.

[77] Y. Bazilevs and T. J. R. Hughes. Weak imposition of Dirichlet boundary conditions in fluid mechanics. *Computers & Fluids*, 36(1):12–26, 2007.

[78] Stefano Mattei. A semi-continuous approach to reduced-order modelling. Master's thesis, Delft University of Technology, 2010.

[79] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer-Verlag New York, 2nd edition, 2006.

[80] J. Dennis and R. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Society for Industrial and Applied Mathematics, 1996.

[81] C. Kelley. *Iterative Methods for Optimization*. Society for Industrial and Applied Mathematics, 1999.

[82] Nicholas I. M. Gould, Stefano Lucidi, Massimo Roma, and Philippe L. Toint. Solving the Trust-Region Subproblem using the Lanczos Method. *SIAM Journal on Optimization*, 9(2):504–525, 1999.

[83] Philippe L. Toint. Non-monotone trust-region algorithms for nonlinear optimization subject to convex constraints. *Mathematical Programming*, 77(3):69–94, 1997.

[84] Stanley C. Eisenstat and Homer F. Walker. Choosing the Forcing Terms in an Inexact Newton Method. *SIAM Journal on Scientific Computing*, 17(1):16–32, 1996.

[85] Ron S. Dembo, Stanley C. Eisenstat, and Trond Steihaug. Inexact Newton Methods. *SIAM Journal on Numerical Analysis*, 19(2):400–408, 1982.

[86] Trond Steihaug. The Conjugate Gradient Method and Trust Regions in Large Scale Optimization. *SIAM Journal on Numerical Analysis*, 20(3):626–637, 1983.

[87] Lei Cheng, Stefano Mattei, Peter W. Fick, and Steve J. Hulshoff. A semi-continuous formulation for goal-oriented reduced-order models: 1D problems. *International Journal for Numerical Methods in Engineering*, 105(6):464–480, 2016. nme.4990.

[88] Thomas J. R. Hughes. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. Dover Publications, 2000.

[89] George Em Karniadakis and Spencer Sherwin. *Spectral/hp Element Methods for Computational Fluid Dynamics*. Oxford University Press, 1999.

[90] W. N. Edeling. Improved representations of unresolved scales using optimization techniques. Master's thesis, Delft University of Technology, 2011.

[91] Guglielmo Scovazzi. *Multiscale methods in science and engineering*. PhD thesis, Stanford university, 2004.

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

[93] Mojtaba Rajaee, Sture K. F. Karlsson, and Lawrence Sirovich. Low-dimensional description of free-shear-flow coherent structures and their dynamical behaviour. *Journal of Fluid Mechanics*, 258:1–29, 1994.

[94] J. Moehlis, T. R. Smith, P. Holmes, and H. Faisst. Models for turbulent plane Couette flow using the proper orthogonal decomposition. *Physics of Fluids*, 14(7):2493–2507, 2002.

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

[96] Bernd R. Noack, Paul Papas, and Peter A. Monkewitz. The need for a pressure-term representation in empirical Galerkin models of incompressible shear flows. *Journal of Fluid Mechanics*, 523:339–365, 2005.

[97] B. J. O'Donnell and B. T. Helenbrook. Proper orthogonal decomposition and incompressible flow: An application to particle modeling. *Computers & Fluids*, 36(7):1174 – 1186, 2007.

[98] Fariduddin Behzad. *Proper Orthogonal Decomposition Based Reduced Order Modeling for Fluid Flow*. PhD thesis, Clarkson University, 2014.

[99] Clarence W. Rowley, Tim Colonius, and Richard Murray. Dynamical models for control of cavity oscillations. In *7th AIAA/CEAS Aeroacoustics Conference and Exhibit*, 2001.

[100] Clarence W. Rowley. *Modeling, Simulation, and Control of Cavity Flow Oscillations*. PhD thesis, California Institute of Technology, 2002.

[101] Clarence W. Rowley, Tim Colonius, and Richard M. Murray. Model reduction for compressible flows using POD and Galerkin projection. *Physica D: Nonlinear Phenomena*, 189(1-2):115 – 129, 2004.

[102] Ron S. Dembo and Trond Steihaug. Truncated-Newton algorithms for large-scale unconstrained optimization. *Mathematical Programming*, 26(2):190–212, 1983.

[103] J. Chung and G. M. Hulbert. A Time Integration Algorithm for Structural Dynamics With Improved Numerical Dissipation: The Generalized-α Method. *Journal of Applied Mechanics-transactions of The Asme*, 60, 1993.

[104] Kenneth E. Jansen, Christian H. Whiting, and Gregory M. Hulbert. A generalized-α method for integrating the filtered Navier–Stokes equations with a stabilized finite element method. *Computer Methods in Applied Mechanics and Engineering*, 190(3–4):305 – 319, 2000.

[105] A. Sartenaer. Automatic Determination of an Initial Trust Region in Nonlinear Programming. *SIAM Journal on Scientific Computing*, 18(6):1788–1803, 1997.

[106] Ju-Liang Zhang and Xiang-Sun Zhang. A Nonmonotone Adaptive Trust Region Method and Its Convergence. *Computers & Mathematics with Applications*, 45(10):1469 – 1477, 2003.

[107] Zhen-Jun Shi and Jin-Hua Guo. A new trust region method for unconstrained optimization. *Journal of Computational and Applied Mathematics*, 213(2):509 – 520, 2008.

[108] Masoud Ahookhosh and Keyvan Amini. A nonmonotone trust region method with adaptive radius for unconstrained optimization problems. *Computers & Mathematics with Applications*, 60(3):411 – 422, 2010.

[109] Lawrence Sirovich. Turbulence and the dynamics of coherent structures Part I: coherent structures. *Quarterly of Applied Mathematics*, 45(3):561–571, 1987.

[110] Jonathan Richard Shewchuk. An Introduction to the Conjugate Gradient Method Without the Agonizing Pain. Technical report, 1994.

ACKNOWLEDGEMENTS

This thesis contains the outcomes of my Ph.D. research carried out in the Aerodynamic Group in the faculty of Aerospace Engineering of Delft University of Technology during the past seven years. The research is funded by Chinese Scholarship Council (CSC). I hereby express my appreciation to the CSC who has given financial supports during my doctoral study.

I would like to sincerely give the most thanks to my daily supervisor, Dr. Steven Hulshoff, for the continuous support throughout my Ph.D. study, for the patience to answer all my questions, and for the encouragements when I was pointless. I have benefited greatly from your guidance, suggestions and professionalism. Discussions with you were pleasant and always provided me lots of ideas on the topic of the research. Your wide range of knowledge and strong analytic ability help me more optimistically pursue an academic career. Besides my supervisor, appreciation will be also given to my promotor Prof. dr. ir. drs. Hester Bijl. Thanks for taking time to follow my thesis progress.

It is a great pleasure to have a Ph.D. study in the aerodynamic group. I am thankful to all the members in this group. Particular thanks go to Linfeng Chen for the constructive suggestions and fruitful discussions on my research work. Talking with you about the programming language and codes used in the research made me save much time. Thank Stefano Mattei for the help on the programming code at the beginning of my research. I also give thanks to Ilya Ipopov and Vahid Kamyab for the help on my research and spoken English.

Next, I would like to thank my friends: Xiaojia Zhao, Shuanghou Deng, Yannian Yang, Qingqing Ye, Zi Wang, Ye Zhang, Zhengzhong Sun, Weiling Zheng, Linfeng Chen, and any more, for the parties, interesting talks and the shared joy time with you during my stay in Netherlands.

Lastly, I will give the deep gratitude to all my family members. Thank my parents for their unconditional love, encouragements, and supporting all my decisions. Thank my brother for accompanying our parents when I am far away from them.

Lei Cheng
Delft, July 2017

CURRICULUM VITÆ

Lei CHENG

16-01-1985 Born in Hebei province, China.

EDUCATION

2010–2017 Ph.D. candidate in Aerodynamics Group
Faculty of Aerospace Engineering
Delft University of Technology, Delft, the Netherlands
Thesis Title: A Semi-Continuous Formulation for Goal-Oriented Reduced-Order Models
Promotor: Prof. Hester Bijl
Supervisor: Dr. Steven J. Hulshoff

2007–2010 Master degree in Fluid Mechanics
School of Aeronautics
Northwestern Polytechnical University, Xi'an, China.
Research Topic: Aerodynamic characteristic of boxwing configuration for transport aircraft
Supervisor: Prof. Binqian Zhang

2003–2007 Bachelor degree in Aircraft Design and Engineering (with honour)
School of Aeronautics
Northwestern Polytechnical University, Xi'an, China.
Thesis Title: Plane shape optimization design for high aspect ratio lambda wings