DATA-DRIVEN REDUCED-ORDER MODELING FOR COMPUTATIONAL FLUID DYNAMICS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

DAMLA SAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
MECHANICAL ENGINEERING

SEPTEMBER 2023

Approval of the thesis:

DATA-DRIVEN REDUCED-ORDER MODELING FOR COMPUTATIONAL FLUID DYNAMICS

submitted by **DAMLA SAN** in partial fulfillment of the requirements for the degree of **Master of Science in Mechanical Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. M. A. Sahir Arıkan
Head of Department, **Mechanical Engineering** _____

Assist. Prof. Dr. Ali Karakuş
Supervisor, **Mechanical Engineering, METU** _____

Assist. Prof. Dr. Romit Maulik
Co-supervisor, **Information Science and Technology, Penn State University** _____

Examining Committee Members:

Prof. Dr. Metin Yavuz
Mechanical Engineering, METU _____

Assist. Prof. Dr. Ali Karakuş
Mechanical Engineering, METU _____

Assist. Prof. Dr. Özgür Uğraş Baran
Mechanical Engineering, METU _____

Assist. Prof. Dr. Altuğ Özçelikkale
Mechanical Engineering, METU _____

Assist. Prof. Dr. Onur Baş
Mechanical Engineering, TEDU _____

Date:11.09.2023

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Damla San

Signature :

ABSTRACT

DATA-DRIVEN REDUCED-ORDER MODELING FOR COMPUTATIONAL FLUID DYNAMICS

San, Damla
M.S., Department of Mechanical Engineering
Supervisor: Assist. Prof. Dr. Ali Karakuş
Co-Supervisor: Assist. Prof. Dr. Romit Maulik

September 2023, 100 pages

Reduced order models (ROM) play a crucial role in tackling the computational challenges posed by complex flow simulations. They provide an effective solution to the resource-intensive demands of direct numerical simulations. Among the techniques utilized for constructing reduced-order models, proper orthogonal decomposition (POD) stands out as a frequently employed method with applications spanning diverse fields of engineering and science. Despite its initial appeal as a means to attain both computational efficiency and precise representation of coherent structures in turbulent flows at high Reynolds numbers, the full realization of this potential remains a goal to be achieved. Appropriate closure modeling methodologies should be used to achieve an ideal combination between the lowest computing cost inherent in ROM and the complexities of the targeted flows.

This thesis employs a combination of innovative physics-based and data-driven modeling tools to develop more robust and improved frameworks for ROM in complex flows. In this thesis, the core concepts of the ROM are introduced, centered around utilizing reduced bases generated from snapshots. Applying techniques like POD and

Galerkin Projection is demonstrated through numerical results involving diverse flow equations, ranging from the Burgers equation to the Navier-Stokes equations. The primary focus is on addressing the challenges arising from the dynamic changes inherent in turbulent flows, which can limit conventional ROM methods. To enhance the accuracy of the ROM approximation, a closure term is formulated using machine learning methods.

Each chapter begins by outlining the full-order model (FOM) employed to generate snapshots representing the flow across different instantaneous time points for each equation. Subsequently, the steps taken in reduced-order modeling are detailed. By comparing outcomes obtained from the full-order model to those derived using POD and Galerkin Projection, an assessment is made to demonstrate the accuracy and efficiency of these methods in reducing computational complexity and providing accurate solutions for complex fluid flow problems.

Keywords: numerical methods, computational fluid dynamics, reduced order model, proper orthogonal decomposition, Galerkin projection, artificial neural networks

ÖZ

HESAPLAMALI AKIŞKANLAR DİNAMİĞİ İÇİN VERİYE DAYALI İNDİRGENMİŞ MODELLEME

San, Damla
Yüksek Lisans, Makina Mühendisliği Bölümü
Tez Yöneticisi: Dr. Öğr. Üyesi. Ali Karakuş
Ortak Tez Yöneticisi: Dr. Öğr. Üyesi. Romit Maulik

Eylül 2023 , 100 sayfa

İndirgenmiş modeller, karmaşık akış simülasyonlarının ortaya çıkardığı hesaplama zorluklarının üstesinden gelmede çok önemli bir rol oynar. Doğrudan sayısal simülasyonların yoğun taleplerine etkili bir çözüm sunarlar. İndirgenmiş modeller oluşturmak için kullanılan teknikler listesi arasında uygun dik ayrıştırma, çeşitli mühendislik ve bilim alanlarını kapsayan uygulamalarda sıklıkla kullanılan bir yöntemdir. Yüksek Reynolds sayılarında türbülanslı akışlarda tutarlı yapıların hem hesaplama verimliliğine hem de kesin temsiline ulaşmanın bir aracı olarak ilk iyi sonuçlarına rağmen, bu potansiyelin tam olarak gerçekleştirilmesi, ulaşılması gereken bir hedef olmaya devam etmektedir. İndirgenmiş modellemenin doğasında bulunan en düşük bilgi işlem maliyeti ile hedeflenen akışların karmaşıklığı arasında ideal bir kombinasyon elde etmek için uygun kapatma modelleme metodolojileri kullanılmalıdır.

Bu tezde, karmaşık akışlarda indirgenmiş modelleme için daha etkili ve gelişmiş çerçeveler oluşturmak için en son fizik tabanlı ve veri odaklı modelleme araçlarından yararlanıyoruz. İlk olarak, uygun dik ayrıştırma kullanılarak anlık görüntülerden he-

saplanan indirgenmiş tabanlar üzerine inşa edilen indirgenmiş mertebeli modellerin temel yönleri sunulmaktadır. Burgers denklemi, girdap akış denklemler ve son olarak Navier-Stokes denklemleri dahil olmak üzere çeşitli akış denklemlerine POD ve Galerkin Projeksiyonunun uygulanması için sayısal sonuçlar sunarken, ortaya çıkan türbülanslı akışların gelişen dinamikleriyle ilgili zorlukların üstesinden gelmeye odaklanıyoruz. Daha sonra makine öğrenmesi teknikleri kullanılarak azaltılmış mertebeli model yaklaşımının sayısal doğruluğunu artırmayı amaçlayan kapanma terimi oluşturulur.

Her denklem için zamanın farklı noktalarındaki akışı temsil eden anlık görüntüleri oluşturmak için kullanılan tam sıralı model, her bölümün başında açıklanmış ve ardından indirgenmiş sıralı modelleme adımları izlenmiştir. Tam sıralı modelin sonuçlarını UDA ve Galerkin projeksiyonu kullanılarak elde edilen sonuçlarla karşılaştırarak, bu denklemler için bu model sıra azaltma tekniklerinin doğruluğunu ve etkinliğini değerlendiriyoruz. Bu sonuçlar aracılığıyla, hesaplama karmaşıklığını azaltmada ve karmaşık sıvı akışı problemlerine doğru çözümler sağlamada bu yöntemlerin etkinliğini göstermeyi amaçlıyoruz.

Anahtar Kelimeler: sayısal yöntemler, hesaplamalı akışkanlar dinamiği, indirgenmiş mertebeli model, uygun ortogonal ayrıştırma, Galerkin projeksiyonu, yapay sinir ağları

Dedicated to my family.

ACKNOWLEDGMENTS

First, I want to express my sincere gratitude to my supervisor, Assist. Prof. Dr. Ali Karakuş and Assist. Prof. Dr.Romit Maulik. Thanks to Dr. Karakuş, I was privileged to join the Accelerated Multiphysics Research Group at the Middle East Technical University of Mechanical Engineering Department. Embarking on a journey in a new field is definitely a significant challenge. However, I found great enjoyment in taking my initial steps into the world of science, thanks to the constant support and guidance of Dr. Karakuş.

I am deeply grateful for all our insightful discussions with the Accelerated Multiphysics Research Group members. These discussions played an essential role in helping me structure my thoughts and direct my attention toward the important questions. Thanks to Atakan Aygün for all his support, understanding, and endless coffee during the challenging phases of working on this thesis. I am confident that he will develop into an exceptional researcher in his own right.

I would like to take this moment to express my deep gratitude to my beloved friends, Berk Kıymaz and Orhun Taşoğlu. Your friendship, whether during the joyful times or the challenging ones, is invaluable in my life, helping me maintain my spiritual balance. Special thanks to Hakan Bostan for his endless commitment to studying with me and being there for me whenever I need it, even when separated by great distances. Without his support and help, completing this thesis would not have been possible.

Last but certainly not least, I want to express my heartfelt gratefulness to my family. Their encouragement during the challenging times and celebration of my achievements have meant so much to me. I see my mother and father as my first teachers, and this thesis is dedicated to them. I am also especially thankful to my sister, Defne San, for her remarkable ability to bring a smile to my face, even during the most challenging times. Her support and positivity have been truly inspiring.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

2D	2 Dimensional
3D	3 Dimensional
PDE	Partial Differential Equation
FDM	Finite Difference Method
FVM	Finite Volume Method
FEM	Finite Element Method
DG-FEM	Discontinuous Galerkin Finite Element Method
CFD	Computational Fluid Dynamics
DOF	Degrees of Freedom
FOM	Full Order Model
ROM	Reduced Order Model
RBM	Reduced Basis Method
POD	Proper Orthogonal Decomposition
PGD	Proper Generalized Decomposition
DMD	Dynamic Mode Decomposition
LES	Large Eddy Simulation
VMS	Variational Multiscale
AD	Approximate Deconvolution
SVD	Singular Value Decomposition
ODE	Ordinary Differential Equation
RIC	Relative Information Content
ANN	Artificial Neural Network
GROM	Galerkin Reduced Order Modeling
INS	Incompressible Navier-Stokes Equation

LIST OF ABBREVIATIONS

$\mathbf{u}$	arbitrary function to be generated
a_k	time coefficient of the k^{th} mode
φ_k	POD basis modes
M	number of snapshots
N	number of nodes
$\mathbf{C}$	correlation matrix
S	snapshot matrix
λ	eigenvalues
r	number of modes retained in approximation
Ω	domain of interest
$\widehat{a_k}$	Galerkin time coefficient of the k^{th} mode
c_k	correction term
u	x-component of the velocity
v	y-component of the velocity
t	time
ν	kinematic viscosity
$\mathbf{u}'$	fluctuating velocity with respect to the mean flow
ψ	streamfunction
ω	vorticity
p	pressure

CHAPTER 1

INTRODUCTION

In this chapter, the conventional differential equation solvers are discussed first. Brief information about the numerical methods used for solving partial differential equations is presented, followed by an introduction to reduced-order modeling. The main modeling principles used to construct reduced-order models and their applications are explained with a primary focus on one of the most popular model reduction techniques used within this thesis's scope: proper orthogonal decomposition (POD). Finally, closure modeling, a method to increase the accuracy of the reduced-order model, is introduced.

1.1 Conventional Numerical Methods

Many physical phenomena, such as the propagation of heat or sound, fluid flow, waves, elasticity, electrodynamics, etc., can be mathematically formulated and modeled by a set of governing equations, several of them being partial differential equations (PDEs) involving functions of multiple variables. Some of these equations are difficult or impossible to compute using analytical methods exactly. Numerical methods provide a way to approximate their solutions to predict the system's real-world behavior. Converting these partial equations into a system of algebraic equations is the key to various numerical methods [2].

PDEs are commonly approached through numerical means, employing methods such as the finite difference method (FDM), finite volume method (FVM), and finite element method (FEM). Among these, FDM is the most straightforward method with the most extended historical usage. In FDM, a grid is constructed using K differ-

ent points with neighboring x^k, $k = 1...K$, and the idea is to discretize the variable derivatives using a Taylor series expansion [1]. For simple problems discretized on structured grids, FDM is intuitive and easy to implement, leading to very efficient schemes for many problems. Furthermore, extending the higher-order approximations using higher-order local polynomial approximations is relatively simple. For example, up to tenth-order compact finite difference methods were introduced by Lele [3]. For numerous applications, these schemes have demonstrated their ability to achieve the goals of high accuracy and low computing cost. However, complex or multiscale geometries, as in the case of real-life applications, are typically difficult to deal with effectively and expensive to compute for FDM because of the requirement for ordered grids [4]. Element-based discretization is introduced to overcome the difficulties that arise while solving the complex geometries and discontinuous regions with FDM. Instead of using simple one-dimensional points located structurally in the domain, elements D^k, generally triangles or quadrilaterals in two-dimension, and tetrahedra or hexahedra in three-dimension, are used in an unstructured manner to construct the grid [1]. The FVM and FEM are commonly used examples for element-based methods with significantly less strict requirements for the structured grid compared to FDM [4].

In the case of FVM, each spatial location represents a local cell that can have a solution discontinuous to neighboring cells. This method can be applied to all differential equations, which can be written in the divergence form [4]. Conservation laws are written for each volume; boundary and initial conditions are applied. Using Gauss' theorem, the volume integral over the divergence can be changed into a surface integral across the borders. That is, Gauss' theorem enables transforming the integration of the differential of the dependent variable inside the cells to integrating the dependent variable's fluxes across the cell boundaries [4]. The equation is then integrated over the volume. Generally, one may calculate these integrals using the proper numerical approximation methods, which yields a differential equation that is considerably simplified [4]. However, problems arise in FVM while attempting to enhance the order of accuracy. Since the solution value for each cell is defined as the value at its center, the solution on the boundary is not clearly defined. Therefore, this method must be supported with some form of reconstruction method. A local interpolation

method is typically utilized, with nearby cell values considered. More cell values and specific mesh structures must be considered to obtain higher-order interpolation of the solution and flux. For example, Wang [5] developed the spectral volume method in which each mesh cell, or spectral volume, is subdivided. The sub-cell state averages are then utilized to generate a high-order reconstruction of the solution inside the spectral volume. However, this contradicts the fundamental requirement for geometric flexibility and leads to a less local method that is both complex and unstable [1]. One can be encouraged to adopt an alternative strategy and add more degrees of freedom to the element which yields in FEM.

Above all the methods explained so far, FEM is by far the most adaptable method, and it can be implemented to solve various numerical problems. Because of this, FEM may be used to solve any differential equation numerically [4]. The basic idea behind the FEM is to convert the problem into a vector problem using the properties of Hilbert spaces, i.e. a linear vector space that has an inner product operation so that the original problem can be expressed as the sum of coefficients multiplied by basis functions that are symmetric in space [1]. The solution function over each finite element domain is then approximated using the weighted residual concept. The ability to formulate methods for basis functions of various orders is a benefit of the FEM. Higher basis function orders result in higher-order, accurate approaches, which have the crucial advantage of increasing accuracy for a certain mesh. The FEM is frequently superior for multiphysics problems when there is a significant degree of coupling and nonlinearity because of its capacity to resolve fine details in the solution.

While Finite Element Method (FEM) can provide more precise solutions using lower-resolution grids by adjusting the order of basis functions, its symmetric nature can introduce a complicated challenge for situations where data propagates directionally, as seen in wave problems and conservation laws [1]. Finite difference and finite volume methods address this challenge by employing upwinding techniques, achieved through stencil choice or the development of reconstruction methods. These approaches mitigate the issue by favoring specific directions of information propagation [1]. A viable solution to this issue involves a strategic combination of finite element and finite volume methods. This entails employing a collection of basis and test functions that emulate the finite element approach while solving the equation more aligned

with the finite volume method. This blend of techniques led to the development of the discontinuous Galerkin Finite Element Method (DG-FEM). [1].

What sets the discontinuous Galerkin (DG) approach apart from the conventional finite element method is that the resulting equations are related only to the generating element. This means that the linked mass matrix remains localized for each element. The solution within each element is not rebuilt by examining neighboring elements. Hence, each element can be seen as a distinct entity requiring some boundary information from its neighbors [6]. Consequently, the mass matrix can be efficiently inverted, leading to an economical computation and an explicit semidiscrete approach. Reconstruction of the solution is not required either, except for the zeroth-order method, which is equivalent to the finite volume method. Moreover, by carefully constructing the numerical flux to align with the fundamental dynamics, an improved ability is achieved to guarantee stability for convection-dominated problems, exceeding the capabilities of conventional FEM approaches. [1]. Unlike the Finite Volume Method, the DG-FEM tackles the main problem of achieving accurate results on different grids using a local element-based approach [1]—table 1.1 lists the general characteristics of popular numerical approaches.

Table 1.1: Generic properties of well-known numerical methods. ✓ and ✗ indicate success and failure, respectively, while the (✓) symbol indicates that the method requires modifications to solve the problem [1].

Properties	**Numerical Methods**			
	FDM	**FVM**	**FEM**	**DG**
Complex geometries	✗	✓	✓	✓
High-order accuracy	✓	✗	✓	✓
Explicit semi-discrete form	✓	✓	✗	✓
Wave dominated problems	✓	✓	(✓)	✓
Elliptic problems	✓	(✓)	✓	(✓)

The main drawback of DG-FEM is the relatively high number of degrees of freedom introduced due to the decoupling of the elements [1]. Compared to the continuous FEM, this doubles the number of degrees of freedom for linear elements [1].

1.2 Turbulence

For over five decades, researchers have been actively studying the simulation of turbulent flows, given its significant influence on various practical uses like ocean modeling, weather prediction, aviation, energy generation, and more [7]. From the beginning of turbulence research, effectively forecasting, controlling, and understanding the chaotic aspects of turbulent phenomena has consistently proven challenging. This is due to the intricate combination of disorder and structure and the extensive range of lengths and time scales inherent in turbulent flows. The Navier-Stokes equations, a rather straightforward set of equations, govern the complex behavior of turbulence. However, there are no analytical solutions to even the simplest turbulent flows. The Navier-Stokes equations can exclusively be solved through the aforementioned methods numerically to generate a comprehensive representation of the turbulent flow. In this representation, the flow properties like velocity and pressure are expressed as functions of both space and time [8].

Advanced numerical models, like direct numerical simulation (DNS), can encompass a broad spectrum of dynamic spatiotemporal scales within turbulent flows. This is achievable by accurately representing the entire energy spectrum of turbulence down to the Kolmogorov scale by utilizing exceptionally fine grid resolutions [7]. Because of the continual increase in computer power, DNS can be obtained for various simple turbulent flow problems. However, as the system's dynamics grow into a more detailed and complex version, the need for more accurate and high-fidelity simulations with large numbers of degrees of freedom (DOF) advances, leading to a considerable amount of computational power and computing times ranging from hours to days depending on the availability of resources. This turns out to be a much more severe problem when a serious amount of analysis has to be conducted. [9]. It should also be mentioned that the ongoing increase in computing power and performance driven by Moore's law has reached saturation levels in recent years [10]. As a result, many active research efforts in various fields are committed to developing efficient and robust modeling algorithms. These efforts aim to enhance the achievable quality of numerical simulations while optimizing computational expenses to the fullest extent [7]. For example, instead of trying to solve for the immediate flow pattern, approaches cen-

tered around the Reynolds-averaged Navier-Stokes (RANS) equations are pursued, or the dominant energy-containing patterns are directly calculated while representing the impact of the smaller patterns through modeling, as seen in large eddy simulation (LES). [8]. Closure models are additionally employed to enhance these methods, addressing the consequences of truncated scales in low-fidelity or lower-resolution models.

Distinctive recurring shapes, known as coherent structures, are common features of numerous turbulent flows. It is evident that in flows where these dominant structures prevail, as in the case of turbulent flows, it might be achievable to construct a relatively accurate, low-dimensional model of the flow by focusing solely on these dominant structures, which leads to the development of cost-effective reduced order models (ROMs). The influence of the smaller, less energetic, seemingly less organized component of the flow could then be simulated in an alternative manner as in the case of conventional numerical methods.

The concept of constructing ROMs for nonlinear dynamical systems was introduced in computational fluid dynamics to reduce computational expenses. Over time, various techniques for creating ROMs have been put forward. These methods have found diverse applications such as flow control, incorporating data into weather and climate models, assessing uncertainty, and more. The following section will delve into comprehensive insights regarding these Reduced Order Models.

1.3 Reduced Order Modelling

Reduced order models (ROMs) now play a central role in the modern design, optimization, and control of complex systems. These models are computational representations characterized by significantly fewer dimensions than full order models (FOM), derived from conventional numerical techniques such as the finite element method [11]. They are formulated to decrease the system's degrees of freedom while preserving its essential characteristics [12]. These models aim to offer accurate depictions of system dynamics while substantially lowering the computational costs compared to the original numerical model [13].

ROMs have been successfully demonstrated to provide a fast approximation of a selected objective at significantly lower computing costs when compared to the currently used traditional modeling techniques, drawing a great deal of attention from the CFD research community [7, 14]. The construction of these ROM methodologies is guided by the coherent structures that arise in the flow field. Coherent structures refer to organized and persistent motion patterns resulting from instability in a material region's past or present life. They may possess many shapes, including vortices, eddies, swirls, jets, and waves. These structures are significant because they can considerably impact the transport of momentum, heat, and other quantities within a fluid. They can be used to identify dominant patterns of behavior of the fluid flow that need to be accurately captured in the reduced model, which makes them an attractor or manifold with a lower intrinsic dimension [15, 16, 17].

Recently, there has been a growing interest in employing reduced-order models for linear time-varying and nonlinear systems. The reduced-order methods involve mapping the dynamical system onto subspaces that encompass features of the anticipated solution. In comparison, finite element methodologies utilize basis elements within the subspaces that lack any connection to the actual parameters of the system being approximated [18].

A ROM can be conceptualized as an extension of the latent space or manifold, essentially an approximate simplification in terms of kinematics [19]. In this context, it is assumed that the FOM solution for a function $u(x,t)$ is effectively captured by the basis functions of the ROM :

$$u_r(x,t) = \sum_{i=1}^{r} a_i(t)\varphi_i(x) \tag{1.1}$$

Here, x represents the spatial coordinates, and t denotes time. The term $\varphi_i(x)$ corresponds to the i^{th} basis function, accompanied by its corresponding amplitude or time coefficient $a_i(t)$. These basis functions convey physical significance, particularly when prominent coherent structures are present [19]. While the conventional assumption suggests that the basis functions $\varphi_i(x)$ remain independent of time, and the dynamics are captured within the coefficients a_i, recent studies have shown the potential for time-varying basis functions [20]. However, in the scope of this thesis, we will consider the case in which the basis functions are only a function of space.

ROMs are conventionally divided into two main categories: physics-based and data-based. Physics-based models are constructed using fundamental governing equations, wherein the operators of the full order model (FOM) are projected onto the reduced subspace through methods like Galerkin-type techniques to form a ROM. [21]. These models offer benefits because of their ability to be understood and applied broadly, along with extensive methods for performing stability and uncertainty analysis [21]. However, turbulent and convection-dominated flows are typically expensive to represent using physics-based ROMs, resulting in an increase in the number of modes (or degrees of freedom) to be preserved in ROM. On the other hand, data-based models rely entirely on archival data (hence known as non-intrusive) to determine the underlying relations that drive the system's dynamical evolution [21]. In contrast to intrusive alternatives, non-intrusive ROMs have experienced advantages from the extensive utilization of machine learning (ML) technologies, which have facilitated the development of robust and accurate models. Neural networks, in particular, have been widely used to simulate the dynamical evolution of ROMs. On the other hand, neural networks often lack human comprehensibility and broad applicability, potentially becoming excessively reliant on substantial amounts of data [21].

Different approaches can be utilized to formulate ROM basis functions. Frequently used ones are the Reduced Basis Method (RBM) [22], Proper Orthogonal Decomposition (POD) [16], Proper Generalized Decomposition (PGD) [23], and Dynamic Mode Decomposition (DMD) [24]. When it comes to reducing the order of nonlinear problems with varying parameters, the two most prevalent used methods are the Reduced Basis method (RBM) and the Proper Orthogonal Decomposition (POD) [25]. Although the main principle behind the two methods is the Galerkin projection, which creates a dynamic system spanned by the basis functions, they differ in the computation of the subspace's so-called reduced or reduced-order basis. The RBM is typically used to provide foundations for stationary problems when many parameters are sought as a solution [25]. Recently, POD, which involves basis functions incorporating data from solutions of the dynamic system at predetermined time points referred to as snapshots, has emerged as perhaps the most widely utilized and efficient model reduction method for nonlinear problems. [18].

Karl Pierson [26] and Harold Hotelling [27] laid the mathematical foundations for

the Proper Orthogonal Decomposition by introducing principal component analysis, a statistical approach for simplifying a data set. The method minimizes multivariate data's dimensionality while maintaining as much relevant information as necessary. POD methods are widely used in many research fields for order reduction of practical engineering systems [28]. John Lumley [16] was the first to introduce this method from the fluid dynamics point of view with the name Proper Orthogonal Decomposition. In the context of turbulence, he introduced this technique to break down the random vector field symbolizing turbulent fluid movement into several deterministic functions. Each function aims to contain a portion of the overall fluctuating kinetic energy within the flow, offering a representation based on coherent structures. [16]. Then, to calculate the POD modes for large-scale fluid dynamics problems, Lawrance Sirovich [29] formulated the method of snapshots. The first POD model was developed by Aubry et al. [15] to model the wall region of a turbulent boundary layer. This study appears to be among the first to propose a coherent relationship between low-dimensional chaotic dynamics and realistic turbulent systems.

POD uses an appropriate collection of modes derived from the full-order model, which involves partial differential equations in their discrete or semi-discrete form to capture the dominant aspects of an efficiently multi-degree-of-freedom system and represent it to the necessary precision [30]. As a fluid dynamics problem is approximated using a considerable number of snapshots, generating a substantial amount of basis vectors, the singular values diminish rapidly. As a result, a restricted collection of basis vectors becomes sufficient for reconstructing and approximating the snapshots, as they preserve the primary portion of collective energy [31]. Subsequently, the CFD model is projected onto a more compact domain defined by a subset of appropriate orthogonal modes or POD eigenfunctions through Galerkin projection. This process leads to constructing a ROM, enabling the determination of dynamics within each distinct subspace [32]. The Galerkin projection is formulated to generate a simplified model wherein the truncated linear coefficients of the POD bases transform over time as a collection of ordinary differential equations (ODEs). As a result, the system is substantially reduced in complexity, yet crucial attributes of the governing equations are preserved [33]:

$$\frac{d\mathbf{a}}{dt} = f(\mathbf{a}) \tag{1.2}$$

where **a** is the time coefficient found by POD as in Equation 1.1 and f contains the ROM operators (e.g., vectors, matrices, and tensors) that are constructed during the offline phase using ROM basis functions. The low-dimensional ROM provided by Equation 1.2 is then utilized in the online phase for parameter values that differ from those utilized during training. The computational cost of ROM is orders of magnitude less than that of FOM since it is low-dimensional [19].

The POD technique has been widely applied in numerous fields such as detection, estimation, pattern recognition, image processing, and system control. It is an efficient method for representing complex processes and a valuable tool for storing information for various purposes. It has been employed in fluid mechanics to reconstruct a flow field using limited data [34]. This approach draws inspiration from Everson and Sirovich's method, originally used for reconstructing human face images with incomplete data [35]. In this work, using only limited surface pressure data, they accurately reconstructed the entire pressure field using just 6 POD modes.

Although the POD combined with the Galerkin projection technique has been extensively used for capturing temporal changes in fluid dynamic problems, the ROMs can also be utilized to capture the system's parametric variations. For example, Epureanu et al. [36] developed a model by gathering snapshots not only across different time instances but also across various inter-blade phase angles. Despite constructing the snapshot collection at a single Mach number, the resultant reduced models were effectively employed for flow analysis at different Mach numbers. Bui-Thanh et al. [37, 34] also extended the application of the POD technique to address a steady transonic external aerodynamic problem. In [37], the POD approach was combined with a cubic-spline interpolation technique to create reduced-order models capable of effectively representing parameter variations. This method was then, relying on the dataset of snapshots, utilized to forecast the flow pressure patterns at any Mach number and angle of attack encompassed by the defined range.

Applying Galerkin's reduced-order modeling can also effectively diminish the overall computational time for problems requiring repetitive simulations, such as inverse design optimization problems [38]. The studies conducted by LeGresley and Alonso [39], as well as Bui-Thanh et al. [34], showcased the viability of employing the POD

technique as an economical and reduced-order solution for aerodynamic shape optimization. Bui-Thanh et al.'s approach draws inspiration from Everson and Sirovich's gappy POD reconstruction method [35], while LeGresley and Alonso's approach builds upon the gradient-based methodology for optimizing cost functions.

Conventional discretization methods often result in spaces containing millions of degrees of freedom, whereas ROMs typically encompass spaces of modest scale, ranging from 10 to 100. In practical terms, utilizing ROM approximations can lead to substantial acceleration, often on the order of magnitudes, along with notable reductions in memory demands. Due to the quadratic nonlinearity, the resulting dense system in fluid dynamics applications has triadic interactions with an order of $O(r^3)$ computational load, where r refers to the maintained number of modes.

Although these reduced-order models have many advantages, one drawback is that the number of modes needed to represent a flow can quickly grow too big to handle since the linear superposition of orthogonal modes must approximate nonlinear processes. Even though the excluded POD modes might not incorporate a significant portion of the system's kinetic energy, they still exert a considerable impact on the dynamics of the reduced-order system. This phenomenon contributes to the unsatisfactory performance of the POD Galerkin ROM. In fact, an accurate prediction of the dynamics of the ROM depends on the interaction between the discarded POD modes and those kept in the ROM [40]. This issue is especially noticeable in non-stationary and rapidly changing turbulent flows. This is attributed to the flows being dominated by convection and characterized by multiscale phenomena across extensive spatial and temporal scales [41]. Consequently, a considerable number of modes might be required for the ROM to ensure solution accuracy, thereby constraining computational efficiency. On the other hand, if the number of modes is restricted due to modal truncation, the ROM could experience a substantial information loss. [41]. Therefore, although ROMs can be used for academic test problems with simple dynamics, their usage in real-life engineering problems is limited since the current ROMs do not have enough DOF to represent the under-resolved regime that is critical in complex flows [19, 11]. In recent years, closure models have been established to enhance the accuracy of ROMs and mitigate stability issues arising from truncation within the POD-Galerkin methodology.

1.4 Closure Modelling

The requirement for closure modeling is inherent when simulating dynamic turbulent flows, a task usually tackled through methods like detached eddy simulation (DES) and large eddy simulation (LES), both involving subgrid modeling. Similar challenges are encountered in reduced order models; for instance, the error arising from truncating the POD basis, referred to as "mode truncation instability," is a widely recognized source of inaccuracy in ROMs [41, 42]. Based on the Kolmogorov hypotheses outlined in [43], turbulent flow at a high Reynolds number is distinguished by an energy cascade phenomenon. In this cascade, larger flow scales containing energy undergo disintegration, transferring energy to progressively smaller scales. This process continues until the length scales become sufficiently small; at this point, viscous forces take over and contribute to the dissipation of turbulent kinetic energy [44]. The outcome indicates that when the ROM truncates the less significant higher-order modes within the subspace approximation, its capability to identify the dissipation range of the turbulent energy spectrum diminishes. Consequently, the ROM becomes inefficient at dissipating adequate energy, leading to elevated energy levels and eventual instability [41]. An under-resolved simulation is a numerical simulation in which the number of ROM modes is insufficient to capture the dynamics of the underlying system as in the case of turbulent flows [45]. There are two different ways to increase the numerical accuracy of an under-resolved simulation:

- Increasing the ROM dimension, i.e. r in Equation 1.1
- Adding a low-dimensional closure term.

Given that one of the key advantages of a ROM lies in its reduced computational cost, enhancing numerical accuracy while upholding computational efficiency is crucial [45]. Therefore, addressing the ROM closure problem becomes essential to attain accurate results. This involves modeling the impact of the discarded ROM modes on the ROM dynamics or the temporal development of resolved ROM modes [19].

$$\frac{d\mathbf{a}}{dt} = f(\mathbf{a}) + \textbf{Closure}(\mathbf{a}) \tag{1.3}$$

where the low-dimensional term $\textbf{Closure}(\mathbf{a})$ simulates the impact of the discarded

ROM modes. The closure problem frequently arises when numerically simulating turbulent flows. Even conventional numerical discretization techniques (such as finite volume or finite element methods) necessitate modeling the impacts of sub-grid scales that are inherently present in the under-resolved regions. [19]. Although there is a great number of closure models can be found in the area of CFD, such as large eddy simulation (LES) based on physical understanding from Kolmogorov's statistical theory of turbulence, in the case of ROM, only a few closure models have been explored [19]. The contrast in closure modeling between POD-based reduced-order modeling and traditional turbulence modeling seems even more noticeable, particularly considering that the concept of an energy cascade, a fundamental modeling principle in Large Eddy Simulation (LES), remains applicable within POD. [46].

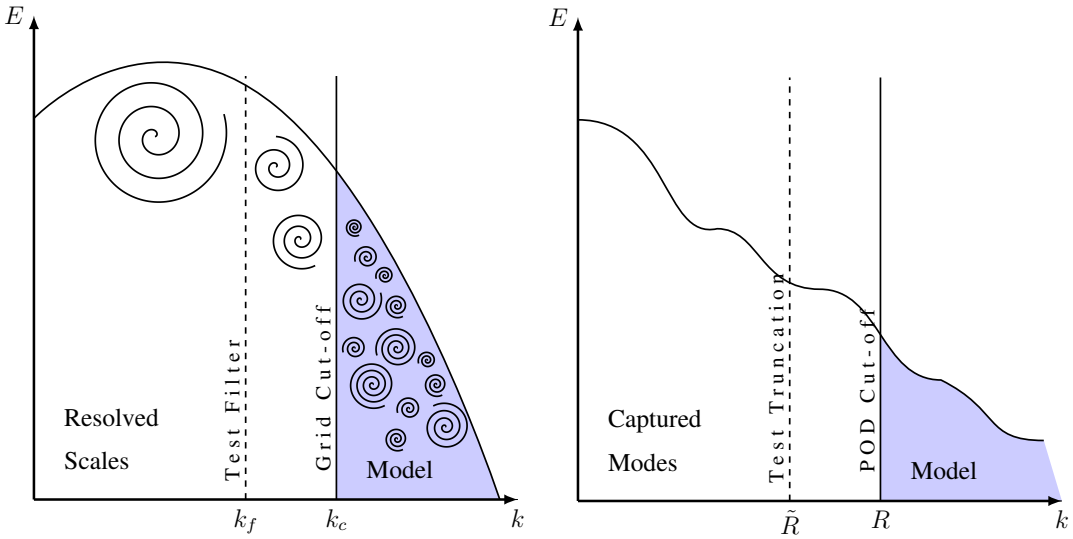


Figure 1.1: Comparison between closure modeling in LES on the left and ROM on the right. In this analogy, the higher k index represents smaller scales. In LES, k_c and k_f represent the grid cut-off and test filtering scales, respectively. Meanwhile, in ROM, R and $\tilde{R}$ denote the number of POD modes retained in the model and the test truncation process, respectively. (adapted from [7]).

The process of constructing a closure model for ROM shares similarities with that of LES closure models [47]. Closure models can also be classified as physics-based or data-driven, as in the case of ROMs. The physics-based category relies heavily on the intuitive understanding of the physical aspects, like the fundamental notion of energy cascade in turbulence and conservation principles. This leads to proposing a specific

form for the closure term by incorporating physical insights such as a dissipative term. These models are also called functional closure modeling methods [19, 42]. Lumley and his team's mixing length model was perhaps the first practical, functional ROM closure model [15]. After this approach, more accurate closure models for ROM have emerged, including the Smagorinsky model, in which eddy viscosity is added to the system's physical viscosity to drain the surplus energy [48, 49]. The variational multiscale (VMS) model [50] is another example of a functional ROM closure model. Within the VMS methodology, the inherent hierarchical arrangement of the ROM basis is exploited to establish precise mathematical expressions for interactions among different spatial scales. This is achieved by leveraging the provided FOM data to construct a closure model that captures these inter-scale interactions. [40, 51].

The streamline-upwind Petrov-Galerkin (SUPG) ROMs are the foundation for another physics-based closure model [52]. This scenario calculates stabilizing parameters based on the underlying finite element discretization and the POD basis truncation with the proper theoretical foundation provided by finite element error analysis. Unfortunately, it has been found that the effectiveness of this kind of closure is typically inadequate when modeling complex flows involving many POD modes [46] and is likely to be useless when employed for simulating chaotic, turbulent flows [42].

Another popular closure modeling technique is Mori–Zwanzig formalism, originally developed by Mori [53] and Zwanzig [54]. It is based on the idea of breaking down the behavior of a complex system into simpler components, such as resolved and unresolved parts, that can be analyzed and modeled more easily [55]. Following the principles of the Mori–Zwanzig formalism, the impact of scales that remain unresolved is depicted in the exact dynamics of the resolved scales through the memory term [56].

The construction of data-driven ROM closures, in which existing data is used to generate the ROM closure model, is the most active field of study in ROM closure modeling [57]. The rise in data-driven closure modeling can be attributed to factors such as the generation of extensive datasets, the accessibility of user-friendly machine-learning libraries, affordable computing resources, and the availability of high-quality training materials, all of which have contributed to its increased adoption

[19]. Given that projection-based ROMs are commonly constructed using snapshot data, whether obtained from experiments or computational sources, it is logical to expand this dataset to estimate the closure term efficiently. For example, enhancing the model can be accomplished by considering insights from POD temporal modes obtained directly from the FOM. This leads to introducing correction coefficients to the initial ROM structure, a process often referred to as calibration [58, 59, 60]. Alternately, the eigenvalues of the linear dynamics matrix can be reassigned to achieve ROM stabilization [61]. Eigenvalue correction typically has to be used with optimization strategies to ensure accuracy. Other techniques [41] use direct snapshot analysis to restore the ROM.

Early investigations about data-driven closure modeling revolved around regression techniques, where the closure model was partially or entirely constructed as a function of accessible information (such as resolved scales). [19]. These methods employ traditional least-squares methods in which the error between the ROM and FOM caused by the discarded modes is minimized. The reader may refer to [51, 62, 63] for a detailed analysis of reduced order models coupled with the data-driven closure models.

Machine learning, i.e., the neural network regression approach, can be a promising choice for data-driven closure modeling. The objective of integrating neural network regression into ROMs is to establish non-intrusive ROM frameworks that exclusively utilize data to grasp the dynamics of the relevant solution space. This technique is appealing because it is portable and can function with various data generation methods while eliminating the requirement to access the governing equations. For example, neural networks can compute stabilization elements like eddy viscosity as a closure model. Another application involves predicting the evolution of POD time coefficients using neural networks based on their previous values [33, 64, 65]. Notably, avoiding the Galerkin projection using artificial neural networks has outperformed the conventional Galerkin projection approach in accuracy. Despite a minimal computational overhead, these models offer dependable predictions even when confronted with parameter variations [64].

1.5 Motivation

This study aims to introduce an automated framework that establishes stable reduced-order models by utilizing Galerkin projection. We focus on achieving this goal while employing a limited number of modes, ensuring computational efficiency without significantly compromising accuracy. A key aspect of our approach involves incorporating closure concepts, which are pivotal in enhancing the robustness and reliability of the resulting reduced-order models.

To increase the accuracy and stability of Galerkin ROMs, we apply data-driven closure model correction on POD temporal coefficients. For the construction of the closure model, machine learning techniques are exploited to find the corrective terms to minimize the error between the full and reduced order models. This methodology offers the potential for significant improvements in the predictive capabilities of reduced-order models and presents opportunities to explore and understand the underlying physical processes of complex flows. Our investigation extends beyond assessing the ROM‘s suitability for the temporal evolution of PDE solutions. We explore the ROM's capacity to solve PDEs across various parameter values, including scenarios with different Reynolds numbers.

Initially, our study embarks on a comprehensive examination of the efficacy of the proposed techniques for reduced-order modeling. This evaluation predicts a nonlinear wave propagation scenario where the viscous Burgers equation governs the underlying dynamics. This equation is often considered a foundational framework for early assessment of numerical techniques in fluid flow studies due to its ability to contain the traits of general nonlinear multidimensional advection-diffusion problems.

As the subsequent test case for the formulated approach, we then seek to validate the adaptability and efficiency of the framework in addressing a more complex and demanding fluid dynamics problem, the 2D vorticity-streamfunction equation. The snapshots are generated using the source code in [66].

Following these cases, the same reduced-order model is utilized to reconstruct the flow field around a square-cylinder governed by the Navier-Stokes equation. We employ libParanumal, a state-of-the-art high-order discontinuous Galerkin finite-element

flow solver, to obtain high-fidelity data. This solver has been developed by the Parallel Numerical Algorithms Group at Virginia Tech [67] and plays a pivotal role in obtaining accurate and detailed flow information for our analyses.

Our closure modeling framework is constructed using JAX [68], a Python-based open-source numerical computing library designed for high-performance machine learning research. The analysis presented in the following sections involves the comparison of different reduction sizes against the closure model. We evaluate the effectiveness of the reduced-order models in contrast to an artificial neural network-based closure model. Additionally, this study highlights the challenges associated with accurately capturing the complex dynamics inherent to the system.

1.6 The Outline of the Thesis

The remainder of the thesis study is structured as follows,

Chapter 2. This chapter presents the methodologies adopted for ROM. It begins with mathematical foundations for basis formulation using POD, through which fundamental flow patterns are uncovered. Then, its integration into the Galerkin projection will be explained. Concluding the chapter, the closure model principle based on artificial neural networks adopted in this study will be introduced, emphasizing JAX.

Chapter 3. This chapter explores the effectiveness of the proposed ROM techniques in predicting nonlinear wave propagation governed by the viscous Burgers equation. The performance of the models is evaluated, considering their capabilities and limitations in capturing the intricate dynamics of the system.

Chapter 4. The 2D vorticity-streamfunction equation is tested in this chapter. Singular value decomposition is applied while obtaining the POD modes. The fundamental idea underlying the closure model based on neural networks remains consistent, with adjustments in the count of layers and neurons.

Chapter 5. The dynamic behavior of fluid flow around a cylinder can be effectively described by the two-dimensional, incompressible, and laminar Navier-Stokes equations for viscous flow. To further test the applicability of the proposed reducer-order

methods in more realistic problems, fluid flow around a square cylinder has been investigated and presented in this chapter. The reason behind eliminating the pressure contribution in the standard Galerkin Reduced Order Model (GROM) is explained with reference to certain common assumptions.

CHAPTER 2

REDUCED ORDER MODELING BASED ON PROPER ORTHOGONAL DECOMPOSITION

Proper Orthogonal Decomposition (POD) and Galerkin Projection are essential mathematical tools in various fields, including engineering, physics, and data analysis. These strategies entail reducing complex high-dimensional data or systems to a smaller number of modes or basis functions that capture the fundamental characteristics of the original data. Examining reduced-order models based on the aforementioned methods is the primary focus of this thesis. Three main steps need to be carried out to construct such a model. The first step is obtaining the so-called snapshots from the numerical solution of the parametrized differential equations. Then, the reduced-order bases are calculated using the POD by using this snapshot data. Finally, the Galerkin reduced-order model (G-ROM) is created by projecting the governing equations to these reduced-order bases. In this section, specifics of both the POD and Galerkin Projection methods, including their theoretical foundations and practical applications to well-known partial differential equations, will be presented.

2.1 Proper Orthogonal Decomposition

Similar to the conventional Fourier decomposition [69], the POD projects a target function onto a set of basis functions called modes. This process yields a finite set of scalar coefficients that characterize the behavior of the given function. The POD provides a distinct collection of modes guaranteed to establish the most optimal linear foundation for representing a finite-size set of observations. In other scientific fields, the same procedure is also called Singular Value Decomposition (numerical math-

ematics), Karhuen–Loeve Decomposition (stochastic) [70], Principal Components Analysis (statistical analysis) [27], Empirical Orthogonal Functions (meteorology) [71], and Singular Spectrum Analysis (time series analysis).

There will often be patterns in space and time in the context of turbulence and other complicated spatiotemporal phenomena [72]. Numerical and experimental data regarding these complex fields can be analyzed using the POD to identify key trends and features, particularly coherent structures. However, the objective here is to find a low-dimensional subspace to build a model by projecting the governing equations, not to examine the data. We want to use the POD to offer a "relevant" collection of basis functions, as described in Chapter 1. The POD will produce fundamental spatial components, which our models will subsequently dynamically reproduce as combinations of time-varying POD modes [72]. In this section, the mathematical background for POD will be displayed by following the notation presented in [72].

2.1.1 Theoretical Background

We consider a collection of M sample or representative data sets, denoted as $\mathbf{u}^k$. These data sets might originate from experimental observations or, as in this instance, numerical solutions to a system of differential equations with varying parameters. To obtain appropriate representations for elements within $\mathbf{u}^k$, it is necessary to project each $\mathbf{u}$ onto suitable basis functions. These basis functions are assumed to belong to an inner product space. In this context, the goal is to identify an optimal basis $\varphi_i(x)_{k=1}^{\infty}$ for the data set, aiming for finite-dimensional representations of the following form:

$$\mathbf{u}_M = \sum_{k=1}^{M} a_k \varphi_k(x) \tag{2.1}$$

We aim to choose φ in a manner that offers a more accurate representation of the ensemble $\mathbf{u}^k$ compared to any alternative linear basis.

We introduce an averaging operator represented as $\langle \cdot \rangle$ to define this optimality criterion formally. This operator represents the spatial integration involved in the inner product operation. The choice of basis functions should be oriented towards maxi-

mizing the average projection of the collection of functions $\mathbf{u}^k$ onto φ.

$$\max_{\varphi} \frac{\left\langle |(\mathbf{u}, \varphi)|^2 \right\rangle}{||\varphi^2||} \tag{2.2}$$

where $|\cdot|$ denotes the modulus and $||\cdot||$ is the L^2 - norm given by

$$||f|| = (f, f)^{1/2} \tag{2.3}$$

and the expression $(\cdot, \cdot)$ represents the inner product of two functions across a predetermined range or domain.

However, numerous local maxima may exist in the maximizing problem stated in Equation 2.2, which would add new basis functions to the decomposition in Equation 2.1. As a result, we have a calculus of variations problem where we would like to maximize $\left\langle |(\mathbf{u}, \varphi)|^2 \right\rangle$ while still meeting the requirement $||\varphi^2|| = 1$. The function that must be maximized in this constrained optimization problem is determined by [69]:

$$J[\varphi] = \left\langle |(\mathbf{u}, \varphi)|^2 \right\rangle - \lambda(||\varphi^2|| - 1) \tag{2.4}$$

where λ is a Lagrange multiplier.

2.1.2 Autocorrelation Matrix of a Finite Dimensional POD

In practical applications, the objective is to find the most suitable low-dimensional representation for data collection. Suppose this data corresponds to the numerical results of a partial differential equation, as represented by a finite element approach. Each data element incorporates a finite element solution obtained at a distinct time point for time-dependent problems or a specific parameter setting for problems involving parameters. For the context of this study, the data is considered to encompass finite element solutions of time-dependent partial differential equations. In this scenario, the snapshots are treated as vectors, and the collection of functions $\mathbf{u}^k$ transforms into a set of M vectors, each with a dimension of N. Consequently, the autocorrelation function takes the form of an autocorrelation tensor of dimensions $N \times N$:

$$\mathbf{C} = \langle \mathbf{u} \otimes \mathbf{u} \rangle \tag{2.5}$$

Here, N represents the total degrees of freedom (DOF) within the system, M represents the count of snapshots, and $\otimes$ denotes the tensor product. The eigenvectors

of the problem correspond to the principal axis of the data points $\mathbf{u}^k$ within the M-dimensional space. The autocorrelation matrix can be seen in Figure 2.1.

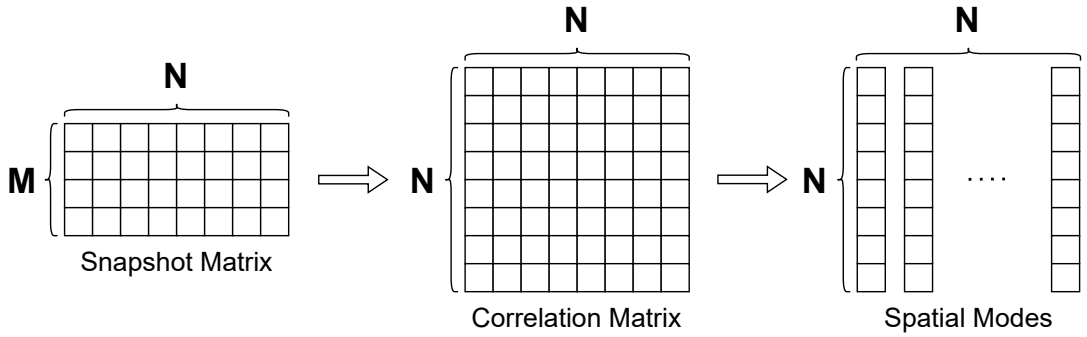


Figure 2.1: Autocorrelation matrix $\mathbf{C}$ resulted from POD, N is the number of nodes and M is the total number of snapshots

The equation mentioned in Equation 2.4 results in the desired optimal POD basis. This basis is formed by the eigenfunctions φ_i of an integral equation whose kernel corresponds to the averaged autocorrelation function.

$$\mathbf{C}\varphi = \lambda\varphi \tag{2.6}$$

The member vectors of the ensemble $\{\mathbf{u}^k\}$ can be decomposed as follows once the basis vectors from this eigenvalue problem have been determined.

$$\mathbf{u} = \sum_{k=1}^{\infty} a_k \varphi_k \tag{2.7}$$

The problem at hand determines the precise definition of the eigenvalues. For example, eigenvalues measure twice the flow's kinetic energy when applied to incompressible fluid flow. If the POD modes are calculated using the H^1 - Sobolov norm rather than the L^2 - norm, then eigenvalues are a measure of dissipation or vorticity [73].

Arranging the eigenvectors by the magnitude of their associated eigenvalues enables the creation of truncated models containing the highest energy components. For instance, this is an appropriate ordering for incompressible flow applications in which the velocities are represented with POD modes, and corresponding eigenvalues represent kinetic energy. Nevertheless, its applicability in different systems varies. Alternate arrangements can be established for systems where achieving a more favorable maximization is feasible and more convenient [69].

2.1.3 Method of Snapshots

Solving the eigenvalue problem is the simplest way to compute the POD modes. Nonetheless, since the dimension N, which corresponds to every point in the mesh for each of the scalar quantities of the finite element basis, is typically quite large, practical implementation can generate significant costs, even when only a limited number of modes are required to depict a function. The method of snapshots was introduced in [29] as an alternative method for obtaining the POD basis in which underlying numerical solutions are called snapshots. The basic idea is to use algebraic manipulations to reduce the large eigenvalue problem to a much smaller one, as seen in Figure 2.2. As a result, the POD basis functions require significantly less computational effort to obtain.

If φ is an eigenvector, it can be expressed as a linear combination of the snapshots throughout the basis.

$$\varphi = \sum_{k=1}^{M} w_k \mathbf{u}^k \tag{2.8}$$

where the coefficients w_k remain to be determined. The N-dimensional eigenfunction problem may then be written as

$$\left(\frac{1}{M} \sum_{i=1}^{M} \mathbf{u}^i \otimes \mathbf{u}^i, \sum_{k=1}^{M} w_k \mathbf{u}^k \right) = \lambda \sum_{k=1}^{M} w_k \mathbf{u}^k \tag{2.9}$$

Rearranging the left-hand side to yields

$$\sum_{i=1}^{M} \left[\sum_{k=1}^{M} \frac{1}{M} \left(\mathbf{u}^i, \mathbf{u}^k \right) w_k \right] \mathbf{u}^i, \tag{2.10}$$

it may be concluded that the solution of Equation 2.9 has coefficients w_k such that

$$\sum_{i=1}^{M} \frac{1}{M} \left(\mathbf{u}^i, \mathbf{u}^k \right) w_k = \lambda a_i \quad i = 1, \ldots, M. \tag{2.11}$$

Thus, with the method of snapshots, the resulting elements of the modified autocorrelation matrix are given by

$$\mathbf{C}_{ij} = (\mathbf{u}(\mathbf{x}, t^{(i)}), \mathbf{u}(\mathbf{x}, t^{(j)})) \tag{2.12}$$

Here, the index i corresponds to the i-th instance of the solution snapshot, with i and j taking values from 1 to M, where M represents the total number of provided snapshots. The integral expression represents the inner product of two snapshots, i and j,

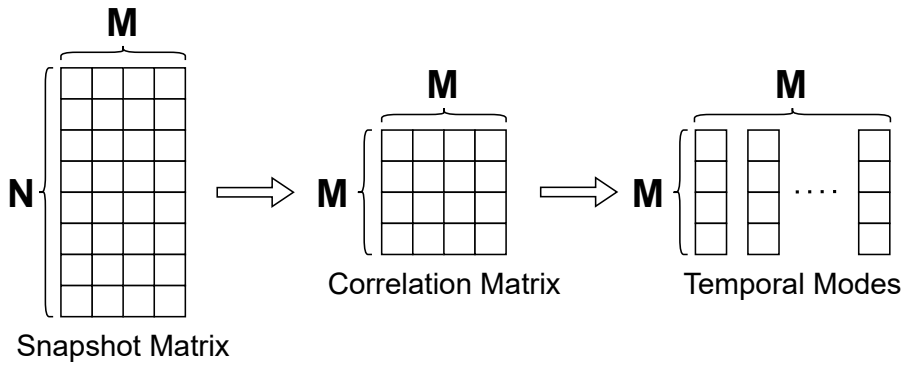


Figure 2.2: Correlation matrix C resulted from method of snapshots, N is the number of nodes and M is the total number of snapshots

resulting in a matrix $\mathbf{C}$. This matrix is non-negative, definite, and symmetric, ensuring its eigenvalues are non-negative, and it has a complete set of orthogonal eigenvectors. The computation of these eigenvectors for matrix $\mathbf{C}$ serves as an intermediary step in determining the actual POD modes.

$$\mathbf{C}W = \Lambda W \tag{2.13}$$

where $\Lambda = \operatorname{diag}[\lambda_1, \lambda_2, \ldots, \lambda_M]$ is a diagonal matrix containing the eigenvalues of this decomposition and $W = \left[\boldsymbol{w}^1, \boldsymbol{w}^2, \ldots, \boldsymbol{w}^M\right]$. The eigenvalues are stored in descending order, $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_M$.

The eigenvalues obtained using the snapshot method align with those acquired from the direct method. Nonetheless, the spatial coefficients and temporal modes derived through the snapshot method differ from the spatial modes and temporal coefficients of the direct method by a scaling factor. In the direct method, spatial modes are orthonormal due to their origin from the eigendecomposition of the symmetric matrix $\mathbf{C}$, while temporal coefficients are not. Conversely, temporal modes are orthonormal in the snapshot method, but spatial coefficients are not. To achieve matching outcomes between both approaches, it is necessary to normalize each spatial coefficient within the snapshot method and adjust the scaling of temporal modes accordingly [74].

Then, the orthogonal POD basis functions can be obtained by projecting the field

variable u into eigenvectors resulting from the Equation 2.13. They can be written as

$$\begin{gathered}\varphi_1(\mathbf{x}) = \sum_{k=1}^{M} w_k^1 u\left(\mathbf{x}, t^{(k)}\right), \quad \varphi_2(\mathbf{x}) = \sum_{k=1}^{M} w_k^2 u\left(\mathbf{x}, t^{(k)}\right), \quad \ldots, \\ \varphi_M(\mathbf{x}) = \sum_{k=1}^{M} w_k^M u\left(\mathbf{x}, t^{(k)}\right)\end{gathered} \tag{2.14}$$

where w_k^n is the kth component of the n th eigenvector $\boldsymbol{w}^n$. The eigenvectors need to be normalized to fulfill the requirement of being orthogonal to each other and having a unit length. It can be shown that, for this to be true, the eigenvector $\boldsymbol{w}^n$ must satisfy the following equation:

$$\sum_{k=1}^{M} w_k^n w_k^n = \frac{1}{\lambda_n} \tag{2.15}$$

where λ_n is the n th eigenvalue associated to the eigenvector $\boldsymbol{w}^n$. In practical cases, most algorithms used to solve the eigensystem, as outlined in Equation 2.13, provide an eigenvector matrix W where all the eigenvectors are normalized to have a unit length. In that case, the orthogonal POD bases are given by

$$\varphi_n(\mathbf{x}) = \frac{1}{\sqrt{\lambda_n}} \sum_{k=1}^{M} w_k^n u\left(\mathbf{x}, t^{(k)}\right) \tag{2.16}$$

where $\varphi_n(\mathbf{x})$ is the n th POD basis function.

It's important to emphasize that the POD should be performed individually for each field variable in the case of multidimensional governing equations.

Then, the field variables and their temporal evolution can be approximated using these POD basis functions and their corresponding time coefficients, where r is considerably smaller than M.

$$\mathbf{u}_r'(x,t) = \sum_{k=1}^{r} a_k(t)\varphi_k(x) \tag{2.17}$$

Introducing the mean velocity subtracted at the initial stage of our analysis, a reconstructed flow field can be generated.

$$\mathbf{u}_r = \bar{\mathbf{u}}(x) + \sum_{k=1}^{r} a_k(t)\varphi_k(x), \tag{2.18}$$

2.1.4 The Singular Value Decomposition

The equivalent results obtained from direct POD and snapshot POD arise from their strong connection to the singular value decomposition (SVD) of the snapshot matrix.

The snapshot matrix can be represented as $\mathbf{S}$ with dimension $N \times M$:

$$\mathbf{S} = \begin{bmatrix} | & | & & | \\ \mathbf{u}^1 & \mathbf{u}^2 & \cdots & \mathbf{u}^M \\ | & | & & | \end{bmatrix} \tag{2.19}$$

The SVD is a unique matrix decomposition that exists for every complex-valued matrix $\mathbf{S}$ with dimension $N \times M$:

$$\mathrm{S} = \mathbf{U}\Sigma\mathbf{V}^* \tag{2.20}$$

In this context, $\mathbf{U}$ is an $N \times N$ matrix, $\mathbf{V}$ is an $M \times M$ matrix with columns that are orthogonal and normalized, and Σ is an $N \times M$ matrix containing non-negative real values on its diagonal and zeros elsewhere. The symbol * refers to the conjugate transpose operation.

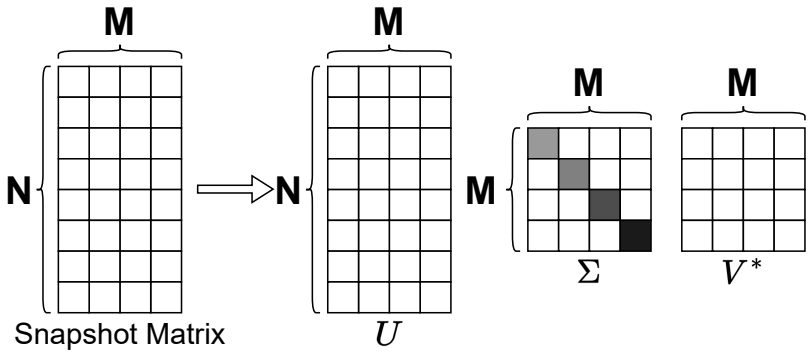


Figure 2.3: Singular value decomposition of the snapshot matrix, N is the number of nodes and M is the total number of snapshots

The non-zero diagonal elements of $\mathbf{\Sigma}$ are positive numbers arranged in decreasing order, i.e., $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r \geq 0$. These are known as the singular values of matrix $\mathbf{S}$. The SVD method can transform any rectangular matrix into a diagonal form, a distinctive feature compared to eigenvalue decomposition, which applies only to square matrices. The schematic of the SVD can be seen in Figure 2.3.

Most widely used numerical implementations are well-established, and many modern programming languages offer straightforward interfaces. That being said, the SVD of a matrix will be computed in this thesis using the Python libraries, especially the most common one `NumPy` [75].

2.1.5 Practical Aspects

In practical applications, constructing the POD basis commonly involves utilizing the snapshot technique, similar to the approach outlined in Section 2.1.3. This can be achieved by solving the eigenvalue problem described in Section 2.1.2 or by performing Singular Value Decomposition (SVD) on a slightly adapted snapshot matrix. A multitude of software tools are accessible to perform these operations. For instance, the linear algebra modules in Python and MATLAB provide diverse algorithms for solving eigenvalue problems and computing the SVD [75, 76].

The dimension of the applied POD basis stands out as a critical factor influencing the model's accuracy. In the existing literature, a heuristic approach is commonly employed to ascertain the POD base's rank [25]. This method relies on a metric termed the energy ratio, denoted as relative information content (RIC), which quantifies the proportion of captured energy in relation to the system's total energy. The energy ratio RIC is given by

$$RIC = \frac{\sum_{k=1}^{r} \lambda_k}{\sum_{k=1}^{M} \lambda_k} \times 100, \tag{2.21}$$

where λ_i are the POD eigenvalues from Equation 2.11. For example, suppose the intention is to encompass at least 99.9% of the overall energy within the POD basis. In that case, the value of r is selected as the smallest whole number for which the cumulative residual information content RIC satisfies the condition $RIC \geq 0.999$.

In cases where the POD eigenvalues exhibit a rapid decline, only a limited number of POD modes is necessary to simulate elements from the snapshot matrix effectively. However, in scenarios like convection-dominated problems, where the POD eigenvalues decrease slower, a larger number of POD modes are required to capture a specific energy threshold within the system adequately.

The other important aspect of POD lies in its orthogonality, which implies that in a suitable function space, one can write:

$$(\varphi_i, \varphi_j) = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \tag{2.22}$$

The orthonormality characteristic is beneficial because it indicates that each coefficient a_i^k depends exclusively on the spatial mode φ^k [74].

2.2 Galerkin Reduced Order Modeling

In numerical fluid flow simulations, one can only integrate a finite number of differential equations or look for solutions on a finite spatial grid. To address this limitation, Galerkin projection emerges as a technique for converting an evolution problem or partial differential equation of infinite dimensions into a finite ensemble of ordinary differential equations [72]. This section provides a concise overview of this method.

The Galerkin projection, also known as the Galerkin method, was introduced by the Russian mathematician and engineer Boris Galerkin. This method is designed to approximate an infinite-dimensional partial differential equation by converting it into a system of ordinary differential equations. This conversion involves projecting the equation onto a finite-dimensional space constructed using suitable basis functions [25]. The Galerkin method is closely related to the weak formulation of the partial differential equation and finds significant application, notably in the Galerkin finite element method.

The basis functions of the standard finite element method are piecewise polynomial functions within the cells of some computational mesh, continuous over the cell edges, and non-zero only within the cell [38]. A finite element model's spatial resolution can be tailored to the problem. For example, the underlying mesh can be refined locally (h-adaptivity), the polynomial degree can be adjusted locally (p-adaptivity), or the mesh points can be relocated (r-adaptivity) [38]. Even when adaptivity is used, the solution space can become quite high-dimensional for some applications [38].

A Galerkin reduced-order model (GROM), as opposed to a typical finite element

model, is based on basis functions more closely related to the solutions of the examined PDE problem, i.e., snapshots. Several techniques exist for deriving the Galerkin reduced-order model's basis functions. One such strategy is the Taylor method, wherein basis functions are constructed using derivatives of solutions at a specific reference point within the parameter space. Another approach, the Lagrange method, involves solutions to the problem at different parameter values as the foundation for basis functions. The Hermite method, which combines aspects of both the Lagrange and Taylor methodologies, incorporates solutions and their first derivatives at multiple parameter values to establish the basis functions [38, 77, 78, 79]. This thesis will obtain the basis functions of interest through POD.

2.2.1 Galerkin Projection

The system of PDEs that represents the dynamics of a variable of interest $\mathbf{u}$ can be depicted in its basic form:

$$\frac{d\mathbf{u}}{dt} = F(\mathbf{u}) \tag{2.23}$$

where $F(\cdot)$ represents a nonlinear operator that may involve spatial derivatives, integrals, and $\mathbf{u} = \mathbf{u}(\mathbf{x}, t)$ is a variable defined on a spatial domain Ω. The solution $\mathbf{u}$ corresponds to an appropriate phase or state space, which we designate by X, at each (fixed) time t. Hence, if $\mathbf{u} = \mathbf{u}(\mathbf{x}, t)$ is a function defined over the spatial domain Ω, X should be a space of functions defined on the domain Ω that is sufficiently flexible to encompass all feasible solutions.

Given a finite-dimensional linear subspace S of X, we wish to determine a dynamical system that evolves on S and approximates Equation 2.23 in some sense. This dynamic system may be denoted

$$\frac{d\mathbf{u}}{dt} = F_S(\mathbf{u}) \tag{2.24}$$

where $\mathbf{u}(\mathbf{x}, t) \in S$ and F_S is a vector field on S. Galerkin projection specifies this vector field by

$$F_S(\mathbf{u}) = PF(\mathbf{u}) \tag{2.25}$$

where $P : X \to S$ is the orthogonal projection onto S. To apply this method explic-

itly, we expand $\mathbf{u}(\mathbf{x}, t)$ as

$$\mathbf{u}_r(\mathbf{x}, t) = \sum_{k=1}^{r} a_k(t)\varphi_k(\mathbf{x}) \tag{2.26}$$

where $\varphi_k(\mathbf{x})$ are basis functions described by POD modes that span the subspace S.

Consider a general nonlinear PDE that describes the evolution of a vector field $\mathbf{u}(\mathbf{x}, t)$, over time. This equation includes both a linear component and quadratic nonlinear terms and can be represented as

$$\frac{\partial \mathbf{u}}{\partial t} = L[\mathbf{u}] + N[\mathbf{u}, \mathbf{u}] \tag{2.27}$$

The operator L represents a linear operation, while N represents a quadratic nonlinear operation. Galerkin projection of the Equation 2.27 onto $\boldsymbol{\varphi}_k$, the results in an expression as follows:

$$\left(\frac{\partial \mathbf{u}}{\partial t}, \varphi_k\right) = (L[\mathbf{u}], \boldsymbol{\varphi}_k) + (N[\mathbf{u}, \mathbf{u}], \boldsymbol{\varphi}_k) \tag{2.28}$$

By inserting Equation 2.26 into Equation 2.28 and then simplifying the resulting equation using the orthonormality condition provided in Equation 2.22, we can express the implementation of the GROM in the following manner:

$$\frac{d\hat{a}_k}{dt} = \mathcal{B}_k + \sum_{i=1}^{r} \mathcal{L}_{ik}\hat{a}_i + \sum_{i=1}^{r}\sum_{j=1}^{r} \mathcal{N}_{ijk}\hat{a}_i\hat{a}_j, \quad \text{for} \quad k = 1, 2, \ldots, r \tag{2.29}$$

Here, a_k represents the vector containing the time coefficients of the GROM, $\mathcal{B}_k$ is a vector with dimensions $r \times 1$, $\mathcal{L}_{ik}$ is a matrix of size $r \times r$, and $\mathcal{N}_{ijk}$ is a tensor with dimensions $r \times r \times r$.

The system given by the above equation consists of r coupled ODEs for modal coefficients, a_k, which can be solved numerically by any suitable time integration scheme. Additionally, the POD basis functions and ROM operators ($\mathcal{B}_k$, $\mathcal{L}_{ik}$, $\mathcal{N}_ijk$) can be precomputed from the data, which makes the system more efficient.

ODEs obtained from the Galerkin projection explicitly depend on the parameter Re through a linear operator. As a result, we derive a fluid flow model that is applicable across all Re. However, it is important to note that the POD modes are optimized specifically for the Re used while obtaining the FOM, so the model's accuracy is expected to be highest near that Re. It has been suggested that the model's applicability

range can be expanded by incorporating snapshots from various parameter values, effectively "stacking" the ensemble [80]. We will apply this technique to compute the optimal POD modes by creating a snapshot matrix that consists of snapshots obtained from different Re.

2.2.2 Inner Product

The role of the inner product within the GROM procedure is crucial. Primarily, it plays a fundamental role in establishing the underlying Hilbert space for the analytical process. Furthermore, it defines how the governing equations are projected onto the POD basis, thus determining the mathematical aspect that the POD basis most effectively represents [81]. This step subsequently forms the basis for constructing the GROM dynamical model.

Most GROM applied to fluid flow analysis have commonly utilized the incompressible Navier-Stokes equations as the governing equations. In such problems, a suitable and intuitive selection for the inner product is the L^2 inner product. This inner product is defined over the spatial domain Ω, reflecting the region of interest. L^2 - norm of a function f is also given in Equation 2.3.

$$(u, v) = \int_{\Omega} uv \, \mathrm{d}\Omega \tag{2.30}$$

Considering the solution vector for the velocity denoted as u, the chosen inner product aligns with the overall kinetic energy across the system. This interpretation conveys physical significance to the POD basis, as these modes are ideally suited to represent the kinetic energy distribution within the ensemble from which they are derived.

The computations of integrals can be approximated along the specified axis using numerical integration methods such as the trapezoidal rule or the well-known Simpson's 1/3 rule. In this thesis, the composite trapezoidal rule is utilized. This rule operates by estimating the area beneath the function's curve $f(x)$ as a trapezoid and then determining its enclosed area. Consequently for $\int_a^b f(x)dx$,

$$\begin{aligned} h &= \frac{b-a}{n} \\ p_i &= \frac{y_i + y_{i-1}}{2} h \end{aligned} \tag{2.31}$$

Given a subinterval length represented by h and the function values y_i obtained from $f(x_i)$ for $i =$ ranging from $i = 1, \ldots, n$ the term p_i represents the area of the trapezoid over the interval $[x_{i-1}, x_i]$ [82]. The complete integration over the interval $[a, b]$ can be expressed as :

$$\int_a^b f(x)dx = \sum_{i=1}^{n} p_i \tag{2.32}$$

2.2.3 Initial and Boundary Conditions

The algorithms employed to calculate the POD basis $\{\varphi_i\}$ use snapshots that may not always exhibit zero values on the Dirichlet boundary. Each POD mode can be viewed as a specific linear combination of snapshots. Consequently, the POD modes derived from the original snapshots, as demonstrated in Equation 2.8, tend to have non-zero values along the Dirichlet boundary in most cases. Consequently, when computing the POD basis and constructing a ROM, careful attention must be given to handling the boundary conditions to ensure that the resulting POD modes are zero on the Dirichlet boundary.

There are numerous approaches can be found in the literature [83, 84, 85, 86] for dealing with both steady and time-dependent inhomogeneous Dirichlet boundary conditions. For the sake of this thesis, however, only steady inhomogeneous Dirichlet boundary conditions will be presented. The reader may refer to [25] for more details.

The finite element solution of the problem with the steady Dirichlet boundary condition is represented with snapshots $\left\{\mathbf{u}^k\right\}_{k=1}^{M}$ where $\mathbf{u}$ is a function of both space x and time t.

$$u(x,t) = g(x) \text{ on } \quad [0,T] \times \Gamma_{\mathrm{D}} \subset \Gamma. \tag{2.33}$$

In this approach, the POD is employed to modified snapshots $\left\{\mathbf{u}^k - \bar{u}\right\}_{k=1}^{M}$ rather than raw snapshots $\left\{\mathbf{u}^k\right\}_{k=1}^{M}$ where $\bar{u}$ is an extension of g into Ω which satisfies the boundary condition. This function must then be added to the reduced-order approximation of u. The most common choice for $\bar{u}$ is the temporal average of the snapshots, i.e.

$$\bar{u} := \frac{1}{M} \sum_{k=1}^{M} u_k. \tag{2.34}$$

As a result, the POD modes are calculated from the fluctuations of the snapshots. This approach is also recognized as the centered-trajectory method [25]. Employing the snapshot averages $\bar{u}$ displays a notable benefit in flow-related scenarios by maintaining the linear characteristics of the solution, including features like divergence within the velocity field.

The mean field $\bar{u}$ and the elements of the POD basis $\{\varphi_k\}$ are constructed from linear combinations of the snapshots $\left\{\mathbf{u}^k\right\}_{k=1}^{M}$, resulting in divergence-free velocity POD modes and an approximation that automatically fulfills the continuity equation [87, 88, 89].

The modified snapshots $\left\{\mathbf{u}^k - \bar{u}\right\}_{k=1}^{M}$ satisfy homogeneous Dirichlet boundary conditions. As a result, creating a ROM using the POD basis functions is possible, which also satisfies homogeneous Dirichlet boundary conditions.

Regarding the initial condition for a projection-based reduced-order model, i.e., Equation 2.26, a_k^0 is typically derived by projecting $u^0 - \bar{u}$ in the L^2 sense onto the POD basis as follows

$$a_k^0 = \left(u^0 - \bar{u}, \varphi_k\right), \quad k = 1, \ldots, r \tag{2.35}$$

where $\bar{u}$ can be found using Equation 2.34 and r is the number of POD modes. Consequently, the reduced-order approximation of the initial condition u^0 has the form

$$u^0 \approx \bar{u} + \sum_{k=1}^{r} a_k^0 \varphi_k \tag{2.36}$$

The POD space represents the best possible approximation in the L^2 sense of the initial condition u^0. Unless explicitly stated, all projection-based ROMs presented in the thesis will adopt this initial condition.

2.2.4 Numerical Methods

The first and second-order derivatives are calculated by employing second-order accurate central differences in the interior points and second-order accurate one-sided (forward or backward) differences at the edges to approximate the differential operators (linear and nonlinear terms) in the GROM.

$O(\Delta x^2)$ centered difference approximations for interior points:

$$\begin{aligned} f'(x): &\quad \{f(x+\Delta x) - f(x-\Delta x)\}/(2\Delta x) \\ f''(x): &\quad \{f(x+\Delta x) - 2f(x) + f(x-\Delta x)\}/\Delta x^2 \end{aligned} \tag{2.37}$$

$O(\Delta x^2)$ forward and backward difference approximations for boundaries:

$$\begin{aligned} f'(x): &\quad \{-3f(x) + 4f(x+\Delta x) - f(x+2\Delta x)\}/(2\Delta x) \\ f''(x): &\quad \{2f(x) - 5f(x+\Delta x) + 4f(x+2\Delta x) - f(x+3\Delta x)\}/\Delta x^3 \end{aligned} \tag{2.38}$$

$$\begin{aligned} f'(x): &\quad \{3f(x) - 4f(x-\Delta x) + f(x-2\Delta x)\}/(2\Delta x) \\ f''(x): &\quad \{2f(x) - 5f(x-\Delta x) + 4f(x-2\Delta x) - f(x-3\Delta x)\}/\Delta x^3 \end{aligned} \tag{2.39}$$

Once Equation 2.29 is derived through applying Galerkin projection, the system's temporal evolution at any parameter can be predicted using a suitable time integration technique. In the scope of this thesis, we want to approximate the solution to a first-order differential equation given by

$$\frac{dy(x)}{dt} = y'(x) = f(x,y), \quad \text{with } y(x_0) = y_0 \tag{2.40}$$

The third-order Runge-Kutta method involves approximating the solution of the initial value problem by calculating the integrand, denoted as $f(x,y)$, three times during each step. For step $i+1$.

$$y_{i+1} = y_i + 1/6\,(k_1 + 4k_2 + k_3)\,, \tag{2.41}$$

where

$$\begin{aligned} k_1 &= hf\,(x_i, y_i)\,, \\ k_2 &= hf\,(x_i + h/2, y_i + k_1/2)\,, \\ k_3 &= hf\,(x_i + h, y_i - k_1 + 2k_2)\,, \end{aligned}$$

and $x_i = x_0 + ih$.

2.2.5 A Summary of the Procedure for Building GROM

The following algorithm outlines the steps of the Galerkin projection-based ROM (GROM) process:

- Take the ensemble average of the snapshots obtained from the FOM. $\bar{u}(\mathbf{x}) = \frac{1}{M}\sum_{k=1}^{M} u^{(k)}(\mathbf{x})$

- Decomposed each snapshot into its mean and instantaneous fluctuation parts. $u\left(\mathbf{x}, t^k\right) = \bar{u}(\mathbf{x}) + u'\left(\mathbf{x}, t^k\right)$

- Construct a data correlation matrix of the fluctuating part, $\mathbf{C} = [c_{ij}]$ from the snapshots where $c_{ij} = \int_{\Omega} u'\left(\mathbf{x}, t^i\right) u'\left(\mathbf{x}, t^j\right) d\Omega$. Here, i and j refer to the snapshot indices.

- Compute the optimal POD basis functions by solving the eigenvalue problem denoted by Equation 2.12 or by applying singular value decomposition stated in Equation 2.20.

- Using the eigenvalues stored in descending order in the diagonal matrix, Λ, or singular values stored in the diagonal matrix Σ, define the orthogonal POD basis functions for the velocity field.

- Rewrite the fluctuating component of the field variables into the POD modes.

$$\mathbf{u}'_M(\mathbf{x}, t) = \sum_{k=1}^{M} a_k(t)\varphi_k(\mathbf{x})$$

  where $a_k(t)$ is the time-dependent modal coefficients and $\varphi_k(\mathbf{x})$ refer to the POD modes.

- Select r modes, where r is much smaller than M, in a way that these r modes capture the most significant energy, aligning with the largest eigenvalues $(\lambda_1, \ldots, \lambda_r)$. The complete representation of the field variables can then be expressed as follows:

$$\mathbf{u_r} = \bar{\mathbf{u}}(\mathbf{x}) + \sum_{k=1}^{r} a_k(t)\varphi_k(\mathbf{x})$$

- Conduct an orthogonal Galerkin projection by multiplying the governing equation with the POD basis functions and integrating across the domain Ω. This process leads to the dynamic GROM system for $\hat{a_k}$ as in Equation 2.29.

- Numerically solve the above equation using an appropriate time integration method, such as the Runge–Kutta method, to predict either the time evolution of the system or the dynamic behavior of the system at an unknown parameter.

2.3 Closure Modelling

As outlined in Section 1.4, there exist primarily two approaches for enhancing the accuracy of Galerkin projection-based ROMs (GROMs). The initial method involves augmenting the number of retained modes, i.e., expanding the ROM's dimension during the procedure. Yet, owing to the non-linearity inherent in the resultant ROM, the computational cost of the GROM scales with $O(r^3)$, thereby imposing significant limitations on the practical use of larger ROM dimensions. Therefore, a secondary approach will be employed in this thesis, which entails introducing a low-dimensional closure term predicted by the artificial neural network (ANN).

A basic feed-forward ANN composed of multiple layers is employed in closure modeling. The physics-constrained Galerkin projection-based ROM (GROM) is maintained to effectively capture the behavior of the larger scales within the system, and the closure model is built as an extension of the GROM. This decision enhances the framework's interpretability and extends its applicability across diverse control parameters.

The training process of an ANN involves minimizing the error between the intended target and the input values to determine an optimal set of parameters. This approach is called supervised learning, wherein labeled data is employed for optimizing the model using gradient-based techniques. The optimized parameters encompass biases and linear weights that effectively encapsulate the underlying correlation between the input values and the intended targets. Subsequently, these parameters can be utilized to predict target values for new input data.

In contrast to traditional statistical regression models, the notable advantage of utilizing the ANN methodology is its ability to yield accurate results even with relatively smaller training data sets. In the following sections, we will present the specific components of the ANN architecture that pertain to our particular test case.

2.3.1 Artificial Neural Networks

An artificial neural network is a computational technique that constructs multiple interconnected processing units. These units, also known as neurons or nodes, form a network with various cells, linking input data to the desired output. Neurons are tightly interconnected and arranged in layers within the neural network. The input layer takes in the initial data, while the output layer produces the outcome. One or more hidden layers are usually situated between these two layers.

The artificial neuron is designed to replicate the fundamental traits of a biological neuron's behavior. Similar to its biological counterpart, the artificial neuron accepts multiple inputs representing other neurons' outputs. Each input is scaled by a specific weight, akin to the strength of synapses in biological neurons. These weighted inputs are summoned and subjected to an activation function, ultimately deciding the neuron's output as in Figure 2.4.

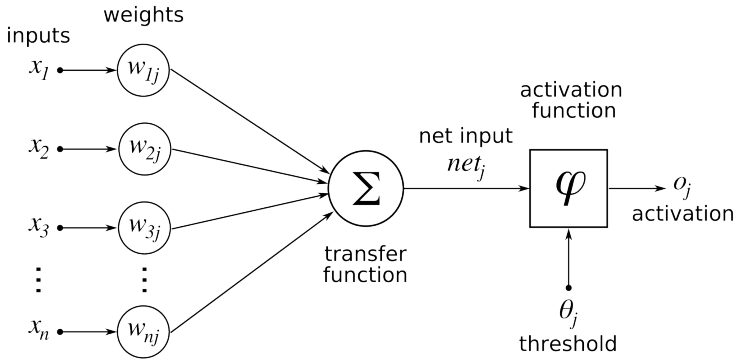


Figure 2.4: Schematic of an Artificial Neuron

The most basic form of a neural network, often called a fully connected network or multilayer perceptron (MLP), comprises multiple interconnected artificial units. The term "perceptron" denotes an artificial neuron that takes multiple input values, each multiplied by a corresponding weight, and produces an output through an activation function, while "multi" in multilayer perceptron indicates the incorporation of multiple concealed layers, enabling the network to comprehend intricate data relationships. The mathematical representation of the artificial neuron can be expressed as follows:

$$y = \sigma\left(\sum_{k=1}^{N} w_k x_k + b\right), \tag{2.42}$$

Here, σ denotes the activation function, and N represents the number of preceding data units. Both the synaptic weights and the activation function influence the performance of the artificial neuron. The activation function brings non-linear characteristics to the neuron's output. The absence of such a function restricts a neural network to linear mappings. Typical activation functions include the rectified linear unit (ReLU), hyperbolic tangent, sigmoid, and swish. They are represented in Figure 2.5.

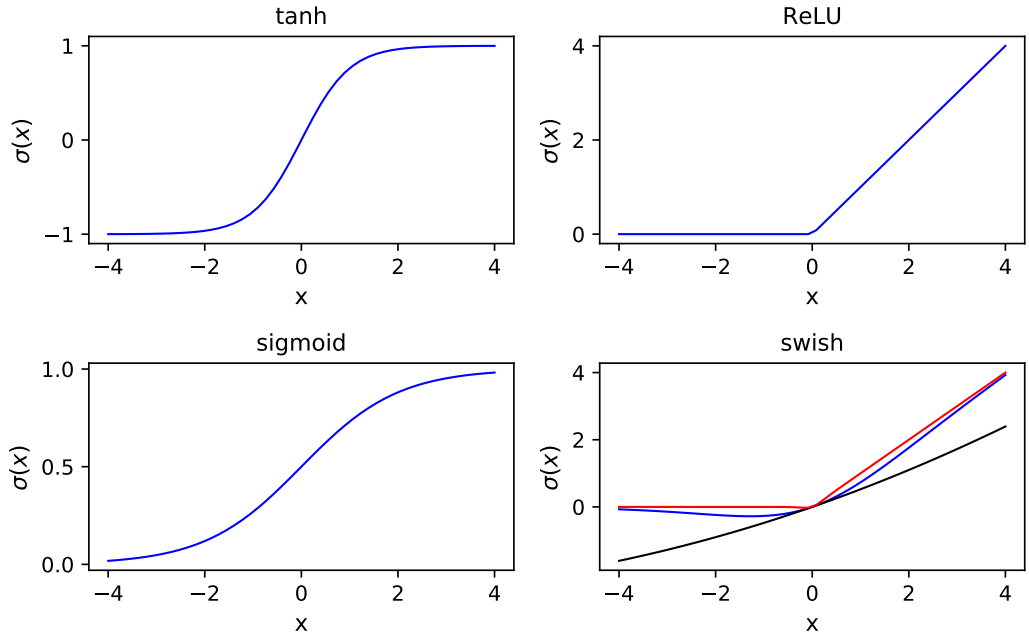


Figure 2.5: Plots depicting frequently used activation functions

Nonlinear activation functions are applied within the hidden layers. However, an identity function serves as the activation function in the output layer. This choice is because the variable in the output layer typically represents real-valued targets for regression or class scores for classification. In general, an L-layered network can be written as:

$$\mathcal{N}^0(\mathbf{x}) = \mathbf{x} \tag{2.43a}$$

$$\mathcal{N}^i(\mathbf{x}) = \sigma(\mathbf{W}^i\mathcal{N}^{i-1}(\mathbf{x}) + \mathbf{b}^i) \tag{2.43b}$$

$$\mathcal{N}^L(\mathbf{x}) = \mathbf{W}^L\mathcal{N}^{L-1}(\mathbf{x}) + \mathbf{b}^L, \tag{2.43c}$$

Here, $\mathbf{W}$ denotes the weight matrix, and $\mathbf{b}$ represents the bias vector. The matrix $\mathbf{W}$ is of size $(m \times n)$, where n represents the input vector's length, and m indicates the

length of the output vector. $\mathbf{b}$ is a vector with dimensions $(m \times 1)$. This multiplication yields an output vector y with dimensions $(m \times 1)$. In a matrix representation, inter-layer calculations can be expressed as follows:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \sigma \left(\begin{bmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,n} \\ w_{2,1} & w_{2,2} & \dots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & \dots & w_{m,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \right) \tag{2.44}$$

In the majority of neural networks, a portion of historical data with known correct outputs is employed to establish the weights, biases, or, in other words, hyperparameters through a procedure known as neural network training. This involves assigning initial values to the hyperparameters and systematically adjusting them step by step until a satisfactory parameter configuration is achieved. The core technique used for this purpose is referred to as Backpropagation.

The concept involves treating the loss resulting from incorrect classifications the neural network makes as a function related to the network's hyperparameters. The goal is to select hyperparameter values that minimize this loss. A widely used loss function is the mean squared error (MSE), which calculates the average of the squared difference between the predicted and actual values. It can be mathematically represented as:

$$\mathcal{L}_{MSE} = \frac{1}{N_d} \sum_{i=1}^{N_d} |\hat{y}_i - y_i|^2, \tag{2.45}$$

Given the primary objective of accurately predicting the exact value, the main focus is to minimize the loss. This optimization problem can be depicted as follows:

$$\theta^* = \arg\min_{\theta} J(\theta; \mathbf{x}), \tag{2.46}$$

Here, J represents the objective function, as seen in Equation 2.45. The hyperparameters can be refined in consecutive iterations by approximating solutions to this minimization problem. Gradient Descent is an iterative optimization method extensively applied in this process to reduce a loss function.

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; \mathbf{x}), \tag{2.47}$$

where η is the learning rate.

Gradient descent aims to locate the nearby minimum of the loss function through successive steps in the direction that minimizes loss the most. The learning rate is pivotal in this mechanism, as it defines the step size and can impact convergence speed and algorithm stability. A commonly employed method for incorporating learning rates is known as learning rate decay. This strategy initiates with a higher learning rate during initial iterations (epochs) and subsequently decreases the learning rate gradually according to a specific pattern:

$$\eta(t) = \eta_0 d^{t/T}, \tag{2.48}$$

Here, t represents the iteration count, T represents the number of steps for decay, and d denotes the decay rate. This decay aims to begin with a higher rate in the early iterations, which helps prevent the memorization of noisy data, and then gradually decreases the rate to a smaller value. This adjustment is made to prevent the occurrence of oscillations around a local minimum, as noted in [90]

Various versions of gradient descent, like stochastic gradient descent (SGD) and mini-batch gradient descent, utilize data subsets to calculate gradients in each step. These variations enhance efficiency, particularly when handling extensive datasets.

The ADAM optimizer is incorporated in this study as a fundamental element in the training process of the neural network model. The selection of ADAM was motivated by its established track record of providing efficient and proficient optimization solutions. ADAM, known as Adaptive Moment Estimation, is remarkably esteemed for its ability to fine-tune learning rates on individual parameters adaptively. In each iteration of the Adam algorithm, the process begins by computing the gradient, denoted as g_t, based on the gradients from the preceding iteration ($(t-1)$).

$$g_t = \nabla_\theta J(\theta^{(t-1)}). \tag{2.49}$$

Next, the biased first (m_t) and second raw moment (v_t) estimates are updated:

$$m_t = \beta_1 \cdot m_{t-1} + (1-\beta_1) g_t \tag{2.50a}$$

$$v_t = \beta_2 \cdot v_{t-1} + (1-\beta_2) g_t^2. \tag{2.50b}$$

Then, the biases are corrected for the first and second raw moment estimates:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{2.51a}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}. \tag{2.51b}$$

Lastly, the parameters are updated:

$$\theta_t = \theta_{t-1} - \eta \frac{\hat{m}_t}{\epsilon + \sqrt{\hat{v}_t}}. \tag{2.52}$$

β_1 and β_2 serve as hyperparameters dictating the exponential decay rate for the moving averages of the gradient (m_t) and squared gradient (v_t), respectively. The parameter ϵ is chosen to be a small number to avoid division by zero.

2.3.2 ANN Training

In our approach to the closure problem, the main idea is to enhance the parameters obtained from model reduction. Therefore, we initially implement a projection onto the full-order data instead of formulating our learning task within the complete high-dimensional space. This projection yields coefficients within a reduced subspace, subsequently serving as the foundation for building our learning framework.

The closure model employed in this thesis is an extension of the Galerkin projection-based ROM (GROM). In this model, the output of GROM at each time step is used as an initial prediction for the time coefficient rather than treating it as the final approximation. To elaborate, a correction term will be introduced to guide the time coefficients $\{\hat{\mathbf{a}}_k\}_{k=1}^r$ obtained directly from the Galerkin projection to form a more accurate representation of the time coefficient.

$$\{\mathbf{a}_k\}_{k=1}^r = \{\hat{\mathbf{a}}_k\}_{k=1}^r + \{\mathbf{c}_k\}_{k=1}^r \tag{2.53}$$

where c is a correction (closure) term defined as

$$\mathbf{c}_k = \mathbf{a}_k - \hat{\mathbf{a}}_k \tag{2.54}$$

In the ANN training phase, we assume we can access true field data at various time points, which can be derived from experiments or numerical simulations. This allows us to acquire the true modal coefficients $\mathbf{a}_k$ associated with these data. Galerkin projection is used to calculate the time progress of each time coefficient $\hat{\mathbf{a}}_k$ based on the

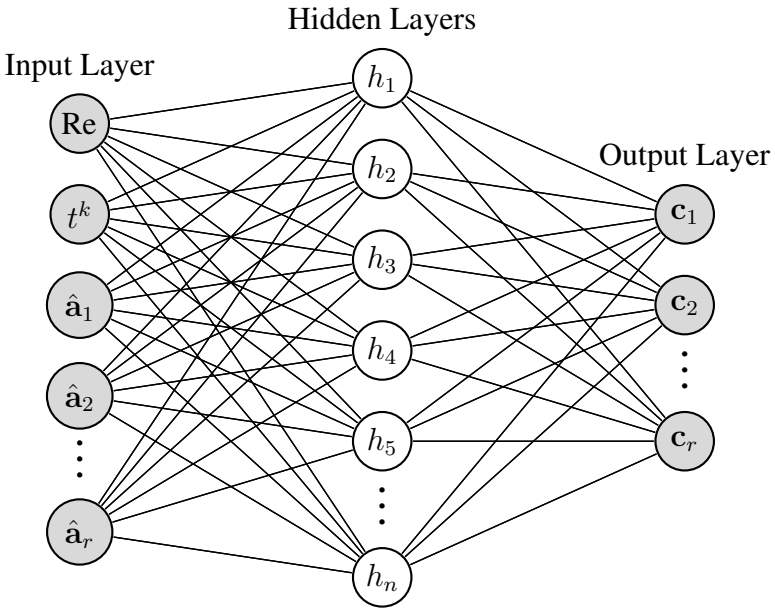


Figure 2.6: ANN architecture to map time coefficients calculated from Galerkin projection to correction term

basis functions obtained from the POD. The difference between true time coefficients and those obtained from the Galerkin projection leads to a correction term. Therefore, an ANN is trained to map $\hat{\mathbf{a}}_k$ to $\mathbf{c}_k$. Here, it is important to highlight that both input and output features are normalized within the range of their minimum and maximum values, and these normalization parameters are retained for deployment purposes.

2.3.3 JAX: High-Performance Array Computing

JAX [68] stands as a Python library tailored for efficient numerical computations, leveraging XLA library to execute code on diverse accelerators such as CPUs, GPUs, and TPUs. Its primary focus is supporting automatic differentiation, making it suitable for machine learning tasks. Notably, JAX includes a fully differentiable variant of the well-known NumPy library, referred to as JAX NumPy. Python and NumPy are extensively utilized and offer user-friendly features.

JAX introduces the Just-In-Time (JIT) compilation option for functions, substantially enhancing their performance. The static compilation, often called "ahead-of-time compilation," seeks to transform a high-level programming language program into a

more basic representation in object code or assembly language. This conversion occurs before the actual execution of the program. During JIT compilation, an abstract version of the function is cached, designed to operate for any conceivable argument values. The function is not compiled for specific input values but to accommodate the entire spectrum of potential input values, with only the array's shape and type being fixed.

For example, we define a matrix multiplication function intended for JIT compilation. During the initial call, compilation occurs, and if we provide two floating-point matrices with dimensions $(200, 100)$ and $(100, 250)$, this specific matrix shape and floating-point data type information are stored in the cache. This results in the availability of a JIT-compiled version of the function, which can be efficiently reused for diverse matrices sharing the same $(200, 100)$ and $(100, 250)$ shape.

Just-In-Time (JIT) compilation in JAX entails particular prerequisites and limitations that must be met for the smooth functioning of JIT mechanics. Fulfilling these conditions ensures optimal performance of JIT-compiled functions during execution. Firstly, the function intended for JIT compilation should be a pure function, meaning that it yields identical outputs for identical input arguments and does not produce any side effects. Secondly, control flow statements within the function must not rely on the values of input arguments. When the JIT compiler encounters a line of code containing an `if - statement`, it tries to assess this expression using the abstract value of the input variable `x`. However, due to the absence of a definite or concrete value, the evaluation process encounters a setback, leading to the termination of the tracing process at that point.

JAX is compatible with the Flax [91] and Optax [92] libraries. Flax is a versatile neural network library tailored for JAX, prioritizing high performance and adaptability. `flax.linen.Module` is the base class for all neural network modules. Layers and models are subclasses of this class. Arbitrary forward pass methods can be defined within the `flax.linen.Module` subclass. Although no methods are treated as exceptional cases, the `call` method is commonly favored, as it permits the utilization of module instances in a manner similar to functions. The `init` operation initializes the module by applying variables and subsequently provides the modified variables

as its output, while the `apply` method is responsible for performing the network's feedforward operation.

Optax, on the other hand, is a JAX library focused on processing gradients and optimization. Its purpose is to provide optimization methods, such as the Adam optimizer, that can be flexibly combined to create customized approaches for optimizing parametric models.

CHAPTER 3

APPLICATION TO BURGERS EQUATION

3.1 Governing Equations

This section employs the viscous Burgers equation as the first benchmark to evaluate the proposed model reduction approach.

The Burgers equation can be expressed as

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \frac{1}{Re}\frac{\partial^2 u}{\partial x^2}, \quad x \in [0,1], \quad t \in [0,1] \tag{3.1}$$

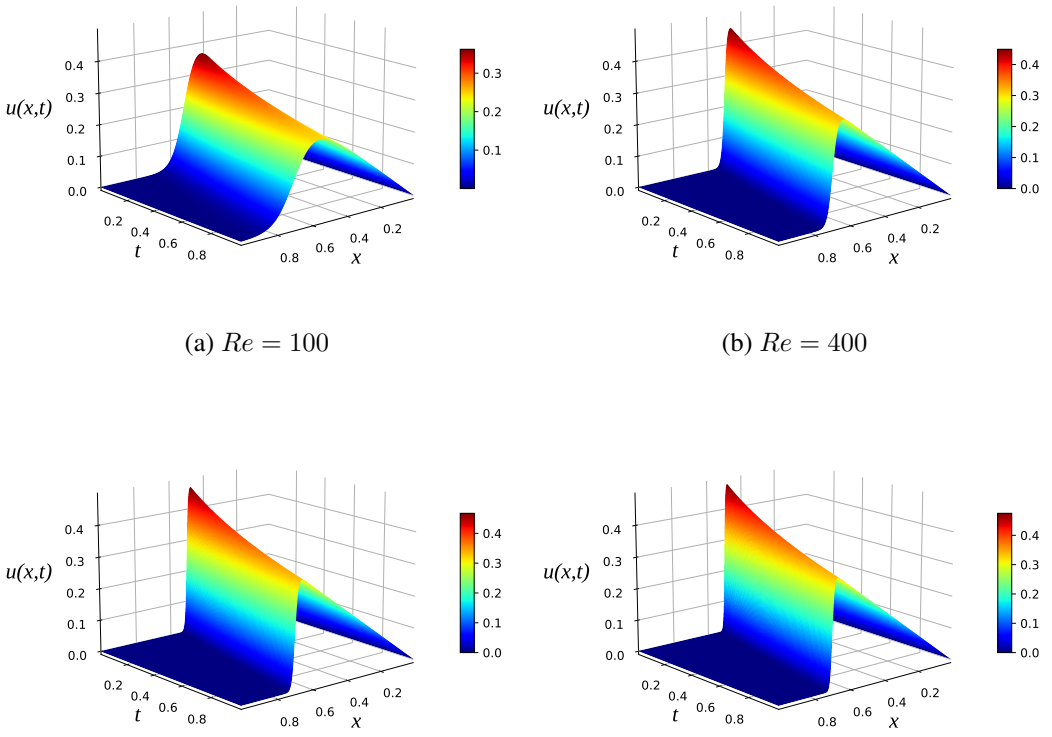
where Re is a non-dimensional Reynolds number and x refers to the spatial coordinates, t is the time.

3.2 Full Order Model

The PDE represented by Equation 3.1 can be exactly solved, leading to an analytical expression for the time-dependent behavior of the field variable $u(x,t)$ [93]. This solution is expressed as follows:

$$u(x,t) = \frac{\frac{x}{t+1}}{1 + \sqrt{\frac{t+1}{t_0}}\exp\left(Re\frac{x^2}{4t+4}\right)}, \tag{3.2}$$

where $t_0 = \exp\left(\frac{Re}{8}\right)$. For our upcoming model order reduction analysis, snapshot data is produced using this exact solution in Equation 3.2, employing $N_x = 1024$ spatial collocation points for each snapshot. The snapshots in our database were captured at 10 equidistant Reynolds number points, specifically $Re = [100, 200, \ldots, 1000]$. The spatial and temporal characteristics of the four illustrative solutions are illustrated in Figure 3.1 across different Reynolds number values.



(a) $Re = 100$ (b) $Re = 400$

(c) $Re = 700$ (d) $Re = 1000$

Figure 3.1: Spatial-temporal solution of the Burgers equation across different Reynolds numbers in our dataset

3.3 Proper Orthogonal Decomposition

Any arbitrary scalar field variable u can be used to create a representative POD basis at different points in time. These snapshots are typically obtained using a standard FOM approach to solve the governing equations we are trying to model. FOM can be considered the exact solution in this case, given in Equation 3.2 to the Burgers equation. The index k denotes a specific point in time. We use M snapshots for the field variable, i.e. $u(x, t^k)$ for $k = 1, 2, \ldots, M$ in the POD approach. The snapshot matrix

is given below. The snapshot matrix comprises the data collected across various Re.

$$\mathbf{S} = \begin{pmatrix} u_{11} & \dots & u_{1M} \\ u_{21} & \dots & u_{2M} \\ \vdots & & \vdots \\ u_{N_x 1} & \dots & u_{N_x M} \end{pmatrix} \tag{3.3}$$

The ensemble mean of the snapshots is subtracted from each snapshot to obtain the instantaneous fluctuating velocity components:

$$\mathbf{S} = \begin{pmatrix} u'\left(x_1, t^1\right) & \dots & u'\left(x_1, t^M\right) \\ u'\left(x_2, t^1\right) & \dots & u'\left(x_2, t^M\right) \\ \vdots & & \vdots \\ u'\left(x_{N_x}, t^1\right) & \dots & u'\left(x_{Nx}, t^M\right) \end{pmatrix} \tag{3.4}$$

Then, the discrete representation of the correlation between instantaneous fluctuations is formed in a matrix denoted as $\mathcal{C}$.

$$\mathcal{C}_{ij} := \left(u'\left(x, t^i\right), u'\left(x, t^j\right)\right)_\Omega \tag{3.5}$$

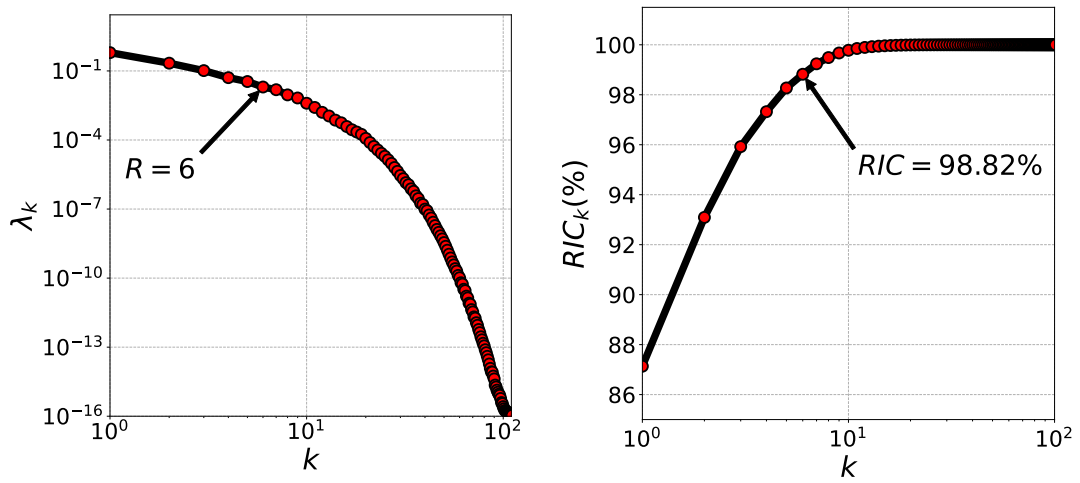
Here, Ω represents the complete spatial domain, and the indices i and j refer to the i^{th} and j^{th} snapshots. The time correlation data matrix $\mathcal{C}$ is a symmetric square matrix of size $M \times M$. The eigenvalue problem can be formulated as follows:

$$\mathcal{C}W = \Lambda W \tag{3.6}$$

A collection of eigenvectors, along with associated eigenvalues, are found in the process of eigendecomposition applied to the correlation matrix.

To find the POD modes required to reconstruct the flow field, fluctuating velocity components must be projected into the eigenvectors (by taking the dot product of the fluctuating velocity with eigenvectors).

$$\varphi = \begin{pmatrix} \varphi_1\left(x_1\right) & \dots & \varphi_r\left(x_1\right) \\ \varphi_1\left(x_2\right) & \dots & \varphi_r\left(x_2\right) \\ \vdots & & \vdots \\ \varphi_1\left(x_{N_x}\right) & \dots & \varphi_r\left(x_{Nx}\right) \end{pmatrix} \tag{3.7}$$



(a) Eigenvalues

(b) RIC Indices

Figure 3.2: Eigenvalues originating from the temporal correlation matrix (a) are paired with their corresponding RIC indices (b) for the space-time solution of the viscous Burgers equation. We retain only the 6 most dominant POD modes in our ROM.

which leads to a reduced order space spanned by POD modes.

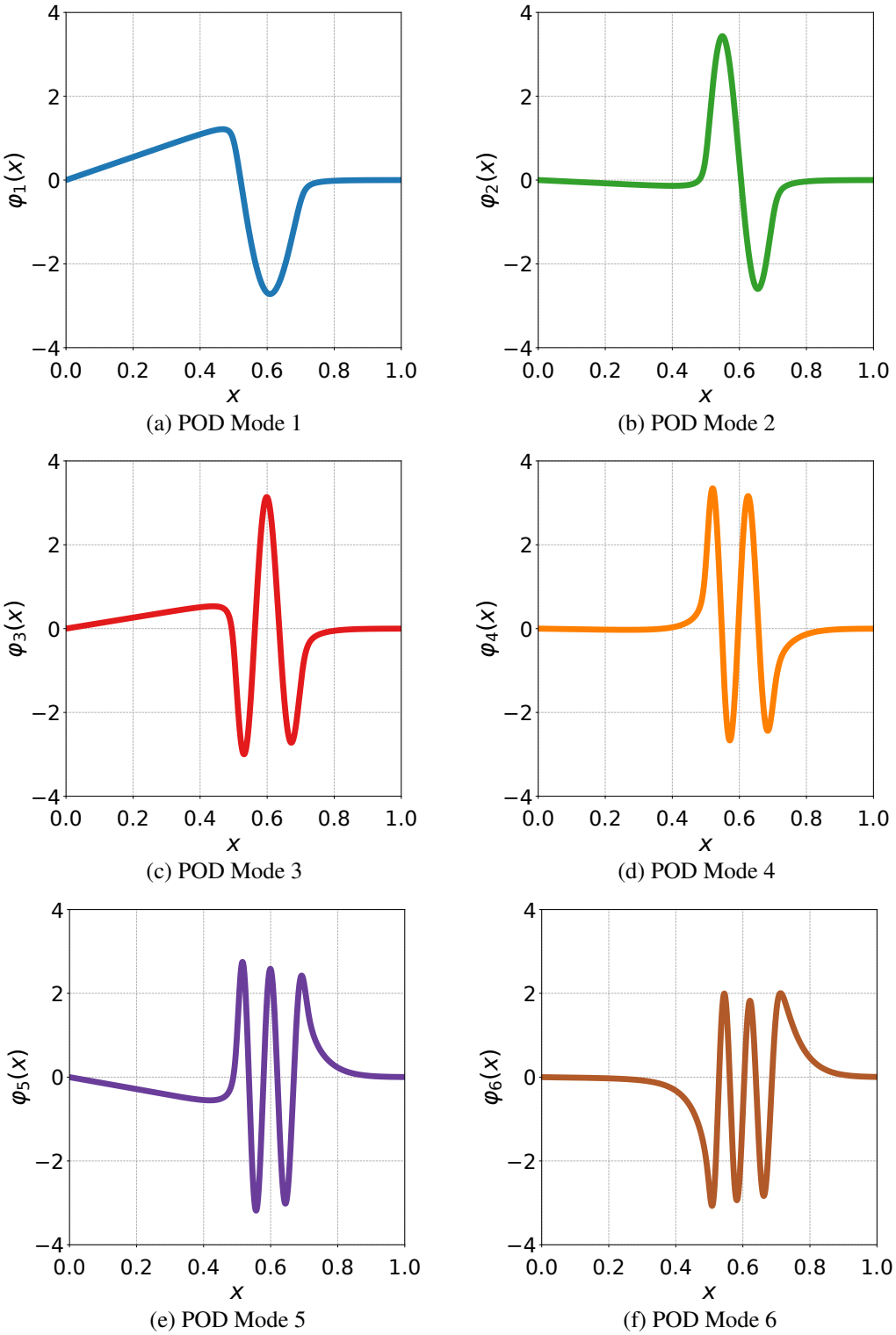
$$\boldsymbol{X}^r := \operatorname{span}\{\varphi_1, \ldots, \varphi_r\}, \tag{3.8}$$

Examining the characteristics and distribution of eigenvalues concerning their modal index is a fundamental step in the analysis of POD since it guides dimensionality reduction and aids in interpreting the physical significance of the modes.

For that purpose, the relative information content (RIC) is calculated using Equation 2.21 and it is shown in Figure 3.2 along with the eigenvalues of the time correlation matrix $\mathcal{C}$.

It becomes apparent that a limited set of modes suffices to illustrate the spatial and temporal characteristics of the system accurately. Hence, our analysis employs the leading $r = 6$ modes, representing the highest energy content and encapsulating roughly 98.82% of the overall energy. The associated POD basis functions employed for our investigation are illustrated in Figure 3.3.

Subsequently, the time coefficients are obtained through the inner product of two



(a) POD Mode 1

(b) POD Mode 2

(c) POD Mode 3

(d) POD Mode 4

(e) POD Mode 5

(f) POD Mode 6

Figure 3.3: Illustration of the most energetic POD basis functions for viscous Burgers Equation

vectors: one encompassing the fluctuating velocity components at any Re and the other containing the POD modes. We will call these true time coefficients since they result directly from the projection of the FOM.

$$\mathbf{a} = \begin{pmatrix} a_1(t_1) & \dots & a_r(t_1) \\ a_1(t_2) & \dots & a_r(t_2) \\ \vdots & & \vdots \\ a_1(t_M) & \dots & a_r(t_M) \end{pmatrix} \tag{3.9}$$

Consequently, the field variables and their progression over time can be estimated using these POD basis functions alongside their associated time coefficients. Notably, the value of r is significantly smaller than M. Introducing the mean velocity subtracted at the initial stage of our analysis, a reconstructed flow field can be generated.

3.4 Galerkin System

The most energetic POD basis functions can model the fundamental behavior of the underlying governing equations. Once these empirically derived POD basis functions are obtained, they can be utilized to formulate a set of nonlinear ODEs through Galerkin projection. We simplify the system by considering only the first r most significant POD basis functions to construct a ROM based on projection. These modes correspond to the r largest eigenvalues.

To create the GROM, we must first rewrite the Burgers equation 3.1 in the following form

$$\frac{\partial u}{\partial t} = L[u] + N[u;u] \tag{3.10}$$

where $L[f] = \frac{1}{Re}\frac{\partial^2 f}{\partial x^2}$ is the linear operator and $N[f;g] = -f\frac{\partial g}{\partial x}$ is the nonlinear operator. We obtain the GROM by applying this projection to our nonlinear system (i.e., multiplying 3.10 with the basis functions and integrating over the domain by replacing u with u_r as in Equation 2.18).

$$\frac{d\hat{a}_k}{dt} = \mathcal{B}_k + \sum_{i=1}^{r} \mathcal{L}_{ik}\hat{a}_i + \sum_{i=1}^{r}\sum_{j=1}^{r} \mathcal{N}_{ijk}\hat{a}_i\hat{a}_j, \quad \text{for} \quad k = 1, 2, \dots, r, \tag{3.11}$$

where

$$\begin{aligned}
\mathcal{B}_k &= (L[\bar{u}] + N[\bar{u};\bar{u}], \varphi_k) \\
\mathcal{L}_{ik} &= (L[\varphi_i] + N[\bar{u};\varphi_i] + N[\varphi_i;\bar{u}], \varphi_k), \\
\mathcal{N}_{ijk} &= (N[\varphi_i;\varphi_j], \varphi_k).
\end{aligned} \tag{3.12}$$

The GROM depicted by Equation 3.12 comprises a set of r interconnected ordinary differential equations (ODEs), which can be effectively solved numerically using standard techniques (such as the third-order Runge-Kutta method employed in this thesis). As a result of this approach, the system's degrees of freedom are significantly reduced. Equation 3.12 contains pre-computed vectors, matrices, and tensors contributing to the formulation of a dynamic system that can be efficiently solved. The spatial derivatives required for the computation of these pre-computed operators are calculated using second-order accurate central differences within the interior points of the spatial domain, along with second-order accurate one-sided (forward or backward) differences at the domain boundaries. The initial condition is established by applying the following projection to finalize the representation of the dynamic system defined by Equation3.11:

$$\hat{a}_k(t=0) = (u(x,t=0) - \bar{u}(x), \varphi_k), \tag{3.13}$$

where $u(x,t=0)$ is the physical initial condition of the problem.

However, the modal truncation process and the introduced errors and instabilities within the Galerkin projection-based ROM (GROM), as elaborated in Section 1.4, lead to prediction inaccuracies. To address this issue, a closure model should be constructed to increase the ROM's accuracy and eliminate the instabilities caused by the Galerkin projection while preserving computational efficiency.

3.5 Closure Model

ANN will be employed to predict a low-dimensional closure term to improve our ROM. Table 3.1 summarizes the adopted hyperparameters. We observed that the neural networks we created display a degree of robustness towards hyperparameters. However, it is important to note that various methods, such as grid search, can be

employed for fine-tuning to achieve the best possible selection of hyperparameters, which is out of the scope of this thesis.

Table 3.1: A list of hyperparameters utilized to train the ANN for Burgers Equation

Variables	**1D Burgers Equation**
Number of hidden layers	4
Number of neurons in each hidden layer	40
Epochs	1000
Activation functions in the hidden layers	ReLU
Loss function	MSE
Optimizer	ADAM
Learning rate	0.01

True time coefficients are obtained by projecting the FOM to the POD modes at $Re = [100, 200, \ldots, 1000]$. They form the input variables for our ANN model. The correction term is found by subtracting the time coefficients obtained through the Galerkin projection from the true time coefficients. These are then used as the output variables for the ANN model. Details for the training procedure of the ANN have been described in Section 2.3.2.

The ANN methodology is created using Flax, as explained in Section 2.3.3. This choice is due to Flax's smooth integration with JAX, facilitating efficient computations and automatic differentiation. Flax also provides essential support for distributed training across multiple GPUs and TPUs, a significant advantage for accelerating the training of expansive models.

3.6 Results

Here, we provide data on the temporal evolution of modal coefficients for our test case and evaluate the comparative effectiveness of the GROM and GROM(6) + Closure. While assessing the error in temporal evolution may not align precisely with measuring errors in conventional function norms or quantities of interest, we noticed

that this approach clarifies our presentation and offers a convenient means to compare the proposed methods against the standard GROM model. Therefore, in addition to the reconstructed fields for different frameworks, the evolution of the temporal coefficients will also be presented.

3.6.1 $Re = 650$: demonstrating ability for interpolation

It should be emphasized that the subsequent results are derived from Reynolds number values that lie outside the range covered by the snapshot dataset. In other words, following the derivation of the basis functions using snapshot data acquired for $Re = [100, 200, \ldots, 1000]$, a Galerkin projection is executed to compute the time coefficients for $Re = 650$ and the correction term is predicted using the ANN. The FOM solution is given in Figure 3.4.

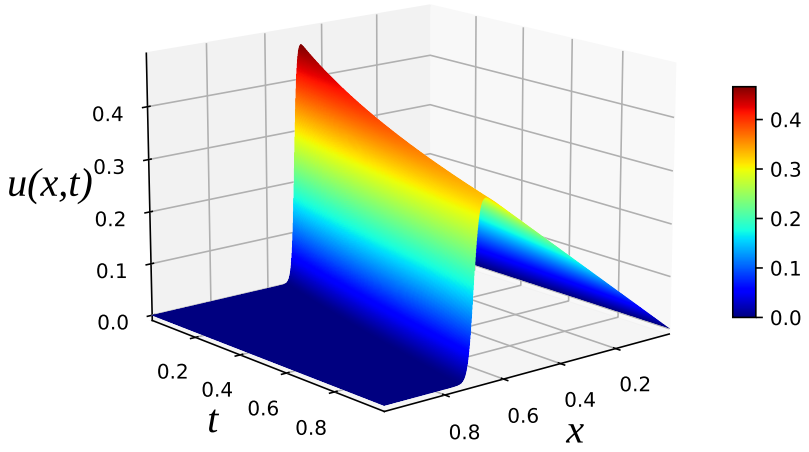
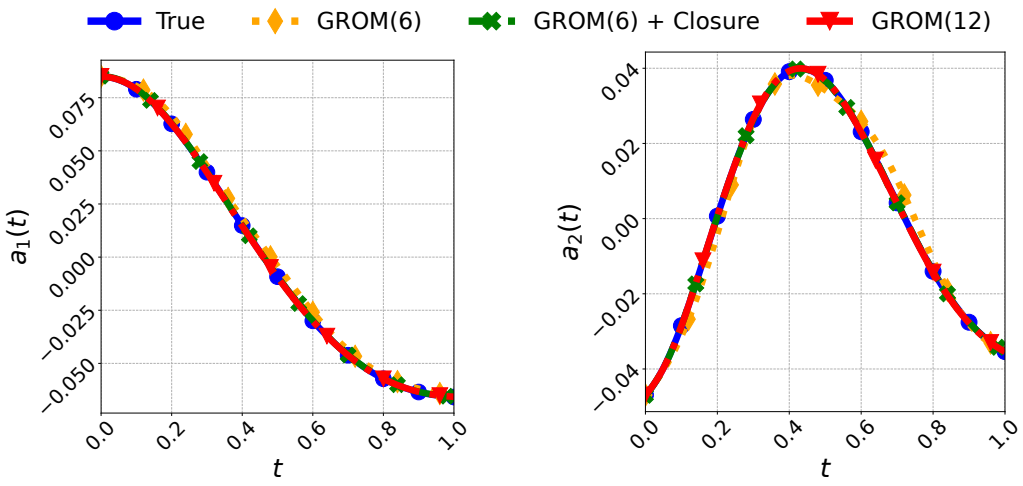


Figure 3.4: Exact solution obtained from the full order model at $Re = 650$

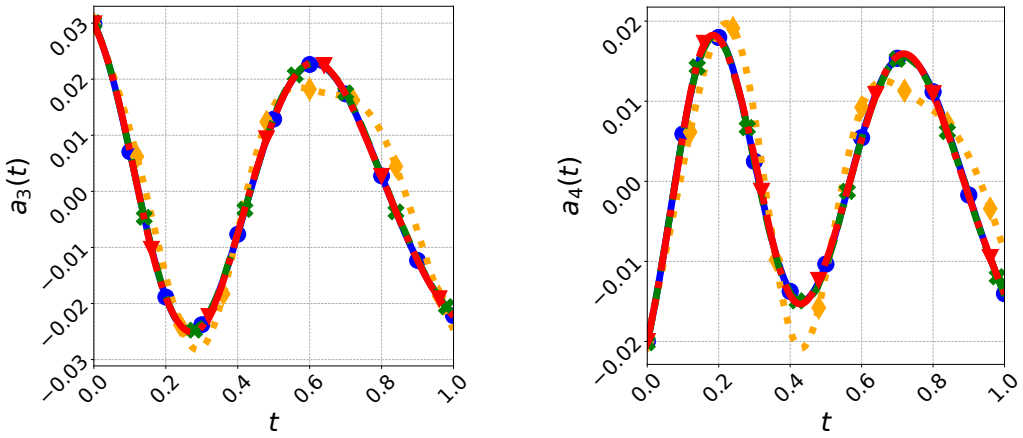
Figure 3.5 illustrates the progression of the initial 6 POD temporal coefficients across distinct frameworks. GROM(6) struggles to capture accurate dynamics due to considerable mode truncation. Conversely, GROM(12) and GROM(6) + Closure yield favorable outcomes, yet GROM(12) demands higher computational resources.

Regarding the reconstruction of the field, we show the temporal evolution in Fig-



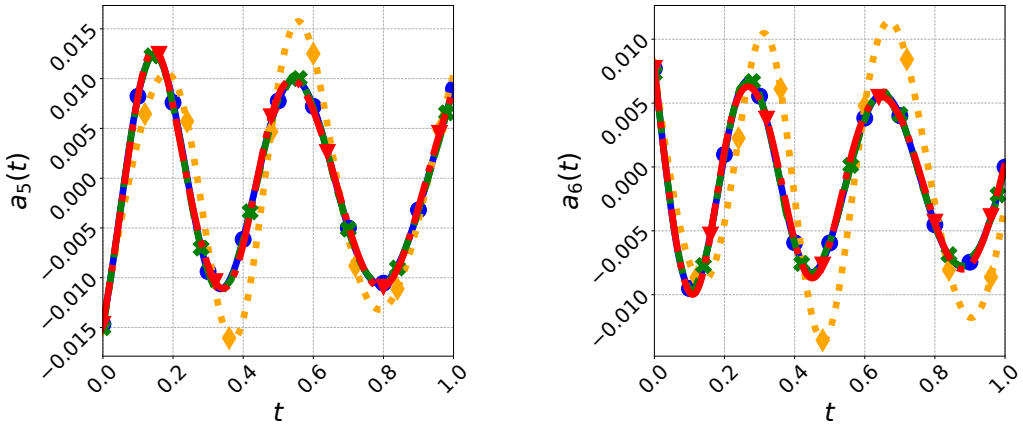
(a) Time Coefficient 1

(b) Time Coefficient 2



(c) Time Coefficient 1

(d) Time Coefficient 2



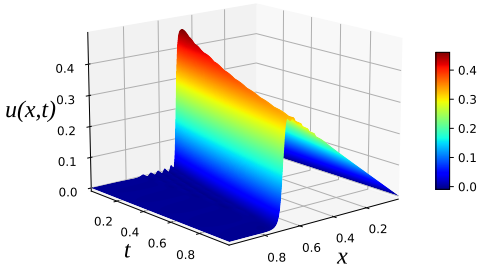
(e) Time Coefficient 1

(f) Time Coefficient 2

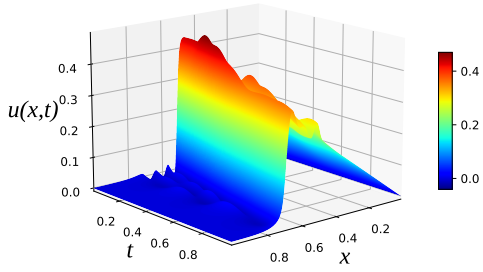
Figure 3.5: Numerical assessments of time coefficients provided by different frameworks applied to the Burgers equation for an interpolatory out-of-sample parameter $Re = 650$

ure 3.6, encompassing snapshots from the exact solution, true projection, as well as GROM(6) + Closure, GROM(6), and GROM(12). The field reconstruction performed by GROM(6) + Closure demonstrates notably accurate predictions compared to the reconstructed field by GROM(6).

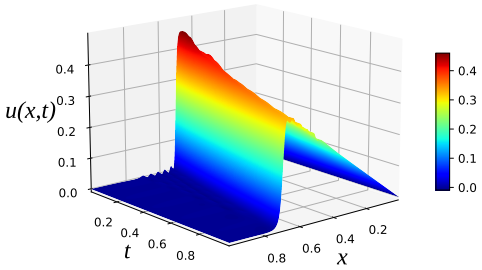
In Figure 3.7, we present a visualization depicting the magnitudes of errors in the proposed model. These errors were computed by taking the difference between the flow field generated by the GROM + Closure model and the true projection. The errors are prominently visible in regions where the flow experiences rapid changes.



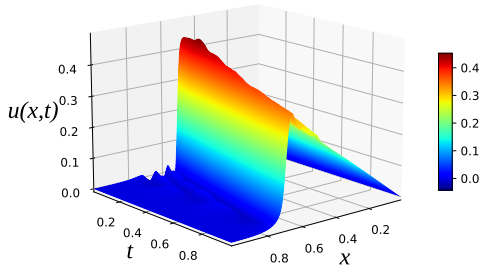
(a) True Projection



(b) Reconstruction with 6 modes



(c) Reconstruction with 12 modes



(d) Reconstruction with 6 modes + Closure

Figure 3.6: Illustration of the temporal evolution of velocity fields for Burgers problem at $Re = 650$ obtained from GROM and GROM(6) + Closure

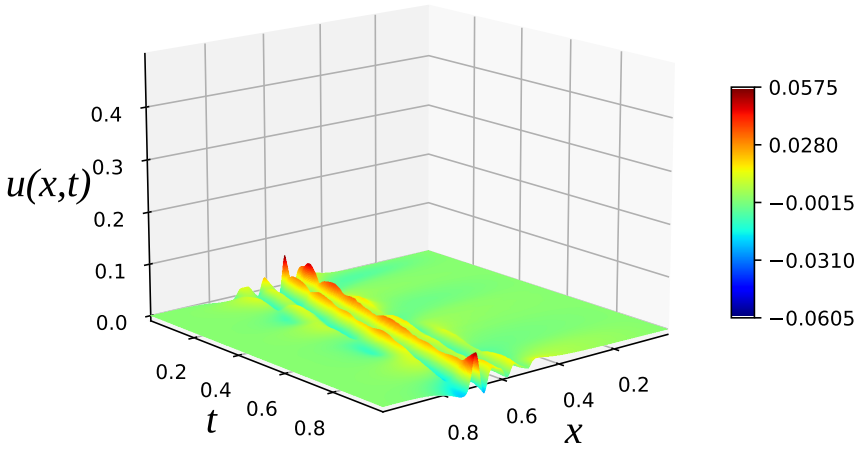


Figure 3.7: Difference between the reconstructed flow fields obtained from GROM + Closure and true projection models for $Re = 650$

3.6.2 $Re = 1250$: demonstrating ability for extrapolation

The upcoming $Re = 1250$ test extrapolation capabilities for the GROM and GROM(6) + Closure models. Figure 3.8 shows the exact solution obtained from the FOM.

Our examination of the given test cases reveals that utilizing a greater number of modes becomes necessary as the Reynolds number increases. At $Re = 650$, it was observed that increasing the number of modes led to improvements in the results. However, this enhancement came at the expense of increased computational expenses. The closure model we introduced proved to be advantageous in this context. It provided a level of accuracy comparable to that of the model utilizing many modes while maintaining relatively lower computational costs.

The discrepancy between the reconstructed fields and the time evolution of the temporal coefficients becomes more evident at $Re = 1250$. In a similar study in Figure 3.9, we first employed 6 POD modes to predict the flow field. Then, we increased the number of modes to 12. As expected, the results showed improvements as in the case of $Re = 650$. However, examining the final 2 time coefficients depicted in Figure 3.9, it becomes apparent that the suggested closure model produced outcomes that closely

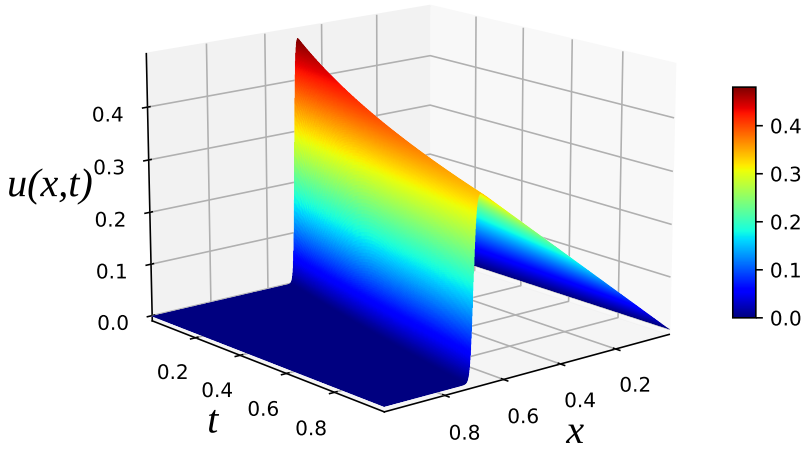
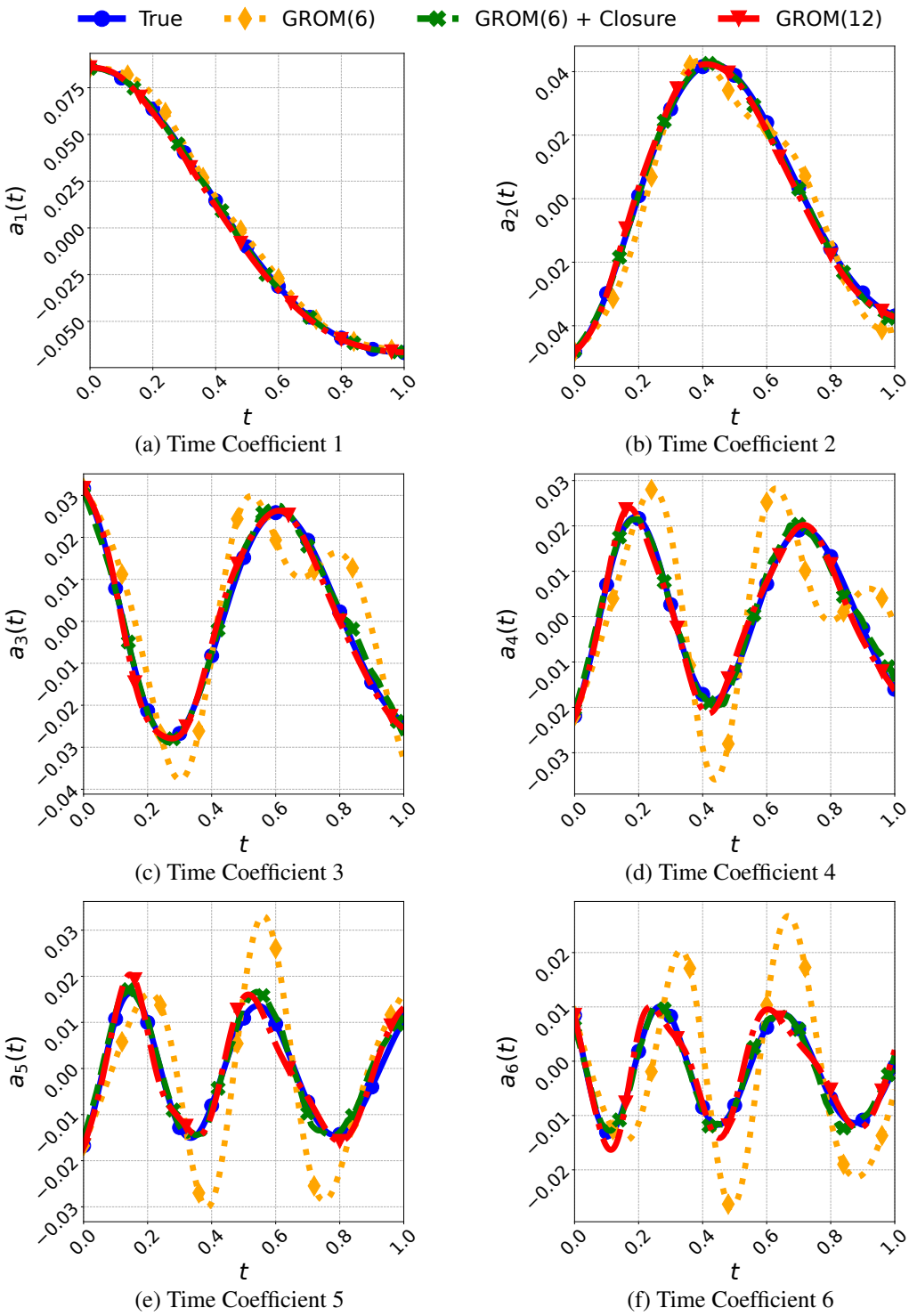


Figure 3.8: Exact solution obtained from the full order model at $Re = 1250$

resemble the actual projection. Also, the reconstructed fields in Figure 3.10 show that utilizing even the 12 most dominant POD modes might not adequately capture the underlying transient behavior. By adopting the GROM(6) + Closure method, we achieve an accuracy comparable, even higher to that of GROM(12), but with minimal computational overhead.

Stability is also achieved compared to closure models incorporating only non-intrusive methods because the physics-based GROM approach provides a base for the proposed framework, with ANN utilization required only for the correction terms.

The magnitudes of the errors shown in Figure 3.11 have noticeably increased compared to the previous case. This observation aligns with our earlier discussions, where we mentioned that the performance of the ROMs tends to decrease as the Re increases.



(a) Time Coefficient 1

(b) Time Coefficient 2

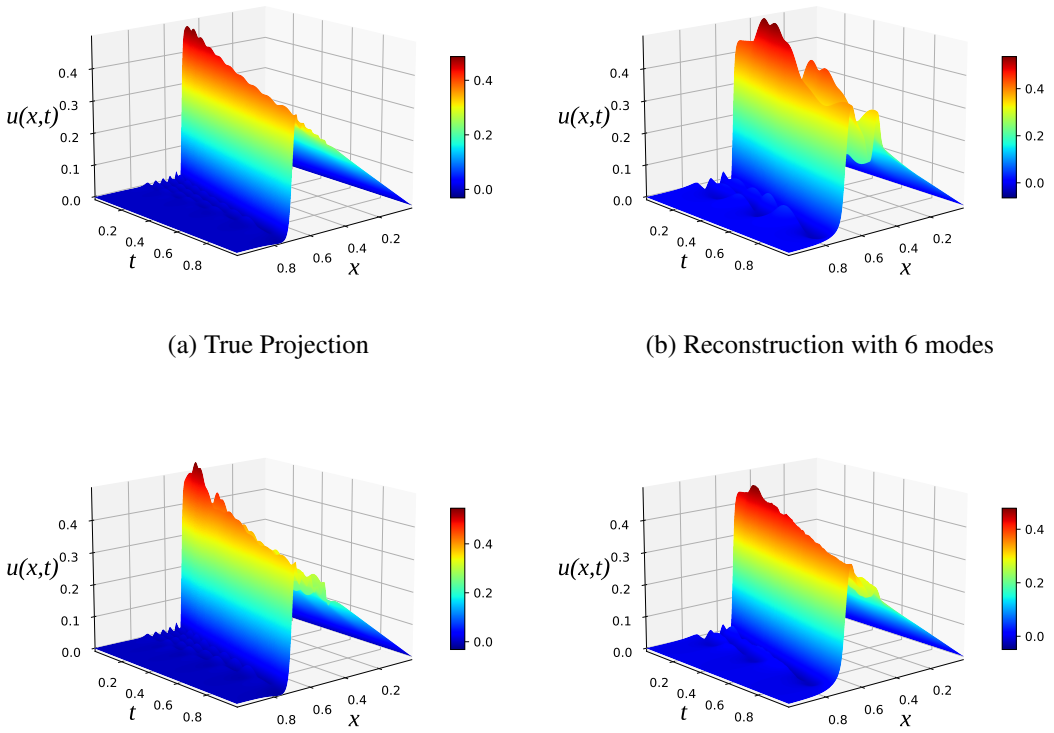
(c) Time Coefficient 3

(d) Time Coefficient 4

(e) Time Coefficient 5

(f) Time Coefficient 6

Figure 3.9: Numerical assessments of time coefficients provided by different frameworks applied to the Burgers equation for an extrapolatory out-of-sample parameter $Re = 1250$



(a) True Projection

(b) Reconstruction with 6 modes

(c) Reconstruction with 12 modes

(d) Reconstruction with 6 modes + Closure

Figure 3.10: Illustration of the reconstructed temporal evolution of velocity fields for Burgers problem at $Re = 1250$ obtained from GROM and GROM(6) + Closure

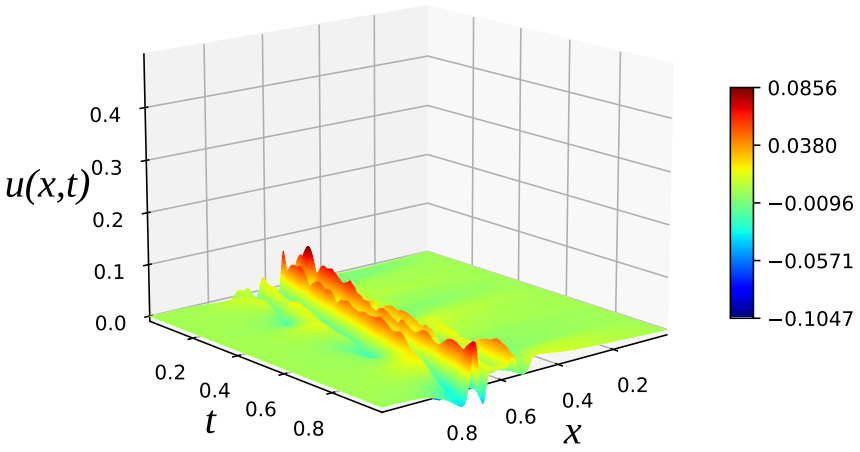


Figure 3.11: Difference between the reconstructed flow fields obtained from GROM + Closure and true projection models for $Re = 1250$

CHAPTER 4

APPLICATION TO VORTICITY-STREAMFUNCTION EQUATION

4.1 Governing Equations

In vector form, we have

$$\nabla^2\psi = -\omega \tag{4.1}$$

where ψ s the streamfunction and ω is the vorticity and ∇^2 is the 2D Laplacian operator. vorticity-streamfunction formulation of the 2D Navier–Stokes equations can be written as

$$\frac{\partial\omega}{\partial t} + (\mathbf{u}\cdot\nabla)\omega = v\nabla^2\omega \tag{4.2}$$

where the vortex stretching term can be seen to vanish. We can also write

$$\mathbf{u}\cdot\nabla = u\frac{\partial}{\partial x} + v\frac{\partial}{\partial y} = \frac{\partial\psi}{\partial y}\frac{\partial}{\partial x} - \frac{\partial\psi}{\partial x}\frac{\partial}{\partial y} \tag{4.3}$$

The vorticity equation in two dimensions then becomes

$$\frac{\partial\omega}{\partial t} + \left(\frac{\partial\psi}{\partial y}\frac{\partial\omega}{\partial x} - \frac{\partial\psi}{\partial x}\frac{\partial\omega}{\partial y}\right) = v\nabla^2\omega \tag{4.4}$$

4.2 Full Order Model

To put the 2D vorticity-function equation into practical use, we investigate a phenomenon known as the vortex merger problem. This scenario involves merging a pair of vortices that rotate in the same direction. When these vortices have parallel axes and come within a specific critical distance of each other, they merge to form a single vortex that is nearly axisymmetric. To explore this problem, we start with an initial vorticity field that consists of two vortices [94]. These vortices are distributed in a

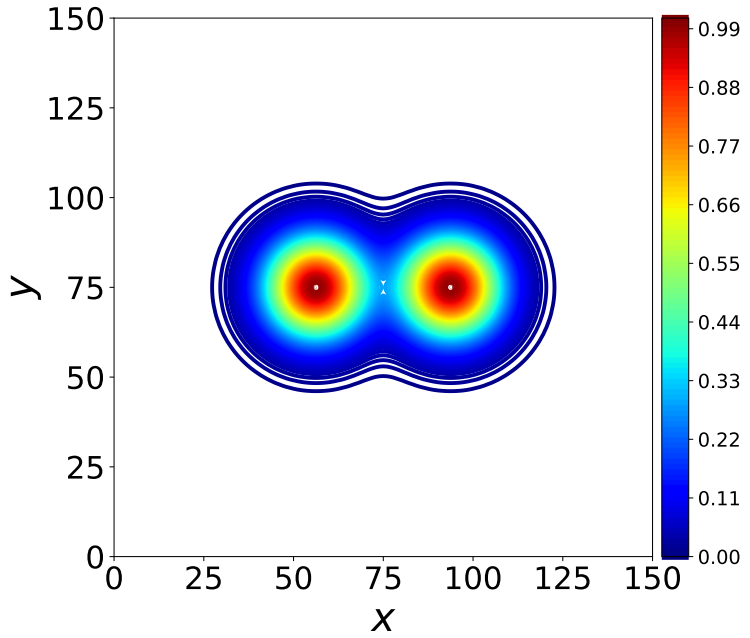


Figure 4.1: Initial field for the 2D vorticity - streamfunction equation

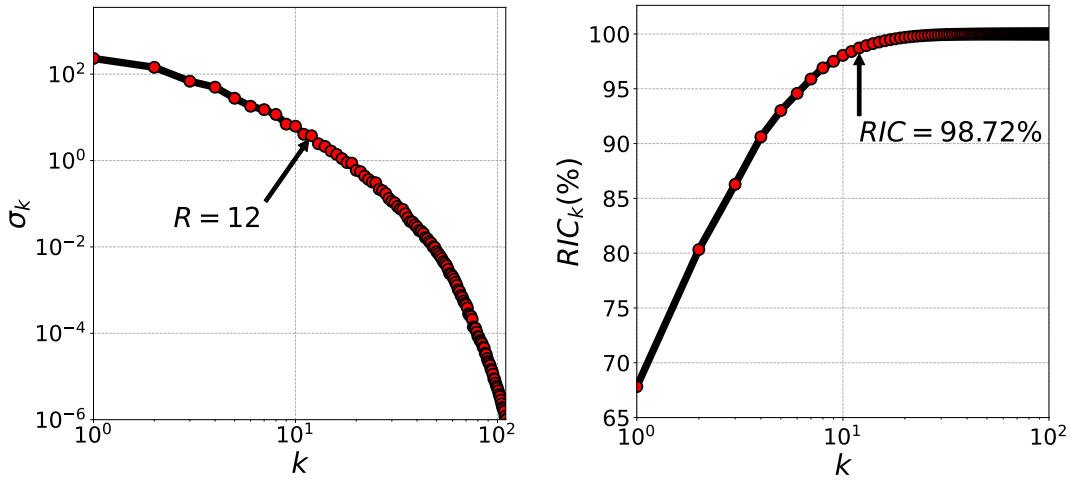
Gaussian pattern, carrying a unit circulation.

$$\begin{aligned}\omega(x,y,0) &= \exp\left(-\rho\left[(x-x_1)^2+(y-y_1)^2\right]\right)\\ &+\exp\left(-\rho\left[(x-x_2)^2+(y-y_2)^2\right]\right)\end{aligned} \tag{4.5}$$

The interaction constant, denoted by the symbol ρ, is established at a specific value of π. Initially, the centers of the vortices are positioned at coordinates $(x_1, y_1) = \left(\frac{3\pi}{4}, \pi\right)$ and $(x_2, y_2) = \left(\frac{5\pi}{4}, \pi\right)$. The initial field is shown in Figure 4.1.

The domain is discretized into 150×150 points, with periodic boundary conditions. We gather a total of 200 snapshots spanning the time interval $t \in [0, 20]$ by using the source code provided in [66], adjusting the Reynolds number across a range of values: $\mathrm{Re} \in [300, 500, 700, 900]$. The snapshot matrix consists of all the data obtained at different Reynolds numbers.

$$\mathbf{S} = \begin{bmatrix} | & | & & | \\ \omega^1 & \omega^2 & \cdots & \omega^M \\ | & | & & | \end{bmatrix}. \tag{4.6}$$



(a) Singular values

(b) RIC Indices

Figure 4.2: Singular values originating from the snapshot matrix (a) are paired with their corresponding RIC indices (b) for the space-time solution of the vorticity - streamfunction equation. We retain only the 12 most dominant POD modes in our ROM.

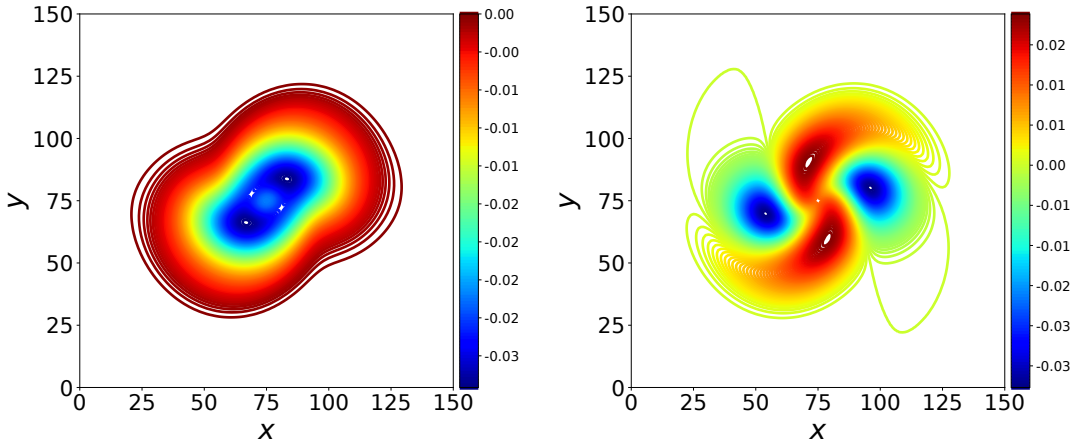
4.3 Proper Orthogonal Decomposition

Since the domain is considerably larger than that of the Burgers problem, computing times of the eigenvalue problem resulted from the correlation matrix dramatically increased. Therefore, the POD modes are calculated using the built-in function `numpy.linalg.svd` without decomposing the vorticity field into its mean and fluctuating parts. The singular values resulting from the SVD can be seen in Figure 4.2.

Our analysis focuses on the top $r = 12$ modes, encompassing the most dominant features and approximately 98.72% of the total energy. Only the first 6 vorticity modes will be presented in Figure 4.3 for illustration purposes.

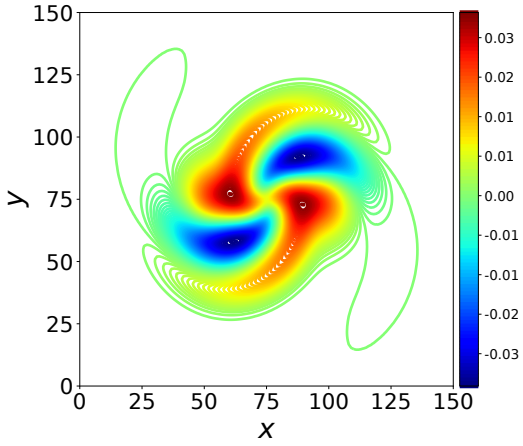
It is evident from Figure 4.3 that POD Mode 1 represents the mean vorticity field that we did not subtract at the beginning of our analysis.

The true time coefficients are obtained by projecting the snapshots taken at the prescribed Reynolds numbers to the POD modes as in the case of the Burgers equation.

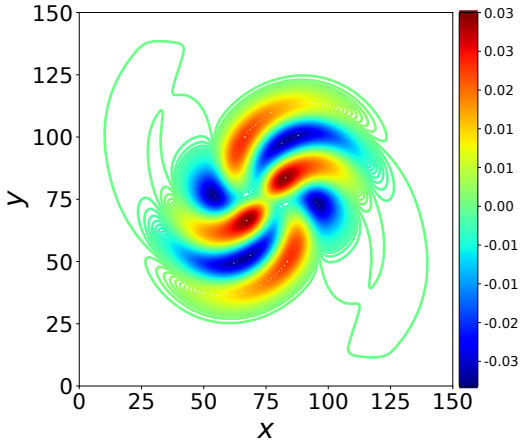


(a) POD Mode 1

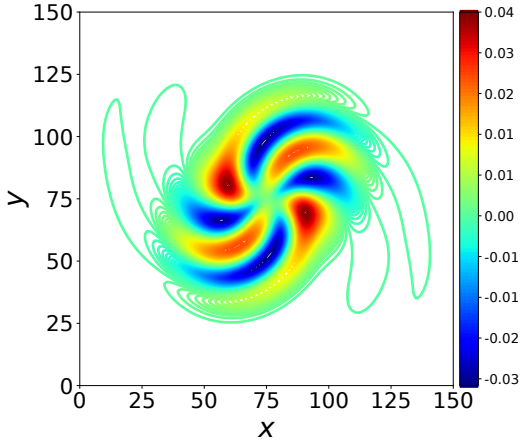
(b) POD Mode 2



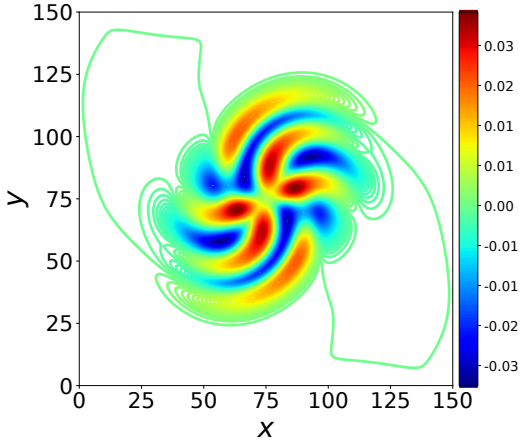
(c) POD Mode 3



(d) POD Mode 4



(e) POD Mode 5



(f) POD Mode 6

Figure 4.3: Illustration of the 6 most energetic POD basis functions for vorticity - streamfunction equation.

It is important to observe that the vorticity and streamfunction share identical time coefficients, owing to their connection through the kinematic relationship defined in Equation 4.1 [66]. Additionally, due to the linear characteristics preserved by the POD, the spatial modes for the streamfunction can be derived from the vorticity modes by solving the Poisson equations.

4.4 Galerkin System

To create the GROM, we must first rewrite the vorticity-streamfunction formulation given in Equation 4.4 in the following form:

$$\frac{\partial \omega}{\partial t} = L[\omega] - J[\omega, \psi] \tag{4.7}$$

where $L[u]$ is the linear operator and $J[\omega, \psi]$ is the non-linear operator defined as:

$$\begin{aligned} J[f;g] &= \frac{\partial f}{\partial x}\frac{\partial g}{\partial y} - \frac{\partial f}{\partial y}\frac{\partial g}{\partial x} \\ L[f] &= \frac{1}{Re}\left(\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}\right) \end{aligned} \tag{4.8}$$

multiplying Equation 4.7 with the basis functions and integrating over the domain and by replacing ω with POD modes:

$$\frac{d\hat{a}_k}{dt} = \sum_{i=1}^{r} \mathcal{L}_{ik}\hat{a}_i + \sum_{i=1}^{r}\sum_{j=1}^{r} \mathcal{N}_{ijk}\hat{a}_i\hat{a}_j, \quad \text{for} \quad k = 1, 2, \ldots, r, \tag{4.9}$$

where

$$\begin{aligned} \mathcal{L}_{ik} &= \left(L\left[\varphi_i^{\omega}\right], \varphi_k^{\omega}\right), \\ \mathcal{N}_{ijk} &= \left(-J\left[\varphi_i^{\omega}; \varphi_j^{\omega}\right], \varphi_k^{\omega}\right). \end{aligned} \tag{4.10}$$

and φ^{ω} represents the POD modes obtained for the vorticity field. The operators will be calculated in a manner similar to how the Burgers equation is handled, employing second-order central differencing methods for the derivatives. The reconstruction process will be applied to the vorticity field exclusively, meaning that the Galerkin projection will only concern the vorticity modes. For the time integration, the third-order Runge-Kutta method will be employed.

4.5 Closure Model

A similar ANN algorithm using Flax for the closure model is developed here utilizing JAX properties. The hyperparameters utilized are outlined in Table 4.1.

Table 4.1: A list of hyperparameters utilized to train the ANN for 2D vorticity-streamfunction equation

Variables	**2D Vorticity-Streamfunction Equation**
Number of hidden layers	6
Number of neurons in each hidden layer	120
Epochs	3000
Activation functions in the hidden layers	ReLU
Loss function	MSE
Optimizer	ADAM
Learning rate	0.01

True time coefficients are obtained by projecting the FOM onto POD modes at different Reynolds numbers, serving as inputs for the ANN, while the correction term, obtained by subtracting Galerkin projection coefficients from the true coefficients, serves as outputs for the ANN; additional training specifics can be found in Section 2.3.2.

Due to the significantly nonlinear nature of the temporal evolution of the time coefficients derived from the vorticity-streamfunctıon equation, more hidden layers and neurons are used.

4.6 Results

To show the characteristics and advantages of the proposed framework, we present results for the two test cases using Re that are beyond the range of those used during model training, including both interpolation and extrapolation problems.

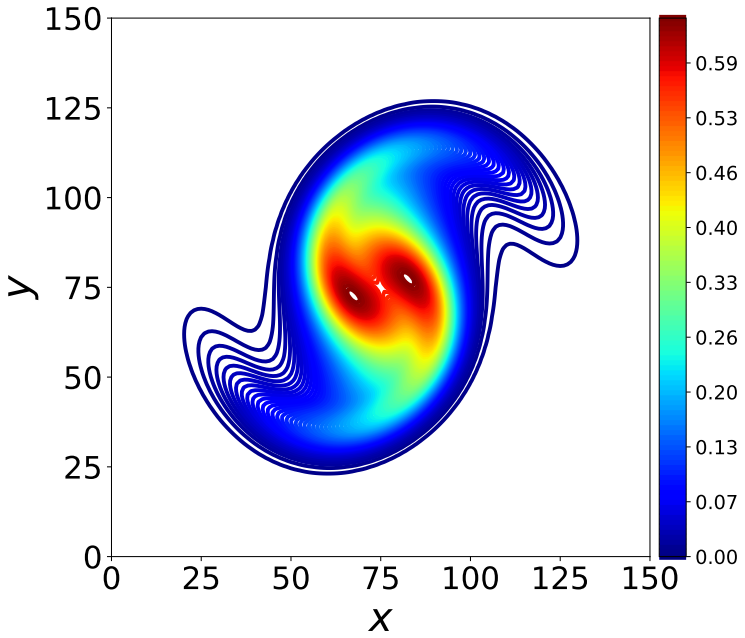
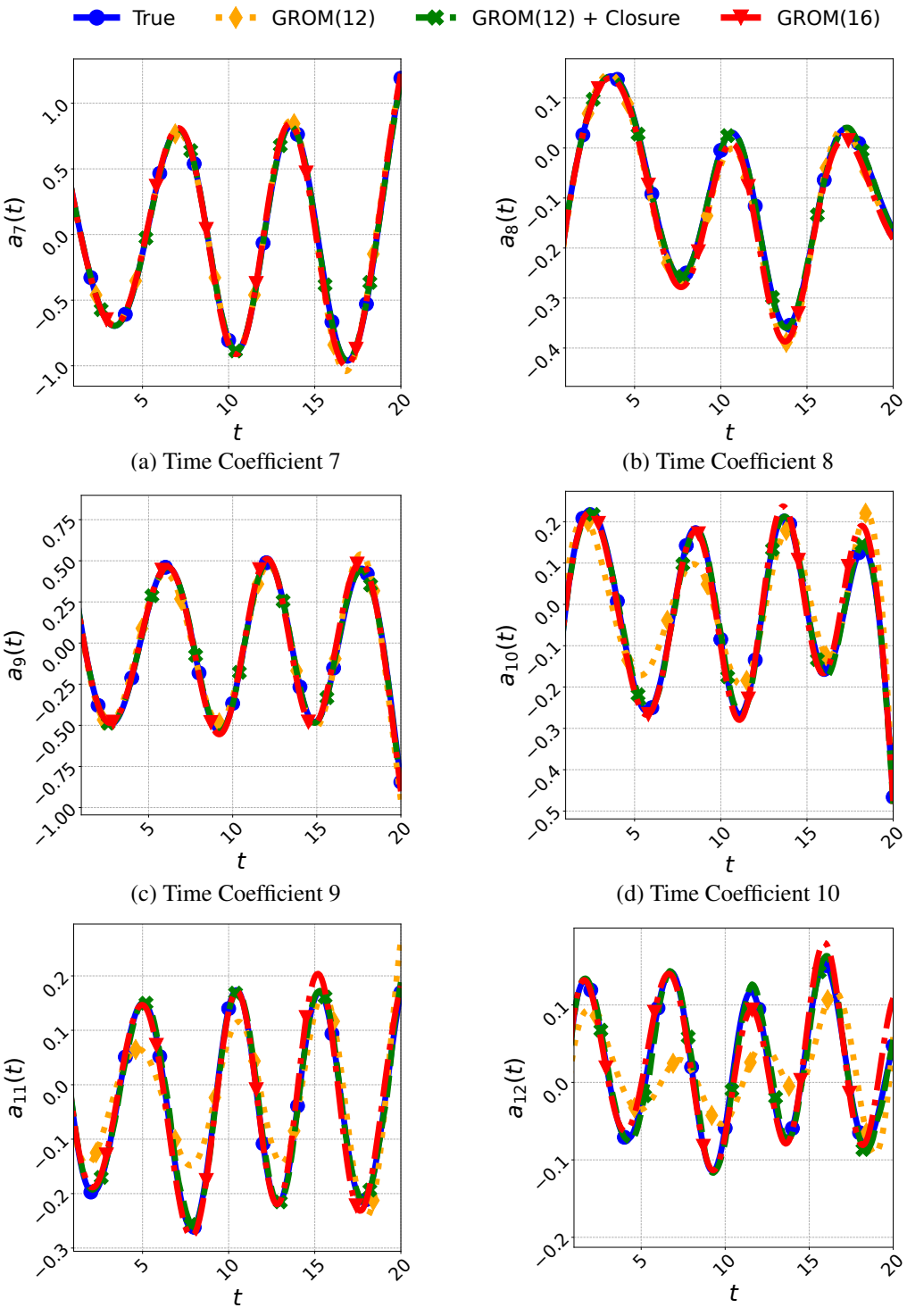


Figure 4.4: Vorticity field obtained from the full order model at $t = 20$ for $Re = 600$.

4.6.1 $Re = 600$: demonstrating ability for interpolation

Galerkin projection has been employed to calculate the temporal coefficients for $Re = 600$. Then, ANN is utilized to estimate the correction term. The FOM solution taken at the last time step for this problem can be observed in the provided Figure 4.4.

While the time coefficients illustrated in Figure 4.5 do not exhibit significant variations, particularly in the lower modes, the differences become noticeable as we move towards higher modes. The last 6 time coefficients are presented since the most significant differences are observed around these modes. These higher modes correspond to the finer scales of motion within the flow. In this context, the results obtained from the closure model exhibit remarkable similarity to the true projection, showing its efficiency in approximating the finer details of the flow. As seen from Figure 4.7, since 12 modes can capture the most dominant structures present in the flow field for this Re, there are no major differences between the reconstructed flow fields at the last time step. However, minor differences can be observed through the time evolution of these fields because of the differences in time coefficients. A higher Re case should be tested to further investigate the framework's effectiveness.



(a) Time Coefficient 7

(b) Time Coefficient 8

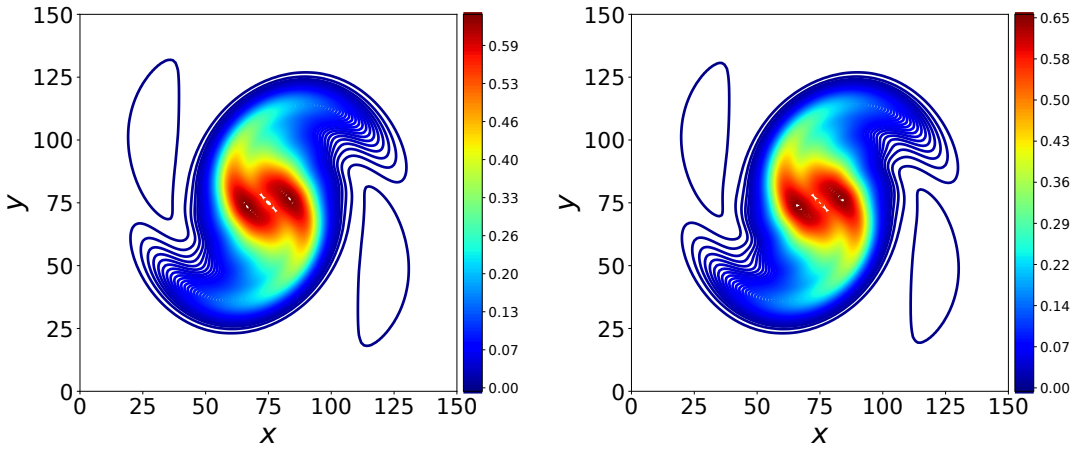
(c) Time Coefficient 9

(d) Time Coefficient 10

(e) Time Coefficient 11

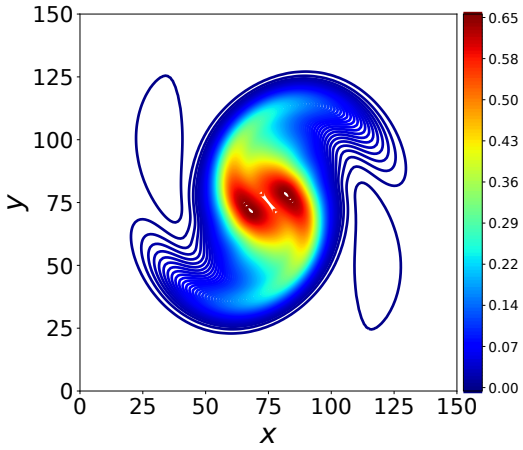
(f) Time Coefficient 12

Figure 4.5: Numerical assessments of time coefficients provided by different frameworks applied to the vorticity-streamfunction equation for an interpolatory out-of-sample parameter $Re = 600$

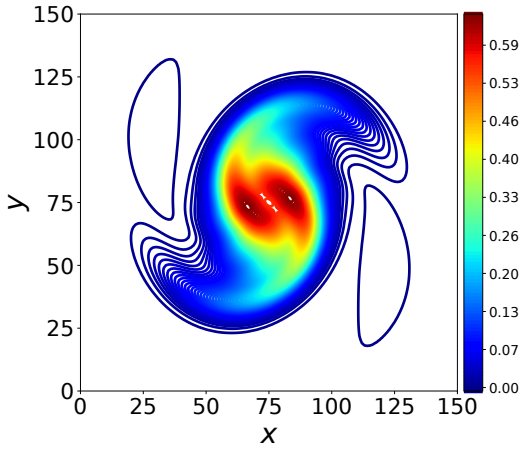


(a) True Projection

(b) Reconstruction with 12 modes



(c) Reconstruction with 16 modes



(d) Reconstruction with 12 modes + Closure

Figure 4.6: Illustration of the vorticity fields at $t = 20$ for vorticity-streamfunction equation at $Re = 600$ obtained from GROM and GROM(12) + Closure

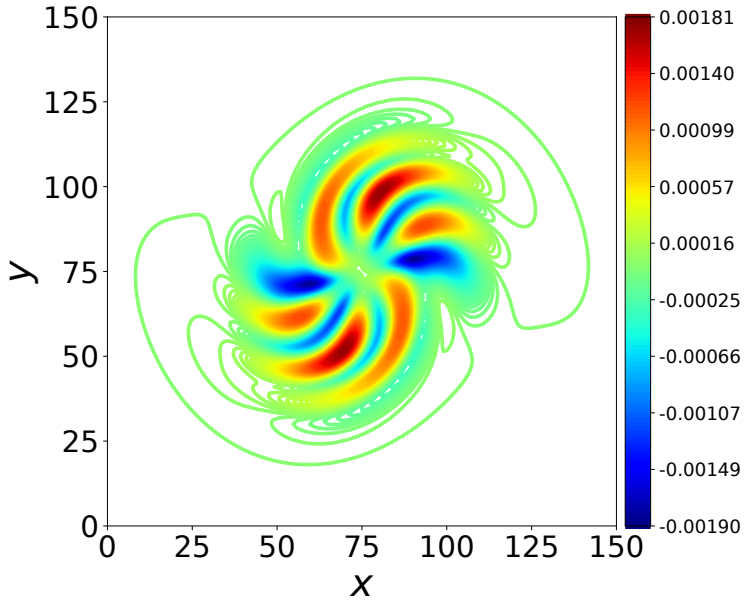


Figure 4.7: Difference between the reconstructed flow fields obtained from GROM + Closure and true projection models for $Re = 600$.

4.6.2 $Re = 1100$ demonstrating ability for extrapolation

To test the ability of the framework to predict the flow field at higher Re numbers, Galerkin projection has been employed to calculate the temporal coefficients for $Re = 1100$. Then, ANN is utilized to estimate the correction term. The FOM solution taken at the last time step for this problem can be observed in the provided Figure 4.8.

The last 6 time coefficients are provided in Figure 4.9. The major differences are observed at the last 3 time coefficients, as in the case of the lower Re. The reconstructed flow fields depicted in Figure 4.10 show that the closure model can predict the flow field with an accuracy close to the true projection of the FOM. Although all the models can correctly predict the orientation of the merging vortices, the instabilities in the models that only use Galerkin projection are more visible. The result obtained from the closure model shows more similarities with the true projection, indicating that the accuracy of the true projection can be obtained while using less number of modes.

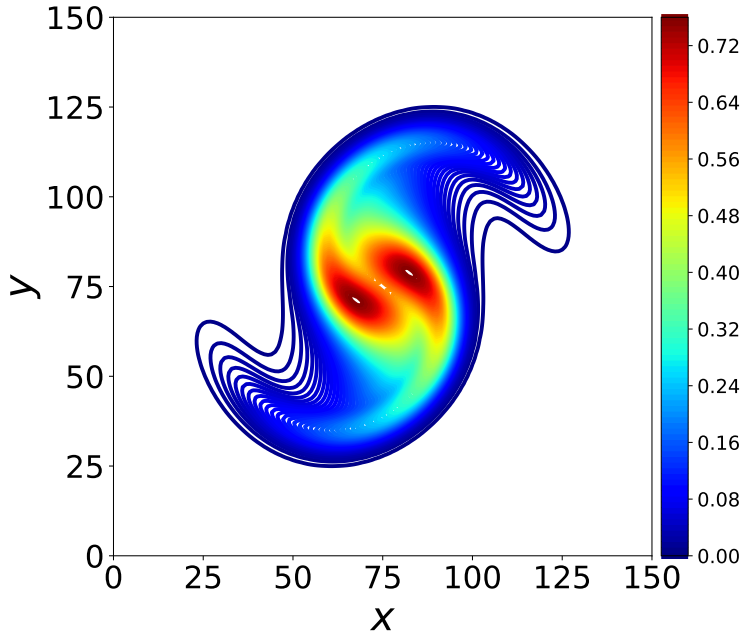
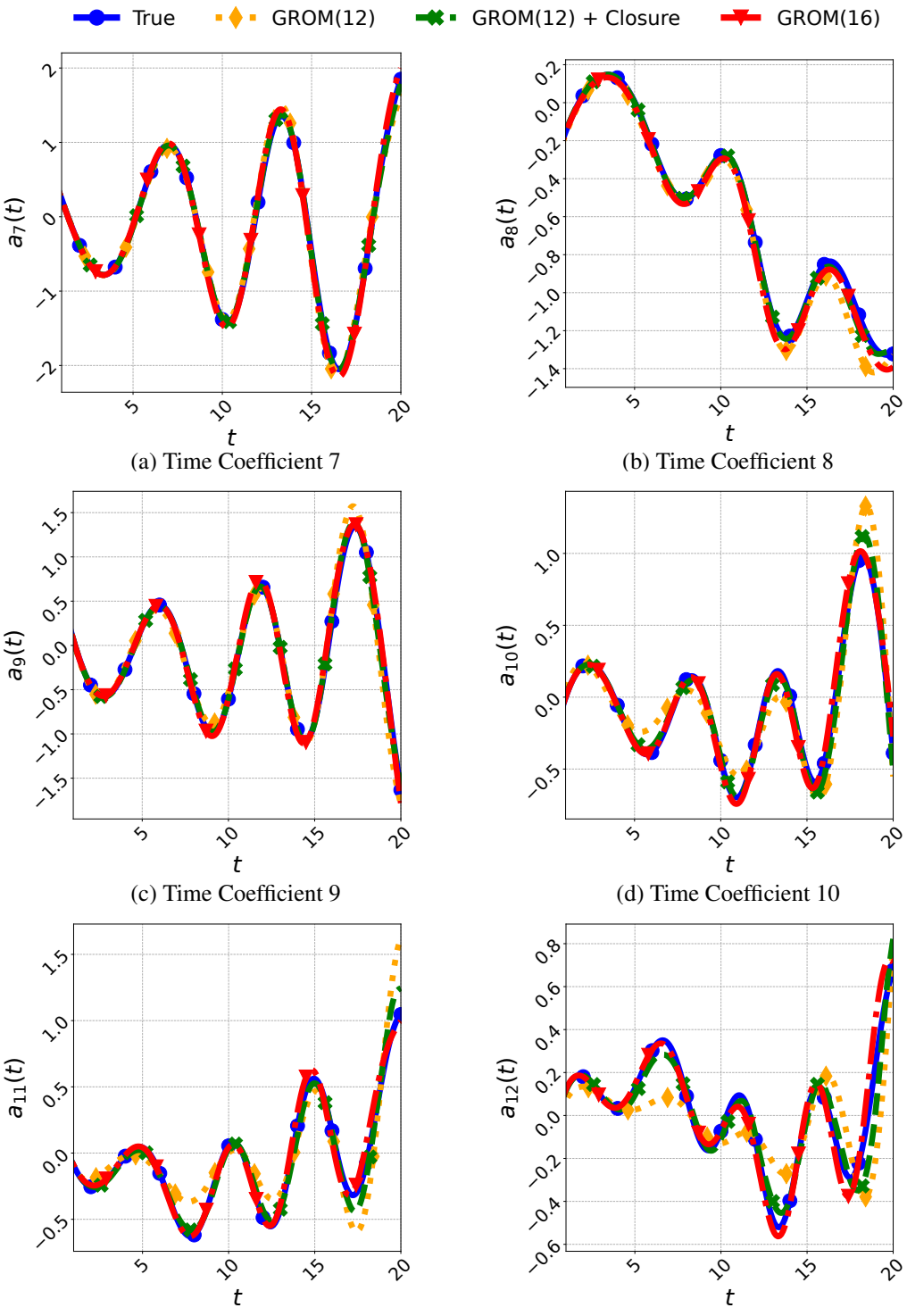


Figure 4.8: Vorticity field obtained from the full order model at $t = 20$ for $Re = 1100$.



(a) Time Coefficient 7

(b) Time Coefficient 8

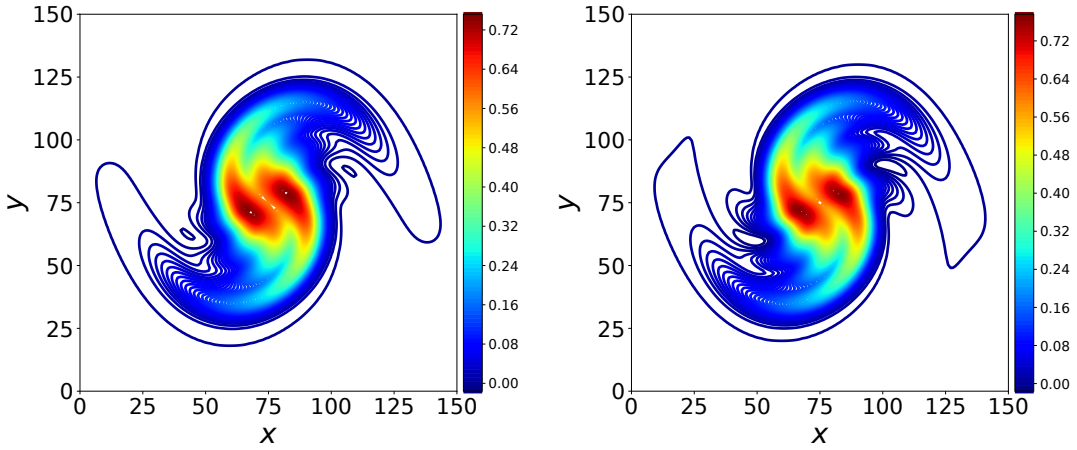
(c) Time Coefficient 9

(d) Time Coefficient 10

(e) Time Coefficient 11

(f) Time Coefficient 12

Figure 4.9: Numerical assessments of time coefficients provided by different frameworks applied to the vorticity-streamfunction equation for an interpolatory out-of-sample parameter $Re = 1100$



(a) True Projection

(b) Reconstruction with 12 modes

(c) Reconstruction with 16 modes

(d) Reconstruction with 12 modes + Closure

Figure 4.10: Illustration of reconstructed the vorticity fields at $t = 20$ for vorticity-streamfunction equation at $Re = 1100$ obtained from GROM and GROM(12) + Closure

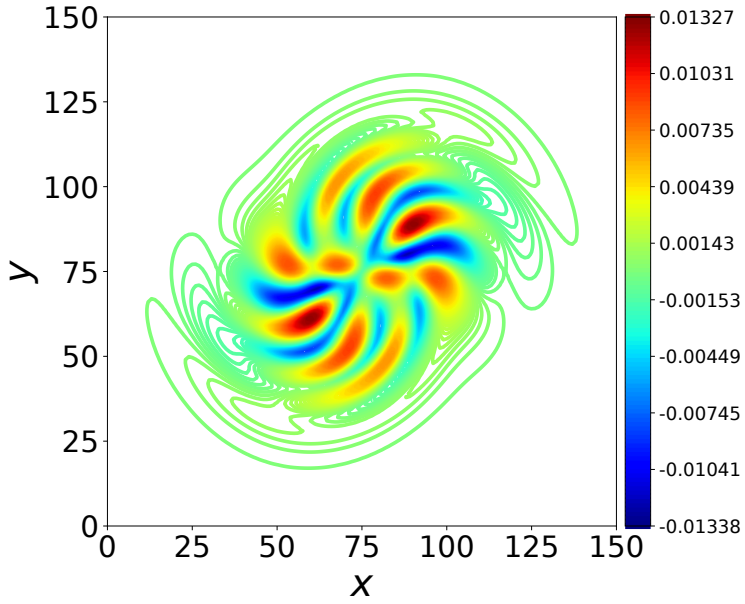


Figure 4.11: Difference between the reconstructed flow fields obtained from GROM + Closure and true projection models for $Re = 1100$.

CHAPTER 5

APPLICATION TO NAVIER-STOKES EQUATION

5.1 Governing Equations

We are interested in the approximation of the constant density incompressible Navier-Stokes (INS) equations:

$$\begin{aligned} &\frac{\partial \mathbf{u}}{\partial t} - Re^{-1}\Delta \mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u} + \nabla p = \mathbf{f} \\ &\nabla\cdot\mathbf{u} = 0 \end{aligned} \tag{5.1}$$

subject to the initial condition

$$\mathbf{u} = \mathbf{u}_0 \quad \text{for } t = 0, \mathbf{x} \in \Omega, \tag{5.2}$$

and the boundary conditions

$$\begin{aligned} \mathbf{u} &= \mathbf{g}_D && \text{on } \mathbf{x} \in \partial\Omega_D, t \in (0,T], \\ \frac{\partial \mathbf{u}}{\partial \mathbf{n}} &= 0,\ p = 0 && \text{on } \mathbf{x} \in \partial\Omega_N, t \in (0,T]. \end{aligned} \tag{5.3}$$

5.2 Full Order Model

The analyses are conducted using libParanumal, which is a high-order discontinuous Galerkin finite-element flow solver developed by the Parallel Numerical Algorithms Group at Virginia Tech [67]. For the context of this thesis, more features about spatial and temporal discretization will not be presented here. The reader may refer to [95] for a more thorough investigation of the mathematical formulation for the GPU-accelerated version of a high-order discontinuous Galerkin discretization of the unsteady INS equations.

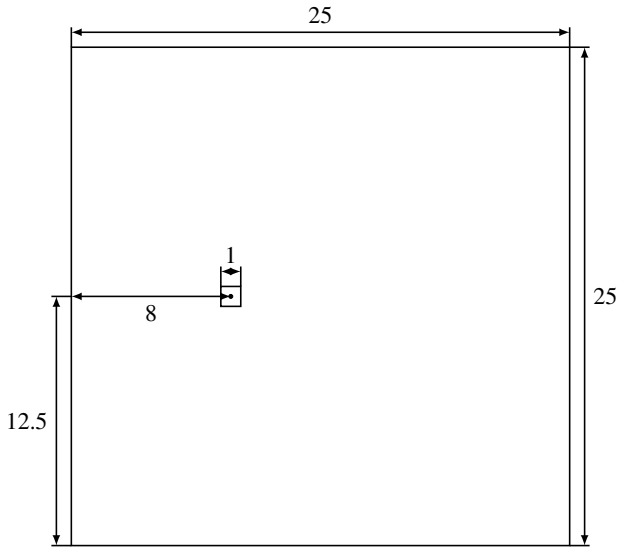


Figure 5.1: Geometry of the flow past a square cylinder

Figure 5.1 illustrates the computational region used to simulate a 2D flow around a square cylinder. The domain is square with dimensions of 25×25, and the square cylinder, centered at the origin $(0, 0)$, has a side length of 1.

No-slip boundary conditions are imposed on both the walls and the cylinder. There is no forcing, and the flow starts from rest.

5.3 Proper Orthogonal Decomposition

In the context of this 2D problem, a total of 500 snapshots for the field variable, i.e. $\mathbf{u}(\mathbf{x}, t^k)$ for $k = 1, 2, \ldots, M$ are gathered over the time interval $t \in [0, 10]$. The index k denotes a specific point in time. The Reynolds number is adjusted across $Re \in [100, 200, 300, 400]$.

The FOM simulations reach a statistically steady state at different time points for the four Reynolds numbers employed in the numerical study. To eliminate the unsteady effects for each Re, the last 200 steps correspond to a 10 s. will be stored to generate the snapshot matrix.

Unlike the previous problems, the dataset for INS consists of velocity vector fields encompassing the x and y directions across a sequence of time steps. After subtract-

ing the mean velocity field from each snapshot, the arrangement of the data within the matrix can be illustrated as follows:

$$\mathbf{S} = \begin{pmatrix} u'_1(t^1) & \dots & u'_1(t^M) \\ u'_2(t^1) & \dots & u'_2(t^M) \\ \vdots & & \vdots \\ u'_N(t^1) & \dots & u'_N(t^M) \\ v'_1(t^1) & \dots & v'_1(t^M) \\ v'_2(t^1) & \dots & v'_2(t^M) \\ \vdots & & \vdots \\ v'_N(t^1) & \dots & v'_N(t^M) \end{pmatrix} \tag{5.4}$$

In this context, the subscript N corresponds to the number of nodes present in the mesh, 34240, and the superscript M represents the number of snapshots, 500. In our approach, we have incorporated both the velocity components represented by u and v in the snapshot matrix. Nevertheless, creating individual snapshot matrices for each component is also possible.

After applying eigenvalue decomposition to the correlation matrix, RIC is calculated to determine the number of modes required to reconstruct the flow field.

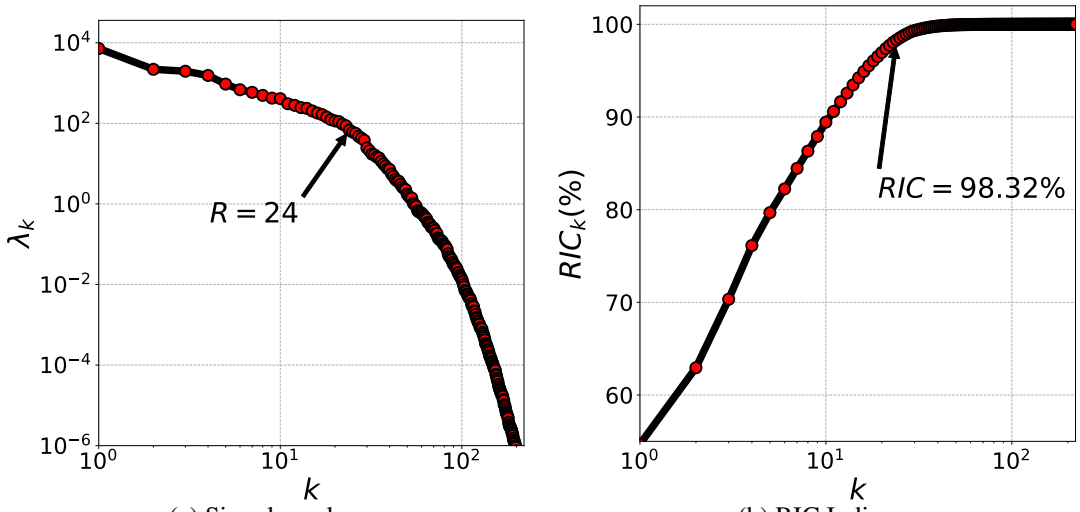
As it is observed from Figure 5.2, eigenvalues do not decrease rapidly, and more modes are required to reach a certain level of information as opposed to previous cases.

The same procedure as in Chapter 3 is followed while obtaining the orthonormal POD modes representing the recurrent spatial structures of the fluctuating flow field. 6 most dominant POD modes for u are depicted in Figure 5.3.

5.4 Galerkin System

Galerkin system for INS can be obtained by replacing the $\mathbf{u}$ in Equation 5.1 with $\mathbf{u_r}$ approximated within the reduced order space $\mathbf{X}^r := \text{span}\{\varphi_1, \dots, \varphi_r\}$.

$$\mathbf{u_r} = \bar{\mathbf{u}}(\mathbf{x}) + \sum_{k=1}^{r} a_k(t)\varphi_k(\mathbf{x}). \tag{5.5}$$



(a) Singular values (b) RIC Indices

Figure 5.2: Eigenvalues originating from the snapshot matrix (a) are paired with their corresponding RIC indices (b) for the space-time solution of the Navier-Stokes equation. We retain the 24 most dominant POD modes in our ROM.

Following this substitution, the ensuing equations are projected onto the ROM space denoted as $\mathbf{X}^r$. As a result of this projection, the GROM with a dimension of r is obtained:

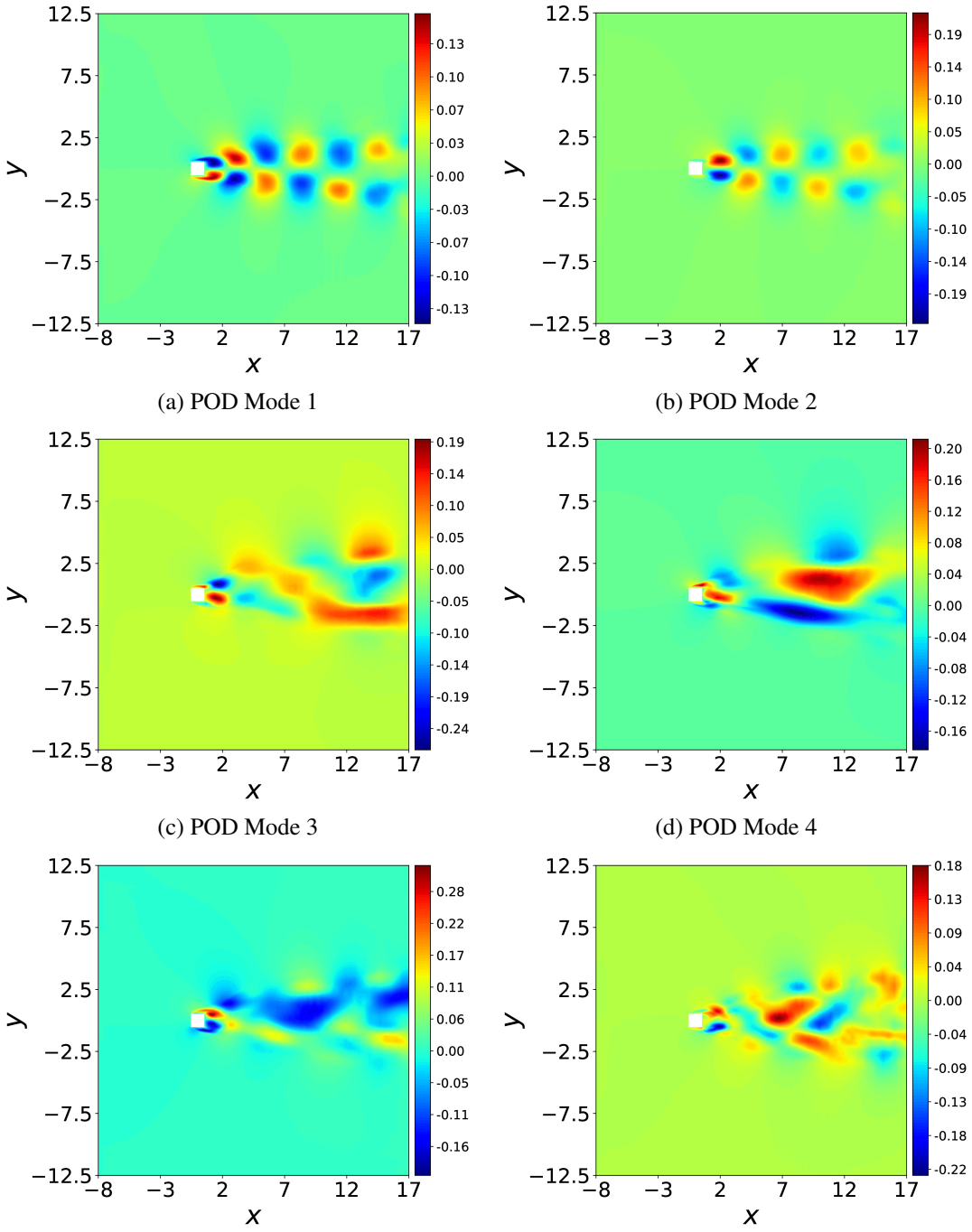
$$\left((\mathbf{u}_r)_t, \boldsymbol{\varphi}_r\right) + Re^{-1}\left(\nabla \mathbf{u}_r, \nabla \boldsymbol{\varphi}_r\right) + \left(\mathbf{u}_r \cdot \nabla \mathbf{u}_r, \boldsymbol{\varphi}_r\right) = 0 \quad \forall \boldsymbol{\varphi}_r \in \mathbf{X}^r. \tag{5.6}$$

The second term on the left-hand side is obtained from the Laplace term through the utilization of integration by parts and the Gaussian theorem. The reader may refer to [25] for the details of the mathematical procedure.

By using a suitable time discretization method, such as the third-order Runge-Kutta method employed in this thesis, we get the full discretization of the r dimensional GROM as follows:

$$\frac{d\hat{a}_k}{dt} = \mathcal{B}_k + \sum_{i=1}^{r} \mathcal{L}_{ik}\hat{a}_i + \sum_{i=1}^{r}\sum_{j=1}^{r} \mathcal{N}_{ijk}\hat{a}_i\hat{a}_j \tag{5.7}$$

for $k = 1, 2, \ldots, r$ where $\hat{a}_k$ is the vector of unknown ROM coefficients, $\mathcal{B}$ is a $r \times 1$



(a) POD Mode 1

(b) POD Mode 2

(c) POD Mode 3

(d) POD Mode 4

(e) POD Mode 5

(f) POD Mode 6

Figure 5.3: Illustration of the 6 most energetic POD basis functions for the x component of the velocity for incompressible Navier-Stokes equation.

vector, $\mathcal{L}$ is a $r \times r$ matrix, and $\mathcal{N}$ is a $r \times r \times r$ tensor for $1 \leq i, j, k \leq r$.

$$\begin{aligned} \mathcal{B}_k &= (L[\bar{\mathbf{u}}] + N[\bar{\mathbf{u}}; \bar{\mathbf{u}}], \boldsymbol{\varphi}_k) \\ \mathcal{L}_{ik} &= (L[\boldsymbol{\varphi}_i] + N[\bar{\mathbf{u}}; \boldsymbol{\varphi}_i] + N[\boldsymbol{\varphi}_i; \bar{\mathbf{u}}], \boldsymbol{\varphi}_k), \\ \mathcal{N}_{ijk} &= \left(N\left[\boldsymbol{\varphi}_i; \boldsymbol{\varphi}_j\right], \boldsymbol{\varphi}_k\right). \end{aligned} \tag{5.8}$$

where $L[\mathbf{v}] = Re^{-1}\Delta\mathbf{v}$ is the diffusion operator and $N[\mathbf{v}; \mathbf{w}] = -\mathbf{v} \cdot \nabla\mathbf{w}$ is the convection operator derived from Equation 5.1.

Our key assumption here revolves around the absence of a pressure term in Equation 5.5. This can be rationalized by recognizing that the POD modes are constructed as linear combination snapshots, which inherently obey the continuity equation, ensuring divergence-free behavior. Consequently, these POD modes maintain the property of being divergence-free. The Galerkin projection of the pressure term is expressed as follows.:

$$(\nabla p, \boldsymbol{\varphi}_k) = \int_\Omega \boldsymbol{\varphi}_k \cdot \nabla p d\boldsymbol{x} = -\int_\Omega p \cdot (\nabla \cdot \boldsymbol{\varphi}_k)\, d\boldsymbol{x} + \int_{\partial\Omega} p \cdot (\boldsymbol{\varphi}_k \cdot \boldsymbol{n})\, d\boldsymbol{x}$$

In enclosed flows, the first term becomes null, and the second term equals zero [96]. The pressure term can be disregarded under the conditions of having a sufficiently large computational domain or appropriate boundary conditions are ensured [46]. However, there are scenarios where omitting this term is not feasible. Consequently, alternative terms might be introduced [97], or an approach involving the formation of a pressure basis becomes necessary [50].

5.5 Closure Model

As the non-linearity in the problem increases, the ANN structure adapted for the closure model should be enhanced. Increasing the number of hidden layers is one way to deal with this problem. It enhances the neural network's ability to grasp intricate patterns and representations within the data, potentially enabling it to closely match the training data and capture intricate features. Since the true time coefficients we obtained for this problem show highly non-linear behavior, we increased the number of hidden layers in the ANN.

Table 5.1: A list of hyperparameters utilized to train the ANN for 2D Navier-Stokes Equation

Variables	**2D Navier-Stokes Equation**
Number of hidden layers	8
Number of neurons in each hidden layer	240
Epochs	5000
Activation functions in the hidden layers	ReLU
Loss function	MSE
Optimizer	ADAM
Learning rate	0.001

5.6 Results

Similarly to the approach adopted for the previous problem, the framework's performance is assessed at $Re = 250$. The FOM solution taken at the last time step for this problem can be observed in the provided Figure 5.4.

However, despite POD providing sets of modes that effectively capture the majority of average turbulent kinetic energy, the importance of the remaining modes remained evident in the context of the vortex shedding problem. This is because POD does not inherently decide which modes are critical for the system's dynamics, and this becomes apparent when observing the gradual (rather than rapid) decrease in the eigenvalues of the correlation matrix depicted in Figure 5.2. At approximately 24 modes, the eigenvalues still exhibit relatively high values, remaining in the order of magnitude of around 10^2. This indicates that the remaining POD modes still contain information that can directly impact the solution to the problem.

In Figure 5.7, we depict the error magnitudes within the proposed model. This error map was generated by calculating the difference between the flow field obtained using the GROM + Closure model and the true projection. Notably, we observe significant deviations, particularly in regions with strong vortices.

Even when employing GROM with an expanded number of modes, such as 50 in

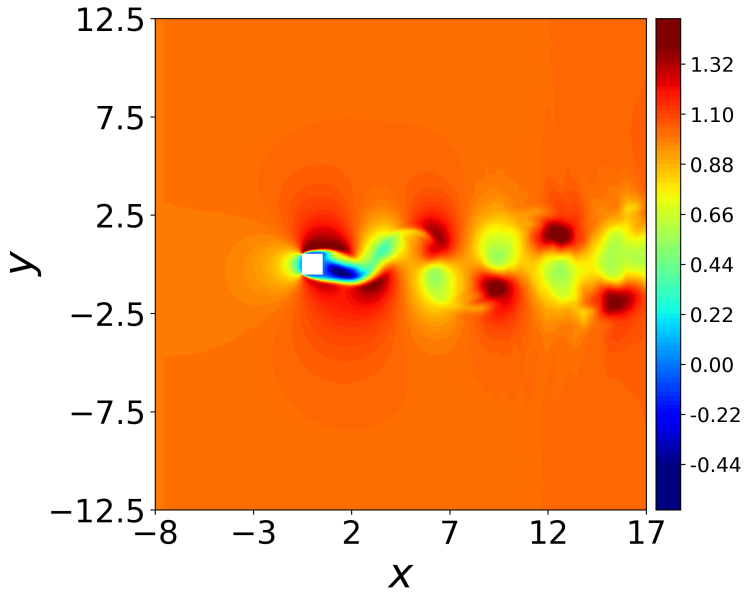
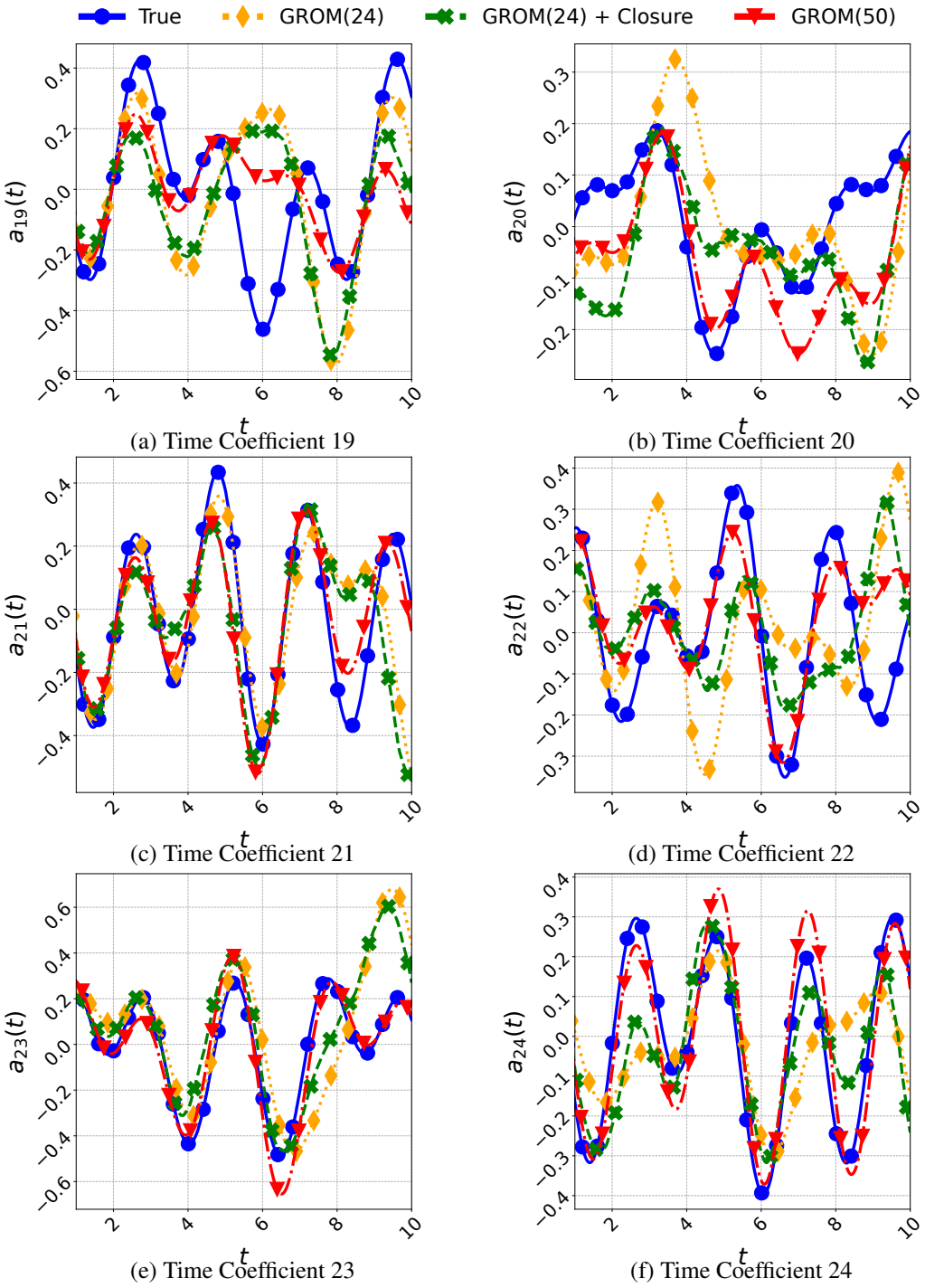


Figure 5.4: u velocity field obtained from the full order model for 2D incompressible Navier-Stokes equation at $t = 10$ for $Re = 250$.

our specific case, it fails to accurately predict the flow field when confronted with an unknown parameter. The further increase in the number of modes would result in a dramatic increase in the computational costs because of the complexity of the problem, i.e., $O(r^3)$, and will lead to a decrease in the efficiency of the proposed framework.

Consequently, the difference between true time coefficients and Galerkin time coefficients becomes highly complex, making it challenging for relatively simple Artificial Neural Network (ANN) structures to make accurate predictions.

As mentioned in Chapter 2.2.1, the POD modes are obtained from a snapshot matrix, which consists of solutions at different Re. This allows us to construct an ODE, which can then be used to find the solution to control parameters not included in the snapshot matrix. If we aimed to predict only the temporal behavior of the dynamical system beyond the range covered by the available data, the snapshot matrix would exclusively consist of solutions corresponding to the specific control parameter of interest. In that case, the decrease in the eigenvalues would be more sudden, and a limited number of modes, such as 10, would be enough to create a ROM.



(a) Time Coefficient 19

(b) Time Coefficient 20

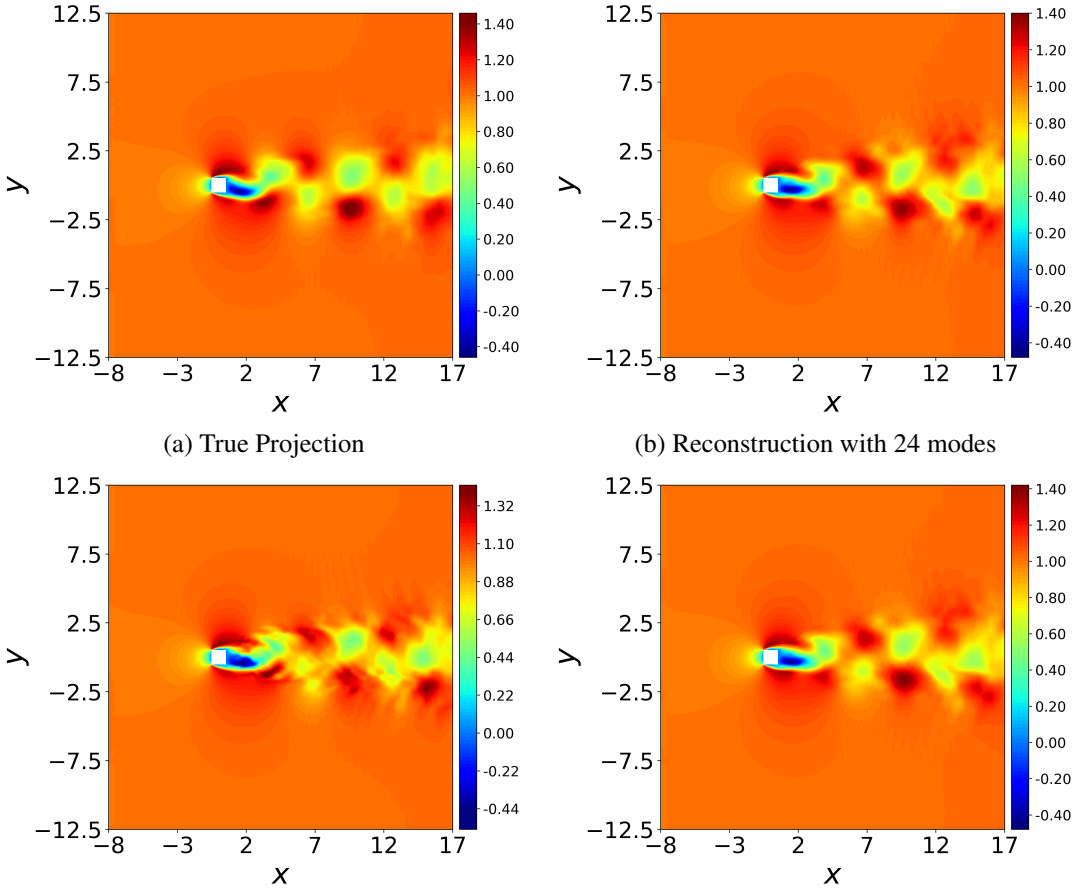
(c) Time Coefficient 21

(d) Time Coefficient 22

(e) Time Coefficient 23

(f) Time Coefficient 24

Figure 5.5: Numerical assessments of time coefficients provided by different frameworks applied to the Navier-Stokes equation for an interpolatory out-of-sample parameter $Re = 250$



(a) True Projection

(b) Reconstruction with 24 modes

(c) Reconstruction with 50 modes

(d) Reconstruction with 24 modes + Closure

Figure 5.6: Illustration of reconstructed the u velocity fields at $t = 10$ for Navier-Stokes equation at $Re = 250$ obtained from GROM and GROM(24) + Closure

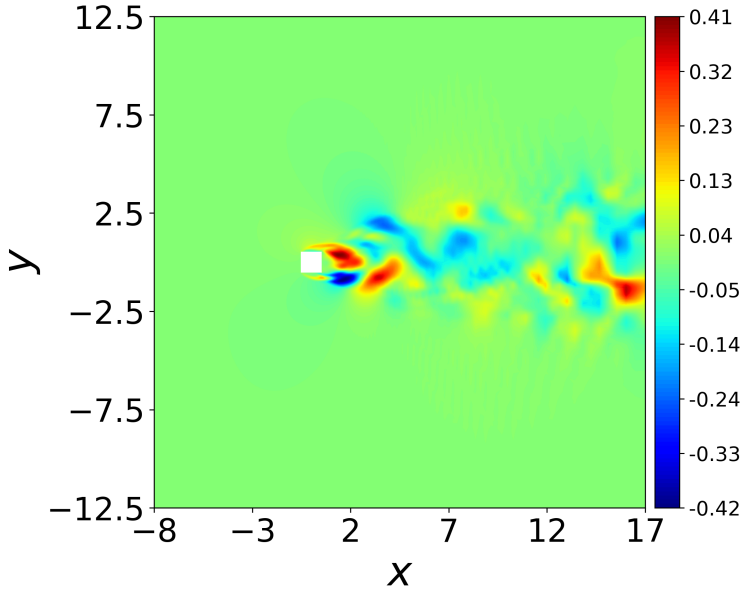


Figure 5.7: Difference between the reconstructed flow fields obtained from GROM + Closure and true projection models for $Re = 250$

CHAPTER 6

CONCLUDING REMARKS

6.1 Summary and Discussions

In this thesis, we have outlined the application of proper orthogonal decomposition (POD) for creating low-dimensional ordinary differential equation (ODE) models in the context of fluid flows. The POD method helps identify empirical eigenfunctions, known as POD modes, that efficiently capture the predominant patterns observed in experimental or numerical data. By projecting the governing partial differential equations (PDEs) of fluid flow onto these POD modes and applying truncation techniques, we can derive simplified, low-dimensional ODE models for the numerical fluid dynamics simulation. In addition to providing a general overview of this methodology, various approaches for numerically calculating the POD modes are presented in this thesis. We have discussed how exploiting the method of snapshots and singular value decomposition (SVD) can simplify and provide insights into these modes. We have presented how Galerkin projection can be used to capture the essential features of the solution while reducing the dimensionality of the problem.

Although reduced order models (ROMs) have been acknowledged as promising in significantly decreasing computational costs while maintaining an acceptable level of accuracy, the complex nature of the flows under consideration necessitates implementing suitable closure modeling strategies. Therefore, we have introduced a closure modeling approach to enhance the conventional Galerkin projection reduced order modeling (GROM) based on artificial neural networks (ANN). First, we provided a brief overview of JAX-based ANN and explained its application in predicting the correction term resulting from the differences between time coefficients obtained

from the true and Galerkin projections. This method can be viewed as a hybrid technique, combining aspects of physics-based and purely data-driven approaches since it builds upon Galerkin projection, thereby improving the model's generalizability and interpretability.

To assess the proposed model‘s performance, we employed three test cases representing convection-dominated flows, the first in a 1D scenario and the others in a 2D. These test cases allowed us to evaluate the proposed ROM under different conditions by examining its performance with two out-of-sample control parameters, which helped us assess its abilities in interpolation and extrapolation scenarios.

In each test case, we first developed GROM from snapshots obtained either through the exact solution of the governing equation or the numerical solution of the problem at different control parameters, i.e., Reynolds number. While the method of snapshots is utilized for computing the POD modes for the 1D Burgers equation and for the 2D incompressible Navier-Stokes equation in Chapters 3 and 5, SVD is used for the same purpose for 2D vorticity-streamfunction equation in Chapter 4. The snapshots are then projected into these POD modes to obtain the true time coefficients, and the ODE resulting from the Galerkin projection was utilized to calculate the Galerkin time coefficients for each Re number. The difference between the actual time coefficients and those acquired through the Galerkin projection results in a correction term. As a closure model, ANN is trained to establish a mapping between Galerkin time coefficients and correction terms.

We observed that the combination of GROM and ANN-based closure model exhibited better performance compared to the GROM approach without any closure model for the first two cases. This performance improvement is maintained across various testing scenarios, demonstrating its effectiveness in interpolating and extrapolating when dealing with simulations featuring different control parameters. We attained a level of accuracy that was previously only achievable by employing a larger number of modes. However, for the 2D incompressible Navier-Stokes equation, this procedure was not entirely straightforward. Even though POD provided sets of modes that capture the most average turbulent kinetic energy, the significance of the remaining modes persisted for the vortex shedding problem. This is because POD does not in-

herently identify which modes are essential for the system's dynamics, and this fact becomes evident when examining the gradual (definitely not rapid) decrease in the eigenvalues of the correlation matrix. Even the GROM with an increased number of modes fails to estimate the flow field at an unknown parameter. As a result, the difference in true time coefficients and Galerkin time coefficients becomes highly complex for simple ANN structures to predict.

6.2 Future Work

Constructing ROMs is a powerful technique that balances computational efficiency with reasonable accuracy, making it applicable to various scientific, engineering, and decision-making tasks. These capabilities will become increasingly important in the future when the numerous engineered systems will be equipped with huge amounts of sensor data, necessitating rapid, real-time simulation. Therefore, in this thesis, our focus has been on enhancing the performance of POD-based ROM by applying machine learning strategies. The results of our proposed models have demonstrated a noteworthy improvement in performance in simple problems. However, when addressing more complex problems, a couple of limitations have been identified. At this point, as a natural extension of this research, the following studies can be considered:

- For estimating POD modes related to an unknown control parameter, more advanced techniques like Grassmann manifold interpolation can be employed [98].
- To simplify calculating derivatives, our framework was designed to accommodate structured meshes exclusively. However, we can incorporate support for unstructured meshes within the framework to expand its applicability to more complex geometries.
- JAX is a remarkably powerful library, particularly in its parallel and distributed computing capabilities and its seamless integration with Graphics Processing Units (GPUs) and Tensor Processing Units (TPUs). These features make it a valuable tool for tackling computationally intensive tasks in machine learning, scientific computing, and beyond. However, transitioning from an object-

oriented programming approach to a functional one can be challenging, requiring a shift in programming paradigms and thinking. Therefore, only the framework in the 1D Burgers equation is completely written in JAX. Transitioning other problems can lead to more modular, maintainable, and parallelizable code.

.

REFERENCES

[1] J. S. Hesthaven and T. Warburton, *Nodal discontinuous Galerkin methods: algorithms, analysis, and applications*. Springer Science & Business Media, 2007.

[2] H. H. Hu, "Computational fluid dynamics," in *Fluid mechanics*, pp. 421–472, Elsevier, 2012.

[3] S. K. Lele, "Compact finite difference schemes with spectral-like resolution," *Journal of computational physics*, vol. 103, no. 1, pp. 16–42, 1992.

[4] B. E. Rapp, *Microfluidics: modeling, mechanics and mathematics*. Elsevier, 2022.

[5] Z. J. Wang, "Spectral (finite) volume method for conservation laws on unstructured grids. basic formulation: Basic formulation," *Journal of computational physics*, vol. 178, no. 1, pp. 210–251, 2002.

[6] A. Baggag, H. Atkins, and D. Keyes, "Parallel implementation of the discontinuous Galerkin method," tech. rep., 1999.

[7] S. M. Rahman, *Reduced Order Modeling of Geophysical Flows Using Physics-based and Data-driven Modeling Techniques*. PhD thesis, Oklahoma State University, 2019.

[8] P. Moin and K. Mahesh, "Direct numerical simulation: a tool in turbulence research," *Annual review of fluid mechanics*, vol. 30, no. 1, pp. 539–578, 1998.

[9] E. Iuliano, "Adaptive sampling strategies for surrogate-based aerodynamic optimization," *Application of Surrogate-based Global Optimization to Aerodynamic Design*, pp. 25–46, 2016.

[10] G. E. Moore *et al.*, "Progress in digital integrated electronics," in *Electron devices meeting*, vol. 21, pp. 11–13, Washington, DC, 1975.

[11] W. Snyder, C. Mou, H. Liu, O. San, R. DeVita, and T. Iliescu, "Reduced order model closures: A brief tutorial," in *Recent Advances in Mechanics and Fluid-Structure Interaction with Applications: The Bong Jae Chung Memorial Volume*, pp. 167–193, Springer, 2022.

[12] U. Baur, P. Benner, and L. Feng, "Model order reduction for linear and nonlinear systems: a system-theoretic perspective," *Archives of Computational Methods in Engineering*, vol. 21, no. 4, pp. 331–358, 2014.

[13] D. J. Lucia, P. S. Beran, and W. A. Silva, "Reduced-order modeling: new approaches for computational physics," *Progress in aerospace sciences*, vol. 40, no. 1-2, pp. 51–117, 2004.

[14] E. Iuliano and D. Quagliarella, "Aerodynamic shape optimization via non-intrusive Galerkin-based surrogate modelling," in *2013 IEEE Congress on Evolutionary Computation*, pp. 1467–1474, IEEE, 2013.

[15] N. Aubry, P. Holmes, J. L. Lumley, and E. Stone, "The dynamics of coherent structures in the wall region of a turbulent boundary layer," *Journal of fluid Mechanics*, vol. 192, pp. 115–173, 1988.

[16] G. Berkooz, P. Holmes, and J. L. Lumley, "The proper orthogonal decomposition in the analysis of turbulent flows," *Annual review of fluid mechanics*, vol. 25, no. 1, pp. 539–575, 1993.

[17] Z. Bai and L. Peng, "Non-intrusive nonlinear model reduction via machine learning approximations to low-dimensional operators," *Advanced Modeling and Simulation in Engineering Sciences*, vol. 8, no. 1, p. 28, 2021.

[18] E. Sachs and S. Volkwein, "Reduced-order Galerkin approximations in PDE constrained optimization,"

[19] S. E. Ahmed, S. Pawar, O. San, A. Rasheed, T. Iliescu, and B. R. Noack, "On closures for reduced order models—a spectrum of first-principle to machine-learned avenues," *Physics of Fluids*, vol. 33, no. 9, p. 091301, 2021.

[20] P. Patil and H. Babaee, "Real-time reduced-order modeling of stochastic partial differential equations via time-dependent subspaces," *Journal of Computational Physics*, vol. 415, p. 109511, 2020.

[21] S. E. Ahmed, S. Pawar, O. San, and A. Rasheed, "Reduced order modeling of fluid flows: Machine learning, Kolmogorov barrier, closure modeling, and partitioning," in *AIAA Aviation 2020 forum*, p. 2946, 2020.

[22] J. S. Hesthaven, G. Rozza, B. Stamm, *et al.*, *Certified reduced basis methods for parametrized partial differential equations*, vol. 590. Springer, 2016.

[23] A. Dumon, C. Allery, and A. Ammar, "Proper general decomposition (PGD) for the resolution of Navier-Stokes equations," *Journal of Computational Physics*, vol. 230, no. 4, pp. 1387–1407, 2011.

[24] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor, *Dynamic mode decomposition: data-driven modeling of complex systems*. SIAM, 2016.

[25] S. Giere, *Numerical and analytical aspects of POD-based reduced-order modeling in computational fluid dynamics*. PhD thesis, 2016.

[26] K. Pearson, "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, vol. 2, no. 11, pp. 559–572, 1901.

[27] H. Hotelling, "Analysis of a complex of statistical variables into principal components.," *Journal of educational psychology*, vol. 24, no. 6, p. 417, 1933.

[28] K. Lu, Y. Jin, Y. Chen, Y. Yang, L. Hou, Z. Zhang, Z. Li, and C. Fu, "Review for order reduction based on proper orthogonal decomposition and outlooks of applications in mechanical systems," *Mechanical Systems and Signal Processing*, vol. 123, pp. 264–297, 2019.

[29] L. Sirovich, "Turbulence and the dynamics of coherent structures. i. coherent structures," *Quarterly of applied mathematics*, vol. 45, no. 3, pp. 561–571, 1987.

[30] S. A. Renganathan, R. Maulik, and V. Rao, "Machine learning for nonintrusive model order reduction of the parametric inviscid transonic flow past an airfoil," *Physics of Fluids*, vol. 32, no. 4, p. 047110, 2020.

[31] E. Iuliano and D. Quagliarella, "Proper orthogonal decomposition, surrogate modelling and evolutionary optimization in aerodynamic design," *Computers & Fluids*, vol. 84, pp. 327–350, 2013.

[32] C. W. Rowley, T. Colonius, and R. M. Murray, "Model reduction for compressible flows using POD and Galerkin projection," *Physica D: Nonlinear Phenomena*, vol. 189, no. 1-2, pp. 115–129, 2004.

[33] O. San, R. Maulik, and M. Ahmed, "An artificial neural network framework for reduced order modeling of transient flows," *Communications in Nonlinear Science and Numerical Simulation*, vol. 77, pp. 271–287, 2019.

[34] T. Bui-Thanh, M. Damodaran, and K. Willcox, "Aerodynamic data reconstruction and inverse design using proper orthogonal decomposition," *AIAA journal*, vol. 42, no. 8, pp. 1505–1516, 2004.

[35] R. Everson and L. Sirovich, "Karhunen–loeve procedure for gappy data," *JOSA A*, vol. 12, no. 8, pp. 1657–1664, 1995.

[36] B. I. Epureanu, E. H. Dowell, and K. C. Hall, "A parametric analysis of reduced order models of potential flows in turbomachinery using proper orthogonal decomposition," in *Turbo Expo: Power for Land, Sea, and Air*, vol. 78507, p. V001T03A056, American Society of Mechanical Engineers, 2001.

[37] T. Bui-Thanh, M. Damodaran, and K. Willcox, "Proper orthogonal decomposition extensions for parametric applications in compressible aerodynamics," in *21st AIAA applied aerodynamics conference*, p. 4213, 2003.

[38] S. Ullmann, *POD-Galerkin modeling for incompressible flows with stochastic boundary conditions*. 2014.

[39] P. LeGresley and J. Alonso, "Investigation of non-linear projection for Galerkin based reduced order models for aerodynamics," in *39th AIAA Aerospace Sciences Meeting and Exhibit*, p. 926, 2001.

[40] Z. Wang, *Reduced-order modeling of complex engineering and geophysical flows: analysis and computations*. PhD thesis, Virginia Tech, 2012.

[41] S. Grimberg, C. Farhat, and N. Youkilis, "On the stability of projection-based model order reduction for convection-dominated laminar and turbulent flows," *Journal of Computational Physics*, vol. 419, p. 109681, 2020.

[42] V. Zucatti and W. Wolf, "Data-driven closure of projection-based reduced order models for unsteady compressible flows," *Computer Methods in Applied Mechanics and Engineering*, vol. 386, p. 114120, 2021.

[43] A. N. Kolmogorov, "The local structure of turbulence in incompressible viscous fluid for very large Reynolds numbers," *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, vol. 434, no. 1890, pp. 9–13, 1991.

[44] S. B. Pope and S. B. Pope, *Turbulent flows*. Cambridge university press, 2000.

[45] B. Koc, *Numerical Analysis for Data-Driven Reduced Order Model Closures*. PhD thesis, Virginia Tech, 2021.

[46] Z. Wang, I. Akhtar, J. Borggaard, and T. Iliescu, "Proper orthogonal decomposition closure models for turbulent flows: a numerical comparison," *Computer Methods in Applied Mechanics and Engineering*, vol. 237, pp. 10–26, 2012.

[47] X. Xie, D. Wells, Z. Wang, and T. Iliescu, "Approximate deconvolution reduced order modeling," *Computer Methods in Applied Mechanics and Engineering*, vol. 313, pp. 512–534, 2017.

[48] S. Ullmann and J. Lang, "A POD-Galerkin reduced model with updated coefficients for Smagorinsky LES," in *V European conference on computational fluid dynamics, ECCOMAS CFD*, p. 2010, 2010.

[49] Z. Wang, I. Akhtar, J. Borggaard, and T. Iliescu, "Two-level discretizations of nonlinear closure models for proper orthogonal decomposition," *Journal of Computational Physics*, vol. 230, no. 1, pp. 126–146, 2011.

[50] M. Bergmann, C.-H. Bruneau, and A. Iollo, "Enablers for robust Galerkin models," *Journal of Computational Physics*, vol. 228, no. 2, pp. 516–538, 2009.

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

[52] S. Giere, T. Iliescu, V. John, and D. Wells, "SUPG reduced order models

for convection-dominated convection–diffusion–reaction equations," *Computer Methods in Applied Mechanics and Engineering*, vol. 289, pp. 454–474, 2015.

[53] H. Mori, "Transport, collective motion, and brownian motion," *Progress of Theoretical Physics*, vol. 33, no. 3, pp. 423–455, 1965.

[54] R. Zwanzig, "Nonlinear generalized Langevin equations," *Journal of Statistical Physics*, vol. 9, no. 3, pp. 215–220, 1973.

[55] P. Stinis, "Renormalized Mori-Zwanzig-reduced models for systems without scale separation," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 471, no. 2176, p. 20140446, 2015.

[56] Q. Wang, N. Ripamonti, and J. S. Hesthaven, "Recurrent neural network closure of parametric POD-Galerkin reduced-order models based on the mori-zwanzig formalism," *Journal of Computational Physics*, vol. 410, p. 109402, 2020.

[57] B. Koc, C. Mou, H. Liu, Z. Wang, G. Rozza, and T. Iliescu, "Verifiability of the data-driven variational multiscale reduced order model," *Journal of Scientific Computing*, vol. 93, no. 2, p. 54, 2022.

[58] B. Galletti, A. Bottaro, C.-H. Bruneau, and A. Iollo, "Accurate model reduction of transient and forced wakes," *European Journal of Mechanics-B/Fluids*, vol. 26, no. 3, pp. 354–366, 2007.

[59] M. Couplet, C. Basdevant, and P. Sagaut, "Calibrated reduced-order POD-Galerkin system for fluid flow modelling," *Journal of Computational Physics*, vol. 207, no. 1, pp. 192–220, 2005.

[60] R. Bourguet, M. Braza, and A. Dervieux, "Reduced-order modeling for unsteady transonic flows around an airfoil," *Physics of Fluids*, vol. 19, no. 11, 2007.

[61] I. Kalashnikova, B. van Bloemen Waanders, S. Arunajatesan, and M. Barone, "Stabilization of projection-based reduced order models for linear time-invariant systems via optimization-based eigenvalue reassignment," *Computer Methods in Applied Mechanics and Engineering*, vol. 272, pp. 251–270, 2014.

[62] X. Xie, M. Mohebujjaman, L. G. Rebholz, and T. Iliescu, "Data-driven filtered reduced order modeling of fluid flows," *SIAM Journal on Scientific Computing*, vol. 40, no. 3, pp. B834–B857, 2018.

[63] C. Mou, H. Liu, D. R. Wells, and T. Iliescu, "Data-driven correction reduced order models for the quasi-geostrophic equations: A numerical investigation," *International Journal of Computational Fluid Dynamics*, vol. 34, no. 2, pp. 147–159, 2020.

[64] O. San and R. Maulik, "Neural network closures for nonlinear model order reduction," *Advances in Computational Mathematics*, vol. 44, pp. 1717–1750, 2018.

[65] O. San and R. Maulik, "Machine learning closures for model order reduction of thermal fluids," *Applied Mathematical Modelling*, vol. 60, pp. 681–710, 2018.

[66] S. E. Ahmed, O. San, A. Rasheed, and T. Iliescu, "A long short-term memory embedding for hybrid uplifted reduced order models," *Physica D: Nonlinear Phenomena*, vol. 409, p. 132471, 2020.

[67] N. Chalmers, A. Karakus, A. P. Austin, K. Swirydowicz, and T. Warburton, "libParanumal: a performance portable high-order finite element library," 2022. Release 0.5.0.

[68] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," 2018.

[69] P. A. LeGresley, *Application of proper orthogonal decomposition (POD) to design decomposition methods*. Stanford University, 2006.

[70] M. Loeve, *Probability theory: foundations, random sequences*. New York, NY: Van Nostrand, 1955.

[71] E. N. Lorenz, *Empirical orthogonal functions and statistical weather prediction*, vol. 1. Massachusetts Institute of Technology, Department of Meteorology Cambridge, 1956.

[72] P. Holmes, J. L. Lumley, G. Berkooz, and C. W. Rowley, *Turbulence, coherent structures, dynamical systems and symmetry*. Cambridge university press, 2012.

[73] P. J. Holmes, J. L. Lumley, G. Berkooz, J. C. Mattingly, and R. W. Wittenberg, "Low-dimensional models of coherent structures in turbulence," *Physics Reports*, vol. 287, no. 4, pp. 337–384, 1997.

[74] J. Weiss, "A tutorial on the proper orthogonal decomposition," in *AIAA Aviation 2019 Forum*, p. 3333, 2019.

[75] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, Sept. 2020.

[76] T. M. Inc., "MATLAB version: 9.13.0 (r2022b)," 2022.

[77] J. S. Peterson, "The reduced basis method for incompressible viscous flow calculations," *SIAM Journal on Scientific and Statistical Computing*, vol. 10, no. 4, pp. 777–786, 1989.

[78] A. K. Noor and J. M. Peters, "Reduced basis technique for nonlinear analysis of structures," *AIAA journal*, vol. 18, no. 4, pp. 455–462, 1980.

[79] K. Ito and S. S. Ravindran, "A reduced basis method for control problems governed by PDEs," tech. rep., North Carolina State University. Center for Research in Scientific Computation, 1997.

[80] T. R. Smith, J. Moehlis, and P. Holmes, "Low-dimensional modelling of turbulence using the proper orthogonal decomposition: a tutorial," *Nonlinear Dynamics*, vol. 41, pp. 275–307, 2005.

[81] M. Barone, D. Segalman, H. Thornquist, and I. Kalashnikova, "Galerkin reduced order models for compressible flow with structural interaction," in *46th AIAA Aerospace Sciences Meeting and Exhibit*, p. 612, 2008.

[82] A. Winnicka, "Comparison of numerical integration methods," *CEUR-WS. org*, vol. 2468, p. 1, 2019.

[83] M. Bergmann, L. Cordier, and J.-P. Brancher, "Optimal rotary control of the cylinder wake using proper orthogonal decomposition reduced-order model," *Physics of fluids*, vol. 17, no. 9, p. 097101, 2005.

[84] W. R. Graham, J. Peraire, and K. Tang, "Optimal control of vortex shedding using low-order models. part i—open-loop model development," *International Journal for Numerical Methods in Engineering*, vol. 44, no. 7, pp. 945–972, 1999.

[85] M. D. Gunzburger, J. S. Peterson, and J. N. Shadid, "Reduced-order modeling of time-dependent PDEs with multiple parameters in the boundary data," *Computer methods in applied mechanics and engineering*, vol. 196, no. 4-6, pp. 1030–1047, 2007.

[86] S. Ravindran, "Control of flow separation over a forward-facing step by model reduction," *Computer methods in applied mechanics and engineering*, vol. 191, no. 41-42, pp. 4599–4617, 2002.

[87] J. Baiges, R. Codina, and S. R. Idelsohn, "Reduced-order modelling strategies for the finite element approximation of the incompressible Navier-Stokes equations," *Numerical Simulations of Coupled Problems in Engineering*, pp. 189–216, 2014.

[88] A. Falaize, E. Liberge, and A. Hamdouni, "POD-based reduced order model for flows induced by rigid bodies in forced rotation," *Journal of Fluids and Structures*, vol. 91, p. 102593, 2019.

[89] J. Roth, "Proper orthogonal decomposition for fluid mechanics problems," *Leibniz Universitat Hannover*, 2021.

[90] A. Aygün, "Physics informed neural networks for computational fluid dynamics," Master's thesis, Middle East Technical University, 2023.

[91] J. Heek, A. Levskaya, A. Oliver, M. Ritter, B. Rondepierre, A. Steiner, and M. van Zee, "Flax: A neural network library and ecosystem for JAX," 2023.

[92] I. Babuschkin, K. Baumli, A. Bell, S. Bhupatiraju, J. Bruce, P. Buchlovsky, D. Budden, T. Cai, A. Clark, I. Danihelka, A. Dedieu, C. Fantacci, J. Godwin, C. Jones, R. Hemsley, T. Hennigan, M. Hessel, S. Hou, S. Kapturowski, T. Keck, I. Kemaev, M. King, M. Kunesch, L. Martens, H. Merzic, V. Mikulik, T. Norman, G. Papamakarios, J. Quan, R. Ring, F. Ruiz, A. Sanchez, R. Schneider, E. Sezener, S. Spencer, S. Srinivasan, W. Stokowiec, L. Wang, G. Zhou, and F. Viola, "The DeepMind JAX Ecosystem," 2020.

[93] M. Maleewong and S. Sirisup, "On-line and off-line POD assisted projective integral for non-linear problems: A case study with burgers-equation," *International Journal of Mathematical and Computational Sciences*, vol. 5, no. 7, pp. 984–992, 2011.

[94] J. Von Hardenberg, J. McWilliams, A. Provenzale, A. Shchepetkin, and J. Weiss, "Vortex merging in quasi-geostrophic flows," *Journal of Fluid Mechanics*, vol. 412, pp. 331–353, 2000.

[95] A. Karakus, N. Chalmers, K. Świrydowicz, and T. Warburton, "A GPU accelerated discontinuous Galerkin incompressible flow solver," *Journal of Computational Physics*, vol. 390, pp. 380–404, 2019.

[96] S. Lorenzi, A. Cammi, L. Luzzi, and G. Rozza, "POD-Galerkin method for finite volume approximation of Navier-Stokes and RANS equations," *Computer Methods in Applied Mechanics and Engineering*, vol. 311, pp. 151–179, 2016.

[97] B. R. Noack, P. Papas, and P. A. Monkewitz, "The need for a pressure-term representation in empirical Galerkin models of incompressible shear flows," *Journal of Fluid Mechanics*, vol. 523, pp. 339–365, 2005.

[98] O. Friderikos, E. Baranger, M. Olive, and D. Néron, "On the stability of pod basis interpolation on grassmann manifolds for parametric model order reduction," *Computational Mechanics*, vol. 70, no. 1, pp. 181–204, 2022.