Data-driven flow modelling using machine learning and data assimilation approaches

Nishant Kumar

THÈSE

Pour l'obtention du Grade de

DOCTEUR DE L'UNIVERSITÉ DE POITIERS
Faculté des Sciences Fondamentales et Appliquées
Diplôme National - Arrêté du 25 mai 2016

Ecole Doctorale : École Doctorale Sciences et Ingénierie des Matériaux, Mécanique, Énergétique (ED 609)

Secteur de Recherche : Mécanique des milieux fluides

Présentée par

Nishant KUMAR

DATA-DRIVEN FLOW MODELLING USING MACHINE LEARNING AND DATA ASSIMILATION APPROACHES

Directeurs de Thèse : Laurent CORDIER et Franck KERHERVÉ

Defense planned on 8th September 2021

— JURY —

M. BOCQUET	Professeur, CEREA, Champs-sur-Marne	Rapporteur
B. PODVIN	Chargée de recherches CNRS - HDR, LISN, Orsay	Rapporteur
R. FABLET	Professeur, Lab-STICC, Brest	Examinateur
L. MATHELIN	Chargé de recherches CNRS, LISN, Orsay	Examinateur
E. MÉMIN	Directeur de recherche INRIA, Rennes	Examinateur
D. SIPP	Directeur de recherche ONERA, Meudon	Examinateur
L. CORDIER	Directeur de recherche CNRS, Institut Pprime	Directeur de thèse
F. KERHERVÉ	Maître de conférences, Institut Pprime	Directeur de thèse

— INVITÉ —

M. MELDI Maître de conférences, Institut Pprime Examinateur

Contents

Chapter 1

Introduction

Contents

1.1 Motivation

A wide range of systems in the field of fluid mechanics exhibit *complex* dynamics. The complexity pertains to spatial and temporal features originating from instabilities, nonlinearities, or turbulence which span a range from small to large scales. This is especially the case in flows involving high Reynolds number, compressibility, or combustion, to cite just a few. For example, the flow in the wake of a circular cylinder, for a range of Reynolds number, exhibits different flow regimes characterized by small and large scale vortical structures, as shown in Fig. 1.1. In these flow cases, very fine spatial and temporal discretizations are required for the adequate resolution and propagation of the flow states. One of the ways to study the complex underlying physical phenomena is provided by the numerical analysis of associated full-scale models governing the flow dynamics. Generally, a detailed analysis of the flow features demands the availability of highly resolved spatio-temporal data. The discretized systems for complex problems can often have a huge number of degrees of freedom, *e.g.* $Re^{9/4}$ for direct numerical simulations. Also, the growing requirement for improved accuracy requires the inclusion of more details in the modeling stage, which inevitably leads to larger-scale, more complex models of dynamical systems (Benner et al., 2015).

Over the past few decades, the analysis of more and more complex flow systems has been made possible by the advancements in numerical simulation and experimental measurement tools. However, the analysis of complex fluid flows poses the challenge of handling large data representing the high-dimensional nonlinear dynamics. In these large-scale settings, the

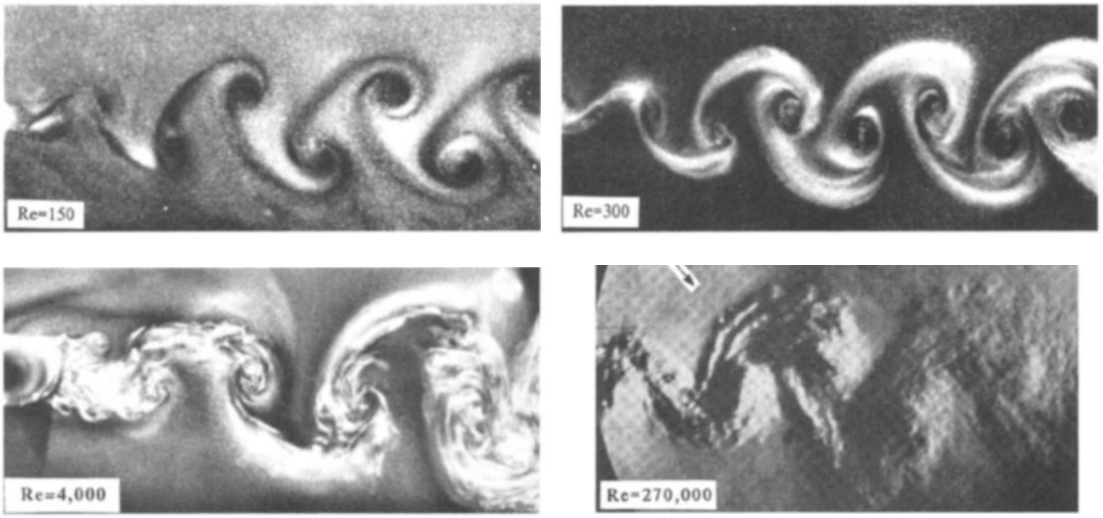


Figure 1.1: Visualization of laminar and turbulent vortex streets in the wake of a circular cylinder, over a wide range of Reynolds numbers. The flow features vortices which are shed in pairs, alternately from each side of the rear surface of the cylinder, when the wake starts to oscillate at $Re > 35$, forming a Kármán vortex street. As Re increases ($Re > 200$), the wake becomes more complex and turbulent, but the vortices can still be detected in the form of coherent structures. For even higher values of Re, the vortex street starts to degenerate into a turbulent wake with unstable vortices splitting into smaller ones and vanishing downstream. The visualizations are from Williamson (1996).

challenge appears in the form of high computational cost, which often hinders the use of high-fidelity tools in applications where repeated realizations of the system are required. These applications include real-time control (Bergmann and Cordier, 2008a), multidisciplinary optimization (MDO) (LeGresley and Alonso, 2000), and uncertainty quantification (UQ) (Mathelin et al., 2005), for example, which are driven by the ever-increasing push towards improving system performance. **To alleviate this computational burden, it is of key interest to develop and study approaches that seek to reduce the size and cost of computational models while minimizing the loss of physical fidelity.** Specifically, one of the key requirements for closed-loop control is the possibility to robustly predict the state of a fluid system and forecast its evolution. This is the main motivation for deriving reduced, low-dimensional, efficient models that are computationally tractable and accurate.

Reduced-order modeling has been demonstrated as a viable approach to mitigate the issue of high computational cost as it offers the potential to simulate physical and dynamical systems with substantially increased computational efficiency without sacrificing accuracy (Alfio Quarteroni, 2014). Reduced-order modeling is a rapidly growing field, offering advantage in terms of facilitating the real-time turn-around of computational results. Over the years, a large variety of reduced-order modeling methodologies have been proposed to address the reduction of degrees of freedom of the dynamics of a system. These methods are mostly centered on physics-based approach, such as, projection-based models. However, there is a more recent growing attention on non-intrusive approaches in which empirical models are derived from the input-output data, such as, subspace identification and neural networks. **The main objective of this work, therefore, is to develop methodologies for reduced-order modeling in order to enable the dynamical prediction of fluid flow states.**

1.2 From physical to latent space

Even though at first glance, the analysis of flow systems may seem daunting, it has been observed that similar spatial flow features – like the von Kármán vortex street and Kelvin-Helmholtz instability – emerge across a wide range of flow geometries or over a wide range of flow parameters. These features are often visually recognizable even under the presence of perturbations in the flow. The occurrence of these similar prominent features across a range of flows indicates the existence of common underlying phenomena that capture the essence of the flow (Taira, Brunton, et al., 2017). The *modal decomposition* techniques are used to mathematically extract these underlying flow features based on some energetic or dynamical criteria. Modal decomposition provides *modes*, which are the spatial features of the flow, and *modal coefficients*, which are the associated characteristic values representing the energy content levels or growth rates and frequencies. The modes enable the realization of a low-dimensional *latent space* coordinate system (reduced basis) from a high-dimensional *physical space*. The reduced basis in the selected coordinate system allows a low-order approximation, with as few degrees of freedom as possible, of the flow dynamics in terms of its dominant components.

Several modal decomposition methods have been employed in literature for the purpose of analyzing a wide range of fluid flows. Each method is unique in terms of extraction of the modal structures which highlight different aspects of the flow field. Based on the inputs used for obtaining the modes, the modal decomposition methods are classified into two main groups (Taira, Brunton, et al., 2017). One of the categories, the *data-based* methods, uses flow-field data from numerical simulations or experiments to obtain the modes and does not necessitate the knowledge of the laws governing the dynamics. It includes methods like proper orthogonal decomposition (POD) (Holmes et al., 2012) and dynamic mode decomposition (DMD) (Schmid, 2010). In contrast to the data-based methods, the other category of *operator-based* methods uses the operator describing the state dynamics to obtain the modes. It includes methods like Koopman analysis (Mezić, 2013), global linear stability analysis (Theofilis, 2011), and resolvent analysis (Schmid et al., 2002). The first operator-based method rigorously connects DMD to nonlinear dynamical systems while the latter two techniques provide the growth or decay characteristics of perturbations with respect to a given base or mean flow. It must be acknowledged that these methods are constrained by the accuracy of the information provided. In this work, the data-based modal decomposition methods are considered.

In the context of fluid dynamics, POD analysis was first introduced by Lumley (1967) as a method for extracting and analyzing coherent structures in experimental turbulent flows, and was later used for analyzing numerical simulations of turbulent flows (Podvin and Lumley, 1998; Wang, Akhtar, et al., 2012). The relationship between spatial and temporal structures was investigated by Aubry (1991), who presented a variant of the decomposition, referred to as bi-orthogonal decomposition (BOD), which focused on the temporal structures of the modes. A *snapshot POD* variant was proposed by Sirovich (1987) to compute both the spatial and temporal components of the decomposition. Subsequently, a link between POD and an eigen-decomposition method known as *singular value decomposition* (SVD) enabled the formulation of a discrete dataset in terms of a simple matrix factorization, as demonstrated by Kunisch and Volkwein (1999). The eigendecomposition method is in contrast to the formulations based on statistics or dynamical system theory, such as the relation between POD symmetry and ergodicity reviewed by Berkooz (1992), or the hierarchy of low-dimensional Galerkin models presented by Noack, Afanasiev, et al. (2003). Owing to the simple matrix factorization framework, POD is a popular tool for analyzing experimental and numerical fluid mechanics. Typical applications of POD include the identification of coherent structures from experimental data (Delville,

1994), flow control (Bergmann and Cordier, 2008b; Brunton and Noack, 2015), reduced-order modeling (Deane et al., 1991), and data-driven identification of nonlinear systems (Brunton, Proctor, et al., 2016b; Loiseau and Brunton, 2018). Excellent reviews on POD can be found in Holmes et al. (2012), and Chapter 3 of Cordier and Bergmann (2008a).

DMD is another decomposition method which, unlike POD, extracts the dynamic modes along with the associated growth rates and frequencies from the flow field data. This decomposition originates from the Koopman theory, which permits the study of a finite dimensional nonlinear system by mapping it onto an infinite dimensional linear one (Brunton, Brunton, et al., 2016). The extraction of finite dimensional approximations of such Koopman (linear) dynamical system from the data for reduced-order modeling applications was performed by Rowley, Mezić, et al. (2009) and Schmid (2010). The connection of the Koopman framework to the DMD has been reviewed by Mezić (2013) and an extensive overview of the DMD and its application is given by Kutz, Brunton, et al. (2016).

In the manuscript, both the modal decomposition methods (POD and DMD) have been introduced and subsequent development of the data-driven reduced-order modeling has been performed in the latent space provided by POD modes.

1.3 Reduced-order modeling in latent space

As already highlighted in Sec. 1.1, obtaining low-dimensional and computationally tractable models as an alternative to the more accurate but computationally intractable high-fidelity models is crucial for the successful implementation in applications requiring repeated realizations of the dynamical system. The field of model reduction encompasses a broad set of mathematical methods which use the reduced basis obtained from modal decomposition techniques to generate and evaluate these low dimensional models known as *reduced-order models* (ROMs) (Noack, Morzyński, et al., 2011; Mendonça et al., 2019). The possibility to obtain approximations of the high-fidelity dynamics with low computational times has led both the industry and the scientific community to investigate ROMs. Specifically, projection-based reduced-order modeling is the most common approach of flow estimation in the flow control community. The low number of degrees of freedom of ROMs makes the classical control techniques feasible. However, the complexity of the models used in practical applications means that numerical packages for creating and computing these models are not readily available.

ROMs are fundamentally characterized by the reduced basis that defines them. As a consequence, the choice of the projection method to represent the dynamics has direct implications on the success of the subsequent modeling and control strategy. Some of the commonly used projection methods for constructing ROMs are POD method (Berkooz et al., 1993), DMD method (Tissot et al., 2014), reduced basis method (Peterson, 1989; Ito and Ravindran, 2001; Haasdonk and Ohlberger, 2008), cross Gramian method (Baur and Benner, 2008; Himpe and Ohlberger, 2014), piecewise tangential interpolation method (Baur, Beattie, et al., 2011), matrix interpolation method (Degroote et al., 2010; Panzer et al., 2010), approximate balancing method (Chiu, 1996), and balanced truncation method (Gugercin and Antoulas, 2004; Lall et al., 2002; Mehrmann and Stykel, 2005). In this work, the data-driven projection method of POD has been used for constructing ROMs for fluid dynamics problems. The methods based on POD and its variants have been applied successfully to numerous research fields including fluid flow control (Kunisch, Volkwein, and Xie, 2004; Bergmann, Cordier, and Brancher, 2005b; Hoepffner et al., 2006; Bagheri et al., 2009; Barbagallo et al., 2009; Ahuja and Rowley, 2010) and data assimilation (Qiu and Chou, 2006; Cao et al., 2007; Dimitriu and Apreutesei, 2007).

The reduced-order models are divided into two categories based on their dependency on governing equations (Frangos et al., 2010) – physics-based *intrusive* ROM and purely data-driven *non-intrusive* ROM. **In this work, ROMs belonging to both these categories have been introduced and analyzed**.

1.3.1 Intrusive reduced-order modeling

By construction, intrusive ROM is dependent on governing equations. In this manuscript, ROM belonging to this category is derived using a method combining POD and Galerkin projection. Among the various reduced-order modeling techniques, this approach has attracted considerable attention for applications in fluid flow analysis. The POD-Galerkin ROM reduces the number of degrees of freedom required to numerically solve the Navier-Stokes equations by introducing spatial basis functions which are specifically adapted to the system. This is unlike the finite volume, finite difference, finite element, or spectral methods that are used to solve the Navier-Stokes equations, where the basis functions typically have little connection with the problem being solved and thus require many functions to represent the solution. As the POD modes are generated for a specific flow, an accurate model of that flow is obtained using only a small number of modes. The model is able to retain, to a large extent, the physical characteristics from the original system and it can be used for prediction of the flow behavior. This POD-Galerkin procedure was first introduced by Aubry et al. (1988) where it was used to derive low-order dynamical system of a turbulent shear flow. Subsequently, this modeling approach has been applied to several flow configurations such as turbulent mixing layer (Ukeiley et al., 2001), and turbulent cylinder wake (Noack and Eckelmann, 1994). All the POD-Galerkin systems that have been investigated in the literature exhibit a polynomial form of either a quadratic (Rajaee et al., 1994) or cubic (Aubry et al., 1988) order depending on the approach to building the model (Rempfer, 1996).

The POD-Galerkin ROM gives the evolution of the time varying coefficients of the POD modes. Usually, a number of modes in the order of 10^0–10^2 are used to approximate the original flow state. This leads to a drastic reduction in the number of degrees of freedom retained as compared to the original number of grid points (10^6–10^9). Several techniques have been presented in existing literature for the identification of the parameters of the ROMs approximating the temporal dynamics of the projection coefficients; particularly applicable in experimental investigations where only limited data is available. The identification of ROM parameters from the knowledge of uncorrelated samples of temporal POD coefficients and their time derivatives obtained from an experimental database was performed by Perret et al. (2006). Autoregressive (AR) (Jeong and Bienkiewicz, 1997) or autoregressive moving average (ARMA) (Kho et al., 2002) models have also been used to construct ROMs based on POD analysis of pressure fluctuations. An approach based on using flow visualizations for obtaining ROMs in order to develop a closed-loop control strategy was presented by Park et al. (2004).

In this work, the linear regression approaches, namely of ordinary least squares (OLS) algorithm (Press et al., 1993), sparse identification of nonlinear dynamics (SINDy) algorithm (Brunton, Proctor, et al., 2016a), and least angle regression (LARS) algorithm (Efron, Hastie, et al., 2004), have been applied for the identification of polynomial coefficients of the POD-Galerkin ROM. A bootstrap method (Efron, 1979) is used to quantify the uncertainty associated with these system identification methods.

The intrusive ROM usually suffers with the issue of *instability* (Schlegel and Noack, 2015; Östh et al., 2014). The instability arises due to the truncation of the POD basis used in the development of POD-Galerkin ROM. The truncated modal basis often fails to capture the small-scale dynamics and the predictive ability of the ROMs is thus limited. In order to alleviate

it, high order modes must be taken into account in the POD-Galerkin approach, even if their combined energy content is low, in order to improve the long-term stability and accuracy of the model. Inspired by the successes in RANS and LES methods, empirical dissipation terms can be added to ROMs to account for the high order modes and ensure model stability (Iollo, Lanteri, et al., 2000; Östh et al., 2014). Other methods for ensuring numerical stability have also been proposed in the literature. A stable symmetrical inner product that guarantees certain stability bounds for the linearized compressible Euler equations was proposed by Kalashnikova and Barone (2010). A stabilization method for the POD-Galerkin ROM involving the calculation of POD for both the function and gradient values (POD in H_1) was proposed by Iollo, Dervieux, et al. (2000). A regularization method to replace the POD modes of the nonlinear terms by their Helmholtz filtered counterparts for strongly-stiff systems was proposed by Sabetghadam and Jafarpour (2012). Bond and Daniel (2008) introduced a set of linear constraints for projection matrix to guarantee a stable ROM.

In this work, the POD-Galerkin ROM has been augmented with modal eddy viscosity (Protas et al., 2015) for stabilization. The calibration of the parameters associated with the nonlinear terms in the closure term is performed using data assimilation (Kutz, 2013; Asch et al., 2016). The data assimilation tool provides an optimal weighted mean between the measurement data and forecast obtained from a prescribed model. This can be extended to tune the parameters and improve the predictive ability of the model. In this work, the dual ensemble Kalman filter (Dual-EnKF) method (Moradkhani et al., 2005) has been used in which both the state variables and model parameters are estimated simultaneously for given erroneous forcing data (input) and observations (output). In essence, the Dual-EnKF uses the ensemble of model trajectories in an interactive parameter-state space and provides the confidence interval of the parameter-state estimation. This framework also allows to take into account the multiplicative nature of errors in the forcing data and observation by assembling the parameter adaptation in the state evolution and forecasting system (Young, 2002).

After the identification of the full set of parameters, *i.e.* both the polynomial coefficients and the stabilizing term parameters, of the POD-Galerkin ROM, the ensemble Kalman filter (EnKF) algorithm (Evensen, 1994) is used to assimilate the model output and observations in order to provide dynamical state estimation.

1.3.2 Non-intrusive reduced-order modeling

Unlike the POD-Galerkin ROM, non-intrusive ROMs (NIROMs) require no knowledge of the physical system. One of the motivations for developing NIROMs is that, in most cases, the source code describing the physical model has to be modified in order to generate the reduced order model and these modifications can be complex, especially in legacy codes, or may not be possible if the source code is not available. In this regards, several NIROMs have been proposed in the literature pertaining to various fields. An approach for reduced-order modeling based on *neural networks* was introduced by Noack, Morzyński, et al. (2011). A neural network is a form of machine learning method which is capable of approximating an arbitrary function using observed data. The neural network based reduced-order models has been applied in several fluid dynamics applications (Peña et al., 2012; Casenave et al., 2015; Hesthaven and Ubbiali, 2018; Pawar et al., 2019; Wang, Hesthaven, et al., 2019). A neural network derived from simulation of a cylinder wake flow was used by Gillies (1998) for controlling a self-excited cylinder wake oscillations. This empirical model of the modal response of the wake to external forcing was then used to design a closed-loop control algorithm. A non-intrusive POD-ROM for aerodynamic shape optimization was developed by Iuliano and Quagliarella (2013). A NIROM framework based on POD and radial basis function (RBF) and the discrete empirical

interpolation method (DEIM) algorithm using artificial neural network (ANN) was proposed by Winter and Breitsamter (2014) and Guenot et al. (2013). A kernel method based on both support vector machines (SVMs) and a vectorial kernel greedy algorithm was proposed by Wirtz and Haasdonk (2012). A NIROM based on constrained POD (CPOD) and Kriging interpolation method was proposed by Xiao et al. (2010). A non-intrusive method for the polynomial chaos representation to extract a set of optimal basis functions from a coarser mesh and use it to perform finer mesh analysis was presented by Raisee et al. (2015).

In this work, in order to bypass the Galerkin projection step in the derivation of POD-ROM, a novel artificial neural network (ANN) framework – named NN-ROM for deep neural network reduced-order model – has been presented in the form of a NIROM method combining two recently presented strategies. The first strategy is based on the parametrized framework presented by Wang, Hesthaven, et al. (2019) in which the DNN approximates the map between the time and parameter values as input and the POD projection coefficients as output. The second strategy is based on the multistep configuration of the DNN presented by Pawar et al. (2019) where the values of the states at previous time steps are used to obtain the time evolution of the POD projection coefficients. Additionally, as this NIROM framework provides sequential estimates of the dynamics, it can be considered as a forward model in the data assimilation paradigm of EnKF while assimilating the estimates and observations.

1.4 Contributions and outline of the thesis

The novel contributions of the thesis are listed here:

1. A comprehensive evaluation of the performance of the linear regression methods used to identify the coefficients of the polynomial terms in the POD-ROM has been undertaken in a probabilistic framework. For this, a bootstrap resampling method has been used to obtain the probability distributions and confidence intervals associated with the identified coefficients.

2. A simultaneous state and parameter estimation in a data assimilation paradigm has been implemented for the estimation of the parameters associated with the closure term in the POD-ROM. This utilizes the Dual-EnKF algorithm which seamlessly integrates the model output and measurements for the data-driven parameter estimation.

3. A neural network based non-intrusive ROM framework has been developed for the time series prediction of transient dynamics of the POD modes. The multistep, residual-based, parameterized neural network framework is augmented with EnKF to provide long-term dynamical predictions with an improved accuracy.

The outline of the manuscript along with chapter-wise keywords is listed in Tab. 1.1 and the schematic of the dynamical modeling and reconstruction approach is shown in Fig. 1.2. As discussed in Sec. 1.2 and Sec. 1.3, the workflow involves a sequence of three main operations. First, the model order reduction is performed using high-fidelity snapshots in the physical space and data-based modal decomposition to extract the reduced basis in the latent space. Next, the reduced-order model in terms of the latent space variables is identified using either an intrusive or non-intrusive approach. Lastly, the identified model is used to obtain future estimates of the latent space variables which, in turn, can be used to provide high-fidelity reconstructions in the physical space.

The contents of the subsequent chapters in the manuscript are briefly presented here. In Chap. 2, the data-based modal decomposition techniques, namely POD and DMD, used to

Table 1.1: Outline of the manuscript.

	Chapter	Keywords
1	Introduction	–
2	Reduced-order modeling	POD, DMD, Galerkin projection, POD-ROM, eddy viscosity stabilization
3	Data-driven dynamical system identification	OLS, SINDy, LARS, Kalman filter, EnKF, Dual-EnKF, artificial neural network, non-intrusive ROM
4	System identification by linear regression models	OLS, SINDy, LARS, noise filtering, system identification, bootstrap estimation
5	Improving POD-ROM using the Dual Ensemble Kalman filter	SINDy, EnKF, Dual-EnKF, stabilization parameter identification, long-term flow prediction
6	Predicting transient dynamics using non-intrusive ROM	deep neural network, non-intrusive ROM, EnKF, long-term flow prediction
7	Conclusion and future work	–

extract the reduced bases from high-dimensional nonlinear flow data are described. A discussion comparing the applicability of both the techniques for model reduction is also presented and the POD-based Galerkin projection is selected for subsequent development of the ROM. The derivation is performed by the projection of the incompressible Navier-Stokes equations on the modes resulting from POD. An eddy viscosity model is introduced for closure and stabilization of the ROM.

In Chap. 3, we introduce different methods considered in this manuscript for the identification of flow dynamics from data. The methods used for data-driven identification of the POD-based ROM are classified into two categories as interpretable, closed-form *grey-box* functions or surrogate *black-box* functions. For the grey-box approach, different methods, namely OLS, SINDy, and LARS, are introduced as regression tools for the identification of the quadratic dynamical evolution component of the POD-ROM. For the nonlinear parameters in the POD-ROM, data assimilation algorithms, in general, and Dual-EnKF, in particular, are introduced. For the black-box approach, a non-intrusive ROM framework based on deep neural network, which is capable of providing iterative predictions of the dynamical component of POD, is presented. This approach is formulated as a combination of two strategies. The first strategy approximates the map between the time and parameter values as inputs and the POD coefficients as outputs. The second strategy uses the values of the states at previous time steps to predict the time evolution of the residuals of the solution.

In Chap. 4, we use linear regression methods such as OLS, SINDy, and LARS to identify the dynamics of the systems. We also show that a bootstrap method can be used to evaluate the dependence of the identified parameters on the snapshots retained in the database. This probabilistic framework is developed for toy models which mimic the POD-ROM without the nonlinear residual term. Different preprocessing tools to handle noisy data and assemble matrices in the linear system are discussed. The probability distributions and confidence intervals associated with the learned parameters are obtained using the circular block bootstrap resampling technique.

In Chap. 5, an eddy viscosity closure term is incorporated in the POD-ROM for a proper description of the energy dissipation mechanism and the Dual-EnKF algorithm is used to identify the corresponding nonlinear parameters. The data assimilation framework is demonstrated for the identification of Lorenz-63 system and a simulated flow around a circular cylinder at a low Reynolds number. The application is also extended to experimental cylinder wake flow at higher Reynold numbers and a Mach 0.9 turbulent jet. The sequential data assimilation

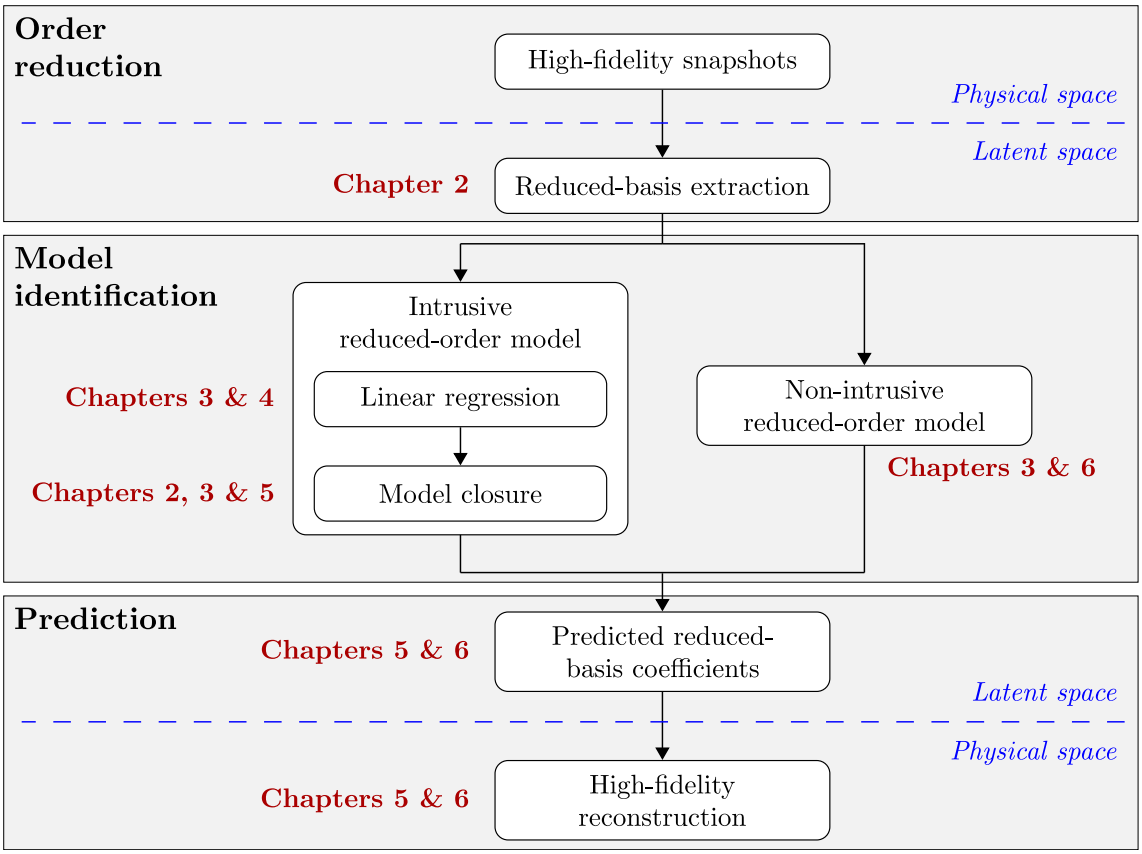


Figure 1.2: Schematic of dynamical modeling and reconstruction approach. The chapters corresponding to the entities are also indicated.

technique is used to recover the dynamics of the full state in a long time horizon.

In Chap. 6, the limitations of the POD-ROM in terms of satisfying the linearity assumption and requirement of an *a priori* knowledge of the model are addressed by using a neural network based non-intrusive ROM (NN-ROM). The consideration of memory effect by the NN-ROM is demonstrated for Lorenz-63 system. Next, the NN-ROM is constructed on a parameter space to extrapolate the full-scale dynamics of a numerical cylinder wake flow at low Reynolds number. Lastly, the sequential NN-ROM is combined with the EnKF data assimilation algorithm to enable long-term predictions of the dynamics of a cylinder wake flow at high Reynolds number.

Finally, Chap. 7 summarizes the most important results and proposes some perspectives for future work.

Chapter 2

Reduced-order modeling

Contents

This chapter discusses some of the most widely used model reduction techniques used in the field of fluid mechanics which will serve as a foundation for the development of dynamical estimation approaches in the subsequent chapters. In Sec. 2.1, two modal analysis methods, namely proper orthogonal decomposition (POD) and dynamic mode decomposition (DMD) are described. The modal decomposition methods provide a reduced basis for the representation of high-dimensional nonlinear flow systems. The applicability of both the methods for model reduction is also compared. A POD-based method, which has been proven to be one of the most efficient model reduction techniques, is considered next for dimensionality reduction. In Sec. 2.2, a dynamic reduced model formulated by Galerkin projection of the incompressible Navier-Stokes equations on to the modes resulting from POD is discussed. To tackle the issue of stability associated with this method, a model closure is also introduced.

2.1 Modal analysis

Fluid flow applications involve a wide range of strongly interacting spatial and temporal scales arising from instabilities, nonlinearities, and turbulence. In the past few decades, rapid advancements in numerical simulation and experimental measurement techniques have led to the availability of large-scale high-fidelity flow field data. In order to study the fluid flows and develop models to capture their dynamical behavior, it is imperative to compress the vast

amount of data to a low-dimensional form. The analysis of these flows is performed by extracting physically important features, or *modes*, from high-resolution spatio-temporal data that capture the intricate physics.

Analysis of flow data has shown that the high-dimensional dynamics are composed of low-dimensional features that are commonly shared by a wide range of flows (Taira, Hemati, et al., 2020). In other words, there exist key underlying spatial features that represent the most important aspects of the flow physics. The identification of an effective low-dimensional coordinate system, or *reduced basis*, for capturing the energetically and dynamically dominant flow mechanisms is performed by *modal decomposition* (Taira, Brunton, et al., 2017). The decomposition results in spatial features called *modes* and corresponding characteristic values, representing either the energy content levels or growth rates and frequencies.

The modes can be determined either from the flow field data or directly from the governing equations. The former is referred as *data-based* technique, which only requires flow field data as input and does not necessitate an *a priori* knowledge of the flow dynamics. The latter, referred as *operator-based* technique, requires a more theoretical framework or a knowledge of the discrete operators of Navier–Stokes equations. In this work, data-based modal decomposition methods has been considered. A method to determine energetically optimal set of modes to represent the data, called proper orthogonal decomposition (POD), is discussed in Sec. 2.1.1. Another data-based method which captures dynamic modes with the associated growth rates and frequencies using a linear approximation to nonlinear dynamics, called dynamic mode decomposition (DMD), is discussed in Sec. 2.1.2. A justification to opt for POD for reduced-order modeling in the subsequent developments in this thesis is also given towards the end of the section.

2.1.1 Proper Orthogonal Decomposition (POD)

POD has been introduced in the fluid mechanics community by Lumley (1967) in the late 1960s as a mean to identify, in a deterministic manner, the large-scale structures present in turbulent flows. POD is a data-based method and therefore only requires the flow field data, which can be associated with linear or nonlinear dynamics. POD extracts modes which are optimal in terms of the mean square of the field variable under observation. In other words, it provides an objective algorithm to decompose a set of data into a minimal number of basis functions which capture as much "energy"[1] as possible (Cordier and Bergmann, 2008a).

Let us consider a field $\boldsymbol{q}(\boldsymbol{\chi}, t)$ which can be composed of either scalar (*e.g.* pressure, temperature) or vectorial (*e.g.* velocity, vorticity) elements. Here, $\boldsymbol{\chi}$ is a vector representing the relevant spatial coordinates and t is a scalar time. Let $\boldsymbol{q}'(\boldsymbol{\chi}, t)$ be the fluctuating component which is obtained by subtracting the temporal mean $\overline{\boldsymbol{q}}(\boldsymbol{\chi})$ from the field $\boldsymbol{q}$,

$$\boldsymbol{q}'(\boldsymbol{\chi}, t) = \boldsymbol{q}(\boldsymbol{\chi}, t) - \overline{\boldsymbol{q}}(\boldsymbol{\chi}). \tag{2.1}$$

The goal is to decompose the random field $\boldsymbol{q}'(\boldsymbol{\chi}, t)$ into a set of deterministic spatial functions modulated by time coefficients, *i.e.*

$$\boldsymbol{q}'(\boldsymbol{\chi}, t) = \sum_{i=1}^{+\infty} a_i(t)\boldsymbol{\Phi}_i(\boldsymbol{\chi}), \tag{2.2}$$

where, $\boldsymbol{\Phi}_i$ and a_i represent the POD (spatial) *modes* and the *modal coefficients*, respectively.

[1] Energy is defined in terms of a given inner product, see App. B.8 for the definition of an inner product and (2.3) for the one used.

The decomposition (2.2) can be performed using different spectral methods, *e.g.* using the standard Fourier decomposition. In the POD framework, we seek the "proper" or optimal basis functions $\boldsymbol{\Phi}_i$ such that the projection of the field $\boldsymbol{q}'$ on the first K spatial function $\boldsymbol{\Phi}_i$ is maximized[2] on average, irrespective of the value of K. In order to rigorously define the projection, we introduce an inner product and its induced norm. Let $\boldsymbol{q}^{\mathrm{I}}$ and $\boldsymbol{q}^{\mathrm{II}}$ be two given fields defined in a spatial domain, the spatial inner product is defined as

$$\left\langle \boldsymbol{q}^{\mathrm{I}}(\boldsymbol{\chi},t), \boldsymbol{q}^{\mathrm{II}}(\boldsymbol{\chi},t) \right\rangle_{\Omega} := \int_{\Omega} \boldsymbol{q}^{\mathrm{I}}(\boldsymbol{\chi},t) \cdot \boldsymbol{q}^{\mathrm{II}}(\boldsymbol{\chi},t)\, \mathrm{d}\boldsymbol{\chi}, \tag{2.3}$$

where the dot represents the Euclidean inner product. The induced norm is $\left\|\boldsymbol{q}^{\mathrm{I}}\right\|_{\Omega} = \sqrt{\langle \boldsymbol{q}^{\mathrm{I}}, \boldsymbol{q}^{\mathrm{I}} \rangle_{\Omega}}$.

We now assume that the field $\boldsymbol{q}'$ is defined on discrete times t_k ($k = 1, \ldots, N_t$) where N_t is the number of temporal snapshots. The corresponding POD maximization problem is given as

$$\max_{\{\boldsymbol{\Phi}_i\}_{i=1}^{K}} \sum_{j=1}^{N_t} \left\| \Pi_{\mathsf{POD}} \boldsymbol{q}'(\boldsymbol{\chi}, t_j) \right\|_{\Omega}^2, \tag{2.4a}$$

subject to

$$\|\boldsymbol{\Phi}_k\|_{\Omega}^2 = 1 \quad k = 1, \cdots, K, \tag{2.4b}$$

where Π_{POD} is the orthogonal projector on the space spanned by the first K functions $\boldsymbol{\Phi}_i$, *i.e.*

$$\Pi_{\mathsf{POD}} \boldsymbol{q}'(\boldsymbol{\chi}, t_j) = \sum_{k=1}^{K} \left\langle \boldsymbol{q}'(\boldsymbol{\chi}, t_j), \boldsymbol{\Phi}_k(\boldsymbol{\chi}) \right\rangle_{\Omega} \boldsymbol{\Phi}_k(\boldsymbol{\chi}).$$

Since Π_{POD} is an orthogonal projector, the maximization problem (2.4) is equivalent to the minimization problem given by

$$\min_{\{\boldsymbol{\Phi}_i\}_{i=1}^{K}} \sum_{j=1}^{N_t} \left\| \boldsymbol{q}'(\boldsymbol{\chi}, t_j) - \Pi_{\mathsf{POD}} \boldsymbol{q}'(\boldsymbol{\chi}, t_j) \right\|_{\Omega}^2. \tag{2.5}$$

This means that for any subspace of size K, POD optimizes the averaged residual with respect to the norm (2.3). It can be shown (see Cordier and Bergmann, 2008a, for instance) that the POD modes are energetically optimal for the norm (2.3). In that sense, POD allows the representation of a signal using a minimum number of modes, opening the opportunity of reduced-order modeling. The constrained maximization problem (2.4) can be solved with classical Lagrange multipliers (Volkwein, 2013). Hereafter, we introduce the solution in terms of correlation matrix.

The data obtained from experiments or numerical simulations are always finite-dimensional. Consequently, the field $\boldsymbol{q}'(\boldsymbol{\chi}, t_j)$ is considered to be defined at discrete spatial points $\boldsymbol{\chi}_j$ ($j = 1, \ldots, N_\chi$) where N_χ is the number of spatial grid points. The fluctuating component of the field in (2.1) can be re-written in the discrete space as

$$\boldsymbol{q}'(\boldsymbol{\chi}_j, t_k) = \boldsymbol{q}(\boldsymbol{\chi}_j, t_k) - \overline{\boldsymbol{q}}(\boldsymbol{\chi}_j), \quad j = 1, \ldots, N_\chi, \quad k = 1, \ldots, N_t. \tag{2.6}$$

Here, $\boldsymbol{q}'(\boldsymbol{\chi}_j, t_k) \in \mathbb{R}^{N_c \times 1}$, where N_c is the number of components of the input data (*e.g.* $N_c =$

[2] This maximization problem is directly linked to the Eckart-Young theorem (see App. B.13.6).

2 for a two-dimensional velocity field, $N_c = 1$ for a pressure field). Let's denote the individual component of $\boldsymbol{q}'$ as q'_m with $m = 1, \ldots, N_c$. We also introduce $N_s = N_\chi \times N_c$ as the number of spatial points saved per time snapshot.

The ensemble of snapshots $\boldsymbol{q}'(\boldsymbol{\chi}, t_k) \in \mathbb{R}^{N_s \times 1}$ is summarized into a snapshot matrix $\boldsymbol{X}$ as

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{q}'(\boldsymbol{\chi}, t_1) & \boldsymbol{q}'(\boldsymbol{\chi}, t_2) & \cdots & \boldsymbol{q}'(\boldsymbol{\chi}, t_{N_t}) \end{bmatrix} \in \mathbb{R}^{N_s \times N_t}. \tag{2.7}$$

To visualize the elements of the matrix $\boldsymbol{X}$, we consider an example of a two-component vector field given as $\boldsymbol{q}' = (q'_1, q'_2)$ in a two-dimensional Cartesian grid in the $\boldsymbol{\chi} = (x, y)$ plane. The expanded form of (2.7) can then be written as

$$\boldsymbol{X} = \begin{bmatrix} q'_1(x_1, y_1, t_1) & q'_1(x_1, y_1, t_2) & \ldots & \ldots & q'_1(x_1, y_1, t_{N_t}) \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ q'_1(x_{N_x}, y_{N_y}, t_1) & q'_1(x_{N_x}, y_{N_y}, t_2) & \ldots & \ldots & q'_1(x_{N_x}, y_{N_y}, t_{N_t}) \\ q'_2(x_1, y_1, t_1) & q'_2(x_1, y_1, t_2) & \ldots & \ldots & q'_2(x_1, y_1, t_{N_t}) \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ q'_2(x_{N_x}, y_{N_y}, t_1) & q'_2(x_{N_x}, y_{N_y}, t_2) & \ldots & \ldots & q'_2(x_{N_x}, y_{N_y}, t_{N_t}) \end{bmatrix}, \tag{2.8}$$

where N_x and N_y are the number of grid points in the example along the x and y directions, respectively. The total number of spatial grid points is given as $N_\chi = N_x \times N_y$. Consequently, the number of rows in the snapshot matrix for vector field with two components ($N_c = 2$) is $N_s = N_\chi \times N_c = 2N_\chi$. If the vector field has more than two components, the data is stacked as additional rows in the matrix $\boldsymbol{X}$, such that the total number of rows becomes $3N_\chi$ for a three-component vector field, and so on.

In addition to some randomness, turbulence is also known to be partially driven by some coherence and order. This order is observed through the *spatial correlation* between the time series $\boldsymbol{q}'(\boldsymbol{\chi}_j, t)$ for each component of $\boldsymbol{q}'$ at each spatial grid point $\boldsymbol{\chi}_j$. Correlation provides a metric to identify the regions where the flow is synchronized and the velocity fluctuations are correlated. The spatial covariance matrix[3] $\boldsymbol{C}_s$ is computed using the snapshot matrix[4] $\boldsymbol{X}$ as[5]

$$\boldsymbol{C}_s = \frac{1}{N_t - 1} \boldsymbol{X} \boldsymbol{X}^\top \in \mathbb{R}^{N_s \times N_s}, \tag{2.9}$$

[3] An element $(C_s)_{\alpha\beta}$ corresponding to the α-th row and β-th column of $\boldsymbol{C}_s$ is given as

$$(C_s)_{\alpha\beta} = \frac{1}{N_t - 1} \sum_{k=1}^{N_t} q'_{m(\alpha)}(\boldsymbol{\chi}_{i(\alpha)}, t_k) q'_{n(\beta)}(\boldsymbol{\chi}_{j(\beta)}, t_k),$$
$$m(\alpha) = \lceil \alpha / N_\chi \rceil,\ n(\beta) = \lceil \beta / N_\chi \rceil,\ i(\alpha) = \alpha - (m-1)N_\chi,\ j(\beta) = \beta - (n-1)N_\chi,$$

where $\lceil \cdot \rceil$ is the ceiling function. Along the diagonal, we have $\alpha = \beta$ (implying $m = n$ and $i = j$) such that the elements correspond to the variances of $q'_m(\boldsymbol{\chi}_i, t)$ while the off-diagonal terms represent the covariances. The covariance matrix is symmetric.

[4] For simplicity, we have considered the field data to be placed on a uniform grid. For this reason, we do not take into account the quadrature rule to describe the effect of the volume integral present in the spatial inner product (2.3). This is equivalent to replacing the spatial inner product (2.3) with an Euclidean inner product in $\mathbb{R}^{N_s}$. In general, the cell volume needs to be included in the formulation to represent the spatial inner product. The covariance matrix should therefore be written as $\boldsymbol{C}_s = \boldsymbol{X} \boldsymbol{W} \boldsymbol{X}^\top / (N_t - 1)$ where $\boldsymbol{W}$ holds the weights of the spatial quadrature. With our assumption, the orthonormality relation (2.10) is given by $\boldsymbol{\Phi}^\top \boldsymbol{\Phi} = \boldsymbol{I}_{N_s}$ and not $\boldsymbol{\Phi}^\top W \boldsymbol{\Phi} = \boldsymbol{I}_{N_s}$ as expected.

[5] We decide to keep the factor $1/(N_t - 1)$ in the definition of the covariance matrix. We could have also removed the factor and lumped it into the eigenvalue λ_i as in Taira, Brunton, et al. (2017).

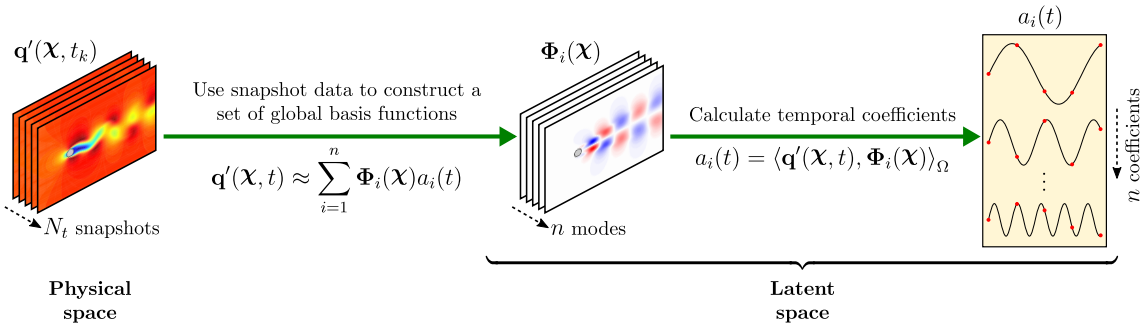


Figure 2.1: Schematic of the classical POD method. A truncated series of $n \leq N_s$ modes is depicted.

where $\square^\top$ represents the matrix transpose.

It can be shown that the optimal basis vectors $\boldsymbol{\Phi}_i$ are obtained from the eigenvectors of the covariance matrix $\boldsymbol{C}_s$ (Lumley, 1967). This can be understood intuitively: in the proper orthogonal basis, the variance along each axis (basis vector) is maximized and, as a consequence, the covariance between the axes (given by the off-diagonal terms of $\boldsymbol{C}_s$) should ideally be zero. This means that the otherwise dense covariance matrix $\boldsymbol{C}_s$ reduces to a $N_s \times N_s$ diagonal matrix. Now, recalling that $\boldsymbol{C}_s$ is symmetric, we can conclude that its eigenvectors form an orthonormal basis in which the matrix $\boldsymbol{C}_s$ is *orthogonally diagonalizable* (see App. B.11 for normal matrices). Therefore, the optimized basis to express the data is simply the set of eigenvectors of the covariance matrix. The POD modes are then orthonormal, *i.e.* the basis functions satisfy

$$\langle \boldsymbol{\Phi}_i(\boldsymbol{\chi}), \boldsymbol{\Phi}_j(\boldsymbol{\chi}) \rangle_\Omega = \delta_{ij} = \begin{cases} 1, & \text{if } i = j\,; \\ 0, & \text{if } i \neq j. \end{cases} \tag{2.10}$$

The fact that the modes are orthonormal allows the modal coefficients a_i to be solely dependent on the modes $\boldsymbol{\Phi}_i$. This becomes evident by taking the Euclidean inner product of both sides of (2.2) with $\boldsymbol{\Phi}_i$, and integrating over the spatial domain. We finally get an expression for a_i as

$$a_i(t) = \left\langle \boldsymbol{q}'(\boldsymbol{\chi}, t), \boldsymbol{\Phi}_i(\boldsymbol{\chi}) \right\rangle_\Omega, \tag{2.11}$$

where the orthonormality property of the modes (2.10) is used. Formally, the covariance matrix $\boldsymbol{C}_s$ is diagonalized as[6]

$$\boldsymbol{C}_s = \boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{\Phi}^{-1} = \boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{\Phi}^\top, \tag{2.12}$$

where the columns of $\boldsymbol{\Phi} \in \mathbb{R}^{N_s \times N_s}$ are the eigenvectors of $\boldsymbol{C}_s$, *i.e.*

$$\boldsymbol{\Phi} = \begin{bmatrix} | & | & & | \\ \boldsymbol{\Phi}_1 & \boldsymbol{\Phi}_2 & \dots & \boldsymbol{\Phi}_{N_s} \\ | & | & & | \end{bmatrix}. \tag{2.13}$$

In the second equality of (2.12), we use the fact that as $\boldsymbol{C}_s$ is symmetric, the eigenvectors are orthonormal, which makes $\boldsymbol{\Phi}$ an orthogonal matrix such that $\boldsymbol{\Phi}^{-1} = \boldsymbol{\Phi}^\top$. The matrix $\boldsymbol{\Lambda} = \mathrm{Diag}\,(\lambda_1, \dots, \lambda_{N_s})$ is assembled such that an eigenvalue λ_i corresponds to an eigenvector given by the i-th column of $\boldsymbol{\Phi}$. By convention, the eigenvalues are arranged in descending

[6] The links between eigenvalue decomposition and the Singular Value Decomposition (SVD) are discussed in App. B.13.5 or in Taira, Brunton, et al. (2017, Sec. II).

order such that $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_{N_s} \geq 0$. In terms of practical model reduction, it is desirable that this decrease is sufficiently fast in order to have the possibility to conserve only a few modes.

As described by (2.2), the basic idea of the POD is to decompose the original fluctuations into a sum of contributions from the individual modes. As a consequence, the original dataset matrix $\boldsymbol{X}$ of the fluctuating field can be represented as a sum of projected data $\boldsymbol{A}$ along the proper orthogonal basis $\boldsymbol{\Phi}$. We therefore get $\boldsymbol{X} = \boldsymbol{\Phi}\boldsymbol{A}$, where

$$\boldsymbol{A} = \begin{bmatrix} a_1(t_1) & a_1(t_2) & \ldots & a_1(t_{N_t}) \\ a_2(t_1) & a_2(t_2) & \ldots & a_2(t_{N_t}) \\ \vdots & \vdots & \ddots & \vdots \\ a_{N_s}(t_1) & a_{N_s}(t_2) & \ldots & a_{N_s}(t_{N_t}) \end{bmatrix} \in \mathbb{R}^{N_s \times N_t}. \tag{2.14}$$

The rows of the projected data matrix[7] $\boldsymbol{A}$ are the temporal coefficients $a_i(t)$ corresponding to a mode $\boldsymbol{\Phi}_i$ represented by the i-th column of $\boldsymbol{\Phi}$. Consequently, in the finite-dimensional space, the columns of $\boldsymbol{X}$ representing the snapshot of the field at each time instant can be expressed as

$$\boldsymbol{q}'(\boldsymbol{\chi}, t) \approx \sum_{i=1}^{N_s} \boldsymbol{\Phi}_i(\boldsymbol{\chi}) a_i(t), \tag{2.15}$$

where $\boldsymbol{\Phi}_i(\boldsymbol{\chi}) \in \mathbb{R}^{N_s \times 1}$ is the basis vector corresponding to the i-th mode (i-th column of $\boldsymbol{\Phi}$) and $a_i(t) \in \mathbb{R}^{1 \times N_t}$ is the corresponding projection coefficient (i-th row of $\boldsymbol{A}$). A discrete counterpart to the continuous form (2.11) can be formulated for obtaining the projection coefficients a_i. The original dataset can be projected on the modes $\boldsymbol{\Phi}$ as[8]

$$\boldsymbol{A} = \boldsymbol{\Phi}^\top \boldsymbol{X}. \tag{2.16}$$

An element A_{ik} of the matrix $\boldsymbol{A}$, representing the projection of the data measured at time t_k on mode Φ_i, is given as

$$A_{ik} = a_i(t_k) = \boldsymbol{\Phi}_i(\boldsymbol{\chi})^\top \boldsymbol{q}'(\boldsymbol{\chi}, t_k) \tag{2.17}$$

$$= \sum_{j=1}^{N_s} \left(\boldsymbol{\Phi}_i\right)_{m(j)} \left(\boldsymbol{\chi}_{n(j)}\right) q'_{m(j)}(\boldsymbol{\chi}_{n(j)}, t_k), \quad \text{where} \quad m(j) = \lceil j/N_\chi \rceil. \tag{2.18}$$

Here $m(j) \in [1, N_c]$ indicates the component of the fluctuation vector, and $n(j) = j - (m(j) - 1) N_\chi \in [1, N_\chi]$ is the spatial coordinate index. By extension of the notations used for the components of the fluctuations $\boldsymbol{q}'$, we denote the m-th component of $\boldsymbol{\Phi}_i(\boldsymbol{\chi})$ as $\left(\boldsymbol{\Phi}_i\right)_m(\boldsymbol{\chi})$ with $m = 1, \ldots, N_c$. A schematic of the POD method is shown in Fig. 2.1.

The importance of the POD basis can be understood from the fact that the eigenvectors give an idea of how the fluctuations $\boldsymbol{q}'$ are correlated. In a proper orthogonal basis, the correlation is not given by the off-diagonal terms of the covariance matrix as this matrix is

[7] An important property of the matrix of projected data $\boldsymbol{A}$ is that its covariance matrix $\boldsymbol{C}_A$ is given by the eigenvalue matrix $\boldsymbol{\Lambda}$. It can be proved as

$$\boldsymbol{C}_A = \frac{1}{N_t - 1} \boldsymbol{A}\boldsymbol{A}^\top = \frac{1}{N_t - 1} (\boldsymbol{\Phi}^\top \boldsymbol{X})(\boldsymbol{\Phi}^\top \boldsymbol{X})^\top = \frac{1}{N_t - 1} \boldsymbol{\Phi}^\top \boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{\Phi} = \boldsymbol{\Phi}^\top \boldsymbol{C} \boldsymbol{\Phi} = \boldsymbol{\Phi}^\top (\boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{\Phi}^\top)\boldsymbol{\Phi} = \boldsymbol{\Lambda}.$$

[8] This decomposition can be obtained by multiplying both sides of $\boldsymbol{X} = \boldsymbol{\Phi}\boldsymbol{A}$ with $\boldsymbol{\Phi}^\top$ and using the orthogonality property $\boldsymbol{\Phi}^\top \boldsymbol{\Phi} = \boldsymbol{I}_{N_s}$.

diagonal. The correlation is implicit in the directions of the eigenvectors. The projected coefficients a_i in (2.16) are specifically constructed to be uncorrelated so that each coefficient can be interpreted as a variation of one independent mode of fluctuation. In fluid dynamics, the POD method is applied with the same idea to obtain modes that can be linked to an independent coherent structure responsible for the fluctuating field. But, for turbulent data, the connection between the POD modes and the physical coherent structures is far from trivial.

Furthermore, the eigenvalues rank the correlation with respect to the variance of the data. In fluid dynamics, if the fluctuations $\boldsymbol{q}'$ are obtained from velocity measurements, and if the inner product (2.3) is used, this corresponds to a ranking based on the turbulent kinetic energy (TKE) of the velocity fluctuations. The individual eigenvalues λ_i therefore represent the TKE associated with the i-th mode and are ordered in the matrix $\boldsymbol{\Lambda}$ based on their contribution to the total TKE. The total TKE is given as $\frac{1}{2}\sum_{i=1}^{N_s}\lambda_i$. Based on this relationship between the eigenvalues and TKE, we introduce an energetic criterion to determine the number of modes that are necessary to sufficiently capture the flow dynamics. The criterion, known as *relative information content* or RIC (Bergmann and Cordier, 2008b), gives the indication of the fraction of total energy that is represented by an ensemble of first n modes. By definition, we have

$$RIC(n) = \sum_{i=1}^{n}\lambda_i \Bigg/ \sum_{i=1}^{N_s}\lambda_i \quad \forall n \leq N_s, \tag{2.19}$$

where $RIC(n) \in (0, 1]$.

2.1.1.1 Snapshot POD

So far in the discussion, we have considered the decomposition into deterministic spatial modes and stochastic time coefficients. However, the decomposition can also be done into deterministic temporal modes and stochastic spatial coefficients. This is possible as the POD equation (2.2) is symmetric in the variables t and $\boldsymbol{\chi}$. An alternate method to evaluate POD modes, called *snapshot POD*, was first introduced by Sirovich (1987). The two POD problems can be linked easily (see App. B.13.5).

The method begins with the same snapshot matrix $\boldsymbol{X} \in \mathbb{R}^{N_s \times N_t}$ defined in (2.8) and consists of constructing a temporal correlation matrix as

$$\boldsymbol{C}_t = \frac{1}{N_s - 1}\boldsymbol{X}^\top \boldsymbol{X} \in \mathbb{R}^{N_t \times N_t}. \tag{2.20}$$

The eigendecomposition of $\boldsymbol{C}_t$ gives the eigenvectors[9] $\boldsymbol{A}_t \in \mathbb{R}^{N_t \times N_t}$ and eigenvalues[10] $\boldsymbol{\Lambda} \in \mathbb{R}^{N_t \times N_t}$ as

$$\boldsymbol{C}_t = \boldsymbol{A}_t \boldsymbol{\Lambda} \boldsymbol{A}_t^{-1}. \tag{2.21}$$

The spatial coefficients $\boldsymbol{\Phi}_t$ can be obtained by projecting the fluctuation data matrix $\boldsymbol{X}$ on the temporal basis $\boldsymbol{A}_t$ as

$$\boldsymbol{\Phi}_t = \boldsymbol{X}\boldsymbol{A}_t^\top \in \mathbb{R}^{N_s \times N_t}. \tag{2.22}$$

[9] Since $\boldsymbol{C}_t$ is symmetric, $\boldsymbol{A}_t$ is orthogonal, *i.e.* $\boldsymbol{A}_t^{-1} = \boldsymbol{A}_t^\top$ (see App. B.11).

[10] By considering the SVD of $\boldsymbol{X}$, it can be shown easily (see App. B.13.5) that the matrices $\boldsymbol{X}^\top\boldsymbol{X}$ and $\boldsymbol{X}\boldsymbol{X}^\top$ share the same eigenvalues. Therefore, we use the same notation $\boldsymbol{\Lambda}$ for the eigenvalues of $\boldsymbol{C}_t$. We remind that the matrix $\boldsymbol{\Lambda} = \mathrm{Diag}\,(\lambda_1, \ldots, \lambda_{N_t})$ is assembled such that an eigenvalue λ_i corresponds to a *temporal mode* given by the i-th column of $\boldsymbol{A}_t$. By convention, the eigenvalues are arranged in descending order such that $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_{N_t} \geq 0$.

The matrix $\boldsymbol{\Phi}_t$ contains the N_t spatial coefficients (equivalent to the spatial modes of the direct method), ordered energetically along the columns.

The spatial coefficients and temporal modes of the snapshot POD obtained so far differ from the spatial modes and temporal coefficients of the direct method by a multiplicative factor. The spatial modes $\boldsymbol{\Phi}$ in the direct method are orthonormal as they originate from the eigendecomposition of the symmetric correlation matrix $\boldsymbol{C}_s$. On the other hand, the spatial modes $\boldsymbol{\Phi}_t$ in the snapshot POD are not orthonormal as $\boldsymbol{\Phi}_t$ is not obtained directly from eigendecomposition but derived through projection (2.22). In order to match the results obtained from both the methods, the spatial coefficients of the snapshot POD are normalized as

$$\boldsymbol{\Phi}_i = \frac{1}{\sqrt{N_s-1}}\frac{1}{\sqrt{\lambda_i}}\boldsymbol{X}\boldsymbol{A}_{t,i} \in \mathbb{R}^{N_s\times 1}, \quad i = 1,\ldots,N_t, \tag{2.23}$$

which gives the columns of $\boldsymbol{\Phi}$, written in matrix form as[11]

$$\boldsymbol{\Phi} = \frac{1}{\sqrt{N_s-1}}\boldsymbol{X}\boldsymbol{A}_t\boldsymbol{\Lambda}^{-1/2} \in \mathbb{R}^{N_s\times N_t}. \tag{2.24}$$

This way, the matrix $\boldsymbol{\Phi}$ is equivalent to the spatial modes of the direct method[12]. Finally, the temporal coefficients of the direct method are obtained by using (2.24), *i.e.*

$$\boldsymbol{A} = \boldsymbol{\Phi}^\top\boldsymbol{X} \in \mathbb{R}^{N_t\times N_t}. \tag{2.25}$$

In most practical cases, especially in fluid dynamics, the data involves planar or volumetric measurements of quantities like velocity and pressure. The number of measurement points in such cases are very large as compared to the number of snapshots, *i.e.* $N_s \gg N_t$. This makes the correlation matrix $\boldsymbol{C}_t$ smaller and much more convenient to store. Subsequently the eigendecomposition is faster to perform (Rowley and Dawson, 2017). The snapshot POD method is therefore preferred in the studies involving PIV and CFD datasets, and the direct method is preferred for measurements involving limited number of single-point probes with high temporal resolution.

Demo 2.1: POD of numerical 2D-cylinder wake flow dataset

The snapshot POD method is applied to a cylinder wake flow at a Reynolds number, based on the diameter, of $Re = 100$. Refer App. A for details regarding the numerical simulation. The snapshot matrix $\boldsymbol{X}$ is constructed from $N_t = 1000$ snapshots, extracted from a post-transient time range of $t \in [150, 250]$ and consisting of $N_c = 2$ components of velocity fluctuations.

The contribution of the first few dominant modes to the TKE is shown in Fig. 2.2. For this type of very simple dynamics, characterized by strong convective phenomena, the POD

[11] The POD basis functions are then represented as linear combinations of the snapshots. Therefore, all the properties of the snapshots that can be written as linear and homogeneous equations pass directly to the POD basis functions. See Sec. 2.2.1.2 for the applications to incompressibility and to the boundary conditions.

[12] Indeed, we have:

$$\boldsymbol{\Phi}^\top\boldsymbol{\Phi} = \frac{1}{N_s-1}\boldsymbol{\Lambda}^{-1}\left(\boldsymbol{X}\boldsymbol{A}_t\right)^\top\left(\boldsymbol{X}\boldsymbol{A}_t\right) = \frac{1}{N_s-1}\boldsymbol{\Lambda}^{-1}\boldsymbol{A}_t^\top\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{A}_t = \boldsymbol{I}_{N_t}.$$

modes clearly appear in pairs. The eigenvalue problem is almost degenerate. It can be seen that the first and second modes dominate the spectrum. The relative energy content of the first ten modes is $RIC(n = 10) = 0.9999$. Therefore, we can expect that these modes can accurately reproduce the flow dynamics (at least in the energetic sense), consequently forming a low-dimensional basis. The low-dimensionality can be attributed to the simplicity of the coherent vortical structures that are even visually identifiable in the snapshots. In the case of high Reynolds number, a higher number of modes will be required to have a sufficiently good approximation of the TKE. In addition, the interpretation of the modes may be more difficult. Nonetheless, as we have $n \ll N_t$, this example offers an insight into the huge computational advantage that we obtain with the dimensionality reduction using POD modal decomposition.

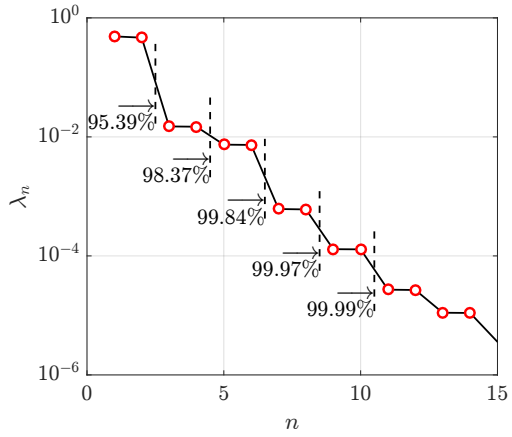


Figure 2.2: Eigenvalues spectrum of the POD modes. The percentage $RIC(n)$ corresponding to the fraction of the total fluctuating kinetic energy captured by the first n POD modes is also shown.

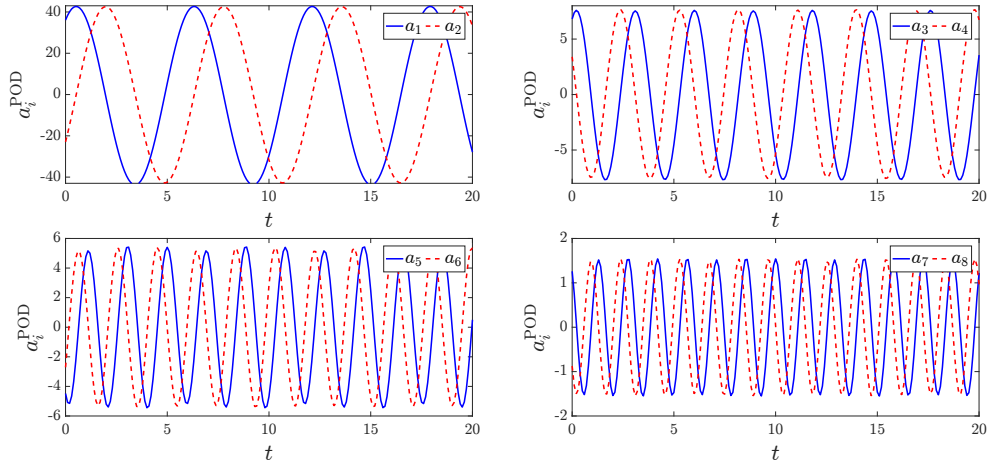


Figure 2.3: Time evolution of the temporal POD coefficients a_i^{POD} for $i = 1, \ldots, 8$ corresponding to the most energetic modes for the 2D-cylinder wake flow at $Re = 100$.

To distinctly identify the modal decomposition performed using the POD method, we introduce the "POD" superscript. The most dominant temporal coefficients are plotted in Fig. 2.3. The temporal modes still appear in pairs of same amplitude, one being shifted in time from the other. The amplitude of the temporal coefficients a_i^{POD} for the first pair (a_1^{POD}, a_2^{POD}) is greater by an order of magnitude when compared with the next pair of

modes (a_3^{POD}, a_4^{POD}). For the subsequent pairs of modes, the amplitude keeps reducing subsequently, consistent with the energy spectrum shown in Fig. 2.2. We also clearly observe that the temporal frequency of the modes increases with the POD index.

The spatial modes are shown in Fig. 2.4. As the cylinder wake is dominated by vortical features, structures resembling the vortices appear in the spatial modes as well. The spatial mode pairs are shifted spatially as a result of the convective nature of the flow. The first spatial mode pair ($\mathbf{\Phi}_1^{\mathrm{POD}}$ and $\mathbf{\Phi}_2^{\mathrm{POD}}$) depicts the dynamical vortex shedding and their downstream convection. The subsequent modes ($\mathbf{\Phi}_3^{\mathrm{POD}}$ and $\mathbf{\Phi}_4^{\mathrm{POD}}$) correspond to smaller scale structures which are attributed to the manifestation of the separated shear layers along the sides of the cylinder and their longitudinal expansion further downstream.

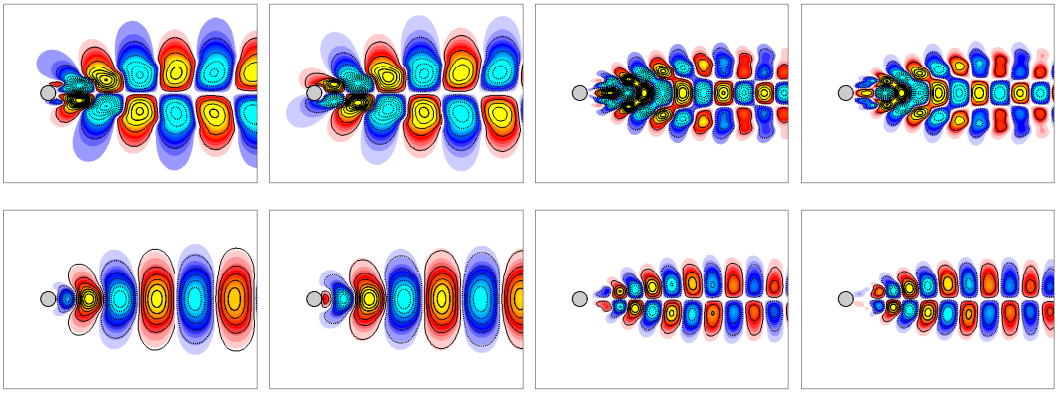


Figure 2.4: Spatial POD modes $\mathbf{\Phi}_i^{\mathrm{POD}}$, $i = 1, 2, 3, 4$ (from left to right) for the streamwise (top) and vertical (bottom) velocity fluctuations for the 2D-cylinder wake flow at $Re = 100$.

2.1.1.2 Limitations of POD

The POD method was introduced in the field of fluid dynamics with the idea that the modes would be able to directly represent the coherent structures in the flow. However, this is not true in general for most of the cases. Recall from (2.12) that the spatial modes are essentially a measure of the correlation of the state variables, represented in terms of the fluctuating field $\boldsymbol{q}'$, between different spatial grid points. Even though the individual zones of correlation may represent coherent structures, the complete flow is composed of the contribution of several modes (2.2). In the case of turbulent flows, the zones of correlation appear randomly as a manifestation of the randomness associated with turbulence (Weiss, 2019). Therefore, depending on the problem, the modes may capture more than just the coherent structures.

POD is optimal in the energetic sense but is generally not *complete*[13]. This can be explained by the limited resolution accuracy of the modes (Noack, Morzyński, et al., 2011). In fluid flow applications, due to the existence of grid-based discretization of the Navier-Stokes equation, the number of spatial modes is tied to the grid. Increasing the resolution accuracy requires the increase of grid resolution which is not always feasible. This also results from the fact that POD is based on a second-order correlation (2.9) and that the higher-order correlations are ignored. Moreover, the calculated modes have in general very restricted applicability outside the parameter space (*e.g.* Reynolds number) and configuration (*e.g.* transients, actu-

[13]Mathematically, a set of orthogonal bases $\mathbf{\Phi}_i$ is termed complete in the closed domain Ω if, for every piecewise continuous function $\boldsymbol{q}'$ defined in the domain, the squared error $\|\boldsymbol{q}' - \sum_{i=1}^{n} a_i \mathbf{\Phi}_i\|_2^2$ converges to zero as $n \to \infty$.

ation) of the considered datasets. These limitations need to be considered when developing methods that rely on the POD method for the modal decomposition of datasets. Such methods have been proposed in Bergmann and Cordier (2008a) for flow control applications. POD mode interpolation methods on Grassman manifolds have been developed in Amsallem and Farhat (2011).

Despite these few limitations, the POD optimality may be useful for constructing reduced-order models (ROM) of fluid dynamics. In Sec. 2.2, POD modes will be used to construct Galerkin projection based ROM for incompressible fluid flows.

2.1.2 Dynamic Mode Decomposition (DMD)

DMD is another data-based method for modal decomposition. The DMD originated from the work of Schmid (2010) in the fluid dynamics community as a method to decompose flows into modes that are representative of the dynamics. DMD modes are spatially correlated and are associated with a given frequency. This is in contrast to the POD modes which are based on energetic optimality of the data and are therefore not distinctly defined in terms of spectral properties.

The method relies on the time-resolved snapshots $\boldsymbol{q}(\boldsymbol{\chi}, t_k) \in \mathbb{R}^{N_s}$, or $\boldsymbol{q}_k$ $(k = 1, \ldots, N_t)$ for simplicity, obtained from a dynamical system[14]. Mathematically, DMD can be thought of as a regression of data on locally linear dynamics. In other words, we search $\boldsymbol{A} \in \mathbb{R}^{N_s \times N_s}$, a linear operator for all pairs of measurements, given as

$$\boldsymbol{q}_{k+1} \approx \boldsymbol{A}\boldsymbol{q}_k. \tag{2.30}$$

The goal of the DMD algorithm is to determine the matrix $\boldsymbol{A}$ that optimally fits the trajectory $\boldsymbol{q}_k$ for $k = 1, \ldots, N_t - 1$ such that the least-square error $\left\|\boldsymbol{q}_{k+1} - \boldsymbol{A}\boldsymbol{q}_k\right\|_2$ is minimized. To arrive at the optimally constructed matrix $\boldsymbol{A}$, we begin with arranging the snapshots into two data matrices given as

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{q}_1 & \boldsymbol{q}_2 & \cdots & \boldsymbol{q}_{N_t-1} \end{bmatrix} \in \mathbb{R}^{N_s \times (N_t-1)}, \tag{2.31}$$

$$\boldsymbol{X}' = \begin{bmatrix} \boldsymbol{q}_2 & \boldsymbol{q}_3 & \cdots & \boldsymbol{q}_{N_t} \end{bmatrix} \in \mathbb{R}^{N_s \times (N_t-1)}. \tag{2.32}$$

[14] Let $\boldsymbol{q}(t)$ be the solution of the continuous dynamical system defined as

$$\frac{d\boldsymbol{q}}{dt} = \boldsymbol{\mathcal{A}}\boldsymbol{q}. \tag{2.26}$$

The solution is given by

$$\boldsymbol{q}(t) = \exp\left(\boldsymbol{\mathcal{A}}t\right)\boldsymbol{q}(0) = \boldsymbol{\Phi}\exp(\boldsymbol{\Omega}t)\boldsymbol{\Phi}^{-1}\boldsymbol{q}(0) = \boldsymbol{\Phi}\exp(\boldsymbol{\Omega}t)\boldsymbol{b}, \tag{2.27}$$

where $\boldsymbol{\Phi}$ and $\boldsymbol{\Omega}$ (diagonal) are the eigenvectors and eigenvalues matrices of the matrix $\boldsymbol{\mathcal{A}}$ (see App. B.12.2). The vector $\boldsymbol{b}$ can be interpreted as the coordinates of $\boldsymbol{q}(0)$ in the eigenvector basis. Given a continuous dynamical system (2.26), it is always possible to define a discrete-time system sampled at every Δt in time:

$$\boldsymbol{q}_{k+1} \approx \boldsymbol{A}\boldsymbol{q}_k, \qquad \text{where} \tag{2.28}$$

$$\boldsymbol{A} = \exp\left(\boldsymbol{\mathcal{A}}\Delta t\right). \tag{2.29}$$

The matrices can be used to re-write the locally linear approximation (2.30) as

$$\boldsymbol{X}' \approx \boldsymbol{A}\boldsymbol{X}. \tag{2.33}$$

The matrix $\boldsymbol{A}$ is obtained as

$$\boldsymbol{A} = \boldsymbol{X}'\boldsymbol{X}^{+}, \tag{2.34}$$

where $\square^{+}$ represents the Moore-Penrose pseudoinverse (see App. B.13). Singular value decomposition (SVD) gives a computationally efficient method to obtain the pseudoinverse (see App. B.14). The SVD of $\boldsymbol{X}$ gives

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathsf{H}} = \begin{bmatrix}\boldsymbol{U}_r & \boldsymbol{U}_{\text{rem}}\end{bmatrix} \begin{bmatrix}\boldsymbol{\Sigma}_r & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\Sigma}_{\text{rem}}\end{bmatrix} \begin{bmatrix}\boldsymbol{V}_r^{\mathsf{H}} \\ \boldsymbol{V}_{\text{rem}}^{\mathsf{H}}\end{bmatrix} \approx \boldsymbol{U}_r\boldsymbol{\Sigma}_r\boldsymbol{V}_r^{\mathsf{H}}, \tag{2.35}$$

where

- $\boldsymbol{U} \in \mathbb{R}^{N_s \times N_s}$ is the matrix with columns of *left-singular vectors*,
- $\boldsymbol{\Sigma} \in \mathbb{R}^{N_s \times (N_t-1)}$ is the diagonal matrix of *singular values*, *i.e.* $\boldsymbol{\Sigma} = \text{Diag}\left(\sigma_1, \cdots, \sigma_p, 0, \cdots, 0\right)$ with $p = \min\left(N_s, N_t - 1\right)$ and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p$,
- $\boldsymbol{V}^{\mathsf{H}} \in \mathbb{R}^{(N_t-1)\times(N_t-1)}$ is the complex conjugate transpose of matrix $\boldsymbol{V}$ with columns of *right-singular vectors*.

The vectors in $\boldsymbol{U}$ are the POD modes (see Sec. 2.1.1). A low dimensional representation of the data matrix can be obtained by choosing a truncation value $r \leq \min(N_s, N_t - 1)$ for the singular values. With truncation, we get $\boldsymbol{U}_r \in \mathbb{R}^{N_s \times r}$, $\boldsymbol{\Sigma}_r \in \mathbb{R}^{r \times r}$, and $\boldsymbol{V}_r^{\mathsf{H}} \in \mathbb{R}^{r \times (N_t-1)}$. The remainder matrices are $\boldsymbol{U}_{\text{rem}} \in \mathbb{R}^{N_s \times (N_s-r)}$, $\boldsymbol{\Sigma}_{\text{rem}} \in \mathbb{R}^{(N_s-r)\times(N_t-r-1)}$, and $\boldsymbol{V}_{\text{rem}}^{\mathsf{H}} \in \mathbb{R}^{(N_t-r-1)\times(N_t-1)}$. This SVD reduction is useful in flows with low-dimensional structures where the number of singular values in $\boldsymbol{\Sigma}$ decreases sharply to zero, leaving just limited number r of dominant modes (see App. B.13.3). We also note that the singular vectors are unitary (orthonormal in the real space, see App. B.9.2), *i.e.* $\boldsymbol{U}^{\mathsf{H}}\boldsymbol{U} = \boldsymbol{I}_{N_s}$ and $\boldsymbol{V}^{\mathsf{H}}\boldsymbol{V} = \boldsymbol{I}_{N_t-1}$.

An approximation $\widehat{\boldsymbol{A}}$ of the matrix $\boldsymbol{A}$ can now be obtained from (2.34) by introducing the pseudoinverse of $\boldsymbol{X}$ determined via the SVD:

$$\boldsymbol{A} \approx \widehat{\boldsymbol{A}} = \boldsymbol{X}'\boldsymbol{V}_r\boldsymbol{\Sigma}_r^{-1}\boldsymbol{U}_r^{\mathsf{H}}, \tag{2.36}$$

where $\widehat{\boldsymbol{A}} \in \mathbb{R}^{N_s \times N_s}$. An eigendecomposition of $\widehat{\boldsymbol{A}}$ gives the dynamic modes and associated eigenvalues of the system. In many cases, the data matrix $\boldsymbol{X}$ is high-dimensional with $N_s \gg 1$. Computationally, this makes it difficult to represent or decompose the matrix $\widehat{\boldsymbol{A}}$ of size $\mathbb{R}^{N_s \times N_s}$. The dimensionality of the problem can be reduced by projecting[15] the matrix $\widehat{\boldsymbol{A}}$ on the low-rank subspace of size r spanned by the POD modes. The most-dominant POD modes are already stored in the columns of $\boldsymbol{U}_r$. Starting from the evolution of the state given by

$$\boldsymbol{q}_{k+1} = \widehat{\boldsymbol{A}}\boldsymbol{q}_k, \tag{2.37}$$

[15] After projection, we obtain the matrix $\boldsymbol{U}_r^{\mathsf{H}}\widehat{\boldsymbol{A}}\boldsymbol{U}_r$ that is *similar* to the matrix $\widehat{\boldsymbol{A}}$ since $\boldsymbol{U}_r$ is unitary. As a consequence of this similarity transformation, $\widehat{\boldsymbol{A}}$ and $\boldsymbol{U}_r^{\mathsf{H}}\widehat{\boldsymbol{A}}\boldsymbol{U}_r$ share the same eigenelements (see App. B.10).

we obtain the state evolution in the projected space ($\boldsymbol{q}_k = \boldsymbol{U}_r \widetilde{\boldsymbol{q}}_k$) as

$$\widetilde{\boldsymbol{q}}_{k+1} = \boldsymbol{U}_r^{\mathsf{H}} \widehat{\boldsymbol{A}} \boldsymbol{U}_r \widetilde{\boldsymbol{q}}_k \tag{2.38}$$

$$= \boldsymbol{U}_r^{\mathsf{H}} \boldsymbol{X}' \boldsymbol{V}_r \boldsymbol{\Sigma}_r^{-1} \widetilde{\boldsymbol{q}}_k = \widetilde{\boldsymbol{A}} \widetilde{\boldsymbol{q}}_k, \tag{2.39}$$

where

$$\widetilde{\boldsymbol{A}} = \boldsymbol{U}_r^{\mathsf{H}} \boldsymbol{X}' \boldsymbol{V}_r \boldsymbol{\Sigma}_r^{-1} \in \mathbb{R}^{r \times r}. \tag{2.40}$$

The most significant modes of the full-dimensional operator $\boldsymbol{A}$ are reconstructed using $\widetilde{\boldsymbol{A}}$. The eigendecomposition of $\widetilde{\boldsymbol{A}}$ is performed as

$$\widetilde{\boldsymbol{A}} \boldsymbol{W} = \boldsymbol{W} \boldsymbol{\Lambda}^{-1}, \tag{2.41}$$

where $\boldsymbol{W} \in \mathbb{R}^{r \times r}$ is a matrix with the i-th column corresponding to the eigenvector $\boldsymbol{w}_i$, and the diagonal matrix $\boldsymbol{\Lambda} \in \mathbb{R}^{r \times r}$ contains the corresponding eigenvalue λ_i.

Finally, the eigenvector $\boldsymbol{\Phi}$ of $\widehat{\boldsymbol{A}}$ (the DMD modes) can be determined from the eigendecomposition (2.41) of $\widetilde{\boldsymbol{A}}$. The dynamic mode corresponding to the i-th column of the matrix $\boldsymbol{\Phi}$ ($i = 1, \cdots, r$) is given as

$$\boldsymbol{\Phi}_i = \boldsymbol{X}' \boldsymbol{V}_r \boldsymbol{\Sigma}_r^{-1} \boldsymbol{w}_i, \quad \text{for } \lambda_i \neq 0. \tag{2.42}$$

These modes are known as *exact DMD modes* because it was proved in Tu et al. (2014) that these are exact eigenvectors of the matrix $\boldsymbol{A}$. This expression differs from the formula

$$\boldsymbol{\Phi}_i = \boldsymbol{U}_r \boldsymbol{w}_i, \tag{2.43}$$

used in Schmid (2010). The expressions (2.42) and (2.43) tend to converge if the matrices $\boldsymbol{X}$ and $\boldsymbol{X}'$ have the same column spaces. The modes given by (2.43) are referred as *projected DMD modes*. If $\lambda_i = 0$ then the expression (2.42) may be used if $\boldsymbol{\Phi}_i = \boldsymbol{X}' \boldsymbol{V}_r \boldsymbol{\Sigma}_r^{-1} \boldsymbol{w}_i \neq 0$. Otherwise, the projected DMD formulation (2.43) $\boldsymbol{\Phi}_i = \boldsymbol{U}_r \boldsymbol{w}_i$ should be used.

The low-rank approximations of the eigenvalues and DMD modes can be used to forecast the future states of the dynamical system. The approximate solution is given as

$$\boldsymbol{q}(t) \approx \sum_{i=1}^{r} \boldsymbol{\Phi}_i \exp(\omega_i t) b_i = \boldsymbol{\Phi} \exp(\boldsymbol{\Omega} t) \boldsymbol{b}, \tag{2.44}$$

where

- $\omega_i = \ln(\lambda_i)/\Delta t$ is the i-th complex eigenvalue of the matrix $\boldsymbol{\mathcal{A}}$ introduced in (2.29). $\boldsymbol{\Omega}$ is diagonal and equal to $\mathrm{Diag}\,(\omega_1, \cdots, \omega_r) \in \mathbb{R}^{r \times r}$. We define the growth rate α_i and frequency β_i for each DMD mode i as

$$\alpha_i = \frac{\ln(\mid \lambda_i \mid)}{\Delta t} \quad \text{and} \quad \beta_i = \frac{\arg(\lambda_i)}{\Delta t}. \tag{2.45}$$

- b_i is the i-th element of the vector $\boldsymbol{b} \in \mathbb{R}^{r \times 1}$ which contains the initial amplitude of all the modes. To obtain $\boldsymbol{b}$, we consider (2.44) at $t = t_1 = 0$ which gives $\boldsymbol{q}_1 = \boldsymbol{\Phi} \boldsymbol{b}$. The initial amplitudes are then given as

$$\boldsymbol{b} = \boldsymbol{\Phi}^{+} \boldsymbol{q}_1, \tag{2.46}$$

  where $\boldsymbol{\Phi}^{+}$ is the Moore-Penrose pseudoinverse as $\boldsymbol{\Phi}$ is generally not a square matrix.

The pseudoinverse gives a best-fit solution $\boldsymbol{b}$ in the least-squares sense.

Demo 2.2: DMD of numerical 2D-cylinder wake flow dataset

The DMD method is applied to the 2D-cylinder wake flow described in App. A. A set of $N_t = 1000$ snapshots, with $N_c = 2$ components of velocity fluctuations, is selected from a post-transient time range of $t \in [150, 250]$. The velocity fields are first transformed into vorticity fields and then arranged in matrices $\boldsymbol{X}$ and $\boldsymbol{X}'$. The DMD modes $\boldsymbol{\Phi}_i$ are obtained from (2.43) along with the associated eigenvalues from (2.41) that determine a low-dimensional dynamical system governing how the mode amplitudes evolve in time. The spectrum of complex eigenvalues is shown in Fig. 2.5. Unlike POD, the temporal mean is not subtracted from the snapshot ensemble. The first mode represents the steady background mode. It is the most energetic mode and typically has eigenvalue $\lambda \approx 1$, with zero imaginary part. The background and 8 harmonic modes are shown in Fig. 2.6. The background mode resembles, but is not exactly the same as, the mean of the data set. For the dataset containing the snapshots from the limit cycle (characterized by a periodic shedding of vortices), the spectrum appears well-ordered; all the modes in Fig. 2.5 lie on the unit circle. Looking at first two harmonics, labeled 2 and 3 in Fig. 2.6, we observe that the first harmonic is symmetric in vorticity about the horizontal axis while the second harmonic is antisymmetric. Thus, the former corresponds to the large-scale convection of fluid structures in the wake, and the latter corresponds to the periodic shedding of vortices from the top and bottom surfaces. The DMD modes look similar to POD modes as POD gives a harmonic decomposition so that the modal amplitudes are approximately sinusoidal in time at harmonic frequencies of the dominant vortex shedding. In fact, in the regime of limit cycle, the POD modes are similar to the complex (oscillatory) DMD modes up to a complex multiplicative factor (Chen, Tu, et al., 2012).

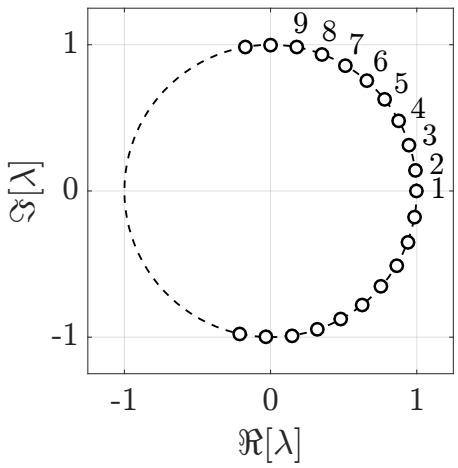


Figure 2.5: Spectral distribution of the complex eigenvalues $\boldsymbol{\Lambda}$ for 2D-cylinder wake flow at $Re = 100$. The dashed line represents a unit circle which indicates the region inside of which the modes decay and outside of which the modes grow.

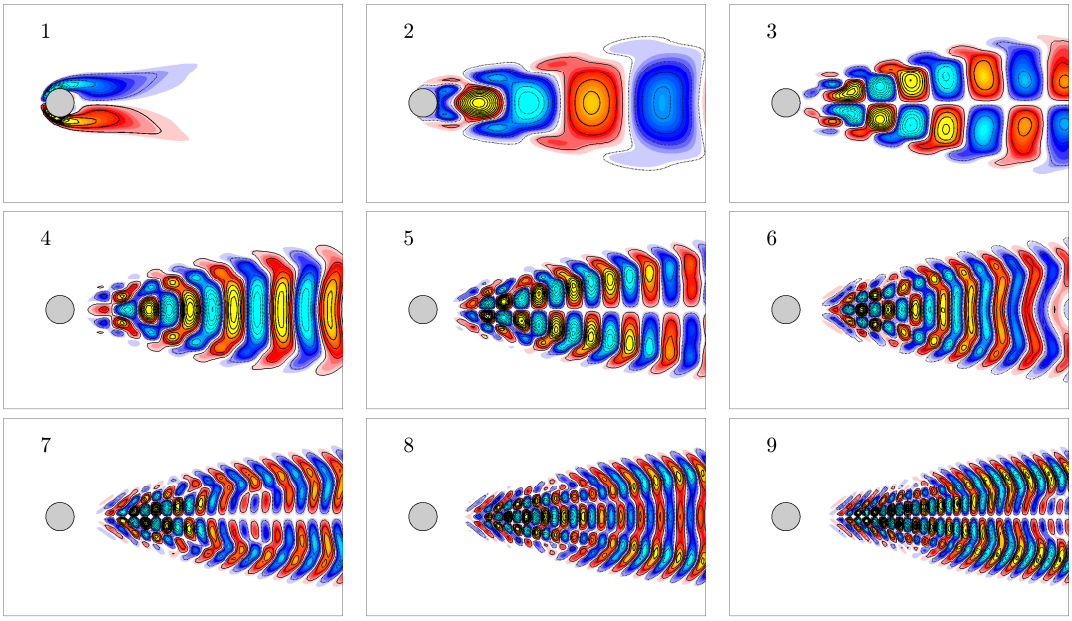


Figure 2.6: Real component of the spatial DMD modes $\Re[\Phi_i^{\mathrm{DMD}}]$ calculated using vorticity fields of 2D-cylinder wake flow at $Re = 100$.

2.1.2.1 Limitations of DMD

The DMD method relies on SVD to extract correlated patterns in the data. As SVD is invariant to *unitary* transformations[16], the SVD-based approaches are sensitive to the alignment of the data and are unable to efficiently handle invariances in the data (Brunton and Kutz, 2019). In other words, translational, rotational or scaling invariances of the low-rank objects embedded in the data are not well captured by DMD. Also, the transient dynamics are not well characterized by DMD (Kutz, Brunton, et al., 2016). A potential strategy to handle the invariances and transient behaviors is offered by multiresolution DMD (Kutz, Fu, et al., 2016).

2.1.2.2 DMD versus POD for reduced-order modeling

So far, two complementary decomposition techniques have been considered: the proper orthogonal decomposition (POD) based on the time-averaged spatial correlation matrix gathered from the snapshots, and the dynamic mode decomposition (DMD) extracting a low-dimensional evolution matrix from the time-resolved data sequence. The POD modes are designed to contain the largest amount of energy with any given number of modes, whereas the DMD modes give the energy of the fluctuations at distinct frequencies. As we have seen, both POD and DMD methods have certain limitations, which can often be mitigated by incorporating some specific strategies. Therefore, the choice boils down to the specific aspects of the given problem (Kutz, 2013).

From the perspective of applications involving the use of modal decomposition methods to construct reduced-order model (ROM), the two approaches have been extensively studied

[16]Unitary transformation is a transformation which preserves the inner product. SVD is fundamentally *geometric* (see App. B.13.4), which implies that, except for the unitary transformation, the results of SVD depend on the transformation of the coordinate system in which the data is represented.

and compared in the literature (see Tissot, 2014, for instance). The POD method preserves the nonlinear dynamics by projecting the governing equations onto the low-dimensional modes, while the DMD method builds the Koopman operator that approximates the nonlinear dynamics without the need of a governing equation. Both methods are able to capture the frequency of pressure fluctuations in flow past square cylinder and cavity (Seid et al., 2012). However, the reconstruction using the DMD method shows a time shift and a more pronounced amplitude deviation in some cases. The discrepancy of DMD to fully describe the fluctuations has been attributed to the fact that while the POD modes oscillate at multiple frequencies, a DMD mode oscillates at a single frequency. The applications on shallow water equations (Bistrian and Navon, 2014) and porous media flow (Bao and Gildin, 2017) show that while the standard DMD-ROM method is less expensive with respect to the numerical implementation costs, the POD-ROM provides higher precision. Therefore, for the subsequent implementation of the system identification methods, we opt for a relatively more precise POD-ROM over the standard DMD-ROM. Some of the limitations of the POD method will be addressed during the construction of the ROM.

2.2 Reduced-order modeling

Formulating a reduced-order model (ROM) is a crucial step in the pursuit of model-based control for fluid flow configurations. This model serves as a way to simplify the Navier-Stokes (N-S) equations into a minimal set of ordinary differential equations (ODEs) which are more manageable. The model-reduction methods are classified as either *projection* or *non-projection* methods (Antoulas, 2005). The first group is most widely used in fluid mechanics and within it, the Galerkin projection approach (Cordier and Bergmann, 2008b) is the most popular. In this work, ROM obtained from projection of the N-S equations onto a POD basis is considered.

2.2.1 Galerkin projection of Navier-Stokes equation

In this section, we present the Galerkin method for incompressible flow. The method approximately solves an initial boundary value problem which is formulated in Sec. 2.2.1.1. The *Galerkin system*, which gives the evolution equation for the mode amplitudes, is then described in Sec. 2.2.1.2.

2.2.1.1 Problem formulation

The incompressible, viscous flow is described in a finite steady domain Ω with Cartesian coordinates $\boldsymbol{\chi} = (x, y, z)$ and time t. The velocity field is represented in terms of its space- and time-dependent components as $\boldsymbol{u}(\boldsymbol{\chi}, t) = (u_x(\boldsymbol{\chi}, t), u_y(\boldsymbol{\chi}, t), u_z(\boldsymbol{\chi}, t))$ and the pressure field is represented as $p(\boldsymbol{\chi}, t)$.

The flow is kinematically characterized by a characteristic length L and a characteristic velocity U. For a Newtonian fluid, density ρ and dynamic viscosity μ are constant. Hereafter, we assume that all the variables have been non-dimensionalized with L, U, ρ and μ. The flow is characterized by the Reynolds number $Re = \rho UL/\mu$, or its reciprocal $\nu := 1/Re$. The flow obeys mass and momentum balance given by the continuity and N-S equation as

$$\nabla \cdot \boldsymbol{u} = 0, \tag{2.47a}$$

$$\partial_t \boldsymbol{u} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} = -\nabla p + \nu \nabla^2 \boldsymbol{u}. \tag{2.47b}$$

As the focus of the current work is on system identification, the external volumetric forces are not considered in the right-hand side of the momentum equation (2.47b). Refer Noack, Morzyński, et al. (2011) for a more general formulation.

Next, an initial condition at $t = 0$ and a boundary condition on the domain boundary $\partial\Omega$ in the time interval $t \in [0, T]$ are defined which allow an unique solution of the initial and boundary value problem (IBVP) (2.47). These conditions are given as

$$\boldsymbol{u}(\boldsymbol{\chi}, 0) = \boldsymbol{u}_{\mathrm{IC}}(\boldsymbol{\chi}) \quad \forall\, \boldsymbol{\chi} \in \Omega, \tag{2.48}$$

$$\boldsymbol{u}(\boldsymbol{\chi}, t) = \boldsymbol{u}_{\mathrm{BC}}(\boldsymbol{\chi}) \quad \forall\, \boldsymbol{\chi} \in \partial\Omega, t \in [0, T]. \tag{2.49}$$

We assume $\boldsymbol{u}_{\mathrm{IC}} = \boldsymbol{u}_{\mathrm{BC}}$ for $\boldsymbol{\chi} \in \partial\Omega$. Pressure is typically considered as a Lagrange multiplier of (2.47a). It can be computed from the velocity field by taking the divergence of (2.47b) and, with the help of (2.47a), obtaining the Poisson equation as

$$\nabla^2 p = -(\nabla \boldsymbol{u})^\top : \nabla \boldsymbol{u}, \tag{2.50}$$

where : represents the dyadic product[17]. A Robin boundary condition for p is obtained from the product of (2.47b) and the wall normal $\boldsymbol{n}$. This boundary condition and the Poisson equation uniquely define p up to an arbitrary constant. As the constant does not affect the velocity field, we can consider the residual $\mathcal{N}(\boldsymbol{u})$ as a nonlinear function of only the velocity field, given as

$$\mathcal{N}(\boldsymbol{u}) \coloneqq \partial_t \boldsymbol{u} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} + \nabla p - \nu \nabla^2 \boldsymbol{u}. \tag{2.51}$$

The formulation applies to low-Mach number incompressible flows of Newtonian fluids in steady domains.

2.2.1.2 Galerkin system

The Galerkin method approximates the solution of the IBVP formulated in Sec. 2.2.1.1 as

$$\boldsymbol{u}^{\mathrm{Gal}}(\boldsymbol{\chi}, t) = \boldsymbol{\Phi}_0(\boldsymbol{\chi}) + \sum_{i=1}^{N_{\mathrm{Gal}}} a_i(t) \boldsymbol{\Phi}_i(\boldsymbol{\chi}), \tag{2.52}$$

$$\frac{\mathrm{d}\boldsymbol{a}(t)}{\mathrm{d}t} = \boldsymbol{f}^{\mathrm{Gal}}(\boldsymbol{a}; t). \tag{2.53}$$

Here, $\boldsymbol{u}^{\mathrm{Gal}}$ is an approximation of the velocity field $\boldsymbol{u}$ in terms of the base flow $\boldsymbol{\Phi}_0$ and an expansion using spatial modes $\boldsymbol{\Phi}_i$ and the corresponding temporal coefficients a_i, for all $i = 1, \ldots, N_{\mathrm{Gal}}$. In (2.53), a system of ODEs for the evolution of the temporal coefficients $\boldsymbol{a}(t) \coloneqq [a_1(t), \ldots, a_{N_{\mathrm{Gal}}}(t)]^\top$ is obtained through the propagators $\boldsymbol{f}^{\mathrm{Gal}} \coloneqq [f_1^{\mathrm{Gal}}, \ldots, f_{N_{\mathrm{Gal}}}^{\mathrm{Gal}}]^\top$. For simplicity, we define $a_0 = 1$ and rewrite (2.52) as

$$\boldsymbol{u}^{\mathrm{Gal}}(\boldsymbol{\chi}, t) = \sum_{i=0}^{N_{\mathrm{Gal}}} a_i(t) \boldsymbol{\Phi}_i(\boldsymbol{\chi}). \tag{2.54}$$

This ansatz is compatible with the weak solution of the N-S equation (Noack, Morzyński, et al., 2011). This property is used to obtain the *Galerkin projection*. The residual (2.51) is

[17] Let $\boldsymbol{A}$ and $\boldsymbol{B}$ be two matrices, then $\boldsymbol{A} : \boldsymbol{B} \coloneqq \sum_i \sum_j A_{ij} B_{ji}$.

projected on the subspace spanned by the N_{Gal} expansion modes $\boldsymbol{\Phi}_i$ as

$$\left\langle \boldsymbol{\Phi}_i, \mathcal{N}(\boldsymbol{u}^{\mathrm{Gal}}) \right\rangle_{\Omega} = 0, \quad i = 1, \ldots, N_{\mathrm{Gal}}. \tag{2.55}$$

We briefly mention some of the properties of the Galerkin expansion (2.52) before performing the projection (2.55). Firstly, the modes satisfy the incompressibility criterion, *i.e.*

$$\nabla \cdot \boldsymbol{\Phi}_i = 0, \quad i = 1, \ldots, N_{\mathrm{Gal}}, \tag{2.56}$$

and are orthonormal, such that

$$\langle \boldsymbol{\Phi}_i, \boldsymbol{\Phi}_j \rangle_{\Omega} = \delta_{ij}, \quad \forall\, i, j \in \{1, \ldots, N_{\mathrm{Gal}}\}. \tag{2.57}$$

Secondly, the orthonormality allows to determine the mode amplitudes for a given velocity field $\boldsymbol{u}$ by minimizing the residual $\left\| \boldsymbol{u}(\boldsymbol{\chi}, t) - \boldsymbol{u}^{\mathrm{Gal}}(\boldsymbol{\chi}, t) \right\|_{\Omega}$, which gives

$$a_i(t) = \langle \boldsymbol{u}(\boldsymbol{\chi}, t) - \boldsymbol{\Phi}_0(\boldsymbol{\chi}), \boldsymbol{\Phi}_i(\boldsymbol{\chi}) \rangle_{\Omega}, \quad i = 1, \ldots, N_{\mathrm{Gal}}. \tag{2.58}$$

We note that the mode amplitude a_i is independent of N_{Gal} as it is not influenced by the orthogonal residual $\sum_{i=N_{\mathrm{Gal}}+1}^{+\infty} a_i(t) \boldsymbol{\Phi}_i(\boldsymbol{\chi})$. Lastly, the boundary condition (2.49) is incorporated as

$$\boldsymbol{\Phi}_0(\boldsymbol{\chi}) = \boldsymbol{u}_{\mathrm{BC}}(\boldsymbol{\chi}), \quad \text{and} \quad \boldsymbol{\Phi}_i(\boldsymbol{\chi}) = 0, \quad i = 1, \ldots, N_{\mathrm{Gal}}, \quad \forall\, \boldsymbol{\chi} \in \partial\Omega. \tag{2.59}$$

Consolidating these properties, the Galerkin projection of each term of the residual in (2.55) is performed individually on the reduced basis space as follows:

1. The local acceleration term yields

$$\left\langle \boldsymbol{\Phi}_i, \partial_t \boldsymbol{u}^{\mathrm{Gal}} \right\rangle_{\Omega} = \left\langle \boldsymbol{\Phi}_i, \partial_t \sum_{j=0}^{N_{\mathrm{Gal}}} a_j \boldsymbol{\Phi}_j \right\rangle_{\Omega} = \sum_{j=1}^{N_{\mathrm{Gal}}} \dot{a}_j \underbrace{\langle \boldsymbol{\Phi}_i, \boldsymbol{\Phi}_j \rangle_{\Omega}}_{\delta_{ij}} = \dot{a}_i, \tag{2.60}$$

   since $\dot{a}_0 = 0$ and the modes are orthonormal.

2. The convective term yields

$$\begin{aligned} \left\langle \boldsymbol{\Phi}_i, -(\boldsymbol{u}^{\mathrm{Gal}} \cdot \nabla) \boldsymbol{u}^{\mathrm{Gal}} \right\rangle_{\Omega} &= \left\langle \boldsymbol{\Phi}_i, -\left[\left(\sum_{j=0}^{N_{\mathrm{Gal}}} a_j \boldsymbol{\Phi}_j \right) \cdot \nabla \right] \left(\sum_{k=0}^{N_{\mathrm{Gal}}} a_k \boldsymbol{\Phi}_k \right) \right\rangle_{\Omega} \\ &= \sum_{j=0}^{N_{\mathrm{Gal}}} \sum_{k=0}^{N_{\mathrm{Gal}}} q^c_{ijk} a_j a_k, \end{aligned} \tag{2.61}$$

   where the coefficient q^c_{ijk} of the quadratic term is defined as

$$q^c_{ijk} = -\langle \boldsymbol{\Phi}_i, \boldsymbol{\Phi}_j \cdot \nabla \boldsymbol{\Phi}_k \rangle_{\Omega}. \tag{2.62}$$

3. The viscous term yields

$$\left\langle \boldsymbol{\Phi}_i, \nu \nabla^2 \boldsymbol{u}^{\mathrm{Gal}} \right\rangle_{\Omega} = \left\langle \boldsymbol{\Phi}_i, \nu \nabla^2 \left(\sum_{j=0}^{N_{\mathrm{Gal}}} a_j \boldsymbol{\Phi}_j \right) \right\rangle_{\Omega} = \nu \sum_{j=0}^{N_{\mathrm{Gal}}} l^{\nu}_{ij} a_j, \tag{2.63}$$

where the coefficient l_{ij}^{ν} of the linear term is defined and transformed using Green's first identity as

$$l_{ij}^{\nu} = \left\langle \boldsymbol{\Phi}_i, \nabla^2 \boldsymbol{\Phi}_j \right\rangle_{\Omega} = -\left\langle \nabla \boldsymbol{\Phi}_i, \nabla \boldsymbol{\Phi}_j \right\rangle_{\Omega} + [\boldsymbol{\Phi}_i \nabla \boldsymbol{\Phi}_j]_{\partial\Omega}, \tag{2.64}$$

where the second term in the summation is the surface integral which is defined as

$$[\boldsymbol{\Phi}_i \nabla \boldsymbol{\Phi}_j]_{\partial\Omega} = \oint_{\partial\Omega} \boldsymbol{\Phi}_i \nabla \boldsymbol{\Phi}_j \cdot \mathrm{d}\boldsymbol{\Gamma} = \oint_{\partial\Omega} \boldsymbol{\Phi}_i (\nabla \boldsymbol{\Phi}_j \cdot \boldsymbol{n}) \, \mathrm{d}\Gamma, \tag{2.65}$$

where $\partial\Omega$ is the boundary region of Ω, $\boldsymbol{n}$ is the outward pointing unit normal of surface element $\mathrm{d}\Gamma$, and $\mathrm{d}\boldsymbol{\Gamma}$ is the oriented surface element. Note that the surface integral vanishes for Dirichlet boundary condition as $\boldsymbol{\Phi}_i = 0$ at the domain boundary.

4. For the pressure term, we first assume that pressure can be represented in a pseudo-quadratic Galerkin ansatz as

$$p^{\mathrm{Gal}}(\boldsymbol{\chi}, t) = \sum_{i=0}^{N_{\mathrm{Gal}}} \sum_{j=0}^{N_{\mathrm{Gal}}} a_i(t) a_j(t) p_{ij}(\boldsymbol{\chi}), \tag{2.66}$$

where the partial pressures p_{ij} satisfies the Poisson equation in (2.50) as

$$\nabla^2 p_{ij} = -(\nabla \boldsymbol{\Phi}_i)^{\top} : \nabla \boldsymbol{\Phi}_j. \tag{2.67}$$

The pressure term yields

$$\left\langle \boldsymbol{\Phi}_i, -\nabla p^{\mathrm{Gal}} \right\rangle_{\Omega} = \left\langle \boldsymbol{\Phi}_i, -\nabla \left(\sum_{j=0}^{N_{\mathrm{Gal}}} \sum_{k=0}^{N_{\mathrm{Gal}}} a_j a_k p_{jk} \right) \right\rangle_{\Omega} = \sum_{j=0}^{N_{\mathrm{Gal}}} \sum_{k=0}^{N_{\mathrm{Gal}}} q_{ijk}^{p} a_j a_k, \tag{2.68}$$

where the coefficient q_{ijk}^{p} of the quadratic term is defined as

$$\begin{aligned} q_{ijk}^{p} &= -\left\langle \boldsymbol{\Phi}_i, \nabla p_{jk} \right\rangle_{\Omega} && (2.69)\\ &= \left\langle (\nabla \cdot \boldsymbol{\Phi}_i) p_{jk} \right\rangle_{\Omega} - [\boldsymbol{\Phi}_i p_{jk}]_{\partial\Omega} && (2.70)\\ &= -[\boldsymbol{\Phi}_i p_{jk}]_{\partial\Omega}, && (2.71) \end{aligned}$$

where we have used the Green's first identity and the incompressibility criterion of (2.56). This term is zero along rigid walls as the surface integral vanishes due to Dirichlet boundary conditions, and under periodic boundary conditions.

Finally, using the results of Galerkin projection for individual terms, the weak form in (2.55) can be expanded to yield a Galerkin system as

$$\underbrace{\dot{a}_i}_{\substack{\text{Local}\\ \text{acceleration}\\ \text{term}}} = \underbrace{\nu \sum_{j=0}^{N_{\mathrm{Gal}}} l_{ij}^{\nu} a_j}_{\substack{\text{Viscous}\\ \text{term}}} + \underbrace{\sum_{j,k=0}^{N_{\mathrm{Gal}}} q_{ijk}^{c} a_j a_k}_{\substack{\text{Convective}\\ \text{term}}} + \underbrace{\sum_{j,k=0}^{N_{\mathrm{Gal}}} q_{ijk}^{p} a_j a_k}_{\substack{\text{Pressure}\\ \text{term}}}, \quad \forall\, i = 0, 1, \ldots, N_{\mathrm{Gal}}. \tag{2.72}$$

For dynamical analysis purpose, we exclude the mode $a_0 = 1$ associated with the base flow

and re-write the Galerkin system as

$$\dot{a}_i = C_i + \sum_{j=1}^{N_{\mathrm{Gal}}} L_{ij} a_j + \sum_{j=1}^{N_{\mathrm{Gal}}} \sum_{k=j}^{N_{\mathrm{Gal}}} Q_{ijk} a_j a_k, \quad \forall\, i = 1, \ldots, N_{\mathrm{Gal}}, \tag{2.73}$$

where $C_i \coloneqq \nu l^{\nu}_{i0} + q^{c}_{i00} + q^{p}_{i00}$ are the constant terms, $L_{ij} \coloneqq \nu l^{\nu}_{ij} + q^{c}_{ij0} + q^{c}_{i0j} + q^{p}_{ij0} + q^{p}_{i0j}$ are the coefficients of the linear terms, and Q_{ijk} are the coefficients of the quadratic terms. Note that the summation in last term of (2.73) is different from the corresponding term in (2.72). As we have $a_j a_k = a_k a_j$ for the quadratic terms, the coefficients can be represented once in the equation instead of separately for $a_j a_k$ and $a_k a_j$. Therefore, the quadratic term coefficients are given as

$$Q_{ijk} \coloneqq \begin{cases} q^{c}_{ijj} + q^{p}_{ijj} & \text{if} \quad j = k \\ q^{c}_{ijk} + q^{c}_{ikj} + q^{p}_{ijk} + q^{p}_{ikj} & \text{if} \quad j \neq k \end{cases}. \tag{2.74}$$

These polynomial coefficients are collected in a parameter vector given as

$$\boldsymbol{\theta} \coloneqq [\cdots C_i \cdots L_{ij} \cdots Q_{ijk} \cdots]^{\top} \in \mathbb{R}^{N_\theta \times 1}, \tag{2.75}$$

where $N_\theta = N_{\mathrm{Gal}} \times N_{\theta_i}$, and

$$N_{\theta_i} = 1 + N_{\mathrm{Gal}} + \frac{N_{\mathrm{Gal}}(N_{\mathrm{Gal}} + 1)}{2}, \tag{2.76}$$

is the number of parameters in the evolution model for the i-th mode with the corresponding coefficients $\boldsymbol{\theta}_i \coloneqq [C_i\, L_{i1} \cdots L_{iN_{\mathrm{Gal}}}\, Q_{i11} \cdots Q_{iN_{\mathrm{Gal}}N_{\mathrm{Gal}}}]^{\top} \in \mathbb{R}^{N_{\theta_i} \times 1}$.

2.2.2 POD reduced-order model (POD-ROM)

The Galerkin system given by (2.73) provides a way to construct a low-dimensional model using modes that span a subspace in which the dynamics is sufficiently well resolved. The performance of the Galerkin method is sensitive to the choice of the expansion modes. Mathematically, the Hilbert space of square-integrable solenoidal vector fields guarantees the existence of a complete orthonormal system[18] (Simader and Sohr, 1992). This property was exploited successfully for simple flow configurations. However, the method poses severe analytical and numerical challenges for turbulent flows.

In contrast, the choice of the mode $\boldsymbol{\Phi}_0$ corresponding to the base flow in (2.52) is not critical. The base mode is considered as either a steady solution of the N-S equations or the mean flow. In practice, both alternatives satisfy the incompressibility criterion (2.47a) and boundary condition (2.49). In this work, the mean field $\overline{\boldsymbol{u}}(\boldsymbol{\chi})$ is considered for the base mode (Cordier, El Majd, et al., 2010). This quantity is easily accessible from the experimental and numerical data.

The POD Galerkin reduced-order model (POD-ROM) is introduced as a problem-oriented empirical ansatz. By construction, POD-ROM is able to resolve the coherent structures and the associated nonlinearities. This makes the model suitable for applications to flow control which is generally effected by influencing the coherent structures (Bergmann, Cordier, and Brancher, 2005a; Bergmann and Cordier, 2008a). However, the POD model fails in general to resolve the turbulent dissipation and the stabilizing base flow variations. Periodic flows are

[18] The *completeness* of an orthonormal system implies that an arbitrary accuracy of (2.52) is achievable by increasing the number of modes N_{Gal}.

fully resolved by 4 to 10 modes (Deane et al., 1991; Noack, Afanasiev, et al., 2003; Noack, Papas, et al., 2005). Transitional flows may need $\mathcal{O}(10^3)$ modes to resolve 90% of turbulent kinetic energy. Fully turbulent flows may, in contrast, require $\mathcal{O}(Re^{9/4})$ modes (Landau and Lifshitz, 1987). The computational cost of using a Galerkin system scales in $\mathcal{O}(N_{\mathrm{Gal}}^3)$ due to the quadratic term. This can inhibit the practicality of Galerkin system for $\mathcal{O}(10^2)$ or more number of modes when compared to the cost of discretized Navier-Stokes solvers which increases linearly with the number of grid points. The Galerkin approximation truncated to N_{Gal} modes therefore gives a significant residual $\delta\boldsymbol{u}$,

$$\boldsymbol{u}(\boldsymbol{\chi},t) = \overline{\boldsymbol{u}}(\boldsymbol{\chi}) + \sum_{i=1}^{N_{\mathrm{Gal}}} a_i(t)\boldsymbol{\Phi}_i^{\mathrm{POD}}(\boldsymbol{\chi}) + \delta\boldsymbol{u}(\boldsymbol{\chi},t), \tag{2.77}$$

where

$$\delta\boldsymbol{u}(\boldsymbol{\chi},t) = \sum_{i=N_{\mathrm{Gal}}+1}^{+\infty} a_i(t)\boldsymbol{\Phi}_i^{\mathrm{POD}}(\boldsymbol{\chi}), \tag{2.78}$$

and $\boldsymbol{\Phi}_i^{\mathrm{POD}}$ is the spatial mode obtained from POD.

Using the ansatz (2.78) for the residual term $\delta\boldsymbol{u}$ leads to an additional term $\mathcal{R}_i(\boldsymbol{a};\boldsymbol{\theta}_s,t)$ in the Galerkin system (2.73),

$$\begin{cases} \dot{a}_i^{\mathrm{ROM}}(t) = f_i^{\mathrm{Gal}}(t) = f_i(\boldsymbol{a}^{\mathrm{ROM}};\boldsymbol{\theta}_i,t) + \mathcal{R}_i(\boldsymbol{a}^{\mathrm{ROM}};\boldsymbol{\theta}_s,t) \\ a_i^{\mathrm{ROM}}(0) = \left\langle \boldsymbol{u}(\boldsymbol{\chi},0) - \overline{\boldsymbol{u}}(\boldsymbol{\chi}), \boldsymbol{\Phi}_i^{\mathrm{POD}}(\boldsymbol{\chi}) \right\rangle_\Omega \end{cases}, \forall\, i = 1,\dots,N_{\mathrm{Gal}}. \tag{2.79}$$

The superscript $\square^{\mathrm{ROM}}$ is introduced to distinguish the solutions of the POD-ROM from the temporal coefficients determined by POD that are denoted by $\square^{\mathrm{POD}}$. f_i^{Gal} is the propagator in (2.53), which is the sum of the polynomial function f_i accounting for the N_{Gal} dominant modes,

$$f_i(\boldsymbol{a}^{\mathrm{ROM}};\boldsymbol{\theta}_i,t) = C_i + \sum_{j=1}^{N_{\mathrm{Gal}}} L_{ij}a_j^{\mathrm{ROM}}(t) + \sum_{j=1}^{N_{\mathrm{Gal}}}\sum_{k=j}^{N_{\mathrm{Gal}}} Q_{ijk}a_j^{\mathrm{ROM}}(t)a_k^{\mathrm{ROM}}(t), \tag{2.80}$$

and a residual term $\mathcal{R}_i(\boldsymbol{a}^{\mathrm{ROM}};\boldsymbol{\theta}_s,t)$ accounting for the higher modes, where $\boldsymbol{\theta}_s$ is a vector of the closure parameters. Here, we have replaced the spatial modes $\boldsymbol{\Phi}_i$ in the expansion (2.52) with the POD modes $\boldsymbol{\Phi}_i^{\mathrm{POD}}$ and N_{Gal} is the number of POD modes kept in the Galerkin projection. For later reference, unless mentioned specifically, the modes and modal coefficients denoted as $\boldsymbol{\Phi}_i$ and a_i will represent those obtained either from the snapshot POD or from the integration of the POD-ROM. The residual term $\mathcal{R}_i$ reads

$$\mathcal{R}_i(\boldsymbol{a}^{\mathrm{ROM}};\boldsymbol{\theta}_s,t) = \sum_{j=N_{\mathrm{Gal}}+1}^{+\infty} L_{ij}a_j^{\mathrm{ROM}}(t) + \sum_{j=N_{\mathrm{Gal}}+1}^{+\infty}\sum_{k=j}^{+\infty} Q_{ijk}a_j^{\mathrm{ROM}}(t)a_k^{\mathrm{ROM}}(t). \tag{2.81}$$

The coefficients C_i, L_{ij} and Q_{ijk} can be consequently represented as functions of the spatial eigenfunctions $\boldsymbol{\Phi}_i$ following the definitions in Sec. 2.2.1. Introducing compact notations, the propagator (2.53) can be represented as $f_i^{\mathrm{Gal}} = f_i(\boldsymbol{a};\boldsymbol{\theta}_i,t) + \mathcal{R}_i(\boldsymbol{a};\boldsymbol{\theta}_s,t)$ and all the model parameters can be assembled in a vector $\boldsymbol{\Theta} = [\boldsymbol{\theta}\ \boldsymbol{\theta}_s]^\top \in \mathbb{R}^{N_\Theta \times 1}$, where N_Θ is the sum of the number of elements in $\boldsymbol{\theta}$ and $\boldsymbol{\theta}_s$.

In practice, the energetic criterion given by (2.19) is used to determine the value of N_{Gal} based on the flow dynamics requirement. The quality of estimation can be evaluated using a

normalized mean square error which is given as

$$NMSE = \frac{1}{N_t}\sum_{k=1}^{N_t}\left[\frac{\left\|\boldsymbol{a}^{\mathrm{POD}}(t_k) - \boldsymbol{a}^{\mathrm{ROM}}(t_k)\right\|_2^2}{\left\|\boldsymbol{a}^{\mathrm{POD}}(t_k)\right\|_2^2}\right], \tag{2.82}$$

where $\boldsymbol{a}(t_k) = \{a_i(t_k)\}_{i=1,\ldots,N_{\mathrm{Gal}}}$ is a vector of temporal POD coefficients at the k-th time step.

Demo 2.3: Galerkin POD-ROM of the numerical 2D-cylinder wake flow dataset

For the flow past a cylinder at $Re = 100$, the POD-ROM is obtained by the Galerkin projection of the Navier-Stokes equations onto the POD modes (see Sec. 2.2.1.2). It has been shown that ROM with about 6 modes is accurate over short time for simple flow configuration (Deane et al., 1991). The coefficients C_i, L_{ij} and Q_{ijk} for the constant, linear, and quadratic terms are visualized in Fig. 2.7 after normalization. Here, $N_{\mathrm{Gal}} = 10$ modes have been chosen, fixing the number of coefficients for each mode to $N_{\theta_i} = 66$. We see that the modes are mostly harmonically related, with the dominant contribution from linear terms.

We observe that the Galerkin system is able to approximate the oscillations of the mode amplitude over a finite time horizon, see Fig. 2.8. The results are almost indistinguishable for the first mode and the main characteristics of the long-term behavior are reproduced for the higher modes. However, we observe discrepancies in the amplitude for the higher modes. The $NMSE$ over the range of $N_t = 1000$ is 0.047, indicating that the POD-ROM remains bounded over the long integration interval. The modal energy spectrum obtained from the POD Galerkin system behaves similarly as the POD modes as shown in Fig. 2.9. There is a good agreement for the main harmonic contributions from the mode pair a_1, a_2 while small deviations are observed in the higher harmonics which represent less than 5% in total TKE.

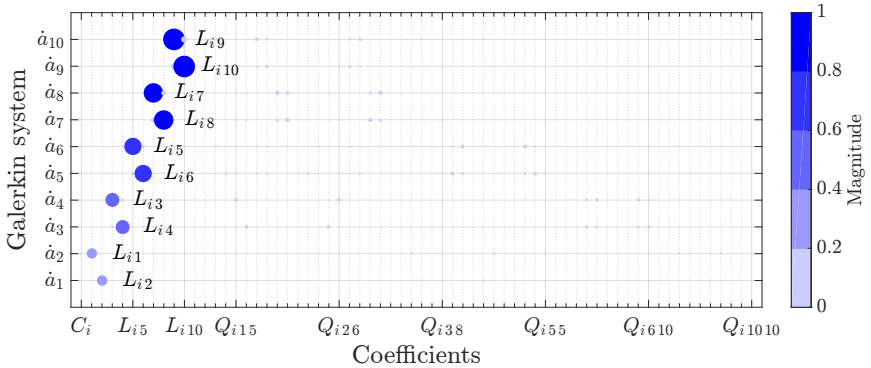


Figure 2.7: Normalized values of the parameter vector $\boldsymbol{\theta}$ of POD-ROM (2.79) with $\mathcal{R}_i = 0$. The size and color of the circles correspond to the magnitude of the coefficients.

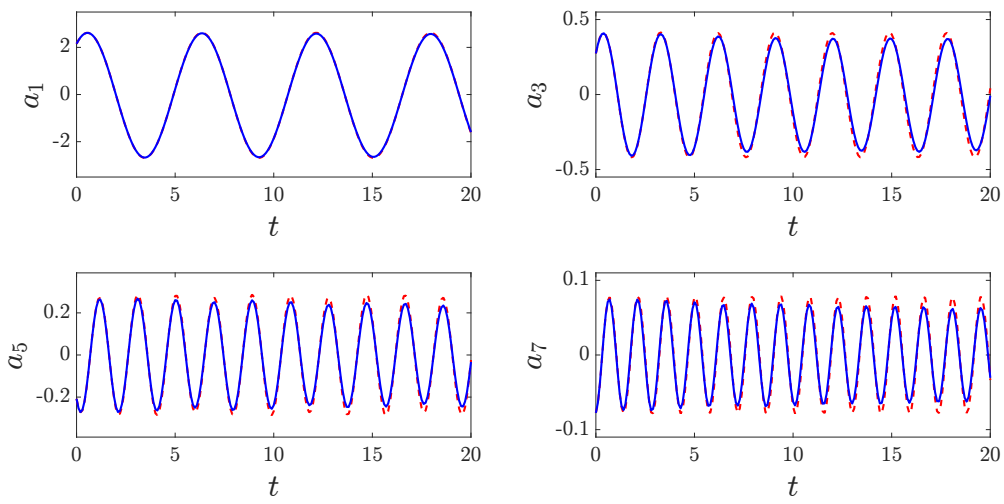


Figure 2.8: Time evolution of the predicted (blue solid line) POD coefficients $a_i^{\mathrm{ROM}}(t)$ $(i = 1, 3, 5, 7)$ compared with the exact evolution $a_i^{\mathrm{POD}}(t)$ (red dashed line) obtained from snapshot POD.

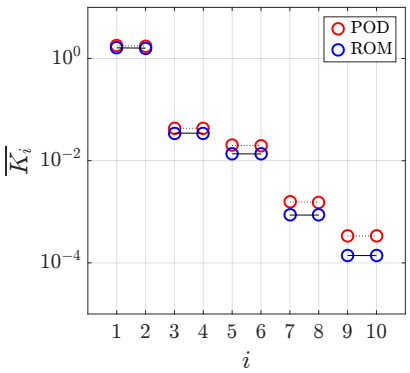


Figure 2.9: Modal energy distribution $\overline{K_i}$, $i = 1, \ldots, 10$ of exact POD modes and the modes obtained from the POD-ROM system.

2.2.3 Limitations of POD-ROM

For general control-oriented problems, the POD Galerkin method is often unable to yield robust models. The reason is either the incompleteness of the POD modes, or the intentional truncation of the number of modes performed in order to reduce the dimension of the problem.

Even though the POD modes are optimal in L^2 sense, the corresponding Galerkin system may fail to fully resolve the Navier-Stokes equation due to the lack of completeness of the POD modes in the Hilbert space of square integrable vector fields $L^2(\Omega)$ (Kutz, 2013).

To counter the truncation effect, the Galerkin projection on any orthonormal basis is usually supplied with calibrated corrections of the constant and linear terms (Galletti et al., 2004) or for the quadratic terms (Cordier, El Majd, et al., 2010). These corrections can also account for the pressure term or general ill-conditioning of the constant, linear, and quadratic terms. The Galerkin models have been constructed for wide range of flow configurations but additional care is needed for robust estimation. For a stable system, POD-ROM is supported with auxiliary model that accounts for the unresolved sub-scale turbulence. The POD modes also often need

to be augmented with additional modes for the base flow variations. These so-called *shift modes* (Bergmann and Cordier, 2008a, for instance) are not included in the current work as only post-transient states has been considered.

2.2.4 Stabilization of POD-ROM

The model reduction effected by the truncation in (2.79) leads to a ROM which, in general, is not sufficiently accurate to estimate the long-term dynamics correctly. As discussed, this truncation is equivalent to keeping only the most energetic modes that are associated with the large coherent structures. The viscous dissipation in turbulent flows takes place at small scales that are represented by low energy modes and are therefore not taken into account. As a consequence, the contribution of the viscous dissipation terms are neglected and the solution of (2.79) may diverge and/or drift in time as the leading ROM is not able to dissipate enough energy. To account for the error introduced by the truncation in the POD-ROM, various stabilization methods are used (Bergmann, Bruneau, et al., 2009).

The focus of stabilization is on the residual $\mathcal{R}_i(\boldsymbol{a};\boldsymbol{\theta}_s,t)$ component of the propagator (2.79), including the identification of the closure parameters $\boldsymbol{\theta}_s$. The residual accounts for the interaction between the resolved modes and the unresolved modes. The flow can be mentally divided on spectral basis into coherent structures with characteristic dominant frequency ω_c, base flow variations at low frequencies $\omega \ll \omega_c$, and small-scale turbulent fluctuations at high frequencies $\omega \gg \omega_c$. Generally, the first N_{Gal} modes represent the low to dominant frequencies $\omega \lesssim \omega_c$ and the residual describes the high frequencies $\omega \gg \omega_c$. The linear term of the residual (2.81) contributes to higher frequencies while the quadratic term in (2.81) contributes to the whole frequency spectrum (Noack, Morzyński, et al., 2011).

2.2.4.1 Modal constant eddy viscosity

One of the classes of stabilization methods is based on the use of eddy viscosity to close the ROM (Aubry et al., 1988). The eddy viscosity can be thought of as a subscale turbulence representation of the POD, trying to introduce the dissipative effect by changing the parameters on which the model depends. In the context of fluid flows, the parameter is the global dimensionless viscosity ν which is augmented with "modal eddy viscosity" ν_i^T (Protas et al., 2015) following the idea of Boussinesq's turbulent viscosity. The residual (2.81) is modified as

$$\mathcal{R}_i(\boldsymbol{a};\boldsymbol{\theta}_s,t) = \nu_i^T \sum_{j=1}^{N_{\mathrm{Gal}}} L_{ij} a_j(t). \tag{2.83}$$

This yields the growth rate associated with the residual term to be $D_i = \nu_i^T L_{ii}$. A convenient way is to consider $\nu_i^T = \nu_c$ as a constant viscosity acting in the same way on all the modes. This approach is known as the Heisenberg model (Podvin and Lumley, 1998). A more established approach is to consider ν_i^T as a function of the mode number i to make the viscous dissipation vary for each POD mode. One such approach is to represent ν_i^T as a function of the relative mode number (i/N_{Gal}) (see Rempfer and Fasel, 1994; Rempfer, 1996). This closure was recently generalized by Pyta and Abel (2017) as

$$\nu_i^T = \nu_c \left(\frac{i}{N_{\mathrm{Gal}}}\right)^{\alpha}, \tag{2.84}$$

where $\nu_c \in \mathbb{R}$ and $\alpha \in \mathbb{R}$ are the closure coefficient and the associated exponent, respectively. These values are together represented as stabilization parameter vector $\boldsymbol{\theta}_s \coloneqq [\nu_c \ \alpha]^\top$. This ansatz works well with small-bandwidth dynamics such as laminar and transitional flows.

2.2.4.2 Modal nonlinear eddy viscosity – FTT closure formalism

The propagator (2.79) often diverges for broadband turbulence (Noack, Morzyński, et al., 2011) due to the non-physical representation of the nonlinear energy cascade for $\mathcal{R}_i = 0$. A finite-time thermodynamics (FTT) formalism (Noack, Schlegel, et al., 2008) yields a nonlinear eddy viscosity model which tries to match the energy transfer rates of the accurate dynamical system. The propagator residual obtained from the power balance (Östh et al., 2014) is represented as

$$\mathcal{R}_i(\boldsymbol{a};\boldsymbol{\theta}_s,t) = \nu_i^T \sqrt{\frac{K(t)}{K^\infty}} \sum_{j=1}^{N_{\mathrm{Gal}}} L_{ij} a_j(t). \tag{2.85}$$

where $K(t)$ is the turbulent kinetic energy (TKE) of the fluctuation $\boldsymbol{u}'(\boldsymbol{\chi},t) \coloneqq \boldsymbol{u}(\boldsymbol{\chi},t) - \overline{\boldsymbol{u}}(\boldsymbol{\chi})$ and is given as

$$K(t) \coloneqq \frac{1}{2} \left\|\boldsymbol{u}'\right\|_\Omega^2 \approx \sum_{i=1}^{N_{\mathrm{Gal}}} K_i(t), \tag{2.86}$$

with $K_i(t)$ quantifying the energy content of each modal component $\boldsymbol{u}^{[i]}(\boldsymbol{\chi},t) = a_i(t)\boldsymbol{\Phi}_i(\boldsymbol{\chi})$ as

$$K_i(t) \coloneqq \frac{1}{2} \left\|\boldsymbol{u}^{[i]}\right\|_\Omega^2 = \frac{1}{2} a_i^2(t), \tag{2.87}$$

using the orthonormality of the modes. The total TKE K^∞ is defined as

$$K^\infty = \sum_{i=1}^{N_{\mathrm{Gal}}} \frac{\lambda_i}{2}, \tag{2.88}$$

where λ_i are the POD eigenvalues. This formulation takes into consideration the fluctuation levels for the broadband turbulence and leads to a higher damping when the fluctuations are high $(K(t)/K^\infty > 1)$ as compared to the linear subscale turbulence representation in Sec. 2.2.4.1. The parameter vector $\boldsymbol{\theta}_s$ for the closure coefficients takes the form $\boldsymbol{\theta}_s \coloneqq [\nu_1^T \ \nu_2^T \ \dots \ \nu_{N_{\mathrm{Gal}}}^T]^\top$. The nonlinear eddy viscosity has been demonstrated to guarantee bounded Galerkin solutions (Cordier, Noack, et al., 2013).

The polynomial coefficients $\boldsymbol{\theta}$ of POD-ROM are generally calibrated using *least-square method* (Perret et al., 2006; Cordier, El Majd, et al., 2010) such that the coefficients a_i^{ROM} are as close as possible to a_i^{POD}. The calibration can also be supplemented with *sparse identification*, leveraging the fact that most physical systems have only a few nonlinear terms in the dynamics (Brunton, Proctor, et al., 2016a). Different regression methods to identify dynamical systems that are linear in parameters are discussed in Sec. 3.1.

Additionally, data-driven system identification approaches can potentially serve as tools for calibration. The identification methods considered in this work can be classified into two different types of classes depending on the point of view kept:

- *data-assimilation method* (Kutz, 2013) introduced in Sec. 3.2 and applied in Chap. 5, which combines the measurement data and the results from a prescribed model to improve the predictive power of the model, and
- *regression via neural network* (Brunton, Noack, and Koumoutsakos, 2019) introduced in Sec. 3.3 and applied in Chap. 6, which identifies or corrects nonlinear systems using deep neural networks.

If the original number of modes is not large enough to provide appropriate dissipation, these methods are able to stabilize the model by allowing the representation of the inter-modal dependence through the residual terms or surrogate models.

Chapter 3

Data-driven dynamical system identification

Contents

The identification of physical laws from simulation or experimental data can be critical to applications where governing equations are unknown. Attempts towards the automated data-driven inference of dynamical systems have been actively made since several decades (Crutchfield and McNamara, 1987). In practical applications, it is often possible to obtain the data by measurement of the time evolution of different observables. A natural question that follows is whether these time-series data can be used to extract the dynamical system from which the data is obtained. In some cases, it is possible to derive physical laws from first principals using data as well as some *a priori* knowledge of the system (*white-box* modelling). However, in cases with high-dimensional and/or unknown physics, the derivation remains a challenge.

There has been substantial research focused towards automating the process of data-driven model discovery (Montáns et al., 2019). Data-driven identification methods can be grouped into two categories: i) methods that provide interpretable closed-form *grey-box* functions for the dynamics as ordinary and partial differential equations, and ii) methods that accurately reproduce the observed dynamical features at the cost of interpretability using *black-box* functions (*e.g.* neural networks). In the grey-box approach, a specific model is assumed for the

data with known physics, while in the black-box approach, models seek accurate predictions. System identification using these data-driven approaches has emerged as a viable alternative to expert knowledge and first-principles derivations. Novel approaches belonging to both these classes are introduced in this chapter and explored in the subsequent chapters.

For systems exhibiting linear dynamics, methods like eigensystem realization algorithm (ERA) (Juang and Pappa, 1985) and dynamic mode decomposition (DMD) (Rowley, Mezić, et al., 2009; Schmid, 2010) have been used for model extraction using time series data. For nonlinear systems, the same has been effected with black-box approaches, using methods like nonlinear autoregressive models with moving average and exogenous input (NARMAX) (Chen and Billings, 1989), equation-free algorithms (Kevrekidis and Samaey, 2009), and neural networks (González-García et al., 1998). For extracting physical laws in the form of ODEs without any prior knowledge about physics or kinematics, Schmidt and Lipson (2009) proposed a symbolic regression approach based on *genetic programming*, providing momentum conservation laws from experimental data.

Often, a combination of first-principle structure determination and empirical parameter estimation is employed for data-driven identification. Following Ockham's principle "All things being equal, the simplest solution tends to be the best one", we search for reduced-order models. In Sec. 3.1, different *regression* methods is discussed to identify dynamical systems that are linear in parameters. As a starting point, the structure of the system is considered to be same as that obtained from the Galerkin projection of the incompressible Navier-Stokes equation onto the POD modes. For this reason, we search for quadratic dynamical evolution models. The use of a pre-defined library based on the known understanding of physics saves from the task of simultaneously identifying the model structure and parameters. For the given model structure, the unknown parameters $\boldsymbol{\theta}$ of (2.79) are identified from limited and potentially noisy data.

For nonlinear parameter estimation, data assimilation tools are used. In Sec. 3.2, *Kalman filter* and its variants are introduced. The Kalman filter, a recursive data-processing algorithm, is the most commonly used sequential data assimilation technique. For linear dynamical models with Gaussian uncertainties, an optimal estimation is obtained. In fluid mechanics, the dynamics is non-linear and very often of very large size. For this reason, we have used an Ensemble Kalman filter method (EnKF) which, via a Monte Carlo approach, estimates the corrections to be made to the state of the system and its parameters according to observations (Moradkhani et al., 2005).

Finally, a black-box approach based on a deep neural network is discussed in Sec. 3.3. We introduce a non-intrusive model reduction method as a new modular approach for supervised learning of the dynamics. The objective is to determine a regression model to iteratively predict the POD coefficients. For this purpose, we present a method based on a multistep, residual-based, parametrized deep neural network.

3.1 Linear regression models

As shown in Sec. 2.2.2, the incompressible Navier-Stokes equations may be represented by the following POD-based reduced-order model[1]:

$$\begin{cases} \dot{a}_i(t) = f_i(\boldsymbol{a}(t);\boldsymbol{\theta}_i) + \mathcal{R}_i(\boldsymbol{a}(t);\boldsymbol{\theta}_s) \\ \\ a_i(0) = a_i^\star(0) \end{cases} , \forall i = 1,\dots,N_{\text{Gal}}, \tag{3.1}$$

where $a_i^\star(0)$ are known initial conditions determined, for instance, by projecting the reference solution onto the POD modes. The propagator f_i is a quadratic model in a_i given by

$$f_i(\boldsymbol{a}(t);\boldsymbol{\theta}_i) = C_i + \sum_{j=1}^{N_{\text{Gal}}} L_{ij} a_j(t) + \sum_{j=1}^{N_{\text{Gal}}} \sum_{k=j}^{N_{\text{Gal}}} Q_{ijk} a_j(t) a_k(t). \tag{3.2}$$

The residual term $\mathcal{R}_i$ can be formulated in terms of different expressions, as detailed in Sec. 2.2.4. For a simpler presentation, $\mathcal{R}_i = 0$ is considered, *i.e.* the closure parameters $\boldsymbol{\theta}_s$ that define the residual $\mathcal{R}_i$ are not determined using the regression tool in this section. It is equivalent to assuming that long-term stable models can be obtained by resolving the dynamics sufficiently well with the retained N_{Gal} POD modes. However, note that in scenarios where $\mathcal{R}_i$ is defined by a linear model, the identification of the parameters $\boldsymbol{\theta}_s$ can also be performed simultaneously using the treatment discussed in this section.

Given a set of N_t samples of quantities a_i^{POD}, *i.e.* the reference data at discrete times t_k, the objective is to determine the dynamical parameters $\boldsymbol{\theta}_i$ in (3.2). The linear propagator (3.2) is rewritten as

$$\dot{a}_i(t_k) = f_i(\boldsymbol{a}(t_k);\boldsymbol{\theta}_i) = \sum_{j=1}^{N_{\theta_i}} \Pi_j(\boldsymbol{a}(t_k)) \left(\boldsymbol{\theta}_i\right)_j + \epsilon_i(t_k), \tag{3.3}$$

where $i = 1,\dots,N_{\text{Gal}}$ and $k = 1,\dots,N_t$. The different terms are introduced in (3.3) as:

- Π_j with $j = 1,\cdots,N_{\theta_i}$, the set of monomials[2] of the quadratic functions in $\boldsymbol{a}(t_k)$ where N_{θ_i} is the number of coefficients of $\boldsymbol{\theta}_i$ given by (2.76),
- $\left(\boldsymbol{\theta}_i\right)_j$, the j-th component of $\boldsymbol{\theta}_i$, and
- $\epsilon_i(t_k)$, the error term that adds numerical "noise" to the linear relationship.

The expression (3.3) can be rewritten in matrix form as[3]

$$\boldsymbol{y}_i = \boldsymbol{X}\boldsymbol{\beta}_i + \boldsymbol{\epsilon}_i \qquad i = 1,\dots,N_{\text{Gal}}. \tag{3.4}$$

[1] In this chapter, the notations are simplified by dropping the superscript ROM in (2.80). A restriction is made to only consider the case of autonomous dynamical systems for which the propagator does not depend explicitly on time.

[2] 1 constant term: $\Pi_1(\boldsymbol{a}(t_k)) = 1$; N_{Gal} linear terms: $\Pi_2(\boldsymbol{a}(t_k)) = a_1(t_k)$, $\cdots$, $\Pi_{N_{\text{Gal}}+1}(\boldsymbol{a}(t_k)) = a_{N_{\text{Gal}}}(t_k)$; $N_{\text{Gal}}(N_{\text{Gal}}+1)/2$ quadratic terms: $\Pi_{N_{\text{Gal}}+2}(\boldsymbol{a}(t_k)) = a_1(t_k)a_1(t_k)$, $\cdots$, $\Pi_{N_{\theta_i}}(\boldsymbol{a}(t_k)) = a_{N_{\text{Gal}}}(t_k)a_{N_{\text{Gal}}}(t_k)$

[3] A choice is made here to stick to the traditional notations of statistical learning (see Sec. 3.2 of Hastie et al., 2009, for instance). Hence, $\boldsymbol{X}$ is clearly different than the one defined in Chap. 2.

Assuming that the time derivatives $\dot{\boldsymbol{a}}_i$ can be determined from the coefficients $\boldsymbol{a}_i$, $\boldsymbol{y}_i$ is defined as the vector of the time derivatives $\dot{a}_i(t_k)$ of the i-th modal coefficient,

$$\boldsymbol{y}_i \coloneqq \dot{\boldsymbol{a}}_i = [\dot{a}_i(t_1)\, \dot{a}_i(t_2)\, \dots\, \dot{a}_i(t_{N_t})]^\top \in \mathbb{R}^{N_t \times 1}. \tag{3.5}$$

$\boldsymbol{X}$ is the so-called *design matrix*[4] defined for $j, k = 1, \dots, N_{\text{Gal}}$ as

$$\boldsymbol{X} \coloneqq \begin{bmatrix} 1 & \cdots & a_j(t_1) & \cdots & a_j(t_1)a_k(t_1) & \cdots \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots \\ 1 & \cdots & a_j(t_{N_t}) & \cdots & a_j(t_{N_t})a_k(t_{N_t}) & \cdots \end{bmatrix} \in \mathbb{R}^{N_t \times N_{\theta_i}}, \tag{3.6}$$

where each row corresponds to the vector of quadratic monomials in terms of $\boldsymbol{a}(t_k)$. The vector of unknown parameters is given by

$$\boldsymbol{\beta}_i \coloneqq \boldsymbol{\theta}_i = [C_i \cdots L_{ij} \cdots Q_{ijk} \cdots]^\top \in \mathbb{R}^{N_{\theta_i} \times 1} \quad \forall\, j, k = 1, \dots, N_{\text{Gal}}. \tag{3.7}$$

Finally, the error term is defined as

$$\boldsymbol{\epsilon}_i \coloneqq [\epsilon_i(t_1)\, \epsilon_i(t_2)\, \dots\, \epsilon_i(t_{N_t})]^\top \in \mathbb{R}^{N_t \times 1}. \tag{3.8}$$

The key point to note is that the system (3.4) is linear in terms of parameters. $\boldsymbol{y}_i$ is a vector of *observed* values or *dependent variables*. The columns of $\boldsymbol{X}$ are known as *regressors*, *explanatory variables* or *independent variables*. $\boldsymbol{\beta}_i$ is the vector of *regression coefficients* and $\boldsymbol{\epsilon}_i$ is the *error term* or sometimes *noise*. In order to simplify the presentation, the index i is dropped from (3.4) which gives the general linear model considered in this section as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}. \tag{3.9}$$

Details on linear regression models are given in App. C. For consistency, the number of unknown parameters is also referred as N_θ instead of N_{θ_i}.

3.1.1 Ordinary least-squares (OLS) method

In Perret et al. (2006), a least mean-squares estimation procedure is used to determine the solution of (3.9). The coefficients $\boldsymbol{\beta}$ obtained by OLS correspond to the minimization of the error term given by[5]

$$\varepsilon = \|\boldsymbol{\epsilon}\|_2^2 = \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2. \tag{3.10}$$

The minimization[6] of the error function leads to the normal equation $(\boldsymbol{X}^\top \boldsymbol{X})\boldsymbol{\beta} = \boldsymbol{X}^\top \boldsymbol{y}$. After inverting $\boldsymbol{X}^\top \boldsymbol{X}$, an estimation of $\boldsymbol{\beta}$ is obtained. A disadvantage of this approach is that the condition number of $\boldsymbol{X}^\top \boldsymbol{X}$ is square of the condition number of $\boldsymbol{X}$. Hence, the problem may quickly become ill-conditioned which can lead to an increase in the number of iterations required for the inversion and limits the accuracy of the solution.

These limitations are overcome by using singular value decomposition (SVD) which is theoretically equivalent to the direct solution. The SVD (see App. B.13) is performed on the

[4] In order to have a unique solution for the linear system (3.4), the case with $N_t > N_{\theta_i}$ is considered so that there are more temporal data than parameters to identify.

[5] Some penalization term can also be added, see Sec. 3.1.2 for ℓ_1-regularization and Cordier, El Majd, et al. (2010) for Tikhonov regularization.

[6] OLS, along with penalized least squares, is presented in more general terms in App. C.2.

matrix $\boldsymbol{X}$ which gives

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top. \tag{3.11}$$

Here, $\boldsymbol{U} \in \mathbb{R}^{N_t \times N_\theta}$ and $\boldsymbol{V} \in \mathbb{R}^{N_\theta \times N_\theta}$ are orthogonal matrices such that $\boldsymbol{U}^\top\boldsymbol{U} = \boldsymbol{V}^\top\boldsymbol{V} = \boldsymbol{I}_{N_\theta}$. $\boldsymbol{\Sigma} \in \mathbb{R}^{N_\theta \times N_\theta}$ is the diagonal matrix of singular values σ_i, arranged such that $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{N_\theta} \geq 0$. The solution for the OLS problem (3.10) is given as (Press et al., 1993)

$$\widehat{\boldsymbol{\beta}} = \sum_{n=1}^{N_\theta} \frac{1}{\sigma_n} \boldsymbol{U}_{:,n}^\top \boldsymbol{y} \boldsymbol{V}_{:,n}, \tag{3.12}$$

where $\boldsymbol{U}_{:,n}$ and $\boldsymbol{V}_{:,n}$ represent the n-th columns of the matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ for all $n = 1, \ldots, N_\theta$. If $\sigma_n \ll 0$, the value of $1/\sigma_n$ can be set to zero. This corresponds to adding a zero multiple of any linear combination of basis functions that are degenerate in the fit. Determining the cut-off index n_c, after which all the singular values are neglected, amounts to empirically determining the rank of the matrix $\boldsymbol{X}$. For this, hard- and soft-thresholding methods are often used (see App. B.15).

The variance in the estimate of $\widehat{\beta}_j$, the j-th component of $\widehat{\boldsymbol{\beta}}$, is given (Press et al., 1993) as

$$\mathrm{Var}\left(\widehat{\beta}_j\right) = \sum_{n=1}^{N_\theta} \frac{1}{\sigma_n^2} \left[\left(\boldsymbol{V}_{:,n}\right)_j\right]^2 = \sum_{n=1}^{N_\theta} \left(\frac{V_{jn}}{\sigma_n}\right)^2 \quad \forall\, j = 1, \ldots, N_\theta. \tag{3.13}$$

The SVD approach is numerically robust and helps to eliminate the roundoff error by monitoring the singular values. The quality of estimation is dependent largely on the number of samples N_t used. It must be sufficiently high to ensure good statistical convergence. A pseudo-code of the implementation of the linear system solver with threshold SVD is given in App. B.15.

Typically, the data is noisy and calculating the time derivative $\dot{\boldsymbol{a}}_i$ directly from the raw data can be problematic. The conventional finite-difference approximations will greatly amplify any noise present in the data. Therefore, the numerical derivative is obtained using a total-variation regularization method introduced by Chartrand (2011). This method allows the differentiation of nonsmooth and discontinuous time-signal data, which is often encountered, with adjustments in regularization parameters.

3.1.2 Sparse identification with SINDy

The sparse identification of nonlinear dynamics (SINDy) algorithm (Brunton, Proctor, et al., 2016a) is a data-driven regression method that can be used as an alternative to OLS for identifying the model (3.9). Like the previous method, the algorithm enables the discovery of a low-order model based on time-series data of observed values and regressors. However, in the current context, the SINDy algorithm seeks a sparse vector $\boldsymbol{\beta}$ satisfying (3.9). The motivation behind the sparse regression approach is the understanding that physical systems in general have only a few nonlinear terms active in the dynamics, accounting for a significantly low number of overall terms in the basis functions. This results in a model (3.9) with a sparse representation of $\boldsymbol{y}$ in the space of high-dimensional basis functions given by $\boldsymbol{X}$.

The active terms in the model are obtained by penalizing the number of terms in the dynamics by convex ℓ_1-regularization[7]

$$\widehat{\boldsymbol{\beta}}_{\text{LASSO}} = \arg\min_{\boldsymbol{\beta}} \left(\|\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_1 \right), \tag{3.14}$$

[7] See App. B.3.3 for the definition of the 1-norm. Other equivalent expressions of LASSO are given in App. C.3.

where $\lambda > 0$ is a sparsity promoting parameter. This parameter is selected heuristically to give the right trade-off between minimizing the residual given by the first term on the right-hand side of (3.14) while keeping the space of the permissible basis functions sparse with the second term. This way the model is interpretable and avoids overfitting. A least absolute selection and shrinkage operator (LASSO) algorithm introduced by Tibshirani (1996) can be used to solve (3.14). However, it may be computationally expensive for large datasets. Alternatively, a sequentially hard-thresholded least-squares algorithm is proposed in Brunton, Proctor, et al. (2016a). In this algorithm, a least squares approximation of $\boldsymbol{\beta}$ is iteratively thresholded against a sparsification knob $\lambda > 0$. Let $k \geq 0$ be the iteration index, the steps of the algorithm are given as[8]

$$\widehat{\boldsymbol{\beta}}^0 = \boldsymbol{X}^+ \boldsymbol{y}, \tag{3.15a}$$

$$S^k = \{1 \leq j \leq N_\theta \mid |\widehat{\beta}_j^k| \geq \lambda\}, \tag{3.15b}$$

$$\widehat{\boldsymbol{\beta}}^{k+1} = \underset{\boldsymbol{\beta} \in \mathbb{R}^{N_\theta} \,|\, \mathrm{supp}(\boldsymbol{\beta}) \subseteq S^k}{\arg\min} \|\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y}\|_2^2 . \tag{3.15c}$$

Before starting the iterations, the coefficient vector is initialized as $\widehat{\boldsymbol{\beta}}^0$ in (3.15a) by using the Moore-Penrose pseudoinverse $\boldsymbol{X}^+$ (see App. B.14). Then the iterations begin and the set of coefficients S^k with magnitude greater than the threshold λ are identified in (3.15b). Lastly, the coefficients in the support set defined by S^k are updated with the solution of the minimization problem (3.15c). The stopping criteria is defined as $S^{k+1} = S^k$, *i.e.* no further update in the set of retained coefficients takes place. It has been shown in Zhang and Schaeffer (2019) that the SINDy algorithm converges in at most N_θ steps and that it approximates the local minimizers of[9]

$$\min_{\boldsymbol{\beta}} \left(\|\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y}\|_2^2 + \lambda^2 \|\boldsymbol{\beta}\|_0 \right), \tag{3.16}$$

where ℓ_0-regularization is used.

The SINDy algorithm is a popular tool for estimating a linear regression model, and even discovering physics from data (Silva et al., 2020), but it has some constraints. Firstly, the linear regression solution is not unique when the number of coefficients exceeds the number of observations ($N_\theta > N_t$) (see Tibshirani, 2013). Secondly, the SINDy algorithm is customized according to the least squares formulation and does not readily accommodate extensions including incorporation of additional constraints, or nonlinear parameter estimation (Champion et al., 2020). For this reason, Zheng et al. (2018) introduced the sparse relaxed regularized regression (SR3) approach that extends the optimization formulation by including additional structure, robustness to outliers, and nonlinear parameter estimation. SR3 introduces an additional relaxation parameter to deal with nonconvex problems that is encountered when $N_\theta > \mathrm{Rank}(\boldsymbol{X})$. Finally, the regularization method may impose a strong bias in the distribution of the identified parameters. For example, the least-squares method corresponds to a Gaussian prior and a strong regularization may push the learned coefficients away from the values that give a best-fit model and towards the values conforming to the prior distribution. Therefore, an extra unbiasing step can be performed after the least-squares regularization to remove the bias from the identified nonzero coefficients (Silva et al., 2020).

[8] See App. B.3.4 for the definition of the support set.

[9] See App. B.3.4 for the definition of the 0-norm.

3.1.3 Sparse identification with LARS

The constraints of SINDy are addressed by using a Least Angle Regression (LARS) algorithm proposed by Efron, Hastie, et al. (2004); the additional "S" refers to the LASSO variant of the basic least angle regression (LAR) method, see Hastie et al. (2009). The LAR algorithm performs model regression in piecewise linear forward steps, accumulating at each step one explanatory coefficient which contributes to the active term in the basis function space. The name LAR reflects the mechanism of taking each of these steps along the equiangular direction between the identified explanatory variables. This allows smooth blending of the new variables instead of introducing them discontinuously as it is done in the classical forward stepwise regression. Simple modifications can be introduced in the LAR procedure to produce efficient LASSO solutions along the piecewise linear equiangular paths, resulting in the LARS algorithm.

3.1.3.1 LARS algorithm

The LARS algorithm gives all possible LASSO estimates for a given problem, the so-called *LASSO path*. A set of solutions is obtained, parameterized by the level of sparsity that is considered sufficient or desirable for the application. If the sequence of solutions is required to correspond to the regularization path of LASSO, it is necessary to allow the variables to be removed from the active set when any coefficient goes to zero (see below). The steps in the LARS procedure are given as follows[10] (see Sec. 3.4.4 Hastie et al., 2009) where k is the index of the algorithm:

Step 0 : Initialization: iteration $k = 0$

a) Standardize the outputs $\boldsymbol{y}$ and the columns of $\boldsymbol{X}$ (the predictors) with the exception of the first column (intercept) to have zero mean and unit variance (see Sec. 3.1.3.2 and App. C.1 for the procedure of standardization). Set the first column of $\boldsymbol{X}$ *i.e.* $\boldsymbol{X}^0$ with unit entries.

b) All the components $\widehat{\beta}_j$ $(j = 0, \cdots, p)$ of $\widehat{\boldsymbol{\beta}}$ are initialized to zero ($\widehat{\boldsymbol{\beta}} = \boldsymbol{0}$).

c) The residual is set to $\boldsymbol{r}_0 = \boldsymbol{y}$.

d) The active set $\mathcal{A}_0$ is empty ($\mathcal{A}_0 = \emptyset$).

Step 1 : Iteration $k = 1$

a) Find j_1, the column of $\boldsymbol{X}$ that is the most correlated with the residual $\boldsymbol{r}_0$, *i.e.*

$$j_1 = \underset{j=0,\cdots,p}{\arg\max} \langle \boldsymbol{X}^j, \boldsymbol{r}_0 \rangle, \tag{3.17}$$

where $\langle \cdot, \cdot \rangle$ represents the Euclidean inner product.

b) The coefficient $\widehat{\beta}_{j_1}$ corresponding to $\boldsymbol{X}^{j_1}$ enters the active set $\mathcal{A}_1$ ($\mathcal{A}_1 = \{\widehat{\beta}_{j_1}\}$).

c) The coefficient $\widehat{\beta}_{j_1}$ is updated following

$$\widehat{\beta}_{j_1}(\alpha) = \widehat{\beta}_{j_1} + \alpha \underbrace{\langle \boldsymbol{X}^{j_1}, \boldsymbol{r}_0 \rangle}_{\boldsymbol{\delta_1}} \tag{3.18}$$

[10] Note that $\boldsymbol{X}^j$ is the j-th column of the matrix $\boldsymbol{X}$ (where $j = 0, \cdots, p$).

where α is the step size[11] in the least angle direction $\boldsymbol{\delta_1}$, see the general definition (3.21). It corresponds to moving $\widehat{\beta}_{j_1}$ in the direction of $\boldsymbol{\delta_1}$ until another candidate term, say $\boldsymbol{X}^{j_2}$, has the same value of correlation with the current residual as $\boldsymbol{X}^{j_1}$.

d) The residual is updated using (3.22).

Step 2 : Iteration k

a) Find j_k the column of $\boldsymbol{X}$, that is the most correlated with the residual $\boldsymbol{r}_{k-1}$, *i.e.*

$$j_k = \underset{j=0,\cdots,p}{\arg\max} \langle \boldsymbol{X}^j, \boldsymbol{r}_{k-1} \rangle. \tag{3.19}$$

b) Update the active set: $\mathcal{A}_k = \mathcal{A}_{k-1} \cup \left\{ \widehat{\beta}_{j_k} \right\}$.

c) The coefficient vector then evolves as

$$\widehat{\boldsymbol{\beta}}_{\mathcal{A}_k}(\alpha) = \widehat{\boldsymbol{\beta}}_{\mathcal{A}_{k-1}} + \alpha \boldsymbol{\delta}_k, \tag{3.20}$$

where the direction of update for the current step is given as

$$\boldsymbol{\delta}_k = (\boldsymbol{X}_{\mathcal{A}_{k-1}}^\top \boldsymbol{X}_{\mathcal{A}_{k-1}})^{-1} \boldsymbol{X}_{\mathcal{A}_{k-1}}^\top \boldsymbol{r}_{k-1} \in \mathbb{R}^{N_\theta \times 1}. \tag{3.21}$$

Updates are performed in the direction $\boldsymbol{\delta}_k$ until another candidate has the same correlation with the current residual. Note that the update $\boldsymbol{X}_{\mathcal{A}_k} \boldsymbol{\delta}_k \in \mathbb{R}^{N_t \times 1}$ of the fit vector makes the smallest and equal angle with each predictor in the active set $\mathcal{A}_k$, hence the name "least angle" (Hastie et al., 2009).

d) The residual is updated as:

$$\boldsymbol{r}_k = \boldsymbol{y} - \boldsymbol{X}_{\mathcal{A}_k} \widehat{\boldsymbol{\beta}}_{\mathcal{A}_k} \in \mathbb{R}^{N_t \times 1}. \tag{3.22}$$

Step 3 : (*LASSO modification*) If at the k-th step, any coefficient already in the active set $\mathcal{A}_k$ reaches zero, then the term is dropped from the active set and the current least angle direction $\boldsymbol{\delta}_k$ is recomputed.

Step 4 : Continue the steps until all the N_θ predictors have been entered. The full least-squares solution is reached after $\min(N_t - 1, N_\theta)$ steps.

The LARS algorithm is extremely efficient with the entire sequence of steps requiring $\mathcal{O}(N_\theta^3 + N_t N_\theta^2)$ computations for $N_\theta < N_t$ variables, same as the cost of a least squares fit using N_θ variables (Efron, Hastie, et al., 2004). The LASSO modification enables the solution to problems with $N_t \ll N_\theta$. The LARS algorithm has been shown to be able to handle cases where there are less observations than the variables $N_t \ll N_\theta$ (Efron, Hastie, et al., 2004). In such cases, the LARS terminates at the saturated least squares fit after $N_t - 1$ variables have been accumulated in the active set (the "-1" is because the data is centered as described in Sec. 3.1.3.2 and App. C.1).

3.1.3.2 Standardized regression

Standardized regression is a method where the underlying variables are *standardized* prior to performing the least squares multiple regression analysis. To perform standardized regression

[11] Remarkably, one can determine an analytical expression of α to quickly jump to the next point on the regularization path where the active set changes.

using (3.9), the vector $\boldsymbol{y} \in \mathbb{R}^{N_t}$ and the columns $\boldsymbol{X}^j = X_{:,j} \in \mathbb{R}^{N_t}$ of matrix $\boldsymbol{X}$ (here $j = 1, \ldots, N_\theta$) are transformed as standardized variables (or z-scores) $\widehat{\square}$ defined as

$$\widehat{\boldsymbol{y}} = \frac{\boldsymbol{y} - \overline{y}\,\mathbf{1}_{N_t}}{\sigma(\boldsymbol{y})}, \quad \text{and} \quad \widehat{\boldsymbol{X}}^j = \frac{\boldsymbol{X}^j - \overline{X}^j\,\mathbf{1}_{N_t}}{\sigma(\boldsymbol{X}^j)}, \quad \forall j = 2, \ldots, N_\theta, \tag{3.23}$$

where $\mathbf{1}_{N_t}$ is the all-ones vector of size N_t. Note that the column with unit entries $\boldsymbol{X}^1 = \mathbf{1}_{N_t}$ is not transformed. The mean and standard deviation of $\boldsymbol{y}$ are given as

$$\overline{y} = \frac{1}{N_t}\sum_{k=1}^{N_t} y_k, \quad \text{and} \quad \sigma(\boldsymbol{y}) = \sqrt{\frac{1}{N_t}\sum_{k=1}^{N_t}(y_k - \overline{y})^2}. \tag{3.24}$$

The mean and standard deviation of the columns $\boldsymbol{X}^j$ of $\boldsymbol{X} \in \mathbb{R}^{N_t \times N_\theta}$ are given as

$$\overline{X}^j = \frac{1}{N_t}\sum_{k=1}^{N_t} X_{k,j}, \quad \text{and} \quad \sigma(\boldsymbol{X}^j) = \sqrt{\frac{1}{N_t}\sum_{k=1}^{N_t}(X_{k,j} - \overline{A}^j)^2} \quad \forall\, j = 2, \ldots, N_\theta. \tag{3.25}$$

A regression carried out on the standardized variables produces standardized coefficients $\widehat{\boldsymbol{\beta}}$.

Standardization is performed in order to scale the different explanatory variables in $\boldsymbol{y}$ and $\boldsymbol{X}$. This is especially important in the regression with $\boldsymbol{X}$ involving polynomial and interaction terms which lead to orders of magnitude difference between the regressors. The differences in scale obscure the statistical significance of model terms and may produce imprecise coefficients, thus making it difficult to identify the correct model. Standardizing offers a way to reduce the multicollinearity produced by the higher-order terms.

As the LARS method performs regression by accumulating explanatory coefficients that have the highest correlation with the residual (see Sec. 3.1.3.1), standardization facilitates the identification of the regressor that has the largest impact on the dependent variable. It is the regressor with the highest absolute value of the corresponding coefficient. A typical deviation of that regressor from its mean produces the largest effect, as compared to the effects produced by similar deviations of the other regressors.

We must note that the coefficients $\widehat{\boldsymbol{\beta}}$ are not invariant to the standardization. In order to retrieve the unstandardized result $\boldsymbol{y}$, the original mean and standard deviation used while performing the standardized regression are used. The unstandardized result $\boldsymbol{y}$ is obtained as

$$\boldsymbol{y} = \widehat{\boldsymbol{y}}\,\sigma(\boldsymbol{y}) + \overline{y}\,\mathbf{1}_{N_t}, \tag{3.26}$$

where

$$\widehat{\boldsymbol{y}} = \widehat{\boldsymbol{X}}\,\widehat{\boldsymbol{\beta}} \quad \text{and} \quad \widehat{\boldsymbol{X}} = [\widehat{\boldsymbol{X}}^1\ \widehat{\boldsymbol{X}}^2\ \ldots\ \widehat{\boldsymbol{X}}^{N_t}]. \tag{3.27}$$

3.1.3.3 Model selection criteria

The LASSO estimator (3.14) is characterized by the tuning parameter $\lambda > 0$. This hyper-parameter controls the trade-off between the fit (accuracy) and complexity (sparsity) of the statistical model, also known as the bias-variance trade-off. The objective of regularization is to control the complexity of the fitted model. As λ increases, the model complexity is reduced through shrinkage – from least regularized LASSO ($\lambda = 0$) corresponding to OLS to most regularized LASSO ($\lambda = \infty$) corresponding to a constant fit. The model complexity can be quantitatively estimated in terms of the effective degrees of freedom $df(\lambda)$ for a given

regularization parameter λ. The evolution of $df(\lambda)$ can be used to select the optimal λ in the LASSO estimator. However, due to the nonlinear nature of LASSO, no explicit expression of $df(\lambda)$ exists.

The LARS algorithm presented in Sec. 3.1.3.1 performs regression with piece-wise linear updates in the current equiangular direction. The model complexity evolves as the active set of coefficients ($\boldsymbol{\mathcal{A}}_k$) updates with the LARS steps. The LARS step (k) can be considered as a metaparameter controlling the model complexity. Following Zou et al. (2007), the degrees of freedom of LARS equals the number of nonzero coefficients in the active set at the k-th step, which gives

$$df(\lambda) = \mathbb{E}[|\boldsymbol{\mathcal{A}}_k|], \tag{3.28}$$

where $|\boldsymbol{\mathcal{A}}_k|$ is the size of the active set. We have $\widehat{df}(\lambda) = |\boldsymbol{\mathcal{A}}_k|$ as the unbiased estimator of $df(\lambda)$.

Unlike the SINDy algorithm where the regularization parameter is selected heuristically, in the LARS algorithm, an information criterion is used to adaptively select an optimal value of the regularization parameter. An information criterion is a measure of the quality of a statistical model based on the level of fit (quantified in terms of the mean squared error) and complexity (a penalty for parameter identification using the sample data). The Akaike information criterion (AIC) (Akaike, 1974) and the Bayesian information criterion (BIC) (Schwarz, 1978) are two of the most used information criteria for model selection. Using the information criteria, the optimal regularization parameter (λ_{opt}) for the LASSO model is given as (Zou et al., 2007)

$$\lambda_{\text{opt}} = \arg\min_{\lambda} \frac{\|\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y}\|_2^2}{\sigma^2} + w_{N_t}\widehat{df}(\lambda), \tag{3.29}$$

where $\sigma^2 = \text{var}(\boldsymbol{y})$. The penalty factor is $w_{N_t} = 2$ for AIC and $w_{N_t} = \ln(N_t)$ for BIC. The LARS model efficiently solves the LASSO estimation problem for all λ with the cost of a single least square fit. Therefore, the optimal λ_{opt} can be immediately obtained once the entire LASSO solution path is solved by the LARS algorithm.

Although the information-criterion based model selection is fast, it must be noted that the method relies on the estimation of the degrees of freedom $df(\lambda)$, on the availability of large sample sizes ($N_t \gg 1$), and on a correct selection of the basis functions $\boldsymbol{X}$. AIC relies on the asymptotic approximation and may not be appropriate for finite data set. The BIC relies on the assumption of i.i.d.[12] random variable and normal distribution of the model errors. In regression, AIC has been shown to be asymptotically optimal for selecting the model with the least-mean squared error under the assumption that the set of basis functions $\boldsymbol{X}$ does not contain the true model (Yang, 2005). In this work, we implement the AIC for the LASSO estimation with LARS using the `LassoLarsIC` function from the `scikit-learn` package (Buitinck et al., 2013).
The LARS method is illustrated in Sec. 4.4 on different test cases.

In the next section, the data-driven techniques based on data assimilation are discussed.

[12] *i.e.* independent and identically distributed

3.2 Data assimilation (DA) methods

In Sec. 3.1, we considered the case where the identification is equivalent to a regression model which is linear in parameters. This is a simplification because in the general case, the residual term $\mathcal{R}_i$ may involve parameters associated with nonlinear terms, like (2.84). In order to identify the full set of model parameters $\boldsymbol{\Theta} = [\boldsymbol{\theta}\ \boldsymbol{\theta}_s]^\top$, we propose to use data assimilation methods.

Data assimilation (DA) is a generic methodology which combines heterogeneous observations with the underlying dynamical principles, governing the system under consideration, to estimate the states and/or physical quantities parameterizing the dynamics in an optimal way. Starting from a background solution (from *model*) and incoming imperfect information (from *observations*), an optimal estimation of the unknowns of the system is performed which takes into account the respective statistical confidences of the two complimentary, but incomplete and inaccurate, sources of information.

For the subsequent discussion, we assume that a nonlinear, time-discrete model for the dynamics of interest is available as

$$\boldsymbol{q}_k = \mathcal{M}_{k:k-1}(\boldsymbol{q}_{k-1}, \boldsymbol{\Theta}_k) + \boldsymbol{\eta}_k. \tag{3.30}$$

Here, $k = 1, \ldots, N_t$ is the time index with N_t being the total number of time steps. The propagation of the state $\boldsymbol{q}_{k-1} \in \mathbb{R}^{N_s}$, where N_s is the dimension of the state vector after spatial discretization, is performed from time t_{k-1} to t_k by a nonlinear function $\mathcal{M}_{k:k-1} : \mathbb{R}^{N_s} \to \mathbb{R}^{N_s}$. The model also depends on the model parameters vector $\boldsymbol{\Theta}_k \in \mathbb{R}^{N_p}$ at time t_k, where N_p is the number of model parameters. For simplicity, we have considered the full vector $\boldsymbol{\Theta}$ here but a partial set of parameters may also be considered without loss of generality. The term $\boldsymbol{\eta}_k \in \mathbb{R}^{N_s}$ is the model error of the true (unknown) process at time t_k. The error is represented here as a stochastic additive term and accounts for the cumulative errors in the parameters, the numerical scheme used for integrating the model (3.30), and the unresolved scales. The objective of the assimilation is to estimate $\boldsymbol{q}_k^t$, the *true* (unobservable) state.

For the observations, we denote the time index j to highlight the difference of the observation instances from the time steps t_k of the model state. The observation time steps $t_{k(j)}$ are a subset of the model integration steps (see Fig. 3.1) with $j = 1, \ldots, N_t^o$; $N_t^o \leq N_t$ being the total number of time steps at which the measurements are available. In the special case where the observations are available at all the model integration steps, we have $N_t^o = N_t$ and $t_{k(j)} = t_k$. The noisy discrete time observations of the state $\boldsymbol{q}_{k(j)}$ are represented as the components of the observation vector $\boldsymbol{y}_j^o \in \mathbb{R}^{N_o}$. The superscript "o" is used to differentiate the imperfect (noisy) observations from the actual (noise-free) observations $\boldsymbol{y}_j$. The relation between the observations and the state is given as

$$\boldsymbol{y}_j^o = \mathcal{H}_j(\boldsymbol{q}_j) + \boldsymbol{\epsilon}_j^o, \tag{3.31}$$

where $\mathcal{H}_j : \mathbb{R}^{N_s} \to \mathbb{R}^{N_o}$ denotes the observation function, which is nonlinear in general. Also, similar to the model error, we have the term $\boldsymbol{\epsilon}_j^o \in \mathbb{R}^{N_o}$ as the additive observation error. The error accounts for the instrument error, deficiencies in the observation operator, and the representation error arising from unresolved scales.

In several problems, the amount of available observation is not sufficient to fully describe the dynamical system ($N_o \ll N_s$). In such cases, the state estimation cannot fully rely on the data-driven approaches and a state evolution model becomes essential to fill the spatial and temporal gaps in the observations (Carrassi et al., 2018). The DA methods are designed to achieve the best use of the observational data and obtain an efficient merger of data and

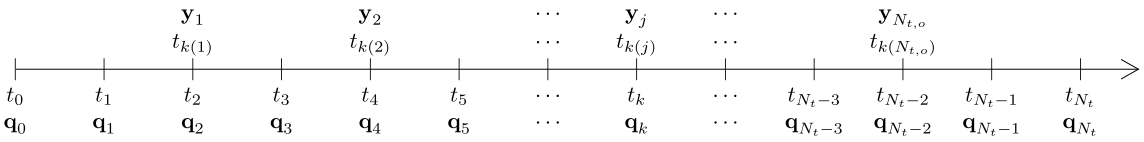


Figure 3.1: Time discretization of the interval $[t_0, t_{N_t}]$ into $N_t + 1$ nodes where the model states in the vector $\boldsymbol{q}_k = \boldsymbol{q}(\boldsymbol{\chi}, t_k)$ are defined. The measurement vectors $\boldsymbol{y}_j$ are available at time steps $t_{k(j)}$ where $j = 1, \ldots, N_{t,o}$.

model.

The DA methods are classically categorized either as *variational DA* or as *sequential DA*, also known as "filtering" or "probabilistic" DA (Asch et al., 2016). The goal of both methods is to seek an optimal solution in a given sense. In the variational approach, the solution minimizes a suitable cost function, and in the sequential approach, the solution is the one with the minimum variance. In certain special cases, both approaches reach the same solution. The sequential approach can be computationally expensive due to its statistical nature but provide rich information in terms of the average solution and its probability distribution. The two may also be combined into an *hybrid* approach which takes the advantage of the robustness of variational approach, and the information-rich solution of the statistical approach (Asch et al., 2016). However, the combined approach is highly problem dependent and can be nontrivial and computationally expensive to implement.

In this work, the statistical approaches based on one of the most well-known and often-used toolbox for stochastic estimation, the Kalman filter, will be discussed and implemented. In Sec. 3.2.1, the data assimilation is first introduced as a filtering problem and then the Kalman filter (KF) algorithm, used to optimally fit the model trajectory to the observations, is discussed. However, the KF assumes the uncertainties in the stochastic model states, parameters, observations and prior to be Gaussian distributed, and requires the model and observation operators to be linear. For handling models following non-Gaussian statistics and/or nonlinear operators, the ensemble Kalman filter (EnKF) is discussed in Sec. 3.2.2. This framework is further extended for parametrized models in Sec. 3.2.3 by considering the Dual-EnKF algorithm for joint estimation of the model state and parameters.

3.2.1 Kalman filter (KF)

The Kalman filter (KF) is a very well-known statistical method named after Rudolph E. Kalman who developed an algorithm to obtain recursive solution to the time-dependent discrete-data linear filtering problems (Kalman, 1960). A *filtering problem* is characterized by *sequential* processing, in which the observations are utilized chronologically as they become available. The filtering solution at a given time step t_k $(0 \leq k \leq N_t)$ is obtained by sequentially updating the solution up to the same time t_k. The solution thus accounts for all the observations before t_k.

We consider a dynamical system and seek to estimate the sequence of true states $\boldsymbol{q}_k^t$ at discrete times t_k when the observations $\boldsymbol{y}_k$ are available. A sequential DA scheme is set up to optimally fit an initially unconstrained model trajectory to the observations taking their respective uncertainties into consideration. We address the statistical DA approach from a Bayesian point of view and discuss the KF algorithm in detail.

3.2.1.1 Bayesian formulation of the state estimation problem

The complimentary sets of information from the model and data need to be combined to obtain a state estimation. We note that, both the model (3.30) and observations (3.31) are random due to the additive stochastic errors. Therefore, they can be described in terms of *probability density functions* (pdfs). The two sources of information are combined using the Bayesian formulation which offers a natural framework to approach the statistical DA problem. Hereafter, we follow Evensen's approach (Evensen, 2009) and present a mathematically and statistically consistent formulation of the combined parameter and state estimation.

In the Bayesian formulation, the output of the estimation process is given by the *posterior distribution*

$$p(\boldsymbol{q}, \boldsymbol{\Theta} \,|\, \boldsymbol{y}) = \frac{p(\boldsymbol{y} \,|\, \boldsymbol{q}, \boldsymbol{\Theta}) p(\boldsymbol{q}, \boldsymbol{\Theta})}{p(\boldsymbol{y})}. \tag{3.32}$$

The posterior distribution gives the joint pdf of the random variables in the form of the unknown state $\boldsymbol{q}$ and parameter $\boldsymbol{\Theta}$ conditioned on the observations $\boldsymbol{y}$. The components of the Bayesian inference on the right-hand side of (3.32) are: i) $p(\boldsymbol{y} \,|\, \boldsymbol{q}, \boldsymbol{\Theta})$, the *likelihood* of the data conditioned to the model state and parameters, *i.e.* what would be the observation if the true state and parameters were known, ii) $p(\boldsymbol{q}, \boldsymbol{\Theta})$, the joint *prior pdf* that describes the state and parameters before the assimilation, and iii) $p(\boldsymbol{y})$, the *marginal distribution* of the observation. The marginal distribution is independent of the model and only contributes as a normalization constant. The posterior distribution (3.32) is therefore more commonly written in terms of the product of the likelihood and prior distributions

$$p(\boldsymbol{q}, \boldsymbol{\Theta} \,|\, \boldsymbol{y}) \propto p(\boldsymbol{y} \,|\, \boldsymbol{q}, \boldsymbol{\Theta}) p(\boldsymbol{q}, \boldsymbol{\Theta}). \tag{3.33}$$

Generally, the dynamical model is supported with initial and boundary conditions. We consider them as random variables and define the pdfs $p(\boldsymbol{q}_0)$ and $p(\boldsymbol{q}_b)$ for the estimates of the initial condition $\boldsymbol{q}_0$ and the boundary condition $p(\boldsymbol{q}_b)$. Similarly, we define the pdf of the estimate of the parameters $\boldsymbol{\Theta}$ as $p(\boldsymbol{\Theta})$. The prior pdf $p(\boldsymbol{q}, \boldsymbol{\Theta})$ is then rewritten as

$$p(\boldsymbol{q}, \boldsymbol{\Theta}, \boldsymbol{q}_0, \boldsymbol{q}_b) = p(\boldsymbol{q}, \boldsymbol{\Theta} \,|\, \boldsymbol{q}_0, \boldsymbol{q}_b) p(\boldsymbol{q}_0) p(\boldsymbol{q}_b) = p(\boldsymbol{q} \,|\, \boldsymbol{\Theta}, \boldsymbol{q}_0, \boldsymbol{q}_b) p(\boldsymbol{q}_0) p(\boldsymbol{q}_b) p(\boldsymbol{\Theta}). \tag{3.34}$$

Then (3.33) is rewritten as

$$p(\boldsymbol{q}, \boldsymbol{\Theta}, \boldsymbol{q}_0, \boldsymbol{q}_b \,|\, \boldsymbol{y}) \propto p(\boldsymbol{q} \,|\, \boldsymbol{\Theta}, \boldsymbol{q}_0, \boldsymbol{q}_b) p(\boldsymbol{q}_0) p(\boldsymbol{q}_b) p(\boldsymbol{\Theta}) p(\boldsymbol{y} \,|\, \boldsymbol{q}, \boldsymbol{\Theta}), \tag{3.35}$$

where it has been assumed that the model and observational sequences, and the initial and boundary conditions are independent in time and also mutually uncorrelated. Here the pdf $p(\boldsymbol{q} \,|\, \boldsymbol{\Theta}, \boldsymbol{q}_0, \boldsymbol{q}_b)$ gives the prior density of the model solution for given parameters, and initial and boundary conditions.

In practice, the model state and observations are discretized in time. The state at the specific time intervals is given as $\boldsymbol{q}_k(\boldsymbol{\mathcal{X}}) = \boldsymbol{q}(\boldsymbol{\mathcal{X}}, t_k)$ for all $k = 0, \ldots, N_t$ with the corresponding boundary condition $\boldsymbol{q}_{b,k}$. For the state evolution from time t_{k-1} to t_k, we assume that the model follows a first-order Markov process[13] and gives the pdf as $p(\boldsymbol{q}_k \,|\, \boldsymbol{q}_{k-1}, \boldsymbol{\Theta}, \boldsymbol{q}_{b,k})$. The

[13] A random process $\{X(t), t \in T\}$ is called a first-order Markov process if for any $t_0 < t_1 < \ldots < t_k$, the conditional probability of $X(t_k)$ for given values of $X(t_0), X(t_1), \ldots, X(t_{k-1})$ depends only on $X(t_{k-1})$, *i.e.* $p(X(t_k) \,|\, X(t_{k-1}), X(t_{k-2}), \ldots, X(t_0)) = p(X(t_k) \,|\, X(t_{k-1}))$.

joint pdf for the model states $\{\boldsymbol{q}_k\}_{k=1}^{N_t}$ and parameters $\boldsymbol{\Theta}$ in (3.34) is rewritten as[14]

$$p(\{\boldsymbol{q}_k\}_{k=1}^{N_t}, \boldsymbol{\Theta}, \boldsymbol{q}_0, \boldsymbol{q}_b) \propto p(\boldsymbol{q}_0)p(\boldsymbol{q}_b)p(\boldsymbol{\Theta}) \prod_{k=1}^{N_t} p(\boldsymbol{q}_k \,|\, \boldsymbol{q}_{k-1}, \boldsymbol{\Theta}, \boldsymbol{q}_b). \tag{3.36}$$

Similarly, as shown in Fig. 3.1, the discrete measurements $\boldsymbol{y}_j$ are collected at times $t_{k(j)}$, where the observation indices $j = 1, \ldots, N_{t,o}$ are such that $\{t_{k(j)}\} \subseteq \{t_k\}$ and $0 < k(1) < \ldots < k(N_{t,o}) < N_t$. The measurement errors are assumed to be uncorrelated in time. The likelihood pdf of the observations $\{\boldsymbol{y}_j\}_{j=1}^{N_{t,o}}$ is then given as

$$p(\{\boldsymbol{y}_j\}_{j=1}^{N_{t,o}} \,|\, \{\boldsymbol{q}_k\}_{k=1}^{N_t}, \boldsymbol{\Theta}) = \prod_{j=1}^{N_{t,o}} p(\boldsymbol{y}_j \,|\, \boldsymbol{q}_{k(j)}, \boldsymbol{\Theta}). \tag{3.37}$$

Finally, the discrete formulation of the Bayes' theorem is obtained by substituting (3.36) and (3.37) in (3.35) as

$$p(\{\boldsymbol{q}_k\}_{k=1}^{N_t}, \boldsymbol{\Theta}, \boldsymbol{q}_0, \boldsymbol{q}_b \,|\, \{\boldsymbol{y}_j\}_{j=1}^{N_{t,o}}) \propto p(\boldsymbol{q}_0)p(\boldsymbol{q}_b)p(\boldsymbol{\Theta}) \prod_{k=1}^{N_t} p(\boldsymbol{q}_k \,|\, \boldsymbol{q}_{k-1}, \boldsymbol{\Theta}, \boldsymbol{q}_b) \prod_{j=1}^{N_{t,o}} p(\boldsymbol{y}_j \,|\, \boldsymbol{q}_{k(j)}, \boldsymbol{\Theta}). \tag{3.38}$$

For a model following a first-order Markov process, the sequential processing of the observations is performed by rewriting the Bayesian formulation (3.38) as

$$\begin{aligned} p(\{\boldsymbol{q}_k\}_{k=1}^{N_t}, \boldsymbol{\Theta}, \boldsymbol{q}_0, \boldsymbol{q}_b \,|\, \{\boldsymbol{y}_j\}_{j=1}^{N_{t,o}}) \propto\ & p(\boldsymbol{q}_0)p(\boldsymbol{q}_b)p(\boldsymbol{\Theta}) \prod_{k=1}^{k(1)} p(\boldsymbol{q}_k \,|\, \boldsymbol{q}_{k-1}, \boldsymbol{\Theta}, \boldsymbol{q}_b)p(\boldsymbol{y}_1 \,|\, \boldsymbol{q}_{k(1)}, \boldsymbol{\Theta}) \ldots \\ & \prod_{k=k(N_{t,o}-1)+1}^{k(N_{t,o})} p(\boldsymbol{q}_k \,|\, \boldsymbol{q}_{k-1}, \boldsymbol{\Theta}, \boldsymbol{q}_b)p(\boldsymbol{y}_{N_{t,o}} \,|\, \boldsymbol{q}_{k(N_{t,o})}, \boldsymbol{\Theta}) \prod_{k=k(N_{t,o})+1}^{N_t} p(\boldsymbol{q}_k \,|\, \boldsymbol{q}_{k-1}, \boldsymbol{\Theta}, \boldsymbol{q}_b). \end{aligned} \tag{3.39}$$

This expression can be evaluated sequentially in time to obtain an expression identical to the one obtained by the direct evaluation of (3.38). After the first observation ($\boldsymbol{y}_1$), we have:

$$p(\{\boldsymbol{q}_k\}_{k=1}^{k(1)}, \boldsymbol{\Theta}, \boldsymbol{q}_0, \boldsymbol{q}_b \,|\, \boldsymbol{y}_1) \propto p(\boldsymbol{q}_0)p(\boldsymbol{q}_b)p(\boldsymbol{\Theta}) \prod_{k=1}^{k(1)} p(\boldsymbol{q}_k \,|\, \boldsymbol{q}_{k-1}, \boldsymbol{\Theta}, \boldsymbol{q}_b)p(\boldsymbol{y}_1 \,|\, \boldsymbol{q}_{k(1)}, \boldsymbol{\Theta}). \tag{3.40}$$

After n observations ($n = 2, \ldots, N_{t,o}$), we have:

$$\begin{aligned} p(\{\boldsymbol{q}_k\}_{k=1}^{k(n)}, \boldsymbol{\Theta}, \boldsymbol{q}_0, \boldsymbol{q}_b \,|\, \{\boldsymbol{y}_j\}_{j=1}^{n}) \propto\ & p(\{\boldsymbol{q}_k\}_{k=1}^{k(n-1)}, \boldsymbol{\Theta}, \boldsymbol{q}_0, \boldsymbol{q}_b \,|\, \{\boldsymbol{y}_j\}_{j=1}^{n-1}) \\ & \prod_{k=k(n-1)+1}^{k(n)} p(\boldsymbol{q}_k \,|\, \boldsymbol{q}_{k-1}, \boldsymbol{\Theta}, \boldsymbol{q}_b)p(\boldsymbol{y}_n \,|\, \boldsymbol{q}_{k(n)}, \boldsymbol{\Theta}), \end{aligned} \tag{3.41}$$

[14] To simplify the notation, we write $p(\{\boldsymbol{q}_k\}_{k=1}^{N_t}, \boldsymbol{\Theta}, \boldsymbol{q}_0, \boldsymbol{q}_b) = p(\boldsymbol{q}_1, \cdots, \boldsymbol{q}_{N_t}, \boldsymbol{\Theta}, \boldsymbol{q}_0, \boldsymbol{q}_b)$.

and

$$p(\{\boldsymbol{q}_k\}_{k=1}^{N_t}, \boldsymbol{\Theta}, \boldsymbol{q}_0, \boldsymbol{q}_b \,|\, \{\boldsymbol{y}_j\}_{j=1}^{N_{t,o}}) \propto p(\{\boldsymbol{q}_k\}_{k=1}^{i(N_{t,o})}, \boldsymbol{\Theta}, \boldsymbol{q}_0, \boldsymbol{q}_b \,|\, \{\boldsymbol{y}_j\}_{j=1}^{N_{t,o}}) \prod_{k=k(N_{t,o})+1}^{N_t} p(\boldsymbol{q}_k \,|\, \boldsymbol{q}_{k-1}, \boldsymbol{\Theta}, \boldsymbol{q}_b). \tag{3.42}$$

These equations show that the joint conditional pdf of the model solution in the time interval $[t_1, t_{k(n)}]$ $(n = 2, \ldots, N_{t,o})$ is obtained using the prior information from the previous time interval $[t_1, t_{k(n-1)}]$ and the observation $\boldsymbol{y}_n$. This is the most general formulation of the state and parameter estimation problem using Bayesian statistics. It states that as long as the state model is a first order Markov process and the observation errors remain uncorrelated in time, a recursive formulation of processing the observations sequentially can be used for the Bayes' theorem. It has been claimed that for many DA problems, the sequential procedure is better posed than processing all the observations simultaneously as is done in variational formulations (Evensen, 2009).

3.2.1.2 Filtering problem

We now return to the filtering problem and discuss it in terms of the conditional distributions. In the subsequent discussion, we omit the distributions for the initial and boundary conditions. The task of filtering is to use the available posterior at the $(n-1)$-th observation time step $p(\boldsymbol{q}_{k(n-1)}, \boldsymbol{\Theta} \,|\, \{\boldsymbol{y}_j\}_{j=1}^{n-1})$ to find the *forecast* distribution $p(\boldsymbol{q}_{k(n)}, \boldsymbol{\Theta} \,|\, \{\boldsymbol{y}_j\}_{j=1}^{n-1})$, and the *analysis* distribution $p(\boldsymbol{q}_{k(n)}, \boldsymbol{\Theta} \,|\, \{\boldsymbol{y}_j\}_{j=1}^{n})$. From the Markov process assumption, the forecast distribution is obtained as (see Wikle and Berliner, 2007)

$$p(\boldsymbol{q}_{k(n)}, \boldsymbol{\Theta} \,|\, \{\boldsymbol{y}_j\}_{j=1}^{n-1}) = p(\boldsymbol{\Theta}) \int p(\boldsymbol{q}_{k(n)} \,|\, \boldsymbol{q}_{k(n)-1}, \boldsymbol{\Theta}) p(\boldsymbol{q}_{k(n)-1}, \boldsymbol{\Theta} \,|\, \{\boldsymbol{y}_j\}_{j=1}^{n-1}) \, \mathrm{d}\boldsymbol{q}_{k(n)-1}. \tag{3.43}$$

This forecast is used to obtain the analysis distribution by Bayes' rule as

$$\begin{aligned} p(\boldsymbol{q}_{k(n)}, \boldsymbol{\Theta} \,|\, \{\boldsymbol{y}_j\}_{j=1}^{n}) &= p(\boldsymbol{q}_{k(n)}, \boldsymbol{\Theta} \,|\, \boldsymbol{y}_n, \{\boldsymbol{y}_j\}_{j=1}^{n-1}) \\ &\propto p(\boldsymbol{y}_n \,|\, \boldsymbol{q}_{k(n)}, \boldsymbol{\Theta}) p(\boldsymbol{q}_{k(n)}, \boldsymbol{\Theta} \,|\, \{\boldsymbol{y}_j\}_{j=1}^{n-1}). \end{aligned} \tag{3.44}$$

The sequential alternating forecast and analysis is performed each time new observation is available at the discrete time steps $t_{k(n)}$ with $n = 1, \ldots, N_{t,o}$.

From the Bayesian point of view, DA leads to a complete knowledge of the posterior pdf, *i.e.* the conditional distribution of the state given the observations. Even though it is true that the pdf describes the probability of all possible states of the system, it is virtually impossible to determine the complete distribution and it does not make good targets for the estimation algorithm (Carrassi et al., 2018). So we seek instead an estimate of the statistical parameters of the distribution, such as its mean and/or variance (Evensen, 2009). Several statistical methods have been developed to give the optimal estimates, *e.g.* the minimum variance (MV) estimator provides the conditional mean of the states subject to the observations, and the maximum *a posteriori* (MAP) estimator provides the mode[15] of the conditional distribution.

Although the Bayesian approach is appealing, the large dimensions that are generally encountered in fluid flow problems make its application intractable. The problem dimension

[15] The mode is the peak of the distribution and gives the most probable state.

impacts the possibility to define and evolve the pdfs. In order to combat the computational challenge, it is a common practice to assume the uncertainties in each stochastic information – model states, parameters, observations and prior – to be Gaussian distributed. The Gaussian hypothesis leads to major simplification as it allows the pdfs to be generalized in terms of the mean and covariance matrices.

3.2.1.3 Kalman filter algorithm

The *Kalman filter* (KF) approach solves the state estimation problem introduced in Sec. 3.2.1.1 for the case of linear model operators and Gaussian error distributions (Carrassi et al., 2018). In the following, the model parameters are considered as fixed and, as such, only the state estimation is performed. We consider the state model given by the linear counterpart of (3.30) as[16]

$$\boldsymbol{q}_k = \boldsymbol{M}_{k:k-1}\boldsymbol{q}_{k-1} + \boldsymbol{\eta}_k, \quad \boldsymbol{\eta}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}_k). \tag{3.45}$$

$\boldsymbol{M}_{k:k-1} \in \mathbb{R}^{N_s \times N_s}$ is the linear model operator that maps the evolution of the state in time and gives the distribution $p(\boldsymbol{q}_k \,|\, \boldsymbol{q}_{k-1})$. The model error $\boldsymbol{\eta}_k$ is assumed to be uncorrelated in time and Gaussian distributed with zero mean and a time-dependent noise covariance matrix $\boldsymbol{Q}_k \in \mathbb{R}^{N_s \times N_s}$. Similarly, the observation is given by the linear counterpart of (3.31) as

$$\boldsymbol{y}_k^o = \boldsymbol{H}_k\boldsymbol{q}_k + \boldsymbol{\epsilon}_k^o, \quad \boldsymbol{\epsilon}_k^o \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}_k). \tag{3.46}$$

Here we have assumed, for simple notations and without loss of generality, that the observations are available at all the model integration steps, allowing the usage of the same time index k for both the variables. $\boldsymbol{H}_k \in \mathbb{R}^{N_o \times N_s}$ is the linear observation operator that maps the states to the measurements and gives the distribution $p(\boldsymbol{y}_k \,|\, \boldsymbol{q}_k)$. Similar to the model error, the observation error $\boldsymbol{\epsilon}_k^o$ is assumed to be uncorrelated in time and Gaussian distributed with zero mean and a time-dependent noise covariance matrix $\boldsymbol{R}_k \in \mathbb{R}^{N_o \times N_o}$. Typically, it is assumed that the model and observation noises are independent and mutually uncorrelated.

Now we introduce the conditional expectations for the *forecast* and *analysis* as $\boldsymbol{q}_k^f \equiv \mathbb{E}[\boldsymbol{q}_k^t | \{\boldsymbol{y}_j\}_{j=1}^{k-1}]$ and $\boldsymbol{q}_k^a \equiv \mathbb{E}[\boldsymbol{q}_k^t | \{\boldsymbol{y}_j\}_{j=1}^{k}]$, respectively. By extension, the conditional forecast and analysis error covariances are given by

$$\boldsymbol{P}_k^f = \mathbb{E}[(\boldsymbol{q}_k^t - \boldsymbol{q}_k^f)(\boldsymbol{q}_k^t - \boldsymbol{q}_k^f)^\top | \{\boldsymbol{y}_j\}_{j=1}^{k-1}] \in \mathbb{R}^{N_s \times N_s}, \tag{3.47}$$

and

$$\boldsymbol{P}_k^a = \mathbb{E}[(\boldsymbol{q}_k^t - \boldsymbol{q}_k^a)(\boldsymbol{q}_k^t - \boldsymbol{q}_k^a)^\top | \{\boldsymbol{y}_j\}_{j=1}^{k}] \in \mathbb{R}^{N_s \times N_s}. \tag{3.48}$$

It can be shown (see Wikle and Berliner, 2007, section 3.1) that the forecast distribution can be obtained as

$$\boldsymbol{q}_k | \{\boldsymbol{y}_j\}_{j=1}^{k-1} \sim \mathcal{N}(\boldsymbol{q}_k^f, \boldsymbol{P}_k^f), \tag{3.49}$$

where the mean is given as

$$\begin{aligned} \boldsymbol{q}_k^f &= \mathbb{E}[\boldsymbol{q}_k^t | \{\boldsymbol{y}_j\}_{j=1}^{k-1}] = \mathbb{E}[\mathbb{E}[\boldsymbol{q}_k^t | \boldsymbol{q}_{k-1}] | \{\boldsymbol{y}_j\}_{j=1}^{k-1}] \\ &= \mathbb{E}[\boldsymbol{M}_{k:k-1}\boldsymbol{q}_{k-1}^t | \{\boldsymbol{y}_j\}_{j=1}^{k-1}] = \boldsymbol{M}_{k:k-1}\boldsymbol{q}_{k-1}^a, \end{aligned} \tag{3.50}$$

[16] The symbol "$\sim$" is read as "is distributed as". The notation $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ refers to a multivariate Gaussian (or normal) distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$.

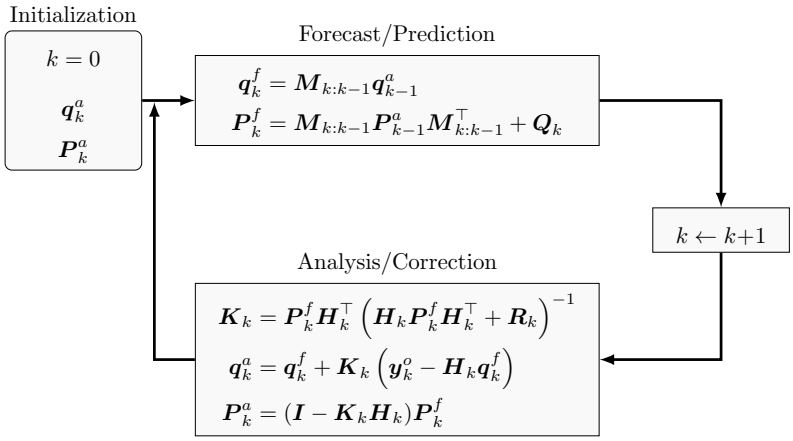


Figure 3.2: Kalman Filter algorithm. The initialization is made with the analysed state.

and the variance is given as

$$\begin{aligned}\boldsymbol{P}_k^f &= \mathrm{var}(\boldsymbol{q}_k^t|\{\boldsymbol{y}_j\}_{j=1}^{k-1}) = \mathbb{E}[\mathrm{var}(\boldsymbol{q}_k^t|\boldsymbol{q}_{k-1})|\{\boldsymbol{y}_j\}_{j=1}^{k-1}] + \mathrm{var}(\mathbb{E}[\boldsymbol{q}_k^t|\boldsymbol{q}_{k-1}]|\{\boldsymbol{y}_j\}_{j=1}^{k-1}) \\ &= \mathbb{E}[\boldsymbol{Q}_k|\{\boldsymbol{y}_j\}_{j=1}^{k-1}] + \mathrm{var}(\boldsymbol{M}_{k:k-1}\boldsymbol{q}_{k-1}^t|\{\boldsymbol{y}_j\}_{j=1}^{k-1}) = \boldsymbol{Q}_k + \boldsymbol{M}_{k:k-1}\boldsymbol{P}_{k-1}^a\boldsymbol{M}_{k:k-1}^\top.\end{aligned} \tag{3.51}$$

The posterior distribution of $\boldsymbol{q}_k|\{\boldsymbol{y}_j\}_{j=1}^k$ is derived in App. E. The analysis follows a normal distribution with the mean and variance given by

$$\begin{aligned}\boldsymbol{q}_k^a &= (\boldsymbol{H}_k^\top\boldsymbol{R}_k^{-1}\boldsymbol{H}_k + (\boldsymbol{P}_k^f)^{-1})^{-1}(\boldsymbol{H}_k^\top\boldsymbol{R}_k^{-1}\boldsymbol{y}_k + \boldsymbol{P}_k^f\boldsymbol{q}_k^f) \\ &= \boldsymbol{q}_k^f + \boldsymbol{K}_k(\boldsymbol{y}_k - \boldsymbol{H}_k\boldsymbol{q}_k^f),\end{aligned} \tag{3.52}$$

and

$$\begin{aligned}\boldsymbol{P}_k^a &= (\boldsymbol{H}_k^\top\boldsymbol{R}_k^{-1}\boldsymbol{H}_k + (\boldsymbol{P}_k^f)^{-1})^{-1} \\ &= (\boldsymbol{I}_{N_s} - \boldsymbol{K}_k\boldsymbol{H}_k)\boldsymbol{P}_k^f,\end{aligned} \tag{3.53}$$

where the Kalman gain is defined as

$$\boldsymbol{K}_k = \boldsymbol{P}_k^f\boldsymbol{H}_k^\top(\boldsymbol{H}_k\boldsymbol{P}_k^f\boldsymbol{H}_k^\top + \boldsymbol{R}_k)^{-1} \in \mathbb{R}^{N_s\times N_o}. \tag{3.54}$$

Therefore, from the model and observation operators $\boldsymbol{M}_{k:k-1}$ and $\boldsymbol{H}_k$, the covariance matrices $\boldsymbol{Q}_k$ and $\boldsymbol{R}_k$ ($k = 1, \ldots, N_t$), and from initial (background) information $\boldsymbol{q}_0^a = \boldsymbol{q}^b$, and $\boldsymbol{P}_0^a = \boldsymbol{P}^b$, the KF algorithm can be used to obtain sequential estimates of the state and associated covariance matrices. The KF algorithm is given in Alg. 3.1 and represented schematically in Fig. 3.2. The prediction-correction trajectory of the model state vector is shown in Fig. 3.3.

The KF algorithm is only optimal when the assumptions of Gaussian statistics and linear operators are valid. However, when the models are non-Gaussian and/or the operators are nonlinear, the mean value and error covariances are not sufficient to describe the pdfs and matrices entering the estimation problem. Consequently, the forecast and analysis distributions given by (3.43) and (3.44) cannot be obtained explicitly. The KF is no longer optimal and can easily fail the estimation process. Moreover, the direct use of the KF approach for high dimensional systems is delicate, if not impossible (inversion of very large matrices for example). These limitations necessitate the use of another approach for the DA problem. A traditional approach is to use tangent linear models for local linearization of the nonlinear evolution

Algorithm 3.1: KF algorithm

Input: For $k = 1, \ldots, N_t$: linearized forward model $\boldsymbol{M}_{k:k-1}$, model error covariance $\boldsymbol{Q}_k$, linearized observation operator $\boldsymbol{H}_j$, observation error covariance $\boldsymbol{R}_j$.
For $j = 1, \ldots, N_{t,o}$: noisy observations $\{\boldsymbol{y}_j^o\}$.

Output: Updated model states $\{\boldsymbol{q}_k^a\}$, $\forall k = 1, \ldots, N_t$

begin

Initialize with estimates for $\{\boldsymbol{q}_0^a\}$ and $\{\boldsymbol{P}_0^a\}$ and $j = 1$

for $k \leftarrow 1$ **to** N_t **do**

Project the state forward

$\boldsymbol{q}_k^f = \boldsymbol{M}_{k:k-1}\boldsymbol{q}_{k-1}^a$

Project the error covariance ahead

$\boldsymbol{P}_k^f = \boldsymbol{Q}_k + \boldsymbol{M}_{k:k-1}\boldsymbol{P}_{k-1}^a\boldsymbol{M}_{k:k-1}^\top$

if $j \leq N_{t,o}$ ***and*** $t_k = t_{k(j)}$ **then**

Compute the Kalman gain

$\boldsymbol{K}_k = \boldsymbol{P}_k^f\boldsymbol{H}_k^\top(\boldsymbol{H}_k\boldsymbol{P}_k^f\boldsymbol{H}_k^\top + \boldsymbol{R}_k)^{-1}$

Update the estimate with measurement

$\boldsymbol{q}_k^a = \boldsymbol{q}_k^f + \boldsymbol{K}_k(\boldsymbol{y}_{k(j)}^o - \boldsymbol{H}_k\boldsymbol{q}_k^f)$

Update the error covariance

$\boldsymbol{P}_k^a = (\boldsymbol{I}_{N_s} - \boldsymbol{K}_k\boldsymbol{H}_k)\boldsymbol{P}_k^f$

$j \leftarrow j + 1$

else

Skip the state update

$\boldsymbol{q}_k^a \leftarrow \boldsymbol{q}_k^f$

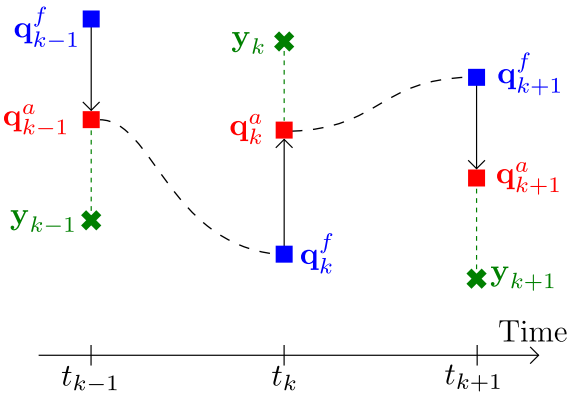


Figure 3.3: Schematic of the KF data assimilation method. The dashed line represents the trajectory of the state vector propagated by the model from one time step to the next. The corrections of the propagated values obtained at the observation times is indicated by black arrows.

and observation operators. However, this method, known as *extended Kalman filter* in the sequential case, works well only when the problem is moderately nonlinear and non-Gaussian. Alternatively, Monte Carlo (MC) based sequential assimilation methods have been developed to

deal with the nonlinearities and non-Gaussian statistics. The ensemble Kalman filter (EnKF) is one such approach (Evensen, 1994) where the pdf is described by an ensemble of time-dependent states. In the subsequent sections, EnKF and its extension will be discussed.

3.2.2 Ensemble Kalman filter (EnKF)

The *ensemble Kalman filter* (EnKF) uses the Monte Carlo (MC) method to empirically represent the statistical characteristics of the estimator while still retaining the nonlinear forward model (3.30) for the state evolution. In particular, the covariance matrix of forecast error $\boldsymbol{P}_k^f$ is not derived from the covariance matrix of analysis error $\boldsymbol{P}_{k-1}^a$, but rather estimated from the propagation of a finite *ensemble* of samples (particles) generated using a forced random walk of the parameters of interest.

The EnKF can be seen as a reduced-order KF as it only handles first two moments (mean and covariance) of the error statistics which loosely mimics the Gaussian filter. Generally, due to this truncation of statistics, the EnKF does not solve the Bayesian filtering problem. However, EnKF has been shown to provide a good approximate algorithm to the filtering problem.

In this section, the following nonlinear, time-discrete model of the dynamical system and the observation equation are considered

$$\begin{cases} \boldsymbol{q}_k^t = \mathcal{M}_{k:k-1}(\boldsymbol{q}_{k-1}) + \boldsymbol{\eta}_k, \quad \boldsymbol{\eta}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}_k), \\ \boldsymbol{y}_k^o = \mathcal{H}_k(\boldsymbol{q}_k) + \boldsymbol{\epsilon}_k^o, \quad \boldsymbol{\epsilon}_k^o \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}_k). \end{cases} \tag{3.55}$$

The model parameters $\boldsymbol{\Theta}$ are not considered because they are assumed to be known in this section. The model error $\boldsymbol{\eta}_k$ and observation error $\boldsymbol{\epsilon}_k^o$ are assumed to be unbiased and mutually and temporally uncorrelated. The EnKF is performed sequentially by following a forecast-analysis (or prediction-correction) scheme.

3.2.2.1 Forecast step

During the prediction step, an ensemble of forecasted states is constructed by propagating each of the N_e members (particles) of the ensemble with the evolution model in (3.55). Each particle in the state-space evolves independently as

$$\boldsymbol{q}_k^{f,(n)} = \mathcal{M}_{k:k-1}(\boldsymbol{q}_{k-1}^{a,(n)}) + \boldsymbol{\eta}_k^{(n)}, \quad \boldsymbol{\eta}_k^{(n)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}_k), \tag{3.56}$$

where $n = 1, \ldots, N_e$ is the sample index. The vector $\boldsymbol{q}_k^{f,(n)}$ represents the n-th member of the ensemble of forecast states at the instant t_k and $\boldsymbol{\eta}_k^{(n)}$ is the associated model error. Note that in the cases where no model error is assumed, the samples are evolved without the additive noise term in (3.56). The vector $\boldsymbol{q}_{k-1}^{a,(n)}$ is the corresponding member of the ensemble of corrected (analyzed) states at a previous time t_{k-1}. The forecast error covariance matrix $\boldsymbol{P}_k^f$ can be estimated empirically. The unbiased empirical estimator for $\boldsymbol{P}_k^f$, $\boldsymbol{P}_k^{f,e}$ is obtained as

$$\boldsymbol{P}_k^{f,e} = \frac{1}{N_e - 1} \sum_{n=1}^{N_e} (\boldsymbol{q}_k^{f,(n)} - \overline{\boldsymbol{q}}_k^f)(\boldsymbol{q}_k^{f,(n)} - \overline{\boldsymbol{q}}_k^f)^\top, \tag{3.57}$$

where the empirical mean $\overline{\boldsymbol{q}}_k^f$ of the ensemble of forecast state is obtained as

$$\overline{\boldsymbol{q}}_k^f = \frac{1}{N_e}\sum_{n=1}^{N_e} \boldsymbol{q}_k^{f,(n)}. \tag{3.58}$$

To avoid the phenomena of coalescence of the ensemble members, it is essential to perturb the observations for each member (data randomization). We will also empirically estimate the covariance matrix of the observation error used to obtain the Kalman gain. This way, the error statistics is continually updated to guarantee sufficient diffusion of the ensemble members and avoid divergence of the algorithm. The random observations are built to be centered around the actual observation with covariance $\boldsymbol{R}_k$ and given as

$$\boldsymbol{y}_k^{o,(n)} = \boldsymbol{y}_k^o + \boldsymbol{\epsilon}_k^{o,(n)}, \quad \boldsymbol{\epsilon}_k^{o,(n)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}_k). \tag{3.59}$$

The vector $\boldsymbol{y}_k^{o,(n)}$ represents the n-th member of the observation ensemble at the instant t_k and $\boldsymbol{\epsilon}_k^{o,(n)}$ denotes the associated perturbation. The unbiased (zero-mean) empirical estimator $\boldsymbol{R}_k^e$ of the observation error covariance matrix is given as

$$\boldsymbol{R}_k^e = \mathbb{E}[\boldsymbol{\epsilon}_k^{o,(n)}(\boldsymbol{\epsilon}_k^{o,(n)})^\top] = \frac{1}{N_e-1}\sum_{n=1}^{N_e} \boldsymbol{\epsilon}_k^{o,(n)}(\boldsymbol{\epsilon}_k^{o,(n)})^\top. \tag{3.60}$$

As the number of MC elements tends to infinity, the empirical estimator $\boldsymbol{R}_k^e$ tends to the full-rank observation error covariance matrix $\boldsymbol{R}_k$. In practice, the estimation $\boldsymbol{R}_k^e$ is used to counter the computational cost of using the full-rank matrix.

3.2.2.2 Analysis step

The correction step consists of updating each member available from the ensemble of forecasted states using the actual observation. For this, a linear correction is applied as follows

$$\begin{aligned} \boldsymbol{q}_k^{a,(n)} &= \boldsymbol{q}_k^{f,(n)} + \boldsymbol{K}_k^e(\boldsymbol{y}_k^{o,(n)} - \boldsymbol{y}_k^{f,(n)}) \\ &= \boldsymbol{q}_k^{f,(n)} + \boldsymbol{K}_k^e(\boldsymbol{y}_k^{o} + \boldsymbol{\epsilon}_k^{o,(n)} - \mathcal{H}_k(\boldsymbol{q}_k^{f,(n)})). \end{aligned} \tag{3.61}$$

The vector $\boldsymbol{q}_k^{a,(n)}$ represents the n-th member of the ensemble of analyzed state at the instant t_k. To mimic the Kalman gain of KF given by (3.54), the ensemble Kalman gain $\boldsymbol{K}_k^e$ is expressed in terms of the ensemble covariances as follows

$$\boldsymbol{K}_k^e = \boldsymbol{P}_k^{f,e}\boldsymbol{H}_k^\top(\boldsymbol{H}_k\boldsymbol{P}_k^{f,e}\boldsymbol{H}_k^\top + \boldsymbol{R}_k)^{-1}. \tag{3.62}$$

This terminates the analysis step. We note that the linearized evolution model has not been used in the algorithm which makes it useful in a significantly nonlinear regime.

Though not necessary for the algorithm, the availability of the ensemble of analyzed states $\boldsymbol{q}_k^{a,(n)}$ makes it possible to obtain the corresponding unbiased empirical estimator of the correlation matrix as

$$\boldsymbol{P}_k^{a,e} = \frac{1}{N_e-1}\sum_{n=1}^{N_e}(\boldsymbol{q}_k^{a,(n)} - \overline{\boldsymbol{q}}_k^a)(\boldsymbol{q}_k^{a,(n)} - \overline{\boldsymbol{q}}_k^a)^\top. \tag{3.63}$$

where $\overline{\boldsymbol{q}}_k^a$ is the empirical mean over the ensemble of analyzed states, given as

$$\overline{\boldsymbol{q}}_k^a = \frac{1}{N_e}\sum_{n=1}^{N_e} \boldsymbol{q}_k^{a,(n)}. \tag{3.64}$$

This can be used as an optional diagnostic in the scheme.

3.2.2.3 Kalman gain for nonlinear observation operator

We notice that the Kalman gain (3.62) requires the tangent linear model $\boldsymbol{H}_k$ of the observation operator $\mathcal{H}_k$ for the terms $\boldsymbol{P}_k^{f,e}\boldsymbol{H}_k^\top$ and $\boldsymbol{H}_k\boldsymbol{P}_k^{f,e}\boldsymbol{H}_k^\top$. In principle, it is therefore necessary to determine $\boldsymbol{H}_k$ at each time step k, which is expensive. However, these two terms can be estimated by using the full nonlinear observation operator (Asch et al., 2016). For this, the following assumption – called the *secant method* – is made

$$\boldsymbol{H}_k(\boldsymbol{q}_k^{f,(n)} - \overline{\boldsymbol{q}}_k^f) \simeq \boldsymbol{y}_k^{f,(n)} - \overline{\boldsymbol{y}}_k^f, \tag{3.65}$$

where

$$\boldsymbol{y}_k^{f,(n)} = \mathcal{H}_k(\boldsymbol{q}_k^{f,(n)}), \quad \text{and} \quad \overline{\boldsymbol{y}}_k^f = \frac{1}{N_e}\sum_{n=1}^{N_e} \boldsymbol{y}_k^{f,(n)}. \tag{3.66}$$

The approximation (3.65) can now be used to replace the nonlinear operator in the expression for the Kalman gain. The term $\boldsymbol{P}_k^{f,e}\boldsymbol{H}_k^\top$ is given in terms of its sample estimate $\boldsymbol{P}_{qy,k}^{f,e}$, a cross-covariance of the forecast state $\boldsymbol{q}_k^{f,(n)}$ and forecast observation $\boldsymbol{y}_k^{f,(n)}$,

$$\begin{aligned}\boldsymbol{P}_k^{f,e}\boldsymbol{H}_k^\top &= \frac{1}{N_e-1}\sum_{n=1}^{N_e}(\boldsymbol{q}_k^{f,(n)} - \overline{\boldsymbol{q}}_k^f)[\boldsymbol{H}_k(\boldsymbol{q}_k^{f,(n)} - \overline{\boldsymbol{q}}_k^f)]^\top \\ &\simeq \frac{1}{N_e-1}\sum_{n=1}^{N_e}(\boldsymbol{q}_k^{f,(n)} - \overline{\boldsymbol{q}}_k^f)(\boldsymbol{y}_k^{f,(n)} - \overline{\boldsymbol{y}}_k^f)^\top := \boldsymbol{P}_{qy,k}^{f,e}.\end{aligned} \tag{3.67}$$

Similarly, the term $\boldsymbol{H}_k\boldsymbol{P}_k^{f,e}\boldsymbol{H}_k^\top$ is given in terms of its sample estimate $\boldsymbol{P}_{yy,k}^{f,e}$, a covariance of the forecast observation $\boldsymbol{y}_k^{f,(n)}$,

$$\begin{aligned}\boldsymbol{H}_k\boldsymbol{P}_k^{f,e}\boldsymbol{H}_k^\top &= \frac{1}{N_e-1}\sum_{n=1}^{N_e}[\boldsymbol{H}_k(\boldsymbol{q}_k^{f,(n)} - \overline{\boldsymbol{q}}_k^f)][\boldsymbol{H}_k(\boldsymbol{q}_k^{f,(n)} - \overline{\boldsymbol{q}}_k^f)]^\top \\ &\simeq \frac{1}{N_e-1}\sum_{n=1}^{N_e}(\boldsymbol{y}_k^{f,(n)} - \overline{\boldsymbol{y}}_k^f)(\boldsymbol{y}_k^{f,(n)} - \overline{\boldsymbol{y}}_k^f)^\top := \boldsymbol{P}_{yy,k}^{f,e}.\end{aligned} \tag{3.68}$$

Substituting the terms from (3.67) and (3.68) in (3.62) gives a fully ensemble based Kalman gain as

$$\boldsymbol{K}_k^e = \boldsymbol{P}_{qy,k}^{f,e}(\boldsymbol{P}_{yy,k}^{f,e} + \boldsymbol{R}_k^e)^{-1}. \tag{3.69}$$

When the number of measurements is greater than the number of ensemble members, the inverse term in the calculation of the Kalman gain may become singular. In this case, a pseudo-inverse based on the singular value decomposition can be employed. However, the advantage of this representation particularly resides in the fact that both the linearization and the calculation of the covariance matrix of the prediction error are not required, therefore

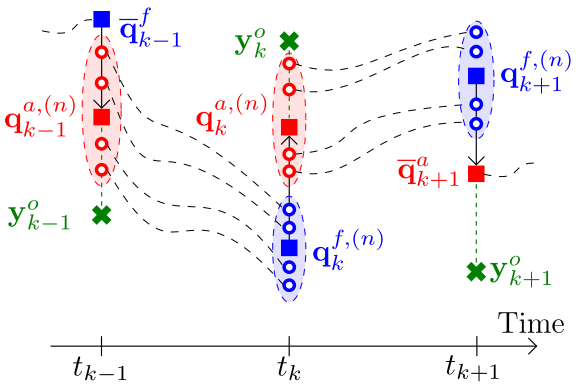


Figure 3.4: Schematic of the EnKF data assimilation method. The filled symbols represent the mean values and the hollow circles represent the ensemble members; the ensemble set itself is indicated by the shaded region around the respective mean values of the analyzed (red) and forecast (blue) states. The dashed lines represent the trajectories of the particles in the state ensemble propagated by the model from one time step to the next (*e.g.* $\mathcal{M}_{k:k-1}$ propagates the state from t_{k-1} to t_k). The corrections of the propagated values obtained at the observation times is indicated by black arrows.

resulting in a considerable reduction in the computation cost and requirement of storage space. The EnKF algorithm is given in Alg. 3.2 and the prediction-correction trajectory of the model state vector is shown in Fig. 3.4.

3.2.3 Dual-ensemble Kalman filter (Dual-EnKF)

In this section, we extend the previous EnKF framework by considering the case of a parameterized model given by (3.30). If the *true* model parameters $\boldsymbol{\Theta}$ are known, then the results presented in Sec. 3.2.2 can be applied immediately for the state estimation. On the other hand, in practice, we frequently encounter cases where the parameters are unknown or imprecise. Then the goal of DA is to estimate both the state variables $\boldsymbol{q}_k$ and the parameters $\boldsymbol{\Theta}_k$ given random observation $\boldsymbol{y}^o_k$. One approach is the *joint estimation* where the state and parameter vectors are concatenated into a single joint state vector in a step known as state augmentation. The drawback of such strategy is that, by increasing the number of unknowns (*i.e.* a combination of model states and parameters), the degree of freedom in the system increases and makes the estimation unstable and intractable, especially in the nonlinear dynamical model. Another possible approach is the *dual estimation* where at each iteration the filtering is alternatively applied to estimate the state and the parameters. More precisely, in the first stage, a correction of the model parameters is obtained from the analyzed state at the previous time state, and in the second stage, the new state is evaluated from the corrected parameters. This sequential double prediction-correction scheme, formally known as Dual-EnKF (Moradkhani et al., 2005), is based on the mechanism of evolution of the traditional EnKF. The Dual-EnKF algorithm is discussed in the following section.

3.2.3.1 Prediction-correction of the parameters vector

For the extension of the EnKF to the dual estimation problem, the parameters $\boldsymbol{\Theta}_k$ are treated as random variables, in the same way as it was done for the state $\boldsymbol{q}_k$. Samples of the prediction (forecast) parameters are generated from the available (analyzed) parameters $\boldsymbol{\Theta}^{a,(n)}_k$ by adding

Algorithm 3.2: Stochastic EnKF algorithm

Input: For $k = 1, \ldots, N_t$: forward model $\mathcal{M}_{k:k-1}$, observation operator $\mathcal{H}_k$, observation error covariance $\boldsymbol{R}_k$.
For $j = 1, \ldots, N_{t,o}$: noisy observations $\{\boldsymbol{y}_j^o\}$.

Output: Updated model states $\{\boldsymbol{q}_k^{a,(n)}\}$, $\forall k = 1, \ldots, N_t$

begin

Initialize the ensemble of analyses $\{\boldsymbol{q}_0^{a,(n)}\}$; $n = 1, \ldots, N_\mathrm{e}$, and $j = 1$

for $k \leftarrow 1$ **to** N_t **do**

Compute the ensemble forecast; $n = 1, \ldots, N_e$

$\boldsymbol{q}_k^{f,(n)} = \mathcal{M}_{k:k-1}(\boldsymbol{q}_{k-1}^{a,(n)})$

if $j \leq N_{t,o}$ ***and*** $t_k = t_{k(j)}$ **then**

Draw a statistically consistent observation set; $n = 1, \ldots, N_e$

$$\boldsymbol{y}_k^{o,(n)} = \boldsymbol{y}_{k(j)}^o + \boldsymbol{\epsilon}_k^{o,(n)}, \quad \boldsymbol{\epsilon}_k^{o,(n)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}_k)$$

(Optional) Compute the estimated observation error covariance

$$\boldsymbol{R}_k^e = \frac{1}{N_e - 1} \sum_{n=1}^{N_e} \boldsymbol{\epsilon}_k^{o,(n)} (\boldsymbol{\epsilon}_k^{o,(n)})^\top$$

Compute the model counterparts of the observation set; $n = 1, \ldots, N_e$

$\boldsymbol{y}_k^{f,(n)} = \mathcal{H}_k(\boldsymbol{q}_k^{f,(n)})$

Compute the ensemble means

$$\overline{\boldsymbol{q}}_k^f = \frac{1}{N_e} \sum_{n=1}^{N_e} \boldsymbol{q}_k^{f,(n)} \;\; ; \;\; \overline{\boldsymbol{y}}_k^f = \frac{1}{N_e} \sum_{n=1}^{N_e} \boldsymbol{y}_k^{f,(n)}$$

Compute the estimated forecast error covariances

$$\boldsymbol{P}_{qy,k}^{f,e} = \frac{1}{N_e - 1} \sum_{n=1}^{N_e} (\boldsymbol{q}_k^{f,(n)} - \overline{\boldsymbol{q}}_k^f)(\boldsymbol{y}_k^{f,(n)} - \overline{\boldsymbol{y}}_k^f)^\top$$

$$\boldsymbol{P}_{yy,k}^{f,e} = \frac{1}{N_e - 1} \sum_{n=1}^{N_e} (\boldsymbol{y}_k^{f,(n)} - \overline{\boldsymbol{y}}_k^f)(\boldsymbol{y}_k^{f,(n)} - \overline{\boldsymbol{y}}_k^f)^\top$$

Compute the Kalman gain

$\boldsymbol{K}_k^e = \boldsymbol{P}_{qy,k}^{f,e} (\boldsymbol{P}_{yy,k}^{f,e} + \boldsymbol{R}_k)^{-1}$, or

$\boldsymbol{K}_k^e = \boldsymbol{P}_{qy,k}^{f,e} (\boldsymbol{P}_{yy,k}^{f,e} + \boldsymbol{R}_k^e)^{-1}$

Update the ensemble; $n = 1, \ldots, N_e$

$\boldsymbol{q}_k^{a,(n)} = \boldsymbol{q}_k^{f,(n)} + \boldsymbol{K}_k^e (\boldsymbol{y}_k^{o,(n)} - \boldsymbol{y}_k^{f,(n)})$

$j \leftarrow j + 1$

else

Skip the ensemble update

$\boldsymbol{q}_k^{a,(n)} \leftarrow \boldsymbol{q}_k^{f,(n)}$

an artificial random walk, *i.e.*

$$\boldsymbol{\Theta}_k^{f,(n)} = \boldsymbol{\Theta}_{k-1}^{a,(n)} + \boldsymbol{\xi}_k^{(n)}, \quad \boldsymbol{\xi}_k^{(n)} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{C}_k), \tag{3.70}$$

where $n = 1, \ldots, N_e$. The additive random perturbation $\boldsymbol{\xi}_k^{(n)}$ is Gaussian distributed with zero mean and covariance $\boldsymbol{C}_k$. However, as the random perturbation is not governed by any physical law, it poses the challenges of over-dispersion of the parameter samples and the loss of information during the propagation of ensemble, even when the parameters are considered to be fixed. We then follow the work of Moradkhani et al. (2005) and introduce *kernel smoothing* and *shrinkage* to tackle the loss of information and over-diffused posteriors. The parameter forecast (3.70) is modified to give

$$\boldsymbol{\Theta}_k^{f,(n)} = s\boldsymbol{\Theta}_{k-1}^{a,(n)} + (1-s)\overline{\boldsymbol{\Theta}}_{k-1}^{a} + \boldsymbol{\xi}_k^{(n)}, \quad \boldsymbol{\xi}_k^{(n)} \sim \mathcal{N}(\mathbf{0}, h^2\boldsymbol{C}_k), \tag{3.71}$$

where

$$\overline{\boldsymbol{\Theta}}_{k-1}^{a} = \frac{1}{N_e}\sum_{n=1}^{N_e} \boldsymbol{\Theta}_{k-1}^{a,(n)}. \tag{3.72}$$

Here h is the smoothing or variance reducing parameter and s is the shrinkage parameter. The two parameters are related as $h = \sqrt{1-s^2}$, with $s = (3\delta - 1)/(2\delta)$ where $\delta \in (0, 1]$ is a constant.

The subsequent steps focus on updating the predicted ensemble of parameters by assimilating the observations similar to the state update of EnKF. For this, the ensemble of predicted state is first obtained by using the predicted parameters for the model $\mathcal{M}_{k:k-1}$ as

$$\boldsymbol{q}_k^{f,(n)} = \mathcal{M}_{k:k-1}(\boldsymbol{q}_{k-1}^{a,(n)}, \boldsymbol{\Theta}_k^{f,(n)}). \tag{3.73}$$

The ensemble of predicted states is used to obtain the ensemble of forecast observations using the observation operator $\mathcal{H}_k$ as

$$\boldsymbol{y}_k^{f,(n)} = \mathcal{H}_k(\boldsymbol{q}_k^{f,(n)}). \tag{3.74}$$

Now, similar to the EnKF algorithm, an ensemble of random observations is built centered around the actual observation $\boldsymbol{y}_k^o$ using an additive noise as

$$\boldsymbol{y}_k^{o,(n)} = \boldsymbol{y}_k^o + \boldsymbol{\epsilon}_k^{o,(n)}, \tag{3.75}$$

with the observation error covariance matrix given as

$$\boldsymbol{R}_k^e = \frac{1}{N_e - 1}\sum_{n=1}^{N_e} \boldsymbol{\epsilon}_k^{o,(n)} (\boldsymbol{\epsilon}_k^{o,(n)})^\top. \tag{3.76}$$

Given the random observations, the update of the predicted ensemble of parameters $\boldsymbol{\Theta}_k^{f,(n)}$ is obtained by the Kalman analysis step which reads as

$$\boldsymbol{\Theta}_k^{a,(n)} = \boldsymbol{\Theta}_k^{f,(n)} + \boldsymbol{K}_k^{\Theta,e}(\boldsymbol{y}_k^{o,(n)} - \boldsymbol{y}_k^{f,(n)}). \tag{3.77}$$

The expression for the Kalman gain of the parameter update $\boldsymbol{K}_k^{\Theta,e}$ follows the definition of EnKF (3.69) and is represented in terms of the error covariances as

$$\boldsymbol{K}_k^{\Theta,e} = \boldsymbol{P}_{\Theta y,k}^{f,e}(\boldsymbol{P}_{yy,k}^{f,e} + \boldsymbol{R}_k^e)^{-1}. \tag{3.78}$$

The unbiased estimators of the observation correlation matrix $\boldsymbol{P}_{yy,k}^{f,e}$ and cross-correlation matrix between the observation and parameters $\boldsymbol{P}_{\theta y,k}^{f,e}$ are obtained as

$$\boldsymbol{P}_{yy,k}^{f,e} = \frac{1}{N_e - 1} \sum_{n=1}^{N_e} (\boldsymbol{y}_k^{f,(n)} - \overline{\boldsymbol{y}}_k^f)(\boldsymbol{y}_k^{f,(n)} - \overline{\boldsymbol{y}}_k^f)^\top, \tag{3.79}$$

and

$$\boldsymbol{P}_{\theta y,k}^{f,e} = \frac{1}{N_e - 1} \sum_{n=1}^{N_e} (\boldsymbol{\Theta}_k^{f,(n)} - \overline{\boldsymbol{\Theta}}_k^f)(\boldsymbol{y}_k^{f,(n)} - \overline{\boldsymbol{y}}_k^f)^\top, \tag{3.80}$$

where the mean values over the ensemble are obtained as

$$\overline{\boldsymbol{y}}_k^f = \frac{1}{N_e} \sum_{n=1}^{N_e} \boldsymbol{y}_k^{f,(n)}, \quad \text{and} \quad \overline{\boldsymbol{\Theta}}_k^f = \frac{1}{N_e} \sum_{n=1}^{N_e} \boldsymbol{\Theta}_k^{f,(n)}. \tag{3.81}$$

This terminates the parameter update part of the dual estimation algorithm.

3.2.3.2 Prediction-correction of the state variable

Once the predicted parameters have been updated, the second filter is implemented to estimate the state variable at the instant t_k by considering the parameters $\boldsymbol{\Theta}_k^{a,(n)}$ as fixed. In other words, the parameters just corrected by the first filter are trusted. The following prediction-correction steps are identical to the classical EnKF. In the first step, the predicted ensemble of states is built starting from the ensemble of known (analyzed) states at the instant t_{k-1} and the corrected ensemble of parameters at the instant t_k to give

$$\boldsymbol{q}_k^{f,(n)} = \mathcal{M}_{k:k-1}(\boldsymbol{q}_{k-1}^{a,(n)}, \boldsymbol{\Theta}_k^{a,(n)}). \tag{3.82}$$

For the second time, the forecast ensemble of observations is obtained as

$$\boldsymbol{y}_k^{f,(n)} = \mathcal{H}_k(\boldsymbol{q}_k^{f,(n)}). \tag{3.83}$$

At this stage, the observations are again considered as random variables and the covariance matrix of the observation error used to obtain the Kalman gain is also empirically estimated. Next, the update of the predicted ensemble of the state is realized by Kalman approach as

$$\boldsymbol{q}_k^{a,(n)} = \boldsymbol{q}_k^{f,(n)} + \boldsymbol{K}_k^e(\boldsymbol{y}_k^{o,(n)} - \boldsymbol{y}_k^{f,(n)}). \tag{3.84}$$

The Kalman gain for correcting the state trajectories is given in terms of the error covariances as

$$\boldsymbol{K}_k^e = \boldsymbol{P}_{qy,k}^{f,e}(\boldsymbol{P}_{yy,k}^{f,e} + \boldsymbol{R}_k^e)^{-1}. \tag{3.85}$$

In this expression, $\boldsymbol{P}_{qy,k}^{f,e}$ and $\boldsymbol{P}_{yy,k}^{f,e}$ denote the unbiased empirical estimators of the cross-correlation matrix between the state and the observations, and the correlation matrix between the observations, respectively. They are explicitly given as

$$\boldsymbol{P}_{yy,k}^{f,e} = \frac{1}{N_e - 1} \sum_{n=1}^{N_e} (\boldsymbol{y}_k^{f,(n)} - \overline{\boldsymbol{y}}_k^f)(\boldsymbol{y}_k^{f,(n)} - \overline{\boldsymbol{y}}_k^f)^\top, \tag{3.86}$$

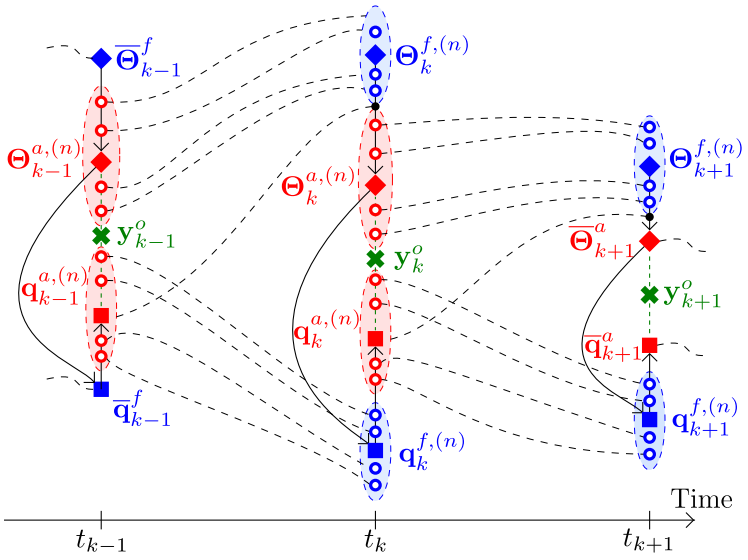


Figure 3.5: Schematic of the Dual-EnKF data assimilation method. Refer to the caption of Fig. 3.4 for details. Apart from the state vectors, the dashed lines here also represent the propagation of the parameter ensemble from one time step to the next. The use of analyzed parameter in the state forecast at the current time step is also indicated by black arrows.

and

$$\boldsymbol{P}^{f,e}_{qy,k} = \frac{1}{N_e - 1}\sum_{n=1}^{N_e}(\boldsymbol{q}^{f,(n)}_k - \overline{\boldsymbol{q}}^{f}_k)(\boldsymbol{y}^{f,(n)}_k - \overline{\boldsymbol{y}}^{f}_k)^{\top}, \tag{3.87}$$

where the mean values over the ensemble are calculated as

$$\overline{\boldsymbol{y}}^{f}_k = \frac{1}{N_e}\sum_{n=1}^{N_e}\boldsymbol{y}^{f,(n)}_k, \quad \text{and} \quad \overline{\boldsymbol{q}}^{f}_k = \frac{1}{N_e}\sum_{n=1}^{N_e}\boldsymbol{q}^{f,(n)}_k. \tag{3.88}$$

This terminates the state update part of the dual estimation algorithm.

This double prediction-correction scheme is sequentially repeated over time. Note that the step of prediction-correction of the system state can precede the operation on the parameters without consequence on the quality of the estimation. We can now return to the POD-ROM and present the use of the Dual-EnKF method to correctly identify the model parameters. In the DA problem, the temporal coefficients $a_i^{\mathrm{ROM}}(t_k)$ form the state vector $\boldsymbol{q}_k$. The POD-ROM in (2.79) serves as the dynamical model $\mathcal{M}_{k:k-1}$ with the full parameters $\boldsymbol{\Theta} = [\boldsymbol{\theta}\ \boldsymbol{\theta}_s]^{\mathbb{T}}$. The initial background condition $\boldsymbol{q}^b$ is same as the POD temporal coefficients. Finally, the observations vector $\boldsymbol{y}^o_k$ can be built, say, using velocity measurements in the computational domain. The Dual-EnKF algorithm is given in Alg. 3.3 and the prediction-correction trajectory of the model state and parameters vector is shown in Fig. 3.5.

Algorithm 3.3: Dual-EnKF algorithm

Input: For $k = 1, \ldots, N_t$: forward model $\mathcal{M}_{k:k-1}$, observation operator $\mathcal{H}_k$, observation error covariance $\boldsymbol{R}_k$.
For $j = 1, \ldots, N_{t,o}$: noisy observations $\{\boldsymbol{y}_j^o\}$.

Output: Updated model parameters $\{\boldsymbol{\Theta}_k^{a,(n)}\}$ and states $\{\boldsymbol{q}_k^{a,(n)}\}$, $\forall k = 1, \ldots, N_t$

begin

Initialize the ensemble of analyses $\{\boldsymbol{\Theta}_0^{a,(n)}\}$ and $\{\boldsymbol{q}_0^{a,(n)}\}$; $n = 1, \ldots, N_\mathsf{e}$, and $j = 1$

for $k \leftarrow 1$ **to** N_t **do**

Compute the parameter forecast; $n = 1, \ldots, N_e$

$$\boldsymbol{\Theta}_k^{f,(n)} = s\boldsymbol{\Theta}_{k-1}^{a,(n)} + (1-s)\overline{\boldsymbol{\Theta}}_{k-1}^{a} + \boldsymbol{\xi}_k^{(n)}, \quad \boldsymbol{\xi}_k^{(n)} \sim \mathcal{N}(\boldsymbol{0}, h^2\boldsymbol{C}_k)$$

if $j \leq N_{t,o}$ ***and*** $t_k = t_{k(j)}$ **then**

Draw a statistically consistent observation set; $n = 1, \ldots, N_e$

$$\boldsymbol{y}_k^{o,(n)} = \boldsymbol{y}_{k(j)}^o + \boldsymbol{\epsilon}_k^{o,(n)}, \quad \boldsymbol{\epsilon}_k^{o,(n)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}_k)$$

$$\boldsymbol{R}_k^e = \frac{1}{N_e - 1} \sum_{n=1}^{N_e} \boldsymbol{\epsilon}_k^{o,(n)} (\boldsymbol{\epsilon}_k^{o,(n)})^\top$$

Compute the model counterparts of the observation set; $n = 1, \ldots, N_e$

$$\boldsymbol{q}_k^{f,(n)} = \mathcal{M}_{k:k-1}(\boldsymbol{q}_{k-1}^{a,(n)}, \boldsymbol{\Theta}_k^{f,(n)})$$

$$\boldsymbol{y}_k^{f,(n)} = \mathcal{H}_k(\boldsymbol{q}_k^{f,(n)})$$

Update the parameter ensemble; $n = 1, \ldots, N_e$

$$\boldsymbol{K}_k^{\Theta,e} = \boldsymbol{P}_{\Theta y,k}^{f,e} (\boldsymbol{P}_{yy,k}^{f,e} + \boldsymbol{R}_k^e)^{-1}$$

$$\boldsymbol{\Theta}_k^{a,(n)} = \boldsymbol{\Theta}_k^{f,(n)} + \boldsymbol{K}_k^{\Theta,e} (\boldsymbol{y}_k^{o,(n)} - \boldsymbol{y}_k^{f,(n)})$$

else

Skip the parameter ensemble update

$$\boldsymbol{\Theta}_k^{a,(n)} \leftarrow \boldsymbol{\Theta}_k^{f,(n)}$$

Compute the state ensemble forecast using the updated parameter; $n = 1, \ldots, N_e$

$$\boldsymbol{q}_k^{f,(n)} = \mathcal{M}_{k:k-1}(\boldsymbol{q}_{k-1}^{a,(n)}, \boldsymbol{\Theta}_k^{a,(n)})$$

if $j \leq N_{t,o}$ ***and*** $t_k = t_{k(j)}$ **then**

Compute the model counterparts of the observation set; $n = 1, \ldots, N_e$

$$\boldsymbol{y}_k^{f,(n)} = \mathcal{H}_k(\boldsymbol{q}_k^{f,(n)})$$

Update the state ensemble; $n = 1, \ldots, N_e$

$$\boldsymbol{K}_k^{e} = \boldsymbol{P}_{qy,k}^{f,e} (\boldsymbol{P}_{yy,k}^{f,e} + \boldsymbol{R}_k^e)^{-1}$$

$$\boldsymbol{q}_k^{a,(n)} = \boldsymbol{q}_k^{f,(n)} + \boldsymbol{K}_k^{e} (\boldsymbol{y}_k^{o,(n)} - \boldsymbol{y}_k^{f,(n)})$$

$j \leftarrow j + 1$

else

Skip the state ensemble update

$$\boldsymbol{q}_k^{a,(n)} \leftarrow \boldsymbol{q}_k^{f,(n)}$$

3.3 Deep neural network modeling

Model reduction by Galerkin projection of the Navier-Stokes equations onto a POD basis (see Sec. 2.1) has the advantage to be strongly connected to the governing equations issued from physical modeling, facilitating the model interpretation. However, this method is *intrusive*, requiring human expertise to develop models from a working simulation. Also, as discussed in Sec. 2.2.3, models obtained from the modal decomposition methods are often not robust when applied to control-oriented problems. This lack of robustness is mainly caused by the fact that the entire envelope of the flow dynamics cannot be captured accurately by a few dominant spatial modes. For this reason, there is a need to develop a *non-intrusive* model, *i.e.* a dynamical model that can be derived solely from data, without any prior information of the underlying physics (Casenave et al., 2015). In the following, this model will be used to approximate the POD reduced-order model and extended for problems involving parametric flow conditions. Optionally, the framework will be extended to reproduce the dynamics over the control parameter domain, by using a database containing information for different values of the control parameter.

In the recent past, machine learning (ML) algorithms have been used to develop non-intrusive approaches for dimensionality reduction and reduced-order modeling of different physical systems. Brunton and Kutz (2019) gives a broad discussion on the application of data-driven methods for dynamical systems with limited or no access to the full-order model operators. With advances in simulation capabilities and experimental techniques, fluid dynamics is becoming a data-rich field, which is amenable to ML algorithms. A recent review article by Brunton, Noack, and Koumoutsakos (2019) gives a comprehensive overview of the application of machine learning to fluid mechanics problems. Several studies have been performed to apply data-driven techniques in the prediction of dynamical systems. San and Maulik (2018) recently proposed a supervised machine learning framework for the closure and stabilization of a highly truncated POD-ROM. A multistep neural network was recently proposed by Raissi et al. (2018). This method combines classical multistep time-stepping schemes with nonlinear function approximators, namely deep neural networks (DNN), for the identification of nonlinear dynamical systems. The constraint of limited data was addressed by physics-informed neural network (PINN), first presented by Raissi et al. (2019), where the ML based approach uses a neural network augmented with the knowledge of the governing equations. For data organized as sequence, the long short-term memory (LSTM) variant of the recurrent neural network (RNN) has been used to model the temporal dynamics of turbulence in a ROM framework (Mohan and Gaitonde, 2018).

In this study, to bypass the Galerkin projection step used in POD-ROM, deep neural network (DNN) is used to derive a regression model. A DNN is an artificial neural network (ANN) with multiple layers between the input and output layers. ANN is a class of machine learning methods that mathematically represents the biological neural networks. In simple terms, an ANN can be used to model a nonlinear mapping from desired set of features to the corresponding targets. For the *training* phase of the ANN, it is necessary to have at disposal a sufficiently large amount of data governed by the underlying governing equations. After learning, the artificial neural network is described by a set of transfer functions which replicate the mean behavior (in a given sense) of the underlying dynamics to which the input features belonged. ANNs have been used, for example, as regression models to identify non-intrusive ROM of the nonlinear Poisson and steady-state incompressible Navier-Stokes equations in Hesthaven and Ubbiali (2018). Unlike reduced-order models based on Galerkin projection, an ANN is not physically interpretable. However, the ANN construction allows to represent nonlinear relationships that cannot be expressed explicitly in functional form. This minimizes

the model uncertainties in the forward simulations. The development of robust non-intrusive ROM based on ANN is an active field of research (Lui and Wolf, 2019; Wang, Hesthaven, et al., 2019).

In this manuscript, two strategies, recently presented in the literature, are combined to develop a non-intrusive ROM based on neural networks. The first strategy corresponds to the case where the DNN approximates the map between the time and parameter values as inputs and the projection coefficients obtained from the projection of the high-fidelity model on a low-dimensional space as outputs (Wang, Hesthaven, et al., 2019). The second strategy corresponds to a DNN architecture which uses the values of the states at previous time steps to predict the time evolution of the residuals of the solution (Pawar et al., 2019). The current algorithm – named NN-ROM for Deep Neural Network Reduced-Order Model – is therefore presented as an alternative to the Galerkin projection based POD-ROM for estimating the projection coefficients directly from the information of the time, parameter and past coefficient values.

In the next sections, the main elements of the NN-ROM are presented. In Sec. 3.3.1, the regression method that employs a multistep, residual-based, parametrized DNN approach for the iterative prediction of the POD coefficients is presented. The training of the DNN model and its subsequent use for prediction are discussed in Sec. 3.3.2.

3.3.1 Regression via neural network

Previous studies (Chen, Rubanova, et al., 2018, for instance) have demonstrated that deep neural networks (DNNs) are capable of approximating nonlinear dynamical systems. This motivates the use of a surrogate model obtained by DNN as an alternative to the POD-ROM (2.79). In the literature, different strategies have already been proposed to iteratively predict the temporal dynamics of POD coefficients via a DNN. In this manuscript, the *multistep*, *residual-based* and *parametrized* approaches of Wang, Hesthaven, et al. (2019) and Pawar et al. (2019) have been combined.

With the multistep approach (Raissi et al., 2018), the aim is to use the evolution of a set of POD coefficients observed at several previous time steps to predict a future state of the system. This approach is similar to classical multistep family of time-stepping schemes from numerical analysis (*e.g.* Adams-Bashforth, Adams Moulton, BDF). The use of multiple steps allows to incorporate the memory effects in learning the temporal dynamics and helps to tackle nonlinear dynamical features.

In the residual approach (Qin et al., 2019; San, Maulik, and Ahmed, 2019), instead of the solution trajectory given by the sequential POD coefficients $(\boldsymbol{a}(t_k), \boldsymbol{a}(t_{k+1}), \ldots)$, their increments, given in terms of the residuals $\boldsymbol{r}(t_{k+1}) = \boldsymbol{a}(t_{k+1}) - \boldsymbol{a}(t_k)$, are used to train the DNN model. Although both formulations are mathematically equivalent, this simple transformation has been shown to yield relatively more stable and accurate results.

Finally, in the parametrized approach, some parameter value defining the flow configuration is also introduced as input of the DNN model to estimate POD coefficients at new values of the parameter. The POD method leads to an orthogonal basis that approximately spans the state solution space of the model for a specific parameter configuration. A deviation from the reference parametric configuration requires the construction of new bases since the original ROM is in general no longer accurate. However, if states depend continuously on flow parameters, the POD basis determined for one parameter configuration can approximate the solution space in a local vicinity of the parameter (Moosavi et al., 2015).

Before proceeding with the regression setup, it is noteworthy that the DNN approach classically includes an *offline* and an *online* stage. In the offline stage, the regression model

is trained in a supervised learning paradigm by using high-fidelity POD data. The objective is to determine approximate maps between the inputs (past projection coefficients, time, and parameter value) and residuals (with respect to the projection coefficients) at the subsequent time steps. In the online stage, the learned regression models are rapidly evaluated to obtain the predictions (projection coefficients for the new sets of inputs).

Combining the above concepts, the DNN model $\boldsymbol{f}^{\mathrm{NN}}$ can be introduced, which maps the available sequence of input features *i.e.* time steps t_k, parameters $\boldsymbol{\mu}_j$, and POD coefficients $\boldsymbol{a}^{\mathrm{POD}}(t_k)$ to the target given in terms of the residuals $\boldsymbol{r}(t_{k+1}) = \boldsymbol{a}^{\mathrm{POD}}(t_{k+1}) - \boldsymbol{a}^{\mathrm{POD}}(t_k)$. Let $\boldsymbol{W}$ and $\boldsymbol{b}$ be the weights and biases of the model, the discrete form of $\boldsymbol{f}^{\mathrm{NN}}$ is given by

$$\boldsymbol{r}(t_{k+1};\boldsymbol{\mu}_j) = \boldsymbol{f}^{\mathrm{NN}}\left(\boldsymbol{a}^{\mathrm{POD}}(t_{k-p+1};\boldsymbol{\mu}_j),\ldots,\boldsymbol{a}^{\mathrm{POD}}(t_k;\boldsymbol{\mu}_j);\boldsymbol{W},\boldsymbol{b}\right)+\boldsymbol{\epsilon}_{k+1}, \tag{3.89}$$

$$\text{for } p=1,\ldots,N_t^{\mathrm{Train}}-1,\quad k=p-1,\ldots,N_t^{\mathrm{Train}}-2 \quad \text{and} \quad j=1,\ldots,N_p^{\mathrm{Train}},$$

where $\boldsymbol{\epsilon}_{k+1}$ is the modeling error at time step $k+1$. After training (offline) the DNN, the learned weights and biases ($\boldsymbol{W}^*$ and $\boldsymbol{b}^*$) are used (online) to predict sequentially the temporal dynamics as

$$\boldsymbol{a}^{\mathrm{NN}}(t_{k+1};\boldsymbol{\mu}_j) = \boldsymbol{a}^{\mathrm{NN}}(t_k;\boldsymbol{\mu}_j)+\boldsymbol{f}^{\mathrm{NN}}(\boldsymbol{a}^{\mathrm{NN}}(t_{k-p+1};\boldsymbol{\mu}_j),\ldots,\boldsymbol{a}^{\mathrm{NN}}(t_k;\boldsymbol{\mu}_j);\boldsymbol{W}^*,\boldsymbol{b}^*), \tag{3.90}$$

$$\text{for } p=1,\ldots,N_t^{\mathrm{Pred}}-1,\quad k=p-1,\ldots,N_t^{\mathrm{Pred}}-1 \quad \text{and} \quad j=1,\ldots,N_p^{\mathrm{Pred}}.$$

In (3.89) and (3.90), the superscripts "Train" and "Pred" are used to distinguish the variables pertaining to the training and prediction regimes. Here, $\boldsymbol{a}^{\mathrm{NN}}(t_k;\boldsymbol{\mu}_j) \in \mathbb{R}^{N_r\times 1}$ represents the POD projection coefficients obtained from the DNN framework at time instant t_k for the parameters $\boldsymbol{\mu}_j \in \mathbb{R}^{N_\mu\times 1}$ characterizing the dynamics. $\boldsymbol{a}^{\mathrm{POD}}(t_k;\boldsymbol{\mu}_j) \in \mathbb{R}^{N_r\times 1}$ represents the coefficients obtained from high-fidelity dataset used to train the DNN model. The model here is trained using data from N_t^{Train} time steps and used to predict the evolution for N_t^{Pred} time steps. N_r is the number of modes considered in the reduced space (similar to N_{Gal} in the Galerkin method (2.52)). N_μ is the number of parameters characterizing the dynamics, *e.g.* $N_\mu = 2$ in a forced convection problem which can be uniquely defined by a Reynolds number and a Prandtl number, $\boldsymbol{\mu} = [Re\ Pr]^\top$, as parameters. N_p^{Train} is the number of parameter configurations available for training, *e.g.* $\{(Re_1,Pr_1),\ldots,(Re_{N_p^{\mathrm{Train}}},Pr_{N_p^{\mathrm{Train}}})\}$. N_p^{Pred} is similarly defined as the number of target parameter configurations for prediction. The multistep model permits the use of the coefficient values from previous p time steps before the time instant t_k to estimate the values at the next time step t_{k+1}.

The input and output data are arranged in matrices that will be referred during the subsequent training of the regression model. The computed POD coefficients $a_i^{\mathrm{POD}}(t_k;\boldsymbol{\mu}_j)$, for all $i=1,\ldots,N_r$; $k=0,\ldots,N_t^{\mathrm{Train}}$ and $j=1,\ldots,N_p^{\mathrm{Train}}$, are used to define the matrix of

input features $\boldsymbol{A} \in \mathbb{R}^{(N_t^{\text{Train}}-p)N_p^{\text{Train}} \times (N_r \times p + N_\mu + 1)}$ given as

$$\boldsymbol{A} = \begin{bmatrix} t_p & \boldsymbol{\mu}_1^\top & \widetilde{\boldsymbol{a}}^{\text{POD}}(t_{p-1};\boldsymbol{\mu}_1)^\top \\ t_{p+1} & \boldsymbol{\mu}_1^\top & \widetilde{\boldsymbol{a}}^{\text{POD}}(t_p;\boldsymbol{\mu}_1)^\top \\ \vdots & \vdots & \vdots \\ t_{N_t^{\text{Train}}-1} & \boldsymbol{\mu}_1^\top & \widetilde{\boldsymbol{a}}^{\text{POD}}(t_{N_t^{\text{Train}}-2};\boldsymbol{\mu}_1)^\top \\ \vdots & \vdots & \vdots \\ t_{k+1} & \boldsymbol{\mu}_j^\top & \widetilde{\boldsymbol{a}}^{\text{POD}}(t_k;\boldsymbol{\mu}_j)^\top \\ \vdots & \vdots & \vdots \\ t_{N_t^{\text{Train}}-1} & \boldsymbol{\mu}_{N_p^{\text{Train}}}^\top & \widetilde{\boldsymbol{a}}^{\text{POD}}(t_{N_t^{\text{Train}}-2};\boldsymbol{\mu}_{N_p^{\text{Train}}})^\top \end{bmatrix}, \tag{3.91}$$

where $\boldsymbol{\mu}_j = [\mu_{1,j}\ \mu_{2,j}\ \dots\ \mu_{N_\mu,j}]^\top$ is the parameter vector and $\widetilde{\boldsymbol{a}}^{\text{POD}} \in \mathbb{R}^{(N_r \times p)\times 1}$ is the vector of POD coefficients spanning a time lag interval of $[t_{k-p+1}, t_k]$ *i.e.*

$$\begin{aligned}\widetilde{\boldsymbol{a}}^{\text{POD}}(t_k;\boldsymbol{\mu}_j) = [a_1^{\text{POD}}(t_{k-p+1};\boldsymbol{\mu}_j)\ a_2^{\text{POD}}(t_{k-p+1};\boldsymbol{\mu}_j) \cdots a_{N_r}^{\text{POD}}(t_{k-p+1};\boldsymbol{\mu}_j) \\ \cdots a_i^{\text{POD}}(t_{k'};\boldsymbol{\mu}_j) \cdots a_{N_r}^{\text{POD}}(t_k;\boldsymbol{\mu}_j)]^\top.\end{aligned} \tag{3.92}$$

In addition, we introduce $\boldsymbol{R}$ the matrix of residuals $r_i(t_{k+1};\boldsymbol{\mu}_j) = a_i^{\text{POD}}(t_{k+1};\boldsymbol{\mu}_j) - a_i^{\text{POD}}(t_k;\boldsymbol{\mu}_j)$ as

$$\boldsymbol{R} = \begin{bmatrix} r_1(t_p;\boldsymbol{\mu}_1) & r_2(t_p;\boldsymbol{\mu}_1) & \cdots & r_{N_r}(t_p;\boldsymbol{\mu}_1) \\ r_1(t_{p+1};\boldsymbol{\mu}_1) & r_2(t_{p+1};\boldsymbol{\mu}_1) & \cdots & r_{N_r}(t_{p+1};\boldsymbol{\mu}_1) \\ \vdots & \vdots & \ddots & \vdots \\ r_1(t_{N_t^{\text{Train}}-1};\boldsymbol{\mu}_1) & r_2(t_{N_t^{\text{Train}}-1};\boldsymbol{\mu}_1) & \cdots & r_{N_r}(t_{N_t^{\text{Train}}-1};\boldsymbol{\mu}_1) \\ \vdots & \vdots & \ddots & \vdots \\ r_1(t_{k+1};\boldsymbol{\mu}_j) & r_2(t_{k+1};\boldsymbol{\mu}_j) & \cdots & r_{N_r}(t_{k+1};\boldsymbol{\mu}_j) \\ \vdots & \vdots & \ddots & \vdots \\ r_1(t_{N_t^{\text{Train}}-1};\boldsymbol{\mu}_{N_p^{\text{Train}}}) & r_2(t_{N_t^{\text{Train}}-1};\boldsymbol{\mu}_{N_p^{\text{Train}}}) & \cdots & r_{N_r}(t_{N_t^{\text{Train}}-1};\boldsymbol{\mu}_{N_p^{\text{Train}}}) \end{bmatrix}. \tag{3.93}$$

The regression problem (3.89) can be rewritten as

$$\boldsymbol{R} \approx \widehat{\boldsymbol{R}} = \boldsymbol{f}^{\text{NN}}(\boldsymbol{A};\boldsymbol{W},\boldsymbol{b}) \in \mathbb{R}^{(N_t^{\text{Train}}-p)N_p^{\text{Train}} \times N_r}, \tag{3.94}$$

where we denote the output of the neural network as $\widehat{\boldsymbol{R}}$ in order to distinguish it from $\boldsymbol{R}$. Each row of the input features in the matrix $\boldsymbol{A}$, *i.e.* current time instant, relevant parameters, and available POD coefficients at previous p time steps, corresponds to the target row in the matrix $\boldsymbol{R}$.

The DNN model (3.90) can be considered as a one-step numerical integrator equivalent to the POD-ROM (2.79). Note that this integrator is "exact" in time (*i.e.* no temporal errors with respect to discretization, order of approximation, etc.) in the sense that the only error appears from the neural network approximation of the model operators defining the governing equations. This framework provides an equation-free or non-intrusive regression modeling alternative to the Galerkin projection based POD-ROM. It presents the advantage to be purely data-driven which enables the reduction of the uncertainties associated with the model-form.

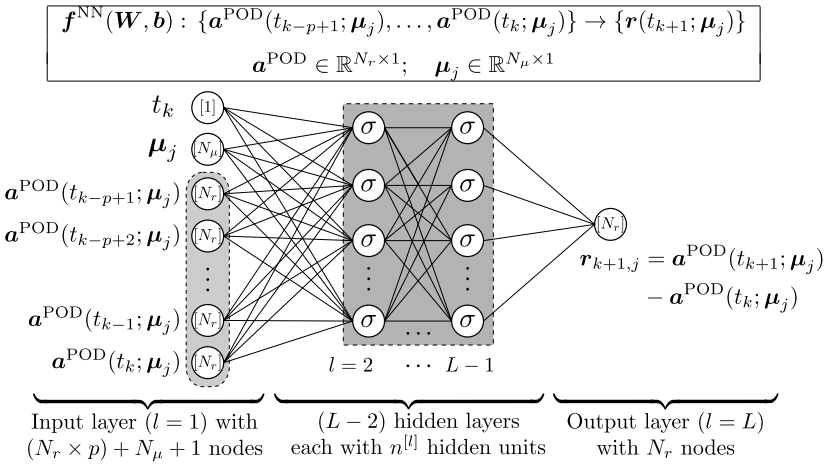


Figure 3.6: DNN architecture of the non-intrusive ROM $\boldsymbol{f}^{\mathrm{NN}}$. In the regression model, the training data (offline stage) consist of the time (t), parameters ($\boldsymbol{\mu}$) and POD modal coefficients ($\boldsymbol{a}^{\mathrm{POD}}(t;\boldsymbol{\mu})$) as inputs, and the residuals ($\boldsymbol{r}(t;\boldsymbol{\mu})$) as outputs. The DNN contains $L-2$ hidden layers each with $n^{[l]}$ hidden units. For simplicity, only one node is shown for each input and output in vector form and the corresponding actual number of nodes is indicated within square brackets. For the hidden layers, σ is the activation function associated with each unit.

3.3.2 DNN ROM training and prediction

This section provides a brief introduction to the backpropagation algorithm which is a supervised learning method for DNN frameworks. The principle of the backpropagation approach is to model a given function by modifying internal weightings of input signals to produce an expected output signal. The system is trained using a supervised learning method, where the error between the system's output and a known expected output is presented to the system and used to optimize its internal state using gradient-based methods. The trained model is used to recursively obtain sequential updates of the state of the dynamical system. The training is performed using the open-source machine learning library TensorFlow (Abadi et al., 2016).

The principle steps of the backpropagation algorithm are described. The construction of the deep neural network and the initialization of the training variables are discussed in Sec. 3.3.2.1. The propagation of the input features through the neural network and the objective function used for the optimization of the training variables are considered in Sec. 3.3.2.2. The backward propagation of the objective function through the neural network for the gradient calculation is discussed in Sec. 3.3.2.3. A gradient descent-based optimization algorithm used to update the training variables is presented in Sec. 3.3.2.4. Finally, the application of the feedforward network as a model for online predictions is described in Sec. 3.3.2.5.

3.3.2.1 DNN architecture and initialization

Backpropagation requires a network structure with one or more densely connected layers, *i.e.* each layer is fully connected to the next layer. Such a DNN architecture used for the offline training is shown in Fig. 3.6. This neural network consists of L layers composed of predefined number of nodes (also called neurons). The first layer is called the input

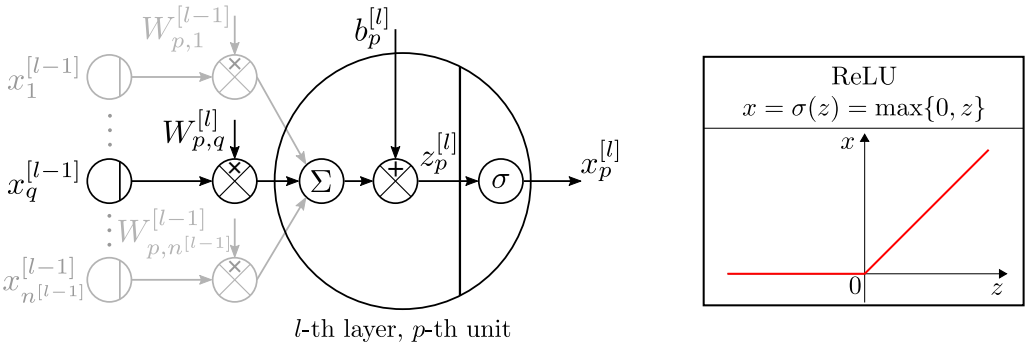


Figure 3.7: Illustration of the sequence of operations performed by a generic neural network node $p \in [1, n^{[l]}]$ in the hidden layer $l \in [2, L-1]$ of a DNN. This network consists of two units connected in series - one unit receives the weighted inputs and applies the bias, and the next one applies the activation function and produces an output. The rectified linear unit (ReLU) used as the activation function σ is shown in the inset.

layer, with $n^{[1]} = (N_r \times p) + N_\mu + 1$ nodes which correspond to the current time instant $t_k \in \mathbb{R}^1$, the relevant parameter characterizing the dynamics $\boldsymbol{\mu}_j \in \mathbb{R}^{N_\mu \times 1}$, and the available POD coefficients at previous p time steps $\boldsymbol{a}^{\text{POD}}(t_{k'}; \boldsymbol{\mu}_j) \in \mathbb{R}^{N_r \times 1}$, for all $k' = k - p + 1, \ldots, k$. The last layer is called the output layer, with $n^{[L]} = N_r$ nodes corresponding to the target residual vector $\boldsymbol{r}(t_{k'}; \boldsymbol{\mu}_j) \in \mathbb{R}^{N_r \times 1}$. The remaining $(L-2)$ layers are called hidden layers, each one consisting of $n^{[l]}$ hidden units $(l = 2, \ldots, L-1)$. Note that the number of nodes $n^{[1]}$ and $n^{[L]}$ in the input and output layers are assigned vis-à-vis the number of columns in the input feature matrix $\boldsymbol{A}$ (3.91) and the target residual matrix $\boldsymbol{R}$ (3.93), respectively.

The offline training is carried out by minimizing the error between the inputs and targets to determine a set of best regression parameters of the DNN model. This procedure is also known as supervised learning where pairs of inputs and targets are utilized for gradient based optimization. The best regression parameters obtained from optimization are the linear weights $\boldsymbol{W}^{[l]}$ and biases $\boldsymbol{b}^{[l]}$ associated with the neural network nodes in the hidden layers. The weights are a set of coefficients associated with the inputs that are combined at a node. These weights either amplify or dampen the input and thereby assign the significance to the input with respect to the output that the DNN is trying to learn. In addition, the nodes have a bias for each input to the node. The product of the inputs and weights, also known as weighted node inputs, and the bias are summed. The result is passed through a node's predefined *activation function* (σ) to obtain the node output. The sequence of operations pertaining to a generic node in the hidden layer, accepting a input vector $\boldsymbol{x}^{[l-1]} \in \mathbb{R}^{n^{[l-1]} \times 1}$, is given as

$$z^{[l]}_p = \sum_{q=1}^{n^{[l-1]}} W^{[l]}_{p,q} x^{[l-1]}_q + b^{[l]}_p, \tag{3.95}$$

$$x^{[l]}_p = \sigma(z^{[l]}_p). \tag{3.96}$$

This sequence is also illustrated in Fig. 3.7. Here, the weight associated with an input $x^{[l-1]}_q$ from the node $q \in [1, n^{[l-1]}]$ in the layer $l-1$ to the node $p \in [1, n^{[l]}]$ in the layer l is denoted as $W^{[l]}_{p,q}$. The corresponding bias associated with the node is simply represented

as $b_p^{[l]}$. The weights and biases pertaining to each hidden layer can be stored in dictionaries as $\boldsymbol{W} \coloneqq \{\boldsymbol{W}^{[2]}, \ldots, \boldsymbol{W}^{[L-1]}\}$, with $\boldsymbol{W}^{[l]} \in \mathbb{R}^{n^{[l]} \times n^{[l-1]}}$, and $\boldsymbol{b} \coloneqq \{\boldsymbol{b}^{[2]}, \ldots, \boldsymbol{b}^{[L-1]}\}$, with $\boldsymbol{b}^{[l]} \in \mathbb{R}^{n^{[l]} \times 1}$, which together form the model training variables $\boldsymbol{\Theta} \coloneqq \{\boldsymbol{W}, \boldsymbol{b}\}$. The affine transformation in (3.95) can also be rewritten in matrix form as

$$\boldsymbol{z}^{[l]} = \boldsymbol{W}^{[l]} \boldsymbol{x}^{[l-1]} + \boldsymbol{b}^{[l]}. \tag{3.97}$$

The subsequent activation $\boldsymbol{x}^{[l]} = \sigma(\boldsymbol{z}^{[l]})$ is applied element-wise. Note that the output of the last layer does not pass through the activation function, thus $\boldsymbol{x}^{[L]} = \boldsymbol{z}^{[L]}$. Learning the weights and biases of the nodes enables the DNN to capture the underlying relationship between the targets and inputs in the training dataset. These parameters will be used later to predict targets for inputs that do not belong to the training set.

An activation function is a fixed, nonlinear function applied to the output of the linear transformation (3.97). The activation function is key to the functioning of a DNN, as without it, the sole linear transformations in (3.95) would render the whole neural network replaceable by an equivalent linear function in terms of the inputs. In this work, the rectified linear unit (ReLU) is used. This function has been successfully implemented in modern neural networks. The ReLU is defined by the nonlinear transformation $\sigma(z) = \max\{0, z\}$ as depicted in Fig. 3.7. This unit corresponds to a piecewise linear function with two linear pieces. After Goodfellow et al. (2016), these units are easy to optimize with gradient-based methods because of their similarity with linear functions. Indeed, the only difference between a linear unit and a ReLU is that the latter outputs zero across half its domain. This makes the gradients through the unit remain large as the derivative of the rectifying operation is equal to 1 where the unit is active. The gradients are also consistent as the second derivative of the rectifying operation is 0 almost everywhere. Therefore, like linear models, the rectified unit provides gradient direction far more useful for learning than it would be with other activation functions that may introduce second-order effects. Although the ReLU function is not differentiable at $z = 0$, the gradient algorithm still performs sufficiently well as the learning algorithm does not aim to determine the local minimum of the cost function, but rather to reduce its value significantly. It has also been shown that piece-wise linear units can compute highly complex and structured functions (Montúfar et al., 2014), thus enabling ReLU as a universal function approximator.

The DNN training algorithm is usually iterative in nature and thus requires the specification of the initial value of the training variables from where iterations can start. Moreover, most algorithms are strongly affected by the choice of initialization. The initial point can determine whether the algorithm converges or not and the speed of convergence if it does (Goodfellow et al., 2016). One heuristic is to initialize the weights of a fully connected layer with $n^{[l-1]}$ inputs and $n^{[l]}$ outputs by sampling each weight from a uniform distribution given as

$$W_{p,q}^{[l]} \sim U\left(-\sqrt{\frac{6}{n^{[l-1]} + n^{[l]}}}, \sqrt{\frac{6}{n^{[l-1]} + n^{[l]}}}\right). \tag{3.98}$$

This method known as normalized initialization was suggested by Glorot and Bengio (2010). This initialization is designed to compromise between the goal of initializing all

layers to have the same activation variance and the goal of initializing all layers to have the same gradient variance. On the other hand, the bias may be assigned such as to avoid causing too much saturation (functions becoming flat) at initialization, causing complete loss of gradient through the saturated units. For this reason, the bias of a ReLU hidden unit is generally set to a small value, such as 0.1 rather than 0.

Also as part of initializing the inputs for training, a scaling of the features is performed. As the input is made of non-homogeneous features, *i.e.* time t, parameters $\boldsymbol{\mu}$ and POD coefficients $a^{\mathrm{POD}}(t)$, the individual scale and distribution can be different for each variable. Differences in the scales across input variables may increase the difficulty to train the DNN. For instance, large values in the input can result in a model that learns corresponding large weight values. Such a model with large weight values is often unstable. It may lead to poor performance during learning and sensitivity to input values, hence resulting in higher generalization[17] error. Therefore, it is recommended to scale the input and output variables before training the neural network. Normalization is one of the ways of rescaling the data from the original range, so that all values are within the range of 0 and 1. Normalization requires the knowledge of the minimum and maximum of the observable values in the input. The row-wise normalization of input matrix (3.91) is performed as

$$\boldsymbol{A}_{:,i} \leftarrow \frac{\boldsymbol{A}_{:,i} - \min(\boldsymbol{A}_{:,i})}{\max(\boldsymbol{A}_{:,i}) - \min(\boldsymbol{A}_{:,i})}, \tag{3.99}$$

to obtain the normalized input features. The application of normalization has been demonstrated to accelerate the training (Ioffe and Szegedy, 2015).

3.3.2.2 Forward propagation

The forward propagation calculates the output residual values $\widehat{\boldsymbol{R}}$ from the DNN by propagating the input features $\boldsymbol{A}$ forward through the hidden layers. The *feedforward network* defines the mapping $\boldsymbol{f}^{\mathrm{NN}}(\boldsymbol{\Theta})\colon \boldsymbol{A} \rightarrow \widehat{\boldsymbol{R}}$ in (3.94), where $\boldsymbol{\Theta} = \{\boldsymbol{W}, \boldsymbol{b}\}$ is the dictionary of training variables of the DNN. There are no feedback connections in which the information flows from the outputs of the model back into itself. During the training, forward propagation generates the predictions that are compared with the actual residuals $\boldsymbol{R}$ in order to tune the training variables $\boldsymbol{\Theta}$.

In the supervised learning paradigm, the forward propagation can also be seen as a mapping of the training variables $\boldsymbol{\Theta}$ to a loss function $\mathcal{L}(\widehat{\boldsymbol{R}}, \boldsymbol{R})$ associated with an input-target training pair $(\boldsymbol{A}, \boldsymbol{R})$. The loss is generally formulated using small subsets of the input features and target data called *minibatch*. These minibatches are disjoint subsets of the training dataset that are randomly selected. From the input features matrix $\boldsymbol{A}$ defined in (3.91), we introduce the minibatches $\{\boldsymbol{A}^{(j)} \in \mathbb{R}^{N_b \times (N_r \times p + N_\mu + 1)} \,|\, j = 1, \ldots, N_{\mathrm{mb}}\}$, which are subsets of $\boldsymbol{A}$ of size $N_b < (N_t^{\mathrm{Train}} - p) N_p^{\mathrm{Train}}$. Here, N_{mb} is the total number of minibatches. Similarly, the corresponding minibatches of output $\{\boldsymbol{R}^{(j)} \in \mathbb{R}^{N_b \times N_r} \,|\, j = 1, \ldots, N_{\mathrm{mb}}\}$ are also drawn from the target matrix $\boldsymbol{R}$ defined in (3.93). Training on minibatches increases the number of optimization steps by a factor of N_{mb}. However, it has been shown that most optimization algorithms converge much faster in terms of the overall computation time, if the approximate estimates of

[17] *i.e.* expected value of the error on a new input

the gradient are rapidly computed over minibatches, as compared to when the exact gradients are computed over the entire training set (Goodfellow et al., 2016).

In neural networks, the loss function is often defined as the mean squared error between the DNN prediction and the training values. At the end of the forward propagation, the loss function is evaluated over a given minibatch j as

$$\begin{aligned}\mathcal{L}(\widehat{\boldsymbol{R}}^{(j)}, \boldsymbol{R}^{(j)}) &= \mathbb{E}[\|\widehat{\boldsymbol{R}}^{(j)} - \boldsymbol{R}^{(j)}\|^2] \\ &= \frac{1}{2N_b}\sum_{i=1}^{N_r}\sum_{k=1}^{N_b}(\widehat{R}_{k,i}^{(j)} - R_{k,i}^{(j)})^2, \quad \forall j = 1, \ldots, N_{\text{mb}}.\end{aligned} \tag{3.100}$$

This standard loss function forms one part of the objective function. The other part corresponds to a *regularization* factor defined in terms of the norm of the training variables as

$$\Omega(\boldsymbol{W}) = \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} = \frac{1}{2}\sum_{l=1}^{L}\sum_{q=1}^{n^{[l-1]}}\sum_{p=1}^{n^{[l]}}(W_{p,q}^{[l]})^2, \tag{3.101}$$

where $\boldsymbol{w}$ is the vector of all the weights contained in the dictionary $\boldsymbol{W}$.

Using the loss function (3.100) along with the penalty (3.101), the L^2 regularized cost function (or objective function) associated with the minibatch j is obtained as

$$\mathcal{C}^{(j)} = \mathcal{L}(\widehat{\boldsymbol{R}}^{(j)}, \boldsymbol{R}^{(j)}) + \lambda\Omega(\boldsymbol{W}), \quad \forall j = 1, \ldots, N_{\text{mb}}. \tag{3.102}$$

Here, $\lambda \in [0, \infty)$ is an hyperparameter that weights the relative contribution of the norm penalty term with respect to the standard loss function. The regularization is introduced to the objective function in order to reduce the generalization error of the learning algorithm. Setting $\lambda = 0$ results in no regularization while increasing the value of λ corresponds to an increase in regularization. During training, the backpropagation algorithm (see Sec. 3.3.2.3) minimizes $\mathcal{C}^{(j)}$ by decreasing both the loss function and a measure of the size of the training variables. Note that the penalty Ω is only a function of the weight and that the biases are left unregularized. Unlike the weight which specifies the interaction between two variables, the bias only controls a single variable. Hence, ignoring the bias does not induce much variance in the objective function. The weight-based regularization attempts to limit the influence of irrelevant connections on the network's predictions by penalizing large weights that do not have a large contribution in the reduction of the cost function.

The algorithm for forward propagation of j-th minibatch and cost function computation is outlined in Alg. 3.4. This forward propagation routine is part of the main loop which iterates over the minibatches obtained from the training set. The iteration over all the minibatches constitutes one *training epoch*. The total number of training epochs N_{epochs} is defined by the user and is sufficiently large in order to reach convergence.

3.3.2.3 Backpropagation

The backpropagation algorithm (Rumelhart et al., 1986) allows the information from the scalar cost function $\mathcal{C}^{(j)}$, obtained from the forward propagation, to flow backwards through the network in order to compute the gradients with respect to the training

Algorithm 3.4: Forward propagation and cost function computation

Input: For $j = 1, \ldots, N_{\text{mb}}$: input to process $\boldsymbol{A}^{(j)}$, target output $\boldsymbol{R}^{(j)}$.
For $l = 2, \ldots, L$: weight matrices $\boldsymbol{W}^{[l]}$, bias parameters $\boldsymbol{b}^{[l]}$.
Hyperparameter for regularization λ.

Output: For $j = 1, \ldots, N_{\text{mb}}$: model prediction $\widehat{\boldsymbol{R}}^{(j)}$, cost function $\mathcal{C}^{(j)}$.

Initialize input layer: $\boldsymbol{x}^{[1]} = \boldsymbol{A}^{(j)}$

Propagate the information forward through the hidden layers:

for $l \leftarrow 2$ **to** $L-1$ **do**

  $\boldsymbol{z}^{[l]} = \boldsymbol{W}^{[l]}\boldsymbol{x}^{[l-1]} + \boldsymbol{b}^{[l]}$

  $\boldsymbol{x}^{[l]} = \sigma(\boldsymbol{z}^{[l]})$

Compute prediction at output layer:

$\boldsymbol{x}^{[L]} \leftarrow \boldsymbol{z}^{[L]} = \boldsymbol{W}^{[L]}\boldsymbol{x}^{[L-1]} + \boldsymbol{b}^{[L]}$

$\widehat{\boldsymbol{R}}^{(j)} = \boldsymbol{x}^{[L]}$

Compute cost function:

$\mathcal{C}^{(j)} = \mathcal{L}(\widehat{\boldsymbol{R}}^{(j)}, \boldsymbol{R}^{(j)}) + \lambda\Omega(\boldsymbol{W})$

variables, $\nabla_{\boldsymbol{W}^{[l]}}\mathcal{C}^{(j)}$ and $\nabla_{\boldsymbol{b}^{[l]}}\mathcal{C}^{(j)}$. The gradients are subsequently used in a gradient-based optimization (see Sec. 3.3.2.4) to update the weights and biases, such that the corresponding model output $\widehat{\boldsymbol{R}}^{(j)}$ minimizes the cost function with respect to the target $\boldsymbol{R}^{(j)}$.

For the output layer ($l = L$), the gradient of the cost function (3.102) with respect to the weight gives

$$\begin{aligned}\nabla_{\boldsymbol{W}^{[L]}}\mathcal{C}^{(j)} &= \nabla_{\boldsymbol{z}^{[L]}}\mathcal{C}^{(j)} \cdot \nabla_{\boldsymbol{W}^{[L]}}\boldsymbol{z}^{[L]} + \lambda\nabla_{\boldsymbol{W}^{[L]}}\Omega(\boldsymbol{W}) \\ &= \nabla_{\boldsymbol{z}^{[L]}}\mathcal{C}^{(j)} \cdot (\boldsymbol{x}^{[L-1]})^{\top} + \lambda\nabla_{\boldsymbol{W}^{[L]}}\Omega(\boldsymbol{W}),\end{aligned} \tag{3.103}$$

where (3.97) has been used to evaluate the gradient $\nabla_{\boldsymbol{W}^{[L]}}\boldsymbol{z}^{[L]}$. Using (3.100) and (3.102), the gradient of the cost function with respect to the neural network output is given as

$$\nabla_{\boldsymbol{z}^{[L]}}\mathcal{C}^{(j)} = \nabla_{\widehat{\boldsymbol{R}}^{(j)}}\mathcal{C}^{(j)} = \nabla_{\widehat{\boldsymbol{R}}^{(j)}}\mathcal{L}(\widehat{\boldsymbol{R}}^{(j)}, \boldsymbol{R}^{(j)}) = \frac{1}{N_b}(\widehat{\boldsymbol{R}}^{(j)} - \boldsymbol{R}^{(j)}). \tag{3.104}$$

Similarly, the gradient of the cost function with the bias for the output layer is given as

$$\nabla_{\boldsymbol{b}^{[L]}}\mathcal{C}^{(j)} = \nabla_{\boldsymbol{z}^{[L]}}\mathcal{C}^{(j)} \cdot \nabla_{\boldsymbol{b}^{[L]}}\boldsymbol{z}^{[L]} = \nabla_{\boldsymbol{z}^{[L]}}\mathcal{C}^{(j)}, \tag{3.105}$$

since $\nabla_{\boldsymbol{b}^{[l]}}\boldsymbol{z}^{[l]} = \boldsymbol{1}_{n^{[l]}}$ from (3.97).

Next, the gradient of the cost function for the output layer $\nabla_{\boldsymbol{z}^{[L]}}\mathcal{C}^{(j)}$ is propagated backwards recursively through the hidden layers, from $l = L-1$ to $l = 2$. For a generic layer in the sequence $l = L-1, L-2, \ldots, 2$, the gradient of the cost function with respect to the layer input is calculated as

$$\nabla_{\boldsymbol{x}^{[l]}}\mathcal{C}^{(j)} = \nabla_{\boldsymbol{z}^{[l+1]}}\mathcal{C}^{(j)} \cdot \nabla_{\boldsymbol{x}^{[l]}}\boldsymbol{z}^{[l+1]} = (\boldsymbol{W}^{[l+1]})^{\top} \cdot \nabla_{\boldsymbol{z}^{[l+1]}}\mathcal{C}^{(j)}, \tag{3.106}$$

using (3.97). This gradient on the layer's input is converted into a gradient on the layer's output $\boldsymbol{z}^{[l]}$ as

$$\nabla_{\boldsymbol{z}^{[l]}}\mathcal{C}^{(j)} = \nabla_{\boldsymbol{x}^{[l]}}\mathcal{C}^{(j)} \cdot \nabla_{\boldsymbol{z}^{[l]}}\boldsymbol{x}^{[l]} = \nabla_{\boldsymbol{x}^{[l]}}\mathcal{C}^{(j)} \odot \sigma'(\boldsymbol{z}^{[l]}), \tag{3.107}$$

where $\sigma'(\boldsymbol{z}^{[l]})$ is the derivative of the activation function. Here, $\odot$ represents the Hadamard or element-wise product. These gradients can now be used to evaluate the cost function gradients with respect to the training variables associated with the layer l. Using (3.102), the weight gradient is given as

$$\begin{aligned}\nabla_{\boldsymbol{W}^{[l]}}\mathcal{C}^{(j)} &= \nabla_{\boldsymbol{z}^{[l]}}\mathcal{C}^{(j)} \cdot \nabla_{\boldsymbol{W}^{[l]}}\boldsymbol{z}^{[l]} + \lambda\nabla_{\boldsymbol{W}^{[l]}}\Omega(\boldsymbol{W}) \\ &= \nabla_{\boldsymbol{z}^{[l]}}\mathcal{C}^{(j)} \cdot (\boldsymbol{x}^{[l-1]})^\top + \lambda\nabla_{\boldsymbol{W}^{[l]}}\Omega(\boldsymbol{W}),\end{aligned} \tag{3.108}$$

and the bias gradient is given as

$$\nabla_{\boldsymbol{b}^{[l]}}\mathcal{C}^{(j)} = \nabla_{\boldsymbol{z}^{[l]}}\mathcal{C}^{(j)} \cdot \nabla_{\boldsymbol{b}^{[l]}}\boldsymbol{z}^{[l]} = \nabla_{\boldsymbol{z}^{[l]}}\mathcal{C}^{(j)}. \tag{3.109}$$

Given the neural network output $\widehat{\boldsymbol{R}}^{(j)}$ and the form of the cost function $\mathcal{C}^{(j)}$, the general DNN model can be optimized using methods based on gradient descent. Referring (3.106) and (3.107), it can be seen how the recursive definition of $\nabla_{\boldsymbol{z}^{[l]}}\mathcal{C}^{(j)}$ propagates the information back from the cost function.

The algorithm for backpropagation of the cost function associated with j-th mini-batch is outlined in Alg. 3.5. The schematic of the sequence of operations in the forward pass and backpropagation pass for a minibatch is shown in Fig. 3.8.

Algorithm 3.5: Backpropagation

Input: For $j = 1, \ldots, N_{\text{mb}}$: cost function $\mathcal{C}^{(j)}$, target output $\boldsymbol{R}^{(j)}$.
For $l = 2, \ldots, L$: node data $\boldsymbol{z}^{[l]}$, $\boldsymbol{x}^{[l]}$, weight matrices $\boldsymbol{W}^{[l]}$.

Output: For $j = 1, \ldots, N_{\text{mb}}$: training variable gradients $\nabla_{\boldsymbol{W}}\mathcal{C}^{(j)}$, $\nabla_{\boldsymbol{b}}\mathcal{C}^{(j)}$.

Gradient on output layer:

$\nabla_{\boldsymbol{z}^{[L]}}\mathcal{C}^{(j)} = \nabla_{\widehat{\boldsymbol{R}}^{(j)}}\mathcal{L}(\widehat{\boldsymbol{R}}^{(j)}, \boldsymbol{R}^{(j)})$

$\nabla_{\boldsymbol{W}^{[L]}}\mathcal{C}^{(j)} = \nabla_{\boldsymbol{z}^{[L]}}\mathcal{C}^{(j)} \cdot (\boldsymbol{x}^{[L-1]})^\top + \lambda\nabla_{\boldsymbol{W}^{[L]}}\Omega(\boldsymbol{W})$

$\nabla_{\boldsymbol{b}^{[L]}}\mathcal{C}^{(j)} = \nabla_{\boldsymbol{z}^{[L]}}\mathcal{C}^{(j)}$

Backpropagate the gradients:

for $l \leftarrow L-1$ **to** 2 **do**

Propagate gradient w.r.t. next lower-level hidden layer's pre-activation variable:

$\nabla_{\boldsymbol{x}^{[l]}}\mathcal{C}^{(j)} = (\boldsymbol{W}^{[l+1]})^\top \cdot \nabla_{\boldsymbol{z}^{[l+1]}}\mathcal{C}^{(j)}$

Convert gradient into a gradient on post-activation variable:

$\nabla_{\boldsymbol{z}^{[l]}}\mathcal{C}^{(j)} = \nabla_{\boldsymbol{x}^{[l]}}\mathcal{C}^{(j)} \odot \sigma'(\boldsymbol{z}^{[l]})$

Compute gradients on weights and biases:

$\nabla_{\boldsymbol{W}^{[l]}}\mathcal{C}^{(j)} = \nabla_{\boldsymbol{z}^{[l]}}\mathcal{C}^{(j)} \cdot (\boldsymbol{x}^{[l-1]})^\top + \lambda\nabla_{\boldsymbol{W}^{[l]}}\Omega(\boldsymbol{W})$

$\nabla_{\boldsymbol{b}^{[l]}}\mathcal{C}^{(j)} = \nabla_{\boldsymbol{z}^{[l]}}\mathcal{C}^{(j)}$

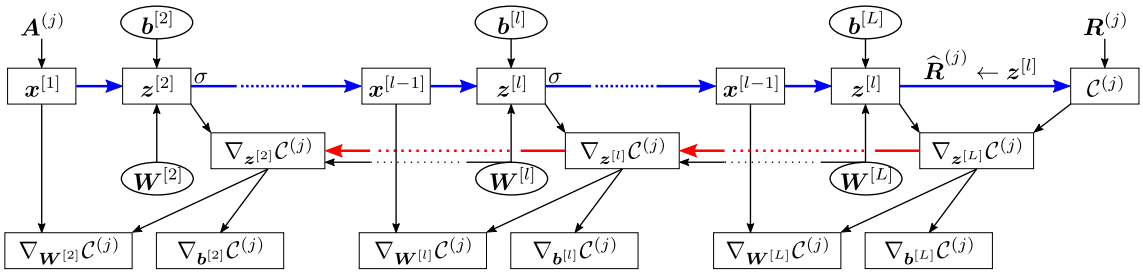


Figure 3.8: Illustration of the sequence of operations in the feedforward and backward propagation steps in a DNN framework. The forward computation from the inputs to the cost function is indicated by blue arrows. The backpropagation of the cost function through the layers is indicated by red arrows. The gradient of the cost function with respect to the training variables is calculated at each layer and used for subsequent gradient-based optimization.

3.3.2.4 Gradient descent optimization

To update the training variables (weights and biases) of the DNN model, the gradients computed from the minibatches of input data (see Sec. 3.3.2.3) are exploited via a gradient descent approach. In this work, the adaptive learning rate optimization algorithm known as Adam (Kingma and Ba, 2014) is used. Contrary to gradient descent optimizers, which adapt the weights with a fixed learning rate across all parameters, adaptive optimizers have more flexibility built-in. The algorithms of this class address the high sensitivity of the cost function to some directions in the space of training variables by altering the learning rate throughout the course of training (Goodfellow et al., 2016). The learning rate used for gradient descent is one of the hyperparameters that has a very significant impact on the model performance. The Adam algorithm therefore identifies the directions of high sensitivity in the space of training variables and automatically adapts the learning rates. The algorithm for the optimization of the training variables, which are collectively referred as $\boldsymbol{\theta}$ (for either $\boldsymbol{\theta} \leftarrow \boldsymbol{W}$ or $\boldsymbol{\theta} \leftarrow \boldsymbol{b}$), is outlined in Alg. 3.6.

The performance of the optimization algorithm over the course of the training epochs is monitored by plotting an error metric evaluated for a validation dataset that the training algorithm does not observe. The validation dataset is constructed by splitting the available input features and targets into disjoint subsets, one of size $N_t^{\text{Train}} \times N_p^{\text{Train}}$, which has been used to train the model variables, and another of size $N_t^{\text{Val}} \times N_p^{\text{Val}}$ used for validation[18]. The validation set is used to estimate the generalization error during training, allowing the training hyperparameters to be updated accordingly. The model

[18] A third disjoint subset of size $N_t^{\text{Test}} \times N_p^{\text{Test}}$, called the test set, is also drawn from the available input features and targets. This set is used for estimating the generalization error after all hyperparameter optimization is complete. Usually the splitting is done such that 70% of the available data is used for learning the model, which is a combination of training (55% of total) and validation (15% of total) datasets. The remaining 30% is used for testing the trained model.

Algorithm 3.6: Adam optimization

Input: For $j = 1, \ldots, N_{\text{mb}}$: gradients $\nabla_{\boldsymbol{\theta}^{[l]}} \mathcal{C}^{(j)}$.
For $l = 2, \ldots, L$: initial values $\boldsymbol{\theta}^{[l]}$.
Hyperparameters (default value): learning rate α (1.0×10^{-3}),
exponential decay rates for first and second moment estimates – β_1 (0.9), β_2 (0.999),
constant for numerical stabilization ϵ (1.0×10^{-8}),
maximum number of iterations N_{epochs}.

Output: For $l = 2, \ldots, L$: updated training variables $\boldsymbol{\theta}^{[l]}$.

Initialize first and second moment variables: $\boldsymbol{s} = \boldsymbol{0}$, $\boldsymbol{r} = \boldsymbol{0}$
Initialize iteration step: $n = 0$
Parameter update:
while $n < N_{\text{epochs}}$ **do**
  for $l \leftarrow 2$ **to** L **do**
    Average gradient over minibatches:
    $\boldsymbol{g} \leftarrow \frac{1}{N_{\text{mb}}} \sum_{j=1}^{N_{\text{mb}}} \nabla_{\boldsymbol{\theta}^{[l]}} \mathcal{C}^{(j)}$
    Update biased first moment estimate:
    $\boldsymbol{s} \leftarrow \beta_1 \boldsymbol{s} + (1 - \beta_1)\boldsymbol{g}$
    Update biased second moment estimate:
    $\boldsymbol{r} \leftarrow \beta_2 \boldsymbol{r} + (1 - \beta_2)\boldsymbol{g} \odot \boldsymbol{g}$
    Correct bias in first moment:
    $\widehat{\boldsymbol{s}} \leftarrow \boldsymbol{s}(1 - \beta_1^n)^{-1}$
    Correct bias in second moment:
    $\widehat{\boldsymbol{r}} \leftarrow \boldsymbol{r}(1 - \beta_2^n)^{-1}$
    Compute update step size (element-wise operations):
    $\Delta\boldsymbol{\theta}^{[l]} = -\alpha\widehat{\boldsymbol{s}}(\sqrt{\widehat{\boldsymbol{r}}} + \epsilon)^{-1}$
    Apply update:
    $\boldsymbol{\theta}^{[l]} \leftarrow \boldsymbol{\theta}^{[l]} + \Delta\boldsymbol{\theta}^{[l]}$
  $n \leftarrow n + 1$

is evaluated throughout the training epochs to obtain the validation set estimates as

$$\begin{aligned} \boldsymbol{a}^{\text{Val,NN}}(t_{k+1}; \boldsymbol{\mu}_j) = \boldsymbol{a}^{\text{Val,POD}}(t_k; \boldsymbol{\mu}_j) + \\ \boldsymbol{f}^{\text{NN}}(\boldsymbol{a}^{\text{Val,POD}}(t_{k-p+1}; \boldsymbol{\mu}_j), \ldots, \boldsymbol{a}^{\text{Val,POD}}(t_k; \boldsymbol{\mu}_j); \boldsymbol{W}^*, \boldsymbol{b}^*), \end{aligned}$$

$$\text{for } p = 1, \ldots, N_t^{\text{Val}} - 1, \quad k = p - 1, \ldots, N_t^{\text{Val}} - 1 \quad \text{and} \quad j = 1, \ldots, N_p^{\text{Val}}. \tag{3.110}$$

The number of time lag steps p is the same as the one used during training and $\boldsymbol{f}^{\text{NN}}(\boldsymbol{W}^*, \boldsymbol{b}^*)$ is the DNN model with trained weights and biases. Here, $\boldsymbol{a}^{\text{Val,POD}}(t_k; \boldsymbol{\mu}_j) \in \mathbb{R}^{N_r \times 1}$ is the validation subset of the POD coefficients dataset, separate from the training dataset, and $\boldsymbol{a}^{\text{Val,NN}}(t_{k+1}; \boldsymbol{\mu}_j)$ is the NN estimation. In this work, the root-mean-square error ($RMSE_{\text{Val}}$) is used to gauge the performance of the optimization algorithm. The

validation error is given as

$$RMSE_{\text{Val}} = \sqrt{\frac{1}{N_t^{\text{Val}} N_p^{\text{Val}}} \sum_{k=0}^{N_t^{\text{Val}}-1} \sum_{j=1}^{N_p^{\text{Val}}} \left\| \boldsymbol{a}^{\text{Val,NN}}(t_k; \boldsymbol{\mu}_j) - \boldsymbol{a}^{\text{Val,POD}}(t_k; \boldsymbol{\mu}_j) \right\|_2^2} \quad (3.111)$$

where the estimation from (3.110) is used. This value is usually compared with the training error $RMSE_{\text{Train}}$ which is also evaluated during the course of the training for the training dataset. The comparison determines if the model is *underfitting* or *overfitting*. Underfitting occurs when the model is not able to obtain a sufficiently low training error value on the training set, while overfitting occurs when the gap between the training error and test error is too large. The difference between the two errors is monitored. This difference is used to tune the regularization coefficient in (3.102) and the model hyperparameters. The objective is to obtain a model with sufficiently low generalization error at the end of the offline training stage.

3.3.2.5 Prediction using the DNN model

The trained DNN model $\boldsymbol{f}^{\text{NN}}$ can be used to obtain online predictions of the POD coefficients using (3.90). This online vision is outlined in Fig. 3.9. As the DNN model is trained using the residuals $\boldsymbol{r}(t_k; \boldsymbol{\mu}_j)$ as target, the model output during prediction is also in terms of the residual. This residual is converted to the estimates of the POD coefficients at the next time step t_{k+1} by adding it to the coefficient vector at time step t_k, already available as an input feature, as follows

$$\boldsymbol{a}^{\text{NN}}(t_{k+1}; \boldsymbol{\mu}_j) = \boldsymbol{a}^{\text{NN}}(t_k; \boldsymbol{\mu}_j) + \boldsymbol{r}(t_k; \boldsymbol{\mu}_j). \quad (3.112)$$

The prediction of the dynamics at subsequent time steps $(t_p, \ldots, t_{N_t^{\text{Pred}}-1})$ is obtained by recursive feedback of the output POD coefficients to the input sequence of p time lag steps. Note that during prediction, the sequence of data at the first p time steps $(t_0, \ldots, t_{p-1})$ must be provided as input.

To summarize, the various steps in the online and offline stages of the non-intrusive, DNN-based ROM approach are shown in Fig. 3.10. The whole framework is based on an encoder-decoder approach to transfer data from the high-fidelity space to a reduced-order space and vice-versa. The encoder step is used to construct the time series of POD coefficients from the snapshot data using (2.11). The decoder step is used to construct the field at any time t by inverse transform (2.15).

The main advantage of the non-intrusive ROM framework is that it does not require information about the governing equations constituting the full order model. However, as the non-intrusive ROM framework is generated solely from the snapshot data reconstructed onto a reduced POD-spanned space, it may still suffer from fundamental challenges of traditional POD-Galerkin models. One way to mitigate this lack of accuracy is to merge this framework into a data assimilation algorithm. The ability of the DNN framework to take the memory effect into account in order to predict the future state of the system makes it amenable to be used as a forecast model in sequential data assimilation algorithms discussed in Sec. 3.2, replacing the Galerkin projection based ROM. This gives a possibility to provide an optimal estimation of the dynamical system,

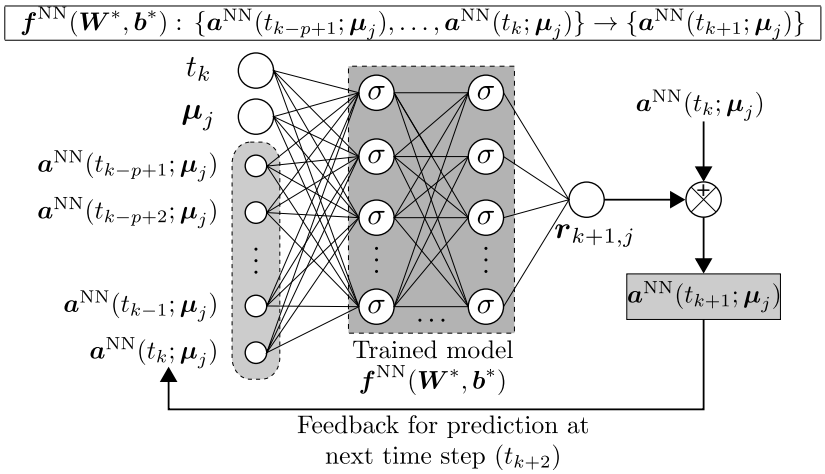


Figure 3.9: DNN architecture of the non-intrusive ROM in the online stage. The learned forward model $\boldsymbol{f}^{\mathrm{NN}}(\boldsymbol{W}^*, \boldsymbol{b}^*)$ is used to obtain the sequential prediction of the POD coefficients $\boldsymbol{a}^{\mathrm{NN}}(t; \boldsymbol{\mu})$.

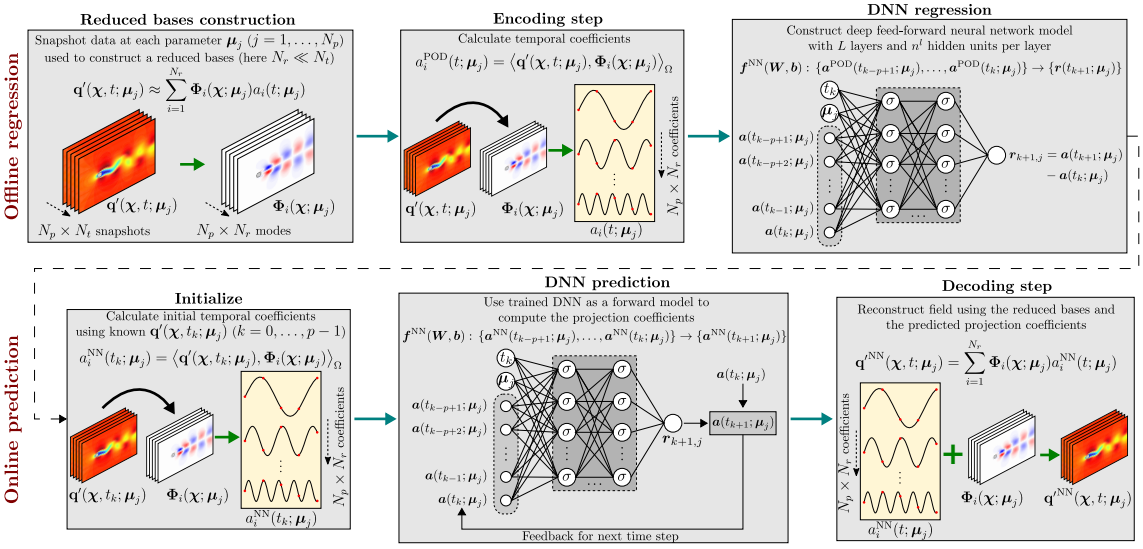


Figure 3.10: Nonintrusive ROM framework.

by taking into consideration the statistical confidences of both the model outputs and the observations, and nudging the ROM solution towards the true dynamics.

Chapter 4

System identification by linear regression models

Contents

In Sec. 3.1, different strategies - namely ordinary least-squares (OLS), sparse identification of nonlinear dynamics (SINDy), and least angle regression (LARS) – have been introduced with the purpose of identifying the coefficients of reduced-order models from limited data only. The implementation of such methods when considering either numerical or experimental data is, however, far from straightforward. The available data are

generally incomplete or corrupted by noises from different sources leading to numerical problems. In order to alleviate these difficulties, it is necessary to invest time in preparing the data that will be used in the regression methods. For this reason, we begin this chapter by introducing some useful numerical tools such as low-pass filtering for noise suppression and total-variation regularization for numerical differentiation. The performance of the system identification methods considered in this work is then compared and evaluated in a probabilistic framework using the bootstrap method. This method can be used to quantify the uncertainty associated with a given estimator, a feature which is not readily available with other methods such as cross-validation. Three toy models are then considered: i) a three-dimensional linear system, ii) the Lorenz-63 model, and iii) the Lorenz-96 model. These three test cases are considered to be representatives of dynamics with increasing level of complexity.

The current chapter is organized as follows. The numerical tools that are necessary to prepare the noisy data are discussed in detail in Sec. 4.1. In Sec. 4.2, the bootstrap method is reviewed. The three toy models are then presented in Sec. 4.3. Finally, the performance of the different identification strategies to recover the parameters of the model equation is compared and discussed in Sec. 4.4. Some conclusions are drawn in Sec. 4.5.

4.1 Data preparation of noisy data

Throughout this chapter, we consider systems for which the dynamics can be represented by governing equations consisting of a second-order polynomial. The POD-ROM discussed in Sec. 2.2.2 entered in this class when the nonlinear residual term is not considered. In the following, we treat the case of a linear regression model of the type (3.4), *i.e.*

$$\dot{\boldsymbol{a}}_i = \boldsymbol{A}\boldsymbol{\theta}_i \quad \forall\, i = 1, \dots, N_s, \tag{4.1}$$

where N_s is the size of the state vector $\boldsymbol{a}(t) = \{a_i(t)\}_{i=1}^{N_s}$. The definitions[1] of $\boldsymbol{A} \in \mathbb{R}^{N_t \times N_{\theta_i}}$, $\dot{\boldsymbol{a}}_i \in \mathbb{R}^{N_t \times 1}$, and $\boldsymbol{\theta}_i \in \mathbb{R}^{N_{\theta_i} \times 1}$ follow from Sec. 3.1. Here, the objective is to estimate the matrix $\boldsymbol{\theta} \in \mathbb{R}^{N_{\theta_i} \times N_s}$, *i.e.* the model parameter matrix with column vectors $\boldsymbol{\theta}_i$, such that the residual (3.10) is minimized. As mentioned in the introduction and discussed in the previous chapter, numerical difficulties occur when solving such linear problems with the presence of noise in the dataset. The data collection are in general corrupted by noise from different sources depending on, for instance, whether the data have been obtained experimentally or from a numerical simulation. It is also common to consider the noise as white and Gaussian, which can sometimes be a too crude assumption. For this reason, it is therefore necessary to have, at disposal, tools which may allow the identification process to be less sensitive to the signal-to-noise ratios.

4.1.1 Low-pass filter

Rare events contained in the time series (*e.g.* outliers) may cause the estimation to become statistically biased. To minimize the effect of these outliers, a solution is to use

[1] In Sec. 3.1, the matrix $\boldsymbol{A}$ is denoted by $\boldsymbol{X}$ to fit the notations generally used in statistical learning.

digital filtering. Before assembling the matrix of basis functions $\boldsymbol{A}$ in the POD-ROM (4.1), the signal $\boldsymbol{a}(t)$ can be filtered using a *low-pass filter* (Sagara et al., 1991). A low-pass (LP) filter passes low-frequency signals and reduces the amplitude of signals with frequencies higher than the cutoff frequency (f_c). It was shown in Sagara et al. (1991) that the identification of parameters is not very sensitive to the cutoff frequency. If the filter is designed so that its pass-band matches that of the system closely, the noise effects are sufficiently reduced and the least-squares identification is able to perform well in the presence of low measurement noise.

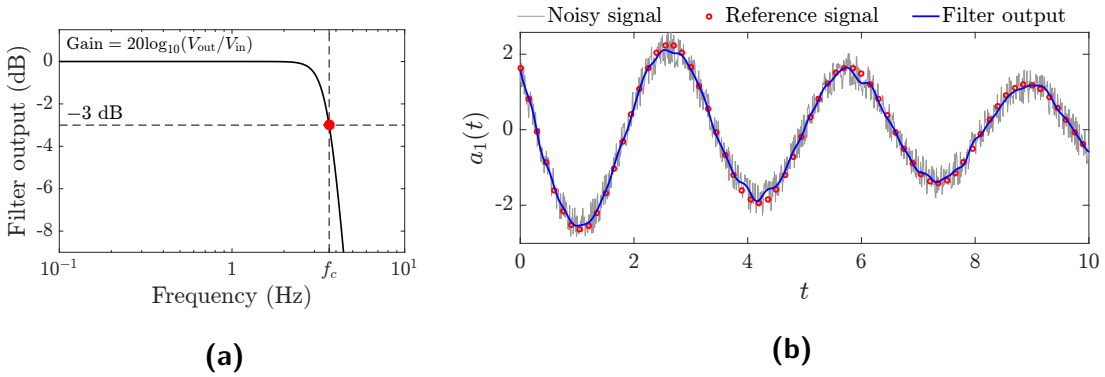


Figure 4.1: (a) Bode plot showing the 5-th order Butterworth LP filter response. The cut-off frequency $f_c = 3.65$ Hz is indicated by the red dot. (b) Output of LP filter applied to noisy trajectory obtained by applying high level noise ($\eta = 10\%$) to the state $a_1(t)$ (reference signal) of a example three-dimensional linear system (Sec. 4.3.1).

A 5-th order digital Butterworth filter (Parks and Burrus, 1987) has been used later in this work. The Butterworth filter is a type of digital infinite impulse response (IIR) filter[2] which is designed in the continuous-time domain and discretized by bilinear transformation.

The designed filter and its application to a noisy arbitrary signal with an additive noise of 10% is shown in Fig. 4.1. Plotting the filter's output signal (V_{out}) against different values of input frequency f gives the frequency response curve or Bode magnitude plot function. The Bode plot shows the frequency response of the filter relative to the cutoff frequency point f_c. The response is nearly flat for low frequencies in the passband ($f < f_c$) and all of the input signal is passed directly to the output with a unity gain $V_{\text{out}}/V_{\text{in}} = 1$. After this cutoff frequency point, the response of the 5-th order filter *rolls-off* at -30 dB/octave. Any high frequency signal applied to the low pass filter in the stopband ($f \geq f_c$) is significantly attenuated. Butterworth filters are optimal in the sense of having a maximally flat amplitude response with a quick roll-off around the cutoff frequency, which improves with increasing order.

This cutoff frequency is defined as the frequency point where the output signal is attenuated to 70.7% of the input signal value or a gain of -3 dB of the input. This

[2]The infinite impulse response (IIR) filter is a class of digital filters in which the impulse response never decays exactly to zero as the time elapses. This is in contrast to the finite impulse response (FIR) filter class in which the response falls exactly to zero in a finite time period. The IIR filters are, in general, more efficient in terms of implementation in order to meet a specific requirement of passband, stopband, and/or roll-off as compared to FIR filters of same order.

corresponds to an attenuation of input power by a factor of $1/2$ and that of the amplitude by a factor of $1/\sqrt{2}$. The gain of the filter, expressed in decibels, is a function of the ratio of output value (V_{out}) and its corresponding input value (V_{in}). It is given as:

$$\text{Gain} = 20 \log_{10} \left(\frac{V_{\text{out}}}{V_{\text{in}}} \right), \quad V_{\text{in}} > 0. \tag{4.2}$$

In the context of the model parameter identification problem from noisy data, we use the LP digital filter to pre-filter the sampled time series data. A recursive identification algorithm is then used to estimate the system parameters from the filtered input-output sampled data.

4.1.2 Numerical differentiation of noisy data

Computing the derivative of functions specified by data is another component of the model parameter identification problem which needs to be addressed. In the case of POD-ROM, the time derivative $\dot{\boldsymbol{a}}_i(t)$ is needed in order to assemble the left-hand side of the system described by (4.1). The standard finite difference approximations of the derivatives are computationally inexpensive and work well with smooth data sequences (*e.g.* data obtained from toy models). However, in the presence of noise in the data, the derivatives are greatly amplified. In addition, there are cases where denoising the data using digital filters presented in Sec. 4.1.1 may not be sufficient.

The *total-variation regularization* based differentiation method, hereafter known as TVreg, was proposed by Chartrand (2011) specifically to handle the derivatives of functions specified by noisy data. The principle is to regularize the differentiation process in order to avoid the noise amplification. The derivative of a function $f(t)$ on $[0, T]$ is given as

$$f'(t) = \underset{u(t)}{\arg\min}\, F(u) = \alpha \int_0^T |u'(t)|\, \mathrm{d}t + \frac{1}{2} \int_0^T |Au(t) - f(t)|\, \mathrm{d}t. \tag{4.3}$$

Here, $u'(t)$ represents the derivative with respect to t, and A is the antidifferentiation[3] operator such that $Au(t) = \int_0^T u(t)\, \mathrm{d}t$. The first term on the right-hand side of (4.3) is a regularization term which penalizes any irregularity in $u(t)$, while the second term is a data fidelity term which penalizes any discrepancy between $Au(t)$ and $f(t)$. The regularization parameter α controls the relative significance of the two terms in the estimation of the minimizer. Usually, a higher value of α increases the regularization strength and improves conditioning. The value can be fixed by using the discrepancy principle (see Chartrand, 2011) but it requires the knowledge of the noise variance in the signal f which is not always known and can only be estimated. In the subsequent applications of TVreg, we opt to select α heuristically.

As an example, a comparison of the result of TVreg differentiation with the fourth-order accurate finite difference scheme is shown in Fig. 4.2 for the arbitrary noise signal used in Fig. 4.1. The result using the TVreg scheme has been obtained after selecting

[3] Antidifferentiation is not strictly equivalent to integration. Antidifferentiation is purely defined as the process of finding a function whose derivative is given. Integration, in the sense of Riemann, is roughly defined as the limit of sum of rectangles under a curve.

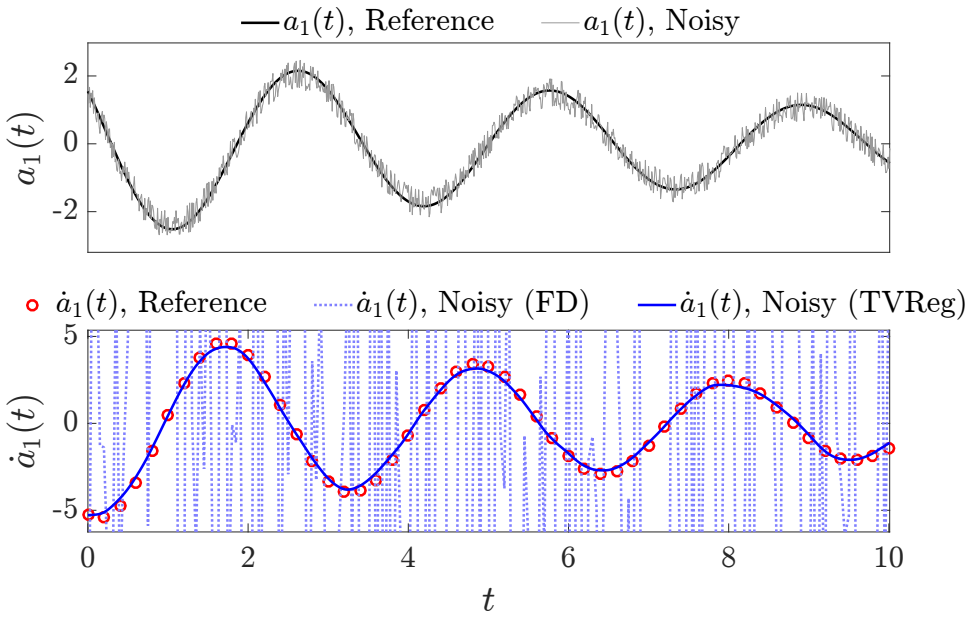


Figure 4.2: Time derivative of the component $a_1(t)$ of a three-dimensional linear system. a_1 is the same as in Fig. 4.1. (Top) Time evolution of signal $a_1(t)$ without (reference) and with 10% noise. (Bottom) Time evolution of derivative $\dot{a}_1(t)$ of the noisy signal obtained using TVreg differentiation compared with that of the reference and noisy signal, both obtained using fourth-order accurate finite difference (FD).

the value of the regularization parameter as $\alpha = 10$. We observe that the derivatives calculated using the finite difference method is amplified by the presence of noise in the signal while the regularized derivative accurately estimates the derivative.

4.2 Bootstrap method

In order to have a critical evaluation of the different identification methods by linear regression models, we would like to associate uncertainties to each estimator. For this, a probabilistic framework such as the one offered by the *bootstrap* method is necessary.

4.2.1 Motivation

The system identification methods proposed in Sec. 3.1 give estimations of the parameters assuming a full knowledge of the system. In such cases, a fixed model setup (dynamical model, parameters, boundary and initial conditions, geometries, etc.) produces unique outputs. However, we often encounter uncertainty of varying extents in physical processes. Model structure is one of the contributors to the uncertainty. The model structural uncertainty is attributed to numerical errors, incomplete representations of all the physical processes important to the state being simulated, nonrepresentative model formulations due to an incomplete understanding of the physics involved, and/or overrepresentation of some rare events during sampling. In the context of POD-ROM,

the calculation of the POD modes on limited sample size inhibits the true knowledge of the full low-dimensional manifold and introduces uncertainties when the model is used for estimation. With very few exceptions, it is often not possible to make highly accurate estimations using such models. Therefore, a measure of the uncertainties is required in order to characterize the estimators. For this purpose, we introduce a statistical method of *bootstrapping* where we estimate the properties of an estimator (*e.g.* variance) by evaluating the said properties over samples drawn from an approximating distribution.

4.2.2 Bootstrap principle

The bootstrap method[4], first introduced in[5] Efron (1979), is a general approach for statistical inference of the distribution of an estimator or interested variable by resampling the data from random independent observations. The method is based on the *plug-in principle* which states that a feature of a given distribution (*e.g.* the expected value, the variance, a quantile) that cannot be computed exactly can be approximated by the same feature of the empirical distribution[6] of a sample of observations drawn from the given distribution. In this work, we use bootstrapping to derive the probabilistic estimate and confidence intervals of the unknown parameters of the POD-ROM identified using the system identification methods.

Let[7] $\boldsymbol{X} = (X_1, \dots, X_n)$ and $\boldsymbol{x} = (x_1, \dots, x_n)$ denote the random sample and its observed realization, respectively (see App. D.1). The random variables X_i (for all $i = 1, \dots, n$) are i.i.d.[8] and have for cumulative distribution function, the unknown function F, *i.e.* $X_i \sim F$. The problem we wish to solve with the bootstrap approach is the following. How to estimate a parameter of interest say, $\theta = T(F)$, on the basis of the observation $\boldsymbol{x}$? If the feature of interest is, for example, the mean of the distribution of a univariate population, we have $T(F) = \mathbb{E}[X] = \int x \, \mathrm{d}F(x)$. A common estimate of θ is the *plug-in* estimate given as $\widehat{\theta} = T(\widehat{F})$ where $\widehat{F}$ is the empirical distribution (see App. D.3). For example, the estimator of the mean of the population is given as the sample mean $\widehat{\theta} = \int x \, \mathrm{d}\widehat{F}(x) = \sum_{i=1}^{n} x_i / n$.

In statistical inference, we are usually interested in estimating the sampling distribution of a random variable $\mathcal{R}$, possibly depending on both $\boldsymbol{X}$ and the unknown distribution F, *i.e.* $\mathcal{R}(\boldsymbol{X}, F)$. For example, we can have $\mathcal{R}(\boldsymbol{X}, F) = T(\widehat{F}) - T(F) = \widehat{\theta} - \theta$. The bootstrap provides an approximation of the sampling distribution of $\mathcal{R}(\boldsymbol{X}, F)$ based on

[4] The use of the term bootstrap derives from the phrase "to pull oneself up by one's bootstraps", widely thought to be based on the 18th century book "The Surprising Adventures of Baron Munchausen" by Rudolph Erich Raspe: "The Baron had fallen to the bottom of a deep lake. Just when it looked like all was lost, he thought to pick himself up by his own bootstraps.".

[5] The International Prize in Statistics has been awarded in 2018 to Bradley Efron, professor of statistics and biomedical data science at Stanford University, in recognition of the "bootstrap" method.

[6] In statistics, an empirical distribution function is the distribution function associated with the measure of a sample *i.e.* a random measure arising from a realization of a sequence of random variables. A formal definition is given in App. D.3.

[7] In the following, we use the classical notations of bootstrap methods. The differences with the notations used in the previous chapters may be obvious depending on the context.

[8] Independent and identically distributed random variable

the empirical distribution function $\widehat{F}$ of the observed data $\boldsymbol{x}$. Knowing $\widehat{F}$, we draw from it, a random sample of size n called *bootstrap sample i.e.* $\boldsymbol{X}^* = (X_1^*, \ldots, X_n^*)$ and $\boldsymbol{x}^* = (x_1^*, \ldots, x_n^*)$ where $X_i^* \sim \widehat{F}$ for $i = 1, \ldots, n$. The values of $\boldsymbol{X}^*$ are selected *with* replacement from the set $\{x_1, \ldots, x_n\}$. This way a permutation distribution of the original observations is not obtained. By the bootstrap principle, the sampling distribution of $\mathcal{R}(\boldsymbol{X}, F)$ is approximated by $\mathcal{R}^* = \mathcal{R}(\boldsymbol{X}^*, \widehat{F})$.

The bootstrap algorithm begins by generating a large number of independent bootstrap samples $\boldsymbol{x}^{*(j)}$ (for $j = 1, \ldots, N_b$), each of size n. For each bootstrap sample, the statistic of interest is then computed. These values are used to estimate the sampling distribution of the statistics of interest (see below). Bootstrapping exploits the following central analogy: "The population is to the sample as the sample is to the bootstrap samples". The estimation error may be made arbitrarily small by increasing N_b. It must be noted that for realistic sample sizes n, the number of potential bootstrap samples N_b is very large[9]. For this reason, considering all the possible combinations may not be possible. The limitation of the number of bootstrap samples therefore has the consequence of introducing estimation errors.

Bootstrap bias and standard error

Here, we give useful measures of statistical errors of the estimators $\widehat{\theta}$, namely the bias and the standard error. The bootstrap algorithm presented before may be adapted to give estimates of these values.

The bias is the difference between the expected value of $\widehat{\theta}$ and the quantity θ being estimated. We consider the quantity $\mathcal{R}(\boldsymbol{X}, F) = T(\widehat{F}) - T(F) = \widehat{\theta} - \theta$. The bias of $\widehat{\theta}$, $B_F(\widehat{\theta})$, is the expectation of $\mathcal{R}(\boldsymbol{X}, F)$ which is given as

$$B_F(\widehat{\theta}) = \mathbb{E}_F[\widehat{\theta} - \theta] = \mathbb{E}_F[\widehat{\theta}] - \theta. \tag{4.4}$$

The bootstrap estimate of bias is defined to be the estimate $\widehat{B}_{\widehat{F}}$ that we obtain by substituting $\widehat{F}$ for F in (4.4), *i.e.*

$$\widehat{B}_{\widehat{F}}(\widehat{\theta}^*) = \mathbb{E}_{\widehat{F}}[\widehat{\theta}^*] - \widehat{\theta}. \tag{4.5}$$

This estimate may be approximated by Monte Carlo simulation. Hence, we generate N_b independent bootstrap samples $\boldsymbol{x}^{*(j)}$ $(j = 1, \ldots, N_b)$ and evaluate the bootstrap replications $\widehat{\theta}^{*(j)}$. The bootstrap estimate of bias based on the N_b replications is obtained as

$$\widehat{B}^*(\widehat{\theta}^*) = \overline{\theta}^* - \widehat{\theta}, \tag{4.6}$$

where the bootstrap expectation is approximated by the average

$$\overline{\theta}^* = \frac{1}{N_b} \sum_{j=1}^{N_b} \widehat{\theta}^{*(j)}. \tag{4.7}$$

[9] The number of unique bootstrap samples for a dataset with n observations is $N_b = {}^{2n-1}C_{n-1}$. This number increases exponentially with n. Stirling's formula $\left(n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n\right)$ gives as approximation $N_b \sim (n\pi)^{-\frac{1}{2}}\, 2^{2n-1}$. It can be shown that $N_b \sim 10^{n/2}$.

The standard error of the parameter estimate $\widehat{\theta} = T(\widehat{F})$ is given as

$$SE_F(\widehat{\theta}) = \sqrt{\text{var}(\widehat{\theta})} = \sqrt{\mathbb{E}_F[(\widehat{\theta} - \theta)^2]}. \tag{4.8}$$

Following the same logic as for the bias (substitution of $\widehat{F}$ for F), we define the bootstrap estimate of the standard error as

$$\widehat{SE}_{\widehat{F}}(\widehat{\theta}^*) = \sqrt{\text{var}(\widehat{\theta}^*)} = \sqrt{\mathbb{E}_{\widehat{F}}[(\widehat{\theta}^* - \widehat{\theta})^2]}. \tag{4.9}$$

The corresponding simulated bootstrap estimate is obtained as

$$\widehat{SE}^*(\widehat{\theta}^*) = \sqrt{\frac{1}{N_b - 1} \sum_{j=1}^{N_b} (\widehat{\theta}^{*(j)} - \overline{\theta}^*)^2}. \tag{4.10}$$

The standard error is generally corrected by multiplying the right-hand side of (4.10) with $\sqrt{n/(n-1)}$. We note that as the number of bootstrap sample increases, the bootstrap estimates approach the theoretical values

$$\lim_{N_b \to \infty} \widehat{B}^*(\widehat{\theta}^*) = \widehat{B}_{\widehat{F}}(\widehat{\theta}^*), \quad \text{and} \quad \lim_{N_b \to \infty} \widehat{SE}^*(\widehat{\theta}^*) = \widehat{SE}_{\widehat{F}}(\widehat{\theta}^*). \tag{4.11}$$

Bootstrap confidence interval

One of the key applications of the bootstrap method is to produce automatically good confidence intervals of the parameter estimates of the desired statistics $\mathcal{R}(\boldsymbol{X}, F)$. A confidence interval (CI) is defined as the interval between the lower and upper bounds of a parameter estimate at a prespecified confidence level (*e.g.* 95% confidence interval). When the value of N_b is large, it turns out that most bootstrap statistics are asymptotically normally distributed, *i.e.* the estimator $\widehat{\theta} = T(\widehat{F})$ tends to follow a normal distribution with mean $\theta = T(F)$ and standard deviation $\widehat{SE}_{\widehat{F}}(\widehat{\theta}^*)$ (see Fox, 2015). We have

$$\mathcal{R}(\boldsymbol{X}, F) = \frac{\widehat{\theta} - \theta}{\widehat{SE}_{\widehat{F}}(\widehat{\theta}^*)} \sim \mathcal{N}(0, 1). \tag{4.12}$$

The $100(1-a)\%$ confidence interval for θ based on the *normal-theory intervals* is given as

$$\theta = \widehat{\theta} \pm z_{a/2} \widehat{SE}^*(\widehat{\theta}^*), \tag{4.13}$$

where $z_{a/2}$ is the unit-normal value with a probability $a/2$ to the right. Results presented by Efron and Tibshirani (1994) suggest that basing bootstrap confidence intervals on $N_b = 1000$ bootstrap samples generally provides sufficiently accurate results. The method however uses the tails of the distribution of $\mathcal{R}(\boldsymbol{X}, F)$ which requires large value of N_b to be well described. Also, the method is only applicable to parameters that shift according to shift in their distribution, *i.e. location* parameters such as mean, median or sample percentile.

To overcome these issues, the *accelerated bias-corrected percentile* (BCa) method (Efron, 1987) is used. Thereafter, we follow the presentation of Fox (2015). As before,

the goal is to obtain a $100(1-a)\%$ confidence interval for θ using the sample estimate $\widehat{\theta}$ and the bootstrap estimates $\widehat{\theta}^{*(j)}$. The BCa method corrects for bias and skewness of the bootstrap estimates by using a bias-correction Z and an acceleration factor A. The *bias-correction* is given as

$$Z = \Phi^{-1}\left[\frac{1}{N_b}\overset{N_b}{\underset{j=1}{\#}}(\widehat{\theta}^{*(j)} < \widehat{\theta})\right], \tag{4.14}$$

where Φ^{-1} is the inverse of the standard normal cumulative distribution function[10] and $\#(\widehat{\theta}^{*(j)} < \widehat{\theta})/N_b$ is the proportion of bootstrap estimates below the estimate $\widehat{\theta}$. If the bootstrap sampling distribution is symmetric and the estimate $\widehat{\theta}$ is unbiased, the proportion is approximately equal to 0.5 and the correction Z is approximately equal to zero. Next, the *acceleration factor* is estimated through a jackknife resampling (also known as "leave one out" resampling), which involves generating n replicates of the original sample by omitting one observation for each replicate. These are known as jackknife replicates; the i-th replicate $\widehat{\theta}_{(-i)}$ (for all $i = 1, \ldots, n$) represents the value of $\widehat{\theta}$ when the i-th observation is left out from the sample. Let $\overline{\theta}$ be the average of the jackknife replicates, *i.e.* $\overline{\theta} \coloneqq \sum_{i=1}^{n} \widehat{\theta}_{(-i)}/n$. The acceleration factor is given as

$$A = \frac{\sum_{i=1}^{n}(\widehat{\theta}_{(-i)} - \overline{\theta})^3}{6\left(\sum_{i=1}^{n}(\widehat{\theta}_{(-i)} - \overline{\theta})^2\right)^{3/2}}. \tag{4.15}$$

Using the correction factors Z and A, we compute

$$A_{1,2} = \Phi\left(Z + \frac{Z \mp z_{a/2}}{1 - A(Z \mp z_{a/2})}\right). \tag{4.16}$$

The values A_1 and A_2 are used to locate the endpoints of the percentile confidence interval as $[\widehat{\theta}^{*(j_1)}, \widehat{\theta}^{*(j_2)}]$, with the indices given by the products $j_{1,2} = N_b A_{1,2}$ (rounded or interpolated to obtain an integer value). In this work, the `conf_int` function from the `arch` toolbox (Sheppard et al., 2020) is used to obtain the BCa confidence intervals.

In Demo. 4.1, a demonstration of the bootstrap method introduced previously is given for the simple case of linear regression.

[10] The inverse function (also known as quantile function) of the standard normal cumulative distribution gives the value x of the distribution function corresponding to a probability value $0 < p < 1$

$$x = \Phi^{-1}(p) = \{x : \Phi(x) = p\}, \quad \text{where} \quad p = \Phi(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} \exp(-t^2/2)\,\mathrm{d}t.$$

For example, $\Phi^{-1}(p = 0.975) = 1.96$.

Demo 4.1: Bootstrapping regression models

The bootstrapping procedures presented in the previous section are extended to linear regression models. The objective is to study the uncertainty (*e.g.* variance, mean-square error or confidence interval) of the fitted parameters. To simplify the presentation, we consider the case of one response variable $y_k \in \mathbb{R}$ and one regressor $x_k \in \mathbb{R}$, *i.e.*

$$y_k = \beta_0 + \beta_1 x_k.$$

The extension to the N_s-vector of regressors $\boldsymbol{x}_k$ is immediate.

The most straightforward approach is to collect the response-variable value y_k and regressor x_k as the observations $\boldsymbol{z}_k$, which gives

$$\boldsymbol{z}_k \coloneqq [y_k, x_k]^\top \in \mathbb{R}^2, \quad \forall\, k = 1, \dots, n.$$

After obtaining the regression coefficients $\widehat{\theta} = [\widehat{\beta}_0, \widehat{\beta}_1]$, the predicted value is given by

$$\widehat{y}_k = \widehat{\beta}_0 + \widehat{\beta}_1 x_k.$$

The residuals of the fitted regression model are also obtained as $\epsilon_k = y_k - \widehat{y}_k$.

To bootstrap the regression model, there are two approaches (Fox, 2015): i) treating the regressor x_k as random and selecting bootstrap samples directly from the observations $\boldsymbol{z}_k$, also called *paired bootstrap*, and ii) treating the regressor as fixed and resampling the residual ϵ_k, also called *residual bootstrap*. A disadvantage of fixed regressor is that the procedure implicitly assumes that the regression model fit to the data is correct and that the errors are identically distributed. On its side, the paired bootstrap might lead to a bad result especially in the presence of influential observations (some x_k very far away from the others).

In this demonstration, we apply a slightly modified form of the paired bootstrap where the regressor is fixed to its sample value $\boldsymbol{x} = \{x_1, \dots, x_n\}$. The objective is to show the influence of the number of bootstrap samples N_b on the estimation of θ.

We perform four separate bootstrap regressions with different numbers of samples $N_b = 1, 10, 100, 1000$. The bootstrap samples are selected from y_k as $\boldsymbol{y}^{*(j)} = \{y_1^{*(j)}, \dots, y_n^{*(j)}\}$ for all $j = 1, \dots, N_b$. The bootstrapped $\boldsymbol{y}^{*(j)}$ are regressed on the fixed $\boldsymbol{x}$ to obtain the bootstrap regression coefficients $\widehat{\theta}^{*(j)}$. In case of least-squares regression, for example, the coefficients are estimated as $\widehat{\theta}^{*(j)} = (\boldsymbol{x}^\top \boldsymbol{x})^{-1} \boldsymbol{x}^\top \boldsymbol{y}^{*(j)}$ for all $j = 1, \dots, N_b$. The resampled bootstrap coefficients $\widehat{\theta}^{*(j)}$ are then used to obtain the bootstrap CI. For qualitative assessment, the relative sum of squares (RSS) is calculated as

$$RSS = \frac{\sum_{k=1}^{n} (y_k - \widehat{y}_k)^2}{\sum_{k=1}^{n} y_k^2}.$$

We perform regression on noisy data generated by the model $y_k = \beta_0 + \beta_1 x_k + \eta_k$, where the coefficients are assigned as $\theta = [\beta_0, \beta_1] = [15, 1.5]$. η_k is a random uniformly distributed additive noise, where $\eta_k \in \mathcal{U}\{-15, 15\}$ is added to all the samples as noise, and $\eta_k \in \mathcal{U}\{-50, 50\}$ is added to half of the samples as outliers. The fitted coefficients of the regression model and some bootstrapping performance criteria are given in Tab. 4.1. We observe that the values of the coefficients obtained for $N_b = 1$ are incorrect, thus highlighting the poor identifiability due to the presence of outliers (or rare events) in $\boldsymbol{y}^{*(j)}$ when the random observations are used for regression without bootstrapping. As the number of bootstrap samples increases, the coefficients tend towards the true value. The CI is calculated using the BCa method which gives an asymmetric lower and upper bound to the mean bootstrap fitted coefficient value. We observe that the 95% CI has a small range for $N_b = 10$ which can be attributed to lower number of samples which may not be sufficient to identify the full distribution of the parameters. However, as N_b increases, the CI range increases and the upper and lower bounds stabilize for $N_b = 100$ and $N_b = 1000$. A minor improvement in the RSS is observed as N_b increases, resulting from the improvement in the identification of the coefficients.

Table 4.1: Bootstrap fitted regression parameters, 95% confidence intervals (CI), and relative sum of squares (RSS) corresponding to different number of bootstrap samples N_b.

	$N_b = 1$		$N_b = 10$		$N_b = 100$		$N_b = 1000$	
	$\widehat{\beta_0}$	$\widehat{\beta_1}$	$\widehat{\beta_0}$	$\widehat{\beta_1}$	$\widehat{\beta_0}$	$\widehat{\beta_1}$	$\widehat{\beta_0}$	$\widehat{\beta_1}$
$\overline{\theta}^*$	13.57	1.53	13.83	1.52	14.87	1.51	15.04	1.51
CI	–	–	(12.64,17.34)	(1.49,1.53)	(11.13,18.29)	(1.46,1.57)	(11.32,18.64)	(1.44,1.58)
RSS	0.04413		0.04415		0.04409		0.04409	

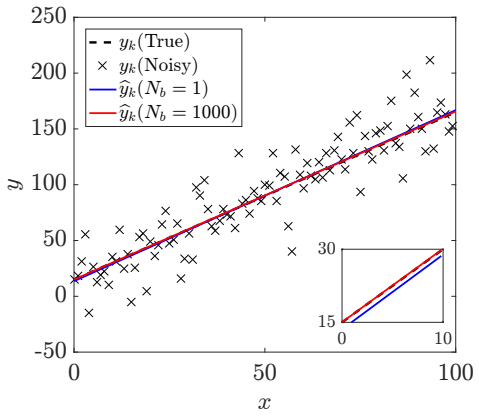


Figure 4.3: Result of bootstrapped linear regression fit for noisy observations using $N_b = 1$ and 1000 samples compared with the true solution. A magnified view in the inset figure shows the difference in the results of two regressed models.

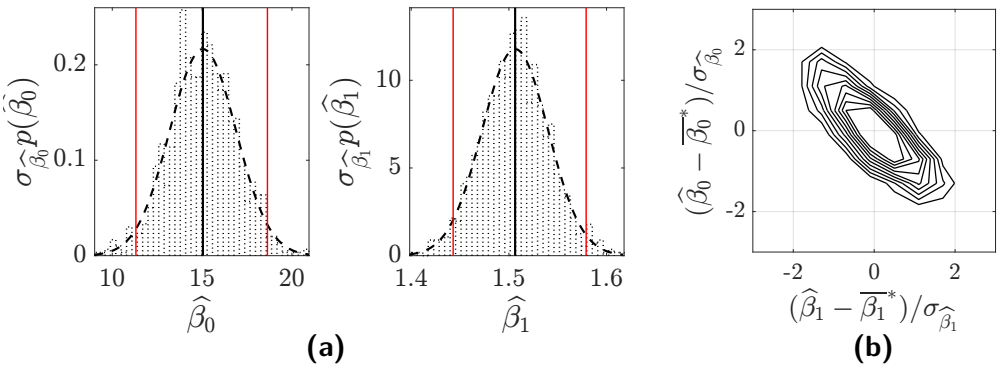


Figure 4.4: Stochastic description of the linear model parameters obtained from bootstrapping noisy observations with $N_b = 1000$ samples. (a) The unbiased bootstrap estimation $\overline{\beta_i}^*$ for $i = 1, 2$ (black solid line) are shown along with the bootstrap estimates (dotted bars), the Gaussian model approximation of the probability density of the parameter estimates (dashed black thick line), and the endpoints of the BCa confidence interval (red). (b) Normalized joint probability of the reduced centered coefficients (the contours correspond to values from 0.04 to 0.2 at increments of 0.02).

The results obtained with the regressed models for $N_b = 1$ and $N_b = 1000$ are shown in Fig. 4.3. The bootstrap fitted model is able to predict the true y_k, which follows from the close proximity of the true θ and identified $\overline{\theta}^*$ coefficients. The distributions of the bootstrapped regression coefficients obtained for $N_b = 1000$ are shown in Fig. 4.4 along with the percentile CI determined with the BCa method. The joint probability distribution of the coefficients gives an indication of the covariation between the bootstrap estimates of β_0 and β_1; the contours of joint pdf[11] indicate a clear negative correlation.

4.2.3 Block bootstrap

We now bring our attention to the objective of obtaining the statistics associated with the POD-ROM formulated as a system of linear equations (4.1). Here, the target $\dot{\boldsymbol{a}}_i$ and regressor $\boldsymbol{A}$ are constructed from the time-series data of the temporal POD coefficients $a_i(t_k)$. However, the ordinary bootstrap method discussed in Sec. 4.2.2 is appropriate only for a sample of i.i.d. observations. This hypothesis does not account for the inherent order which exists in a time series data and which can be mathematically represented by an autocorrelation function. Therefore, modifications to the previous bootstrap method must be made so that the resampling of the data preserve the time series dependency structure within a pseudo-sample.

An extensive review of the bootstrap methods for dependent data has been given by Kreiss and Paparoditis (2011). Out of the group of bootstrap procedures in the time domain, the *block bootstrap* procedure proposed by Hall (1985) has been shown to

[11] The joint probability distribution is obtained by calculating the bivariate histogram bin counts (for bin edges between -4 and 3.8 with an interval of 0.6 in each direction β_0 and β_1). The pdf estimate is given as the bin value $v_i = c_i/(N\,A_i)$ for each i-th bin where c_i is the number of elements in the bin, A_i is the bin area calculated using bin widths in each direction, and N is the number of elements in the input data.

correctly approximate the distribution of the statistics of interest (*e.g.* mean and variance) which can also be an estimator of some parameter. The block bootstrap method is a widely used bootstrap method in the domain of time series and is the method employed in the current work. In block bootstrap, the original time-series data is divided into N_b blocks of observations, each block with N_ℓ consecutive data points (also known as the block length). The bootstrap sample is constructed by randomly sampling the N_b blocks with replacement and concatenating them into a series of observations. In this way, a strong correlation is obtained within a block as compared to a relatively weaker correlation between the blocks.

4.2.3.1 Block bootstrap method

Different block bootstrap methods have been proposed in the literature in the context of bootstrapping time series data. These different methods attempt to reproduce different aspects of the dependence structure of the observed data in the resampled data. The most commonly used block bootstrap methods are:

- the moving block bootstrap (MBB), see Kunsch (1989);
- the nonoverlapping block bootstrap (NBB), see Carlstein (1986);
- the circular block bootstrap (CBB), see Politis and Romano (1992);
- the stationary bootstrap (SB), see Politis and Romano (1994).

All of these methods except SB use nonrandom *i.e.* constant block length. These bootstrap methods have been extensively discussed and compared by Lahiri (1999). It has been shown[12] that in terms of the mean-squared error (MSE) of a block bootstrap estimator, the MBB and the CBB estimators outperform the NBB estimators, which in turn outperform the SB estimators. We will subsequently describe the MBB and the CBB but first we briefly discuss the generalized block bootstrap.

We have $\boldsymbol{X}_n = \{X_1, \ldots, X_n\}$ as the available observations. Let $1 < N_\ell < n$ denote the block length for the block bootstrap methods. Given the time series $\boldsymbol{X}_n$, we define a new time series $\{Y_{n,i}\}_{i\geq 1}$ by *periodic extension*, where for $i \geq 1$, we have $Y_{n,i} = X_k$ when $i = mn+k$ for $m \geq 0$ (period index) and $1 \leq k \leq n$ (see Fig. 4.6). The blocks of length N_ℓ based on the new time series $\{Y_{n,i}\}$ are defined as $\boldsymbol{B}^{(j)} = \{Y_{n,i}, \ldots, Y_{n,(i+N_\ell-1)}\}$, $j \geq 1$. The different variants of block bootstrap methods are obtained by resampling from the subcollection of blocks $\{\boldsymbol{B}^{(j)} : j \geq 1\}$.

In the *MBB method*, the blocks are resampled with replacement from the subcollection $\{\boldsymbol{B}^{(j)} : j = 1, \ldots, N_{\text{mbb}}\}$ of overlapping blocks, where $N_{\text{mbb}} = n - N_\ell + 1$, as illustrated in Fig. 4.5. Let $I_1, \ldots, I_{N_{\text{mbb}}}$, be the conditionally i.i.d. random variables with the discrete uniform distribution on $\{1, \ldots, N_{\text{mbb}}\}$, *i.e.* the conditional probability[13] of

[12] All the four methods have the same amount of bias asymptotically. However, the leading terms in the variance part are not the same for all methods. The first-order terms in the expansions for the variances of the MBB and the CBB estimators are identical, making them equivalent in the MSE sense. In terms of the variance, the NBB estimators have an asymptotic efficiency of 2/3 compared to the corresponding MBB or CBB estimators. The variances of the SB estimators are always at least twice as large as the variances of the corresponding NBB estimators and at least three times as large as those of the MBB and CBB estimators.

[13] Here and in the next paragraph, we use P_* to denote the conditional probability.

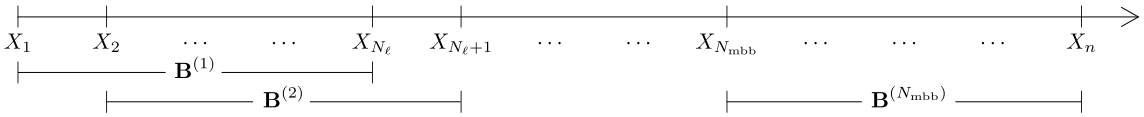


Figure 4.5: Collection of overlapping moving blocks $\{\boldsymbol{B}^{(j)} : j = 1, \ldots, N_{\mathrm{mbb}}\}$ for MBB.

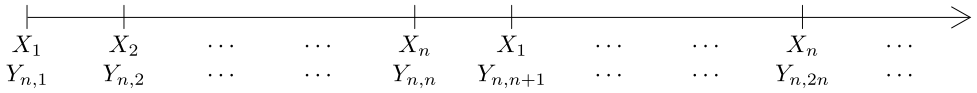


Figure 4.6: Periodically extended time series $\{Y_{n,i}\}_{i\geq 1}$ obtained from the observations $\boldsymbol{X}_n = \{X_1, \ldots, X_n\}$ for CBB.

each element is $P_*(I_1 = j) = 1/N_{\mathrm{mbb}}$ for $1 \leq j \leq N_{\mathrm{mbb}}$. Then the resampled blocks are drawn from $\boldsymbol{B}^{(j)}$ to give the blocks of MBB as $\{\boldsymbol{B}^{*(I_1)}, \ldots, \boldsymbol{B}^{*(I_{N_{\mathrm{mbb}}})}\}$. The bootstrap sample contains $N_{\mathrm{mbb}}N_\ell$ elements (*i.e.* N_{mbb} blocks of length N_ℓ) which are arranged in the sequence $X_1^{*(I_1)}, \ldots, X_{N_\ell}^{*(I_1)}, X_1^{*(I_2)}, \ldots, X_{N_\ell}^{*(I_2)}, \ldots, X_{N_\ell}^{*(I_{N_{\mathrm{mbb}}})}$.

In the *CBB method*, the blocks are resampled with replacement from the subcollection $\{\boldsymbol{B}^{(j)} : j = 1, \ldots, n\}$ of overlapping blocks. Therefore, in contrast to the MBB, the CBB uses elements from the periodically extended time series $\{Y_{n,i}\}_{i\geq 1}$ beyond $Y_{n,n}$, as illustrated in Fig. 4.6. Let $I_1, \ldots, I_{N_{\mathrm{cbb}}}$, where $N_{\mathrm{cbb}} \geq 1$ is the number of blocks of observed time series, be the conditionally i.i.d. random variables with the conditional probability $P_*(I_1 = j) = 1/n$, $j = \{1, \ldots, n\}$. Then the blocks of CBB drawn from $\boldsymbol{B}^{(j)}$ are obtained as $\boldsymbol{B}^{*(I_1)}, \ldots, \boldsymbol{B}^{*(I_{N_{\mathrm{cbb}}})}$ with the elements $X_1^{*(I_1)}, \ldots, X_{N_{\mathrm{cbb}}N_\ell}^{*(I_{N_{\mathrm{cbb}}})}$.

The MBB resampling scheme has an undesirable boundary effect as it assigns lesser weights to the observations at the start and end of the time series as compared to the observations in the middle of the dataset (Lahiri, 1999). Since there are no observations beyond the dataset, additional blocks cannot be defined to mitigate these boundary effects. In this work, the block booststrap method with CBB resampling scheme is used to obtain the estimates and confidence intervals of the unknown parameters of the POD-ROM (see Sec. 4.2.2). Precisely, the `CircularBlockBootstrap` function from the `arch` toolbox (Sheppard et al., 2020) is used for CBB resampling using blocks of the same length with end-to-start wrap around.

4.2.3.2 Block length

The performance of the block bootstrap method crucially depends on the choice of the block length N_ℓ. If N_ℓ is large, then there may be very few unique blocks available to form bootstrap samples. On the other hand, if N_ℓ is small, the observations in different blocks may not be independent and the i.i.d. bootstrapping of the blocks cannot be performed. The idea is then to select a block length large enough, so that the observations more than N_ℓ time steps apart are nearly independent, while the correlation between the observations less than N_ℓ time steps apart is retained. It should be noted that even for a properly chosen N_ℓ, the correlation between the observations in the block bootstrapped sample will be less than that in the original sample as the blocks are independent from each other.

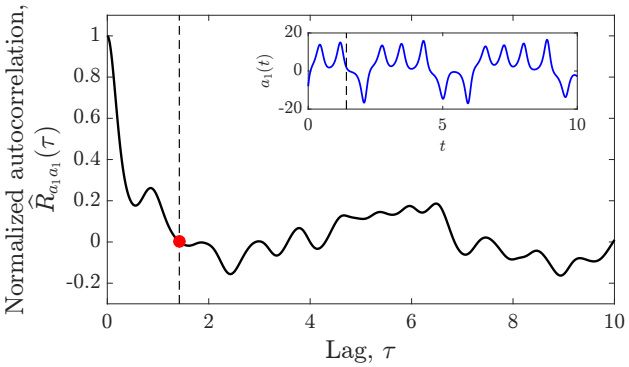


Figure 4.7: Optimal block length estimation for the CBB resampling of the state $a_1(t)$ of the Lorenz-63 system. The red dot at $\tau = 1.42$ corresponds to the optimal block length $N_\ell = 143$. For visual reference, the block size indicated by the dashed line is shown for the state dynamics in the inset figure.

A novel methodology for automatic selection of the block length was proposed by Politis and White (2004). For the selection, we look at the problem of approximating the sampling distribution of the sample mean $\overline{X} = (1/n)\sum_{t=1}^{n} X_t$ as both the population mean μ and population autocovariance sequence $R(\tau)$ are unknown. The autocovariance sequence is given as

$$R(\tau) = \frac{1}{n-\tau}\sum_{t=1}^{n-\tau}(X_t - \overline{X})(X_{t+\tau} - \overline{X}), \quad \forall\, 0 < \tau < n. \tag{4.17}$$

The normalized autocorrelation can be obtained as $\widehat{R}(\tau) = R(\tau)/\sigma_n^2$, where $\sigma_n^2 = \mathrm{var}(\sqrt{n}\,\overline{X}) = R(0) + 2\sum_{\tau=1}^{n}(1-\tau/n)R(\tau)$ is the sample variance. As the distribution of $\overline{X}$ is asymptotically normal, *i.e.* $\sqrt{n}(\overline{X} - \mu)$ converges to $\mathcal{N}(0, \sigma_\infty^2)$ as $n \longrightarrow \infty$, estimating the variance $\sigma_\infty^2 = \sum_{\tau=-\infty}^{\infty} R(\tau)$ is important. The optimal (expected) block length is obtained when the mean square error of the CBB estimates of σ_∞^2 is minimized.

The formal derivation of the block length can be found in Politis and White (2004) with some corrections done by Patton et al. (2009). This block length selection method is implemented in the `optimal_block_length` function from the `arch` toolbox (Sheppard et al., 2020), which has been used in this work. The application of the above steps to the time series of the state $a_1(t)$ as obtained in the Lorenz system studied in Sec. 4.3.2 is shown in Fig. 4.7. The optimal block length for the CBB resampling of the time series is obtained as $N_\ell = 143$. It can be seen that for this value of N_ℓ, the autocorrelation is zero which implies that the blocks of consecutive data of size N_ℓ will be mutually uncorrelated. In case of multivariate time series (*e.g.* $\boldsymbol{a}(t)$), we obtain block lengths $N_\ell^{(i)}$ for each of the $i = 1, \ldots, N_s$ states. Here, the state-wise maximum block length is used for CBB resampling of all the components of the multivariate time series as the block-wise standard deviation does not change with the excess block length (Flyvbjerg and Petersen, 1989).

4.3 Toy models

In this section, we describe the three toy models used to evaluate the performances of the system identification methods introduced in Sec. 3.1. The toy models conform to the formulation of POD-ROM without the nonlinear residual term, given by (4.1).

4.3.1 Toy model 1: Three-dimensional linear system

A three-dimensional linear system offers a simple dynamics which helps to establish the capabilities of the different system identification strategies. The dynamics is obtained from the solution of a system of $N_s = 3$ linear homogeneous ordinary differential equations (ODEs) of the form

$$\begin{cases} \dot{a}_1(t) = -\sigma a_1(t) - \rho a_2(t), \\ \dot{a}_2(t) = \quad \rho a_1(t) - \sigma a_2(t), \\ \dot{a}_3(t) = -\beta a_3(t), \end{cases} \tag{4.18}$$

where the parameters are $[\sigma\ \rho\ \beta] = [0.1\ 2.0\ 0.3]$ and the initial state is given as $\boldsymbol{a}_0 = [a_1(0)\ a_2(0)\ a_3(0)]^\mathbb{T} = [2\ 2\ 1]^\mathbb{T}$. As we are interested in second order dynamical system, the polynomial library $\boldsymbol{A}$ is constructed[14] with polynomials up to second order in terms of $\boldsymbol{a}$. The number of parameters, associated with each state evolution equation, is $N_{\theta_i} = 10$. We observe from (4.18) that for this linear system, the parameter matrix $\boldsymbol{\theta} \in \mathbb{R}^{N_{\theta_i} \times N_s}$ is sparse; the only nonzero elements are $L_{11} = L_{22} = -\sigma$, $L_{12} = -\rho$, $L_{21} = \rho$, and $L_{33} = -\beta$. The state dynamics is obtained by integrating the system of ODEs using the LSODA integrator[15]. The evolution of the state trajectories is shown in Fig. 4.8.

4.3.2 Toy model 2: Lorenz-63 system

The second test case considered is the very familiar Lorenz-63 nonlinear system. This system was developed to describe the phenomenon of natural convection in a rectangular cavity with a heated lower wall. It leads to a quadratic system with $N_s = 3$ ODEs in terms of $\boldsymbol{a}$ of the form

$$\begin{cases} \dot{a}_1(t) = \sigma(a_2(t) - a_1(t)), \\ \dot{a}_2(t) = a_1(t)(\rho - a_3(t)) - a_2(t), \\ \dot{a}_3(t) = a_1(t)a_2(t) - \beta a_3(t). \end{cases} \tag{4.19}$$

For the assigned parameter values of $[\sigma\ \rho\ \beta] = [10\ 28\ 8/3]$, the Lorenz system exhibits a chaotic dynamics characterized by a limit cycle and a strange attractor as shown

[14] This is not necessary in this case since we know that the dynamical system to be identified is linear. Considering a general case allows us to test the efficiency of the identification methods to evaluate a sparse representation.

[15] LSODA (Petzold, 1983) differs from the solver LSODE (Livermore Solver for Ordinary Differential Equations) on which it is based in that it switches automatically between stiff (BDF) and non-stiff (Adams) methods. It uses the non-stiff method initially, and dynamically monitors data in order to decide which method to use. The LSODA integrator is provided by the FORTRAN library ODEPACK and is accessible through the `solve_ivp` function of `scipy`.

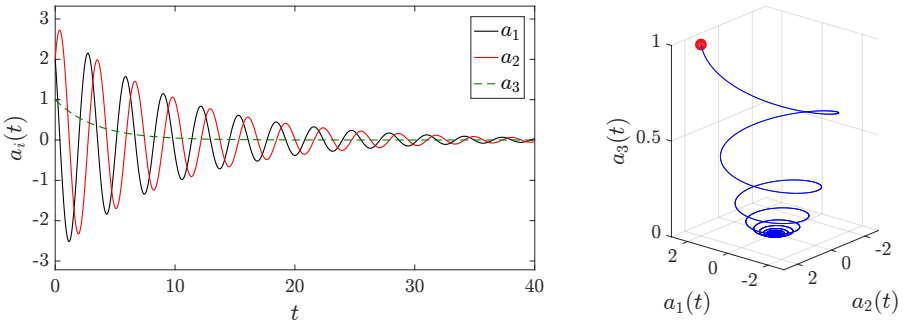


Figure 4.8: Trajectories of the states $a_i(t)$ $(i = 1, 2, 3)$ of a three-dimensional linear system. The initial state $\boldsymbol{a}(0)$ is indicated by a red dot in the trajectory of the combined evolution (right panel).

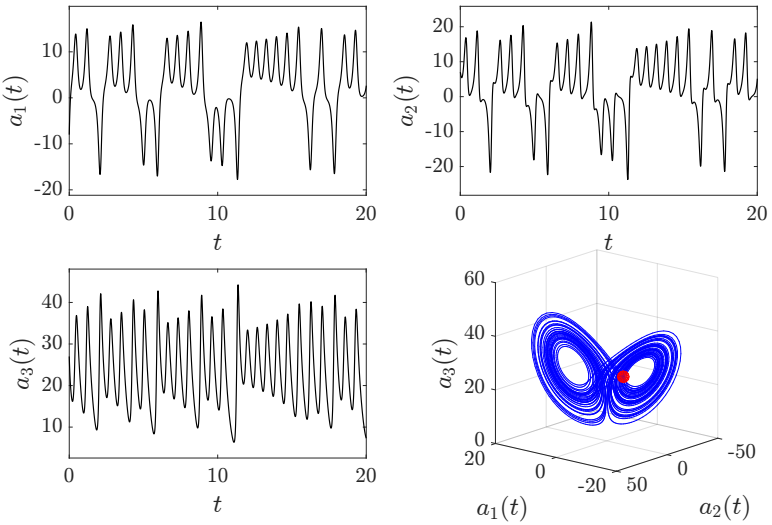


Figure 4.9: Trajectories of the states $a_i(t)$ $(i = 1, 2, 3)$ of the Lorenz-63 system.

in Fig. 4.9. The initial state is set as $\boldsymbol{a}_0 = [a_1(0)\ a_2(0)\ a_3(0)]^\top = [-8\ 7\ 27]^\top$. The quadratic polynomial library $\boldsymbol{A}$ is used. The parameter matrix $\boldsymbol{\theta}$ is sparse.

4.3.3 Toy model 3: Lorenz-96 system

To evaluate the performance of the identification strategies for systems with higher dimensions, the Lorenz-96 system is considered. This system of ODEs describes a single scalar quantity as it evolves on a circular array of sites, undergoing forcing, dissipation, and advection. It was initially introduced as a test problem for numerical weather prediction. The dynamical system corresponding to the classical N_s dimensional Lorenz-96 system is given as

$$\dot{a}_i(t) = (a_{i+1}(t) - a_{i-2}(t))a_{i-1}(t) - a_i(t) + F, \quad \forall\, i = 1, \ldots, N_s, \tag{4.20}$$

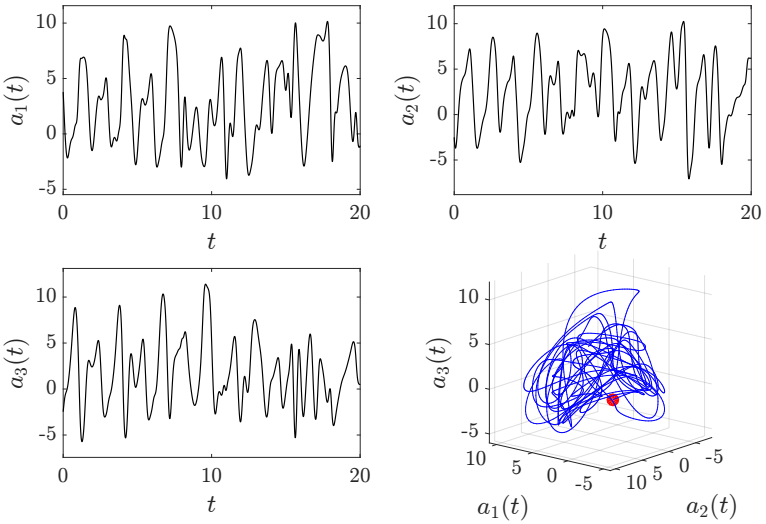


Figure 4.10: Trajectories of the first three states $a_i(t)$ $(i = 1, 2, 3)$ of the Lorenz-96 system of dimension $N_s = 10$.

where it is assumed that $N_s \geq 4$, $a_{-1}(t) = a_{N_s-1}(t)$, $a_0(t) = a_{N_s}(t)$, and $a_{N_s+1}(t) = a_1(t)$. This system belongs to a class of problems with a single bifurcation parameter (rescaled forcing term F). The forcing parameter is set as $F = 8$ which is known to cause chaotic behavior as shown in Fig. 4.10. The initial condition is $\boldsymbol{a}(0) = \mathbf{0}_{N_s \times 1}$ with the last element perturbed such that $a_{N_s}(0) = 0.01$. The number of states N_s can be varied to obtain models of different dimensions. Similar to the previous test cases, we consider a quadratic polynomial library $\boldsymbol{A}$ with a sparse parameter matrix $\boldsymbol{\theta}$ for the system identification.

4.3.4 Artificial noise

In order to simulate physical processes, additive noise is introduced to the numerical solution of the dynamical systems discussed above. The noise is defined by noise level $\eta \geq 0$, and a sequence of uniformly distributed random numbers $\varepsilon_u(t) \sim \mathcal{U}\{-1, 1\}$ of the same length as the signal. In the following, the state variable $a_i(t)$ is modified as

$$a_i(t) \leftarrow a_i(t) + \eta \varepsilon_u(t) \Delta a_i, \quad \forall\, i = 1, \ldots, N_s, \tag{4.21}$$

where $\Delta a_i = \max(a_i(t)) - \min(a_i(t))$ is the range of the time signal over the period of observation. Noise levels of $\eta = 0\%$ (no noise), 5% (low), and 10% (high) have been considered in this work.

4.4 Results

In this section, the result of system identification of the dynamical systems (Sec. 4.3) with different noise levels $\eta = 0\%$, 5% and 10% will be evaluated. The identification is performed using the system identification methods discussed in Sec. 3.1 and the

numerical tools discussed in Sec. 4.1. Initially, the three regression methods – OLS, SINDy, and LARS – are used to identify the three-dimensional linear system (Sec. 4.4.2) and their relative performance is assessed. In the subsequent sections, the SINDy and LARS methods are applied to the chaotic systems of Lorenz-63 (Sec. 4.4.3) and Lorenz-96 (Sec. 4.4.4).

4.4.1 Training and testing datasets

As a common practice in the statistical system identification approach, the time series data are usually split into two separate sets, known as the *training* and *testing* sets, before performing the regression. The multivariate dataset $\boldsymbol{X} = [\boldsymbol{a}_1\, \boldsymbol{a}_2\, \ldots\, \boldsymbol{a}_{N_s}] \in \mathbb{R}^{N_t \times N_s}$ with the elements $X_{k,j} = a_j(t_k)$, is split in order to form a training set

$$\boldsymbol{X}^{\text{Train}} = \{X_{k,:}\}_{k=1}^{N_t^{\text{Train}}} \in \mathbb{R}^{N_t^{\text{Train}} \times N_s}, \tag{4.22}$$

where N_t^{Train} is the number of data snapshots assigned to the training set, and a testing set

$$\boldsymbol{X}^{\text{Test}} = \{X_{k,:}\}_{k=N_t^{\text{Train}}+1}^{N_t} \in \mathbb{R}^{N_t^{\text{Test}} \times N_s}, \tag{4.23}$$

where $N_t^{\text{Test}} = N_t - N_t^{\text{Train}}$ is the remaining number of data snapshots in the original dataset. We note that the time series are not shuffled and the order of the time sequence is maintained in both the subsets.

We use the training dataset $\boldsymbol{X}^{\text{Train}}$ in the system identification step to fit the regression model. This requires an alteration of the model such that $\boldsymbol{A} \in \mathbb{R}^{N_t^{\text{Train}} \times N_{\theta_i}}$ and $\dot{\boldsymbol{a}}_i \in \mathbb{R}^{N_t^{\text{Train}} \times 1}$ correspond to the data in the training dataset. The testing dataset $\boldsymbol{X}^{\text{Test}}$ is used to evaluate the fit of the regressed model. The training and testing sets are vital to ensure that the model does not overfit to the data used for the identification and is able to generalize well to new data. A prediction $\boldsymbol{a}^{\text{Pred}}(t)$ is obtained by integrating the *trained* model in the time window $[t_{N_t^{\text{Train}}+1}, t_{N_t}]$ using $\boldsymbol{a}(t_{N_t^{\text{Train}}+1})$ as the initial condition. To evaluate the performance of the system identification methods, the prediction error between $\boldsymbol{a}^{\text{Test}}(t)$ and $\boldsymbol{a}^{\text{Pred}}(t)$ is obtained by calculating the normalized mean square error corresponding to each state variable and averaging over the states,

$$E_{\text{Pred}} = \frac{1}{N_s} \sum_{i=1}^{N_s} \sqrt{\frac{\|\boldsymbol{a}_i^{\text{Test}}(t) - \boldsymbol{a}_i^{\text{Pred}}(t)\|_2^2}{\left\|\boldsymbol{a}_i^{\text{Test}}(t) - \overline{\boldsymbol{a}_i^{\text{Test}}}\right\|_2^2}}, \tag{4.24}$$

where $\overline{\boldsymbol{a}_i^{\text{Test}}}$ is the temporal mean state vector of the reference test data. The prediction error quantifies the ability of the learned model to predict the dynamics when it is supplied with a new dataset outside the training set. In this work, for sufficiently large datasets ($N_t \gtrsim 1000$), we consider $N_t^{\text{Train}} = \lfloor 0.7 N_t \rfloor$.

4.4.2 Toy model 1: Three-dimensional linear system

The dataset for the three-dimensional linear system is generated by integrating the system of ordinary differential equations (4.18) using the LSODA integrator (Petzold, 1983).

The dataset consists of $N_t = 3000$ discrete snapshots at regular time steps $\Delta t = 0.01$. As discussed in Sec. 4.4.1, the dataset is divided into training and testing sets with $N_t^{\text{Train}} = 2100$ and $N_t^{\text{Test}} = 900$ snapshots, respectively. We perform system identification on the original dataset (without noise, $\eta = 0\%$) and two additional datasets obtained by introducing additive noise of levels $\eta = 5\%$ and 10% using (4.21). For the dataset without noise, the system (4.1) is constructed by using the time series $\boldsymbol{a}(t)$ to assemble the matrix of basis functions $\boldsymbol{A}$, and by utilizing the fourth-order accurate finite difference scheme to obtain the time derivatives $\dot{\boldsymbol{a}}_i(t)$. On the other hand, for the datasets with noise, the noisy time series $\boldsymbol{a}(t)$ are first passed through a LP filter (see Sec. 4.1.1) to remove the high-frequency contribution of the noise. The filtered time series are then used to assemble the matrix $\boldsymbol{A}$, while the TVreg differentiation method, described in Sec. 4.1.2, is employed to obtain $\dot{\boldsymbol{a}}_i(t)$ directly from the noisy data. The LP filter and TVreg differentiation contribute to minimizing the influence of the non-physical artefacts introduced in the system (4.18) by the noise. These techniques are of great help for recovering the true but unknown underlying dynamics (fitting the systematic component rather than the noise). Also, for the identification by LARS, we use the standardized form of the time derivative vector and basis function matrix, $\widehat{\dot{\boldsymbol{a}}}_i(t)$ and $\widehat{\boldsymbol{A}}$ respectively, to identify the corresponding parameters $\widehat{\boldsymbol{\theta}}_i$ (see Sec. 3.1.3.2). The parameters are not invariant to the standardization and are therefore not equivalent to the true parameters $\boldsymbol{\theta}_i$. However, $\boldsymbol{\theta}_i$ can be retrieved from $\widehat{\boldsymbol{\theta}}_i$ with the help of the transformation (3.26).

As discussed in Sec. 4.2.3, the circular block bootstrap (CBB) method is used to identify the model parameters in a probabilistic framework. The training dataset is used to obtain blocks of consecutive data points. The block length N_ℓ is obtained automatically using spectral estimation, as discussed in Sec. 4.2.3. The blocks are resampled with replacement to obtain $N_{\text{cbb}} = 1000$ bootstrap samples. The system (4.1) is identified for each of the N_{cbb} samples. The learned parameters $\boldsymbol{\theta}_i$ are obtained as the mean over the sample estimates.

The scaled color representation of the true and learned parameter matrix $\boldsymbol{\theta}$ are shown in Fig. 4.11 for the OLS, SINDy and LARS identification methods. For the dataset without noise ($\eta = 0\%$), we observe that the coefficients obtained from the three system identification methods replicate the true sparse parameter matrix. As the noise level increases ($\eta = 5\%$ and 10%), additional nonzero coefficients enter the parameter matrix apart from the ones appearing in the true system. This behavior will be discussed in the next paragraph. The BCa method described in Sec. 4.2.2 can be used to obtain the confidence interval (CI) of the empirical distribution of the bootstrap parameter estimates. As an example, the 95% CI of the coefficients obtained from LARS for the dataset corresponding to the noise level $\eta = 10\%$ is shown in Fig. 4.12. We observe that the mean values are not centered with respect to the CI which is common when the percentile based BCa estimation of the CI is applied to skewed distributions. The distributions of some selected significant nonzero coefficients (L_{11}, L_{22} and L_{33}) are visually presented with the aid of box plots in Fig. 4.13. The plot indicates that the distribution of the bootstrap estimates is skewed (non-Gaussian), *e.g.* considering the coefficient L_{33} estimated for the dataset corresponding to the noise level $\eta = 5\%$ for SINDy and LARS. This type of non-normal distribution necessitates the use of the bias corrected methods like BCa to estimate correctly the CI.

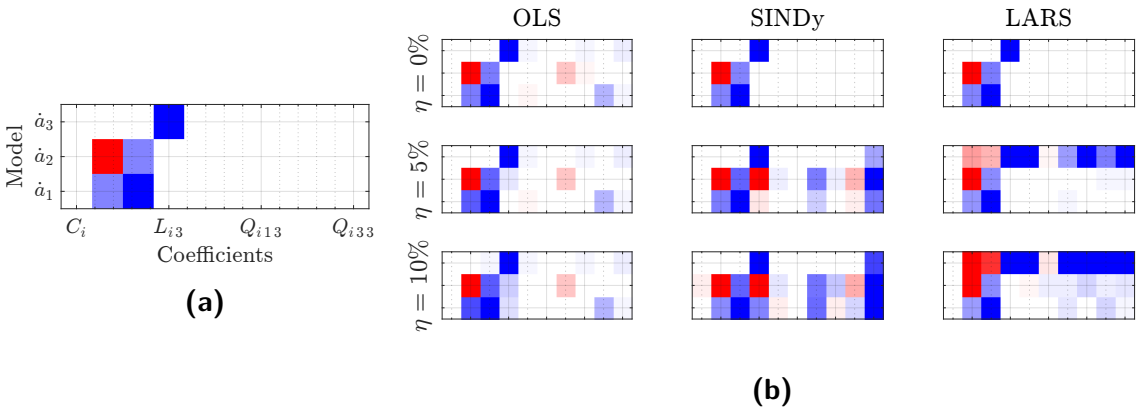


Figure 4.11: Scaled color representation of the true (a), and learned (b) parameters for the three-dimensional linear system. In (b), we compare the results obtained by OLS, SINDy and LARS identification methods. Each row corresponds to the parameter vector $\boldsymbol{\theta}_i$ consisting of the coefficients of the model for $\dot{a}_i(t)$, where $i = 1, 2, 3$. Note that the standardized parameters are plotted for the LARS results. The negative and positive parameter values are indicated by blue and red, respectively, and fade to white as the value tends to zero.

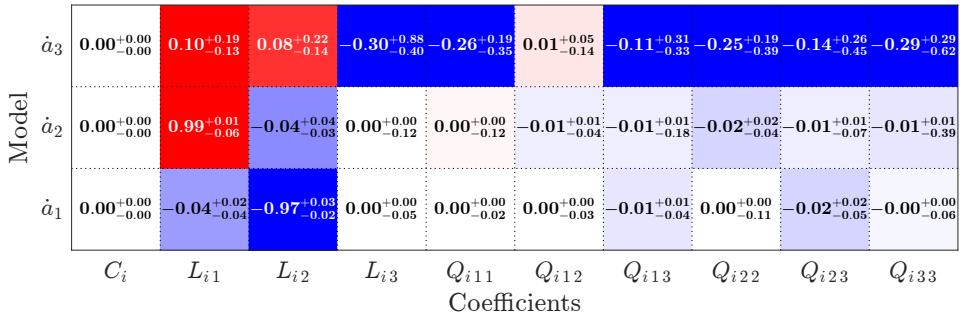


Figure 4.12: Scaled color representation of the (standardized) learned parameters from LARS for the three-dimensional linear system with additive noise $\eta = 10\%$. The values correspond to the bootstrap mean of the parameters along with the difference from the lower (subscript) and upper (superscript) bounds of the 95% confidence interval obtained from the BCa method.

For the noisy datasets, the OLS method yields a sparse parameter matrix which is close to the true one. In contrast, a penalized least-squares (PLS) method (see App. C.2) introduces non-zero values for parameters that should be zero. This difference in the results can be attributed to the regression principles of OLS and PLS methods. Using OLS, we estimate the coefficients of the linear model by minimizing (3.10), the squared difference from the target $\dot{\boldsymbol{a}}_i(t)$. On the other hand, PLS consists of solving a minimization problem which includes a penalty term constraining the coefficients. Considering an example of ridge regression, the PLS solution for a smooth finite dimensional problem is given by (C.24); re-written in terms of the relevant variables as

$$\boldsymbol{\theta}_i = (\boldsymbol{A}^\top \boldsymbol{A} + \lambda I_{N_{\theta_i}})^{-1} \boldsymbol{A}^\top \dot{\boldsymbol{a}}_i. \tag{4.25}$$

In (4.25), if $\lambda = 0$, the OLS solution is obtained which is an unbiased estimator (specifically minimum-variance linear unbiased estimator) according to Gauss–Markov theorem.

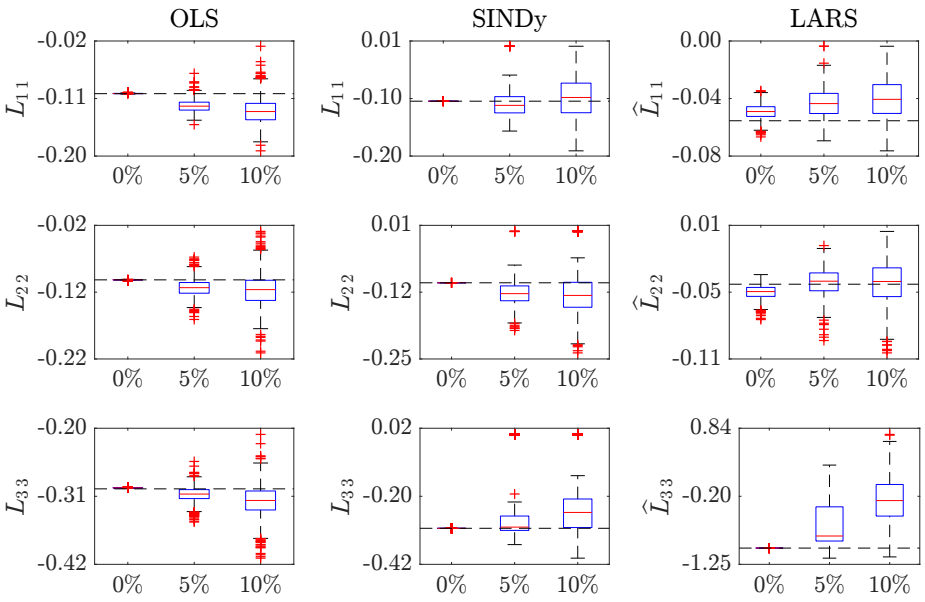


Figure 4.13: Box plots of the learned coefficients by OLS, SINDy and LARS along with the true value (dashed horizontal line) for the three-dimensional linear system with noise levels $\eta = 0\%$, 5% and 10%. The standardized coefficients obtained from LARS are indicated as $\widehat{\square}$. The central line, represented in red, of the blue box indicates the median value. The bottom and top edges of the blue box indicate the 25th and 75th percentiles, respectively. The whiskers extend to the most extreme data points without considering outliers. The outliers are plotted individually using the "+" symbol.

On the other hand, a nonzero term $\lambda I_{N_{\theta_i}}$ introduces a bias in the PLS solution which leads to a difference between the estimator's expected value and the true value of the parameter being estimated. A trade-off between bias and variance is obtained by manipulating λ. Despite the unbiased estimation, the OLS is not preferred as it is not guaranteed to give a lower value of the loss function (3.10) as compared to PLS. Moreover, it risks overfitting the target values. The optimal prediction accuracy is obtained by using a nonzero λ which introduces bias to reduce variance. Another class of PLS methods based on L_1 regularization offer many of the beneficial properties of the ridge regression, but yields sparse models that are more easily interpreted (Hastie et al., 2009). In this thesis, the L_1 regularization based methods of SINDy (Sec. 3.1.2) and LARS (Sec. 3.1.3) are used. As already discussed, these PLS methods are able to solve the linear systems when the number of unknown parameters exceeds the number of target values $N_\theta > N_t$. In such settings, the OLS is ill-posed as the associated optimization problem has infinitely many solutions.

The performance of the bootstrap estimate of the parameters can be investigated in the training and testing windows. For the study in the training window $[t_1, t_{N_t^{\text{Train}}}]$, samples of the parameter matrix $\boldsymbol{\theta}$ are drawn from a uniform distribution over the 95% confidence interval obtained from the BCa method. We generate in this way $N_{\text{Prop}} = 200$ sets of parameters which are used to propagate an initial state in the training window. As a result, we obtain a series of equiprobable trajectories in the phase space. The range

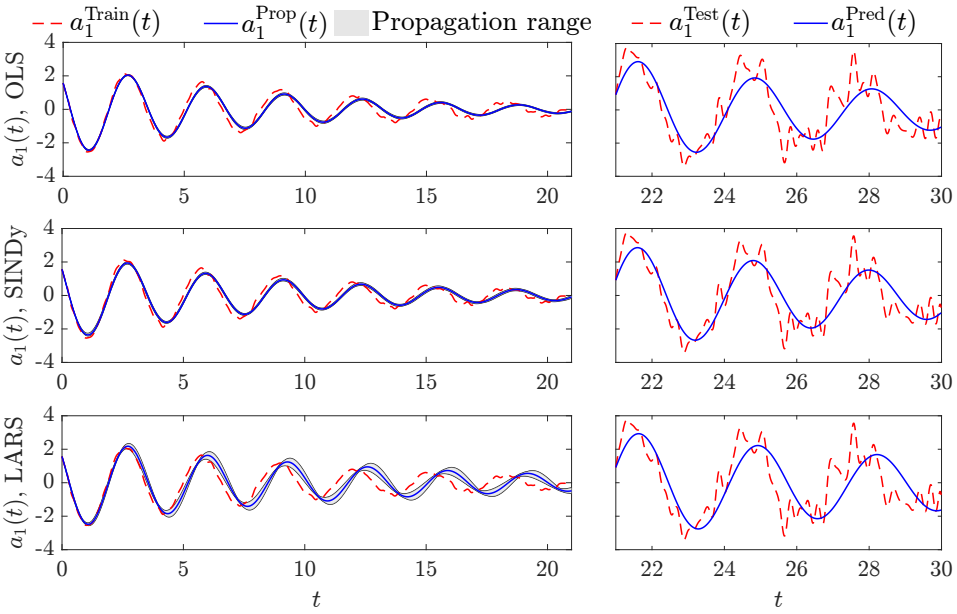


Figure 4.14: Bootstrap analysis of the time evolution of the state $a_1(t)$ for different regression methods (OLS, SINDy, and LARS). Case of the three-dimensional linear system with an additive noise ($\eta = 10\%$). The two columns correspond to the propagation in the training window (left), and the prediction in the testing window (right). The gray shaded area represents the region between the minimum and maximum values of the evolution of the state obtained by integrating 200 bootstrap dynamical models. The dashed red line corresponds to the dynamics in the training and testing windows, respectively. The solid blue line corresponds to the dynamics for the mean bootstrap parameters in the training and testing windows, respectively.

of the propagated trajectories is shown by the shaded region in the training window in Fig. 4.14. For the three system identification methods considered, we observe that the state propagation remains bounded and correlated to the dynamics of the training data over the time span of the training window. The mean bootstrap estimate of the parameter is also used to propagate the states in the training window as shown in Fig. 4.14. The goodness of fit of this trajectory $\boldsymbol{a}_i^{\text{Prop}}(t)$ to the training data $\boldsymbol{a}_i^{\text{Train}}(t)$ is evaluated using the averaged normalized root mean square error E_{Prop} (propagation error). This quantity is calculated in the same way as the prediction error (4.24) *i.e.*

$$E_{\text{Prop}} = \frac{1}{N_s}\sum_{i=1}^{N_s}\sqrt{\frac{\left\|\boldsymbol{a}_i^{\text{Train}}(t) - \boldsymbol{a}_i^{\text{Prop}}(t)\right\|_2^2}{\left\|\boldsymbol{a}_i^{\text{Train}}(t) - \overline{\boldsymbol{a}_i^{\text{Train}}}\right\|_2^2}}. \tag{4.26}$$

In the testing window $[t_{N_t^{\text{Train}}+1}, t_{N_t}]$, the states trajectory is obtained by integrating the model with the mean bootstrap estimate of the parameter from the initial state $\boldsymbol{a}^{\text{Test}}(t_{N_t^{\text{Train}}+1})$. The prediction error E_{Pred} is then calculated. The predicted trajectories obtained from integration of the learned model using the three system identification methods is shown in the testing window in Fig. 4.14. We observe that the prediction using

the bootstrap estimation of the parameters closely follows the expected time evolution of the reference state. Let R^2 denote the *coefficient of determination i.e.* the proportion of total variation of the test data explained by the learned model. For the prediction trajectory, this quantity is given as

$$R^2 = 1 - \frac{1}{N_s}\sum_{i=1}^{N_s}\left[\frac{\sum_{k=t_{N_t^{\text{Train}}+1}}^{N_t}(a_i^{\text{Test}}(t_k) - a_i^{\text{Pred}}(t_k))^2}{\sum_{k=t_{N_t^{\text{Test}}+1}}^{N_t}(a_i^{\text{Test}}(t_k) - \overline{a_i^{\text{Train}}})^2}\right]. \tag{4.27}$$

The value of R^2 provides a measure of the strength of the relationship between the test data and the dynamics replicated by the model based on the parameters learned in the training window. By definition, R^2 ranges from 0 to 1. In general, the higher the value of R^2, the better the model explains all the variability of the test data around its mean. It must be noted that R^2 cannot determine whether the predictions are biased and therefore serve only as an intuitive measure of the goodness of fit.

The performance metrics E_{Prop}, E_{Pred} and R^2 in the training and testing windows are listed in Tab. 4.2 for the noise levels $\eta = 0\%$, 5% and 10%. In the training window, the performance of the three identification methods is almost identical, yielding the same order of magnitude for E_{Prop}. Also, the errors corresponding to the noisy data are higher than that for the data without noise. In terms of actual value of the error, the OLS method marginally outperforms the SINDy method while the LARS method yields the least accurate results amongst the three.

In the testing window, for the data without noise, the magnitudes of E_{Pred} corresponding to the SINDy and LARS methods are lower than that for the OLS method. However, for the LARS method, the values of E_{Pred} obtained for the noisy data are higher than those corresponding to the OLS and SINDy methods. This behavior is typical of cases where the LARS method is used to identify dynamics where the amplitude decays or grows in time. Indeed, as the LARS method relies on standardization of the target vector and the basis function matrix (see Sec. 3.1.3.2), the prediction in the testing window requires to be unstandardized using the statistics of the training dataset. For decaying or growing dynamics, the variance σ^2 evolves with time. The use of the variance obtained in the training window to rescale the results in the testing window then leads to the bias. We also observe in Fig. 4.14 the influence of the standardization in the form of a small phase difference in the prediction for LARS. The R^2 scores corresponding to the different noise levels are approximately equal for the three identification methods. The R^2 values closer to 1 for $\eta = 0\%$ and 5% indicate that the predicted trajectories are able to explain the variability of the dynamics in the testing window. The lower R^2 score for $\eta = 10\%$ indicates the influence of the noise on the biased bootstrap estimation of the coefficient values; the learned model, however, is still able to predict the dynamics in the testing window as shown in Fig. 4.14.

The computational costs associated with the three methods are compared in the training window for 100 bootstrap samples and $\eta = 0\%$. The total elapsed time and the number of bootstrap samples per second are reported in Tab. 4.3. It is observed that OLS is one order of magnitude slower than the SINDy and LARS identification. This is attributed to the cost of calculation of the SVD (3.11) for OLS which has the computational complexity of $\mathcal{O}(\min(N_t N_{\theta_i}^2, N_{\theta_i} N_t^2))$ which is higher than the complexity

Table 4.2: Performance of the system identification methods (OLS, SINDy, and LARS) for the three-dimensional linear system with noise levels $\eta = 0\%$, 5% and 10%.

η	OLS			SINDy			LARS		
	E_{Prop}	E_{Pred}	R^2	E_{Prop}	E_{Pred}	R^2	E_{Prop}	E_{Pred}	R^2
0%	0.0037	0.0496	1.0000	0.0060	0.0024	1.0000	0.0093	0.0102	0.9998
5%	0.1892	0.7622	0.9032	0.1902	0.7039	0.9122	0.3645	1.1924	0.9064
10%	0.2395	0.8999	0.6685	0.2366	0.8675	0.6961	0.3003	0.9970	0.6876

Table 4.3: Total elapsed time and number of bootstrap samples per second for the three-dimensional linear system in the training window (100 bootstrap samples and $\eta = 0\%$).

	OLS	SINDy	LARS
Elapsed time (s)	11.486	0.926	1.102
Bootstrap samples per second	8.619	106.926	89.845

associated with the iterative convex optimization based methods of SINDy and LARS. The higher cost associated with OLS makes it computationally intractable for the problems with higher dimensions or higher number of samples.

In the subsequent examples, due to the higher costs associated with the OLS method, we only discuss the system identification using the SINDy and LARS methods.

4.4.3 Toy model 2: Lorenz-63 system

In this section, the relative performance of the SINDy and LARS based system identification methods are assessed for the Lorenz-63 system in chaotic regime. The dataset used for the identification is generated by integrating the system of ordinary differential equations (4.19). The hyperparameters associated with the data generation and bootstrap estimation are assigned the same values as in the case of the three-dimensional linear system discussed in Sec. 4.4.2. For the LARS algorithm, the AIC model selection criteria and LARS solution path is demonstrated in Demo. 4.2.

Demo 4.2: LARS identification of Lorenz-63 system

The selection of λ_{opt} for the LASSO model fit of the Lorenz-63 system using LARS with AIC is shown in Fig. 4.15. The regularization parameter λ is updated along with the iterations of LARS and λ_{opt} corresponding to the minimum value of AIC is used in the LASSO fit. We observe that the AIC increases with the increasing noise levels. This can be attributed to the higher contribution to the residual term, *i.e.* the first term on the right-hand side of (3.29), due to perturbation in the dataset. The AIC obtained from the datasets corresponding to different bootstrap samples $\boldsymbol{\Xi}^{*(j)}$ also varies but the selection of λ_{opt} remains bounded within the same order of magnitude. Fig. 4.16 shows the stepwise algorithm of LARS used to trace out the LASSO solution path. LARS computes a path solution only for each kink in the path which can be

seen in the LASSO path as addition or deletion of the parameters from the active set. The LASSO path is able to identify the parameters in the active set that correspond to the ones in the true dynamical model. We observe that for higher noise level, more number of LARS updates are performed and additional nonzero coefficients apart from the ones in the true model remain the identified active set. However, the dominant coefficients still correspond to the ones in the true model.

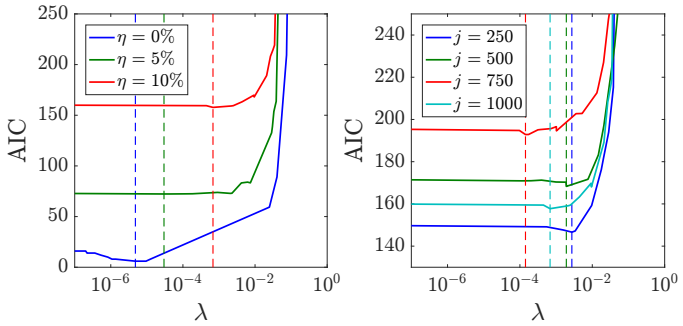


Figure 4.15: Plot of AIC against the regularization parameter λ for LASSO estimator of the dynamics of state $a_1(t)$ of the Lorenz-63 system. AIC is compared for datasets with different noise levels η for an arbitrarily selected bootstrap sample $j = 1000$ (left), and different bootstrap samples (denoted by index j) for the dataset with noise level $\eta = 10\%$ (right). The dashed vertical lines indicate the values of λ corresponding to the minimum AIC.

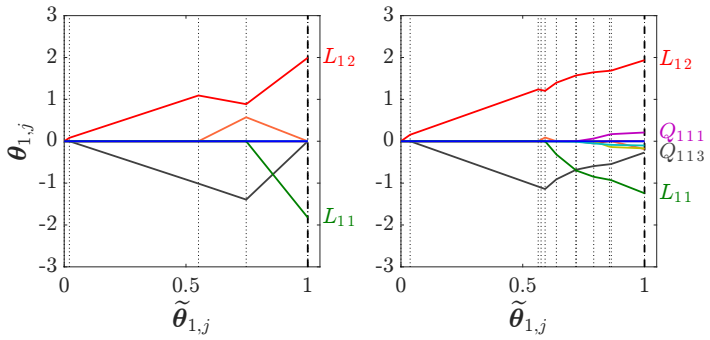


Figure 4.16: LARS solution path of the coefficients $\boldsymbol{\theta}_{1,j}$ plotted against the normalized coefficients $\widetilde{\boldsymbol{\theta}}_{1,j} = \sum_j |\boldsymbol{\theta}_{1,j}| / \max(\sum_j |\boldsymbol{\theta}_{1,j}|)$ for LASSO estimator of the dynamics of state $a_1(t)$ of the Lorenz-63 system with different noise levels: $\eta = 0\%$ (left) and $\eta = 10\%$ (right). The dotted vertical lines indicate the locations where the parameters are added or removed from the active set. The dashed vertical line represents the location corresponding to the LASSO model fit obtained with LARS using AIC.

The scaled color representation of the true and learned parameter matrix $\boldsymbol{\theta}$ are shown in Fig. 4.17. For the case without noise ($\eta = 0\%$), the SINDy and LARS methods lead to a parameter structure matrix equivalent to that of the true parameters. However, for the case with noise, both identification methods introduce nonzero values in the parameter matrix for the coefficients which are zero valued in the true parameter matrix. The

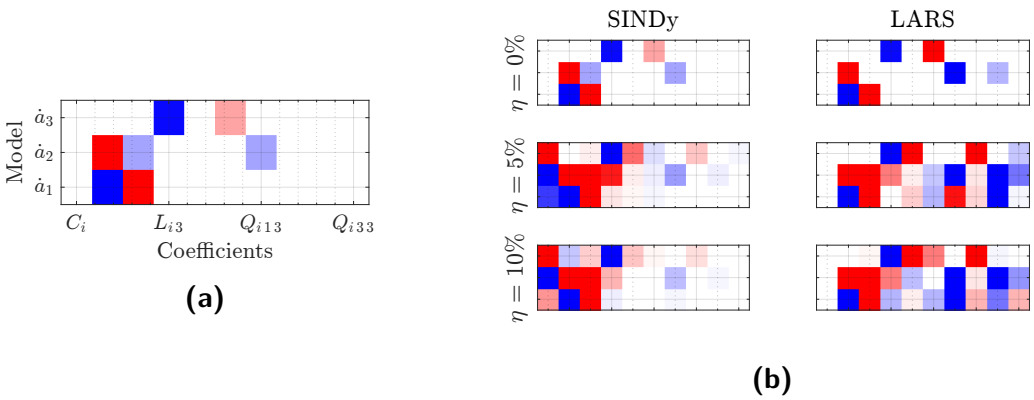


Figure 4.17: Scaled color representation of the true (a), and learned (b) parameters for the Lorenz-63 system. In (b), we compare the results obtained by SINDy and LARS identification methods. Refer to the caption of Fig. 4.11 for a detailed description.

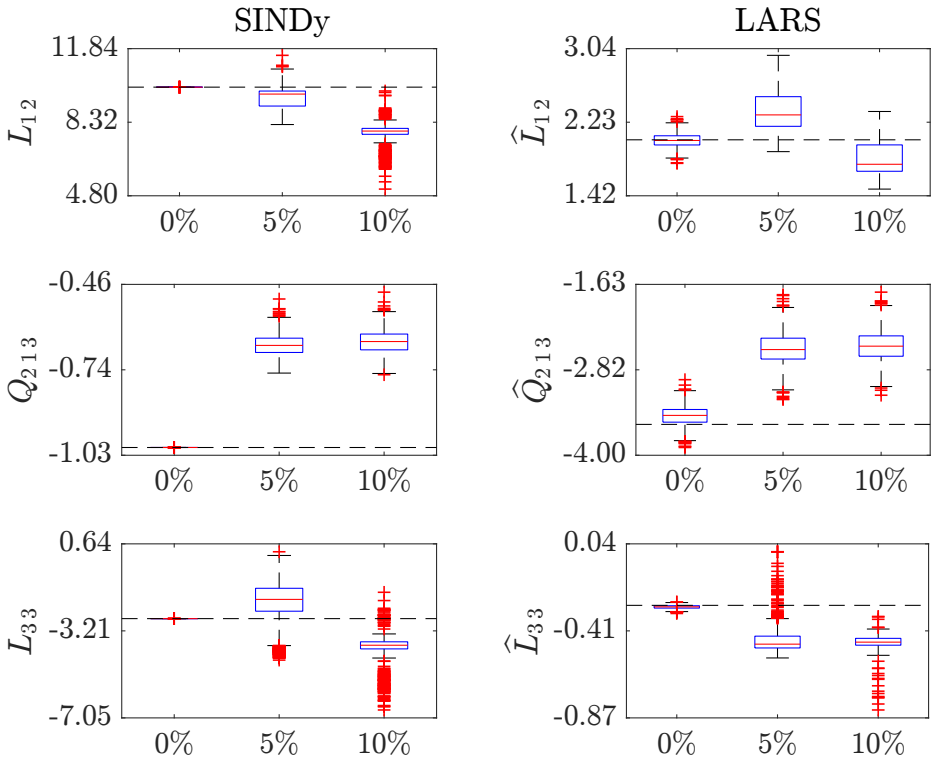


Figure 4.18: Box plots of the learned coefficients by SINDy and LARS along with the true value (dashed horizontal line) for the Lorenz-63 system with noise levels $\eta = 0\%$, 5% and 10%. Refer to the caption of Fig. 4.13 for a detailed description.

parameter matrix obtained using the SINDy method is sparser than the one obtained using the LARS method. As we will see, the prediction of the dynamics using the learned parameters from both methods are comparable in terms of the error metrics. The distribution of some selected significant nonzero coefficients (L_{12}, L_{33} and Q_{213})

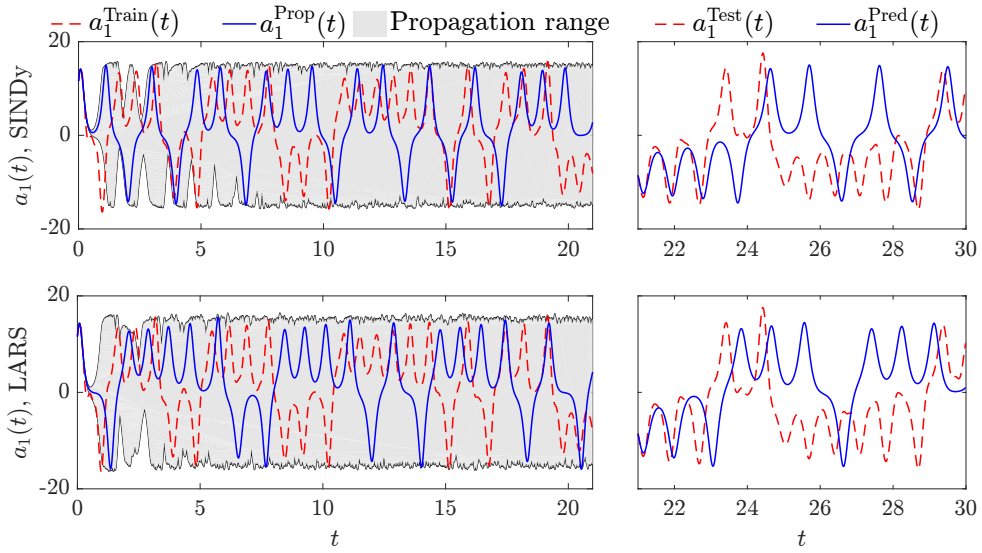


Figure 4.19: Bootstrap analysis of the time evolution of the state $a_1(t)$ for different regression methods (SINDy and LARS). Case of the Lorenz-63 system with an additive noise ($\eta = 10\%$). Refer the caption of Fig. 4.14 for a detailed description.

are visually presented with the aid of box plots in Fig. 4.18. The parameter estimates corresponding to the three noise levels are close to the true values.

The performance of the bootstrap estimates of the parameters is investigated in the training and testing windows in the same way as in Sec. 4.4.2. The propagation and prediction trajectories corresponding to the noise level $\eta = 10\%$ are shown in Fig. 4.19. As the system is chaotic, propagated trajectories start deviating from the training data at around $t = 0.5$. The range of the trajectories shows that the propagation remains bounded for the coefficients selected from within the bounds of the BCa CI. The trajectory of the dynamics obtained from the model using the mean bootstrap parameter estimates shows that the learned model is able to replicate the dynamics of the Lorenz-63 attractor. Moreover, the comparison of the predicted trajectory with the test data shows that the learned model gives a sufficiently accurate estimation of the state for a short time span outside the training window and is able to replicate the attractor dynamics.

The performance metrics in the training and testing windows for the propagated and predicted trajectories corresponding to the datasets with noise levels $\eta = 0\%, 5\%, 10\%$ are listed in Tab. 4.4. The errors E_{Prop} and E_{Pred} for the propagation and prediction are comparable for both the LARS and SINDy methods. The R^2 score corresponding to the different noise levels is approximately equal and very close to the value of 1 for the two identification methods which indicates that the predicted trajectories are able to explain the variability of the dynamics in the testing window.

The computational times associated with the application of the two system identification methods using 100 bootstrap samples are compared in Tab. 4.5 for the no noise case ($\eta = 0\%$). The estimation using the LARS method is observed to be slower than that using the SINDy method by a factor of 3 for the Lorenz-63 system.

Table 4.4: Performance of the system identification methods (SINDy and LARS) for the Lorenz-63 system with noise levels $\eta = 0\%$, 5% and 10%.

η	SINDy			LARS		
	E_{Prop}	E_{Pred}	R^2	E_{Prop}	E_{Pred}	R^2
0%	1.3248	1.0904	1.0000	1.3770	1.3797	0.9995
5%	1.3962	1.4650	0.9735	1.3614	1.4971	0.9639
10%	1.4437	1.4428	0.9502	1.3786	1.4898	0.9391

Table 4.5: Total elapsed time and number of bootstrap samples per second for the Lorenz-63 system in the training window (100 bootstrap samples and $\eta = 0\%$).

	SINDy	LARS
Elapsed time (s)	0.897	2.835
Bootstrap samples per second	110.409	34.924

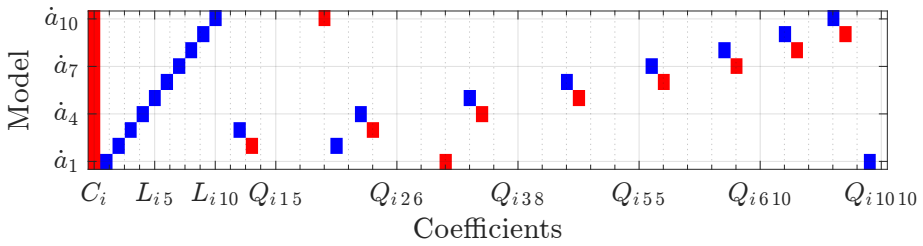
4.4.4 Toy model 3: Lorenz-96 system

In this section, the relative performance of the SINDy and LARS based system identification methods are assessed for the Lorenz-96 system in chaotic regime. The dataset for a 10 dimensional Lorenz-96 system ($N_s = 10$) is generated by integrating the system of ordinary differential equations (4.20). The hyperparameters associated with the data generation and bootstrap estimation are kept the same as in the case of the three-dimensional linear system presented in Sec. 4.4.2.

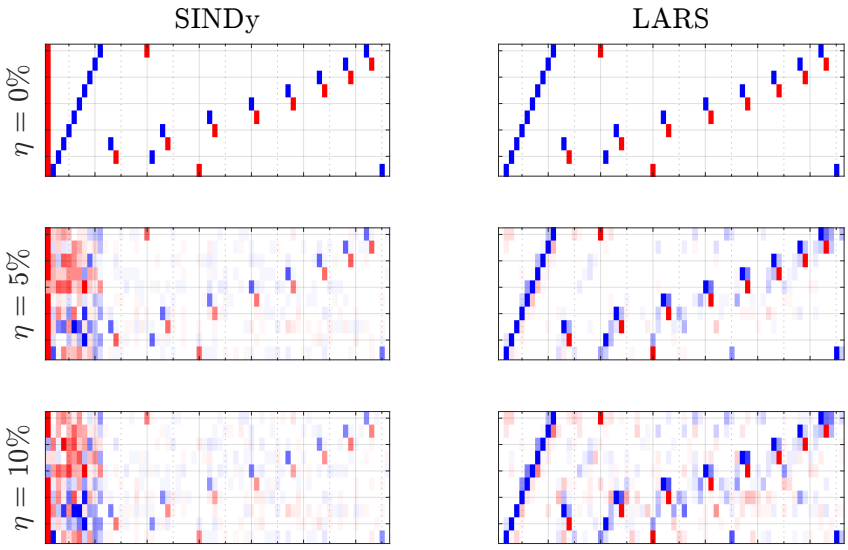
The scaled color representation of the true and learned parameter matrix $\boldsymbol{\theta}$ are shown in Fig. 4.20. For the case without noise ($\eta = 0\%$), the SINDy and LARS methods find a parameter matrix structure very similar to the true solution[16]. However, for the case with noise, both identification methods introduce nonzero values in the parameter matrix for the coefficients which are zero valued in the true parameter matrix. Contrary to the case of the Lorenz-63 system analyzed in Sec. 4.4.3, the parameter matrix obtained using the LARS method is observed to be sparser than the one obtained using the SINDy method. The additional nonzero values introduced by SINDy for the noisy data affect the sparse structure of the linear coefficients L_{ij} ($i = 1, \ldots, N_s$ and $j = i, \ldots, N_s$). On the other hand, the parameter matrix obtained for the noisy data using the LARS method retains the same structure as the true parameter matrix with the additional nonzero values mostly assigned to the quadratic coefficients Q_{ijk} ($i = 1, \ldots, N_s$, $j = i, \ldots, N_s$, and $k = j, \ldots, N_s$). The distribution of some selected significant nonzero coefficients (L_{11}, L_{99} and Q_{534}) are visually presented with the aid of box plots in Fig. 4.21. The parameter estimates corresponding to the three noise levels are close to the true values.

The performance of the bootstrap estimates of the parameters is investigated in the training and testing windows in the same way as in Sec. 4.4.2. The propagation and prediction trajectories corresponding to $\eta = 10\%$ are shown in Fig. 4.22. Similar to the Lorenz-63 system, the propagated trajectories start deviating from the training data at around $t = 0.5$. The range of the trajectories shows that the propagation remains bounded for the coefficients selected from within the bounds of the BCa CI.

[16] Note that for the LARS method, the constant terms corresponding to $C_i = F$, where F is the forcing parameter in (4.20), appear equal to zero due to the standardization before identification.



(a)



(b)

Figure 4.20: Scaled color representation of the true (a), and learned (b) parameters for the Lorenz-96 system ($N_s = 10$). In (b), we compare the results obtained by SINDy and LARS identification methods. Refer the caption of Fig. 4.11 for a detailed description.

Table 4.6: Performance of the system identification methods (SINDy and LARS) for the Lorenz-96 system ($N_s = 10$) with noise levels $\eta = 0\%$, 5% and 10%.

η	SINDy			LARS		
	E_{Prop}	E_{Pred}	R^2	E_{Prop}	E_{Pred}	R^2
0%	1.1074	0.6121	1.0000	1.2152	1.2729	0.9860
5%	1.2334	0.9813	0.9335	1.3322	1.3326	0.9636
10%	1.0902	1.0247	0.8523	1.3662	1.3489	0.9213

The trajectory of the dynamics obtained from the model using the mean bootstrap parameter estimates shows that the learned model is able to replicate the true reference dynamics. Moreover, the comparison of the predicted trajectory with the test data shows that the learned model gives a sufficiently accurate estimation of the state for a short time span outside the training window. In this example, the estimation obtained from the LARS method replicates the attractor dynamics for a longer time span than that obtained from the SINDy method.

The performance metrics in the training and testing windows for the propagated and predicted trajectories are listed in Tab. 4.6 for the noise levels $\eta = 0\%$, 5% and

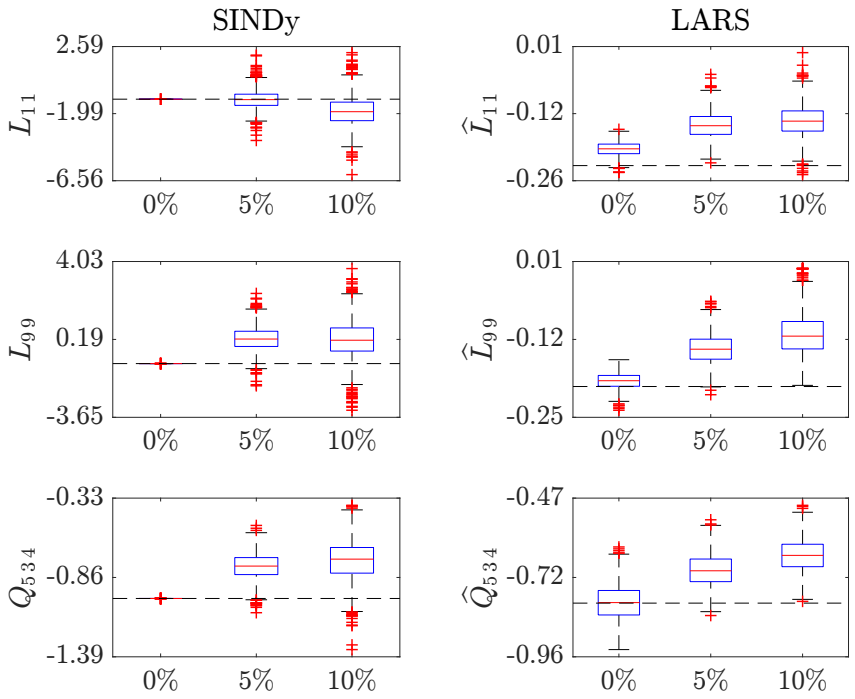


Figure 4.21: Box plots of the learned coefficients by SINDy and LARS along with the true value (dashed horizontal line) for the Lorenz-96 ($N_s = 10$) system with noise levels $\eta = 0\%$, 5% and 10%. Refer the caption of Fig. 4.13 for a detailed description.

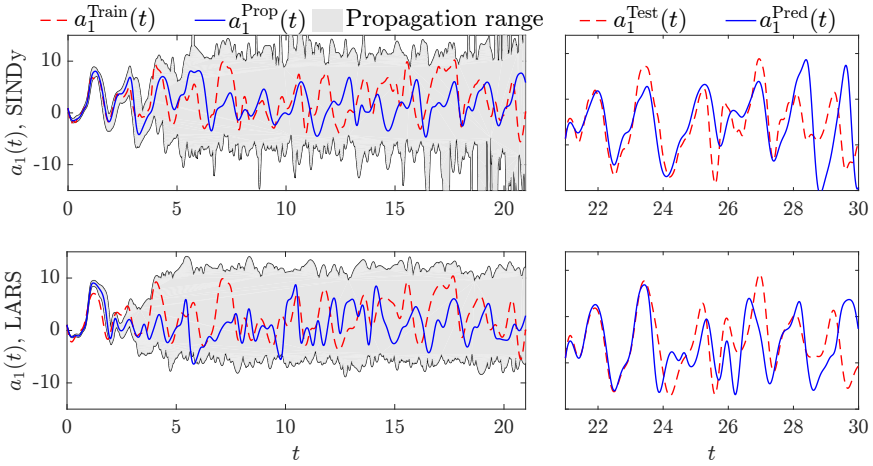


Figure 4.22: Bootstrap analysis of the time evolution of the state $a_1(t)$ for different regression methods (SINDy and LARS). Case of the Lorenz-96 system ($N_s = 10$) with an additive noise ($\eta = 10\%$). Refer the caption of Fig. 4.14 for a detailed description.

10%. The errors E_{Prop} and E_{Pred} for the propagation and prediction are comparable for both the LARS and SINDy methods. The error values associated with LARS are slightly higher than those associated with SINDy. The R^2 score corresponding to the

Table 4.7: Total elapsed time and number of bootstrap samples per second for the Lorenz-96 system in the training window ($N_s = 10$, 100 bootstrap samples and $\eta = 0\%$).

	SINDy	LARS
Elapsed time (s)	87.326	27.983
Bootstrap samples per second	1.134	3.538

different noise levels is very close to the value of 1 for the two identification methods. For noisy data, the R^2 score corresponding to the LARS method is higher than that corresponding to the SINDy method. This result indicates that the predicted trajectory using the estimated parameters from the LARS method is able to better explain the variability of the dynamics in the testing window.

The computational times associated with the two identification methods are compared in Tab. 4.7 for 100 bootstrap samples and $\eta = 0\%$. Unlike the Lorenz-63 system, the estimation of the Lorenz-96 system using the LARS method is observed to be faster by a factor of 3 than that using the SINDy method. This suggests that for a high-dimensional system, the LARS method is computationally more efficient than the SINDy method.

4.5 Conclusion

The uncertainties of three system identification methods (namely OLS, SINDy and LARS) have been quantified in a probabilistic framework provided by the bootstrap method. Toy models – that mimic POD-ROM without the nonlinear residual term – have been considered for the evaluation of the identification methods. In particular, the performance of the three identification methods in handling noisy and imperfect data to recover the model parameters vector has been examined. Preprocessing steps consisting of a low-pass filtering and a total-variation numerical differentiation method have also been introduced.

The use of the circular block bootstrap resampling technique allows to estimate for the three toy models the distribution and the confidence intervals associated with the learned coefficients of interest. The OLS identification method is found to be comparatively accurate but computationally expensive. The SINDy and LARS identification methods exhibit a comparable performance in terms of the error metrics when applied to problems with chaotic dynamics. However, the SINDy method has been observed to be computationally more efficient than the LARS method for a low-dimensional identification problem, while the LARS method is more efficient for a high-dimensional problem. We also note that the LARS algorithm has an advantage over the SINDy algorithm in terms of the tuning of the regularization parameter. Indeed, unlike SINDy, the LARS algorithm uses the information criterion to select the regularization parameter that optimally balances parsimony and accuracy.

Following the results obtained in this chapter, the linear regression method will be used in the subsequent chapter to initialize the POD-ROMs corresponding to different test cases.

Chapter 5

Improving POD-ROM using the Dual Ensemble Kalman filter

Contents

The identification of POD-ROMs by linear regression models (see Chap. 4) leads in general to a dynamical system with imperfect long-time predictability. As already discussed,

one of the main reasons is the truncation error introduced by neglecting the high-order modes in the POD expansion. Another reason is the imperfect identification of the model parameters due to numerical errors inherent to the chosen identification method or due to limited available data. Typically, the temporal coefficients can experience an amplitude growth and/or loss of the phase information due to the incomplete or improper description of the energy dissipation mechanism in the model, resulting in large error in the estimated dynamics. At this stage, the quadratic structure of the POD-ROM may lead to numerical instabilities of the model causing the dynamics to diverge in finite time. In this chapter, as a possible cure to the model prediction problem, the residual term of the POD-ROM associated with the high-order modes is first modeled, thanks to a linear eddy viscosity term parameterized with a vector $\boldsymbol{\theta}_s$. Secondly, the Dual Ensemble Kalman filter (hereafter Dual-EnKF), introduced and detailed in Sec. 3.2.3, is used to simultaneously correct the estimated state and predict the vector $\boldsymbol{\theta}_s$ thanks to assimilation of observations into the model. As a demonstration, three test cases are considered. In Sec. 5.1, the Lorenz-63 system is first considered as a toy model to analyze the performance of the Dual-EnKF method to recover both the state and the coefficients of the model equation. We investigate the influence of different parameters, such as the noise levels or the dimension of the observation data set. In Sec. 5.2, the case of a simulated flow around a circular cylinder at a low Reynolds number is considered. A POD-ROM with a linear eddy viscosity residual term given by (2.83) is first built. The Dual-EnKF algorithm is then applied to estimate the values of the eddy viscosity directly from the data. In Sec. 5.3 and Sec. 5.4, the same methodology is applied to an experimental cylinder wake flow at two different Reynolds numbers. Finally, the case of a Mach 0.9 turbulent jet is considered in Sec. 5.5. In this last example, which follows the work of Kerhervé et al. (2012), a twenty degree of freedom POD-ROM is constructed to predict, over a long-term horizon, the dynamics of the acoustically-important flow motions. The main outcomes of the chapter are finally drawn in Sec. 5.6.

5.1 Validation of Dual-EnKF method for the Lorenz-63 system

The Lorenz-63 oscillator introduced in Sec. 4.3 is considered again as a toy model to investigate this time the performance of the Dual-EnKF to simultaneously recover the state of the system and the model coefficients. The results presented in this section are largely inspired by the work of Bourgois et al. (2011).

5.1.1 Reference and observation data setup

We rewrite the system of ordinary differential equations (4.19) defining the Lorenz-63 system as

$$\begin{cases} \dot{a}_1(t) = \sigma(a_2(t) - a_1(t)), \\ \dot{a}_2(t) = a_1(t)(\rho - a_3(t)) - a_2(t), \\ \dot{a}_3(t) = a_1(t)a_2(t) - \beta a_3(t), \end{cases} \tag{5.1}$$

with the state vector $\boldsymbol{a}(t) = [a_1(t)\ a_2(t)\ a_3(t)]^\top$ of length $N_s = 3$ and the true parameter vector $\boldsymbol{\theta}^t = [\sigma\ \rho\ \beta]^\top = [10\ 28\ 8/3]^\top$ of length $N_b = 3$. As a true solution, (5.1) is

solved using the time marching scheme LSODA with initial condition $\boldsymbol{a}_0 = [1 \ -1 \ 25]^\top$. This results in a state evolving at the discrete time steps t_k denoted $a_{i,k} = a_i(t_k)$, where $i = 1, \ldots, N_s$ and $k = 1, \ldots, N_t$, with a regular step size of $\Delta t = 0.002$. To simulate the presence of model errors at the time instants t_k, a time-dependent Gaussian distributed noise $\boldsymbol{\eta}_k$ is added to the true solution, leading to, hereafter, the "reference" solution $\boldsymbol{a}^{\text{ref}}$:

$$a_{i,k}^{\text{Ref}} \leftarrow a_{i,k} + \eta_{i,k}, \quad \forall\, i = 1, \ldots, N_s. \tag{5.2}$$

Here $\boldsymbol{\eta}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}_k) \in \mathbb{R}^{N_s \times 1}$ is Gaussian distributed with zero mean and a covariance matrix $\boldsymbol{Q}_k \in \mathbb{R}^{N_s \times N_s}$.

Next, we assume a series of observations, denoted as $a_{i,k}^{\text{o}}$, obtained from the true solution $a_{i,k}$ perturbed by a Gaussian distributed noise $\boldsymbol{\epsilon}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}_k)$. The noise $\boldsymbol{\epsilon}_k \in N_o \times 1$ simulates the measurement errors characterized by the corresponding covariance matrix $\boldsymbol{R}_k \in \mathbb{R}^{N_o \times N_o}$, where N_o is the number of components in the observation vector $\boldsymbol{a}_k^{\text{o}}$. In the following, full $(N_o = N_s)$ and partial $(N_o < N_s)$ observation will be considered. A total number of $N_{t,o} = 100$ observations are sampled at regularly spaced time intervals with a step size of $\Delta t_o = 100\Delta t$.

5.1.2 Objective of the assimilation and initial guess

In the following, the performance of the Dual-EnKF algorithm to simultaneously predict the state trajectory and to recover the model parameter vector θ^t when starting with an incorrect initial guess is investigated. In Sec. 5.1.3, the influence of different levels of model and observation covariance matrices ($\boldsymbol{Q}_k$ and $\boldsymbol{R}_k$) on the estimation is discussed. The values of the elements forming the two covariance matrices can be seen as the degree of confidence associated with the model and observations which shows up in the weights associated with the correction step of the states and parameters in the Dual-EnKF algorithm. Next, the influence of different ensemble sizes is discussed in Sec. 5.1.4. Note that the time evolution of the parameters is not presented in these two sections as the focus is on the influence of hyperparameters on the quality of prediction of the Dual-EnKF algorithm. Lastly, the influence of partial observation in the state space is discussed in Sec. 5.1.5.

As discussed in Sec. 3.2.3, the Dual-EnKF propagates, in time, an ensemble of state and parameter vectors from which the estimated state and parameter vector at each time steps are obtained via means of the ensembles. At the start of the assimilation process, ensembles of N_e members are thus initiated for the state and parameter vectors. The initial ensemble state $\{\boldsymbol{a}_0^{a,(n)}\}_{n=1,\cdots,N_e}$ is randomly selected from a normal distribution $\mathcal{N}(\boldsymbol{a}_0, \boldsymbol{Q}_0 = \boldsymbol{I}_3)$ while the initial ensemble parameter vector $\{\boldsymbol{\theta}_0^{a,(n)}\}_{n=1,\cdots,N_e}$ is randomly selected from a normal distribution $\mathcal{N}(\boldsymbol{\theta}_0, \boldsymbol{C}_0 = \boldsymbol{I}_3)$. During the assimilation process, the parameter covariance matrix $\boldsymbol{C}_k$ is maintained equal to $0.001\boldsymbol{I}_3$.

5.1.3 Influence of model and observation errors

As a first instance, we consider all the components of the state vector $\boldsymbol{a}_k$ as observables such that the size of the observation vector is $N_o = N_s = 3$. To study the influence of varying the covariance error levels on the estimation, we choose four pairs of model and observation covariance matrices as listed in Tab. 5.1; each matrix pair being identified

Table 5.1: Test cases with varying model and observation errors. $N_e = 100$.

	$\boldsymbol{Q}_k$	$\boldsymbol{R}_k$
A-I	$0.01\boldsymbol{I}_3$	$0.01\boldsymbol{I}_3$
A-II	$\boldsymbol{I}_3$	$0.01\boldsymbol{I}_3$
A-III	$0.01\boldsymbol{I}_3$	$\boldsymbol{I}_3$
A-IV	$\boldsymbol{I}_3$	$\boldsymbol{I}_3$

Table 5.2: Tests cases with varying ensemble sizes and $\boldsymbol{Q} = 0.01\boldsymbol{I}_3$, $\boldsymbol{R} = 0.01\boldsymbol{I}_3$.

	N_e
B-I	10
B-II	50
B-III	100
B-IV	200

with a label from A-I to A-IV. The ensemble size is fixed as $N_e = 100$ which is sufficiently high to ensure that enough members are available in the ensemble set to represent the distribution of the unknown variable (state or parameter) during the assimilation period. This relatively favorable data assimilation setup allows to gauge the impact of the model and observation error levels on the quality of the estimation. As $N_{t,o} = 100$ observations are available, the Dual-EnKF assimilation involves $N_{\text{Iter}} = 100$ iterations in which both the state estimation and the model parameters are predicted and updated sequentially as described by Alg. 3.2.

As a metric of the estimation quality, we introduce the time-dependent mean square error (hereafter MSE) over the assimilation period defined with respect to the reference solution as,

$$MSE(t_k) = \frac{1}{N_s}\frac{1}{N_e}\sum_{i=1}^{N_s}\sum_{n=1}^{N_e}(a_{i,k}^{a,(n)} - a_{i,k}^{\text{Ref}})^2, \tag{5.3}$$

where $a_{i,k}^{a,(n)}$ is the i-th element of the n-th analyzed state vector $\boldsymbol{a}_k^{a,(n)}$ at the time instant t_k.

The time evolution of the MSE over the assimilation window for the different test cases considered in Tab. 5.1 is reported in Fig. 5.1 (left).

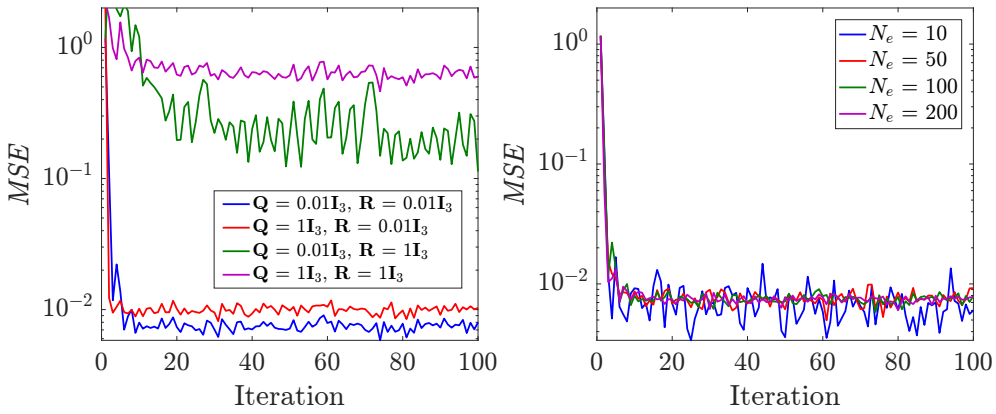


Figure 5.1: Influence on the MSE of varying model and observation noises with fixed ensemble size $N_e = 100$ (left) and varying ensemble size with constant $\boldsymbol{Q} = 0.01\boldsymbol{I}_3$ and $\boldsymbol{R} = 0.01\boldsymbol{I}_3$ (right).

In the test case A-I, the low level error covariance matrices $\boldsymbol{Q}_k = 0.01\boldsymbol{I}_3$ and $\boldsymbol{R}_k = 0.01\boldsymbol{I}_3$ correspond to a situation with good confidence in both the dynamical model and

the observations. As a result, the Dual-EnKF algorithm is able to update the state to match the reference trajectory within about 5 iterations, as can be seen from the error metric value dropping and stabilizing at $MSE \approx 7.5 \times 10^{-3}$.

In the test case A-II, the covariance matrices $\boldsymbol{Q}_k = \boldsymbol{I}_3$ and $\boldsymbol{R}_k = 0.01\boldsymbol{I}_3$ correspond to a case with a lower confidence in the dynamical model. For these covariance matrices, the ratio of the error levels of the state and the observations is equal to 100, which signifies a large Kalman gain, and therefore a larger weight associated with the correction of the estimates. As a consequence, the error metric drops to a low stable level of $MSE \approx 1.0 \times 10^{-2}$ from the first iteration onwards. As the model error level here is higher than in the previous test case, the members of the state ensemble resulting from the covariance matrix span a larger domain in the state space, which improves the estimation. The estimation error however increases by an order of 10 as compared to the test case A-I due to the low confidence in the model estimates.

In the test case A-III, the covariance matrices $\boldsymbol{Q}_k = 0.01\boldsymbol{I}_3$ and $\boldsymbol{R}_k = \boldsymbol{I}_3$ correspond to a case with low confidence in the observations. In this case, the ratio of the state and observation error levels is 0.01, which leads to a low magnitude of the Kalman gain. Therefore, the observations are less reliable and weakly modify the current estimate. Also, while not reported here, the span of the state ensemble members is very limited due to the low level of model noise. As a result, the error metric stabilizes slowly after approximately 20 iterations to a relatively high value of $MSE \approx 2.0 \times 10^{-1}$.

Finally, in the test case A-IV, the high level covariance matrices $\boldsymbol{Q}_k = \boldsymbol{I}_3$ and $\boldsymbol{R}_k = \boldsymbol{I}_3$ correspond to a case with low confidence in the dynamical model and the observations. This results in a higher value of the estimation error after convergence of $MSE \approx 6.3 \times 10^{-1}$ as compared to the previous tests. However, as the ensemble members are spread over a larger span, owing to the higher covariance levels, the performance in terms of the rate of convergence remains comparable, with the error metric dropping to a stable value in around 10 iterations.

As preliminary conclusion, the model and observation error covariance matrices are found to influence the predictability and accuracy of the estimation. Care must therefore be taken while setting up these matrices in order to accurately represent the model and observation errors and to seek remedies if the Dual-EnKF estimation diverges.

5.1.4 Influence of ensemble size

The influence of the ensemble size of the state and parameter forecasts on the Dual-EnKF estimation is now investigated. In this regard, we consider four different values of ensemble sizes ranging from 10 to 200 as listed in Tab. 5.2 and labeled from B-I to B-IV.

The influence of the size of the ensemble on the quality of the estimation can be observed from the evolution of the error metric MSE in Fig. 5.1 (right). In each test case, the covariance matrices associated with the model and the observations have been assigned to $\boldsymbol{Q}_k = 0.01\boldsymbol{I}_3$ and $\boldsymbol{R}_k = 0.01\boldsymbol{I}_3$ (test case A-I) which was found in Sec. 5.1.3 to correspond to the minimum estimation error across the test cases considered.

For the test case with $N_e = 10$, we observe that even though the MSE value drops to a low value as soon as the assimilation starts, it shows relatively high fluctuations instead to converging to a stable value. This effect can be attributed to the low number

of members in the forecast ensemble which leads to a weak exploration of the state space. For the test cases with $N_e = 50$, 100 and 200, we observe from the mean MSE values that the performances are similar with respect to the quality of estimation but the fluctuations are comparatively negligible. The Dual-EnKF filter is able to rapidly converge to a low error level of the order of 1.0×10^{-3} when the ensemble size is sufficiently large.

The ensemble size has been shown to have some significant impact on the accuracy of the Dual-EnKF estimation. It should be noted that the selection of the ensemble size is generally dependent on the problem. When too small, it may lead to a false representation of the domain of forecast variables with respect to the true values. In contrast, when too large, it may render the iterations computationally intractable.

5.1.5 Influence of partial observation

We now investigate the influence of partial observations on the Dual-EnKF estimation, *i.e.* the scenarios where, instead of assuming that the full state vector $\boldsymbol{a}^{\mathrm{o}}(t)$ is known at each time instant, only a subset of the component states are observable and available for assimilation. As an example, we consider the case where only the state $a_1(t)$ is observed.

The estimation problem is set up with the model state and observation covariances assigned as $\boldsymbol{Q}_k = 0.01\boldsymbol{I}_3$ and $\boldsymbol{R}_k = 0.01\boldsymbol{I}_3$, and the ensemble size set to $N_e = 100$. To begin the estimation with an initial guess which is well separated from the true value, the state and parameter covariance matrices at t_0 are assigned as $\boldsymbol{Q}_0 = 100\boldsymbol{I}_3$ and $\boldsymbol{C}_0 = 100\boldsymbol{I}_3$. The evolution of the Lorenz-63 system corresponding to the cases with full and partial observations is shown in Fig. 5.2a, and the respective evolution of the model parameters $\boldsymbol{\theta}_k$ is shown in Fig. 5.2b.

From the state evolution, we observe that when all the components of the state vector are observed and used for the Dual-EnKF estimation, the analyzed state immediately follows the reference trajectory $\boldsymbol{a}^{\mathrm{Ref}}(t)$. This accurate estimation is attributed to the fast convergence of the parameter to the true values $\boldsymbol{\theta}^t$, within approximately 10 iterations as shown in Fig. 5.2b. When partial observation is considered, the true state trajectory is still well recovered, even if more iterations are needed to make the parameter vector converging towards the true values.

It is therefore found that when the parameters are constant in time and the number of iterations N_{Iter} sufficiently large, the Dual-EnKF assimilation method estimates the complete model space with partial observations.

5.1.6 Estimation of slow time-varying parameter

The case studied so far correspond to the classical Lorenz-63 system where the model parameters vector $\boldsymbol{\theta}$ is constant with time. In many practical situations, the system state may however be better represented by a time-varying parameters model. Such situation is familiar, for instance, in hydrological processes where climate change, afforestation, urbanization or again seasonal variations can lead to time-variant hydrological model parameters (Deng et al., 2016). Closer to the present context, one can imagine the case where the Reynolds number of the flow is slowly varying in time and that we would like to "track" the changes in this number thanks to limited flow data. While such a problem

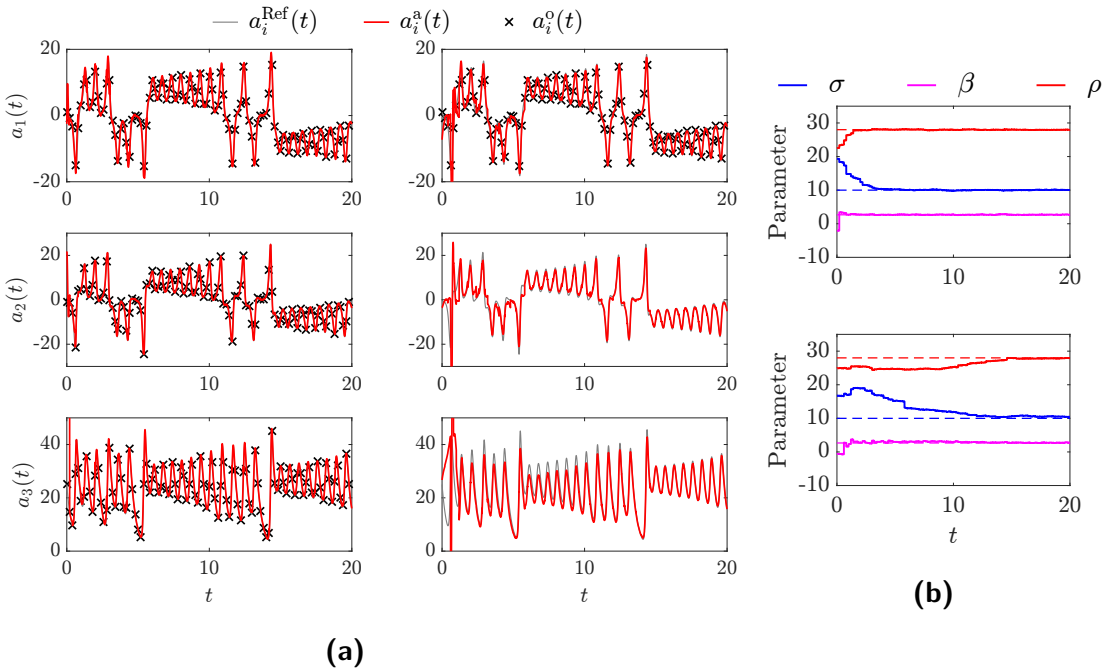


Figure 5.2: (a) Influence of the partial observation on the state estimation. Comparison is made between the scenarios where all three state variables are observed (left), and where only the component $a_1(t)$ is observed (right). (b) Influence of the partial observation on the estimated parameter values. The true values of parameters $\boldsymbol{\theta}^t = [10\ 28\ 8/3]^\top$ are shown by dashed lines.

is out of the scope of the present work, it can also be demonstrated that the Dual-EnKF performs well in the case of slowly-time-varying parameter models.

As demonstration, a new true solution of the Lorenz-63 system (5.1) is generated assuming that the second coefficient of the parameter vector $\boldsymbol{\theta}$ is slowly varying in time according to,

$$\rho(t) = 2\sin(2\pi t/T),\ \forall\, t \in [0, T] \quad \text{with } T = 20. \tag{5.4}$$

The estimation problem (full observation) is set up with the model state and observation covariances assigned as $\boldsymbol{Q}_k = 0.01\boldsymbol{I}_3$ and $\boldsymbol{R}_k = 0.01\boldsymbol{I}_3$, and the ensemble size set to $N_e = 100$. For initialization, the state and parameter covariance matrices at t_0 are assigned as $\boldsymbol{Q}_0 = 100\boldsymbol{I}_3$ and $\boldsymbol{C}_0 = 100\boldsymbol{I}_3$. The evolution of the state $a_2(t)$ is shown in Fig. 5.3a, and the respective evolution of the model parameters $\boldsymbol{\theta}_k$ is shown in Fig. 5.3b.

From the state evolution, it can be seen that the assimilation method is able to provide accurate estimations with respect to the reference trajectory. This is attributed to the accurate estimation of the time varying parameter $\rho(t)$. In the applications where the transient dependence of the parameters is not known *a priori*, the Dual-EnKF estimation is able to predict the slow time varying components.

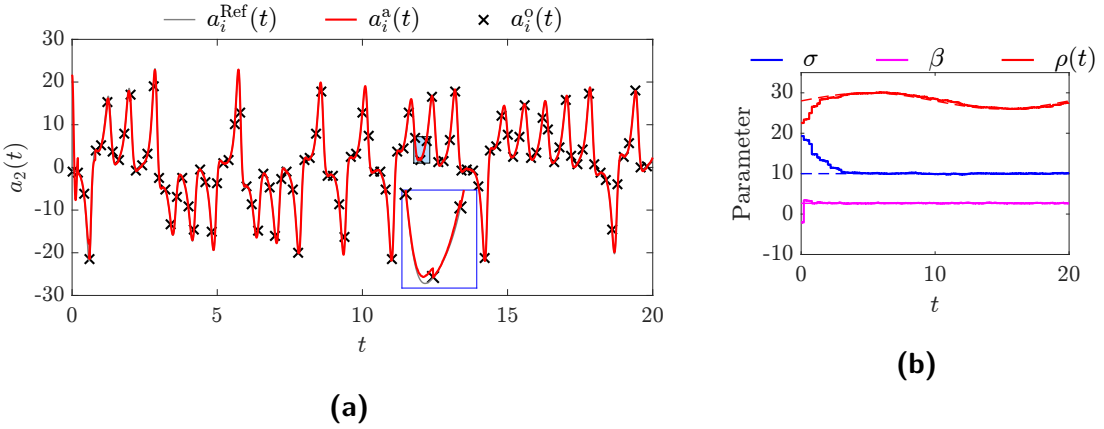


Figure 5.3: (a) Trajectory of the state $a_2(t)$ with time varying parameter. The maximized view in the inset shows the ability of the assimilation method to correct the analyzed state $a_2^a(t)$ at the time steps where the observation $a_2^o(t)$ is available. (b) Estimated parameter values with constant σ and β, and slowly time varying $\rho(t)$. The true values of parameters $\boldsymbol{\theta}^t(t)$ are denoted by dashed lines.

5.2 Test case 1: Numerical cylinder wake flow at $Re_D = 200$

The effects of the Dual-EnKF tuning hyperparameters on the state and parameter identification were discussed for a simple toy model in Sec. 5.1. Proceeding further, the case of a simulated cylinder wake flow at $Re = 200$ is considered. The objective is to demonstrate the ability of the DA method to identify the closure parameters $\boldsymbol{\theta}_s$ characterizing the residual term $\mathcal{R}_i(\boldsymbol{a}; \boldsymbol{\theta}_s, t)$, added in Sec. 2.2.4.1 to model in the POD-ROM the interactions between the resolved and unresolved modes. Here, the flow dynamics is essentially driven by quasi-periodic cycles of vortex shedding, but the number of modes used to construct the POD-ROM is such that the model can be considered high-dimensional compared to the Lorenz-63 system examined in the previous section. As a first step, a POD-ROM in the form of (2.79) is identified using the sparse-identification method SINDy detailed in Sec. 3.1.2. As discussed below, due to the "simple" dynamics of the flow considered, the identified model gives already satisfying results in terms of prediction of the states over few cycles of vortex shedding. For the purpose of the current study, the identified model is therefore deteriorated in a second step such that it can be recovered exactly by adding a residual term in the form discussed in Sec. 2.2.4.1. In the third and final step, the Dual-EnKF is applied to identify the closure parameters introduced to model the residual term and demonstrate some robustness with regards to observation noise.

5.2.1 Identification of the POD-ROM

The numerical set-up, except for the Reynolds number, is similar to the one discussed in App. A used to simulate a 2D cylinder wake flow at $Re = 100$. For the current purpose, 1000 snapshots (including streamwise and transverse velocity components) are obtained with a time step of $\Delta t = 0.1$ and used to build the POD basis. The energetic content,

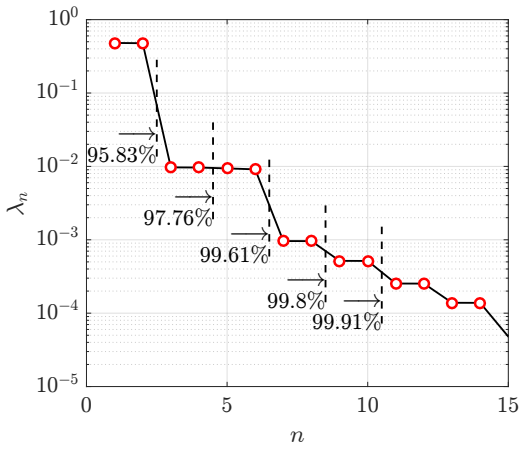


Figure 5.4: POD energy content of the most energetic modes for the 2D-cylinder wake flow at $Re = 200$. Eigenvalue spectrum of the POD modes (λ_n). The values of $RIC(n)$ are also indicated, representing the fraction of the total fluctuating kinetic energy captured if only the POD modes up to the dashed lines are considered.

POD eigenvalues and relative information content, are reported in Fig. 5.4 as a function of the mode number. As expected for a cylinder wake flow at low Reynolds numbers, the eigenvalues appear in pairs. This characteristic is attributed to the convection of the coherent structures which cannot be represented as a single mode (Rempfer and Fasel, 1994). The first ten modes, which together represent 99.91% of the overall energy, are retained to build a reduced-order-model of the general form given by (2.79). The number of modes considered in the following study is therefore fixed as $N_{\mathrm{Gal}} = 10$.

The structure of the spatial modes $\mathbf{\Phi}_i^{x,\mathrm{POD}}(\boldsymbol{\chi})$ $(i = 1, 5, 9)$ corresponding to the streamwise velocity fluctuations and the time evolution of the pairs of temporal POD coefficients a_i^{POD} $(i = 1, 2,\ i = 5, 6$ and $i = 9, 10)$ are shown in Fig. 5.5. The mode $\mathbf{\Phi}_1^{x,\mathrm{POD}}$ represents the coherent vortex-shedding structures and pairs with the mode $\mathbf{\Phi}_2^{x,\mathrm{POD}}$ (not shown) with similar structures but shifted in the streamwise advection direction. The corresponding POD coefficients a_1 and a_2 are analogously shifted in time. These two modes together are representative of the Kármán vortex-shedding mode and contribute to 95.83% of the total energy. The eight subsequent modes correspond to higher-order harmonic modes which together represent 4.08% of the total energy. These modes can be associated with smaller scale structures attributed to the manifestation of the separated shear layers along the sides of the cylinder and their longitudinal expansion further downstream. From the evolution of the temporal coefficients, it can be observed that the amplitude of the dynamics drops from the mode pairs (a_1, a_2) to (a_5, a_6) to (a_9, a_{10}) which is consistent with the energy spectrum of Fig. 5.4. From the energy spectrum, it can also be seen that the modes with indices $i = 3, 4, 5, 6$ have contributions to the total energy in the same order of magnitude and, as such, exhibit mixed harmonics, as observed in the temporal evolution of coefficients a_5 and a_6 in Fig. 5.5d. The first 10 modes together therefore exhibit a range of different features associated with the vortex shedding, such as the large-amplitude low-frequency scales in the pair of the most dominant modes or again mixed harmonics in the subsequent pairs of modes.

A reduced-order model of the form given by (2.79) is considered. It is recalled here

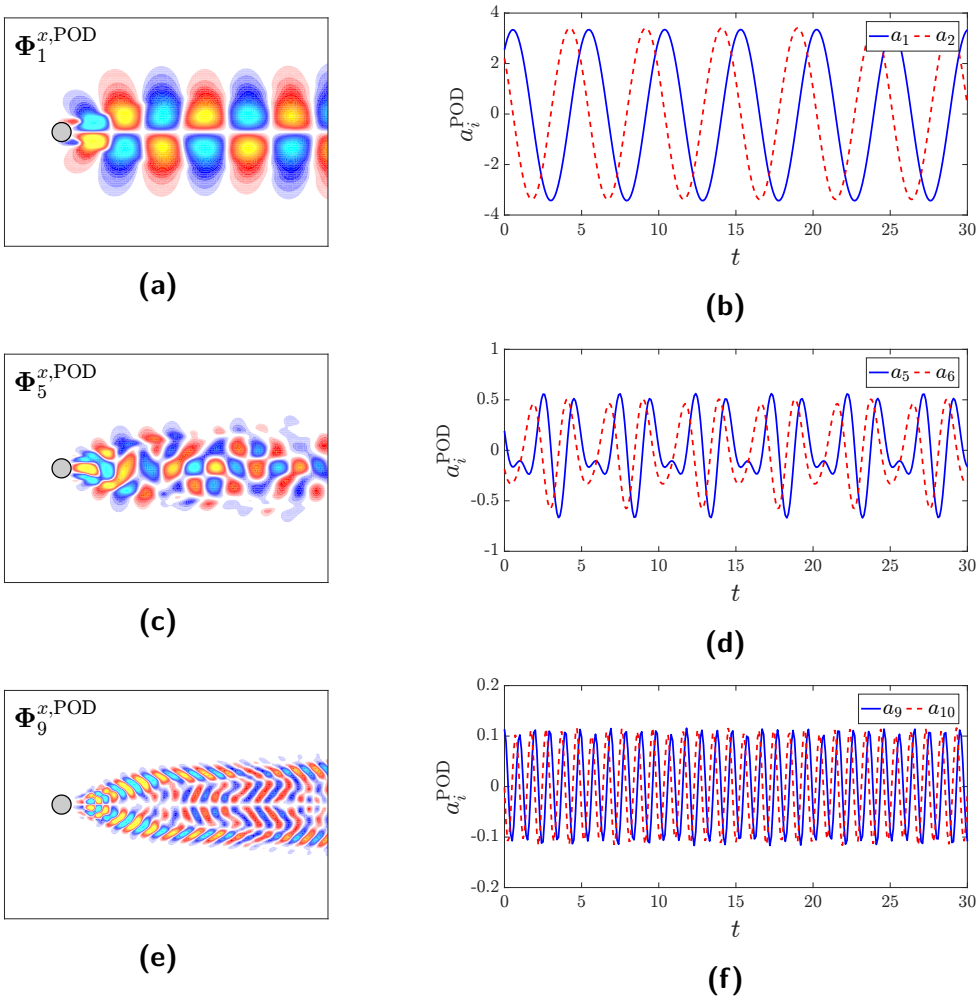


Figure 5.5: (a,c,e) Spatial POD modes $\mathbf{\Phi}_i^{x,\mathrm{POD}}$ $(i = 1, 5, 9)$ corresponding to the streamwise velocity fluctuations. (b,d,f) Time evolution of the temporal POD coefficients a_i^{POD} $(i = 1, 2, 5, 6, 9, 10)$ corresponding to the most energetic modes of the 2D-cylinder wake flow at $Re = 200$.

for convenience,

$$\begin{aligned}\dot{a}_i^{\mathrm{ROM}}(t) &= f_i(\boldsymbol{a}^{\mathrm{ROM}}; \boldsymbol{\theta}_i, t) \\ &= C_i + \sum_{j=1}^{N_{\mathrm{Gal}}} L_{ij} a_j^{\mathrm{ROM}}(t) + \sum_{j=1}^{N_{\mathrm{Gal}}} \sum_{k=j}^{N_{\mathrm{Gal}}} Q_{ijk} a_j^{\mathrm{ROM}}(t) a_k^{\mathrm{ROM}}(t). \end{aligned} \tag{5.5}$$

The identification is done using only the first 105 snapshots of the data collection (training dataset). The rest is used for testing the learned model. The parameters vectors θ_i are identified using SINDy with $\lambda = 1.0 \times 10^{-3}$ as sparsification parameter. The POD-ROM thus identified is considered as a basis for further alteration (see Sec. 5.2.2).

Using the same representation as in Sec. 4.4, the identified parameters are shown in Fig. 5.6(a). As expected and already discussed in the case of the cylinder wake flow at $Re = 100$, the SINDy identification method provides a sparse solution with

regards to the model coefficients. Close examination of this result and that of Fig. 5.5 shows that the temporal coefficients $a_i(t)$ $(i = 1, 2, 7, 8, 9, 10)$ which exhibit quasi-single harmonics, contribute only linearly to the ROM. On the other hand, the temporal coefficients a_i $(i = 3, 4, 5, 6)$ which exhibit mixed harmonics, have contribution from linear and quadratic terms as well.

Time evolution of the true temporal POD coefficients $a_i^{\text{Train}}(t) = a_i^{\text{POD}}(t)$ $(i = 1, 5, 9)$ in the training window, and that obtained from the integration of the identified POD-ROM with $a_i^{\text{Prop}}(0) = a_i^{\text{POD}}(0)$ as initial condition, can be seen in Fig. 5.6(b-d)(left). The results show that the identified model is able to replicate the original dynamics over a complete cycle of vortex shedding after which some deviations are manifest. When testing the identified POD-ROM outside the training window, as shown in Fig. 5.6(b-d)(right), similar conclusions can be drawn. While the identified POD-ROM is not perfect, it can be used to predict, with sufficient accuracy, the evolution of the state in a short time window corresponding, at least, to a complete cycle of vortex shedding.

5.2.2 Deterioration of the POD-ROM

In what follows, it is assumed that the SINDy identification gives rise to a satisfactory POD-ROM which would not require any stabilization. Therefore, to evaluate the ability of the Dual-EnKF, a new POD-ROM is built based on the parameters C_i, L_{ij} and Q_{ijk} identified previously by SINDy. We consider

$$\dot{a}_i^{\text{ROM}}(t) = C_i + \sum_{j=1}^{N_{\text{Gal}}} L'_{ij} a_j^{\text{ROM}}(t) + \sum_{j=1}^{N_{\text{Gal}}} \sum_{k=j}^{N_{\text{Gal}}} Q_{ijk} a_j^{\text{ROM}}(t) a_k^{\text{ROM}}(t), \tag{5.6}$$

where the coefficients L'_{ij}, associated with the linear terms, are now given as,

$$L'_{ij} = \frac{1}{1 + \hat{\nu_i}^T} L_{ij}. \tag{5.7}$$

The added viscosity term $\hat{\nu_i}^T$ is given by the modal constant eddy viscosity closure model of (2.84),

$$\hat{\nu_i}^T = \hat{\nu}_c \left(\frac{i}{N_{\text{Gal}}} \right)^{\hat{\alpha}}. \tag{5.8}$$

While not shown here, the new model gives rise to significant errors in the prediction and requires stabilization. One can show that adding to the model (5.6) a residual term of the form,

$$\mathcal{R}_i(\boldsymbol{a}; \boldsymbol{\theta}_s, t) = \nu_c \left(\frac{i}{N_{\text{Gal}}} \right)^{\alpha} \sum_{j=1}^{N_{\text{Gal}}} L'_{ij} a_j(t) \quad \text{and} \quad \boldsymbol{\theta}_s = [\nu_c \, \alpha]^\top \tag{5.9}$$

the original model (5.5) is exactly recovered if and only if $\boldsymbol{\theta}_s \equiv [\hat{\nu_c} \, \hat{\alpha}]^\top$. This mathematical trick allows to deteriorate arbitrarily, with a known parameter vector $\hat{\boldsymbol{\theta}}_s$, the POD-ROM identified previously.

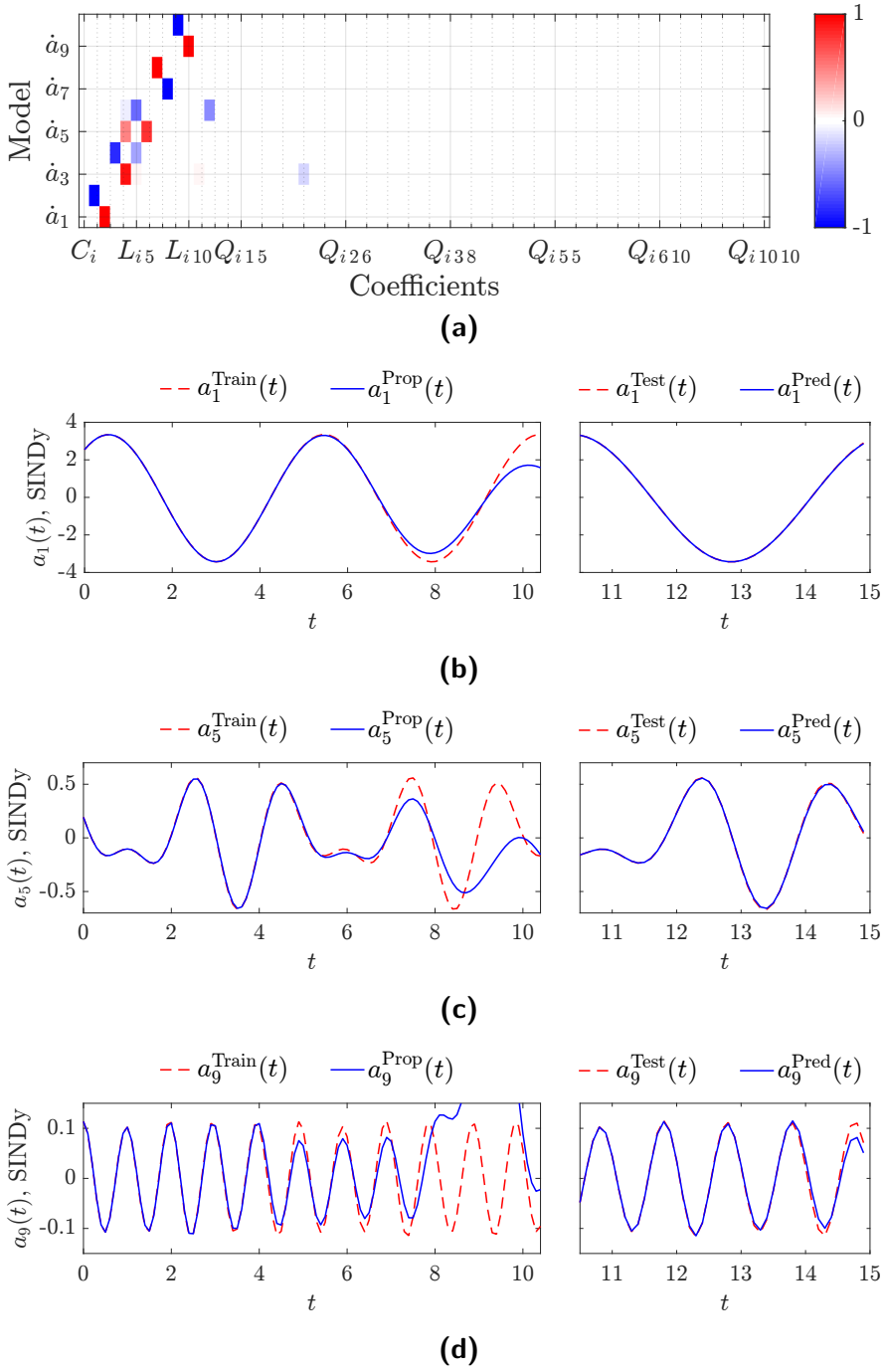


Figure 5.6: (a) Scaled color representation of the parameters identified using SINDy for the POD-ROM of a 2D-cylinder wake flow at $Re = 200$. (b,c,d) Evolution of coefficients $a_i(t)$ $(i = 1, 5, 9)$ in the training regime (left) and in the testing regime (right). True (dashed line) and estimated (solid line) trajectories.

In the following, we consider as model the POD-ROM (5.6) to which the residual term given by (5.9) is added. The parameters are arbitrarily set as $\hat{\boldsymbol{\theta}}_s = [\hat{\nu}_c \ \hat{\alpha}]^\top = [2.3 \ 0.4]^\top$. The Dual-EnKF method is then applied to recover the correct parameters vector $\hat{\boldsymbol{\theta}}_s$ from observations generated using the original snapshots perturbed by different noise levels.

5.2.3 Dual-EnKF estimation of stabilization parameters

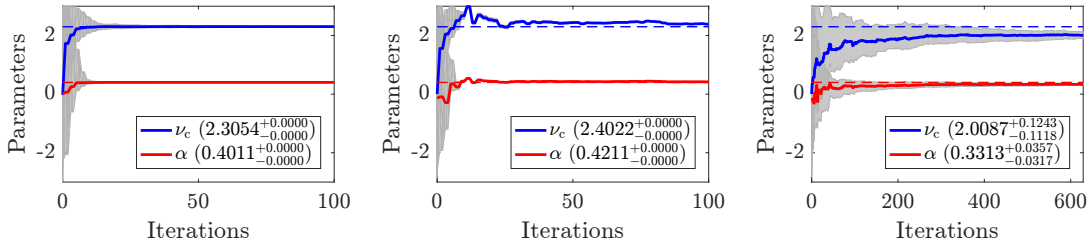
In this section, the general set-up of the Dual-EnKF data assimilation algorithm is first detailed. Then, its performance with regards to retrieving the stabilizing parameters $\hat{\boldsymbol{\theta}}_s$ is discussed.

Following the notations introduced in Sec. 3.2, the number of state elements in the data-assimilation problem is $N_s = N_{\text{Gal}} = 10$. The model error is assumed to be given by a Gaussian distribution $\boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{Q})$ with zero mean and covariance $\boldsymbol{Q} = 1.0 \times 10^{-5} \boldsymbol{I}_{N_s}$. The initial estimate of the state vector is considered as the same as the temporal POD coefficient at $t_1 = 0$, *i.e.* $\boldsymbol{a}^{\text{DA}}(0) = \boldsymbol{a}^{\text{POD}}(0)$, where the superscripts "DA" and "POD" are used to distinguish the state vectors obtained from DA and POD. As mentioned previously, the parameter vector is given as $\boldsymbol{\Theta} = \boldsymbol{\theta}_s = [\nu_c \ \alpha]^\top$ (here $N_p = 2$). Its propagation follows the random walk given by (3.71) with a Gaussian distributed additive perturbation $\boldsymbol{\xi}^{(n)} \sim \mathcal{N}(\mathbf{0}, h^2 \boldsymbol{C})$ with zero mean and covariance $\boldsymbol{C} = 1.0 \boldsymbol{I}_{N_p}$. The estimated parameter vector is initialized as $\boldsymbol{\Theta}_k = \mathbf{0}_{N_p \times 1}$. Ensembles with $N_e = 200$ members are considered. Finally, to test the robustness of the Dual-EnKF method with regards to noise, uniformly distributed noises with different noise levels (0%, 1% and 10%) are applied to the observations generated from the original data. For the observation equation, the noise is assumed to be given by a Gaussian distribution $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{R})$ with zero mean and covariance $\boldsymbol{R} = \eta \boldsymbol{I}_{N_o}$, where the covariance level η is representative of the noise level used to generate the observation, *i.e.* $\eta = 1.0 \times 10^{-6}$ (a very low value, $\eta \ll 1$) for 0% noise, $\eta = 0.01$ for 1% noise and $\eta = 0.1$ for 10% noise. Training is effected during 630 time units (corresponding to $t = 63$) during which assimilation of new observations is realized every two time steps, and after which the assimilation of new observations is stopped. In the subsequent sections, the DA procedure will consider two forms of observations: i) the temporal POD coefficients values (Sec. 5.2.3.1), and ii) point velocity "measurements" at few locations in the flow field (Sec. 5.2.3.2).

5.2.3.1 DA with temporal POD coefficients as observation

The case where the full original temporal POD coefficients vector can be observed is considered first. Referring to the observation equation (3.31), the observation vector is given as $\boldsymbol{y}_j^o = \boldsymbol{a}^{\text{POD}}(t_j) \in \mathbb{R}^{N_o \times 1}$ (here $N_o = N_s$ and $j = 1, \ldots, N_{t,o}$).

Time evolution of the estimated parameters $\boldsymbol{\theta}_s = [\nu_c \ \alpha]^\top$ during the course of DA using observations for the three different noise levels are shown in Fig. 5.7. In the three cases, the analyzed parameters during the assimilation regime tend towards the target value of $[\hat{\nu}_c \ \hat{\alpha}] = [2.3 \ 0.4]$. The variance of the parameter ensemble (represented by the gray shaded area) also reduces as the assimilation proceeds, hence resulting in a smaller confidence interval at the end of the assimilation window. For the lower noise levels (0% and 1%), the parameters at the end of the assimilation regime have values very close to

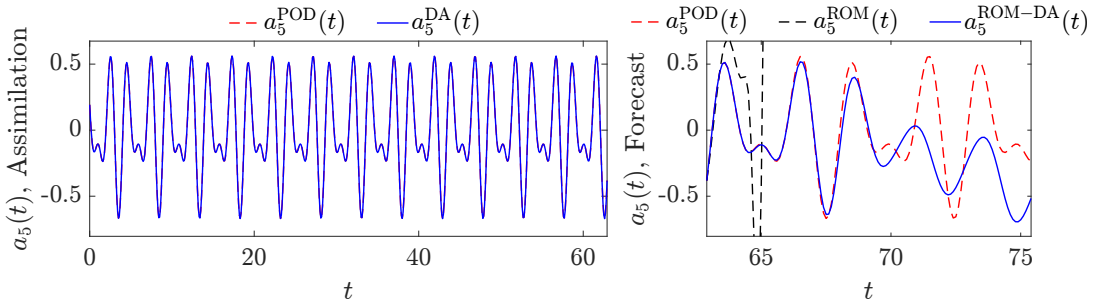


(a) 0% noise **(b)** 1% noise **(c)** 10% noise

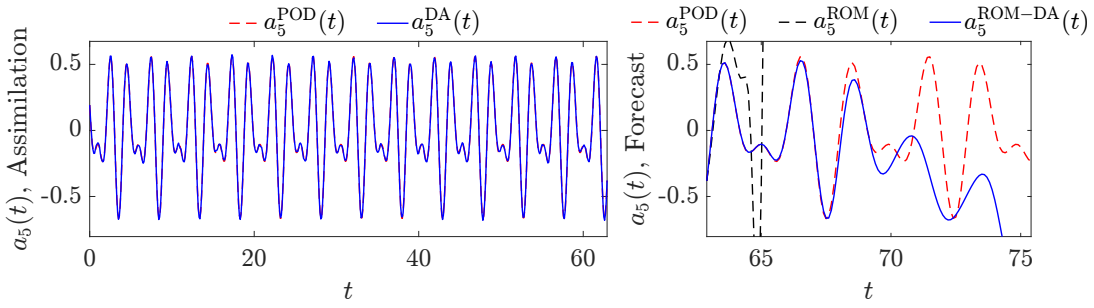
Figure 5.7: Evolution of the stabilizing parameters $\boldsymbol{\theta}_s = [\nu_c\ \alpha]^\top$ of the POD-ROM in the assimilation regime corresponding to three noise levels of observations in the form of the temporal POD coefficient. The gray shaded area shows the spread of the parameter ensemble. The dashed lines represent the artificial perturbed values $[\hat{\nu}_c\ \hat{\alpha}] = [2.3\ 0.4]$. Only the initial 100 iterations are shown in (a) and (b). The parameter values at the end of the assimilation (630 iterations) are shown in the legend along with the lower and upper values of the ensemble at that step.

the target value. For the case with 10% noise level, the difference between the estimated and target values, as well as the confidence interval are larger. The Dual-EnKF algorithm is therefore found to be able to assimilate the noisy observations and provide an estimate for the stabilizing parameters.

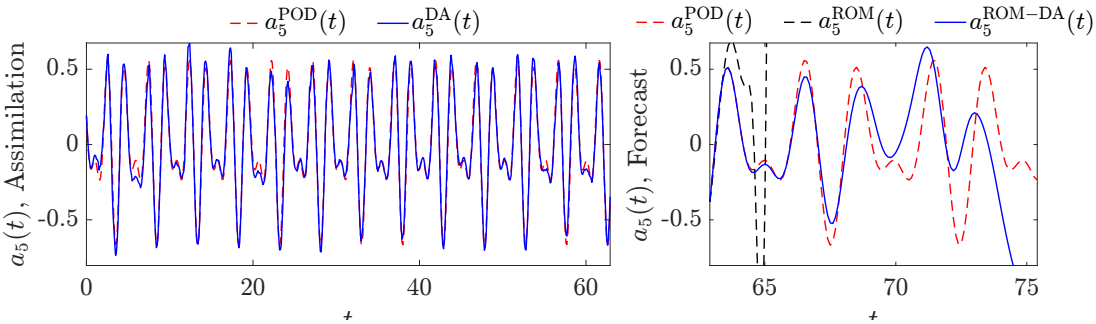
The evolution of the temporal POD mode a_5, considered here as an example as it exhibits mixed harmonics, is shown in Fig. 5.8 during the course of DA for observations with different noise levels. It is observed that the state estimation $a_i^{\mathrm{DA}}(t)$ in the assimilation regime corresponding to the lower noise levels (0% and 1%) is able to accurately replicate the reference trajectory $a_i^{\mathrm{POD}}(t)$. However, for the state estimation corresponding to 10% of noise level, the estimated value is not replicating the reference trajectory initially. As the assimilation proceeds, the accuracy of the estimation improves, thanks to the corresponding update of the parameter values as seen in Fig. 5.7c, thus correcting the forward model. The figure also shows the forecast regime where the model is integrated for $N_t^f = N_t/5 = 126$ time steps with the estimated parameters determined at the end of the assimilation regime. This long term forecast of the POD coefficient is denoted as $a_i^{\mathrm{ROM-DA}}(t)$. The initial value for the forecast is the value of the temporal POD coefficient at the end of the assimilation regime *i.e.* $\boldsymbol{a}^{\mathrm{POD}}(t_{N_t})$. The forecast is compared with the corresponding trajectory obtained by integrating the artificially perturbed POD-ROM, $a_i^{\mathrm{ROM}}(t)$, and the reference trajectory $a_i^{\mathrm{POD}}(t)$. It can be seen that the perturbed POD-ROM is unstable within approximately 20 time steps and gives an inaccurate state estimate, while the stabilized POD-ROM provides an accurate forecast for about 70 time steps. Again, a slightly higher deviation of the forecast from the reference trajectory is observed for the 10% noise level than that corresponding to the low noise levels (0% and 1%). Note that as the exact parameters $\hat{\boldsymbol{\theta}}_s$ are not obtained towards the end of the assimilation regime, the forecast $a_i^{\mathrm{ROM-DA}}(t)$ also starts deviating from the reference trajectory over higher time steps for all the noise levels. However, in the forecast window, observations can be introduced to correct the ROM estimates at regular intervals. The time between two observations could be much larger than the observation time step Δt_o used during the DA, leading to a less expensive computational



(a) 0% noise



(b) 1% noise



(c) 10% noise

Figure 5.8: Evolution of the temporal POD coefficient $a_5(t)$ in the assimilation (left) and forecast (right) regimes corresponding to three noise levels of observations in the form of the temporal POD coefficient. The assimilated ($a_5^{\mathrm{DA}}(t)$), forecast ($a_5^{\mathrm{ROM-DA}}(t)$) and artificially perturbed ROM ($a_5^{\mathrm{ROM}}(t)$) trajectories are compared with the reference trajectory ($a_5^{\mathrm{POD}}(t)$).

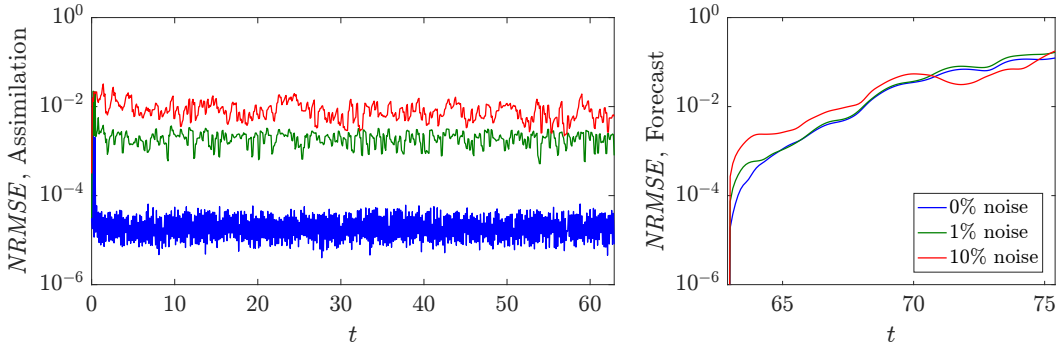


Figure 5.9: Time evolution of $NRMSE$ in the assimilation (left) and forecast (right) regimes corresponding to three noise levels of observations in the form of the temporal POD coefficient.

procedure. This approach will be illustrated in the next section.

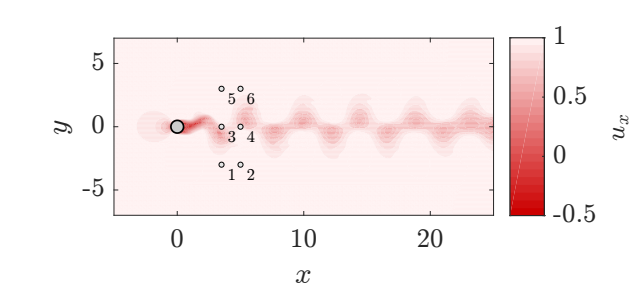
So far, it has been seen that the Dual-EnKF DA algorithm can be used to obtain a sufficiently accurate estimate of the temporal POD coefficients within the assimilation regime, while the stabilized ROM can be used to obtain an accurate long term forecast. The assimilation and forecast values of the POD coefficients $a_i(t)$ $(i = 1, \ldots, N_{\text{Gal}})$, can be used along with the known POD modes $\mathbf{\Phi}(\boldsymbol{\chi})$ to approximation by (2.2) the flow field variables. The performance of the Dual-EnKF algorithm can be quantified in terms of the normalized root-mean-square error ($NRMSE$). Here, the error is calculated with respect to the streamwise velocity fluctuations $u'_x(\boldsymbol{\chi}, t)$ obtained from the snapshot database using (2.1). The error $NRMSE$ is defined as

$$NRMSE(t) = \frac{\sqrt{\sum_{\ell=1}^{N_\chi} \left(u'_x(\boldsymbol{\chi}_\ell, t) - \widetilde{u}'_x(\boldsymbol{\chi}_\ell, t)\right)^2}}{\sqrt{\sum_{\ell=1}^{N_\chi} \left(u'_x(\boldsymbol{\chi}_\ell, t)\right)^2}}, \tag{5.10}$$

where N_χ is the spatial degree of freedom and $\widetilde{u}'_x(\boldsymbol{\chi}, t)$ represents the estimated velocity obtained by either using the assimilated $(a_i^{\text{DA}}(t))$ or forecast $(a_i^{\text{ROM-DA}}(t))$ temporal POD coefficients. The time evolution of $NRMSE$ is shown in Fig. 5.9. In the assimilation regime, it can be seen that as the noise level in the observation increases, the estimation error increases. For all the noise levels, the order of magnitude of the error practically remains constant during the assimilation window. In the forecast regime, the errors corresponding to different noise levels appear to have the same order of magnitude. However, unlike in the assimilation regime, the error in the forecast regime increases with time owing to the difference between the estimated and target values of the stabilizing parameters.

5.2.3.2 DA with streamwise velocity measurements as observation

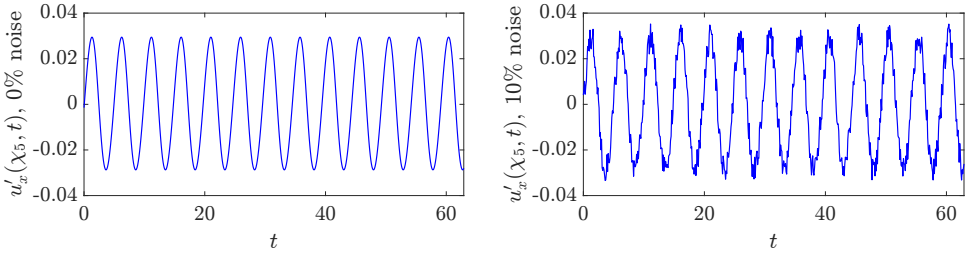
In this section, instead of taking the full temporal coefficient vector as observation, point measurements of the streamwise velocity component at few locations will be considered. Such consideration is motivated by what can be done in practice in an experimental context. Time survey of the velocity at discrete points of the flow is easily accessible in experiments, thanks to velocity probe measurements such as hot-wire anemometry for example. Here, the observation vector in (3.31) is thus defined as the streamwise



ℓ	$\boldsymbol{\chi}_\ell \coloneqq (x_\ell, y_\ell)$
1	$(3.5, -3.0)$
2	$(5.0, -3.0)$
3	$(3.5, 0.0)$
4	$(5.0, 0.0)$
5	$(3.5, 3.0)$
6	$(5.0, 3.0)$

Probe coordinates

(a)



(b)

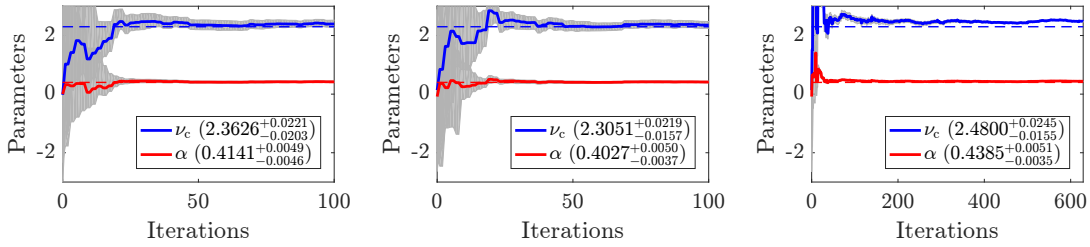
Figure 5.10: (a) Visual representation and coordinates of the probe locations for the streamwise velocity component measurements in the flow-field. The flow field is illustrated with a contour plot of the instantaneous streamwise velocity at $t = 50$. (b) Velocity fluctuation signals with no noise (left) and 10% noise (right) obtained from the probe at location $\boldsymbol{\chi}_5$.

velocity fluctuation, *i.e.* $\boldsymbol{y}_j^o = \{u'_x(\boldsymbol{\chi}_\ell, t_j)\}_{\ell=1}^{N_o}$, where $N_o = 6$ is the number of fictive probes and $\boldsymbol{\chi}_\ell$ is the coordinate vector of the ℓ-th probe. The fluctuating component $u'_x(\boldsymbol{\chi}, t)$ is obtained after subtracting the averaged component. The location of the probes in the flow-field and sample measurement signals are shown in Fig. 5.10. The spatial POD mode at each probe location serves as the linear observation operator $\boldsymbol{H}$ to map the temporal POD coefficients obtained from the forward model to the observed velocity fluctuations. The observation model (3.31) can thus be reformulated as,

$$\boldsymbol{u}'_x(\boldsymbol{\chi}, t_k) = \begin{bmatrix} u'_x(\chi_1, t_k) \\ \vdots \\ u'_x(\chi_{N_o}, t_k) \end{bmatrix} = \underbrace{\begin{bmatrix} \Phi^x_1(\chi_1) & \cdots & \Phi^x_{N_{\text{Gal}}}(\chi_1) \\ \vdots & \ddots & \vdots \\ \Phi^x_1(\chi_{N_o}) & \cdots & \Phi^x_{N_{\text{Gal}}}(\chi_{N_o}) \end{bmatrix}}_{\boldsymbol{H}} \boldsymbol{a}(t_k), \tag{5.11}$$

where $\Phi^x_i(\chi_j)$ $(i = 1, \cdots, N_{\text{Gal}})$ represents the streamwise component of the i-th spatial POD mode of the velocity fluctuations at the probe location χ_j.

The evolution of the parameters $\boldsymbol{\theta}_s$ during the course of DA using observations with three different noise levels is shown in Fig. 5.11. Again, it is observed that for all three noise levels, the analyzed parameters in the assimilation regime tend towards the target artificial value of $[\hat{\nu}_c \ \hat{\alpha}] = [2.3 \ 0.4]$. We also observe a reduction of the variance of the ensemble. For the lower noise levels (0% and 1%), the parameters at the end of the assimilation regime have values very close to the target value, while for the 10% noise level, the difference between the estimated and target values is larger. Like in



(a) 0% noise **(b)** 1% noise **(c)** 10% noise

Figure 5.11: Evolution of the stabilizing parameters $\boldsymbol{\theta}_s = [\nu_c \ \alpha]^\top$ of the POD-ROM in the assimilation regime corresponding to three noise levels of observations in the form of the streamwise velocity fluctuations. Refer to the caption of Fig. 5.7 for a detailed description.

the previous section, the Dual-EnKF algorithm is able to assimilate the noisy velocity fluctuation measurements and provide an estimate for the stabilizing parameters close to the target value.

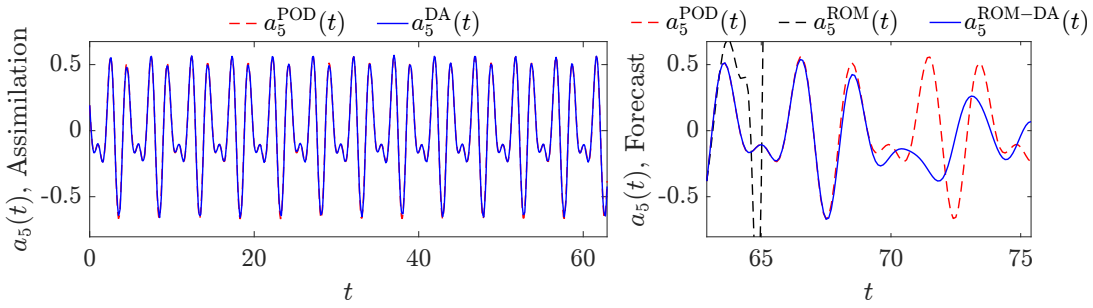
The evolution of the temporal POD mode $a_5(t)$ in the assimilation and forecast regimes are shown in Fig. 5.12(left) and Fig. 5.12(right), respectively. During the assimilation regime, the estimated coefficients $a_i^{\mathrm{DA}}(t)$ correctly replicate the true trajectory. For the 10% noise level, some small discrepancies are observed but the description remains well acceptable. In the forecast window, we observe similar results than those obtained when the full temporal coefficient vector was considered: the original trajectory is well replicated over a short-term horizon.

The time-dependent $NRMSE$ for this case is reported in Fig. 5.13. In the assimilation regime, the estimation error corresponding to the 10% noise level is slightly higher than for the other levels of noise but belong to the same order of magnitude. However, when comparing the error for the noiseless data in Fig. 5.9, the use of observations in the form of the temporal POD coefficient results in a relatively lower magnitude of error. In the forecast regime, the error increases with time which is similar in trend as that observed in the previous section in Fig. 5.9.

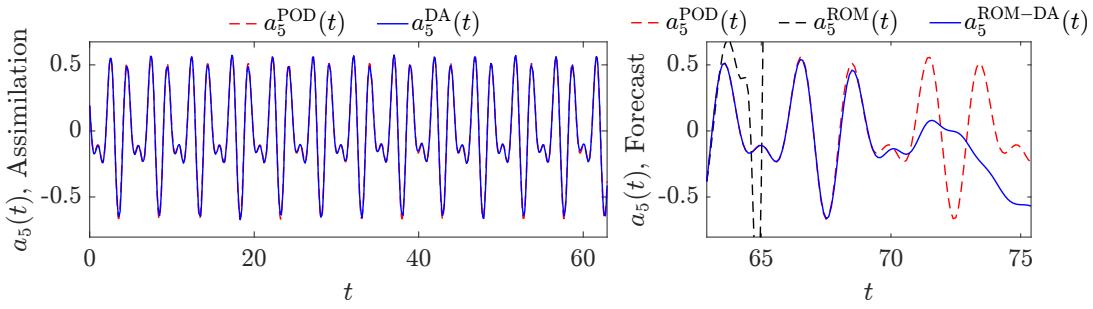
5.2.4 Intermediate outcomes

In this second test case, the Dual-EnKF has been shown to perform well to recover the parameter vector driving a residual term added to the conventional POD-ROM to take into account the energy dissipation. Another lesson learned is that, despite the good performance of the Dual-EnKF for recovering this parameter vector, even small errors in the estimate results in a corrected POD-ROM which can not be used for prediction over long-term horizon. Assimilation of new observations needs to be maintained, at least at regular time intervals when the model starts to drift from the original trajectory.

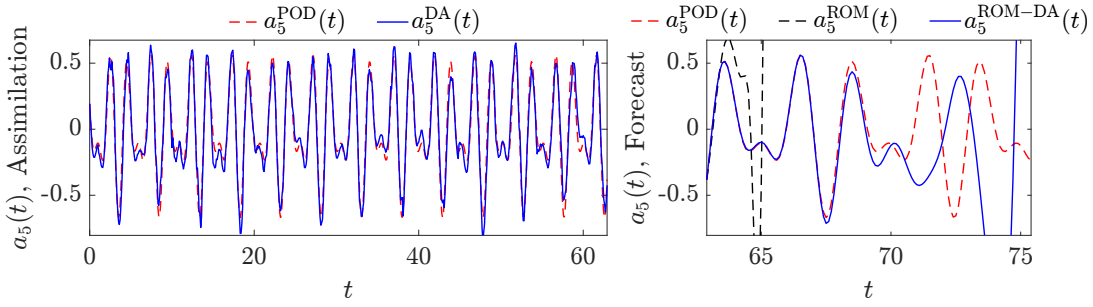
One way to obtain an accurate forecast over longer time range is to use the learned stabilized model in conjunction with observations from the flow-field to correct the model estimates using EnKF (see Sec. 3.2.2). As the model parameters have been corrected, we can expect that the state estimate does not deviate from the reference trajectory within a few time steps of integration. For this reason, the assimilation can be performed at larger



(a) 0% noise



(b) 1% noise



(c) 10% noise

Figure 5.12: Evolution of the temporal POD coefficient $a_5(t)$ in the assimilation (left) and forecast (right) regimes corresponding to three noise levels of observations in the form of the streamwise velocity fluctuations. Refer to the caption of Fig. 5.8 for a detailed description.

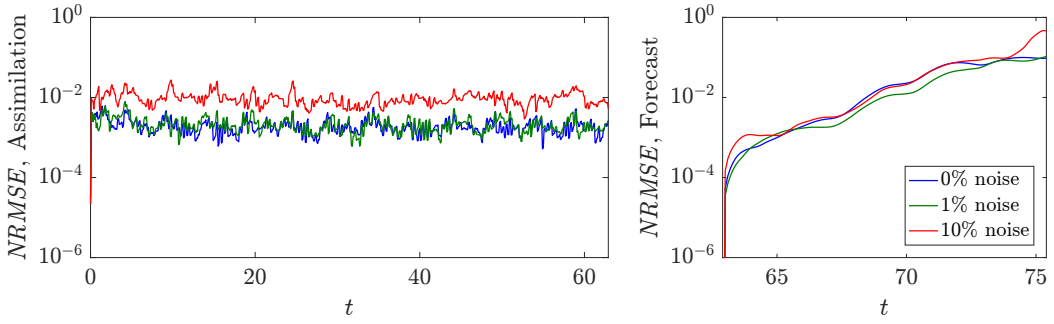
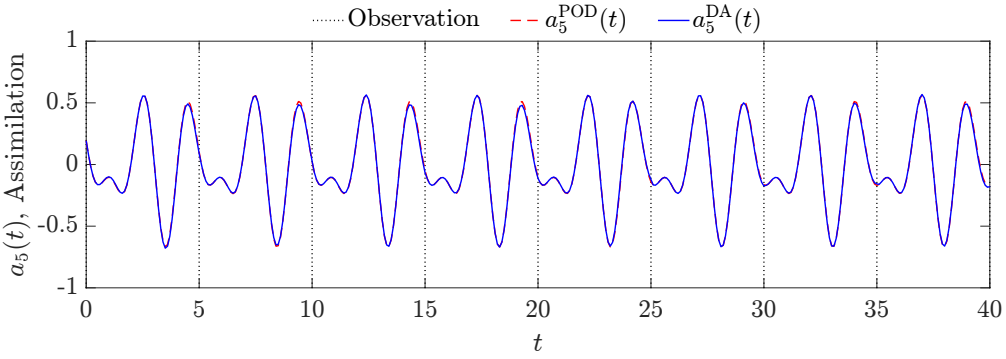
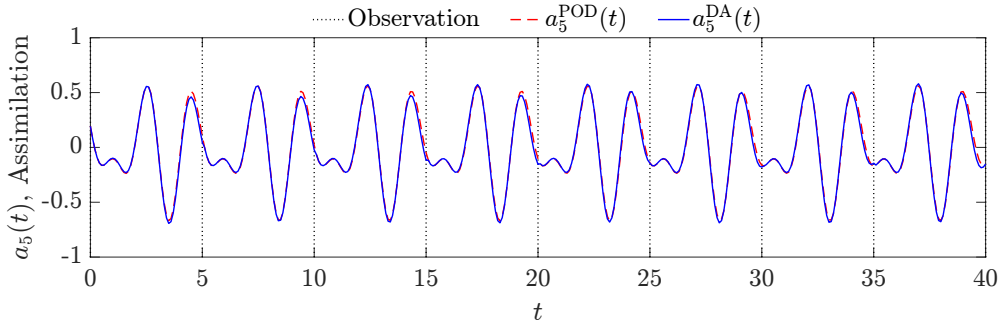


Figure 5.13: Time evolution of $NRMSE$ in the assimilation (left) and forecast (right) regimes corresponding to three noise levels of observations in the form of the streamwise velocity fluctuations.



(a) 0% noise



(b) 10% noise

Figure 5.14: Time evolution of the temporal POD coefficient $a_5^{\mathrm{DA}}(t)$ obtained from EnKF assimilation of the observations (streamwise velocity fluctuations) compared with the reference trajectory $a_5^{\mathrm{POD}}(t)$. The model parameters of the stabilized POD-ROM are used. The black dotted lines indicate the instants when the observations are assimilated.

time interval as compared to that used in the Dual-EnKF algorithm. As an illustration, Fig. 5.14 shows the long-term forecast obtained from the EnKF assimilation of observations. It is noted that as the forecast window is short, the demonstration is performed on a dataset which belongs to the assimilation window (*i.e.* $t = 0$ corresponds to the start of the assimilation window) to have a possibility to demonstrate the performance of the proposed framework over a long time range. Also, for this specific case where the dynamics is periodic, the generality of the results is not lost. For the assimilation, the observations are considered to be available at a time step of $\Delta t_o = 50\Delta t = 5$ (the time step used in Dual-EnKF was $\Delta t_o = 2\Delta t$). It is observed that the state estimated from the time integration of the model follows the reference trajectory in the interval between two observations. Moreover, the EnKF algorithm is able to update the prediction when observation is available, thus ensuring that the model forecast does not deviate much from the reference trajectory. This also offers a reduction in the computational cost as the assimilation is performed over significantly large time intervals.

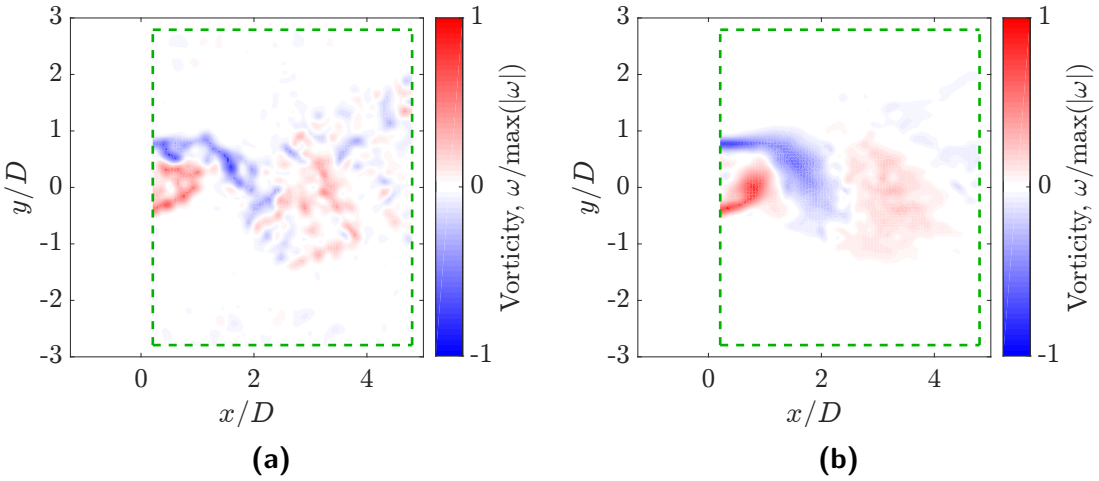


Figure 5.15: Normalized vorticity field at a time instant $t = 100$ s obtained from the PIV measurements (a), and reconstructed using the 10 most energetic POD modes (b) for the cylinder wake flow at $Re = 1.5 \times 10^4$. The PIV measurement field in the near-wake region is depicted by the green box.

5.3 Test case 2: Experimental cylinder wake flow at $Re_D = 1.5 \times 10^4$

The Dual-EnKF data assimilation technique is now applied to experimental data obtained for a cylinder wake flow at two different Reynolds numbers. The available data, obtained from Particle Image Velocimetry (PIV), are necessarily limited both in terms of spatial and temporal resolution, and are corrupted with unknown measurement noise. The objective here is therefore to demonstrate that the analysis framework discussed so far can accommodate these limitations.

5.3.1 Snapshot dataset

The dataset were obtained from the experimental work performed by Benard et al. (2010). The experimental setup is briefly described in this section.

The cylinder wake flow was experimentally studied in an open Eiffel-type wind tunnel. The cylinder, which spans the entire test section, has a diameter of $D = 40$ mm and an aspect ratio of $L/D = 7.5$. The upstream velocity in the measurement section was set as $U_\infty = 5.6$ m/s, giving a Reynolds number based on the cylinder diameter of $Re = 1.5 \times 10^4$. The Reynolds number is therefore an order of magnitude lower than the critical Reynolds number ($Re_c \sim 2.0 \times 10^5$, see Williamson 1996) at which turbulence transition occurs in the detached shear layers. The flow over the cylinder remains in the sub-critical regime with laminar boundary layer separation, while a wide turbulent wake develops downstream. The natural frequency of the vortex shedding, obtained from the post-processing of the available data, is equivalent to a Strouhal number, based on the cylinder diameter and freestream velocity, of $St = 0.18$ (or a non-dimensional time period of $T_s U_\infty / D = 5.39$).

The two components of velocity in the near wake of the cylinder were measured with a

particle image velocimetry (PIV) system consisting of a fast CCD camera (Photron, APX-RS), a fast dual oscillator single-head laser (Quantronix, Darwin-Duo), a synchronization unit (EG, R&D Vision) and an acquisition PC. The cylinder was illuminated in the mid-span by a 1 mm thin laser sheet. The measurement field, shown in Fig. 5.15a, covers the region defined by the bounds $0.2 < x/D < 4.8$ and $-2.8 < y/D < 2.8$. A CCD sensor of resolution 1024×1024 pixel^2 was used. The acquisition was carried out at a frequency of 1000 Hz, which corresponds to a non-dimensional time interval between two consecutive snapshots of $\Delta t U_\infty/D = 0.14$. This implies that, in theory, a shedding cycle is discretized by about 38 instantaneous snapshots. The velocity vectors were obtained using LaVision's Davis software by computing cross-correlations with windows ranging in size from 64×64 pixel^2 to 16×16 pixel^2 in the final pass, each with a 50% overlap. This resulted in a uniform grid of size $N_y \times N_x = 108 \times 89$, *i.e.* $N_\chi = N_x N_y = 9612$ nodes, with a spatial resolution of $\Delta x = \Delta y = 2.09$ mm. The overall time sequence spans the time range $t \in [0,\ 1]$ s (*i.e.* 1001 snapshots), which is equivalent to $tU_\infty/D \in [0,\ 140]$ in terms of the non-dimensional variable.

5.3.2 Identification of the POD-ROM

The $N_t = 1001$ snapshots obtained from PIV measurements are used to determine the POD basis. The relative information content (RIC) are reported in Fig. 5.16 as a function of the mode number. The convergence rate is quite slow and the first 43 modes contain about 90% of the total kinetic energy in the system. The first $N_{\text{Gal}} = 10$ modes, which together account for approximately 79% of the overall energy, are retained to build a reduced-order model of the general form given by (2.79). The reconstructed vorticity field using the N_{Gal} first modes is shown in Fig. 5.15b. Comparing with the corresponding field obtained from the PIV measurements, it can be seen that while the small scale structures are not resolved by the reconstruction, the large scale structures are still well represented. Therefore, the reduced-order system based on the truncated modes serves the objective of modeling the dynamics of the coherent structures, *i.e.* the temporally persistent regions of concentrated vorticity.

The structure of the spatial modes $\mathbf{\Phi}_i^{x,\text{POD}}(\boldsymbol{\chi})$ $(i = 1, 9)$ corresponding to the streamwise velocity fluctuations and the time evolution of the pairs of temporal POD coefficients a_i^{POD} $(i = 1, 2$ and $i = 9, 10)$ are shown in Fig. 5.17. Similarly to the cylinder wake flow at low Reynolds number flow discussed in Sec. 5.2, the mode $\mathbf{\Phi}_1$ represents the coherent vortex-shedding structures and pairs with the mode $\mathbf{\Phi}_2$ (not shown) with similar structures but shifted in the streamwise advection direction. The corresponding POD coefficients a_1 and a_2 are also analogously shifted in time. The higher modes $(n > 2)$ represent subscale turbulence structures and exhibit non-periodic temporal dynamics. In the current context, however, the mode structures contribute little to the reduced-order system identification, and the focus will be on the temporal evolution of the modes.

Following the same approach as in the previous section, a reduced-order model without the stabilization terms, is first calibrated. The identification is done using the first 700 snapshots (*i.e.* 70% of the total number of snapshots) which form the training dataset. The remaining snapshots form the testing dataset used to evaluate the learned model. The train-test split for the reduced-order model identification is illustrated in Fig. 5.18. The parameters vectors θ_i are identified using the sparse-identification method SINDy

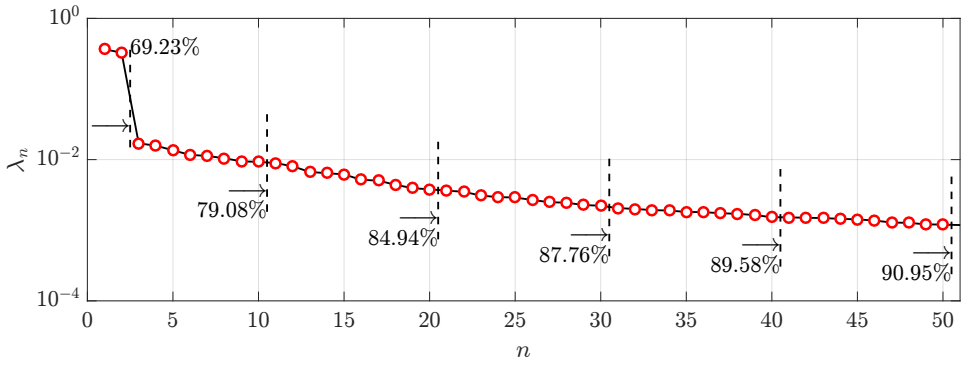


Figure 5.16: POD energy content of the most energetic modes for the cylinder wake flow at $Re = 1.5 \times 10^4$. Eigenvalue spectrum of the POD modes (λ_n). The values of $RIC(n)$ are also indicated, representing the fraction of the total fluctuating kinetic energy captured if only the POD modes up to the dashed lines are considered.

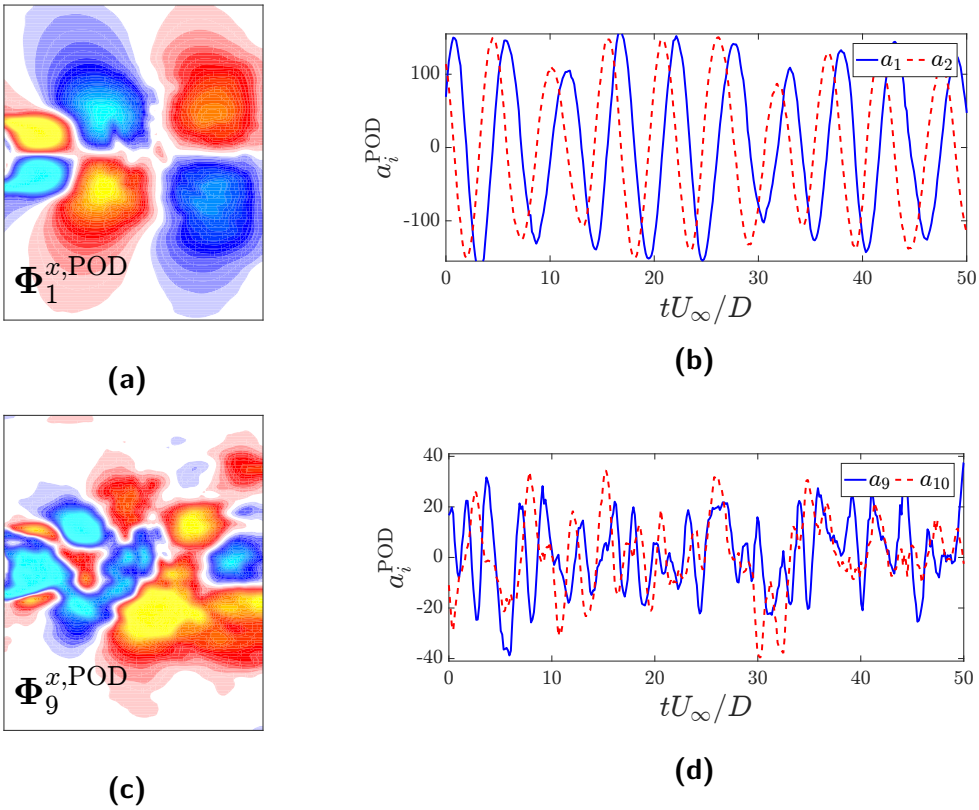


Figure 5.17: (a,c) Spatial POD modes $\mathbf{\Phi}_i^{x,\mathrm{POD}}$ $(i = 1, 9)$ corresponding to the streamwise velocity fluctuations. (b,d) Time evolution of the temporal POD coefficients a_i^{POD} $(i = 1, 2, 9, 10)$ corresponding to the most energetic modes of the cylinder wake flow at $Re = 1.5 \times 10^4$.

with $\lambda = 1.0 \times 10^{-3}$ as sparsification parameter. The POD-ROM thus identified is considered as a basis for further alteration (see Sec. 5.3.3).

Using the same representation as in Sec. 4.4, the identified parameters are shown in

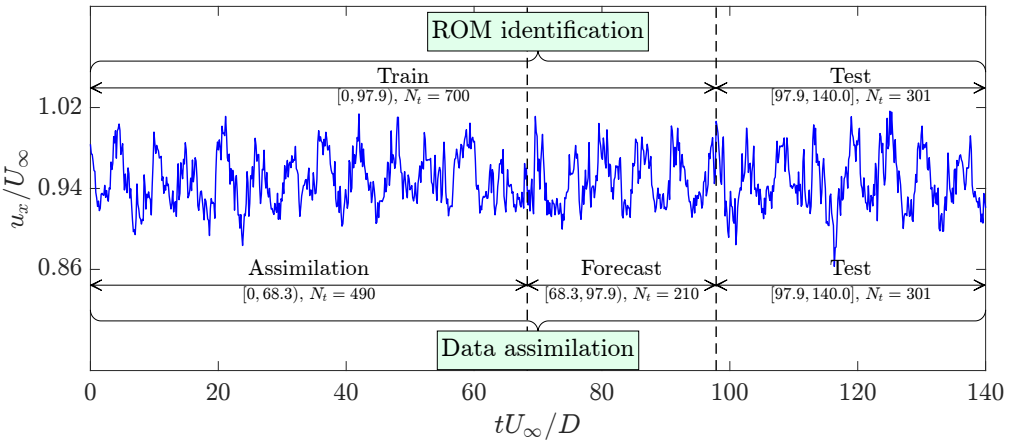


Figure 5.18: Illustration of the splitting of the $N_t = 1001$ snapshots obtained from PIV measurements for the ROM identification and data assimilation. For the ROM identification with SINDy, the full time series of snapshots is split in $7:3$ ratio to obtain the training and testing datasets. For the data assimilation, the previous training dataset is further split in $7:3$ ratio to obtain the assimilation and forecasting sub-datasets. The remaining snapshots are used as testing dataset. The sample signal corresponds to the time evolution of streamwise velocity magnitude at a location $(x/D, y/D) = (2.6, 2.8)$ within the measurement field.

Fig. 5.19a. The SINDy identification method provides a sparse solution with regards to the model coefficients. In terms of magnitude, the major contribution to the dynamics is found to appear only from the constant and linear terms in the ROM.

The time evolution of the estimated POD coefficient $a_i^{\text{Prop}}(t)$ ($i = 1, 5$ and 9) obtained from the time integration of the identified POD-ROM in the training window, with $a_i^{\text{Prop}}(0) = a_i^{\text{POD}}(0)$ as initial condition, is reported in Fig. 5.19(b-d)(left). The time evolution of the actual POD coefficients $a_i^{\text{POD}}(t)$ is also reported for comparison. The dominant mode $a_1(t)$ is predicted with a phase shift with respect to the actual dynamics – Fig. 5.19b(left). This implies that the large scale structures associated with the mode $\mathbf{\Phi}_1$ are well represented by the reduced-order model albeit with a phase shift. In contrast, the reduced-order model is unable to represent the high frequency dynamics of the actual POD coefficients for the subsequent modes. This can be attributed to the truncation of the POD modes to construct the reduced-order model which results in the loss of information of the less energetic modes which are associated with the higher frequency dynamics. When testing the identified POD-ROM outside the training window, as shown in Fig. 5.19(b-d)(right), similar conclusions can be drawn.

5.3.3 Dual-EnKF estimation of stabilization parameters

While the identified POD-ROM is not perfect, it can, once again, serve as a first guess. Similarly to the previous section, a modal constant eddy viscosity closure model given by (2.84) is added to the identified model as residual term. The Dual-EnKF is then applied to recover an estimation of the stabilizing parameter vector $\mathbf{\Theta} = \boldsymbol{\theta}_s = [\nu_c \ \alpha]^\top$ and to correct for drift in the estimated trajectories of the temporal coefficients.

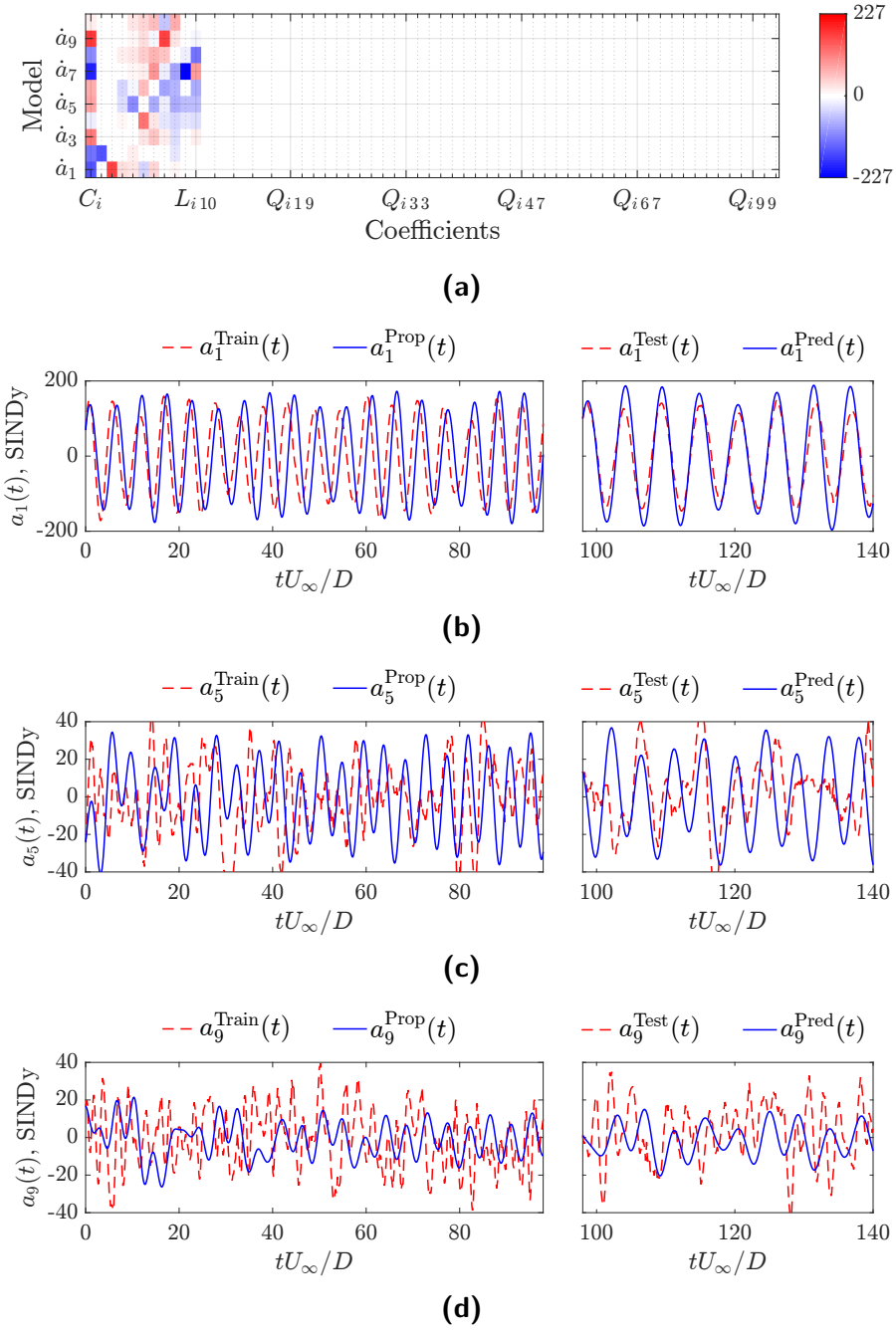
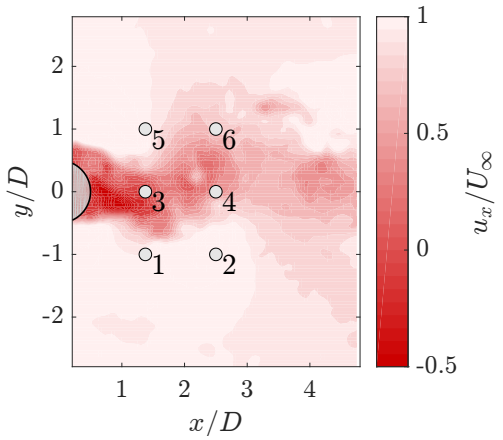


Figure 5.19: (a) Scaled color representation of the parameters identified using SINDy for the POD-ROM of a cylinder wake flow at $Re = 1.5 \times 10^4$. (b,c,d) Evolution of coefficients $a_i(t)$ $(i = 1, 5, 9)$ in the training regime (left) and in the testing regime (right). True (dashed line) and estimated (solid line) trajectories.



ℓ	$\boldsymbol{\chi}_\ell := (x_\ell/D, y_\ell/D)$
1	$(1.4, -1.0)$
2	$(2.5, -1.0)$
3	$(1.4, 0.0)$
4	$(2.5, 0.0)$
5	$(1.4, 1.0)$
6	$(2.5, 1.0)$

Probe coordinates

Figure 5.20: Visual representation and coordinates of the probe locations for the streamwise velocity component measurements used for parameter identification. The flow field is illustrated with a contour plot of the instantaneous streamwise velocity at $tU_\infty/D = 50$.

The assimilation-forecast split of the dataset for the DA is illustrated in Fig. 5.18. The 700 snapshots used to identify the ROM is split in $7:3$ ratio, resulting in 490 snapshots allocated for the Dual-EnKF assimilation and 210 snapshots for the forecast step which is used to validate the assimilated model. The subsequent 301 snapshots, which are hidden from both the ROM calibration and DA steps, are used for evaluating the performance of the final model.

Following the notations introduced in Sec. 3.2, the number of state elements in the data-assimilation problem is $N_s = N_{\text{Gal}} = 10$. The model error is assumed to be given by a Gaussian distribution $\boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{Q})$ with zero mean and covariance $\boldsymbol{Q} = 1.0 \times 10^{-3} \boldsymbol{I}_{N_s}$. The initial estimate of the state vector is given as $\boldsymbol{a}^{\text{DA}}(0) = \boldsymbol{a}^{\text{POD}}(0)$. The propagation of the stabilizing parameters vector $\boldsymbol{\theta}_s$ follows the random walk given by (3.71) with a Gaussian distributed additive perturbation $\boldsymbol{\xi}^{(n)} \sim \mathcal{N}(\mathbf{0}, h^2\boldsymbol{C})$ with zero mean and covariance $\boldsymbol{C} = 1.0\boldsymbol{I}_{N_p}$. The estimated parameter vector is initialized as $\boldsymbol{\Theta}_k = \mathbf{0}_{N_p \times 1}$.

The data assimilation is performed using point measurements of the streamwise velocity component at few locations in the flow field, motivated by the discussion in Sec. 5.2.3.2. As before, the observation vector in (3.31) is defined accordingly as the streamwise component of the velocity fluctuation, *i.e.* $\boldsymbol{y}_j^o = \{u_x'(\boldsymbol{\chi}_\ell, t_j)\}_{\ell=1}^{N_o}$, where $N_o = 6$ is the number of fictive probes and $\boldsymbol{\chi}_\ell$ is the coordinate vector of the ℓ-th probe. The locations of the probes in the flow field are shown in Fig. 5.20. The observation equation (5.11) maps the temporal POD coefficients obtained from the forward model to the observed velocity fluctuations. For the assimilation, the noise is assumed to be given by a Gaussian distribution $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{R})$ with zero mean and covariance $\boldsymbol{R} = \eta\boldsymbol{I}_{N_o}$, where the covariance level $\eta = 0.01$ is representative of the observed streamwise velocity fluctuation level of $\sim 1\%$.

The data assimilation is performed for different values of the ensemble sizes N_e to verify the convergence of the identified parameters. To evaluate the performance of the Dual-EnKF assimilation, the standard average normalized RMS error between the actual

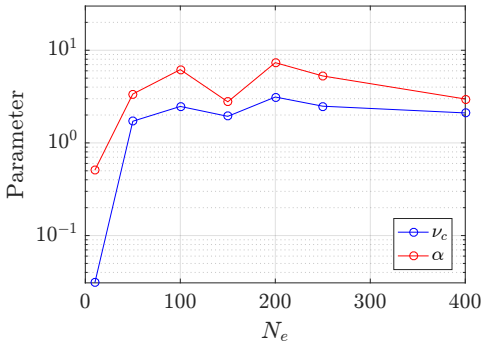


Figure 5.21: Values of the stabilizing parameters ν_c and α obtained using the Dual-EnKF data assimilation for different values of ensemble size N_e.

Table 5.3: Average $NRMSE$ in the forecast window for different values of ensemble size N_e.

N_e	Average $NRMSE$
10	1.5146
50	1.4784
100	1.4457
150	1.8391
200	1.5224
250	1.4215
400	1.7811

and estimated temporal coefficients is calculated in the forecast window as

$$\text{Average } NRMSE = \frac{1}{N_t} \sum_{k=1}^{N_t} \sqrt{\frac{\sum_{i=1}^{N_s} \left(a_i^{\text{DA}}(t_k) - a_i^{\text{POD}}(t_k)\right)^2}{\sum_{i=1}^{N_s} \left(a_i^{\text{POD}}(t_k)\right)^2}}. \tag{5.12}$$

The values of the components in the stabilizing parameters vector $\boldsymbol{\theta}_s$ is shown in Fig. 5.21 for different ensemble sizes. The corresponding values of the average $NRMSE$ are given in Tab. 5.3. It is observed that the parameter values converge to nearly the same order of magnitude for ensemble sizes $N_e > 10$ and that the value of the error also remains of the same order of magnitude. In the following, we choose for the ensemble size $N_e = 250$ to keep a good compromise between the number of elements and the error value.

The evolution of the parameters $\boldsymbol{\theta}_s$ during the course of DA using streamwise velocity measurements as observations is shown in Fig. 5.22. The analyzed parameters in the assimilation window tend towards $[\nu_c \ \alpha] = [2.49 \ 5.28]$ and are accompanied with a reduction of the variance of the ensemble.

The evolution of the POD coefficients $a_1(t)$ and $a_9(t)$ in the assimilation and forecast windows are shown in Fig. 5.23. For the coefficient $a_1(t)$, the Dual-EnKF method is able to assimilate the observations such that it follows the actual trajectory obtained from POD in the assimilation window. For the coefficient $a_9(t)$, which exhibits high frequency dynamics, the DA method initially fails to follow the actual trajectory but subsequently improves during the course of assimilation. In the forecast window, for the coefficient $a_1(t)$, the results obtained from both the initial ROM and the ROM with stabilization parameters are comparable. However, for the coefficient $a_9(t)$, the dynamics obtained from the ROM with stabilization parameters appears to be an improvement over the results obtained from the initial ROM, especially in terms of replicating the harmonics.

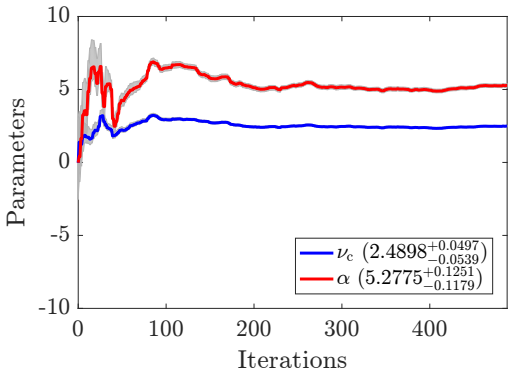


Figure 5.22: Evolution of the stabilizing parameters $\boldsymbol{\theta}_s = [\nu_c\ \alpha]^\top$ of the POD-ROM in the assimilation window for the modal constant closure and $\eta = 1\%$. The gray shaded area shows the spread of the parameter ensemble. The parameter values at the end of the assimilation along with the ensemble interval is shown in the legend.

5.3.4 Flow reconstruction using the ROM with stabilization parameters

The performance of the initial ROM and the ROM with stabilizing terms obtained from data assimilation will be discussed in this section for the test dataset. To obtain a better prediction over longer time range, the learned stabilized ROM and regular new observations are used in common to solve by EnKF (see Sec. 3.2.2) a data assimilation problem in the test window. Assimilation of new observations is maintained at regular time intervals when the model starts to drift from the original trajectory. To showcase the independence of the assimilation method to the observations, a probe configuration different from the one used during the Dual-EnKF assimilation will be used. The probe locations for the streamwise velocity component measurements are shown in Fig. 5.24.

As the model parameters have been corrected in the assimilation window, in the test window we observe that the state estimates do not diverge from the actual trajectory within a few time steps of integration. Hence, the EnKF assimilation can be performed at larger time interval as compared to that used in the Dual-EnKF algorithm. A similar procedure was illustrated in Sec. 5.2.4 for the cylinder wake flow at low Reynolds number. As an illustration, Fig. 5.25a shows the long-term prediction obtained from the EnKF assimilation of observations in the test window. The observations are assimilated from $t_0 U_\infty/D = 98.4$ (in non-dimensional form) at $N_{t,o} = 4$ time instants shown by vertical dotted lines in Fig. 5.25a. The step size between two assimilations is equal to $\Delta t_o U_\infty/D = 11.2$ and corresponds to 80 times the acquisition step size ($\Delta t U_\infty/D = 0.14$). This assimilation step size (Δt_o) approximately spans two cycles of vortex shedding. It is observed that the POD coefficient estimated from the time integration of the initial ROM starts to deviate from the actual trajectory towards the end of the test window. In contrast, the assimilated trajectory of the coefficient estimated from the stabilized ROM follows the actual trajectory in the interval between the observations. The EnKF algorithm which updates the prediction when an observation is available, ensures that the model prediction does not deviate much from the actual

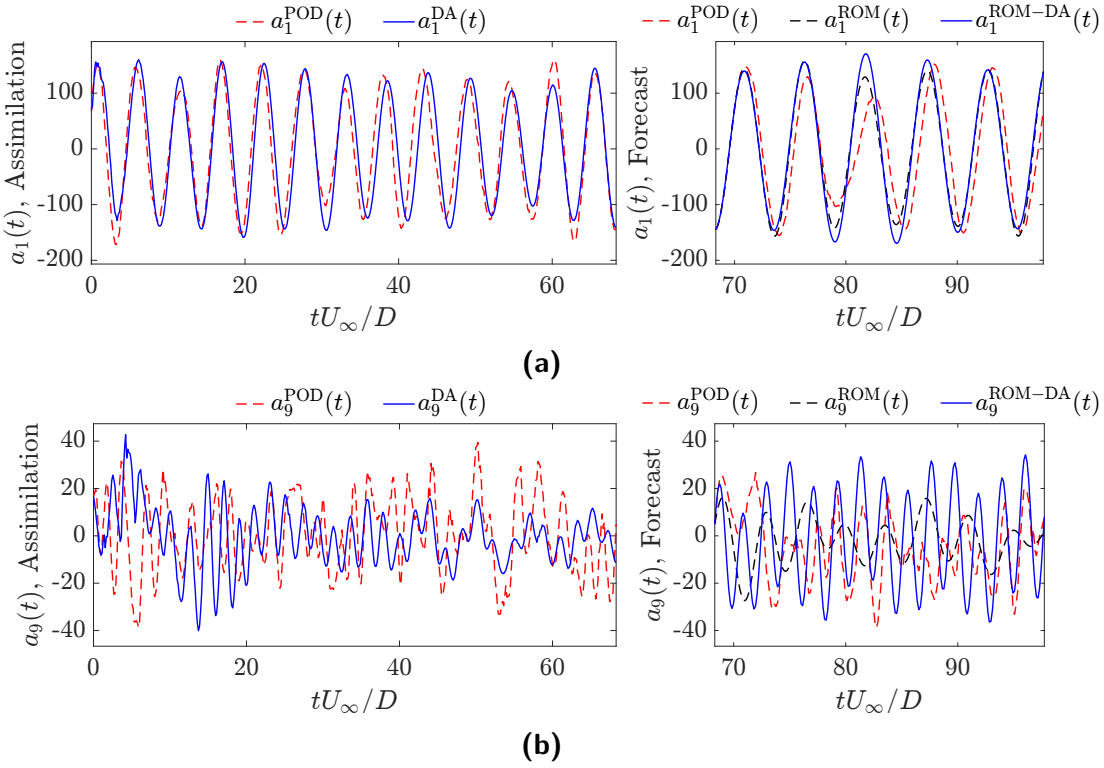
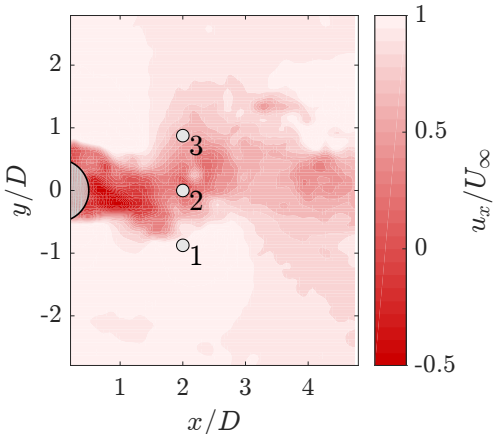


Figure 5.23: Evolution of the temporal POD coefficients $a_1(t)$ (a) and $a_9(t)$ (b) in the assimilation (left) and forecast (right) windows after Dual-EnKF data assimilation using streamwise velocity fluctuations as observation. The assimilated ($a_i^{\mathrm{DA}}(t)$), forecast ($a_i^{\mathrm{ROM-DA}}(t)$) and ROM solution ($a_i^{\mathrm{ROM}}(t)$) trajectories are compared with the actual trajectory ($a_i^{\mathrm{POD}}(t)$).



ℓ	$\boldsymbol{\chi}_\ell := (x_\ell/D, y_\ell/D)$
1	$(2.0, -0.9)$
2	$(2.0, 0.0)$
3	$(2.0, 0.9)$

Probe coordinates

Figure 5.24: Visual representation and coordinates of the probe locations for the streamwise velocity component measurements used for the EnKF data assimilation in the test window. The flow field is illustrated with a contour plot of the instantaneous streamwise velocity at $tU_\infty/D = 50$.

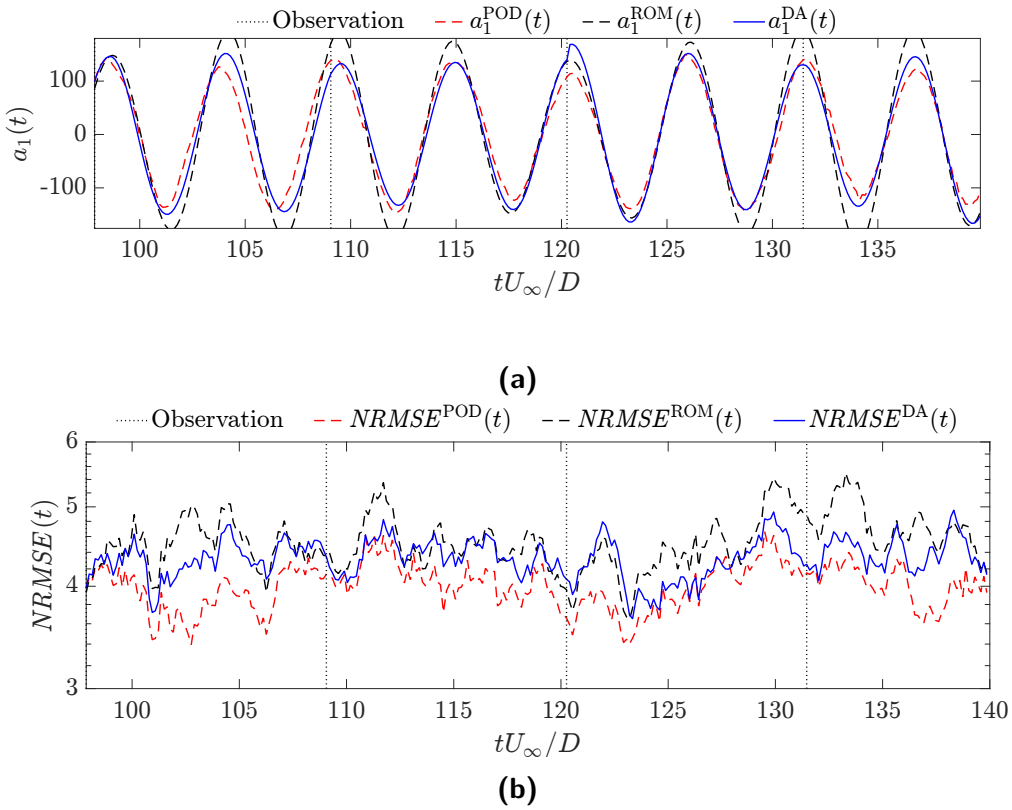


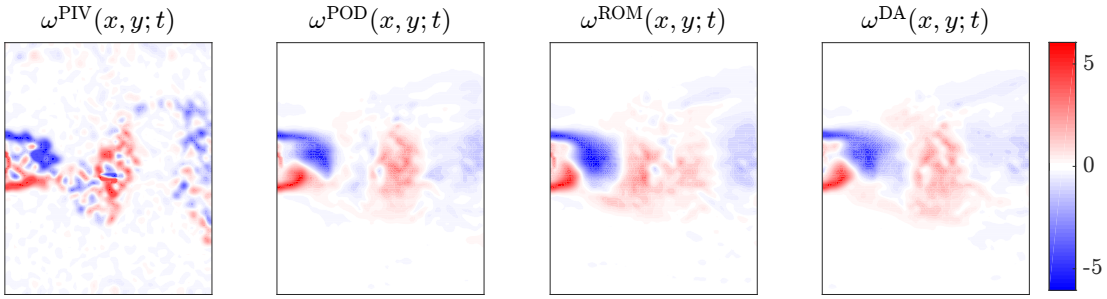
Figure 5.25: (a) Time evolution of the temporal POD coefficient $a_1^{\mathrm{DA}}(t)$ obtained from EnKF assimilation of the observations (streamwise velocity fluctuations) in the test window. Comparison with the model estimates obtained from integrating the stabilized POD-ROM and the actual trajectory $a_1^{\mathrm{POD}}(t)$. The black vertical dotted lines indicate the instants when the observations are assimilated. (b) Time evolution of the error $NRMSE$ of the vorticity fields with respect to the vorticity calculated from the PIV measurements.

trajectory. To evaluate and compare the performance of the initial ROM and stabilized and assimilated ROM in the test window, the vorticity field $\omega^{\mathrm{ROM}}(\boldsymbol{\chi}, t)$ is reconstructed. We then define the error $NRMSE$ in terms of vorticity as

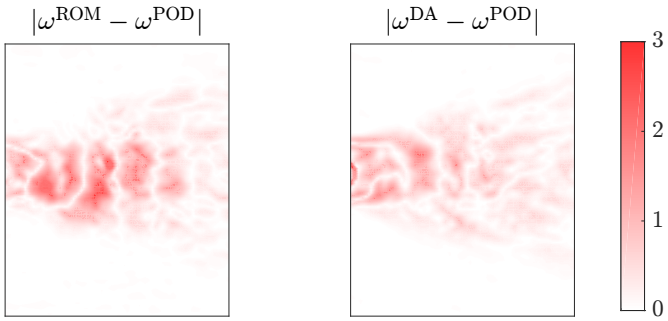
$$NRMSE(t) = \sqrt{\frac{\sum_{\ell=1}^{N_\chi} \left(\omega(\boldsymbol{\chi}_\ell, t) - \omega^{\mathrm{PIV}}(\boldsymbol{\chi}_\ell, t)\right)^2}{\sum_{\ell=1}^{N_\chi} \left(\omega^{\mathrm{PIV}}(\boldsymbol{\chi}_\ell, t)\right)^2}}, \tag{5.13}$$

where N_χ is the spatial degree of freedom and $\omega^{\mathrm{PIV}}(\boldsymbol{\chi}, t)$ is the vorticity field obtained from the PIV measurements. The time evolution of the error $NRMSE$ is shown in Fig. 5.25b. It is observed that the error corresponding to the vorticity field reproduction using the stabilized and assimilated ROM is lower than that using the initial ROM, resulting in a 4.6% lower average $NRMSE$ over the testing window. The error also remains bounded and reaches levels close to that obtained from the calculated POD modes in the long-term. This highlights the improvement of reconstructions in the physical space offered by the stabilized and assimilated ROM.

For a qualitative comparison, the reconstructed vorticity fields at an arbitrary time



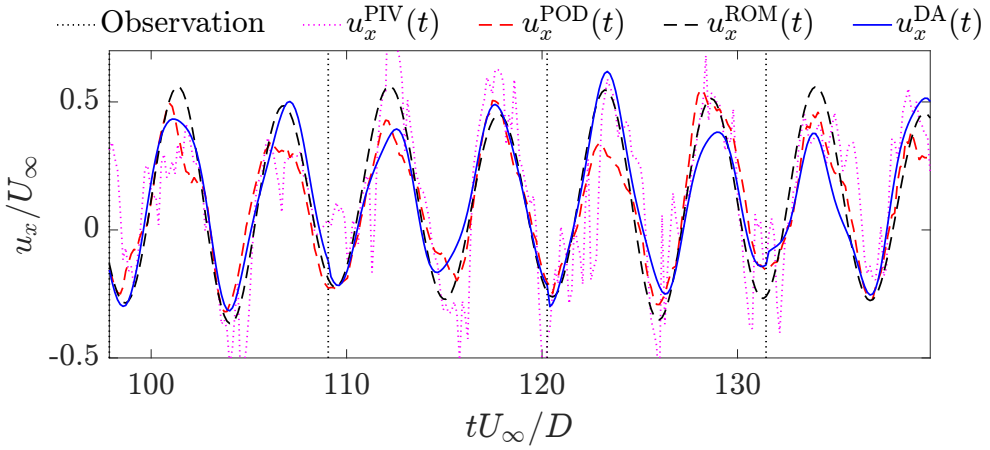
(a)



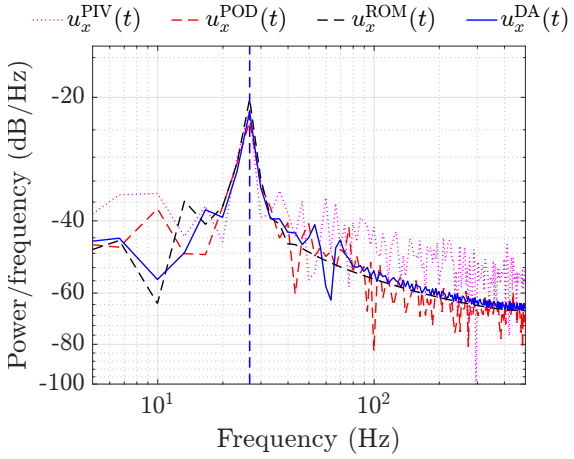
(b)

Figure 5.26: (a) Instantaneous vorticity fields at $tU_\infty/D = 100$ obtained using the velocity obtained from the PIV measurements ($\omega^{\mathrm{PIV}}(x,y;t)$) and different reconstructions using the POD coefficients ($\omega^{\mathrm{POD}}(x,y;t)$), the initial ($\omega^{\mathrm{ROM}}(x,y;t)$) and stabilized ($\omega^{\mathrm{DA}}(x,y;t)$) ROMs. (b) Absolute error between the reconstructed vorticity fields using the initial and stabilized ROMs and that using the POD coefficients.

instant $t = 100$ in the test window are shown in Fig. 5.26. The vorticity field obtained from the stabilized and assimilated ROM exhibits an improvement over that obtained using the initial ROM when compared with the reconstructed field from the POD modes. The measured and reconstructed time evolution of the streamwise velocity component and its power spectrum at a near-wake location are also compared in Fig. 5.27. The reconstructed time evolution of the velocity component obtained from the stabilized ROM follows closely the time evolution of that obtained from the POD modes. The dominant frequency is given correctly by both the initial and stabilized ROM. However, the amelioration of the ROM using the observations in the test window leads to a better representation of the energy associated with the near-dominant frequencies by the stabilized and assimilated ROM. Therefore, it can be inferred that the reconstruction using the stabilized and assimilated ROM offers a definite improvement in the time and frequency domains. The inclusion of the stabilizing parameters and the assimilation of observations during the model integration provides a framework for robust long term estimation of the flow dynamics.



(a)



(b)

Figure 5.27: (a) Time evolution of the streamwise velocity component at a location $(x/D, y/D) = (0.5, -0.2)$ in the flow field. Comparisons of the velocity obtained from the PIV measurements ($u_x^{\mathrm{PIV}}(t)$) and different reconstructions using the POD coefficients ($u_x^{\mathrm{POD}}(t)$), the initial ROM ($u_x^{\mathrm{ROM}}(t)$) and the stabilized and assimilated ROM ($u_x^{\mathrm{DA}}(t)$). (b) Power spectrum densities of the velocity measured using PIV and the reconstructed velocity signals using the POD coefficients and the initial and stabilized ROMs. The peak frequencies of the measured signal and that reconstructed from the stabilized ROM (which coincide here) are indicated by dashed lines.

5.4 Test case 3: Experimental cylinder wake flow at $Re_D = 5.5 \times 10^4$

To test the robustness of the proposed strategy with regards to an increase of the Reynolds number, and indirectly to an increase of the number of degrees of freedom of the flow itself, the same methodology is now applied to an experimental cylinder wake flow at a Reynolds number equal to 5.5×10^4. The experimental configuration is similar to that described previously. Only the freestream velocity is different, now $U_\infty = 20.6$ m/s.

The database, again, consists of $N_t = 1001$ snapshots obtained from the PIV measurements. These snapshots are used to determine the POD basis. The relative information content (RIC) is reported in Fig. 5.28 as a function of the mode number. Comparing with the results of Fig. 5.16 at $Re_D = 1.5 \times 10^4$, it can be observed that the convergence rate is slower. This is well reflected in the first ten modes which together account for only approximately 73% of the overall energy. In the following, these first $N_{\text{Gal}} = 10$ modes are retained to build a reduced-order model of the general form given by (2.79).

As in the previous test case, the database is split into two parts. The first 700 snapshots (*i.e.* 70% of the total number of snapshots) form the training dataset and are used to calibrate an initial ROM without the residual term. The remaining part of the database forms the testing dataset and is used to evaluate the learned model. The sparse-identification method SINDy with $\lambda = 1.0 \times 10^{-2}$ as sparsification parameter is used to identify the parameters vector $\boldsymbol{\theta}$. Using the same representation as in Sec. 4.4, the result of the identification in terms of parameters is shown in Fig. 5.29a. Similar to the previous case, the SINDy identification method provides a sparse solution where the major contribution to the dynamics is found to appear only from the constant and linear terms of the ROM. Time history of the estimated POD coefficient $a_1^{\text{Prop}}(t)$ and $a_1^{\text{Pred}}(t)$ in the training and testing windows as obtained from the integration of the identified ROM are reported in Fig. 5.29b. The actual temporal trajectory of the POD coefficient is also reported for comparison. The estimated coefficients are found well bounded over time but a phase shift with respect to the actual dynamics is observed. Again, this discrepancy can be mainly attributed to the truncation of the POD modes in the ROM construction.

The modal constant eddy viscosity closure is then introduced. The identified ROM is augmented with the residual term (5.9). The Dual-EnKF method is applied to identify the parameter vector $\boldsymbol{\theta}_S$. The same observers than that used at $Re_D = 1.5 \times 10^4$ are considered here. We can see in the testing window of Fig. 5.29b that the error between the ROM estimates and the actual dynamics is an order of magnitude higher than in the lower Reynolds number case. Hence, we heuristically fixed the covariances corresponding to the model and observations errors at relatively higher levels. The model error covariance is fixed as $\boldsymbol{Q} = 1.0 \times 10^1 \boldsymbol{I}_{N_s}$ and the observation covariance is fixed as $\boldsymbol{R} = 1.0 \boldsymbol{I}_{N_o}$. The ensemble size ($N_e = 250$) and the parameter propagation covariance ($\boldsymbol{C} = 1.0 \boldsymbol{I}_{N_p}$) are assigned to the same values as in the previous case. The evolution of the parameters $\boldsymbol{\theta}_s$ during the course of DA is shown in Fig. 5.30. Once again, the analyzed parameters vector in the assimilation window is found to converge to a fixed value. This convergence is accompanied with a reduction of the ensemble variance.

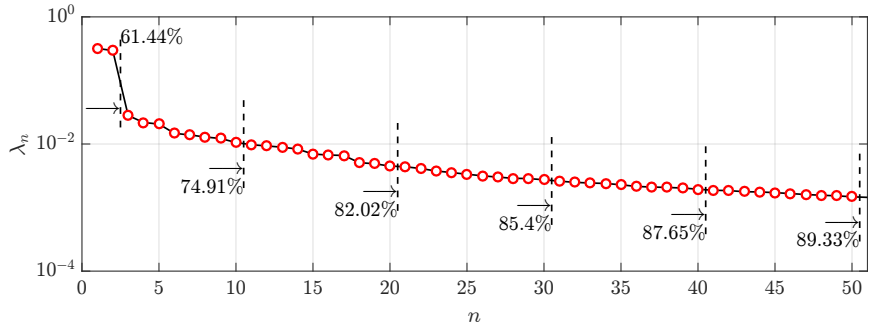


Figure 5.28: POD energy content of the most energetic modes for the cylinder wake flow at $Re = 5.5 \times 10^4$. Eigenvalue spectrum of the POD modes (λ_n). The values of $RIC(n)$ are also indicated, representing the fraction of the total fluctuating kinetic energy captured if only the POD modes up to the dashed lines are considered.

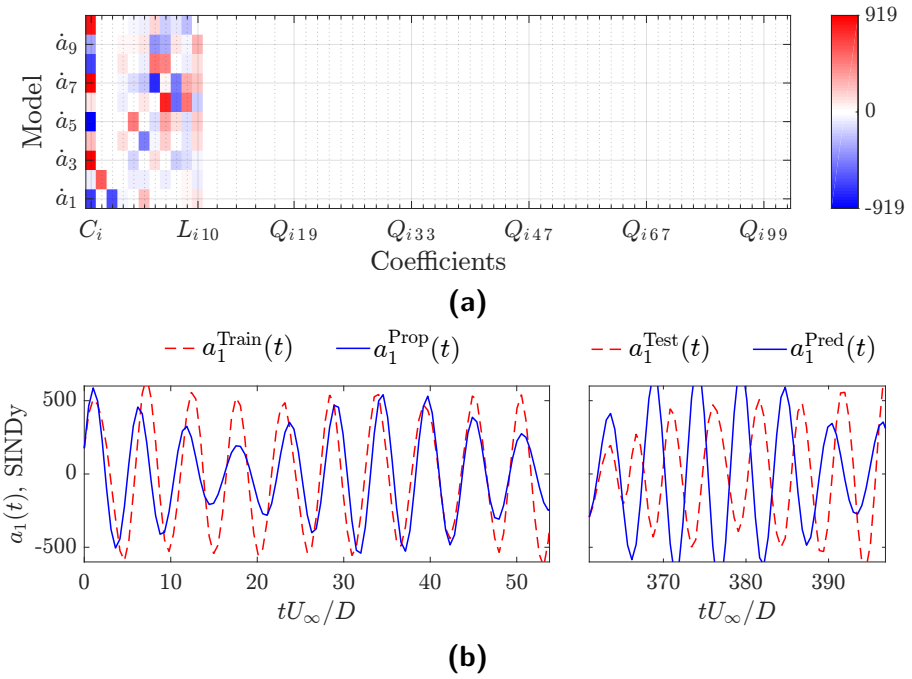


Figure 5.29: (a) Scaled color representation of the parameters identified using SINDy for the POD-ROM of a cylinder wake flow at $Re = 5.5 \times 10^4$. (b) Evolution of coefficients $a_i(t)$ ($i = 1, 5, 9$) in the training regime (left) and in the testing regime (right). True (dashed line) and estimated (solid line) trajectories.

At the end of the assimilation window, the parameters vector is found to take values $[\nu_c \ \alpha] = [0.26 \ 2.58]$.

Estimation of the temporal coefficients over a long-time horizon is then undertaken with a simple EnKF by following the same assimilation arrangement than in the previous case (see Fig. 5.24 for the probes configuration). Following the same procedure as in the previous section, the new observations are assimilated in the test window at time steps spanning about two cycles of vortex shedding, giving an assimilation step size of $\Delta t_o U_\infty/D = 10.8$, *i.e.* 21 times the acquisition step size, $\Delta t U_\infty/D = 0.51$. The

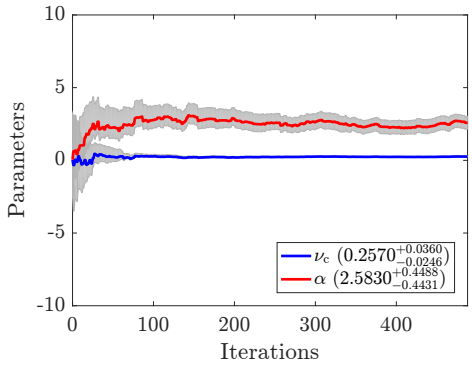


Figure 5.30: Evolution of the stabilizing parameters $\boldsymbol{\theta}_s = [\nu_c \ \alpha]^\top$ of the POD-ROM in the assimilation window for the modal constant closure. Refer to the caption of Fig. 5.22 for a detailed description.

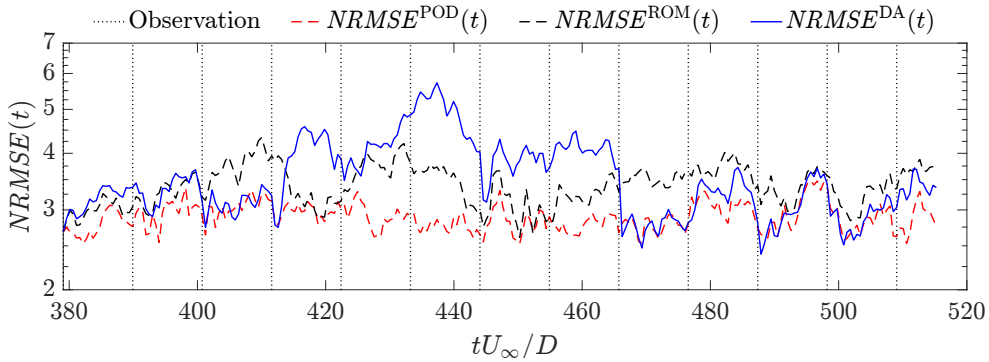
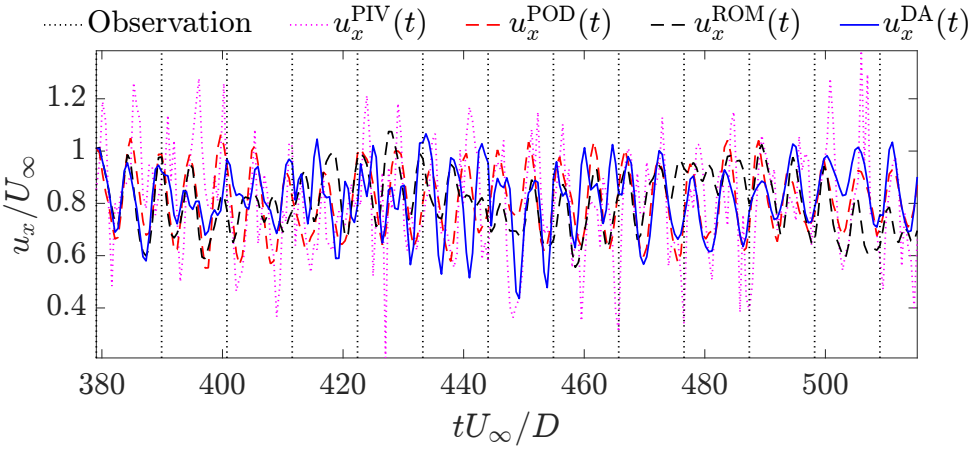
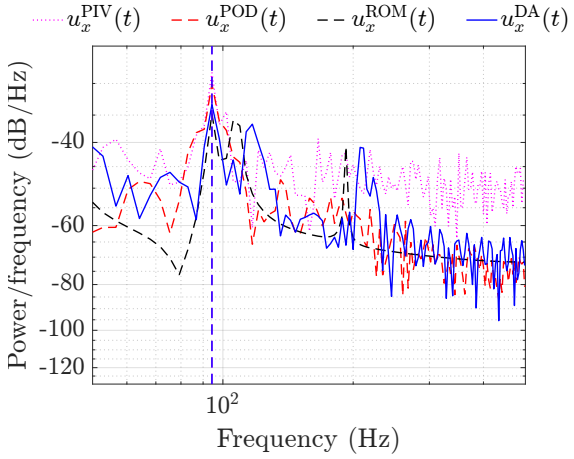


Figure 5.31: Time evolution of the error $NRMSE$ of the vorticity fields with respect to the vorticity calculated from the PIV measurements.

$NRMSE$ shown in Fig. 5.31, the time evolution of the estimated streamwise velocity component at farfield location in the wake represented in Fig. 5.32a, and the corresponding frequency spectra plotted in Fig. 5.32b, indicate that the overall methodology holds for this higher value of Reynolds number. In comparison with the error from the initial ROM, we observe in Fig. 5.31 a higher error for the stabilized and assimilated ROM in the middle span of the testing window. This indicates a limitation of the ROM to provide accurate solutions for an arbitrary initial conditions, causing a deviation from the reference trajectory (indicated by an increase in error at $tU_\infty/D = 415$). However, the continued assimilation of new observations in the test window corrects the solution trajectory. We note that the reference trajectory is accurately reproduced in the long-term (from $tU_\infty/D = 465$ onwards). For the rest, similar conclusions as that drawn for the lower Reynolds number can be made here. Improvement in the accuracy of the estimated state is observed when observations are assimilated regularly thanks to the EnKF. In terms of the frequency content, the dynamics of the original velocity field is well replicated without the loss of phase information. At this stage, a remaining question is the ability of the methodology to accurately predict a flow state when the dynamics requires a larger number of modes (hence a larger number of model parameters) to be kept in the construction of the reduced-order model. This point is addressed in the next section where the case of a turbulent jet is considered.



(a)



(b)

Figure 5.32: (a) Time evolution of the streamwise velocity component at a location $(x/D, y/D) = (4.8, 0.9)$ in the flow field. Comparisons of the velocity obtained from the PIV measurements ($u_x^{\mathrm{PIV}}(t)$) and different reconstructions using the POD coefficients ($u_x^{\mathrm{POD}}(t)$), the initial ROM ($u_x^{\mathrm{ROM}}(t)$) and the stabilized and assimilated ROM ($u_x^{\mathrm{DA}}(t)$). (b) Power spectrum densities of the velocity measured using PIV and the reconstructed velocity signals using the POD coefficients and the initial and stabilized ROMs. The peak frequencies of the measured signal and that reconstructed from the stabilized ROM (which coincide here) are indicated by dashed lines.

5.5 Test case 4: Numerical Mach 0.9 turbulent jet

In this section, the case of a Mach 0.9 single-stream jet at Reynolds number 4.0×10^5 is considered. The data was obtained from Large Eddy Simulation (LES) by Bogey and Bailly (2006). Building on previous efforts to investigate noise mechanisms in turbulent jets, Kerhervé et al. (2012) proposed a methodology to educe, from the overall turbulent flow field, the sound-producing events associated with low-angle radiation. In the present work, the Dual-EnKF is employed as a strategy to build a reduced-order dynamical model for this flow configuration. For convenience, the coefficients of the POD-ROM are identified thanks to a regularized L_2 minimization procedure. A twenty degree-of-freedom ROM is established as a first start. The identification scheme is found to lead to a bounded ROM, but with estimated trajectories drifting from the true one. A data-driven correction based on the Dual-EnKF is then applied to calibrate an empirical non-linear eddy viscosity closure model added to the initial POD-ROM with the objective to achieve "long-term" prediction of the acoustically-important flow motions. Details of the methodology and the obtained results are extensively detailed in the next paper. The results show that the assimilation strategy based on the Dual-EnKF is able to properly replicate over a long-time horizon the dynamics of the original data thanks to a bounded reduced-order model and observations collected over time. In this last test case, the observation data were designed to replicate experimental conditions.

A data-driven low-order model of sound source dynamics in a turbulent jet

N. Kumar [1] and F. Kerhervé [2] and L. Cordier [3]

Abstract

Building on previous efforts dedicated to the eduction of flow motions associated with low-angle sound emission from turbulent jets [25], the current paper proposes a strategy for the reduced-order dynamical modelling of these motions. The technique is data-driven and comprises (i) a dynamic *Ansatz* obtained by Galerkin projection of the Navier-Stokes equations onto a set of POD orthonormal spatial basis functions coupled with (ii) a dual-Kalman filter which allows long term prediction of the state and in line correction of the model parameters. Instead of directly computing the coefficients of the model equation driving the flow state, which is only possible when full flow information is available, the coefficients are first identified thanks to a regularised L_2 minimisation procedure. A twenty degree-of-freedom reduced-order dynamical model (ROM) is then established. The identification scheme is found to produce bounded ROMs but with estimated trajectories drifting from the true ones. A data-driven correction based on a dual-ensemble Kalman filter is therefore implemented and an empirical non-linear eddy viscosity model is added to the ROM to solve its inherent lack of stability. This correction allows the estimated state to be corrected "on-line" when a measurement of the true state (here point velocity and far-field pressure) is available, as well as to find the optimal set of parameters associated with the eddy viscosity term. The described approach, though validated here using partial flow information from a numerical simulation, has been designed from the perspective of implementation in an experimental context.

1. Introduction

Description of turbulent flows using stochastic distribution of eddies has long been served as a paradigm [27]. Since then, further experimental evidences of the role played by well-coherent large structures in the dynamics have considerably changed this point of view [15]. As far as jet flows are concerned, evidences that part of the noise radiated is essentially driven by these large-scale well-coherent structures has led to a significant progress in the understanding of noise generation and new methodologies for noise prediction have been derived. Following the original works of Michalke [30], Jordan and co-workers [21, 10] have proposed wavepackets *Ansatz* as the main source of jet noise. These entities may be seen as a low-order description of jet flows. More generally, assumption

[1]Institut PPRIME CNRS UPR 3346, Université de Poitiers, ENSMA (France)
[2]Institut PPRIME CNRS UPR 3346, Université de Poitiers, ENSMA (France)
[3]Institut PPRIME CNRS UPR 3346, Université de Poitiers, ENSMA (France)

that turbulent flows can thus be meaningfully reduced to simplified kinematics pushes forward for low-dimensional modelling of their dynamics. However, where radiated noise is of concern, it is essential that the low-dimensional model also satisfies the inhomogeneous linear wave equation.

One of the key objectives of low-dimensional modelling is to obtain a dynamical model with as few degrees of freedom as possible. Generally, this is also motivated by the need to perform real-time sensing and control. The number of degrees of freedom is dictated by the number of basis functions deemed necessary for an accurate estimation of the dynamics. This raises a question regarding the selection of the basis functions that are best suited to a given problem. The empirical eigenfunctions obtained by Proper Orthogonal Decomposition (hereafter POD) are frequently used, largely on account of the optimality property of this kind of decomposition in the sense of energy, and will also be used here. However, rather than applying it to the full flow solution, here it is applied to the reduced complexity sound-producing flow skeleton identified in a previous work [25]. This last work showed that the number of modes necessary to capture a given percentage of fluctuation energy of the acoustically-important coherent structures is an order of magnitude less than the number necessary to capture the same percentage of fluctuation energy of the full field. The POD modes identified can thus be considered as acoustically-optimised and are somewhat equivalent to the Most Observable Decomposition modes of Jordan et al. [22]. While the kinematic features of the sound-producing flow skeleton identified were fully described in Kerhervé et al. [25], here we seek a low-dimensional dynamical system driving the associated motions.The conventional Galerkin projection method is employed here. Projection of the Navier-Stokes equations on a truncated POD basis results in a Galerkin system which consists in a low-dimensional system of ordinary differential equations (ODE) with a tractable limited rank. In the present case, the corresponding reduced-order model (POD-ROM) is built with the first twenty POD modes. The linear combination of these modes was found to reproduce the dynamics of the essential sound-producing flow motions. The main difficulty resides in the calibration of the POD-ROM itself. By construction, this methodology is known to result in unstable models; after a relatively short time horizon, the estimated state drifts from the true trajectory and corrections are required to guarantee not only the stability of the POD-ROM but also the requirement that the estimated state converges towards the true value. Keeping only the lower POD modes in the truncation is equivalent to conserving only the large-scale coherent structures associated with the production of turbulent kinetic energy while omitting the small-scale fluctuations associated with dissipation. The dissipation loss due to mode truncation is generally considered as the main contributor to the lack of stability of the POD-ROM which therefore needs to be amended. Common approaches fall into two classes of strategies. The first class is closure model dependent and derives from the pioneering work of Aubry et al. [1] who proposed to add dissipation terms to the POD-ROM in the form of an empirical "eddy viscosity". Strategies based on eddy viscosity have since been refined by numerous authors considering either linear [36, 18, 38] or non-linear [14, 34, 37] additional terms. As the Reynolds number increases, non-linear subscale turbulence models are found mandatory. In such cases, Östh et al. [34] recently showed that only non-linear modal eddy viscosity can guarantee stabilisation and robustness of the POD-ROM. Spectral eddy viscosity mod-

els, inspired by LES models, have also been proposed [8, 32, 39]. Energy conservation concept has also been used to derived closure modelling strategies. This has been initially proposed by Cazemier et al. [11] and do not require the specification of any free parameter. The second class of strategies is closure model free. The most recent strategies are formulated as constrained optimisation problem resulting in POD-ROM where part of the small scales are included [2]. In contrast to the first class, this type of strategy has the benefit that it guarantees power balance in the low-dimensional space and does not require a subgrid-turbulence model.

The lesson from several of these studies is that while stability of the estimated POD-ROM can be maintained, accurate tracking over long time horizon is generally not achievable despite large efforts in the calibration procedure. In the present case, the calibration technique of Perret et al. [35] combined with a Tikhonov regularisation [19], such as the one proposed by Cordier et al. [13], is considered to obtain a stable dynamical system in the first step. To improve the predictability of the POD-ROM over long time horizons, different strategies are then presented, among which a sequential data-assimilation technique known as dual-ensemble Kalman filter (Dual-EnKF) is considered here. Thanks to the advancements in computational capabilities, Kalman filters have regained new interest in the fluid mechanics community and have been used to develop data-driven simulations [24, 41] or again to solve data-driven estimation problems [26]. Kalman filters have the ability to improve the state estimation by assimilating available informations. Since in the present paper the interest is essentially focused on low-angle sound emission, the acoustic pressure being radiated downstream will be considered as observation.

The paper is organised as follows. In §2, the flow configuration and the main elements of the database are described. Elaboration of a bounded reduced-order dynamical model is then presented in §3 and its limitation to only provide short-term predictions is discussed in §4. Strategies for long-term prediction are presented in detail in §5. Main characteristics of the reconstructed spatio-temporal flow field dynamics associated with low angle radiation over long a time horizon is addressed in §6. Conclusions and perspectives finally closes the paper in §7.

2. Flow configuration and previous work

The flow under investigation is a Mach 0.9 single-stream jet with a Reynolds number of 4×10^5, based on the jet diameter D and exit velocity U, obtained from Large Eddy Simulation (LES) by Bogey and Bailly [6]. Details of the simulation as well as validation of the flow and sound properties can be found in Bogey et al. [7] and Bogey and Bailly [5, 6]. As mentioned in the introduction, here we focus on the flow dynamics responsible for the noise being radiated downstream. In the work of Kerhervé et al. [25], a methodology has been developed in order to educe the sound-producing events from the overall flow field. The core idea is to use the radiated pressure field to filter out flow mechanisms not involved in noise production. Linear Stochastic Estimation (hereafter LSE) was used to reconstruct a time-resolved conditional space-time flow field associated with the low-angle sound emission extracted from the overall radiated pressure field, thanks to an angle-dependent wavenumber-frequency filter. Figure 1(a) shows the LES solution, while figure 1(b) shows, for the same time step: in zone Ω_A, the low-angle component of the sound field,

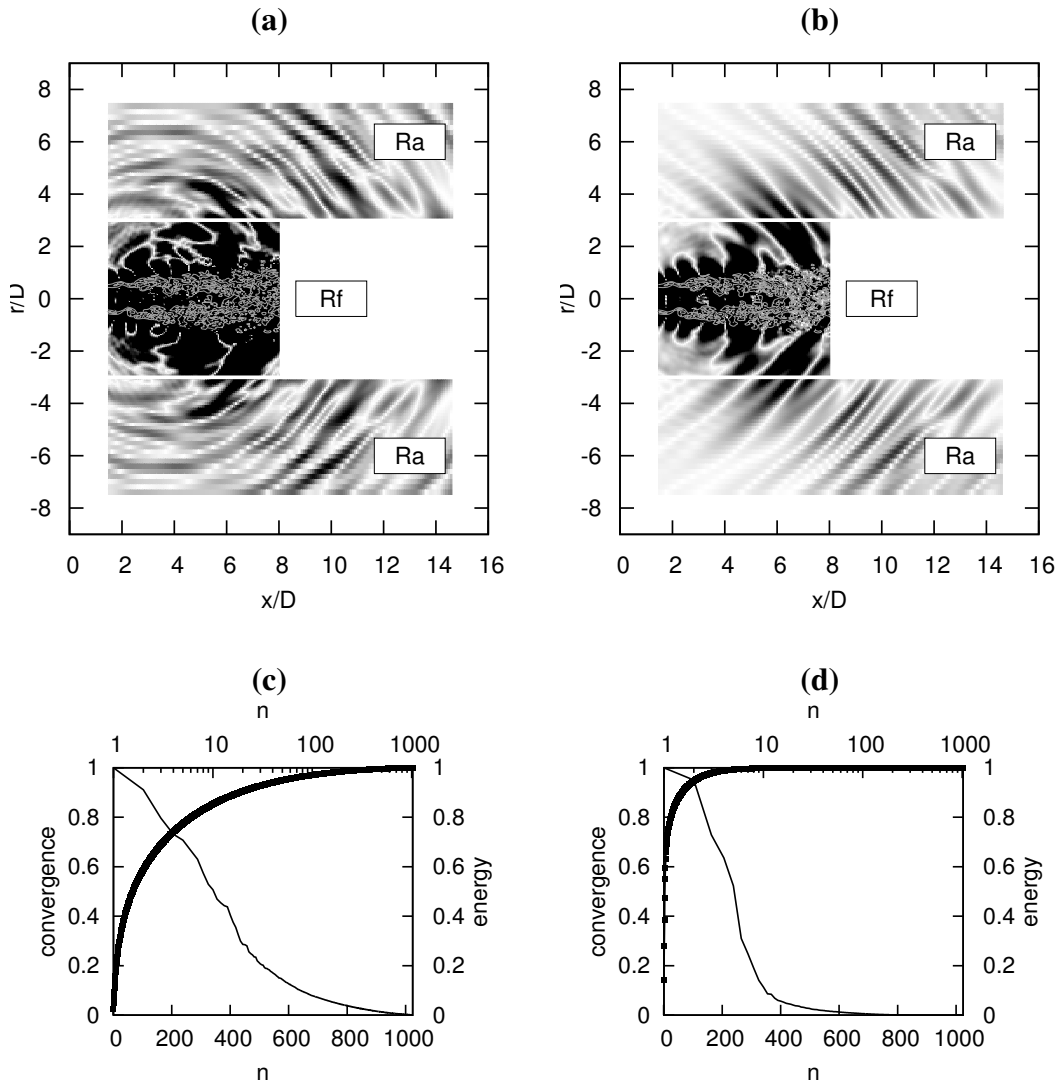


Figure 1: (a) Snapshot of vorticity and pressure fields of the LES solution. (b) Zone Ω_A: snapshot of the low-angle component of the LES sound field (obtained using a frequency-wavenumber filter); zone Ω_F: snapshot of pressure field associated with the sound-producing flow skeleton (this is a conditional field, computed by LSE from the information in Ω_A). (c) POD eigenspectrum (open symbols) and its convergence (solid symbols) of LES velocity fields corresponding to the sub-region Ω_F^{rot} of Ω_F (see (a) and the zoom given in figure 2). (d) POD eigenspectrum (open symbols) and its convergence (solid symbols) of the conditional field presented in (b).

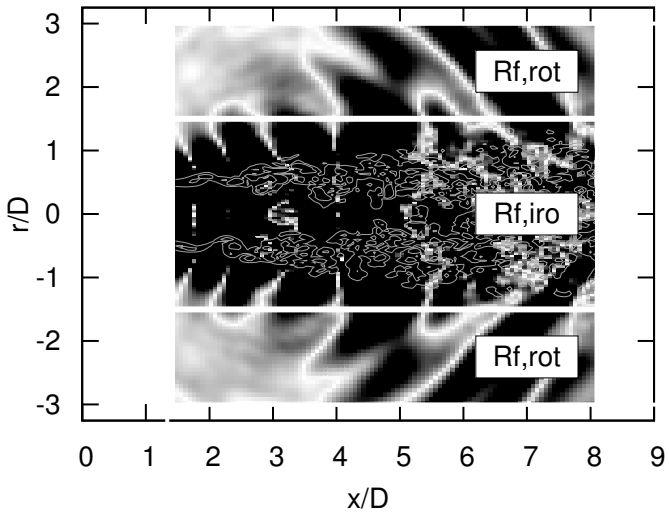


Figure 2: Zoomed view of region Ω_F, which is further broken down into rotational and irrotational regions, respectively Ω_F^{rot} and Ω_F^{irot}.

and, in zone Ω_F, the associated conditional pressure field (a conditional velocity field, not shown, is also computed).

3. Elaboration of the dynamical system

3.1. Model equation

A reduced-order model describing the time evolution of the conditional field $\hat{\mathbf{q}}$, which can be considered here as the high-fidelity system, is sought. The Petrov-Galerkin projection method is implemented with POD modes as the basis of projection and the Navier-Stokes equations as the high-fidelity model equation. This results in a system of ordinary differential equations (ODE) with finite dimension, which is in contrast to the Navier-Stokes equations. Unfortunately, these models are known to be intrinsically unstable, mainly due to mode truncation. For this reason, the calibration method elaborated by [13] is implemented here and is described in the following sections.

3.1.1. Proper Orthogonal Decomposition

Let $\{\boldsymbol{u}(\boldsymbol{x}, t_k)\}_{k=1,\cdots,N_s}$ be a set of N_s velocity snapshots equally spaced over a time interval T_s with $\boldsymbol{x} \in \Omega_F^{\text{rot}}$. The velocity field can be decomposed into an orthonormal basis consisting of spatial basis functions and temporal coefficients $\{\boldsymbol{\Phi}_i^p(\boldsymbol{x}), a_i^p(t)\}_{i=1,\cdots,N_s}$, such that the normalised mean-square projection of the basis functions on the velocity field is maximised [29] and is given by,

$$\boldsymbol{u}(\boldsymbol{x}, t) = \sum_{i=1}^{N_s} a_i^p(t)\boldsymbol{\Phi}_i^p(\boldsymbol{x}). \tag{1}$$

When the "snapshot POD" method [40] is implemented, the maximisation problem leads

to an eigenvalue problem which reads as,

$$\mathcal{C}\boldsymbol{a}_i^p = \lambda_i \boldsymbol{a}_i^P, \tag{2}$$

where the superscript ^{p} denotes temporal POD coefficients obtained from the high-fidelity system (represented by the N_s velocity snapshots) and $\mathcal{C} \in \mathbb{R}^{N_s \times N_s}$ denotes a correlation matrix, averaged over the spatial direction, whose elements can be expressed as,

$$\mathcal{C}_{ij} = \frac{1}{N_s} \left\langle \boldsymbol{u}(\boldsymbol{x}, t_i), \boldsymbol{u}(\boldsymbol{x}, t_j) \right\rangle_{\Omega_F^{rot}} = \sum_{k=1}^{n_c} \iint_{\Omega_F^{rot}} u_k(\boldsymbol{x}, t_i) u_k(\boldsymbol{x}, t_j) d\boldsymbol{x}, \tag{3}$$

with n_c as the number of velocity components considered (here $n_c = 2$). In practice, when the eigenvalue problem (eq. 2) is solved, the calculated temporal POD coefficients are normalised so that the variance of the coefficients corresponding to each mode is equal to unity. Formally, these coefficients should be properly rescaled so that their variance is equal to the corresponding eigenvalue. However, in what follows, this fully suits the numerical constraints of linear regression problem.

In the following, the objective is to build a reduced-order model providing long-term prediction. The overall database includes a total of $N_T = 19000$ snapshots sampled at a Strouhal number, based on the jet diameter, of $St_D = 3.9$ (corresponding to a total duration of $tU/D = 4900$). Here, the snapshot POD method, recalled previously, is applied to a set of $N_s = 4096$ snapshots which has been randomly sampled from the full database. Solving Eq.(2) results in N_s temporal POD modes, randomly sampled over time, which, when projected onto the initial randomly selected snapshots lead to N_s spatial eigenfunctions $\boldsymbol{\Phi}_i(\boldsymbol{x})$. In order to build a ROM and to compare the estimated temporal POD modes over the full duration of the database, the eigenfunctions are reprojected into $u(\boldsymbol{x}, t)$ to obtain a set of N_s temporal POD functions sampled at the same rate as the velocity snapshots.

The POD eigenspectra obtained for the LES field and the conditional acoustically-filtered velocity field are reported in Figure 1(c and d). While 20 modes suffice to recover 80% of the fluctuation energy for the conditional field, over 200 are required to retain the same amount of energy for the LES field. As discussed in detail in Kerhervé et al. [25], the acoustically-filtered field is found to comprise considerably lower degrees of freedom which suggests that a low-rank model can be found to properly estimate its dynamics.

3.1.2. POD reduced-order model based on Galerkin projection

A Galerkin projection of the Navier-Stokes equations onto the POD basis defined previously is effected next. After manipulations, this leads to a set of ordinary differential equations (ODEs) which reads as [20, 4, 13],

$$\dot{a}_i^P(t) = D_i + \sum_{j=1}^{N_{\text{gal}}} L_{ij} a_j^P(t) + \sum_{j=1}^{N_{\text{gal}}} \sum_{k=1}^{N_{\text{gal}}} Q_{ijk} a_j^P(t) a_k^P(t) + \mathcal{R}_i(t) \quad i = 1, \cdots, N_{\text{gal}}, \tag{4}$$

where $\boldsymbol{y}_i = \left(D_i, L_{i1}, \cdots, L_{iN_{\text{gal}}}, Q_{i11}, \cdots, Q_{iN_{\text{gal}}N_{\text{gal}}}\right)^\top \in \mathbb{R}^{N_{\boldsymbol{y}_i}}$ denotes a vector of unknown real-valued coefficients, $\dot{q}$ denotes the time derivative of q and $\mathcal{R}_i$ denotes the resid-

ual term which is read as,

$$\mathcal{R}_i(t) = \sum_{j=N_{\text{gal}}+1}^{N_{\text{s}}} L_{ij} a_j^P(t) + \sum_{\substack{j=1 \\ max(j,k)>N_{\text{gal}}+1}}^{N_{\text{s}}} \sum_{k=1}^{N_{\text{s}}} Q_{ijk} a_j^P(t) a_k^P(t). \tag{5}$$

As a first approximation, the case with $\mathcal{R}_i = 0$ is considered, which is equivalent to neglecting the convective and viscous terms associated with the unresolved modes $i > N_{\text{gal}}$. The POD-reduced order model can consequently be written as,

$$\dot{a}_i^R(t) = D_i + \sum_{j=1}^{N_{\text{gal}}} L_{ij} a_j^R(t) + \sum_{j=1}^{N_{\text{gal}}} \sum_{k=j}^{N_{\text{gal}}} Q_{ijk} a_j^R(t) a_k^R(t) \quad i = 1, \cdots, N_{\text{gal}}, \tag{6}$$

where $a_i^R(t)$ denotes the i-th estimated temporal coefficient.

The coefficients D_i, L_{ij} and Q_{ijk} can be computed directly if the full flow information is available, for instance from a sufficiently resolved numerical simulation. However, in situations where only incomplete flow information is available, such as in an experiment or in the case of an incomplete numerical data, the coefficients must be determined by alternative means. This is discussed in §3.2.

Introducing,

$$\boldsymbol{a}^R(t) = \left(a_1^R(t), \cdots, a_{N_{\text{gal}}}^R(t)\right)^\top,$$

Eq.(4) can be rewritten as,

$$\dot{a}_i^R(t) = f_i(\boldsymbol{a}^R(t), \boldsymbol{y}_i), \tag{7}$$

where f_i is the model equation for the i-th mode. The dynamical system (7) can be written in a more compact form,

$$\dot{\boldsymbol{a}}^R(t) = \boldsymbol{f}\left(\boldsymbol{a}^R(t), \boldsymbol{y}\right), \tag{8}$$

where $\boldsymbol{f} = \left(f_1, \cdots, f_{N_{\text{gal}}}\right)^\top \in \mathbb{R}^{N_{\text{gal}}}$ and $\boldsymbol{y} = (\boldsymbol{y}_1, \cdots, \boldsymbol{y}_{N_{\text{gal}}})^\top \in \mathbb{R}^{N_{\boldsymbol{y}}}$ is the full set of unknown coefficients, with $N_{\boldsymbol{y}} = N_{\text{gal}} N_{\boldsymbol{y}_i}$.

In the following sections, this compact form is conserved for a discussion on the identification procedure of the coefficients vector $\boldsymbol{y}_i$.

3.2. Model identification

A number of studies have been performed with regards to the identification of model coefficients. These methods differ based on the definition of the error to be minimised. An intuitive choice is the error with respect to the time derivative of the temporal coefficients since the objective of the reduced-order model is to predict as accurately as possible the dynamics of the system. This has been initially proposed by Perret et al. [35] and revisited by Cordier et al. [13] or more recently by Suzuki [42]. Here, the flow calibration method of Cordier et al. [13] is presented.

Let $\boldsymbol{e}(\boldsymbol{y}, t)$ be the error function defined by,

$$\boldsymbol{e}(\boldsymbol{y}, t) = \dot{\boldsymbol{a}}^P(t) - \boldsymbol{f}(\boldsymbol{a}^P(t), \boldsymbol{y}), \tag{9}$$

The flow calibration procedure consists in minimising the cost functional $\mathcal{I}(\boldsymbol{y}) = \langle ||\boldsymbol{e}(\boldsymbol{y}, t)||_2^2 \rangle_{T_s}$, where $||\cdot||_2$ denotes the L_2 norm, and $\langle\cdot\rangle_{T_s}$ a time-average operator defined as

$$< q(t) >_{T_s} = \frac{1}{N} \sum_{k=1}^{N} q(t_k), \tag{10}$$

where $q(t_i)$ denotes N discrete values of q equally distributed over the time period $[0, T_s]$. Equivalently, the functional $\mathcal{I}$ can then be written as,

$$\mathcal{I}(\boldsymbol{y}) = \frac{1}{N} \sum_{k=1}^{N} \sum_{i=1}^{N_{\text{gal}}} \left[\dot{a}_i^P(t_k) - f_i(a_i^P(t_k), \boldsymbol{y}) \right]^2. \tag{11}$$

Minimisation of $\mathcal{I}$ leads to the following linear system [13],

$$A\boldsymbol{y} = \boldsymbol{b} \quad \text{with} \quad A \in \mathbb{R}^{N_{\boldsymbol{y}} \times N_{\boldsymbol{y}}} \quad \text{and} \quad \boldsymbol{b} \in \mathbb{R}^{N_{\boldsymbol{y}}}, \tag{12}$$

where,

$$A = \left\langle E^\top(t)\, E(t) \right\rangle_{T_s} \quad \text{and} \quad \boldsymbol{b} = -\left\langle E^\top(t) \boldsymbol{e}(\boldsymbol{0}, t) \right\rangle_{T_s}. \tag{13}$$

The time-dependent array, $E(t) \in \mathbb{R}^{N_{\text{gal}} \times N_{\boldsymbol{y}}}$, is defined as

$$E(t)\boldsymbol{y} = -\boldsymbol{f}(\boldsymbol{a}^P(t), \boldsymbol{y}). \tag{14}$$

The minimisation procedure is therefore equivalent to solving the linear system given in Eq.(12), where the unknown vector contains the coefficients of the dynamical system. Note that this procedure is similar to that proposed by Perret et al. [35] except that here the coefficients of the entire ODE system are calibrated in an one-shot procedure and is computationally more efficient.

3.3. Regularisation of the solution

The linear system Eq.(12) is generally ill-conditioned. It can be easily inferred since the term $\boldsymbol{b}$ on the right-hand side of the equation may be contaminated through $\boldsymbol{e}(\boldsymbol{0}, t) = \dot{\boldsymbol{a}}^P(t)$ due to approximation errors associated with the numerical evaluation of time-derivatives of the temporal POD coefficients. Here, a 2nd-order centered finite-difference scheme is used for estimating these time derivatives.

As a remedy to solve the potentially ill-conditioned system, a regularisation procedure is therefore needed. In this work, a Tikhonov regularisation of zero-th order is implemented [19] . The basic idea of Tikhonov regularisation is to find a solution $\boldsymbol{y}_\rho$ of the linear system Eq.(12) that minimises the residual $||A\boldsymbol{y}_\rho - \boldsymbol{b}||_2$ without penalising too much the value of $||\boldsymbol{y}_\rho||_2$. This is equivalent to minimising the following functional,

$$\phi_\rho(\boldsymbol{y}) = ||A\boldsymbol{y} - \boldsymbol{b}||^2 + \rho||\boldsymbol{y}||_2^2, \tag{15}$$

where ρ is a regularisation parameter to be determined.

To understand the influence of an ill-conditioned Eq.(12) on the solution of the linear system, the concept of filter factors is now introduced. To do so, the singular value decomposition of A is considered, and can be read as,

$$A = U\Sigma V^\top = \sum_{i=1}^{N_{\boldsymbol{y}}} \boldsymbol{u}_i \sigma_i \boldsymbol{v}_i^\top, \tag{16}$$

where U and V are orthogonal matrices containing left, $\boldsymbol{u}_i$, and right, $\boldsymbol{v}_i$, singular vectors, and σ_i are the singular values of A arranged in decreasing order. The solution for $\boldsymbol{y}$ can thus be written as

$$\boldsymbol{y} = \sum_{i=1}^{N_{\boldsymbol{y}}} h_i \frac{1}{\sigma_i} \boldsymbol{u_i}^\top \boldsymbol{b} \boldsymbol{v_i} \quad \text{with} \quad h_i = 1 \quad \text{for} \quad i = 1, \cdots, N_{\boldsymbol{y}}. \tag{17}$$

In this expression, some additional factors, h_i, have been artificially introduced. When these factors are fixed to unity, $|\boldsymbol{u_i}^\top \boldsymbol{b}|$ may not decrease sufficiently fast as compared to $|\boldsymbol{u_i}^\top \boldsymbol{b}/\sigma_i|$ when σ_i becomes small. This can result in a solution with a large Euclidean norm [19, 13]. To minimise the contribution of modes i associated with small σ_i in the summation defined in Eq.(17), these components are "low-pass filtered" by modifying the value of h_i. Given a value of ρ, the solution of the linear system can be expressed by Eq.(17) with filter factors equal to,

$$h_i = \frac{\sigma_i^2}{\sigma_i^2 + \rho^2}. \tag{18}$$

Different strategies may be used to determine the parameter ρ. In the present case, the L-curve method described in Hansen [19] is used. This consists in an iterative procedure to identify $\boldsymbol{y}_\rho = \arg\min_{\boldsymbol{y}} \phi_\rho$ which balances the values of the two residuals $||A\boldsymbol{y_\rho} - \boldsymbol{b}||_2^2$ and $||\boldsymbol{y_\rho}||_2^2$.

4. Bounded reduced-order model

4.1. Minimisation of cost functional $\mathcal{I}$

As discussed previously, identification of the dynamical model given by Eq.(8) is equivalent to solving the linear system Eq.(12). The matrix A and vector $\boldsymbol{b}$ are evaluated according to Eq.(13). Performance of the minimisation procedure is evaluated in figure 3 where the time derivative $\dot{a}_i^P$ of the POD temporal coefficients in comparison with that obtained by the identified dynamical model $\boldsymbol{f}(\boldsymbol{a}^P(t), \boldsymbol{y})$ for the first and fourth POD modes are reported. Identification of $\boldsymbol{f}$ is in good agreement with the original data and, as expected, very similar results are observed without and with application of the Tikhonov regularisation. However, as we will see in the next section, the very small differences that do exist between the two estimates have important consequences when the dynamical system is integrated in time.

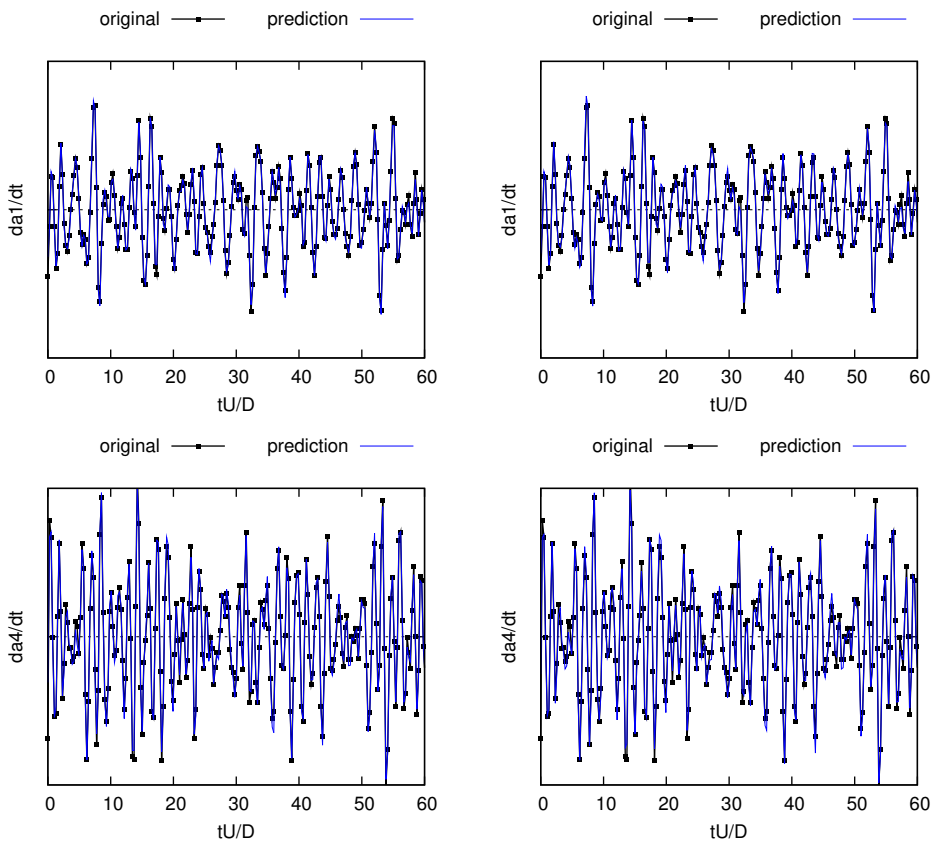


Figure 3: Visual evaluation of error function $\boldsymbol{e}(\boldsymbol{y}, t)$: comparison between (line with dots) time derivative of POD temporal coefficient $\dot{a}_i^P(t)$ and (blue line) identified model $f_i(\boldsymbol{a}^P(t), \boldsymbol{y})$ for (top) mode 1 and (bottom) mode 4. (a) Without and (b) with Tikhonov regularisation.

4.2. Short-term prediction

Once the coefficients D_i, L_{ij} and Q_{ijk} have been determined, temporal integration of the dynamical system Eq.(4) is performed using a 4-th order Runge-Kutta scheme. The initial condition at $t = 0$ corresponds to the initial sample of the data sequence considered for the POD analysis. The time step used for the integration is equal to the time resolution of the data described in §3.

Two solutions are obtained: one where the coefficients of the model are calibrated using Tikhonov regularisation, and a second where no regularisation is used. Time histories of the first temporal POD modes corresponding to both the solutions are shown in figure 4. The dynamical system obtained without regularisation is unstable and diverges after 20 time units, whereas the other is stable. Both systems nonetheless provide good predictions beyond a short time horizon. The stable system begins to drift from the trajectory described by the original data after about six convective time units as illustrated in figures 4(c and d). This result is not surprising as discussed in the introduction and reported by many others. In the following sections, different strategies for long-term predictions will thus be considered.

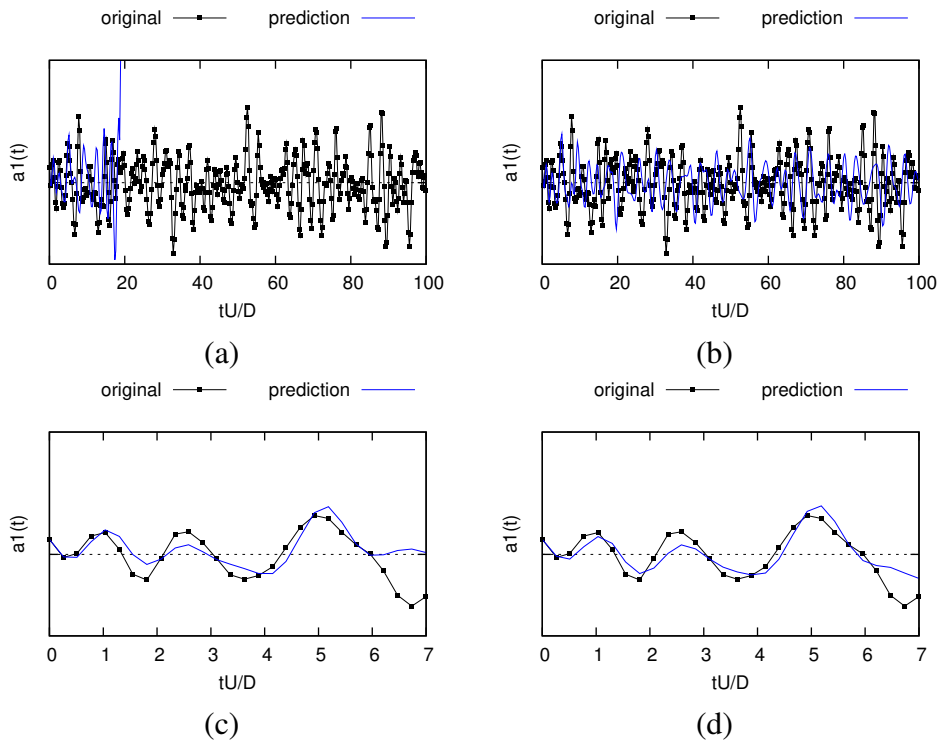


Figure 4: Time histories of the first POD temporal coefficients obtained directly from POD (line with dots) and predicted by the dynamical model (blue line) without (a,c) and with (b,d) Tikhonov regularisation. (top) Time histories over 100 time units and (bottom) zoomed view over the first 7 time units.

5. Strategies for long-term predictions

5.1. Dependence of the initial condition and intrinsic stabilisation

It is of practical interest to establish if the identified dynamical system can provide correct predictions when the time integration is started at different points on the POD attractor. As illustrated in figure 5, which shows the trajectories of estimated temporal coefficients $a_1(t)$ obtained from initiating the time integration at different time instances with the true value as initial condition, the dynamical model performs equally well. Furthermore, when an initial condition which was not contained in the data set for the identification procedure is used, as it is the case in figure 5(d), the model continues to provide correct prediction over the short time horizon. This is of considerable importance because it implies that the model coefficients identified from a given set of realisations (learning sequence) can be re-employed to predict another sequence of realisations. Note that this remains valid only if the matrix A and vector $\boldsymbol{b}$ continue to statistically well represent the new realisations. In the present case, the results are an indication that A and $\boldsymbol{b}$ are sufficiently well estimated and that a longer data sequence for learning would not have been necessary.

Since a correct prediction can be obtained for a short time duration by assimilating a new initial condition, a basic procedure for long-term prediction is now evaluated. This procedure consists in injecting new initial conditions every 6 time units. This procedure is somewhat similar to the intrinsic stabilisation scheme proposed by Kalb and Deane [23]. The results are shown in figure 6 for the first POD temporal coefficient (similar results

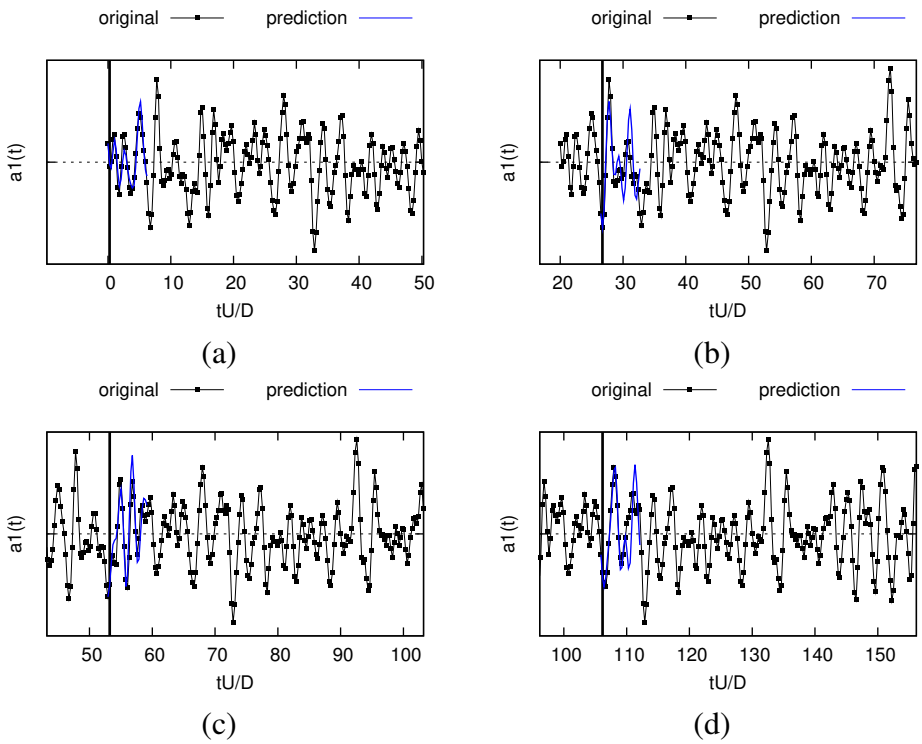


Figure 5: Time histories of the first POD temporal coefficients (line with dots) and its prediction (blue line) obtained from different initial conditions.

are obtained for the higher modes). As expected , the long-term prediction is found to be satisfactory. This procedure is however of limited practical interest. It requires temporal POD coefficients to be estimated in order to provide a new initial condition. This can only be obtained if velocity snapshots at the corresponding instants are available from measurements. Also, measurement noise or, again, incoming perturbation may deteriorate the prediction since the model is not designed to take into account such deviations from the original trajectory. In the next section, we thus introduce additional correction to the model in perspective of long-term prediction to take into account for such deviations.

5.2. Long-term prediction with data assimilation

Despite efforts to enable the calibration procedure to provide accurate estimates, the low-order dynamical systems inherently drift from the true trajectory when integrated in time if no correction is applied. As already mentioned, the model equation does not drive the examined real flow system perfectly. Injecting periodically a new initial condition, as performed previously, is the zero-th step of more advanced data assimilation technique. Suzuki [42] recently proposed a data-driven approach where a proportional feedback term is added to the model equation. When the gain of the feedback term is properly selected – typically comparable to the magnitude of the linear coefficients L_{ij} of the reduced-order model Eq.(4) – the estimates are found to be sufficiently robust. This approach and the one proposed previously, have a disadvantage in terms of the requirement of a perfect knowledge of the real trajectory of the state vector. In practice, it may not be possible to have an easy on-line access to this quantity. In contrast, the user may be able to instantaneously

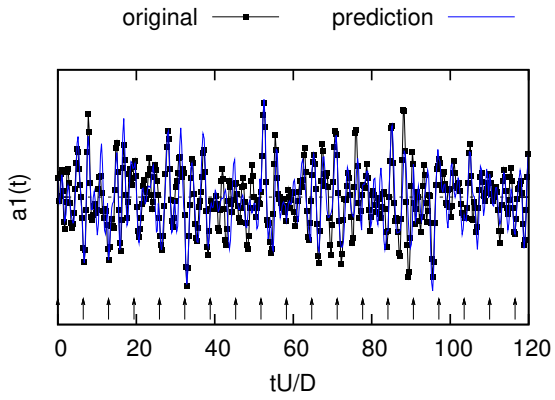


Figure 6: Time histories of the first POD temporal coefficients (line with dots) and its prediction (blue line) obtained by intrinsic stabilisation. Vertical arrows indicate the time steps where the initial condition is updated by observing the original POD coefficient.

access the "observable" quantities such as radiated pressure or, again, velocity at given locations. Among the range of sequential data assimilation techniques used, for example, in meteorology for long-term weather prediction, here we introduce the Ensemble Kalman filter (EnKF), detailed in Appendix A. The latter is an extended version of the more conventional Kalman filter (KF) and is appropriate for systems described by non-linear model equation. Unlike the KF, the EnKF may be interpreted as a framework providing statistically suboptimal estimate of the underlying system state for given noisy and/or inaccurate observations. The EnKF generates an analysis ensemble, that is, an ensemble of model states that reflects both an estimate of the true state (through its mean) and the uncertainty of this estimate (through its spread). This probabilistic approach allows to deal with non-linear model equations and the final estimated state vector may be seen as a higher order linearisation of the true state. Like the KF, a recursive resolution is employed. At each time step, a model equation is first used to generate an *a priori* (or forecast) background ensemble of estimated states and then the available observations are assimilated to build the *a posteriori* (or analysis) ensemble thanks to an optimal filter. Extensive details on the EnKF can be find in the literature (see Evensen [17] for a thorough review). Its implementation thus requires a model equation of the system state, which is provided here by the reduced-order model of Eq.(8), and an observation equation relating the state vector and the available measurements. For the observation equation, here we propose to start with a fictive velocity sensor located at $(x/D, y/D) = (3, 0.5)$ and a fictive pressure sensor in Ω_F^{rot} located at $(x/D, y/D) = (7, 2.5)$ for local surveys of the velocity and the radiated pressure (hereafter referred to $u(t)$ and $p(t)$ respectively). This choice is motivated from a perspective of practical implementation. The observation equation can thus be read, in a discrete form, as,

$$\mathbf{z}(t_k) = H\mathbf{a}^R(t_k) \;\text{ with }\; \mathbf{z}(t_k) = \begin{bmatrix} u(t_k) \\ p(t_k) \end{bmatrix} \in \mathbb{R}^2. \tag{19}$$

Here matrix $H \in \mathbb{R}^{2\times N_{gal}}$ maps the estimated state vector $\mathbf{a}^R(t)$ to the available observation vector $\mathbf{z}(\mathbf{t})$. Its first row contains the values of the spatial eigenfunctions, calculated *a priori*, at the sensor location. For the mapping between the estimated temporal POD

coefficients and the pressure in the radiated field, we propose the use of a linear stochastic estimate of the pressure field, which can be written as,

$$\tilde{p}(t+\tau_{ac}) = \sum_{i=1}^{N_{gal}} \alpha_i a_i^R(t), \tag{20}$$

where $\{\alpha_i\}_{i=1,\dots,N_{gal}}$ defines a set of coefficients which minimises $||p(t) - \tilde{p}(t)||_2^2$ in the L_2 norm sense. The time-delay τ_{ac} introduced in the last expression takes the acoustic propagation time into account. This quantity needs to be optimised but we will not discuss this point here. The second row of the observation matrix H contains the values of the coefficients α_i. Finally, the non-linear discrete system considered for application of the EnKF may thus be read as,

$$\left\{ \begin{array}{ll} \mathbf{a}^R(t_k) & = \mathbf{F}\left(\mathbf{a}^R(t_{k-1}), \mathbf{y}\right) + \mathbf{u}_k \\ \mathbf{z}(t_k) & = H\mathbf{a}^R(t_k) + \mathbf{v}_k \end{array} \right. \tag{21}$$

where $\mathbf{u}_k$ and $\mathbf{v}_k$ are white zero-mean uncorrelated process and observation noise respectively with known covariance matrices. These noise components are introduced to take into account the errors in the model equations and errors in the measurements. These can also be interpreted as levels of trust in both the model and the available observations. Here, the time-discrete form of the model equation used in the implementation of the EnKF is obtained by applying a 4-th order Runge-Kutta scheme to Eq.(8). The white Gaussian noises are initialised identically as $\mathcal{N}(0, 0.01\mathbf{I}_{N_{\text{gal}}})$ and the size of the state ensemble is fixed as 500.

Results obtained using this framework are illustrated in figure 7 for the first temporal POD coefficient. The top part of the figure shows the predictive variable $\mathbf{z}(\mathbf{t})$ obtained with the observation equation of Eq.(21), allowing for measurement noise, while the bottom part shows the time history of the estimated first POD coefficient. In contrast to the intrinsic stabilization or feedback procedures discussed above, here it is important to point out that the values of the true temporal POD coefficients have not been used. Only the observations are assimilated. Comparing with the results obtained without assimilation, as shown in figure 5, here the time history of the temporal POD coefficient is well recovered over a long time duration while the assimilation procedure is maintained. The error of reconstruction is found to increase slightly with the mode number. In order to increase the level of accuracy of the prediction, additional fictive sensors can be used for observation. Figure 8 shows the results of assimilation procedure for the first four temporal POD coefficients with velocity sensors distributed along the mixing layer axis ($y/D = 0.5$) between $x/D = 2$ and $x/D = 6.5$ at a step of $0.5D$. The estimates obtained without assimilation are also reported for comparison. As illustrated, time histories of the temporal coefficients are well reproduced with very small discrepancies. While the trajectory estimated without assimilation rapidly drifts from the correct one, that estimated with the EnKF fits well with the original data, thanks to only a limited number of observations and a stable model equation.

5.3. Dual parameters and state estimation

As mentioned in the introduction, the typically observed drift of POD-ROMs is believed to be due to the omission of the residual term $\mathcal{R}_i$ in Eq.(4). Empirical subscale linear and

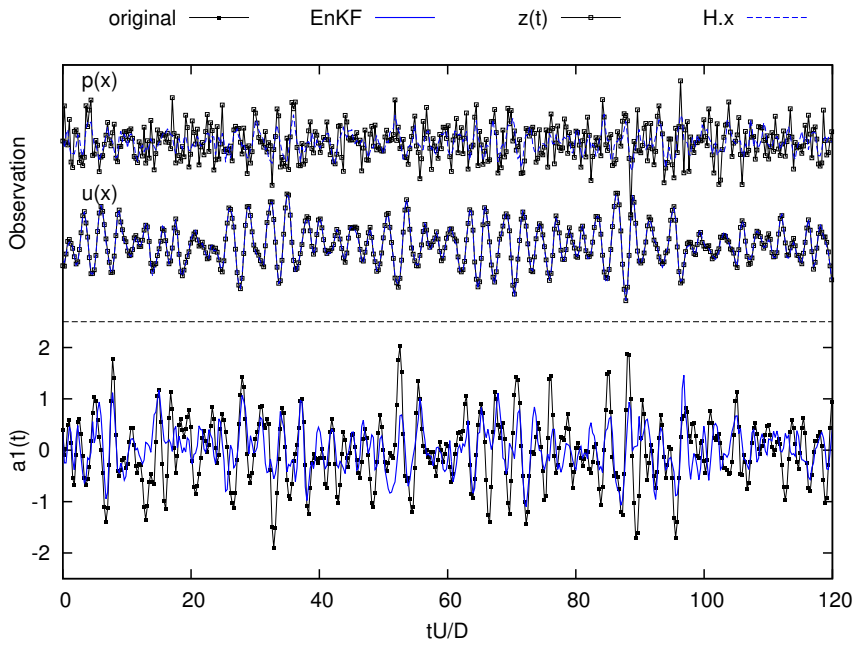


Figure 7: (Bottom part) Time histories of the first POD temporal coefficient (line with dots) and its prediction using the EnKF (blue line). (Top part) Time histories of the observations and their estimates through 19 used for assimilation.

non-linear models have been proposed for this term in the literature [34]. The modal linear eddy viscosity model proposed by Östh et al. [34] is considered here, mainly in order to demonstrate the potential of the assimilation scheme detailed below. The residual term in Eq.(4) is hence written as,

$$\mathcal{R}_i(t) = \nu_i^t \sum_{j=1}^{N_{\text{gal}}} L_{ij} a_j^R(t). \tag{22}$$

Traditionally, ν_i^t can be obtained by solution matching [14] or a modal power balance [33]. In this study, the time-independent unknown parameter vector $\theta = (\nu_1^t, \cdots, \nu_{N_{\text{gal}}}^t)^\top$ is estimated simultaneously with the state of the system.

While the EnKF presented previously offers the potential of online correction of the state estimate, it does not allow an improvement of the model itself through, for example, the parameters vector θ. Therefore, here we introduce a dual state-parameter prediction technique based on EnKF, known as Dual-EnKF, as proposed initially by Moradkhani et al. [31]. The Dual-EnKF requires separate state-space representation for the state and parameters through two parallel filters. The equation of evolution for the parameter vector is set up artificially assuming a random walk with a zero-mean white Gaussian noises, hereafter $\mathbf{w}$. At each time step, the usual EnKF described in Appendix A is applied twice: firstly to update the parameter vector, and secondly to update the state vector using the analysis parameter vector. This dual approach is generally found to outperform joint Kalman filters where state and parameters are updated at the same time step, by overcoming two limitations [3]. The first one is that in a joint approach, the number of parameters that can be estimated is restricted to the number of measurements available. The second is that if

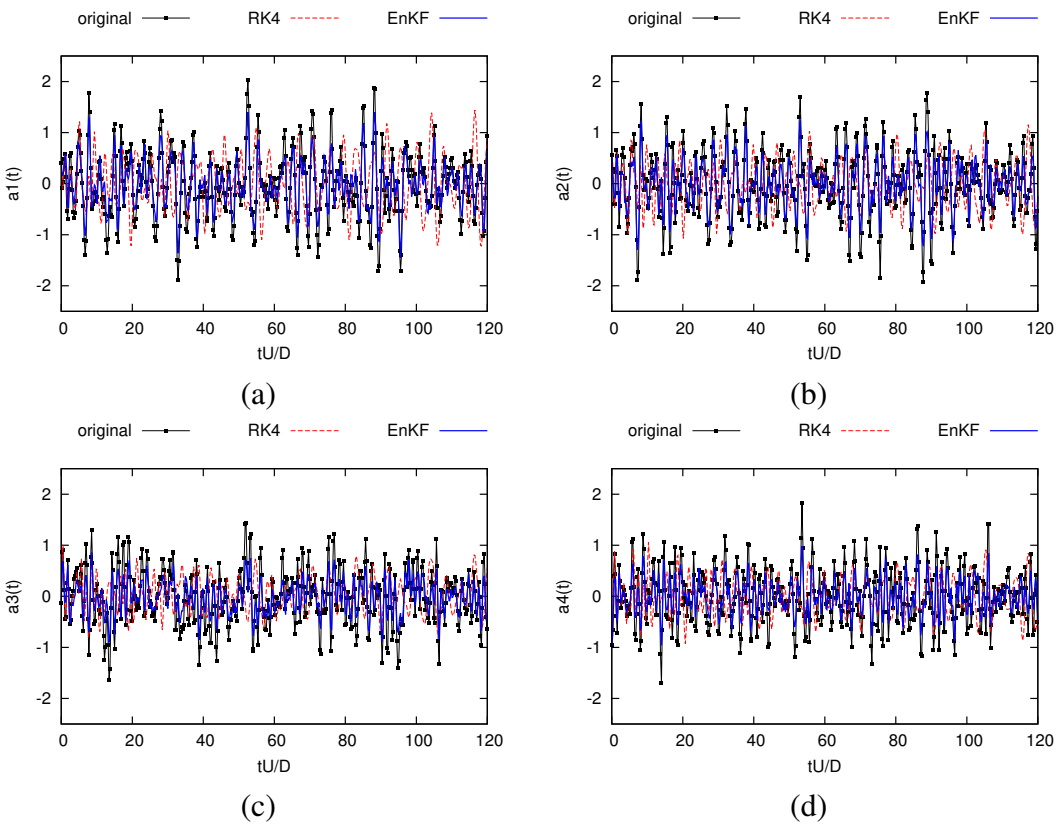


Figure 8: Time histories of the first four POD temporal coefficients (line with dots) and that of their estimates without (red dashed line) and with (blue line) data assimilation using the ensemble Kalman filter of Eq.(21). The observation vector includes far-field pressure sensor and longitudinal velocity sensors along the shear layer axis.

the sensitivity matrix of the state-to-output equations is ill-conditioned, the joint estimation results in deteriorated accuracy for parameter estimation. Finally, the non-linear discrete system considered here can be read as,

$$\left\{\begin{array}{ll} \boldsymbol{\theta}_k & = \boldsymbol{\theta}_{k-1} + \mathbf{w}_k \\ \mathbf{a}^R(\boldsymbol{\theta}_k; t_k) & = \mathbf{F}\left(\mathbf{a}^R(\boldsymbol{\theta}_k, t_{k-1}), \mathbf{y}\right) + \mathbf{u}_k \\ \mathbf{z}(t_k) & = H\mathbf{a}^R(\boldsymbol{\theta}_k; t_k) + \mathbf{v}_k \end{array}\right. . \tag{23}$$

Demonstration of the Dual-EnKF applied on standard Lorenz model equation in order to recover the coefficients of the model equation is discussed in Appendix A. For the present purpose, the white Gaussian noises are initialised as $\mathbf{u}_0 = \mathbf{v}_0 = \mathcal{N}(0, 0.01\mathbf{I}_{N_{\text{gal}}})$ and $\mathbf{w}_0 = \mathcal{N}(0, \mathbf{I}_{N_{\text{gal}}})$. An ensemble of 500 members is chosen and is found to give satisfactory results. Note that the same observation vector as that used in §5.2 is considered. The parameter vector $\boldsymbol{\theta}$ is initialised with null values. The entire time sequence is divided into a learning sequence of duration 265 time units during which the observations are assimilated at each time step, followed by a validation sequence during which observations are considered to be available at only every 50 time steps (with a corresponding time interval of $\Delta t_z U/D \approx 13$).

The value of ν_1^t over the learning window is reported in figure 9. The shaded area cor-

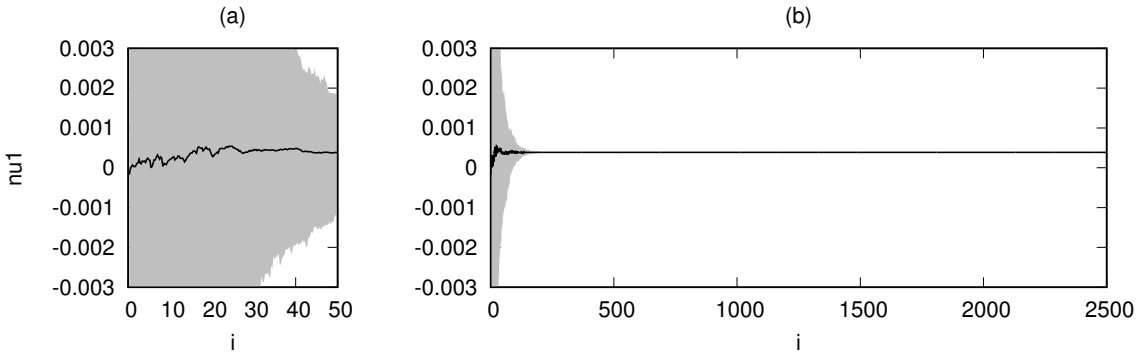


Figure 9: Time evolution of the estimated ν_1^t parameter of Eq.(22) obtained with Dual-EnKF. (a) Close view of the first 50 time units, (b) Complete view of the full learning sequence.

.

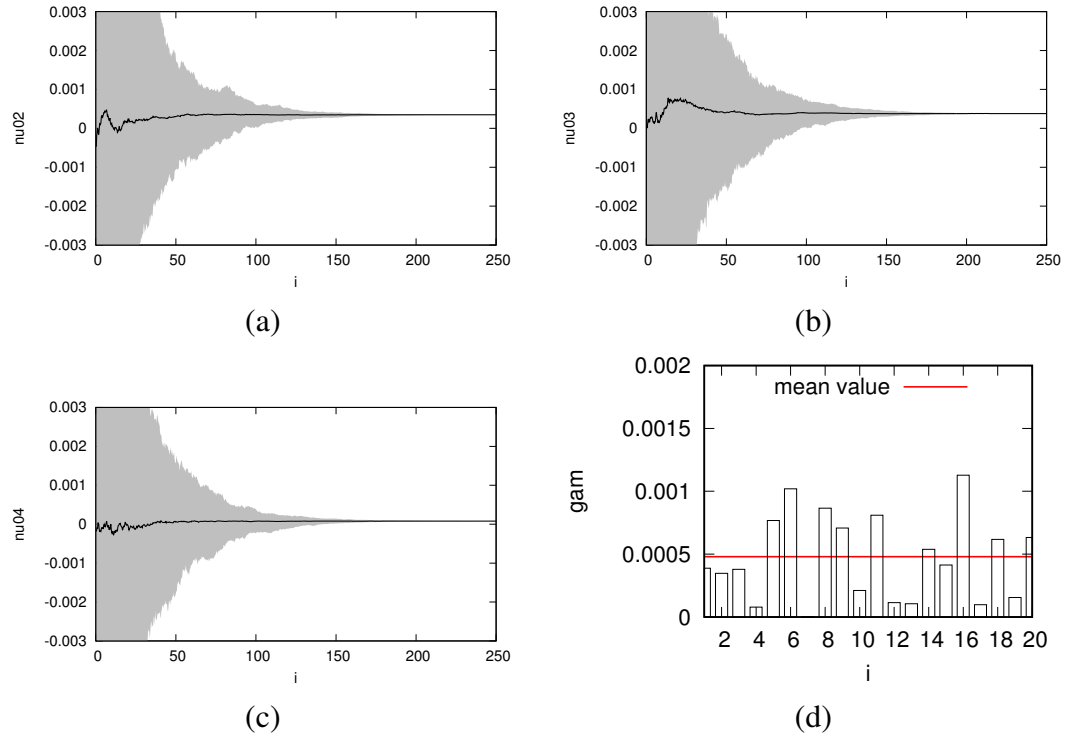


Figure 10: (a,b and c) Evolution of the first estimated parameters during the assimilation window. (d) Values of the parameter vector at the end of the assimilation window with (red line) mean value of θ.

.

responds to the variance in the ensemble members (or confidence interval) while the black line represents the ensemble mean, i.e. the estimated parameter value. The variance of the ensemble is arbitrarily initialised to be large. The parameter value is found to converge smoothly in a given region of the parameter space towards a solution with negligible variance or, in other words, uncertainty. Similar results are obtained for the other parameters as reported in figure 10(a-c). Time evolution of the temporal coefficient $a_1(t)$ during the learning window is reported in figure 11(a). The error between the reference and estimated

trajectories can be quantified with the help of normalised root-mean-square error (hereafter NRMSE) defined as the error between the state estimate and the state reference given by,

$$\text{NRMSE}(t) = \sqrt{\frac{\sum_{i=1}^{N_{\text{gal}}} \left[a_i^P(t) - a_i^R(t)\right]^2}{\sum_{i=1}^{N_{\text{gal}}} \left[a_i^P(t)\right]^2}}. \tag{24}$$

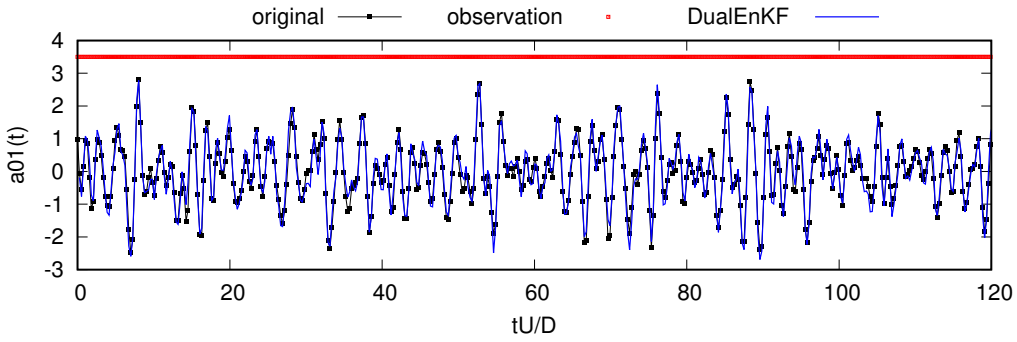
As shown in figure 11(b), the NRMSE remains low during the learning window. At the end of the learning window, as mentioned previously, the observations are then assimilated only every 13 time units. Time evolution of the estimated coefficient $a_1(t)$ and NRMSE after the learning window is reported in figure 12(a and b). The NRMSE only slightly increases and the reference trajectory for $a_1(t)$ is well recovered. Similar results are obtained for other modes, as illustrated in figure 13 where estimated trajectories for $a_2(t),\ a_4(t),\ a_6(t)$ and $a_{10}(t)$ are shown. Also, while not shown here, the NRMSE is found to slightly increase with the mode number but the estimation remains satisfactory. Additionally, though not shown here, it was observed that if the assimilation of new observations is stopped, the estimated trajectories remain bounded but the reference dynamics is lost as observed initially in §4.2. At this stage, this raises the question of the reliability of POD-ROMs to provide long-term prediction without correction of the estimate thanks, for example, to assimilation of new observations.

6. Reconstruction of the spatio-temporal flow field dynamics associated with low-frequency sound emission

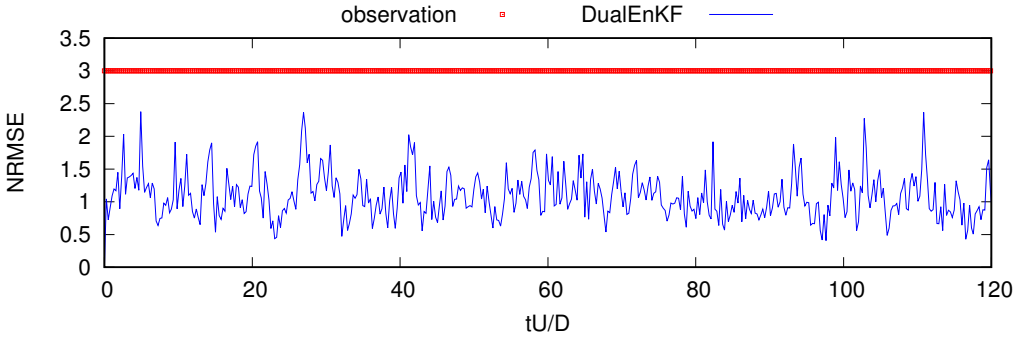
In this last part, we test the degree to which the corrected ROM can reproduce the dynamics of the reference spatio-temporal flow field associated with sound-producing mechanisms.

First, an unrolled phase portrait over a long time horizon using the first two estimated temporal POD coefficients is reported in figure 14. The phase portrait is found to be well-bounded, thanks to stabilization of the dynamical system, and to properly follow the original trajectory. Using the distance defined by $d(t) = \sqrt{a_1(t)^2 + a_2(t)^2}$ to identify the location of a point along the phase portrait with coordinates $(a_1(t), a_2(t))$ at any time, the trajectory described by the portrait may be seen as an orbit with center close to coordinate (0,0) and essentially bounded. However, the trajectory is found to jump intermittently to another orbit characterised by a larger diameter. The phase portrait shown in figure 14 is coloured in red when the diameter of the trajectory is above a given threshold to highlight these intermittent jumps. These jumps are connected to the sound producing intermittent events discussed by Cavalieri et al. [9] and highlighted in Kerhervé et al. [25] for the present data.

The full space-time velocity flow field can be obtained by combining the spatial POD eigenfunction extracted from the original LES data with the estimated temporal POD coefficients. A quantitative evaluation of the error in the flow estimate is provided by comparing the centerline velocity spectra at two different stations downstream of the exit nozzle for the original and estimated reduced-order flows, as illustrated in figure 15 using the temporal coefficients estimated with the Dual-EnKF previously discussed. The overall frequency content below $St_D \sim 1$ of the reference data is well reproduced. Above $St_D \sim 1$,



(a)



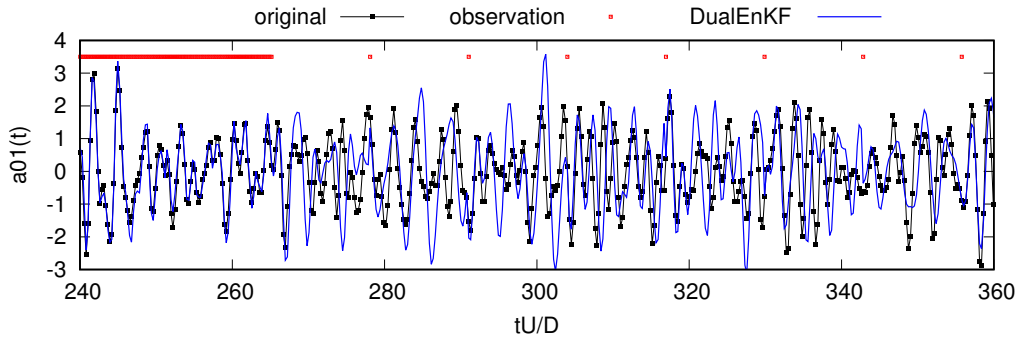
(b)

Figure 11: Time evolution in the learning window of (top) the first temporal POD coefficient estimated using the Dual-EnKF scheme and (b) the associated root-mean-squre error defined in Eq.(24). The red dots indicate the time instants at which the observations have been assimilated. (Black square dots) Orignal POD coefficient and (blue line) coefficient estimated with Dual-EnKF.

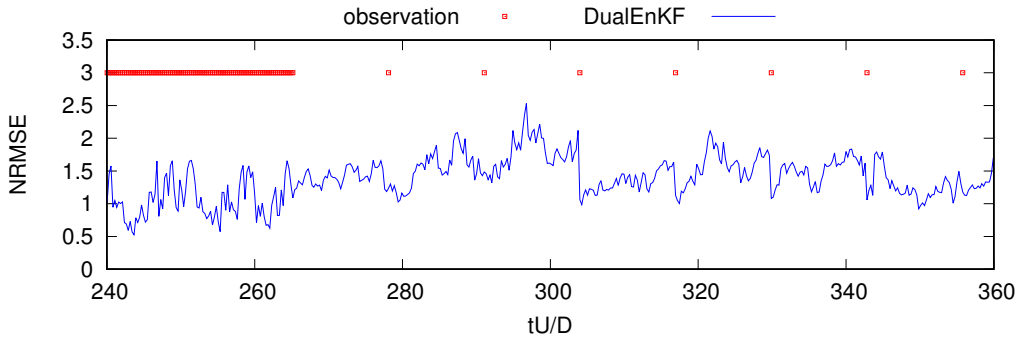
the discrepancies with the reference data are observed due to, firstly, the POD truncation and, secondly, the higher discrepancies observed in the higher modes between the reference and estimated temporal coefficients (see figure 13). Spurious frequency peaks are observed for the estimated field. These peaks are harmonics of the low frequency peak around $St_D \approx 0.08$ which is exactly the frequency associated with the time interval of the assimilation of new observations ($\Delta t_z U/D = 13$). Another estimation was effected with assimilation at time intervals $\Delta t_z U/D = 4$. The same frequency spectra for the estimated field are also plotted in figure 15. The spurious frequency peaks have almost disappeared, with only one peak close to $St_D \approx 0.25$, which is again the frequency associated with the time interval of assimilation. In conclusion, the energy content of the dominated coherent structures of interest is well reproduced by the low-order modelling framework.

7. Conclusion

A data-driven strategy for the elaboration of POD reduced-order models capable of predicting flow states over a long-time horizon is proposed. The methodology combines (i) a regularised least-square identification of the coefficients of the POD-ROM, (ii) a modal linear eddy viscosity model parametrised with an unknown parameters vector to mimic the high-order modes, and (iii) a sequential data assimilation technique known as the dual-

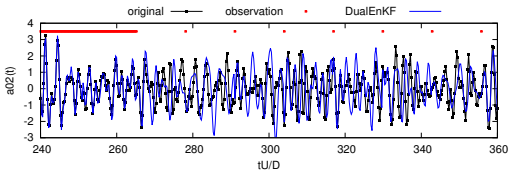


(a)

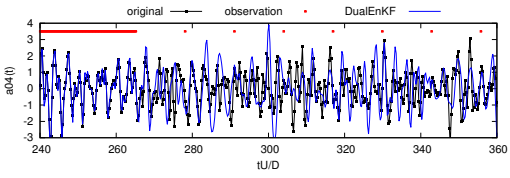


(b)

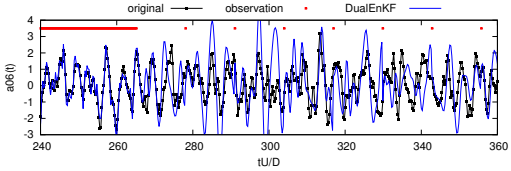
Figure 12: Time evolution in the testing window of (top) the first temporal POD coefficient estimated using the dual-EnKF scheme and (b) the associated root-mean-squre error defined in Eq.(24) . The learning window ends at $tU/D = 265$, beyond which observations are assimilated every 13 time units. Refer the caption of figure 11 for more information.



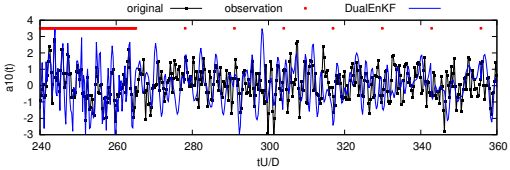
(a)



(b)



(c)



(b)

Figure 13: Time evolution in the testing window of different temporal POD coefficients: (a) $a_2(t)$, (b) $a_4(t)$, (c) $a_6(t)$ and (d) $a_10(t)$. Refer the caption of figure 11 for more information.

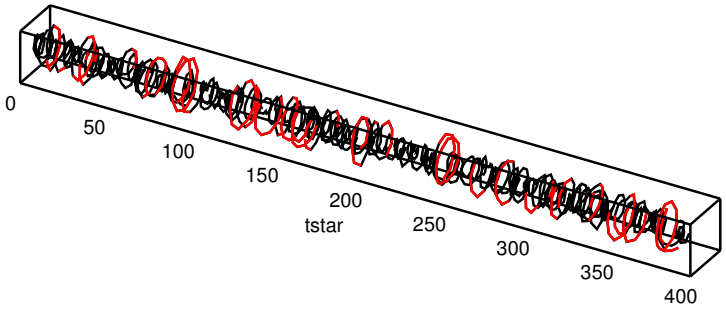
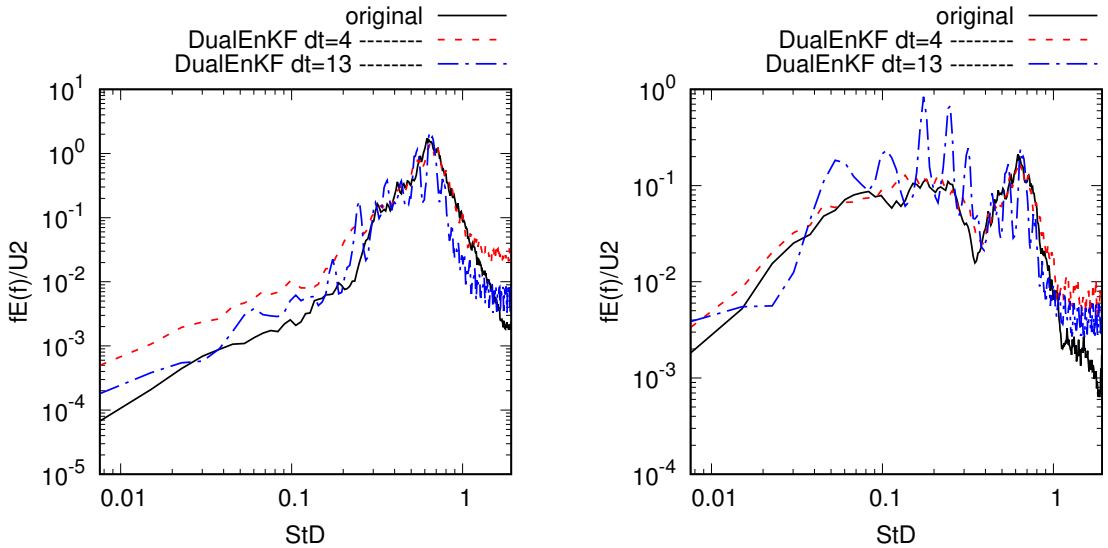


Figure 14: Unrolled phase portrait of POD modes 1 and 2. When the trajectory of the portrait is observed to follow an orbit at a larger distance from the centerline, it is coloured in red.



(a) $x/D = 2$ (b) $x/D = 8$

Figure 15: Energy spectrum of the fluctuating axial velocity component at a point located along the jet centerline at (a) $x/D = 2$ and (b) $x/D = 8$ for the LES and modelled fields using the long-term prediction procedure.

ensemble Kalman filter (Dual-EnKF). The latter is used to sequentially propagate and update the estimated state and to discover, simultaneously, the parameter vector of the residual term by assimilating observations at regular time steps. A Mach 0.9 turbulent jet, simulated using Large Eddy Simulation by Bogey and Bailly [6], is considered as a test case. The present work follows that of Kerhervé et al. [25] who proposed a methodology to educe the flow motions associated with sound-producing events and responsible for low-angle sound radiation. The data-driven strategy is applied here with the objective to build a low-order dynamical system of these specific motions. A twenty mode POD-ROM is therefore identified and used in combination with the Dual-EnKF to recover the dynamics of the sound-producing flow motions over long time horizon using only a limited number of observations. The types of observation have been chosen to mimic what can be replicated in experiments such as, for instance, point velocity and far-field pressure measurements. The regularised identification procedure allows to obtain a bounded dynamical system but displays a behaviour where the trajectory drifts from the true one beyond initial few time steps. However, when combined with the Dual-EnKF, the trajectory of the estimation is corrected and is observed to be replicating the trajectory of the true state, albeit with small errors. As proposals for improvements in the current strategy, further investigation into the selection of the observation vector, the estimation of the model and error covariances, or the identification of reduced-order models which are non-linear in the coefficients can be carried out.

References

[1] Aubry, N., Holmes, P., Lumley, J.L., Stone, E., 1988. The dynamics of coherent structures in the wall region of a turbulent boundary layer. Journal of Fluid Mechanics 192, 115–173.

[2] Balajewicz, M.J., Dowell, E.H., Noack, B.R., 2013. Low-dimensional modelling of high-reynolds-number shear flows incorporating constraints from the navier–stokes equation. Journal of Fluid Mechanics 729, 285–308.

[3] Bavdekar, V.A., Prakash, J., Shah, S.L., Gopaluni, R.B., 2013. Constrained dual ensemble kalman filter for state and parameter estimation, in: 2013 American Control Conference, IEEE. pp. 3093–3098.

[4] Bergmann, M., Cordier, L., 2008. Control of the circular cylinder wake by Trust-Region methods and POD Reduced-Order Models. Research Report 6552, INRIA 06, URL https://hal.inria.fr/inria-00284258 .

[5] Bogey, C., Bailly, C., 2005. Effects of inflow conditions and forcing on subsonic jet flows and noise. AIAA J. 43(5), 1000–1007.

[6] Bogey, C., Bailly, C., 2006. Computation of a high Reynolds number jet and its radiated noise using large eddy simulation based on explicit filtering. Comp. Fluids 35(10), 1344–1358.

[7] Bogey, C., Bailly, C., Juvé, D., 2003. Noise investigation of a high subsonic moderate Reynolds number jet using a compressible LES. Theor. Comput. Fluid Dyn. 16(4), 273–297.

[8] Borggaard, J., Iliescu, T., Wang, Z., 2011. Artificial viscosity proper orthogonal decomposition. Mathematical and Computer Modelling 53, 269–279.

[9] Cavalieri, A., Jordan, P., Agarwal, A., Gervais, Y., 2011. Jittering wave-packet models for subsonic jet noise. J. Sound Vib. 330, 4474–4492.

[10] Cavalieri, A.V., Jordan, P., Lesshafft, L., 2019. Wave-packet models for jet dynamics and sound radiation. Applied Mechanics Reviews 71.

[11] Cazemier, W., Verstappen, R., Veldman, A., 1998. Proper orthogonal decomposition and low-dimensional models for driven cavity flows. Physics of Fluids (1994-present) 10, 1685–1699.

[12] Chen, M., Liu, S., Tieszen, L., Hollinger, D., 2008. An improved state-parameter analysis of ecosystem models using data assimilation. ecological modelling 219, 317–326.

[13] Cordier, L., Abou El Majd, B., Favier, J., 2010. Calibration of POD reduced-order models using tikhonov regularization. Int. J. Num. Meth. in Fluids 63, 269–296.

[14] Cordier, L., Noack, B.R., Tissot, G., Lehnasch, G., Delville, J., Balajewicz, M., Daviller, G., Niven, R.K., 2013. Identification strategies for model-based control. Experiments in fluids 54, 1–21.

[15] Crow, S., Champage, F., 1971. Orderly structures in jet turbulence. J. Fluid Mech. 48(3), 547–591.

[16] Evensen, G., 2003. The ensemble kalman filter: Theoretical formulation and practical implementation. Ocean dynamics 53, 343–367.

[17] Evensen, G., 2009. Data assimilation: the ensemble Kalman filter. Springer Science & Business Media.

[18] Galletti, B., Bruneau, C., Zannetti, L., Iollo, A., 2004. Low-order modelling of laminar flow regimes past a confined square cylinder. Journal of Fluid Mechanics 503, 161–170.

[19] Hansen, P., 1994. Regularization tools: A matlab package for analysis and solution of discrete ill-posed problems. Numerical algorithms 6, 1–35.

[20] Holmes, P., Lumley, J., Berkooz, G., 1998. Turbulence, coherent structures, dynamical systems and symmetry. Cambridge University Press .

[21] Jordan, P., Colonius, T., 2013. Wave packets and turbulent jet noise. Annual review of fluid mechanics 45, 173–195.

[22] Jordan, P., Schlegel, M., Stalnov, O., Noack, B., Tinney, C., 2007. Identifying noisy and quiet modes in jet. 13th AIAA/CEAS Aeroacoustics Conference, Paper 2007-3602 .

[23] Kalb, V., Deane, A., 2007. An intrinsic stabilization scheme for proper orthogonal decomposition based low-dimensional models. Physics of fluids 19, 054106.

[24] Kato, H., Yoshizawa, A., Ueno, G., Obayashi, S., 2015. A data assimilation methodology for reconstructing turbulent flows around aircraft. Journal of Computational Physics 283, 559–581.

[25] Kerhervé, F., Jordan, P., Cavalieri, A., Delville, J., Bogey, C., Juvé, D., 2012. Educing the source mechanism associated with downstream radiation in subsonic jets. J. Fluid Mech. 710, 606–640. doi:http://dx.doi.org/10.1017/jfm.2012.378.

[26] Leroux, R., Chatellier, L., David, L., 2014. Bayesian inference applied to spatio-temporal reconstruction of flows around a naca0012 airfoil. Experiments in Fluids 55, 1–19.

[27] Lighthill, M., 1952. On sound generated aerodynamically: I. general theory. Proc. R. Soc. A 222, 564–587.

[28] Liu, J., West, M., 2001. Combined parameter and state estimation in simulation-based filtering, in: Sequential Monte Carlo methods in practice. Springer, pp. 197–223.

[29] Lumley, J., 1967. The structure of inhomogeneous turbulent flows. Atmospheric Turbulence and Wave Propagation (Moscow, Nauka) , 166–178.

[30] Michalke, A., 1970. A wave model for sound generation in circular jets .

[31] Moradkhani, H., Sorooshian, S., Gupta, H.V., Houser, P.R., 2005. Dual state–parameter estimation of hydrological models using ensemble kalman filter. Advances in Water Resources 28, 135–147.

[32] Noack, B., Papas, P., Monkewitz, P., 2002. Low-dimensional Galerkin model of a laminar shear-layer. Technical Report. Tech. Rep. 2002-01. Laboratoire de Mecanique des Fluides, Departement de Genie Mecanique, Ecole Polytechnique Fédérale de Lausanne, Switzerland.

[33] Noack, B.R., Papas, P., Monkewitz, P.A., 2005. The need for a pressure-term representation in empirical galerkin models of incompressible shear flows. Journal of Fluid Mechanics 523, 339.

[34] Östh, J., Noack, B.R., Krajnović, S., Barros, D., Borée, J., 2014. On the need for a nonlinear subscale turbulence term in pod models as exemplified for a high-reynolds-number flow over an ahmed body. Journal of Fluid Mechanics 747, 518–544.

[35] Perret, L., Collin, E., Delville, J., 2006. Polynomial identification of POD based low-order dynamical system. J. Turb. 7, 1–15.

[36] Podvin, B., 2009. A proper-orthogonal-decomposition–based model for the wall layer of a turbulent channel flow. Physics of Fluids (1994-present) 21, 015111.

[37] Protas, B., Noack, B.R., Östh, J., 2015. Optimal nonlinear eddy viscosity in galerkin models of turbulent flows. Journal of Fluid Mechanics 766, 337–367.

[38] Rempfer, D., Fasel, H.F., 1994. Evolution of three-dimensional coherent structures in a flat-plate boundary layer. Journal of Fluid Mechanics 260, 351–375.

[39] Sirisup, S., Karniadakis, G., 2004. A spectral viscosity method for correcting the long-term behavior of pod models. Journal of Computational Physics 194, 92–116.

[40] Sirovich, L., 1987. Turbulence and the dynamics of coherent structures. i. coherent structures. Quarterly of applied mathematics 45, 561–571.

[41] Suzuki, T., 2012. Reduced-order kalman-filtered hybrid simulation combining particle tracking velocimetry and direct numerical simulation. Journal of Fluid Mechanics 709, 249–288.

[42] Suzuki, T., 2014. Pod-based reduced-order hybrid simulation using the data-driven transfer function with time-resolved ptv feedback. Experiments in Fluids 55, 1–17.

[43] West, M., 1993. Mixture models, monte carlo, bayesian updating, and dynamic models. Computing Science and Statistics , 325–325.

Appendix A. Ensemble and dual-Ensemble Kalman filters

Appendix A.1. Ensemble Kalman filter

The main elements of the implementation of the dual-Ensemble Kalman filter for state and parameters estimation are detailed here, following the original formulation of Evensen [16] and Moradkhani et al. [31]. Here, the dimension of the state space is denoted as n while the true state of the system at time t_k is denoted as $\mathbf{x}_k \in \mathbb{R}^n$. Note that for practical implementation, the time is discretised, which justifies the subscript notation. Let the dynamics of the system be described by a non-linear operator $\mathcal{M}(\mathbf{x}_k, \theta)$ where θ forms a vector of time-invariant parameters of dimention n_b. In addition, the dimension of the observation space is considered as n_o while the observation vector at time t_k is denoted by $\mathbf{y}_k \in \mathbb{R}^{n_o}$. Here, we consider the case where the observation and state vectors are related through $\mathbf{y}_k = \mathcal{H}(\mathbf{x}_k)$, where $\mathcal{H}$ denotes either a linear or non-linear operator. Kalman filtering is a two steps procedure, alternating forecast and analysis of the system state. In the following, the forecast and analysis of the state at time t_k are denoted by $\mathbf{x}_k^f$ and $\mathbf{x}_k^a$ respectively. In the sequential procedure, the dynamical model $\mathcal{M}$ is first used to evolve the state estimate forward in time, forming the forecast estimate $\mathbf{x}_k^f$. When an observation is available, a correction of $\mathbf{x}_k^f$ is effected to generate the analysis estimate $\mathbf{x}_k^a$ such that the mean-square error between the prediction $\mathcal{H}(\mathbf{x}_k^f)$ and the observation $\mathbf{y}_k$ is minimised.

When the dynamics of the state is described by a linear model, the linear Kalman filter (KF) offers a statistically optimal estimate of the underlying system state for given noisy observations and model. The generalisation to non-linear system is offered by the Extended

Kalman filter (EKF) which requires the Jacobian of the dynamics matrix. This filter mimics the classical Kalman filter by propagating a matrix (the surrogate covariance) that is analogous to the error covariance in the linear case. Another approach belonging to the category of particle filters which is widely used in weather forecasting, which involves high order non-linear models and the initial states that are uncertain, is the ensemble Kalman filter (EnKF). While the KF uses a single state estimate, the EnKF uses a statistical sample of state estimates, called an ensemble. It is a suboptimal estimator based on Monte Carlo or ensemble generations where the approximation of the forecast state error covariance matrix is made by propagating an ensemble of model states using the updated states (ensemble members) from the previous time step. In contrast to the standard Kalman filter, the propagation of covariance matrices is not necessary. This results in a reduction of computational cost, albeit with the inclusion of an additional cost in terms of maintaining the ensemble members throughout the time marching.

Let the discrete-time non-linear system be expressed as,

$$\begin{cases} \mathbf{x}_k &= \mathcal{M}(\mathbf{x}_{k-1}, \theta) + \eta_{k-1} \\ \mathbf{y}_k &= \mathcal{H}(\mathbf{x}_k) + \epsilon_k \\ \eta_k &\sim \mathcal{N}(0, \Sigma_k^{\mathcal{M}}) \\ \epsilon_k &\sim \mathcal{N}(0, \Sigma_k^{\mathcal{H}}) \end{cases}, \tag{A.1}$$

where $\eta_k \in \mathbb{R}^n$ and $\epsilon_k \in \mathbb{R}^{n_o}$ are white, zero-mean, uncorrelated process and observation noises respectively with known covariance matrices $\Sigma_k^{\mathcal{M}}$ and Σ_k^y. The EnKF propagates ensembles of state vectors in parallel such that each state vector represents one realisation of generated model replicates. In the following, ensembles at a time instant t_k are denoted as X_k with superscripts f or a for forecast and analysis respectively. These ensembles are formed with n_q replicates of the state vector, $\mathbf{x}_k^{(i)}$,

$$X_k = (\mathbf{x}_k^{(1)}, \cdots, \mathbf{x}_k^{(n_q)}).$$

For a given ensemble, the state estimate, hereafter $\overline{\mathbf{x}}_k$, must be interpreted as the most expected value of the ensemble:

$$\overline{\mathbf{x}}_k = \frac{1}{n_q} \sum_{i=1}^{n_q} \mathbf{x}_k^{(i)}. \tag{A.2}$$

The EnKF uses the standard KF equations to propagate the members of the ensembles. The forecast ensemble members are first obtained by propagating the analysis ensemble through the non-linear dynamics,

$$\mathbf{x}_k^{f,(i)} = \mathcal{M}(\mathbf{x}_{k-1}^{a,(i)}, \theta) + \eta_k^{(i)}, \text{ with } \eta_k^{(i)} = \mathcal{N}(0, \Sigma_k^{\mathcal{M}}). \tag{A.3}$$

A linear correction according to the standard KF is then used to update the members of the forecast ensemble as

$$\mathbf{x}_k^{a,(i)} = \hat{\mathbf{x}}_k^{f,(i)} + K_k \left(\mathbf{y}_k^{(i)} - \hat{\mathbf{y}}_k^{(i)} \right), \tag{A.4}$$

where $\hat{\mathbf{y}}_k^{(i)}$ denotes the predicted variable given by,

$$\hat{\mathbf{y}}_k^{(i)} = \mathcal{H}(\mathbf{x}_k^{f,(i)}), \tag{A.5}$$

and $\mathbf{y}_k^{(i)}$ denotes the replicate members of the observation vector $\mathbf{y}_k$ generated by,

$$\mathbf{y}_k^{(i)} = \mathbf{y}_k + \epsilon_k^{(i)} \text{ with } \epsilon_k^{(i)} = \mathcal{N}(0, \Sigma_k^{\mathcal{H}}). \tag{A.6}$$

The Kalman gain K_k is computed as,

$$K_k = \Sigma_k^{xy,f} \left[\Sigma_k^{yy} + \Sigma_k^{\mathcal{H}}\right]^{-1},$$

where Σ_k^{yy} denotes the forecast error covariance matrix of the prediction ensemble given by,

$$\Sigma_k^{yy} = \frac{1}{n_q - 1} \sum_{i=1}^{n_q} \hat{y}_k^{(i)} \left(\hat{y}_k^{(i)}\right)^\top,$$

and $\Sigma_k^{xy,f}$ denotes the forecast cross-covariance matrix between the forecast ensemble $\mathbf{x}_k^{f,(i)}$ and prediction $\hat{\mathbf{y}}_k^{(i)}$ given by,

$$\Sigma_k^{yy} = \frac{1}{n_q - 1} \sum_{i=1}^{n_q} x_k^{f,(i)} \left(\hat{y}_k^{(i)}\right)^\top.$$

In contrast to the Kalman filter, propagation of the covariance matrices is not necessary, which results in in a reduction of computational cost. A key of the EnKF however resides in the perturbation of the forcing and observation data by adding noise. The noise variances must be representatives of the uncertainty in these data.

Appendix A.2. Dual-Ensemble Kalman filter

When there is no guarantee that the model parameters will be constant in time or when the values chosen for these parameters represent only an initial guess, a simultaneous state-parameter estimation may be necessary. An extension of the EnKF to state-parameter estimation, known as Dual-EnKF, has been proposed by Moradkhani et al. [31]. The parameters are treated similarly as the state variables except for the time evolution, which is assumed to follow a random walk such that,

$$\theta_{k+1}^{f,(i)} = \theta_k^{a,(i)} + \tau_k^i. \tag{A.7}$$

Let θ_k denote the ensemble of parameter members $\theta^{(i)_k}$ at time t_k, with superscripts a or f for analysis and forecast estimates respectively, and with associated covariance Σ_k^θ. This artificial evolution is however known to result in an over-dispersion of parameter members leading to a loss of continuity between two consecutive time steps [12]. This flaw can be corrected thanks to kernel smoothing with location shrinkage [43, 28] where the individual ensemble members are drawn from a truncated multivariate normal distribution (*TMVN*),

$$\theta_k^{f,(i)} = \textit{TMVN}(a\theta_{k-1}^{a,(i)} + (1-a)\overline{\theta}_k^a, h^2 \Sigma_k^{\theta,a}) \tag{A.8}$$

where $h^2 = 1 - a^2$, $a = (3\delta - 1)/2\delta$ and $\delta \in \mathbb{R}$ is the smoothing parameter, typically between 0.95 and 0.99 [28] . In the Dual-EnKF, the standard EnKF equations are first

applied to update the parameters ensemble and to obtain the analysis parameter estimate accordingly:

$$\begin{cases} \theta_k^{f,(i)} = \mathit{TMVN}(a\theta_{k-1}^{a,(i)} + (1-a)\overline{\theta}_k^a, h^2\,\Sigma_k^{\theta,f}) \\ \mathbf{x}_k^{f,(i)} = \mathcal{M}(\mathbf{x}_{k-1}^{a,(i)}, \theta_k^{f,(i)}) \\ \hat{\mathbf{y}}_k^{(i)} = \mathcal{H}(\mathbf{x}_k^{f,(i)}) \\ \theta_k^{a,(i)} = \theta_k^{f,(i)} + K_k^{\theta}\left(\mathbf{y}_k^{(i)} - \hat{\mathbf{y}}_k^{(i)}\right) \\ K_k^{\theta} = \Sigma_k^{\theta y -}\left[\Sigma_k^{yy} + \Sigma_k^{\mathcal{H}}\right] \end{cases} . \tag{A.9}$$

For the same time step, the EnKF equations are then applied to update the state variables taking into account the updated parameters ensemble according to,

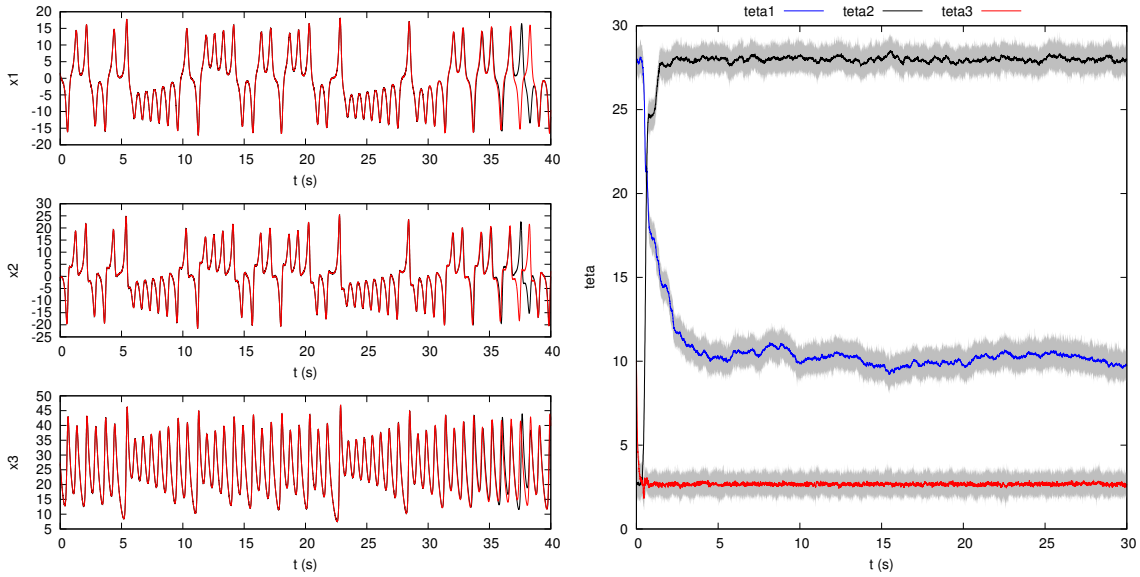
$$\begin{cases} \mathbf{x}_k^{f,(i)} = \mathcal{M}(\mathbf{x}_{k-1}^{a,(i)}, \theta_k^{a,(i)}) \\ \hat{\mathbf{y}}_k^{(i)} = \mathcal{H}(\mathbf{x}_k^{f,(i)}) \\ \mathbf{x}_k^{a,(i)} = \mathbf{x}_k^{f,(i)} + K_k^{x}\left(\mathbf{y}_k^{(i)} - \hat{\mathbf{y}}_k^{(i)}\right) \\ K_k^{x} = \Sigma_k^{xy,f}\left[\Sigma_k^{yy} + \Sigma_k^{\mathcal{H}}\right] \end{cases} . \tag{A.10}$$

It is noteworthy that, theoretically, the two previous steps in the sequence can be performed in reverse order.

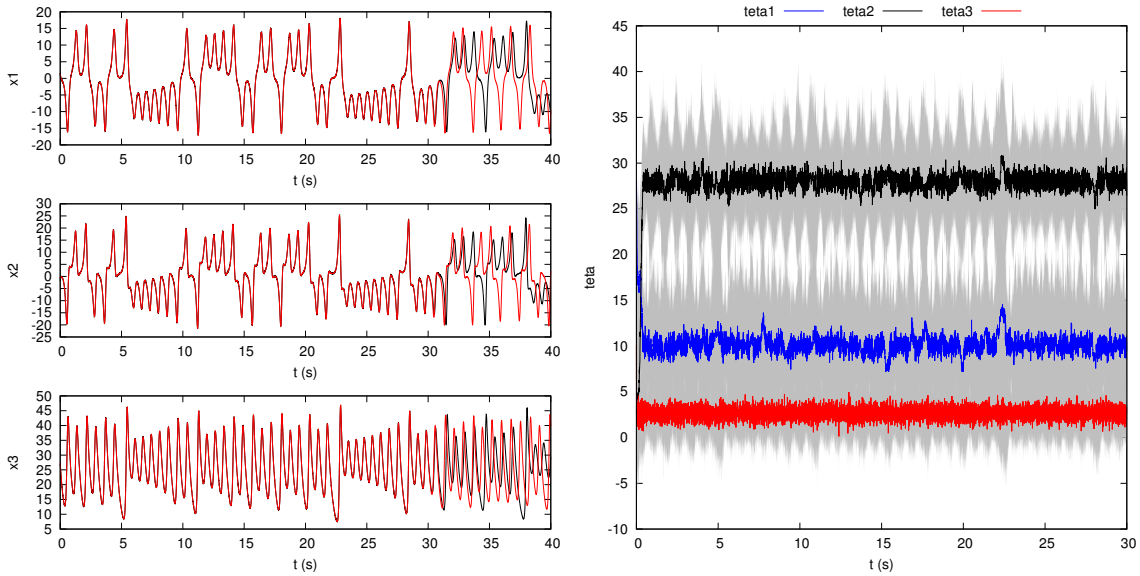
To illustrate the capability of the Dual-EnKF technique to recover both the state variable and the model's parameter vector, we apply the framework to the Lorenz-63 model,

$$\begin{cases} \dot{x}_1 = \sigma(x_2 - x_1) \\ \dot{x}_2 = x_1(\rho - x_3) - x_2 \\ \dot{x}_3 = x_1 x_2 - \beta x_3 \end{cases} , \tag{A.11}$$

where $\mathbf{x} = (x_1, x_2, x_3)^\top$ denotes the three-dimensional state vector and $\theta = (\sigma, \beta, \rho)^\top$ denotes the parameter vector. This model equations give a non-linear dynamical system. For $\theta = (10, 8/3, 28)$, the system is known to converge towards a chaotic solution known as strange attractor. The initial condition considered here is $\mathbf{x}(t=0) = \mathbf{x_0} = (5,5,5)^\top$. The model Eq.(A.11) can easily be rewritten in the general form given in Eq.(A.1). In the present case, the operator $\mathcal{H}$ is defined as the identity matrix. The parameter vector θ is initialised with incorrect values such that $\theta_0 = (8/3, 28, 10)^\top$, and random noise is introduced in the initial condition $\mathbf{x}_0$. Observations are assimilated until $t^* = 30$. During this learning sequence, the parameter vector is updated. At the end of the sequence, the last value obtained for the parameter vector is conserved and time integration is maintained for the state variable up to $t^* = 40$. To test the procedure with regards to observation noise, two cases are considered: (case A) $\Sigma^y = 0.01\mathbf{I}_3$ and (case B) $\Sigma^y = \mathbf{I}_3$. Results of the estimation procedure for the state and parameter vectors are reported in figure A.16. Due to incorrect initial parameter vector and initial condition, discrepancies between the true and estimated trajectories of the state variables are observed for the two cases in the first time steps. The estimated state trajectory and the parameter vector are then rapidly corrected and fast convergence towards the true system is observed. Even with large noisy observations, as shown in figure A.16(b), the system's trajectory is well recovered when



(a) Case A: $\Sigma^y = 0.01\mathbf{I}_3$



(b) Case B: $\Sigma^y = \mathbf{I}_3$

Figure A.16: Results of the dual state-parameter ensemble Kalman filter applied to the Lorenz-63 model equation with observation noise covariance (a) $\Sigma^y = 0.01\mathbf{I}_3$ and (b) $\Sigma^y = \mathbf{I}_3$. Left column: Time history of (black) true and (red) estimated state variables. Right column: Time history of the estimated parameters through the learning sequence. The learning sequence stops at $t = 30$.

observations are assimilated, thanks to the Dual-EnKF. When the assimilation is stopped, the estimated state's trajectory remains well estimated over duration which decreases as the observation noise level increases due to small errors in the parameter vector but continue to exhibit similar dynamics.

5.6 Conclusion

In this chapter, we considered the sequential data assimilation technique known as the Dual Ensemble Kalman Filter or Dual-EnKF (see the presentation in Sec. 3.2.3) as a cure to the inherent drift of ROM obtained from limited data. Through different test cases, it was demonstrated that such assimilation technique can be used for both long-term prediction and calibration of model parameters. In the present case, the constant, linear and quadratic coefficients of a POD-ROM are first identified thanks to the sparse identification method SINDy and used as a first guess model equation. A modal linear eddy viscosity model is then added to the POD-ROM as a residual term to mimic the high-order modes. This closure is parameterized with an unknown parameters vector. This modified POD-ROM then serves as the model equation in the Dual-EnKF algorithm to propagate in time the estimated state and to discover, simultaneously, the unknown parameter vector by assimilating sequentially new observations. Influence of the model and observation covariances, or of the size of the ensemble, on the estimation error were first examined. For this parametric investigation, the standard Lorenz-63 system was considered. The results show that the mean square error is primarily dependent on the ratio of the observation and model covariance levels. Moreover, the Dual-EnKF is found to be able to recover both the state evolution and the model parameters, even in the presence of noisy observations, as soon as special attention is paid on the choice of the initial covariance levels. The overall methodology is then replicated for different test cases with increasing number of degrees of freedom, such as a numerical cylinder wake flow at Reynolds number of 200, an experimental cylinder wake flow at Reynolds numbers equal to 1.5×10^4 and 5.5×10^4, and finally a numerical Mach 0.9 turbulent jet at Reynolds number of 4.0×10^5. For all these test cases, the original dataset, formed by velocity snapshots, is conventionally split into training and testing parts. The training part is used to identify an initial POD-ROM and to discover the parameters vector of its residual term. The testing part is then used to evaluate the learning POD-ROM in terms of mean square error and frequency content. For all the test cases, the sequential data assimilation technique is found to properly recover the dynamics of the full state, even in a long term horizon, with acceptable errors as soon as new observations are assimilated regularly.

The current strategy has however at least two severe limitations. The first one relates with the form of the reduced-order model given by (3.2) which is linear in coefficients. All the identification procedures discussed previously are only applicable for such linear problems. If the reduced-order model do not satisfy anymore this property, new identification strategies are therefore required. The second limitation is that an *a priori* model is mandatory. Again, it is not sure that a ROM of the form given by (3.2) is appropriate in the general case. To leverage these two limitations, the next chapter discusses the use of neural-networks to discover a surrogate model based only on data.

Chapter 6

Predicting transient dynamics using non-intrusive ROM

Contents

The reduced-order models implemented so far have been derived from the Galerkin projection of the high-fidelity Navier-Stokes equations onto the reduced space spanned by the POD modes, refer Sec. 2.2.2. In this projection-based approach, formally classified as an *intrusive* method, the reduced (POD) coefficients are determined by solving the system of reduced-order equations. However, as seen in Chap. 5 and in literature (Aubry et al., 1988; Iollo, Lanteri, et al., 2000), the intrusive method lacks an *a priori* guarantee of stability and requires modeling of the unresolved modes in some alternate way to achieve numerical stability.

On the other hand, a *non-intrusive* reduced-order model bypasses the Galerkin projection in order to obtain approximate maps between a set of parameter values and the reduced coefficients of the high-fidelity solution in low-dimensional space. In this work, this is realized by regression models based on artificial neural networks (ANN). The later are trained using high-fidelity data in a supervised learning paradigm during an *offline* stage. The learned ANN can then be used in an *online* stage to propagate the estimates of states in time. In this regard, the deep neural network-based data-driven method described in Sec. 3.3 provides a multistep, residual-based and parametrized framework for reduced-order modeling. This serves as a black-box alternative to the system identification methods discussed in the previous chapters.

In this chapter, the neural network-based non-intrusive ROM presented in Sec. 3.3.1, hereafter called NN-ROM, is applied for time series prediction of POD modal coefficients. In Sec. 6.1, the ability of the multistep, residual-based NN-ROM framework to take into consideration the memory effect is demonstrated over the Lorenz-63 dynamical system. In Sec. 6.2, the NN-ROM is constructed on a parameter space, characterizing different numerical cylinder wake flows obtained at different Reynolds numbers, and used to interpolate and extrapolate the dynamics corresponding to a Reynolds number not provided during the training. Also, the sequential NN-ROM is augmented with the EnKF data assimilation algorithm to provide long-term predictions. In Sec. 6.3, the EnKF augmented NN-ROM is applied to obtain predictions of the dynamics of a cylinder wake flow at high Reynolds number. Finally, concluding remarks are given in Sec. 6.4.

6.1 Validation using a toy model

Before implementing the proposed framework on fluid flow applications, the capability of NN-ROM to model nonlinear dynamical systems is demonstrated using a toy model. Specifically, the NN-ROM will be trained and evaluated for the highly nonlinear and chaotic Lorenz-63 system.

6.1.1 Training and model-evaluation dataset

The equations along with the parameter values and initial condition introduced in Sec. 4.3 are used to obtain the discrete state evolution data. The size of the state vector is $N_r = 3$. For learning NN-ROM, the data is generated in the time range $t = [0, 20]$ with a time step of 0.01, giving $N_t = 2001$ snapshots. The dataset is split such that state evolution corresponding to the initial 85% of the time steps is used for training, *i.e.* $N_t^{\text{Train}} = 1700$, and the remaining data is used for validation, *i.e.* $N_t^{\text{Val}} = 301$. At the end of the learning stage, the performance of the trained model is evaluated based on the reproduction capability with respect to the full solution trajectory, *i.e.* $N_t^{\text{Test}} = 2001$.

Note that as only the ability of the neural network framework to provide state estimates is evaluated in this section, the toy model defined by a single parameter set is considered, *i.e.* $N_p^{\text{Train}} = 1$. As such, the parametrized framework of NN-ROM will not be evaluated in this section.

6.1.2 Hyperparameter selection

Training NN-ROM, which is constructed as a feedforward network (see Sec. 3.3.2.2), requires making some design decisions. The performance of the NN-ROM is largely dictated by the selection of hyperparameters used for defining the architecture and the optimization of the neural network. The architecture is defined by the number of hidden layers, the connection between the consecutive layers, and the number of hidden units in each layer. The output of the hidden layer is governed by the choice of the activation function. The gradient-based learning also requires choosing the optimizer and the cost function.

Finding an optimal set of hyperparameters which minimizes the loss function over a hyperparameter space is a challenging task given the substantial number of free parameters involved. Manual search is one of the the most widely used hyperparameter optimization methods. In this work, the hyperparameters have been selected heuristically by monitoring the influence of the assigned values on model performance 'on the fly'. More sophisticated methods for hyperparameter selection are random search (Bergstra and Bengio, 2012) and Bayesian optimization (Brochu et al., 2010). However, following Goodfellow et al. (2016), general guidelines can be formulated. The effect of key hyperparameters on the performance of the NN-ROM are listed below:

1. Number of layers (L): Increasing L augments the model's capacity to represent more complex functions with a simultaneous increase in the computational cost of training.
   ▶ In this example, the network is constructed with $L = 7$ layers, which includes the input and output layers and 5 hidden layers.

2. Number of units in each layer ($n^{[l]}$): Increasing $n^{[l]}$ has the similar effect as L on the representation ability of the model. The number of units in the input and output layers is fixed for a problem based on the size of the feature and target. Note that for a large value of either L or $n^{[l]}$, the model has a chance of overfitting the training data.
   ▶ In this example, the number of units in the input layer is $n^{[1]} = N_r p+1 = 3p+1$, where p is the number of past input steps defining the multistep framework (see Sec. 3.3.1). Note that the '$+1$' unit corresponds to the time variable and no additional units are assigned for parameters as the training is performed using data belonging to the solution set corresponding to the same parameters. The size of the hidden layers is assigned to $n^{[l=2,\dots,6]} = 128$ units. The output layer contains $n^{[7]} = N_r = 3$ units.

3. Activation function (σ): The output of the hidden units is dictated by the activation function. The function σ must be chosen such that the layers do not saturate, *i.e.* the gradients of the cost function do not become very flat and remain large enough to guide the learning algorithm.
   ▶ In this example, the widely used and easy to optimize rectified linear unit (ReLU) is used as the activation function (see Sec. 3.3.2.1).

4. Size of minibatch (N_b): The stability and speed of the learning algorithm are generally optimized by dividing the input features (here, time and state evolution) and

the target into subsets, the training being made on these minibatches. Minibatches typically contain two to several hundred samples, although for large models the choice may be constrained by computational resources.
▶ In this example, the minibatches are randomly selected disjoint subsets of the training data of size $N_b = 501$.

5. Number of minibatches (N_{mb}): The number of minibatches dictates the number of updates of the training variables during each training epoch. The number is fixed by monitoring the learning curve. A higher value of N_{mb} may lead the model to overfit the training data.
▶ In this example, $N_{\text{mb}} = 10$ is used.

6. Learning rate (α): The success of the gradient-based optimization depends on the choice of the learning rate. A small α slows down the computation while a large α may lead to overshooting the local minima of the cost function. The value of α is fixed by monitoring the learning curve.
▶ In this example, the learning rate $\alpha = 0.001$ is used.

7. Regularization parameter (λ): The generalization error of the NN-ROM is tuned by the regularization parameter in the cost function. A lower value of λ may make the model more prone to overfit to the training data, while a higher λ may cause the weights of the neural network to vanish, severely underfitting the training data. The value of λ is fixed by monitoring the learning curve.
▶ In this example, the regularization parameter $\lambda = 1.0 \times 10^{-5}$ is used.

8. Number of training epochs (N_{Epochs}): The number of training epochs is related to the learning rate. As N_{Epochs} increases, the model transforms from underfitting, to optimal, and to potentially overfitting the training data. The value of N_{Epochs} is fixed by monitoring the learning curve.
▶ In this example, the model is trained over $N_{\text{Epochs}} = 20000$ cycles.

The choice of these hyperparameters is summarized in Tab. 6.1.

Table 6.1: Hyperparameters of NN-ROM for the Lorenz-63 system.

DNN architecture ($p = 1, 5, 10$)				Training data		Minibatch		Optimization parameters		
$n^{[1]}$	$n^{[2,\ldots,6]}$	$n^{[7]}$	σ	N_t^{Train}	N_p^{Train}	N_b	N_{mb}	α	λ	N_{Epochs}
$3p+1$	128	3	ReLU	1700	1	501	10	1.0×10^{-3}	1.0×10^{-5}	20000

6.1.3 Online training

Once the hyperparameters are assigned, the training variables of NN-ROM (weights and biases) are optimized over the N_{Epochs} training epochs using the Adam algorithm (see Sec. 3.3.2.4). One training epoch involves the evaluation of the gradients of the cost function (3.102) for each minibatch by backpropagation (see Sec. 3.3.2.3). The deep neural network utilizes high-capacity architecture which, due to a large number of training variables, are susceptible to overfitting even when sufficient training data is available. This behavior of the model is monitored during the course of training using

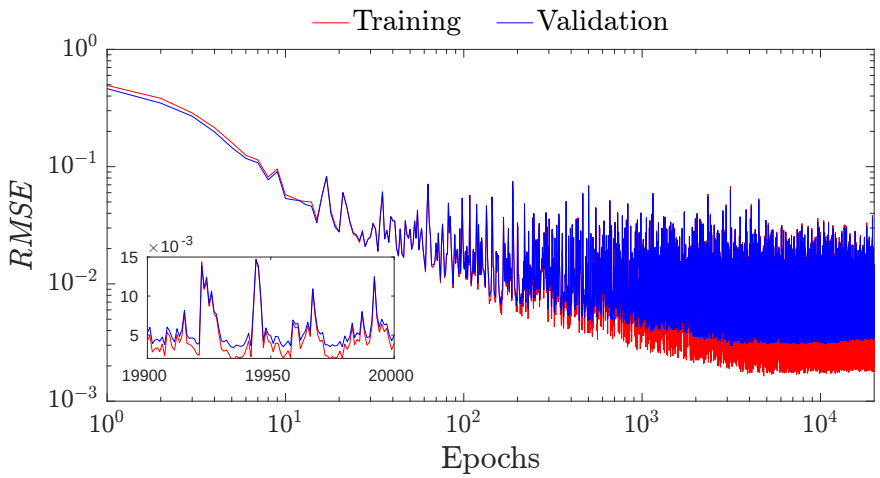


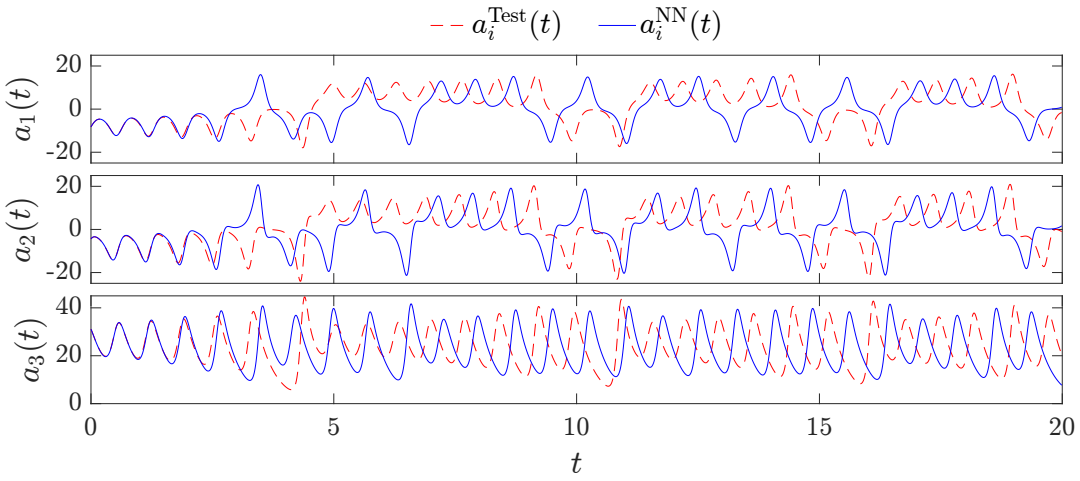
Figure 6.1: Evolution of the training and validation errors during the training epochs of NN-ROM for the Lorenz-63 system ($p = 10$). The error evolution in the last 100 epochs is also shown in the inset.

the learning curves which are plots of the error metric (3.111) calculated for the training set ($RMSE_{\text{Train}}$) and the validation set ($RMSE_{\text{Val}}$) as a function of the training epochs.

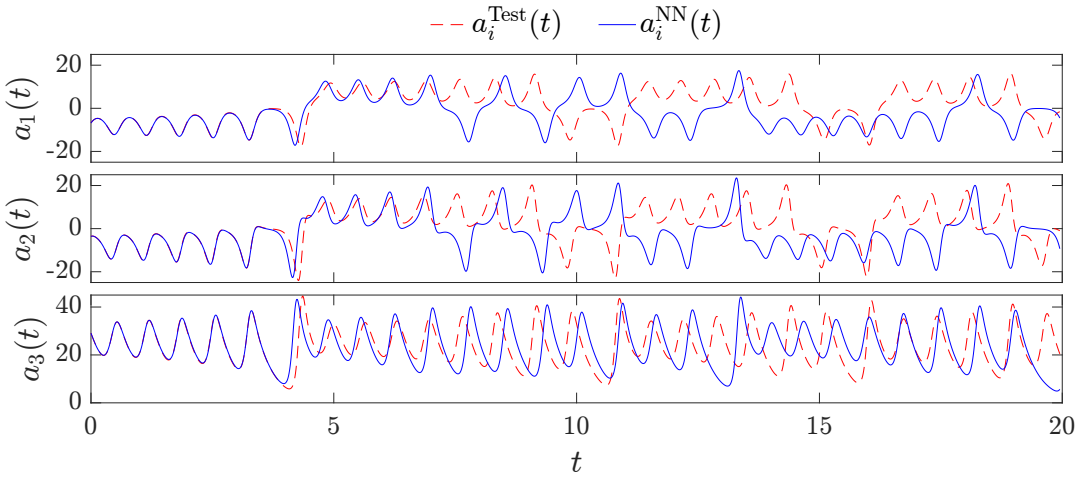
In order to demonstrate the influence of the number of past time steps p in the input on the performance of NN-ROM, three values of p are considered, namely $p = 1, 5, 10$. The learning curves for the training of NN-ROM for the Lorenz-63 system with $p = 10$ is shown in Fig. 6.1. Similar curves are obtained for other values of p. It is observed that the error for the validation set drops consistently with the corresponding decrease in the error for the training set. This implies that the hyperparameters selected for designing and training the neural network lead to a NN-ROM with a low generalization error, *i.e.* the model neither underfits not overfits the data used for training. Note that the noisy learning curve is common to the stochastic gradient-descent methods which can be attributed to the frequent updates of the training variables, allowing the model to avoid local minima and hence, an early convergence.

6.1.4 Offline prediction

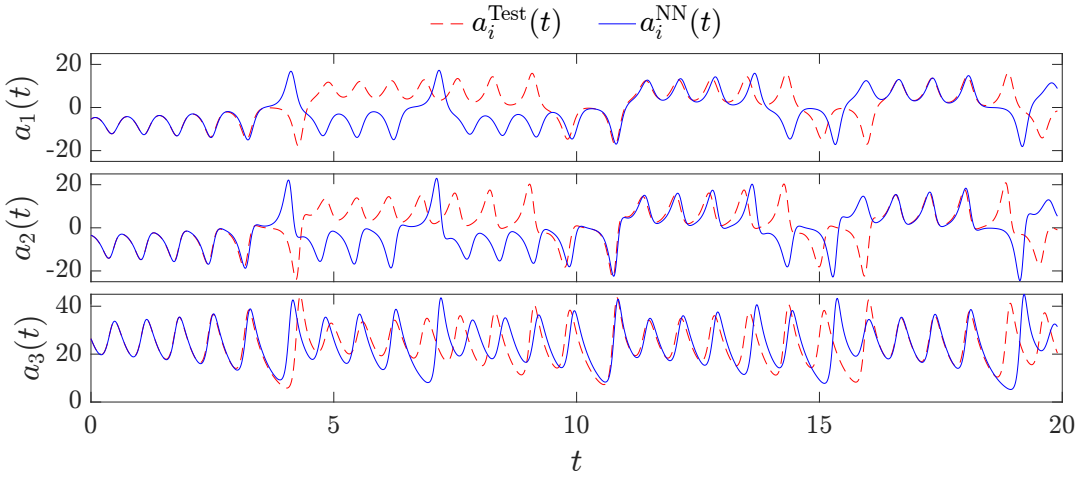
To elucidate the multistep, residual-based framework of the neural network, the learned NN-ROM models, corresponding to the three values of p in the input training data, are used to obtain the estimation of the state trajectories $\boldsymbol{a}^{\text{NN}}(t)$ of the Lorenz-63 system using the forward model (3.90). The comparison of the estimated trajectories with the reference solution trajectories is shown in Fig. 6.2. The Lorenz system has a chaotic behavior. Hence, a small error in the estimated state of the system can lead to a larger error in the subsequent time steps. The time period for which the reproduced trajectory is the same as the reference trajectory varies for different values of p. In general, the inclusion of temporal history of the state of the system in the input to NN-ROM, *i.e.* $p > 1$, leads to an improved estimation where the reference trajectory is followed over a longer time span.



(a) $p = 1$



(b) $p = 5$



(c) $p = 10$

Figure 6.2: Evolution of the states $\boldsymbol{a}(t)$ of the Lorenz-63 system obtained from the trained NN-ROM in the testing window corresponding to different numbers of past input steps (p) and compared with the true reference trajectories.

Table 6.2: Time averaged $NRMSE$ in the testing window corresponding to different numbers of past input steps (p) for the Lorenz-63 system.

	$p = 1$	$p = 5$	$p = 10$	$p = 100$
Average $NRMSE$	0.7524	0.5481	0.4905	0.6282

The performance of NN-ROM is quantified in terms of the normalized root-mean-square error ($NRMSE$). Here, the $NRMSE$ is calculated with respect to the reference solution $\boldsymbol{a}^{\text{Ref}}(t)$, which is the solution obtained from the numerical integration of the Lorenz-63 system, as

$$NRMSE(t) = \frac{\sqrt{\sum_{i=1}^{N_r}(a_i^{\text{NN}}(t) - a_i^{\text{Ref}}(t))^2}}{\sqrt{\sum_{i=1}^{N_r}(a_i^{\text{Ref}}(t))^2}}, \tag{6.1}$$

where $N_r = 3$. The averaged $NRMSE$ over the testing time span is given in Tab. 6.2 for the three values of p. An extreme case where $p = 100$ time steps has been considered is also included to observe the effect of a long time past window on the accuracy of the estimation. It is observed that the estimation is more accurate for the cases with $p > 1$. This result encourages to take the memory effect into account in the input of NN-ROM for a more accurate reproduction of the transient dynamics. The effect is however not monotonous (*i.e.* higher number of time steps p does not correspond necessary to a lower value of error), as indicated by the higher value of estimation error corresponding to $p = 100$ as compared to $p = 10$. This is exacerbated by the sensitivity of the Lorenz-63 system to the state space, *i.e.* featuring a positive Lyapunov exponent. The number of past time steps (p) is thus a hyperparameter which can be optimized to obtain a sufficiently accurate estimation framework. However, in the subsequent test cases, this parameter has been selected heuristically.

6.2 Test case 1: Parametrized reconstruction of a cylinder wake flow

In the previous section, the multistep, residual-based framework of NN-ROM has been evaluated for a toy dynamical model. In this section, we focus on the parametrized framework of the non-intrusive NN-ROM. For this, we evaluate the reconstruction of a cylinder wake flow dynamics obtained numerically at low Reynolds numbers ($Re \in [100, 210]$).

6.2.1 Training and testing dataset

Numerical simulations are performed using the same setup as described in App. A for a set of nine, non-equally spaced Reynolds numbers, namely $Re_{\text{Set}} = \{100, 110, 120, 130, 140, 160, 170, 190, 210\}$. From the post-transient, periodic vortex shedding regime, 1001 snapshots of two-component velocity fields are sampled for each Reynolds number in the time span $t \in [0, 100]$ with time step of $\Delta t = 0.1$.

Table 6.3: Values of the Reynolds number (Re) considered in the training and testing sets for the two cases representing the interpolation and extrapolation problem.

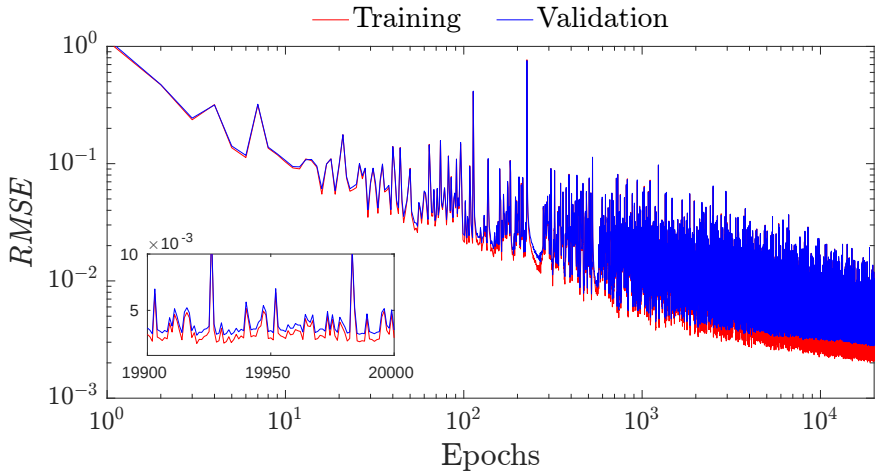
	Problem	Training parameter set	Test parameter
Case-I	Interpolation	$\{100, 110, 120, 130, 140, 170, 190\}$	160
Case-E	Extrapolation	$\{100, 110, 120, 130, 140, 160, 170, 190\}$	210

Snapshot POD of the fluctuating velocity flow fields is performed to obtain the reduced basis. The detail for building the POD basis is the same as discussed in Sec. 5.2.1. For the set of Reynolds numbers considered, similar eigenspectra with paired modes are obtained. In each case, the most dominant, first $N_r = 10$ modes are preserved for reduced-order modeling, which together represent more than $\sim 99.95\%$ of the total energy of the respective flows. The input features and targets for training the NN-ROM are subsets of the calculated temporal POD coefficients $\boldsymbol{a}^{\mathrm{POD}}(t_k; \mu_j)$, where $\mu_j \in Re_{\mathrm{Set}}$ is the parameter characterizing the dynamics. The POD modes $\boldsymbol{\Phi}^{\mathrm{POD}}(\boldsymbol{\chi}; \mu_j)$ are also conserved for reconstruction of the flow fields.

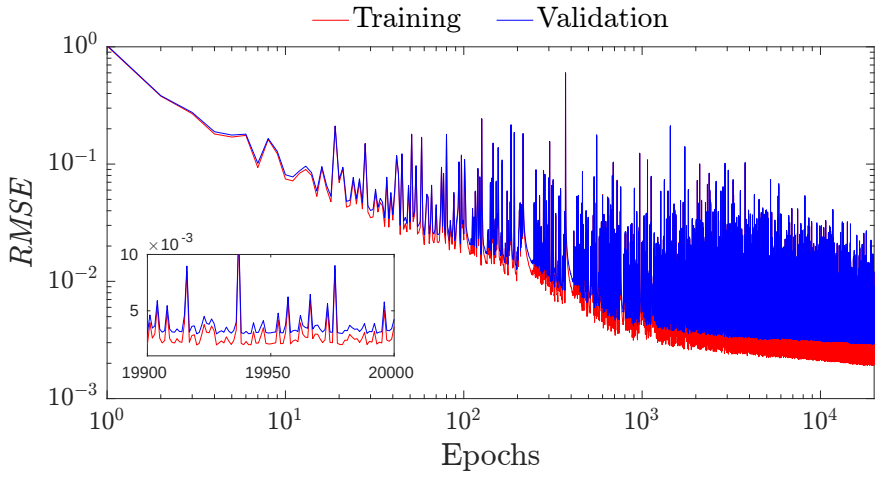
To evaluate the performance of the NN-ROM for parametric estimation, interpolation and extrapolation problems have been considered. These problems are denoted as Case-I and Case-E, respectively. The corresponding training-testing split of parameters is given in Tab. 6.3. The POD coefficients corresponding to several unique Reynold numbers constitute the training set and are used to learn the NN-ROM. For the interpolation problem (Case-I), $N_p^{\mathrm{Train}} = 7$ parameters are considered in the training set. In this case, the test parameter is chosen as $Re = 160$ such that it lies within the range of Reynold numbers considered in the training parameter set. On the other hand, for the extrapolation problem (Case-E), $N_p^{\mathrm{Train}} = 8$ parameters are considered in the training set. Here, the selected test parameter ($Re = 210$) lies outside the range of the training parameter set. In both cases, data (*i.e.* POD coefficients) from the initial 501 time steps for each parameter is used for learning, out of which $N_t^{\mathrm{Train}} = 425$ snapshots is used for training and $N_t^{\mathrm{Val}} = 76$ snapshots is used for validation of the learned model.

6.2.2 NN-ROM architecture and optimization

The NN-ROM is constructed in a similar manner as for the toy model considered in Sec. 6.1. The hyperparameters associated with the construction and training of the neural network are assigned manually. The major difference appears in the input layer as, apart from the units for time t_k (for all $k = 1, \ldots, N_t^{\mathrm{Train}}$) and POD coefficient $\boldsymbol{a}^{\mathrm{POD}}(t_k; \mu_j)$, the input feature set contains an additional unit for the parameters μ_j (for all $j = 1, \ldots, N_p^{\mathrm{Train}}$). Note that here μ is scalar, *i.e.* only the Reynolds number is the parameter. The total number of units in the input layer is therefore $n^{[1]} = N_r p + 2 = 10p + 2$. The same DNN architecture and optimization parameters have been used for learning the model for both the interpolation and extrapolation problems. The number of units in the hidden layers is fixed as 256 to accommodate large range of dynamics exhibited by the POD coefficients. In this demonstration, history of POD coefficients from previous $p = 5$ time steps is used in the input feature set. A summary of the hyperparameters is given in Tab. 6.4. Refer to Sec. 6.1.2 for a brief discussion on the influence of each hyperparameter on the performance of NN-ROM.



(a) Case-I - Interpolation problem



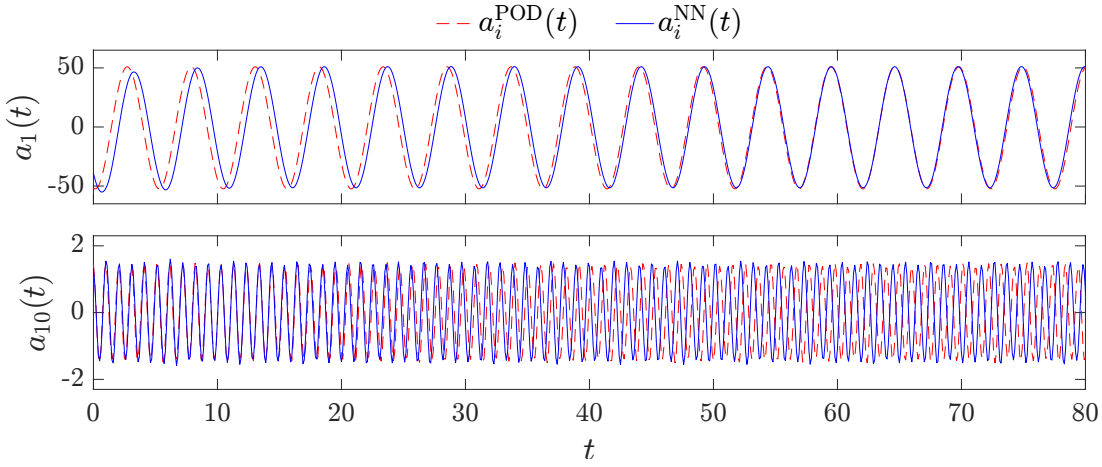
(b) Case-E - Extrapolation problem

Figure 6.3: Evolution of the training and validation errors during the learning epochs of NN-ROMs. (a) interpolation, and (b) extrapolation problems for the numerical cylinder wake flow. The error evolution in the last 100 epochs is also shown in the inset.

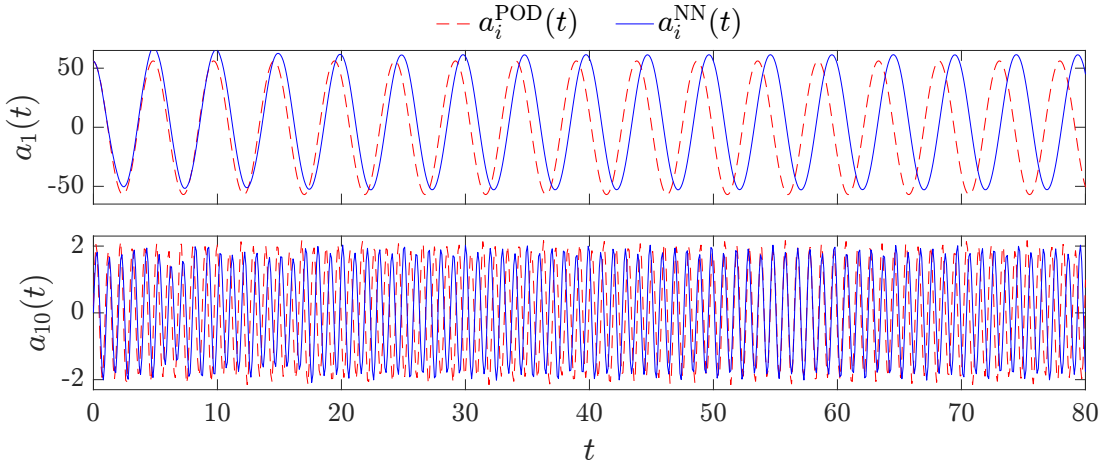
Table 6.4: Hyperparameters of NN-ROM for the numerical cylinder wake flow. Note that $N_p^{\text{Train}} = 7$ for Case-I and $N_p^{\text{Train}} = 8$ for Case-E.

DNN architecture ($p = 5$)				Training data		Minibatch		Optimization parameters		
$n^{[1]}$	$n^{[2,\ldots,6]}$	$n^{[7]}$	σ	N_t^{Train}	N_p^{Train}	N_b	N_{mb}	α	λ	N_{epochs}
$10p+2$	256	10	ReLU	501	7 or 8	501	50	1.0×10^{-3}	1.0×10^{-5}	20000

The training variables of the NN-ROM with the assigned hyperparameters are optimized over the training epochs using the Adam algorithm. The learning curves for the training stage is shown in Fig. 6.3. For both the interpolation and extrapolation problems, the $RMSE$ for training and validation sets are observed to decrease with the number of epochs, as expected. After 10^4 epochs, the $RMSE$ is found acceptable.



(a) Case-I - Interpolation problem, $Re = 160$



(b) Case-E - Extrapolation problem, $Re = 210$

Figure 6.4: Prediction by NN-ROM of the temporal POD coefficients $a_1(t)$ and $a_{10}(t)$ for the test dataset of the cylinder wake flow at $Re = 160$ for the interpolation problem (a), and $Re = 210$ for the extrapolation problem (b). Comparison with the reference trajectory.

6.2.3 Estimation of POD coefficients using trained NN-ROM

The trained NN-ROM is used to obtain with the forward model (3.90) the estimates of the POD coefficients $\boldsymbol{a}^{\mathrm{NN}}(t_k; \mu_j)$ for the interpolation and extrapolation problems. The comparison of the trajectories of the coefficients $a_1(t)$ and $a_{10}(t)$ with the reference trajectory obtained from POD is shown in Fig. 6.4. It is observed that NN-ROM is able to provide fairly accurate estimates of the dynamics of the POD coefficients corresponding to $Re = 160$ for the interpolation problem (Case-I) and $Re = 210$ for the extrapolation problem (Case-E). In Case-I, both the amplitude and phase behavior of the coefficients are preserved for a long time span. On the other hand, the extrapolated estimation of the POD coefficient in Case-E is accurate in a smaller time span. A noticeable phase difference is observed between the predicted and reference trajectories over longer time span. However, NN-ROM performs well in capturing the amplitude for a long time span for all $N_r = 10$ POD coefficients for both the testing sets.

The performance of NN-ROM is quantified in terms of the normalized root-mean-

Table 6.5: Time averaged $NRMSE$ for the test parameter for the interpolation (Case-I) and extrapolation (Case-E) problems for the numerical cylinder wake flow.

	Case-I	Case-E
Average $NRMSE$	0.4401	0.9827

square error ($NRMSE$). Here, the $NRMSE$ is calculated with respect to the POD coefficients $\boldsymbol{a}^{\mathrm{POD}}(t)$ using (6.1). The time averaged value of $NRMSE$ is given in Tab. 6.5. The NN-ROM provides more accurate estimates for the interpolation problem as compared to the extrapolation problem. This implies that the estimation benefits when the test parameter lies within the range of the parameter space used for training. However, the error in both the cases remain of the same order of magnitude as the evolution remains bounded over a long time range.

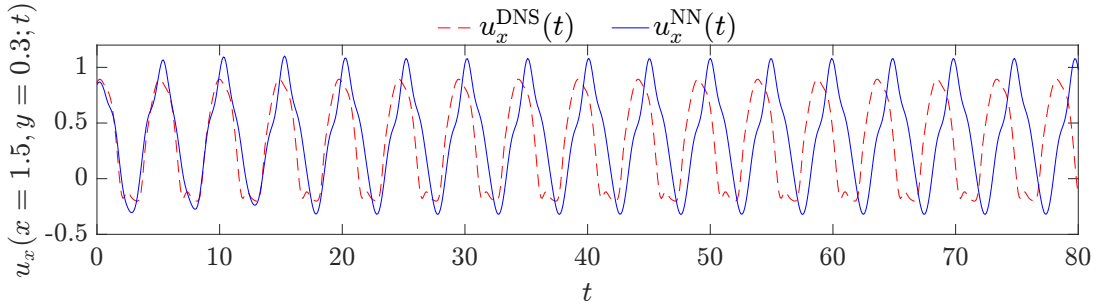
With the NN-ROM framework, the estimation of the POD coefficients has been obtained by bypassing the computation of the reduced system through Galerkin projection. The non-intrusive nature of the NN-ROM and the lack of system of equations that needs to be solved for estimating the solution makes it an attractive alternative to the POD-ROM implementation.

The proposed parametrized framework has been shown to provide fairly accurate short-term estimates for the interpolated and extrapolated parameters outside the training set. However, a phase shift is observed over the long time range for the extrapolation problem. The observable deviation between the estimated and reference dynamics can be a result of the limited number of parameters ($N_p^{\mathrm{Train}} = 8$) available for training the NN-ROM. However, the predictions in a short time span and the overall preservation of the amplitude encourage the use of data assimilation algorithms as demonstrated in Sec. 5.2.4 to modify the predicted trajectory over long term estimation. This approach is demonstrated in the next section for flow field reconstruction over a long time range.

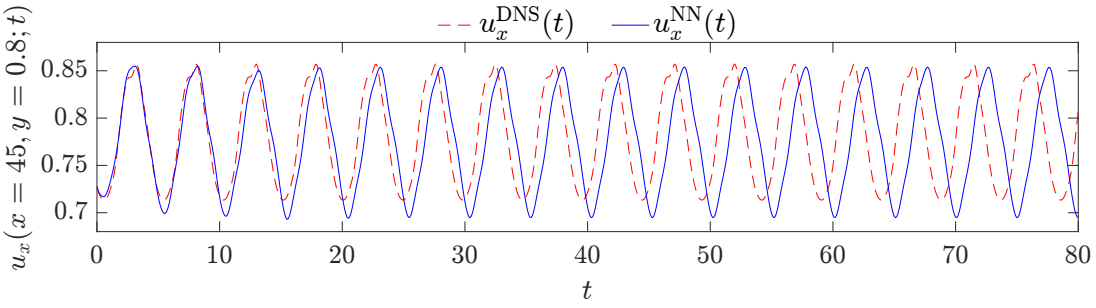
6.2.4 Flow field reconstruction using EnKF augmented NN-ROM estimation

In this section, we introduce an ensemble Kalman filter (EnKF) augmented version of the NN-ROM method presented in Sec. 6.3.1. Hereafter, this method is referenced as NN-ROM-DA. In the following, we comparatively analyze the performance of the standard NN-ROM and the NN-ROM-DA method in the case of the extrapolation problem (Case-E) at $Re = 210$.

First, the flow field is reconstructed from the NN-ROM estimates. The extrapolated POD coefficients obtained by NN-ROM and the spatial POD modes at $Re = 210$ are combined to reconstruct with (2.15) the two-component velocity fields. In Fig. 6.5, the time evolution of the estimated streamwise component of velocity $u_x^{\mathrm{NN}}(t)$ at locations in the near-wake and farfield region are compared with the corresponding reference velocities obtained from the numerical simulation $u_x^{\mathrm{DNS}}(t)$. The estimated velocity accurately captures the initial dynamics over a short time span but a phase shift is observed over longer time range.



(a) Probe location $(x, y) = (1.5, 0.3)$



(b) Probe location $(x, y) = (45, 0.8)$

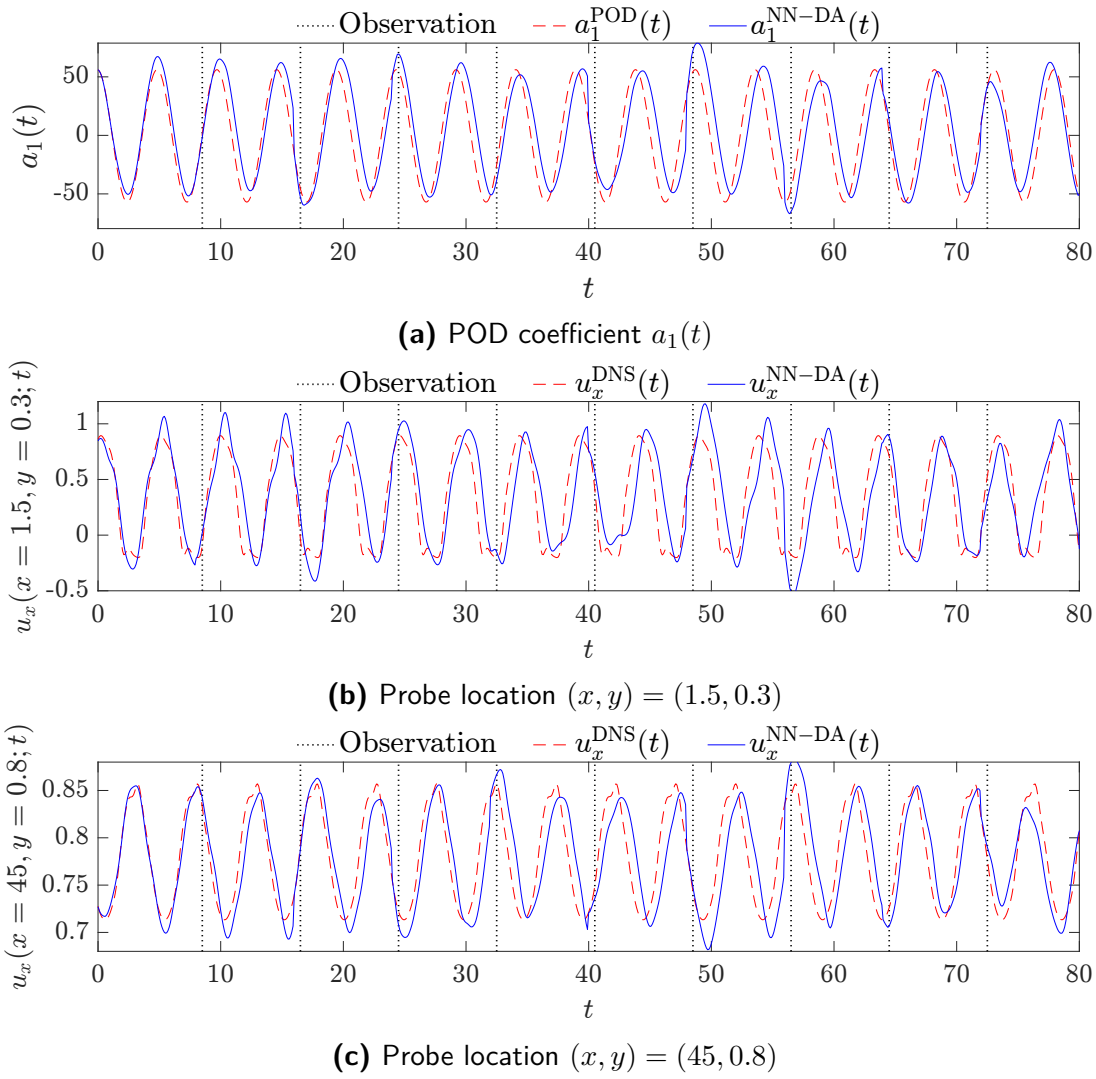
Figure 6.5: Time evolution of the streamwise velocity component at near-wake (a), and farfield (b) locations in the flow field, comparing the trajectory obtained from the numerical simulation ($u_x^{\mathrm{DNS}}(t)$) with the reconstructions using the POD coefficients obtained from the NN-ROM ($u_x^{\mathrm{NN}}(t)$) for the cylinder wake flow at $Re = 210$.

Next, the NN-ROM-DA estimates used to obtain an improved reconstruction of the flow field is considered. The introduced formulation is briefly described before presenting the results. As already discussed, the sequential state evolution provided by the NN-ROM (3.90) is a viable alternative to the POD-ROM as a forward model. The estimates provided by the imperfect learned NN-ROM can be combined with heterogeneous observations in the data assimilation paradigm (see Sec. 3.2). Specifically, the learned NN-ROM can be used as the forward model (3.56) in the EnKF framework. This dynamical model is given as

$$\boldsymbol{a}^{f,(n)}(t_{k+1};\boldsymbol{\mu}_j) = \boldsymbol{a}^{a,(n)}(t_k;\boldsymbol{\mu}_j) + \boldsymbol{f}^{\mathrm{NN}}(\{\boldsymbol{a}^{a,(n)}(t_l;\boldsymbol{\mu}_j)\}_{l=k-p+1}^{k};\boldsymbol{W},\boldsymbol{b}) + \boldsymbol{\eta}_{k+1}^{(n)}, \quad (6.2)$$

where $n = 1,\ldots,N_e$ is the index of the sample in the ensemble. The superscripts 'f' and 'a' represent the forecast and analyzed POD coefficients. $\boldsymbol{\eta}_{k+1}^{(n)} \sim \mathcal{N}(\boldsymbol{0},\boldsymbol{Q}_{k+1})$ is the Gaussian representation of the model error with zero mean and time-dependent covariance $\boldsymbol{Q}_{k+1}$. Using the EnKF algorithm given by Alg. 3.2, the state estimates trajectory can be updated sequentially whenever observations are available.

The same measurement configuration (number of probes and location) as discussed in Sec. 5.2.3.2 and shown in Fig. 5.10 is used to obtain observations of streamwise velocity component at fixed intervals. The observations are assimilated at time steps of size $\Delta t_o = 80\Delta t = 8$, *i.e.* 80 times larger than the simulation time step. This corresponds to performing 9 equispaced assimilations in the span of the long time range window defined over 800 time steps. The time evolution of the POD coefficient $a_1(t)$ using the NN-



(a) POD coefficient $a_1(t)$

(b) Probe location $(x, y) = (1.5, 0.3)$

(c) Probe location $(x, y) = (45, 0.8)$

Figure 6.6: Time evolution of the POD coefficient $a_1(t)$ (a), and the streamwise velocity components at near-wake (b) and farfield (c) locations in the flow field. For (a), a comparison is made with the POD coefficients. For (b) and (c), we compare the trajectory obtained from the numerical simulation ($u_x^{\mathrm{DNS}}(t)$) with the reconstructions using the POD coefficients obtained from the NN-ROM-DA ($u_x^{\mathrm{NN-DA}}(t)$). Case of the cylinder wake flow at $Re = 210$.

ROM-DA formulation is compared with the values obtained directly from the snapshots in Fig. 6.6a. The plots show that the estimated trajectory of the POD coefficient obtained using NN-ROM-DA follows the reference trajectory more accurately over a longer time range as compared to the estimate obtained from NN-ROM (see Fig. 6.4b). The estimated streamwise component of velocity $u_x^{\mathrm{NN-DA}}(t)$ at locations in the near-wake and farfield region are also compared with the corresponding reference velocities obtained from the numerical simulation $u_x^{\mathrm{DNS}}(t)$ in Fig. 6.6b and Fig. 6.6c. Comparing with the results shown in Fig. 6.5, it can be observed that after each observation step, the assimilation procedure improves the accuracy of the trajectories in the time span between the observations.

In order to quantify the performance of the different approaches, the normalized

Table 6.6: Time averaged $NRMSE$ of the velocity magnitude $|\boldsymbol{u}|$ for the extrapolation problem corresponding to $Re = 210$.

	POD	NN-ROM	NN-ROM-DA
Average $NRMSE$	4.836×10^{-3}	1.545×10^{-1}	8.307×10^{-2}

root-mean-square error of the velocity magnitude $|\boldsymbol{u}|$ with respect to the DNS values is calculated over the prediction time span. This error is defined as

$$NRMSE(t) = \frac{\sqrt{\sum_{i=1}^{N_\chi}(|\boldsymbol{u}^{\mathrm{NN}}(\boldsymbol{\chi}_i, t)| - |\boldsymbol{u}^{\mathrm{DNS}}(\boldsymbol{\chi}_i, t)|)^2}}{\sqrt{\sum_{i=1}^{N_\chi}(|\boldsymbol{u}^{\mathrm{DNS}}(\boldsymbol{\chi}_i, t)|)^2}}. \tag{6.3}$$

In Tab. 6.6, the averaged $NRMSE$ over the testing time window obtained for the NN-ROM and NN-ROM-DA methods are compared to the results obtained directly by POD. The error corresponding to the NN-ROM-DA is one order of magnitude higher than the error corresponding to the POD estimation but lower than the error corresponding to NN-ROM (*i.e.* without assimilation). These values imply that a more accurate estimation is obtained from NN-ROM-DA than NN-ROM in the testing time span. It highlights the benefit of augmenting NN-ROM with data assimilation.

6.3 Test case 2: Long-term estimation of an experimental cylinder wake flow

In this section, we consider a cylinder wake flow configuration at high Reynolds number. This configuration is much more challenging in terms of dynamical complexity than the low Reynolds number wake considered in Sec. 6.2. Thereafter, as we have already done in Sec. 6.2.4, we combine sequential updates obtained by NN-ROM and data assimilation.

6.3.1 NN-ROM setup and estimation

The case of the cylinder wake flow at $Re = 1.5 \times 10^4$ introduced in Sec. 5.3 is considered. POD is performed on the snapshots of velocity field obtained using PIV as discussed in Sec. 5.3.2. The most dominant $N_r = 10$ modes, which represent 79% of total energy, are preserved for reduced-order modeling.

The initial 701 (over 1001) time steps of the POD coefficients are used for learning NN-ROM. The time series is split such that the data belonging to the first $N_t^{\mathrm{Train}} = 595$ snapshots is used for training and that belonging to the next $N_t^{\mathrm{Val}} = 106$ snapshots is used for validation of the learned model. The summary of the manually assigned hyperparameters associated with the construction and training of the neural network is given in Tab. 6.7. The NN-ROM is based on using the history of POD coefficients from previous $p = 10$ time steps in the input feature set. Note that as the dataset is characterized by a single parameter, the unit in the input layer corresponding to parameters in the feature set is dropped. The details regarding the rest of the setup and optimization of the model are the same as the ones discussed in Sec. 6.1.2.

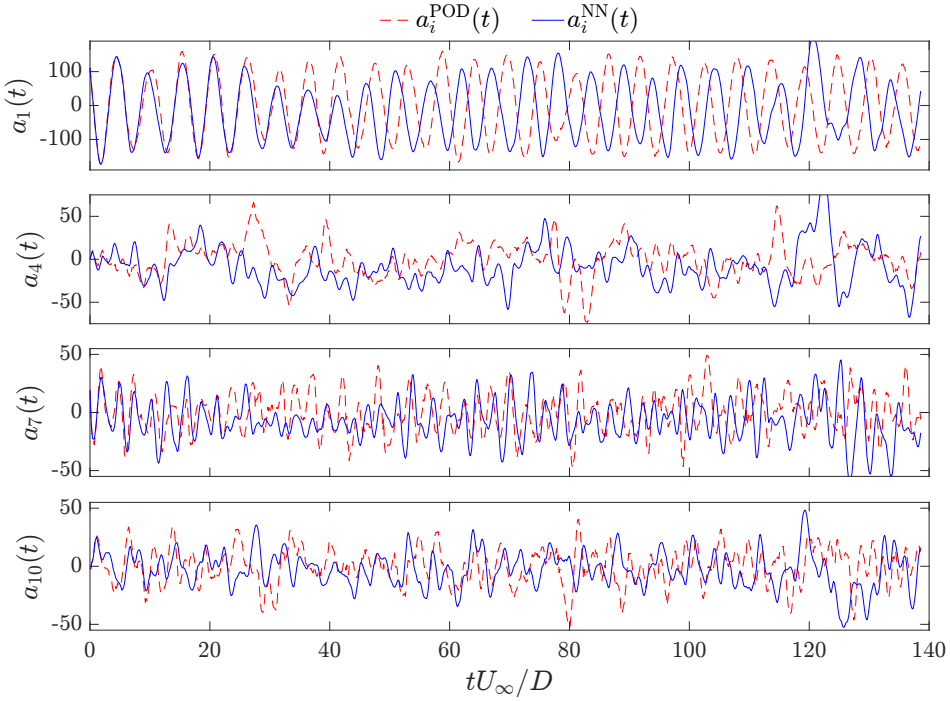


Figure 6.7: Evolution of the temporal POD coefficients $a_i(t)$ $(i = 1, 4, 7, 10)$ for the testing dataset for the cylinder wake flow at $Re = 1.5 \times 10^4$. The results are obtained from the NN-ROM and compared with the reference trajectory.

Table 6.7: Hyperparameters of the NN-ROM for the cylinder wake flow at $Re = 1.5 \times 10^4$.

DNN architecture ($p = 10$)				Training data		Minibatch		Optimization parameters		
$n^{[1]}$	$n^{[2,\ldots,6]}$	$n^{[7]}$	σ	N_t^{Train}	N_p^{Train}	N_b	N_{mb}	α	λ	N_{Epochs}
$10p+1$	256	10	ReLU	595	1	401	10	1.0×10^{-3}	1.0×10^{-2}	20000

The learned NN-ROM is then used to obtain estimates of the POD coefficients. Due to limited size of data, the testing is performed on the whole time range of POD coefficients, $N_t^{\mathrm{Test}} = 1001$. The evolution of the temporal POD coefficients for the testing dataset is shown in Fig. 6.7. Overall, the amplitudes of the estimated coefficients remain consistent with respect to the reference trajectory and the evolution remains bounded over the full testing time span. It is observed that in the short-term, the trajectory of the POD coefficients estimated from NN-ROM follows the reference trajectory of the POD coefficients directly obtained from the snapshots. However, the trajectories start to deviate after about the first 20 time units.

6.3.2 EnKF augmented NN-ROM estimation

The NN-ROM learned in Sec. 6.3.1 is augmented with EnKF data assimilation. The reader is referred to the discussion of the previous test case (see Sec. 6.2.4) for a brief discussion on the NN-ROM-DA formulation. The same measurement configuration (number of probes and location) as discussed in Sec. 5.3.4 and shown in Fig. 5.24 is used to obtain observations of streamwise velocity component at fixed intervals.

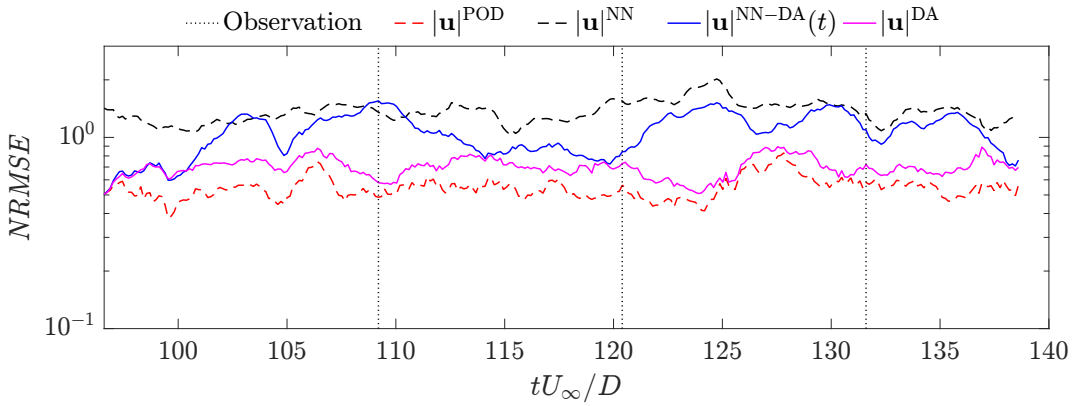


Figure 6.8: Time evolution of the $NRMSE$ with respect to the velocity magnitude in the testing window for the NN-ROM, NN-ROM-DA and DA estimates. Comparison with the results obtained with the POD coefficients. Cylinder wake flow configuration at $Re = 1.5 \times 10^4$.

Before discussing the results in detail, the performance of the non-intrusive NN-ROM-DA approach is briefly compared with the intrusive data assimilation approach of Chap. 5. This is done by evaluating for the estimated data the normalized root-mean-square given by (6.3) in the same time window ($t \in [97.9, 140.0]$) as the one used in assimilation (see Sec. 5.3). In Fig. 6.8, we compare the errors obtained from the NN-ROM, NN-ROM-DA and DA estimates with those obtained from the calculated POD coefficients. Initially, the error magnitudes for the NN-ROM-DA and DA estimates are observed to be the same. Then, for a convective time approximately equal to 100, the error corresponding to the NN-ROM-DA estimates increases sharply. However, the magnitude stays bounded and remains of the same order as that of the DA estimates. The key observation is that a more accurate estimate is obtained from NN-ROM-DA which implies that NN-ROM benefits from the EnKF augmented approach.

Next, an additional hyperparameter, p^{DA}, is introduced which is defined as the number of consecutive observations used in assimilation. The justification of this multistep assimilation approach is to augment the capability of the NN-ROM to take into consideration the memory effect of the temporal dynamics. In this case study, two values of p^{DA} are considered, namely $p^{\mathrm{DA}} = 1$, which corresponds to the classical single update method, and $p^{\mathrm{DA}} = 5$. Note that both are smaller than the number of time steps passed as the input to NN-ROM, $p = 10$.

For the DA augmented approach, the observations are assimilated at time steps of size $\Delta t_o U_\infty/D = 28.1$. This value is 200 times larger than the PIV acquisition step size and spans 5 cycles of vortex shedding (refer to Sec. 5.3.4 for a detailed discussion). This equates to performing 4 steps of equispaced assimilations over the estimation window (1001 time steps).

The time evolution of the temporal POD coefficients obtained by NN-ROM-DA for the two values of consecutive assimilations $p^{\mathrm{DA}} = 1$ and $p^{\mathrm{DA}} = 5$ is shown in Fig. 6.9 and Fig. 6.10, respectively. The plots show that the estimated trajectories of the POD coefficients obtained using NN-ROM-DA follow the reference trajectories more accurately as compared to the estimates obtained from NN-ROM (see Fig. 6.7). As

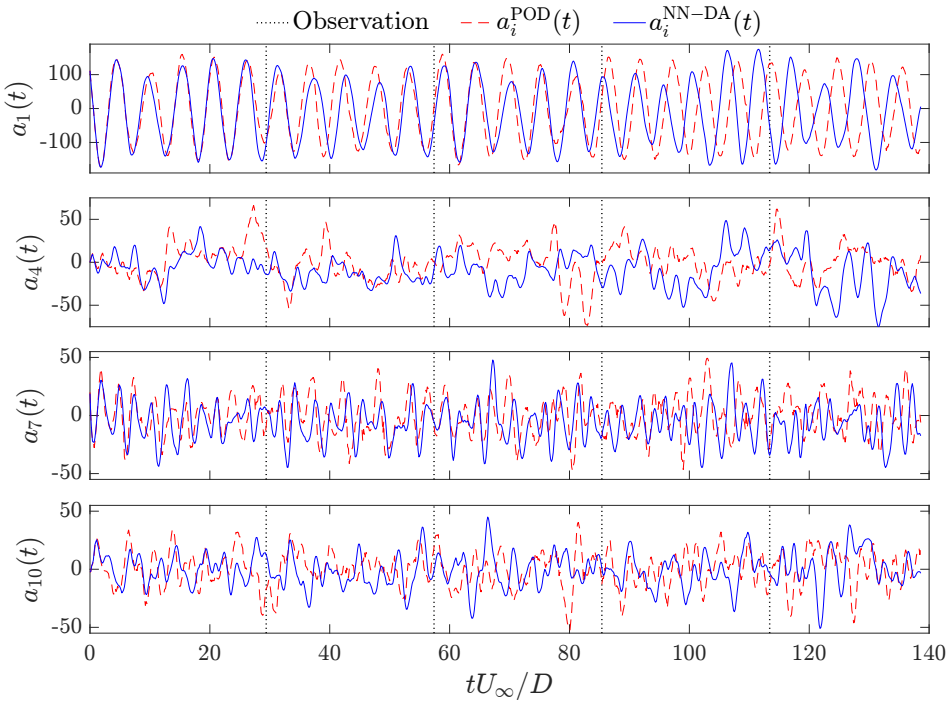


Figure 6.9: Same comparison as in Fig. 6.7 but for the evolution obtained from NN-ROM-DA with $p^{\mathrm{DA}} = 1$ consecutive assimilation.

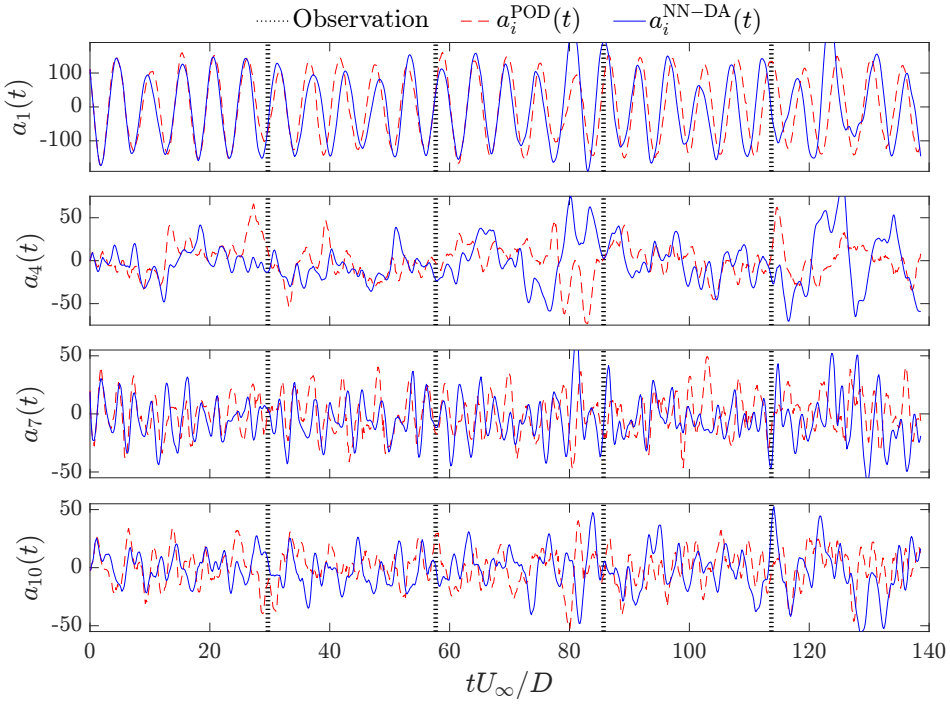


Figure 6.10: Same comparison as in Fig. 6.7 but for the evolution obtained from NN-ROM-DA with $p^{\mathrm{DA}} = 5$ consecutive assimilations.

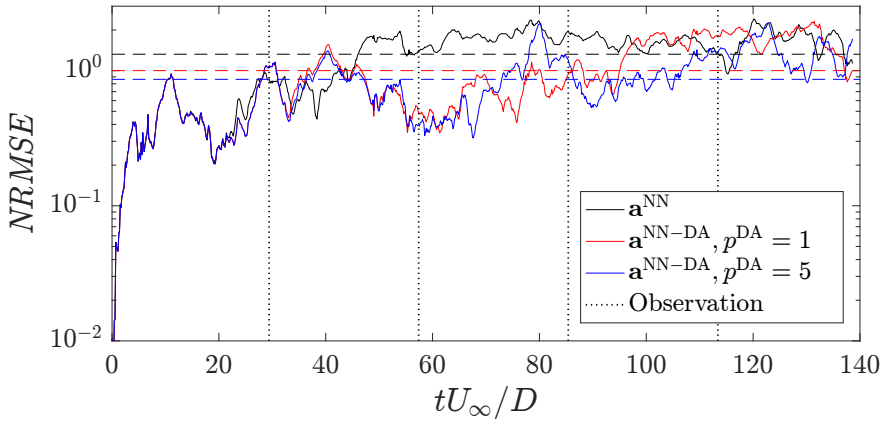
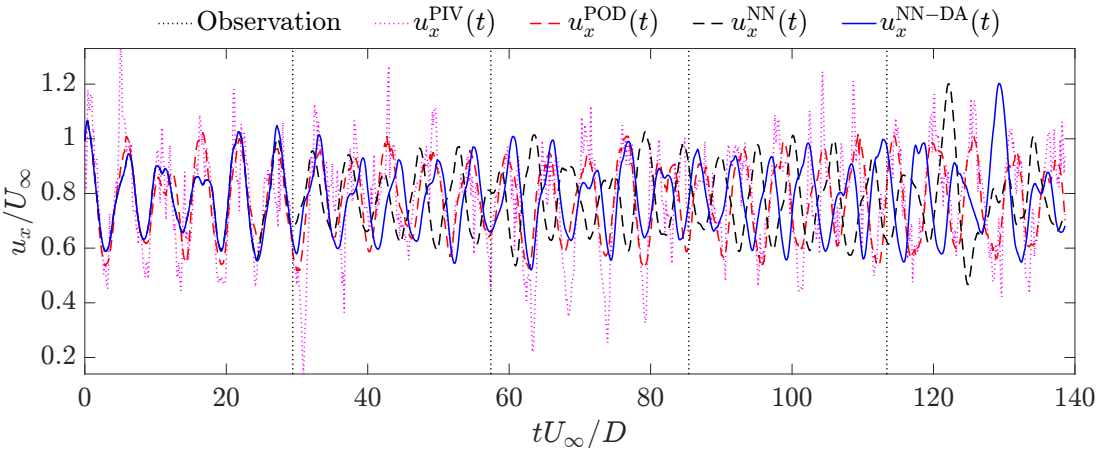


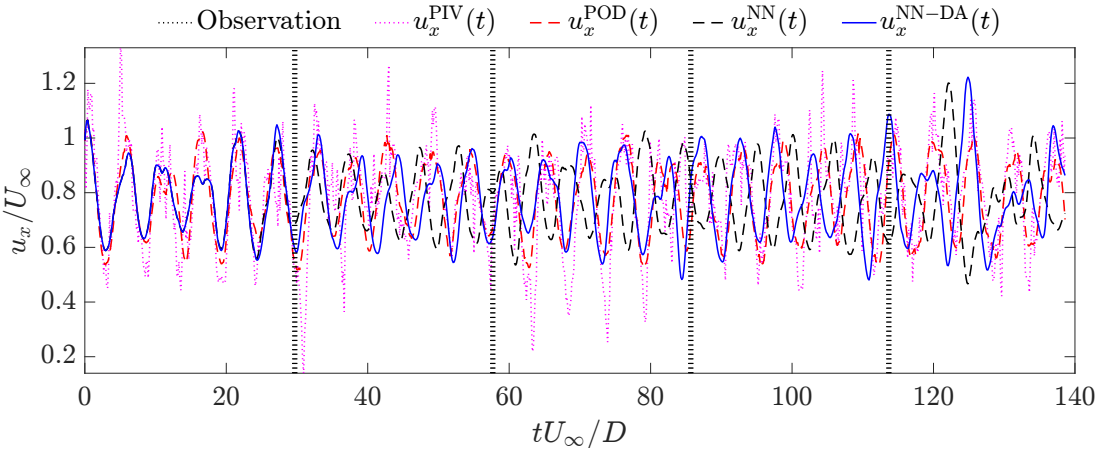
Figure 6.11: Time evolution of $NRMSE$ in the testing window for the POD coefficients obtained from NN-ROM and NN-ROM-DA with $p^{\mathrm{DA}} = 1$ and 5. Cylinder wake flow configuration at $Re = 1.5 \times 10^4$. The time averaged $NRMSE$ values over the testing window are indicated by horizontal dashed lines. The dotted lines represent the time steps at which the observations are initiated for the two NN-ROM-DA frameworks.

observed, after each observation step, the EnKF assimilation procedure improves the accuracy of the trajectories in the time span between the observations. Moreover, a slight improvement in accuracy can be observed when multiple consecutive observations are available ($p^{\mathrm{DA}} = 5$) as compared to when only single observation is available at each assimilation step ($p^{\mathrm{DA}} = 1$). This indicates that the NN-ROM estimates benefit from the greater number of assimilated states available in the memory for forward propagation. In Fig. 6.11, the $NRMSE$ defined by (6.1) is compared for the NN-ROM (*i.e.* without assimilation) and for the NN-ROM-DA with $p^{\mathrm{DA}} = 1$ and 5. It is observed that a more accurate estimation is obtained from NN-ROM-DA than NN-ROM in the majority of the testing time span. The corresponding average errors are also lower.

The estimated POD coefficients are used to reconstruct the two-component velocity fields. In Fig. 6.12, the time evolution of the streamwise components of velocity, obtained from the NN-ROM estimates ($u_x^{\mathrm{NN}}(t)$) and the NN-ROM-DA estimates ($u_x^{\mathrm{NN-DA}}(t)$), are compared at a location in the flow field with the reference trajectories determined from the calculated POD coefficients ($u_x^{\mathrm{POD}}(t)$) and the PIV experiments ($u_x^{\mathrm{PIV}}(t)$). Owing to the more accurate estimation of the POD coefficients, the corresponding reconstruction of the velocity is also more accurate for the estimates from NN-ROM-DA as compared to those obtained from NN-ROM. Moreover, the phase shift observed in the time span between the assimilation steps is minimized when multiple consecutive observations are used to nudge the estimated trajectory towards the reference trajectory. To gauge the performance in frequency domain, the power spectra of the reconstructed velocity signals using POD, NN-ROM and NN-ROM-DA (with $p^{\mathrm{DA}} = 5$) are compared in Fig. 6.13 with that using the signal obtained from experiment. It is observed that the peak frequency of the signal obtained from experiments (0.18 Hz) is captured by the estimated dynamics using NN-ROM-DA (0.1874 Hz). In Fig. 6.14, we represent the temporal evolution of the normalized root-mean-square error defined in (6.3) as obtained from NN-ROM and NN-ROM-DA on the one hand, and the coefficients determined by



(a) $p^{\mathrm{DA}} = 1$ consecutive assimilation



(b) $p^{\mathrm{DA}} = 5$ consecutive assimilations

Figure 6.12: Time evolution of the streamwise velocity components at a location $(x/D, y/D) = (4.8, 0.9)$ in the flow field. Comparison of the trajectory obtained from the PIV measurements with the reconstructions using the POD coefficients determined from the NN-ROM and NN-ROM-DA with $p^{\mathrm{DA}} = 1$ (a), and $p^{\mathrm{DA}} = 5$ (b). Cylinder wake flow configuration at $Re = 1.5 \times 10^4$.

POD on the other hand. Again, the NN-ROM-DA estimates show an improvement over the NN-ROM estimates. Indeed, the reconstructions obtained from NN-ROM-DA have the same order of magnitude of error as those obtained from the direct reconstruction using POD modes. A summary of the averaged $NRMSE$ values for the predictions in latent space and the reconstructions in physical space is given in Tab. 6.8. In the latent space, the use of NN-ROM-DA (with $p^{\mathrm{DA}} = 5$) leads to a 35% reduction in the error value as compared to NN-ROM. A corresponding reduction of 32% and 21% in the averaged error values are obtained in the physical space, respectively for the streamwise velocity measurement at a location in the farfield wake and the velocity magnitude. The lower error magnitudes obtained by NN-ROM-DA demonstrate the interest of coupling non-intrusive models determined by neural networks to the data assimilation framework.

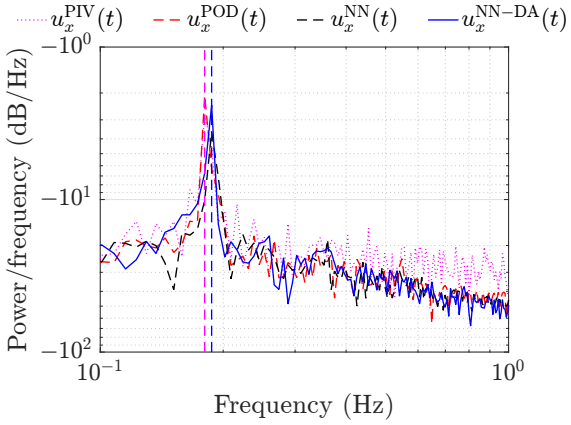


Figure 6.13: Power spectrum densities of the velocity at a location $(x/D, y/D) = (4.8, 0.9)$ in the flow field. Comparison of the results obtained from PIV measurements and the reconstructed velocity signals using the POD coefficients determined from the NN-ROM and NN-ROM-DA with $p^{\mathrm{DA}} = 5$. Cylinder wake flow configuration at $Re = 1.5 \times 10^4$. The peak frequencies of the measured signal and that reconstructed from the augmented ROM are indicated by vertical dashed lines.

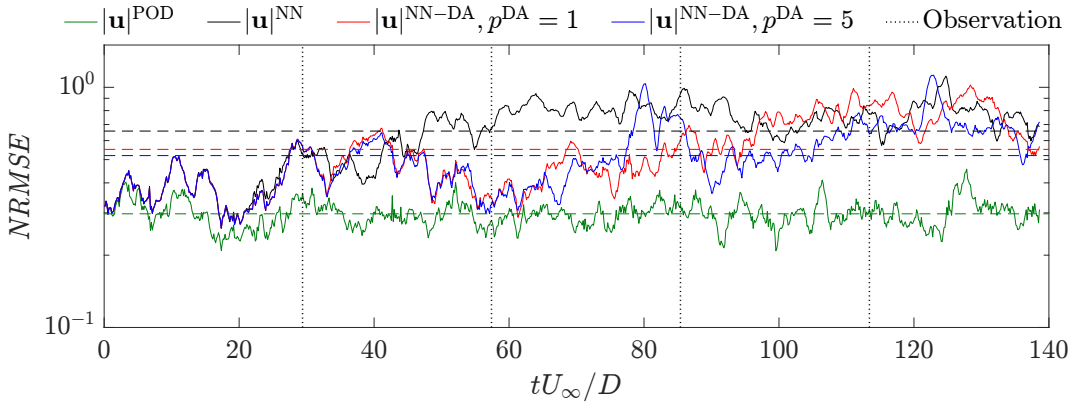


Figure 6.14: Time evolution of $NRMSE$ in the testing window for the velocity magnitude calculated from the POD coefficients obtained from the reference values and those determined from the NN-ROM and NN-ROM-DA with $p^{\mathrm{DA}} = 1$ and 5. Cylinder wake flow configuration at $Re = 1.5 \times 10^4$. The time averaged $NRMSE$ values over the testing window are indicated by horizontal dashed lines. The dotted lines represent the time steps at which the observations are initiated for the two NN-ROM-DA frameworks.

6.4 Conclusion

The neural network-based, data-driven reduced-order model (NN-ROM) presented in Sec. 3.3 was considered in this chapter as a surrogate, non-intrusive alternative to the intrusive POD-Galerkin ROM. Through applications to a toy model and two fluid flow cases, it was demonstrated that NN-ROM offers a viable replacement as a forward model to provide long-term prediction. The whole framework can be considered in an offline-online paradigm. In the offline stage, NN-ROM is trained on variables in the latent space, namely the POD coefficients. These are obtained by projecting a large amount of high-

Table 6.8: Time averaged $NRMSE$ values in the testing window corresponding to the POD coefficients $\boldsymbol{a}$, the streamwise velocity component u_x at location $(x/D, y/D) = (4.8, 0.9)$, and the full-field velocity magnitude $|\boldsymbol{u}|$ for the cylinder wake flow at $Re = 1.5 \times 10^4$.

Average $NRMSE$ $NRMSE(t;\boldsymbol{x},\boldsymbol{x}^{\mathrm{Ref}}) = \sqrt{\overline{(\boldsymbol{x}-\boldsymbol{x}^{\mathrm{Ref}})^2}/\overline{(\boldsymbol{x}^{\mathrm{Ref}})^2}}$		POD	NN-ROM	NN-ROM-DA	
$\boldsymbol{x}$	$\boldsymbol{x}^{\mathrm{Ref}}$			$p^{\mathrm{DA}}=1$	$p^{\mathrm{DA}}=5$
$\boldsymbol{a}^{\mathrm{NN}}$, $\boldsymbol{a}^{\mathrm{NN-DA}}$	$\boldsymbol{a}^{\mathrm{POD}}$	–	1.3234	1.0009	0.8625
u_x^{POD}, u_x^{NN}, $u_x^{\mathrm{NN-DA}}$	u_x^{PIV}	0.1332	0.2765	0.2203	0.1886
$\lvert\boldsymbol{u}^{\mathrm{POD}}\rvert$, $\lvert\boldsymbol{u}^{\mathrm{NN}}\rvert$, $\lvert\boldsymbol{u}^{\mathrm{NN-DA}}\rvert$	$\lvert\boldsymbol{u}^{\mathrm{PIV}}\rvert$	0.2973	0.6573	0.5516	0.5196

fidelity snapshots onto the reduced space. In the online stage, the trained NN-ROM is rapidly evaluated to obtain the future state of the latent space variables. These variables are used for reconstruction in the physical space, thus returning predicted high-fidelity flow field.

The NN-ROM is designed such that it learns on residual targets, *i.e.* the difference of the state of interest between two consecutive time steps, rather than the value of the future state itself. The model takes into account the memory effect by considering a sequential input of fixed length (p) of the latent variable in order to provide estimates at subsequent time steps. Influence of the inclusion of temporal history of the dynamics on the performance of NN-ROM was evaluated for the standard Lorenz-63 system. The results show that an increase in the time period for which the trajectory of true solution is followed by the NN-ROM estimates is observed with the increase in the length of the temporal history data.

In the parametrized framework of NN-ROM, parameters characterizing the dynamics are included in the input feature set while training with the objective to enhance the generalization of the model to provide estimation for configurations outside the training data. The performance of this framework was evaluated for the reconstruction of cylinder wake flow dynamics at low Reynolds number. Data from simulations at 9 Reynolds numbers in the range $Re \in [100, 210]$ is considered and divided into training and testing subsets based on the parameter. The trained model was evaluated to provide estimation of the dynamics corresponding to two parameters - one lying within the range of parameters used for training (interpolation problem) and one outside the range of the training parameter set (extrapolation problem). For the interpolation problem, NN-ROM provides a sufficiently accurate estimate of the dynamics over a long time range. On the other hand, for the extrapolation problem, NN-ROM gives a sufficiently accurate initial estimate of the dynamics but a phase shift is observed in the long-term prediction. However, the state trajectories remain bounded and the amplitude matches that of the reference trajectory.

In order to ensure accurate long-term predictions, NN-ROM is augmented with an Ensemble Kalman Filter (EnKF) algorithm introduced in Sec. 3.2.2. The ability of NN-ROM to provide sequential updates makes it amenable to be used as a forecast model in the EnKF algorithm. The performance of NN-ROM augmented with EnKF (NN-ROM-DA) is evaluated for reconstruction of the flow field. NN-ROM-DA is found to be effective in mitigating the observed phase shift over a long time span and consequently, an improvement in the estimation is observed overall.

The performance of NN-ROM-DA is evaluated on experimental cylinder wake flow data at $Re = 1.5 \times 10^4$. Observations in the form of streamwise velocity components are used to improve the accuracy of NN-ROM estimates. The results show a more accurate long-term prediction of dynamics in both the latent and physical space as compared to NN-ROM. Moreover, using multiple consecutive assimilations ($p^{\mathrm{DA}} > 1$) has been shown to further improve the long-term predictions.

Chapter 7

Conclusion and future work

The main objective of this thesis was to address the challenge of high computational costs posed by full-scale numerical models. Specifically, strategies for reduced-order modeling of high-dimensional estimation problems in fluid dynamics have been developed.

An estimation methodology, in which reduced-order models (ROMs) are extracted from numerical simulations or experimental measurements and are subsequently used as estimator to predict the temporal evolution of the system dynamics, has been employed to investigate the performance of the data-driven reduced-order modeling frameworks belonging to two categories – intrusive and nonintrusive. These frameworks have been demonstrated to provide tools that combine the fidelity and robustness of a high-dimensional representation of dynamical systems with the computational efficiency of a low-order approximator in terms of latent space variables, *i.e.* the temporal modal coefficients.

In the pursuit of this objective, several interesting aspects of the estimation framework have been investigated and novel approaches for model parameter identification and surrogate modeling were proposed. The intrusive and nonintrusive reduced-order modeling approaches were successfully applied to the estimation of the temporal dynamics of the latent space variables and reconstruction of high-dimensional flow fields.

7.1 Intrusive framework

The data-based method involving the identification of a ROM obtained from the Galerkin projection of the incompressible Navier-Stokes equations on the proper orthogonal decomposition (POD) modes has been considered. For dataset which sufficiently represents the typical variations of a system, the intrusive POD-ROM provides estimates which are valid within this range. In order to account for the inaccuracy introduced in the model due to reduction of the dimension, an additive modal eddy viscosity term is introduced for stabilization. This model serves as a low-order approximation of the full-scale system governing the dynamics.

The coefficients of the polynomial terms have been identified using linear regression method. Using toy models, three system identification methods – namely, ordinary least squares (OLS), sparse identification of nonlinear dynamics (SINDy), and least angle regression (LARS) – have been evaluated in a probabilistic framework provided by the bootstrap method. The OLS identification method was found to be accurate but

computationally expensive for problems with high degrees of freedom. The SINDy and LARS identification methods exhibited comparable performances in terms of accuracy of the estimates. However, upon comparing the elapsed time for the parameter identification, it was concluded that the SINDy method is computationally more efficient for identification problems involving low degrees of freedom, while the LARS method is more efficient for the problems involving high degrees of freedom.

Next, the identification of the parameters associated with the modal eddy viscosity terms, crucial for the stabilization of the POD-ROM, has been performed in a data assimilation paradigm. The dual ensemble Kalman filter (Dual-EnKF) algorithm, which seamlessly integrates the model output and measurements, was used for the data-driven parameter estimation. Assuming that the source of the forecast errors is known, this sequential data assimilation technique is able to alternately update the model parameters and provide long-term predictions. The influence of the model and observation covariances and the size of the ensemble, on the estimation error was examined for a toy model. it was found that the accuracy of the estimation depends primarily on the ratio of the observation and model covariance levels which is fixed by the problem. It was also observed that accurate estimates with a low error level of the order of 1.0×10^{-3} were obtained for sufficiently large ensemble sizes. Given an appropriate choice of the initial covariance levels, the algorithm was also able to recover both the state evolution and the model parameters even in the presence of noisy observations. In numerical and experimental cylinder wake applications, an operational ensemble forecast-analysis system which produces substantially improved flow state information was demonstrated. The inclusion of the stabilizing parameters and the assimilation of observations during the model integration was found to provide a framework for robust long term estimation. When the estimated POD coefficients were used for the reconstruction of the vorticity fields, more accurate results were obtained using the stabilized and assimilated ROM as compared to the initial ROM, with a reduction of 4.6% in the error metric. The dynamics of the original velocity field in terms of frequency content was also well replicated without loss in the phase information.

In problems where the prediction of long-term evolution based on the stabilized reduced-order models was found to be unstable, it was overcome by introducing a feedback system using EnKF algorithm. For all the test cases, this sequential data assimilation technique was able to successfully capture the full state dynamics using regular assimilation of new observations. Nonetheless, it must be noted that the more accurate prediction of the next step provided by the stabilized POD-ROM helps in relaxing the model error covariance levels and suppressing the accumulated errors. Thus, such improvement of the POD-ROM is a prerequisite for accurate data-driven simulations of unsteady flows.

7.2 Nonintrusive framework

A novel framework based on deep neural network (DNN) has been developed as a surrogate, nonintrusive reduced-order modeling approach to bypass the Galerkin projection step used to obtain the intrusive POD-ROM. The nonintrusive ROM, referred as NN-ROM, operated in an offline-online paradigm to estimate the temporal POD coefficients. During training, the NN-ROM allows to take into account the memory effect in the form

of temporal sequence as input features, and to consider residuals between two consecutive time steps as output targets. The performance of the NN-ROM was assessed by implementing it as a forward model to obtain long-term estimates for toy model and fluid flow problems.

For the standard Lorenz-63 system, it was observed that the NN-ROM estimator was able to accurately predict the dynamics over a longer time range as the length of the temporal history data in the input increased. However, the reduction in estimation error was not monotonous for this example featuring a positive Lyapunov exponent. The number of past time steps is therefore treated as a hyperparameter which has been selected heuristically in this thesis.

Next, parameters characterizing the flow problems were included in the input features of the NN-ROM and the trained model was used to estimate the dynamics corresponding to the configurations lying outside the parameter set used for training. Data from numerical cylinder wake flow at low Reynolds number in the range of $Re \in [100, 210]$ was considered to evaluate the performance of the parametrized framework. For the interpolation problem, the model provided sufficiently accurate estimate of the dynamics over a long time range. On the other hand, for the extrapolation problem, only the initial estimate was found to be accurate and a phase shift was observed in the long-term prediction. As a remedy, the NN-ROM estimates were augmented with flow field observations using the ensemble Kalman filter (EnKF) algorithm, where the trained NN-ROM served as a forecast model in the data assimilation paradigm. It was observed that the NN-ROM augmented with EnKF, referred as NN-ROM-DA, was able to mitigate the observed phase-shift and therefore improve the accuracy of long-term predictions, evident from a decrease of 45% in the error metric as compared to the initial NN-ROM estimation.

Lastly, the performance of the NN-ROM-DA framework was evaluated for a high Reynolds number experimental cylinder wake flow problem. A similar improvement as the numerical case was observed in the long-term prediction of the dynamics. Moreover, using multiple consecutive assimilations, a decrease of 21% in the error metric was observed as compared to the initial NN-ROM estimation.

In conclusion, the multistep, residual-based, parametrized neural network framework which is augmented with EnKF can be employed as a sufficiently robust approach to provide accurate long-term dynamical predictions.

7.3 Recommendations for future work

In this work, relatively low-order models, *i.e.* degrees of freedom less than or equal to 20, have been considered. This allowed to concentrate on the development, testing, and evaluation of the different reduced-order modeling strategies rather than dealing with the model complexities. The discussions in Chap. 5 and Chap. 6 provided valuable insights into the performance of the intrusive and nonintrusive methods in applications involving a range of dynamical systems.

A natural extension of the current work is to use the data-based ROM identification in operational forecasting and in applications requiring repeated realizations of the system such as real-time control, multidisciplinary optimization, and uncertainty quantification. Working in a practical setup would provide an opportunity to assess the feasibility and

to gain a deeper insight into the capabilities of the data-based reduced-order modeling frameworks.

In terms of data assimilation, a more intricate representation of the errors and error covariances is required for application to more complex problems. The amplitudes of the model and observation error covariance define the weights given to the background (model forecasts) and the observations which consequently impact the accuracy of the estimation. Also, the consideration of the type and availability of the observational data, the nonlinear observation operators, representativity and instrument errors will be important while assimilating real observations.

A comparative study of the parameter estimation methods proposed in this work must be performed against other data-driven calibration techniques (Brunton, Noack, and Koumoutsakos, 2019). This would allow an evaluation of the cost effectiveness of the current method with respect to the currently used modeling alternatives.

Except for the parametrized NN-ROM, it was assumed that the reduced-order models are applicable only within the operating range of the snapshot dataset used to build the model. Robustness analysis of the models must be performed in order to have a detailed knowledge of the range of validity of the reduced-order model, *i.e.* to understand if the identified model will still be able to provide accurate estimates if the parameters like Reynolds number vary between specified minimum and maximum values. This can be challenging as, in governing nonlinear PDEs, the change of physical parameters results in a change in the spatial distribution of the solution which the initial POD modes may not be able to approximate. One of the scenarios in which the robustness analysis can be beneficial is when uncertainties are introduced in terms of time varying parameter disturbances, which in turn affect the estimation of the POD coefficients.

Lastly, it is acknowledged that the performance of the parametric framework of NN-ROM was tested with the limited available parametrized dataset and needs to be further evaluated with datasets involving more number of parameters and those obtained from experiments. Also, for NN-ROM architecture and optimization, more sophisticated methods like random search (Bergstra and Bengio, 2012) and Bayesian optimization (Brochu et al., 2010) should be introduced to tune the model hyperparameters according to the problem.

Appendix A

Numerical simulation of 2D-cylinder wake flow

Contents

In the fluid dynamics community, a large body of literature exists in which the two-dimensional incompressible viscous flow past a circular cylinder has been chosen to illustrate modal decomposition (Bagheri, 2013) and model identification techniques (Noack, Afanasiev, et al., 2003; Brunton, Proctor, et al., 2016b; Rowley and Dawson, 2017). Owing to its simple dynamics, this particular set-up has been used extensively in this work as a benchmark dynamical system to illustrate the model identification strategies. In this appendix, the details of the numerical simulation are discussed.

A.1 Mesh

The numerical simulation is performed using a finite-element based incompressible Navier-Stokes equations solver written in *FreeFem++* (Hecht, 2013). The computational domain along with the dimensions and boundary conditions is shown in Fig. A.1. All the dimensions have been parameterized with the diameter of the cylinder D, here set to one.

For the purpose of mesh size control, the domain is divided into sub-domains and the automatic mesh generation offered by the `buildmesh` command in *FreeFem++* is used. The mesh has 10263 vertices and 20360 triangles. See Fig. A.2.

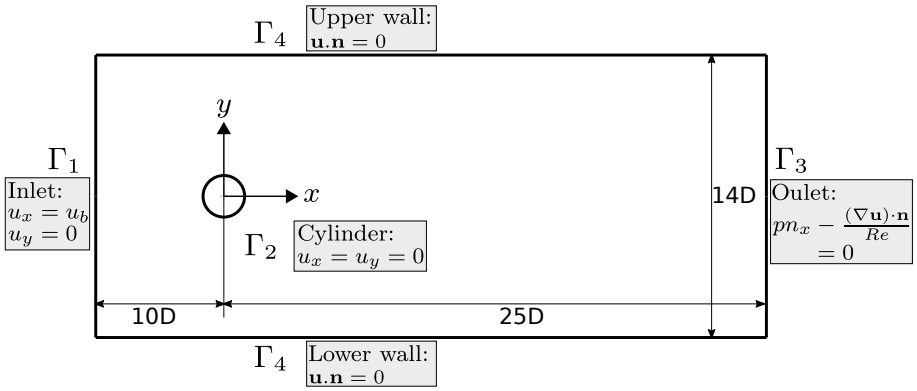


Figure A.1: Schematic of the computational domain and boundary conditions for the 2D-cylinder wake flow simulation.

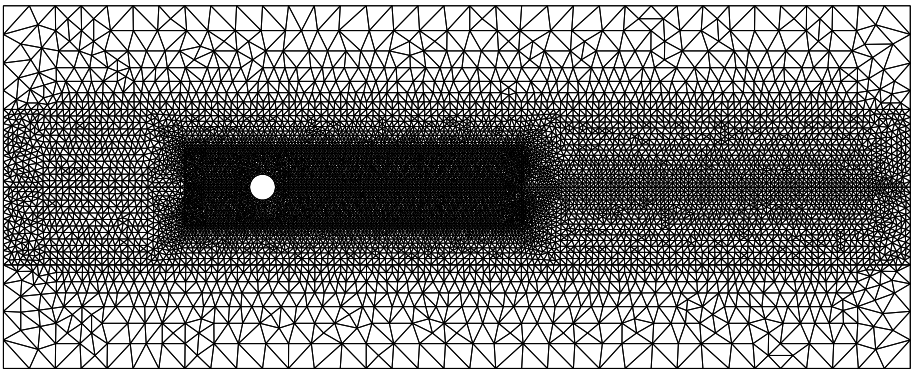


Figure A.2: Finite-element discretization of the domain obtained from *FreeFem++*.

A.2 Boundary conditions

The schematic representation of the boundary conditions for the $2D$ cylinder flow domain is shown in Fig. A.1. At the inlet, we consider a uniform flow in the x-direction. Here, $u_\mathsf{b} = 1$. On the upper and lower walls and the cylinder surface, we have no-slip boundary conditions. The free outflow condition is used at the outlet. In $FreeFem++$, this condition is implicitly applied.

A.3 Validation of results of flow at $Re_D = 100$

To solve the space discretized Navier-Stokes equations, an optimized Newton method is used. This approach is a variant of the classical Newton method for which the nonlinear term is discretized semi-implicitly. The velocity and pressure variables are discretized using $P2$ and $P1$ finite element spaces, respectively. After convergence of the iterative procedure, the pressure field is such that the resulting velocity is divergence free. A sample of the vorticity field at $t = 150$ is shown in Fig. A.3.

Table A.1: Post-transient parameters of 2D-cylinder wake flow at $Re = 100$

	$\overline{C_d}$	$C_{l,\max}$	$\overline{\Delta p}$	St
Simulation	1.36	0.3354	0.9307	0.168
Bounds (Muddada and Patnaik, 2010)	1.28–1.41	–	–	0.160–0.170

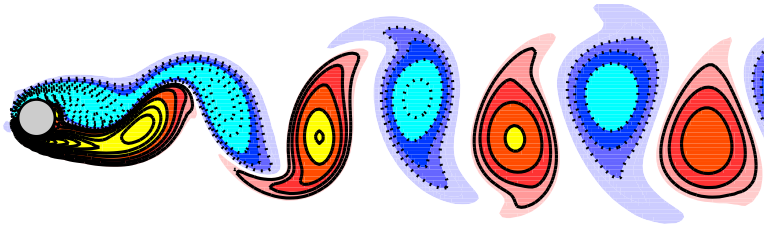
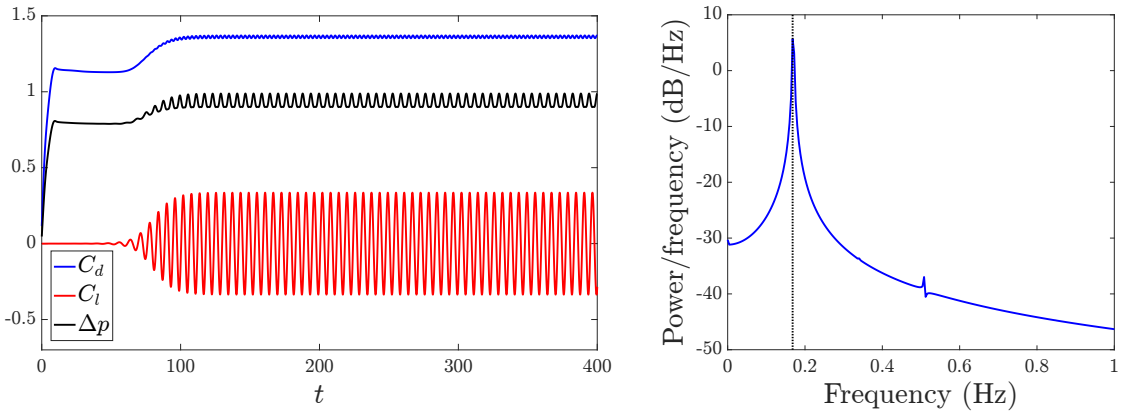


Figure A.3: Vorticity field obtained from the numerical simulation of 2D-cylinder wake flow at $Re = 100$ at the instant $t = 150$.



(a) **(b)**

Figure A.4: (a) Time evolution of drag coefficient (C_d), lift coefficient (C_l), and pressure drop across the cylinder's leading and trailing surfaces (Δp). (b) Power spectral density of the C_l evolution in the post-transient stable vortex shedding window. The dotted line indicates the peak frequency.

The Reynolds number of the numerical simulation is set to $Re = 100$, based on the cylinder diameter D, the free-stream velocity u_b, and the kinematic viscosity ν. This Reynolds number is well above the critical Reynolds number ($Re_c = 48$) for the onset of the two-dimensional vortex shedding (Zebib, 1987) and below the critical Reynolds number ($Re_c = 188$) for the onset of three-dimensional instabilities (Zhang, Fey, et al., 1995).

The snapshots are taken in a time range of $t = [150, 250]$ after all initial disturbances have been damped. The simulation generates $N_t = 1000$ snapshots at a sampling frequency of $f_s = 10$ Hz. The snapshots contain the information of the two components of the fluctuating velocity vector $\boldsymbol{u}' = [u'_x, u'_y]^\top$. The number of degrees of freedom for the problem (size of the discretized state variables) is $N_s = 81772$. The time evolution of the drag and lift coefficient are shown in Fig. A.4a. From the evolution, it is observed that after the initial transient phase, the periodic vortex shedding is observed from

$t = 100$ onward. This implies that the snapshots will represent the dynamics of the post-transient stable vortex shedding.

For validation, the post-transient values of mean drag coefficient $\overline{C_d}$ and the Strouhal number St are compared with the range reported in (Muddada and Patnaik, 2010). The values of maximum lift coefficient $C_{l,\max}$ and mean pressure drop $\overline{\Delta p}$ are also obtained. The power spectral density of the post-transient C_l evolution is obtained, as shown in Fig. A.4b, and the value of the peak frequency f is used to calculate the Strouhal number $St = fD/u_b$, where $u_b = 1$ and $D = 1$. The values are reported in Tab. A.1. It is observed that the values of $\overline{C_d}$ and St lie within the bounds reported in the literature.

Appendix B

Elements of linear algebra

Contents

B.1 Dot product

Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be two column vectors in $\mathbb{R}^n$. The dot product is defined as

$$\boldsymbol{x} \cdot \boldsymbol{y} = \boldsymbol{x}^\top \boldsymbol{y} = \sum_{i=1}^{n} x_i y_i \in \mathbb{R}.$$

B.2 Outer product

Let $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{y} \in \mathbb{R}^m$ be two column vectors. Their outer product is defined as $\boldsymbol{y}\boldsymbol{x}^\top = \boldsymbol{A}$ where

$$(\boldsymbol{A})_{ij} = y_i x_j \qquad \text{with} \quad 1 \leq i \leq m, \quad \text{and} \quad 1 \leq j \leq n.$$

B.3 Vector norm

B.3.1 Definition

Given a vector space V over a field K (real numbers $\mathbb{R}$ or complex numbers $\mathbb{C}$), a norm on V is a non negative-valued function $p : V \to \mathbb{R}^+$ with the following properties:

For all $a \in K$ and all $\boldsymbol{u}, \boldsymbol{v} \in V$,

1. $p(\boldsymbol{u} + \boldsymbol{v}) \leq p(\boldsymbol{u}) + p(\boldsymbol{v})$ (*triangle inequality*).
2. $p(a\boldsymbol{u}) = |a|\, p(\boldsymbol{u})$ (*absolutely homogeneous* or *absolutely scalable*).
3. if $p(\boldsymbol{u}) = 0$ then $\boldsymbol{u} = \boldsymbol{0}$ (*positive definite*).

The norm of a vector $\boldsymbol{u} \in V$ is usually denoted by $p(\boldsymbol{u}) = \|\boldsymbol{u}\|$.

A *seminorm* on V is a function $p : V \to \mathbb{R}^+$ with only the properties 1 and 2 above.

B.3.2 Equivalent norms

Suppose that p and q are two norms (or seminorms) on a vector space V. Then p and q are called *equivalent*, if there exists two real constants c and C with $c > 0$ such that for every vector $\boldsymbol{v} \in V$, we have

$$cq(\boldsymbol{v}) \leq p(\boldsymbol{v}) \leq Cq(\boldsymbol{v}).$$

In a finite-dimensional space, any tow norms are equivalent but this is not true in infinite-dimensional spaces.

B.3.3 p-norm ($p \geq 1$)

Let $p \geq 1$ be a real number. The p-norm (also called ℓ_p-norm) of vector $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ is

$$\|\boldsymbol{x}\|_p := \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}.$$

For $p = 1$, we get the Taxicab norm or Manhattan norm $\|\boldsymbol{x}\|_1 := \sum_{i=1}^{n} |x_i|$. It can be viewed as counting the number of blocks you would have to walk on a n-dimensional grid. For $p = 2$, we get the Euclidean norm $\|\boldsymbol{x}\|_2 := \sqrt{x_1^2 + \cdots + x_n^2}$. As p approaches ∞, the p-norm approaches the infinity norm or maximum norm: $\|\boldsymbol{x}\|_\infty := \max_i |x_i|$. The norm is a measure of length. All these norms are equivalent, since

$$\|\boldsymbol{x}\|_\infty \leq \|\boldsymbol{x}\|_p \leq n^{\frac{1}{p}} \|\boldsymbol{x}\|_\infty.$$

B.3.4 0-norm

Let $\boldsymbol{x} \in \mathbb{R}^n$ and define the support set of $\boldsymbol{x}$ as the set of indices corresponding to its nonzero elements, *i.e.*

$$\operatorname{supp}(\boldsymbol{x}) := \{j \in \{1, 2, \cdots, n\} \mid x_j \neq 0\}.$$

The ℓ_0 penalty of $\boldsymbol{x}$ measures the number of nonzero elements in the vector and is defined as:

$$\|\boldsymbol{x}\|_0 := \operatorname{Card}(\operatorname{supp}(\boldsymbol{x})).$$

The vector $\boldsymbol{x}$ is called s-sparse if it has at most s nonzero elements, thus $\|\boldsymbol{x}\|_0 \leq s$.

B.4 Matrix norm

B.4.1 Definition

Let $K^{m\times n}$ be the vector space of all matrices of size $m \times n$ with entries in the field K (real numbers $\mathbb{R}$ or complex numbers $\mathbb{C}$). A matrix norm is a function $\|\cdot\| : K^{m\times n} \to \mathbb{R}$ that must satisfy the following properties:

- $\|\alpha \boldsymbol{A}\| = |\alpha| \|\boldsymbol{A}\|$ (*absolutely homogeneous*)
- $\|\boldsymbol{A} + \boldsymbol{B}\| \leq \|\boldsymbol{A}\| + \|\boldsymbol{B}\|$ (*sub-additive* or *triangle inequality*)
- $\|\boldsymbol{A}\| \geq 0$ (*positive-valued*)
- $\|\boldsymbol{A}\| = 0 \iff \boldsymbol{A} = 0_{m,n}$ (*definite*)

for all scalars $\alpha \in K$ and for all matrices $\boldsymbol{A}, \boldsymbol{B} \in K^{m\times n}$.

Additionally, in the case of square matrices ($m = n$), some (but not all) matrix norms satisfy the additional property given by

$$\|\boldsymbol{AB}\| \leq \|\boldsymbol{A}\| \|\boldsymbol{B}\|.$$

A matrix norm that satisfies this additional property is called a *submultiplicative* norm.

B.4.2 Matrix norms induced by vector norms

Let $\|\cdot\|$ be a vector norm for both spaces K^m and K^n. The *induced norm* on the space $K^{m\times n}$ of all $m \times n$ matrices is defined as follows:

$$\begin{aligned}\|\boldsymbol{A}\| &= \sup\{\|\boldsymbol{Ax}\| : \boldsymbol{x} \in K^n \text{ with } \|\boldsymbol{x}\| = 1\} \\ &= \sup\left\{\frac{\|\boldsymbol{Ax}\|}{\|\boldsymbol{x}\|} : \boldsymbol{x} \in K^n \text{ with } \boldsymbol{x} \neq \boldsymbol{0}\right\}.\end{aligned}$$

If the p-norm for vectors ($1 \leq p \leq \infty$) is used (see Sec. B.3), then

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

In the special cases of $p = 1, 2, \infty$, the induced matrix norms can be computed as

$$\|\boldsymbol{A}\|_1 = \max_{1 \le j \le n} \sum_{i=1}^{m} |a_{ij}|,$$

which is simply the maximum absolute column sum of the matrix;

$$\|\boldsymbol{A}\|_\infty = \max_{1 \le i \le m} \sum_{j=1}^{n} |a_{ij}|,$$

which is simply the maximum absolute row sum of the matrix;

$$\|\boldsymbol{A}\|_2 = \sigma_{\max}(\boldsymbol{A}),$$

where $\sigma_{\max}(\boldsymbol{A})$ represents the largest singular value of matrix $\boldsymbol{A}$. The following inequality holds:

$$\|\boldsymbol{A}\|_2 = \sigma_{\max}(\boldsymbol{A}) \le \|\boldsymbol{A}\|_{\mathsf{F}} = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2 \right)^{\frac{1}{2}},$$

where $\|\boldsymbol{A}\|_{\mathsf{F}}$ is the Frobenius norm.

B.4.3 "Entrywise" matrix norms

The *Frobenius* norm or the *Hilbert–Schmidt* norm is defined as:

$$\|\boldsymbol{A}\|_{\mathsf{F}} = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2} = \sqrt{\operatorname{trace}\left(\boldsymbol{A}^{\mathsf{H}} \boldsymbol{A}\right)} = \sqrt{\sum_{i=1}^{\min\{m,n\}} \sigma_i^2(\boldsymbol{A})},$$

where $\sigma_i(\boldsymbol{A})$ are the singular values of $\boldsymbol{A}$.

B.5 Injectivity, surjectivity and bijection

Let f be a function mapping the domain X to the *codomain* Y, *i.e.* $f : X \to Y$.

B.5.1 Injective function

By definition, the function f is said to be *injective*, if

$$\forall \boldsymbol{x_1}, \boldsymbol{x_2} \in X, \quad f(\boldsymbol{x_1}) = f(\boldsymbol{x_2}) \Rightarrow \boldsymbol{x_1} = \boldsymbol{x_2},$$

or, using the contrapositive, if

$$\forall \boldsymbol{x_1}, \boldsymbol{x_2} \in X, \quad \boldsymbol{x_1} \neq \boldsymbol{x_2} \Rightarrow f(\boldsymbol{x_1}) \neq f(\boldsymbol{x_2}).$$

An injective function (also known as *injection*, or *one-to-one* function) is a function that maps distinct elements of its domain to distinct elements of its codomain. In other

words, every element of the function's codomain is the image of at most one element of its domain.

B.5.2 Surjective function

By definition, the function f is said to be *surjective*, if

$$\forall \boldsymbol{y} \in Y, \exists \boldsymbol{x} \in X, \quad f(\boldsymbol{x}) = \boldsymbol{y}.$$

A surjective function is also known as *surjection*, or *onto* function. It is not required that $\boldsymbol{x}$ be unique; the function f may map one or more elements of X to the same element of Y.

B.5.3 Bijective function

By definition, the function f is a *bijection*, *bijective function*, *one-to-one correspondence*, or *invertible* function, if f is a one-to-one (injective) and onto (surjective) mapping of a set X to a set Y. In other words, each element of X is paired with exactly one element of Y, and each element of Y is paired with exactly one element of X. There are no unpaired elements.

B.6 Basis

Let V be a vector space over a field K (real numbers $\mathbb{R}$ or complex numbers $\mathbb{C}$). A subset B of V is a *basis* if it satisfies the two conditions:

1. the *linear independence* property, *i.e.* for every finite subset $\{\boldsymbol{v_1}, \ldots, \boldsymbol{v_n}\}$ of B:

   if $\quad c_1\boldsymbol{v_1} + \cdots + c_n\boldsymbol{v_n} = 0$, for some $c_1, \ldots, c_n \in K \quad$ then $\quad c_1 = \cdots = c_n = 0$;

2. the *spanning* property, *i.e.* for every vector $\boldsymbol{v}$ in V, one can write:

   $$\boldsymbol{v} = c_1\boldsymbol{v_1} + \cdots + c_n\boldsymbol{v_n} \text{ with } c_1, \ldots, c_n \in K \quad \text{and} \quad \boldsymbol{v_1}, \ldots, \boldsymbol{v_n} \in B.$$

The scalars c_i are called the coordinates of the vector $\boldsymbol{v}$ with respect to the basis B. By the first property, the coordinates are uniquely determined. The *dimension* of a subspace is the largest number of vectors in the subspace that are linearly independent.

B.7 Range space, Null space and Rank

B.7.1 Definitions

Let $\boldsymbol{A} \in \mathbb{R}^{n \times m}$ be an arbitrary matrix, we can associate to $\boldsymbol{A}$ the linear map $f : \mathbb{R}^n \to \mathbb{R}^m$ such that $\boldsymbol{x} \mapsto \boldsymbol{A}\boldsymbol{x}$ where $\boldsymbol{x} \in \mathbb{R}^n$. For f to be an injective function, it is necessary that $n \leq m$ and that the columns of $\boldsymbol{A}$ be linearly independent. If the columns are not linearly independent, then there exists $\boldsymbol{z} \in \mathbb{R}^n$ such that $\boldsymbol{A}\boldsymbol{z} = \boldsymbol{0}$. Due to the linearity,

there are an infinite number of vectors that map to zero. This set of vectors is called the *null space* of $\boldsymbol{A}$ and is denoted

$$\mathcal{N}(\boldsymbol{A}) = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}\}.$$

Due to the linearity of the mapping, $\mathcal{N}(\boldsymbol{A})$ is a subspace.

Now consider the range of f. The range

$$\mathcal{R}(\boldsymbol{A}) = \{\boldsymbol{A}\boldsymbol{x} \mid \boldsymbol{x} \in \mathbb{R}^n\}.$$

is the set of vectors mapped to $\mathbb{R}^m$ by $\boldsymbol{x} \mapsto \boldsymbol{A}\boldsymbol{x}$. For an arbitrary $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ is a linear combination of the columns of $\boldsymbol{A}$. The range of $\boldsymbol{A}$ is then the span of the columns of $\boldsymbol{A}$:

$$\mathcal{R}(\boldsymbol{A}) = \mathrm{Span}(\boldsymbol{a_1}, \boldsymbol{a_2}, \cdots, \boldsymbol{a_n}).$$

The column-rank is the dimension of $\mathcal{R}(\boldsymbol{A})$ and the row-rank is the dimension of $\mathcal{R}(\boldsymbol{A}^\top)$. The column-rank of a matrix is equal to its row-rank and is called the *rank* of the matrix. A matrix is said to have *full rank* if $\mathrm{Rank}(\boldsymbol{A}) = \min(m, n)$. f is an injective function if $m \geq n$ and $\boldsymbol{A}$ has full rank. In that case, we have

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}\boldsymbol{y} \Rightarrow \boldsymbol{x} = \boldsymbol{y}.$$

For f to be a surjective function, the column rank must be m. A square matrix is full rank if and only if f is a bijective function. Such a matrix is called *non singular*. For a non singular matrix, there exists a unique *inverse*. A square matrix with rank less than its size is called *singular*.

B.7.2 Fundamental theorem of linear algebra

The fundamental theorem of linear algebra states that for a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, we have:

$$\dim(\mathcal{R}(\boldsymbol{A})) + \dim(\mathcal{N}(\boldsymbol{A})) = n.$$

B.8 Inner product

B.8.1 Definition

Let V be a vector space over the field K (real numbers $\mathbb{R}$ or complex numbers $\mathbb{C}$). The map

$$\langle \cdot, \cdot \rangle : V \times V \to K$$

is called an inner product, if the following conditions (1), (2) and (3) are satisfied for all vectors $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in V$ and all scalars $a \in K$:

1. *Linearity* in the first argument:

$$\langle a\boldsymbol{x}, \boldsymbol{y} \rangle = a\langle \boldsymbol{x}, \boldsymbol{y} \rangle$$
$$\langle \boldsymbol{x} + \boldsymbol{y}, \boldsymbol{z} \rangle = \langle \boldsymbol{x}, \boldsymbol{z} \rangle + \langle \boldsymbol{y}, \boldsymbol{z} \rangle$$

2. *Hermitian* symmetry:
$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \langle \boldsymbol{y}, \boldsymbol{x} \rangle^{\mathsf{H}}.$$

3. *Positive-definite*:
$$\langle \boldsymbol{x}, \boldsymbol{x} \rangle > 0, \qquad \text{if } \boldsymbol{x} \neq \boldsymbol{0}$$

Assuming (1) holds, condition (3) will hold if and only if conditions (4) and (5) below hold:

4. *Positive semi-definite* or nonnegative-definite:
$$\langle \boldsymbol{x}, \boldsymbol{x} \rangle \geq 0$$

5. *Definite condition*
$$\langle \boldsymbol{x}, \boldsymbol{x} \rangle = 0 \Rightarrow \boldsymbol{x} = \boldsymbol{0}$$

Conditions (1) through (5) are satisfied by every inner product. Inner product spaces are normed vector spaces for the norm defined by $\|x\| = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle}$.

B.8.2 Examples

B.8.2.1 Euclidean vector space

A common special case of the inner product is the dot product defined in Sec. B.1. $\mathbb{R}^n$ with the dot product defines the Euclidean vector space:

$$\left\langle \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \right\rangle_{\mathbb{R}^n} := \boldsymbol{x}^\top \boldsymbol{y} = \sum_{i=1}^{n} x_i y_i = x_1 y_1 + \cdots + x_n y_n.$$

B.8.2.2 Inner product with respect to matrix

Let $\boldsymbol{A} \in \mathbb{C}^{n \times n}$ be any Hermitian positive-definite[1] matrix. The inner product with respect to $\boldsymbol{A}$ of $\boldsymbol{x} \in \mathbb{C}^n$ and $\boldsymbol{y} \in \mathbb{C}^n$ is given by

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\boldsymbol{A}} := \boldsymbol{y}^{\mathsf{H}} \boldsymbol{A} \boldsymbol{x} = \left(\boldsymbol{x}^{\mathsf{H}} \boldsymbol{A} \boldsymbol{y} \right)^{\mathsf{H}}.$$

The inner product can be used to define a norm

$$\|\boldsymbol{x}\|_{\boldsymbol{A}} = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle_{\boldsymbol{A}}},$$

which is called the $\boldsymbol{A}$-norm. When $\boldsymbol{A} = \boldsymbol{I}$, this is just the 2-norm.

$\boldsymbol{A}$ is an Hermitian positive semidefinite matrix if and only if it can be decomposed as a product
$$\boldsymbol{A} = \boldsymbol{M}^{\mathsf{H}} \boldsymbol{M}.$$

[1] $\boldsymbol{A}$ is said to be positive-definite if the scalar $\boldsymbol{z}^{\mathsf{H}} \boldsymbol{A} \boldsymbol{z}$ is strictly positive for every non-zero column vector $\boldsymbol{z}$ of n complex numbers.

With that in mind, the $\boldsymbol{A}$ inner product can be written:

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\boldsymbol{A}} = \boldsymbol{y}^{\mathsf{H}} \boldsymbol{A} \boldsymbol{x} = \boldsymbol{y}^{\mathsf{H}} \boldsymbol{M}^{\mathsf{H}} \boldsymbol{M} \boldsymbol{x} = (\boldsymbol{M}\boldsymbol{y})^{\mathsf{H}} (\boldsymbol{M}\boldsymbol{x}) = \langle \boldsymbol{M}\boldsymbol{y}, \boldsymbol{M}\boldsymbol{x} \rangle_{\mathbb{C}^n}.$$

In terms of norm, we obtain:

$$\|\boldsymbol{x}\|_{\boldsymbol{A}} = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle_{\boldsymbol{A}}} = \sqrt{\langle \boldsymbol{M}\boldsymbol{x}, \boldsymbol{M}\boldsymbol{x} \rangle_{\mathbb{C}^n}} = \|\boldsymbol{M}\boldsymbol{x}\|_{\mathbb{C}^n}.$$

B.9 Orthogonality and orthonormality

B.9.1 Definition

Two vectors, $\boldsymbol{x}$ and $\boldsymbol{y}$, in an inner product space, V, are orthogonal if their inner product $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = 0$. We denote this relation $\boldsymbol{x} \perp \boldsymbol{y}$. These vectors are $\boldsymbol{A}$-orthogonal if $\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\boldsymbol{A}} = 0$.

Let $\langle \cdot, \cdot \rangle$ be the inner product defined over V. A set of vectors $\{\boldsymbol{u_1}, \boldsymbol{u_2}, \ldots, \boldsymbol{u_n}\} \in V$ is called *orthonormal* if and only if

$$\langle \boldsymbol{u_i}, \boldsymbol{u_j} \rangle = \delta_{ij}, \quad \forall i, j$$

where δ_{ij} is the Kronecker delta. $\boldsymbol{A}$-orthonormality is defined by extension with the $\boldsymbol{A}$-inner product. Every orthonormal set of vectors is linearly independent.

B.9.2 Unitary/orthogonal matrices

B.9.2.1 Definition

$\boldsymbol{A} \in \mathbb{C}^{n \times n}$ is *unitary*, if

$$\boldsymbol{A}^{\mathsf{H}} \boldsymbol{A} = \boldsymbol{A} \boldsymbol{A}^{\mathsf{H}} = \boldsymbol{I}_n.$$

By extension, if $\boldsymbol{A} \in \mathbb{R}^{n \times n}$, we define an *orthogonal* matrix as:

$$\boldsymbol{A}^{\top} \boldsymbol{A} = \boldsymbol{A} \boldsymbol{A}^{\top} = \boldsymbol{I}_n.$$

The columns and rows of $\boldsymbol{A}$ are orthonormal for the usual inner product.

B.9.2.2 Properties

If $\boldsymbol{A}$ is a unitary matrix, then the following hold:

P_1 : Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be two complex vectors, multiplication by $\boldsymbol{A}$ preserves their inner product, *i.e.* $\langle \boldsymbol{A}\boldsymbol{x}, \boldsymbol{A}\boldsymbol{y} \rangle = \langle \boldsymbol{x}, \boldsymbol{y} \rangle$. See C_6 for the consequence.

P_2 : $\boldsymbol{A}$ is *normal* (see Sec. B.11 for the definition and properties) : $\boldsymbol{A}^{\mathsf{H}} \boldsymbol{A} = \boldsymbol{A} \boldsymbol{A}^{\mathsf{H}}$. See P_3 and C_7 for the consequence.

P_3 : $\boldsymbol{A}$ is diagonalizable and its eigenvectors form an orthonormal basis, *i.e.* $\boldsymbol{A}$ has a decomposition of the form

$$\boldsymbol{A} = \boldsymbol{U} \boldsymbol{\Lambda} \boldsymbol{U}^{\mathsf{H}}$$

where $\boldsymbol{U}$ is unitary, and $\boldsymbol{\Lambda}$ is diagonal and unitary. $\boldsymbol{A}$ is similar to the diagonal matrix $\boldsymbol{\Lambda}$.

P_4 : $|\det(\boldsymbol{A})| = 1$. See C_7 for the consequence in terms of eigenvalues of $\boldsymbol{A}$.

B.9.2.3 Equivalent conditions

If $\boldsymbol{A} \in \mathbb{C}^{n\times n}$, then the following conditions are equivalent:

C_1 : $\boldsymbol{A}$ is unitary.

C_2 : $\boldsymbol{A}^{\mathsf{H}}$ is unitary.

C_3 : $\boldsymbol{A}$ is invertible with $\boldsymbol{A}^{-1} = \boldsymbol{A}^{\mathsf{H}}$.

C_4 : The columns of $\boldsymbol{A}$ form an orthonormal basis of $\mathbb{C}^n$ with respect to the usual inner product, *i.e.* $\boldsymbol{A}^{\mathsf{H}}\boldsymbol{A} = \boldsymbol{I}_n$.

C_5 : The rows of $\boldsymbol{A}$ form an orthonormal basis of $\mathbb{C}^n$ with respect to the usual inner product, *i.e.* $\boldsymbol{A}\boldsymbol{A}^{\mathsf{H}} = \boldsymbol{I}_n$.

C_6 : $\boldsymbol{A}$ is an *isometry* with respect to the usual norm, *i.e.* $\|\boldsymbol{A}\boldsymbol{x}\|_2 = \|\boldsymbol{x}\|_2$ for all $\boldsymbol{x} \in \mathbb{C}^n$, where $\|\boldsymbol{x}\|_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2}$. Orthogonal matrices $\boldsymbol{A}$ are often called rotations or reflections.

C_7 : $\boldsymbol{A}$ is a normal matrix (equivalently, there is an orthonormal basis formed by eigenvectors of $\boldsymbol{A}$). Since $|\det(\boldsymbol{A})| = 1$ (see P_4), then the eigenvalues of $\boldsymbol{A}$ lie on the unit circle.

B.9.3 Gram-Schmidt orthogonalization process

B.9.3.1 Theorem

If $\{\boldsymbol{v_1}, \ldots, \boldsymbol{v_n}\}$ is a linearly independent set of vectors in an inner-product space V, then there exists an orthonormal set $\{\boldsymbol{e_1}, \ldots, \boldsymbol{e_n}\}$ of vectors in V such that

$$\operatorname{Span}(\boldsymbol{v_1}, \boldsymbol{v_2}, \cdots, \boldsymbol{v_n}) = \operatorname{Span}(\boldsymbol{e_1}, \boldsymbol{e_2}, \cdots, \boldsymbol{e_n}).$$

B.9.3.2 Gram-Schmidt algorithm

Let $\operatorname{proj}_{\boldsymbol{u}}(\boldsymbol{v})$ be the operator that projects the vector $\boldsymbol{v}$ orthogonally onto the line spanned by vector u. This projection operator is defined by

$$\operatorname{proj}_{\boldsymbol{u}}(\boldsymbol{v}) = \frac{\langle \boldsymbol{u}, \boldsymbol{v} \rangle}{\langle \boldsymbol{u}, \boldsymbol{u} \rangle} \boldsymbol{u}.$$

If $\boldsymbol{u} = \boldsymbol{0}$, we define $\text{proj}_{\boldsymbol{0}}(\boldsymbol{v}) := \boldsymbol{0}$ *i.e.* , the projection map is the zero map, sending every vector to the zero vector. The Gram–Schmidt process then works as follows:

$$\boldsymbol{u}_k = \boldsymbol{v}_k - \sum_{j=1}^{k-1} \text{proj}_{\boldsymbol{u}_j}(\boldsymbol{v}_k), \qquad \boldsymbol{e}_k = \frac{\boldsymbol{u}_k}{\|\boldsymbol{u}_k\|} \quad k = 1, \cdots, n. \tag{B.1}$$

The calculation of the sequence $\boldsymbol{u_1}, \cdots, \boldsymbol{u_n}$ is known as Gram–Schmidt orthogonalization, while the calculation of the sequence $\boldsymbol{e_1}, \cdots, \boldsymbol{e_n}$ is known as Gram–Schmidt orthonormalization as the vectors are normalized. This algorithm is numerically unstable leading to loss of orthogonality through rounding errors. The Gram–Schmidt process can be stabilized by a small modification. In this version, sometimes referred to as *modified Gram-Schmidt*, $\boldsymbol{u_k}$ is computed iteratively as:

$$\begin{aligned}
\boldsymbol{u}_k^{(1)} &= \boldsymbol{v}_k - \text{proj}_{\boldsymbol{u}_1}(\boldsymbol{v}_k), \\
\boldsymbol{u}_k^{(2)} &= \boldsymbol{u}_k^{(1)} - \text{proj}_{\boldsymbol{u}_2}(\boldsymbol{u}_k^{(1)}), \\
&\vdots \\
\boldsymbol{u}_k^{(k-2)} &= \boldsymbol{u}_k^{(k-3)} - \text{proj}_{\boldsymbol{u}_{k-2}}(\boldsymbol{u}_k^{(k-3)}), \\
\boldsymbol{u}_k^{(k-1)} &= \boldsymbol{u}_k^{(k-2)} - \text{proj}_{\boldsymbol{u}_{k-1}}(\boldsymbol{u}_k^{(k-2)}), \\
\boldsymbol{u}_k &= \frac{\boldsymbol{u}_k^{(k-1)}}{\|\boldsymbol{u}_k^{(k-1)}\|}
\end{aligned}$$

B.9.4 The QR matrix factorization

B.9.4.1 Definition

Let $\boldsymbol{A} \in \mathbb{C}^{m \times n}$ be a matrix with $m \geq n$. $\boldsymbol{A}$ may be decomposed as the product of an $m \times m$ unitary matrix $\boldsymbol{Q}$ ($\boldsymbol{Q}^{\mathsf{H}}\boldsymbol{Q} = \boldsymbol{Q}\boldsymbol{Q}^{\mathsf{H}} = \boldsymbol{I}$) and an $m \times n$ upper triangular matrix $\boldsymbol{R}$. As the bottom $m - n$ rows of an $m \times n$ upper triangular matrix consist entirely of zeros, it is often useful to partition $\boldsymbol{R}$, or both $\boldsymbol{R}$ and $\boldsymbol{Q}$:

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R} = \boldsymbol{Q}\begin{bmatrix}\boldsymbol{R_1} \\ \boldsymbol{0}\end{bmatrix} = \begin{bmatrix}\boldsymbol{Q_1}, \boldsymbol{Q_2}\end{bmatrix}\begin{bmatrix}\boldsymbol{R_1} \\ \boldsymbol{0}\end{bmatrix} = \boldsymbol{Q_1}\boldsymbol{R_1},$$

where $\boldsymbol{R_1}$ is an $n \times n$ upper triangular matrix, $\boldsymbol{0}$ is an $(m-n) \times n$ zero matrix, $\boldsymbol{Q_1}$ is $m \times n$, $\boldsymbol{Q_2}$ is $m \times (m-n)$, and $\boldsymbol{Q_1}$ and $\boldsymbol{Q_2}$ both have orthogonal columns. $\boldsymbol{Q_1}\boldsymbol{R_1}$ is called the thin QR factorization of $\boldsymbol{A}$ or reduced QR factorization. If $\boldsymbol{A}$ is of full rank n and we require that the diagonal elements of $\boldsymbol{R_1}$ are positive then $\boldsymbol{R_1}$ and $\boldsymbol{Q_1}$ are unique, but in general $\boldsymbol{Q_2}$ is not.

B.9.4.2 Computing the QR decomposition

There are several methods for computing the QR decomposition, such as by means of the Gram–Schmidt process, Householder transformations, or Givens rotations. Each has a number of advantages and disadvantages.

B.9.4.3 Use of QR for solving a linear system

Let $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ be a matrix, we consider the linear problem $\boldsymbol{Ax} = \boldsymbol{b}$. Compared to the direct matrix inverse, solutions using QR decomposition are more numerically stable due to their reduced condition numbers.

To find the solution $\hat{\boldsymbol{x}}$ to the *overdetermined* ($m \geq n$) problem $\boldsymbol{Ax} = \boldsymbol{b}$ which minimizes the norm $\|\boldsymbol{A}\hat{\boldsymbol{x}} - \boldsymbol{b}\|$, first find the QR factorization of $\boldsymbol{A}$ ($\boldsymbol{A} = \boldsymbol{QR}$). The solution can then be expressed as $\hat{\boldsymbol{x}} = \boldsymbol{R_1}^{-1}\left(\boldsymbol{Q_1}^{\top}\boldsymbol{b}\right)$, where $\boldsymbol{Q_1}$ is an $m \times n$ matrix containing the first n columns of the full orthonormal basis $\boldsymbol{Q}$ and where $\boldsymbol{R_1}$ is a square $m \times m$ right triangular matrix.

To solve the *underdetermined* $m < n$ linear problem, first find the QR factorization of the transpose of $\boldsymbol{A}$, *i.e.* $\boldsymbol{A}^{\mathsf{T}} = \boldsymbol{QR}$. $\boldsymbol{Q}$ is orthogonal and $\boldsymbol{R} = \begin{bmatrix} \boldsymbol{R_1} \\ \boldsymbol{0} \end{bmatrix}$ where $\boldsymbol{R_1}$ is as before, and the zero matrix has dimension $(n-m) \times m$. After some algebra, it can be shown that a solution to the linear problem can be expressed as:

$$\boldsymbol{x} = \boldsymbol{Q}\begin{bmatrix} \left(\boldsymbol{R_1}^{\top}\right)^{-1}\boldsymbol{b} \\ \boldsymbol{0} \end{bmatrix}$$

where one may either find $\left(\boldsymbol{R_1}^{\top}\right)^{-1}$ by Gaussian elimination or compute $\left(\boldsymbol{R_1}^{\top}\right)^{-1}\boldsymbol{b}$ directly by forward substitution. The latter technique enjoys greater numerical accuracy and lower computations.

B.9.5 Orthogonal complement

If S is a subset of a Hilbert space H, the set of vectors orthogonal to S, called the orthogonal complement of S, is defined as

$$S^{\perp} = \{\boldsymbol{x} \in H \mid \langle \boldsymbol{x}, \boldsymbol{s} \rangle = 0,\ \forall \boldsymbol{s} \in S\}\,.$$

$S^{\perp}$ is a closed subset of H. Every $\boldsymbol{x} \in H$ can then be written uniquely as $\boldsymbol{x} = \boldsymbol{s} + \boldsymbol{s}^{\perp}$, with $\boldsymbol{s} \in S$ and $\boldsymbol{s}^{\perp} \in S^{\perp}$.

B.10 Matrix similarity

In linear algebra, two n-by-n matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are called *similar* if there exists an invertible n-by-n matrix $\boldsymbol{P}$ such that:

$$\boldsymbol{B} = \boldsymbol{P}^{-1}\boldsymbol{AP}.$$

Similar matrices represent the same linear map under two (possibly) different bases, with $\boldsymbol{P}$ being the change of basis matrix. A transformation $\boldsymbol{A} \mapsto \boldsymbol{P}^{-1}\boldsymbol{AP}$ is called a *similarity transformation* or *conjugation* of the matrix $\boldsymbol{A}$. The matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ share the same eigenvalues (see Sec. B.12).

B.11 Normal matrix

Let $\boldsymbol{A}$ be a complex matrix. $\boldsymbol{A}$ is normal, if and only if, we have:

$$\boldsymbol{A}^{\mathsf{H}}\boldsymbol{A} = \boldsymbol{A}\boldsymbol{A}^{\mathsf{H}}.$$

The *spectral theorem* states that a matrix $\boldsymbol{A}$ is normal if and only if there exists a diagonal matrix $\boldsymbol{\Lambda}$ and a unitary matrix $\boldsymbol{U}$ such that $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{\mathsf{H}}$. Since $\boldsymbol{U}^{-1} = \boldsymbol{U}^{\mathsf{H}}$, the matrix $\boldsymbol{A}$ is similar (see Sec. B.10) to a diagonal matrix $\boldsymbol{\Lambda}$. Since $\boldsymbol{U}$ is unitary, the eigenvectors of $\boldsymbol{A}$ form an orthonormal basis for the usual inner product.

A symmetric matrix $\boldsymbol{C} \in \mathbb{R}^{n\times n}$ is a special case of normal matrix. As a consequence $\boldsymbol{C}$ is necessarily orthogonally diagonalizable. This implies that there always exists an orthogonal matrix $\boldsymbol{S} \in \mathbb{R}^{n\times n}$ (*i.e.* $\boldsymbol{S}^{\top}\boldsymbol{S} = \boldsymbol{I}_n$) such that $\boldsymbol{S}^{-1}\boldsymbol{C}\boldsymbol{S}$ is diagonal. The columns of the matrix $\boldsymbol{S}$ correspond to the eigenvectors of $\boldsymbol{C}$ (see Sec. B.12).

B.12 Eigenvalue Decomposition

B.12.1 Definition

Let $\boldsymbol{A} \in \mathbb{C}^{n\times n}$ be an arbitrary square matrix. $\boldsymbol{v_i} \in \mathbb{C}^n$ and $\lambda_i \in \mathbb{C}$ are eigenvectors/-values if:

$$\boldsymbol{A}\,\boldsymbol{V} = \boldsymbol{V}\,\boldsymbol{\Lambda},$$

with $\boldsymbol{V} = (\boldsymbol{v_1}, \boldsymbol{v_2}, \ldots, \boldsymbol{v_n}) \in \mathbb{C}^{n\times n}$ and $\boldsymbol{\Lambda} = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$.

If $\boldsymbol{A}$ has n linearly independent eigenvectors $\boldsymbol{v_i}$ then $\boldsymbol{A}$ can be factorized as:

$$\boldsymbol{A} = \boldsymbol{V}\,\boldsymbol{\Lambda}\,\boldsymbol{V}^{-1} \quad \text{eigendecomposition of } \boldsymbol{A}.$$

This is equivalent to say that $\boldsymbol{A}$ and $\boldsymbol{\Lambda}$ are similar (see Sec. B.10) with $\boldsymbol{\Lambda}$ a diagonal matrix. The eigenvectors capture the directions in which vectors can grow or shrink (see Fig. B.1).

B.12.2 Application to linear dynamical system

The eigenvalue decomposition is extremely useful to describe the long-term evolution of linear dynamical systems: $\dot{\boldsymbol{x}} = \boldsymbol{A}\,\boldsymbol{x}$. Indeed, we have:

$$\begin{aligned}
\boldsymbol{x}(t) &= \exp(\boldsymbol{A}\,t)\;\boldsymbol{x}(t_0)\,, \\
&= \boldsymbol{V}\,\exp(\boldsymbol{\Lambda}\,t)\;\boldsymbol{V}^{-1}\,\boldsymbol{x}(t_0) \\
&= \sum_{k=1}^{n} \boldsymbol{v_k}\exp(\lambda_k\,t)\,b_k,
\end{aligned}$$

where

- $\boldsymbol{b} = \boldsymbol{V}^{-1}\,\boldsymbol{x}(t_0)$ *i.e.* $\boldsymbol{x}(t_0)$ in the eigenvector basis
- $\operatorname{Re}(\lambda_k)$: growth rate (> 0) ; decay rate (< 0)

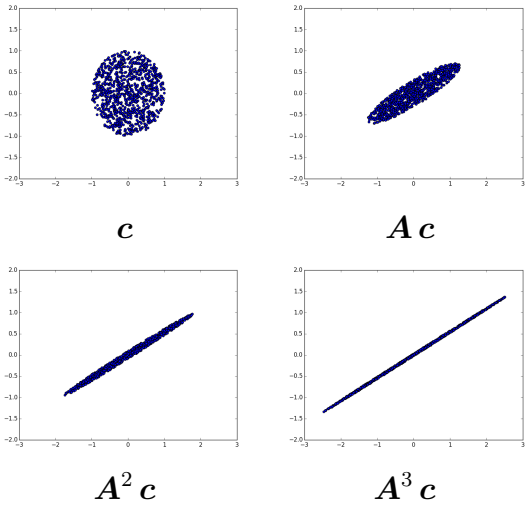


Figure B.1: Successive applications of the matrix $\boldsymbol{A} = \begin{pmatrix} 1.2 & 0.4 \\ 0.5 & 0.5 \end{pmatrix}$ to a set of unit norm initial conditions defined by $\mathcal{C} = \{\boldsymbol{c_i} \mid \|\boldsymbol{c_i}\|_2 = 1\}$.

- $\mathrm{Im}(\lambda_k)$: frequency
- System stable if $\mathrm{Re}(\lambda_k) < 0 \quad \forall k$.

B.13 Singular Value Decomposition

B.13.1 Definition

Let $\boldsymbol{A} \in \mathbb{C}^{m \times n}$ be an arbitrary matrix. We note $\boldsymbol{A}^{\mathsf{H}}$ the Hermitian or complex conjugate transpose matrix of $\boldsymbol{A}$ ($\boldsymbol{A}^{\mathsf{H}} = \overline{\boldsymbol{A}^{\top}}$). If $\boldsymbol{A}$ is real, then $\boldsymbol{A}^{\mathsf{H}} = \boldsymbol{A}^{\top}$.

The *Singular Value Decomposition* or SVD is a generalization of the eigenvalue decomposition to non-square matrices. Every matrix $\boldsymbol{A}$ can be decomposed as:

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathsf{H}} \tag{B.2}$$

where

- $\boldsymbol{U} \in \mathbb{C}^{m \times m}$ is the unitary[2] matrix ($\boldsymbol{U}^{\mathsf{H}}\boldsymbol{U} = \boldsymbol{U}\boldsymbol{U}^{\mathsf{H}} = \boldsymbol{I}_m$) with columns of *left-singular vectors* $\boldsymbol{u_i}$, *i.e.* $\boldsymbol{U} = (\boldsymbol{u_1}, \boldsymbol{u_2}, \cdots, \boldsymbol{u_m})$.
- $\boldsymbol{\Sigma} \in \mathbb{R}^{m \times n}$ is the diagonal matrix of singular values, *i.e.*

$$\boldsymbol{\Sigma} = \mathrm{Diag}\,(\sigma_1, \sigma_2, \cdots, \sigma_p, 0, \cdots, 0)$$

[2] If $\boldsymbol{A}$ is real then $\boldsymbol{U}$ is orthogonal, *i.e.* $\boldsymbol{U}^{\top}\boldsymbol{U} = \boldsymbol{U}\boldsymbol{U}^{\top} = \boldsymbol{I}_m$.

with $p = \min(m, n)$. The SVD is not unique but it is always possible to choose the decomposition so that the singular values σ_i are in descending order, *i.e.*

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \cdots = \sigma_p = 0$$

where r the rank of $\boldsymbol{A}$ is the number of non-zero singular values ($r \leq p = \min(m, n)$).

- $\boldsymbol{V}^{\mathsf{H}} \in \mathbb{C}^{n \times n}$ is the unitary matrix ($\boldsymbol{V}^{\mathsf{H}}\boldsymbol{V} = \boldsymbol{V}\boldsymbol{V}^{\mathsf{H}} = \boldsymbol{I}_n$) with columns of *right-singular vectors* $\boldsymbol{v_i}$, *i.e.* $\boldsymbol{V} = (\boldsymbol{v_1}, \boldsymbol{v_2}, \cdots, \boldsymbol{v_n})$.

The range of $\boldsymbol{A}$ is given by $\mathcal{R}(\boldsymbol{A}) = \mathrm{Span}(\boldsymbol{u_1}, \boldsymbol{u_2}, \cdots, \boldsymbol{u_r})$ and the null space of $\boldsymbol{A}$ by $\mathcal{N}(\boldsymbol{A}) = \mathrm{Span}(\boldsymbol{v_{r+1}}, \cdots, \boldsymbol{v_n})$.

B.13.2 Examples

B.13.2.1 Case $n > m$ *i.e.* $p = m$

We consider the SVD decomposition of $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathsf{H}}$ where $\boldsymbol{A}$ has more columns than rows ($n > m$). The case where $n \gg m$ is called "short/fat" matrix.

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{u_1} & \cdots & \boldsymbol{u_m} \end{pmatrix} \left(\begin{array}{cccc|cccc} \sigma_1 & & & & 0 & \cdots & \cdots & 0 \\ & \ddots & & & \vdots & & & \vdots \\ & & \ddots & & \vdots & & & \vdots \\ & & & \sigma_m & 0 & \cdots & \cdots & 0 \end{array}\right) \left(\begin{array}{c} \boldsymbol{v_1}^{\mathsf{H}} \\ \vdots \\ \vdots \\ \boldsymbol{v_m}^{\mathsf{H}} \\ \hline \boldsymbol{v_{m+1}}^{\mathsf{H}} \\ \vdots \\ \vdots \\ \boldsymbol{v_n}^{\mathsf{H}} \end{array}\right)$$

B.13.2.2 Case $m > n$ *i.e.* $p = n$

We consider the SVD decomposition of $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathsf{H}}$ where $\boldsymbol{A}$ has more rows than columns ($m > n$). The case where $m \gg n$ is called "tall/skinny" matrix.

$$\boldsymbol{A} = \left(\begin{array}{cccccc} \boldsymbol{u_1} & \cdots & \boldsymbol{u_n} & \boldsymbol{u_{n+1}} & \cdots & \boldsymbol{u_m} \end{array} \right) \left(\begin{array}{ccccc} \sigma_1 & & & & \\ & \ddots & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & \sigma_n \\ \hline 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & & & & \vdots \\ \vdots & & & & \vdots \\ 0 & \cdots & \cdots & \cdots & 0 \end{array} \right) \left(\begin{array}{c} \boldsymbol{v_1}^{\mathsf{H}} \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \boldsymbol{v_n}^{\mathsf{H}} \end{array} \right).$$

B.13.3 Truncated SVD approximations: Dyadic expansion

If $r = \text{rank}(\boldsymbol{A})$, then the SVD of $\boldsymbol{A} \in \mathbb{C}^{m \times n}$ can be written as

$$\boldsymbol{A} = \left(\begin{array}{cc} \underline{\boldsymbol{U}}_{m \times r} & \overline{\boldsymbol{U}}_{m \times (m-r)} \end{array} \right) \left(\begin{array}{cc} \underline{\boldsymbol{\Sigma}}_{r \times r} & 0 \\ 0 & 0 \end{array} \right) \left(\begin{array}{cc} \underline{\boldsymbol{V}}_{n \times r} & \overline{\boldsymbol{V}}_{n \times (n-r)} \end{array} \right)^{\mathsf{H}}$$

$$\boldsymbol{A} = \underline{\boldsymbol{U}}_{m \times r} \underline{\boldsymbol{\Sigma}}_{r \times r} \underline{\boldsymbol{V}}^{\mathsf{H}}_{n \times r}$$

i.e.

$$\boldsymbol{A} = \sigma_1 \, \boldsymbol{u_1} \boldsymbol{v_1}^{\mathsf{H}} + \sigma_2 \, \boldsymbol{u_2} \boldsymbol{v_2}^{\mathsf{H}} + \cdots + \sigma_r \, \boldsymbol{u_r} \boldsymbol{v_r}^{\mathsf{H}}.$$

If we truncate this expansion to $k < r$ terms, then

$$\boldsymbol{A}_k = \boldsymbol{U}_k \boldsymbol{\Sigma}_k \boldsymbol{V}_k^{\mathsf{H}} = \sigma_1 \, \boldsymbol{u_1} \boldsymbol{v_1}^{\mathsf{H}} + \sigma_2 \, \boldsymbol{u_2} \boldsymbol{v_2}^{\mathsf{H}} + \cdots + \sigma_k \, \boldsymbol{u_k} \boldsymbol{v_k}^{\mathsf{H}}.$$

The matrix $\boldsymbol{A}_k$ is an approximation of $\boldsymbol{A}$. The quality of the approximation is evaluated by Eckart Young's theorem (see Sec. B.13.6).

B.13.4 Geometric interpretation

The SVD factorization maps the sphere S of radius one in $\mathbb{R}^n$ onto an ellipsoid in $\mathbb{R}^m$. Non zero singular values are simply the lengths of the semi-axes of this ellipsoid. When $n = m$, the SVD can be interpreted as a succession of three consecutive moves: an initial rotation $\boldsymbol{V}^{\mathsf{H}}$, a scaling $\boldsymbol{\Sigma}$ along the coordinate axes, and a final rotation $\boldsymbol{U}$ (see Fig. B.2).

The relation

$$\boldsymbol{A} \boldsymbol{v_i} = \boldsymbol{U} \boldsymbol{\Sigma} \boldsymbol{V}^{\mathsf{H}} \boldsymbol{v_i} = \boldsymbol{U} \boldsymbol{\Sigma} \boldsymbol{e_i} = \sigma_i \boldsymbol{u_i} \quad i = 1, \cdots, r$$

shows that $\boldsymbol{A}$ maps input $\boldsymbol{v_i}$ to output $\boldsymbol{u_i}$ with amplification σ_i. We deduce from it that

- the columns $\boldsymbol{u_i}, i = 1, \cdots, r$ define an orthonormal basis of $\boldsymbol{A}$,
- the columns $\boldsymbol{v_i}, i = 1, \cdots, r$ define an orthonormal basis of $\boldsymbol{A}^{\mathsf{H}}$, and

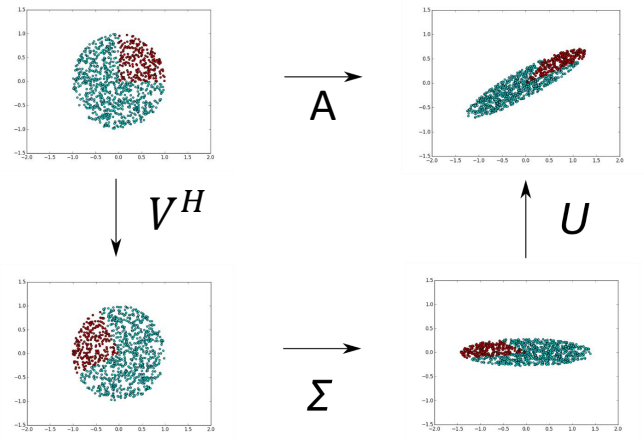


Figure B.2: Geometric interpretation of SVD.

- the singular values σ_i indicate amplification factors.

B.13.5 Links between SVD and eigenvalue problems

The SVD and the eigenvalue problems are directly linked. Some algebraic manipulations thus allow to connect the "classical" POD and "snapshot" POD approaches.

- Classical POD (Lumley, 1967)

$$\begin{aligned}\boldsymbol{A}\boldsymbol{A}^{\mathsf{H}} &= \left(\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathsf{H}}\right)\left(\boldsymbol{V}\boldsymbol{\Sigma}^{\mathsf{H}}\boldsymbol{U}^{\mathsf{H}}\right) = \boldsymbol{U}\boldsymbol{\Sigma}\underbrace{\boldsymbol{V}^{\mathsf{H}}\boldsymbol{V}}_{\boldsymbol{I}_n}\boldsymbol{\Sigma}^{\mathsf{H}}\boldsymbol{U}^{\mathsf{H}} \\ &= \boldsymbol{U}\boldsymbol{\Sigma}^2\boldsymbol{U}^{\mathsf{H}} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{\mathsf{H}}\end{aligned}$$

  $\implies \left(\boldsymbol{A}\boldsymbol{A}^{\mathsf{H}}\right)\boldsymbol{U} = \boldsymbol{U}\boldsymbol{\Sigma}^2 = \boldsymbol{U}\boldsymbol{\Lambda}$, *i.e.* the columns of $\boldsymbol{U}$ are the eigenvectors of $\boldsymbol{A}\boldsymbol{A}^{\mathsf{H}} \in \mathbb{C}^{m\times m}$.

- Snapshot POD (Sirovich, 1987)

$$\begin{aligned}\boldsymbol{A}^{\mathsf{H}}\boldsymbol{A} &= \left(\boldsymbol{V}\boldsymbol{\Sigma}^{\mathsf{H}}\boldsymbol{U}^{\mathsf{H}}\right)\left(\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathsf{H}}\right) = \boldsymbol{V}\boldsymbol{\Sigma}^{\mathsf{H}}\underbrace{\boldsymbol{U}^{\mathsf{H}}\boldsymbol{U}}_{\boldsymbol{I}_m}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathsf{H}} \\ &= \boldsymbol{V}\boldsymbol{\Sigma}^2\boldsymbol{V}^{\mathsf{H}} \\ &= \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{\mathsf{H}}\end{aligned}$$

  $\implies \left(\boldsymbol{A}^{\mathsf{H}}\boldsymbol{A}\right)\boldsymbol{V} = \boldsymbol{V}\boldsymbol{\Sigma}^2 = \boldsymbol{V}\boldsymbol{\Lambda}$, *i.e.* the columns of $\boldsymbol{V}$ are the eigenvectors of $\boldsymbol{A}^{\mathsf{H}}\boldsymbol{A} \in \mathbb{C}^{n\times n}$.

- Singular values

$$\sigma_i = \sqrt{\lambda_i(\boldsymbol{A}^{\mathsf{H}}\boldsymbol{A})} = \sqrt{\lambda_i(\boldsymbol{A}\boldsymbol{A}^{\mathsf{H}})} \quad i = 1, \cdots, r$$

B.13.6 Low rank approximation: Eckart-Young theorem

Let $\boldsymbol{A} \in \mathbb{C}^{m\times n}$, the objective is to determine $\boldsymbol{A}_k \in \mathbb{C}^{m\times n}$ such that $\text{rank}(\boldsymbol{A}_k) = k < \text{rank}(\boldsymbol{A})$. $\boldsymbol{A}_k$ is found as the solution that minimizes the Frobenius norm (see the definition in Sec. B.4.3) of the error $\boldsymbol{A} - \boldsymbol{A}_k$. The Eckart-Young theorem states that

$$\min_{\text{rank}\,(\boldsymbol{X})\leq k} \|\boldsymbol{A} - \boldsymbol{X}\|_{\mathsf{F}} = \|\boldsymbol{A} - \boldsymbol{A}_k\|_{\mathsf{F}} = \sqrt{\sum_{i=k+1}^{r} \sigma_i^2(\boldsymbol{A})}$$

with

$$\boldsymbol{A}_k = \boldsymbol{U}\begin{pmatrix} \boldsymbol{\Sigma}_k & 0 \\ 0 & 0 \end{pmatrix}\boldsymbol{V}^{\mathsf{H}} = \sigma_1 \boldsymbol{u_1}\boldsymbol{v_1}^{\mathsf{H}} + \sigma_2 \boldsymbol{u_2}\boldsymbol{v_2}^{\mathsf{H}} + \cdots + \sigma_k \boldsymbol{u_k}\boldsymbol{v_k}^{\mathsf{H}}.$$

This theorem establishes a relationship between the rank k of the approximation, and the singular values of $\boldsymbol{A}$.

B.14 Moore-Penrose inverse

The Moore-Penrose inverse $\boldsymbol{A}^+$ of a matrix $\boldsymbol{A}$ is the most widely known generalization of the inverse matrix. The term *pseudoinverse* is often used to indicate the Moore-Penrose inverse. The pseudoinverse is defined and unique for all matrices whose entries are real or complex numbers. It can be computed using the singular value decomposition.

B.14.1 Definition

Let K be the fields of real numbers $\mathbb{R}$ or complex numbers $\mathbb{C}$. The vector space of $m \times n$ matrices over K is denoted by $K^{m\times n}$.

For $\boldsymbol{A} \in K^{m\times n}$, a pseudoinverse of $\boldsymbol{A}$ is defined as a matrix $\boldsymbol{A}^+ \in K^{m\times n}$ satisfying all of the following four criteria, known as the Moore-Penrose conditions:

1. $\boldsymbol{A}\boldsymbol{A}^+\boldsymbol{A} = \boldsymbol{A}$
   $\boldsymbol{A}\boldsymbol{A}^+$ need not be the general identity matrix, but it maps all column vectors of $\boldsymbol{A}$ to themselves;

2. $\boldsymbol{A}^+\boldsymbol{A}\boldsymbol{A}^+ = \boldsymbol{A}^+$

3. $\left(\boldsymbol{A}\boldsymbol{A}^+\right)^{\mathsf{H}} = \boldsymbol{A}\boldsymbol{A}^+$
   $\boldsymbol{A}\boldsymbol{A}^+$ is Hermitian;

4. $\left(\boldsymbol{A}^+\boldsymbol{A}\right)^{\mathsf{H}} = \boldsymbol{A}^+\boldsymbol{A}$
   $\boldsymbol{A}^+\boldsymbol{A}$ is also Hermitian.

B.14.2 Left and right inverse

$\boldsymbol{A}^+$ exists for any matrix $\boldsymbol{A}$, but, when the latter has full rank (*i.e.* the rank of $\boldsymbol{A}$ is $\min(m, n)$), then $\boldsymbol{A}^+$ can be expressed as a simple algebraic formula.

In particular, when $\boldsymbol{A}$ has linearly independent columns (and thus matrix $\boldsymbol{A}^{\mathsf{H}}\boldsymbol{A}$ is invertible), $\boldsymbol{A}^{+}$ can be computed as

$$\boldsymbol{A}^{+} = \left(\boldsymbol{A}^{\mathsf{H}}\boldsymbol{A}\right)^{-1}\boldsymbol{A}^{\mathsf{H}}.$$

This particular pseudoinverse constitutes a *left inverse*, since, in this case, $\boldsymbol{A}^{+}\boldsymbol{A} = \boldsymbol{I}$.

When $\boldsymbol{A}$ has linearly independent rows (matrix $\boldsymbol{A}\boldsymbol{A}^{\mathsf{H}}$ is invertible), $\boldsymbol{A}^{+}$ can be computed as

$$\boldsymbol{A}^{+} = \boldsymbol{A}^{\mathsf{H}}\left(\boldsymbol{A}\boldsymbol{A}^{\mathsf{H}}\right)^{-1}.$$

This is a *right inverse*, as $\boldsymbol{A}\boldsymbol{A}^{+} = \boldsymbol{I}$.

B.14.3 Pseudoinverse by SVD

A computationally simple and accurate way to compute the pseudoinverse is by using the SVD. If $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathsf{H}}$ is the singular value decomposition of $\boldsymbol{A}$, then

$$\boldsymbol{A}^{+} = \boldsymbol{V}\boldsymbol{\Sigma}^{+}\boldsymbol{U}^{\mathsf{H}}.$$

For a rectangular diagonal matrix such as $\boldsymbol{\Sigma}$, we get the pseudoinverse by taking the reciprocal of each non-zero element on the diagonal, leaving the zeros in place, and then transposing the matrix. In numerical computation, only elements larger than some small tolerance are taken to be nonzero, and the others are replaced by zeros. For example, in the MATLAB, GNU Octave, or NumPy function `pinv`, the tolerance is taken to be $t = \epsilon \cdot \max(m, n) \cdot \max(\boldsymbol{\Sigma})$, where ϵ is the machine precision.

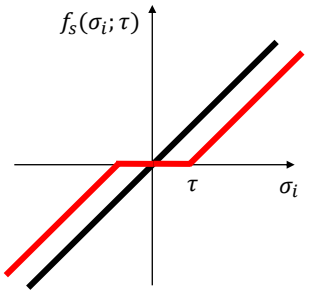
B.15 Thresholded SVD

The thresholded SVD approach is classically used to solve a linear system of equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ when the matrix $\boldsymbol{A}$ is numerically ill-conditioned (presence of noise for example) and may cause round-off error (Gavish and Donoho, 2014). This method relies on the use of the SVD decomposition of $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\top}$. The principle is to determine $\boldsymbol{x}$ by only keeping the singular triplets $(\sigma, \boldsymbol{u}, \boldsymbol{v})$ with singular values larger than a given positive threshold which is dictated by the value of threshold level τ fixed *a priori*. This is equivalent to empirically searching for the rank of the matrix $\boldsymbol{A}$, and to use a pseudo-inverse approach for determining $\boldsymbol{x}$. In the algorithms, a difference is made between *hard-thresholding* and *soft-thresholding* (see Murphy, 2012, p. 433).

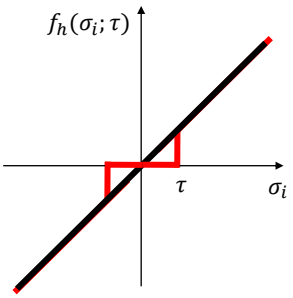
Let $\boldsymbol{\Sigma}_{+}$, the matrix of filtered singular values[3], be defined as $\boldsymbol{\Sigma}_{+} = \mathrm{Diag}((\sigma_i)_{+})$. In hard-thresholding, the filter function f_h is defined as

$$f_h(\sigma_i; \tau) = (\sigma_i)_{+} = \begin{cases} \sigma_i, & \text{if } \sigma_i > \tau \\ 0, & \text{otherwise} \end{cases},$$

[3] This matrix should not be confused with $\boldsymbol{\Sigma}^{+}$ defined in Sec. B.14.3.



(a) Soft thresholding.



(b) Hard thresholding.

Figure B.3: Comparison between soft thresholding and hard thresholding. The flat region is the interval $[-\tau;\tau]$.

whereas in soft-thresholding, the filter function f_s is defined as

$$f_s(\sigma_i;\tau) = (\sigma_i)_+ = \begin{cases} \sigma_i - \tau, & \text{if } \sigma_i > \tau \\ 0, & \text{otherwise} \end{cases}.$$

Algorithm B.1: Thresholded SVD solution of a linear system of equations

Input: $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ with $m \geq n$, $\boldsymbol{b} \in \mathbb{R}^m$ and threshold level $\tau \geq 0$
Output: Solution $\boldsymbol{x} \in \mathbb{R}^n$
$\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}^\top \leftarrow \mathbf{svd}(\boldsymbol{A})$
$r \leftarrow 1$
$\sigma_{\text{Max}} \leftarrow \tau \cdot \boldsymbol{\Sigma}(1) \cdot \mathbf{max}(\mathbf{size}(\boldsymbol{A}))$
while $r < min(m,n)$ **and** $\boldsymbol{\Sigma}(r) \geq \sigma_{Max}$ **do**
  $r \leftarrow r + 1$
$\boldsymbol{z} \leftarrow \boldsymbol{U}^\top \boldsymbol{b}$
$\boldsymbol{w}(1:r) \leftarrow \boldsymbol{z}(1:r)/\boldsymbol{\Sigma}_+(1:r)$
$\boldsymbol{x} \leftarrow \boldsymbol{V}\ [w \quad \mathbf{zeros}(n-r)]^\top$
return $\boldsymbol{x}$

Appendix C

Linear regression model

Contents

In this appendix, we consider the general linear regression model given by

$$y = \beta_0 + \sum_{j=1}^{p} x_j \beta_j \tag{C.1}$$

where

- y is the *dependent variable*;
- x_j, $j = 1, \cdots, p$ the p *regressors*;
- β_0 the *intercept* or *bias*;
- β_j, $j = 1, \cdots, p$ the p *parameters*.

All these quantities are real scalars. We consider that we have a set of training data y_i, $(x_j)_i$, $i = 1, \cdots, n$. Let $\boldsymbol{y} \in \mathbb{R}^{n \times 1}$ be the n-vector of outputs in the training set, and $\boldsymbol{X} \in \mathbb{R}^{n \times p}$ be the *design matrix*. This matrix can be defined in terms of its column vectors, *i.e.*

$$\boldsymbol{X^j} = \begin{bmatrix} (x_j)_1 \\ (x_j)_2 \\ \vdots \\ (x_j)_n \end{bmatrix} \in \mathbb{R}^n \quad (j = 1, \cdots, p) \quad \text{and} \quad \boldsymbol{X} = \begin{bmatrix} | & | & & | \\ \boldsymbol{X^1} & \boldsymbol{X^2} & \dots & \boldsymbol{X^p} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times p} \tag{C.2}$$

or, in terms of its row vectors, here written in columns:

$$\boldsymbol{X_i} = \begin{bmatrix} (x_1)_i \\ (x_2)_i \\ \vdots \\ (x_p)_i \end{bmatrix} \in \mathbb{R}^p \quad (i = 1, \cdots, n) \quad \text{and} \quad \boldsymbol{X} = \begin{bmatrix} — & \boldsymbol{X_1}^\top & — \\ — & \boldsymbol{X_2}^\top & — \\ & \vdots & \\ — & \boldsymbol{X_n}^\top & — \end{bmatrix} \in \mathbb{R}^{n \times p}. \tag{C.3}$$

Each column vectors $\boldsymbol{X^j}$ represent the n training data of the j-th predictor, whereas each row vector represent the p regressors for one particular training data i. Since we have n training data, we deduce from (C.1) that:

$$y_i = \beta_0 + \sum_{j=1}^{p} (x_j)_i \beta_j = \beta_0 + \boldsymbol{X_i}^\top \boldsymbol{\beta} \quad (i = 1, \cdots, n) \text{ where } \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix} \in \mathbb{R}^p. \tag{C.4}$$

If we now define an extended version of $\boldsymbol{X}$ and $\boldsymbol{\beta}$ as

$$\boldsymbol{X}_e = \begin{bmatrix} | & | & | & & | \\ \mathbf{1}_n & \boldsymbol{X^1} & \boldsymbol{X^2} & \dots & \boldsymbol{X^p} \\ | & | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times (p+1)} \quad \text{and} \quad \boldsymbol{\beta}_e = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix} \in \mathbb{R}^{p+1}, \tag{C.5}$$

then (C.4) simplifies as $\boldsymbol{y} = \boldsymbol{X}_e \boldsymbol{\beta}_e$.

C.1 Standardization, or mean removal and variance scaling

Standardization of datasets is a common requirement for many machine learning estimators. The objective of this section is to explain and to justify the standardization procedure of the variables.

We define $\overline{y} \in \mathbb{R}$, the average of y_i over the n training data. Starting from (C.4), we deduce that

$$\overline{y} = \frac{1}{n} \sum_{i=1}^{n} y_i = \beta_0 + \sum_{j=1}^{p} \left[\frac{1}{n} \sum_{i=1}^{n} (x_j)_i \right] \beta_j \tag{C.6}$$

$$= \beta_0 + \sum_{j=1}^{p} \overline{x_j} \beta_j \tag{C.7}$$

$$= \beta_0 + \overline{\boldsymbol{X}}^\top \boldsymbol{\beta} \tag{C.8}$$

where

$$\overline{x_j} = \frac{1}{n}\sum_{i=1}^{n}(x_j)_i \quad \text{and} \quad \overline{\boldsymbol{X}} = \begin{bmatrix} \overline{x_1} \\ \overline{x_2} \\ \vdots \\ \overline{x_p} \end{bmatrix} \in \mathbb{R}^p. \tag{C.9}$$

From (C.8), we deduce that the estimated value of β_0 is given by:

$$\widehat{\beta_0} = \overline{y} - \overline{\boldsymbol{X}}^\top \boldsymbol{\beta}. \tag{C.10}$$

By inserting (C.10) into (C.4), we obtain:

$$y_i - \overline{y} = \left(\boldsymbol{X_i} - \overline{\boldsymbol{X}}\right)^\top \boldsymbol{\beta} \tag{C.11}$$

where

$$\boldsymbol{X_i} - \overline{\boldsymbol{X}} = \begin{bmatrix} (x_1)_i - \overline{x_1} \\ (x_2)_i - \overline{x_2} \\ \vdots \\ (x_p)_i - \overline{x_p} \end{bmatrix} \in \mathbb{R}^p.$$

Equation (C.11) is a modified version of the original problem (C.4) where we have removed the mean of the dependent and independent variables, without changing the unknowns. Starting from the solution $\boldsymbol{\beta}$ of (C.11), we can deduce the value of the intercept by using (C.10).

Starting with the Penalized Least Squares (PLS), it is common in Statistical Learning to introduce constraints that penalize the value of the coefficients for each regressor. This value should not depend on the magnitude of each variable. For this reason, we center and reduce, or standardize, the different variables. This is equivalent to introduce *score* variables. If μ and σ are the mean and standard deviation of a variable x, the standard score is defined as:

$$z = \frac{x - \mu}{\sigma}. \tag{C.12}$$

Hence, we define the standard deviation of the output y as

$$\sigma(y) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \overline{y})^2 \tag{C.13}$$

and the standard deviation of each regressor x_j $(j = 1, \cdots, p)$ as

$$\sigma\left(x_j\right) = \frac{1}{n}\sum_{i=1}^{n}\left[(x_j)_i - \overline{x_j}\right]^2. \tag{C.14}$$

After standardization of the output and regressor variables, (C.11) becomes:

$$\frac{y_i - \overline{y}}{\sigma(y)} = \begin{bmatrix} \dfrac{(x_1)_i - \overline{x_1}}{\sigma(x_1)} \\ \dfrac{(x_2)_i - \overline{x_2}}{\sigma(x_2)} \\ \vdots \\ \dfrac{(x_p)_i - \overline{x_p}}{\sigma(x_p)} \end{bmatrix}^\top \boldsymbol{\gamma} \quad \text{where} \quad \boldsymbol{\gamma} = \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_p \end{bmatrix} \in \mathbb{R}^p. \tag{C.15}$$

The standardized coefficients obtained as solution of (C.15) are linked to the unstandardized coefficients, solution of (C.11) by the relations:

$$\gamma_j = \beta_j \frac{\sigma(x_j)}{\sigma(y)} \qquad j = 1, \cdots, p. \tag{C.16}$$

Standardizing variables before regression leads to:

1. no need for intercept,
2. the ability to rank the coefficient importance by the relative magnitude of post-shrinkage coefficient estimates, facilitating the interpretability of coefficients.

C.2 Ordinary least-squares (OLS) and Penalized least-squares (PLS)

In the language of optimization, OLS corresponds to the following optimization problem:

$$\widehat{\boldsymbol{\beta}}_{\text{OLS}} = \arg\min_{\boldsymbol{\beta}} \|\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y}\|_2^2 = \arg\min_{\boldsymbol{\beta}} \|\boldsymbol{\epsilon}\|_2^2 . \tag{C.17}$$

The solution of (C.17) is given by

$$\widehat{\boldsymbol{\beta}}_{\text{OLS}} = \left(\boldsymbol{X}^\top \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{y} = \boldsymbol{X}^+ \boldsymbol{y}, \tag{C.18}$$

where $\square^+$ represents the Moore-Penrose pseudoinverse (see App. B.14). It is interesting to note that OLS can also have a geometric interpretation, leading to find $\widehat{\boldsymbol{\beta}}$ as the solution of a successive orthogonalization algorithm (Alg. 3.1 of Hastie et al., 2009). Least squares estimates are often not satisfying for two reasons:

1. their prediction accuracy associated to low bias and large variance,
2. the interpretability of the solution.

The OLS solution is also known to generate an over-fitting phenomenon. To avoid this, one approach consists in penalizing the solution vector using so-called *shrinking methods*. As a first example, we introduce the *Ridge Regression* or *Penalized least-squares* problem

defined by:

$$\widehat{\boldsymbol{\beta}}_{\text{PLS}} = \arg\min_{\boldsymbol{\beta}} \left(\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} (x_j)_i \beta_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right), \tag{C.19}$$

$$= \arg\min_{\boldsymbol{\beta}} \left(\|\boldsymbol{X}_e \boldsymbol{\beta}_e - \boldsymbol{y}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_2^2 \right), \tag{C.20}$$

where λ is a hyperparameter. The intercept β_0 is not introduced in the penalty term. It can be shown (Hastie et al., 2009, p. 64) that penalizing the intercept would make the procedure depending on the origin of the output. We also show in (C.11) that by centering the data, we can separate the determination of the coefficient $\boldsymbol{\beta}$ in two subproblems, β_0 on one hand, and the remaining components, on the other hand. If we assume that the centering has been done, then (C.19) is equivalent to

$$\widehat{\boldsymbol{\beta}}_{\text{PLS}} = \arg\min_{\boldsymbol{\beta}} \left(\|\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_2^2 \right), \tag{C.21}$$

where centered variables are used. An equivalent way to write the PLS problem is

$$\widehat{\boldsymbol{\beta}}_{\text{PLS}} = \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} (x_j)_i \beta_j \right)^2, \tag{C.22}$$

$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 \leq t \tag{C.23}$$

There is a one-to-one correspondence between the parameter λ in (C.19) and t in (C.23). We can show that the solution of (C.21) is given by:

$$\widehat{\boldsymbol{\beta}}_{\text{PLS}} = \left(\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I}_p \right)^{-1} \boldsymbol{X}^\top \boldsymbol{y}. \tag{C.24}$$

As the value of λ increases, the solution becomes more robust to possible problems of collinearity of the column vectors of the matrix $\boldsymbol{X}$. To set the regularization parameter λ, we can use generalized cross-validation technique (leave-one-out cross validation).

For simplicity, we can assume that $\boldsymbol{X}$ is composed of orthonormal variables, *i.e.* $\boldsymbol{X}^\top \boldsymbol{X} = \boldsymbol{I}$. It then follows that

$$\widehat{\boldsymbol{\beta}}_{\text{PLS}} = \left(\boldsymbol{I} + \lambda \boldsymbol{I}_p \right)^{-1} \boldsymbol{X}^\top \boldsymbol{y} = \left((1+\lambda)\, \boldsymbol{I}_p \right)^{-1} \boldsymbol{X}^\top \boldsymbol{y} = \frac{1}{1+\lambda} \widehat{\boldsymbol{\beta}}_{\text{OLS}}. \tag{C.25}$$

This shows that the ridge estimator is simply a downweighted version of the OLS estimator.

Some additional insight can be obtained by using the SVD decomposition of the centered matrix $\boldsymbol{X}$. The SVD of the $n \times p$ matrix $\boldsymbol{X}$ has the form (see App. B.13):

$$\boldsymbol{X} = \boldsymbol{U} \boldsymbol{\Sigma} \boldsymbol{V}^\top. \tag{C.26}$$

Using this decomposition, we can write the least squares fitted vector as[1]

$$\widehat{\boldsymbol{y}}_{\text{OLS}} = \boldsymbol{X}\widehat{\boldsymbol{\beta}}_{\text{OLS}} \tag{C.27}$$

$$= \boldsymbol{X}\left(\boldsymbol{X}^\top\boldsymbol{X}\right)^{-1}\boldsymbol{X}^\top\boldsymbol{y}$$

$$= \boldsymbol{U}\boldsymbol{U}^\top\boldsymbol{y}. \tag{C.28}$$

Here $\boldsymbol{U}^\top\boldsymbol{y}$ are the coordinates of $\boldsymbol{y}$ with respect to the orthonormal basis $\boldsymbol{U}$. Similarly, the ridge solution is given by

$$\widehat{\boldsymbol{y}}_{\text{PLS}} = \boldsymbol{X}\widehat{\boldsymbol{\beta}}_{\text{PLS}} \tag{C.29}$$

$$= \boldsymbol{X}\left(\boldsymbol{X}^\top\boldsymbol{X} + \lambda\boldsymbol{I}_p\right)^{-1}\boldsymbol{X}^\top\boldsymbol{y}$$

$$= \sum_{j=1}^{p}\boldsymbol{u_j}\frac{\sigma_j^2}{\sigma_j^2+\lambda}\boldsymbol{u_j}^\top\boldsymbol{y}, \tag{C.30}$$

where the $\boldsymbol{u_j}$ are the columns of $\boldsymbol{U}$. Since $\lambda \geq 0$, we have $\frac{\sigma_j^2}{\sigma_j^2+\lambda} \leq 1$. The effect of this term is then to shrink the coordinates of $\boldsymbol{y}$ with respect to the orthonormal basis $\boldsymbol{U}$.

C.3 Least Absolute Shrinkage and Selection Operator (LASSO)

The LASSO estimate is defined as

$$\widehat{\boldsymbol{\beta}}_{\text{LASSO}} = \arg\min_{\boldsymbol{\beta}}\left(\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\left(x_j\right)_i\beta_j\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j|\right), \tag{C.31}$$

$$= \arg\min_{\boldsymbol{\beta}}\left(\sum_{i=1}^{n}\left(y_i - \langle\boldsymbol{X_i},\boldsymbol{\beta}\rangle\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j|\right), \tag{C.32}$$

where $\boldsymbol{X}_i$ denotes the i-th row of $\boldsymbol{X}$ and $\langle\cdot,\cdot\rangle$ represents the Euclidean inner product. This last expression will be interesting for discussing LASSO in terms of geometry. An equivalent form for the LASSO problem is

$$\widehat{\boldsymbol{\beta}}_{\text{LASSO}} = \arg\min_{\boldsymbol{\beta}}\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\left(x_j\right)_i\beta_j\right)^2, \tag{C.33}$$

$$\text{subject to } \sum_{j=1}^{p}|\beta_j| \leq t \tag{C.34}$$

Similarly as in App. C.2, we can re-parametrize the problem by standardizing the

[1] We assume in the last line that we applied the economy SVD. Hence, $\boldsymbol{U}$, and possibly $\boldsymbol{V}^\top$, are now *semi-unitary* matrices, which means that only $\boldsymbol{U}^\top\boldsymbol{U} = \boldsymbol{V}^\top\boldsymbol{V} = \boldsymbol{I}$.

output and the regressors. We then arrive at

$$\widehat{\boldsymbol{\beta}}_{\text{LASSO}} = \underset{\boldsymbol{\beta}}{\arg\min} \left(\|\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_1 \right), \tag{C.35}$$

where $\boldsymbol{X}$ is now an $n \times p$ matrix.

If we assume that $n = p$ and $\boldsymbol{X}$ is orthogonal, we can show that

$$\widehat{\boldsymbol{\beta}}_{\text{LASSO}} = \left(\widehat{\boldsymbol{\beta}}_{\text{OLS}} - \lambda \right)_+ \tag{C.36}$$

where $(\cdot)_+$ corresponds here to the soft-thresholding operator introduced in App. B.15.

Appendix D

Elements of statistics

Contents

Most of the results presented in this appendix are from Taboga (2021).

D.1 Random variable

A random variable X is a function defined from a sample space Ω to a measurable space. The value of X at a point $\omega \in \Omega$ is the realization $x = X(\omega)$ of X.

D.2 Indicator function

For any given set A, the indicator function is defined as

$$\mathbb{1}_A(x) := \begin{cases} 1 & \text{if } x \in A \,, \\ 0 & \text{if } x \notin A \,. \end{cases} \tag{D.1}$$

D.3 Empirical distribution

D.3.1 Definition

Let $\boldsymbol{x} = \{x_1, \ldots, x_n\}$ be a sample of size n of a random variable X, where $x_1, \cdots, x_n$ are the n observations from the sample. The empirical distribution $\widehat{F}$ is defined by giving

the mass $1/n$ at each data point x_i, *i.e.*

$$\widehat{F}(x) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{x_i \leq x}(x) \tag{D.2}$$

where $\mathbb{1}_{x_i \leq x}$, the indicator function, is equal to 1 if $x_i \leq x$ and 0 otherwise (see Sec. D.2).

In other words, the value of the empirical distribution function at a given point x is obtained by:

1. counting the number of observations that are less than or equal to x;
2. dividing the number thus obtained by the total number of observations, so as to obtain the proportion of observations that is less than or equal to x.

Suppose we observe a sample made of four observations: $\boldsymbol{x} = \{x_1, x_2, x_3, x_4\}$ where $x_1 = 3$, $x_2 = 2$, $x_3 = 5$, $x_4 = 2$. We have:

$$\begin{aligned}
\widehat{F}(4) &= \frac{1}{4}\sum_{i=1}^{4} \mathbb{1}_{x_i \leq 4}(4) \\
&= \frac{1}{4}\left(\sum_{i=1}^{4} \mathbb{1}_{x_1 \leq 4}(4) + \sum_{i=1}^{4} \mathbb{1}_{x_2 \leq 4}(4) + \sum_{i=1}^{4} \mathbb{1}_{x_3 \leq 4}(4) + \sum_{i=1}^{4} \mathbb{1}_{x_4 \leq 4}(4)\right) \\
&= \frac{1}{4}\left(1 + 1 + 0 + 1\right) \\
&= \frac{3}{4}.
\end{aligned} \tag{D.3}$$

D.3.2 Interpretation

The empirical distribution can be interpreted as the distribution function of a discrete variable.

Let $x_{(1)}, \ldots, x_{(n)}$ be the sample observations ordered from the smallest to the largest. The empirical distribution function can be written as

$$\widehat{F}(x) = \begin{cases} 0 & \text{if } \; x < x_{(1)} \; , \\ \frac{1}{n} & \text{if } \; x_{(1)} \leq x < x_{(2)} \\ \vdots & \\ \frac{n-1}{n} & \text{if } \; x_{(n-1)} \leq x < x_{(n)} \\ 1 & \text{if } \; x_{(n)} \leq x. \end{cases} \tag{D.4}$$

This is a function that is everywhere flat except at sample points, where it jumps by $\frac{1}{n}$. It is the distribution function of a discrete random variable X that can take any one of the values $x_{(1)}$,...,$x_{(n)}$ with probability $1/n$. In other words, it is the distribution

function of a discrete variable X having probability mass function given by

$$p_X(x) = \begin{cases} \frac{1}{n} & \text{if } x = x_{(1)} \\ \frac{1}{n} & \text{if } x = x_{(2)} \\ \vdots & \\ \frac{1}{n} & \text{if } x = x_{(n)} \\ 0 & \text{otherwise.} \end{cases} \tag{D.5}$$

D.3.3 Properties

D.3.3.1 Finite sample properties

When the n observations from the sample $x_1, \ldots, x_n$ are the realizations of n random variables $X_1, \ldots, X_n$, then the value $\widehat{F}(x)$ taken by the empirical distribution at a given point x can also be regarded as a random variable. Under the hypothesis that all the random variables $X_1, \ldots, X_n$ have the same distribution, the expected value and the variance of $\widehat{F}(x)$ are given by:

$$\begin{aligned} \mathbb{E}[\widehat{F}(x)] &= F(x), \\ \text{Var}\left(\widehat{F}(x)\right) &= \frac{1}{n} F(x)\left(1 - F(x)\right). \end{aligned} \tag{D.6}$$

The empirical distribution function is then an unbiased estimator of the true distribution function. Furthermore, its variance tends to zero as the sample size (n) becomes large.

D.3.3.2 Large sample properties

An immediate consequence of the previous results is that the empirical distribution converges in mean-square to the true one, *i.e.*

$$\widehat{F}(x) \xrightarrow{L^2} F(x) \quad \forall x. \tag{D.7}$$

It is also possible to prove a much stronger result, called Glivenko-Cantelli theorem which states that not only $\widehat{F}(x)$ converges almost surely (a.s.) to $F(x)$, but it also converges uniformly, *i.e.*

$$\sup_x |\widehat{F}(x) - F(x)| \xrightarrow{\text{a.s.}} 0. \tag{D.8}$$

Appendix

Bayesian inference of Gaussian distributed data

Contents

The Bayes' theorem[1] gives the conditional distribution of the dynamic state of interest $\boldsymbol{q}$ given the observed data $\boldsymbol{y}$,

$$p(\boldsymbol{q} \,|\, \boldsymbol{y}) = \frac{p(\boldsymbol{y} \,|\, \boldsymbol{q}) p(\boldsymbol{q})}{p(\boldsymbol{y})}. \tag{E.1}$$

Here $\boldsymbol{q}$ and $\boldsymbol{y}$ are fixed and observed quantities. We also introduce the stochastic variables $\boldsymbol{q}^t$ for the true unknown quantity and $\boldsymbol{y}^o$ for the imperfect observations. It is for these random variables that the conditional probability distributions is evaluated for the given known quantities. In the subsequent sections, we follow the presentation of Wikle and Berliner (2007) and derive the statistics of the posterior distribution $p(\boldsymbol{q} \,|\, \boldsymbol{y})$ for the univariate and multivariate data following Gaussian (or normal) distributions.

E.1 Univariate state variable

We consider a Gaussian distributed univariate state variable $q^t \sim \mathcal{N}(\mu, \tau^2)$ with mean μ and standard deviation τ. Let $\boldsymbol{y}^o = [y_1^o, \ldots, y_n^o]^\top$ be the n independent but noisy observations conditioned on the true value of the state q^t such that the $y_i^o | q^t \sim \mathcal{N}(q^t, \sigma^2)$.

[1] Let X and Y be two random variables defined on a probability space. By definition of the condition probability, we have $p(x \,|\, y) = \dfrac{p(x,y)}{p(y)}$ where $p(x,y)$ is the joint probability of having $X = x$ and $Y = y$. Similarly, we can write that $p(y \,|\, x) = \dfrac{p(y,x)}{p(x)}$. Hence, we have $p(x,y) = p(x \,|\, y)p(y) = p(y \,|\, x)p(x)$, from where we deduce the Bayes rule $p(x \,|\, y) = \dfrac{p(y \,|\, x)p(x)}{p(y)}$.

Then, the likelihood distribution for the Gaussian distributed data is given as

$$p(\boldsymbol{y}^o \,|\, q^t) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2}\left(\frac{y_i^o - q^t}{\sigma}\right)^2\right] \propto \exp\left[-\frac{1}{2}\sum_{i=1}^{n}\left(\frac{y_i^o - q^t}{\sigma}\right)^2\right]. \quad \text{(E.2)}$$

The Bayes' rule gives the posterior distribution as

$$\begin{aligned} p(q^t \,|\, \boldsymbol{y}^o) &\propto \exp\left[-\frac{1}{2}\sum_{i=1}^{n}\left(\frac{y_i^o - q^t}{\sigma}\right)^2 - \frac{1}{2}\left(\frac{q^t - \mu}{\tau}\right)^2\right] \\ &\propto \exp\left[-\frac{1}{2}\left(\frac{n}{\sigma^2} + \frac{1}{\tau^2}\right)(q^t)^2 + \left(\sum_{i=1}^{n}\frac{y_i^o}{\sigma^2} + \frac{\mu}{\tau^2}\right) q^t\right]. \end{aligned} \quad \text{(E.3)}$$

This is a product of two Gaussian distributions. It can be shown (by completing the square) that the result is also Gaussian which is given as

$$q^t|\boldsymbol{y}^o \sim \mathcal{N}\left(\left(\frac{n}{\sigma^2} + \frac{1}{\tau^2}\right)^{-1}\left(\sum_{i=1}^{n}\frac{y_i^o}{\sigma^2} + \frac{\mu}{\tau^2}\right), \left(\frac{n}{\sigma^2} + \frac{1}{\tau^2}\right)^{-1}\right). \quad \text{(E.4)}$$

The posterior mean is given as

$$\begin{aligned} \mathbb{E}[q^t|\boldsymbol{y}^o] &= \frac{\sigma^2\tau^2}{\sigma^2 + n\tau^2}\left(n\frac{\overline{y}^o}{\sigma^2} + \frac{\mu}{\tau^2}\right) \\ &= w_y\overline{y}^o + w_\mu\mu, \end{aligned} \quad \text{(E.5)}$$

where $\overline{y}^o = \sum_{i=1}^{n} y_i^o/n$ is the mean of the observation sample, $w_y = (n\tau^2)/(\sigma^2 + n\tau^2)$ and $w_\mu = \sigma^2/(\sigma^2 + n\tau^2)$. Note that $w_y + w_\mu = 1$. The posterior mean is then a weighted average of the prior mean (μ) and the natural, data based estimate of q^t, $\overline{y}^o$. We can show that the data model is generally the major controller of the posterior.

In data assimilation, we write (E.5) as

$$\begin{aligned} \mathbb{E}[q^t|\boldsymbol{y}^o] &= \mu + \frac{n\tau^2}{\sigma^2 + n\tau^2}(\overline{y}^o - \mu) \\ &= \mu + K(\overline{y}^o - \mu), \end{aligned} \quad \text{(E.6)}$$

where $K = (n\tau^2)/(\sigma^2 + n\tau^2)$ is the *gain* which adjusts the prior mean μ towards the estimate $\overline{y}^o$. The posterior variance is given as

$$\mathrm{var}(q^t|\boldsymbol{y}^o) = (1 - K)\tau^2, \quad \text{(E.7)}$$

which uses the gain K to update the prior variance τ^2.

E.2 Multivariate state variable

We now assume the state variable to be a m-dimensional vector with a prior distribution $\boldsymbol{q}^t \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{P})$, where the mean $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{P}$ are known. Also, we

have a n-dimensional observation vector conditioned on the state variables such that $\boldsymbol{y}^o|\boldsymbol{q}^t \sim \mathcal{N}(\boldsymbol{H}\boldsymbol{q}, \boldsymbol{R})$, with the known observation matrix $\boldsymbol{H} \in \mathbb{R}^{n\times m}$ and covariance matrix $\boldsymbol{R}$. Similar to the univariate case, the posterior is Gaussian which follows the mean and variance expression as in (E.4),

$$\boldsymbol{q}^t|\boldsymbol{y}^o \sim \mathcal{N}((\boldsymbol{H}^\top\boldsymbol{R}^{-1}\boldsymbol{H} + \boldsymbol{P}^{-1})^{-1}(\boldsymbol{H}^\top\boldsymbol{R}^{-1}\boldsymbol{y}^o + \boldsymbol{P}^{-1}\boldsymbol{\mu})^{-1}, (\boldsymbol{H}^\top\boldsymbol{R}^{-1}\boldsymbol{H} + \boldsymbol{P}^{-1})^{-1}). \tag{E.8}$$

The posterior mean may be rewritten as

$$\mathbb{E}[\boldsymbol{q}^t|\boldsymbol{y}^o] = \boldsymbol{\mu} + \boldsymbol{K}(\boldsymbol{y} - \boldsymbol{H}\boldsymbol{\mu}), \tag{E.9}$$

where $\boldsymbol{K} = \boldsymbol{P}\boldsymbol{H}^\top(\boldsymbol{R} + \boldsymbol{H}\boldsymbol{P}\boldsymbol{H}^\top)^{-1}$. The posterior covariance matrix is given as

$$\mathrm{var}(\boldsymbol{q}^t|\boldsymbol{y}^o) = (\boldsymbol{I}_m - \boldsymbol{K}\boldsymbol{H})\boldsymbol{P}. \tag{E.10}$$

The DA based on linear and Gaussian assumptions relies on (E.9) and (E.10) to correct the prior (forecast) mean $\boldsymbol{\mu}$ according to the gain $\boldsymbol{K}$, which is a function of the forecast and observation error matrices (*i.e.* $\boldsymbol{P}$ and $\boldsymbol{R}$).

Appendix F

Résumé étendu

La majorité des écoulements observés en pratique présentent une dynamique complexe. Cette complexité se caractérisent par une variété de mécanismes d'échelles spatiales et temporelles très variées provenant de divers instabilités et non-linéarités inhérentes à la turbulence. La description et la prédiction de leur évolution spatio-temporelle nécessite, par voie de conséquences, des discrétisations spatiales et temporelles très fines se traduisant par des coûts de calcul souvent trop importants pour être exploités en pratique. La présence d'une certaine organisation sous-jacente dans ces dynamiques complexes a, néanmoins, motivé la représentation des écoulements turbulents dans un espace dit latent de plus petite dimension nécessitant, par là même, des modèles à coût de calculs réduit. Le défi a relevé est alors celui de découvrir, dans ce nouvel espace, un modèle exploitable qui régisse la dynamique de l'écoulement en s'appuyant, éventuellement, sur les équations de conservation mais également et avant tout sur des données issues de mesures expérimentales et/ou de simulations numériques. Pour tenter d'y répondre, ce travail de thèse présente des outils méthodologiques basés sur les développements récents en matière d'assimilation de données et d'apprentissage automatique. La Fig. F.1 illustre la démarche et les différentes étapes décrites dans les différents chapitres de cette thèse. Le premier bloc méthodologique consiste à réduire la dimension du problème physique étudié. Dans ce travail, l'espace latent est construit par la technique de décomposition modale POD largement exploitée en mécanique des fluides. Dans un second bloc, deux approches sont considérées pour identifier le modèle permettant de décrire la dynamique de l'écoulement dans cet espace à plus faible dimension: (i) une approche intrusive dite boite-grise où la structure du modèle est imposée et (ii) une approche non-intrusive dite boite noire basée sur des réseaux de neurones profonds. Enfin, une fois le modèle identifié, une prédiction au temps long est obtenue en corrigeant l'état estimé à l'aide d'observations grâce à une approche par assimilation de données séquentielle effectuée dans l'espace latent. Le retour dans l'espace physique est réalisé en projetant l'état estimé sur la base modale identifiée initialement.

Modèles réduits basés sur la POD

Les systèmes fluides ont la plupart du temps une dynamique relativement complexe mais dans laquelle il est possible d'identifier une structuration organisée. Cela se traduit, entre autres, par la cohabitation de structures dites "cohérentes" à grandes échelles qui

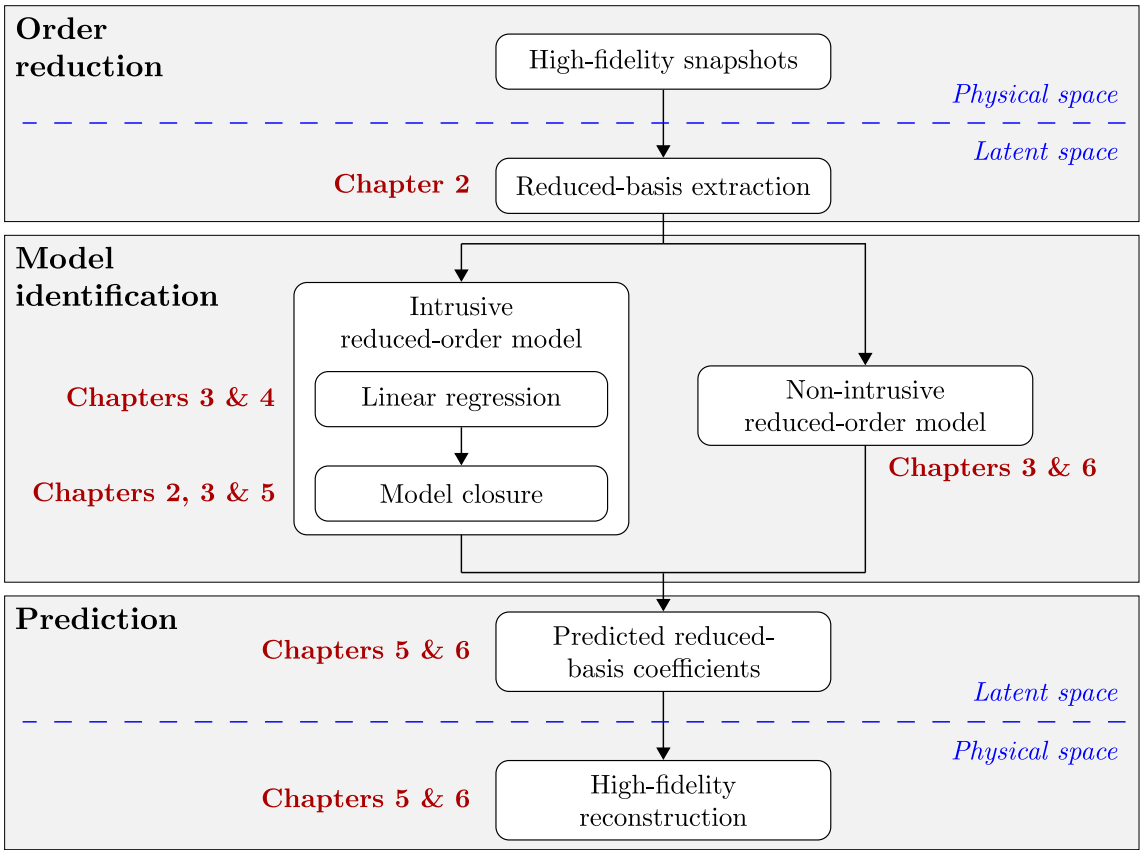


Figure F.1: Diagramme illustrant les différentes étapes méthodologiques du présent travail. Les chapitres associés à chaque étape sont mentionnés.

pilotent la dynamique globale de l'écoulement et de structures à petites échelles participant à la dissipation de l'énergie. C'est pourquoi une modélisation d'ordre-bas basée sur les lois fondamentales de la physique constitue une démarche de modélisation encore aujourd'hui privilégiée. La dynamique de l'écoulement est alors décrite, non plus dans l'espace physique réel, mais dans un espace à plus petite dimension, l'espace latent. Cette représentation peut être obtenue en choisissant une base de projection dans laquelle la dynamique sera décrite à partir d'un nombre limité de fonctions (ou modes). Ce chapitre présente l'approche dite par projection de Galerkin: les équations de la mécanique des fluides, représentées par les équations de Navier-Stokes et formant un système d'équations différentielles partielles, sont projetées sur une base de fonctions de faible dimension permettant d'aboutir à un système d'équations différentielles ordinaires plus exploitable. Ce chapitre constitue donc la première brique du travail présenté consistant à réduire la dimension du problème physique considéré. Cette réduction est obtenue en projetant le système physique réel (ou haute-fidelité) dans un espace de plus petite dimension (latent). La base de projection a pour objectif de représenter de manière précise la dynamique du système par un nombre minimal de fonctions (ou modes). Le choix, non unique, de ces fonctions est lié à ce que l'on considère comme essentiel à modéliser dans le système, ce que l'on peut donc assimiler à un problème d'optimisation. On note $\mathbf{U}(\chi, t) \in \mathbb{R}^{n_s \times n_t}$ une matrice contenant un ensemble de réalisations (snapshots) du système physique aux instants $\{t_k = (k-1)\Delta_t\}_{k=1}^{n_t}$ et aux points $\chi \in \mathbb{R}^{n_s}$. Dans le

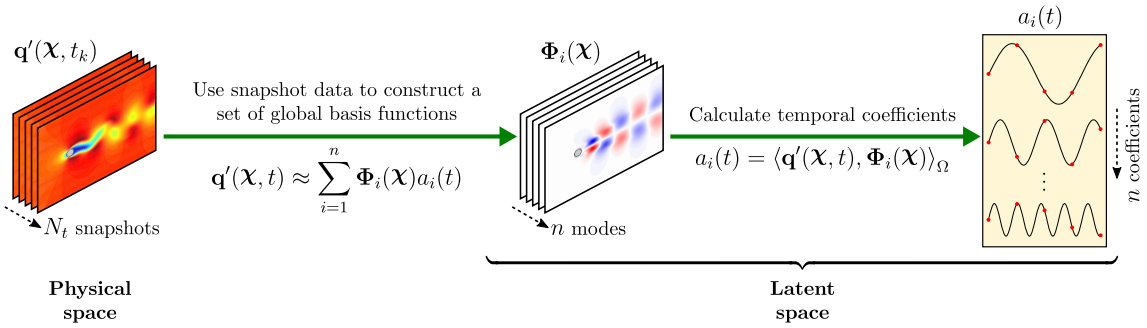


Figure F.2: Projection dans l'espace latent POD.

cas présent, on ne s'intéresse ici, au moins dans un premier temps, qu'aux méthodes de décomposition de type linéaire. L'objectif est donc de décomposer la matrice $\mathbf{U}$ en une somme de contributions de rang 1, qu'on appellera *mode propre* et ayant chacun une structure spatial $\Phi_i(\chi)$, une structure temporelle $a_i(t)$ et une amplitude λ_i. Ce que l'on pourra encore écrire,

$$\mathbf{U}(\chi, t) = \sum_{i=1}^{rk(\mathbf{U})} \lambda_i \Phi_i(\chi) a_r(t) \tag{F.1}$$

Par convention, on notera que les fonctions spatiales et temporelles sont à norme (énergie) unité. L'énergie contenue dans un mode donné est définie par le choix du produit interne dans le temps et dans l'espace. La norme L_2 est ici adoptée. Deux familles de bases pour le choix de $\Phi_i(\chi)$ peuvent être considérées: (i) des bases de fonctions données *a priori* et (ii) des bases de fonctions obtenues *a posteriori* et guidées purement par les données. Parmi la première famille on peut citer, à titre d'exemple, la décomposition en série de Fourier, en ondelettes ou encore en polynômes de Tchebychev. Dans le cas présent, l'hypothèse est faite qu'une base de projection construite *a posteriori* purement à partir des données (expérimentales ou numériques) permet d'obtenir une représentation avec un nombre de modes plus faible. Dans le présent travail, la décomposition aux valeurs propres (ou Proper Orthogonal Decomposition, POD) est considérée comme base de projection. Cette méthode est actuellement la plus répandue dans la littérature pour développer des modèles réduits en mécanique des fluides. Cela s'explique principalement par le fait que, par construction, cette méthode extrait, de réalisations de l'écoulement, une base orthonormale qui maximise la représentation en son contenu énergétique. La section §2.1.1 rappelle les éléments fondamentaux de la POD ainsi que sa variante la "snapshot POD" plus appropriée lorsque la base de données décrivant le problème haute-fidélité est composée de snapshots temporelles en nombre réduit ($n_t << n_s$) comme se sera le cas au cours de ce travail. La Fig. F.2 illustre la projection du système physique dans l'espace latent POD. Une seconde méthode connue sous le nom de Dynamic Mode Decomposition (DMD) est également présentée à la section §2.1.2 en complément. Si la POD est optimale au sens de la représentation énergétique du système, la DMD détermine les éléments des modes propres permettant de passer d'une réalisation à l'autre et est donc intrinsèquement liée à la dynamique du système.

La projection des équations de Navier-Stokes incompressibles sur la base POD conduit à un système d'équations différentielles ordinaires pour les coefficients temporels $a_i(t)$. Ainsi que détaillé en §2.2.2, chacun de ces coefficients peut être décrit par la

forme polynomiale suivante,

$$\dot{a}_i(t) = C_i + \sum_{j=1}^{rk(\mathbf{U})} L_{ij} a_j(t) + \sum_{j=1}^{rk(\mathbf{U})} \sum_{k=j}^{rk(\mathbf{U})} Q_{ijk} a_j(t) a_k(t), \quad \forall\, i = 1, \ldots, rk(\mathbf{U}), \quad \text{(F.2)}$$

avec $\theta := [C_i \ldots L_{ij} \ldots Q_{ijk}]$ un vecteur de paramètres. Un système de plus petite dimension peut alors être obtenu et que l'on écrira sous la forme compacte suivante,

$$\dot{a}_i(t) = f_i(a_i(t); \theta) + \mathcal{R}_i(N_{\text{Gal}}; \mu) \quad \forall\, i = 1, \ldots, N_{\text{Gal}} \quad \text{(F.3)}$$

Le premier membre du terme de droite faire apparaître l'approximation d'ordre bas de $\dot{a}_i(t)$ avec $f_i(a_i(t); \theta)$ obtenu par troncature à l'ordre $N_{\text{Gal}} < rk(\mathbf{U})$. Le terme $\mathcal{R}_i(N_{\text{Gal}}, \mu)$ désigne quant à lui un résidu faisant suite à cette troncature. Cette approximation est appelée modèle dynamique d'ordre-bas (ROM). Le terme de résidu peut être vu comme un modèle de fermeture permettant de modéliser l'effet des modes élevés associés typiquement aux petites échelles de l'écoulement et responsables de la dissipation d'énergie turbulente. La section §2.2.4 discute différents modèles de fermeture exprimés sous la forme d'une fonction éventuellement non linéaire au regard d'un paramétre μ.

Dans le cas idéal, le vecteur de paramètres θ est obtenu implicitement depuis la projection des équations fondamentales sur la base POD. En pratique, les données disponibles sont, de manière générale, incomplètes et/ou bruitées si bien que ce vecteur paramètre est inconnu et doit être identifié. L'identification de ce vecteur paramètre et plus globalement l'identification d'un système d'équation décrivant la dynamique du système à partir de données constitue l'objet du Chap. 3.

Méthodes d'identification purement guidées par les données (Chap. 3)

La connaissance des équations fondamentales et des conditions aux limites n'est, en général, pas systématique. En pratique, l'écoulement considéré peut être décrit par des données chronologiques obtenues par des mesures issues d'expériences ou de simulations et ce pour différentes grandeurs. La question qui vient immédiatement est celle de savoir si ces données, typiquement spatio-temporelles, peuvent être utilisées pour extraire la dynamique du système et découvrir les lois physiques qui la régissent. Dans le cas où la dynamique du système est de très faible dimension, il est possible de dériver des lois physiques à partir des équations fondamentales et des données. On parle alors de modélisation *boîte blanche* ("white-box modeling"). En revanche, pour les écoulements observés en pratique, leur dynamique est généralement décrite par un système à très grande dimension et ce type de modélisation demeure un défi. Pour palier à cette difficulté, nous avons vu au Chap. 2.2 que projeter le système d'étude dans un espace latent dans lequel la dynamique est décrite par un problème à plus petite dimension est une étape centrale. La problématique qui s'en suit est alors d'identifier cette dynamique. Les méthodes d'identification guidées par les données peuvent typiquement être découpées en deux grandes familles: (i) les méthodes dites *boites grises* pour lesquelles la dynamique est décrite par un système d'équations avec une structure fixée *a priori* et (ii) les méth-

odes dites *boites noires* qui cherche à reproduire de manière exacte la dynamique des données au prix de l'interprétabilité du système d'équations identifié. Dans cette seconde famille, nous citerons à titre d'exemple les *réseaux de neurones* sur lesquels nous reviendrons plus loin. Ce chapitre est donc dédié à la présentation des outils méthodologiques implémentés dans ce travail pour la modélisation, dans l'espace latent, de la dynamique du système considéré.

Comme point de départ, nous avons considéré au Chap. 2.2 une structure du système dynamique imposée par la projection des équations fondamentales sur une base POD. Le modèle d'évolution se présente donc sous une forme quadratique. En l'absence de terme de résidus dans l'équation (2.79), on notera que le modèle est, en revanche, linéaire en θ. Différentes méthodes de regression sont ainsi présentées en §3.1 pour identifier des systèmes dynamiques linéaire au regard des paramètres. Comme discuté dans le Chap. 2.2, les modèles réduits de type POD-ROM sont connus pour dériver à court-terme. Des modèles de fermeture peuvent être utilisés pour maintenir l'état estimé dans un espace borné. Ceux introduits dans le travail présent ont une forme éventuellement non-linéaire au regard d'un paramètre μ à identifier. Les méthodes de regression linéaire ne sont donc ici plus appropriées. Pour palier à cette difficulté, des variants du filtre de Kalman sont introduits en §3.2. Ces derniers rentrent dans le cadre des méthodes d'assimilation de données séquentielles. Le filtre de Kalman d'Ensemble est considéré en détails. Cette méthode estime par le biais d'une approche de Monte Carlo les corrections à appliquer à l'état prédit du système et aux paramètres du modèle (ici le vecteur étendu $[\theta; \mu]$) selon les observations disponibles. Une équation d'observation reliant, à un instant donné, les observations disponibles au vecteur d'état estimé est combinée à l'équation d'évolution du système et un problème d'optimisation sous contrainte est résolu.

Enfin, pour s'affranchir d'une structure *a priori* du modèle réduit, une approche de type boite-noire basée sur des réseaux de neurones profonds est présentée en §3.3. Cette méthode, non-intrusive, permet d'apprendre la dynamique du système de manière supervisée en recherchant, uniquement à partir des données, un modèle de régression capable de prédire l'évolution temporelle des coefficients temporels POD.

Méthodes de regression linéaire

La modélisation par l'approche de type POD-Galerkin discutée au Chap. 2.2 rentre, dans le cadre des méthodes dites *boites grises*. La structure du modèle réduit est dans ce cas contrainte par la projection des équations de Navier-Stokes sur la base POD et est pleinement connue une fois le vecteur de paramètres θ identifié. La détermination de θ s'apparente à un problème de régression. Typiquement, le modèle réduit POD-Garlerkin tel que présenté en §2.2.2 décrit certes une dynamique non-linéaire des modes propres du système, mais est linéaire au regard du vecteur de paramètres θ. Aussi, le problème de l'identification de ce paramètre revient à un problème de regression linéaire du type,

$$\boldsymbol{y_i} = \boldsymbol{X}\boldsymbol{\beta}_i + \epsilon_i \quad i = 1, \dots, N_{\text{Gal}} \tag{F.4}$$

En supposant que les dérivées temporelles $\dot{a_i}(t)$ des fonctions propres $a_i(t)$ issues de la POD appliquée aux données puissent être obtenues, $\mathbf{y_i}$ est défini par le vecteur,

$$\boldsymbol{y}_i \coloneqq \dot{\boldsymbol{a}}_i = [\dot{a}_i(t_1)\, \dot{a}_i(t_2)\, \dots\, \dot{a}_i(t_{N_t})]^\top \in \mathbb{R}^{N_t \times 1}. \tag{F.5}$$

La matrice $\boldsymbol{X}$ est appelée *design matrix* et est définie pour $j, k = 1, \ldots, N_{\text{Gal}}$ par,

$$\boldsymbol{X} \coloneqq \begin{bmatrix} 1 & \cdots & a_j(t_1) & \cdots & a_j(t_1)a_k(t_1) & \cdots \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots \\ 1 & \cdots & a_j(t_{N_t}) & \cdots & a_j(t_{N_t})a_k(t_{N_t}) & \cdots \end{bmatrix} \in \mathbb{R}^{N_t \times N_{\theta i}}, \tag{F.6}$$

Le vecteur de paramètres inconnus est donné par,

$$\boldsymbol{\beta}_i \coloneqq \boldsymbol{\theta}_i = [C_i \cdots L_{ij} \cdots Q_{ijk} \cdots]^\top \in \mathbb{R}^{N_{\theta i} \times 1} \quad \forall\, j, k = 1, \ldots, N_{\text{Gal}}. \tag{F.7}$$

et ϵ_i désigne un terme d'erreur défini par,

$$\boldsymbol{\epsilon}_i \coloneqq [\epsilon_i(t_1)\, \epsilon_i(t_2)\, \ldots\, \epsilon_i(t_{N_t})]^\top \in \mathbb{R}^{N_t \times 1}. \tag{F.8}$$

Dans le language commun, $\boldsymbol{y}_i$ est un vecteur de valeurs observées (ou variables dépendantes). Les colonnes de $\boldsymbol{X}$ sont appelés variables explicatives (ou variables indépendantes).

La première partie du chapitre présente trois méthodes de régression linéaire évaluées dans ce travail. En §3.1.1, la procédure d'estimation par moindres carrées (OLS) est présentée en détails. Un estimateur du vecteur β peut ainsi être obtenu comme solution du problème de minimisation en norme ℓ_2 du terme d'erreur soit encore[1]

$$\hat{\beta} = \arg\min_{\boldsymbol{\beta}} \left(\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 \right) \tag{F.9}$$

On montre que cette minimisation est équivalente à la résolution d'un problème linéaire bien souvent mal conditionné en raison de la taille de $\boldsymbol{X}$. Cette difficulté est typiquement soulevée grâce à une troncature de la décomposition aux valeurs propres $\boldsymbol{X}$.

Lorsque la taille du vecteur de paramètres est relativement important (ce qui est typiquement le cas pour un modèle réduit POD même avec un nombre de mode faible), il peut être interessant de chercher une solution dite *creuse*. En §3.1.2, l'algorithme SINDy ("sparse identification of nonlinear dynamics"), introduit par Brunton, Proctor, et al. (2016a), est ainsi présenté. Ce dernier part de l'hypothèse que un système, même complexe, présente en général un nombre limité seulement de termes non-linéaires actifs dans sa dynamique. Dans l'espace de grande dimension formé par les fonctions de bases et donné par $\boldsymbol{X}$, cela signifie qu'une représentation creuse de $\boldsymbol{y}$ peut être obtenue. La pénalisation du nombre de terme non nul dans le vecteur de paramètres inconnus peut ainsi être obtenu au moyen d'une régularisation de type ℓ_1 convexe, soit encore,

$$\hat{\beta} = \arg\min_{\boldsymbol{\beta}} \left(\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_1 \right) \tag{F.10}$$

avec λ un paramètre de parcimonie choisi de manière heuristique. La résolution de cette minimisation peut être obtenue au moyen de l'algorithme LASSO. Dans Brunton, Proctor, et al. (2016a), la résolution est faite de manière séquentielle . A chaque itération, une approximation aux moindres carrées de $\boldsymbol{\beta}$ est "plafonnée" à un seuil $\lambda > 0$. L'algorithme SINDy converge en, au plus, N_θ itérations et on montre que l'estimateur

[1] par mesure de simplification, l'indices i est désormais omis

du vecteur inconnu est donné par le minimum local de,

$$\min_{\boldsymbol{\beta}} \left(\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda^2 \|\boldsymbol{\beta}\|_0 \right) \tag{F.11}$$

Bien que cet algorithme ait démontré un engouement certain comme méthode de régression linéaire, celui-ci présente deux limitations majeures. Dans un premier temps, la solution n'est pas unique quand la dimension du vecteur inconnu excède le nombre d'observations. Par ailleurs, l'algorithme est conçu pour une formulation aux moindres carrées et est peu approprié si un terme de contrainte est ajouté ou encore pour l'estimation non-linéaire. Afin de palier à ces limitations, l'algorithme LARS (Least Angle Regression) de Efron, Hastie, et al. (2004), particulièrement efficient dés lors que $N_t \ll N_\theta$, est ainsi discuté en §3.1.3.

Assimilation de données

Les modèles réduits de type POD-ROM sont connus pour dérivés au-delà d'un temps court. Une première manière de faire pour maintenir le système prédit borné est d'utiliser une méthode de régression avec régularisation. Une première difficulté apparait dés lors que le modèle considéré n'est plus linéaire au regard du vecteur de paramètres inconnu, ce qui est typiquement le cas lorsqu'un modèle de fermeture non-linéaire est utilisé pour le terme de résidu $\mathcal{R}_i(N_{\text{Gal}}, \mu)$ introduit précédemment. Par ailleurs, malgré l'effort apporté pour identifier le système dynamique, l'expérience montre que leur évolution est difficilement prédictible sur le temps long en raison du caractère chaotique des écoulements turbulents considérés. Pour palier à ces deux difficultés, une extension du filtre de Kalman, connue sous le nom de Kalman d'ensemble dual, comme méthode d'assimilation de données séquentielles est introduite en §3.2 pour corriger simultanément l'état prédit et certains paramètres du modèle réduit à partir d'observations du système. Ce type de filtre utilise une représentation de Monte Carlo comme approximation des vecteurs d'états et de paramètres sous forme d'ensembles. Ces deux ensembles sont définis à l'aide de distributions normales gaussiennes autour de leur valeur moyenne respective et de matrice de covariances données. A partir d'un ensemble d'états d'ébauches à l'instant t_{i-1}, un ensemble d'états d'ébauches à l'instant t_i est construit par intégration du modèle d'évolution (modèle réduit identifié au préalable sans terme de fermeture par une méthode de régression linéaire). Une seconde phase, dite d'analyse, permet de corriger l'ensemble des états estimés par un gain de Kalman calculé à partir d'une matrice de covariance d'erreur de précision et de l'ensemble d'observations disponible à cet instant. Le processus itératif est illustré Fig. F.3. En parallèle, l'ensemble des paramètres d'ébauches est également propagé suivant une marche aléatoire puis analyser selon la même procédure. Le filtre de Kalman d'ensemble peut être vu comme une estimation suboptimale de l'état du système étant données des observations bruitées ou/et erronées. Nous montrerons par la suite sur plusieurs cas tests que cet algorithme permet, après un temps dit d'apprentissage, non seulement d'estimer le paramètre inconnu permettant de décrire le modèle de fermeture, mais également de corriger sur le temps long l'état estimé du système.

Réseaux de neurones profonds

La réduction de modèle par une projection de Galerkin des équations de Navier-

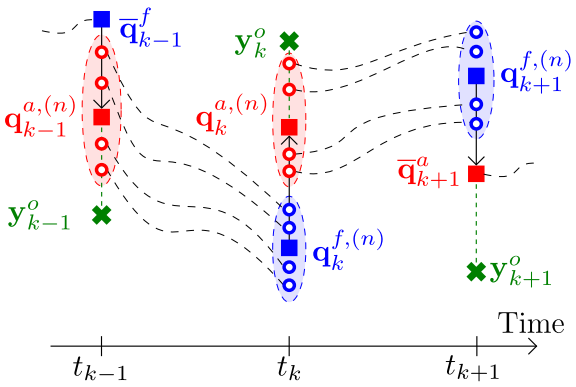


Figure F.3: Illustration du filtre de Kalman d'ensemble. Les symboles pleins représentent les valeurs moyennes des ensembles analysés $\overline{q}^a$ (rouge) et prédits $\overline{q}^f$ (bleu). Les lignes discontinues représentent les trajectoires de chaque membre de l'ensemble propagés de manière séquentielle à l'aide du modèle d'évolution. La phase d'analyse (correction) de l'état prédit à chaque itération étant donné le vecteur d'observation y^o est indiquée par une flèche.

Stokes sur une base POD a l'avantage d'être liée aux équations de conservation, facilitant l'interprétabilité du modèle. Cependant, cette méthode est intrusive au sens où elle nécessite une expertise humaine pour le développement des modèles à partir de données. De plus, les modèles obtenus à partir des méthodes modales sont, en général, peu robustes dans les applications de type contrôle. Ce manque de robustesse est principalement dû au fait que toute l'enveloppe de la dynamique de l'écoulement ne peut pas être capturée avec précision par quelques modes spatiaux dominants. C'est pourquoi, une approche non intrusive est également proposée dans le travail présent et par laquelle un modèle dynamique peut être dérivé uniquement à partir de données, sans aucune information préalable sur la physique sous-jacente. L'objectif est par ailleurs d'utiliser ce modèle pour aproximer le modèle d'ordre réduit POD pour une condition d'écoulement donnée (dépendance au nombre de Reynolds typiquement). Notons que le cadre proposé ici peut être étendu pour reproduire la dynamique sur le domaine des paramètres de contrôle, en utilisant une base de données contenant des informations pour différentes valeurs de ces paramètre.

Dans ce travail, l'étape de projection de Galerkin est contournée au moyen de réseaux de neurones profonds (DNN). Les DNN sont des réseaux de neurones artificiels (ANN) pouvant être vu comme des modèles entrées-sorties composés de plusieurs couches intermédiaires. Un ANN est une classe de méthodes d'apprentissage machine qui imite, mathématiquement, les réseaux de neurones biologiques et qui peuvent être utilisés en tant que méthode de régression non-linéaire. A l'inverse d'un modèle POD-ROM obtenue par projection de Galerkin, les ANN ne sont pas physiquement interprétables. En revanche, ils permettent, par construction, de représenter des relations non-linéaires qui ne peuvent pas être explicitées de manière formelle. Cette particularité permet en particulier de minimiser les incertitudes du modèle durant la phase de prédiction.

Deux stratégies de DNN récemment présentées dans la littérature pour le développement de modèles ROM non intrusifs sont combinées ici. La première stratégie est celle d'un DNN qui approxime les coefficients temporels POD, obtenus depuis les données, comme une fonction du temps et des valeurs de paramètres (Wang, Hesthaven, et al.,

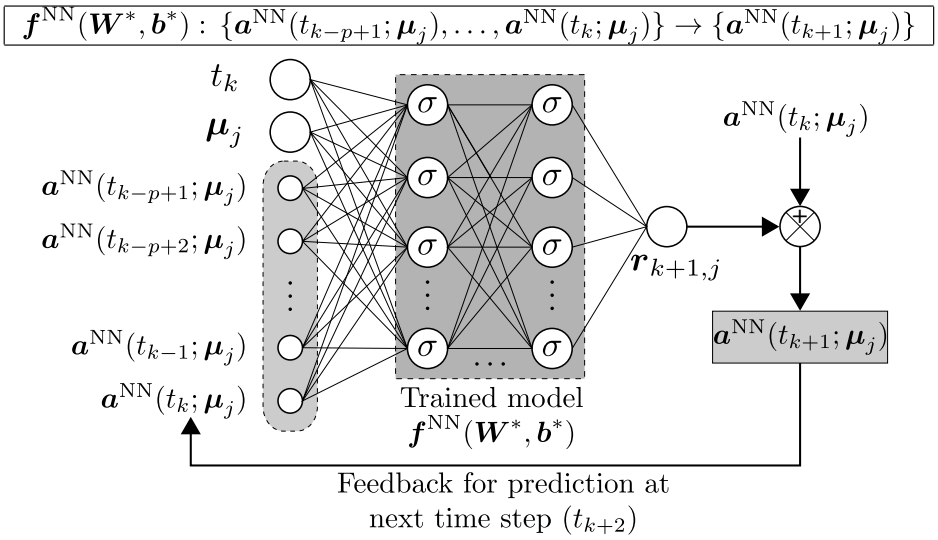


Figure F.4: Architecture de l'algorithme NN-ROM.

2019). Cette stratégie est combinée avec celle employée pour les DNN utilisés pour prédire l'évolution du résidu de la solution à partir des valeurs d'un état aux instants passés (Pawar et al., 2019). L'architecture de l'algorithme, nommé NN-ROM, est illustré Fig. F.4 et discuté en détails en §3.3.1.

Evaluation par technique de bootstrap des méthodes de régression linéaire (Chap. 4)

Afin d'évaluer les performances des méthodes de régression linéaire discutées au Chap. 3, différents cas tests sont ici considérés. La capacité de ces méthodes à identifier un modèle dynamique réduit ayant une structure figée *a priori* à partir de données issus d'expériences ou de simulations, donc incomplètes et éventuellement bruitées, est évaluée de manière quantitative dans un cadre probabiliste grâce à la technique du bootstrap. Cette technique statistique fournit une approximation d'une distribution inconnue par une distribution empirique obtenue par un processus de ré-échantillonnage. Son application aux modèles de régression donne donc la distribution des erreurs de prédiction, soit encore des intervalles de confiance sur des prévisions. Cette technique ainsi que sa variante dite "par blocs circulaires" est décrite en détails en §4.2. Par ailleurs, afin de palier au bruit qui corrompt éventuellement les données disponibles, une technique de différentiation pour l'estimation de la dérivée temporelle des coefficients POD et une technique de filtrage passe-bas sont introduites (§4.1). Ces deux outils sont employés systématiquement dans la suite du travail dés lors que les données considérées sont issues de mesures expérimentales.

Les trois cas tests considérés ici sont les suivants: (i) un système linéaire de dimension 3, (ii) le système chaotic de Lorenz-63 de dimension 3 et enfin (iii) le modèle de Lorenz-96. On fait l'approximation que ces trois cas donnent une bonne représentation de dynamiques de complexités croissantes (au moins en termes de dimension) des écoulements auxquels on s'intéresse dans ce travail. Pour chacun des trois cas, le sys-

tème d'équations décrivant la dynamique du système est connue et est donc utilisé pour générer dans un premier temps une série de données temporelles discrètes auxquelles sont ajoutées du bruit gaussien à moyenne nulle. L'effet du niveau de bruit est également considéré en faisant varier le niveau de variance. Chacune des trois méthodes de régression (OLS, SINDy et LARS) est ensuite appliquée de manière systématique pour retrouver les paramètres du modèle. Notons par ailleurs que chaque jeu de données est découpé en un jeu pour l'apprentissage utilisé pour identifier les paramètres du modèle, et un jeu pour la validation de la prédiction.

En ce qui concerne l'identification des paramètres du modèle, la méthode par moindres carrés (OLS) donne des résultats satisfaisants mais pour un coût de calcul, comparativement aux deux autres méthodes, plus grand. Les méthodes SINDy et LARS montrent des performances comparables en termes d'erreur et ont l'avantage de donner des solutions parcimonieuses. En revanche, la méthode LARS montre une sensibilité plus faible vis à vis du bruit de mesure même dans le cas d'un système à grande dimension, ce qui présente un intérêt tout particulier dans le cas de données expérimentales par exemple. Pour les trois méthodes, les trajectoires des états estimés restent bornées en phase de prédiction quelque soit le cas test considéré. En revanche, la trajectoire vraie est correctement reproduite uniquement sur un temps court (deux ou trois lâchés tourbillonnaires typiquement dans le cas de l'écoulement autour d'un cylindre). Les échelles temporelles caractéristiques de l'évolution des systèmes étudiés sont bien retrouvées mais un déphasage est observé. Ce dernier point indique clairement que l'identification seule du modèle à partir de données chronologiques n'est pas suffisante pour une prédiction sur le temps long et qu'une correction supplémentaire est nécessaire.

Amélioration des modèles réduits POD-ROM par filtre de Kalman d'ensemble dual pour la prédiction au temps long (Chap. 5)

Les efforts méthodologiques pour l'identification du modèle POD-ROM à partir de méthode de régression ne sont bien souvent pas suffisantes pour une prédiction au temps long. A minima, le modèle identifié reste borné. Cela est d'autant plus vrai et critique que le système étudié est caractérisé par une dynamique nécessitant, même dans l'espace latent, un nombre de modes en $\mathcal{O}(10)$ résultant en la détermination d'un vecteur de paramètres inconnu de grande dimension. Notons, à ce stade, que pour des applications de type contrôle, une prédiction à temps court peut être souvent suffisante. Afin d'améliorer la prédiction sur le temps long, l'intérêt du filtre de Kalman d'ensemble dual comme méthode d'assimilation de données est démontré ici pour différents cas tests: (1) un écoulement de sillage derrière un cylindre à nombre de Reynolds de 200 obtenu par simulation numérique, (2) un écoulement de sillage derrière un cylindre à nombre de Reynolds de $1,5 - 5,5 \times 10^4$ décrit par des snapshots PIV et enfin (3) un jet turbulent à Mach 0,9. Pour ces trois configurations, le modèle POD-ROM identifié par une des méthodes de régression vue dans le chapitre précédent constitue le point de départ. Un terme de résidu, éventuellement non-linéaire au regard d'un vecteur de paramètres μ inconnu, est ajouté au modèle. Le filtre de Kalman d'ensemble dual est ensuite appliqué

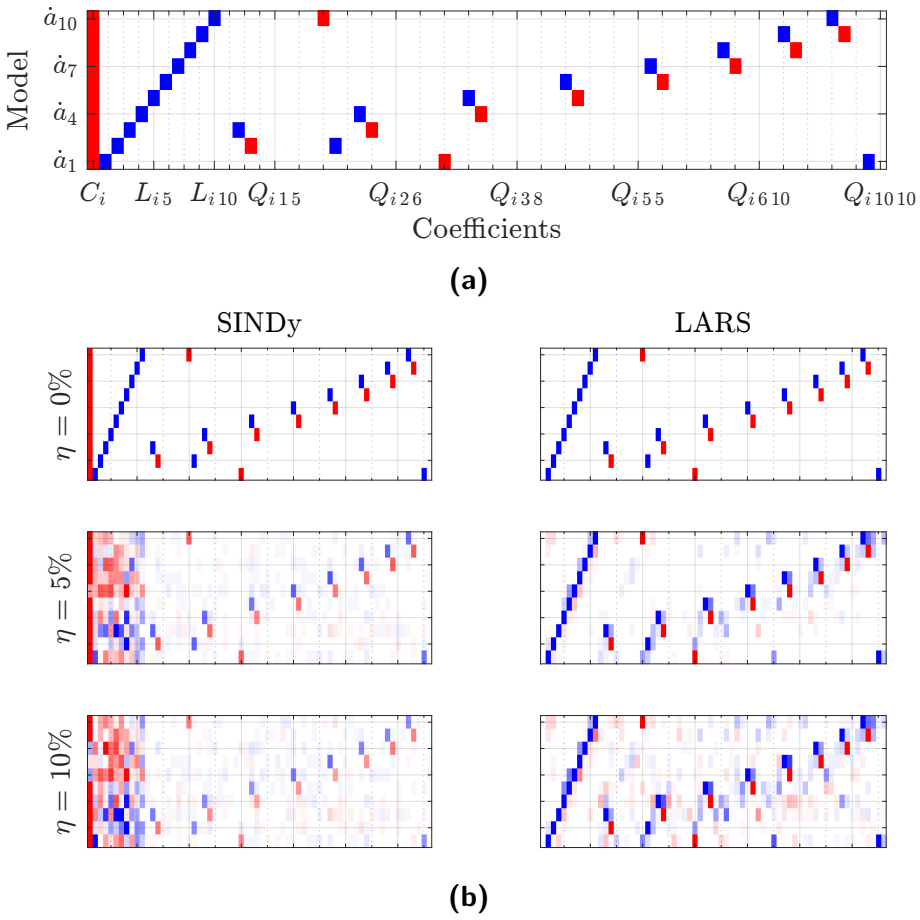


Figure F.5: Cas test: Lorenz-96 de dimension 10. Représentation des coefficients du modèle (a) vrais (b) estimés par les algorithmes (gauche) SINDy et (droite) LARS pour différents niveaux η de bruit.

pour corriger l'estimation sur le temps long des coefficients temporels POD ainsi que pour identifier, en parallèle, le vecteur μ.

Le modèle de Lorenz-63 est d'abord considéré pour mener une étude paramétrique sur l'effet du bruit de mesure et d'observation. On montre en particulier que l'erreur d'estimation dépend essentiellement du rapport de niveau entre les covariances du modèle et d'observation. Pour démontrer le potentiel dual du filtre de Kalman, on définit le vecteur de paramètres μ comme les paramètres du modèle de Lorenz standard. Lorsque initialisé avec un vecteur μ erroné, on montre que le filtre de Kalman d'ensemble permet de faire converger ce vecteur vers la valeur vraie. Le nombre d'itérations nécessaire pour la convergence dépend une nouvelle fois des matrices de covariance choisies pour le modèle et les observations.

Pour les autres cas tests, les données d'origine, formées par des snapshots de vitesse, sont classiquement découpées en un jeu pour l'apprentissage et un jeu de validation. Le premier jeu est utilisé pour identifier un premier modèle POD-ROM grâce à l'algorithme SINDy et pour identifier grâce au filtre de DualEnKF le vecteur de paramètres μ du terme résiduel. Les résultats obtenus montre de nouveau que le vecteur de paramètres converge vers une valeur finie avec une réduction de la covariance de l'ensemble au

cours de la phase d'apprentissage. L'évaluation du modèle POD-ROM+Résidu pour la prédiction sur le temps long est ensuite réalisée à l'aide du second jeu de données. Une illustration des résultats obtenus est présentées Fig. F.6 pour le cas du cylindre. L'erreur d'estimation quadratique moyenne et le contenu spectral des coefficients temporels POD sont examinés. Pour tous les cas tests, des erreurs d'estimation acceptables sont observées à condition que l'assimilation de nouvelles observations soient effectuées à pas de temps réguliers. On note en particulier que la correction du terme de résidu à travers l'estimation du vecteur de paramètre μ à la fin de la fenêtre d'apprentissage permet d'assimiler les observations à des intervalles de temps plus grands qu'en l'absence du terme de résidu. Dans le cas du cylindre par exemple (illustré Fig. F.6), l'intervalle d'assimilation passe de 2 à 3 lâchers tourbillonnaires typiquement.

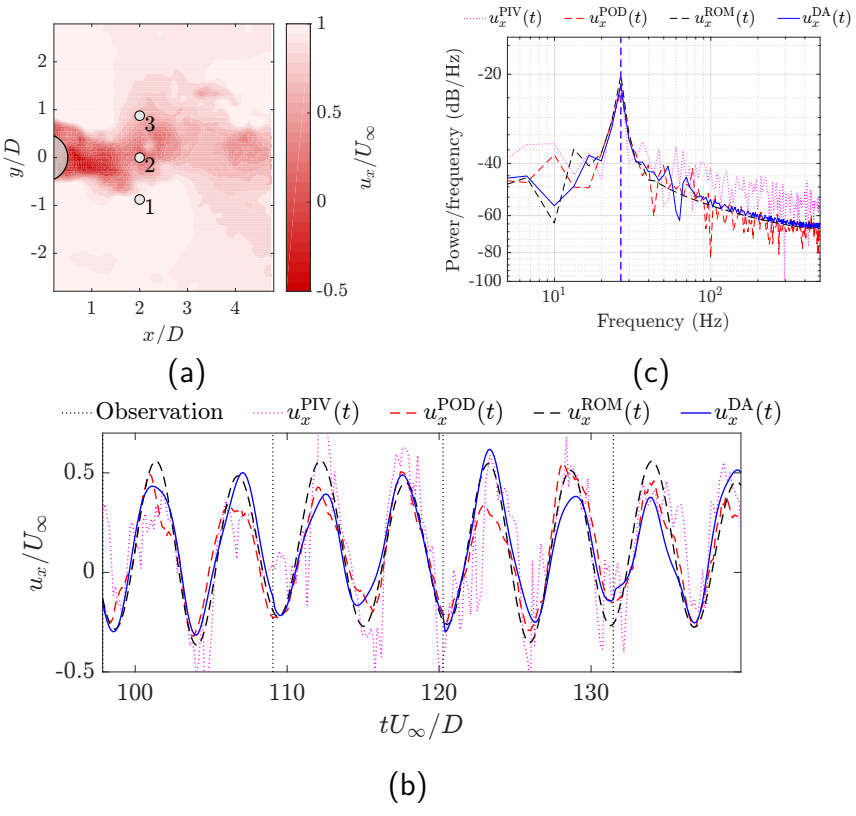


Figure F.6: (a) Snapshot PIV pour le cas d'un écoulement autour d'un cylindre à nombre de Reynolds $1,5 \times 10^4$ et localisation des points observations pour la procédure d'assimilation de données. (b) Evolution temporelle de la composante de vitesse longitudinale pour un point situé à 0,5D derrière le cylindre obtenue par mesure PIV ($u_x^{\mathrm{PIV}}(t)$) et reconstruite à partir des coefficients temporels POD vrais ($u_x^{\mathrm{POD}}(t)$), estimés par le modèle POD-ROM identifié par SINDy ($u_x^{\mathrm{ROM}}(t)$) et estimés par le modèle POD-ROM corrigé par assimilation d'observations ($u_x^{\mathrm{DA}}(t)$). (b) Densités spectrales de puissances associées.

	Problem	Training parameter set	Test parameter
Case-I	Interpolation	$\{100, 110, 120, 130, 140, 170, 190\}$	160
Case-II	Extrapolation	$\{100, 110, 120, 130, 140, 160, 170, 190\}$	210

Table F.1: Valeurs du nombre de Reynolds (Re) retenus pour deux jeux d'apprentissage pour confronter le NN-ROM à un problème d'interpolation et d'extrapolation respectivement. Ecoulement autour d'un cylindre.

Méthode ROM non-intrusive basées sur les réseaux de neurones profonds (Chap. 6)

Jusqu'ici, l'identification du modèle réduit s'est appuyé sur une forme quadratique connue *a priori* trouvant sa justification dans la projection des équations de Navier-Stokes sur une base POD (projection de Galerkin). Cette approche nécessite une expertise humaine et peut donc être qualifiée de "intrusive". Dans cette dernière partie de la thèse, nous nous proposons d'appliquer une approche dite "non-intrusive" dans laquelle la dynamique de l'écoulement dans l'espace latent est représentée au moyen d'un réseau de neurones profond acceptant comme paramètre d'entrée le nombre de Reynolds. Par ailleurs, afin de prendre en compte un effet mémoire de la dynamique passée, le modèle NN-ROM élaboré considère pour entrée les états estimés aux instants passés. L'objectif est ici de démontrer au travers d'un cas test comme celui d'un écoulement autour d'un cylindre, que l'algorithme discuté en Sec. 3.3.1 permet de reconstruire la dynamique pour un nombre de Reynolds donné non disponible dans la base de données initiale. En ce qui concerne les autres hyperparamètres, des règles bien établies dans la littérature permettent d'éviter, à titre d'exemple, le surapprentissage.

Le modèle de Lorenz-63 est, au préalable, considéré pour valider l'algorithme et étudier l'influence des hyperparamètres du modèle tel que le nombre de couche, le nombre de neurones par couche, le nombre d'epoch ou encore le nombre d'instants passés pris en compte. Les résultats montrent que ce dernier paramètre a une influence importante et qu'une réduction d'autant plus significative de l'erreur d'estimation peut être obtenue sur la fenêtre de prédiction que le nombre d'instants passés pris en compte est grand. La dynamique estimée reste bornée au domaine de la dynamique vraie. En comparaison des résultats obtenus pour l'approche POD-ROM discutée dans les chapitres précédent, la dynamique du système est correctement reproduite sur une durée plus importante dans le cas du NN-ROM. Au-delà de ce temps court, les échelles caractéristiques sont retrouvés mais la trajectoire vraie n'est plus reproduite correctement comme dans le cas du POD-ROM.

Une fois identifié l'influence de divers hyperparamètres du modèle, on se propose de valider l'approche NN-ROM pour le cas d'un écoulement de cylindre. Pour cela, on réalise dans un premier temps une base de données constituée de snapshots de vitesse obtenues par simulations numériques pour différentes valeurs du nombre de Reynolds Re compris entre 100 et 210. Pour chaque simulation, les 10 premiers modes POD les plus énergétiques sont calculés et conservés. Le tableau F.1 précise les deux cas retenus pour l'apprentissage du modèle NN-ROM et tester sa capacité à interpoler à $Re = 160$ et extrapoler à $Re = 210$ respectivement la dynamique de l'écoulement. Les résultats mon-

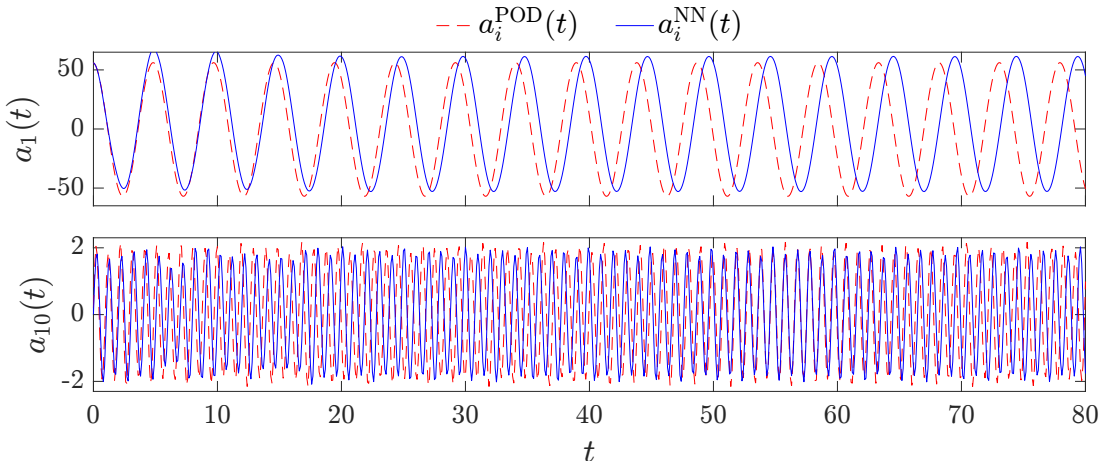


Figure F.7: Evolution of the temporal POD coefficients $a_1(t)$ and $a_{10}(t)$ for the testing dataset for the cylinder wake flow at $Re = 210$ obtained from the NN-ROM and compared with the reference trajectory. Note that data corresponding to this Re was part of the testing dataset and not used during the learning of the NN-ROM.

trent que les dynamiques pour les nombres de Reynolds inclus dans la base de données utilisées pour l'apprentissage sont recouvrées avec une erreur d'estimation négligeable sans overfiting. La dynamique à $Re = 160$ est interpolée avec une erreur quadratique moyenne acceptable. En revanche, un déphasage entre la trajectoire estimée et vraie est observée pour les modes élevés. Pour le cas $Re = 210$ (extrapolation), l'erreur quadratique moyenne reste acceptable mais l'évolution temporelle des coefficients POD (même bas) n'est bien reproduite que sur un temps court suivi d'un déphasage qui s'opère sur le temps long comme l'illustre la Fig. F.7. Une comparaison avec les résultats obtenus par identification du modèle POD-ROM par régression linéaire indique que le déphasage s'observe plus tardivement dans le cas du NN-ROM, suggérant qu'une prédiction au temps long peut être obtenue par la technique du filtre de Kalman d'ensemble (EnKF) grâce à l'assimilation d'observations en des instants plus éloignés. Ceci représente un intérêt tout particulier pour de futurs implémentations dans un objectif de contrôle ou encore de prédiction en "temps-réel" pour lesquelles les contraintes techniques peuvent nécessiter une durée entre l'assimilation de nouvelles observations du même ordre de grandeur que l'échelle caractéristique de l'écoulement étudié.

La robustesse de l'approche NN-ROM + Filtre EnKF au bruit de mesure est démontrée pour le cas de l'écoulement de cylindre à $Re = 1,4 \times 10^5$ considéré au Chap. 5 et pour lequel on dispose d'une base de données constituées de snapshots PIV. Comme précédemment, les 10 premiers modes POD sont calculés et conservés pour construire le modèle NN-ROM. Le nombre d'instants passés considérés pour estimer l'instant suivant est fixé à 10. L'apprentissage est réalisé sur 20000 epochs. Sans assimilation, les résultats montrent une bonne prédictibilité sur un temps court des coefficients temporels correspondant à environ 4 lâchers tourbillonnaires donc une nouvelle fois plus long que le modèle POD-ROM. L'assimilation d'observations à ce même intervalle de temps régulier permet de maintenir une prédiction sur le temps long et une d'erreur d'estimation quadratique faible.

Conclusion & Perspectives

Ce travail de thèse présente un cadre méthodologique pour la modélisation d'écoulements guidée par les données. La modélisation est réalisée dans un espace latent de dimension exploitable, plus faible que le problème initial décrit typiquement par les équations de Navier-Stokes. L'approche par décomposition aux valeurs propres (POD) est ici employée mais d'autres bases de décomposition peuvent être envisagée pour extraire les modes dominants de l'écoulement et ainsi construire l'espace latent. Dans cet espace réduit, la dynamique de l'écoulement peut alors être décrite par celle des coefficients temporels POD. Deux pistes sont ensuite privilégiées pour identifier un modèle décrivant cette dynamique.

La première, dite intrusive, s'appuie sur les équations de la mécanique des fluides. Un système d'équations différentielles ordinaires est dérivé par projection de Galerkin sur la base POD. Un modèle, dit POD-ROM, de forme quadratique au regard des coefficients temporels POD mais linéaire au regard de paramètres à identifier est obtenu. Trois méthodes de régression linéaire (OLS, SINDy et LASSO) sont alors discutées et évaluées dans un cadre probabiliste grâce à la méthode du bootstrap pour estimer les paramètres du modèle uniquement à partir de données. Différents cas test sont considérés tel que un modèle analytique 1D non-linéaire, le système de Lorenz ou encore le cas d'un écoulement autour d'un cylindre. Les méthodes SINDy et LASSO conduisent à des solutions parcimonieuses pour le vecteur de paramètres inconnu mais seule la seconde montre une bonne robustesse au bruit présent dans les données. Afin d'obtenir une prédiction bornée des coefficients temporels, une régularisation de la solution est introduite. Toutefois, les trajectoires vraies ne sont bien reproduites que sur un temps court de l'ordre du temps caractéristique des systèmes considérés. Afin d'améliorer la prédiction sur le temps long, un terme de résidu pour modéliser les modes élevés non pris en compte dans le modèle POD-ROM initial est d'abord ajouté. Le filtre de Kalman d'ensemble dual est alors ensuite introduit pour identifier le paramètre inconnu du terme de résidu tout en corrigeant le vecteur d'état grâce à l'assimilation d'observations éventuellement bruitées. Cette technique d'assimilation de données séquentielle est particulièrement bien adaptée aux applications expérimentales. L'approche combinée POD-ROM et filtre de Kalman d'ensemble est ainsi démontrée dans le présent travail pour un écoulement de sillage derrière un cylindre en considérant à la fois des données issues de simulations numériques mais également d'expérience.

La seconde piste, dite non-intrusive, fait appel aux réseaux de neurones profonds pour identifier, directement dans les données disponibles, la dynamique réduite. Cette approche, dénommée NN-ROM, constitute un changement complet de paradigme, au prix de l'interprétabilité du modèle. Durant l'apprentissage du réseaux de neurones, l'effet mémoire est maintenu grâce à la prise en compte des valeurs des coefficients temporels aux instants passés. Par ailleurs, le résidu entre deux pas de temps consécutifs est considéré comme valeur cible en sortie du réseau. Une nouvelle fois, différents cas tests sont considérés pour évaluer la capacité du NN-ROM à estimer sur le temps long la dynamique du système dans l'espace latent. Les résultats montrent notamment que les trajectoires vraies sont correctement prédite sur un temps court plus long que celui observé avec l'approche POD-ROM. En combinant le NN-ROM avec un filtre de Kalman d'ensemble, des améliorations significatives sont observées. Enfin, l'approche

est étendue au cadre paramétrique avec le problème d'interpolation et d'extrapolation de la dynamique d'un écoulement autour d'un cylindre à des nombres de Reynolds non disponibles dans la base de données initiale. L'extrapolation pour un régime correspondant à un nombre de Reynolds en dehors de la gamme disponible dans les données initiales constitue le cas présentant des erreurs d'estimation significatives mais qu'il est possible de corriger grâce au filtre de Kalman d'ensemble. Au final, l'approche proposée et étendue au cadre paramétrique présente une certaine robustesse aux bruits de mesures pour l'estimation sur le temps long.

Une extension naturelle des travaux en cours consiste à utiliser l'identification ROM guidée par les données à la prévision opérationnelle et aux applications telles que le contrôle en temps réel, l'optimisation multidisciplinaire ou encore la quantification des incertitudes. L'application expérimentale fournirait l'occasion d'évaluer plus largement la faisabilité de ces approches mais également d'acquérir un aperçu plus approfondi de leurs capacités à estimer des dynamiques réels.

References

Page number(s) of citation in this document is provided for information.

Abadi, M., A. Agarwal, P. Barham, E. Brevdo, Z. Chen, et al. (2016). *Tensorflow: Large-scale machine learning on heterogeneous distributed systems*. arXiv: 1603.04467 [cs.DC] (see page 68).

Ahuja, Sunil and Clarence W Rowley (2010). "Feedback control of unstable steady states of flow past a flat plate using reduced-order estimators". In: *Journal of Fluid Mechanics* 645, pp. 447–478. DOI: 10.1017/S0022112009992655 (see page 4).

Akaike, Hirotugu (1974). "A new look at the statistical model identification". In: *IEEE Transactions on Automatic Control* 19.6, pp. 716–723. DOI: 10.1109/TAC.1974.1100705 (see page 46).

Alfio Quarteroni, Gianluigi Rozza (2014). *Reduced Order Methods for Modeling and Computational Reduction*. 1st ed. MS&A - Modeling, Simulation and Applications. Springer International Publishing. DOI: doi.org/10.1007/978-3-319-02090-7 (see page 2).

Amsallem, D. and C. Farhat (2011). "An Online Method for Interpolating Linear Parametric Reduced-Order Models". In: *SIAM Journal on Scientific Computing* 33.5, pp. 2169–2198 (see page 21).

Antoulas, A. C. (2005). *Approximation of Large-Scale Dynamical Systems*. Advances in Design and Control. SIAM (see page 26).

Asch, M., M. Bocquet, and M. Nodet (2016). *Data assimilation*. Fundamentals of Algorithms. Society for Industrial and Applied Mathematics. ISBN: 978-1-61197-453-9 (see pages 6, 48, 57).

Aubry, Nadine (1991). "On the hidden beauty of the proper orthogonal decomposition". In: *Theoretical and Computational Fluid Dynamics* 2.5, pp. 339–352. DOI: 10.1007/BF00271473 (see page 3).

Aubry, Nadine, Philip Holmes, John L. Lumley, and Emily Stone (1988). "The dynamics of coherent structures in the wall region of a turbulent boundary layer". In: *Journal of Fluid Mechanics* 192, pp. 115–173. DOI: 10.1017/S0022112088001818 (see pages 5, 34, 179).

Bagheri, Shervin (2013). "Koopman-mode decomposition of the cylinder wake". In: *Journal of Fluid Mechanics* 726, pp. 596–623. DOI: 10.1017/jfm.2013.249 (see page 205).

Bagheri, Shervin, Luca Brandt, and Dan S Henningson (2009). "Input-output analysis, model reduction and control of the flat-plate boundary layer". In: *Journal of Fluid Mechanics* 620.2, pp. 263–298. DOI: 10.1017/S0022112008004394 (see page 4).

Bao, A. and E. Gildin (2017). "Data-driven model reduction based on sparsity-promoting methods for multiphase flow in porous media". In: *SPE Latin America and Caribbean Petroleum Engineering Conference*. Society of Petroleum Engineers (see page 26).

Barbagallo, Alexandre, Denis Sipp, and Peter J Schmid (2009). "Closed-loop control of an open cavity flow using reduced-order models". In: *Journal of Fluid Mechanics* 641, pp. 1–50. DOI: 10.1017/S0022112009991418 (see page 4).

Baur, Ulrike, Christopher Beattie, Peter Benner, and Serkan Gugercin (2011). "Interpolatory projection methods for parameterized model reduction". In: *SIAM Journal on Scientific Computing* 33.5, pp. 2489–2518. DOI: 10.1137/090776925 (see page 4).

Baur, Ulrike and Peter Benner (2008). "Cross-Gramian based model reduction for data-sparse systems". In: *Electronic Transactions on Numerical Analysis* 31.256-270, p. 27. ISSN: 1068-9613 (see page 4).

Benard, N, A Debien, L David, and E Moreau (2010). "Analyse par PIV rapide du sillage d'un cylindre manipulé par actionneurs plasmas". In: *Congres Francophone de Techniques Laser, Vandoeuvre-les-Nancy*, pp. 14–17 (see page 131).

Benner, Peter, Serkan Gugercin, and Karen Willcox (2015). "A survey of projection-based model reduction methods for parametric dynamical systems". In: *SIAM Review* 57.4, pp. 483–531. DOI: 10.1137/130932715 (see page 1).

Bergmann, M. and L. Cordier (2008a). "Optimal control of the cylinder wake in the laminar regime by Trust-Region methods and POD Reduced Order Models". In: *J. Comp. Phys.* 227, pp. 7813–7840 (see pages 2, 21, 30, 34).

Bergmann, M. and L. Cordier (2008b). "Optimal control of the cylinder wake in the laminar regime by trust-region methods and POD reduced-order models". In: *Journal of Computational Physics* 227.16, pp. 7813–7840. DOI: 10.1016/j.jcp.2008.04.034 (see pages 4, 17).

Bergmann, M., L. Cordier, and J.-P. Brancher (2005a). "Optimal rotary control of the cylinder wake using POD Reduced Order Model". In: *Phys. Fluids* 17.9, 097101:1–21 (see page 30).

Bergmann, Michel, C. -H. Bruneau, and Angelo Iollo (2009). "Enablers for robust POD models". In: *Journal of Computational Physics* 228.2, pp. 516–538. DOI: 10.1016/j.jcp.2008.09.024 (see page 34).

Bergmann, Michel, Laurent Cordier, and Jean-Pierre Brancher (2005b). "Optimal rotary control of the cylinder wake using proper orthogonal decomposition reduced-order model". In: *Physics of fluids* 17.9, p. 097101. DOI: 10.1063/1.2033624 (see page 4).

Bergstra, James and Yoshua Bengio (2012). "Random search for hyper-parameter optimization". In: *Journal of Machine Learning Research* 13.2 (see pages 181, 204).

Berkooz, Gal (1992). "Observations on the proper orthogonal decomposition". In: *Studies in Turbulence*, pp. 229–247. DOI: 10.1007/978-1-4612-2792-2_16 (see page 3).

Berkooz, Gal, Philip Holmes, and John L. Lumley (1993). "The proper orthogonal decomposition in the analysis of turbulent flows". In: *Annual Review of Fluid Mechanics* 25.1, pp. 539–575. DOI: 10.1146/annurev.fl.25.010193.002543 (see page 4).

Bistrian, D. A. and I. M. Navon (2014). "Comparison of optimized Dynamic Mode Decomposition vs POD for the shallow water equations model reduction with large-time-step observations". In: *International Journal for Numerical Methods in Fluids*, pp. 1–25. DOI: 10.1002/fld (see page 26).

Bogey, Christophe and Christophe Bailly (2006). "Computation of a high Reynolds number jet and its radiated noise using large eddy simulation based on explicit filtering". In: *Computers & fluids* 35.10, pp. 1344–1358. DOI: 10.1016/j.compfluid.2005.04.008 (see page 147).

Bond, Bradley N. and Luca Daniel (2008). "Guaranteed stable projection-based model reduction for indefinite and unstable linear systems". In: *2008 IEEE/ACM Interna-*

tional Conference on Computer-Aided Design. IEEE, pp. 728–735. DOI: 10.5555/1509456.1509615 (see page 6).

Bourgois, Laurent, Gilles Roussel, and Mohammed Benjelloun (2011). "Kalman d'ensemble état-paramètres appliqué au modèle de Lorenz". In: *3èmes Journées Identification et Modélisation Expérimentale* (see page 112).

Brochu, Eric, Vlad M. Cora, and Nando de Freitas (2010). *A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning*. arXiv: 1012.2599 [cs.LG] (see pages 181, 204).

Brunton, S. L., B. R. Noack, and P. Koumoutsakos (2019). "Machine learning for fluid mechanics". In: *Annual Review of Fluid Mechanics* 52. DOI: 10.1146/annurev-fluid-010719-060214 (see pages 36, 64, 204).

Brunton, S. L., J. L. Proctor, and J. N. Kutz (2016a). "Discovering governing equations from data by sparse identification of nonlinear dynamical systems". In: *PNAS* 113.15, pp. 3932–3937. DOI: 10.1073/pnas.1517384113 (see pages 5, 36, 41, 42, 250).

Brunton, Steven L and Bernd R Noack (2015). "Closed-loop turbulence control: Progress and challenges". In: *Applied Mechanics Reviews* 67.5. DOI: 10.1115/1.4031175 (see page 4).

Brunton, Steven L, Joshua L Proctor, and J Nathan Kutz (2016b). "Sparse identification of nonlinear dynamics with control (SINDYc)". In: *IFAC-PapersOnLine* 49.18, pp. 710–715. DOI: 10.1016/j.ifacol.2016.10.249 (see pages 4, 205).

Brunton, Steven L., Bingni W. Brunton, Joshua L. Proctor, and J. Nathan Kutz (2016). "Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control". In: *PLOS ONE* 11, pp. 1–19. DOI: 10.1371/journal.pone.0150171 (see page 4).

Brunton, Steven L. and J. Nathan Kutz (2019). *Data-driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press. DOI: 10.1017/9781108380690 (see pages 25, 64).

Buitinck, Lars, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, et al. (2013). "API design for machine learning software: Experiences from the scikit-learn project". In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pp. 108–122 (see page 46).

Cao, Yanhua, Jiang Zhu, I Michael Navon, and Zhendong Luo (2007). "A reduced-order approach to four-dimensional variational data assimilation using proper orthogonal decomposition". In: *International Journal for Numerical Methods in Fluids* 53.10, pp. 1571–1583. DOI: 10.1002/fld.1365 (see page 4).

Carlstein, Edward (1986). "The use of subseries values for estimating the variance of a general statistic from a stationary sequence". In: *The Annals of Statistics* 14.3, pp. 1171–1179. DOI: 10.1214/aos/1176350057 (see page 91).

Carrassi, Alberto, Marc Bocquet, Laurent Bertino, and Geir Evensen (2018). "Data assimilation in the geosciences: An overview of methods, issues, and perspectives". In: *Wiley Interdisciplinary Reviews: Climate Change* 9.5, e535. DOI: 10.1002/wcc.535 (see pages 47, 51, 52).

Casenave, Fabien, Alexandre Ern, and Tony Lelievre (2015). "A nonintrusive reduced basis method applied to aeroacoustic simulations". In: *Advances in Computational Mathematics* 41.5, pp. 961–986. DOI: 10.1007/s10444-014-9365-0 (see pages 6, 64).

Champion, Kathleen, Peng Zheng, Aleksandr Y. Aravkin, Steven L. Brunton, and Kutz J. Nathan (2020). "A unified sparse optimization framework to learn parsimonious physics-informed models from data". In: *arXiv* 1906.10612, p. 22 (see page 42).

Chartrand, Rick (2011). "Numerical differentiation of noisy, nonsmooth data". In: *ISRN Applied Mathematics* 2011. DOI: 10.5402/2011/164564 (see pages 41, 82).

Chen, Kevin K., Jonathan H. Tu, and Clarence W. Rowley (2012). "Variants of Dynamic Mode Decomposition: Boundary condition, Koopman, and Fourier analyses". In: *Journal of Nonlinear Science* 22.6, pp. 887–915. DOI: 10.1007/s00332-012-9130-9 (see page 24).

Chen, Ricky T. Q., Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud (2018). "Neural ordinary differential equations". In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., pp. 6571–6583 (see page 65).

Chen, Sheng and Steve A. Billings (1989). "Representations of non-linear systems: The NARMAX model". In: *International Journal of Control* 49.3, pp. 1013–1032. DOI: 10.1080/00207178908559683 (see page 38).

Chiu, Tai-Yih (1996). "Model reduction by the low-frequency approximation balancing method for unstable systems". In: *IEEE transactions on Automatic Control* 41.7, pp. 995–997. DOI: 10.1109/9.508903 (see page 4).

Cordier, L. and M. Bergmann (2008a). "Proper Orthogonal Decomposition: an overview". In: *Lecture series 2002-04, 2003-03 and 2008-01 on post-processing of experimental and numerical data*. ISBN 978-2-930389-80-X. Von Kármán Institute for Fluid Dynamics, pp. 1–46 (see pages 4, 12, 13).

Cordier, L. and M. Bergmann (2008b). "Two typical applications of POD: coherent structures eduction and reduced order modelling". In: *Lecture series 2002-04, 2003-03 and 2008-01 on post-processing of experimental and numerical data*. ISBN 978-2-930389-80-X. Von Kármán Institute for Fluid Dynamics, pp. 1–60 (see page 26).

Cordier, L., B. Abou El Majd, and J. Favier (2010). "Calibration of POD reduced-order models using Tikhonov regularization". In: *International Journal for Numerical Methods in Fluids* 63.2, pp. 269–296. ISSN: 1097-0363. DOI: 10.1002/fld.2074 (see pages 30, 33, 36, 40).

Cordier, Laurent, Bernd R. Noack, Gilles Tissot, Guillaume Lehnasch, Joël Delville, Maciej Balajewicz, Guillaume Daviller, and Robert K. Niven (2013). "Identification strategies for model-based control". In: *Experiments in Fluids* 54.8, p. 1580. ISSN: 0723-4864, 1432-1114. DOI: 10.1007/s00348-013-1580-9 (see page 35).

Crutchfield, James P. and B. S. McNamara (1987). "Equations of motion from a data series". In: *Complex Systems* 1, pp. 417–452 (see page 37).

Deane, A. E., I. G. Kevrekidis, G. E. Karniadakis, and S. A. Orszag (1991). "Low-dimensional models for complex geometry flows: Application to grooved channels and circular cylinders". In: *Physics of Fluids A: Fluid Dynamics* 3.10, pp. 2337–2354. DOI: 10.1063/1.857881 (see pages 4, 31, 32).

Degroote, Joris, Jan Vierendeels, and Karen Willcox (2010). "Interpolation among reduced-order matrices to obtain parameterized models for design, optimization and probabilistic analysis". In: *International Journal for Numerical Methods in Fluids* 63.2, pp. 207–230. DOI: 10.1002/fld.2089 (see page 4).

Delville, Joël (1994). "Characterization of the organization in shear layers via the proper orthogonal decomposition". In: *Applied Scientific Research* 53.3, pp. 263–281. DOI: 10.1007/BF00849104 (see page 3).

Deng, Chao, Pan Liu, Shenglian Guo, Zejun Li, and Dingbao Wang (2016). "Identification of hydrological model parameter variation using ensemble Kalman filter". In: *Hydrology and Earth System Sciences* 20.12, pp. 4949–4961. DOI: 10.5194/hess-20-4949-2016 (see page 116).

Dimitriu, Gabriel and Narcisa Apreutesei (2007). "Comparative study with data assimilation experiments using proper orthogonal decomposition method". In: *International Conference on Large-Scale Scientific Computing*. Springer, pp. 393–400. DOI: 10.1007/978-3-540-78827-0_44 (see page 4).

Efron, B. and R.J. Tibshirani (1994). *An Introduction to the Bootstrap*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis. ISBN: 9780412042317 (see page 86).

Efron, Bradley (1987). "Better bootstrap confidence intervals". In: *Journal of the American statistical Association* 82.397, pp. 171–185. DOI: 10.1080/01621459.1987.10478410 (see page 86).

Efron, Bradley (1979). "Bootstrap methods: Another look at the Jackknife". In: *The Annals of Statistics* 7.1, pp. 1–26. DOI: 10.1214/aos/1176344552 (see pages 5, 84).

Efron, Bradley, Trevor Hastie, Iain Johnstone, and Robert Tibshirani (2004). "Least angle regression". In: *The Annals of Statistics* 32.2, pp. 407–499. DOI: 10.1214/009053604000000067 (see pages 5, 43, 44, 251).

Evensen, Geir (2009). *Data Assimilation: The Ensemble Kalman Filter*. 2nd ed. Springer-Verlag. ISBN: 978-3-642-03710-8 (see pages 49, 51).

Evensen, Geir (1994). "Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics". In: *Journal of Geophysical Research: Oceans* 99.C5, pp. 10143–10162. ISSN: 2156-2202. DOI: 10.1029/94JC00572 (see pages 6, 55).

Flyvbjerg, Henrik and Henrik Gordon Petersen (1989). "Error estimates on averages of correlated data". In: *The Journal of Chemical Physics* 91.1, pp. 461–466. DOI: 10.1063/1.457480 (see page 93).

Fox, John (2015). *Applied Regression Analysis and Generalized Linear Models*. 3rd ed. Sage Publications. ISBN: 978-1-483-32131-8 (see pages 86, 88).

Frangos, M., Y. Marzouk, K. Willcox, and B. van Bloemen Waanders (2010). "Surrogate and reduced-order modeling: A comparison of approaches for large-scale statistical inverse problems". In: *Large-Scale Inverse Problems and Quantification of Uncertainty*. John Wiley & Sons, Ltd. Chap. 7, pp. 123–149. DOI: 10.1002/9780470685853.ch7 (see page 5).

Galletti, B, CH Bruneau, Luca Zannetti, and Angelo Iollo (2004). "Low-order modelling of laminar flow regimes past a confined square cylinder". In: *Journal of Fluid Mechanics* 503, pp. 161–170. DOI: 10.1017/S0022112004007906 (see page 33).

Gavish, Matan and David L. Donoho (2014). "The optimal hard threshold for singular values is 4/sqrt(3)". In: *IEEE Transactions on Information Theory* 60.8, pp. 5040–5053 (see page 227).

Gillies, E. A. (1998). "Low-dimensional control of the circular cylinder wake". In: *Journal of Fluid Mechanics* 371, pp. 157–178. DOI: 10.1017/S0022112098002122 (see page 6).

Glorot, Xavier and Yoshua Bengio (2010). "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, pp. 249–256 (see page 70).

González-García, R., R. Rico-Martínez, and I.G. Kevrekidis (1998). "Identification of distributed parameter systems: A neural net based approach". In: *Computers & Chemical Engineering* 22, S965–S968. DOI: 10.1016/S0098-1354(98)00191-4 (see page 38).

Goodfellow, Ian, Yoshua Bengio, Aaron Courville, and Yoshua Bengio (2016). *Deep Learning*. Vol. 1. 2. MIT Press, Cambridge. ISBN: 978-0262035613 (see pages 70, 72, 75, 181).

Guenot, Marc, Ingrid Lepot, Caroline Sainvitu, Jordan Goblet, and Rajan Filomeno Coelho (2013). "Adaptive sampling strategies for non-intrusive POD-based surrogates". In: *Engineering Computations: International Journal for Computer-Aided Engineering* 30.4, pp. 521–547. DOI: 10.1108/02644401311329352 (see page 7).

Gugercin, Serkan and Athanasios C Antoulas (2004). "A survey of model reduction by balanced truncation and some new results". In: *International Journal of Control* 77.8, pp. 748–766. DOI: 10.1080/00207170410001713448 (see page 4).

Haasdonk, Bernard and Mario Ohlberger (2008). "Reduced basis method for finite volume approximations of parametrized linear evolution equations". In: *ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique* 42.2, pp. 277–302. DOI: 10.1051/m2an:2008001 (see page 4).

Hall, Peter (1985). "Resampling a coverage pattern". In: *Stochastic Processes and their Applications* 20.2, pp. 231–246. DOI: 10.1016/0304-4149(85)90212-1 (see page 90).

Hastie, T, R Tibshirani, and J Friedman (2009). *The elements of statistical learning: Data mining, inference, and prediction*. DOI: 10.1007/978-0-387-84858-7 (see pages 39, 43, 44, 100, 232, 233).

Hecht, F. (2013). "New development in FreeFem++". In: *Journal of Numerical Mathematics* 20.3-4, pp. 251–266. ISSN: 1569-3953. DOI: 10.1515/jnum-2012-0013 (see page 205).

Hesthaven, Jan S. and Stefano Ubbiali (2018). "Non-intrusive reduced order modeling of nonlinear problems using neural networks". In: *Journal of Computational Physics* 363, pp. 55–78. DOI: 10.1016/j.jcp.2018.02.037 (see pages 6, 64).

Himpe, Christian and Mario Ohlberger (2014). "Cross-gramian-based combined state and parameter reduction for large-scale control systems". In: *Mathematical Problems in Engineering* 2014. DOI: 10.1155/2014/843869 (see page 4).

Hoepffner, Jérôme, Espen Akervik, Uwe Ehrenstein, and Dan S. Henningson (2006). "Control of cavity-driven separated boundary layer". In: *3rd Conference on Active Flow Control*. Berlin, Germany (see page 4).

Holmes, Philip, John L. Lumley, Gahl Berkooz, and Clarence W. Rowley (2012). *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*. 2nd ed. Cam-

bridge Monographs on Mechanics. Cambridge University Press. DOI: 10.1017/CBO9780511919701 (see pages 3, 4).

Ioffe, Sergey and Christian Szegedy (2015). *Batch normalization: Accelerating deep network training by reducing internal covariate shift*. arXiv: 1502.03167 [cs.LG] (see page 71).

Iollo, Angelo, Alain Dervieux, Jean-Antoine Désidéri, and Stéphane Lanteri (2000). "Two stable POD-based approximations to the Navier–Stokes equations". In: *Computing and Visualization in Science* 3.1-2, pp. 61–66. DOI: 10.1007/s007910050052 (see page 6).

Iollo, Angelo, Stéphane Lanteri, and J-A Désidéri (2000). "Stability properties of POD–Galerkin approximations for the compressible Navier–Stokes equations". In: *Theoretical and Computational Fluid Dynamics* 13.6, pp. 377–396. DOI: 10.1007/s001620050119 (see pages 6, 179).

Ito, Kazufumi and Sivaguru S Ravindran (2001). "Reduced basis method for optimal control of unsteady viscous flows". In: *International Journal of Computational Fluid Dynamics* 15.2, pp. 97–113. DOI: 10.1080/10618560108970021 (see page 4).

Iuliano, Emiliano and Domenico Quagliarella (2013). "Aerodynamic shape optimization via non-intrusive POD-based surrogate modelling". In: *2013 IEEE Congress on Evolutionary Computation*. IEEE, pp. 1467–1474. DOI: 10.1109/CEC.2013.6557736 (see page 6).

Jeong, S. H. and B. Bienkiewicz (1997). "Application of autoregressive modeling in proper orthogonal decomposition of building wind pressure". In: *Journal of Wind Eengineering and Industrial Aerodynamics* 69, pp. 685–695. DOI: 10.1016/S0167-6105(97)00197-9 (see page 5).

Juang, Jer-Nan and Richard S. Pappa (1985). "An eigensystem realization algorithm for modal parameter identification and model reduction". In: *Journal of Guidance, Control, and Dynamics* 8.5, pp. 620–627. DOI: 10.2514/3.20031 (see page 38).

Kalashnikova, Irina and Matthew F. Barone (2010). "On the stability and convergence of a Galerkin reduced order model (ROM) of compressible flow with solid wall and far-field boundary treatment". In: *International Journal for Numerical Methods in Engineering* 83.10, pp. 1345–1375. DOI: 10.1002/nme.2867 (see page 6).

Kalman, R. E. (1960). "A new approach to linear filtering and prediction problems". In: *Journal of Basic Engineering* 82.1, pp. 35–45. DOI: 10.1115/1.3662552 (see page 48).

Kerhervé, Franck, P Jordan, AVG Cavalieri, Joël Delville, Christophe Bogey, and Daniel Juvé (2012). "Educing the source mechanism associated with downstream radiation in subsonic jets". In: *Journal of Fluid Mechanics* 710, p. 606. DOI: 10.1017/jfm.2012.378 (see pages 112, 147).

Kevrekidis, Ioannis G. and Giovanni Samaey (2009). "Equation-free multiscale computation: Algorithms and applications". In: *Annual Review of Physical Chemistry* 60, pp. 321–344. DOI: 10.1146/annurev.physchem.59.032607.093610 (see page 38).

Kho, S., C. Baker, and R. Hoxey (2002). "POD/ARMA reconstruction of the surface pressure field around a low rise structure". In: *Journal of Wind Engineering and Industrial Aerodynamics* 90.12-15, pp. 1831–1842. DOI: 10.1016/S0167-6105(02)00291-X (see page 5).

Kingma, D. P and J. Ba (2014). "Adam: A method for stochastic optimization". In: arXiv: 1412.6980 [cs.LG] (see page 75).

Kreiss, Jens-Peter and Efstathios Paparoditis (2011). "Bootstrap methods for dependent data: A review". In: *Journal of the Korean Statistical Society* 40.4, pp. 357–378. DOI: 10.1016/j.jkss.2011.08.009 (see page 90).

Kunisch, Karl and Stefan Volkwein (1999). "Control of the Burgers equation by a reduced-order approach using proper orthogonal decomposition". In: *Journal of Optimization Theory and Applications* 102.2, pp. 345–371. DOI: 10.1023/A:1021732508059 (see page 3).

Kunisch, Karl, Stefan Volkwein, and Lei Xie (2004). "HJB-POD-based feedback design for the optimal control of evolution problems". In: *SIAM Journal on Applied Dynamical Systems* 3.4, pp. 701–722. DOI: 10.1137/030600485 (see page 4).

Kunsch, Hans R (1989). "The jackknife and the bootstrap for general stationary observations". In: *The Annals of Statistics*, pp. 1217–1241. DOI: 10.1214/aos/1176347265 (see page 91).

Kutz, J. Nathan (2013). *Data-driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data*. Oxford University Press. ISBN: 9780191635885 (see pages 6, 25, 33, 36).

Kutz, J. Nathan, Steven L. Brunton, Bingni W. Brunton, and Joshua L. Proctor (2016). *Dynamic Mode Decomposition: Data-driven Modeling of Complex Systems*. SIAM. DOI: 10.1137/1.9781611974508 (see pages 4, 25).

Kutz, J. Nathan, Xing Fu, and Steven L. Brunton (2016). "Multiresolution Dynamic Mode Decomposition". In: *SIAM Journal on Applied Dynamical Systems* 15.2, pp. 713–735. DOI: 10.1137/15M1023543 (see page 25).

Lahiri, Soumendra N (1999). "Theoretical comparisons of block bootstrap methods". In: *The Annals of Statistics*, pp. 386–404. DOI: 10.1214/aos/1018031117 (see pages 91, 92).

Lall, Sanjay, Jerrold E Marsden, and Sonja Glavaški (2002). "A subspace approach to balanced truncation for model reduction of nonlinear control systems". In: *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal* 12.6, pp. 519–535. DOI: 10.1002/rnc.657 (see page 4).

Landau, L. D. and E. M. Lifshitz (1987). *Fluid Mechanics*. 2nd English edition. Revised. Vol. 6. Course of Theoretical Physics. Pergamon Press, Oxford. ISBN: 9781483161044 (see page 31).

LeGresley, Patrick and Juan Alonso (2000). "Airfoil design optimization using reduced order models based on proper orthogonal decomposition". In: *Fluids 2000 Conference and Exhibit*, p. 2545. DOI: 10.2514/6.2000-2545 (see page 2).

Loiseau, Jean-Christophe and Steven L. Brunton (2018). "Constrained sparse Galerkin regression". In: *Journal of Fluid Mechanics* 838, pp. 42–67. DOI: 10.1017/jfm.2017.823 (see page 4).

Lui, Hugo F. S. and William R. Wolf (2019). "Construction of reduced order models for fluid flows using deep feedforward neural networks". In: *Journal of Fluid Mechanics* 872, pp. 963–994. DOI: :10.1017/jfm.2019.358 (see page 65).

Lumley, J. L. (1967). "The structure of inhomogeneous turbulent flows". In: *Atmospheric Turbulence and Radio Propagation*. Ed. by A. M. Yaglom and V. I. Tatarski. Nauka, pp. 166–178 (see pages 3, 12, 15, 225).

Mathelin, Lionel, M. Yousuff Hussaini, and Thomas A. Zang (2005). "Stochastic approaches to uncertainty quantification in CFD simulations". In: *Numerical Algorithms* 38.1-3, pp. 209–236. DOI: 10.1007/BF02810624 (see page 2).

Mehrmann, Volker and Tatjana Stykel (2005). "Balanced truncation model reduction for large-scale systems in descriptor form". In: *Dimension Reduction of Large-Scale Systems*. Springer, pp. 83–115. DOI: 10.1007/3-540-27909-1_3 (see page 4).

Mendonça, Gonçalo, Frederico Afonso, and Fernando Lau (2019). "Model order reduction in aerodynamics: Review and applications". In: *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering* 233.15, pp. 5816–5836. DOI: 10.1177/0954410019853472 (see page 4).

Mezić, Igor (2013). "Analysis of fluid flows via spectral properties of the Koopman operator". In: *Annual Review of Fluid Mechanics* 45, pp. 357–378. DOI: 10.1146/annurev-fluid-011212-140652 (see pages 3, 4).

Mohan, Arvind T and Datta V Gaitonde (2018). "A deep learning based approach to reduced order modeling for turbulent flow control using LSTM neural networks". In: arXiv: 1804.09269 [physics.comp-ph] (see page 64).

Montáns, Francisco J., Francisco Chinesta, Rafael Gómez-Bombarelli, and J. Nathan Kutz (2019). "Data-driven modeling and learning in science and engineering". In: *Comptes Rendus Mécanique* 347.11, pp. 845–855. DOI: 10.1016/j.crme.2019.11.009 (see page 37).

Montúfar, Guido, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio (2014). *On the number of linear regions of deep neural networks*. arXiv: 1402.1869 [stat.ML] (see page 70).

Moosavi, Azam, Razvan Stefanescu, and Adrian Sandu (2015). *Efficient construction of local parametric reduced order models using machine learning techniques*. arXiv: 1511.02909 [cs.LG] (see page 65).

Moradkhani, Hamid, Soroosh Sorooshian, Hoshin V. Gupta, and Paul R. Houser (2005). "Dual state-parameter estimation of hydrological models using ensemble Kalman filter". In: *Advances in Water Resources* 28.2, pp. 135–147. ISSN: 0309-1708. DOI: 10.1016/j.advwatres.2004.09.002 (see pages 6, 38, 58, 60).

Muddada, Sridhar and B. S. V. Patnaik (2010). "An active flow control strategy for the suppression of vortex structures behind a circular cylinder". In: *European Journal of Mechanics-B/Fluids* 29.2, pp. 93–104. DOI: 10.1016/j.euromechflu.2009.11.002 (see pages 207, 208).

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. Cambridge: MIT Press (see page 227).

Noack, Bernd R, Michael Schlegel, Boye Ahlborn, Gerd Mutschke, Marek Morzyński, Pierre Comte, and Gilead Tadmor (2008). "A finite-time thermodynamics of unsteady fluid flows". In: *Journal of Non-Equilibrium Thermodynamics* 33.2, pp. 103–148. DOI: 10.1515/JNETDY.2008.006 (see page 35).

Noack, Bernd R., Konstantin Afanasiev, Marek Morzyński, Gilead Tadmor, and Frank Thiele (2003). "A hierarchy of low-dimensional models for the transient and post-transient cylinder wake". In: *Journal of Fluid Mechanics* 497, pp. 335–363. DOI: 10.1017/S0022112003006694 (see pages 3, 31, 205).

Noack, Bernd R. and Helmut Eckelmann (1994). "A low-dimensional Galerkin method for the three-dimensional flow around a circular cylinder". In: *Physics of Fluids* 6.1, pp. 124–143. DOI: 10.1063/1.868433 (see page 5).

Noack, Bernd R., Marek Morzyński, and Gilead Tadmor, eds. (2011). *Reduced-order Modelling for Flow Control*. CISM International Centre for Mechanical Sciences. Springer-Verlag. ISBN: 978-3-7091-0757-7 (see pages 4, 6, 20, 27, 34, 35).

Noack, Bernd R., Paul Papas, and Peter A. Monkewitz (2005). "The need for a pressure-term representation in empirical Galerkin models of incompressible shear flows". In: *Journal of Fluid Mechanics* 523, p. 339. DOI: 10.1017/S0022112004002149 (see page 31).

Östh, Jan, Bernd R Noack, Siniša Krajnović, Diogo Barros, and Jacques Borée (2014). "On the need for a nonlinear subscale turbulence term in POD models as exemplified for a high-Reynolds-number flow over an Ahmed body". In: *Journal of Fluid Mechanics* 747, pp. 518–544. DOI: 10.1017/jfm.2014.168 (see pages 5, 6, 35).

Panzer, Heiko, Jan Mohring, Rudy Eid, and Boris Lohmann (2010). "Parametric model order reduction by matrix interpolation". In: *at-Automatisierungstechnik Methoden und Anwendungen der Steuerungs-* 58.8, pp. 475–484. DOI: 10.1524/auto.2010.0863 (see page 4).

Park, Sungbae, Adam Wachsman, Anuradha Annaswamy, Ahmed Ghoniem, B Pang, and Ken Yu (2004). "Experimental study of pod-based control for combustion instability using a linear photodiode array". In: *42nd AIAA Aerospace Sciences Meeting and Exhibit*, p. 639 (see page 5).

Parks, Thomas W. and C. Sidney Burrus (1987). *Digital Dilter Design*. Wiley-Interscience. ISBN: 978-0-471-82896-9 (see page 81).

Patton, Andrew, Dimitris N Politis, and Halbert White (2009). "Correction to "Automatic block-length selection for the dependent bootstrap" by D. Politis and H. White". In: *Econometric Reviews* 28.4, pp. 372–375. DOI: 10.1080/07474930802459016 (see page 93).

Pawar, Suraj, S. M. Rahman, H. Vaddireddy, Omer San, Adil Rasheed, and Prakash Vedula (2019). "A deep learning enabler for nonintrusive reduced order modeling of fluid flows". In: *Physics of Fluids* 31.8, p. 085101. DOI: 10.1063/1.5113494 (see pages 6, 7, 65, 253).

Peña, F Lopez, V Díaz Casás, A Gosset, and RJ Duro (2012). "A surrogate method based on the enhancement of low fidelity computational fluid dynamics approximations by artificial neural networks". In: *Computers & fluids* 58, pp. 112–119. DOI: 10.1016/j.compfluid.2012.01.008 (see page 6).

Perret, L., E. Collin, and J. Delville (2006). "Polynomial identification of POD based low-order dynamical system". In: *Journal of Turbulence* 7, N17. DOI: 10.1080/14685240600559665 (see pages 5, 36, 40).

Peterson, Janet S (1989). "The reduced basis method for incompressible viscous flow calculations". In: *SIAM Journal on Scientific and Statistical Computing* 10.4, pp. 777–786. DOI: 10.1137/0910047 (see page 4).

Petzold, Linda (1983). "Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations". In: *SIAM Journal on Scientific and Statistical Computing* 4.1, pp. 136–148. DOI: 10.1137/0904010 (see pages 94, 97).

Podvin, Bérengère and John Lumley (1998). "A low-dimensional approach for the minimal flow unit". In: *Journal of Fluid Mechanics* 362, pp. 121–155. DOI: 10.1017/S0022112098008854 (see pages 3, 34).

Politis, Dimitris N and Joseph P Romano (1992). "A circular block-resampling procedure for stationary data". In: *Exploring the Limits of Bootstrap*. Wiley, New York, pp. 263–270 (see page 91).

Politis, Dimitris N and Joseph P Romano (1994). "The stationary bootstrap". In: *Journal of the American Statistical association* 89.428, pp. 1303–1313. DOI: 10.2307/2290993 (see page 91).

Politis, Dimitris N and Halbert White (2004). "Automatic block-length selection for the dependent bootstrap". In: *Econometric Reviews* 23.1, pp. 53–70. DOI: 10.1081/ETC-120028836 (see page 93).

Press, William H., Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery (1993). *Numerical Recipes in FORTRAN; The Art of Scientific Computing*. 2nd. Cambridge University Press. ISBN: 0521437164 (see pages 5, 41).

Protas, Bartosz, Bernd R. Noack, and Jan Östh (2015). "Optimal nonlinear eddy viscosity in Galerkin models of turbulent flows". In: *Journal of Fluid Mechanics* 766, pp. 337–367. DOI: 10.1017/jfm.2015.14 (see pages 6, 34).

Pyta, L. and D. Abel (2017). "Online model adaption of reduced order models for fluid flows". In: *IFAC-PapersOnLine* 50.1, pp. 11138–11143. DOI: 10.1016/j.ifacol.2017.08.1006 (see page 34).

Qin, Tong, Kailiang Wu, and Dongbin Xiu (2019). "Data driven governing equations approximation using deep neural networks". In: *Journal of Computational Physics* 395, pp. 620–635. DOI: 10.1016/j.jcp.2019.06.042 (see page 65).

Qiu, C and J Chou (2006). "Four-dimensional data assimilation method based on SVD: Theoretical aspect". In: *Theoretical and applied climatology* 83.1, pp. 51–57. DOI: 10.1007/s00704-005-0162-z (see page 4).

Raisee, M., D. Kumar, and C. Lacor (2015). "A non-intrusive model reduction approach for polynomial chaos expansion using proper orthogonal decomposition". In: *International Journal for Numerical Methods in Engineering* 103.4, pp. 293–312. DOI: 10.1002/nme.4900 (see page 7).

Raissi, M., P. Perdikaris, and G. E. Karniadakis (2018). "Multistep neural networks for data-driven discovery of nonlinear dynamical systems". In: arXiv: 1801.01236 [math.DS] (see pages 64, 65).

Raissi, M., P. Perdikaris, and G. E. Karniadakis (2019). "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational Physics* 378, pp. 686–707. DOI: 10.1016/j.jcp.2018.10.045 (see page 64).

Rajaee, Mojtaba, Sture KF Karlsson, and Lawrence Sirovich (1994). "Low-dimensional description of free-shear-flow coherent structures and their dynamical behaviour". In: *Journal of Fluid Mechanics* 258, pp. 1–29. DOI: 10.1017/S0022112094003228 (see page 5).

Rempfer, D. (1996). "Investigations of boundary layer transition via Galerkin projections on empirical eigenfunctions". In: *Physics of Fluids* 8.1, pp. 175–188. DOI: 10.1063/1.868825 (see pages 5, 34).

Rempfer, D. and H. F. Fasel (1994). "Evolution of three-dimensional coherent structures in a flat-plate boundary layer". In: *Journal of Fluid Mechanics* 260, pp. 351–375. DOI: 10.1017/S0022112094003551 (see pages 34, 119).

Rowley, Clarence W. and Scott T. M. Dawson (2017). "Model reduction for flow analysis and control". In: *Annual Review of Fluid Mechanics* 49.1, pp. 387–417. DOI: 10.1146/annurev-fluid-010816-060042 (see pages 18, 205).

Rowley, Clarence W., Igor Mezić, Shervin Bagheri, Philipp Schlatter, and Dan S. Henningson (2009). "Spectral analysis of nonlinear flows". In: *Journal of Fluid Mechanics* 641.1, pp. 115–127. DOI: 10.1017/S0022112009992059 (see pages 4, 38).

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams (1986). "Learning representations by back-propagating errors". In: *Nature* 323.6088, pp. 533–536. DOI: 10.1038/323533a0 (see page 72).

Sabetghadam, Feriedoun and Alireza Jafarpour (2012). "α regularization of the POD-Galerkin dynamical systems of the Kuramoto–Sivashinsky equation". In: *Applied Mathematics and Computation* 218.10, pp. 6012–6026. DOI: 10.1016/j.amc.2011.11.083 (see page 6).

Sagara, Setsuo, Zi-Jiang Yang, and Kiyoshi Wada (1991). "Identification of continuous systems using digital low-pass filters". In: *International Journal of Systems Science* 22.7, pp. 1159–1176. DOI: 10.1080/00207729108910693 (see page 81).

San, Omer and Romit Maulik (2018). "Neural network closures for nonlinear model order reduction". In: *Advances in Computational Mathematics* 44.6, pp. 1717–1750. DOI: 10.1007/s10444-018-9590-z (see page 64).

San, Omer, Romit Maulik, and Mansoor Ahmed (2019). "An artificial neural network framework for reduced order modeling of transient flows". In: *Communications in Nonlinear Science and Numerical Simulation* 77, pp. 271–287. DOI: 10.1016/j.cnsns.2019.04.025 (see page 65).

Schlegel, Michael and Bernd R Noack (2015). "On long-term boundedness of Galerkin models". In: *Journal of Fluid Mechanics* 765, pp. 325–352. DOI: 10.1017/jfm.2014.736 (see page 5).

Schmid, Peter J. (2010). "Dynamic Mode Decomposition of numerical and experimental data". In: *Journal of Fluid Mechanics* 656, pp. 5–28. DOI: 10.1017/S0022112010001217 (see pages 3, 4, 21, 23, 38).

Schmid, Peter J., Dan S. Henningson, and D. F. Jankowski (2002). *Stability and Transition in Shear Flows*. Vol. 55. Applied Mathematical Reviews 3. ASME, B57–B59. DOI: 10.1115/1.1470687 (see page 3).

Schmidt, M. and H. Lipson (2009). "Distilling free-form natural laws from experimental data". In: *Science* 324.5923, pp. 81–85 (see page 38).

Schwarz, Gideon (1978). "Estimating the dimension of a model". In: *The Annals of Statistics* 6.2, pp. 461–464. DOI: 10.1214/aos/1176344136 (see page 46).

Seid, K. H., Gregor Gilka, R. Leung, and Frank Thiele (2012). "A comparison study of reduced order models for aeroacoustic applications". In: *18th AIAA/CEAS Aeroacoustics Conference (33rd AIAA Aeroacoustics Conference)*, p. 2072 (see page 26).

Sheppard, Kevin, Stanislav Khrapov, Gábor Lipták, et al. (2020). *bashtage/arch*. Version 4.15. DOI: 10.5281/zenodo.593254 (see pages 87, 92, 93).

Silva, Brian de, Kathleen Champion, Markus Quade, Jean-Christophe Loiseau, J Kutz, and Steven Brunton (2020). "PySINDy: A Python package for the sparse identifica-

tion of nonlinear dynamical systems from data". In: arXiv: 2004.08424 [math.DS] (see page 42).

Simader, Christian G. and Hermann Sohr (1992). "A new approach to the Helmholtz decomposition and the Neumann problem in L^q-spaces for bounded and exterior domains". In: *Mathematical problems relating to the Navier-Stokes equations*. Vol. 11. Series on Advances in Mathematics for Applied Sciences. World Scientific, pp. 1–35. DOI: 10.1142/9789814503594_0001 (see page 30).

Sirovich, Lawrence (1987). "Turbulence and the dynamics of coherent structures, Parts I-III". In: *Quarterly of Applied Mathematics* 45.3, pp. 561–590. ISSN: 0033-569X (see pages 3, 17, 225).

Taboga, Marco (2021). *StatLect Digital textbook on probability and statistics*. https://www.statlect.com/. Accessed: 2021-05-02 (see page 237).

Taira, Kunihiko, Steven L. Brunton, Scott T. M. Dawson, Clarence W. Rowley, Tim Colonius, Beverley J. McKeon, Oliver T. Schmidt, Stanislav Gordeyev, Vassilios Theofilis, and Lawrence S. Ukeiley (2017). "Modal analysis of fluid flows: An overview". In: *AIAA Journal* 55.12, pp. 4013–4041. DOI: 10.2514/1.J056060 (see pages 3, 12, 14, 15).

Taira, Kunihiko, Maziar S. Hemati, Steven L. Brunton, Yiyang Sun, Karthik Duraisamy, Shervin Bagheri, Scott T. M. Dawson, and Chi-An Yeh (2020). "Modal analysis of fluid flows: Applications and outlook". In: *AIAA journal* 58.3, pp. 998–1022. DOI: 10.2514/1.J058462 (see page 12).

Theofilis, Vassilios (2011). "Global linear instability". In: *Annual Review of Fluid Mechanics* 43, pp. 319–352. DOI: 10.1146/annurev-fluid-122109-160705 (see page 3).

Tibshirani, Robert (1996). "Regression shrinkage and selection via the lasso". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1, pp. 267–288. DOI: 10.1111/j.2517-6161.1996.tb02080.x (see page 42).

Tibshirani, Ryan J. (2013). "The lasso problem and uniqueness". In: *Electronic Journal of Statistics* 7.0, pp. 1456–1490. DOI: 10.1214/13-ejs815 (see page 42).

Tissot, G. (2014). "Réduction de modèle et contrôle d'écoulements". Thèse de Doctorat. Université de Poitiers (see page 26).

Tissot, Gilles, Laurent Cordier, Nicolas Benard, and Bernd R Noack (2014). "Model reduction using dynamic mode decomposition". In: *Comptes Rendus Mécanique* 342.6-7, pp. 410–416. DOI: 10.1016/j.crme.2013.12.011 (see page 4).

Tu, J., C. Rowley, D. Luchtenburg, S. Brunton, and J. N. Kutz (2014). "On Dynamic Mode Decomposition: Theory and Applications". In: *Journal of Computational Dynamics* 1, pp. 391–421 (see page 23).

Ukeiley, L., L. Cordier, Remi Manceau, J. Delville, M. Glauser, and J. Bonnet (2001). "Examination of large-scale structures in a turbulent plane mixing layer. Part 2. Dynamical systems model". In: *Journal of Fluid Mechanics* 441, pp. 67–108. DOI: 10.1017/S0022112001004803 (see page 5).

Volkwein, S. (2013). *Proper Orthogonal Decomposition: Theory and Reduced-Order Modelling*. Lecture Notes. University of Konstanz. Department of Mathematics and Statistics (see page 13).

Wang, Qian, Jan S Hesthaven, and Deep Ray (2019). "Non-intrusive reduced order modeling of unsteady flows using artificial neural networks with application to a

combustion problem". In: *Journal of Computational Physics* 384, pp. 289–307. DOI: 10.1016/j.jcp.2019.01.031 (see pages 6, 7, 65, 252).

Wang, Zhu, Imran Akhtar, Jeff Borggaard, and Traian Iliescu (2012). "Proper orthogonal decomposition closure models for turbulent flows: A numerical comparison". In: *Computer Methods in Applied Mechanics and Engineering* 237, pp. 10–26. DOI: 10.1016/j.cma.2012.04.015 (see page 3).

Weiss, Julien (2019). "A tutorial on the Proper Orthogonal Decomposition". In: *AIAA Aviation 2019 Forum*, p. 3333. DOI: 10.2514/6.2019-3333 (see page 20).

Wikle, Christopher K and L Mark Berliner (2007). "A Bayesian tutorial for data assimilation". In: *Physica D: Nonlinear Phenomena* 230.1-2, pp. 1–16. DOI: 10.1016/j.physd.2006.09.017 (see pages 51, 52, 241).

Williamson, Charles HK (1996). "Vortex dynamics in the cylinder wake". In: *Annual Review of Fluid Mechanics* 28.1, pp. 477–539. DOI: 10.1146/annurev.fl.28.010196.002401 (see pages 2, 131).

Winter, M and C Breitsamter (2014). *Reduced-order modeling of unsteady aerodynamic loads using radial basis function neural networks*. Deutsche Gesellschaft für Luft-und Raumfahrt-Lilienthal-Oberth eV (see page 7).

Wirtz, Daniel and Bernard Haasdonk (2012). "Efficient a-posteriori error estimation for nonlinear kernel-based reduced systems". In: *Systems & Control Letters* 61.1, pp. 203–211. DOI: 10.3182/20120215-3-AT-3016.00135 (see page 7).

Xiao, Manyu, Piotr Breitkopf, Rajan Filomeno Coelho, Catherine Knopf-Lenoir, Maryan Sidorkiewicz, and Pierre Villon (2010). "Model reduction by CPOD and Kriging". In: *Structural and Multidisciplinary Optimization* 41.4, pp. 555–574. DOI: 10.1007/s00158-009-0434-9 (see page 7).

Yang, Yuhong (2005). "Can the strengths of AIC and BIC be shared? A conflict between model indentification and regression estimation". In: *Biometrika* 92.4, pp. 937–950. DOI: 10.1093/biomet/92.4.937 (see page 46).

Young, Peter C (2002). "Advances in real–time flood forecasting". In: *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 360.1796, pp. 1433–1450. ISSN: 1364503X (see page 6).

Zebib, A. (1987). "Stability of viscous flow past a circular cylinder". In: *Journal of Engineering Mathematics* 21.2, pp. 155–165. DOI: 10.1007/BF00127673 (see page 207).

Zhang, Hong-Quan, Uwe Fey, Bernd R. Noack, Michael König, and Helmut Eckelmann (1995). "On the transition of the cylinder wake". In: *Physics of Fluids* 7.4, pp. 779–794. DOI: 10.1063/1.868601 (see page 207).

Zhang, Linan and Hayden Schaeffer (2019). "On the convergence of the SINDy algorithm". In: *Multiscale Modeling & Simulation* 17.3, pp. 948–972. DOI: 10.1137/18M1189828 (see page 42).

Zheng, Peng, Travis Askham, Steven L Brunton, J Nathan Kutz, and Aleksandr Y Aravkin (2018). "A unified framework for sparse relaxed regularized regression: SR3". In: *IEEE Access* 7, pp. 1404–1423. DOI: 10.1109/ACCESS.2018.2886528 (see page 42).

Zou, Hui, Trevor Hastie, and Robert Tibshirani (2007). "On the "degrees of freedom" of the LASSO". In: *The Annals of Statistics* 35.5, pp. 2173–2192. DOI: 10.1214/009053607000000127 (see page 46).

Résumé

Les modèles *haute fidélité* employés en Turbulence sont inutilisables lorsque des résolutions multiples sont nécessaires, comme c'est le cas en optimisation ou en contrôle des écoulements. La *réduction de modèle* a pour objectif de construire des modèles de dimension réduite (ROMs) afin d'approximer de manière précise la dynamique haute fidélité sous-jacente. La méthode de réduction de modèle par projection de Galerkin, largement utilisée, est *intrusive* car elle nécessite la connaissance des équations d'évolution et/ou d'avoir accès au code source décrivant la physique. La réduction de modèle de type intrusif est donc inadaptée aux problèmes présentant aucune ou une faible connaissance du système physique. Dans ces cas, une alternative est offerte par les méthodes *non intrusives* ou *modélisations basées sur des données* pour lesquelles les modèles réduits sont appris à partir de séries temporelles obtenues par simulations numériques ou expériences. Dans cette thèse, des approches basées "données" de type *intrusif* et *non intrusif* sont présentées pour prédire la dynamique d'écoulements.

Concernant les approches de type intrusif, un modèle réduit de dynamique basé sur la Proper Orthogonal Decomposition (POD-ROM) est considéré. La méthode POD offre l'avantage de préserver la dynamique non linéaire en projetant les équations d'état sur un espace de faible dimension engendré par des modes optimaux. Dans un premier temps, des méthodes de régression creuse issues de l'*apprentissage statistique* sont utilisées pour identifier les inconnues linéaires du modèle réduit. Une *méthode de bootstrap* est ensuite proposée pour quantifier de manière probabiliste les incertitudes associées aux méthodes de régression. Dans un second temps, un modèle non linéaire de viscosité turbulente est ajouté aux équations du modèle réduit. Ce modèle de Turbulence fournit une représentation fermée *basée sur la physique* de la dynamique de l'écoulement. Finalement, les paramètres du modèle de fermeture sont estimés à l'aide d'une approche de type *Dual Ensemble Kalman filter* (Dual EnKF) qui intègre les sorties du modèle et des mesures, tout en prenant en compte les incertitudes respectives.

Concernant les approches de type non intrusif, des modèles de régression basés sur des réseaux de neurones (NN-ROM) sont considérés comme alternative de l'approche POD-ROM. Cette méthode traite les limitations de l'approche POD-ROM – le manque de garantie *a priori* de stabilité, et la nécessité de termes de fermeture pour prendre en compte les modes non résolus – au prix de l'interprétabilité du modèle approché résultant. Le modèle NN-ROM sert de méthode d'intégration temporelle des coefficients de projection POD. Pour cela, un réseau de neurones paramétrisé, multi-pas est introduit pour représenter les termes de résidus. Ce modèle est utilisé dans le cadre d'une méthode d'*assimilation de données* (DA) afin d'améliorer la prédiction à long terme du modèle.

Les approches intrusive et non intrusive proposées sont appliquées sur un système dynamique canonique (Lorenz), sur des données numériques issues de simulations à bas nombre de Reynolds d'un écoulement de sillage et d'un jet à faible nombre de Mach, et enfin sur des données expérimentales d'un écoulement de sillage.

Mots-clés : apprentissage automatique, modélisation réduite (ROM), modèle réduit non intrusif, identification de système, modèle de régression linéaire, estimation bootstrap, assimilation de données, réseaux de neurones profonds, dynamique des fluides

Abstract

High-fidelity models used for solving Turbulent flows are intractable in applications where repeated realizations are required, such as for optimization or flow control. *Model order reduction* aims to construct low-dimensional reduced-order models (ROMs) to accurately approximate the underlying high-fidelity dynamics. The traditional Galerkin projection model reduction is *intrusive* since it requires the knowledge of the governing equations and/or to have access to the source code describing the physical model. Intrusive model reduction is therefore not suited for problems with no or limited knowledge of the physical system. In these cases, an alternative is offered by *nonintrusive* ROMs or pure *data-driven* modelling where reduced models are learnt from time-series data obtained from simulations or experiments. In this thesis, intrusive and nonintrusive data-driven approaches of reduced-order modelling are presented for the dynamical prediction of fluid flows.

For the intrusive approach, Proper Orthogonal Decomposition (POD) based ROM (POD-ROM) is considered. The POD method offers the advantage of preserving the nonlinear dynamics by projecting the governing equations onto low-dimensional optimal modes. First, sparse regression methods issued from *statistical learning* are used to identify the linear unknowns of the ROM. A *bootstrap method* is then proposed to quantify in a probabilistic framework the uncertainties associated with the regression methods. Subsequently, the POD-ROM is augmented with a nonlinear eddy viscosity model that provides an interpretable *physics-based* closed-form representation of the flow dynamics. Finally, the closure term parameters are estimated with a *Dual Ensemble Kalman filter* approach (Dual EnKF) which integrates the model outputs and measurements while taking into account the respective uncertainties.

For the nonintrusive approach, regression models based on Neural Networks (NN-ROM) are considered as an alternative to the POD-ROM. This method addresses the limitations of POD-ROM – the lack of an *a priori* guarantee of stability, and requirement of closure to account for the unresolved modes – at the cost of interpretability of the resulting surrogate model. The derived NN-ROM serves as a time-stepping method for the POD projection coefficients. A novel multistep, residual-based, parametrized neural network is proposed. This framework is augmented with *Data Assimilation* (DA) to provide accurate long-term dynamical predictions.

The proposed intrusive and nonintrusive approaches have been applied on a canonical dynamical system (Lorenz), on numerical data from low Reynolds number simulations of a cylinder wake flow and a low Mach number jet, and finally on experimental data of a wake flow.

Keywords: machine learning, reduced-order modelling (ROM), nonintrusive ROM, system identification, linear regression model, bootstrap estimation, data assimilation, deep neural network, fluid dynamics