Cross-Validation of Data-Driven Correction Reduced Order Modeling

Changhong Mou

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

Master of Science
in
Mathematics

Traian Iliescu, Chair
John A. Burns
Honghu Liu

October 3, 2018
Blacksburg, Virginia

Keywords: Reduced Order Modeling, Data-Driven Modeling, Optimization, Scientific Computing

Cross-Validation of Data-Driven Correction Reduced Order Modeling

Changhong Mou

(ABSTRACT)

In this thesis, we develop a data-driven correction reduced order model (DDC-ROM) for numerical simulation of fluid flows. The general DDC-ROM involves two stages: (1) we apply ROM filtering (such as ROM projection) to the full order model (FOM) and construct the filtered ROM (F-ROM). (2) We use data-driven modeling to model the nonlinear interactions between resolved and unresolved modes, which solves the F-ROM's closure problem.

In the DDC-ROM, a linear or quadratic ansatz is used in the data-driven modeling step. In this thesis, we propose a new cubic ansatz. To get the unknown coefficients in our ansatz, we solve an optimization problem that minimizes the difference between the FOM data and the ansatz. We test the new DDC-ROM in the numerical simulation of the one-dimensional Burgers equation with a small diffusion coefficient. Furthermore, we perform a cross-validation of the DDC-ROM to investigate whether it can be successful in computational settings that are different from the training regime.

Cross-Validation of Data-Driven Correction Reduced Order Modeling

Changhong Mou

(GENERAL AUDIENCE ABSTRACT)

Practical engineering and scientific problems often require the repeated simulation of unsteady fluid flows. In these applications, the computational cost of high-fidelity full-order models can be prohibitively high. Reduced order models (ROMs) represent efficient alternatives to brute force computational approaches. In this thesis, we propose a data-driven correction ROM (DDC-ROM) in which available data and an optimization problem are used to model the nonlinear interactions between resolved and unresolved modes. In order to test the new DDC-ROM's predictability, we perform its cross-validation for the one-dimensional viscous Burgers equation and different training regimes.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we first introduce the basic ideas of reduced order models (ROMs). Then, we discuss the current achievements and challenges of ROMs. In the last section, the work in this thesis is outlined.

1.1 Motivation

In science and engineering, we are often facing challenges in modeling and control of complex systems, such as heat transfer, fluid dynamics, geophysical fluid dynamics, nuclear engineering, chemical reacting flows, biological systems, wave propagation and design of micro-electro-mechanical systems (MEMS) [6, 11, 13, 15, 16, 21, 35, 37, 39, 71, 73, 76, 82, 86, 95, 95, 100, 108]. Dynamical systems are the basic framework of these complex systems. To study the underlying dynamics of these physical phenomena, one efficient approach is to investigate the numerical simulation of the associated models. However, the current scientific and industrial needs require more accuracy, more details, and more parameters. As a result, the corresponding numerical simulation must involve larger scale and more complex numerical schemes. Simulations of many different possible realizations of the system are needed to improve their performance while multiple simulations of large scale settings inevitably require a large amount of computing resources. Reduced order models (ROMs), which are low dimensional, efficient and fast to solve, can be used to alleviate the computational burden.

In this thesis, we consider problems characterized by partial differential equations (PDEs) which feature an infinite number of degrees of freedom (DOF)[40, 96, 103].

1.2 Review of Reduced Order Models (ROMs)

During the past decades, ROM has been developed massively to address the issues mentioned in 1.1. The key idea of the ROM is to replace the original full-order model with a low dimensional one of controlled loss of accuracy while ensuring reliable evaluations of the reduced model at substantially reduced computational cost[19]. In the field of model reduction, a wide span of mathematical methods can be applied [23, 26, 87]. In this thesis, we will focus on the proper orthogonal decomposition (POD-) based reduced order modeling, which is a Galerkin approximation in the spatial variables that is based on solutions at pre-specified times, which are called *snapshots* [42, 55]. Proper orthogonal decomposition (POD) is used to construct a ROM basis from the given set of snapshots.

Proper orthogonal decomposition was first introduced by Lumley in the study of turbulence, while in other disciplines the POD procedure shares other names: Karhunen-Loéve decomposition, principal components analysis (PCA), singular system analysis, and singular value decomposition (SVD) [42, 53, 58, 61, 70, 77, 78] has been successfully applied to various disciplines besides fluid dynamics and coherent structures[42], such as signal analysis and pattern recognition[83], control theory[1, 4] and inverse problems[10].

The procedure for generating POD-based ROMs are described in brief as following steps: (i). simulating the full order models (FOMs); (ii) picking up snapshots (state variable vectors); (iii) finding an optimal basis by applying POD method to the data set and (iv). employing the Galerkin projection for the basis coefficients[17, 44, 101]. Mathematical details of POD procedure will be given in section 2.2.The resulting coefficients can be used to obtain a low-order dynamical system which is named POD Galerkin reduced-order model (POD-G-ROM) or simply G-ROM in this thesis.

1.3 Outline of this Thesis

In this thesis, we first introduce the construction of reduced order models (ROMs) (see chapter 2), then the *data-driven correction* ROM (see chapter 3) , which greatly improves the accuracy of the standard G-ROM. In chapter 4, we apply the Cross-Validation to DDC-ROM numerically. Finally, chapter 5 includes a brief summary and future research plan.

Chapter 2

Reduced Order Models

2.1 Model Description

2.1.1 Navier-Stokes Equations

Before presenting the reduced order models (ROMs), we first introduce the models we are interested in. Consider the flow of a fluid in a bounded region $\Omega \subset \mathbb{R}^2$ or $\mathbb{R}^3$. Suppose $u(x,t)$ for $x \in \bar{\Omega}, 0 \leq t \leq T$ and $p(x,t)$ for $x \in \Omega, 0 < t \leq T$ are fluid velocity and pressure functions respectively, are governed by the incompressible *Navier-Stokes equations (NSE)*[8, 27, 38, 57, 79, 97]:

$$\frac{\partial \mathbf{u}}{\partial t} - Re^{-1}\Delta \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} + \nabla p = 0, x \in \Omega, 0 < t \leq T \tag{2.1}$$

$$\nabla \cdot \mathbf{u} = 0, x \in \Omega \quad \text{with} \quad 0 < t \leq T, \mathbf{u}(x,0) = \mathbf{u}_0(x), x \in \Omega \tag{2.2}$$

$$\mathbf{u} = 0 \quad \text{on} \quad \partial\Omega, \quad \text{with} \quad \int_\Omega p dx = 0 \quad \forall t \in (0,T], \tag{2.3}$$

where Re is Reynolds number.

Suppose $\mathbf{y}(x,t)$ is the state variable and let $\mathcal{H}$ be a Hilbert space. The complex flow described by equations (2.1) to (2.3), which has an infinite number of degrees of freedom, can be abstracted into the form: find $\mathbf{y}(\cdot,t) \in \mathcal{H}$ satisfying,

$$\dot{\mathbf{y}}(x,t) = \mathbf{f}(t, \mathbf{y}(x,t)), \tag{2.4}$$

$$\mathbf{y}(x,t_0) = \mathbf{y}_0(x). \tag{2.5}$$

With appropriate numerical methods, e.g. finite difference method or finite element [92], the problem is approximated with a *finite* number of degrees of freedom: to find $\mathbf{y}(\cdot,t) \in \mathbb{R}^N$

satisfying

$$\dot{\mathbf{y}}(x,t) = \mathbf{f}(t, \mathbf{y}(x,t)), \tag{2.6}$$
$$\mathbf{y}(x,t_0) = \mathbf{y}_0(x). \tag{2.7}$$

2.1.2 Snapshots

Proposed by Sirovich [90, 91], the method of *snapshots*, which startts with a set of instantaneous flow field solutions, *snapshots*, which are either from the *Direct Numerical Simulation (DNS)* of equations (2.1) to (2.3), or experiments, and provides a fast generation of the POD basis $\{\varphi_1(x), \varphi_2(x), \cdots, \varphi_r(x)\}$[22, 32, 43, 45, 69, 98]. The most common approach to generate the snapshots is to simulate the *full order model* (FOM) of equations (2.1) to (2.3).

2.2 Proper Orthogonal Decomposition (POD)

In this section, we will briefly introduce the proper orthogonal decomposition method [34, 42, 54, 55, 90, 91, 99] . Suppose that $\mathbf{y}(x,t) \in L^2(\mathcal{H}, (t_0, t_0+T))$. Suppose that $t_1, t_2, t_3, \cdots, t_M \in [t_0, t_0+T]$ are the time instances that we picked up for the snapshots. Then,

$$\mathcal{R} := span(\mathbf{y}(\cdot,t_1), \mathbf{y}(\cdot,t_2), \cdots, \mathbf{y}(\cdot,t_M))$$

is the ensemble of snapshots with $dim(\mathcal{R}) = d$. The POD method can be described as: find a low dimensional basis $\{\varphi_1, \cdots, \varphi_r\}, r \leq d$, such that

$$\min \frac{1}{M} \sum_{l=1}^{M} \|\mathbf{y}(\cdot,t_l) - \sum_{j=1}^{r} (\mathbf{y}(\cdot,t_l), \varphi_j(\cdot))_{\mathcal{H}} \varphi_j(\cdot)\|_{\mathcal{H}}^2 \tag{2.8}$$
$$\text{subject to} \qquad (\varphi_i, \varphi_j)_{\mathcal{H}} = \delta_{ij}, \quad 1 \leq i,j \leq r \leq d, \tag{2.9}$$

where δ_{ij} is the Kronecker delta and $\|\cdot\|_{\mathcal{H}}$ is the induced norm,

$$\|f\|_{\mathcal{H}} = (f,f)_{\mathcal{H}}^{1/2} \tag{2.10}$$

This is equivalent to the eigenvalue problem,

$$Kv = \lambda v \tag{2.11}$$

where K is the snapshot correlation matrix, $K \in \mathbb{R}^{M \times M}$, and has the form

$$K_{lm} = \frac{1}{M} (\mathbf{y}(\cdot,t_l), \mathbf{y}(\cdot,t_m))_{\mathcal{H}},$$

and $\{v_j\}_{j=1}^d$ are the eigenvectors with corresponding eigenvalues $\{\lambda\}_{j=1}^d$, satisfying $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d > 0$. Hence the solution of equations (2.8) and (2.9) is given by

$$\varphi_j(\cdot) = \frac{1}{\sqrt{\lambda_j}} \sum_{l=1}^{M} (v_j)_l \mathbf{y}(\cdot, t_l), \qquad 1 \leq j \leq r, \tag{2.12}$$

where $(v_j)_l$ denotes the $l-$th component of v_j.

Proposition 2.1. *The error formula for the optimization problem equations* (2.8) *and* (2.9) *yields [7, 9, 55, 80, 107]*

$$\frac{1}{M} \sum_{l=1}^{M} \|\mathbf{y}(\cdot, t_l) - \sum_{j=1}^{r} (\mathbf{y}(\cdot, t_l), \varphi_j(\cdot))_{\mathcal{H}} \varphi_j(\cdot)\|_{\mathcal{H}}^2 = \sum_{j=r+1}^{d} \lambda_j. \tag{2.13}$$

If the *finite element method (FEM)* is used and Y is the snapshot set, then we can perform the singular value decomposition (SVD) on the matrix $\tilde{Y} = (M_h)^{\frac{1}{2}} Y$, where M_h is the mass matrix that contains the mesh and local test function information. Hence,

$$\varphi_j = (M_h)^{\frac{1}{2}} U_j,$$

where U_j is jth left singular vector of $\tilde{Y}$[102, 105].

2.3 Galerkin ROM (G-ROM)

Applying the POD method described in section 2.2, the ROM representation of the flow in equations (2.1) and (2.2), $\mathbf{u}_r \in X^r \subset X^d \subset \mathcal{H}$, can be written as,

$$\mathbf{u}_r(\mathbf{x}, t) \approx \mathbf{u}_r(\mathbf{x}) + \sum_{j=1}^{r} a_j(t) \varphi_j(x), \tag{2.14}$$

where $\mathbf{u}_r(\mathbf{x})$ is the centering trajectory, $\{\varphi_j\}_{j=1}^r \in X^r$ are the first r POD basis vectors, and $\{a_j(t)\}_{j=1}^r$ are the time varying coefficients of the POD-Galerkin trajectories. Note that the POD basis functions satisfy the following solenoidal condition:

$$\nabla \cdot \varphi_i = 0 \qquad \forall \varphi_i \in X^r. \tag{2.15}$$

This is due to the fact that the POD basis are constructed with the snapshots that are solenoidal ($\nabla \cdot \mathbf{u} = 0$).

Since the boundary conditions are homogeneous Dirichlet, the Galerkin projection of the NSE (equations (2.1) and (2.2)) onto X^r space is,

$$\left(\frac{\partial \mathbf{u}_r}{\partial t}, \varphi\right)_{\mathcal{H}} + ((\mathbf{u}_r \cdot \nabla)\mathbf{u}_r, \varphi)_{\mathcal{H}} + (\frac{1}{Re} \nabla \mathbf{u}_r, \varphi)_{\mathcal{H}} = 0 \qquad \forall \varphi \in X^r. \tag{2.16}$$

Note that in equation (2.16), the pressure term is neglected since the ROM basis functions are solenoidal[2, 68]. Furthermore, with the equation (2.14), the POD-Galerkin ROM yields the following autonomous dynamical system:

$$\dot{\mathbf{a}} = \mathbf{b} + \mathbf{A}\mathbf{a} + \mathbf{a}^T\mathbf{B}\mathbf{a} \tag{2.17}$$

where $\mathbf{a}(t)$ denotes the vector of time coefficients and $\mathbf{b}, \mathbf{A}, \mathbf{B}$ correspond to the constant, linear and quadratic terms in the discretization form of the NSE, respectively. Also the initial condition can be obtained via projection, i.e.,

$$a_i(0) = (\phi_i, \mathbf{u}(\cdot, 0) - \mathbf{U}(\cdot))_{\mathcal{H}}, \qquad i = 1, 2, \cdots, r. \tag{2.18}$$

The G-ROM equation (2.17) yields the componentwise form

$$\dot{a}_i(t) = b_i + \sum_{m=1}^{r} A_{im} a_m(t) + \sum_{m=1}^{r}\sum_{n=1}^{r} B_{imn} a_n(t) a_m(t), \tag{2.19}$$

where b_i, A_{im}, B_{imn} are given by

$$b_i = -(\mathbf{U}\cdot\nabla\mathbf{U}, \varphi_i)_{\mathcal{H}} - \frac{1}{Re}\left(\nabla\mathbf{U} + \nabla\mathbf{U}^T, \nabla\varphi_i\right)_{\mathcal{H}}, \tag{2.20}$$

$$A_{im} = -\left(\mathbf{U}\cdot\nabla\varphi_m, \varphi_i\right)_{\mathcal{H}} - \left(\varphi_m\cdot\nabla\mathbf{U}, \varphi_i\right)_{\mathcal{H}} - \frac{1}{Re}\left(\nabla\varphi_m + \nabla\varphi_m^T, \nabla\varphi_i\right)_{\mathcal{H}}, \tag{2.21}$$

$$B_{imn} = -\left(\varphi_m\cdot\nabla\varphi_n, \varphi_i\right)_{\mathcal{H}}. \tag{2.22}$$

2.4 ROM Correction Problem

In section section 2.3, we introduced the G-ROM. Although the dynamical system (equation (2.17)) is often accurate for modeling laminar flow, it will generally produce inaccurate results for turbulent flows even when the truncated Galerkin modes capture most of system's energy. As a remedy, the ROM closure problem should be taken into account. In general, closure modeling has developed in two directions: (1) the stability improvement; [12, 20, 28, 46, 89] (2) the physical accuracy improvement [5, 29, 72]. This thesis will be concerned with the later, i.e. we model the interaction between the retained modes, $\{\varphi_1, \cdots, \varphi_r\}$ and the discarded modes, $\{\varphi_{r+1}, \cdots, \varphi_d\}$ in order to get accurate results.

In [106], a modified G-ROM is used,

$$\dot{\mathbf{a}} = \mathbf{b} + \mathbf{A}\mathbf{a} + \mathbf{a}^T\mathbf{B}\mathbf{a} + \tau, \tag{2.23}$$

where the term τ accounts for the correction term. Most often, τ is modeled with a dissipation mechanism, e.g. eddy viscosity, $\tau \approx EV(\mathbf{a})$. Nevertheless, there are many schemes, including the mixing-length ROM, Smagorinsky ROM, multiscale ROM and dynamic subgrid-scale

ROM (see survey in [73, 81, 100]). In this thesis, a data-driven technique is used to determine the unknown τ , while the other operators, $\mathbf{b}, \mathbf{A}, \mathbf{B}$, are determined by the standard Galerkin method. Specifically, we employ a quadratic ansatz to model τ,

$$\tau = \tilde{A}\mathbf{a} + \mathbf{a}^T \tilde{B}\mathbf{a}, \tag{2.24}$$

where $\tilde{A}$ and $\tilde{B}$ are computed via data-driven modeling.

Chapter 3

Data-driven Correction ROMs (DDC-ROMs)

In this chapter, we present the optimization problem encountered in the computation of the quadratic, we introduce a new algorithm for the optimization problem which is more efficient in computation, and finally we present the numerical results of the DDC-ROMs. Figure 3.1 illustrates the development of the DDC-ROM.

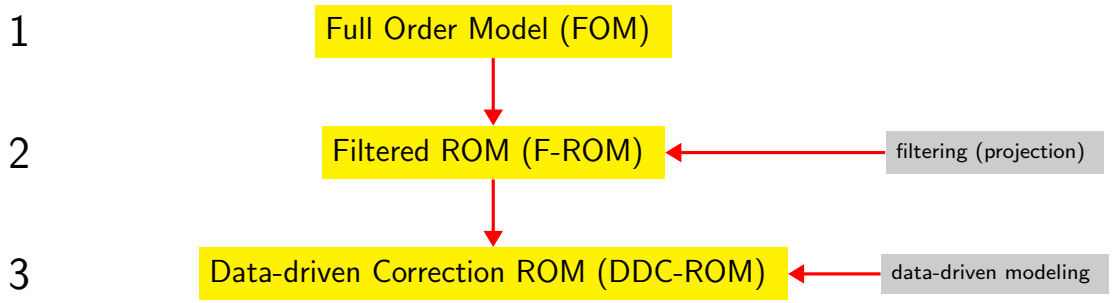


Figure 3.1: Schematic illustration of the DDC-ROM

3.1 Filtered ROM (F-ROM)

ROM Spatial Filtering

To construct the filtered ROM, spatial filtering is applied explicitly[93, 102, 105]. In this thesis, only the ROM projection filter (definition 3.1) is considered.

Definition 3.1. With fixed $r \leq d$ and known $\mathbf{u} \in X^h$, the ROM projection seeks $\overline{\mathbf{u}}^r \in X^r$,

s.t.

$$(\overline{\mathbf{u}}^r, \varphi_j)_{\mathcal{H}} = (\mathbf{u}, \varphi_j)_{\mathcal{H}} \qquad \forall j = 1, 2, \cdots, r. \tag{3.1}$$

Filtered NSE

With the assumption that the commutation error between the differential operator and the filtering is negligible [14, 48, 49], the spatially filtered NSE can be obtained via projecting the equations onto a space of weakly divergence-free function ϕ,

$$\left(\frac{\partial \overline{\mathbf{u}}}{\partial t}, \phi\right)_{\mathcal{H}} + \frac{1}{Re}\left(\nabla \overline{\mathbf{u}}, \nabla \phi\right)_{\mathcal{H}} + \left((\overline{\mathbf{u}} \cdot \nabla)\overline{\mathbf{u}}, \phi\right)_{\mathcal{H}} + \left(\tau^{SFS}, \phi\right)_{\mathcal{H}} = 0, \tag{3.2}$$

where τ^{SFS} is the *subfilter-scale stress tensor*,

$$\tau^{SFS} = \overline{(\mathbf{u} \cdot \nabla)\mathbf{u}} - (\overline{\mathbf{u}} \cdot \nabla)\overline{\mathbf{u}}. \tag{3.3}$$

Note that equation (3.2) holds for at the continuous level. To develop a ROM, first we use the best possible approximation $\mathbf{u}_d \approx \mathbf{u}$, where d is the rank of the snapshots; then we apply the projection filter described in equation (3.1) to replace the filter used at the continuous level, i.e. $\overline{\mathbf{u}}^r \approx \overline{\mathbf{u}}$. Hence, the resulting ROM is,

$$\left(\frac{\partial \overline{\mathbf{u_d}}^r}{\partial t}, \varphi_i\right) + \frac{1}{Re}\left(\nabla \overline{\mathbf{u_d}}^r, \nabla \varphi_i\right) + \left((\overline{\mathbf{u_d}}^r \cdot \nabla)\overline{\mathbf{u_d}}^r, \varphi_i\right) + \left({\tau_{\mathbf{r}}}^{SFS}, \varphi_i\right) = 0, \qquad \forall i = 1, \cdots, r, \tag{3.4}$$

where

$${\tau_{\mathbf{r}}}^{SFS} = \overline{(\mathbf{u_d} \cdot \nabla)\mathbf{u_d}}^r - (\overline{\mathbf{u_d}}^r \cdot \nabla)\overline{\mathbf{u_d}}^r. \tag{3.5}$$

Note that the $\overline{(\cdot)}^r$ means the ROM projection from X^d to X^r, thus

$$\overline{\mathbf{u_d}}^r = \mathbf{u_r}. \tag{3.6}$$

Using equation (3.6) in the equation (3.4), we obtain the following equations:

$$\left(\frac{\partial \mathbf{u_r}}{\partial t}, \varphi_i\right) + \frac{1}{Re}\left(\nabla \mathbf{u_r}, \nabla \varphi_i\right) + \left(\mathbf{u_r} \cdot \nabla)\mathbf{u_r}, \varphi_i\right) + \left({\tau_{\mathbf{r}}}^{SFS}, \varphi_i\right) = 0, \qquad \forall i = 1, \cdots, r. \tag{3.7}$$

$$\text{where} \qquad {\tau_{\mathbf{r}}}^{SFS} = \overline{(\mathbf{u_d} \cdot \nabla)\mathbf{u_d}}^r - (\mathbf{u}_r \cdot \nabla)\mathbf{u}_r \tag{3.8}$$

Using the ROM approximation of the velocity field $\mathbf{u}^r = \sum_{j=1}^{r} \mathbf{a}_j \varphi_j$, the F-ROM for the NSE is obtained:

$$\dot{\mathbf{a}} = A\mathbf{a} + \mathbf{a}^T B\mathbf{a} + \tau. \tag{3.9}$$

The component-wise form can be written as,

$$\dot{a}_i = \sum_{m=1}^{r} A_{im} a_m(t) + \sum_{m=1}^{r}\sum_{n=1}^{r} B_{imn} a_n(t) a_m(t) + \tau_i, \qquad i = 1, \cdots, r, \tag{3.10}$$

where

$$\tau_i = -\left(\tau_r^{SFS}, \varphi_i\right), \tag{3.11}$$

$$A_{im} = -\frac{1}{Re}\left(\nabla\varphi_m, \nabla\varphi_i\right) \tag{3.12}$$

$$B_{imn} = -\left(\varphi_m \cdot \nabla\varphi_n, \varphi_i\right) \tag{3.13}$$

Remark 3.2. The F-ROM is consistent with the NSE, i.e. if $r \to d$, the F-ROM will expectedly converge to the best approximation in X^d.

Remark 3.3. The F-ROM is not a closed system, because ${\tau_{\mathbf{r}}}^{SFS}$ in equation (3.7) depends on $\mathbf{u}_d$. Thus, in order to solve the closure issue, a formula $\tau = \tau(\mathbf{a})$ is needed.

DDC-ROM

The *data-driven correction* ROM (DDC-ROM) yields

$$\dot{\mathbf{a}} = A\mathbf{a} + \mathbf{a}^T B\mathbf{a} + \tau(\mathbf{a}). \tag{3.14}$$

Specifically, $\tau(\mathbf{a})$ is an approximation of τ in F-ROM (equation (3.9)) which is obtained via the *data-driven modeling*. Based on the reasoning that $\tau(\mathbf{a})$ would yield a quadratic ansatz as the G-ROM, we make the following assumption:

$$\tau(\mathbf{a}) = \tilde{\mathbf{A}}\mathbf{a} + \mathbf{a}^{\mathbf{T}}\tilde{\mathbf{B}}\mathbf{a}, \tag{3.15}$$

where $\tilde{A} \in \mathbb{R}^{r\times r}$ and $\tilde{B} \in \mathbb{R}^{r\times r\times r}$ are determined by solving an optimization problem via data-driven technique; this is discussed in detail in section 3.2. Hence, the DDC-ROM yields yields:

$$\dot{\mathbf{a}} = (A + \tilde{A})\mathbf{a} + \mathbf{a}^T(B + \tilde{B})\mathbf{a}. \tag{3.16}$$

To conclude this section, we give the algorithm of constructing the DDC-ROM.

Algorithm 3.4. DDC-ROM Generation

Given $\mathbf{y}(\cdot, t)$ from equations (2.1) and (2.2) for $t \in (t_0, t_0 + T)$

1. Compute a POD basis
$$\{\varphi_1(x), \varphi_2(x), \cdots, \varphi_r(x)\},$$
such that $X^r = span\{\varphi_1(\cdot), \varphi_2(\cdot), \cdots, \varphi_r(\cdot)\}$. The ROM approximation is written as
$$\mathbf{y}(\cdot, t) = \sum_{j=1}^{r} a_j(t)\varphi_j(\cdot) \in X^r,$$
where $\{a_j(t)\}_{j=1}^r$ are the unknown time varying coefficients.

2. Apply ROM projection filter to the full order model (FOM):
$$\dot{\mathbf{a}} = \mathbf{A}\mathbf{a} + \mathbf{a}^T\mathbf{B}\mathbf{a} + \tau(\mathbf{FOM}). \tag{3.17}$$

3. Replace the term $\tau(FOM)$ with $\tau(\mathbf{a})$, where $\tau(\mathbf{a})$ can be modeled by a quadratic ansatz. Use the dataset $\mathbf{y}(\cdot, t)$ and POD basis $\{\varphi_1(x), \varphi_2(x), \cdots, \varphi_d(x)\}$ to compute the coefficient matrix and tensor, $\tilde{A}$ and $\tilde{B}$, respectively:
$$\tau = \tilde{A}\mathbf{a} + \mathbf{a}^T\tilde{B}\mathbf{a}.$$

4. Substitute the POD approximation in step 1 into the full order system, applying the Galerkin procedure with the quadratic ansatz,
$$\dot{\mathbf{a}} = (\mathbf{A} + \tilde{\mathbf{A}})\mathbf{a} + \mathbf{a}^T(\mathbf{B} + \tilde{\mathbf{B}})\mathbf{a}, \tag{3.18}$$
which yields the dynamical system
$$\dot{a}_i(t) = \sum_{m=1}^{r}(A_{im} + \tilde{A}_{im})a_m(t) + \sum_{m=1}^{r}\sum_{n=1}^{r}(B_{imn} + \tilde{B}_{imn})a_n(t)a_m(t). \tag{3.19}$$

3.2 Optimization Problem in DDC-ROM

In section 2.4, $\tilde{A}$ and $\tilde{B}$ need to be computed prior to the construction of DDC-ROM. In fact, solving for $\tilde{A}$ and $\tilde{B}$ yields an optimization problem [104] which minimizes the L^2 norm of the difference between τ computed with the FOM data, and the τ computed with the ansatz equation (3.15) and the ROM coefficients via the snapshots, $\mathbf{a}^{snap}$. Denote the snapshots in $\mathbf{X}^d$ space,
$$\mathbf{a}^{snap} = [\mathbf{a}^{snap}(t_1), \mathbf{a}^{snap}(t_2), \cdots, \mathbf{a}^{snap}(t_M)] \in \mathbb{R}^{d\times M},$$
which are obtained via the projection of $\mathbf{u}(t_j)$ onto the POD basis function $\varphi_i, i = 1, 2, \cdots, d$,
$$\mathbf{a}_i^{snap}(t_j) = (\mathbf{u}(t_j), \varphi_i) \tag{3.20}$$

Specifically, the optimization problem for finding $\tilde{A}$ and $\tilde{B}$ is

$$\min_{\tilde{A}\in\mathbb{R}^{r\times r},\quad \tilde{B}\in\mathbb{R}^{r\times r\times r}} \sum_{j=1}^{M} \|\tau^{true}(t_j) - \tau^{ansatz}(t_j)\|^2, \tag{3.21}$$

where $\|\cdot\|$ is the Euclidean norm and $\tau^{true}(t_j) \in \mathbb{R}^r$ is computed via the snapshots data:

$$\begin{aligned}\tau_i^{true}(t_j) &= -\left(\overline{(\mathbf{u}_d^{snap}(t_j)\cdot\nabla)\mathbf{u}_d^{snap}(t_j)}^{r} - (\mathbf{u}_r^{snap}(t_j)\cdot\nabla)\mathbf{u}_r^{snap}(t_j), \varphi_i\right) \\ &= -\left(\overline{\sum_{k_1=1}^{d}\sum_{k_2=1}^{d} a_{k_1}^{snap}(t_j)a_{k_2}^{snap}(t_j)(\varphi_{k_1}\cdot\nabla)\varphi_{k_2}}^{r} - \sum_{k_3=1}^{r}\sum_{k_4=1}^{r} a_{k_3}^{snap}(t_j)a_{k_4}^{snap}(t_j)(\varphi_{k_3}\cdot\nabla)\varphi_{k_4}\right),\end{aligned} \tag{3.22}$$

where

$$\mathbf{u}_d^{snap}(t_j) = \sum_{k=1}^{d} a_k^{snap}(t_j)\varphi_k, \qquad \mathbf{u}_r^{snap}(t_j) = \sum_{k=1}^{r} a_k^{snap}(t_j)\varphi_k. \tag{3.23}$$

Hence, equation (3.15) yields

$$\tau^{ansatz}(t_j) = \tilde{A}\mathbf{a}^{snap}(t_j) + \mathbf{a}^{snap}(t_j)^T\tilde{B}\mathbf{a}^{snap}(t_j). \tag{3.24}$$

As a result, the optimization problem becomes

$$\min_{\tilde{A}\in\mathbb{R}^{r\times r},\quad \tilde{B}\in\mathbb{R}^{r\times r\times r}} \sum_{j=1}^{M} \|\tau^{true}(t_j) - \tilde{A}\mathbf{a}^{snap}(t_j) - \mathbf{a}^{snap}(t_j)^T\tilde{B}\mathbf{a}^{snap}(t_j)\|^2. \tag{3.25}$$

Solving the optimization problem

To solve the optimization problem 3.25, we first define a vector $\mathbf{X} \in \mathbb{R}^{r^2+r^3\times 1}$ that contains all the entries of $\tilde{A}$ and $\tilde{B}$ in the following structure:

$$\mathbf{X} = \begin{pmatrix} \tilde{A}_{1,1} \\ \tilde{A}_{1,2} \\ \vdots \\ \tilde{A}_{1,r} \\ \tilde{A}_{2,1} \\ \vdots \\ \tilde{A}_{2,r} \\ \vdots \\ \tilde{A}_{r,1} \\ \tilde{A}_{r,2} \\ \vdots \\ \tilde{A}_{r,r} \\ \tilde{B}_{1,1,1} \\ \tilde{B}_{1,2,1} \\ \vdots \\ \tilde{B}_{r,r,1} \\ \tilde{B}_{1,1,2} \\ \vdots \\ \tilde{B}_{r,r,2} \\ \vdots \\ \tilde{B}_{r,1,r} \\ \tilde{B}_{r,2,r} \\ \vdots \\ \tilde{B}_{r,r,r} \end{pmatrix}. \tag{3.26}$$

We also define a vector $\mathbf{f} \in \mathbb{R}^{(\mathbf{Mr})\times \mathbf{1}}$ and a matrix $E \in \mathbb{R}^{(Mr)\times(r^2+r^3)}$, which are computed from $\mathbf{a}^{snap}(t_j)$. Before giving the mathematical expressions of $\mathbf{f}$ and E, we introduce the notation: $a_i^j := a_i^{\text{snap}}(t_j)$ and $\tau_i^{\text{true}}(t_j) := \tau_{i,j}$, $i = 1, 2, \cdots, r$ and $j = 1, 2, \cdots, M$, as well as

$$a'_{\cdot j} = \begin{pmatrix} a_1^{\text{snap}}(t_j) & a_2^{\text{snap}}(t_j) & a_3^{\text{snap}}(t_j) & \cdots & a_r^{\text{snap}}(t_j) \end{pmatrix} \in \mathbb{R}^{1\times r},$$
$$b'_{\cdot\cdot j} = \begin{pmatrix} a_1^{\text{snap}}(t_j)a_1^{\text{snap}}(t_j) & a_1^{\text{snap}}(t_j)a_2^{\text{snap}}(t_j) & \cdots & a_1^{\text{snap}}(t_j)a_r^{\text{snap}}(t_j) & \cdots & \cdots & a_r^{\text{snap}}(t_j)a_r^{\text{snap}}(t_j) \end{pmatrix} \in \mathbb{R}^{1\times r^2}.$$

Now that the vector $\mathbf{f}$ and the matrix E yield

$$E = \begin{pmatrix} a'_{\cdot 1} & 0 & 0 & \cdots & 0 & b'_{\cdot\cdot 1} & 0 & 0 & 0 & \cdots & 0 \\ 0 & a'_{\cdot 1} & 0 & \cdots & 0 & 0 & b'_{\cdot\cdot 1} & 0 & 0 & \cdots & 0 \\ 0 & 0 & a'_{\cdot 1} & \cdots & 0 & 0 & 0 & b'_{\cdot\cdot 1} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & a'_{\cdot 1} & 0 & 0 & 0 & 0 & \cdots & b'_{\cdot\cdot 1} \\ a'_{\cdot 2} & 0 & 0 & \cdots & 0 & b'_{\cdot\cdot 2} & 0 & 0 & 0 & \cdots & 0 \\ 0 & a'_{\cdot 2} & 0 & \cdots & 0 & 0 & b'_{\cdot\cdot 2} & 0 & 0 & \cdots & 0 \\ 0 & 0 & a'_{\cdot 2} & \cdots & 0 & 0 & 0 & b'_{\cdot\cdot 2} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & a'_{\cdot 2} & 0 & 0 & 0 & 0 & \cdots & b'_{\cdot\cdot 2} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \cdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \cdots & \vdots \\ a'_{\cdot M} & 0 & 0 & \cdots & 0 & b'_{\cdot\cdot M} & 0 & 0 & 0 & \cdots & 0 \\ 0 & a'_{\cdot M} & 0 & \cdots & 0 & 0 & b'_{\cdot\cdot M} & 0 & 0 & \cdots & 0 \\ 0 & 0 & a'_{\cdot M} & \cdots & 0 & 0 & 0 & b'_{\cdot\cdot M} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & a'_{\cdot M} & 0 & 0 & 0 & 0 & \cdots & b'_{\cdot\cdot M} \end{pmatrix} \in \mathbb{R}^{Mr\times(r^2+r^3)}, \quad (3.27)$$

$$\mathbf{f} = \begin{pmatrix} \tau_{1,1} \\ \tau_{2,1} \\ \tau_{3,1} \\ \vdots \\ \tau_{r,1} \\ \tau_{1,2} \\ \tau_{2,2} \\ \tau_{3,2} \\ \vdots \\ \tau_{r,2} \\ \vdots \\ \vdots \\ \tau_{1,M} \\ \tau_{2,M} \\ \tau_{3,M} \\ \vdots \\ \tau_{r,M} \end{pmatrix} \in \mathbb{R}^{Mr\times 1}. \quad (3.28)$$

Denoting $E_{k\bullet}$ the kth row of the matrix E, we observe that $E_{k_1\bullet}$ and $E_{k_2\bullet}$ are linearly independent. We remark that $\mathbf{X}, \mathbf{f}, E$, satisfy the relation

$$\sum_{j=1}^{M} \|\tau^{true}(t_j) - \tilde{A}\mathbf{a}^{snap}(t_j) - \mathbf{a}^{snap}(t_j)^T \tilde{B}\mathbf{a}^{snap}(t_j)\|^2 = \|\mathbf{f} - EX\|^2. \tag{3.29}$$

Therefore, the optimization problem 3.25 becomes the least square problem

$$\min_{X \in \mathbb{R}^{(r^2+r^3)\times 1}} \|\mathbf{f} - E\mathbf{X}\|^2. \tag{3.30}$$

Note that the least square problem 3.30 can be solved by the truncated *singular value decomposition* (SVD) algorithm [30]:

Algorithm 3.5. Truncated SVD Method

---

1. Calculate the SVD of E:

$$E = U\Sigma V. \tag{3.31}$$

2. Specify a tolerence tol.

3. Keep the entries in Σ that are larger than tol; the result matrix is $\hat{\Sigma}$ ($\hat{\sigma} = \sigma$ if $\sigma > tol$).

4. Construct the truncated SVD of E, $\hat{E}$:

$$\hat{E} = \hat{U}\hat{\Sigma}\hat{V}, \tag{3.32}$$

   where $\hat{U}$ and $\hat{V}$ are the entries of U, V that correspond to $\hat{\Sigma}$.

5. The solution is given by

$$\mathbf{X} = \left(\hat{V}\hat{\Sigma}^{-1}\hat{U}\right)\mathbf{f}. \tag{3.33}$$

6. Extract each entries of X to construct $\tilde{A}, \tilde{B}$.

Remark 3.6. The data-driven least squares problems 3.25 can be ill-conditioned; hence the *truncated SVD method* is used here as a remedy.

3.3 New Algorithms for DDC-ROM

In solving the optimization problem 3.30, the most expensive step is to solve the truncated SVD problem. Although 3.27 shares the sparsity feature in the matrix computation, it yields $Mr \times (r+r^2)$ non-zero entries, which leads to a large computational burden when performing the SVD.

With the observation of the repetitive pattern in matrix E (3.27), we can reduce the size of E and thus the computational cost of the SVD. First we permute the matrix E to $\tilde{E}$:

$$\tilde{E} = \begin{pmatrix} a'_{\cdot 1} & b'_{\cdot\cdot 1} & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ a'_{\cdot 2} & b'_{\cdot\cdot 2} & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ a'_{\cdot 3} & b'_{\cdot\cdot 3} & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a'_{\cdot M} & b'_{\cdot\cdot M} & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & a'_{\cdot 1} & b'_{\cdot\cdot 1} & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & a'_{\cdot 2} & b'_{\cdot\cdot 2} & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & a'_{\cdot 3} & b'_{\cdot\cdot 3} & 0 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & a'_{\cdot M} & b'_{\cdot\cdot M} & 0 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & a'_{\cdot 1} & b'_{\cdot\cdot 1} \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & a'_{\cdot 2} & b'_{\cdot\cdot 2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & a'_{\cdot M} & b'_{\cdot\cdot M} \end{pmatrix} \in \mathbb{R}^{Mr \times (r^2+r^3)}. \tag{3.34}$$

Similarly, we permute $\tilde{F}$ and $\tilde{X}$ as follows:

$$\tilde{F} = \begin{pmatrix} \tau_{1,1} \\ \tau_{1,2} \\ \tau_{1,3} \\ \vdots \\ \tau_{1,M} \\ \tau_{2,1} \\ \tau_{2,2} \\ \tau_{2,3} \\ \vdots \\ \tau_{2,M} \\ \vdots \\ \vdots \\ \tau_{r,1} \\ \tau_{r,2} \\ \tau_{r,3} \\ \vdots \\ \tau_{r,M} \end{pmatrix} \in \mathbb{R}^{Mr\times 1} \quad \text{and} \quad \tilde{X} = \begin{pmatrix} \tilde{A}_{1,1} \\ \tilde{A}_{1,2} \\ \vdots \\ \tilde{A}_{1,r} \\ \tilde{B}_{1,1,1} \\ \tilde{B}_{1,2,1} \\ \vdots \\ \tilde{B}_{r,r,1} \\ \tilde{A}_{2,1} \\ \vdots \\ \tilde{A}_{2,r} \\ \tilde{B}_{1,1,2} \\ \tilde{B}_{1,2,2} \\ \vdots \\ \tilde{B}_{r,r,2} \\ \vdots \\ \tilde{A}_{r,1} \\ \tilde{A}_{r,2} \\ \vdots \\ \tilde{A}_{r,r} \\ \tilde{B}_{1,1,r} \\ \vdots \\ \tilde{B}_{r,r,r} \end{pmatrix}. \tag{3.35}$$

The problem 3.30 is equivalent to finding the solution of the following least square problem:

$$\min \|\tilde{F} - \tilde{E}\tilde{X}\|^2. \tag{3.36}$$

Defining

$$\tilde{E}_0 = \begin{pmatrix} a'_{\cdot 1} & b'_{\cdot\cdot 1} \\ a'_{\cdot 2} & b'_{\cdot\cdot 2} \\ a'_{\cdot 3} & b'_{\cdot\cdot 3} \\ \vdots & \vdots \\ a'_{\cdot M} & b'_{\cdot\cdot M} \end{pmatrix} \in \mathbb{R}^{M\times(r+r^2)}, \tag{3.37}$$

we get $\tilde{E}$ which can be expressed as a block diagonal matrix about $\tilde{E}_0$:

$$\tilde{E} = \begin{pmatrix} \tilde{E}_0 & \mathcal{O} & \mathcal{O} & \cdots & \mathcal{O} \\ \mathcal{O} & \tilde{E}_0 & \mathcal{O} & \cdots & \mathcal{O} \\ \vdots & \vdots & \ddots & & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ \mathcal{O} & \mathcal{O} & \cdots & \mathcal{O} & \tilde{E}_0 \end{pmatrix} \in \mathbb{R}^{M\times(r+r^2)}.$$

Defining

$$X_{\cdot j} = \begin{pmatrix} \tilde{A}_{j,1} \\ \tilde{A}_{j,2} \\ \vdots \\ \tilde{A}_{j,r} \\ \tilde{B}_{1,1,j} \\ \tilde{B}_{1,2,j} \\ \vdots \\ \tilde{B}_{r,r,j} \end{pmatrix} \in \mathbb{R}^{(r+r^2)\times 1}; j = 1, 2, \cdots, r \tag{3.38}$$

and

$$T_{\cdot k} = \begin{pmatrix} \tau_{k,1} \\ \tau_{k,2} \\ \tau_{k,3} \\ \vdots \\ \tau_{k,M} \end{pmatrix} \in \mathbb{R}^{M\times 1}; k = 1, 2, \cdots, r, \tag{3.39}$$

we obtain

$$\|\tilde{F} - \tilde{E}\tilde{X}\|^2 = \left\| \begin{pmatrix} T_{\cdot 1} \\ T_{\cdot 2} \\ \vdots \\ T_{\cdot r} \end{pmatrix} - \begin{pmatrix} \tilde{E}_0 & \mathcal{O} & \mathcal{O} & \cdots & \mathcal{O} \\ \mathcal{O} & \tilde{E}_0 & \mathcal{O} & \cdots & \mathcal{O} \\ \vdots & \vdots & \ddots & & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ \mathcal{O} & \mathcal{O} & \cdots & \mathcal{O} & \tilde{E}_0 \end{pmatrix} \begin{pmatrix} X_{\cdot 1} \\ X_{\cdot 2} \\ \vdots \\ X_{\cdot r} \end{pmatrix} \right\|^2 = \sum_{j=1}^{r} \left\| T_{\cdot j} - \tilde{E}_0 X_{\cdot j} \right\|^2. \tag{3.40}$$

Thus, we can use the alternate truncated SVD algorithm to solve the least squares problem 3.40:

Algorithm 3.7. Alternative Truncated SVD Method for the optimization problem

1. Calculate the SVD of $\tilde{E}_0 = U_s \Sigma_s V_s^T$,
2. Construct matrix $\hat{\Sigma}_s$ from Σ_s,
3. Let $\tilde{E}_0 = \hat{U}_s \hat{\Sigma}_s \hat{V_s}^T$;
4. Construct
$$\begin{pmatrix} X_{\cdot 1} & X_{\cdot 2} & \cdots & X_{\cdot r} \end{pmatrix} = \hat{C}_s \hat{\Sigma_s}^{-1} \hat{U_s}^T \begin{pmatrix} T_{\cdot 1} & T_{\cdot 2} & \cdots & T_{\cdot r} \end{pmatrix},$$
5. Extract each entries of $X_{\cdot j}, \forall j = 1, \cdots, r$ to construct $\tilde{A}, \tilde{B}$.

Remark 3.8. Algorithm 3.7 proposed in this section reproduces exactly the same result as the algorithm 3.5, i.e., with the same snapshots and numerical setting, they produce the same same $\tilde{A}$ and $\tilde{B}$. This is verified numerically in Tables 3.2 and 3.3.

Remark 3.9. Instead of performing SVD of E of size $Mr \times (r^2 + r^3)$, we only need to perform SVD of E_0, which is of size $M \times (r + r^2)$. Table 3.1 reveals that algorithm 3.7 is *orders of magnitude more efficient* than algorithm 3.5.

The *Frobenius Norm* of a matrix is given by [36]

$$\|A\|_F = \left(\sum_{i=1}^{r} \sum_{j=1}^{r} A_{ij}^2 \right)^{1/2} \quad \text{for } A \in \mathbb{R}^{r \times r}. \tag{3.41}$$

Similarly, we define the *Frobenius Norm* for a tensor as

$$\|B\|_F = \left(\sum_{i=1}^{r} \sum_{j=1}^{r} \sum_{k=1}^{r} B_{ijk}^2 \right)^{1/2} \quad \text{for } B \in \mathbb{R}^{r \times r \times r}. \tag{3.42}$$

We use the *Frobenius Norm* to compare the $\tilde{A}$ and $\tilde{B}$ generated by the two algorithms:

Table 3.1: Comparison of CPU time in the generation of $\tilde{A}$ and $\tilde{B}$.

r values	6		11		20	
	Alg 1	Alg 2	Alg 1	Alg 2	Alg 1	Alg 2
$\nu = 10^{-2}$	$0.73s$	$0.095s$	$8.97s$	0.12	$302s$	$0.24s$
$\nu = 10^{-3}$	$1.08s$	$0.49s$	$8.49s$	$0.797s$	$353s$	$1.58s$
$\nu = 10^{-4}$	$15.5s$	$8.00s$	$22.2s$	$14.9s$	$329s$	$25.3s$
$\nu = 10^{-5}$	$17.0s$	$8.28s$	$23.8s$	$16.1s$	$349s$	$27.0s$

Table 3.2: $\|\tilde{A}^{algorithm1} - \tilde{A}^{algorithm2}\|_F$ for different ν and r values; comparison of $\tilde{A}$ generated by the two different algorithms.

ansatz r values	6	11	20
$\nu = 10^{-2}$	$2.0691e-11$	$5.2789e-11$	$5.7157e-11$
$\nu = 10^{-3}$	$3.1908e-11$	$3.4951e-11$	$6.2159e-10$
$\nu = 10^{-4}$	$1.6734e-11$	$2.0007e-11$	$2.9709e-10$
$\nu = 10^{-5}$	$1.9483e-11$	$2.6633e-11$	$4.0770e-10$

Table 3.3: $\|\tilde{B}^{algorithm1} - \tilde{B}^{algorithm2}\|_F$ for different ν and r values; comparison of $\tilde{B}$ generated by the two different algorithms.

r values	6	11	20
$\nu = 10^{-2}$	$3.7706e-11$	$1.1260e-10$	$1.3575e-10$
$\nu = 10^{-3}$	$9.0420e-11$	$6.3923e-11$	$1.1003e-09$
$\nu = 10^{-4}$	$4.3483e-11$	$3.9628e-11$	$5.6605e-10$
$\nu = 10^{-5}$	$3.7062e-11$	$5.9669e-11$	$1.6415e-09$

3.4 Simple Case for Illustration

To illustrate the truncated *Singular Value Decomposition (SVD)* used in Algorithm 3.7, we consider a two-by-two simple case, i.e., $r = 2$, $\tilde{A} \in \mathbb{R}^{2\times 2}$,and $\tilde{B} \in \mathbb{R}^{2\times 2\times 2}$. This yields

$$\tilde{A} = \begin{bmatrix} \tilde{A}_{11} & \tilde{A}_{12} \\ \tilde{A}_{21} & \tilde{A}_{22} \end{bmatrix}. \tag{3.43}$$

For the tensor, we write

$$\tilde{B}_{..1} = \begin{bmatrix} \tilde{B}_{111} & \tilde{B}_{121} \\ \tilde{B}_{211} & \tilde{B}_{221} \end{bmatrix}, \qquad \tilde{B}_{..2} = \begin{bmatrix} \tilde{B}_{112} & \tilde{B}_{122} \\ \tilde{B}_{212} & \tilde{B}_{222} \end{bmatrix}. \tag{3.44}$$

In addition, for this simple setting, we consider 2 snapshots, $\mathbf{a}(t_1), \mathbf{a}(t_2)$, such that

$$\mathbf{a}(t_1) = \begin{bmatrix} a_1(t_1) \\ a_2(t_1) \end{bmatrix}, \qquad \mathbf{a}(t_2) = \begin{bmatrix} a_1(t_2) \\ a_2(t_2) \end{bmatrix}. \tag{3.45}$$

As a result, from the *full order model* (FOM), $\tau^{true} \in \mathbb{R}^{2\times 2}$ in vector form can be represented as:

$$\tau^{true}(t_1) = \begin{bmatrix} \tau_1^{true}(t_1) \\ \tau_2^{true}(t_1) \end{bmatrix}, \qquad \mathbf{a}(t_2) = \begin{bmatrix} \tau_1^{true}(t_2) \\ \tau_2^{true}(t_2) \end{bmatrix}. \tag{3.46}$$

Therefore, the optimization problem 3.25 yields

$$\min_{\tilde{A}\in\mathbb{R}^{2\times 2},\quad \tilde{B}\in\mathbb{R}^{2\times 2\times 2}} \left\{ \left\| \tau^{true}(t_1) - \left(\tilde{A}\mathbf{a}(t_1) + \mathbf{a}(t_1)^T \tilde{B}\mathbf{a}(t_1) \right) \right\|^2 + \left\| \tau^{true}(t_2) - \left(\tilde{A}\mathbf{a}(t_2) + \mathbf{a}(t_2)^T \tilde{B}\mathbf{a}(t_2) \right) \right\|^2 \right\}. \tag{3.47}$$

Before we write the optimization problem into matrix form, the term $\tilde{A}\mathbf{a}(t_i)+\mathbf{a}(t_i)^T\tilde{B}\mathbf{a}(t_i), i = 1, 2$ can be expressed in vector form:

$$\begin{aligned} &\tilde{A}\mathbf{a}(t_1) + \mathbf{a}(t_1)^T \tilde{B}\mathbf{a}(t_1) \\ &= \begin{bmatrix} \tilde{A}_{11}a_1(t_1)+\tilde{A}_{12}a_2(t_1)+a_1^2(t_1)\tilde{B}_{111}+a_1(t_1)a_2(t_1)\tilde{B}_{121}+a_2(t_1)a_1(t_1)\tilde{B}_{211}+a_2^2(t_1)\tilde{B}_{221} \\ \tilde{A}_{21}a_1(t_1)+\tilde{A}_{22}a_2(t_1)+a_1^2(t_1)\tilde{B}_{112}+a_1(t_1)a_2(t_1)\tilde{B}_{122}+a_2(t_1)a_1(t_1)\tilde{B}_{212}+a_2^2(t_1)\tilde{B}_{222} \end{bmatrix}, \end{aligned} \tag{3.48}$$

$$\begin{aligned} &\tilde{A}\mathbf{a}(t_2) + \mathbf{a}(t_2)^T \tilde{B}\mathbf{a}(t_2) \\ &= \begin{bmatrix} \tilde{A}_{11}a_1(t_2)+\tilde{A}_{12}a_2(t_2)+a_1^2(t_2)\tilde{B}_{111}+a_1(t_2)a_2(t_2)\tilde{B}_{121}+a_2(t_2)a_1(t_2)\tilde{B}_{211}+a_2^2(t_2)\tilde{B}_{221} \\ \tilde{A}_{21}a_1(t_2)+\tilde{A}_{22}a_2(t_2)+a_1^2(t_2)\tilde{B}_{112}+a_1(t_2)a_2(t_2)\tilde{B}_{122}+a_2(t_2)a_1(t_2)\tilde{B}_{212}+a_2^2(t_2)\tilde{B}_{222} \end{bmatrix}. \end{aligned} \tag{3.49}$$

Construction of Algorithm 3.7

Finding the entries of $\tilde{A}, \tilde{B}$ yields $r^2 + r^3 = 2^2 + 2^3 = 12$ unknowns. Once we ensure that the ordering of the elements is consistent, we can vectorize a matrix and tensor [51], i.e., define the vector $\mathbf{x} \in \mathbb{R}^{12\times 1}$

$$\mathbf{x} = \begin{bmatrix} \tilde{A}_{11} \\ \tilde{A}_{12} \\ \tilde{A}_{21} \\ \tilde{A}_{22} \\ \tilde{B}_{111} \\ \tilde{B}_{121} \\ \tilde{B}_{211} \\ \tilde{B}_{221} \\ \tilde{B}_{112} \\ \tilde{B}_{122} \\ \tilde{B}_{212} \\ \tilde{B}_{222} \end{bmatrix}. \tag{3.50}$$

This yields:

$$\tilde{A}\mathbf{a}(t_1) + \mathbf{a}(t_1)^T \tilde{B}\mathbf{a}(t_1)$$

$$= \begin{bmatrix} a_1(t_1) & a_2(t_1) & 0 & 0 & a_1^2(t_1) & a_1(t_1)a_2(t_1) & a_2(t_1)a_1(t_1) & a_2^2(t_1) & 0 & 0 & 0 & 0 \\ 0 & 0 & a_1(t_1) & a_2(t_1) & 0 & 0 & 0 & 0 & a_1^2(t_1) & a_1(t_1)a_2(t_1) & a_2(t_1)a_1(t_1) & a_2^2(t_1) \end{bmatrix} \begin{bmatrix} \tilde{A}_{11} \\ \tilde{A}_{12} \\ \tilde{A}_{21} \\ \tilde{A}_{22} \\ \tilde{B}_{111} \\ \tilde{B}_{121} \\ \tilde{B}_{211} \\ \tilde{B}_{221} \\ \tilde{B}_{112} \\ \tilde{B}_{122} \\ \tilde{B}_{212} \\ \tilde{B}_{222} \end{bmatrix}. \tag{3.51}$$

$$\tilde{A}\mathbf{a}(t_2) + \mathbf{a}(t_2)^T \tilde{B}\mathbf{a}(t_2)$$

$$= \begin{bmatrix} a_1(t_2) & a_2(t_2) & 0 & 0 & a_1^2(t_2) & a_1(t_2)a_2(t_2) & a_2(t_2)a_1(t_2) & a_2^2(t_2) & 0 & 0 & 0 & 0 \\ 0 & 0 & a_1(t_2) & a_2(t_2) & 0 & 0 & 0 & 0 & a_1^2(t_2) & a_1(t_2)a_2(t_2) & a_2(t_2)a_1(t_2) & a_2^2(t_2) \end{bmatrix} \begin{bmatrix} \tilde{A}_{11} \\ \tilde{A}_{12} \\ \tilde{A}_{21} \\ \tilde{A}_{22} \\ \tilde{B}_{111} \\ \tilde{B}_{121} \\ \tilde{B}_{211} \\ \tilde{B}_{221} \\ \tilde{B}_{112} \\ \tilde{B}_{122} \\ \tilde{B}_{212} \\ \tilde{B}_{222} \end{bmatrix}. \tag{3.52}$$

If we consider the Euclidean norm for vectors, the optimization problem 3.47 yields:

$$\min_{\mathbf{x}\in\mathbb{R}^{12}} \|\mathbf{f} - E\mathbf{x}\|^2, \tag{3.53}$$

where

$$\mathbf{f} = \begin{bmatrix} \tau_1^{true}(t_1) \\ \tau_2^{true}(t_1) \\ \tau_1^{true}(t_2) \\ \tau_2^{true}(t_2) \end{bmatrix}, \qquad \mathbf{x} = \begin{bmatrix} \tilde{A}_{11} \\ \tilde{A}_{12} \\ \tilde{A}_{21} \\ \tilde{A}_{22} \\ \tilde{B}_{111} \\ \tilde{B}_{121} \\ \tilde{B}_{211} \\ \tilde{B}_{221} \\ \tilde{B}_{112} \\ \tilde{B}_{122} \\ \tilde{B}_{212} \\ \tilde{B}_{222} \end{bmatrix}, \tag{3.54}$$

$$E = \begin{bmatrix} a_1(t_1) & a_2(t_1) & 0 & 0 & a_1^2(t_1) & a_1(t_1)a_2(t_1) & a_2(t_1)a_1(t_1) & a_2^2(t_1) & 0 & 0 & 0 & 0 \\ 0 & 0 & a_1(t_1) & a_2(t_1) & 0 & 0 & 0 & 0 & a_1^2(t_1) & a_1(t_1)a_2(t_1) & a_2(t_1)a_1(t_1) & a_2^2(t_1) \\ a_1(t_2) & a_2(t_2) & 0 & 0 & a_1^2(t_2) & a_1(t_2)a_2(t_2) & a_2(t_2)a_1(t_2) & a_2^2(t_2) & 0 & 0 & 0 & 0 \\ 0 & 0 & a_1(t_2) & a_2(t_2) & 0 & 0 & 0 & 0 & a_1^2(t_2) & a_1(t_2)a_2(t_2) & a_2(t_2)a_1(t_2) & a_2^2(t_2) \end{bmatrix}, \tag{3.55}$$

with $E \in \mathbb{R}^{12\times 4}$, this yields a *rank-deficient least squares problem* [30] which can be solved with the truncated SVD in Algorithm 3.5.

Relationship between Algorithm 3.5 to Algorithm 3.7

To construct the structure in Algorithm 3.7, first we permute the $\tilde{\mathbf{x}}$ column-wise, .i.e., we perform left multiplication by a permutation matrix $\mathcal{O}_c \in \mathbb{R}^{12\times 12}$

$$\mathcal{O}_c = \begin{bmatrix} \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} \end{bmatrix}, \tag{3.56}$$

such that

$$\tilde{\mathbf{x}} = \mathcal{O}_c\mathbf{x} = \begin{bmatrix} \tilde{A}_{11} \\ \tilde{A}_{12} \\ \tilde{B}_{111} \\ \tilde{B}_{121} \\ \tilde{B}_{211} \\ \tilde{B}_{221} \\ \tilde{A}_{21} \\ \tilde{A}_{22} \\ \tilde{B}_{112} \\ \tilde{B}_{122} \\ \tilde{B}_{212} \\ \tilde{B}_{222} \end{bmatrix}. \tag{3.57}$$

Then, the corresponding permutation of E yields:

$$\tilde{E} = E\mathcal{O}_c^T = \begin{bmatrix} a_1(t_1) & a_2(t_1) & a_1^2(t_1) & a_1(t_1)a_2(t_1) & a_2(t_1)a_1(t_1) & a_2^2(t_1) & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & a_1(t_1) & a_2(t_1) & a_1^2(t_1) & a_1(t_1)a_2(t_1) & a_2(t_1)a_1(t_1) & a_2^2(t_1) \\ a_1(t_2) & a_2(t_2) & a_1^2(t_2) & a_1(t_2)a_2(t_2) & a_2(t_2)a_1(t_2) & a_2^2(t_2) & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & a_1(t_2) & a_2(t_2) & a_1^2(t_2) & a_1(t_2)a_2(t_2) & a_2(t_2)a_1(t_2) & a_2^2(t_2) \end{bmatrix}. \tag{3.58}$$

The permutation matrix satisfies the relation $\mathcal{O}_c^T\mathcal{O}_c = \mathcal{I}$, such that $\tilde{E}\tilde{\mathbf{x}} = E\mathcal{O}_c^T\mathcal{O}_c\mathbf{x} = E\mathbf{x}$. Then,

$$\min_{\mathbf{x}\in\mathbb{R}^{12}} \|f - E\mathbf{x}\|^2 \quad \Leftrightarrow \quad \min_{\tilde{\mathbf{x}}\in\mathbb{R}^{12}} \|f - \tilde{E}\tilde{\mathbf{x}}\|^2. \tag{3.59}$$

If we permute row-wise E and $\mathbf{f}$ simultaneously, i.e., we perform left multiplication with a permutation matrix $\mathcal{O}_r$, we obtain

$$\tilde{f} = \mathcal{O}_r f = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \tau_1^{true}(t_1) \\ \tau_2^{true}(t_1) \\ \tau_1^{true}(t_2) \\ \tau_2^{true}(t_2) \end{bmatrix} = \begin{bmatrix} \tau_1^{true}(t_1) \\ \tau_1^{true}(t_2) \\ \tau_2^{true}(t_1) \\ \tau_2^{true}(t_2) \end{bmatrix}, \tag{3.60}$$

$$\begin{aligned} \tilde{\tilde{E}} &= \mathcal{O}_r\tilde{E} \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \\ &\quad \begin{bmatrix} a_1(t_1) & a_2(t_1) & a_1^2(t_1) & a_1(t_1)a_2(t_1) & a_2(t_1)a_1(t_1) & a_2^2(t_1) & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & a_1(t_1) & a_2(t_1) & a_1^2(t_1) & a_1(t_1)a_2(t_1) & a_2(t_1)a_1(t_1) & a_2^2(t_1) \\ a_1(t_2) & a_2(t_2) & a_1^2(t_2) & a_1(t_2)a_2(t_2) & a_2(t_2)a_1(t_2) & a_2^2(t_2) & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & a_1(t_2) & a_2(t_2) & a_1^2(t_2) & a_1(t_2)a_2(t_2) & a_2(t_2)a_1(t_2) & a_2^2(t_2) \end{bmatrix} \\ &= \begin{bmatrix} a_1(t_1) & a_2(t_1) & a_1^2(t_1) & a_1(t_1)a_2(t_1) & a_2(t_1)a_1(t_1) & a_2^2(t_1) & 0 & 0 & 0 & 0 & 0 & 0 \\ a_1(t_2) & a_2(t_2) & a_1^2(t_2) & a_1(t_2)a_2(t_2) & a_2(t_2)a_1(t_2) & a_2^2(t_2) & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & a_1(t_2) & a_2(t_2) & a_1^2(t_2) & a_1(t_2)a_2(t_2) & a_2(t_2)a_1(t_2) & a_2^2(t_2) \\ 0 & 0 & 0 & 0 & 0 & 0 & a_1(t_1) & a_2(t_1) & a_1^2(t_1) & a_1(t_1)a_2(t_1) & a_2(t_1)a_1(t_1) & a_2^2(t_1) \end{bmatrix}. \end{aligned} \tag{3.61}$$

Since the permutation of a vector does not changes its Euclidean norm, we have the following equivalent least square problems:

$$\min_{\mathbf{x}\in\mathbb{R}^{12}} \|f - E\mathbf{x}\|^2 \quad \Leftrightarrow \quad \min_{\tilde{\mathbf{x}}\in\mathbb{R}^{12}} \|f - \tilde{E}\tilde{\mathbf{x}}\|^2 \quad \Leftrightarrow \quad \min_{\tilde{\mathbf{x}}\in\mathbb{R}^{12}} \|\tilde{f} - \tilde{\tilde{E}}\tilde{\mathbf{x}}\|^2. \tag{3.62}$$

On the other hand, notice that

$$\tilde{\tilde{E}} = \begin{bmatrix} E_0 & \mathbf{0} \\ \mathbf{0} & E_0 \end{bmatrix}, \qquad \text{with} \qquad E_0 = \begin{bmatrix} a_1(t_1) & a_2(t_1) & a_1^2(t_1) & a_1(t_1)a_2(t_1) & a_2(t_1)a_1(t_1) & a_2^2(t_1) \\ a_1(t_2) & a_2(t_2) & a_1^2(t_2) & a_1(t_2)a_2(t_2) & a_2(t_2)a_1(t_2) & a_2^2(t_2) \end{bmatrix}. \tag{3.63}$$

The SVD of E_0 gives $E_0 = U_0 \Sigma_0 V_0^T$. Then,

$$\tilde{\tilde{E}} = \tilde{U}\tilde{\Sigma}\tilde{V}^T = \begin{bmatrix} U_0 & U_0 \end{bmatrix} \begin{bmatrix} \Sigma_0 & \mathbf{0} \\ \mathbf{0} & \Sigma_0 \end{bmatrix} \begin{bmatrix} V_0^T \\ V_0^T \end{bmatrix}, \tag{3.64}$$

with

$$\tilde{U} = \begin{bmatrix} U_0 & U_0 \end{bmatrix}, \quad \tilde{\Sigma} = \begin{bmatrix} \Sigma_0 & \mathbf{0} \\ \mathbf{0} & \Sigma_0 \end{bmatrix}, \quad \tilde{V}^T = \begin{bmatrix} V_0^T \\ V_0^T \end{bmatrix}. \tag{3.65}$$

We define

$$\tilde{\mathbf{x}} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix}, \qquad \text{with} \qquad \mathbf{y}_1 = \begin{bmatrix} \tilde{A}_{11} \\ \tilde{A}_{12} \\ \tilde{B}_{111} \\ \tilde{B}_{121} \\ \tilde{B}_{211} \\ \tilde{B}_{221} \end{bmatrix}, \quad \mathbf{y}_2 = \begin{bmatrix} \tilde{A}_{21} \\ \tilde{A}_{22} \\ \tilde{B}_{112} \\ \tilde{B}_{122} \\ \tilde{B}_{212} \\ \tilde{B}_{222} \end{bmatrix},$$

and

$$\tilde{f} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}, \qquad \text{with} \qquad f_1 = \begin{bmatrix} \tau_1^{true}(t_1) \\ \tau_1^{true}(t_2) \end{bmatrix}, f_2 = \begin{bmatrix} \tau_2^{true}(t_1) \\ \tau_2^{true}(t_2) \end{bmatrix}.$$

The truncated SVD for a block diagonal matrix gives the solution to the *rank deficient least square problem* 3.53

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix} = \hat{\tilde{V}}\hat{\tilde{\Sigma}}^{-1}\hat{\tilde{U}}^T\tilde{f} = \begin{bmatrix} \hat{V}_0 & \hat{V}_0 \end{bmatrix} \begin{bmatrix} \hat{\Sigma}_0^{-1} & \mathbf{0} \\ \mathbf{0} & \hat{\Sigma}_0^{-1} \end{bmatrix} \begin{bmatrix} \hat{U}_0^T \\ \hat{U}_0^T \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}, \tag{3.66}$$

or in compressed form as in Algorithm 3.7,

$$\begin{bmatrix} \mathbf{y}_1 & \mathbf{y}_2 \end{bmatrix} = \hat{V}_0 \hat{\Sigma}_0^{-1} \hat{U}_0^T \begin{bmatrix} f_1 & f_2 \end{bmatrix}. \tag{3.67}$$

Remark 3.10. The most expensive part of Algorithms 3.5 and 3.7 is performing the truncated SVD of matrix E and E_0. For this two by two case, the size of E in equation (3.55) is 4×12 in Algorithm 3.5, while the size of E_0 in equation (3.63) is only 2×6 in Algorithm 3.7 which yields a more efficient algorithm. This is also shown in Table 3.1.

Remark 3.11. Note that Algorithm 3.7 yields: a *decoupled SVD* [74, 75]. In fact, we can show that Algorithm 3.7 solves M independent least square problems. From equation (3.25), we consider the matrix R, which consists of τ^{true} at different times:

$$\mathbf{R} = \left[\tau_i^{true}(t_j)\right]_{i,j}^T \in \mathbb{R}^{M \times r}. \tag{3.68}$$

We also consider $\hat{\mathbf{X}}$ in block form as

$$\hat{\mathbf{X}} = \begin{bmatrix} \mathbf{X}_{\cdot 1} & \mathbf{X}_{\cdot 2} & \cdots & \mathbf{X}_{\cdot r,} \end{bmatrix} \in \mathbb{R}^{(r+r^2) \times r}, \tag{3.69}$$

where $\mathbf{X}_{\cdot j}$ is defined in equation (3.38). Because the squared Frobenius norm of a matrix is the sum of the squares of all entries of the matrix which is equivalent to the sum of the square of Euclidean norm of columns, the sum in equation (3.25) can be written as a Frobenius norm. Thus, the optimization problem 3.25 yields the following least squares problem:

$$\min_{\hat{\mathbf{X}}\in\mathbb{R}^{(r+r^2)\times r}} \|E_0\hat{\mathbf{X}} - \mathbf{R}\|^2 \tag{3.70}$$

where E_0 is defined in equation (3.37). The least square problem 3.70 can be transformed into r independent least square problems in $\|\|_2$ norm, which can be proved with Lemma 1 in [74]. The r dependent least squares problems are

$$\min_{\mathbf{X}_{\cdot j}\in\mathbb{R}^{(r+r^2)}} \|E_0\mathbf{X}_{\cdot j} - \mathbf{R}_{\cdot j}\|^2, \qquad j = 1, 2, \cdots, r, \tag{3.71}$$

where $\mathbf{R}_{\cdot j}$ is the jth column of $\mathbf{R}$. Then, equation (3.71) can be solved by using Algorithm 3.7.

Remark 3.12. In Remark 3.11, we decoupled the original least squares problem into r independent least squares problems. Each one of the r independent least square problems in 3.71 corresponds to a projection: the kth least square problem, $\min_{\mathbf{X}_{\cdot k}\in\mathbb{R}^{(r+r^2)}} \|E_0\mathbf{X}_{\cdot k} - \mathbf{R}_{\cdot k}\|^2$ is related to the projection of the kth POD basis φ_k, i.e., $\mathbf{R}_{\cdot k} = [(\tau(t_i), \varphi_k)_{\mathcal{H}}]_i$.

3.5 DDC-ROM with Different ansatz

3.5.1 DDC-ROM with a Linear Ansatz

In chapter 3, a quadratic ansatz is applied for the ROM closure problem. We can also replace the quadratic ansatz with a linear ansatz in the DDC-ROM:

$$\tau = \tilde{A}\mathbf{a},$$

where $\tilde{A} \in \mathbb{R}^{r\times r}$. In component form, this can be written as

$$\tau_l^{ansatz}(a) = \sum_i \tilde{A}_{il} a_i. \tag{3.72}$$

As a result, the DDC-ROM with a linear ansatz yields

$$\dot{\mathbf{a}} = (\mathbf{A} + \tilde{\mathbf{A}})\mathbf{a} + \mathbf{a}^T\mathbf{B}\mathbf{a}. \tag{3.73}$$

To find $\tilde{\mathbf{A}}, \tilde{\mathbf{B}}$ and $\tilde{\mathbf{C}}$, we solve an optimization problem, which is similar to that solved for the quadratic ansatz:

$$\min_{\tilde{A}\in\mathbb{R}^{r\times r}} \sum_{j=1}^{M} \|\tau^{true}(t_j) - \tilde{A}\mathbf{a}^{snap}(t_j))\|^2. \tag{3.74}$$

We use the truncated *singular value decomposition* (SVD) (as we did in algorithms 3.5 and 3.7) to solve the optimization problem 3.74 and thus get the DDC-ROM with a linear ansatz.

3.5.2 DDC-ROM with a Cubic Ansatz

Instead of a linear or quadratic ansatz, we use a cubic ansatz for the DDC-ROM,:

$$\tau = \tilde{A}\mathbf{a} + \mathbf{a}^T\tilde{B}\mathbf{a} + \mathbf{a}^T(\mathbf{a}^T\tilde{C}\mathbf{a}),$$

where $\tilde{A} \in \mathbb{R}^{r\times r}, \tilde{B} \in \mathbb{R}^{r\times r\times r}$, and $\tilde{C} \in \mathbb{R}^{r\times r\times r\times r}$. In component form, this can be written as

$$\tau^{ansatz}(a)_l = \sum_i \tilde{A}_{il}a_i + \sum_{i,j} \tilde{B}_{ijl}a_ia_j + \sum_{i,j,k} \tilde{C}_{ijkl}a_ia_ja_k. \tag{3.75}$$

As a result, the DDC-ROM with a cubic ansatz yields the form

$$\dot{\mathbf{a}} = (\mathbf{A} + \tilde{\mathbf{A}})\mathbf{a} + \mathbf{a}^T(\mathbf{B} + \tilde{\mathbf{B}})\mathbf{a} + \mathbf{a}^{\mathbf{T}}(\mathbf{a}^{\mathbf{T}}\tilde{\mathbf{C}}\mathbf{a}). \tag{3.76}$$

To find $\tilde{\mathbf{A}}, \tilde{\mathbf{B}}$ and $\tilde{\mathbf{C}}$, we solve an optimization problem, similar to 3.74:

$$\min_{\substack{\tilde{A}\in\mathbb{R}^{r\times r};\\ \tilde{B}\in\mathbb{R}^{r\times r\times r}}} \sum_{j=1}^{M} \|\tau^{true}(t_j) - \tilde{A}\mathbf{a}^{snap}(t_j) - \mathbf{a}^{snap}(t_j)^T\tilde{B}\mathbf{a}^{snap}(t_j) - \mathbf{a}^{snap}(t_j)^T\left(\mathbf{a}^{snap}(t_j)^T\tilde{C}\mathbf{a}^{snap}(t_j)\right)\|^2. \tag{3.77}$$

We again use the truncated SVD to solve such optimization problem 3.77 and get the DDC-ROM with a cubic ansatz.

3.5.3 Idealized DDC-ROM

Instead of solving an optimization problem, we can directly use τ^{true} in the ROM, i.e.,

$$\dot{\mathbf{a}} = \mathbf{A}\mathbf{a} + \mathbf{a}^T\mathbf{B}\mathbf{a} + \tau^{\text{true}}, \tag{3.78}$$

Burgers where τ^{true} is directly computed from the *snapshots*.

3.6 DDC-ROM Numerical Experiments

In this section, we present numerical results for one-dimensional viscous Burgers equation with the DDC-ROM.

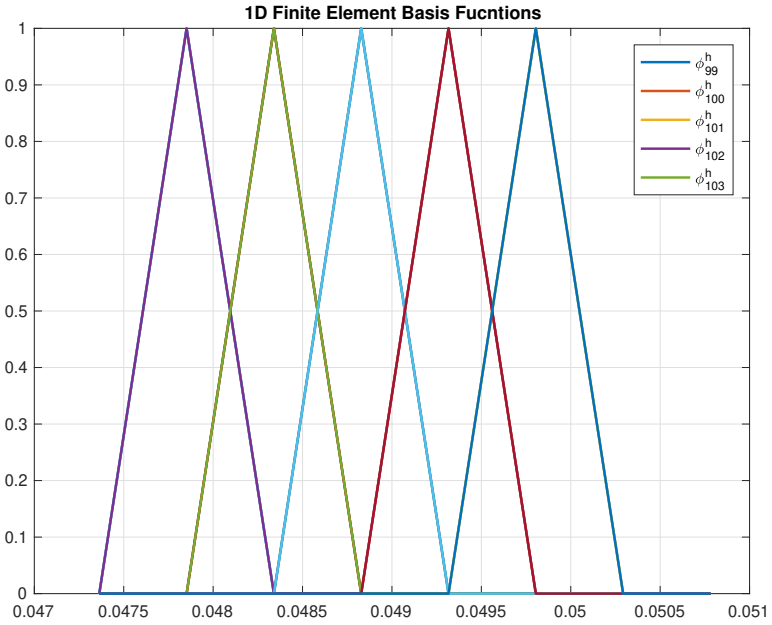


Figure 3.2: 1D piecewise linear finite element basis functions

1-D viscous Burgers equation

Consider the one-dimensional Burgers equation:

$$u_t - \nu u_{xx} + u u_x = f \qquad \text{in } \Omega \times (0,T), \tag{3.79}$$
$$u(x,0) = u_0(x) \qquad \text{in } \Omega, \tag{3.80}$$
$$u(x,t) = g(x,t) \qquad \text{on } \partial\Omega \times (0,T]. \tag{3.81}$$

The initial condition is

$$u_0(x) = \begin{cases} 1 & \text{if } x \in (0, \frac{1}{2}], \\ 0 & \text{if } x \in (\frac{1}{2}, 1). \end{cases} \tag{3.82}$$

The forcing term $f = 0, \Omega = [0,1]$, and $T = 1$. The boundary conditions are homogeneous Dirichlet, that is, $u(0,t) = u(1,t) = 0, \forall t \in [0,1]$. To generate the snapshots, we use piecewise linear finite elements with mesh size $h = 1/2048$ and the backward Euler method with a time step $\Delta t = 10^{-3}$, and save data at each time instance. The linear piecewise finite element shape functions are illustrated in Figure 3.2. The *direct numerical simulations* (DNS) [18, 31, 56, 94] for different ν values are plotted in Figure 3.3. All snapshots are used for the POD basis generation.

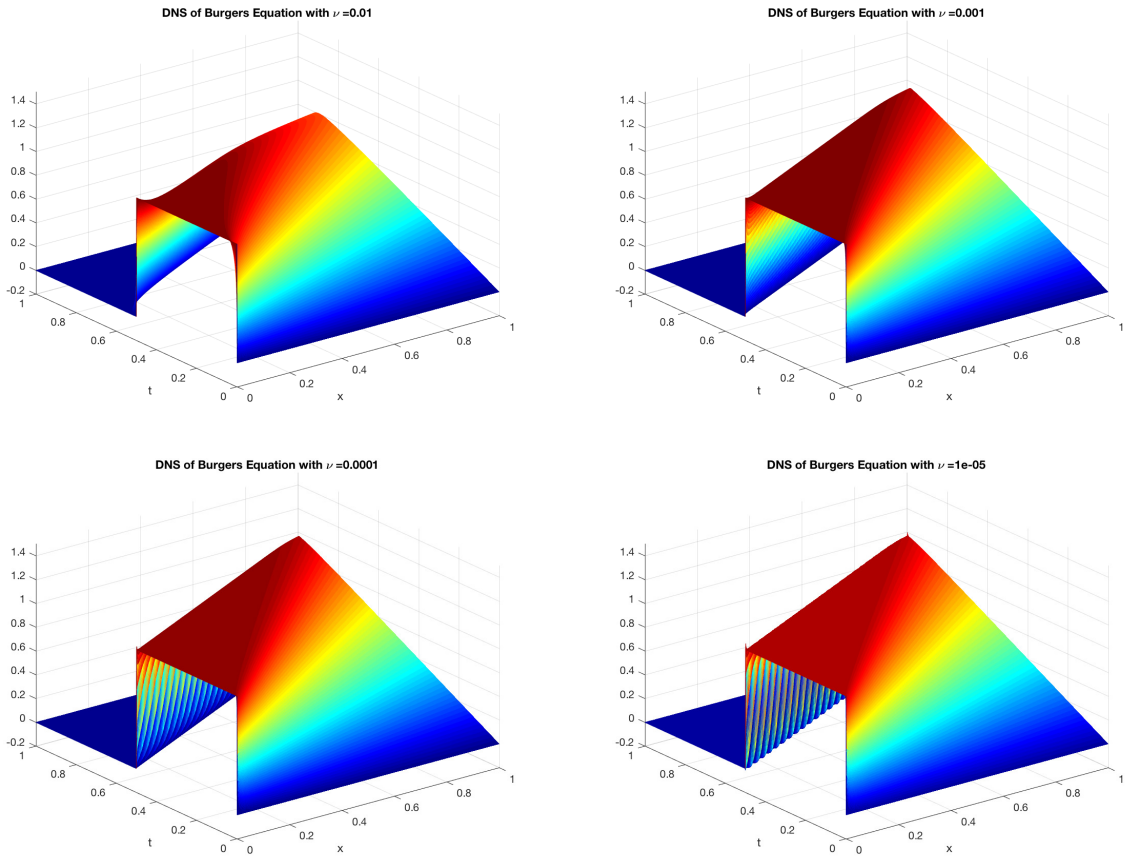


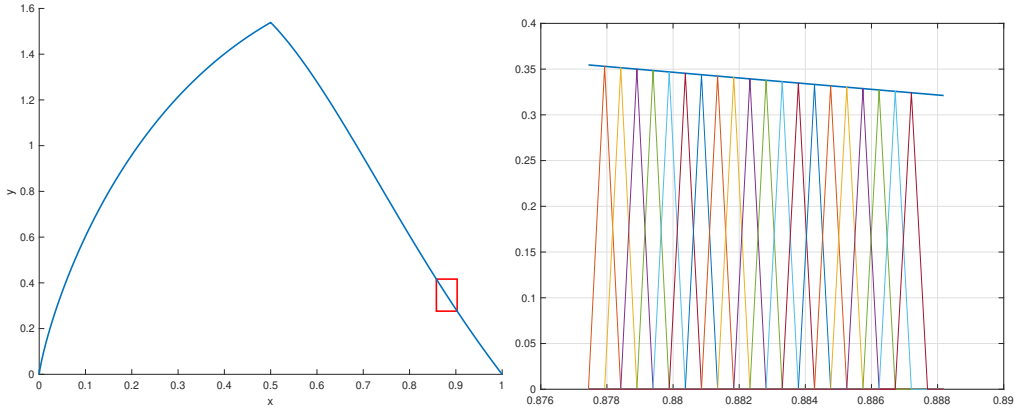
Figure 3.3: Plots of DNS solutions of Burgers equation with different ν values.

Relation between POD basis functions and FEM basis functions

Since the DNS solutions are generated by the *finite element method*, the POD basis functions $\{\varphi_j\}_{i=1}^d$ are interpolated by the FEM basis functions, $\{\phi_i^h\}_{i=1}^N$, i.e.,

$$\varphi_j(x) = \sum_{i=1}^{N} c_{kj}\phi_k^h(x), \qquad \forall j = 1, 2, \cdots, d. \tag{3.83}$$

Figure 3.4 shows that φ_1 is interpolated by finite element shape functions (tent functions) in the red box area. Table 3.4 lists the POD coefficients $\{c_{kj}\}$ in Figure 3.4.



(a) First POD basis function for Burgers equation with $\nu = 10^{-5}$

(b) Piecewise linear FEM shape function interpolation in boxed region. Shape functions $\phi_k^h, k = 1799, \cdots, 1818$ are illustrated.

Figure 3.4: Illustration of POD interpolated by finite element basis function

Table 3.4: The first POD coefficients in the boxed region (Figure 3.4) $c_{1j}, j = 1799, \cdots, 1808$

$c_{1,1799}$	$c_{1,1800}$	$c_{1,1801}$	$c_{1,1802}$	$c_{1,1803}$
$3.5457e-01$	$3.5304e-01$	$3.5151e-01$	$3.4998e-01$	$3.4846e-01$
$c_{1,1804}$	$c_{1,1805}$	$c_{1,1806}$	$c_{1,1807}$	$c_{1,1808}$
$3.4693e-01$	$3.4541e-01$	$3.4389e-01$	$3.4236e-01$	$3.4084e-01$

Different types of ROMs

In the previous chapters, we introduced the *Galerkin ROM* (G-ROM, see section 2.3), *data-driven correction ROM* (DDC-ROM, see chapter 3). For the DDC-ROM, we introduced the

idealized ROM (see section 3.5.3). For the DDC-ROM, we introduced three types of ansatz: *linear, quadratic*, and *cubic* (see section 3.5). To avoid confusion in the nomenclature of different ROMs, the ROM abbreviations are given in Table 3.5.

Table 3.5: Different types of ROMs and their abbreviations.

Type of ROMs	Abbreviation
Galerkin ROM	G-ROM
Idealized Data-driven Correction ROM	IDDC-ROM
Data-driven Correction ROM with a linear ansatz	DDC-ROM-Lin
Data-driven Correction ROM with a quadratic ansatz	DDC-ROM-Qu
Data-driven Correction ROM with a cubic ansatz	DDC-ROM-Cu

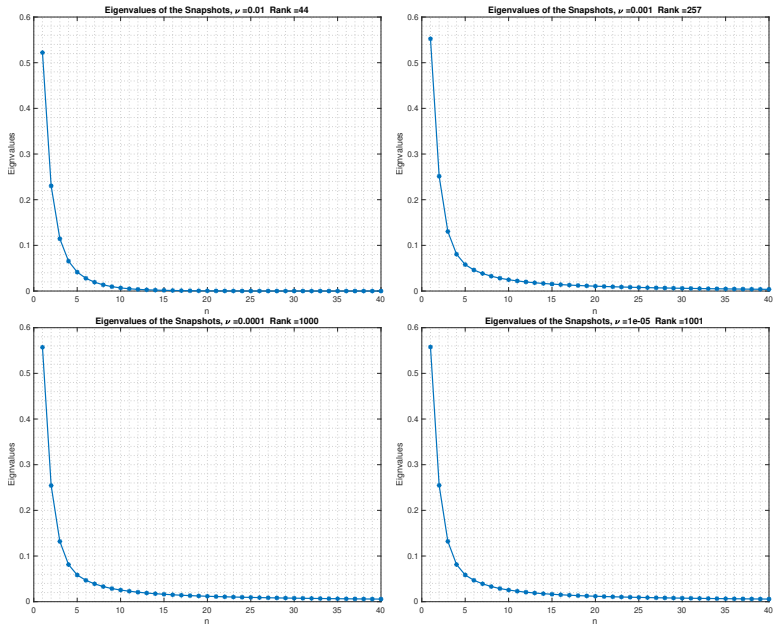
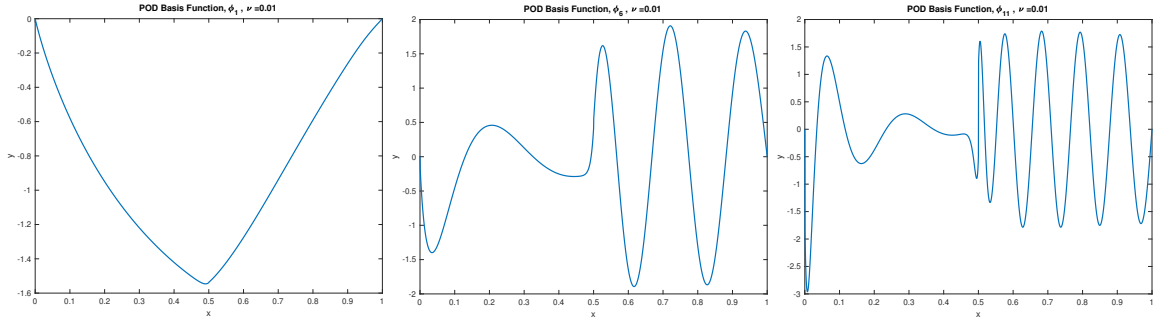


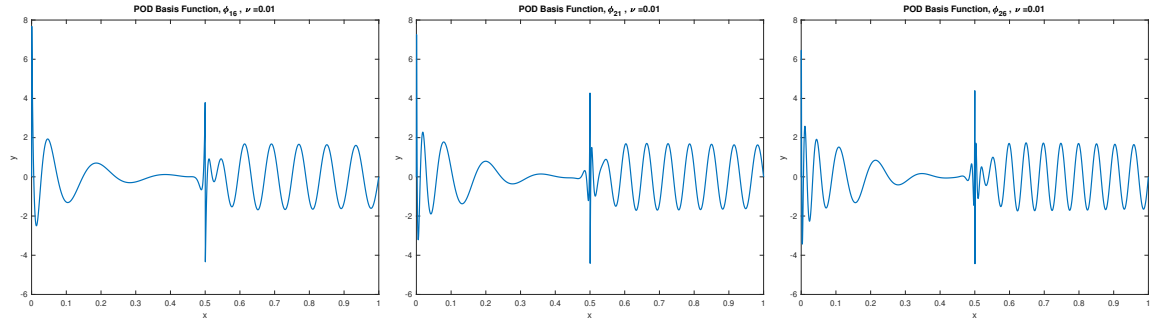
Figure 3.5: Plots of First 40 Eigenvalues of Snapshots with different ν values

Numerical Results

Figures 3.6 to 3.9 illusrate the POD basis functions $\varphi_1, \varphi_6, \varphi_{11}, \varphi_{16}, \varphi_{21}$ and φ_{26} for the one-dimensional Burgers equation with different ν values, i.e., $\nu = 10^{-2}, \nu = 10^{-3}, \nu = 10^{-4}$, and $\nu = 10^{-5}$. Figure 3.5 illustrates the eigenvalues of the *snapshots* for different ν values. Note the different ν may result in different ranks of the snapshots. Although $\tilde{A}, \tilde{B}$, or $\tilde{C}$ are computed in the offline stage, for a three-dimensional fluid problem, their cost can still be

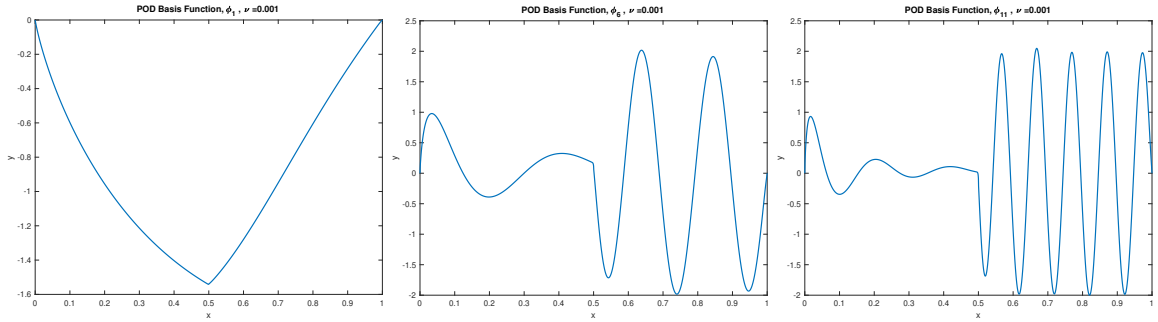


(a) ϕ_1 (b) ϕ_6 (c) ϕ_{11}

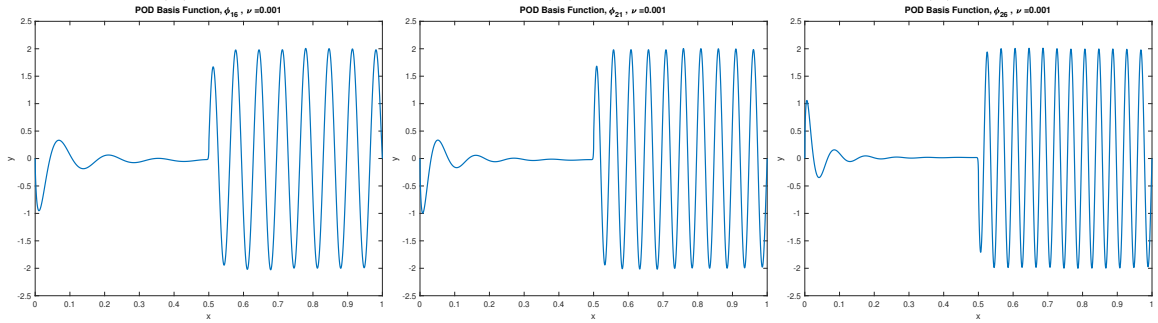


(d) ϕ_{16} (e) ϕ_{21} (f) ϕ_{26}

Figure 3.6: Plots of POD basis functions, *snapshots* for $\nu = 10^{-2}$.

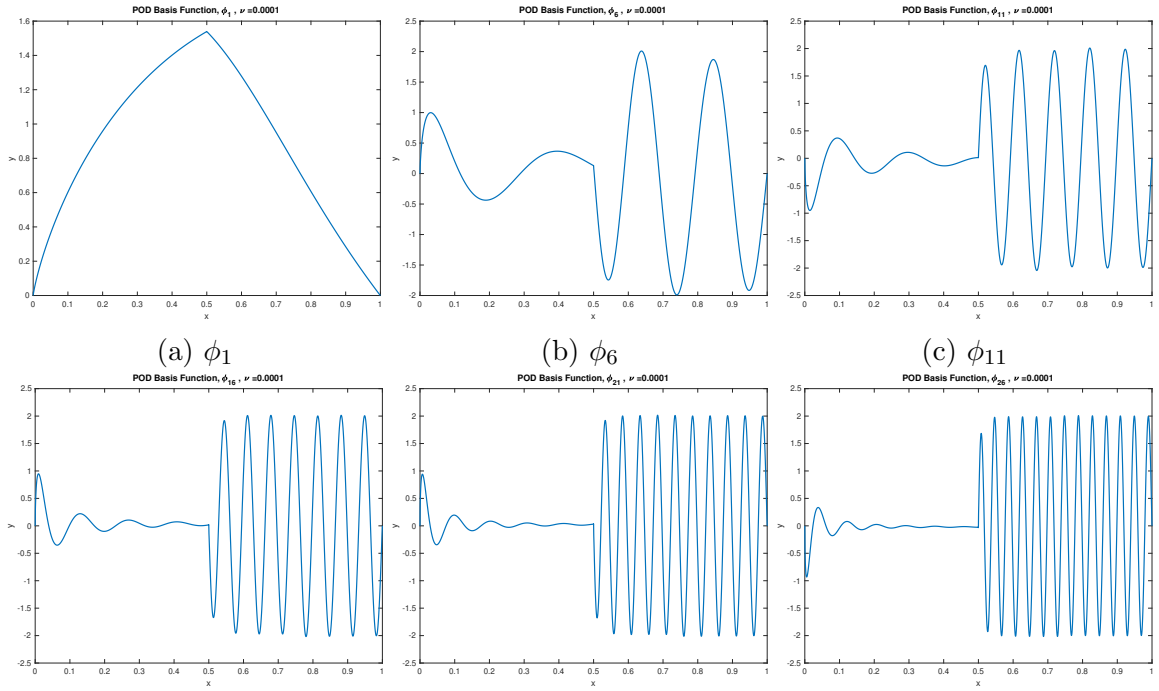


(a) ϕ_1 (b) ϕ_6 (c) ϕ_{11}



(d) ϕ_{16} (e) ϕ_{21} (f) ϕ_{26}

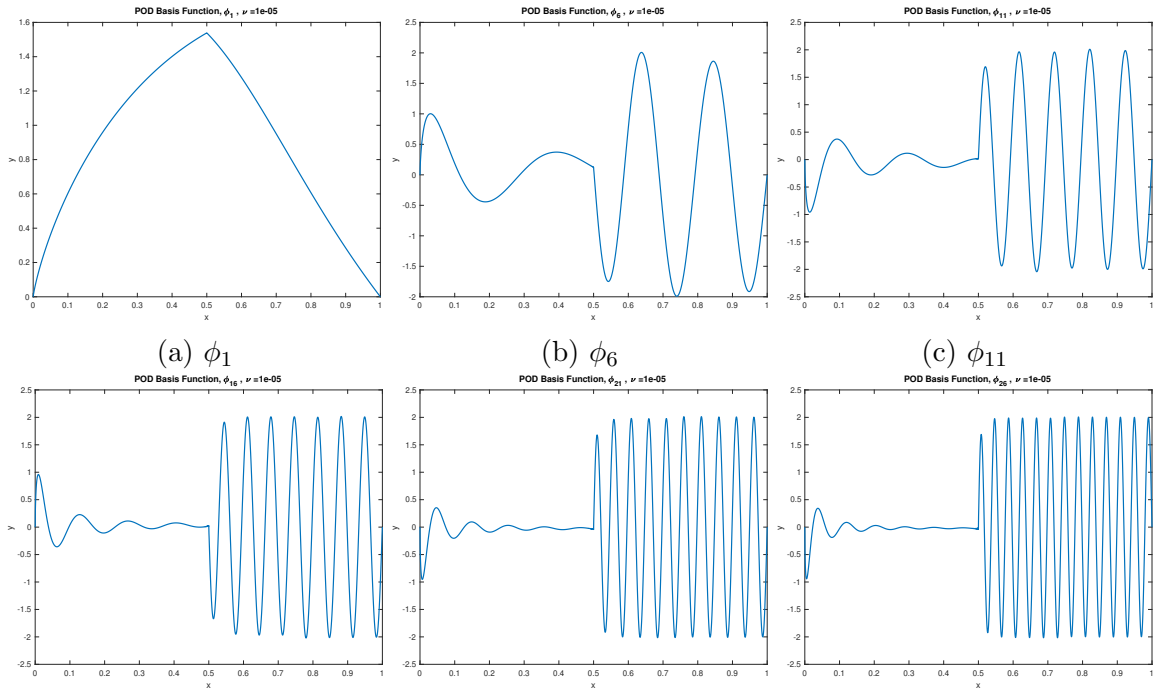
Figure 3.7: Plots of POD basis functions, *snapshots* for $\nu = 10^{-3}$.



(a) ϕ_1 (b) ϕ_6 (c) ϕ_{11}

(d) ϕ_{16} (e) ϕ_{21} (f) ϕ_{26}

Figure 3.8: Plots of POD basis functions, *snapshots* for $\nu = 10^{-4}$.



(a) ϕ_1 (b) ϕ_6 (c) ϕ_{11}

(d) ϕ_{16} (e) ϕ_{21} (f) ϕ_{26}

Figure 3.9: Plots of POD basis functions, *snapshots* for $\nu = 10^{-5}$.

considerable, since the rank of the snapshot matrix can be very large. In [106], a remedy to reduce the offline cost proposed:

$$-\left(\overline{(\mathbf{u_d}\cdot\nabla)\mathbf{u^d}}^r-(\mathbf{u_r}\cdot\nabla)\mathbf{u_r},\varphi_\mathbf{i}\right)\approx-\left(\overline{(\mathbf{u_m}\cdot\nabla)\mathbf{u^m}}^r-(\mathbf{u_r}\cdot\nabla)\mathbf{u_r},\varphi_\mathbf{i}\right),\quad\forall i=1,\cdots,r. \tag{3.84}$$

In equation (3.84), the ROM projection on the space $\mathbf{X}^d$ is approximated by the projection on space $\mathbf{X}^m$, where the parameter m, with the range $r\le m\le d$, is chosen as a compromise between accuracy and efficiency. Note that if $m=r$, then the DDC-ROM becomes the standard *Galerkin ROM*. Tables 3.6 to 3.8, include results for $m=r+1,2r,d$.

We denote the error at each time t_j by $e_j:=u^r(\cdot,t_j)-u^h(\cdot,t_j)$. u^r,u^h denote the ROM and FEM solutions, respectively. We consider the error in the $L^2(0,T;L^2(\Omega))$, which is defined as

$$\mathcal{E}_{L^2(L^2)}=\sqrt{\frac{1}{M+1}\sum_{0\le j\le M}\|e_j\|^2_{L^2(\Omega)}}. \tag{3.85}$$

The errors of the DDC-ROM with different ansatz and Galerkin ROM for the Burgers equation with $\nu=10^{-3}$ and $r=6,11,20$ are listed in Table 3.6; the errors of DDC-ROMs with different ansatz and Galerkin ROM for Burgers equation $\nu=10^{-4}$ with $r=6,11,20$ is listed in Table 3.7; and the errors of DDC-ROMs with different ansatz and Galerkin ROM for Burgers equation $\nu=10^{-5}$ with $r=6,11,20$ is listed in Table 3.8. Figures 3.10 to 3.18 plot the G-ROM and DDC-ROMs with different ν and r values.

The G-ROM with low r values performs poorly for small ν values even though it is cheap computationally. On the other hand, the DDC-ROM family (including DDC-ROM-Lin, DDC-ROM-Qu, DDC-ROM-Cu) are much more accurate than the G-ROM at low r values. Indeed, in general, the DDC-ROM with a quadratic ansatz and with a cubic ansatz perform better than the DDC-ROM with a linear ansatz; unexpectedly, the IDDC-ROM does not perform better than these two DDC-ROMs. On the other hand, DDC-ROM-Cu does not necessarily perform better than DDC-ROM-Qu; this might be due to the fact that a cubic ansatz in the model yields more numerical oscillations than a quadratic ansatz. In addition, it is worth noting that when $r\le m<<d$, the IDDC-ROM performs poorly even though it applies the true ansatz while at the same time, the DDC-ROM with only a part of the true ansatz works much better.

Table 3.6: DDC-ROMs with different ansatzs and G-ROM errors for the Burgers equation with $\nu = 10^{-3}$, using different r and m values.

ROM Type	Proj Space	$r = 6$	$r = 11$	$r = 20$
G-ROM	X^r	$2.1526e-01$	$1.3780e-01$	$5.8418e-02$
DDC-ROM-Lin	X^{r+1}	$1.3010e-01$	$8.5145e-02$	$4.4412e-02$
	X^{2r}	$9.5121e-02$	$5.8310e-02$	$3.2566e-02$
	X^d	$9.0316e-02$	$5.6793e-02$	$3.2324e-02$
DDC-ROM-Qu	X^{r+1}	$1.1635e-01$	$7.8380e-02$	$4.2608e-02$
	X^{2r}	$8.9336e-02$	$5.4921e-02$	$3.1416e-02$
	X^d	$8.5970e-02$	$5.4547e-02$	$3.1374e-02$
DDC-ROM-Cu	X^{r+1}	$1.1415e-01$	$7.7210e-02$	$4.2332e-02$
	X^{2r}	$8.7306e-02$	$5.4923e-02$	$3.1490e-02$
	X^d	$8.6040e-02$	$5.4578e-02$	$3.1441e-02$
IDDC-ROM	X^{r+1}	$1.3521e-01$	$1.0132e-01$	$4.7934e-02$
	X^{2r}	$9.3477e-02$	$5.7001e-02$	$3.1666e-02$
	X^d	$8.5997e-02$	$5.4617e-02$	$3.1464e-02$

Table 3.7: DDC-ROMs with different ansatzs and G-ROM errors for the Burgers equation with $\nu = 10^{-4}$, using different r and m values.

ROM Type	Proj Space	$r = 6$	$r = 11$	$r = 20$
G-ROM	X^r	$2.7233e-01$	$2.8500e-01$	$1.7350e-01$
DDC-ROM-Lin	X^{r+1}	$1.5744e-01$	$1.3169e-01$	$9.7138e-02$
	X^{2r}	$1.0852e-01$	$7.3679e-02$	$5.0081e-02$
	X^d	$9.9837e-02$	$6.9020e-02$	$4.7888e-02$
DDC-ROM-Qu	X^{r+1}	$1.4002e-01$	$1.1697e-01$	$9.3786e-02$
	X^{2r}	$9.9025e-02$	$6.8411e-02$	$4.7656e-02$
	X^d	$9.5316e-02$	$6.6767e-02$	$4.6782e-02$
DDC-ROM-Cu	X^{r+1}	$1.4027e-01$	$1.1526e-01$	$9.2114e-02$
	X^{2r}	$9.8252e-02$	$6.8362e-02$	$4.7677e-02$
	X^d	$9.5347e-02$	$6.6803e-02$	$4.6826e-02$
IDDC-ROM	X^{r+1}	$1.7283e-01$	$2.4185e-01$	$1.7148e-01$
	X^{2r}	$1.1415e-01$	$8.8169e-02$	$5.8319e-02$
	X^d	$9.5352e-02$	$6.6905e-02$	$4.7174e-02$

Table 3.8: DDC-ROMs with different ansatzs and G-ROM errors for the Burgers equation with $\nu = 10^{-5}$, using different r and m values.

ROM Type	Proj Space	$r = 6$	$r = 11$	$r = 20$
G-ROM	X^r	$2.7887e-01$	$3.0715e-01$	$2.0690e-01$
DDC-ROM-Lin	X^{r+1}	$1.6063e-01$	$1.3904e-01$	$1.0820e-01$
	X^{2r}	$1.1006e-01$	$7.5525e-02$	$5.2334e-02$
	X^d	$1.0084e-01$	$7.0308e-02$	$4.9611e-02$
DDC-ROM-Qu	X^{r+1}	$1.4123e-01$	$1.2248e-01$	$1.0451e-01$
	X^{2r}	$1.1569e-01$	$6.9951e-02$	$4.9639e-02$
	X^d	$9.6287e-02$	$6.8064e-02$	$4.8510e-02$
DDC-ROM-Cu	X^{r+1}	$1.4338e-01$	$1.2074e-01$	$1.0259e-01$
	X^{2r}	$9.9446e-02$	$6.9895e-02$	$4.9654e-02$
	X^d	$9.6324e-02$	$6.8101e-02$	$4.8555e-02$
IDDC-ROM	X^{r+1}	$1.7736e-01$	$2.7001e-01$	$2.1821e-01$
	X^{2r}	$1.1685e-01$	$9.5269e-02$	$6.6680e-02$
	X^d	$9.6329e-02$	$6.8213e-02$	$4.8982e-02$

The following plots are the G-ROM and DDC-ROM results: the left column is the G-ROM with $r = 6, 11, 20, 36, 62$, while the right column is the DDC-ROM with the same r values.

3.7 Summary

In this chapter, we proposed a DDC-ROM framework for the numerical simulation of general nonlinear PDEs. This framework is based on the ROM projection and solving a optimization problem. Since the projection and optimization steps are done in the *offline* stage, the *online* stage of the DDC-ROM actually takes a similar amount of CPU time as the G-ROM, which ensures its efficiency. We numerically investigated the DDC-ROM in the simulation of the one-dimensional Burgers equation. We compared the errors of G-ROM with the family of DDC-ROMs, including DDC-ROM with a linear ansatz, DDC-ROM with a quadratic ansatz, DDC-ROM with a cubic ansatz, and Idealized DDC-ROM.

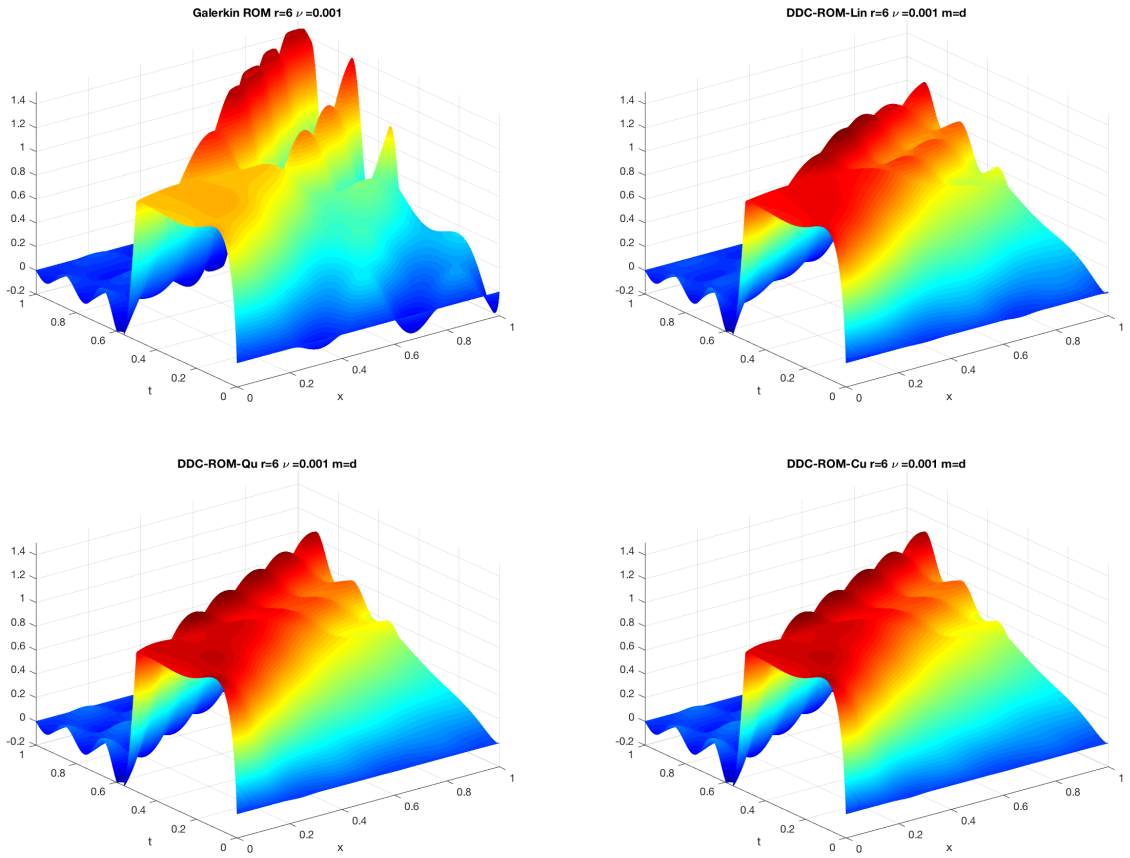


Figure 3.10: Plots of G-ROM (top-left), DDC-ROM-Lin (top-right),DDC-ROM-Qu (bottom-left), and DDC-ROM-Cu (bottom-right), for $\nu = 10^{-3}$ and $r = 6$.

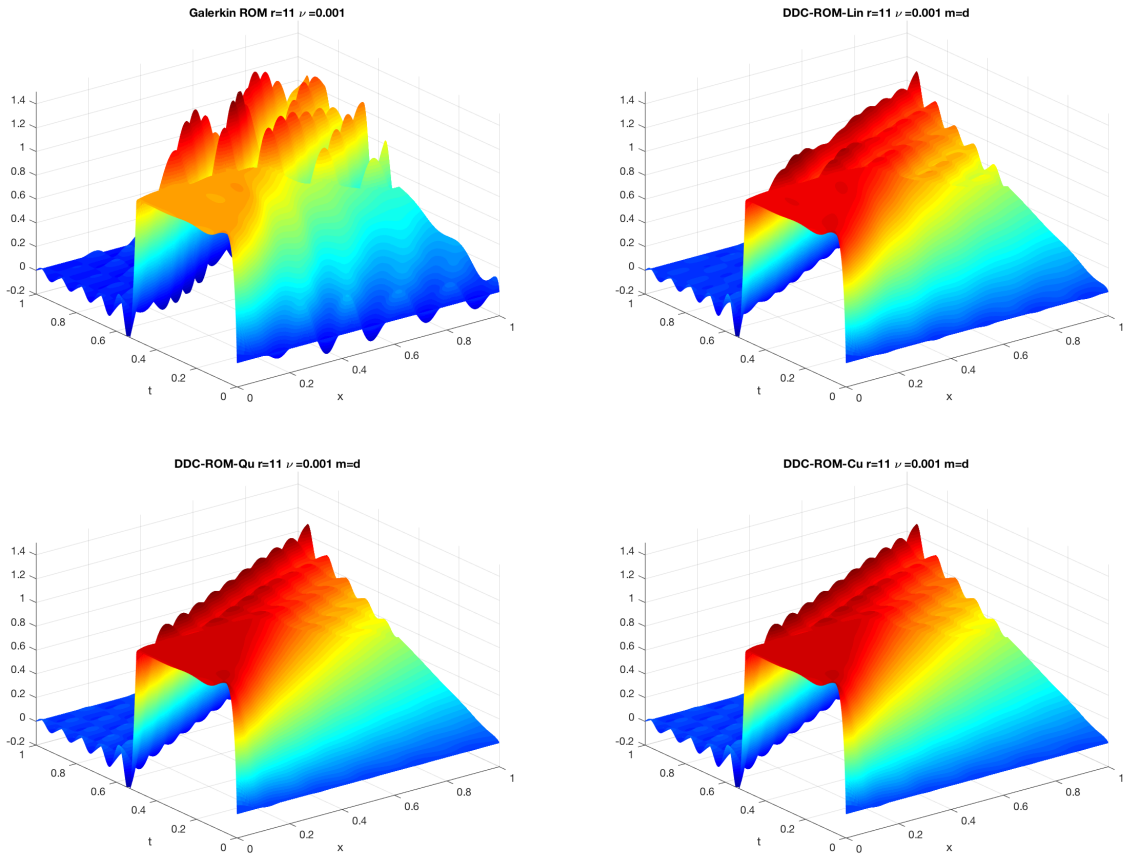


Figure 3.11: Plots of G-ROM (top-left), DDC-ROM-Lin (top-right),DDC-ROM-Qu (bottom-left), and DDC-ROM-Cu (bottom-right), for $\nu = 10^{-3}$ and $r = 11$.

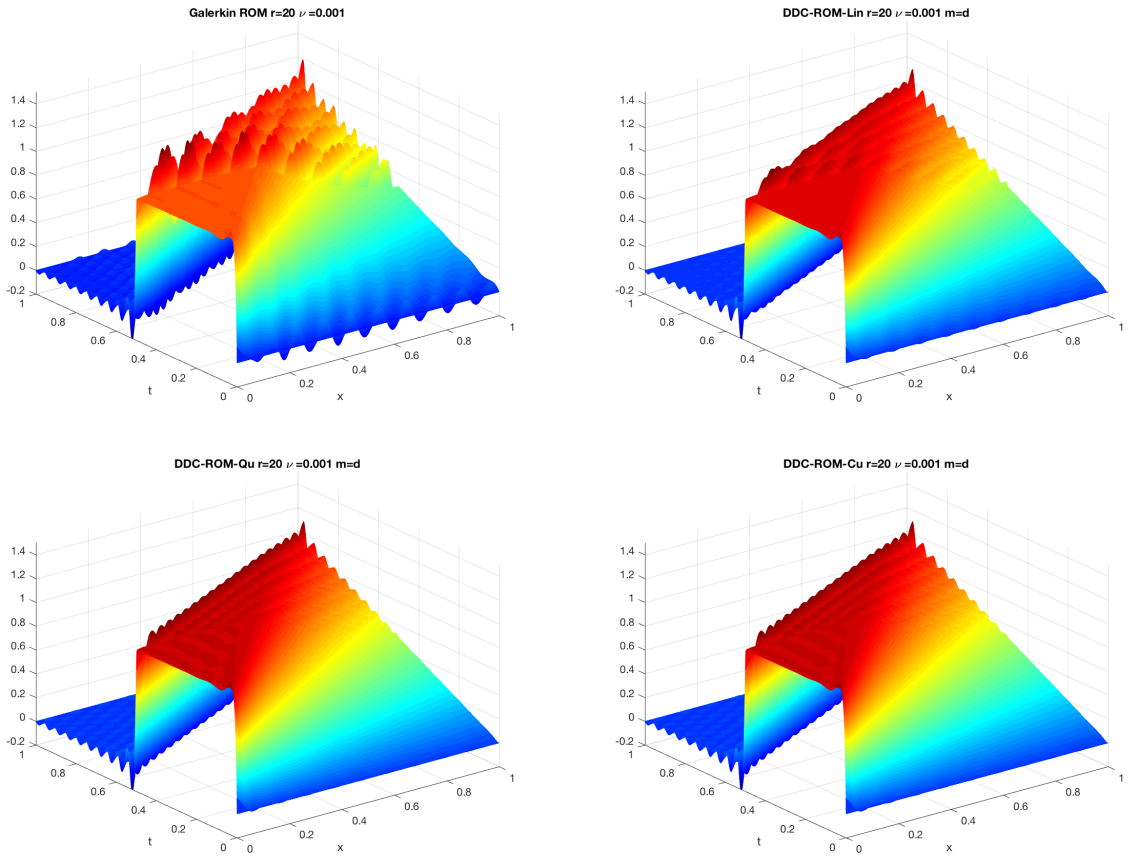


Figure 3.12: Plots of G-ROM (top-left), DDC-ROM-Lin (top-right),DDC-ROM-Qu (bottom-left), and DDC-ROM-Cu (bottom-right), for $\nu = 10^{-3}$ and $r = 20$.

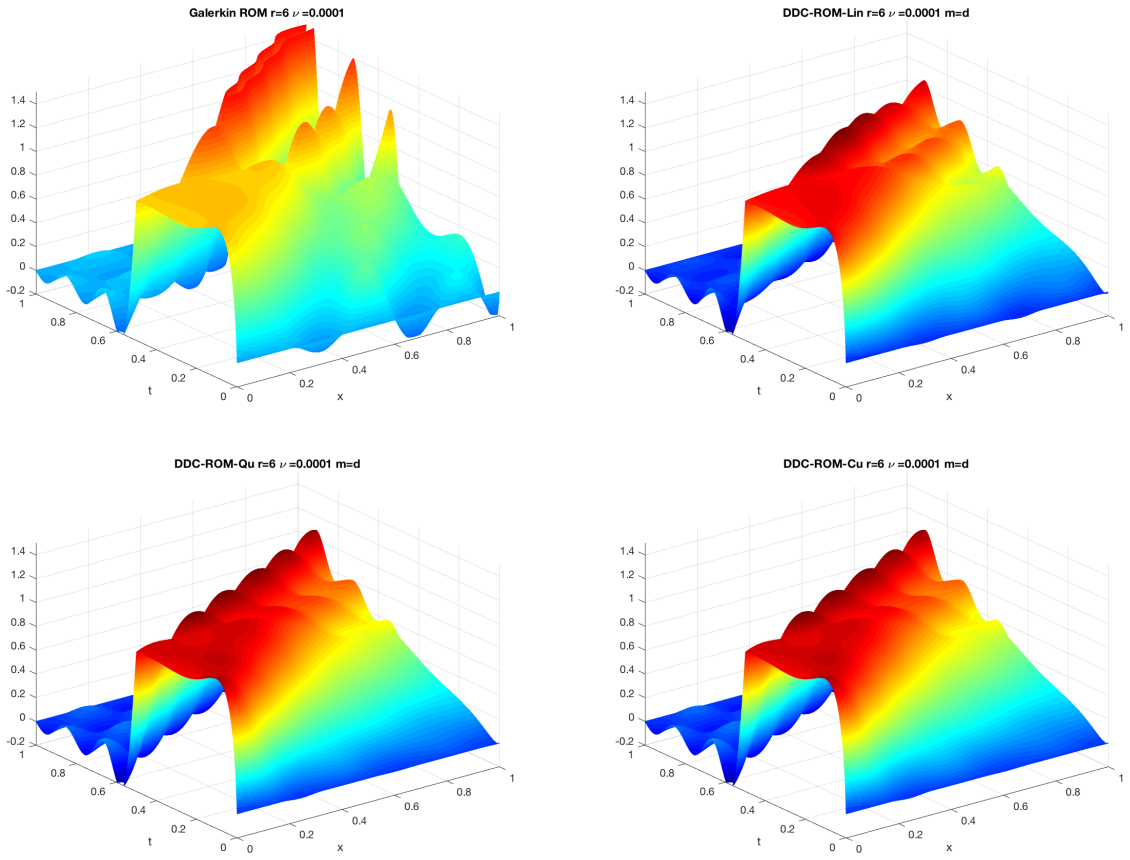


Figure 3.13: Plots of G-ROM (top-left), DDC-ROM-Lin (top-right),DDC-ROM-Qu (bottom-left), and DDC-ROM-Cu (bottom-right), for $\nu = 10^{-4}$ and $r = 6$.

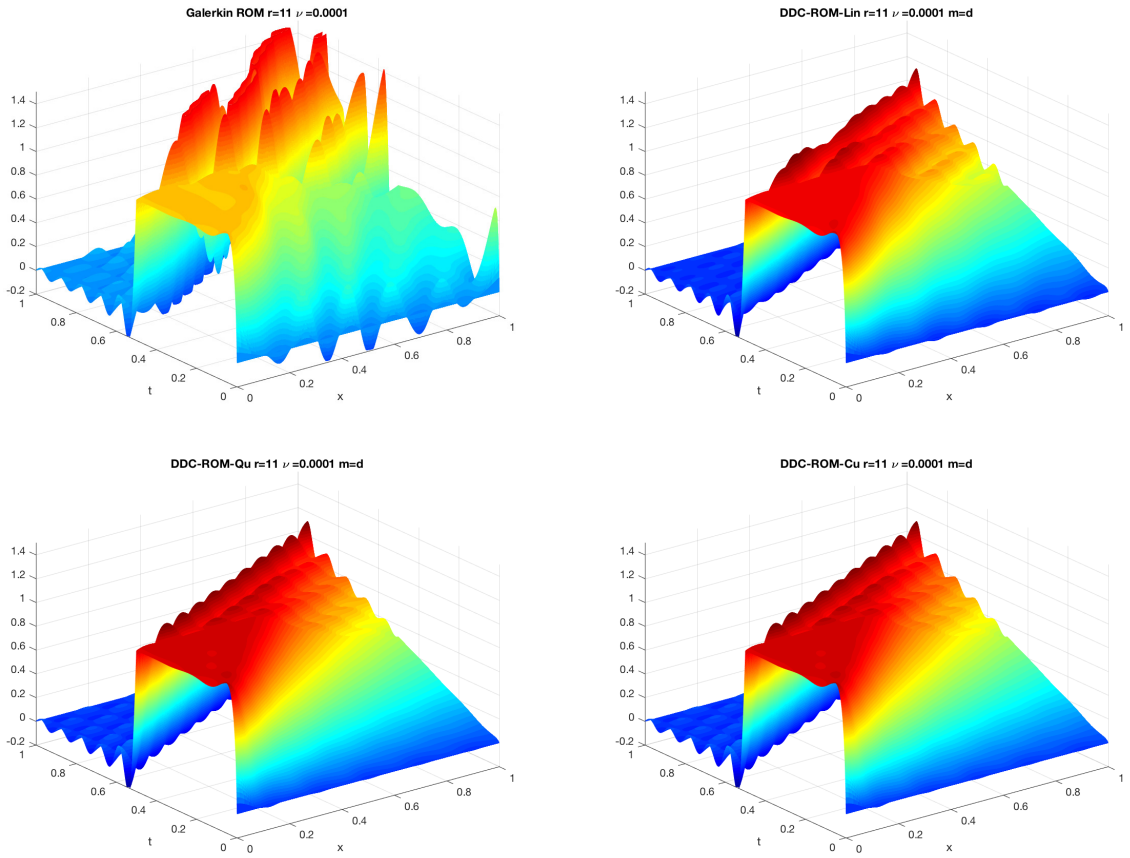


Figure 3.14: Plots of G-ROM (top-left), DDC-ROM-Lin (top-right),DDC-ROM-Qu (bottom-left), and DDC-ROM-Cu (bottom-right), for $\nu = 10^{-4}$ and $r = 11$.

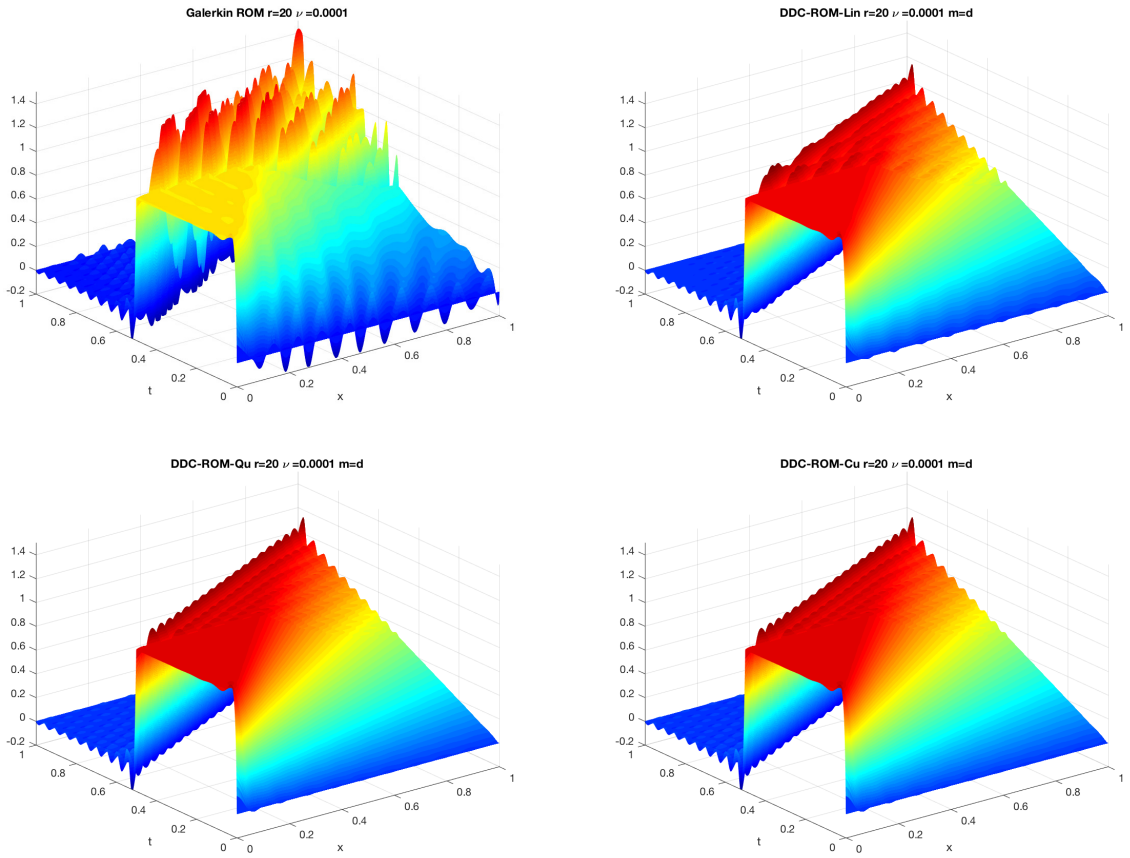


Figure 3.15: Plots of G-ROM (top-left), DDC-ROM-Lin (top-right),DDC-ROM-Qu (bottom-left), and DDC-ROM-Cu (bottom-right), for $\nu = 10^{-4}$ and $r = 20$.

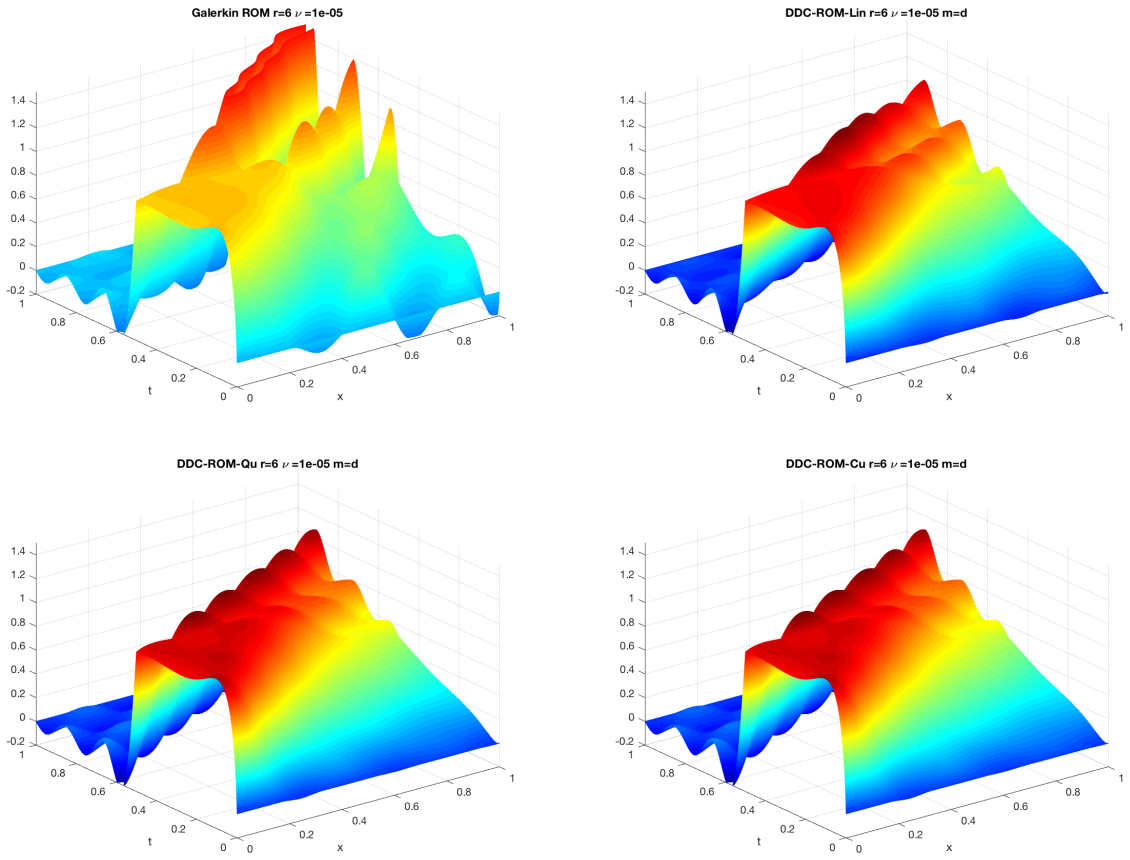


Figure 3.16: Plots of G-ROM (top-left), DDC-ROM-Lin (top-right),DDC-ROM-Qu (bottom-left), and DDC-ROM-Cu (bottom-right), for $\nu = 10^{-5}$ and $r = 6$.

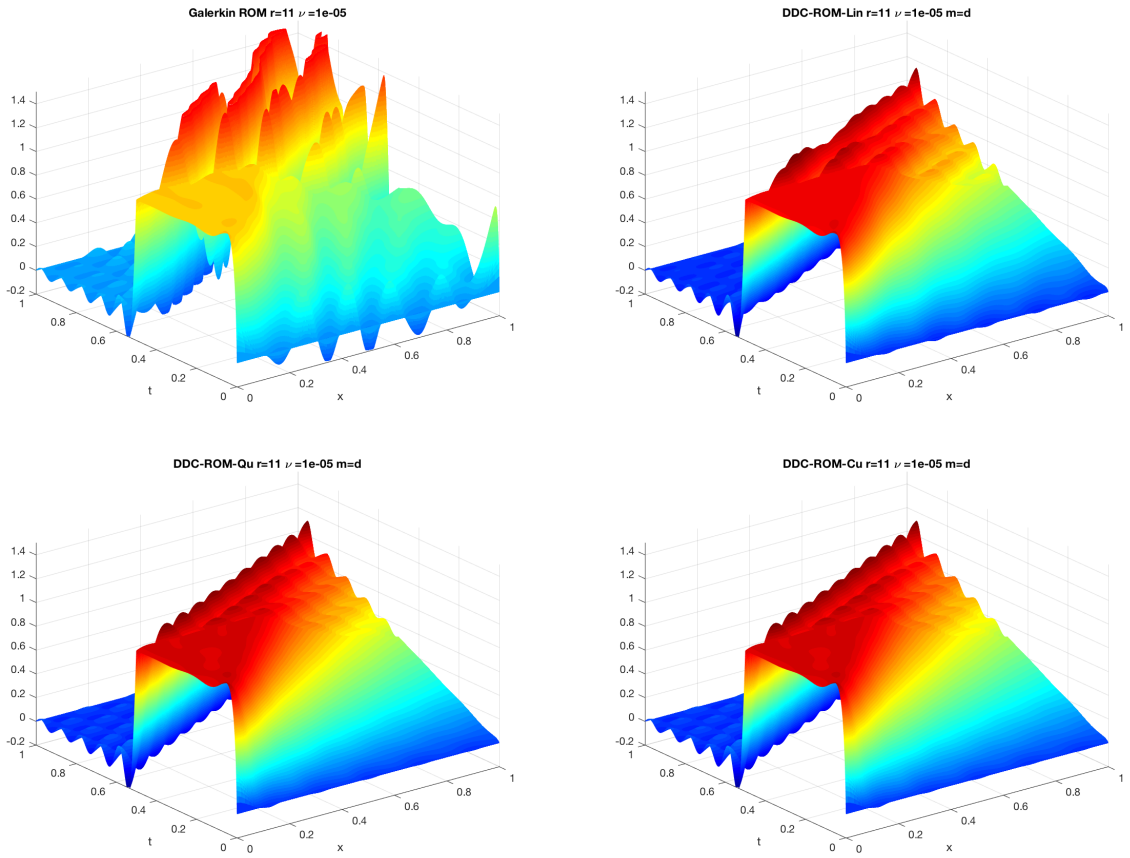


Figure 3.17: Plots of G-ROM (top-left), DDC-ROM-Lin (top-right),DDC-ROM-Qu (bottom-left), and DDC-ROM-Cu (bottom-right), for $\nu = 10^{-5}$ and $r = 11$.

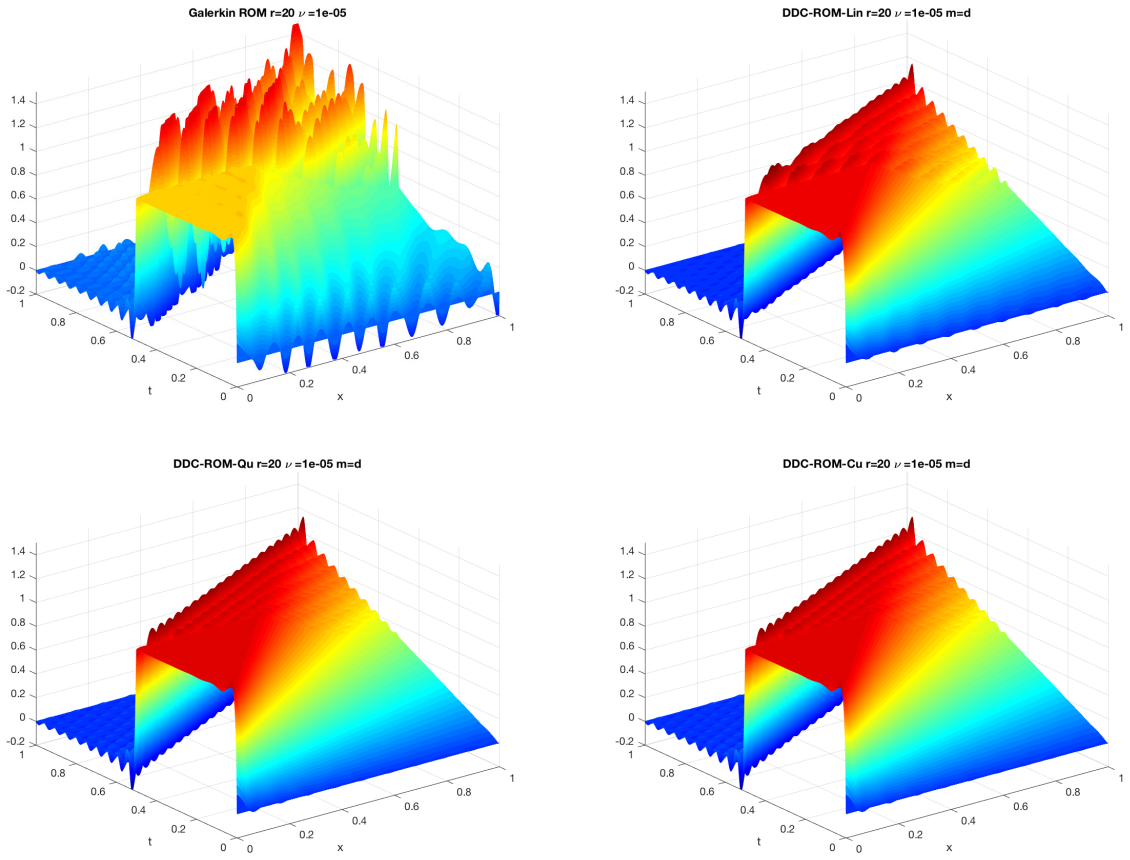


Figure 3.18: Plots of G-ROM (top-left), DDC-ROM-Lin (top-right),DDC-ROM-Qu (bottom-left), and DDC-ROM-Cu (bottom-right), for $\nu = 10^{-5}$ and $r = 20$.

Chapter 4

Cross-Validation of DDC-ROMs

In this chapter, we perform a *cross-validation* of the DDC-ROMs introduced in the previous chapters. That is we investigate whether the operator $\tilde{A}$ and $\tilde{B}$ can be used in parameter settings that are different from the settings used in their training stage.

4.1 Models and Algorithms

The *holdout method* is one of the the simplest kinds of *cross-validation* [3, 50, 88]. This method separates the data into two sets, the training set and the testing set. With the training set of data, we generate the sets, $\{\tilde{A}_k^\dagger\}_{k=1}^s$ and $\{\tilde{B}_k^\dagger\}_{k=1}^s$. Then we apply these two sets to the τ terms to predict the output values in the testing set. The errors are accumulated as before to give the mean absolute test set error, which is used to evaluate the model. The ultimate goal behind this method is to investigate the potential universality of the ansatz τ in the DDC-ROM.

Suppose that a set of FOM solutions is prepared as a database prior to the construction of the ROM with different Reynolds number values ($\mu = \frac{1}{Re}$)[9, 41, 84]. The database can be represented as

$$\mathcal{D} = \{\{(t,\mu), \mathbf{u}_d(t,\mu)\} : (t,\mu) \in \mathcal{T}_M \times \mathcal{P}_s\}, \tag{4.1}$$
$$\mathcal{T}_M = \{t_1, t_2, \cdots, t_M\}, \tag{4.2}$$
$$\mathcal{P}_s = \{\mu_1, \mu_2, \cdots, \mu_s\}. \tag{4.3}$$

In order to cross-validate the model, we separate the set $\mathcal{P}_s$ into two subsets, the training set, $\mathcal{P}_p$, and the test set, $\mathcal{P}_q$:

$$\mathcal{P}_p = \{\mu_{p_1}, \mu_{p_2}, \cdots, \mu_{p_{s_1}}\} \qquad \text{and} \qquad \mathcal{P}_q = \{\mu_{q_1}, \mu_{q_2}, \cdots, \mu_{q_{s_2}}\}, \tag{4.4}$$

with $s_1 + s_2 = s$. Now the data base has two counterparts:

$$\mathcal{D}_{\text{training}} = \{\{(t,\mu), \mathbf{u}_d(t,\mu)\} : (t,\mu) \in \mathcal{T}_M \times \mathcal{P}_p\}, \tag{4.5}$$
$$\mathcal{D}_{\text{test}} = \{\{(t,\mu), \mathbf{u}_d(t,\mu)\} : (t,\mu) \in \mathcal{T}_M \times \mathcal{P}_q\}, \tag{4.6}$$

which satisfy the relation $\mathcal{D}_{\text{training}} \cup \mathcal{D}_{\text{test}} = \mathcal{D}$ and $\mathcal{D}_{\text{training}} \cap \mathcal{D}_{\text{test}} = \emptyset$.

In the cross-validation, we can replace the DDC-ROM for the NSE with

$$\dot{\mathbf{a}} = (\mathbf{A} + \tilde{\mathbf{A}}^{\dagger})\mathbf{a} + \mathbf{a}^T(\mathbf{B} + \tilde{\mathbf{B}}^{\dagger})\mathbf{a}, \tag{4.7}$$

where $\mathbf{A}, \mathbf{A}$ are generated in $\mathcal{D}_{\text{test}}$ and $\tilde{\mathbf{A}}^{\dagger}, \tilde{\mathbf{B}}^{\dagger}$ are generated in $\mathcal{D}_{\text{training}}$.

4.2 Numerical Experiments

Pattern Recognition

To illustrate the pattern of the $\tau \in X^r$ terms in the Burgers equation, we can project the calculated τ back into the X^h space, i.e.,

$$\tau^h(t_j) = \sum_{i=1}^{N} \left(\overline{(\mathbf{u}_d(t_j)\cdot\nabla)\mathbf{u}_d(t_j)}^{r} - (\mathbf{u}_r(t_j)\cdot\nabla)\mathbf{u}_r(t_j), \phi_i^h \right), \quad \text{where} \quad \phi_i^h \in \mathbf{X}^h. \tag{4.8}$$

Figures 4.1 to 4.3 illustrate the time-space pattern of τ with different ν and r values. For each subplot, from top to bottom, $\nu = 10^{-2}, 10^{-3}, 10^{-4}$, and 10^{-5}, respectively. we observe that for fixed r values, different ν will still have the similar patterns.

Errors for Cross-Validation

In the cross-validation of the DDC-ROM, we consider the one-dimensional Burgers equation (3.79) as a test case. The initial condition and the boundary conditions are described in section 3.6. In this section, we consider only the DDC-ROM with a quadratic ansatz; thus DDC-ROM in this section refers to DDC-ROM-Qu if not otherwise specified.

For simplicity, we choose $\mathcal{P}_p = \{10^{-2}, 10^{-3}, 10^{-4}\}$ as the training set. Instead applying deep learning techniques in constructing the ansatz for specific ν values, we simply use the individual quadratic ansatz from the training dataset for specific ν values.

Tables 4.1 to 4.3 show the cross-validation errors, which are defined in equation (3.85). We observe that for comparatively large ν values, the ansatz from the training data does not provide an accurate result for small ν values, since at these ν values the G-ROM yields satisfactory results. We also note that the training data for $\nu = 10^{-3}$ represents a turning

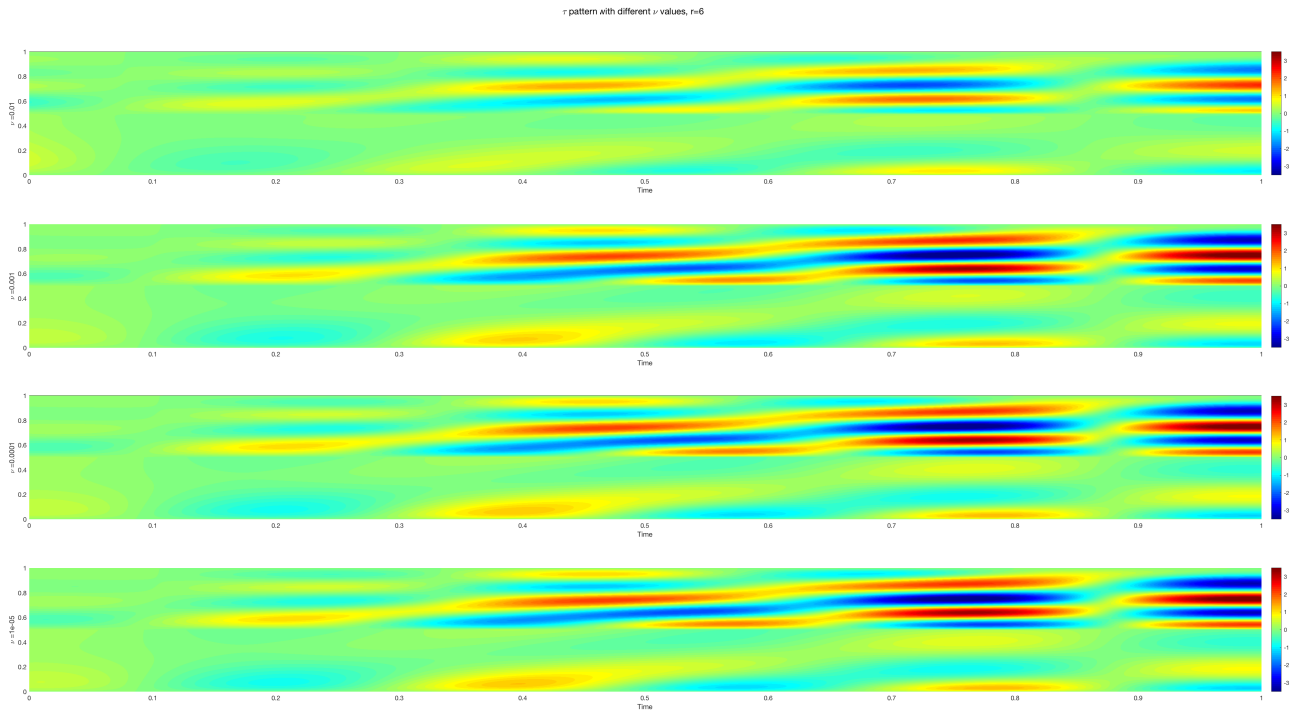


Figure 4.1: Space-time patterns of τ for the Burgers equation with different viscosities, $\nu = 10^{-2}, 10^{-3}, 10^{-4}$, and 10^{-5}, and $r = 6$.

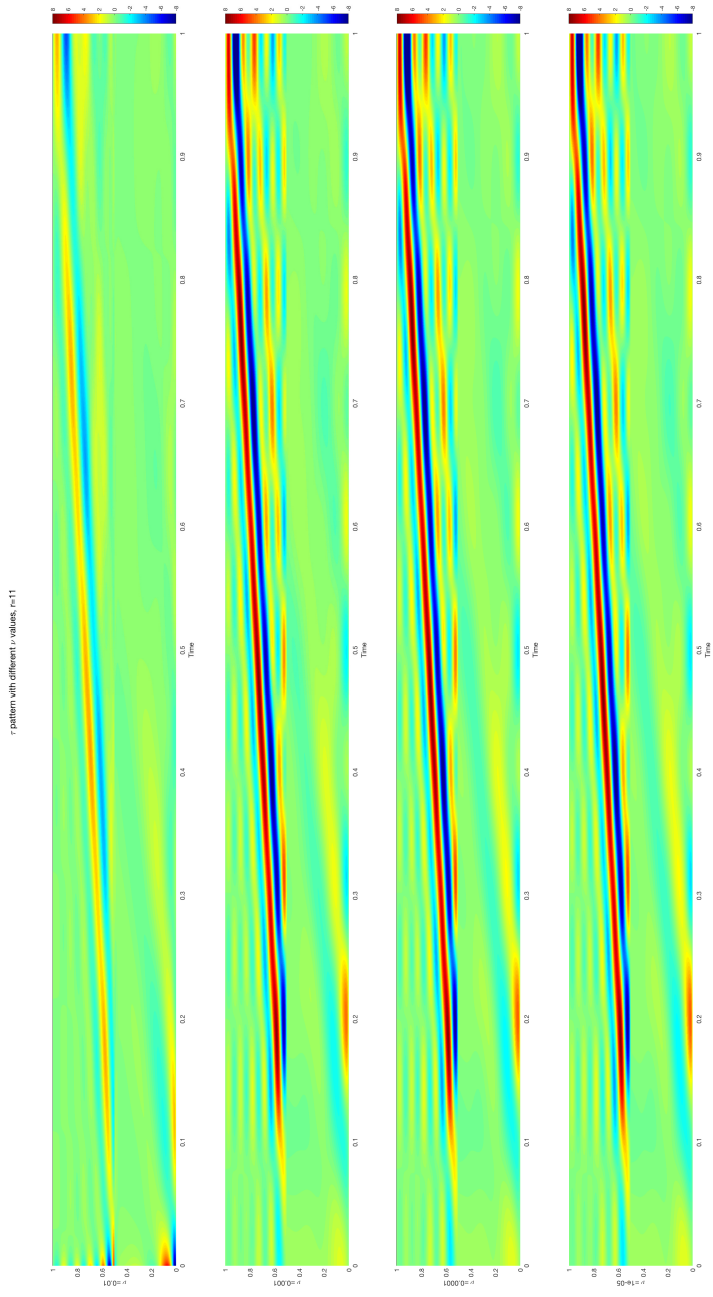


Figure 4.2: Space-time patterns of τ for the Burgers equation with different viscosities, $\nu = 10^{-2}, 10^{-3}, 10^{-4}$, and 10^{-5}, and $r = 11$.

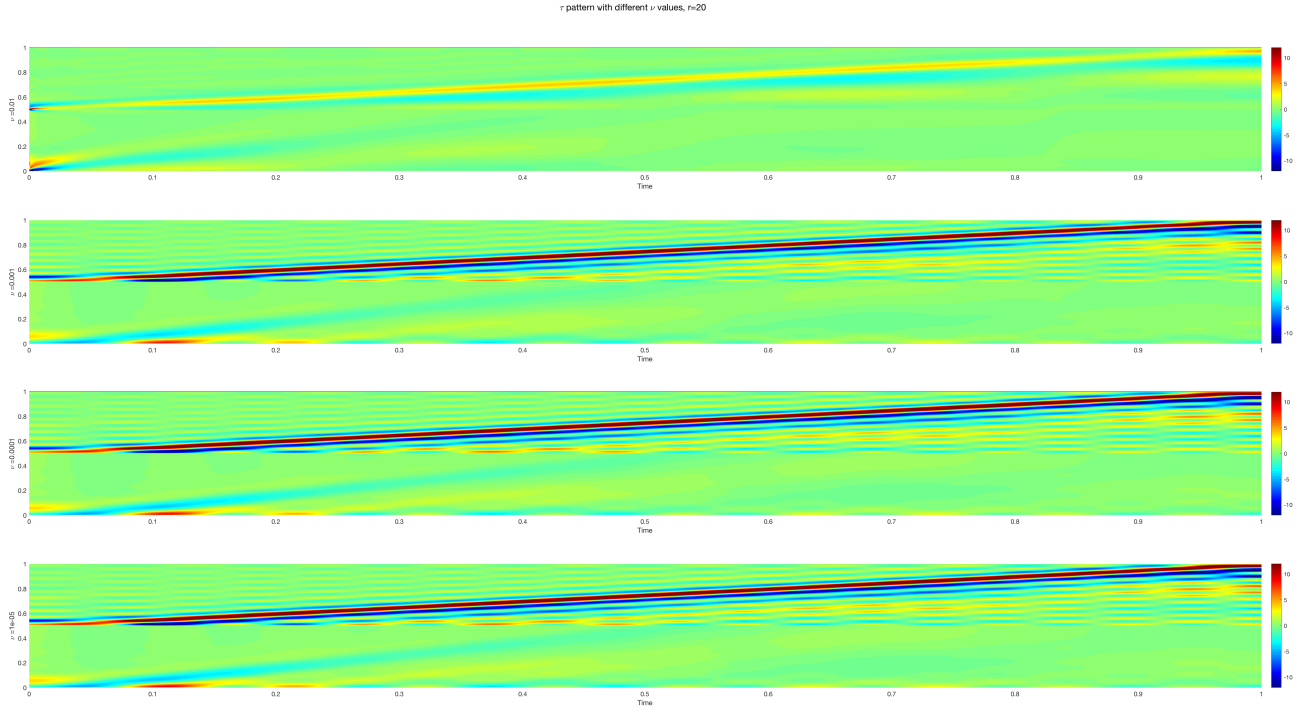


Figure 4.3: Space-time patterns of τ for the Burgers equation with different viscosities, $\nu = 10^{-2}, 10^{-3}, 10^{-4}$, and 10^{-5}, and $r = 20$.

point for the parameter ν: for ν values that are lower than $\mu = 10^{-3}$, the DDC-ROM's accuracy degrades.

Table 4.1: Cross-validation errors of DDC-ROM for the Burgers equation, ansatz generated by training data for $\nu = 10^{-2}$.

	$\nu = 9 \times 10^{-3}$		$\nu = 8 \times 10^{-3}$	
r	G-ROM	DDC-ROM	G-ROM	DDC-ROM
6	$3.8464e-02$	$3.5392e-02$	$4.6190e-02$	$3.8977e-02$
11	$8.4613e-03$	$8.4219e-03$	$1.0077e-02$	$1.0062e-02$
20	$7.49720e-04$	$7.4945e-04$	$1.1119e-03$	$1.1118e-03$
36	$9.2505e-06$	$9.2505e-06$	$1.9661e-05$	$1.9661e-05$
	$\nu = 5 \times 10^{-3}$		$\nu = 10^{-3}$	
r	G-ROM	DDC-ROM	G-ROM	DDC-ROM
6	$8.4069e-02$	$7.3935e-02$	$2.1525e-01$	$1.4262e-01$
11	$2.3106e-02$	$2.1679e-02$	$1.3780e-01$	$9.5911e-02$
20	$4.5967e-03$	$4.5943e-03$	$5.8417e-02$	$5.8317e-02$
36	$2.6564e-04$	$2.6564e-04$	$1.9581e-02$	$1.9581e-02$

Table 4.2: Cross-validation errors of DDC-ROM for the Burgers equation, ansatz generated by training data for $\nu = 10^{-3}$.

	$\nu = 9 \times 10^{-4}$		$\nu = 8 \times 10^{-4}$	
r	G-ROM	DDC-ROM	G-ROM	DDC-ROM
6	$2.2089e-01$	$1.01006e-01$	$2.2670e-01$	$1.0214e-01$
11	$1.4765e-01$	$6.1754e-02$	$1.5884e-01$	$6.268e-02$
20	$6.4068e-02$	$3.433e-02$	$7.0507e-02$	$3.7385e-02$
36	$2.2542e-02$	$1.6314e-02$	$2.6034e-02$	$1.8079e-02$
	$\nu = 5 \times 10^{-4}$		$\nu = 10^{-4}$	
r	G-ROM	DDC-ROM	G-ROM	DDC-ROM
6	$2.4517e-01$	$1.0567e-01$	$2.7233e-01$	$6.4134e-01$
11	$2.0261e-01$	$6.7630e-02$	$2.8500e-01$	$6.1542e+00$
20	$9.6933e-02$	$4.1322e-02$	$1.7350e-01$	$7.9359e-01$
36	$4.1216e-02$	$2.4555e-02$	$8.9464e-02$	$7.3487e-02$

Table 4.3: Cross-validation errors of DDC-ROM for the Burgers equation, ansatz generated by training data for $\nu = 10^{-4}$.

	$\nu = 9 \times 10^{-5}$		$\nu = 8 \times 10^{-5}$	
r	G-ROM	DDC-ROM	G-ROM	DDC-ROM
6	$2.7304e-01$	$9.5553e-02$	$2.7376e-01$	$9.5661e-02$
11	$2.8734e-01$	$6.7030e-02$	$2.8971e-01$	$6.7174e-02$
20	$1.7678e-01$	$4.8578e-02$	$1.8017e-01$	$4.9438e-02$
36	$9.1711e-02$	$3.3120e-02$	$9.4059e-02$	$3.3326e-02$
	$\nu = 5 \times 10^{-5}$		$\nu = 5 \times 10^{-5}$	
r	G-ROM	DDC-ROM	G-ROM	DDC-ROM
6	$2.7594e-01$	$9.5987e-02$	$2.7886e-01$	$9.6425e-02$
11	$2.9696e-01$	$6.7607e-02$	$3.0714e-01$	$6.8190e-02$
20	$1.9106e-01$	$4.8246e-02$	$2.0690e-01$	$4.94991e-02$
36	$1.0178e-01$	$3.4132e-02$	$1.1411e-01$	$3.5420e-02$

Comparisons of Cross-Validation with Different Type of Ansatz

Inspired by the existence of a turning point for the parameter of ν, we performed a careful numerical comparison for different types of ansatz in the cross-validation. We use the Burgers equation with ν in the key region between 10^{-4} and 10^{-3}. The errors are shown in Tables 4.4 and 4.5. The "diverged" entry in the table means that the Newton solver diverges at some time step. We observe a shifting between training data and testing data for the ν values between $\nu = 10^{-3}$ and 10^{-4}, where the ansatz cannot be interchanged if the ν values for training data and testing data differ too much. The numerical results also suggest that cross-validation of the DDC-ROM with a linear ansatz is the most robust model of the DDC-ROM family.

4.3 Summary

In this chapter, we propose a cross-validation framework for the τ terms in the DDC-ROM. We perform some numerical tests for some chosen training data that includes the ansatz from chosen ν values. We perform the cross-validation in the simulation of the one-dimensional Burgers equation. The numerical results shows that the cross-validation of DDC-ROM suggests a potential predictability for nonlinear PDEs with varying parameters.

Table 4.4: Cross-validation of the DDC-ROM with different ansatz for the Burgers equation, ansatz generated by training data for $\nu = 10^{-3}$.

ν values	r	G-ROM $\mathcal{E}_{C^0(L^2)}$	DDC-ROM-Lin	DDC-ROM-Qu	DDC-ROM-Cu	IDDC-ROM	$\sqrt{\sum_{j=r+1}^{d}\lambda_j}$
9×10^{-4}	3	$1.8370e-01$	$1.4517e-01$	$1.3985e-01$	$1.3985e-01$	$1.3985e-01$	$1.3984e-01$
	4	$1.8153e-01$	$1.1851e-01$	$1.1430e-01$	$1.1424e-01$	$1.1425e-01$	$1.1423e-01$
	6	$2.2079e-01$	$9.2445e-02$	$1.5546e-01$	$1.4270e-01$	$3.2945e-01$	$8.6969e-02$
	11	$1.4758e-01$	$5.8442e-02$	$6.7804e-02$	$6.7281e-02$	$1.8822e-01$	$5.5828e-02$
8×10^{-4}	3	$1.8480e-01$	$1.4592e-01$	$1.4059e-01$	$1.4059e-01$	$1.4060e-01$	$1.4058e-01$
	4	$1.8327e-01$	$1.1940e-01$	$1.1534e-01$	$1.1512e-01$	$1.1511e-01$	$1.1508e-01$
	6	$2.2660e-01$	$9.3644e-02$	$1.6669e-01$	$1.4670e-01$	$3.3615e-01$	$8.7983e-02$
	11	$1.5877e-01$	$6.0001e-02$	$6.8318e-02$	$7.0530e-02$	$2.0017e-01$	$5.7134e-02$
5×10^{-4}	3	$1.8814e-01$	$1.4820e-01$	$1.4288e-01$	$1.4287e-01$	$1.4287e-01$	$1.4283e-01$
	4	$1.9027e-01$	$1.2306e-01$	$1.2107e-01$	$1.1896e-01$	$1.1871e-01$	$1.1852e-01$
	6	$2.4505e-01$	$9.7344e-02$	$2.0859e-01$	$1.6007e-01$	$3.5708e-01$	$9.1071e-02$
	11	$2.0251e-01$	$6.4685e-02$	$7.6505e-02$	$7.6332e-02$	$2.1428e-01$	$6.1158e-02$
4×10^{-4}	3	$1.8926e-01$	$1.4897e-01$	$1.4367e-01$	$1.4365e-01$	$1.4364e-01$	$1.4359e-01$
	4	$1.9027e-01$	$1.2889e-01$	$1.2127e-01$	$1.2179e-01$	$1.2569e-01$	$1.1852e-01$
	6	$2.5156e-01$	$9.8930e-02$	$1.0607e-01$	$1.2038e-01$	$2.3850e-01$	$9.2118e-022$
	11	$2.2088e-01$	$6.6103e-02$	$7.0510e-02$	$7.0533e-02$	$2.2269e-01$	$6.2533e-02$
3×10^{-4}	3	$1.9039e-01$	$1.5866e-01$	diverged	$1.7196e-01$	$2.1756e-01$	$1.4436e-01$
	4	$1.9203e-01$	$1.2838e-01$	diverged	$1.6041e-01$	$1.6516e-01$	$1.1940e-01$
	6	$2.5825e-01$	$9.9256e-02$	diverged	$1.6150e-01$	$1.0961e-01$	$9.3172e-02$
	11	$2.4093e-01$	$1.0778e-01$	diverged	$1.6508e-01$	$3.1136e-01$	$6.3925e-02$
2×10^{-4}	3	$1.9151e-01$	$1.5947e-01$	diverged	$1.7296e-01$	$2.1867e-01$	$1.4514e-01$
	4	$1.9378e-01$	$1.2933e-01$	diverged	$1.6187e-01$	$1.6678e-01$	$1.2028e-01$
	6	$2.6512e-01$	$1.2539e-01$	diverged	$1.6503e-01$	$1.1195e-01$	$9.4236e-02$
	11	$2.6230e-01$	$1.1418e-01$	diverged	$1.7675e-01$	$1.9061e-01$	$6.5333e-02$

Table 4.5: Cross-validation of the DDC-ROM with different ansatz for the Burgers equation, ansatz generated by training data for $\nu = 10^{-4}$.

ν values	r	G-ROM $\mathcal{E}_{C^0(L^2)}$	DDC-ROM-Lin	DDC-ROM-Qu	DDC-ROM-Cu	IDDC-ROM	$\sqrt{\sum_{j=r+1}^{d}\lambda_j}$
9×10^{-4}	3	$1.8370e-01$	$1.5415e-01$	diverged	$1.6420e-01$	$2.1057e-01$	$1.3984e-01$
	4	$1.8153e-01$	$1.2273e-01$	diverged	$1.7703e-01$	$1.5549e-01$	$1.1423e-01$
	6	$2.2079e-01$	$9.1956e-02$	$4.4640e+02$	$1.6950e-01$	$3.6719e-01$	$8.6969e-02$
	11	$1.4758e-01$	$5.8517e-02$	diverged	$8.3275e-02$	$1.4051e-01$	$5.5828e-02$
5×10^{-4}	3	$1.8814e-01$	$1.5719e-01$	diverged	$1.6821e-01$	$2.1496e-01$	$1.4283e-01$
	4	$1.8851e-01$	$1.2624e-01$	diverged	$1.8299e-01$	$1.6177e-01$	$1.1765e-01$
	6	$2.4505e-01$	$9.6342e-02$	diverged	$3.5050e-01$	$3.9743e-01$	$9.1071e-02$
	11	$2.0251e-01$	$6.3880e-02$	diverged	$1.9145e-01$	$1.4437e-01$	$6.1158e-02$
4×10^{-4}	3	$1.8926e-01$	$1.5797e-01$	diverged	$1.4252e+00$	$2.1606e-01$	$1.4359e-01$
	4	$1.9027e-01$	$1.2714e-01$	diverged	$9.4529e-01$	$1.6337e-01$	$1.1852e-01$
	6	$2.5156e-01$	$9.7675e-02$	diverged	$3.1552e-01$	$2.6986e-01$	$9.2118e-02$
	11	$2.2088e-01$	$6.5068e-02$	diverged	$3.9441e-01$	$3.8677e-01$	$6.2533e-02$
3×10^{-4}	3	$1.9039e-01$	$1.4961e-01$	$1.4437e-01$	$1.4438e-01$	$1.4437e-01$	$1.4436e-01$
	4	$1.9203e-01$	$1.2344e-01$	$1.1994e-01$	$1.1944e-01$	$1.1943e-01$	$1.1940e-01$
	6	$2.5825e-01$	$9.7453e-02$	$9.3704e-02$	$9.3225e-02$	$9.3308e-02$	$9.3172e-02$
	11	$2.4093e-01$	$6.6008e-02$	$7.0255e-02$	$6.9331e-02$	$2.4619e-01$	$6.3925e-02$
2×10^{-4}	3	$1.9151e-01$	$1.5039e-01$	$1.4514e-01$	$1.4515e-01$	$1.4514e-01$	$1.4514e-01$
	4	$1.9378e-01$	$1.2436e-01$	$1.2042e-01$	$1.2033e-01$	$1.2030e-01$	$1.2028e-01$
	6	$2.6512e-01$	$9.8636e-02$	$9.4376e-02$	$9.4278e-02$	$9.4300e-02$	$9.4236e-02$
	11	$2.6230e-01$	$6.7484e-02$	$6.7803e-02$	$6.7180e-02$	$7.7857e-02$	$6.5333e-02$

Chapter 5

Conclusions and Future Work

5.1 Conclusions

ROM projection spatial filtering as well as data-driven modeling for τ are used to develop the DDC-ROM framework. Within this DDC-ROM framework, the DDC-ROM with a linear ansatz, with a quadratic ansatz and with a cubic ansatz, as well as the idealized DDC-ROM are tested numerically. The numerical results show that in general, DDC-ROMs perform better than the standard Galerkin ROM, while they share similar online stage CPU time. Furthermore, the smaller the ν values, the better DDC-ROM results.

The cross-validation of the DDC-ROM for different ν values is also performed. The numerical results reveal the potential predictability of the DDC-ROM for varying parameter ν values. The cross-validation also shows that the DDC-ROM with a linear ansatz seems to be a robust model, while the DDC-ROM with higher-order ansatz may suffer from numerical instability for some ν values. This instability might be fixed by enforcing conservation properties for the quadratic ansatz [59, 66].

5.2 Future Works

- In this thesis, only the one-dimensional Burgers equation is used in the numerical tests; in [106], three dimensional-simulation of flow past the cylinder is performed for the DDC-ROM with a quadratic ansatz. The numerical investigation of the three-dimensional DDC-ROM with a cubic ansatz will be performed. This investigation could shed new light on the findings in this thesis regarding the pattern in τ, the cubic ansatz, or the cross-validation of the DDC-ROM.

- Some physical constraints (from the nature of the underlying problem) can be added

[47, 67]. As a result, instead of the optimization problem 3.25, we will solve a *constrained* optimization problem.

- The DDC-ROM applies the ROM projection filter to obtain the filtered ROM. On the other hand, filtered ROM can be obtained via other filters, e.g., the ROM differential filter. We plan to test the DDC-ROM with other filters.

- In [39], a data-driven framework for parametrized PDEs is proposed and a regression model is developed to build the ROM. A regression model for τ can be extended . This extension could provide a reliable and efficient way for solving parametrized time-dependent problems.

- Inspired by Chapter 4 (which can be categorized in the area of pattern recognition), statistical tools, e.g., deep learning, stochastic tools [24, 25, 33, 52, 60, 62, 85], and uncertainty quantification [63, 64, 65] can be used to analyse the patterns of the ansatz and thus possibly provides a DDC-ROM that is more robust with respect to changes in the Reynolds numbers.

Bibliography

[1] Konstantin Afanasiev and Michael Hinze. Adaptive control of a wake flow using proper orthogonal decomposition. *Lecture Notes in Pure and Applied Mathematics*, pages 317–332, 2001.

[2] Imran Akhtar, Ali H Nayfeh, and Calvin J Ribbens. On the stability and extension of reduced-order galerkin models in incompressible flows. *Theoretical and Computational Fluid Dynamics*, 23(3):213–237, 2009.

[3] Sylvain Arlot, Alain Celisse, et al. A survey of cross-validation procedures for model selection. *Statistics surveys*, 4:40–79, 2010.

[4] Jeanne A Atwell and Belinda B King. Reduced order controllers for spatially distributed systems via proper orthogonal decomposition. *SIAM Journal on Scientific Computing*, 26(1):128–151, 2004.

[5] 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, 1988.

[6] Zhaojun Bai. Krylov subspace techniques for reduced-order modeling of large-scale dynamical systems. *Applied numerical mathematics*, 43(1-2):9–44, 2002.

[7] Joan Baiges, Ramon Codina, and Sergio Idelsohn. Reduced-order subscales for pod models. *Computer Methods in Applied Mechanics and Engineering*, 291:173–196, 2015.

[8] Francesco Ballarin, Andrea Manzoni, Alfio Quarteroni, and Gianluigi Rozza. Supremizer stabilization of pod–galerkin approximation of parametrized steady incompressible navier–stokes equations. *International Journal for Numerical Methods in Engineering*, 102(5):1136–1161, 2015.

[9] Francesco Ballarin, Elena Faggiano, Sonia Ippolito, Andrea Manzoni, Alfio Quarteroni, Gianluigi Rozza, and Roberto Scrofani. Fast simulations of patient-specific haemodynamics of coronary artery bypass grafts based on a pod–galerkin method and a vascular shape parametrization. *Journal of Computational Physics*, 315:609–628, 2016.

[10] Harvey T Banks, Michele L Joyner, Buzz Wincheski, and William P Winfree. Nondestructive evaluation using a reduced-order computational methodology. *Inverse Problems*, 16(4):929, 2000.

[11] 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.

[12] Mouhacine Benosman, Jeff Borggaard, Omer San, and Boris Kramer. Learning-based robust stabilization for reduced-order models of 2d and 3d boussinesq equations. *Applied Mathematical Modelling*, 49:162–181, 2017.

[13] Michel Bergmann, Andrea Ferrero, Angelo Iollo, Edoardo Lombardi, Angela Scardigli, and Haysam Telib. A zonal galerkin-free pod model for incompressible flows. *Journal of Computational Physics*, 352:301–325, 2018.

[14] Luigi Carlo Berselli, Traian Iliescu, and William J Layton. *Mathematics of large eddy simulation of turbulent flows*. Springer Science & Business Media, 2005.

[15] Diana A Bistrian and I. Michael Navon. An improved algorithm for the shallow water equations model reduction: Dynamic mode decomposition vs pod. *International Journal for Numerical Methods in Fluids*, 78(9):552–580, 2015.

[16] Jeff Borggaard, John A Burns, Amit Surana, and Lizette Zietsman. Control, estimation and optimization of energy efficient buildings. In *American Control Conference, 2009. ACC'09.*, pages 837–841. IEEE, 2009.

[17] Jeff Borggaard, Zhu Wang, and Lizette Zietsman. A goal-oriented reduced-order modeling approach for nonlinear systems. *Computers & Mathematics with Applications*, 71(11):2155–2169, 2016.

[18] Susanne Brenner and Ridgway Scott. *The mathematical theory of finite element methods*, volume 15. Springer Science & Business Media, 2007.

[19] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, page 201517384, 2016.

[20] Marcelo Buffoni, Simone Camarri, Angelo Iollo, and Maria Vittoria Salvetti. Low-dimensional modelling of a confined three-dimensional wake flow. *Journal of Fluid Mechanics*, 569:141–150, 2006.

[21] John A Burns and Belinda B King. A reduced basis approach to the design of low-order feedback controllers for nonlinear continuous systems. *Journal of Vibration and Control*, 4(3):297–323, 1998.

[22] Alfonso Caiazzo, Traian Iliescu, Volker John, and Swetlana Schyschlowa. A numerical investigation of velocity–pressure reduced order models for incompressible flows. *Journal of Computational Physics*, 259:598–616, 2014.

[23] Kevin Carlberg, Matthew Barone, and Harbir Antil. Galerkin v. least-squares petrov–galerkin projection in nonlinear model reduction. *Journal of Computational Physics*, 330:693–734, 2017.

[24] Mickaël D Chekroun, Honghu Liu, and Shouhong Wang. *Stochastic parameterizing manifolds and non-Markovian reduced equations: stochastic manifolds for nonlinear SPDEs II*. Springer, 2014.

[25] Mickaël D Chekroun, Honghu Liu, and Shouhong Wang. *Approximation of stochastic invariant manifolds: stochastic manifolds for nonlinear SPDEs I*. Springer, 2015.

[26] Youngsoo Choi and Kevin Carlberg. Space-time least-squares petrov-galerkin projection for nonlinear model reduction. *arXiv preprint arXiv:1703.04560*, 2017.

[27] Alexandre Joel Chorin and Jerrold E Marsden. *A mathematical introduction to fluid mechanics*, volume 3. Springer, 1990.

[28] M Couplet, C Basdevant, and P Sagaut. Calibrated reduced-order pod-galerkin system for fluid flow modelling. *Journal of Computational Physics*, 207(1):192–220, 2005.

[29] Anil E Deane and Lawrence Sirovich. A computational study of rayleigh–bénard convection. part 1. rayleigh-number scaling. *Journal of Fluid Mechanics*, 222:231–250, 1991.

[30] James W Demmel. *Applied numerical linear algebra*, volume 56. SIAM, 1997.

[31] Howard C Elman, David J Silvester, and Andrew J Wathen. *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*. Numerical Mathematics and Scientific Computation, 2014.

[32] Hiba Fareed and John R Singler. A note on incremental pod algorithms for continuous time data. *arXiv preprint arXiv:1807.00045*, 2018.

[33] Florian Feppon and Pierre FJ Lermusiaux. Dynamically orthogonal numerical schemes for efficient stochastic advection and lagrangian transport. *SIAM Review*, 60(3):595–625, 2018.

[34] Lambert Fick, Yvon Maday, Anthony T Patera, and Tommaso Taddei. A reduced basis technique for long-time unsteady turbulent flows. *arXiv preprint arXiv:1710.03569*, 2017.

[35] Erich L Foster, Traian Iliescu, and Zhu Wang. A finite element discretization of the streamfunction formulation of the stationary quasi-geostrophic equations of the ocean. *Computer Methods in Applied Mechanics and Engineering*, 261:105–117, 2013.

[36] Gene H Golub and Charles F Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

[37] Ayoub Gouasmi, Eric J Parish, and Karthik Duraisamy. A priori estimation of memory effects in reduced-order models of nonlinear systems using the mori–zwanzig formalism. *Proc. R. Soc. A*, 473(2205):20170385, 2017.

[38] Max Gunzburger, Nan Jiang, and Michael Schneier. An ensemble-proper orthogonal decomposition method for the nonstationary navier–stokes equations. *SIAM Journal on Numerical Analysis*, 55(1):286–304, 2017.

[39] Mengwu Guo and Jan S Hesthaven. Data-driven reduced order modeling for time-dependent problems. Technical report, 2018. Preprint on webpage at `https://infoscience.epfl.ch/record/255406/files/Document%20file.pdf`.

[40] Jack K Hale and Hüseyin Koçak. *Dynamics and bifurcations*, volume 3. Springer Science & Business Media, 2012.

[41] Jan S Hesthaven, Gianluigi Rozza, Benjamin Stamm, et al. *Certified reduced basis methods for parametrized partial differential equations*. Springer, 2016.

[42] Philip Holmes, John L Lumley, Gahl Berkooz, and Clarence W Rowley. *Turbulence, coherent structures, dynamical systems and symmetry*. Cambridge university press, 2012.

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

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

[45] Traian Iliescu and Zhu Wang. Variational multiscale proper orthogonal decomposition: Navier-stokes equations. *Numerical Methods for Partial Differential Equations*, 30(2): 641–663, 2014.

[46] Angelo Iollo, Stéphane Lanteri, and J-A Désidéri. Stability properties of pod–galerkin approximations for the compressible navier–stokes equations. *Theoretical and Computational Fluid Dynamics*, 13(6):377–396, 2000.

[47] Volker John, Alexander Linke, Christian Merdon, Michael Neilan, and Leo G Rebholz. On the divergence constraint in mixed finite element methods for incompressible flows. *SIAM Review*, 59(3):492–544, 2017.

[48] Birgul Koc. Commutation error of data-driven correction reduced order modeling. Master's thesis, Virginia Polytechnic Institute and State University.

[49] Birgul Koc, Muhammad Mohebujjaman, Changhong Mou, and Traian Iliescu. Commutation error in reduced order modeling of fluid flows. *arXiv preprint arXiv:1810.00517*, 2018.

[50] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada, 1995.

[51] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[52] Dmitri Kondrashov, Mickaël D Chekroun, and Michael Ghil. Data-driven non-markovian closure models. *Physica D: Nonlinear Phenomena*, 297:33–55, 2015.

[53] DD Kosambi. Statistics in function space. In *DD Kosambi*, pages 115–123. Springer, 2016.

[54] Karl Kunisch and Stefan Volkwein. Galerkin proper orthogonal decomposition methods for parabolic problems. *Numerische mathematik*, 90(1):117–148, 2001.

[55] Karl Kunisch and Stefan Volkwein. Galerkin proper orthogonal decomposition methods for a general equation in fluid dynamics. *SIAM Journal on Numerical analysis*, 40(2): 492–515, 2002.

[56] Mats G Larson and Fredrik Bengzon. *The finite element method: theory, implementation, and applications*, volume 10. Springer Science & Business Media, 2013.

[57] William Layton. *Introduction to the numerical analysis of incompressible viscous flows*, volume 6. Siam, 2008.

[58] Michel Loève. Functions aleatoire de second ordre. *Revue science*, 84:195–206, 1946.

[59] Jean-Christophe Loiseau and Steven L Brunton. Constrained sparse galerkin regression. *Journal of Fluid Mechanics*, 838:42–67, 2018.

[60] Fei Lu, Kevin K Lin, and Alexandre J Chorin. Data-based stochastic model reduction for the kuramoto–sivashinsky equation. *Physica D: Nonlinear Phenomena*, 340:46–57, 2017.

[61] John L Lumley. The structure of inhomogeneous turbulent flows. *Atmospheric turbulence and radio wave propagation*, 1967.

[62] John L Lumley. *Stochastic tools in turbulence*. Courier Corporation, 2007.

[63] Andrew J Majda and Michal Branicki. Lessons in uncertainty quantification for turbulent dynamical systems. *Discrete & Continuous Dynamical Systems-A*, 32(9):3133–3221, 2012.

[64] Andrew J Majda and Nan Chen. Model error, information barriers, state estimation and prediction in complex multiscale systems. *Entropy*, 20(9):644, 2018.

[65] Andrew J Majda and Di Qi. Strategies for reduced-order models for predicting the statistical responses and uncertainty quantification in complex turbulent dynamical systems. *arXiv preprint arXiv:1802.08051*, 2018.

[66] Muhammad Mohebujjaman, Leo G. Rebholz, and Traian Iliescu. Physically-constrained data-driven correction for reduced order modeling of fluid flows. *International Journal for Numerical Methods in Fluids*.

[67] Muhammad Mohebujjaman, Leo G Rebholz, Xuping Xie, and Traian Iliescu. Energy balance and mass conservation in reduced order models of fluid flows. *Journal of Computational Physics*, 346:262–277, 2017.

[68] 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.

[69] Assad A Oberai and Jayanth Jagalur-Mohan. Approximate optimal projection for reduced-order models. *International Journal for Numerical Methods in Engineering*, 105(1):63–80, 2016.

[70] Alexander Obukhov. Statistical description of continuous fields. *Transactions of the Geophysical International Academy Nauk USSR*, 24(24):3–42, 1954.

[71] Jan Östh, Bernd R Noack, Siniša Krajnović, Diogo Barros, and Jacques Borée. On the need for a nonlinear subscale turbulence term in pod models as exemplified for a high-reynolds-number flow over an ahmed body. *Journal of Fluid Mechanics*, 747: 518–544, 2014.

[72] Shaowu Pan and Karthik Duraisamy. Data-driven discovery of closure models. *arXiv preprint arXiv:1803.09318*, 2018.

[73] Eric J. Parish, Christopher Wentland, and Karthik Duraisamy. A residual-based petrov-galerkin reduced-order model with memory effects. *arXiv preprint arXiv:1810.03455*, 2018.

[74] Benjamin Peherstorfer and Karen Willcox. Dynamic data-driven reduced-order models. *Computer Methods in Applied Mechanics and Engineering*, 291:21–41, 2015.

[75] Benjamin Peherstorfer and Karen Willcox. Data-driven operator inference for nonintrusive projection-based model reduction. *Computer Methods in Applied Mechanics and Engineering*, 306:196–215, 2016.

[76] Giuseppe Pitton, Annalisa Quaini, and Gianluigi Rozza. Computational reduction strategies for the detection of steady bifurcations in incompressible fluid-dynamics: Applications to coanda effect in cardiology. *Journal of Computational Physics*, 344: 534–557, 2017.

[77] Bartosz Protas, Bernd R Noack, and Jan Östh. Optimal nonlinear eddy viscosity in galerkin models of turbulent flows. *Journal of Fluid Mechanics*, 766:337–367, 2015.

[78] Vladimir Semenovich Pugachev. The general theory of correlation of random functions. *Izvestiya Rossiiskoi Akademii Nauk. Seriya Matematicheskaya*, 17(5):401–420, 1953.

[79] Alfio Quarteroni and Alberto Valli. *Numerical Approximation of Partial Differential Equations*. Springer, 1994.

[80] Alfio Quarteroni, Andrea Manzoni, and Federico Negri. *Reduced basis methods for partial differential equations: an introduction*, volume 92. Springer, 2015.

[81] Tomás Chacón Rebollo, Enrique Delgado Ávila, Macarena Gómez Mármol, Francesco Ballarin, and Gianluigi Rozza. On a certified smagorinsky reduced basis turbulence model. *SIAM Journal on Numerical Analysis*, 55(6):3047–3067, 2017.

[82] Oleg Roderick, Zhu Wang, and Mihai Anitescu. Dimensionality reduction for uncertainty quantification of nuclear engineering models. *Science and Engineering*, 164(2): 122–138, 2010.

[83] Azriel Rosenfeld. *Digital picture processing*. Academic press, 1976.

[84] Gianluigi Rozza, Dinh Bao Phuong Huynh, and Anthony T Patera. Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations. *Archives of Computational Methods in Engineering*, 15 (3):1, 2007.

[85] Omer San and Romit Maulik. Neural network closures for nonlinear model order reduction. *Advances in Computational Mathematics*, pages 1–34, 2018.

[86] Omer San, Anne E Staples, Zhu Wang, and Traian Iliescu. Approximate deconvolution large eddy simulation of a barotropic ocean circulation model. *Ocean Modelling*, 40 (2):120–132, 2011.

[87] Peter J Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics*, 656:5–28, 2010.

[88] Jun Shao. Linear model selection by cross-validation. *Journal of the American statistical Association*, 88(422):486–494, 1993.

[89] Sirod Sirisup and George E Karniadakis. A spectral viscosity method for correcting the long-term behavior of pod models. *Journal of Computational Physics*, 194(1):92–116, 2004.

[90] Lawrence Sirovich. Turbulence and the dynamics of coherent structures. i. coherent structures. *Quarterly of applied mathematics*, 45(3):561–571, 1987.

[91] Lawrence Sirovich. Turbulence and the dynamics of coherent structures. ii. symmetries and transformations. *Quarterly of Applied mathematics*, 45(3):573–582, 1987.

[92] Giovanni Stabile and Gianluigi Rozza. Finite volume pod-galerkin stabilised reduced order methods for the parametrised incompressible navier–stokes equations. *Computers & Fluids*, 2018.

[93] Anne Staples and Omer San. A posteriori analysis of spatial filters for approximate deconvolution large eddy simulations of homogeneous incompressible flows. *Bulletin of the American Physical Society*, 57, 2012.

[94] Gilbert Strang. *Computational science and engineering*, volume 791. Wellesley-Cambridge Press Wellesley, 2007.

[95] Maria Strazzullo, Francesco Ballarin, Renzo Mosetti, and Gianluigi Rozza. Model reduction for parametrized optimal control problems in environmental marine sciences and engineering. *SIAM Journal on Scientific Computing*, 40(4):B1055–B1079, 2018.

[96] Steven H Strogatz. *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. CRC Press, 2018.

[97] Kunihiko Taira, Steven L Brunton, Scott TM Dawson, Clarence W Rowley, Tim Colonius, Beverley J McKeon, Oliver T Schmidt, Stanislav Gordeyev, Vassilios Theofilis, and Lawrence S Ukeiley. Modal analysis of fluid flows: An overview. *Aiaa Journal*, pages 4013–4041, 2017.

[98] Stefan Volkwein. Proper orthogonal decomposition: Theory and reduced-order modelling. *Lecture Notes, University of Konstanz*, 4(4), 2013.

[99] Zhu Wang, Imran Akhtar, Jeff Borggaard, and Traian Iliescu. Two-level discretizations of nonlinear closure models for proper orthogonal decomposition. *Journal of Computational Physics*, 230(1):126–146, 2011.

[100] Zhu Wang, Imran Akhtar, Jeff Borggaard, and Traian Iliescu. Proper orthogonal decomposition closure models for turbulent flows: a numerical comparison. *Computer Methods in Applied Mechanics and Engineering*, 237:10–26, 2012.

[101] Zhu Wang, Brian McBee, and Traian Iliescu. Approximate partitioned method of snapshots for pod. *Journal of Computational and Applied Mathematics*, 307:374–384, 2016.

[102] David Wells, Zhu Wang, Xuping Xie, and Traian Iliescu. An evolve-then-filter regularized reduced order model for convection-dominated flows. *International Journal for Numerical Methods in Fluids*, 84(10):598–615, 2017.

[103] Stephen Wiggins. *Introduction to applied nonlinear dynamical systems and chaos*, volume 2. Springer Science & Business Media, 2003.

[104] Stephen Wright and Jorge Nocedal. Numerical optimization. *Springer Science*, 35 (67-68):7, 1999.

[105] Xuping Xie, David Wells, Zhu Wang, and Traian Iliescu. Approximate deconvolution reduced order modeling. *Computer Methods in Applied Mechanics and Engineering*, 313:512–534, 2017.

[106] Xuping Xie, Muhammad Mohebujjaman, Leo G. Rebholz, and Traian Iliescu. Data-driven filtered reduced order modeling of fluid flows. *SIAM Journal on Scientific Computing*, 40(3):B834–B857, 2018.

[107] Xuping Xie, David Wells, Zhu Wang, and Traian Iliescu. Numerical analysis of the leray reduced order model. *Journal of Computational and Applied Mathematics*, 328: 12–29, 2018.

[108] Masayuki Yano. Discontinuous galerkin reduced basis empirical quadrature procedure for model reduction of parametrized nonlinear conservation laws.