Data-driven modeling for fluid dynamics and control

Hao Zhang

A Dissertation
Presented to the Faculty
of Princeton University
in Candidacy for the Degree
of Doctor of Philosophy

Recommended for Acceptance
by the Department of
Mechanical and Aerospace Engineering
Adviser: Professor Clarence W. Rowley

April 2020

Abstract

One of the most critical tasks in fluid dynamics and control is to build simple, low-order, and accurate models. The models are essential for understanding dynamics and control. However, in many cases, the models are either unknown or too complicated to be useful. As an example, fluid flows are governed by Navier-Stokes equations (NSE), which remain intractable for real-time applications. Meanwhile, with increasing computational power and advances in experimental and numerical methods, researchers have access to much more data about dynamical systems. For instance, computational fluid dynamics (CFD) produces tons of data, but the data have not been fully utilized.

Data-driven modeling addresses these challenges by learning dynamical system models from data. This thesis focuses on data-driven modeling methods for applications in fluid dynamics and control. First, we propose an evaluation criterion to quantify the accuracy of dynamic mode decomposition (DMD), a data-driven algorithm for extracting spatial and temporal features about dynamical systems from data. DMD is a numerical approximation to the linear Koopman operator associated with a dynamical system. By exploiting this connection, the accuracy criterion is purely data-driven and physically meaningful. It also applies to other variants of DMD algorithms and assists in model selection.

Second, fast algorithms are developed for online dynamic mode decomposition (ODMD). Given real-time measurement about a dynamical system, this algorithm efficiently updates an adaptive model upon each new snapshot. It reduces both the computational time and memory requirements by order of magnitudes compared with existing methods. ODMD algorithm can be modified to gradually forget old data, which enables faster tracking of dynamics. ODMD also extends to both linear and nonlinear system identification, where control is included.

Finally, we study the input-output response of a separated flow past a flat plate. The analysis is based on the frequency-domain transfer function of the linearized NSE about the mean flow. The control input is body forcing, and the output is the flow field. This analysis sheds light on the optimal control placement and reveals that the trailing edge separation bubble is most sensitive to streamwise body force (control) in upstream of the separation point.

Acknowledgements

This part is hard because it is impossible to list all the people that I am grateful to. Coming to Princeton is one of the best decisions I have made, and the past years have shaped me both personally and professionally. Therefore, I want to thank all the people that guided and accompanied me towards and along this journey. You made me who I am today.

First of all, I am deeply indebted to my advisor, Prof. Clancy Rowley, who provided tremendous help during my PhD study. He convinced me that it is really possible to achieve a healthy balance between research, teaching, and life. Clancy is undoubtedly the most knowledgeable person I have worked with. He taught me about critical thinking and rigorous scientific research. He also showed great patience and empathy when I had hard times. This thesis will never be possible without him.

I would like to acknowledge my PhD committee members, Howard Stone and Naomi Leonard, for their time and guidance over the years. Thanks to my thesis readers, Louis Cattafesta and Anirudha Majumdar, for the valuable feedback. I also thank other faculty and students involved in my general exam and PhD research: Yiguang Ju, Daniel Nosenchuck, Egemen Kolemen, Michael Mueller, Eric Deem. I thank all the MAE staff members, especially Jill Ray, for their administrative support.

I want to thank all the lab members in H125, especially Scott Dawson and Michael Fairchild. They have made the lab a great working place, and I will miss the lunch and discussion we had. Scott offered lots of help during the early stage of my PhD study.

I also thank all my Princeton friends, since they have made my Princeton experience much more enjoyable. There are too many people to list here. Thanks to my MAE friends: He Sun, Xinyi Liu, Ying Liu. I also thank other friends including but not limited to Xinlei Sheng, Yingshi Peng, Linguang Zhang, Kai Gong, Maofeng Liu,

Kengran Yang, Xiaogang He, Renzhi Jing, Manzhuang Zheng. I miss the drinking nights we spend together.

I thank the people that have supported and influenced me before Princeton. I must thank Fenghua Zou, my respected junior high school math teacher. He is the first mentor in my life and encouraged me to be a better version of myself. I am grateful to Mr. Jin, who remained anonymous, for generously providing financial aid during my college years. I am deeply indebted to Prof. Zheng Chen from Peking University, for his kindness and guidance during my college study. I view him as my role model since then.

Special gratitude goes to Yizhi Wang for her continuous understanding, support, and encouragement. She has accompanied me through ups and downs and makes me smile when there is no reason to. Over the past few years, she is the largest source of support when I am physically away from my family. I really look forward to our Corgi puppy soon.

Finally and most importantly, I am deeply grateful to my parents Yaofeng Zhang and Chunfang Yao, and my brother Yao Zhang. Thanks to my parents for their sacrifices over the years for my brother and me. They taught me the importance of education when I was a child. Only with this belief in mind, I can make it here. They respect and trust my choice, and support me all the way along. Words cannot express how thankful I am. I dedicate this thesis to you, my beloved family.

This thesis research is supported by AFOSR grant FA9550-14-1-0289 and DARPA award HR0011-16-C-0116. I also acknowledge Dr. Jeffrey Dwoskin, who provided the thesis template.

This dissertation carries T#3394 in the records of the Department of Mechanical and Aerospace Engineering.

道生一，一生二，二生三，三生万物

– 道德经

The Tao produced One; One produced Two; Two produced Three; Three produced All things.

– Tao Te Ching

Contents

List of Tables

List of Figures

Part I

Introduction

Organization

This thesis focuses on data-driven modeling methods for applications in fluid dynamics and control. Part I summarizes the high-level idea of the thesis contributions. The main results are organized into chapters as follows.

- Chapter 1 outlines the modeling challenges in fluid dynamics and control, and proposes the necessity to develop data-driven modeling methods. The canonical flow separation problem is introduced to motivate the thesis research.
- Chapter 2 presents an accuracy criterion for dynamic mode decomposition. It guides the selection of models to better represent the dynamics in the separated flow. More information can be found in chapter 6.
- Chapter 3 describes fast algorithms for online model learning. These online algorithms lay the foundation for efficient real-time model updating, prediction, and control (e.g., it has been applied to the canonical separated flow problem [18]). For more details, refer to chapter 7.
- Chapter 4 investigates the input-output response of the canonical separated flow, and suggests that the separation bubble is most sensitive to streamwise body force (control) in the upstream of the separation point. It is further elaborated in chapter 8.
- Chapter 5 summarizes the thesis contributions and outlines a few directions for future work.

Notation

This thesis covers multiple fields, such as fluid dynamics, control, and Koopman analysis. Each chapter is mostly self-contained. In terms of notation, we decide to

respect the convection in each field. Therefore, some variables can take different meanings depending on the context. Of course, the notation will be clearly defined when introduced. To avoid any confusion, we make efforts to ensure that the meaning of notation is clear from the context.

Chapter 1

Motivation

1.1 Fluid dynamics

The study of fluid dynamics appears in many aspects of science and engineering and plays an important role in broad fields such as energy [40], health [94], and the environment [100]. The well-known Navier-Stokes equations (NSE) describes the motion of fluid flows (established in the early 18th century [12]). After almost two hundred years, the existence and smoothness of its solution remain unknown, and is one of the seven Millennium Prize Problems [28]. At high Reynolds number, a complicated phenomenon called turbulence will occur, but the turbulence mechanism is not well understood. Richard Feynman considered turbulence as the most important unsolved problem of classical physics [29].

The (non-dimensional) incompressible NSE reads [90]

$$\frac{\partial \boldsymbol{u}}{\partial t} = -\boldsymbol{u} \cdot \nabla \boldsymbol{u} - \nabla p + \frac{1}{Re} \nabla^2 \boldsymbol{u}, \tag{1.1a}$$

$$\nabla \cdot \boldsymbol{u} = 0, \tag{1.1b}$$

where $\boldsymbol{u}$ is the velocity field (in three directions), p is the pressure field, and Re is the Reynolds number. Reynolds number is the most important nondimensional

parameter in fluid dynamics, and fluid flows can exhibit different behaviors under different Reynolds number. NSE is a typical example of nonlinear partial differential equations (PDE).

Fluid dynamics is characterized by high dimensionality and nonlinearity. In practice, NSE is usually spatially discretized, resulting in high-dimensional and nonlinear ordinary differential equations (ODE). However, for many real-time applications, we can not afford such high dimensionality due to the high computational time and memory requirements. For example, consider a spatial domain of $1m \times 1m \times 1m$, the resulting discrete state dimension will easily reach one million if the discretization size is $0.01m$. Also, when the Reynolds number is high, a smaller discretization size has to be used in order to resolve the physics. In particular, the discretization size scales like $Re^{-\frac{3}{4}}$ [3], where Re is the Reynolds number. The total computational time of computational fluid dynamics (CFD) [2] simulation will scale as Re^3 [3]. Due to the computational time and memory requirements, CFD is mainly an "offline" method. That is, CFD simulation is run, and the result is analyzed before the actual application of fluid systems. For instance, Boeing utilizes CFD to test its airplanes [50]. The nonlinearity of NSE poses serious computational challenges to numerical simulation. Typically, implicit and iterative methods (high computational time) must be used to ensure the numerical accuracy and stability of CFD [2].

Additionally, CFD simulation of the full NSE requires very detailed and accurate information about the boundary condition, and measurement of the full flow field, which are often unavailable in real-time applications.

Therefore, for real-time applications, we desire simple, low-order, and accurate models. The model will be an approximate model to the NSE and describes the essential features of fluid flows, but is much simpler. The simplified model paves the way for efficient real-time prediction, estimation, and control [95].

1.2 Control

Optimal control deals with finding optimal control law for a dynamical system over a period of time so as to optimize an objective function [6]. It has wide applications in various fields including fluid dynamics [56], aerospace engineering [97], and even economics [87].

The mathematical theory of control is concerned with a dynamical system where control input exists. In general, the state-space representation of a (discrete-time) dynamical system [6] is

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k), \tag{1.2a}$$

$$\boldsymbol{y}_k = \boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{u}_k), \tag{1.2b}$$

where k is the time step, $\boldsymbol{x}_k$ is the state, $\boldsymbol{u}_k$ is the control (input), $\boldsymbol{y}_k$ is the observation (output). The dynamical system is called autonomous if control is absent. Notice that NSE can be cast into the above state-space form after spatial and temporal discretization. Therefore, fluid flow systems are also dynamical systems. The functions $\boldsymbol{f}$ and $\boldsymbol{g}$ together specify the dynamical system model.

In the special case where both $\boldsymbol{f}$ and $\boldsymbol{g}$ are linear and time-invariant in $\boldsymbol{x}, \boldsymbol{u}$, the dynamical system is called linear time-invariant (LTI) [6] and takes the following form:

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k, \tag{1.3a}$$

$$\boldsymbol{y}_k = \boldsymbol{C}\boldsymbol{x}_k + \boldsymbol{D}\boldsymbol{u}_k, \tag{1.3b}$$

where $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}$ are properly sized matrices. LTI system is well studied and comes with many powerful techniques. At present, nonlinear system control still remains very challenging in general. Even seemingly simple nonlinear dynamical systems

can exhibit complex behaviors such as chaos (e.g., Lorenz system [39]), limit cycle [13] (e.g., Van der Pol oscillator [39]), and ergodicity [102] (e.g., irrational rotation map [89]).

The high dimensionality of dynamical systems also poses computational challenges. For example, Kalman filter [53] and linear quadratic regulator (LQR) [6] are probably the two most widely used techniques for control. However, their computational time both scale like $\mathcal{O}(n^3)$, where n is the state dimension. In high-dimensional case, the numerical stability and accuracy also become an issue (e.g., when computing matrix inverse in Kalman filter [53]).

In summary, high dimensionality and nonlinearity also arise as challenges in applications of control theory. For efficient prediction, estimation, and control, we aim to build low-order simplified models.

1.3 Data-driven modeling

In the field of fluid dynamics and control, we both require low-order, simple, and accurate models. However, in many applications of interest, the models are unknown or too complicated to be useful. Consider a flow control problem, where we can only measure the velocity/pressure at a limited number of locations by placing sensors. Even though the NSE governs the motion of the whole flow field, we do not have an explicit model for the measured states. Now imagine that we can measure the full flow field (the discrete state dimension is in the order of millions), it is still inefficient to work with such high dimensionality. In these cases, we hope to build a low-order approximate model for the original high-dimensional systems.

At present, with more powerful computers and advancement in experimental and numerical techniques, simulating and measuring dynamical systems have become much easier. To make better use of the available data, data-driven modeling methods

are desirable. Now, to formalize the notion of data-driven modeling, assume that we have history measurement of state, control, and observation up to time step k,

$$\{\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{y}_i\}_{i=0}^{k}. \tag{1.4}$$

Depending on the application domain, one or two of these quantities may be absent. For example, for an autonomous system, there is no control. In some cases, the observation will be the same as the state. The task is to learn a low-order, simple, and accurate model that describes the essential behaviors of the observed data. Typically, we aim for a state-space form, and it amounts to learning the functions $\boldsymbol{f}, \boldsymbol{g}$.

There are a few widely used data-driven modeling methods including proper orthogonal decomposition (POD) [60], Galerkin projection (GP) [78], dynamics mode decomposition (DMD) [81], input-output analysis (also referred to as resolvent analysis) [63]. For a more comprehensive review of these methods, see [95].

1.4 Flow separation control

To motivate the research work in this thesis, we present the flow separation control problem. At a high angle of attack, the flow past an airfoil will separate, as sketched in Figure 1.1. Flow separation is typically a deleterious phenomenon because of the loss of lift and increase of drag [67]. Flow separation induces complex flow behaviors, including Kelvin-Helmholtz instability, wake shedding, flow recirculation, and separation bubble oscillations [67]. Therefore, flow separation control is an active area of research [85, 33, 68, 76].

A simplified model for airfoil separation flow is proposed as the so-called canonical separated flow [37]. A sketch of the setup is shown in Figure 1.2a. It removes the curvature of the airfoil (by using a flat plate), but retains essential separation characteristics as shown in Figure 1.2b. An adverse pressure gradient is imposed by

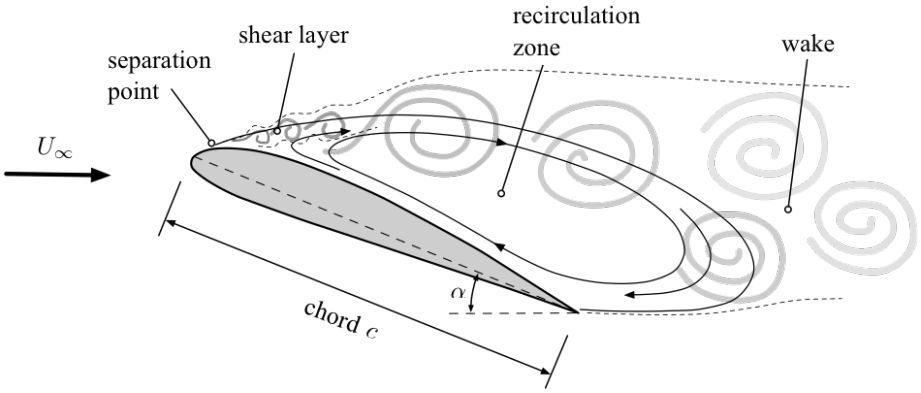
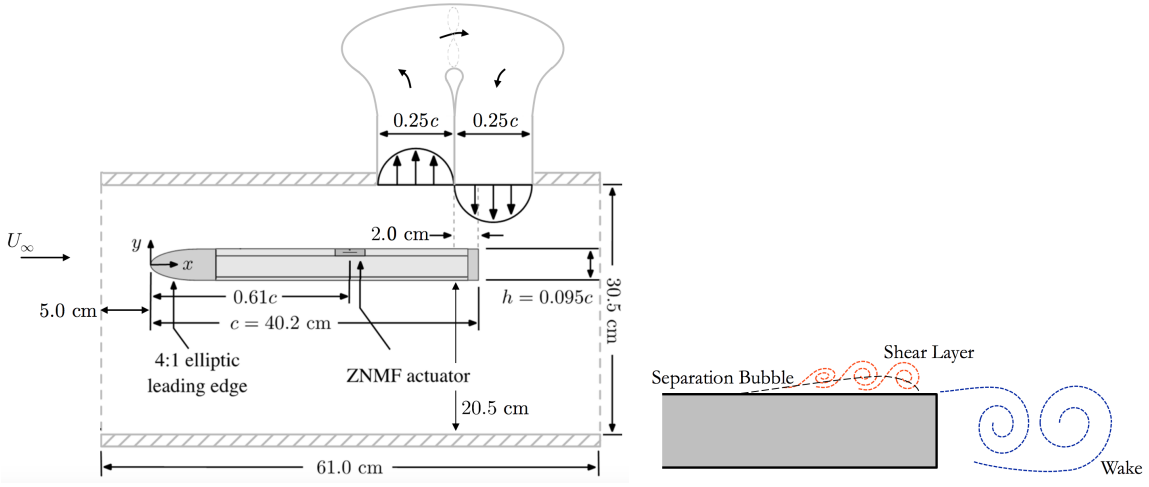


Figure 1.1: Sketch of flow separation around an airfoil. Credit to [18].

the suction fan, and the flow is re-attached by the blowing fan. More information regarding the separation system and the flat plate model can be found in [18].



(a) Flate plate model and separation system (b) Flow physics

Figure 1.2: Canonical separated flow. Images from collaborators [18].

Unsteady pressure fluctuations are measured by an array of 13 microphones within the separation region. The microphones are placed at the centerline of the plate, between $x/c = 0.70$ and 0.94 , with a spacing of $\triangle x/c = 0.02$. A zero-net mass-flux (ZNMF) [32] actuator is employed to control the separated flow, which is located at $x/c = 0.61$. For further details refer to [18].

This is a practical flow control problem. First of all, we can not measure the whole flow field. The only available information is the 13 pressure measurements and

ZNMF control history. The model is unknown because NSE is only satisfied by the full flow field. Second, we can control the ZNMF actuator upstream. We want to adjust the control signal based on the measurements, and this strategy is known as closed-loop (feedback) control [6]. There exist model-free control methods, such as reinforcement learning (RL) [93]. However, RL typically requires many iterations and experiments before learning meaningful control strategies, making it inapplicable to many real-time applications. For robustness, simplicity, and efficiency, it is desirable to construct an explicit model from data.

Chapter 2

Accuracy criterion for dynamic mode decomposition

2.1 Background

Dynamic mode decomposition (DMD) [81] is a data-driven method to decompose spatio-temporal data into spatial modes and temporal functions describing their evolution (in the form of frequency and growth/decay rate). Over the past decade, DMD has been successfully utilized for many applications in fluid dynamics [80, 98, 19, 113].

Consider a discrete-time (autonomous) dynamical system

$$\boldsymbol{x}(k+1) = \boldsymbol{F}(\boldsymbol{x}(k)), \tag{2.1}$$

where $\boldsymbol{x}(k) \in \mathbb{R}^n$ is the state and $\boldsymbol{F} : \mathbb{R}^n \to \mathbb{R}^n$ is the dynamics (map). Let $\psi_1, \psi_2, \cdots, \psi_q$ be scalar-valued functions, and $\boldsymbol{\psi} : \mathbb{R}^n \to \mathbb{R}^q$ be a vector-valued observable function whose components are $(\psi_1, \psi_2, \cdots, \psi_q)$. We consider pairs of snapshots $(\boldsymbol{x}_k, \boldsymbol{x}_k^{\#}), k = 1, 2, \cdots, m$, where $\boldsymbol{x}_k^{\#} = \boldsymbol{F}(\boldsymbol{x}_k)$ is the image of $\boldsymbol{x}_k$ under dynamics $\boldsymbol{F}$. This formulation [99] is slightly more general than the original DMD [81] which is designed for sequential data. For sequential data $\boldsymbol{x}(1), \boldsymbol{x}(2), \cdots, \boldsymbol{x}(m+1)$, it can be

handled in this new formulation by taking $\boldsymbol{x}_k^{\#} = \boldsymbol{x}_{k+1}$. Next, we form two matrices $\boldsymbol{Y} \in \mathbb{R}^{q\times m}, \boldsymbol{Y}^{\#} \in \mathbb{R}^{q\times m}$ whose columns are $\boldsymbol{y}_k = \boldsymbol{\psi}(\boldsymbol{x}_k), \boldsymbol{y}_k^{\#} = \boldsymbol{\psi}(\boldsymbol{x}_k^{\#})$ respectively. DMD aims to find a matrix $\boldsymbol{A} \in \mathbb{R}^{q\times q}$ such that

$$\boldsymbol{y}_k^{\#} = \boldsymbol{A}\boldsymbol{y}_k. \tag{2.2}$$

The optimal (least square or minimum norm) solution is [81]

$$\boldsymbol{A} = \boldsymbol{Y}_k^{\#}\boldsymbol{Y}_k^{+}. \tag{2.3}$$

More details about the DMD algorithm can be found in [99]. Denote the eigenvalues and eigenvectors of $\boldsymbol{A}$ by $(\mu_i, \boldsymbol{v}_i)_{i=1}^q$. Suppose the initial condition is $\boldsymbol{y}_1 = \sum_{i=1}^q c_i\boldsymbol{v}_i$, the prediction will be

$$\boldsymbol{y}_{k+1} = \boldsymbol{A}^k\boldsymbol{y}_1 = \sum_{i=1}^{q} c_i\mu_i^k\boldsymbol{v}_i, \tag{2.4}$$

where μ_i is the DMD eigenvalue, and $\boldsymbol{v}_i$ is the DMD mode. Therefore, each DMD mode is associated with a single frequency and growth/decay rate (DMD eigenvalue).

There are connections between DMD and the infinite-dimensional linear Koopman operator [54]. The high-level idea is that under certain conditions [99], DMD gives a finite-dimensional approximation to the Koopman operator. Koopman operator acts on scalar-valued functions of the state space. Given the dynamics 6.1, and a function space $\Phi \subseteq L^2(\mathbb{R}^n)$, the Koopman operator $\mathcal{K} : \Phi \to \Phi$ is defined as [54]

$$(\mathcal{K}\phi)(\boldsymbol{x}) = (\phi \circ \boldsymbol{F})(\boldsymbol{x}) = \phi(\boldsymbol{F}(\boldsymbol{x})), \tag{2.5}$$

where $\phi \in \Phi$. In other words, $\mathcal{K}$ maps a function ϕ to another function $\phi \circ \boldsymbol{F}$, and $(\mathcal{K}\phi)(\boldsymbol{x})$ gives the values of $\phi(\boldsymbol{x})$ at the next time step under the dynamics. We make two remarks about the Koopman operator. Frist, the Koopman operator acts on functions of the state instead of the state itself. Second, the Koopman operator

is linear even though the dynamics might be nonlinear. By studying the Koopman operator, we are looking at an infinite-dimensional linear operator instead of the original finite-dimensional nonlinear dynamics. Due to the linearity of the Koopman operator, it may have eigenvalues and eigenfunctions, which satisfy

$$\mathcal{K}\varphi = \mu\varphi, \tag{2.6}$$

where φ is the eigenfunction with eigenvalue μ. (μ, φ) is called a Koopman eigenpair. Koopman eigenfunction gives a change of coordinate in which the dynamics becomes linear. To show this, let $z_k = \varphi(\boldsymbol{x}_k)$, we have

$$z_{k+1} = \varphi(\boldsymbol{x}_{k+1}) = \varphi(\boldsymbol{F}(\boldsymbol{x}_k)) = (\mathcal{K}\varphi)(\boldsymbol{x}_k) = \mu\varphi(\boldsymbol{x}_k) = \mu z_k. \tag{2.7}$$

Assume the Koopman eigenfunction $\varphi(\boldsymbol{x})$ (with eigenvalue μ) lies in the span of the observables $\boldsymbol{\psi}_i$, i.e., there exists $\boldsymbol{w}^*$ such that $\varphi(\boldsymbol{x}) = \boldsymbol{w}^*\boldsymbol{\psi}(\boldsymbol{x})$. Under other additional conditions [99], $\boldsymbol{w}^*$ will be a left eigenvector of the DMD matrix $\boldsymbol{A}$ with eigenvalue μ (i.e., $\boldsymbol{w}^*\boldsymbol{A} = \mu\boldsymbol{w}^*$). In practice, the conditions might not be satisfied exactly or the collected data may be corrupted with noise. In this case, DMD only provides an approximation to the Koopman eigenpairs. In summary, by computing the left eigenvalues and eigenvectors $(\mu_i, \boldsymbol{w}_i^*)$ of the DMD matrix, we can get (approximate) Koopman eigenpairs $(\mu_i, \varphi_i(\boldsymbol{x}))$ where $\varphi_i(\boldsymbol{x}) = \boldsymbol{w}_i^*\boldsymbol{\psi}(\boldsymbol{x})$.

2.2 Method

As mentioned above, DMD is only an approximation to the Koopman operator. The first question we should ask is: how accurate is the DMD approximated eigenpairs? We should answer this fundamental question before interpreting DMD results or constructing models. Indeed, it is very tempting to change the coordinate using Koopman

eigenfunctions to make the dynamics linear. However, we must assess the quality of DMD results in advance. In most applications, the true eigenpairs are unknown.

Thanks to the connection between DMD and the Koopman operator, we can evaluate the accuracy of DMD eigenvalue and DMD mode by looking at the accuracy of its corresponding Koopman eigenpair. To be specific, given a DMD approximated eigenpair (μ, φ) (using training data) where $\varphi(\boldsymbol{x}) = \boldsymbol{w}^* \boldsymbol{\psi}(\boldsymbol{x})$, and we want to assess its accuracy. The true eigenpair satisfies the relationship $\varphi \circ \boldsymbol{F} = \mu\varphi$. For a DMD approximated eigenpair, ideally we should compute

$$\frac{\|\varphi \circ \boldsymbol{F} - \mu\varphi\|}{\|\varphi\|}, \tag{2.8}$$

where $\|\cdot\|$ is the norm of a function. However, evaluation of this quantity requires explicit knowledge of $\boldsymbol{F}$, which is unknown in most cases of interest. To bypass this problem, we rely on the collected data. The sampled snapshot pairs are split into training data and testing data. Training data is used in DMD to approximate the Koopman eigenpairs, and testing data will be used to estimate the above quantity. Particularly, we define the following accuracy criterion

$$\alpha = \frac{\sum_k |\varphi(\boldsymbol{x}_k^\#) - \mu\varphi(\boldsymbol{x}_k)|}{\sum_k |\varphi(\boldsymbol{x}_k)|}, \tag{2.9}$$

where $\varphi(\boldsymbol{x}) = \boldsymbol{w}^* \boldsymbol{\psi}(\boldsymbol{x})$, and the summation is over the testing data. A diagram summarizing how this accuracy criterion may be applied is shown Figure 2.1.

The numerator measures to what extent the eigenfunction equation holds, and the denominator gives a measure of the magnitude of the eigenfunction. Here α can be interpreted as the error of a (DMD approximated) Koopman eigenpair. Notice that each eigenpair can be assessed independently. Therefore it makes sense to call α the mode error. Also, it is purely data-driven and does not require any knowledge

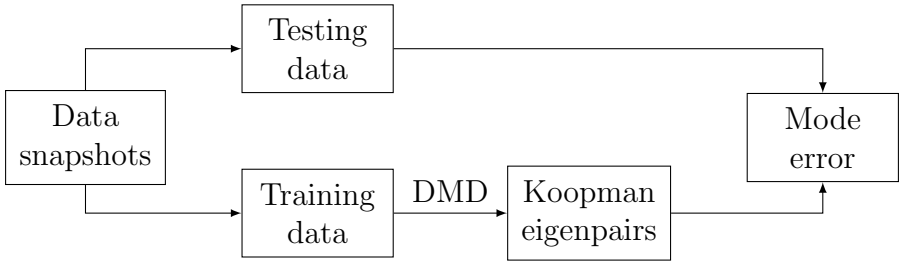


Figure 2.1: A diagram summarizing the implementation of the accuracy criterion. Training data is used to approximate Koopman eigenpairs with variants of DMD, while testing data is used to evaluate the quality of Koopman eigenpairs. Reproduced from Figure 6.1.

of the dynamics. Smaller α indicates higher accuracy and $\alpha = 0$ for a true eigenpair. Typically, we only care about the eigenpairs with $0 \leq \alpha \ll 1$.

2.3 Results

First of all, the accuracy criterion is validated with a dynamical system where the true Koopman eigenpairs are known. Consider a 2D nonlinear map (also considered in [99]) with dynamics defined by

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \mapsto \begin{bmatrix} \gamma x_1 \\ \delta x_2 + (\gamma^2 - \delta) x_1^2 \end{bmatrix}, \tag{2.10}$$

where $\gamma = 0.9, \delta = 0.8$. It is straightforward to verify that γ, δ are Koopman eigenvalues with respective eigenfunctions $\varphi_\gamma(\boldsymbol{x}) = x_1, \varphi_\delta(\boldsymbol{x}) = x_2 - x_1^2$. Additional Koopman eigenvalues and eigenfunctions are given by $\mu_{k,\ell} = \gamma^k \delta^\ell, \varphi_{k,\ell} = \varphi_\gamma^k \varphi_\delta^\ell$, where $k, \ell = 0, 1, 2, \cdots$ are non-negative integers.

In the case where the observable $\boldsymbol{\psi}(\boldsymbol{x})$ is not identity, the DMD algorithm becomes the extended DMD (EDMD) algorithm [104]. Proper data is collected, and observable functions are taken to be monomials up to degree 5 (e.g., $x_1, x_2, x_1^2, x_2^2, x_1 x_2, \cdots$). The accuracy criterion α is compared with true eigenvalue error τ and true eigen-

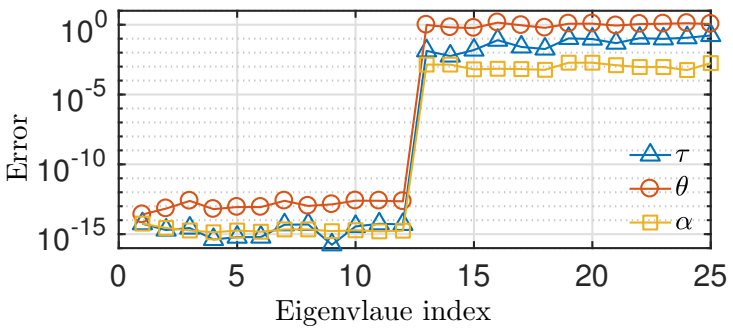


Figure 2.2: Comparison between the accuracy criterion α, eigenvalue error τ, and eigenfunction error θ. The eigenvalues are indexed by their absolute value, in descending order. Reproduced from Figure 6.2.

function error θ, which are defined by comparing with the true Koopman eigenpairs. In particular, they are defined as

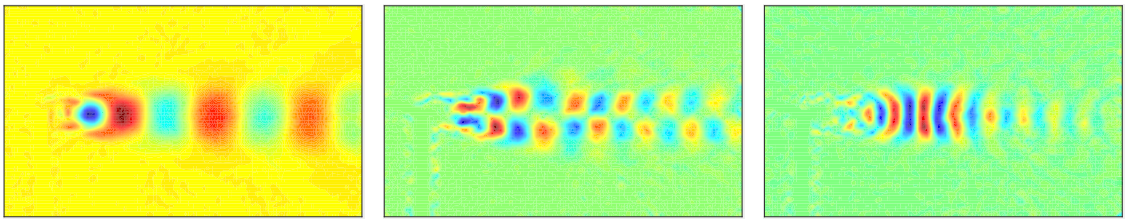
$$\tau = \frac{|\mu - \mu_{\text{true}}|}{|\mu_{\text{true}}|}, \quad \theta = \frac{\|\varphi - \varphi_{\text{true}}\|}{\|\varphi_{\text{true}}\|}. \tag{2.11}$$

The result is shown in Figure 2.2. Obviously, the accuracy criterion highly correlates with the true error. Here the errors are many orders of magnitude apart (almost 15 orders of magnitude), but the accuracy criterion is also effective for cases where the separation of scale is smaller. Keep in mind that the accuracy criterion does not assume any knowledge about the true dynamics, and it is purely data-driven. It is capable of indicating the accuracy very well.

If the the observable function $\boldsymbol{\psi}(\boldsymbol{x})$ is implicitly defined by kernel function, the algorithm is called the kernel DMD (KDMD) [105]. In KDMD, EDMD is reformulated such that only the inner products of observables need to be computed. A kernel function $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is defined as [11]

$$k(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \langle \boldsymbol{\psi}(\boldsymbol{x}), \boldsymbol{\psi}(\hat{\boldsymbol{x}}) \rangle. \tag{2.12}$$

Kernel function allows efficient computation of inner product between high-dimensional (even infinite-dimensional) observables . As an example, consider a



(a) Mode 1, $f_1 = 0.90$ Hz (b) Mode 2, $f_2 = 1.77$ Hz (c) Mode 3, $f_3 = 2.69$ Hz

Figure 2.3: Three dominant DMD modes (real part) picked out by accuracy criterion. Reproduced from Figure 6.7.

polynomial kernel $k(\boldsymbol{x}, \hat{\boldsymbol{x}}) = (1 + \boldsymbol{x}^T \hat{\boldsymbol{x}})^d$. This kernel corresponds to a set of observables $\boldsymbol{\psi}(\boldsymbol{x})$ consisting of all monomials in components of $\boldsymbol{x}$ up to degree d [11]. In particular, taking $n = 2$ and $d = 2$, this kernel function expands to

$$\begin{aligned}(1 + \boldsymbol{x}^T \hat{\boldsymbol{x}})^2 &= 1 + 2x_1\hat{x}_1 + 2x_2\hat{x}_2 + 2x_1^2\hat{x}_1^2 + 2x_1x_2\hat{x}_1\hat{x}_2 + \hat{x}_1^2\hat{x}_2^2 \\ &= \langle \boldsymbol{\psi}(\boldsymbol{x}), \boldsymbol{\psi}(\hat{\boldsymbol{x}}) \rangle,\end{aligned} \tag{2.13}$$

where $\boldsymbol{\psi}(\boldsymbol{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2)$.

The proposed accuracy criterion is shown to be capable of informing the choice of right kernel among various kernel functions. For the considered dynamical system, the polynomial kernel performs best in identifying the correct Koopman eigenpairs, because its implicit observables span the true Koopman eigenfunctions (both are polynomials). It is also robust when the data is corrupted with noise, because the (implicit) observable function is finite-dimensional, making it less likely to overfit in the presence of noise.

Finally, DMD and its variants are applied to experimental data from flow past a cylinder and separated flow past a flat plate. In both cases, the accuracy criterion picks out the important modes that are responsible for the dominant physics in the flows. For example, the identified dominant modes for flow past a cylinder are shown in Figure 2.3. These modes agree with the results in previous work [98].

Chapter 3

Online dynamic mode decomposition

3.1 Background

From now on, we view DMD as a system identification method. To make it clear, we need a slightly different formulation. Consider a dynamical system

$$\boldsymbol{x}_{j+1} = \boldsymbol{f}(\boldsymbol{x}_j, \boldsymbol{u}_j), \tag{3.1}$$

where $\boldsymbol{x}_j \in \mathbb{R}^n, \boldsymbol{u}_j \in \mathbb{R}^p$ are the state and control respectively. Further suppose that we have measurement history $(\boldsymbol{x}_j, \boldsymbol{u}_j)_{j=1}^k$. To learn a model from data, we assume the model can be written in the following form

$$\boldsymbol{x}_{j+1} = \boldsymbol{M}\boldsymbol{\phi}(\boldsymbol{x}_j, \boldsymbol{u}_j), \tag{3.2}$$

where $\boldsymbol{\phi} : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^q$ is the nonlinearity, and must be specified by the users. $\boldsymbol{M} \in \mathbb{R}^{n \times q}$ is unknown, and will be learned from data. For example, the most widely

used form for $\boldsymbol{\phi}$ is the linear form:

$$\boldsymbol{\phi}(\boldsymbol{x}_j, \boldsymbol{u}_j) = \begin{bmatrix} \boldsymbol{x}_j \\ \boldsymbol{u}_j \end{bmatrix}. \tag{3.3}$$

In this case, it amounts to finding a linear-time invariant (LTI) system. In general, $\boldsymbol{f}$ is unknown, so the nonlinearity $\boldsymbol{\phi}$ might not agree with the true nonlinearity in $\boldsymbol{f}$. In fluid applications, second order nonlinearity is widely used for $\boldsymbol{\phi}$ because of the nonlinearity in Navier-Stokes equations. The best model (a matrix) $\boldsymbol{M} \in \mathbb{R}^{n \times q}$ that fits the data can be found by minimizing the cost function,

$$\sum_{j=1}^{k} \|\boldsymbol{x}_{j+1} - \boldsymbol{M}\boldsymbol{\phi}(\boldsymbol{x}_j, \boldsymbol{u}_j)\|^2 = \|\boldsymbol{M}\boldsymbol{X} - \boldsymbol{Y}\|_F^2, \tag{3.4}$$

where $\boldsymbol{X} \in \mathbb{R}^{q \times q}, \boldsymbol{Y} \in \mathbb{R}^{n \times q}$ are two matrices where the columns are $\boldsymbol{\phi}(\boldsymbol{x}_j, \boldsymbol{u}_j), \boldsymbol{x}_{j+1}$ respectively. $\|\cdot\|$ is the Euclidean on vectors and $\|\cdot\|_F$ is the Frobenius norm on matrices. Assuming $\boldsymbol{X}$ has full row rank, the least squares solution is

$$\boldsymbol{M} = \boldsymbol{Y}\boldsymbol{X}^{+}. \tag{3.5}$$

In this formulation, DMD computes a model that can be used for prediction, estimation and control.

The above DMD algorithm is an "offline" algorithm, i.e., it computes the DMD matrix only once given all the data. However, in real-time applications, data usually come in a stream. If we want to control a fluid system in real-time such as the canonical separated flow, the measurements will be a data stream. Intuitively, the model should be updated so that it better tracks the true dynamics. Therefore, we desire an "online" algorithm that allows us to update the model in real-time.

There are a few reasons why we should update the model. First, the dynamical system might be time-varying. In this case, the model should be adaptive in order to account for time-varying behaviors. Second, the dynamics may be nonlinear. If we use a linear form for $\boldsymbol{\phi}(\boldsymbol{x}_j, \boldsymbol{u}_j)$, the learned model is only valid locally (by Taylor expansion). Thus, the model should be updated so that it remains a valid approximation. Third, the nonlinearity of the dynamics is typically unknown, so $\boldsymbol{\phi}(\boldsymbol{x}_j, \boldsymbol{u}_j)$ might not agree with the true nonlinearity. Without any prior knowledge about the form of nonlinearity, we can simply use a linear form. Linear models will be local approximations as long as it is continuously updated.

It is worthwhile to point out that if the true dynamics is time-invariant and the nonlinearity used in $\boldsymbol{\phi}(\boldsymbol{x}_j, \boldsymbol{u}_j)$ agrees with $\boldsymbol{f}(\boldsymbol{x}_j, \boldsymbol{u}_j)$, then updating the model using new data does not change the model at all.

3.2 Method

To distinguish the models across time steps, we will index the relevant matrices by time. Given $(\boldsymbol{x}_j, \boldsymbol{u}_j)_{j=1}^k$, we form matrices $\boldsymbol{X}_k, \boldsymbol{Y}_k$ as before, then the model is $\boldsymbol{M}_k = \boldsymbol{Y}_k \boldsymbol{X}_k^+$. Now, at time step $k+1$, we have a new observation $\boldsymbol{x}_{k+1}$. The question is: how should $\boldsymbol{M}_k$ be updated? First, we rewrite the online model as

$$\boldsymbol{M}_k = \boldsymbol{Q}_k \boldsymbol{P}_k^{-1}, \quad \boldsymbol{Q}_k = \boldsymbol{Y}_k \boldsymbol{X}_k^T, \quad \boldsymbol{P}_k = (\boldsymbol{X}_k \boldsymbol{X}_k^T)^{-1}, \tag{3.6}$$

where $\boldsymbol{Q}_k \in \mathbb{R}^{n \times q}, \boldsymbol{P}_k \in \mathbb{R}^{q \times q}$. At time step $k+1$, we have

$$\boldsymbol{X}_{k+1} = \begin{bmatrix} \boldsymbol{X}_k & \boldsymbol{\phi}_k \end{bmatrix}, \quad \boldsymbol{Y}_{k+1} = \begin{bmatrix} \boldsymbol{Y}_k & \boldsymbol{x}_{k+1} \end{bmatrix}, \tag{3.7}$$

where $\boldsymbol{\phi}_k$ is a shorthand for $\boldsymbol{\phi}(\boldsymbol{x}_k, \boldsymbol{u}_k)$. Therefore,

$$\boldsymbol{Q}_{k+1} = \boldsymbol{Y}_{k+1}\boldsymbol{X}_{k+1}^T = \boldsymbol{Q}_k + \boldsymbol{x}_{k+1}\boldsymbol{\phi}_k^T, \tag{3.8a}$$

$$\boldsymbol{P}_{k+1} = (\boldsymbol{X}_{k+1}\boldsymbol{X}_{k+1}^T)^{-1} = (\boldsymbol{P}_k^{-1} + \boldsymbol{\phi}_k\boldsymbol{\phi}_k^T)^{-1}. \tag{3.8b}$$

The data matrices are related by a rank-one update, and this is the most important observation. Applying Sherman–Morrison formula [88, 41] to $\boldsymbol{P}_{k+1}$, and after some algebraic manipulation, we finally obtain

$$\boldsymbol{P}_{k+1} = \boldsymbol{P}_k - \gamma_{k+1}\boldsymbol{P}_k\boldsymbol{\phi}_k\boldsymbol{\phi}_k^T\boldsymbol{P}_k, \tag{3.9a}$$

$$\boldsymbol{M}_{k+1} = \boldsymbol{M}_k + \gamma_{k+1}(\boldsymbol{x}_{k+1} - \boldsymbol{M}_k\boldsymbol{\phi}_k)\boldsymbol{\phi}_k^T\boldsymbol{P}_k, \tag{3.9b}$$

$$\gamma_{k+1} = \frac{1}{1 + \boldsymbol{\phi}_k^T\boldsymbol{P}_k\boldsymbol{\phi}_k}. \tag{3.9c}$$

Notice that the update to the model $\boldsymbol{M}_k$ is proportional to the prediction error of current model $(\boldsymbol{x}_{k+1} - \boldsymbol{M}_k\boldsymbol{\phi}_k)$, which makes intuitive sense.

In summary, the algorithm works as follows. First, collect snapshots $(\boldsymbol{x}_j, \boldsymbol{u}_j)_{j=1}^k$. Second, compute $\boldsymbol{M}_k, \boldsymbol{P}_k$ by definition (3.6). Third, when a new snapshot pair $(\boldsymbol{\phi}_k, \boldsymbol{x}_{k+1})$ becomes available, update $\boldsymbol{M}_k, \boldsymbol{P}_k$ according to (3.9). Implementations of this algorithm in both Matlab and Python are publicly available at [112].

There are two important extensions to the above algorithm. They are both designed to discount old snapshots. First, an exponential weighting factor can be used to place less weight on old data (weighted online DMD). The cost function will be

$$\sum_{j=1}^{k} \rho^{k-i}\|\boldsymbol{x}_{j+1} - \boldsymbol{M}_k\boldsymbol{\phi}_j\|^2, \qquad 0 < \rho \leq 1,$$

where ρ is the weighting factor. Smaller ρ will result in faster tracking of the dynamics. Second, we can use a sliding window to only use the most recent snapshots (window DMD). Consider recent w snapshot pairs $(\boldsymbol{\phi}_j, \boldsymbol{x}_{j+1})_{j=k-w+1}^k$, and the resulting cost

Aspect	Standard DMD	Online DMD
Computational time	$\mathcal{O}(kq(n+q))$	$\mathcal{O}(q(n+q))$
Memory requirements	$\mathcal{O}(kq(n+q))$	$\mathcal{O}(q(n+q))$

Table 3.1: Computational time and memory requirements of online DMD compared with standard DMD. k is current time step, n is state dimension, q is nonlinearity dimension, $k \gg \max(n, q)$.

function is

$$\sum_{j=k-w+1}^{k} \|\boldsymbol{x}_{j+1} - \boldsymbol{M}_k \boldsymbol{\phi}_j\|^2.$$

The above online algorithms can be modified for these two extensions.

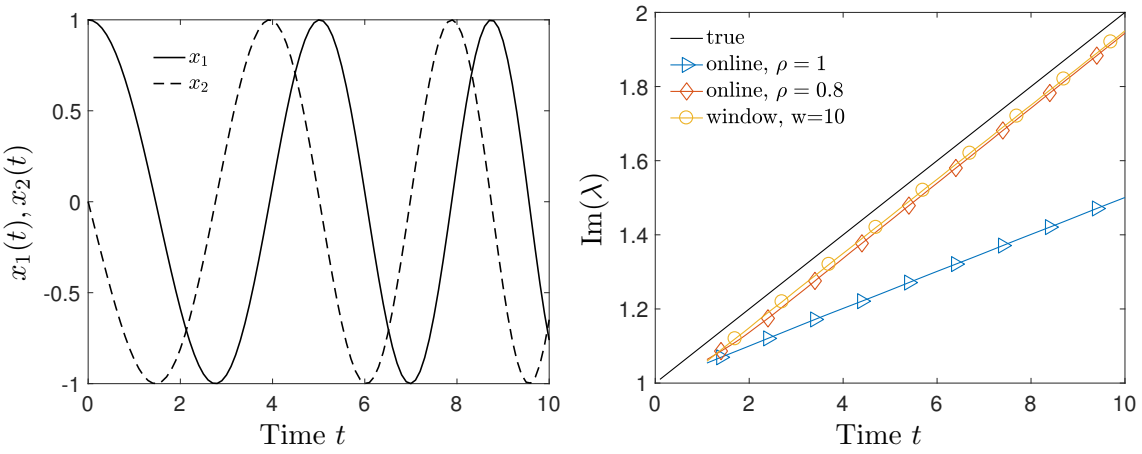
The model form (3.2) is very general and encompasses many familiar dynamical system models. As we have mentioned, LTI system can be represented by taking linear form (3.3) for $\boldsymbol{\phi}(\boldsymbol{x}_j, \boldsymbol{u}_j)$. The well-known autoregressive (AR) model [103], which resembles the Takens' delay embedding for dynamical systems [96, 73], can also be cast into form (3.2). Additionally, many classical nonlinear systems (e.g., logistics map, Duffing oscillator, Lorenz attractor, and Van der Pol oscillator) [39] can be written in this form.

3.3 Results

The online algorithms are superior to existing standard methods in terms of both computational time and memory requirements, as summarized in Table 3.1. Both the computational time and memory requirements are improved from cubic to quadratic. Furthermore, they do not depend on the time step k anymore ($k \gg \max(n, q)$), which essentially goes to infinity in real-time applications.

Consider a 2D time-varying system

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}(t)\boldsymbol{x}(t), \tag{3.10a}$$



(a) State evolution (b) Frequency prediction

Figure 3.1: Solution of the linear time-varying system (3.10), and frequencies predicted by various DMD algorithms. Reproduced from Figure 7.4.

where $\boldsymbol{x}(t) \in \mathbb{R}^2$, and the time-varying matrix $\boldsymbol{A}(t)$ is given by

$$\boldsymbol{A}(t) = \begin{bmatrix} 0 & \omega(t) \\ -\omega(t) & 0 \end{bmatrix}, \tag{3.10b}$$

where $\omega(t) = 1 + \epsilon t$. We set $\epsilon = 0.1$, so that the system is slowly varying in time. The eigenvalues of $\boldsymbol{A}(t)$ are purely imaginary $\pm i\omega(t)$. The system is simulated for $t \in [0, 10]$ starting from initial condition $(1, 0)$, and the snapshots are taken with time step $\triangle t = 0.1$. We apply online DMD algorithm (and its variants) to fit the model and compute the resulting frequency. The result is presented in Figure 3.1. Both window DMD and (weighted) online DMD are able to track the time-varying frequencies. For online DMD, smaller values of the parameter ρ result in faster tracking of the time-varying frequency.

Additionally, the online algorithms have been shown to successfully capture the time-varying dynamics in the pressure fluctuation in the canonical separated flow. In fact, the online DMD algorithm has been applied to this flow separation control problem [18] for system identification and real-time control. The high-level idea is to

learn an adaptive linear model with online DMD, and apply LQR control based on the estimated model.

Chapter 4

Input-output analysis of separated flow

4.1 Background

As mentioned in section 1.4, flow separation is usually an undesirable behavior due to lift decreasing and drag increasing [67]. There have been some studies [85, 33, 68, 76] trying to find the optimal ZNMF actuator parameter by systematic parameter search. However, these approaches suffer from high experimental or simulation costs. They only provide limited insight into the relevant physics in the problem.

To better understand the physics, and gain more insights about optimal actuation, we investigate the problem from the perspective of input-output response. The input-output analysis was originally proposed to study the response of a flow field to disturbances [51, 63]. Later, it is also referred to as the resolvent analysis, and has been widely used in turbulence modeling and building reduced-order models [59, 30, 49, 35]. For a review of various applications, refer to [95].

The input-output analysis utilizes the transfer function (a linear operator) from the input forcing (including nonlinear advective forcing, and any other external forc-

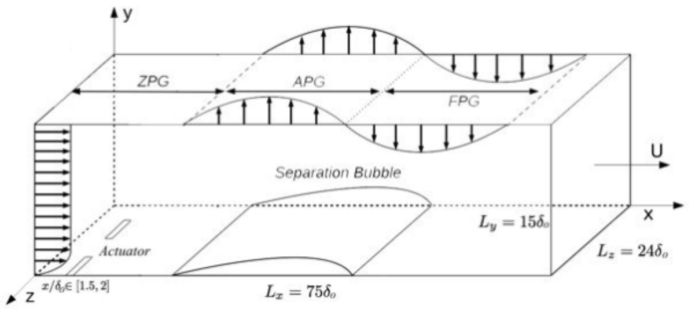
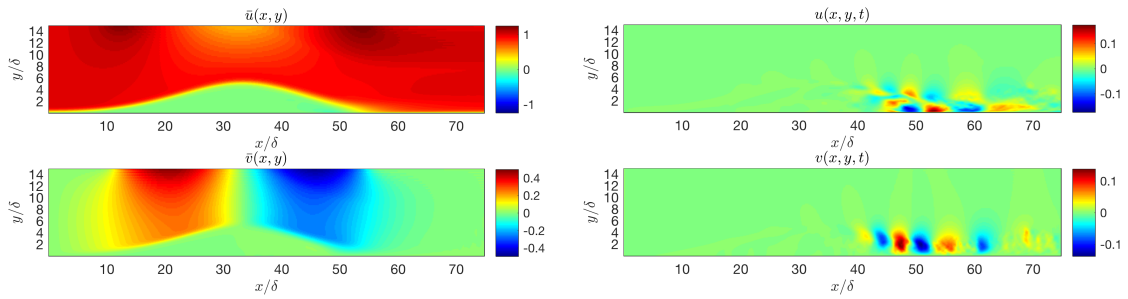


Figure 4.1: Sketch of separated flow on a flat plate (not to scale). Credit to [107]. Reproduced from Figure 8.1.

ing) to the output response (velocity and pressure field). This operator is often low rank (sometimes approximately rank-one). From it, one can find the optimal forcing mode that actuates the optimal response that has maximum amplification. The optimal forcing mode provides clues about where is the most efficient place to actuate. For example, it has guided the design of airfoil separation control [109], where localized unsteady thermal actuation is used.

We consider a laminar boundary layer with a separation bubble along a flat plate. The pressure gradient is imposed by suction and blowing on the top boundary, as shown in Figure 4.1. Three-dimensional numerical simulation is performed, and for more details, refer to [107]. The physical quantities are nondimensionalized by the free-stream velocity $U_{\infty,0}$, and the boundary thickness δ_0 of the Blasius velocity profile at the inlet. The Reynolds number is $Re = U_{\infty,0}\delta_0/\nu = 1000$, where ν is the kinematic viscosity.

A simulation with no forcing is performed, and the result is visualized in Figure 4.2. The flow is mainly two-dimensional, along with small spanwise fluctuations. The separation bubble spans from $x = 10\delta$ to $x = 55\delta$. The fluctuation field is mainly concentrated in and behind the trailing edge of the separation bubble region. The mean flow will be taken as the base flow in the input-output analysis, and we then study the response of the fluctuation field to external forcing.



(a) (b)

Figure 4.2: (a) Time and spanwise averaged field. (b) Spanwise averaged fluctuation field. Reproduced from Figure 8.2.

4.2 Method

To introduce the input-output analysis, we start with the incompressible Navier-Stokes equations

$$\partial_t \tilde{\boldsymbol{u}} = -\tilde{\boldsymbol{u}} \cdot \nabla \tilde{\boldsymbol{u}} - \nabla \tilde{p} + \frac{1}{Re} \nabla^2 \tilde{\boldsymbol{u}}, \tag{4.1a}$$

$$\nabla \cdot \tilde{\boldsymbol{u}} = \boldsymbol{0}. \tag{4.1b}$$

The full flow field $(\tilde{\boldsymbol{u}}, \tilde{p})$ is decomposed into a base flow $(\bar{\boldsymbol{u}}, \bar{p})$ (usually mean flow or steady solution of the Navier-Stokes equations), and a fluctuation field $(\boldsymbol{u}, p)$, and we derive an equation for the fluctuations. Let $\tilde{\boldsymbol{u}} = \bar{\boldsymbol{u}} + \boldsymbol{u}, \tilde{p} = \bar{p} + p$ and substitute into (4.1), we have

$$\partial_t \boldsymbol{u} = L\boldsymbol{u} - \nabla p + \boldsymbol{f}, \tag{4.2a}$$

$$\nabla \cdot \boldsymbol{u} = \boldsymbol{0}, \tag{4.2b}$$

where

$$Lu = -\bar{\boldsymbol{u}} \cdot \nabla \boldsymbol{u} - \boldsymbol{u} \cdot \nabla \bar{\boldsymbol{u}} + \frac{1}{Re} \nabla^2 \boldsymbol{u},$$

and L is called the linearized Navier-Stokes operator. The forcing term $\boldsymbol{f}$ includes all additional terms, including the nonlinear advective term and any external forcing terms such as body forcing and boundary forcing.

Treating the nonlinear advective term and external forcing as input and velocity field as output, we can derive a state-space representation for the fluctuation field. If we define

$$q = \begin{bmatrix} \boldsymbol{u} \\ p \end{bmatrix}, \quad A = \begin{bmatrix} L & -\nabla \\ \nabla\cdot & 0 \end{bmatrix}, \quad M = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 \end{bmatrix},$$

then (4.2) can be put into state-space form

$$M\partial_t q = Aq + B\boldsymbol{f}, \tag{4.3a}$$

$$\boldsymbol{u} = Cq, \tag{4.3b}$$

where $\boldsymbol{f}$ is the input and $\boldsymbol{u}$ is the output.

If we consider sinusoidal forcing $\boldsymbol{f} = \hat{\boldsymbol{f}}e^{i\omega t}$, then the state and output will be sinusoidal because the system is linear. Let $\boldsymbol{u} = \hat{\boldsymbol{u}}e^{i\omega t}$, and use (4.3), we have

$$\hat{\boldsymbol{u}} = H(i\omega)\hat{\boldsymbol{f}},$$

where

$$H(i\omega) = C(i\omega M - A)^{-1}B$$

is the transfer function from input ($\boldsymbol{f} = \hat{\boldsymbol{f}}e^{i\omega t}$) to output ($\boldsymbol{u} = \hat{\boldsymbol{u}}e^{i\omega t}$). This transfer function is a linear operator, and is unique for each frequency ω.

It can be shown (see, e.g., [10]) that the optimal forcing mode that induces maximum amplification in the responese can be found from singular value decomposition (SVD) of the transfer function. Let the SVD of the transfer function be

$$H = \sum_j \sigma_j \psi_j \phi_j^*,$$

where $\psi_i^*\psi_j = \delta_{ij}, \phi_i^*\phi_j = \delta_{ij}$, and $\sigma_j \geq 0$ is in descending order. Then the optimal forcing is the first right singular vector ϕ_1 and the optimal response is the first left singular vector ψ_1. The largest singular value σ_1 gives the corresponding amplification. For an arbitrary forcing $\hat{\boldsymbol{f}}$, the response $\hat{\boldsymbol{u}}$ can be written as

$$\hat{\boldsymbol{u}} = \sum_j \sigma_j \psi_j (\phi_j^* \hat{\boldsymbol{f}}),$$

where $\phi_j^*\hat{\boldsymbol{f}}$ is the projection of $\hat{\boldsymbol{f}}$ in the direction of ϕ_j. For many shear flows of practical interest [63, 59, 30, 10] the linear operator H can be closely approximated by an operator of rank one: $\sigma_1 \gg \sigma_{j\geq 2}$. A rank-one approximation of the response is then

$$\hat{\boldsymbol{u}} \approx \sigma_1 \psi_1 (\phi_1^* \hat{\boldsymbol{f}}), \tag{4.4}$$

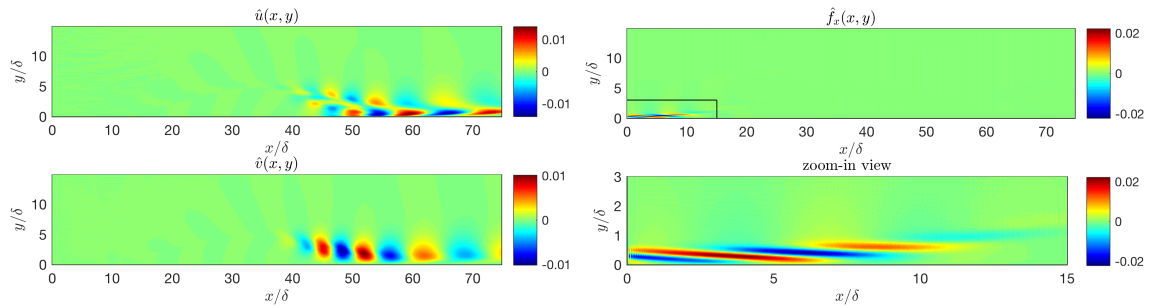
and one expects this to be a close approximation of the response for a typical forcing $\hat{\boldsymbol{f}}$ (i.e., as long as the direction of forcing is not such that $\phi_1^*\hat{\boldsymbol{f}}$ is small).

The spanwise and time-averaged flow is taken as the base flow in input-output analysis. After spatial discretization, the transfer function $H(i\omega)$ is a huge matrix of size 0.26 million by 0.26 million. Therefore, the efficient randomized SVD method of [42] is used to obtain the approximate leading singular values and singular vectors.

4.3 Results

Assuming sinusoidal forcing in both time and the spanwise direction, the output will also be sinusoidal (in time and the spanwise direction), because the system is linear. In particular, let

$$\boldsymbol{f}(x,y,z,t) = \hat{\boldsymbol{f}}(x,y)e^{i(k_z z+\omega t)}, \quad \boldsymbol{u}(x,y,z,t) = \hat{\boldsymbol{u}}(x,y)e^{i(k_z z+\omega t)},$$



(a) (b)

Figure 4.3: Global forcing result. (a) Streamwise and wall-normal optimal response mode $\hat{u}, \hat{v}$. $k_z = 0, \omega = 0.377$. (b) Streamwise optimal forcing mode $\hat{f}_x$ and zoom-in view in the region $(x/\delta, y/\delta) \in [0, 15] \times [0, 3]$. Reproduced from Figure 8.7.

and we consider only two-dimensional forcing in this study, i.e., $k_z = 0$.

It is verified that the transfer function is approximately rank-one for a wide range of frequencies ω. The first singular value is six orders of magnitude larger than the second and the rest. We set $\omega_p = 2\pi f_p = 0.377$, where $f_p = 0.06$ is the peak from discrete Fourier transform of the flow.

The optimal response mode and optimal forcing mode are shown in Fig. 4.3. First, we look at the response mode. The energy in the optimal response mode is mainly concentrated in the streamwise and wall-normal component $(\hat{u}, \hat{v})$. The response is primarily in and behind the trailing edge of the separation bubble, from $x = 40\delta$ to $x = 70\delta$. Therefore, the separation bubble is receptive to disturbances.

As for the optimal forcing mode, we find that $\hat{f}_y$ is much smaller than $\hat{f}_x$, and $\hat{f}_z$ is nine orders of magnitude smaller than $\hat{f}_x$. Therefore, only the streamwise component $\hat{f}_x$ of the forcing is shown. There are three interesting observations. First, the optimal forcing mode reveals that streamwise body force disturbance is much more important. However, the ZNMF actuator produces disturbances mainly in the wall-normal direction. This result implies that a more effective actuator should instead introduce streamwise disturbances, which are much more efficient at exciting a response in the separation bubble.

Second, the optimal forcing energy is distributed along the wall, upstream of the separation bubble. Upstream disturbances to the flow travel downstream and produce a response in the separation bubble. This observation confirms that an actuator should be placed upstream, as in previous studies [67, 76].

Third, the streamwise optimal forcing alternates between positive and negative values in the upstream region. According to (4.4), if the forcing is applied in both the positive region and negative region, their responses will be out of phase and cancel each other. Typically, the location of a ZNMF actuator (which resembles a local body force) is not well-tuned due to a lack of physical insight. The optimal forcing mode suggests that the body force actuator should be carefully placed in order to avoid cancellation in the response.

Chapter 5

Conclusion

5.1 Summary

Data-driven modeling is a new paradigm for understanding and controlling dynamical systems. High dimensionality and nonlinearity pose serious challenges to many applications in fluid dynamics and control. For efficient prediction, estimation, and control, we desire simple, low-order, and accurate models. In some sense, this thesis is motivated by the flow separation control problem. The flow physics is complicated, but we still need to control it in real-time. As a result, we must build simplified models. This is a perfect use for data-driven modeling.

The contribution of this thesis consists of three parts. First, we proposed an accuracy criterion to evaluate the accuracy of DMD results. This criterion assists model selection to better represent the dynamics in separated flow. It is purely data-driven and can be applied to other variants of DMD. It quantifies the accuracy of DMD results in the sense of approximating the Koopman eigenpairs.

Second, fast algorithms for online model learning are proposed. These algorithms have been successfully applied for learning adaptive models in the canonical separated flow [18]. These models allow for efficient real-time prediction and control. The

algorithms are orders of magnitude more efficient than existing standard algorithms, when the state is not compressed by projection. Both the computational time and memory requirements are significantly reduced. Additionally, the algorithms can be modified to discount old data, which allows for faster tracking of the dynamics.

Third, we study the input-output response of separated flow past a flat plate to the external body forcing. It is confirmed that the optimal actuator placement is upstream of the separation location, as used in previous work [67, 76]. Also, the result suggests that the separation region is most receptive to streamwise body force (control). Finally, the control should be properly placed to avoid cancellation effects in the response. Therefore, this study provides a guide for control design.

5.2 Outlook

There have been fruitful advances in data-driven modeling methods for fluid dynamics and control in recent decades [95]. However, plenty of interesting open questions are still waiting to be answered. In this spirit, we outline a few directions for future research. First of all, we discuss some questions that are tightly related to this thesis.

It is unclear how to select a subset of Koopman eigenpairs such that the original (nonlinear) system is accurately approximated. The proposed accuracy criterion sheds light on the problem by quantifying the reliability of Koopman eigenpairs. However, how to select the most dynamically important Koopman eigenpairs remains an open question. Unlike proper orthogonal decomposition (POD) [60], where the modes are orthogonal by construction, Koopman eigenfunctions are generally not orthogonal. Therefore, mode amplitudes (projection coefficients of data onto DMD modes) are not necessarily a meaningful criterion for evaluating importance. An possible method is choosing a sparse set of Koopman eigenfunctions that best predict long term trajectory.

The proposed online model learning algorithm works for weakly nonlinear and slowly time-varying systems. The approach is based on the Taylor expansion: the online model is only valid locally, and the region of validity shrinks when the true dynamics becomes more complicated. It also relies on the fast sampling of the snapshots (fast compared with the characteristic time of dynamics variations). Quantifying the uncertainty and accuracy of the learned model as a function of data quality (e.g., amount and type of noise), dynamics nonlinearity, and the sampling rate is a meaningful research direction. In general, all data-driven modeling methods depend on data, and the quality of the models needs to be characterized.

The online model learning algorithm assumes a very general model form, which encompasses many familiar models. The model can take an arbitrary form of nonlinearity for the state and control, but must be linear in the unknown parameters. Examples include LTI system, polynomial nonlinear models, autoregressive (AR) model [103] (and vector AR), and Takens' delay embedding for dynamical systems [96, 73]. It is also capable of learning static relationships, e.g., nonlinear regression. In brief, it applies to any problem where a function of a particular form needs to be learned from data. The application of this algorithm to other problems is an interesting research direction.

The input-output analysis provides the optimal forcing and optimal response mode corresponding to the maximum amplification. However, the most amplified direction is not necessarily the most physically preferable (e.g., in the sense of control). For example, an unstable system will not be stabilized by amplifying the output response. How to make use of the input-output transfer function to build low-order models for flow prediction, analysis, and control is still an open question.

Next, we suggest several general directions for the development of data-driven modeling methods. These questions may draw attention from the research community in the next decade.

Most existing data-driven modeling methods do not make use of first principles from physics (e.g., Newton's laws of motion). For example, DMD is purely data-driven and has been employed for many fluid dynamics applications. Despite the fact that the Navier-Stokes equations govern the motion of fluid flows, this information is not fully exploited. It is helpful to incorporate physical knowledge into data-driven methods, so they become more robust, require less data, and suffer less from the overfitting problem. Bayesism [8] seems promising because it provides a natural way to include prior information. However, physical knowledge is much more challenging to represent compared with simple prior distribution. Therefore, we need a new approach to express the prior physical knowledge in a meaningful and mathematically rigorous way.

The pursuit of "white box" models has lasted for thousands of years and dominated the history of science. For instance, Newton's laws of motion and laws of universal gravitation are highly interpretable and are considered "white box" theory. Traditionally, scientists hold the belief that having too many parameters in the model is a bad idea. For example, John von Neumann said that "with four parameters I can fit an elephant, and with five I can make him wiggle his trunk." [23]. However, it seems that now lots of problems can only be solved by "black box" methods. In the past few decades, deep neural networks with millions of parameters have been successfully used for many challenging tasks (e.g., image understanding [77], natural language processing [20]) that traditional methods can not solve. In this new era with more and more data, is white box models and interpretability still indispensable? Is it possible that black box modeling leads to the next science explosion?

Data-driven modeling methods share many similarities with machine learning methods. In fact, both research communities focus on the problem of learning models from data. Here are a few examples. First, the proper orthogonal decomposition (POD) [60] (1967) is also called the principal component analysis (PCA) [74] (1901)

in machine learning. Second, recurrent neural networks (RNN) [24] (1990s) are widely used in sequential modeling [58] (e.g., time series and natural language processing), and they are actually dynamical systems (1900s). Third, reinforcement learning [93] (1950s) is closely related to optimal control theory [92] (1950s). Machine learning provides many powerful methods for learning models from data. Therefore, both fields can benefit from bridging the gap between them. Nevertheless, most machine learning algorithms are designed for learning static functions and relationships (e.g., regression algorithms, feed-forward neural networks [82]). Dynamical systems are characterized by their temporal dynamic behaviors. How to effectively leverage machine learning for dynamics learning and understanding remains an unsolved problem.

Part II

Publications

Organization

Part II contains published articles. There might exist minor formatting differences between the published papers and the chapters here. The publications are organized into chapters as follows.

- Chapter 6 proposes an accuracy criterion for dynamic mode decomposition. The criterion is purely data-driven and physically meaningful. It applies to variants of DMD, and assists in model selection.
- Chapter 7 develops fast algorithms for online model learning. It can be applied to a stream of real-time measurements and is capable of discounting old snapshots. Also, the algorithms can be used for both linear and nonlinear system identification, where control is included.
- Chapter 8 studies the input-output response of a separated flow past a flat plate and reveals that the separation bubble is most sensitive to streamwise body force (control) in the upstream of the separation point.

Author contributions

In the following chapters, I did the majority of the research, programming, analysis, and writing. Clancy Rowley guided me on almost every aspect of the research and revised the papers. The other co-authors helped me mainly by providing numerical and experimental data. I describe specific contributions from the co-authors, and the rest unlisted contributions are from myself.

- In chapter 6, Clancy Rowley proposed the counterexample to show that mode amplitude can be misleading and is not always a useful criterion for mode selection. Scott Dawson helped to structure the paper and proofread the draft.

Eric Deem and Louis Cattafesta provided the experimental data for the canonical separated flow. An anonymous reviewer suggested studying the long term Koopman prediction error quantified by the accuracy criterion. Dr. Jessica Shang (appears in the paper acknowledgment) authorized our use of the flow past circular cylinder experiment data.

- In chapter 7, Clancy Rowley suggested a more compact formulation of the window DMD algorithm and proposed the 2D linear time-varying system. Eric Deem provided the data for the canonical separated flow experiment. Louis Cattafesta suggested a power spectral density analysis of the pressure data. An anonymous reviewer suggested changing the fan rotation speed for the system to exhibit time-varying dynamics. Dr. Maziar Hemati (appears in the paper acknowledgment) made helpful suggestions about references.

- In chapter 8, Clancy Rowley suggested a detailed discussion of the optimal forcing mode and the numerical simulation setup in the future work section. Wen Wu, Charles Meneveau, and Rajat Mittal provided the simulation data for the canonical separated flow.

Chapter 6

Evaluating the accuracy of the dynamic mode decomposition

Hao Zhang[1], Scott T. M. Dawson[2], Clarence W. Rowley[1], Eric A. Deem[3], and Louis N. Cattafesta[3]

[1]Mechanical and Aerospace Engineering, Princeton University, Princeton, NJ 08544, USA

[2]Graduate Aerospace Laboratories, California Institute of Technology, Pasadena, CA 91125, USA

[3]Mechanical Engineering, Florida State University, Tallahassee, FL 32310, USA

Appears as [111], Journal of Computational Dynamics, 2019, doi: 10.3934/jcd.2020002.

Dynamic mode decomposition (DMD) gives a practical means of extracting dynamic information from data, in the form of spatial modes and their associated frequencies and growth/decay rates. DMD can be considered as a numerical approximation to the Koopman operator, an infinite-dimensional linear operator defined for (nonlinear) dynamical systems. This work proposes a new criterion to estimate the accuracy of DMD on a mode-by-mode basis, by estimating how closely each individual DMD eigenfunction approximates the corresponding Koopman eigenfunction. This approach does not require any prior knowledge of the system dynamics or the true Koopman spectral decomposition. The method may be applied to extensions of DMD

(i.e., extended/kernel DMD), which are applicable to a wider range of problems. The accuracy criterion is first validated against the true error with a synthetic system for which the true Koopman spectral decomposition is known. We next demonstrate how this proposed accuracy criterion can be used to assess the performance of various choices of kernel when using the kernel method for extended DMD. Finally, we show that our proposed method successfully identifies modes of high accuracy when applying DMD to data from experiments in fluids, in particular particle image velocimetry of a cylinder wake and a canonical separated boundary layer.

6.1 Introduction

The decomposition of spatio-temporal data into spatial modes and temporal functions describing their evolution gives a means to isolate coherent features and assemble low-order representations of complex dynamics. Over the past decade, the dynamic mode decomposition (DMD) [81] has become a routinely-used method for such purposes [80, 98, 19, 113]. See, for example, [57] and [79] for reviews of many ensuing uses and applications of DMD. While successfully used on a range of datasets, general questions still exist in terms of how to select a reduced set of modes, and how to ensure results are quantitatively accurate. On the first point, numerous methods have been proposed to select a reduced number of modes that best represent the dynamics of the system [15, 108, 52, 55]. On the second point, the sensitivity of the outputs of DMD to noisy data has also been investigated [22, 7], and a number of modified algorithms proposed that give improved accuracy for noisy data [46, 17, 4].

The present work differs from these past studies by providing a means of estimating the accuracy of DMD on a mode-by-mode basis, without any a-priori knowledge of the system dynamics, noise characteristics, or truncation of low-energy modes. A related

method has been considered in [21]. It has been shown previously [80, 99] that DMD approximates the Koopman operator, an infinite-dimensional linear operator defined for (nonlinear) dynamical systems. In this work, we will exploit this connection by estimating the accuracy to which we approximate eigenfunctions of the Koopman operator. This approach allows our analysis to naturally generalize to extensions of DMD [104] that are designed to improve the approximation to the Koopman operator for nonlinear systems. Extended DMD uses nonlinear observables to expand the space in which the Koopman operator is approximated. However, EDMD suffers from the curse of dimensionality: that is, the computational cost increases rapidly with the dimension of the state. To circumvent this issue, kernel DMD (KDMD) [105] was proposed as a computationally inexpensive alternative, which makes use of a kernel function to implicitly include a rich (and nonlinear) set of observables, while maintaining the same computational cost as DMD. The optimal choice of kernel function for KDMD is still an open question, and here we demonstrate that the accuracy criterion may be used to evaluate and compare the performance of various kernels.

The structure of this work is as follows. We first review DMD, the Koopman operator, and kernel DMD in section 6.2, before presenting and validating our proposed accuracy criterion in section 6.3. Section 6.4 uses the accuracy criterion to measure the performance of various kernels in KDMD for a simple nonlinear system, while section 6.5 demonstrates that this criterion is effective in selecting accurate DMD modes from experimental data.

6.2 Background

We first give a review of previous results, including the DMD algorithm and its connections to the Koopman operator (section 6.2.1), as well as extensions of DMD that can better approximate the Koopman operator for nonlinear systems (section 6.2.2).

6.2.1 Dynamic mode decomposition

Dynamic mode decomposition was introduced in [81], and our presentation here follows that in [99, 79]. Consider a discrete-time dynamical system whose state space is denoted by $X \subset \mathbb{R}^n$, and suppose the dynamics are given by

$$\boldsymbol{x}(k+1) = \boldsymbol{F}(\boldsymbol{x}(k)), \quad \boldsymbol{x}(k) \in X. \tag{6.1}$$

Let $\psi_1, \ldots, \psi_q$ be real-valued functions on X, which we call observables, and let $\boldsymbol{\psi} : X \to \mathbb{R}^q$ denote the vector-valued function whose components are $(\psi_1, \ldots, \psi_q)$. We may not be able to measure the state $\boldsymbol{x}$ directly, but instead, we can measure the vector

$$\boldsymbol{y} = \boldsymbol{\psi}(\boldsymbol{x}).$$

As a special case, $\boldsymbol{y}$ could be the state itself, i.e., $\boldsymbol{y} = \boldsymbol{\psi}(\boldsymbol{x}) = \boldsymbol{x}$. For complex systems, it can be advantageous to define observables that are nonlinear functions of the state, which will be discussed in more detail in section 6.2.2. For the purposes of describing standard DMD, we assume $\boldsymbol{y} = \boldsymbol{x}$.

We consider pairs of snapshots $(\boldsymbol{x}_k, \boldsymbol{x}_k^{\#})$, with $\boldsymbol{x}_k \in X, k = 1, 2, \ldots, m$, and where $\boldsymbol{x}_k^{\#} = F(\boldsymbol{x}_k)$ is the image of $\boldsymbol{x}_k$ upon application of the dynamics (6.1). For sequential data, $\boldsymbol{x}(1), \ldots, \boldsymbol{x}(m+1)$ satisfying (6.1), one takes $\boldsymbol{x}_k = \boldsymbol{x}(k)$, $\boldsymbol{x}_k^{\#} = \boldsymbol{x}(k+1)$, though non-sequential data may also be used, such as from multiple runs of experiments or

simulations [99]. In DMD, we seek a matrix $\boldsymbol{A} \in \mathbb{R}^{q \times q}$ such that

$$\boldsymbol{y}_k^{\#} = \boldsymbol{A}\boldsymbol{y}_k, \quad k = 1, 2, \ldots, m$$

holds, at least approximately. We form two matrices

$$\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{y}_1 & \boldsymbol{y}_2 & \cdots & \boldsymbol{y}_m \end{bmatrix}, \quad \boldsymbol{Y}^{\#} = \begin{bmatrix} \boldsymbol{y}_1^{\#} & \boldsymbol{y}_2^{\#} & \cdots & \boldsymbol{y}_m^{\#} \end{bmatrix},$$

and define the DMD matrix $\boldsymbol{A}$ by

$$\boldsymbol{A} = \boldsymbol{Y}^{\#}\boldsymbol{Y}^{+}, \tag{6.2}$$

where $\boldsymbol{Y}^{+}$ denotes the Moore-Penrose pseudoinverse [34] of $\boldsymbol{Y}$. DMD modes and eigenvalues are the eigenvectors and eigenvalues of $\boldsymbol{A}$. A typical algorithm to compute these modes and eigenvalues is as follows [99]:

Algorithm (DMD)

1. Compute the reduced SVD $\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$.
2. (Optional) Truncate the SVD by only retaining the first r columns of $\boldsymbol{U}, \boldsymbol{V}$, and the first r rows and columns of $\boldsymbol{\Sigma}$, to obtain $\boldsymbol{U}_r, \boldsymbol{\Sigma}_r, \boldsymbol{V}_r$.
3. Let $\tilde{\boldsymbol{A}} = \boldsymbol{U}_r^T\boldsymbol{A}\boldsymbol{U}_r = \boldsymbol{U}_r^T\boldsymbol{Y}^{\#}\boldsymbol{V}_r\boldsymbol{\Sigma}_r^{-1}$, $\tilde{\boldsymbol{A}} \in \mathbb{R}^{r \times r}$.
4. Find the eigenvalues μ_i and eigenvectors $\tilde{\boldsymbol{v}}_i$ of $\tilde{\boldsymbol{A}}$, such that $\tilde{\boldsymbol{A}}\tilde{\boldsymbol{v}}_i = \mu_i\tilde{\boldsymbol{v}}_i$.
5. The (projected) DMD modes are given by $\boldsymbol{v}_i = \boldsymbol{U}_r^T\tilde{\boldsymbol{v}}_i$, with corresponding (discrete-time) DMD eigenvalues μ_i.

The eigenvectors of the matrix $\boldsymbol{A} \in \mathbb{R}^{q \times q}$ can be found from the eigenvectors of the smaller matrix $\tilde{\boldsymbol{A}} \in \mathbb{R}^{r \times r}$. We denote the eigenvalues and eigenvectors of $\boldsymbol{A}$ by

$\{\mu_i, \boldsymbol{v}_i\}$. In the case of sequential data (for which $\boldsymbol{y}_k^{\#} = \boldsymbol{y}_{k+1}$), suppose that we can express the initial state as

$$\boldsymbol{y}_1 = \sum_{i=1}^{q} c_i \boldsymbol{v}_i.$$

The time evolution of the system (starting at $\boldsymbol{y}_1$) is then predicted by DMD to be

$$\boldsymbol{y}_{k+1} = \boldsymbol{A}^k \boldsymbol{y}_1 = \sum_{i=1}^{q} c_i \mu_i^k \boldsymbol{v}_i. \tag{6.3}$$

Therefore, each DMD mode $\boldsymbol{v}_i$ is associated with a single frequency and growth/decay rate (DMD eigenvalue μ_i). In reality, (6.3) may not hold exactly, depending on the quantity and quality of data used, whether the system dynamics are nonlinear, whether the SVD is truncated in step 2 of the DMD algorithm above, etc. For cases where equation (6.3) does not give an exact description of the dynamics, DMD gives a least-squares fit to the data (as pairs of snapshots).

There are connections between DMD and an infinite-dimensional linear operator called the Koopman operator [99, 80], with the high-level idea being that DMD gives a finite-dimensional numerical approximation of the Koopman operator. Our proposed criterion for evaluating the accuracy of DMD exploits this connection. For a given state-space X, the Koopman operator acts on scalar-valued functions of X, which we referred to earlier as observables. Here, we consider observables in $L^2(X)$, the space of square integrable functions on X. Given the dynamics in (6.1), one defines the Koopman operator $\mathcal{K} : L^2(X) \to L^2(X)$ as follows: for any $f \in L^2(X)$,[1]

$$(\mathcal{K}f)(\boldsymbol{x}) = (f \circ \boldsymbol{F})(\boldsymbol{x}) = f(\boldsymbol{F}(\boldsymbol{x})). \tag{6.4}$$

[1]To be rigorous, one typically assumes there is a measure μ that is preserved under the dynamics (6.1), in which case it follows that for any function $f \in L^2(X, \mu)$, the function $f \circ \boldsymbol{F}$ is also in $L^2(X, \mu)$; in fact, if $\mathbf{F}$ is measure preserving, then $\mathcal{K}$ is an isometry.

That is, $\mathcal{K}$ maps a function f to another function $f \circ \boldsymbol{F}$, and $(\mathcal{K}f)(\boldsymbol{x})$ gives the value of $f(\boldsymbol{x})$ at the next time step. Here we emphasize two points: first, that the Koopman operator acts on functions of the state instead of the state itself; and second, that the Koopman operator is linear, even though the dynamics might be nonlinear. On the second point, note that

$$\mathcal{K}(c_1 f_1 + c_2 f_2) = c_1(\mathcal{K}f_1) + c_2(\mathcal{K}f_2)$$

holds for any functions f_1, f_2 and any scalars c_1, c_2. Since the Koopman operator is linear, it may have eigenvalues and eigenfunctions, which satisfy

$$\mathcal{K}\varphi = \mu\varphi, \tag{6.5}$$

where φ is the eigenfunction with eigenvalue μ.

One reason that one might be interested in finding Koopman eigenfunctions is that such eigenfunctions may be used to define coordinates in which the dynamics are particularly simple—in fact, linear. To see this, let

$$z(k) = \varphi(\boldsymbol{x}(k)) \tag{6.6}$$

be the new coordinate, where φ is an eigenfunction of the Koopman operator. Then the new coordinate evolves according to

$$z(k+1) = \varphi(\boldsymbol{x}(k+1)) = \varphi(\boldsymbol{F}(\boldsymbol{x}(k))) = \mathcal{K}\varphi(\boldsymbol{x}(k)) = \mu\varphi(\boldsymbol{x}(k)) = \mu z(k), \tag{6.7}$$

where we have used the property of a Koopman eigenfunction (6.5). Therefore, the new coordinate $z(k)$ evolves linearly.

Now, suppose we have a given set of observables $\{\psi_1, \psi_2, \ldots, \psi_q\}$, and suppose φ is a Koopman eigenfuntion (with eigenvalue μ) that lies in the span of $\{\psi_j\}$: i.e.,

$$\varphi(\boldsymbol{x}) = \bar{w}_1\psi_1(\boldsymbol{x}) + \ldots + \bar{w}_q\psi_q(\boldsymbol{x}) = \boldsymbol{w}^*\boldsymbol{\psi}(\boldsymbol{x}), \tag{6.8}$$

for some $\boldsymbol{w}^* \in \mathbb{C}^n$. Then one can show (see [99, §4.1]) that under certain conditions on the data, $\boldsymbol{w}^*$ is a left eigenvector of the DMD matrix $\boldsymbol{A}$ with eigenvalue μ (i.e., $\boldsymbol{w}^*\boldsymbol{A} = \mu\boldsymbol{w}^*$). This connection implies that we can approximate Koopman eigenfunctions (and eigenvalues) for a given unknown dynamical system directly from data using DMD. In particular, given left eigenvectors of the DMD matrix ($\boldsymbol{w}_i^*\boldsymbol{A} = \mu_i\boldsymbol{w}_i^*$), we consider $\varphi_i(\boldsymbol{x}) = \boldsymbol{w}_i^*\boldsymbol{\psi}(\boldsymbol{x})$ as a DMD-approximated Koopman eigenfunction, with eigenvalue μ_i.

6.2.2 Extended DMD and kernel DMD

In order to apply the connection between DMD and Koopman mentioned above, the Koopman eigenfunctions must lie within the space spanned by the observables $\{\psi_j\}$. If one takes $\boldsymbol{\psi}(\boldsymbol{x}) = \boldsymbol{x}$, as with standard DMD, then the subspace spanned by $\{\psi_j\}$ consists only of linear functions of $\boldsymbol{x}$, and this subspace is often not large enough to include eigenfunctions of $\mathcal{K}$ (a notable exception being the case in which $\boldsymbol{F}$ is linear). Extended DMD (EDMD) was proposed in [104] in order to enlarge the subspace of observables, and therefore better approximate Koopman eigenfunctions. In particular, Extended DMD approximates the Koopman operator by a weighted residual method, with trial functions given by $\{\psi_j\}$ and a particular choice of test functions specified by the data. Examples of observables $\psi_j(\boldsymbol{x})$ could include polynomials, Fourier modes, indicator functions, or spectral elements, as suggested in [104]. For instance, if we take $\boldsymbol{x} \in \mathbb{R}^2$ and take observables to be monomials in components of $\boldsymbol{x}$ up to degree

$d = 2$ (including the constant 1), then the vector of observables is

$$\boldsymbol{\psi}(\boldsymbol{x}) = \begin{bmatrix} 1 & x_1 & x_2 & x_1^2 & x_1 x_2 & x_2^2 \end{bmatrix}^T.$$

We can potentially approximate many more accurate Koopman eigenfunctions with EDMD than we can with DMD. However, EDMD suffers from the curse of dimensionality [11]. If the state dimension is n and we consider (multivariate) polynomials up to degree d, then the number of observables is $q = \binom{n+d}{d}$, which is approximately n^d for large n. For large problems (as arise in fluids), data might typically have $n \approx 10^6$, so even if one considers only quadratic polynomials, the number of observables is $q \approx 10^{12}$, too large for practical computation. It is thus very computationally expensive to consider large subspaces of observables.

Kernel DMD (KDMD) has been proposed to deal with this curse of dimensionality [105]. In KDMD, EDMD is reformulated such that only inner products of observables need to be computed. The inner product can be evaluated by making use of a kernel function, a common technique in the community of machine learning [11]. A kernel function $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is defined as

$$k(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \langle \boldsymbol{\psi}(\boldsymbol{x}), \boldsymbol{\psi}(\hat{\boldsymbol{x}}) \rangle. \tag{6.9}$$

To appreciate how kernel functions work, consider for example a polynomial kernel $k(\boldsymbol{x}, \hat{\boldsymbol{x}}) = (1 + \boldsymbol{x}^T \hat{\boldsymbol{x}})^d$ as an example. This kernel corresponds to a set of observables $\boldsymbol{\psi}(\boldsymbol{x})$ consisting of all monomials in components of $\boldsymbol{x}$ up to degree d [11]. Taking $n = 2$ and $d = 2$, this kernel function can be expanded as

$$\begin{aligned} (1 + \boldsymbol{x}^T \hat{\boldsymbol{x}})^2 &= 1 + 2x_1\hat{x}_1 + 2x_2\hat{x}_2 + 2x_1^2\hat{x}_1^2 + 2x_1x_2\hat{x}_1\hat{x}_2 + \hat{x}_1^2\hat{x}_2^2 \\ &= \langle \boldsymbol{\psi}(\boldsymbol{x}), \boldsymbol{\psi}(\hat{\boldsymbol{x}}) \rangle, \end{aligned} \tag{6.10}$$

where $\boldsymbol{\psi}(\boldsymbol{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2)$. In the terminology of machine learning, $\boldsymbol{\psi}$ is called the feature map, and $\boldsymbol{\psi}(\boldsymbol{x}) \in \mathbb{R}^q$ is called the feature space (which might be infinite-dimensional). In the example above, the dimension of the (implicitly defined) feature space is $q = 6$, but in order to compute $k(\boldsymbol{x}, \hat{\boldsymbol{x}})$, we require inner products only in state space, which has dimension $n = 2$. Kernel functions hence can be used to evaluate the inner product in a high-dimensional (or even infinite-dimensional) feature space in an efficient way. More examples of kernel functions are given in section 6.4.1.

6.3 Accuracy criterion for DMD

The connection between DMD and the Koopman operator as discussed in section 6.2.1 implies that we can use variants of DMD (e.g., DMD, EDMD, or KDMD) to approximate Koopman eigenfunctions and eigenvalues, given access to data. By applying DMD variants to a given dataset, we can potentially identify many Koopman eigenfunctions and eigenvalues (which we refer to as eigenpairs). However, the reliability of these eigenpairs remains unknown. Before using DMD results for any analysis or reduced-order modeling, it is desirable and necessary to assess the quality (i.e., accuracy) of the results. In this section, we will develop a criterion for evaluating the accuracy of DMD-approximated Koopman eigenpairs. We describe this accuracy criterion in section 6.3.1, and then validate it in section 6.3.2 using a simple nonlinear system where the analytical Koopman eigenpairs are known.

The most common way to select which of the computed DMD modes are most relevant is to use the "mode amplitude": for sequential data, one projects the initial condition onto DMD modes and one views the magnitude of the projection coefficients as the mode amplitudes. It is common practice [99, 44, 80] to retain the modes of largest amplitude. This approach sounds plausible; however, it was observed in [52]

(which used sparsity-promoting techniques to select modes) that mode amplitude is not always a useful criterion for mode selection. Indeed, mode amplitudes can be misleading, as we illustrate below with a simple example.

Suppose we have three DMD modes,

$$\boldsymbol{v}_1 = (1, 0, 0), \qquad \boldsymbol{v}_2 = (0, 1, 0), \qquad \boldsymbol{v}_3 = (0, 1, \epsilon), \tag{6.11}$$

where ϵ is small and thus $\boldsymbol{v}_2$ and $\boldsymbol{v}_3$ are almost parallel. If we consider an initial condition $\boldsymbol{x}_0 = (1, 0, \zeta)$, and project it onto these DMD modes, we obtain

$$\boldsymbol{x}_0 = \boldsymbol{v}_1 - \frac{\zeta}{\epsilon}\boldsymbol{v}_2 + \frac{\zeta}{\epsilon}\boldsymbol{v}_3. \tag{6.12}$$

For instance, if $\zeta = 10^{-3}$ and $\epsilon = 10^{-6}$, then $\zeta/\epsilon = 10^3$, so the mode amplitude (defined as the magnitude of the projection coefficients) indicates that $\boldsymbol{v}_2$ and $\boldsymbol{v}_3$ are much more important than $\boldsymbol{v}_1$. The mode amplitudes indicate that we might be able to neglect $\boldsymbol{v}_1$ without significant adverse effects. However, it is clear that $\boldsymbol{v}_1$ is much more relevant for reconstructing $\boldsymbol{x}_0$: if we use only $\boldsymbol{v}_2$ and $\boldsymbol{v}_3$, we obtain

$$-\frac{\zeta}{\epsilon}\boldsymbol{v}_2 + \frac{\zeta}{\epsilon}\boldsymbol{v}_3 = (0, 0, \zeta), \tag{6.13}$$

which does not accurately approximate $\boldsymbol{x}_0 = (1, 0, \zeta)$. A better approximation to $\boldsymbol{x}_0$ is simply $\boldsymbol{v}_1 = (1, 0, 0)$. This example illustrates that mode amplitude is not always a reliable criterion for selecting which modes are essential, especially when modes are almost parallel. Note that this problem would also arise if using other methods to measure mode amplitude (e.g., [55]).

The accuracy criterion we describe below does not provide a way of selecting which modes are dominant, and in the example above, the accuracy of the modes did not play a role. However, the accuracy criterion can provide a way of eliminating

candidate modes that we know to be inaccurate, so in this way it can help with the problem of mode selection.

6.3.1 Proposed accuracy criterion

Given data from an experiment or simulation, we can split the dataset into training data and testing data. Training data is used to approximate DMD modes (and associated Koopman eigenpairs), while testing data is used to evaluate the quality of these identified modes. Data-driven algorithms may suffer from the problem of over-fitting [43], so any evaluation criteria should use testing data that differs from the training data. The notion of training and testing data is the same as that commonly used in machine learning: training data is used for fitting models (in this case, Koopman eigenpairs), and testing data is used for evaluating models (in this case, accuracy of the Koopman eigenpairs).

The idea of our approach is to evaluate the accuracy of a DMD mode (and eigenvalue) by looking at the accuracy of its corresponding Koopman eigenfunction. Suppose we are given an approximate Koopman eigenpair (μ, φ), and we wish to evaluate its accuracy. If (μ, φ) were a true Koopman eigenpair, then by definition it would satisfy

$$\varphi \circ \boldsymbol{F} = \mu\varphi,$$

where $\boldsymbol{F}$ defines the dynamics in (6.1). Ideally, we would like to compute

$$\frac{\|\varphi \circ \boldsymbol{F} - \mu\varphi\|}{\|\varphi\|}, \tag{6.14}$$

where $\|\cdot\|$ is the norm of a function. (We divide by $\|\varphi\|$ so that the above quantity is independent of the scaling of the eigenfunction $\|\varphi\|$.) However, in order to compute (6.14), we require explicit knowledge of the dynamics $\boldsymbol{F}$, which is unknown in most cases of interest. Instead, we can estimate the above quantity using finite num-

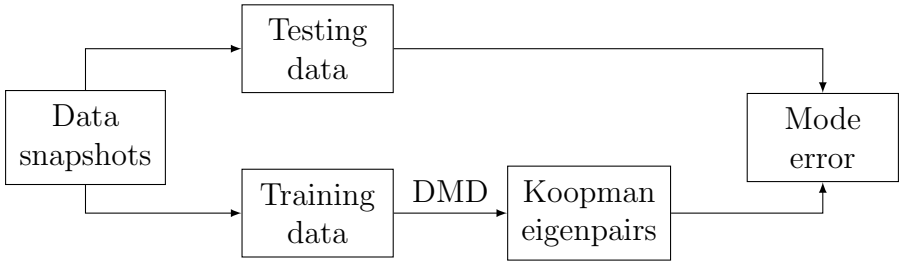


Figure 6.1: A diagram summarizing the implementation of the accuracy criterion. Training data is used to approximate Koopman eigenpairs with variants of DMD, while testing data is used to evaluate the quality of Koopman eigenpairs.

ber of data points (i.e., the testing data). The estimation should give some sense of the quantity in (6.14), using only the testing data, which consists of pairs of samples $(\boldsymbol{x}_k, \boldsymbol{x}_k^{\#})$ with $\boldsymbol{x}_k \in X$ and $\boldsymbol{x}_k^{\#} = \boldsymbol{F}(\boldsymbol{x}_k)$. This observation motivates the following definition of an accuracy criterion:

$$\alpha = \frac{\sum_k |\varphi(\boldsymbol{x}_k^{\#}) - \mu\varphi(\boldsymbol{x}_k)|}{\sum_k |\varphi(\boldsymbol{x}_k)|}, \tag{6.15}$$

where $|\cdot|$ denotes the absolute value, and the summation is over the entire testing dataset. A diagram summarizing how this accuracy criterion may be applied is shown in Figure 6.1. More specifically, given a DMD-approximated eigenfunction $\varphi(\boldsymbol{x}) = \boldsymbol{w}^*\boldsymbol{\psi}(\boldsymbol{x})$ with eigenvalue μ (i.e., $\boldsymbol{w}^*\boldsymbol{A} = \mu\boldsymbol{w}^*$, with $\boldsymbol{A}$ as defined in (6.2)), the accuracy criterion, or estimated mode error, can be written as

$$\alpha = \frac{\sum_k |\boldsymbol{w}^*\boldsymbol{\psi}(\boldsymbol{x}_k^{\#}) - \mu\boldsymbol{w}^*\boldsymbol{\psi}(\boldsymbol{x}_k)|}{\sum_k |\boldsymbol{w}^*\boldsymbol{\psi}(\boldsymbol{x}_k)|}. \tag{6.16}$$

The numerator measures to what extent the eigenfunction equation holds, and the denominator gives a measure of the magnitude of the eigenfunction. Here α can be interpreted as the error of a Koopman eigenpair. The error is defined on a mode-by-mode basis, which enables independent evaluation for each individual DMD mode. Therefore it makes sense to call α the mode error. Observe that α is always non-negative, and it is usually less than 1. When we feed in the true Koopman

eigenfunction and eigenvalue into α in equation (6.15), then $\alpha = 0$ (assuming that the testing data is noise-free). If α is close to 1, the Koopman eigenpair is extremely unreliable, because the discrepancy in the eigenfunction equation is of the same order as the magnitude of the eigenfunction. Therefore, usually we only care about the DMD eigenpairs for which $0 \leq \alpha \ll 1$. In our definition in (6.15), we have used the absolute value to indicate the discrepancy in the eigenfunction equation. However, it is also possible to use other norms, such as the ℓ^2 norm (or its square), which yield similar results in terms of indicating the relative accuracy of modes.

Long-term accuracy. The proposed accuracy criterion may also be used to quantify the accuracy of a long-term simulation of the reduced model, at least in certain situations. Recall that, given a Koopman eigenpair, a new coordinate defined by (6.6) will evolve linearly. Suppose that $\{\varphi, \mu\}$ is a DMD-approximated Koopman eigenpair, and suppose we have sequential testing data $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ (satisfying $\boldsymbol{x}_k^{\#} = \boldsymbol{F}(\boldsymbol{x}_k) = \boldsymbol{x}_{k+1}$). We further assume that $|\mu| \leq 1$; that is, the eigenfunction φ corresponds to a stable direction. Consider the new coordinate $z_k = \varphi(\boldsymbol{x}_k)$, and start from initial condition $\boldsymbol{x}_0$. After m steps, the reduced model prediction is $\mu^m \varphi(\boldsymbol{x}_0)$ (see (7.5)), while the true state in the new coordinate is $\varphi(\boldsymbol{x}_m)$. The error of the long term simulation in the new coordinate is

$$E_m = |\varphi(\boldsymbol{x}_m) - \mu^m \varphi(\boldsymbol{x}_0)|. \tag{6.17}$$

We can rewrite the error as the telescoping sum

$$E_m = \left| \sum_{k=1}^{m} \mu^{m-k} \big(\varphi(\boldsymbol{x}_k) - \mu\varphi(\boldsymbol{x}_{k-1})\big) \right| \leq \sum_{k=1}^{m} |\mu|^{m-k} \big|\varphi(\boldsymbol{x}_k) - \mu\varphi(\boldsymbol{x}_{k-1})\big|,$$

where we have used the triangle inequality. Given the assumption $|\mu| \leq 1$, and for sequential data ($\boldsymbol{x}_k^{\#} = \boldsymbol{x}_{k+1}$), we have

$$E_m \leq \sum_{k=1}^{m} |\varphi(\boldsymbol{x}_k) - \mu\varphi(\boldsymbol{x}_{k-1})| = \sum_{k=0}^{m-1} |\varphi(\boldsymbol{x}_k^{\#}) - \mu\varphi(\boldsymbol{x}_k)| = \alpha \sum_{k=0}^{m-1} |\varphi(\boldsymbol{x}_k)|, \qquad (6.18)$$

where α is the accuracy criterion defined by (6.15). Therefore the long-term simulation error of the corresponding coordinate of the reduced model is also characterized by α.

Scaling. A meaningful evaluation criterion should be (fairly) independent the scaling of the eigenfunctions, the scaling of the testing data, and the size of the testing set. The proposed accuracy criterion approximately satisfies all of these. To show this, we consider the simple case where the full system state is used in DMD, i.e., $\boldsymbol{\psi}(\boldsymbol{x}) = \boldsymbol{x}$ and the DMD-computed eigenfunction is linear, i.e., $\varphi(\boldsymbol{x}) = \boldsymbol{w}^*\boldsymbol{\psi}(\boldsymbol{x}) = \boldsymbol{w}^*\boldsymbol{x}$. The fact that we normalize by the magnitude of the observables means that α is relatively independent of eigenfunction scaling, data scaling, and data quantity, as is desired. In the case where the observable is not the full state (i.e., when using EDMD or KDMD), the scaling of the eigenfunctions and size of testing again do not influence α, for the same reason. However, due to the nonlinear transformation $\boldsymbol{\psi}(\boldsymbol{x})$, the scaling of testing data $\boldsymbol{x}$ may play some role in the size of α. Fortunately, it is reasonable to expect that the relative magnitude of α should still indicate the relative accuracy of different DMD-computed Koopman eigenpairs.

We point out that if the testing data is clean, mode error is determined only by the quality of DMD-approximated Koopman eigenpairs. If the testing data is noisy, mode error is also affected by the noise in testing data. For experimental data, we have access only to the noisy measurements. In these cases, the relative magnitude of α is still expected to indicate the relative accuracy of Koopman eigenpairs. We reiterate

again that this definition of error does not assume access to analytical Koopman spectral decomposition, which is unknown in most cases.

6.3.2 Validating the accuracy criterion

We have proposed an accuracy criterion that exploits the connection between DMD and the Koopman operator. Before applying this criterion to real data, we first seek to validate it as a reliable measure of accuracy. We will first consider a simple 2D nonlinear system for which the analytical Koopman spectral decomposition is known. Given analytical Koopman eigenpairs, we can define the true error to be the distance between the DMD eigenvalue and the true eigenvalue (eigenvalue error), or the difference between the DMD eigenfunction and the true eigenfunction (eigenfunction error). We will validate the accuracy criterion against the true error, and show that the accuracy criterion reliably indicates accuracy.

Here we consider a 2D nonlinear map (also considered in [99]) with dynamics defined by

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \mapsto \begin{bmatrix} \gamma x_1 \\ \delta x_2 + (\gamma^2 - \delta) x_1^2 \end{bmatrix}, \quad \gamma = 0.9, \delta = 0.8. \tag{6.19}$$

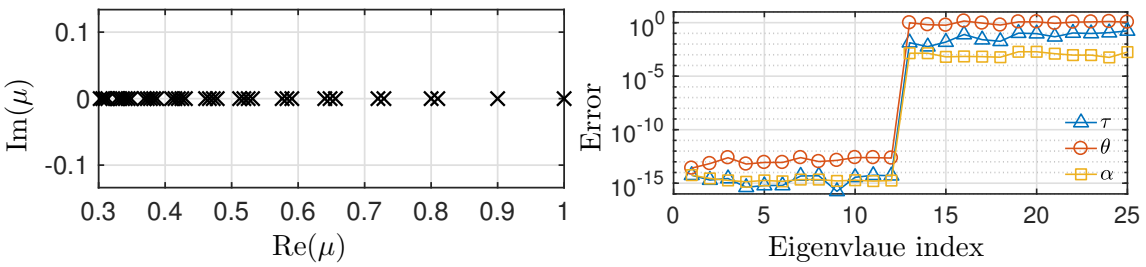
It is straightforward to verify that γ, δ are Koopman eigenvalues with respective eigenfunctions

$$\varphi_\gamma(\boldsymbol{x}) = x_1, \qquad \varphi_\delta(\boldsymbol{x}) = x_2 - x_1^2.$$

Additional Koopman eigenvalues and eigenfunctions are given by

$$\mu_{k,\ell} = \gamma^k \delta^\ell, \qquad \varphi_{k,\ell} = \varphi_\gamma^k \varphi_\delta^\ell, \tag{6.20}$$

where $k, \ell = 0, 1, 2, \ldots$ are non-negative integers. The analytical eigenvalues are both real, and they are shown in Figure 6.2 (a). Notice that the analytical eigenfunctions are multivariate polynomials in the state variables.



(a) Analytical eigenvalues (b) EDMD eigenvalue error

Figure 6.2: (a) Analytical eigenvalues. (b) Comparison between the accuracy criterion α, eigenvalue error τ, and eigenfunction error θ. The eigenvalues are indexed by their absolute value, in descending order.

To collect training data, $m = 100$ random initial points are sampled from a uniform distribution on $[-1, 1] \times [-1, 1]$, and their images are found by applying the map defined in equation (6.19). Similarly we also generate $m_{\text{test}} = 100$ snapshot pairs as the testing data. The generated training and testing dataset are used for subsequent analysis in both this and the next section.

Here we apply EDMD with monomials as observables. In particular, the observables are taken to be

$$\psi_{k,\ell}(\boldsymbol{x}) = x_1^k x_2^\ell, \quad k, \ell = 0, 1, 2, 3, 4, 5,$$

where the feature space dimension is $q = 6 \times 6 = 36$. We report the accuracy criterion for EDMD approximated Koopman eigenvalues in Figure 6.2 (b). We note that mode error indicates that leading eigenvalues are approximated very accurately ($\alpha \sim 10^{-15}$), and this is consistent with the comparison to analytical eigenvalues. As mentioned in section 6.2.1, if the Koopman eigenfunctions lie in the span of the observables, the eigenfunction can be found exactly by EDMD. In this case, monomials up to degree 5 span the leading Koopman eigenfunctions, and hence these eigenvalues can be identified.

To validate that the proposed accuracy criterion does indeed indicate accuracy, now we compare α with the true error. We can compute the discrepancy between DMD eigenvalues, indicated by $\hat{\mu}_i$, and true eigenvalues $\mu_{k,\ell} = \gamma^k \delta^\ell$ given in equation (6.20), by defining the eigenvalue error

$$\tau_i = \frac{|\hat{\mu}_i - \mu_{k,\ell}|}{|\mu_{k,\ell}|}, \tag{6.21}$$

where the indices (k, ℓ) are chosen such that $\mu_{k,\ell}$ is the closest eigenvalue to $\hat{\mu}_i$. We then interpret $\hat{\mu}_i$ as a DMD approximation to the analytical eigenvalue $\mu_{k,\ell}$. We can also compute the discrepancy between DMD eigenfunctions $\hat{\varphi}_i$ and true eigenfunctions $\varphi_{k,\ell}$ given in equation (6.20). We normalize the eigenfunctions $\hat{\varphi}_i$ and $\varphi_{k,\ell}$ so that $|\varphi|_{\max} = 1$ in the domain $\Omega = [-1, 1] \times [-1, 1]$, and define the eigenfunction error as

$$\theta_i = \frac{\|\hat{\varphi}_i - \varphi_{k,\ell}\|}{\|\varphi_{k,\ell}\|}, \tag{6.22}$$

where $\|\cdot\|$ denotes the L^2 norm given by

$$\|f\|^2 = \int_\Omega |f(\boldsymbol{x})|^2 d\boldsymbol{x}. \tag{6.23}$$

In order to validate the accuracy criterion, we compare α_i with the eigenvalue error τ_i and eigenfunction error θ_i in Figure 6.2 (b). We observe that α highly correlates with both τ and θ, even though the proposed accuracy criterion does not assume access to analytical Koopman eigenpairs. The proposed accuracy criterion hence indicates accuracy very well, by comparison with the true error defined using true Koopman eigenpairs. Starting from the 13th eigenvalue $\hat{\mu}_{13} \approx \mu_{6,0} = \gamma^6 \delta^0 = (0.9)^6 (0.8)^0 = 0.531441$, the error dramatically increases, which implies that the remaining eigenfunctions cannot be accurately identified using EDMD with this choice of observables. This is expected, as monomials up to degree 5 can not represent the

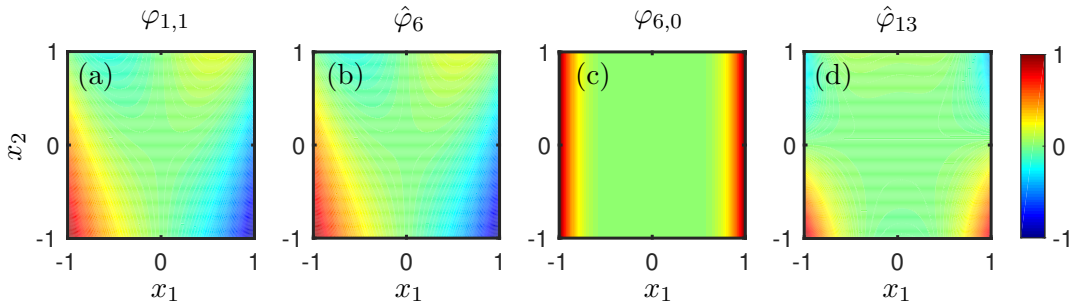


Figure 6.3: Eigenfunctions for the system defined in (6.19), restricted to a domain of $[-1,1] \times [-1,1]$, and normalized such that $|\varphi(\boldsymbol{x})|_{max} = 1$. The analytical eigenfunction $\varphi_{1,1}$ shown in (a) is closely approximated by the eigenfunction $\hat{\varphi}_6$ computed by EDMD, shown in (b). However, the analytical eigenfunction $\varphi_{6,0}$ (with eigenvalue $\mu_{6,0} = 0.531441$) shown in (c) is not closely approximated by its corresponding eigenfunction $\hat{\varphi}_{13}$ computed by EDMD (with eigenvalue $\hat{\mu}_{13} = 0.5250 + 0.0030j$), whose real part is shown in (d).

eigenfunction $\varphi_{6,0}(\boldsymbol{x}) = x_1^6$. This comparison gives us confidence in the reliability of the accuracy criterion.

We now consider the 6th eigenvalue $\hat{\mu}_6 \approx \mu_{1,1} = 0.72$, and the 13th eigenvalue $\hat{\mu}_{13} \approx \mu_{6,0} = 0.531441$. The errors $\tau_6 \approx 10^{-15}$ and $\theta_6 \approx 10^{-13}$ indicate that the 6th eigenpair is approximated very accurately, while $\tau_{13} \approx 10^{-2}, \theta_{13} \approx 10^{0}$ indicate that the 13th eigenpair is approximated with lower accuracy. The EDMD eigenfunctions are compared with the analytical eigenfunctions in Figure 6.3. It is observed that the 6th eigenfunction is indeed approximated very accurately, as $\alpha_6 \approx 10^{-15}$ suggests. The 13th eigenfunction are approximated less accurately, as is expected given that $\alpha_{13} \approx 10^{-3}$. This comparison shows that the accuracy criterion does indicate the accuracy of DMD approximated Koopman eigenpairs, without assuming access to the true Koopman eigenpairs.

6.4 Evaluating the performance of kernels with the accuracy criterion

This section focusses on using the accuracy criterion defined in section 6.3.1 to evaluate the performance of KDMD using various kernel functions. We first introduce a few commonly used kernel functions in section 6.4.1, then we compare the performance of various kernels in section 6.4.2, using the same test problem considered in section 6.3.2. Following this, section 6.4.3 studies the robustness of various kernels for the case where the data are noisy.

6.4.1 Kernel functions

In section 6.2.2 we briefly described KDMD, which makes use of a kernel function to circumvent the curse of dimensionality associated with EDMD. Application of KDMD requires a suitable choice of kernel function. In order to appreciate how a kernel function may implicitly define a observable function, note that Mercer's theorem [65] states that a (quite broad) class of "Mercer kernels" $k(\boldsymbol{x}, \hat{\boldsymbol{x}})$ may be written as

$$k(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \sum_{i=1}^{\infty} c_i \psi_i(\boldsymbol{x}) \psi_i(\hat{\boldsymbol{x}}), \quad c_i \geq c_{i+1} \geq 0. \tag{6.24}$$

Hence there exists an infinite-dimensional implicit observable function (also called feature map in the machine learning community)

$$\boldsymbol{\psi}(\boldsymbol{x}) = \begin{bmatrix} \sqrt{c_1}\psi_1(\boldsymbol{x}) & \sqrt{c_2}\psi_2(\boldsymbol{x}) & \cdots & \sqrt{c_i}\psi_i(\boldsymbol{x}) & \cdots \end{bmatrix}^T \tag{6.25}$$

such that $k(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \langle \boldsymbol{\psi}(\boldsymbol{x}), \boldsymbol{\psi}(\hat{\boldsymbol{x}}) \rangle$. We now introduce a few commonly used kernel functions, and in section 6.4.2 we compare their performance on the example from the previous section.

Polynomial kernel

$$k(\boldsymbol{x}, \hat{\boldsymbol{x}}) = (1 + \boldsymbol{x}^T \hat{\boldsymbol{x}})^d \tag{6.26}$$

The (implicit) observables associated with the polynomial kernel are all monomials in components of $\boldsymbol{x} \in \mathbb{R}^n$ up to degree d. The dimension of the observable vector is $q = \binom{n+d}{d}$. The feature map for arbitrary $n \geq 1, d \geq 0$ is described in details in [16]. The observables when $n = 2, d = 2$ are given by equation (6.10).

Exponential kernel

$$k(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \exp\left(\boldsymbol{x}^T \hat{\boldsymbol{x}}\right) \tag{6.27}$$

The (implicit) observables associated with the exponential kernel are all monomials in components of $\boldsymbol{x}$, up to infinite degree. An explicit feature map can be also found from a Taylor expansion of the exponential kernel [16]. Taking $\boldsymbol{x} \in \mathbb{R}^2$ for example, the kernel can be expanded as

$$\begin{aligned}\exp\{\boldsymbol{x}^T \hat{\boldsymbol{x}}\} &= \sum_{\ell=0}^{\infty} \frac{(\boldsymbol{x}^T \hat{\boldsymbol{x}})^\ell}{\ell!} = \sum_{\ell=0}^{\infty} \frac{(x_1 \hat{x}_1 + x_2 \hat{x}_2)^\ell}{\ell!} \\ &= \sum_{\ell=0}^{\infty} \frac{\sum_{k=0}^{\ell} \binom{l}{k} (x_1 \hat{x}_1)^k (x_2 \hat{x}_2)^{\ell-k}}{\ell!} = \langle \boldsymbol{\psi}(\boldsymbol{x}), \boldsymbol{\psi}(\hat{\boldsymbol{x}}) \rangle,\end{aligned}$$

where the observable is $\psi_{\ell,k}(\boldsymbol{x}) = \left(\binom{\ell}{k}/\ell!\right)^{1/2} x_1^k x_2^{\ell-k}$, where $\ell = 0, 1, 2, \ldots$, and $k = 0, 1, 2, \ldots, \ell$. Notice that the number of observables is infinite, $q = \infty$.

Gaussian kernel

$$k(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \exp\left(-\frac{\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2^2}{\sigma^2}\right), \tag{6.28}$$

where $\|\cdot\|_2$ is the ℓ^2 norm, and σ scales the kernel width [27].

The Gaussian kernel is a Mercer kernel for all dimensions $n \geq 1$ [83]. Take $x \in \mathbb{R}$ as an example, the (implicit) observables as in equation (6.24) are given by [36]

$$\psi_k(x) \propto \exp(-(d-a)x^2) H_k(x\sqrt{2d}),$$

where

$$c_k \propto b^k, \quad b < 1,$$

a, b, d are functions of σ, and H_k is the k-th order Hermite polynomial. The number of observables is infinite, $q = \infty$. For arbitrary n, an explicit feature map can in principle be also found from Taylor expansion of the Gaussian kernel [16].

Laplacian kernel

$$k(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \exp\{-\frac{\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2}{\sigma}\} \tag{6.29}$$

Note the similarity between the Laplacian and Gaussian kernels, with the difference being that that the Laplacian kernel uses the ℓ^2 norm in the exponent without squaring [91]. For arbitrary n, the Laplacian kernel is a valid Mercer kernel [83].

6.4.2 Performance of kernels

We now compare the above kernel functions using the example considered in section 6.3.2. Figure 6.4 shows the performance of polynomial, exponential, Gaussian, and Laplacian kernels in identifying the Koopman eigenvalues of the system, using the same training and testing data as in section 6.3.2.

We find that a polynomial kernel of degree $d = 5$ accurately identifies the leading eigenvalues ($\mu_{k,\ell} \in [0.6, 1]$) with very high accuracy ($\alpha \approx 10^{-14}$), as was the case with EDMD. This is not surprising, as the polynomial kernel implicitly defines monomials of states as observables, which span the same space as the explicitly defined monomials

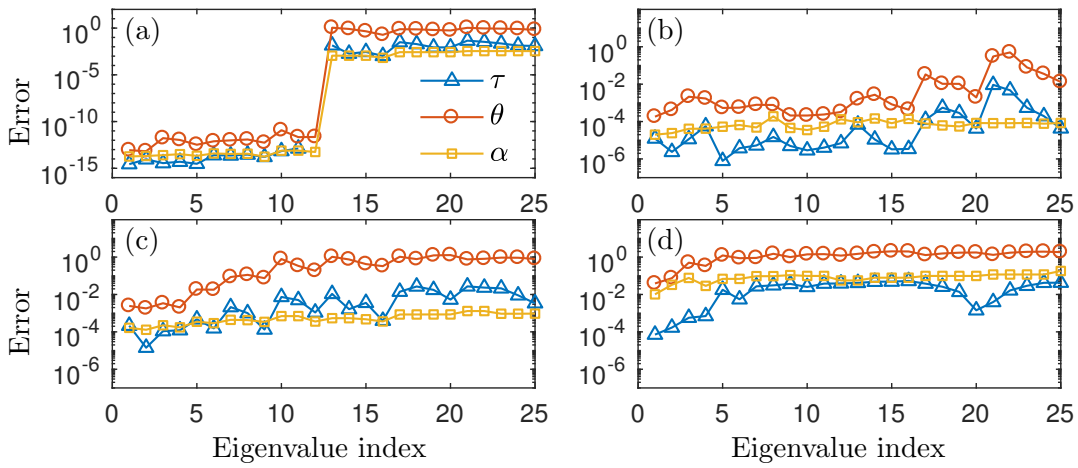


Figure 6.4: Performance of various kernels. Eigenvalue error τ, eigenfunction error θ, and accuracy criterion α are shown. (a) Polynomial kernel of degree $d = 5$, $q = \binom{2+5}{5} = 21$. (b) Exponential kernel, $q = \infty$. (c) Gaussian kernel with $\sigma = 1$, $q = \infty$. (d) Laplacian kernel with $\sigma = 1$, $q = \infty$.

used in EDMD. With the increasing order of the polynomial kernel, more eigenvalues can be accurately identified. It is found that the exponential kernel can identify more eigenvalues ($\mu_{k,\ell} \in [0.5, 1]$) than the polynomial kernel with satisfactory accuracy ($\alpha \approx 10^{-4}$), since the implicit observables associated with the exponential kernel are monomials up to infinite degree. The Gaussian kernel is able to find the leading eigenvalues ($\mu_{k,\ell} \in [0.65, 1]$) with mode error (accuracy criterion) $\alpha \approx 10^{-4}$ to 10^{-3}, even though the implicit observables of the Gaussian kernel are not monomials. This demonstrates the potential power of kernel functions: they are able to span a useful function space, primarily because the dimension of the space of (implicit) observables can be large, and even infinite. The Laplacian kernel can approximate only a few leading eigenvalues ($\mu = 1.0.9, 0.8$), and with a lower accuracy of $\alpha \approx 10^{-2}$.

We emphasize that, while the exact Koopman eigenvalues are known in this case, it is possible to use the accuracy criterion to compare the performance of different kernels even when the true dynamics are unknown. Indeed, using only the results of the accuracy criterion, we would reason that the polynomial kernel is the best choice for identifying the leading Koopman eigenvalues accurately.

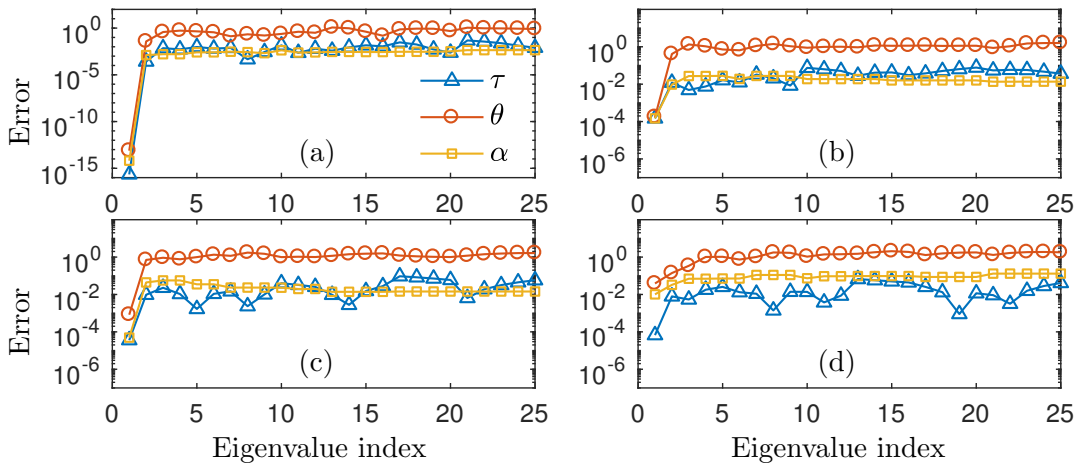


Figure 6.5: Performance of various kernels in the presence of noise. Eigenvalue error τ, eigenfunction error θ, and accuracy criterion α are shown. (a) Polynomial kernel of degree $d = 5$, $q = \binom{2+5}{5} = 21$. (b) Exponential kernel, $q = \infty$. (c) Gaussian kernel with $\sigma = 1$, $q = \infty$. (d) Laplacian kernel with $\sigma = 1$, $q = \infty$.

6.4.3 Sensitivity of kernels to noise

In practice, data is typically corrupted with noise. Here we present a study of the sensitivity of different kernels with respect to the presence of noise. We add zero-mean Gaussian noise with a standard deviation $\sigma_{\text{noise}} = 10^{-3}$ to the 100 random uniformly distributed data pairs taken from $[-1, 1] \times [-1, 1]$, resulting in a signal-to-noise ratio (SNR) of about 10^6. The training data is noisy, but the testing data is "clean". Therefore, the accuracy criterion only accounts for the accuracy of DMD approximated Koopman eigenpairs.

The results are shown in Figure 6.5. We observe that the polynomial kernel is slightly more robust than the other kernels ($\alpha \approx 10^{-3}$) in the presence of noise, and is able to accurately identify the first few leading eigenvalues ($\mu = 1, 0.9$). The reason for this is that the dimension of the implicit observables associated with the polynomial kernel is finite and small ($q = 21$) in comparison to the number of snapshots ($m = 100$), so we avoid problems of overfitting. In KDMD, the Koopman eigenpairs are found from the eigendecomposition of the matrix $\boldsymbol{A}_{\text{KDMD}} = \boldsymbol{Y}^{+}\boldsymbol{Y}^{\#}$, where the columns of $\boldsymbol{Y}$ and $\boldsymbol{Y}^{\#}$ are $\boldsymbol{y} = \boldsymbol{\psi}(\boldsymbol{x}) \in \mathbb{R}^q$ and $\boldsymbol{y}^{\#} = \boldsymbol{\psi}(\boldsymbol{x}^{\#}) \in \mathbb{R}^q$ respectively,

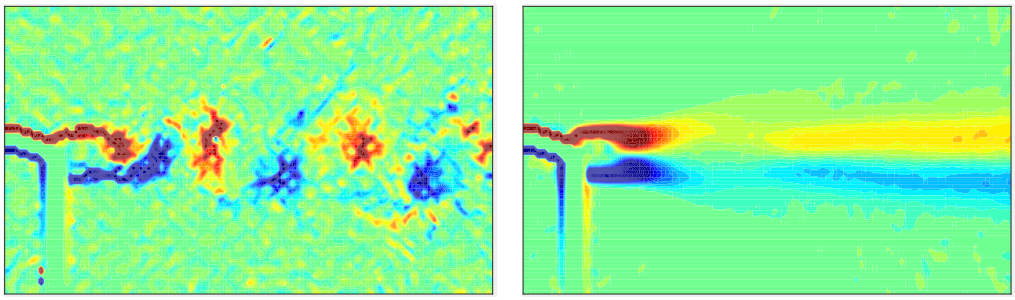
and $\boldsymbol{Y}, \boldsymbol{Y}^{\#} \in \mathbb{R}^{q \times m}$. The matrix $\boldsymbol{A}_{\text{KDMD}}$ has the same non-zero eigenvalues as the DMD matrix $\boldsymbol{A} = \boldsymbol{Y}^{\#}\boldsymbol{Y}^{+}$. $\boldsymbol{A}$ is the optimal (least-square or minimum-norm) solution to $\min_{\boldsymbol{A}} \|\boldsymbol{A}\boldsymbol{Y} - \boldsymbol{Y}^{\#}\|_F$, where $\boldsymbol{Y}, \boldsymbol{Y}^{\#} \in \mathbb{R}^{q \times m}$. For the polynomial kernel, $\boldsymbol{A}$ is the solution to an over-constrained problem ($q < m$), and is hence more robust to noise. In contrast, the exponential kernel, Gaussian kernel, and Laplacian kernel span an infinite-dimensional space of observables ($q = \infty$). The finite-dimensional approximation to the Koopman operator is found by solving an under-constrained problem ($q \gg m$), which makes it more sensitive to noise, as these three kernels tend to over-fit the noise in the training dataset. Given noisy data, they are only able to accurately identify the eigenvalue $\mu = 1$, whose eigenfunction is a constant.

6.5 Identifying accurate DMD modes using experimental data

Having demonstrated the use of the accuracy criterion with synthetic data, now we turn our attention to data from fluids experiments. In these cases, the analytical Koopman spectral decomposition is unknown. An crucial advantage of the proposed accuracy criterion is that it does not rely on known Koopman eigenpairs, and can be applied so long as there is data available. We will use the proposed accuracy criterion to identify accurate DMD modes for vorticity data from flow past a circular cylinder in section 6.5.1, and from a separation experiment in section 6.5.2.

6.5.1 Flow past a circular cylinder

In this example, we use the experimental particle image velocimetry (PIV) data for flow past a circular cylinder at a Reynolds number of 413. The PIV velocity data were sampled at a frequency of 20 Hz with a resolution of 135×80. See [98] for more details about this experiment. This dataset has been used in other studies [47, 105] for



(a) Instantaneous spanwise vorticity (b) Time averaged spanwise vorticity

Figure 6.6: (a) An instantaneous spanwise vorticity field of flow past a circular cylinder at $Re = 413$. (b) Time averaged spanwise vorticity field

testing various proposed DMD algorithms. A typical instantaneous spanwise vorticity field and the time-averaged spanwise vorticity field are shown in Figure 6.6. It is clear that there is vortex shedding behind the cylinder. We will use spanwise vorticity data for DMD, which can be computed from velocity data by finite difference methods. The state dimension is $n = 135 \times 80 = 10800$, and the number of snapshots in training data is taken to be $m = 1000$. We use an additional $m_{\text{test}} = 1000$ snapshot pairs as testing data.

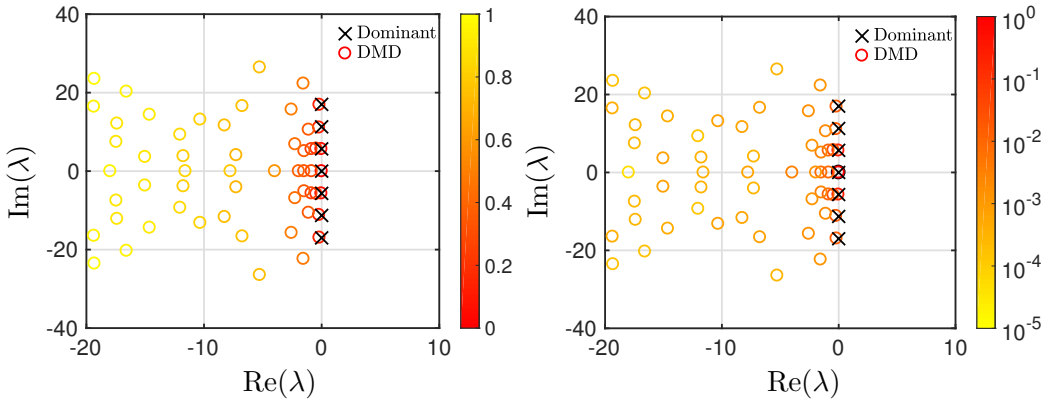
When we apply DMD to sequential data that has time step $\triangle t$, the continuous-time DMD eigenvalues λ_{DMD} are related to the discrete-time DMD eigenvalues μ_{DMD} by

$$\mu_{\text{DMD}} = e^{\lambda_{\text{DMD}} \triangle t}. \tag{6.30}$$

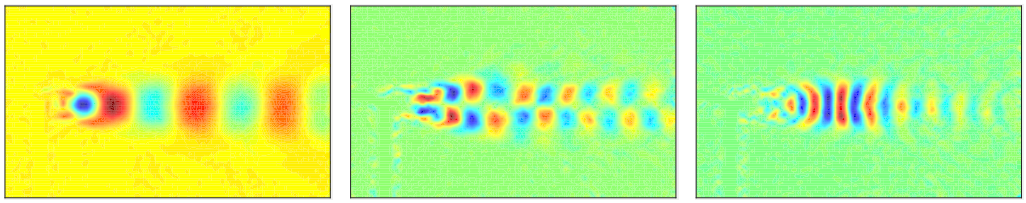
The discrete-time DMD eigenvalues are computed with DMD and converted to continuous-time DMD eigenvalues by equation (7.35), and in this example the time spacing is $\triangle t = (1/20)s$. The DMD frequency f_{DMD} is related to the continuous-time DMD eigenvalues λ_{DMD} by

$$f_{\text{DMD}} = \frac{\text{Im}(\lambda_{\text{DMD}})}{2\pi}, \tag{6.31}$$

where $\text{Im}(\lambda_{\text{DMD}})$ is the imaginary part of λ_{DMD}.



(a) DMD, accuracy criterion (b) DMD, mode amplitude

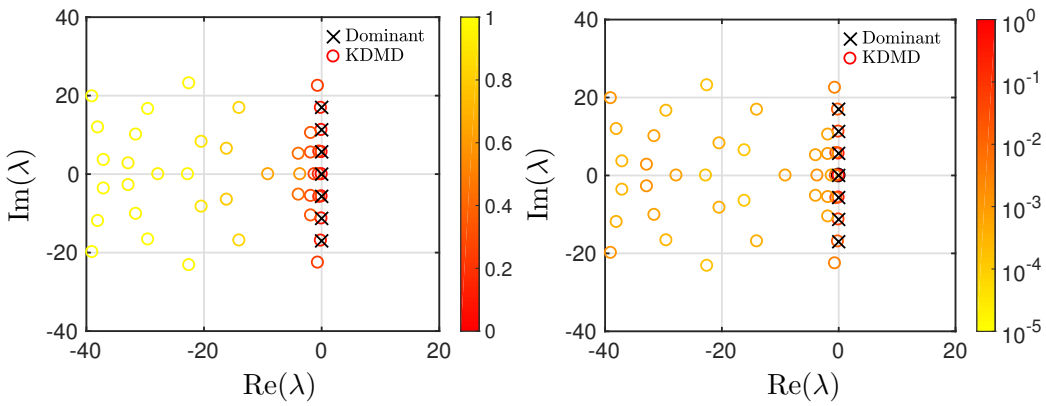


(c) Mode 1, $f_1 = 0.90$ Hz (d) Mode 2, $f_2 = 1.77$ Hz (e) Mode 3, $f_3 = 2.69$ Hz

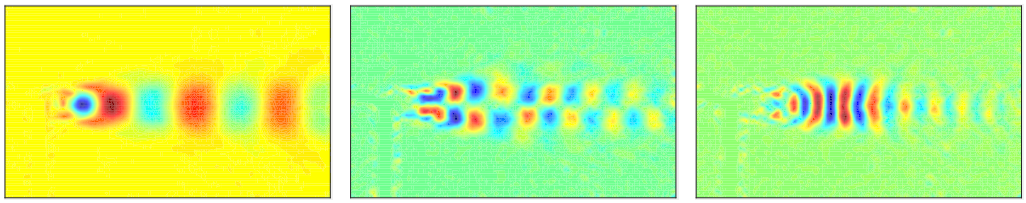
Figure 6.7: (a)–(b) Continuous-time DMD eigenvalues (circles) colored by the accuracy criterion α (a) and mode amplitude β (b). Mode amplitudes are normalized by the maximum amplitude. Dominant frequencies (black cross sign $\times$) are shown for comparison. (c)–(e) Three dominant DMD modes (only show real part) picked out by accuracy criterion and mode amplitude.

We first apply the standard DMD method described in section 6.2.1. We use a truncation level of $r = 100$, which corresponds to preserving 78.16% of the total energy of the snapshots. The continuous-time DMD eigenvalues are shown shaded by the corresponding accuracy criterion values α in Figure 6.7 (a), and time-averaged mode amplitudes β in Figure 6.7 (b) (defined as in [55]).

Inspecting Figure 6.7 (a), we observe that eigenvalues near the imaginary axis are more accurate, and this observation is consistent with physical intuition: this flow exhibits a von Kármán vortex street, whose dominant dynamics evolve on a limit cycle. For this experiment, the wake shedding frequency is $f_{\text{wake}} = 0.889\,\text{Hz}$ [98], In previous work [98], the physically relevant dominant frequencies are reported as $f_0 = 0\,\text{Hz}$, $f_1 = 0.89\,\text{Hz}$, $f_2 = 1.77\,\text{Hz}$, $f_3 = 2.73\,\text{Hz}$. The DMD mode associated with f_0 is the mean of the flow, and f_1, f_2, f_3 are the first, second, and third harmonic of



(a) KDMD, accuracy criterion (b) KDMD, mode amplitude



(c) Mode 1, $f_1 = 0.90$ Hz (d) Mode 2, $f_2 = 1.79$ Hz (e) Mode 3, $f_3 = 2.69$ Hz

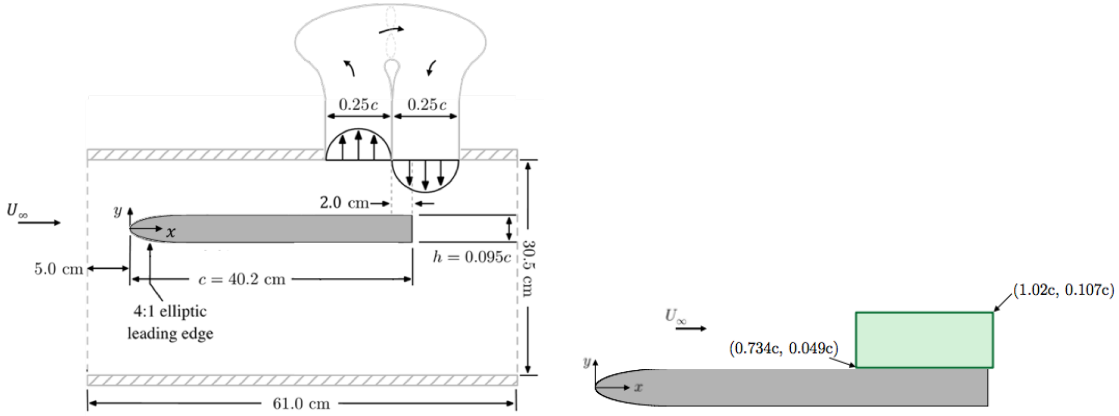
Figure 6.8: (a)–(b) Continuous-time KDMD eigenvalues (circles) colored by the accuracy criterion α (a) and mode amplitude β (b). Mode amplitudes are normalized by the maximum amplitude. Dominant frequencies (black cross sign $\times$) are shown for comparison. (c)–(e) Three dominant KDMD modes (only show real part) picked out by accuracy criterion and mode amplitude.

the fundamental wake frequency f_{wake}. These four frequencies represent the dominant dynamics in this flow. This observation indicates that the proposed accuracy criterion can be used to identify physically relevant DMD modes/eigenvalues, and distinguish relevant modes from irrelevant ones. By comparing Figure 6.7 (a) and Figure 6.7 (b), we verify that the accuracy criterion indicates the same dominant frequencies as the mode amplitude. The DMD modes that have higher accuracy, as indicated by the accuracy criterion are shown in Figure 6.7 (c)–(e). We verify that they look similar to those identified in previous work [98].

Next, we investigate the performance of KDMD on this dataset. Figure 6.8 shows results for a polynomial kernel of degree $d = 5$, again using a truncation level of $r = 100$. The DMD eigenvalues are shown in Figure 6.8 (a)–(b), colored by both accuracy criterion and mode amplitude. The relevant DMD modes picked out by

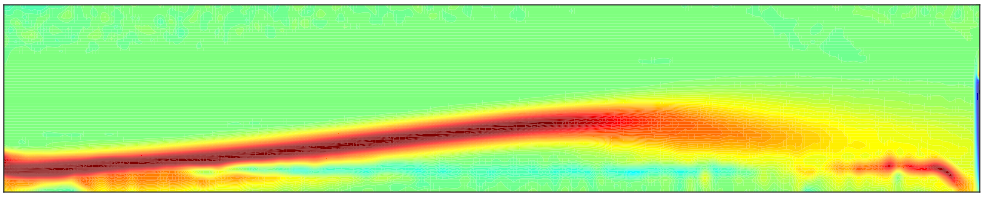
accuracy criterion, and mode amplitude are shown in Figure 6.8 (c)–(e). We verify that the accuracy criterion is able to isolate dominant modes when using KDMD.

6.5.2 Canonical separated flow



(a) Experiment setup

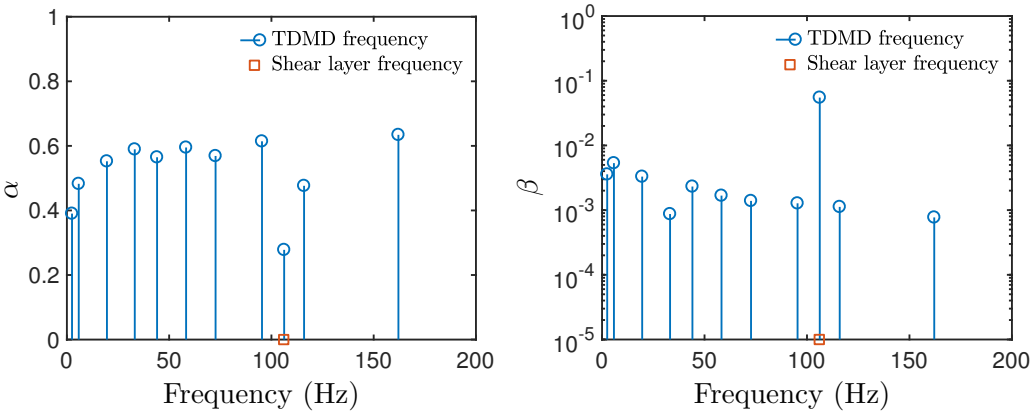
(b) The PIV measurement region



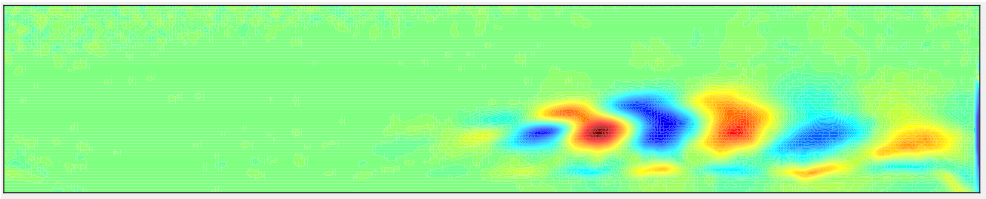
(c) Time-averaged spanwise vorticity field in the PIV measurement region

Figure 6.9: (a) Sketch of the canonical separated flow experiment setup (adapted from [37]). (b) PIV measurement region. (c) Mean spanwise vorticity field

In this example, we use PIV data from a canonical flow separation experiment sketched in Figure 6.9 (a)–(b). Separation is induced on the surface of a flat plate by a zero-net suction/blowing boundary condition imposed on the wall of the wind tunnel, near the trailing edge of the plate. The free-stream velocity is $U_\infty = 3.9\,\mathrm{m/s}$, the chord length is $c = 402\,\mathrm{mm}$, the span is $s = 305\,\mathrm{mm}$, and the height is $h = 0.095c$. The Reynolds number based on chord length is $Re_c = 10^5$, small enough that the boundary layer is laminar upstream of the separation point. The average separation bubble length is $L_{\mathrm{sep}} = 0.2c$. More information regarding the separation system and the flat plate model can be found in [19].



(a) TDMD, accuracy criterion (b) TDMD, mode amplitude

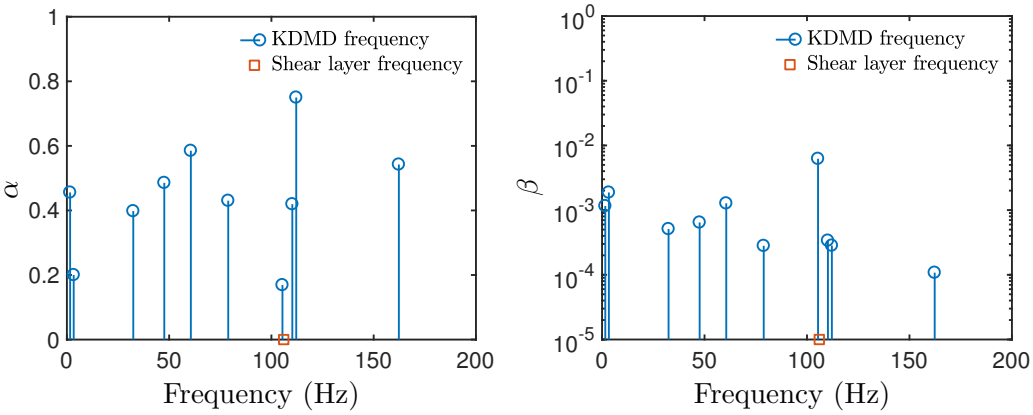


(c) TDMD mode, $f = 106$ Hz

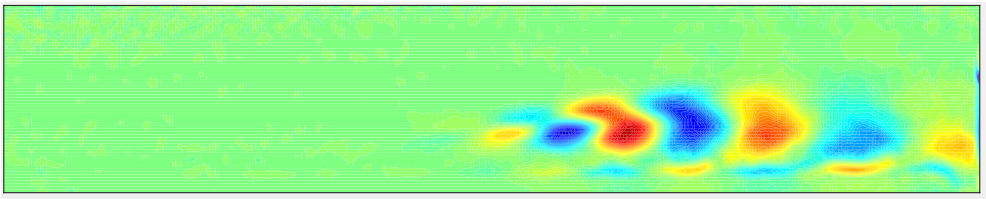
Figure 6.10: TDMD frequency (f_{TDMD}) and corresponding mode error/amplitude. Mode amplitudes are normalized by the maximum mode amplitude. The truncation level is $r = 25$. The shear layer frequency $f_{\text{SL}} = 106$ Hz is denoted with a red square, and corresponds to the most accurate (smallest α) and largest amplitude (largest β) mode.

Two-component PIV velocity data is sampled at $f_s = 1600\,\text{Hz}$, with a resolution of 319×62. The time-averaged spanwise vorticity field in the PIV measurement region is shown in Figure 6.9 (c). The PIV spanwise vorticity dataset for the separated flow studied here consists of $m = 3000$ snapshot pairs (the training data), with a state dimension $n = 319 \times 62 = 19778$. We also take another $m_{\text{test}} = 3000$ snapshot pairs as testing data.

This particular experimental dataset has been used and studied in previous work [44], in which the shear layer frequency was found to be $f_{\text{SL}} = 106\,\text{Hz}$. The shear layer frequency is a periodic roll-up of the shear layer due to the Kelvin-Helmholtz instability. The shear layer frequency f_{SL} can be identified by applying total-least-squares DMD (TDMD), a variant of DMD which makes use of total-least-square regression to improve the accuracy of DMD for noisy data [17, 46]. As in [44], we use



(a) KDMD, accuracy criterion (b) KDMD, mode amplitude



(c) KDMD mode, $f = 105$ Hz

Figure 6.11: KDMD frequency (f_{KDMD}) and corresponding mode error/amplitude. The truncation level is $r = 25$. The shear layer frequency $f_{\text{SL}} = 106$ Hz is denoted with a red square.

a truncation level of $r = 25$, which corresponds to preserving 74% of the energy of the data. In this example the time spacing is $\triangle t = 1/f_s = (1/1600)s$.

For comparison, we also compute the time-averaged mode amplitude β, as in the example in section 6.5.1 (e.g., Figure 6.7 (b)). The DMD frequencies are plotted against their accuracy criterion values and mode amplitudes in Figure 6.10 (a)–(b). It is observed that $f_{\text{SL}} = 106$ Hz is accurately identified by TDMD. In addition, it stands out by having a small mode error. The DMD mode associated with shear layer frequency is plotted in Figure 6.10 (c), and it agrees with the mode identified in previous work [44].

We apply KDMD to this dataset, using polynomial kernels of degree $d = 5$, again with a truncation level of $r = 25$. Eigenvalue frequencies, and corresponding accuracy criterion values and mode amplitudes are plotted in Figure 6.11 (a)–(b). We observe that the shear layer frequency has a small error and large mode amplitude,

and once again verify that the DMD mode associated with shear layer frequency (Figure 6.11 (c)) agrees closely with that found in previous work [44].

6.6 Conclusion and outlook

Exploiting the connection between DMD and the Koopman operator, we have presented an accuracy criterion to evaluate the quality (accuracy) of Koopman eigenpairs approximated with DMD variants. The criterion does not assume access to the analytical Koopman spectral decomposition, which is generally unknown in practice. Furthermore, the proposed accuracy criterion naturally applies to other variants of DMD, because it is based on the general notion of Koopman eigenfunctions. We show that in certain situations, the accuracy criterion may also be used to quantify the long-term simulation error of a reduced-order model. In particular, given a testing dataset, the long-term error is shown to be bounded by a multiple of the accuracy criterion. The proposed accuracy criterion is validated with a synthetic system where the analytical Koopman eigenpairs are known. Using this accuracy criterion, we present a study of the performance of various kernels, and assess their sensitivity to noisy data. In our examples, the polynomial kernel (with finite-dimensional observables) performs well both in the sense of accuracy and robustness to noise. Exponential, Gaussian, and Laplacian kernels are able to span an infinite-dimensional function space, but the tradeoff is that they are significantly more sensitive to noise in the dataset. We demonstrate that the accuracy criterion can assist in identifying accurate and physically relevant DMD modes/eigenvalues from experimental data with measurement noise. The accuracy criterion is conceptually simple and easy to use. As a data-driven algorithm, depending on the nature of the problem, sometimes DMD produces relevant results and sometimes outputs numerical artifacts. For reduced-

order modeling based on DMD/Koopman modes, it is vital to assess the quality of DMD results.

Note that our proposed accuracy criterion requires that some portion of data snapshots are withheld from the DMD analysis for purposes of assessing mode accuracy. However, it would be possible to incorporate this additional data into the DMD analysis after the DMD modes and eigenvalues of interest have been identified.

The demand for accurate reduced-order models (ROM) has increased rapidly in recent years, but it is still unclear how to select a subset of Koopman eigenpairs such that the original (nonlinear) system is accurately approximated. In order to build any meaningful ROM, we need to at least assess the accuracy and importance of DMD-approximated Koopman eigenpairs. The present work has shed some light on the accuracy side. However, how to select the most dynamically important Koopman eigenpairs remains an open question. Unlike techniques such as proper orthogonal decomposition, in which the modes are orthogonal by construction, Koopman eigenfunctions are in general not orthogonal (though orthogonal DMD-like modes may be obtained [72]). Mode amplitudes obtained by a projection of data onto DMD modes are not necessarily always a meaningful criterion for evaluating importance, as demonstrated in the example in section 6.3. It would be desirable to develop a criterion that can guide the selection of modes for the purpose of representing the dynamics accurately.

Acknowledgments

The authors gratefully acknowledge Dr. Jessica Shang for the experimental data of the cylinder flow. Hao Zhang thanks Dr. Matthew O. Williams for the generous guidance and help as a labmate.

Chapter 7

Online dynamic mode decomposition for time-varying systems

Hao Zhang[1], Clarence W. Rowley[1], Eric A. Deem[2], and Louis N. Cattafesta[2]

[1]Mechanical and Aerospace Engineering, Princeton University, Princeton, NJ 08544, USA

[2]Mechanical Engineering, Florida State University, Tallahassee, FL 32310, USA

Appears as [113], SIAM Journal on Applied Dynamical Systems, 2019, doi: 10.1137/18M1192329.

Dynamic mode decomposition (DMD) is a popular technique for modal decomposition, flow analysis, and reduced-order modeling. In situations where a system is time-varying, one would like to update the system's description online as time evolves. This work provides an efficient method for computing DMD in real-time, updating the approximation of a system's dynamics as new data becomes available. The algorithm does not require storage of past data, and computes the exact DMD matrix using rank-1 updates. A weighting factor that places less weight on older data can be incorporated in a straightforward manner, making the method particularly well suited

to time-varying systems. A variant of the method may also be applied to the online computation of "windowed DMD", in which only the most recent data are used. The efficiency of the method is compared against several existing DMD algorithms: for problems in which the state dimension is less than about 200, the proposed algorithm is the most efficient for real-time computation, and it can be orders of magnitude more efficient than the standard DMD algorithm. The method is demonstrated on several examples, including a time-varying linear system and a more complex example using data from a wind tunnel experiment. In particular, we show that the method is effective at capturing the dynamics of surface pressure measurements in the flow over a flat plate with an unsteady separation bubble.

7.1 Introduction

Modal decomposition methods are widely used in studying complex dynamical systems such as fluid flows. In particular, dynamic mode decomposition (DMD) [81, 80] has become increasingly popular in the fluids community. DMD decomposes spatio-temporal data into spatial modes (DMD modes) each of which has simple temporal behavior characterized by single frequency and growth/decay rate (DMD eigenvalues). DMD has been successfully applied to a wide range of problems, for instance as discussed in [79, 57]. The idea of DMD is to fit a linear system to observed dynamics. However, DMD is also a promising technique for nonlinear systems, as it has been shown to be a finite-dimensional approximation to the Koopman operator, an infinite-dimensional linear operator that captures the full behavior of a nonlinear dynamical system [80, 99]. A few methods [111, 21] have been proposed to assess the accuracy (quality) of the DMD approximated Koopman eigenfunctions/eigenvalues.

Recently, several algorithms have been proposed to compute DMD modes efficiently for huge datasets, for instance using randomized methods [25, 26]. In situations in which the incoming data is "streaming" in nature, and one does not wish to store all of the data, a "streaming DMD" algorithm performs online updating of the DMD modes and eigenvalues [47]. Streaming DMD keeps track of a small number of orthogonal basis vectors and updates the DMD matrix projected onto the corresponding subspace. Another related method uses an incremental SVD algorithm to compute DMD modes on the fly [62]. The work proposed here may be viewed as an alternative to streaming DMD, in that we provide a method for updating the DMD matrix in real-time, without the need to store all the raw data. Our method differs from Streaming DMD in that we compute the exact DMD matrix, rather than a projection onto basis functions; in addition, we propose various methods for better approximations of time-varying dynamics, in particular by "forgetting" older snapshots, or giving them less weight than more recent snapshots.

It is worth emphasizing that our proposed algorithm relies on an important assumption: the number of snapshots is much larger than the state dimension. In practice, DMD is often applied to fluid problems for which the opposite is true: the state dimension is high, and much larger than the number of snapshots. In this paper, we explore an algorithm for real-time updating of the linear model (DMD matrix), with the ultimate goal of real-time modeling and control. For offline data analysis, one can have access to datasets with huge numbers of states (e.g., measurements of the full flow field). However, in real-time modeling, it is often the case that we have only a small number of measurements (for instance pressure measurements from an array of sensors). Given this limited amount of information, we are trying to find adaptive real-time models.

The paper is organized as follows. In section 7.2, we give an overview of DMD, and describe the online DMD algorithm. In section 7.3, we discussed a variant called

windowed DMD, in which only the most recent data are used. In section 7.4, we briefly describe how these methods may be used in online system identification, and in section 7.5 we compare the different algorithms on various examples.

7.2 Online dynamic mode decomposition

7.2.1 The problem

We first give a brief summary of the standard DMD algorithm, as described in [99]. Suppose we have a discrete-time dynamical system given by

$$\boldsymbol{x}_{j+1} = \boldsymbol{F}(\boldsymbol{x}_j),$$

where $\boldsymbol{x}_j \in \mathbb{R}^n$ is the state vector, and $\boldsymbol{F} : \mathbb{R}^n \to \mathbb{R}^n$ defines the dynamics. For a given state $\boldsymbol{x}_j$, let $\boldsymbol{y}_j = \boldsymbol{F}(\boldsymbol{x}_j)$; we call $(\boldsymbol{x}_j, \boldsymbol{y}_j)$ a *snapshot pair*. For DMD, we assume we have access to a collection of snapshot pairs $(\boldsymbol{x}_j, \boldsymbol{y}_j)$, for $j = 1, \ldots, k$. (It is often the case that $\boldsymbol{x}_{j+1} = \boldsymbol{y}_j$, corresponding to a sequence of points along a single trajectory, but this is not required.)

DMD seeks to find a matrix $\boldsymbol{A}$ such that $\boldsymbol{y}_j = \boldsymbol{A}\boldsymbol{x}_j$, in an approximate sense. DMD modes are then eigenvectors of the matrix $\boldsymbol{A}$, and DMD eigenvalues are the corresponding eigenvalues. In the present work, we are interested in obtaining a matrix $\boldsymbol{A}$ that varies in time, giving us a local linear model for the dynamics, but in the standard DMD approach, one seeks a single matrix $\boldsymbol{A}$.

Given snapshot pairs $(\boldsymbol{x}_j, \boldsymbol{y}_j)$ for $j = 1, \ldots, k$, we form matrices

$$\boldsymbol{X}_k = \begin{bmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_k \end{bmatrix}, \qquad \boldsymbol{Y}_k = \begin{bmatrix} \boldsymbol{y}_1 & \boldsymbol{y}_2 & \cdots & \boldsymbol{y}_k \end{bmatrix}, \tag{7.1}$$

which both have dimension $n \times k$. We wish to find an $n \times n$ matrix $\boldsymbol{A}_k$ such that $\boldsymbol{A}_k \boldsymbol{X}_k = \boldsymbol{Y}_k$ approximately holds; in particular, we are interested in the *overcon-*

strained problem, in which $k > n$. When the problem is *underconstrained*, the model will tend to overfit the data, and any noise present in the data will lead to poor performance of the model [11]. Note that DMD has typically been used on underconstrained problems, in which the number of states is greater than the number of snapshots (e.g., in [99]), while in this paper we consider the overconstrained case. In either case, the DMD matrix $\boldsymbol{A}_k$ is found by minimizing the cost function [81, 80]

$$J_k = \sum_{i=1}^{k} \|\boldsymbol{y}_i - \boldsymbol{A}_k \boldsymbol{x}_i\|^2 = \|\boldsymbol{Y}_k - \boldsymbol{A}_k \boldsymbol{X}_k\|_F^2, \tag{7.2}$$

where $\|\cdot\|$ denotes the Euclidean norm on vectors and $\|\cdot\|_F$ denotes the Frobenius norm on matrices. The unique minimum-norm solution to this least-squares problem is given by

$$\boldsymbol{A}_k = \boldsymbol{Y}_k \boldsymbol{X}_k^+, \tag{7.3}$$

where $\boldsymbol{X}_k^+$ denotes the Moore-Penrose pseudoinverse of $\boldsymbol{X}_k$.

Here, we shall assume that $\boldsymbol{X}_k$ has full row rank, in which case $\boldsymbol{X}_k \boldsymbol{X}_k^T$ is invertible, and

$$\boldsymbol{X}_k^+ = \boldsymbol{X}_k^T (\boldsymbol{X}_k \boldsymbol{X}_k^T)^{-1}. \tag{7.4}$$

This assumption is essential for the development of the online algorithm, as we shall see shortly. Under this assumption, the $\boldsymbol{A}_k$ given above is the unique solution that minimizes J_k. This corresponds to the case in which the number of snapshots k is large, compared with the state dimension n.

Our primary focus here is systems that may be slowly varying in time, so that the matrix $\boldsymbol{A}_k$ should evolve as k increases. In the following section, we will present an efficient algorithm for updating $\boldsymbol{A}_k$ as more data becomes available. Furthermore, if the system is time-varying, it may make sense to weight more recent snapshots more heavily than less recent snapshots. In this spirit, we will consider minimizing a

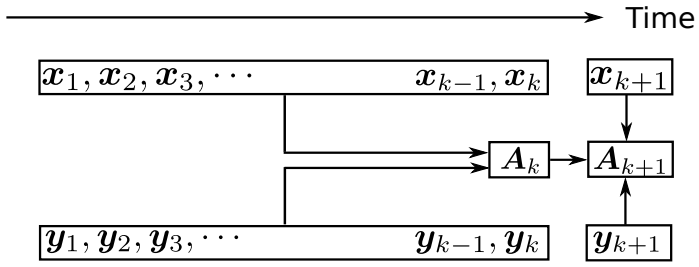


Figure 7.1: A cartoon of the online DMD setup. $\boldsymbol{A}_k$ is the optimal (least-squares) fit that maps $\boldsymbol{X}_k = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_k]$ to $\boldsymbol{Y}_k = [\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_k]$. Arrow indicates the information flow, and box denotes block of information. At time $k+1$, $\boldsymbol{A}_k$ is updated to find $\boldsymbol{A}_{k+1}$, using the information from time k, and new available snapshot pair $\boldsymbol{x}_{k+1}, \boldsymbol{y}_{k+1}$ at time $k+1$. $\boldsymbol{A}_{k+1}$ is the optimal (least-squares) fit that maps $\boldsymbol{X}_{k+1} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_k, \boldsymbol{x}_{k+1}]$ to $\boldsymbol{Y}_{k+1} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_k, \boldsymbol{y}_{k+1}]$.

modified cost function

$$\tilde{J}_k = \sum_{i=1}^{k} \rho^{k-i} \|\boldsymbol{y}_i - \boldsymbol{A}_k \boldsymbol{x}_i\|^2, \tag{7.5}$$

for some constant ρ with $0 < \rho \leq 1$. When $\rho = 1$, this cost function is the same as (7.2), and when $\rho < 1$, errors in past snapshots are discounted. Our algorithm will apply to this minimization problem as well, with only minor modifications and no increase in computational effort.

A sketch of the online DMD setup is shown in Figure 7.1. Suppose we have already computed $\boldsymbol{A}_k$ for a given dataset. As time progresses and a new pair of snapshots $(\boldsymbol{x}_{k+1}, \boldsymbol{y}_{k+1})$ becomes available, the matrix $\boldsymbol{A}_{k+1}$ may be updated according to the formula given in (7.3). If $\boldsymbol{A}_{k+1}$ is computed directly in this manner, we call this the "standard approach".

There are two drawbacks to the "standard approach". First, it requires computing the pseudoinverse of $\boldsymbol{X}_k$ whenever new snapshots are acquired, and for this reason it is computationally expensive. In addition, the method requires storing all the snapshots (i.e., storing the matrix $\boldsymbol{X}_k$), which may be challenging or impossible as the number of snapshots k increases.

7.2.2 Algorithm for online DMD

To overcome the above two shortcomings, we propose a different approach to find the solution to (7.3) in the "online setting", in which we want to compute $\boldsymbol{A}_{k+1}$ given a matrix $\boldsymbol{A}_k$ and a new pair of snapshots $(\boldsymbol{x}_{k+1}, \boldsymbol{y}_{k+1})$. The algorithm we present is based on the idea that $\boldsymbol{A}_{k+1}$ should be close to $\boldsymbol{A}_k$ in some sense. For the online updating, our approach is similar to the classical recursive least-squares estimation algorithm as formulated in [48]. Here we focus on updating the DMD matrix in real-time, but the recursive least-squares algorithm updates a vector in real-time.

First, observe that, using (7.4), we may write (7.3) as

$$\boldsymbol{A}_k = \boldsymbol{Y}_k \boldsymbol{X}_k^T (\boldsymbol{X}_k \boldsymbol{X}_k^T)^{-1} = \boldsymbol{Q}_k \boldsymbol{P}_k, \tag{7.6}$$

where $\boldsymbol{Q}_k$ and $\boldsymbol{P}_k$ are $n \times n$ matrices given by

$$\boldsymbol{Q}_k = \boldsymbol{Y}_k \boldsymbol{X}_k^T, \tag{7.7a}$$

$$\boldsymbol{P}_k = (\boldsymbol{X}_k \boldsymbol{X}_k^T)^{-1}. \tag{7.7b}$$

The condition that $\boldsymbol{X}_k$ has rank n ensures that $\boldsymbol{X}_k \boldsymbol{X}_k^T$ is invertible, and hence $\boldsymbol{P}_k$ is well defined. Note also that $\boldsymbol{P}_k$ is symmetric and strictly positive definite.

At time $k+1$, we wish to compute $\boldsymbol{A}_{k+1} = \boldsymbol{Q}_{k+1} \boldsymbol{P}_{k+1}$. Clearly, $\boldsymbol{Q}_{k+1}, \boldsymbol{P}_{k+1}$ are related to $\boldsymbol{Q}_k, \boldsymbol{P}_k$:

$$\boldsymbol{Q}_{k+1} = \boldsymbol{Y}_{k+1} \boldsymbol{X}_{k+1}^T = \begin{bmatrix} \boldsymbol{Y}_k & \boldsymbol{y}_{k+1} \end{bmatrix} \begin{bmatrix} \boldsymbol{X}_k & \boldsymbol{x}_{k+1} \end{bmatrix}^T = \boldsymbol{Y}_k \boldsymbol{X}_k^T + \boldsymbol{y}_{k+1} \boldsymbol{x}_{k+1}^T,$$

$$\boldsymbol{P}_{k+1}^{-1} = \boldsymbol{X}_{k+1} \boldsymbol{X}_{k+1}^T = \begin{bmatrix} \boldsymbol{X}_k & \boldsymbol{x}_{k+1} \end{bmatrix} \begin{bmatrix} \boldsymbol{X}_k & \boldsymbol{x}_{k+1} \end{bmatrix}^T = \boldsymbol{X}_k \boldsymbol{X}_k^T + \boldsymbol{x}_{k+1} \boldsymbol{x}_{k+1}^T.$$

Because $\boldsymbol{X}_k$ already has rank n, and adding an additional column cannot reduce the rank of a matrix, it follows that $\boldsymbol{X}_{k+1}$ also has rank n, so $\boldsymbol{P}_{k+1}$ is well defined. The

above equation shows that, given $\boldsymbol{Q}_k$ and $\boldsymbol{P}_k^{-1}$, we may find $\boldsymbol{Q}_{k+1}$ and $\boldsymbol{P}_{k+1}^{-1}$ with simple rank-1 updates:

$$\begin{aligned}\boldsymbol{Q}_{k+1} &= \boldsymbol{Q}_k + \boldsymbol{y}_{k+1}\boldsymbol{x}_{k+1}^T, \\ \boldsymbol{P}_{k+1}^{-1} &= \boldsymbol{P}_k^{-1} + \boldsymbol{x}_{k+1}\boldsymbol{x}_{k+1}^T.\end{aligned}$$

The updated DMD matrix is then given by

$$\boldsymbol{A}_{k+1} = \boldsymbol{Q}_{k+1}\boldsymbol{P}_{k+1} = (\boldsymbol{Q}_k + \boldsymbol{y}_{k+1}\boldsymbol{x}_{k+1}^T)(\boldsymbol{P}_k^{-1} + \boldsymbol{x}_{k+1}\boldsymbol{x}_{k+1}^T)^{-1}. \tag{7.8}$$

Then the problem is reduced to how to find $\boldsymbol{P}_{k+1}$ from $\boldsymbol{P}_k$ in an efficient manner. Computing the inverse directly would require $\mathcal{O}(n^3)$ operations, and would not be efficient. However, because $\boldsymbol{P}_{k+1}$ is the inverse of a rank-1 update of $\boldsymbol{P}_k^{-1}$, we may take advantage of a matrix inversion formula known as the Sherman-Morrison formula [88, 41].

Suppose $\boldsymbol{A}$ is an invertible square matrix, and $\boldsymbol{u}, \boldsymbol{v}$ are column vectors. Then $\boldsymbol{A} + \boldsymbol{u}\boldsymbol{v}^T$ is invertible if and only if $1 + \boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{u} \neq 0$, and in this case, the inverse is given by the Sherman-Morrison formula

$$(\boldsymbol{A} + \boldsymbol{u}\boldsymbol{v}^T)^{-1} = \boldsymbol{A}^{-1} - \frac{\boldsymbol{A}^{-1}\boldsymbol{u}\boldsymbol{v}^T\boldsymbol{A}^{-1}}{1 + \boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{u}}. \tag{7.9}$$

This formula is a special case of the more general matrix inversion lemma (or Woodbury formula) [106, 41].

Applying the formula to the expression for $\boldsymbol{P}_{k+1}$, we obtain

$$\boldsymbol{P}_{k+1} = (\boldsymbol{P}_k^{-1} + \boldsymbol{x}_{k+1}\boldsymbol{x}_{k+1}^T)^{-1} = \boldsymbol{P}_k - \gamma_{k+1}\boldsymbol{P}_k\boldsymbol{x}_{k+1}\boldsymbol{x}_{k+1}^T\boldsymbol{P}_k, \tag{7.10a}$$

where

$$\gamma_{k+1} = \frac{1}{1 + \boldsymbol{x}_{k+1}^T \boldsymbol{P}_k \boldsymbol{x}_{k+1}}. \tag{7.10b}$$

Note that, because $\boldsymbol{P}_k$ is positive definite, the scalar quantity $1 + \boldsymbol{x}_{k+1}^T \boldsymbol{P}_k \boldsymbol{x}_{k+1}$ is always nonzero, so the formula applies. Therefore, the updated DMD matrix may be written

$$\begin{aligned} \boldsymbol{A}_{k+1} &= (\boldsymbol{Q}_k + \boldsymbol{y}_{k+1}\boldsymbol{x}_{k+1}^T)(\boldsymbol{P}_k - \gamma_{k+1}\boldsymbol{P}_k\boldsymbol{x}_{k+1}\boldsymbol{x}_{k+1}^T\boldsymbol{P}_k) \\ &= \boldsymbol{Q}_k\boldsymbol{P}_k - \gamma_{k+1}\boldsymbol{Q}_k\boldsymbol{P}_k\boldsymbol{x}_{k+1}\boldsymbol{x}_{k+1}^T\boldsymbol{P}_k \\ &\quad + \boldsymbol{y}_{k+1}\boldsymbol{x}_{k+1}^T\boldsymbol{P}_k - \gamma_{k+1}\boldsymbol{y}_{k+1}\boldsymbol{x}_{k+1}^T\boldsymbol{P}_k\boldsymbol{x}_{k+1}\boldsymbol{x}_{k+1}^T\boldsymbol{P}_k. \end{aligned} \tag{7.11}$$

We can simplify the last two terms, since

$$\begin{aligned} \boldsymbol{y}_{k+1}\boldsymbol{x}_{k+1}^T\boldsymbol{P}_k - \gamma_{k+1}\boldsymbol{y}_{k+1}\boldsymbol{x}_{k+1}^T\boldsymbol{P}_k\boldsymbol{x}_{k+1}\boldsymbol{x}_{k+1}^T\boldsymbol{P}_k &= \gamma_{k+1}\boldsymbol{y}_{k+1}(\gamma_{k+1}^{-1} - \boldsymbol{x}_{k+1}^T\boldsymbol{P}_k\boldsymbol{x}_{k+1})\boldsymbol{x}_{k+1}^T\boldsymbol{P}_k \\ &= \gamma_{k+1}\boldsymbol{y}_{k+1}\boldsymbol{x}_{k+1}^T\boldsymbol{P}_k, \end{aligned}$$

where we have used (7.10b). Substituting into (7.11), we obtain

$$\begin{aligned} \boldsymbol{A}_{k+1} &= \boldsymbol{Q}_k\boldsymbol{P}_k - \gamma_{k+1}\boldsymbol{Q}_k\boldsymbol{P}_k\boldsymbol{x}_{k+1}\boldsymbol{x}_{k+1}^T\boldsymbol{P}_k + \gamma_{k+1}\boldsymbol{y}_{k+1}\boldsymbol{x}_{k+1}^T\boldsymbol{P}_k \\ &= \boldsymbol{A}_k - \gamma_{k+1}\boldsymbol{A}_k\boldsymbol{x}_{k+1}\boldsymbol{x}_{k+1}^T\boldsymbol{P}_k + \gamma_{k+1}\boldsymbol{y}_{k+1}\boldsymbol{x}_{k+1}^T\boldsymbol{P}_k, \end{aligned}$$

and hence

$$\boldsymbol{A}_{k+1} = \boldsymbol{A}_k + \gamma_{k+1}(\boldsymbol{y}_{k+1} - \boldsymbol{A}_k\boldsymbol{x}_{k+1})\boldsymbol{x}_{k+1}^T\boldsymbol{P}_k. \tag{7.12}$$

The above formula gives a rule for computing $\boldsymbol{A}_{k+1}$ given $\boldsymbol{A}_k, \boldsymbol{P}_k$ and the new snapshot pair $(\boldsymbol{x}_{k+1}, \boldsymbol{y}_{k+1})$. In order to use this formula recursively, we also need to compute $\boldsymbol{P}_{k+1}$ using (7.10), given $\boldsymbol{P}_k$ and $\boldsymbol{x}_{k+1}$.

There is an intuitive interpretation for the update formula (7.12). The quantity $(\boldsymbol{y}_{k+1} - \boldsymbol{A}_k\boldsymbol{x}_{k+1})$ can be considered as the prediction error from the current model $\boldsymbol{A}_k$, and the DMD matrix is updated by adding a term proportional to this error.

The updates in (7.10) and (7.12) together require only two matrix vector multiplications ($\boldsymbol{A}_k\boldsymbol{x}_{k+1}$ and $\boldsymbol{P}_k\boldsymbol{x}_{k+1}$, since $\boldsymbol{P}_k$ is symmetric), and two vector outer products, for a total of $4n^2$ floating-point multiplies. This is much more efficient than applying the standard DMD algorithm, which involves a singular value decomposition or pseudoinverse, and requires $O(kn^2)$ multiplies, where $k > n$. In our approach, two $n \times n$ matrices need to be stored ($\boldsymbol{A}_k$ and $\boldsymbol{P}_k$), but the large $n \times k$ snapshot matrices ($\boldsymbol{X}_k, \boldsymbol{Y}_k$) do not need to be stored.

It is worth emphasizing that the update formulas (7.10) and (7.12) compute the DMD matrix $\boldsymbol{A}_{k+1} = \boldsymbol{Y}_{k+1}\boldsymbol{X}_{k+1}^{+}$ exactly (up to machine precision). That is, with exact arithmetic, our formulas give the same results as the standard DMD algorithm. The matrix $\boldsymbol{P}_k$ does involve "squaring up" the matrix $\boldsymbol{X}_k$, which could in principle lead to difficulties with numerical stability for ill-conditioned problems [110, 1]. However, we have not encountered problems with numerical stability in the examples we have tried (see section 7.5).

Initialization The algorithm described above needs a starting point. In particular, to apply the updates (7.10) and (7.12), one needs the matrices $\boldsymbol{P}_k$ and $\boldsymbol{A}_k$ at timestep k. The initialization technique is similar to the initialization of the recursive least-squares estimation described in [48]. Two practical approaches are discussed below. The most straightforward way to initialize the algorithm is to first collect at least n snapshots (more precisely, enough snapshots so that $\boldsymbol{X}_k$ as defined in (7.1) has rank n), and then compute $\boldsymbol{P}_k$ and $\boldsymbol{A}_k$ using the standard DMD algorithm, from (7.6) and (7.7):

$$\boldsymbol{A}_k = \boldsymbol{Y}_k\boldsymbol{X}_k^{+}, \qquad \boldsymbol{P}_k = (\boldsymbol{X}_k\boldsymbol{X}_k^{T})^{-1}. \tag{7.13}$$

If for some reason this is not desirable, then an alternative approach is to initialize $\boldsymbol{A}_0$ to a random matrix (e.g., the zero matrix), and set $\boldsymbol{P}_0 = \alpha\boldsymbol{I}$, where α is a large

positive scalar. Then in the limit as $\alpha \to \infty$, the matrices $\boldsymbol{P}_k, \boldsymbol{A}_k$ computed by the updates (7.10) and (7.12) converge to the true values given by (7.13).

Multiple snapshots In our method, the DMD matrix $\boldsymbol{A}_k$ gets updated at every time step when a new snapshot pair becomes available. In principle, one could update the DMD matrix less frequently (for instance every 10 time steps). The above derivation can be appropriately modified to handle this case, using the more general Woodbury formula (see (7.23)) [106, 41]. However, if s is the number of new snapshots to be incorporated, the computational cost of a single rank-s update is roughly the same as applying the rank-1 formula s times, so there does not appear to be a benefit to incorporating multiple snapshots at once.

Extensions As is the case for most DMD algorithms (including streaming DMD), the online DMD algorithm described above applies more generally to extended DMD (EDMD) [104], simply replacing the state observations $\boldsymbol{x}_k, \boldsymbol{y}_k$ by the corresponding vectors of observables. In addition, the algorithm can be used for real-time online system identification, including both linear and nonlinear system identification, as we shall discuss in section 7.4.

Summary To summarize, the algorithm proceeds as follows:

1. Collect k snapshot pairs $(\boldsymbol{x}_j, \boldsymbol{y}_j)$, $j = 1, \ldots, k$, where $k > n$ is large enough so that $\operatorname{Rank} \boldsymbol{X}_k = n$ (where $\boldsymbol{X}_k$ is given by (7.1)).

2. Compute $\boldsymbol{A}_k$ and $\boldsymbol{P}_k$ from (7.13).

3. When a new snapshot pair $(\boldsymbol{x}_{k+1}, \boldsymbol{y}_{k+1})$ becomes available, update $\boldsymbol{A}_k$ and $\boldsymbol{P}_k$ according to (7.12) and (7.10).

Implementations of this algorithm in both Matlab and Python are publicly available at [112].

7.2.3 Weighted online DMD

As mentioned previously, the online DMD algorithm described above is ideally suited to cases for which the system is varying in time, so that we want to revise our estimate of the DMD matrix $\boldsymbol{A}_k$ in real-time. In such a situation, we might wish to place more weight on recent snapshots, and gradually "forget" the older snapshots, by minimizing a cost function of the form (7.5) instead of the original cost function (7.2). This weighting scheme is analogous to that used in real-time least-squares approximation [48]. This idea may also be used with streaming DMD, and in fact has been considered before (the conference presentation [44] implemented such a "forgetting factor" with streaming DMD, although it did not appear in the associated paper). It turns out that the online DMD algorithm can be adapted to minimize the cost function (7.5) with only minor modifications to the algorithm.

We now consider the cost function

$$\tilde{J}_k = \sum_{i=1}^{k} \rho^{k-i} \|\boldsymbol{y}_i - \boldsymbol{A}_k \boldsymbol{x}_i\|^2, \qquad 0 < \rho \leq 1,$$

where ρ is the weighting factor. For instance, if we wish our snapshots to have a "half-life" of m samples, then we could choose $\rho = 2^{-1/m}$. In practice, ρ should be chosen according to how fast the dynamics are changing. There is a tradeoff between faster tracking and noise filtering: a smaller ρ will result in faster tracking, while inevitably making the identified model more sensitive to noise in the data (since we are forgetting old samples). For convenience, let us take $\rho = \sigma^2$ where $0 < \sigma \leq 1$, and write the cost function as

$$\tilde{J}_k = \sum_{i=1}^{k} \|\sigma^{k-i} \boldsymbol{y}_i - \boldsymbol{A}_k \sigma^{k-i} \boldsymbol{x}_i\|^2.$$

If we define matrices based on scaled versions of the snapshots, as

$$\tilde{\boldsymbol{X}}_k = \begin{bmatrix} \sigma^{k-1}\boldsymbol{x}_1 & \sigma^{k-2}\boldsymbol{x}_2 & \cdots & \boldsymbol{x}_k \end{bmatrix},$$
$$\tilde{\boldsymbol{Y}}_k = \begin{bmatrix} \sigma^{k-1}\boldsymbol{y}_1 & \sigma^{k-2}\boldsymbol{y}_2 & \cdots & \boldsymbol{y}_k \end{bmatrix},$$

then the cost function can be written as

$$\tilde{J}_k = \|\tilde{\boldsymbol{Y}}_k - \boldsymbol{A}_k\tilde{\boldsymbol{X}}_k\|_F^2.$$

The unique least-squares solution that minimizes this cost function (assuming $\tilde{\boldsymbol{X}}_k$ has full row rank) is given by

$$\boldsymbol{A}_k = \tilde{\boldsymbol{Y}}_k\tilde{\boldsymbol{X}}_k^+ = \tilde{\boldsymbol{Y}}_k\tilde{\boldsymbol{X}}_k^T(\tilde{\boldsymbol{X}}_k\tilde{\boldsymbol{X}}_k^T)^{-1} = \tilde{\boldsymbol{Q}}_k\tilde{\boldsymbol{P}}_k,$$

where we define

$$\tilde{\boldsymbol{Q}}_k = \tilde{\boldsymbol{Y}}_k\tilde{\boldsymbol{X}}_k^T,$$
$$\tilde{\boldsymbol{P}}_k = (\tilde{\boldsymbol{X}}_k\tilde{\boldsymbol{X}}_k^T)^{-1}.$$

At step $k+1$, we wish to compute $\boldsymbol{A}_{k+1} = \tilde{\boldsymbol{Q}}_{k+1}\tilde{\boldsymbol{P}}_{k+1}$. We write down $\tilde{\boldsymbol{X}}_{k+1}, \tilde{\boldsymbol{Y}}_{k+1}$ explicitly as

$$\tilde{\boldsymbol{X}}_{k+1} = \begin{bmatrix} \sigma^k\boldsymbol{x}_1 & \sigma^{k-1}\boldsymbol{x}_2 & \cdots & \sigma\boldsymbol{x}_k & \boldsymbol{x}_{k+1} \end{bmatrix} = \begin{bmatrix} \sigma\tilde{\boldsymbol{X}}_k & \boldsymbol{x}_{k+1} \end{bmatrix},$$
$$\tilde{\boldsymbol{Y}}_{k+1} = \begin{bmatrix} \sigma^k\boldsymbol{y}_1 & \sigma^{k-1}\boldsymbol{y}_2 & \cdots & \sigma\boldsymbol{y}_k & \boldsymbol{y}_{k+1} \end{bmatrix} = \begin{bmatrix} \sigma\tilde{\boldsymbol{Y}}_k & \boldsymbol{y}_{k+1} \end{bmatrix}.$$

Therefore, $\tilde{\boldsymbol{Q}}_{k+1}$ can be written

$$\begin{aligned}\tilde{\boldsymbol{Q}}_{k+1} = \tilde{\boldsymbol{Y}}_{k+1}\tilde{\boldsymbol{X}}_{k+1}^T &= \begin{bmatrix}\sigma\tilde{\boldsymbol{Y}}_k & \boldsymbol{y}_{k+1}\end{bmatrix}\begin{bmatrix}\sigma\tilde{\boldsymbol{X}}_k & \boldsymbol{x}_{k+1}\end{bmatrix}^T \\ &= \sigma^2\tilde{\boldsymbol{Y}}_k\tilde{\boldsymbol{X}}_k^T + \boldsymbol{y}_{k+1}\boldsymbol{x}_{k+1}^T \\ &= \rho\tilde{\boldsymbol{Q}}_k + \boldsymbol{y}_{k+1}\boldsymbol{x}_{k+1}^T,\end{aligned}$$

and similarly

$$\tilde{\boldsymbol{P}}_{k+1}^{-1} = \rho\tilde{\boldsymbol{P}}_k^{-1} + \boldsymbol{x}_{k+1}\boldsymbol{x}_{k+1}^T. \tag{7.14}$$

The updated DMD matrix is then given by

$$\boldsymbol{A}_{k+1} = \tilde{\boldsymbol{Q}}_{k+1}\tilde{\boldsymbol{P}}_{k+1} = (\rho\tilde{\boldsymbol{Q}}_k + \boldsymbol{y}_{k+1}\boldsymbol{x}_{k+1}^T)(\rho\tilde{\boldsymbol{P}}_k^{-1} + \boldsymbol{x}_{k+1}\boldsymbol{x}_{k+1}^T)^{-1}.$$

As before, we can apply the Sherman-Morrison formula (7.9) to (7.14) and obtain

$$\tilde{\boldsymbol{P}}_{k+1} = \frac{\tilde{\boldsymbol{P}}_k}{\rho} - \gamma_{k+1}\frac{\tilde{\boldsymbol{P}}_k}{\rho}\boldsymbol{x}_{k+1}\boldsymbol{x}_{k+1}^T\frac{\tilde{\boldsymbol{P}}_k}{\rho},$$

where

$$\gamma_{k+1} = \frac{1}{1 + \boldsymbol{x}_{k+1}^T(\tilde{\boldsymbol{P}}_k/\rho)\boldsymbol{x}_{k+1}}.$$

Let us rescale $\tilde{\boldsymbol{P}}_k$, and define

$$\hat{\boldsymbol{P}}_k = \frac{\tilde{\boldsymbol{P}}_k}{\rho} = \frac{1}{\rho}(\tilde{\boldsymbol{X}}_k\tilde{\boldsymbol{X}}_k^T)^{-1}.$$

Then after some manipulation, the formula for $\boldsymbol{A}_{k+1}$ becomes

$$\boldsymbol{A}_{k+1} = \boldsymbol{A}_k + \gamma_{k+1}(\boldsymbol{y}_{k+1} - \boldsymbol{A}_k\boldsymbol{x}_{k+1})\boldsymbol{x}_{k+1}^T\hat{\boldsymbol{P}}_k, \tag{7.15}$$

where

$$\hat{\boldsymbol{P}}_{k+1} = \frac{1}{\rho}(\hat{\boldsymbol{P}}_k - \gamma_{k+1}\hat{\boldsymbol{P}}_k \boldsymbol{x}_{k+1}\boldsymbol{x}_{k+1}^T \hat{\boldsymbol{P}}_k), \tag{7.16a}$$

$$\gamma_{k+1} = \frac{1}{1 + \boldsymbol{x}_{k+1}^T \hat{\boldsymbol{P}}_k \boldsymbol{x}_{k+1}}. \tag{7.16b}$$

Observe that the update (7.15) for $\boldsymbol{A}_{k+1}$ is identical to the update (7.12) from the previous section, with $\boldsymbol{P}_k$ replaced by $\hat{\boldsymbol{P}}_k$, and the update rule (7.16) for $\hat{\boldsymbol{P}}_{k+1}$ differs from (7.10) only by a factor of ρ. When $\rho = 1$, of course, the above formulas are identical to those given in section 7.2.2.

7.3 Windowed dynamic mode decomposition

In section 7.2.3, we presented a method for gradually "forgetting" older snapshots, by giving them less weight in a cost function. In this section, we discuss an alternative method, which uses a hard cut-off: in particular, we consider a "window" containing only the most recent snapshots, for instance as used in [38, 61].

7.3.1 The problem

If the dynamics are slowly varying with time, we may wish to use only the most recent snapshots to identify the dynamics. Here, we consider the case where we use only a fixed "window" containing the most recent snapshots. Here, we present an "online" algorithm to compute windowed DMD efficiently, again using low-rank updates, as in the previous section. We refer to the resulting method as "windowed dynamic mode decomposition" (windowed DMD).

At time t_k, suppose we have access to past snapshot pairs $\{(\boldsymbol{x}_j, \boldsymbol{y}_j)\}_{j=k-w+1}^{k}$ in a finite time window of size w. We would like to fit a linear model $\boldsymbol{A}_k$, such that

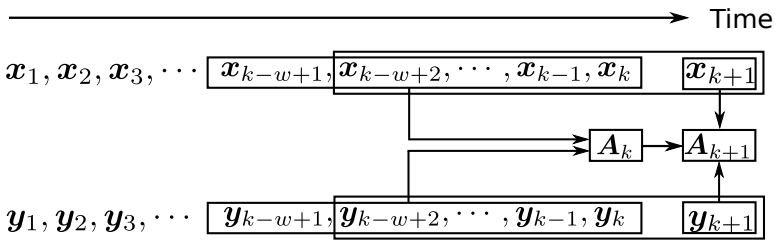


Figure 7.2: A cartoon of the windowed DMD setup. At time k, $\boldsymbol{A}_k$ depends only on the w most recent snapshots. At time $k+1$, one new snapshot is added, and the oldest snapshot is dropped.

$\boldsymbol{y}_j = \boldsymbol{A}_k \boldsymbol{x}_j$ (at least approximately) for all j in this window. Let

$$\boldsymbol{X}_k = \begin{bmatrix} \boldsymbol{x}_{k-w+1} & \boldsymbol{x}_{k-w+2} & \cdots & \boldsymbol{x}_k \end{bmatrix}, \qquad \boldsymbol{Y}_k = \begin{bmatrix} \boldsymbol{y}_{k-w+1} & \boldsymbol{y}_{k-w+2} & \cdots & \boldsymbol{y}_k \end{bmatrix}, \quad (7.17)$$

both $n \times w$ matrices. Then we seek an $n \times n$ matrix $\boldsymbol{A}_k$ such that $\boldsymbol{A}_k \boldsymbol{X}_k = \boldsymbol{Y}_k$ approximately holds. More precisely (as explained in section 7.2.1), the DMD matrix $\boldsymbol{A}_k$ is found by minimizing

$$J_k = \|\boldsymbol{Y}_k - \boldsymbol{A}_k \boldsymbol{X}_k\|_F^2. \quad (7.18)$$

As before, we assume that the rank of $\boldsymbol{X}_k$ is $n \leq w$, so that there is a unique solution to this least-squares problem, given by

$$\boldsymbol{A}_k = \boldsymbol{Y}_k \boldsymbol{X}_k^+, \quad (7.19)$$

where $\boldsymbol{X}_k^+ = \boldsymbol{X}_k^T (\boldsymbol{X}_k \boldsymbol{X}_k^T)^{-1}$ is the Moore-Penrose pseudoinverse of $\boldsymbol{X}_k$. (Note, in particular, that we require that the window size w be at least as large as the state dimension n, so that $\boldsymbol{X}_k \boldsymbol{X}_k^T$ is invertible.)

A sketch of the windowed DMD setup is shown in Figure 7.2. As time progresses, we can update $\boldsymbol{A}_k$ according to the formula given in (7.19). However, evaluating (7.19) involves computing a new pseudoinverse and a matrix multiplication, which are costly

operations. We may compute this update more efficiently using an approach similar to that in the previous section, as we describe below.

7.3.2 Algorithm for windowed DMD

As in the approach presented in section 7.2.2, observe that (7.19) can be written as

$$\boldsymbol{A}_k = \boldsymbol{Y}_k \boldsymbol{X}_k^+ = \boldsymbol{Y}_k \boldsymbol{X}_k^T (\boldsymbol{X}_k \boldsymbol{X}_k^T)^{-1} = \boldsymbol{Q}_k \boldsymbol{P}_k, \tag{7.20}$$

where

$$\boldsymbol{Q}_k = \boldsymbol{Y}_k \boldsymbol{X}_k^T = \sum_{i=k-w+1}^{k} \boldsymbol{y}_i \boldsymbol{x}_i^T,$$

$$\boldsymbol{P}_k = (\boldsymbol{X}_k \boldsymbol{X}_k^T)^{-1} = \left(\sum_{i=k-w+1}^{k} \boldsymbol{x}_i \boldsymbol{x}_i^T \right)^{-1}. \tag{7.21}$$

where $\boldsymbol{Q}_k$ and $\boldsymbol{P}_k$ are $n \times n$ matrices. The condition that $\boldsymbol{X}_k$ has rank n ensures that $\boldsymbol{X}_k \boldsymbol{X}_k^T$ is invertible, so $\boldsymbol{P}_k$ is well defined.

At step $k+1$, we need to compute $\boldsymbol{A}_{k+1} = \boldsymbol{Q}_{k+1} \boldsymbol{P}_{k+1}$. Clearly, $\boldsymbol{Q}_{k+1}, \boldsymbol{P}_{k+1}$ are related to $\boldsymbol{Q}_k, \boldsymbol{P}_k$. To show this, we write them down explicitly as

$$\boldsymbol{Q}_{k+1} = \boldsymbol{Y}_{k+1} \boldsymbol{X}_{k+1}^T = \sum_{i=k-w+2}^{k+1} \boldsymbol{y}_i \boldsymbol{x}_i^T = \boldsymbol{Q}_k - \boldsymbol{y}_{k-w+1} \boldsymbol{x}_{k-w+1}^T + \boldsymbol{y}_{k+1} \boldsymbol{x}_{k+1}^T,$$

$$\boldsymbol{P}_{k+1}^{-1} = \boldsymbol{X}_{k+1} \boldsymbol{X}_{k+1}^T = \sum_{i=k-w+2}^{k+1} \boldsymbol{x}_i \boldsymbol{x}_i^T = \boldsymbol{P}_k^{-1} - \boldsymbol{x}_{k-w+1} \boldsymbol{x}_{k-w+1}^T + \boldsymbol{x}_{k+1} \boldsymbol{x}_{k+1}^T.$$

There is an intuitive interpretation to this relationship: $\boldsymbol{Q}_k, \boldsymbol{P}_k$ forgets the oldest snapshot and incorporates the newest snapshot, and gets updated into $\boldsymbol{Q}_{k+1}, \boldsymbol{P}_{k+1}$. As in section 7.2.2, where we used the Sherman-Morrison formula (7.9) to update $\boldsymbol{P}_k$, we may use a similar approach to update $\boldsymbol{P}_k$ in this case.

Letting

$$\boldsymbol{U} = \begin{bmatrix} \boldsymbol{x}_{k-w+1} & \boldsymbol{x}_{k+1} \end{bmatrix}, \qquad \boldsymbol{V} = \begin{bmatrix} \boldsymbol{y}_{k-w+1} & \boldsymbol{y}_{k+1} \end{bmatrix}, \qquad \boldsymbol{C} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix},$$

we may write $\boldsymbol{Q}_{k+1}, \boldsymbol{P}_{k+1}$ as

$$\boldsymbol{Q}_{k+1} = \boldsymbol{Q}_k + \boldsymbol{V}\boldsymbol{C}\boldsymbol{U}^T,$$

$$\boldsymbol{P}_{k+1}^{-1} = \boldsymbol{P}_k^{-1} + \boldsymbol{U}\boldsymbol{C}\boldsymbol{U}^T,$$

therefore

$$\boldsymbol{A}_{k+1} = \boldsymbol{Q}_{k+1}\boldsymbol{P}_{k+1} = (\boldsymbol{Q}_k + \boldsymbol{V}\boldsymbol{C}\boldsymbol{U}^T)(\boldsymbol{P}_k^{-1} + \boldsymbol{U}\boldsymbol{C}\boldsymbol{U}^T)^{-1}. \tag{7.22}$$

Now, the matrix inversion lemma (or Woodbury formula) [106, 41] states that

$$(\boldsymbol{A} + \boldsymbol{U}\boldsymbol{C}\boldsymbol{V})^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{U}(\boldsymbol{C}^{-1} + \boldsymbol{V}\boldsymbol{A}^{-1}\boldsymbol{U})^{-1}\boldsymbol{V}\boldsymbol{A} \tag{7.23}$$

whenever $\boldsymbol{A}$, $\boldsymbol{C}$, and $\boldsymbol{A}+\boldsymbol{U}\boldsymbol{C}\boldsymbol{V}$ are invertible. Applying this formula to our expression for $\boldsymbol{P}_{k+1}$, we have

$$\boldsymbol{P}_{k+1} = \boldsymbol{P}_k - \boldsymbol{P}_k\boldsymbol{U}\boldsymbol{\Gamma}_{k+1}\boldsymbol{U}^T\boldsymbol{P}_k, \tag{7.24a}$$

where

$$\boldsymbol{\Gamma}_{k+1} = (\boldsymbol{C}^{-1} + \boldsymbol{U}^T\boldsymbol{P}_k\boldsymbol{U})^{-1}. \tag{7.24b}$$

Substituting back into (7.22), we obtain

$$\begin{aligned} \boldsymbol{A}_{k+1} &= (\boldsymbol{Q}_k + \boldsymbol{V}\boldsymbol{C}\boldsymbol{U}^T)(\boldsymbol{P}_k - \boldsymbol{P}_k\boldsymbol{U}\boldsymbol{\Gamma}_{k+1}\boldsymbol{U}^T\boldsymbol{P}_k) \\ &= \boldsymbol{Q}_k\boldsymbol{P}_k - \boldsymbol{Q}_k\boldsymbol{P}_k\boldsymbol{U}\boldsymbol{\Gamma}_{k+1}\boldsymbol{U}^T\boldsymbol{P}_k \\ &\quad + \boldsymbol{V}\boldsymbol{C}\boldsymbol{U}^T\boldsymbol{P}_k - \boldsymbol{V}\boldsymbol{C}\boldsymbol{U}^T\boldsymbol{P}_k\boldsymbol{U}\boldsymbol{\Gamma}_{k+1}\boldsymbol{U}^T\boldsymbol{P}_k. \end{aligned} \tag{7.25}$$

The last two terms simplify, since

$$\begin{aligned}\boldsymbol{V}\boldsymbol{C}\boldsymbol{U}^T\boldsymbol{P}_k - \boldsymbol{V}\boldsymbol{C}\boldsymbol{U}^T\boldsymbol{P}_k\boldsymbol{U}\boldsymbol{\Gamma}_{k+1}\boldsymbol{U}^T\boldsymbol{P}_k &= \boldsymbol{V}\boldsymbol{C}(\boldsymbol{\Gamma}_{k+1}^{-1} - \boldsymbol{U}^T\boldsymbol{P}_k\boldsymbol{U})\boldsymbol{\Gamma}_{k+1}\boldsymbol{U}^T\boldsymbol{P}_k \\ &= \boldsymbol{V}\boldsymbol{C}\boldsymbol{C}^{-1}\boldsymbol{\Gamma}_{k+1}\boldsymbol{U}^T\boldsymbol{P}_k = \boldsymbol{V}\boldsymbol{\Gamma}_{k+1}\boldsymbol{U}^T\boldsymbol{P}_k,\end{aligned}$$

where we have used (7.24b). Substituting into (7.25), we obtain

$$\begin{aligned}\boldsymbol{A}_{k+1} &= \boldsymbol{Q}_k\boldsymbol{P}_k - \boldsymbol{Q}_k\boldsymbol{P}_k\boldsymbol{U}\boldsymbol{\Gamma}_{k+1}\boldsymbol{U}^T\boldsymbol{P}_k + \boldsymbol{V}\boldsymbol{\Gamma}_{k+1}\boldsymbol{U}^T\boldsymbol{P}_k \\ &= \boldsymbol{A}_k - \boldsymbol{A}_k\boldsymbol{U}\boldsymbol{\Gamma}_{k+1}\boldsymbol{U}^T\boldsymbol{P}_k + \boldsymbol{V}\boldsymbol{\Gamma}_{k+1}\boldsymbol{U}^T\boldsymbol{P}_k,\end{aligned}$$

and hence

$$\boldsymbol{A}_{k+1} = \boldsymbol{A}_k + (\boldsymbol{V} - \boldsymbol{A}_k\boldsymbol{U})\boldsymbol{\Gamma}_{k+1}\boldsymbol{U}^T\boldsymbol{P}_k. \tag{7.26}$$

Notice the similarity between this expression with the updating formula (7.12) for online DMD. $\boldsymbol{\Gamma}_{k+1}$ is the matrix version of γ_{k+1} in (7.10b). The matrix $(\boldsymbol{V} - \boldsymbol{A}_k\boldsymbol{U})$ can also be considered as the prediction error based on current model $\boldsymbol{A}_k$, and the correction to DMD matrix is proportional to this error term.

The updates in (7.26), (7.24) require two products of $n \times n$ and $n \times 2$ matrices (to compute $\boldsymbol{A}_k\boldsymbol{U}$ and $\boldsymbol{P}_k\boldsymbol{U}$, since $\boldsymbol{P}_k$ is symmetric), and two products of $n \times 2$ and $2 \times n$ matrices, for a total of $8n^2$ multiplies. This windowed DMD approach is much more efficient than the standard DMD approach, solving (7.20) directly ($\mathcal{O}(wn^2)$ multiplies, with $w \geq n$). Windowed DMD can be initialized in the same manner as online DMD, discussed in section 7.2.2.

In order to implement windowed DMD, we need to store two $n \times n$ matrices $(\boldsymbol{A}_k, \boldsymbol{P}_k)$, as well as the w most recent snapshots. Thus, the storage required is more than in online DMD, or the weighted online DMD approach discussed in section 7.2.3, which also provides a mechanism for "forgetting" older snapshots. It is worth pointing out that the update formulas (7.24), (7.26) give the exact solution $\boldsymbol{A}_{k+1} = \boldsymbol{Y}_{k+1}\boldsymbol{X}_{k+1}^+$ from (7.22), without approximation.

Larger window stride size We can in principle move more than one step for windowed DMD, i.e., forgetting multiple snapshots and remembering multiple snapshots. If we would like to move the sliding window for s steps ($s < n/2$), then after similar derivations, we can show that the computational cost is $8sn^2$ multiplies, which is the same as applying the rank-2 formulas s times. Therefore, there is no obvious advantage to incorporating multiple snapshots at one time.

Extensions Similar to online DMD, we can also incorporate an exponential weighting factor into windowed DMD. In particular, consider the cost function as

$$\tilde{J}_k = \sum_{i=k-w+1}^{k} \rho^{k-i} \|\boldsymbol{y}_i - \boldsymbol{A}_k \boldsymbol{x}_i\|^2, \qquad 0 < \rho \leq 1,$$

where ρ is the weighting factor. Then, proceeding as in section 7.2.3, we obtain the update formulas

$$\boldsymbol{A}_{k+1} = \boldsymbol{A}_k + (\boldsymbol{V} - \boldsymbol{A}_k \boldsymbol{U}) \tilde{\boldsymbol{\Gamma}}_{k+1} \boldsymbol{U}^T \hat{\boldsymbol{P}}_k, \tag{7.27}$$

$$\hat{\boldsymbol{P}}_{k+1} = \frac{1}{\rho} (\hat{\boldsymbol{P}}_k - \hat{\boldsymbol{P}}_k \boldsymbol{U} \tilde{\boldsymbol{\Gamma}}_{k+1} \boldsymbol{U}^T \hat{\boldsymbol{P}}_k), \tag{7.28a}$$

where

$$\tilde{\boldsymbol{\Gamma}}_{k+1} = (\tilde{\boldsymbol{C}}^{-1} + \boldsymbol{U}^T \hat{\boldsymbol{P}}_k \boldsymbol{U})^{-1}, \qquad \tilde{\boldsymbol{C}} = \begin{bmatrix} -\rho^w & 0 \\ 0 & 1 \end{bmatrix}. \tag{7.28b}$$

As with the online DMD algorithm, the above windowed DMD algorithm applies generally to extended DMD (EDMD) [104] as well, if $\boldsymbol{x}_k, \boldsymbol{y}_k$ are simply replaced by the observable vector of the states. In addition, the algorithm can be used for real-time online system identification, including both linear and nonlinear system identification, as discussed in section 7.4.

Summary To summarize, the algorithm proceeds as follows:

1. Collect w snapshot pairs $(\boldsymbol{x}_j, \boldsymbol{y}_j)$, $j = 1, \ldots, w$, where $w \geq n$ is large enough so that $\operatorname{Rank} \boldsymbol{X}_k = n$ (where $\boldsymbol{X}_k$ is given by (7.17)).

2. Compute $\boldsymbol{A}_k$ and $\boldsymbol{P}_k$ from (7.13), where $\boldsymbol{X}_k, \boldsymbol{Y}_k$ is given by (7.17).

3. When a new snapshot pair $(\boldsymbol{x}_{k+1}, \boldsymbol{y}_{k+1})$ becomes available, update $\boldsymbol{A}_k$ and $\boldsymbol{P}_k$ according to (7.26) and (7.24).

Implementations of this algorithm in both Matlab and Python are publicly available at [112].

7.4 Online system identification

As previously mentioned, the online and windowed DMD algorithms discussed above can be generalized to online system identification with control in a straightforward manner. For a review of system identification methods, see [5].

7.4.1 Online linear system identification

Dynamic mode decomposition can be used for system identification, as shown in [75]. Suppose we are interested in identifying a (discrete-time) linear system given by

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k, \tag{7.29}$$

where $\boldsymbol{x}_k \in \mathbb{R}^n, \boldsymbol{u}_k \in \mathbb{R}^p$ are the states and control input respectively, $\boldsymbol{A} \in \mathbb{R}^{n \times n}, \boldsymbol{B} \in \mathbb{R}^{n \times p}$.

At time k, assume that we have access to $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_{k+1}$ and $\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_k$. Letting

$$\tilde{\boldsymbol{Y}}_k = \begin{bmatrix} \boldsymbol{x}_2 & \boldsymbol{x}_3 & \cdots & \boldsymbol{x}_{k+1} \end{bmatrix}, \qquad \tilde{\boldsymbol{X}}_k = \begin{bmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_k \\ \boldsymbol{u}_1 & \boldsymbol{u}_2 & \cdots & \boldsymbol{u}_k \end{bmatrix}, \qquad \tilde{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \end{bmatrix},$$

we may write (7.29) in the form

$$\tilde{\boldsymbol{Y}}_k = \tilde{\boldsymbol{A}}\tilde{\boldsymbol{X}}_k. \tag{7.30}$$

The matrices $\boldsymbol{A}, \boldsymbol{B}$ may then be found by minimizing the cost function

$$J_k = \|\tilde{\boldsymbol{Y}}_k - \tilde{\boldsymbol{A}}_k\tilde{\boldsymbol{X}}_k\|_F^2. \tag{7.31}$$

As before, the solution is given by

$$\tilde{\boldsymbol{A}}_k = \tilde{\boldsymbol{Y}}_k\tilde{\boldsymbol{X}}_k^+. \tag{7.32}$$

At time $k+1$, we add a new column to $\tilde{\boldsymbol{X}}_k$ and $\tilde{\boldsymbol{Y}}_k$, and we would like to update $\tilde{\boldsymbol{A}}_{k+1}$ using our previous knowledge of $\tilde{\boldsymbol{A}}_k$. Using the same approach as in section 7.2, it is straightforward to extend the online DMD and windowed DMD algorithms to this case. In particular, the square matrix $\boldsymbol{A}_k$ from section 7.2 is replaced by the rectangular matrix $\tilde{\boldsymbol{A}}_k$ defined above, and the vector $\boldsymbol{x}_k$ in the formulas in section 7.2 is replaced by the column vector

$$\begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{u}_k \end{bmatrix}.$$

7.4.2 Online nonlinear system identification

The efficient online/windowed DMD algorithms apply to nonlinear system identification as well. In general, nonlinear system identification is a challenging problem; see [71] for an overview. Some interesting methods are to use linear-parameter-varying models [101, 45], or to consider a large dictionary of potential nonlinear functions, and exploit sparsity to select a small subset [14].

Suppose we are interested in identifying a nonlinear system

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k)$$

directly from data, where $\boldsymbol{x}_k \in \mathbb{R}^n, \boldsymbol{u}_k \in \mathbb{R}^p$ are the state vector and control input respectively. The specific form of nonlinearity is unknown, but in order to proceed, we have to make some assumptions about the nonlinear form. Assume that we have q (nonlinear) observables $z_i(\boldsymbol{x}, \boldsymbol{u}), i = 1, 2, \cdots, q$, such that the underlying dynamics can be approximately described by

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{z}_k, \tag{7.33}$$

where $\boldsymbol{A} \in \mathbb{R}^{n \times q}$, and

$$\boldsymbol{z}_k = \begin{bmatrix} z_1(\boldsymbol{x}_k, \boldsymbol{u}_k) & z_2(\boldsymbol{x}_k, \boldsymbol{u}_k) & \cdots & z_q(\boldsymbol{x}_k, \boldsymbol{u}_k) \end{bmatrix}^T.$$

To illustrate how this representation works, we take $\boldsymbol{x} \in \mathbb{R}, \boldsymbol{u} \in \mathbb{R}$ for example, and assume the nonlinear dynamics is given by

$$x_{k+1} = a_1 x_k + a_2 x_k^2 + a_3 u_k + a_4 u_k^2 + a_5 x_k u_k.$$

Then by setting $z_1(x, u) = x, z_2(x, u) = x^2, z_3(x, u) = u, z_4(x, u) = u^2, z_5(x, u) = xu$, we can write the dynamics in the form (7.33), with

$$\boldsymbol{A} = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 \end{bmatrix}.$$

Note that in the above, the state x_k still evolves nonlinearly (i.e., x_{k+1} depends in a nonlinear way on x_k and u_k), but we are able to identify the coefficients a_k using linear regression (i.e., finding the matrix $\boldsymbol{A}$ in (7.33)).

This approach is related to Carleman linearization [9, 31], although in Carleman linearization, the goal is to find a true linear representation of the dynamics in a higher-dimensional state space, and for most nonlinear systems of practical interest, it is not possible to obtain a finite-dimensional linear representation. This is also related to extended DMD [104] and kernel DMD [105] where Koopman eigenfunctions are used to determine coordinates in which the nonlinear system becomes linear. Our approach is to define a collection of observables in order to write the original nonlinear system in a linear fashion. It is generally difficult to determine what observables to use, and sparsity-promoting system identification [14] is one possible method. If one knows something about the form of the nonlinearity (e.g., the nonlinear terms are quadratic), then this can inform the choice of observables (e.g., including all quadratic couplings of the states).

In summary, by assuming a particular form of the nonlinearity, we can find the coefficients of a nonlinear system using the same techniques as used in linear system identification, writing the nonlinear system in the form (7.33).

7.5 Application and results

In this section, we illustrate the methods on a number of examples, first showing results for simple benchmark problems, and then using data from a wind tunnel experiment.

7.5.1 Benchmarks

We now present a study of the computational time of various DMD algorithms. Two benchmark tasks are considered here. In the first task, we wish to know the DMD matrix only at the final time step, at which point we have access to all of the data. In the second task, we wish to compute the DMD matrix at each time, whenever a

new snapshot is acquired. The first task thus represents the standard approach to computing the DMD matrix, while the second task applies to situations where the system is time-varying, and we wish to update the DMD matrix in real-time.

Asymptotic cost First, we examine how the various algorithms scale with the state dimension n and the number of snapshots m, for the two tasks described above. In particular, we are concerned with the over-constrained case in which $n < m$. For the standard algorithm, in which the DMD matrix is computed directly using (7.3), one must compute an $n \times m$ pseudoinverse and an $n \times m$, $m \times n$ matrix multiplication. For the first task, the computational cost (measured by the number of multiplies) is thus

$$T_{\text{standard}} = \mathcal{O}(nm \min(m, n) + mn^2) = \mathcal{O}(mn^2).$$

For the second task, in which we compute the DMD matrix at each time, we refer to the standard algorithm as "batch DMD", since the snapshots are processed all in one batch. The method is initialized and applied after m_0 snapshots are gathered (and in the examples below, we will take $m_0 = n$), so the computational cost is

$$T_{\text{batch}} = \mathcal{O}\left(\sum_{k=m_0}^{m} (nk \min(k, n) + kn^2) \right) = \mathcal{O}(m^2 n^2),$$

Next, we consider windowed DMD, for a window containing w snapshots (with $n < w < m$). In this case, we refer to the standard algorithm, in which DMD matrix is computed directly using (7.19), as "mini-batch DMD". The computational cost is given by

$$T_{\text{mini-batch}} = \mathcal{O}\left(\sum_{k=w}^{m} (nw \min(n, w) + wn^2) \right) = \mathcal{O}(mwn^2),$$

For streaming DMD [47] for a fixed rank r, the cost of one iteration is $\mathcal{O}(r^2 n)$, and for full-rank streaming DMD, the cost of one iteration is $\mathcal{O}(n^2)$. Thus, for either task,

the overall cost after m snapshots is

$$T^{r=n}_{\text{streaming}} = \mathcal{O}(\sum_{k=1}^{m} n^2) = \mathcal{O}(mn^2),$$

and

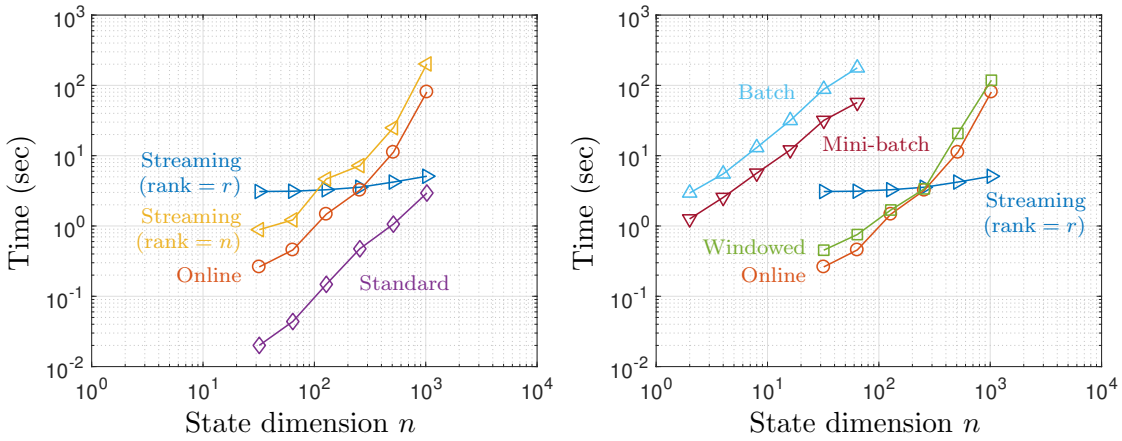
$$T^{r<n}_{\text{streaming}} = \mathcal{O}(\sum_{k=1}^{m} r^2 n) = \mathcal{O}(mr^2 n).$$

(If, in streaming DMD, the compression step (step 3 in the algorithm in [47]) is performed only every r steps, then the cost is reduced to $\mathcal{O}(mrn)$.) Finally, for both online and windowed DMD algorithms, discussed in section 7.2.2 and section 7.3.2, the cost per timestep is $\mathcal{O}(n^2)$. The algorithms are applied after w snapshots are gathered, so the overall cost of either algorithm is

$$T_{\text{online}} = T_{\text{window}} = \mathcal{O}(\sum_{k=w+1}^{m} n^2) = \mathcal{O}(mn^2).$$

Results We now compare the performance of the different algorithms on actual examples, for the two tasks described above. In particular, we consider a system with state $\boldsymbol{x} \in \mathbb{R}^n$, where n varies between 2 and 1024. The entries in the $n \times n$ matrix $\boldsymbol{A}$ are chosen randomly, according to a normal distribution (zero-mean with unit variance). The snapshots $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ are also chosen to be random vectors, whose components are also chosen according to the standard normal distribution. In the tests below, we use a fixed number of snapshots $m = 10^4$. For mini-batch DMD and windowed DMD, the window size is fixed at $w = 2048$, and online DMD and windowed DMD are both initialized after the first w snapshot pairs. For streaming DMD with a fixed rank r, we take $r = 16$. The simulations are performed in MATLAB (R2016b) on a personal computer equipped with a 2.6 GHz Intel Core i5 processor.

The results are shown in Figure 7.3. For the first task (computing the DMD matrix only at the final step), the standard DMD algorithm is the most efficient,



(a) Task: compute DMD matrix at final step.(b) Task: compute DMD matrix at each step

Figure 7.3: Performance of the different DMD algorithms on the benchmark cases described in section 7.5.1. For low-rank streaming, the dimension is limited to $r = 16$.

for the problem sizes considered here. However, note that streaming DMD with a fixed rank r scales much better with the state dimension n, and would be the fastest approach for problems with larger state dimensions.

Our primary interest here is in the second task, shown in Figure 7.3b, in which the DMD matrix is updated at each step. For problems with $n < 256$, online DMD is the fastest approach, and can be orders of magnitude faster than the standard batch and mini-batch algorithms. For problems with larger state dimensions, streaming DMD is the fastest algorithm, as it scales linearly in the state dimension (while the other algorithms scale quadratically). However, note that streaming DMD does not compute the exact DMD matrix: instead, it computes a projection onto a subspace of dimension r (here 16). By contrast, online DMD and windowed DMD both compute the full DMD matrix, without approximation.

These results focus on the time required for these algorithms, but it is worth pointing out the memory requirements as well. Streaming DMD and online DMD do not require storage of any past snapshots, while windowed DMD and mini-batch DMD require storing the w snapshots in the window, and batch DMD requires storage of all past snapshots.

7.5.2 Linear time-varying system

We now test the online DMD and windowed DMD algorithms on a simple linear system that is slowly varying in time. In particular, consider the system

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}(t)\boldsymbol{x}(t), \tag{7.34a}$$

where $\boldsymbol{x}(t) \in \mathbb{R}^2$, and the time-varying matrix $\boldsymbol{A}(t)$ is given by

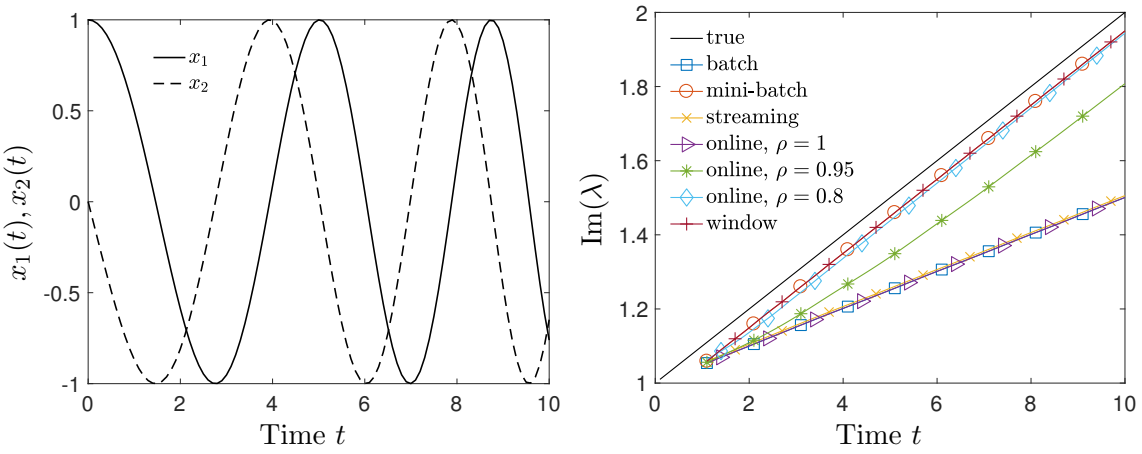
$$\boldsymbol{A}(t) = \begin{bmatrix} 0 & \omega(t) \\ -\omega(t) & 0 \end{bmatrix}, \tag{7.34b}$$

where

$$\omega(t) = 1 + \epsilon t.$$

We take $\epsilon = 0.1$, so that the system is slowly varying in time. The eigenvalues of $\boldsymbol{A}(t)$ are $\pm i\omega(t)$, and it is straightforward to show that $\|\boldsymbol{x}(t)\|$ is constant in t. We simulate the system for $0 < t < 10$ from initial condition $\boldsymbol{x}(0) = (1, 0)^T$, and the snapshots are taken with time step $\Delta t = 0.1$ as shown in Figure 7.4a. It is evident from the figure that the frequency is increasing with time.

Given the snapshots, we apply both brute-force batch DMD and mini-batch DMD as the benchmark, then we compare streaming DMD, online DMD and windowed DMD with these two benchmark brute-force algorithms. The finite time window size of mini-batch DMD and windowed DMD is $w = 10$. Batch DMD takes into account all the past snapshots, while mini-batch DMD only takes the recent snapshots from a finite time window. Streaming DMD, online DMD, and windowed DMD are initialized using the first $w = 10$ snapshot pairs, and they start iteration from time $w+1$. Batch DMD and mini-batch DMD also starts from time $w + 1$. The results for streaming DMD, online DMD ($\rho = 1, \rho = 0.95, \rho = 0.8$), and windowed DMD are shown in



(a) State evolution (b) Frequency prediction

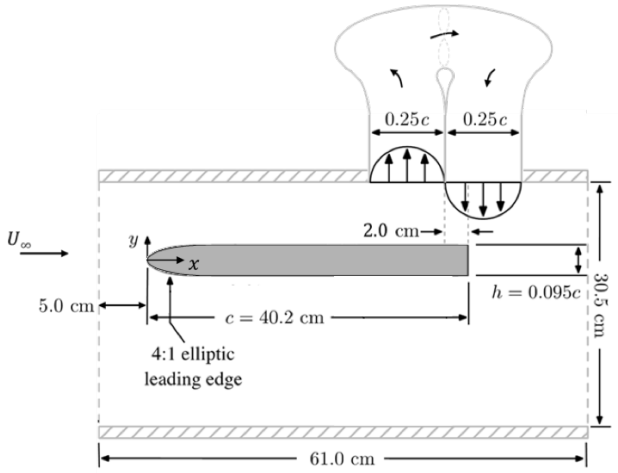
Figure 7.4: Solution of the linear time-varying system (7.34), and frequencies predicted by various DMD algorithms. For mini-batch and windowed DMD, the window size is $w = 10$. For online DMD, smaller values of the parameter ρ result in faster tracking of the time-varying frequency.

Figure 7.4b. DMD finds the discrete-time eigenvalues μ_{DMD} from data, and the figure shows the continuous-time DMD eigenvalues λ_{DMD}, which are related to these by

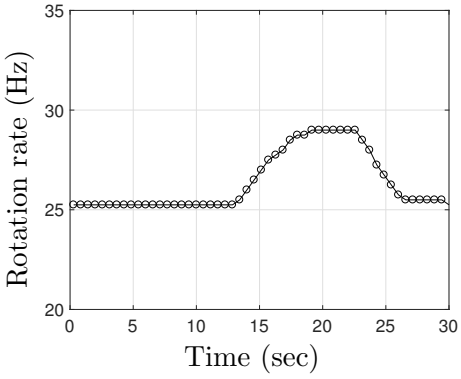
$$\mu_{\mathrm{DMD}} = e^{\lambda_{\mathrm{DMD}} \Delta t}, \tag{7.35}$$

where Δt is the time spacing between snapshot pairs. We show the DMD results starting from time $w + 1$, and the true eigenvalues are also shown for comparison.

Observe from Figure 7.4b that the eigenvalues computed by the standard algorithm (batch DMD) agree with those identified by streaming DMD and online DMD with $\rho = 1$, as expected. Similarly, windowed DMD perfectly overlaps with mini-batch DMD. When the weighting ρ in online DMD is smaller than 1, the identified frequencies shift slightly towards those identified by windowed DMD. If we further decrease the weighting factor ($\rho = 0.8$), online DMD aggressively forgets old data, and the identified frequency adapts more quickly. This example demonstrates that windowed DMD and weighted online DMD are capable of capturing time-variations in



(a) Experiment setup



(b) Separation fan roation speed

Figure 7.5: (a) Schematic of the flat plate model and flow separation system. (b) Separation fan rotation speed (rotation per second).

dynamics, with an appropriate choice of the weight ρ. A Matlab script implementing this example can be found in [112].

7.5.3 Pressure fluctuations in a separation bubble

We now demonstrate the algorithm on a more complicated example, using data obtained from a wind tunnel experiment. In particular, we study the flow over a flat plate with an adverse pressure gradient, and investigate the dynamics of pressure fluctuations in the vicinity of a separation bubble.

The setup of the wind tunnel experiment is shown in Figure 7.5a. A flat plate with an elliptical (4:1) leading edge is placed in the flow, and zero-net mass flux suction and blowing are applied at the ceiling of the wind tunnel in order to impose an adverse and subsequent favorable pressure gradient along the surface of the plate, causing the boundary layer to separate and then re-attach. More information about the separation system and the plate model can be found in [19].

These experiments were conducted in the Florida State Flow Control (FSFC) open-return wind tunnel. The cross-sectional dimensions of the test section are

30.5 cm × 30.5 cm, and the length is 61.0 cm. The contraction ratio of the inlet is 9:1. An aluminum honeycomb mesh and two fine, anti-turbulence screens condition the flow at the inlet and provide a freestream turbulence intensity of 0.5%. The suction/blowing on the ceiling of the wind tunnel test section is provided by a variable-speed fan mounted within a duct fixed to the ceiling of the wind tunnel, which pulls flow from the ceiling and reintroduces it immediately downstream. The chord of the flat plate model is $c = 40.2$ cm, and the height is $h = 0.095c$. For these experiments, the freestream velocity is $U_\infty = 3.9\,\mathrm{m/s}$ and the Reynolds number is $\mathrm{Re}_c = U_\infty c/\nu = 10^5$.

Unsteady surface pressure fluctuations within the separated flow are monitored by an array of 13 surface-mounted Panasonic WM-61A electret microphones located within the separation region. The microphones are placed at the centerline of the plate, between $x/c = 0.70$ and 0.94, with a spacing of $\triangle x/c = 0.02$. More details regarding this microphone array can be found in [19]. These 13 microphone signals provide the data we use for online DMD.

We change the rotation speed of the separation fan slowly, in order to impose time-varying boundary conditions and induce time-varying dynamics in the separation bubble. The size of the separation bubble increases with the rotation speed of the separation fan. The rotation speed starts at about 25 Hz (rotation per second), and increases slowly to reach a high of about 30 Hz, then back to initial speed of 25 Hz. The blade rotation rate is shown in Figure 7.5b. The pressure data sampling frequency is $f_s = 10^4$, and we collect data for $T = 30$ sec. The total number of samples is $m = 3 \times 10^5$, and the state dimension is $n = 13$ (there are 13 pressure sensors).

To study the frequencies from the pressure data, we first present a spectral analysis of the pressure data using short-time discrete Fourier transform (DFT). Figure 7.6 shows the results for the 7-th (medium) pressure sensor; other pressure sensors have

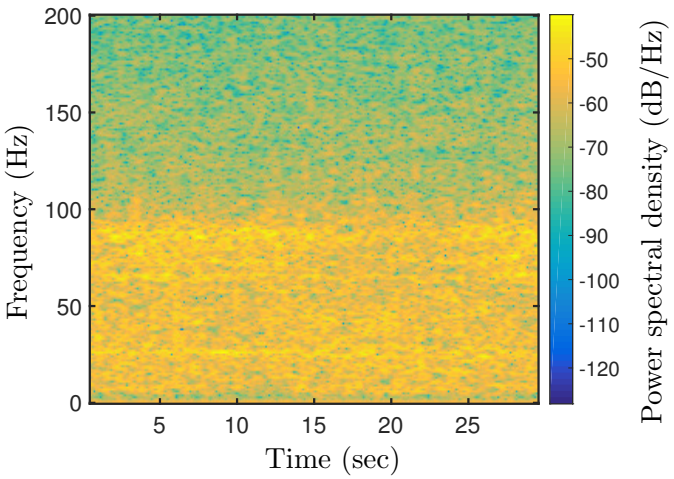


Figure 7.6: Power spectral density (PSD) of the pressure measurement at the 7-th (medium) pressure sensor. There is no persistent peak frequency, and the dominant frequencies are time-varying.

similar results. A window size of $f_s = 10^4$ is used (along with a Hamming window of the same length), with 90% overlap of samples between adjoining sections. It is observed that there is no persistent peak frequency, and the dominant frequencies are time-varying. In order to gain a comprehensive understanding of the frequency variations in all the pressure sensors, and how these might be related to one another, we apply online DMD and windowed DMD to the all the pressure data.

The number of snapshots $m = 3 \times 10^5$ is much larger than the state dimension is $n = 13$, so the over-constrained assumption is satisfied. Recall that our proposed algorithm relies on the assumption that the number of snapshots is larger than the state dimension. Therefore, we can apply both online DMD and windowed DMD for this problem. The dynamics of the pressure fluctuations can be characterized by the DMD frequencies, which may be slowly varying in time. The DMD frequency is defined as

$$f_{\text{DMD}} = \frac{\text{Im}(\lambda_{\text{DMD}})}{2\pi},$$

where $\text{Im}(\lambda_{\text{DMD}})$ is the imaginary part of the continuous-time DMD eigenvalues computed from (7.35). (The discrete-time eigenvalues μ_{DMD} are eigenvalues of the 13×13

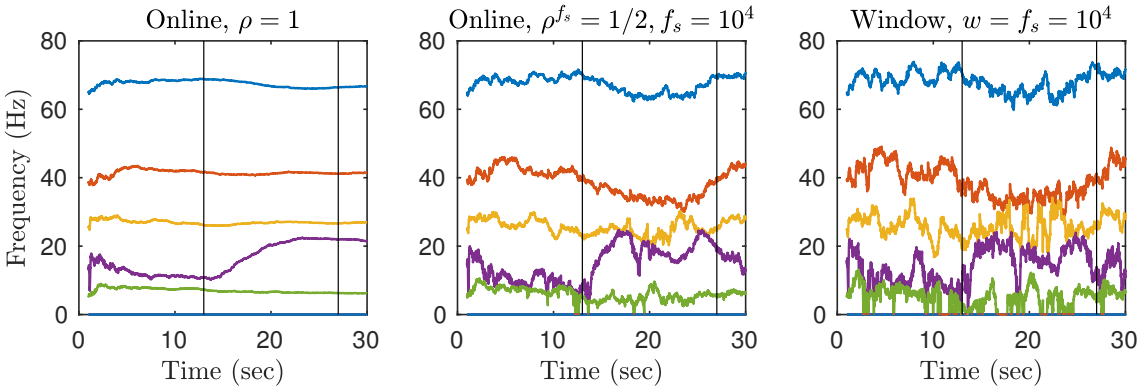


Figure 7.7: DMD frequencies identified by different DMD algorithms from 13 pressure signals, as described in section 7.5.3. The two vertical lines indicate the time when the fan rotation rate starts to increase ($t = 13$ sec) and finally settles down to the original speed ($t = 27$ sec).

matrix matrix $\boldsymbol{A}_k$.) The frequencies computed by the various DMD algorithms are shown in Figure 7.7. There are 13 DMD eigenvalues in total, and one of them is $f_0 = 0$ Hz corresponding to the mean. The remaining DMD eigenvalues consist of six complex-conjugate pairs, corresponding to six non-zero DMD frequencies. The DMD computation reveals three zero frequencies for all cases we run, indicating that there are only five active (non-zero) frequencies present in the system. Online DMD with $\rho = 1$ is run to obtain the standard DMD result. We use a weighting factor of $\rho^{f_s} = 0.5$ (corresponding half-decay life is f_s samples, i.e., one second.) for weighted online DMD and window size of $w = f_s = 10^4$ for windowed DMD. The DMD algorithms are initialized with the first w snapshots, and the resulting DMD frequencies are computed starting from time step $w + 1$.

Recall that with $\rho = 1$, online DMD coincides with the standard DMD algorithm. From Figure 7.7, we see that for this case, the three leading frequencies remain more or less constant in time. However, we expect the dynamics in the separation bubble to be time-varying. In response to the change of fan speed, the frequencies should start to change at $t = 13$ sec, and return close to their original values at $t = 27$ sec (with some lag). The weighted online DMD and windowed DMD both act as desired,

while the standard DMD is clearly erroneous. With weighting factor $\rho^{f_s} = 1/2$, online DMD behaves more like windowed DMD: in particular, the method is better at tracking variations in the frequency. The weighting factor in online DMD acts like a soft cut-off for the old snapshots, compared with the hard cut-off imposed by windowed DMD. Online DMD gradually forgets the old snapshots, so it is expected to be smoother than windowed DMD. From the results, we confirm that the online DMD is smoother compared to the windowed DMD. While the frequency variations shown in Figure 7.7 appear to be rapid or noisy, note that the time interval shown in the figure is quite long (about 300 periods for the lowest frequency of around 10 Hz), so it is reasonable to consider these frequencies as slowly varying in time.

Due to the slowly time-varying boundary condition imposed by the separation fan, the dynamics of the pressure fluctuations in the separation bubble is expected to be slowly time-varying too. When we simply use standard DMD, then the frequencies (and linear models) we identify will not be adaptive to the flow conditions in real-time. In contrast, weighted online DMD and windowed DMD successfully tracks the time variations from the pressure signals. Therefore, the proposed online algorithms are demonstrated to be beneficial in real-time modeling of time-varying systems. In fact, the online DMD algorithm has been applied to this particular separation control problem [18] for system identification and real-time control purposes. The idea is to use online DMD to find an adaptive linear model from pressure measurements and control input history (see section 7.4.1), and apply linear control based on this adaptive model.

7.6 Conclusion and outlook

In this work, we have developed efficient methods for computing online DMD and windowed DMD. The proposed algorithms are especially useful in applications for

which the number of snapshots is huge compared to the state dimension, and when the dynamics are slowly varying in time. The number of snapshots is assumed to be larger than state dimension, because we are primarily interested in real-time modeling (updating the DMD matrix). In real-time, it is often the case that we have a relatively small number of measurements, and as time progresses, eventually we will always have more snapshots than measurements. A weighting factor can be included easily in the online DMD algorithm, which is used to weight recent snapshots more heavily than older snapshots. This approach corresponds to using a soft cut-off for older snapshots, while windowed DMD uses a hard cut-off, from a finite time window. The proposed algorithms can be readily extended to online system identification, even for time-varying systems.

The efficiency is compared against the standard DMD algorithm, both for situations in which one computes the DMD matrix only at the final time, and for situations in which one computes the DMD matrix in an "online" manner, updating it as new snapshots become available. The latter case is applicable, for instance, when one expects the dynamics to be time-varying. For the former case, the standard DMD algorithm is the most efficient, while for the latter case, the new online and windowed DMD algorithms are the most efficient, and can be orders of magnitude more efficient than the standard DMD algorithm. Table 7.1 provides a brief comparison of the main characteristics and features of standard DMD, batch DMD, mini-batch DMD, (low rank) streaming DMD, online DMD, and windowed DMD.

The algorithms are further demonstrated on a number of examples, including a linear time-varying system, and data obtained from a wind tunnel experiment. As expected, weighted online DMD and windowed DMD are effective at capturing time-varying dynamics.

In this paper we assign different weights to different samples, and in particular a weighting factor is used to forget old snapshots. This choice of weight is especially

Aspect	Standard	Batch	Mini-batch	Streaming	Online	Windowed
Computational time	$\mathcal{O}(mn^2)$	$\mathcal{O}(kn^2)$	$\mathcal{O}(wn^2)$	$\mathcal{O}(r^2n)$	$4n^2$	$8n^2$
Memory	mn	kn	wn	$\mathcal{O}(rn)$	$2n^2$	$wn + 2n^2$
Store past snapshots	Yes	Yes	Yes	No	No	Yes
Track time variations	No	No	Yes	Yes	Yes	Yes
Real-time DMD matrix	No	Yes	Yes	Yes	Yes	Yes
Exact DMD matrix	Yes	Yes	Yes	No	Yes	Yes

Table 7.1: Characteristics of the various DMD algorithms considered. Relevant parameters are state dimension n, total number of snapshot pairs $m \gg n$, window size w such that $n < w \ll m$, low rank $r < n$, and discrete time $k > n$. Computational time denotes the required floating-point multplies for one iteration (computing the DMD matrix).

useful for time-varying systems, as older information becomes outdated. As one reviewer pointed out, some real-world applications might involve measurements with different degrees of reliability (e.g., different levels of uncertainty, different measurement methods). One could in principle place different weights on different samples, according to how much one "trusts" them, which would lead to a weighted linear regression problem. This could be an interesting direction for future work.

Another relevant direction for future work is a more detailed study of the application of proposed online/windowed DMD algorithms to system identification. In situations where there are variations in dynamics, or where we desire real-time control, it is crucial to build accurate and adaptive reduced order models for effective control, and the methods proposed here could be useful in these cases.

Acknowledgment

The authors thank Professor Maziar S. Hemati for helpful suggestions about references.

Chapter 8

Input output analysis of separated flow past a flat plate

Hao Zhang[1], Clarence W. Rowley[1], Wen Wu[2], Charles Meneveau[2], and Rajat Mittal[2]

[1]Mechanical and Aerospace Engineering, Princeton University, Princeton, NJ 08544, USA

[2]Mechanical Engineering, Johns Hopkins University, Baltimore, MD 21218, USA

Appears as [114], AIAA Scitech 2019 Forum, 2019, doi: 10.2514/6.2019-0880.

The response of a separated flow to external forcing is investigated using input-output analysis. A laminar separation bubble is induced on the flow past a flat plate by imposing an adverse pressure gradient. We study the response of spanwise-constant perturbations to both global body forcing and local body forcing. Input-output analysis (also referred to as resolvent analysis) uses a linear operator that maps the external forcing to the response of the flow field. For this flow forced at a single frequency, the linear operator is closely approximated by an operator of rank one and describes the optimal spatial forcing for which the resulting response has maximum amplification. The input-output analysis gives valuable information about optimal actuator placement and optimal actuation frequency. It is found that the

flow response is maximized when forcing at the natural frequency of the separation bubble. The optimal response mode implies that the separation bubble is receptive to the upstream body forcing. The optimal forcing mode indicates that body forcing should be applied upstream of the separation bubble, and gives information about the size and shape of the region where forcing should be applied, in order to maximize the response in the separation bubble.

8.1 Introduction

Flow separation is usually an undesirable phenomenon in aerodynamic applications because it reduces lift and increases (pressure) drag [67]. The complicated nonlinear dynamics of flow separation are characterized by Kelvin-Helmholtz instability, wake shedding, and separation bubble oscillations [67]. Efforts have been devoted to designing control methods to suppress separation and reattach a separated flow. A number of studies try to find the optimal actuation parameters by searching the parameter space: see [85, 33, 68, 76] for example. Parameter-searching control methods usually require huge experimental or simulation efforts, and provide only limited physical insights about the problem. Among various separation control methods, zero-net-mass-flux (ZNMF) actuators are simple to implement and have been shown to be effective in reducing flow separation [32, 84, 33, 64, 18].

The input-output analysis was introduced to the fluid community to study the response of a flow field to disturbances [51, 63]. This approach uses the transfer function (a linear operator) from the input forcing (including nonlinear advective forcing, and any other external forcing) to the output response (velocity and pressure field). This linear operator is often low rank (sometimes approximately rank-one), and from it, one can deduce the optimal forcing mode that actuates the optimal

response that has maximum amplification at any given forcing frequency. Input-output analysis, also referred to as resolvent analysis, has been used for a wide range of problems [59, 30, 49, 35], ranging from pipe turbulence control to building reduced-order models. A review of various applications is presented in [95].

The focus of this paper is to better understand the physics of a separated flow, with the ultimate objective of controlling the flow. We consider a laminar boundary layer with a separation bubble along a flat plate. The separation is induced by a pressure gradient imposed through suction and blowing at the upper wall of the numerical domain. The forcing and response determined by input-output analysis can be extremely useful, since the optimal forcing and response can be used to guide the placement of actuators and other aspects of the control design. Input-output analysis has been utilized for the design of airfoil separation control [109], in which localized unsteady thermal actuation is applied near the leading edge. A previous study has looked into the response mode of incompressible jets to both body forcing and boundary forcing [30]. In the present work, we actuate the separated flow by both global and local body forcing.

8.2 Methodology

8.2.1 Flow setup and numerical simulation

In this paper, we study a laminar boundary layer with a separation bubble induced by an adverse pressure gradient. The pressure gradient is imposed by suction and blowing on the top boundary, as shown in Fig. 8.1. This configuration has been widely studied as a proxy for separation in the flow past a wing at a high angle of attack, because it isolates the effects of separation from other geometric effects such as wall curvature [69].

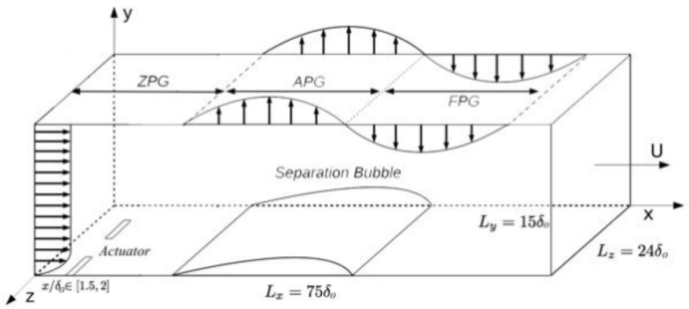
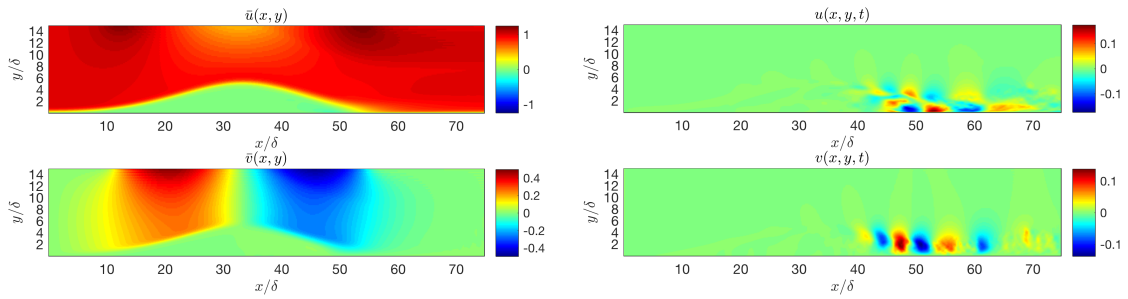


Figure 8.1: Sketch of separated flow on a flat plate (not to scale). Adapted from [107].

The numerical simulation of the three-dimensional incompressible Navier-Stokes equations is performed with a second-order finite difference code ViCar3D [66]. Further details about the numerical schemes of the solver can be found in [86, 107]. The flow variables are nondimensionalized by the free-stream velocity $U_{\infty,0}$, and the boundary thickness δ_0 of the Blasius velocity profile at the inlet of the flow domain. The Reynolds number of the flow is $Re = U_{\infty,0}\delta_0/\nu = 1000$, where ν is the kinematic viscosity. The size of the computational domain is $75\delta_0$ in the streamwise direction, $15\delta_0$ in the wall-normal direction, and $24\delta_0$ in the spanwise direction.

Blasius velocity profiles are imposed at the inlet, and a Neumann boundary condition is used at the outlet (zero normal derivative). A no-slip boundary condition is imposed at the bottom wall, and a suction-blowing boundary condition (for wall-normal velocity) is imposed at the top boundary. At the top boundary, the streamwise velocity is implied by a zero-spanwise-vorticity condition. Periodic boundary conditions are assumed in the spanwise direction. The pressure satisfies a Neumann boundary condition at all boundaries (zero derivative normal to the boundary).

A simulation with no forcing is performed, and that is denoted as the base case ("case B"). For this case, the flow field averaged in time and in the spanwise direction is shown in Fig. 8.2, along with the spanwise-averaged fluctuations (deviations from the time-averaged flow). The time and spanwise averaged field will be used as base flow in the input-output analysis. The spanwise velocity is not reported here



(a) (b)

Figure 8.2: (a) Time and spanwise averaged field. (b) Spanwise averaged fluctuation field.

because it is small and close to zero. The flow is mainly two-dimensional, along with small spanwise fluctuations. Observe first that the imposed adverse pressure gradient successfully induces a separation bubble. The separation bubble spans from $x = 10\delta$ to $x = 55\delta$. The fluctuation field is nonzero mainly in and behind the trailing edge of the separation bubble region. The time- and spanwise-averaged flow field is taken as the base flow for the input-output analysis. We then study the response of the fluctuation field to external body forcing.

8.2.2 Input-output analysis formulation

To formulate the input-output analysis, we start from the incompressible Navier-Stokes equations

$$\partial_t \tilde{\boldsymbol{u}} = -\tilde{\boldsymbol{u}} \cdot \nabla \tilde{\boldsymbol{u}} - \nabla \tilde{p} + \frac{1}{Re} \nabla^2 \tilde{\boldsymbol{u}}, \tag{8.1a}$$

$$\nabla \cdot \tilde{\boldsymbol{u}} = \boldsymbol{0}. \tag{8.1b}$$

The full flow field $(\tilde{\boldsymbol{u}}, \tilde{p})$ is decomposed into a base flow $(\bar{\boldsymbol{u}}, \bar{p})$, which is usually taken to be the mean flow or a steady solution of the Navier-Stokes equations, and a fluctuation field $(\boldsymbol{u}, p)$, and we derive an equation for the fluctuations. Letting

$\tilde{\boldsymbol{u}} = \bar{\boldsymbol{u}} + \boldsymbol{u}, \tilde{p} = \bar{p} + p$ and substituting into Eq. (8.1), we obtain

$$\partial_t \boldsymbol{u} = L\boldsymbol{u} - \nabla p + \boldsymbol{f}, \tag{8.2a}$$

$$\nabla \cdot \boldsymbol{u} = \boldsymbol{0}, \tag{8.2b}$$

where

$$Lu = -\bar{\boldsymbol{u}} \cdot \nabla \boldsymbol{u} - \boldsymbol{u} \cdot \nabla \bar{\boldsymbol{u}} + \frac{1}{Re} \nabla^2 \boldsymbol{u},$$

and L is the linearized Navier-Stokes operator. The forcing term $\boldsymbol{f}$ includes all additional terms, including the nonlinear advective term and any external forcing terms such as body forcing and boundary forcing.

The governing equation of the fluctuation field is linear. Treating the nonlinear advective term and external forcing as input and velocity field as output, we can derive a state-space representation for the fluctuation field. If we define

$$q = \begin{bmatrix} \boldsymbol{u} \\ p \end{bmatrix}, \quad A = \begin{bmatrix} L & -\nabla \\ \nabla\cdot & 0 \end{bmatrix}, \quad M = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 \end{bmatrix},$$

then Eq. (8.2) can be put into state-space form

$$M\partial_t q = Aq + B\boldsymbol{f}, \tag{8.3a}$$

$$\boldsymbol{u} = Cq, \tag{8.3b}$$

where $\boldsymbol{f}$ is the input and $\boldsymbol{u}$ is the output. If we consider sinusoidal forcing $\boldsymbol{f} = \hat{\boldsymbol{f}} e^{i\omega t}$, then both the state q and output $\boldsymbol{u}$ will be sinusoidal because the system is linear. Letting $\boldsymbol{u} = \hat{\boldsymbol{u}} e^{i\omega t}$ and substituting into Eq. (8.3), we obtain

$$\hat{\boldsymbol{u}} = H(i\omega)\hat{\boldsymbol{f}},$$

where

$$H(i\omega) = C(i\omega M - A)^{-1}B$$

is the transfer function from input ($\boldsymbol{f} = \hat{\boldsymbol{f}}e^{i\omega t}$) to output ($\boldsymbol{u} = \hat{\boldsymbol{u}}e^{i\omega t}$). The transfer function $H(i\omega)$ is the centerpiece of input-output analysis. (The operator $(i\omega M - A)^{-1}$ is called the resolvent operator, and analysis of this operator is often referred to as "resolvent analysis.") Notice that $H(i\omega)$ will be different for each different frequency ω. Keep in mind that $H(i\omega)$ is the transfer function from the forcing at a particular frequency ω to the response at the same frequency. In what follows, we fix a particular frequency ω, and we drop the explicit dependence of H on ω. However, note that the forcing and response modes we discuss below depend on the frequency ω.

Now let us consider the maximum amplification problem. We are interested in finding an input forcing with unit norm such that that the output has maximum norm:

$$\max_{\hat{\boldsymbol{f}}} \|\hat{\boldsymbol{u}}\|, \quad s.t. \quad \|\hat{\boldsymbol{f}}\| = 1,$$

where $\|\cdot\|$ denotes a suitable norm on the flow field $\hat{\boldsymbol{u}}$ and forcing term $\hat{\boldsymbol{f}}$. One can show (see, e.g., [10]) that the optimal forcing $\hat{\boldsymbol{f}}$ is an eigenvector of H^*H, where H^* denotes the adjoint of H: in particular, the optimal $\hat{\boldsymbol{f}}$ is the solution of

$$H^*H\hat{\boldsymbol{f}} = \sigma^2\hat{\boldsymbol{f}}$$

corresponding to the largest eigenvalue σ^2. The solution to the eigenvalue problem can be found from the singular value decomposition (SVD) of the linear operator

$$H = \sum_j \sigma_j \psi_j \phi_j^*,$$

where $\psi_i^*\psi_j = \delta_{ij}, \phi_i^*\phi_j = \delta_{ij}$, and $\sigma_j \geq 0$ is in descending order. The optimal forcing is the first right singular vector ϕ_1 and the optimal response is the first left singular

vector ψ_1. The largest singular value σ_1 gives the corresponding amplification. For an arbitrary forcing $\hat{\boldsymbol{f}}$, the response $\hat{\boldsymbol{u}}$ can be written as

$$\hat{\boldsymbol{u}} = \sum_j \sigma_j \psi_j (\phi_j^* \hat{\boldsymbol{f}}),$$

where $\phi_j^* \hat{\boldsymbol{f}}$ is the projection of $\hat{\boldsymbol{f}}$ in the direction of ϕ_j. For many shear flows of practical interest [63, 59, 30, 10] the linear operator H can be closely approximated by an operator of rank one: $\sigma_1 \gg \sigma_{j\geq 2}$. A rank-one approximation of the response is then

$$\hat{\boldsymbol{u}} \approx \sigma_1 \psi_1 (\phi_1^* \hat{\boldsymbol{f}}), \tag{8.4}$$

and one expects this to be a close approximation of the response for a typical forcing $\hat{\boldsymbol{f}}$ (i.e., as long as the direction of forcing is not such that $\phi_1^* \hat{\boldsymbol{f}}$ is small).

The base flow is computed from the simulation previously described as "case B." We select a particular frequency ω, as the dominant frequency observed in the fluctuations for case B (as discussed in the next section). The transfer function $H(i\omega)$ is discretized with a finite-difference method, with appropriate boundary conditions for the fluctuations deduced from the simulation boundary conditions. After discretization, $H(i\omega)$ is a huge matrix of size 0.26 million by 0.26 million. For a matrix of this size, it is too cumbersome to compute the matrix inverse $(i\omega M - A)^{-1}$ explicitly, and accordingly, is extremely computationally expensive to find the SVD of $H(i\omega)$ directly. Fortunately, we only need the first (few) singular values and singular vectors of H. The randomized SVD method of [42] is used to obtain the approximate leading singular values and singular vectors, without the need to explicitly compute the transfer function H or the resolvent $(i\omega M - A)^{-1}$.

8.2.3 Local body forcing

In the presentation above, the body force is global. However, it is usually not practical to impose a body force everywhere in space. We now consider external forcing as a local body force. The governing equation for the fluctuation field is

$$\partial_t \boldsymbol{u} = L\boldsymbol{u} - \nabla p + b(\boldsymbol{x})\boldsymbol{f}, \tag{8.5a}$$

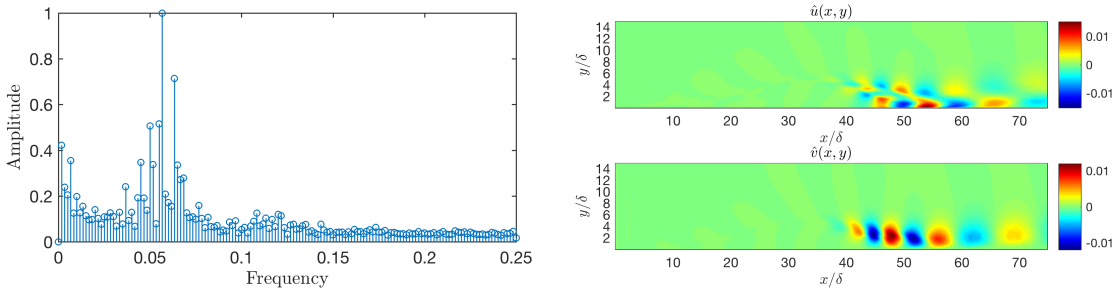
$$\nabla \cdot \boldsymbol{u} = \boldsymbol{0}, \tag{8.5b}$$

with appropriate boundary conditions as specified in previous section. $b(\boldsymbol{x})$ embeds where the local body force is applied. In this study, we consider local body force in the upstream near the wall boundary (due to global forcing analysis). Following the derivation in section 8.2.2, we can obtain a state-space representation similar to Eq. (8.3). For local forcing, we essentially have a different B matrix, which allows us to select which part of the flow field to apply forcing.

8.3 Results and Discussion

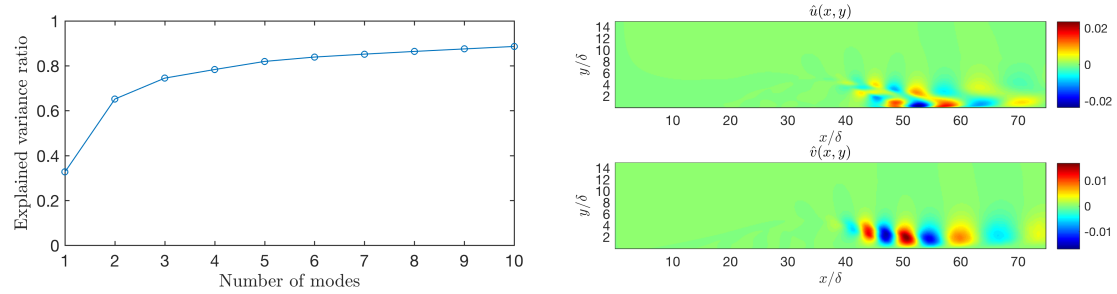
8.3.1 Modal decomposition analysis

We start with a modal decomposition analysis of this problem. In particular, we consider three methods: discrete Fourier transform (DFT), proper orthogonal decomposition (POD) [60], and dynamic mode decomposition (DMD) [81]. They will provide insight into the coherent structures and main flow physics in the flow. Both three methods are based on temporal-spatial data (discrete velocity field measurements that evolve in time). DFT gives spatial mode associated with uniformly spaced frequencies ranging from zero to Nyquist frequency. POD extracts the most energetic spatial mode from data. DMD finds spatial mode that evolves in time with a fixed frequency



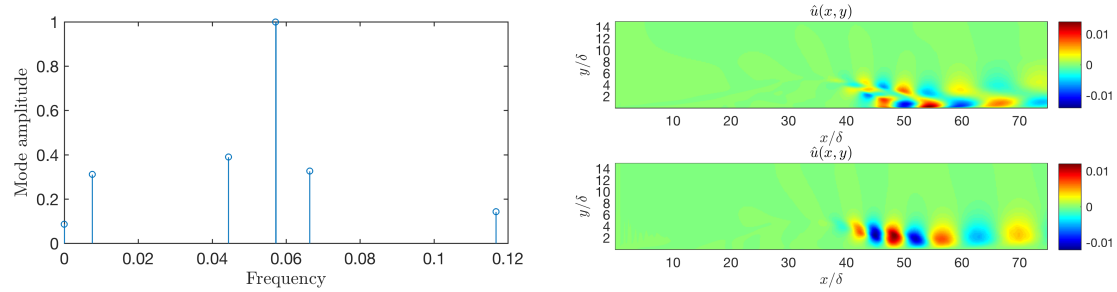
(a) (b)

Figure 8.3: (a) DFT frequency and amplitude (normalized by maximum amplitude). (b) DFT mode $\hat{u}, \hat{v}$ (real part) associated with the peak frequency $f = 0.0567$.



(a) (b)

Figure 8.4: (a) Explained variance ratio with respect to number of POD modes. (b) First POD mode $\hat{u}, \hat{v}$.



(a) (b)

Figure 8.5: (a) DMD frequency and amplitude (normalized by maximum amplitude), largest amplitude occurs at $f = 0.0571$. (b) Dominant (largest amplitude) DMD mode $\hat{u}, \hat{v}$.

and growth/decay rate. For more details and reviews about modal decomposition methods in fluid flows, see [79, 95].

First, we apply DFT to the fluctuation velocity field which consists of 300 snapshots with a time step of $\triangle t = 2$. DFT frequencies, amplitudes, and the first mode

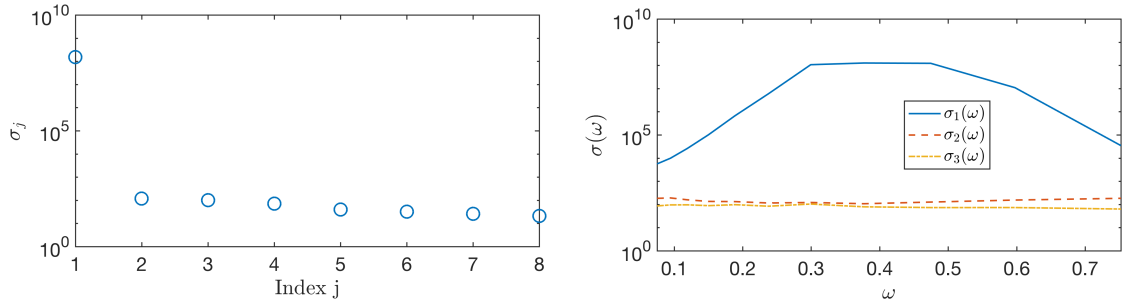
are shown in Fig. 8.3. It is revealed that there is a (non-dimensional) peak frequency at $f = 0.0567$. The associated mode resembles the fluctuation velocity field, as shown in Fig. 8.2 (b). This Fourier mode indicates the coherent structure in this flow.

Next, we analyze the same fluctuation velocity field with POD. The result is presented in Fig. 8.4. We report the explained variance ratio, which is defined as

$$r_k = \sum_{j=1}^{k} \sigma_j^2 / (\sum_{j=1}^{N} \sigma_j^2), \tag{8.6}$$

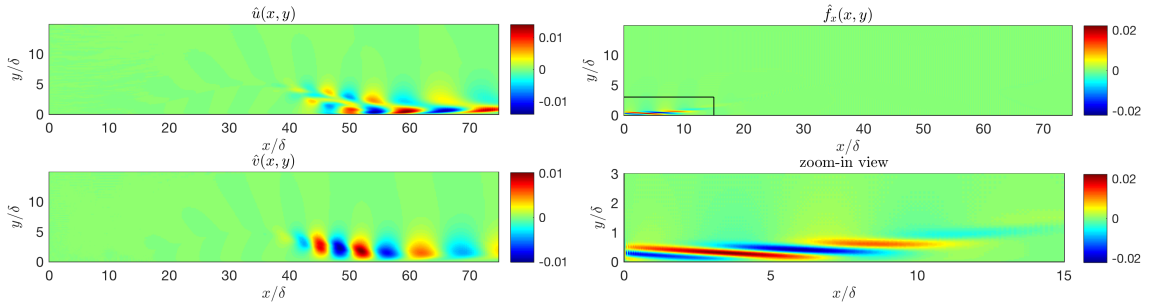
where k is the number of modes used, N is the total number of modes, and σ_j is the j-th singular value (in descending order) of the snapshot matrix (whose column is the discrete velocity field). r_k quantifies how much energy is explained by the first k POD modes. The first 10 POD modes account for about 90% of the total energy in the flow field. The first POD mode is shown in Fig. 8.4 (b), and this is considered as the dominant coherent structure in the fluctuation field. Observe that it is very similar to the dominant DFT mode.

Finally, DMD is used to find coherent structures in the fluctuation flow field. We first project the snapshots onto the subspace spanned by the leading POD modes of the snapshot matrix, and then perform DMD in this subspace. This approach is more numerically stable and computationally efficient [79]. We use 11 POD modes, which amounts to keeping about 90% of the total energy in the snapshots. The result is shown in Fig. 8.5. DMD amplitude is defined as the magnitude of the coefficient when the initial condition is projected onto the DMD modes. DMD identifies a dominant frequency of $f = 0.0571$, which is (almost) the same as the DFT peak frequency $f = 0.0567$. The associated dominant DMD mode is similar to those found by DFT and POD.



(a) (b)

Figure 8.6: Global forcing result. (a) Singular values of the linear operator (transfer function) for $k_z = 0, \omega_p = 0.377$. (b) First three singular values of the transfer function for a range of frequency ω around ω_p. $k_z = 0$ is fixed.



(a) (b)

Figure 8.7: Global forcing result. (a) Streamwise and wall-normal optimal response mode $\hat{u}, \hat{v}$. $k_z = 0, \omega = 0.377$. (b) Streamwise optimal forcing mode $\hat{f}_x$ and zoom-in view in the region $(x/\delta, y/\delta) \in [0, 15] \times [0, 3]$.

8.3.2 Global body forcing and optimal actuator placement

One question we are interested in is the optimal location of an actuator. As discussed in the previous section, the optimal forcing mode gives insight into this question, since it gives the forcing term (with unit norm) that has the largest amplification. We first present results of input-output analysis considering the global body forcing distributed in the whole domain. Because the body force is global instead of local, the optimal forcing mode will shed light on the optimal actuator placement.

Assuming sinusoidal forcing in both time and the spanwise direction, the output will also be sinusoidal (in time and the spanwise direction), because the system is

linear. Let

$$\boldsymbol{f}(x,y,z,t) = \hat{\boldsymbol{f}}(x,y)e^{i(k_z z+\omega t)}, \quad \boldsymbol{u}(x,y,z,t) = \hat{\boldsymbol{u}}(x,y)e^{i(k_z z+\omega t)}.$$

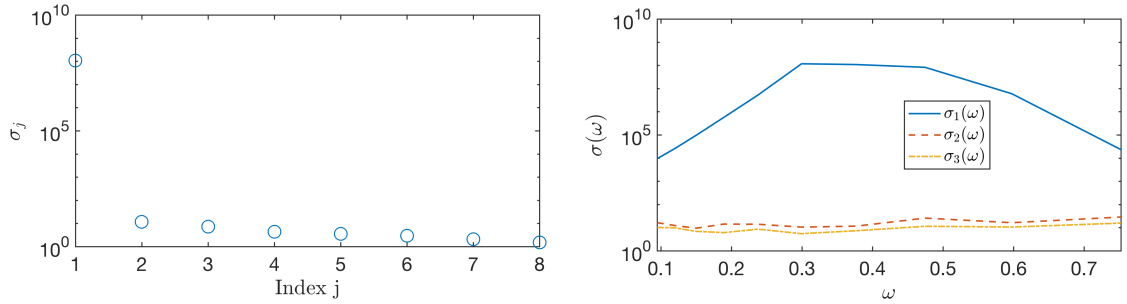
We consider only two-dimensional forcing in this study, i.e., $k_z = 0$. For each ω, we have a different transfer function, and therefore a different set of optimal forcing modes and response modes. We consider a particular frequency ω determined from the simulation previously described as "case B." From a power spectral density analysis of the streamwise velocity of a probe in the separation bubble, a (non-dimensional) peak frequency of $f_p = 0.06$ is found, which corresponds to the natural frequency of the separation bubble [107]. This frequency is also very close to the dominant frequency found from the discrete Fourier transform of the full fluctuation velocity field. To report the result, we set $\omega_p = 2\pi f_p = 0.377$. The first eight singular values computed by randomized SVD are shown in Fig. 8.6 (a). The first singular value and singular vector change no more than 1% when we increase the number of random vectors in the randomized SVD algorithm by a factor of 2, verifying the convergence of the algorithm.

We observe that the first singular value is six orders of magnitude larger than the second and the rest, so the transfer function may indeed be approximated by an operator of rank one. To study the validity of rank-one approximation for other frequencies, we show the first three singular values of the transfer function (found by randomized SVD algorithm) for frequency ω varying around ω_p in Fig. 8.6 (b). It is verified that rank-one approximation is valid for a range of frequencies. Furthermore, we find that $\omega_p = 0.337$ has the maximum amplification among all the frequencies around it. This is consistent with the standard practice to force the flow at its natural frequency.

The optimal response mode and optimal forcing mode are shown in Fig. 8.7. The energy in the optimal response mode is mainly concentrated in the streamwise and wall-normal component $(\hat{u}, \hat{v})$. The spanwise response $\hat{w}$ is found to be five orders of magnitude smaller than $\hat{u}, \hat{v}$. Only two-dimensional disturbances are considered here, so we do not expect a response in the spanwise component. Recall that the optimal response mode is the dominant response to disturbances in the flow. The response is primarily in and behind the trailing edge of the separation bubble, from $x = 40\delta$ to $x = 70\delta$. Therefore, the separation bubble is receptive to disturbances.

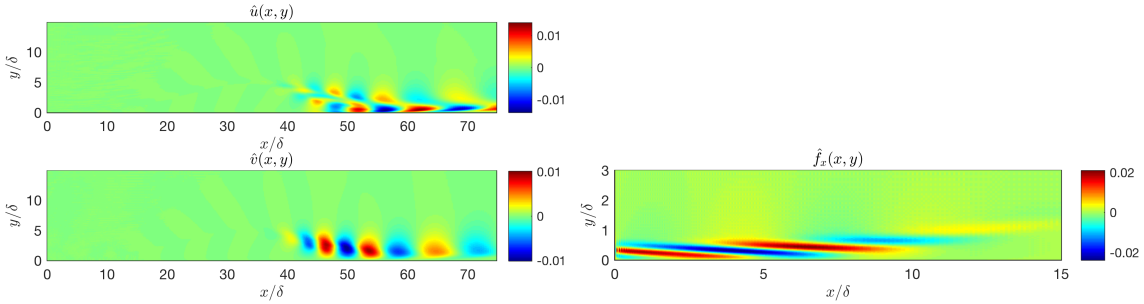
As for the optimal forcing mode, we find that $\hat{f}_y$ is much smaller than $\hat{f}_x$, and $\hat{f}_z$ is nine orders of magnitude smaller than $\hat{f}_x$. Therefore, only the streamwise component $\hat{f}_x$ of the forcing is shown. The optimal forcing mode reveals that streamwise body force disturbance is much more crucial. Nonetheless, the ZNMF actuator produces disturbances mainly in the wall-normal direction. This result implies that a more effective actuator should instead introduce streamwise disturbances, which are much more efficient at exciting a response in the separation bubble. The optimal forcing energy is distributed along the wall, upstream of the separation bubble. Upstream disturbances to the flow travel downstream and produce a response in the separation bubble. This observation indicates that an actuator should be placed upstream, as in previous studies [67, 76]. The forcing mode also shows similarity with the optimal disturbances in the Blasius boundary layer reported in [70]. In [70], the optimal initial condition leading to the largest growth at finite times and the optimal time-periodic forcing leading to the largest asymptotic response are both considered.

The streamwise optimal forcing alternates between positive and negative values in the upstream region. Recall from Eq. 8.4 that the response is approximately proportional to the projection of the forcing in the direction of the optimal forcing mode ϕ_1. When the forcing is applied in both the positive region and negative region, their responses will be out of phase, and cancel each other. Typically, the location of



(a) (b)

Figure 8.8: Local forcing result. (a) Singular values of the linear operator (transfer function) for $k_z = 0, \omega_p = 0.377$. (b) First three singular values of the transfer function for a range of frequency ω around ω_p. $k_z = 0$ is fixed.



(a) (b)

Figure 8.9: Local forcing result. (a) Streamwise and wall-normal optimal response mode $\hat{u}, \hat{v}$. $k_z = 0, \omega = 0.377$. (b) Streamwise optimal forcing mode $\hat{f}_x$.

a ZNMF actuator (which resembles a local body force) is not well-tuned due to a lack of physical insight. The optimal forcing mode suggests that the body force actuator should be carefully placed in order to avoid cancellation in the response.

8.3.3 Local body forcing

The governing equation Eq. (8.5) is spatially semi-discretized and put into state space form. Then we can derive the corresponding transfer function. We are mainly interested in local body forcing near the upstream of the wall. Consider local body forcing in the region $(x/\delta, y/\delta) \in \Omega$, where $\Omega = [0, 15] \times [0, 3]$. This can be achieved by choosing $b(\boldsymbol{x})$ to be 1 in Ω and 0 elsewhere in Eq. (8.5).

Experiment	Simulation	Description
1	S1A	streamwise forcing placed upstream near the wall
	S1B	streamwise forcing placed upstream away from the wall
2	S2A=S1A	see S1A
	S2B	wall-normal forcing applied at the same region as streamwise forcing
3	S3A=S1A	see S1A
	S3B	streamwise forcing with the same shape but different orientation

Table 8.1: Setup of three control experiments. Each control experiment consists of two numerical simulations.

Similar to global forcing, the randomized SVD algorithm is used to compute the leading singular values and singular vectors of the transfer function. The singular values for the case of $\omega_p = 0.337$ is presented in Fig. 8.8 (a). In addition, the first three singular values for a range of frequencies ω is shown in Fig. 8.8 (b). We conclude that rank-one approximation is valid for a range of frequencies ω. The optimal forcing and response mode corresponding to ω_p is illustrated in Fig. 8.9. Again the wall-normal forcing mode is much smaller and thus not shown here. Observe that both the response mode and forcing mode are almost identical to those found by global body forcing.

8.3.4 Implications and future work

In this part, we will further discuss the implications of input-output analysis. While the preceding analysis provides insight about where and how to force the flow field, we still need to validate these implications numerically. We will outline future work.

Based on the discussion in 8.3.2, the optimal forcing mode is concentrated upstream near the wall, and streamwise forcing is much more important (compared with wall-normal forcing). In addition, the forcing mode consists of patches alternating between positive and negative values. As a result, in order to get a significant response, one should pay attention to this pattern to avoid cancellation effects.

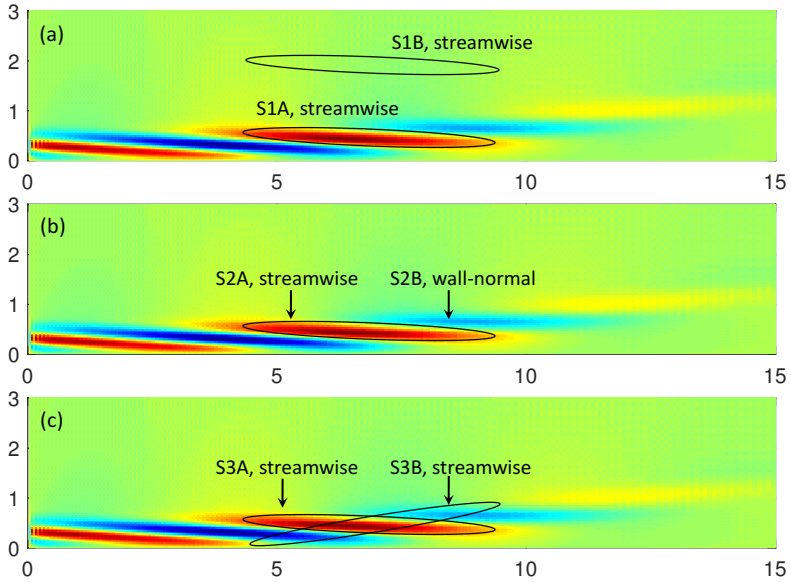


Figure 8.10: (a) Experiment 1: forcing should be concentrated upstream near the wall. (b) Experiment 2: streamwise forcing is more important than wall-normal forcing. (c) Experiment 3: there exists cancellation effect

In order to validate these implications, we will do three control experiments. There are two simulations in each. The first experiment aims to show that forcing should be applied upstream near the wall. The second experiment is designed to validate that streamwise forcing is more critical than wall-normal forcing. The third experiment is intended to demonstrate the cancellation effect due to the pattern in the forcing mode. Details about these three experiments are summarized in Table. 8.1. To better describe the control experiments, we present a visualization of the simulation setup in Fig. 8.10.

8.4 Conclusion

In this work, we performed an input-output analysis for a separated flow past a flat plate. The body forcing (global or local) is taken as input, and the flow field is taken as output. A transfer function that maps input to output can be derived by assuming sinusoidal forcing in time. It is validated that the transfer function

is indeed approximately rank-one. The forcing mode corresponds to the maximum amplification in the response provides insight about optimal forcing location and frequency.

We find that when the flow is forced at the natural frequency of the separation bubble, the response is maximized (in terms of energy). The response is mainly in and behind the trailing edge of the separation bubble, demonstrating the receptivity of the separation bubble to the body forcing. Furthermore, the forcing mode indicates that the optimal forcing location is upstream near the wall. In order to get a significant response, one should carefully place the forcing in the upstream region, to avoid cancellation effects.

In future work, we will perform a numerical simulation to validate these implications. In particular, we would like to verify that (a) the optimal forcing should be concentrated upstream near the wall, (b) streamwise is the optimal forcing direction, and (c) there exists a cancellation effect in the response.

Acknowledgments

This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-17-1-0084.

Bibliography

[1] Eugene L Allower and Kurt Georg. Update methods and their numerical stability. In *Numerical Continuation Methods*, pages 252–265. Springer, 1990.

[2] John David Anderson and J Wendt. *Computational Fluid Dynamics*, volume 206. Springer, 1995.

[3] Hassan Aref and Sivaramakrishnan Balachandar. *A First Course in Computational Fluid Dynamics*. Cambridge University Press, 2017.

[4] Travis Askham and J Nathan Kutz. Variable projection methods for an optimized dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 17(1):380–416, 2018.

[5] Karl Johan Åström and Peter Eykhoff. System identification—a survey. *Automatica*, 7(2):123–162, 1971.

[6] Karl Johan Åström and Richard M Murray. *Feedback Systems: an Introduction for Scientists and Engineers*. Princeton university press, 2010.

[7] Shervin Bagheri. Effects of weak noise on oscillating flows: linking quality factor, Floquet modes, and Koopman spectrum. *Physics of Fluids*, 26:094104, 2014.

[8] David Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

[9] R. Bellman and J. M. Richardson. On some questions arising in the approximate solution of nonlinear differential equations. *Quarterly of Applied Mathematics*, 20:333–339, 1963.

[10] Samir Beneddine, Denis Sipp, Anthony Arnault, Julien Dandois, and Lutz Lesshafft. Conditions for validity of mean flow stability analysis. *Journal of Fluid Mechanics*, 798:485–504, 2016.

[11] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007.

[12] Sylvio R Bistafa. On the development of the Navier-Stokes equation by Navier. *Revista Brasileira de Ensino de Física*, 40(2), 2018.

[13] Geoff Boeing. Visual analysis of nonlinear dynamical systems: chaos, fractals, self-similarity and the limits of prediction. *Systems*, 4(4):37, 2016.

[14] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

[15] Kevin K Chen, Jonathan H Tu, and Clarence W Rowley. Variants of dynamic mode decomposition: boundary condition, Koopman, and Fourier analyses. *Journal of Nonlinear Science*, 22(6):887–915, 2012.

[16] Andrew Cotter, Joseph Keshet, and Nathan Srebro. Explicit approximations of the Gaussian kernel. *arXiv preprint arXiv:1109.4603*, 2011.

[17] Scott T M Dawson, Maziar S Hemati, Matthew O Williams, and Clarence W Rowley. Characterizing and correcting for the effect of sensor noise in the dynamic mode decomposition. *Experiments in Fluids*, 3(57):1–19, 2016.

[18] Eric A Deem, Louis N Cattafesta, Huaijin Yao, Maziar Hemati, Hao Zhang, and Clarence W Rowley. Experimental implementation of modal approaches for autonomous reattachment of separated flows. In *2018 AIAA Aerospace Sciences Meeting*, page 1052, 2018.

[19] Eric A Deem, Louis N Cattafesta, Hao Zhang, Clarence W Rowley, Maziar Hemati, Francois Cadieux, and Rajat Mittal. Identifying dynamic modes of separated flow subject to ZNMF-based control from surface pressure measurements. In *47th AIAA Fluid Dynamics Conference*, page 3309, 2017.

[20] Li Deng and Yang Liu. *Deep Learning in Natural Language Processing*. Springer, 2018.

[21] Zlatko Drmač, Igor Mezić, and Ryan Mohr. Data driven modal decompositions: analysis and enhancements. *SIAM Journal on Scientific Computing*, 40(4):A2253–A2285, 2018.

[22] Daniel Duke, Julio Soria, and Damon Honnery. An error analysis of the dynamic mode decomposition. *Experiments in Fluids*, 52(2):529–542, 2012.

[23] Freeman Dyson. A meeting with Enrico Fermi. *Nature*, 427(6972):297, 2004.

[24] Jeffrey L Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[25] N Benjamin Erichson and Carl Donovan. Randomized low-rank dynamic mode decomposition for motion detection. *Computer Vision and Image Understanding*, 146:40–50, 2016.

[26] N Benjamin Erichson, Lionel Mathelin, J Nathan Kutz, and Steven L Brunton. Randomized dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 18(4):1867–1891, 2019.

[27] Gregory E Fasshauer. Positive definite kernels: past, present and future. *Dolomite Research Notes on Approximation*, 4:21–63, 2011.

[28] Charles L Fefferman. Existence and smoothness of the Navier-Stokes equation. *The Millennium Prize Problems*, 57:67, 2006.

[29] Richard P Feynman, Robert B Leighton, and Matthew Sands. *The Feynman Lectures on Physics, Vol. I: The New Millennium Edition*. Basic books, 2011.

[30] Xavier Garnaud, Lutz Lesshafft, PJ Schmid, and Patrick Huerre. The preferred mode of incompressible jets: linear frequency response analysis. *Journal of Fluid Mechanics*, 716:189–202, 2013.

[31] Alfredo Germani, Costanzo Manes, and Pasquale Palumbo. Polynomial extended Kalman filter. *IEEE Transactions on Automatic Control*, 50(12):2059–2064, 2005.

[32] Ari Glezer and Michael Amitay. Synthetic jets. *Annual Review of Fluid Mechanics*, 34(1):503–529, 2002.

[33] Ari Glezer, Michael Amitay, and Andrew M Honohan. Aspects of low-and high-frequency actuation for aerodynamic flow control. *AIAA Journal*, 43(7):1501–1511, 2005.

[34] Gene H Golub and Charles F Van Loan. *Matrix Computations*. Johns Hopkins University Press, 2012.

[35] F Gómez, HM Blackburn, M Rudman, AS Sharma, and BJ McKeon. A reduced-order model of three-dimensional unsteady flow in a cavity based on the resolvent operator. *Journal of Fluid Mechanics*, 798, 2016.

[36] Arthur Gretton. Introduction to RKHS, and some simple kernel algorithms. *Advanced Topics in Machine Learning. Lecture Conducted from University College London*, 2013.

[37] John C Griffin, Matias Oyarzun, Louis N Cattafesta, Jonathan H Tu, and Clarence W Rowley. Control of a canonical separated flow. In *43rd AIAA Fluid Dynamics Conference*, page 2968, 2013.

[38] Jacob Grosek and J Nathan Kutz. Dynamic mode decomposition for real-time background/foreground separation in video. *arXiv preprint arXiv:1404.7592*, 2014.

[39] John Guckenheimer and Philip Holmes. *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*, volume 42. Springer Science & Business Media, 2013.

[40] Global Wind Energy Council GWEC. Global wind report. 2015. *Brussels: GWEC*, 2017.

[41] William W Hager. Updating the inverse of a matrix. *SIAM Review*, 31(2):221–239, 1989.

[42] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011.

[43] Douglas M Hawkins. The problem of overfitting. *Journal of Chemical Information and Computer Sciences*, 44(1):1–12, 2004.

[44] Maziar Hemati, Eric Deem, Matthew Williams, Clarence W Rowley, and Louis N Cattafesta. Improving separation control with noise-robust variants of dynamic mode decomposition. In *54th AIAA Aerospace Sciences Meeting*, page 1103, 2016.

[45] Maziar S Hemati, Scott TM Dawson, and Clarence W Rowley. Parameter-varying aerodynamics models for aggressive pitching-response prediction. *AIAA Journal*, 55(3):1–9, March 2017.

[46] Maziar S Hemati, Clarence W Rowley, Eric A Deem, and Louis N Cattafesta. De-biasing the dynamic mode decomposition for applied Koopman spectral analysis of noisy datasets. *Theoretical and Computational Fluid Dynamics*, pages 1–20, 2017.

[47] Maziar S Hemati, Matthew O Williams, and Clarence W Rowley. Dynamic mode decomposition for large and streaming datasets. *Physics of Fluids*, 26(11):111701, 2014.

[48] Tien C Hsia. *System Identification: Least-Squares Methods*. Lexington Books, 1977.

[49] Jinah Jeun, Joseph W Nichols, and Mihailo R Jovanović. Input-output analysis of high-speed axisymmetric isothermal jet noise. *Physics of Fluids*, 28(4):047101, 2016.

[50] Forrester T Johnson, Edward N Tinoco, and N Jong Yu. Thirty years of development and application of CFD at Boeing Commercial Airplanes, Seattle. *Computers & Fluids*, 34(10):1115–1151, 2005.

[51] Mihailo R Jovanović and Bassam Bamieh. Componentwise energy amplification in channel flows. *Journal of Fluid Mechanics*, 534:145–183, 2005.

[52] Mihailo R Jovanović, Peter J Schmid, and Joseph W Nichols. Sparsity-promoting dynamic mode decomposition. *Physics of Fluids*, 26(2):024103, 2014.

[53] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

[54] Bernard O Koopman. Hamiltonian systems and transformation in Hilbert space. *Proceedings of the National Academy of Sciences*, 17(5):315–318, 1931.

[55] Jiaqing Kou and Weiwei Zhang. An improved criterion to select dominant modes from dynamic mode decomposition. *European Journal of Mechanics-B/Fluids*, 62:109–129, 2017.

[56] Petros D Koumoutsakos and Igor Mezic. *Control of Fluid Flow*. Springer, 2006.

[57] J Nathan Kutz, Steven L Brunton, Bingni W Brunton, and Joshua L Proctor. *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*. SIAM, 2016.

[58] Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

[59] Mitul Luhar, Ati S Sharma, and Beverley J McKeon. Opposition control within the resolvent analysis framework. *Journal of Fluid Mechanics*, 749:597–626, 2014.

[60] John Leask Lumley. The structure of inhomogeneous turbulent flows. *Atmospheric Turbulence and Radio Wave Propagation*, 1967.

[61] Senka Maćešić, Nelida Črnjarić-Žic, and Igor Mezić. Koopman operator family spectrum for nonautonomous systems. *SIAM Journal on Applied Dynamical Systems*, 17(4):2478–2515, 2018.

[62] Daiki Matsumoto and Thomas Indinger. On-the-fly algorithm for dynamic mode decomposition using incremental singular value decomposition and total least squares. *arXiv preprint arXiv:1703.11004*, 2017.

[63] BJ McKeon and AS Sharma. A critical-layer framework for turbulent pipe flow. *Journal of Fluid Mechanics*, 658:336–382, 2010.

[64] LaTunia Pack Melton, Chung Sheng Yao, and Avi Seifert. Active control of separation from the flap of a supercritical airfoil. *AIAA Journal*, 44(1):34–41, 2006.

[65] James Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London, Series A*, 209:415–446, 1909.

[66] Rajat Mittal, Haibo Dong, Meliha Bozkurttas, FM Najjar, Abel Vargas, and Alfred von Loebbecke. A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries. *Journal of Computational Physics*, 227(10):4825–4852, 2008.

[67] Rajat Mittal, Rupesh Kotapati, and Louis Cattafesta. Numerical study of resonant interactions and flow control in a canonical separated flow. In *43rd AIAA Aerospace Sciences Meeting and Exhibit*, page 1261, 2005.

[68] Rajat Mittal and Rupesh B Kotapati. Resonant mode interaction in a canonical separated flow. In *IUTAM Symposium on Laminar-Turbulent Transition*, pages 341–348. Springer, 2006.

[69] Abdelouahab Mohammed-Taifour and Julien Weiss. Unsteadiness in a large turbulent separation bubble. *Journal of Fluid Mechanics*, 799:383–412, 2016.

[70] Antonios Monokrousos, Espen Åkervik, Luca Brandt, and Dan S Henningson. Global three-dimensional optimal disturbances in the blasius boundary-layer flow using time-steppers. *Journal of Fluid Mechanics*, 650:181–214, 2010.

[71] Oliver Nelles. *Nonlinear System Identification: from Classical Approaches to Neural Networks and Fuzzy Models*. Springer Science & Business Media, 2013.

[72] Bernd R Noack, Witold Stankiewicz, Marek Morzyński, and Peter J Schmid. Recursive dynamic mode decomposition of transient and post-transient wake flows. *Journal of Fluid Mechanics*, 809:843–872, 2016.

[73] Shaowu Pan and Karthik Duraisamy. On the structure of time-delay embedding in linear models of non-linear dynamical systems. *arXiv preprint arXiv:1902.05198*, 2019.

[74] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[75] Joshua L Proctor, Steven L Brunton, and J Nathan Kutz. Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems*, 15(1):142–161, 2016.

[76] Reni Raju, Rajat Mittal, and Louis Cattafesta. Dynamics of airfoil separation control using zero-net mass-flux forcing. *AIAA Journal*, 46(12):3103–3115, 2008.

[77] Waseem Rawat and Zenghui Wang. Deep convolutional neural networks for image classification: a comprehensive review. *Neural Computation*, 29(9):2352–2449, 2017.

[78] Clarence W Rowley, Tim Colonius, and Richard M Murray. Model reduction for compressible flows using POD and Galerkin projection. *Physica D: Nonlinear Phenomena*, 189(1-2):115–129, 2004.

[79] Clarence W Rowley and Scott TM Dawson. Model reduction for flow analysis and control. *Annual Review of Fluid Mechanics*, 49:387–417, 2017.

[80] Clarence W Rowley, Igor Mezić, Shervin Bagheri, Philipp Schlatter, and Dan S Henningson. Spectral analysis of nonlinear flows. *Journal of Fluid Mechanics*, 641:115–127, 2009.

[81] Peter J Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28, 2010.

[82] Jürgen Schmidhuber. Deep learning in neural networks: an overview. *Neural Networks*, 61:85–117, 2015.

[83] Clint Scovel, Don Hush, Ingo Steinwart, and James Theiler. Radial kernels and their reproducing kernel Hilbert spaces. *Journal of Complexity*, 26(6):641–660, 2010.

[84] Avi Seifert, David Greenblatt, and Israel J Wygnanski. Active separation control: an overview of Reynolds and Mach numbers effects. *Aerospace Science and Technology*, 8(7):569–582, 2004.

[85] Avraham Seifert, A Darabi, and Israel Wyganski. Delay of airfoil stall by periodic excitation. *Journal of Aircraft*, 33(4):691–698, 1996.

[86] JH Seo, F Cadieux, R Mittal, E Deem, and L Cattafesta. Effect of synthetic jet modulation schemes on the reduction of a laminar separation bubble. *Physical Review Fluids*, 3(3):033901, 2018.

[87] Suresh P Sethi. What is optimal control theory? In *Optimal Control Theory*, pages 1–26. Springer, 2019.

[88] Jack Sherman and Winifred J Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, 21(1):124–127, 1950.

[89] César Ernesto Silva. *Invitation to Ergodic Theory*, volume 42. American Mathematical Soc., 2008.

[90] Alexander J Smits. *A Physical Introduction to Fluid Mechanics*. Wiley, 2000.

[91] César R Souza. Kernel functions for machine learning applications. *Creative Commons Attribution-Noncommercial-Share Alike*, 3, 2010.

[92] Robert F Stengel. *Optimal Control and Estimation*. Courier Corporation, 1994.

[93] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: an Introduction*. MIT press, 2018.

[94] Patrick Tabeling. *Introduction to Microfluidics*. OUP Oxford, 2005.

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

[96] Floris Takens. Detecting strange attractors in turbulence. In *Dynamical Systems and Turbulence, Warwick 1980*, pages 366–381. Springer, 1981.

[97] Emmanuel Trélat. Optimal control and applications to aerospace: some results and challenges. *Journal of Optimization Theory and Applications*, 154(3):713–758, 2012.

[98] Jonathan H Tu, Clarence W Rowley, J Nathan Kutz, and Jessica K Shang. Spectral analysis of fluid flows using sub-Nyquist-rate PIV data. *Experiments in Fluids*, 55(9):1–13, 2014.

[99] Jonathan H Tu, Clarence W Rowley, Dirk M Luchtenburg, Steven L Brunton, and J Nathan Kutz. On dynamic mode decomposition: theory and applications. *Journal of Computational Dynamics*, 1(2):391–421, 2014.

[100] Geoffrey K Vallis. Geophysical fluid dynamics: whence, whither and why? *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 472(2192):20160140, 2016.

[101] V. Verdult and M. Verhaegen. Kernel methods for subspace identification of multivariable LPV and bilinear systems. *Automatica*, 41(9):1557–1565, 2005.

[102] Peter Walters. *An Introduction to Ergodic Theory*, volume 79. Springer Science & Business Media, 2000.

[103] Peter Whitle. *Hypothesis Testing in Time Series Analysis*, volume 4. Almqvist & Wiksells, 1951.

[104] Matthew O Williams, Ioannis G Kevrekidis, and Clarence W Rowley. A data-driven approximation of the Koopman operator: extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, 2015.

[105] Matthew O Williams, Clarence W Rowley, and Ioannis G Kevrekidis. A kernel-based method for data-driven Koopman spectral analysis. *Journal of Computational Dynamics*, 2(2):247–265, 2015.

[106] Max A Woodbury. Inverting modified matrices. *Memorandum Report*, 42(106):336, 1950.

[107] Wen Wu, Jung-Hee Seo, Charles Meneveau, and Rajat Mittal. Response of a laminar separation bubble to forcing with zero-net mass flux jets. In *Proc. 2018 Flow Control Conference, AIAA AVIATION Forum,(AIAA 2018-4018)*, 2018.

[108] A Wynn, DS Pearson, B Ganapathisubramani, and PJ Goulart. Optimal mode decomposition for unsteady flows. *Journal of Fluid Mechanics*, 733:473, 2013.

[109] Chi-An Yeh and Kunihiko Taira. Resolvent-analysis-based design of airfoil separation control. *Journal of Fluid Mechanics*, 867:572–610, 2019.

[110] Elizabeth L Yip. A note on the stability of solving a rank-p modification of a linear system by the Sherman–Morrison–Woodbury formula. *SIAM Journal on Scientific and Statistical Computing*, 7(2):507–513, 1986.

[111] Hao Zhang, Scott Dawson, Clarence W Rowley, Eric A Deem, and Louis N Cattafesta. Evaluating the accuracy of the dynamic mode decomposition. *Journal of Computational Dynamics*, 2019.

[112] Hao Zhang and Clarence W Rowley. Online DMD and window DMD implementation in Matlab and Python. *Github, https://github.com/haozhg/odmd*, 2017.

[113] Hao Zhang, Clarence W Rowley, Eric A Deem, and Louis N Cattafesta. Online dynamic mode decomposition for time-varying systems. *SIAM Journal on Applied Dynamical Systems*, 18(3):1586–1609, 2019.

[114] Hao Zhang, Clarence W Rowley, Wen Wu, Charles Meneveau, and Rajat Mittal. Input-output analysis of a separated flow past a flat plate. In *AIAA Scitech 2019 Forum*, page 0880, 2019.