Chapter 1
Dynamic Mode Decomposition: An Introduction

The data-driven modeling and control of complex systems is a rapidly evolving field with great potential to transform the engineering, biological, and physical sciences. There is unprecedented availability of high-fidelity measurements from historical records, numerical simulations, and experimental data, and although data is abundant, models often remain elusive. Modern systems of interest, such as a turbulent fluid, an epidemiological system, a network of neurons, financial markets, or the climate, may be characterized as high-dimensional, nonlinear dynamical systems that exhibit rich multiscale phenomena in both space and time. However complex, many of these systems evolve on a low-dimensional attractor that may be characterized by spatiotemporal coherent structures. In this chapter, we will introduce the topic of this book, dynamic mode decomposition (DMD), which is a powerful new technique for the discovery of dynamical systems from high-dimensional data.

The DMD method originated in the fluid dynamics community as a method to decompose complex flows into a simple representation based on spatiotemporal coherent structures. Schmid and Sesterhenn [250] and Schmid [247] first defined the DMD algorithm and demonstrated its ability to provide insights from high-dimensional fluids data. The growing success of DMD stems from the fact that it is an *equation-free*, data-driven method capable of providing an accurate decomposition of a complex system into spatiotemporal coherent structures that may be used for short-time future-state prediction and control. More broadly, DMD has quickly gained popularity since Mezić et al. [196, 194, 195] and Rowley et al. [235] showed that it is connected to the underlying nonlinear dynamics through Koopman operator theory [162] and is readily interpretable using standard dynamical systems techniques.

The development of DMD is timely due to the concurrent rise of data science, encompassing a broad range of techniques, from machine learning and statistical regression to computer vision and compressed sensing. Improved algorithms, abundant data, vastly expanded computational resources, and interconnectedness of data streams make this a fertile ground for rapid development. In this chapter, we will introduce the core DMD algorithm, provide a broader historical context, and lay the mathematical foundation for the future innovations and applications of DMD discussed in the remaining chapters.

1.1 ▪ DMD

DMD is the featured method of this book. At its core, the method can be thought of as an ideal combination of spatial dimensionality-reduction techniques, such as the proper orthogonal decomposition (POD),[1] with Fourier transforms in time. Thus, correlated spatial modes are also now associated with a given temporal frequency, possibly with a growth or decay rate. The method relies simply on collecting snapshots of data $\mathbf{x}_k$ from a dynamical system at a number of times t_k, where $k = 1, 2, 3, \ldots, m$. As we will show, DMD is algorithmically a regression of data onto locally linear dynamics $\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k$, where $\mathbf{A}$ is chosen to minimize $\|\mathbf{x}_{k+1} - \mathbf{A}\mathbf{x}_k\|_2$ over the $k = 1, 2, 3, \ldots, m-1$ snapshots. The advantages of this method are that it is very simple to execute and it makes almost no assumptions about the underlying system. The cost of the algorithm is a singular value decomposition (SVD) of the snapshot matrix constructed from the data $\mathbf{x}_k$.

DMD has a number of uses and interpretations. Specifically, the DMD algorithm can generally be thought to enable three primary tasks.

I. Diagnostics. At its inception, the DMD algorithm was used as a diagnostic tool to characterize complex fluid flows [247, 235]. In particular, the algorithm extracts key low-rank spatiotemporal features of many high-dimensional systems, allowing for physically interpretable results in terms of spatial structures and their associated temporal responses. Interestingly, this is still perhaps the primary function of DMD in many application areas. The diagnostic nature of DMD allows for the data-driven discovery of fundamental, low-rank structures in complex systems analogous to POD analysis in fluid flows, plasma physics, atmospheric modeling, etc.

II. State estimation and future-state prediction. A more sophisticated and challenging use of the DMD algorithm is associated with using the spatiotemporal structures that are dominant in the data to construct dynamical models of the underlying processes observed. This is a much more difficult task, especially as DMD is limited to constructing the best-fit (least-square) linear dynamical system to the nonlinear dynamical system generating the data. Thus, unlike the diagnostic objective, the goal is to anticipate the state of the system in a regime where no measurements were made. Confounding the regressive nature of DMD is the fact that the underlying dynamics can exhibit multiscale dynamics in both time and space. Regardless, there are a number of key strategies, including intelligent sampling of the data and updating the regression, that allow DMD to be effective for generating a useful linear dynamical model. This generative model approach can then be used for future-state predictions of the dynamical systems and has been used with success in many application areas.

III. Control. Enabling viable and robust control strategies directly from data sampling is the ultimate, and most challenging, goal of the DMD algorithm. Given that we are using a linear dynamical model to predict the future of a nonlinear dynamical system, it is reasonable to expect that there is only a limited, perhaps short-time, window in the future where the two models will actually agree. The hope is that this accurate prediction window is long enough in duration to enable a control decision capable of influencing the future state of the system. The DMD algorithm in this case

[1] POD is often computed using the singular value decomposition (SVD). It is also known as principal component analysis (PCA) in statistics [214, 147], empirical orthogonal functions (EOF) in climate and meteorology [181], the discrete Karhunen–Loeve transform, and the Hotelling transform [134, 135].

allows for a completely data-driven approach to control theory, thus providing a compelling mathematical framework for controlling complex dynamical systems whose governing equations are not known or are difficult to model computationally.

When using the DMD method, one should always be mindful of the intended use and the expected outcome. Further, although it seems quite obvious, the success of the DMD algorithm will depend largely on which of the above tasks one is attempting to achieve. Throughout this book, many of the chapters will address one, two, or all three of the above objectives. In some applications, it is only reasonable at this point to use DMD as a diagnostic tool. In other applications, limited success has been achieved even for the task of control. Undoubtedly, many researchers are working hard to move emerging application areas through this task list toward achieving accurate modeling and control of dynamical systems in an equation-free framework.

1.2 ▪ Formulating the DMD architecture

In the DMD architecture, we typically consider data collected from a dynamical system

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, t; \mu), \tag{1.1}$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ is a vector representing the state of our dynamical system at time t, μ contains parameters of the system, and $\mathbf{f}(\cdot)$ represents the dynamics. Generally, a dynamical system is represented as a coupled system of ordinary differential equations that are often nonlinear. The state $\mathbf{x}$ is typically quite large, having dimension $n \gg 1$; this state may arise as the discretization of a partial differential equation at a number of discrete spatial locations. Finally, the continuous-time dynamics from (1.1) may also induce a corresponding discrete-time representation, in which we sample the system every Δt in time and denote the time as a subscript so that $\mathbf{x}_k = \mathbf{x}(k\Delta t)$. We denote the discrete-time *flow* map obtained by evolving (1.1) for Δt by $\mathbf{F}$:

$$\mathbf{x}_{k+1} = \mathbf{F}(\mathbf{x}_k). \tag{1.2}$$

Measurements of the system

$$\mathbf{y}_k = \mathbf{g}(\mathbf{x}_k) \tag{1.3}$$

are collected at times t_k from $k = 1, 2, \ldots, m$ for a total of m measurement times. In many applications, the measurements are simply the state, so that $\mathbf{y}_k = \mathbf{x}_k$. The data assimilation community often represents these measurements by the implicit expression $G(\mathbf{x}, t) = 0$. The initial conditions are prescribed as $\mathbf{x}(t_0) = \mathbf{x}_0$.

As already highlighted, $\mathbf{x}$ is an n-dimensional vector ($n \gg 1$) that arises from either discretization of a complex system, or in the case of applications such as video streams, from the total number of pixels in a given frame. The governing equations and initial condition specify a well-posed initial value problem.

In general, it is not possible to construct a solution to the governing nonlinear evolution (1.1), so numerical solutions are used to evolve to future states. The DMD framework takes the equation-free perspective, where the dynamics $\mathbf{f}(\mathbf{x}, t; \mu)$ may be unknown. Thus, data measurements of the system alone are used to approximate the dynamics and predict the future state. The DMD procedure constructs the proxy, approximate locally linear dynamical system

$$\frac{d\mathbf{x}}{dt} = \mathcal{A}\mathbf{x} \tag{1.4}$$

with initial condition $\mathbf{x}(0)$ and well-known solution [33]

$$\mathbf{x}(t) = \sum_{k=1}^{n} \boldsymbol{\phi}_k \exp(\omega_k t) b_k = \boldsymbol{\Phi} \exp(\boldsymbol{\Omega} t)\mathbf{b}, \tag{1.5}$$

where $\boldsymbol{\phi}_k$ and ω_k are the eigenvectors and eigenvalues of the matrix $\mathcal{A}$, and the coefficients b_k are the coordinates of $\mathbf{x}(0)$ in the eigenvector basis.

Given continuous dynamics as in (1.4), it is always possible to describe an analogous discrete-time system sampled every Δt in time:

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k, \tag{1.6}$$

where

$$\mathbf{A} = \exp(\mathcal{A}\Delta t). \tag{1.7}$$

$\mathcal{A}$ refers to the matrix in the continuous-time dynamics from (1.4). The solution to this system may be expressed simply in terms of the eigenvalues λ_k and eigenvectors $\boldsymbol{\phi}_k$ of the discrete-time map $\mathbf{A}$:

$$\mathbf{x}_{k+1} = \sum_{j=1}^{r} \boldsymbol{\phi}_j \lambda_j^k b_j = \boldsymbol{\Phi}\boldsymbol{\Lambda}^k \mathbf{b}. \tag{1.8}$$

As before, $\mathbf{b}$ are the coefficients of the initial condition $\mathbf{x}_1$ in the eigenvector basis, so that $\mathbf{x}_1 = \boldsymbol{\Phi}\mathbf{b}$. The DMD algorithm produces a low-rank eigendecomposition (1.8) of the matrix $\mathbf{A}$ that optimally fits the measured trajectory $\mathbf{x}_k$ for $k = 1, 2, \ldots, m$ in a least-square sense so that

$$\|\mathbf{x}_{k+1} - \mathbf{A}\mathbf{x}_k\|_2 \tag{1.9}$$

is minimized across all points for $k = 1, 2, \ldots, m-1$. The optimality of the approximation holds only over the sampling window where $\mathbf{A}$ is constructed, and the approximate solution can be used to not only make future state predictions but also decompose the dynamics into various time scales, since the λ_k are prescribed.

Arriving at the optimally constructed matrix $\mathbf{A}$ for the approximation (1.6) and the subsequent eigendecomposition in (1.8) is the focus of the remainder of this introductory chapter. In subsequent chapters, a deeper theoretical understanding will be pursued, along with various demonstrations of the power of this methodology.

To minimize the approximation error (1.9) across all snapshots from $k = 1, 2, \ldots, m$, it is possible to arrange the m snapshots into two large data matrices:

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_{m-1} \\ | & | & & | \end{bmatrix}, \tag{1.10a}$$

$$\mathbf{X}' = \begin{bmatrix} | & | & & | \\ \mathbf{x}_2 & \mathbf{x}_3 & \cdots & \mathbf{x}_{m} \\ | & | & & | \end{bmatrix}. \tag{1.10b}$$

Recall that these data snapshots are likely sampled from a nonlinear dynamical system (1.1), although we are finding an optimal local linear approximation. The locally linear approximation (1.6) may be written in terms of these data matrices as

$$\mathbf{X}' \approx \mathbf{A}\mathbf{X}. \tag{1.11}$$

The best-fit **A** matrix is given by

$$\mathbf{A} = \mathbf{X}'\mathbf{X}^{\dagger}, \tag{1.12}$$

where † is the Moore–Penrose pseudoinverse. This solution minimizes the error

$$||\mathbf{X}' - \mathbf{A}\mathbf{X}||_F, \tag{1.13}$$

where $||\cdot||_F$ is the Frobenius norm, given by

$$||\mathbf{X}||_F = \sqrt{\sum_{j=1}^{n}\sum_{k=1}^{m} X_{jk}{}^2}. \tag{1.14}$$

It is important to note that (1.13) and the solution (1.12) may be thought of as a linear regression of data onto the dynamics given by **A**. However, there is a key difference between DMD and alternative regression-based system identification and model reduction techniques. Importantly, we are assuming that the snapshots $\mathbf{x}_k$ in our data matrix **X** are high dimensional so that the matrix is *tall and skinny*, meaning that the size n of a snapshot is larger than the number $m-1$ of snapshots. The matrix **A** may be high dimensional; if $n = 10^6$, then **A** has 10^{12} elements, so that it may be difficult to represent or decompose. However, the rank of **A** is at most $m-1$, since it is constructed as a linear combination of the $m-1$ columns of $\mathbf{X}'$. Instead of solving for **A** directly, we first project our data onto a low-rank subspace defined by at most $m-1$ POD modes and then solve for a low-dimensional evolution $\tilde{\mathbf{A}}$ that evolves on these POD mode coefficients. The DMD algorithm then uses this low-dimensional operator $\tilde{\mathbf{A}}$ to reconstruct the leading nonzero eigenvalues and eigenvectors of the full-dimensional operator **A** without ever explicitly computing **A**. This is discussed in § 1.3 below.

1.2.1 ▪ Defining DMD

The DMD method provides a spatiotemporal decomposition of data into a set of dynamic modes that are derived from snapshots or measurements of a given system in time. A schematic overview is provided in Figure 1.1. The mathematics underlying the extraction of dynamic information from time-resolved snapshots is closely related to the idea of the Arnoldi algorithm [247], one of the workhorses of fast computational solvers. The data collection process involves two parameters:

$$\begin{aligned} n &= \text{number of spatial points saved per time snapshot,} \\ m &= \text{number of snapshots taken.} \end{aligned}$$

The DMD algorithm was originally designed to collect data at regularly spaced intervals of time, as in (1.10). However, new innovations allow for both sparse spatial [48] and sparse temporal [289] collections of data, as well as irregularly spaced collection times. Indeed, Tu et al. [290] provides the most modern definition of the DMD method and algorithm.

Definition: Dynamic mode decomposition (Tu et al. 2014 [290]): *Suppose we have a*

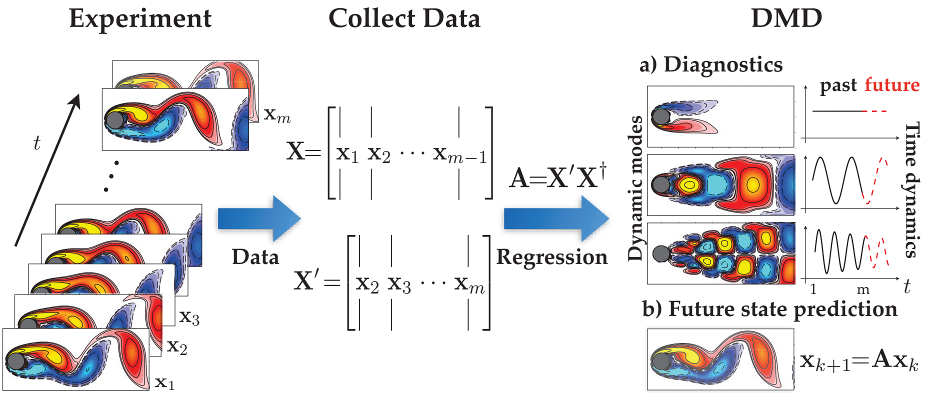
Experiment
Collect Data
DMD
t
$\mathbf{x}_m$
$\mathbf{x}_3$
$\mathbf{x}_2$
$\mathbf{x}_1$
Data
$\mathbf{X}=\begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_{m-1} \\ | & | & & | \end{bmatrix}$
$\mathbf{X}'=\begin{bmatrix} | & | & & | \\ \mathbf{x}_2 & \mathbf{x}_3 & \cdots & \mathbf{x}_m \\ | & | & & | \end{bmatrix}$
$\mathbf{A}=\mathbf{X}'\mathbf{X}^\dagger$
Regression
a) Diagnostics
Dynamic modes
past future
Time dynamics
1
m
t
b) Future state prediction
$\mathbf{x}_{k+1}=\mathbf{A}\mathbf{x}_k$

Figure 1.1. *Schematic overview of DMD on a fluid flow example that will be explored further in Chapter 2. Note that the regression step does not typically construct* $\mathbf{A}$ *but instead constructs* $\tilde{\mathbf{A}}$*, which evolves the dynamics on a low-rank subspace via POD. The eigendecomposition of* $\tilde{\mathbf{A}}$ *is then used to approximate the eigendecomposition of the high-dimensional matrix* $\mathbf{A}$.

dynamical system (1.1) *and two sets of data,*

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_{m-1} \\ | & | & & | \end{bmatrix}, \tag{1.15a}$$

$$\mathbf{X}' = \begin{bmatrix} | & | & & | \\ \mathbf{x}'_1 & \mathbf{x}'_2 & \cdots & \mathbf{x}'_{m-1} \\ | & | & & | \end{bmatrix}, \tag{1.15b}$$

so that $\mathbf{x}'_k = \mathbf{F}(\mathbf{x}_k)$*, where* $\mathbf{F}$ *is the map in* (1.2) *corresponding to the evolution of* (1.1) *for time* Δt*. DMD computes the leading eigendecomposition of the best-fit linear operator* $\mathbf{A}$ *relating the data* $\mathbf{X}' \approx \mathbf{A}\mathbf{X}$*:*

$$\mathbf{A} = \mathbf{X}'\mathbf{X}^\dagger. \tag{1.16}$$

The DMD modes, also called dynamic modes, are the eigenvectors of $\mathbf{A}$*, and each DMD mode corresponds to a particular eigenvalue of* $\mathbf{A}$.

1.2.2 ▪ DMD and the Koopman operator

The DMD method approximates the modes of the so-called *Koopman operator*. The Koopman operator is a linear, infinite-dimensional operator that represents the action of a nonlinear dynamical system on the Hilbert space of measurement functions of the state [235, 195]. It is important to note that the Koopman operator does not rely on linearization of the dynamics but instead represents the flow of the dynamical system on measurement functions as an infinite-dimensional operator.

The DMD method can be viewed as computing, from the experimental data, the eigenvalues and eigenvectors (low-dimensional modes) of a finite-dimensional linear model that approximates the infinite-dimensional Koopman operator. Since the operator is linear, the decomposition gives the growth rates and frequencies associated with each mode. If the underlying dynamics in (1.1) are linear, then the DMD method recovers the leading eigenvalues and eigenvectors normally computed using standard solution methods for linear differential equations.

Mathematically, the Koopman operator $\mathcal{K}$ is an infinite-dimensional linear operator that acts on the Hilbert space $\mathcal{H}$ containing *all* scalar-valued measurement functions $g : \mathbb{C}^n \to \mathbb{C}$ of the state $\mathbf{x}$. The action of the Koopman operator on a measurement function g equates to composition of the function g with the flow map $\mathbf{F}$:

$$\mathcal{K} g = g \circ \mathbf{F} \tag{1.17a}$$

$$\implies \quad \mathcal{K} g(\mathbf{x}_k) = g(\mathbf{F}(\mathbf{x}_k)) = g(\mathbf{x}_{k+1}). \tag{1.17b}$$

In other words, the Koopman operator, also known as the composition operator, advances measurements along with the flow $\mathbf{F}$. This is incredibly powerful and general, since it holds for all measurement functions g evaluated at any state vector $\mathbf{x}$.

The approximation of the Koopman operator is at the heart of the DMD methodology. As already stated, the mapping over Δt is linear even though the underlying dynamics that generated $\mathbf{x}_k$ may be nonlinear. It should be noted that this is fundamentally different than linearizing the dynamics. However, the quality of any finite-dimensional approximation to the Koopman operator depends on the measurements $\mathbf{y} = \mathbf{g}(\mathbf{x})$ chosen. This will be discussed in depth in Chapter 3.

1.3 ▪ The DMD algorithm

In practice, when the state dimension n is large, the matrix $\mathbf{A}$ may be intractable to analyze directly. Instead, DMD circumvents the eigendecomposition of $\mathbf{A}$ by considering a rank-reduced representation in terms of a POD-projected matrix $\tilde{\mathbf{A}}$. The DMD algorithm, also shown in Algorithm **1.1** as a function, proceeds as follows [290]:

1. First, take the singular value decomposition (SVD) of $\mathbf{X}$ [283]:

   $$\mathbf{X} \approx \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^*, \tag{1.18}$$

   where $*$ denotes the conjugate transpose, $\mathbf{U} \in \mathbb{C}^{n \times r}$, $\Sigma \in \mathbb{C}^{r \times r}$, and $\mathbf{V} \in \mathbb{C}^{m \times r}$. Here r is the rank of the reduced SVD approximation to $\mathbf{X}$. The left singular vectors $\mathbf{U}$ are POD modes. The columns of $\mathbf{U}$ are orthonormal, so $\mathbf{U}^*\mathbf{U} = \mathbf{I}$; similarly, $\mathbf{V}^*\mathbf{V} = \mathbf{I}$.

   The SVD reduction in (1.18) is exploited at this stage in the algorithm to perform a low-rank truncation of the data. Specifically, if low-dimensional structure is present in the data, the singular values of Σ will decrease sharply to zero with perhaps only a limited number of dominant modes. A principled way to truncate noisy data is given by the recent hard-thresholding algorithm of Gavish and Donoho [106], as discussed in § 8.2.

2. The matrix $\mathbf{A}$ from (1.16) may be obtained by using the pseudoinverse of $\mathbf{X}$ obtained via the SVD:

   $$\mathbf{A} = \mathbf{X}'\mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^*. \tag{1.19}$$

   In practice, it is more efficient computationally to compute $\tilde{\mathbf{A}}$, the $r \times r$ projection of the full matrix $\mathbf{A}$ onto POD modes:

   $$\tilde{\mathbf{A}} = \mathbf{U}^*\mathbf{A}\mathbf{U} = \mathbf{U}^*\mathbf{X}'\mathbf{V}\boldsymbol{\Sigma}^{-1}. \tag{1.20}$$

   The matrix $\tilde{\mathbf{A}}$ defines a low-dimensional linear model of the dynamical system on POD coordinates:

   $$\tilde{\mathbf{x}}_{k+1} = \tilde{\mathbf{A}}\tilde{\mathbf{x}}_k. \tag{1.21}$$

   It is possible to reconstruct the high-dimensional state $\mathbf{x}_k = \mathbf{U}\tilde{\mathbf{x}}_k$.

3. Compute the eigendecomposition of $\tilde{\mathbf{A}}$:

$$\tilde{\mathbf{A}}\mathbf{W} = \mathbf{W}\boldsymbol{\Lambda}, \tag{1.22}$$

   where columns of $\mathbf{W}$ are eigenvectors and $\boldsymbol{\Lambda}$ is a diagonal matrix containing the corresponding eigenvalues λ_k.

4. Finally, we may reconstruct the eigendecomposition of $\mathbf{A}$ from $\mathbf{W}$ and $\boldsymbol{\Lambda}$. In particular, the eigenvalues of $\mathbf{A}$ are given by $\boldsymbol{\Lambda}$ and the eigenvectors of $\mathbf{A}$ (DMD modes) are given by columns of $\boldsymbol{\Phi}$:

$$\boldsymbol{\Phi} = \mathbf{X}'\mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{W}. \tag{1.23}$$

Note that (1.23) from [290] differs from the formula $\boldsymbol{\Phi} = \mathbf{U}\mathbf{W}$ in [247], although these will tend to converge if $\mathbf{X}$ and $\mathbf{X}'$ have the same column spaces. The modes in (1.23) are often called *exact DMD modes*, because it was proven in Tu et al. [290] that these are exact eigenvectors of the matrix $\mathbf{A}$. The modes $\boldsymbol{\Phi} = \mathbf{U}\mathbf{W}$ are referred to as *projected DMD modes* [247]. To find a dynamic mode of $\mathbf{A}$ associated with a zero eigenvalue $\lambda_k = 0$, the exact formulation in (1.23) may be used if $\boldsymbol{\phi} = \mathbf{X}'\mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{w} \neq 0$. Otherwise, the projected DMD formulation $\boldsymbol{\phi} = \mathbf{U}\mathbf{w}$ should be used [290].

With the low-rank approximations of both the eigenvalues and the eigenvectors in hand, the projected future solution can be constructed for all time in the future. By first rewriting for convenience $\omega_k = \ln(\lambda_k)/\Delta t$, the approximate solution at all future times is given by

$$\mathbf{x}(t) \approx \sum_{k=1}^{r} \boldsymbol{\phi}_k \exp(\omega_k t) b_k = \boldsymbol{\Phi} \exp(\boldsymbol{\Omega} t)\mathbf{b}, \tag{1.24}$$

where b_k is the initial amplitude of each mode, $\boldsymbol{\Phi}$ is the matrix whose columns are the DMD eigenvectors $\boldsymbol{\phi}_k$, and $\boldsymbol{\Omega} = \text{diag}(\omega)$ is a diagonal matrix whose entries are the eigenvalues ω_k. The eigenvectors $\boldsymbol{\phi}_k$ are the same size as the state $\mathbf{x}$, and $\mathbf{b}$ is a vector of the coefficients b_k.

As discussed earlier, it is possible to interpret (1.24) as the least-square fit, or regression, of a best-fit linear dynamical system $\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k$ for the data sampled. The matrix $\mathbf{A}$ is constructed so that $||\mathbf{x}_{k+1} - \mathbf{A}\mathbf{x}_k||_2$ is minimized across all snapshots.

It only remains to compute the initial coefficient values b_k. If we consider the initial snapshot $\mathbf{x}_1$ at time $t_1 = 0$, then (1.24) gives $\mathbf{x}_1 = \boldsymbol{\Phi}\mathbf{b}$. The matrix of eigenvectors $\boldsymbol{\Phi}$ is generically not a square matrix so that the initial conditions

$$\mathbf{b} = \boldsymbol{\Phi}^{\dagger}\mathbf{x}_1 \tag{1.25}$$

can be found using a pseudoinverse. Indeed, $\boldsymbol{\Phi}^{\dagger}$ denotes the Moore–Penrose pseudoinverse that can be accessed in MATLAB via the `pinv` command. The pseudoinverse is equivalent to finding the best-fit solution $\mathbf{b}$ in the least-squares sense.

A MATLAB implementation of the DMD algorithm (`DMD.m`) is provided in Algorithm **1.1**. The DMD algorithm presented here takes advantage of low dimensionality in the data to make a low-rank approximation of the linear mapping that best approximates the nonlinear dynamics of the data collected for the system. Once this is done, a prediction of the future state of the system is achieved for all time. Unlike the POD-Galerkin method, which requires solving a low-rank set of dynamical quantities to predict the future state, no additional work is required for the future state prediction besides plugging the desired future time into (1.24). Thus, the advantages of DMD revolve around the fact that (i) it is an equation-free architecture and (ii) a future-state prediction is possible to construct for any time t (of course, provided the DMD approximation holds).

A key benefit of the DMD framework is the simple formulation in terms of well-established techniques from linear algebra. This makes DMD amenable to a variety of powerful extensions, including multiresolution (Chapter 5), actuation and control (Chapter 6), time-delay coordinates and probabilistic formulations (Chapter 7), noise mitigation (Chapter 8), and sparse measurements via compressed sensing (Chapter 9). Another important advantage of DMD is that it is formulated *entirely* in terms of measurement data. For this reason, it may be applied to a broad range of applications, including fluid dynamics (Chapter 2), video processing (Chapter 4), epidemiology (Chapter 11), neuroscience (Chapter 12), and finance (Chapter 13).

***Algorithm* 1.1. DMD function (`DMD.m`).**

```
function [Phi,omega,lambda,b,Xdmd] = DMD(X1,X2,r,dt)
% function [Phi,omega,lambda,b,Xdmd] = DMD(X1,X2,r,dt)
% Computes the Dynamic Mode Decomposition of X1, X2
%
% INPUTS:
% X1 = X, data matrix
% X2 = X', shifted data matrix
% Columns of X1 and X2 are state snapshots
% r = target rank of SVD
% dt = time step advancing X1 to X2 (X to X')
%
% OUTPUTS:
% Phi, the DMD modes
% omega, the continuous-time DMD eigenvalues
% lambda, the discrete-time DMD eigenvalues
% b, a vector of magnitudes of modes Phi
% Xdmd, the data matrix reconstructed by Phi, omega, b

%% DMD
[U, S, V] = svd(X1, 'econ');
r = min(r, size(U,2));

U_r = U(:, 1:r); % truncate to rank-r
S_r = S(1:r, 1:r);
V_r = V(:, 1:r);
Atilde = U_r' * X2 * V_r / S_r; % low-rank dynamics
[W_r, D] = eig(Atilde);
Phi = X2 * V_r / S_r * W_r; % DMD modes

lambda = diag(D); % discrete-time eigenvalues
omega = log(lambda)/dt; % continuous-time eigenvalues

%% Compute DMD mode amplitudes b
x1 = X1(:, 1);
b = Phi\x1;

%% DMD reconstruction
mm1 = size(X1, 2); % mm1 = m - 1
time_dynamics = zeros(r, mm1);
```

```
t = (0:mm1-1)*dt; % time vector
for iter = 1:mm1,
    time_dynamics(:,iter) = (b.*exp(omega*t(iter)));
end;
Xdmd = Phi * time_dynamics;
```

1.3.1 ▪ Historical perspective on the DMD algorithm

Historically, the original DMD algorithm [247] was formulated in the context of its connection to Krylov subspaces and the Arnoldi algorithm. In this context, it is assumed that the data is sampled from a single trajectory in phase space, although this assumption has subsequently been relaxed in the general definition presented in § 1.2.1. For completeness, as well as understanding this connection, we present the DMD algorithm as originally presented by Schmid [247]. To illustrate the algorithm, consider the following regularly spaced sampling in time:

$$\text{data collection times:} \ \ t_{k+1} = t_k + \Delta t, \tag{1.26}$$

where the collection time starts at t_1 and ends at t_m and the interval between data collection times is Δt.

As before, the data snapshots are then arranged into an $n \times m$ matrix

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}(t_1) & \mathbf{x}(t_2) & \cdots & \mathbf{x}(t_m) \\ | & | & & | \end{bmatrix}, \tag{1.27}$$

where the vector $\mathbf{x}$ contains the n measurements of the state variable of the system of interest at the data collection points. The objective is to mine the data matrix $\mathbf{X}$ for important dynamical information. For the purposes of the DMD method, the following matrix is also defined:

$$\mathbf{X}_j^k = \begin{bmatrix} | & | & & | \\ \mathbf{x}(t_j) & \mathbf{x}(t_{j+1}) & \cdots & \mathbf{x}(t_k) \\ | & | & & | \end{bmatrix}. \tag{1.28}$$

Thus, this matrix includes columns j through k of the original data matrix.

To construct the Koopman operator that best represents the data collected, the matrix $\mathbf{X}_1^m$ is considered:

$$\mathbf{X}_1^m = \begin{bmatrix} | & | & & | \\ \mathbf{x}(t_1) & \mathbf{x}(t_2) & \cdots & \mathbf{x}(t_m) \\ | & | & & | \end{bmatrix}. \tag{1.29}$$

Making use of (1.6), this matrix reduces to

$$\mathbf{X}_1^m = \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{A}\mathbf{x}_1 & \cdots & \mathbf{A}^{m-1}\mathbf{x}_1 \\ | & | & & | \end{bmatrix}. \tag{1.30}$$

Here is where the DMD method connects to Krylov subspaces and the Arnoldi algorithm. Specifically, the columns of $\mathbf{X}_1^m$ are each elements in a Krylov space. This matrix attempts to fit the first $m-1$ data collection points using the matrix $\mathbf{A}$ that approximates the Koopman operator. In the DMD technique, the final data point $\mathbf{x}_m$ is represented, as best as possible, in terms of this Krylov basis; thus

$$\mathbf{x}_m = \sum_{k=1}^{m-1} b_k \mathbf{x}_k + \mathbf{r}, \tag{1.31}$$

where the b_k are the coefficients of the Krylov space vectors and $\mathbf{r}$ is the residual (or error) that lies outside (orthogonal to) the Krylov space. Ultimately, this best fit to the data using this DMD procedure will be done in an ℓ_2 sense using a pseudoinverse.

In this notation, we have the key result (compare to the definition (1.16)):

$$\mathbf{X}_2^m = \mathbf{A}\mathbf{X}_1^{m-1}, \tag{1.32}$$

where the operator $\mathbf{A}$ is chosen to minimize the Frobenius norm of $||\mathbf{X}_2^m - \mathbf{A}\mathbf{X}_1^{m-1}||_F$. In other words, the operator $\mathbf{A}$ advances each snapshot column in $\mathbf{X}_1^{m-1}$ a single time step, Δt, resulting in the future snapshot columns in $\mathbf{X}_2^m$. The DMD outlined previously can then be enacted. This definition of DMD is similar to the more modern definition with $\mathbf{X} \rightarrow \mathbf{X}_1^{m-1}$, $\mathbf{X}' \rightarrow \mathbf{X}_2^m$, and the formula $\mathbf{A}\mathbf{X}_1^{m-1} = \mathbf{X}_2^m$ equivalent to (1.16). However, this formulation can be thought of more broadly and does not necessarily need to relate directly to (1.1).

1.4 ▪ Example code and decomposition

To demonstrate the DMD algorithm, we consider a simple example of two mixed spatiotemporal signals. In particular, our objective is to demonstrate the ability of DMD to efficiently decompose the signal into its constituent parts. To build intuition, we also compare DMD against results from principal component analysis (PCA) and independent component analysis (ICA).

The two signals of interest are

$$\begin{aligned} f(x,t) &= f_1(x,t) + f_2(x,t) \\ &= \operatorname{sech}(x+3)\exp(i2.3t) + 2\operatorname{sech}(x)\tanh(x)\exp(i2.8t). \end{aligned} \tag{1.33}$$

The individual spatiotemporal signals $f_1(x,t)$ and $f_2(x,t)$ are illustrated in Figure 1.2(a)–(b). The two frequencies present are $\omega_1 = 2.3$ and $\omega_2 = 2.8$, which have distinct spatial structures. The mixed signal $f(x,t) = f_1(x,t) + f_2(x,t)$ is illustrated in Figure 1.2(c). The following code in MATLAB constructs the spatiotemporal signals.

***Algorithm* 1.2. Mixing of two spatiotemporal signals.**

```
%% Define time and space discretizations
xi = linspace(-10,10,400);
t = linspace(0,4*pi,200);
dt = t(2) - t(1);
[Xgrid,T] = meshgrid(xi,t);

%% Create two spatiotemporal patterns
f1 = sech(Xgrid+3) .* (1*exp(1j*2.3*T));
f2 = (sech(Xgrid).*tanh(Xgrid)).*(2*exp(1j*2.8*T));

%% Combine signals and make data matrix
```

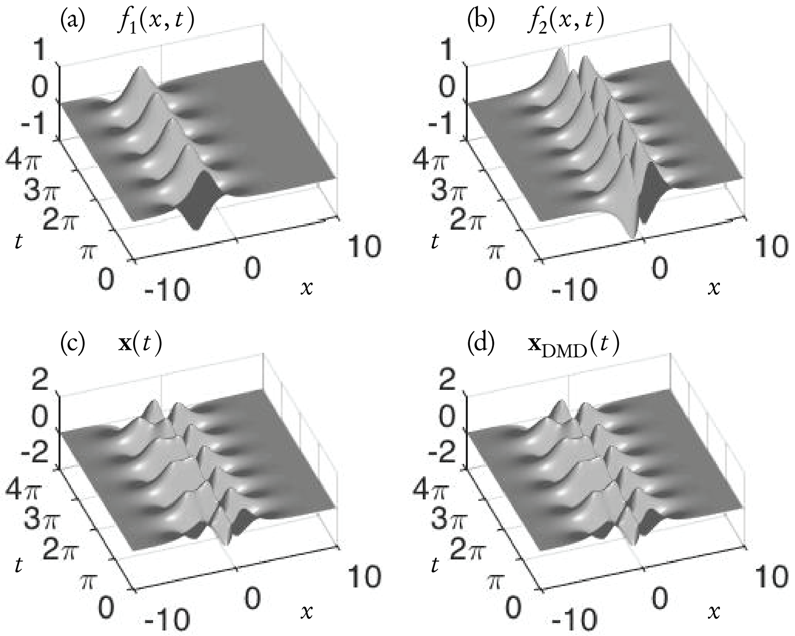

Figure 1.2. *An example of spatiotemporal dynamics of two signals* (a) $f_1(x,t)$ *and* (b) $f_2(x,t)$ *of* (1.33) *that are mixed in* (c) $f(x,t) = f_1(x,t) + f_2(x,t)$*. The function* $f(x,t)$ *can be represented instead by* $\mathbf{x}(t)$*. The DMD of* $\mathbf{x}$ *was computed, and a rank-2 approximate reconstruction* (1.24) *of the signal* $\mathbf{x}_{DMD}(t)$ *is shown in panel* (d)*. The reconstruction is almost perfect, with the DMD modes and spectra closely matching those of the underlying signals* $f_1(x,t)$ *and* $f_2(x,t)$*.*

```
f = f1 + f2;
X = f.'; % Data Matrix

%% Visualize f1, f2, and f
figure;
subplot(2,2,1);
surfl(real(f1));
shading interp; colormap(gray); view(-20,60);
set(gca, 'YTick', numel(t)/4 * (0:4)),
set(gca, 'Yticklabel',{'0','\pi','2\pi','3\pi','4\pi'});
set(gca, 'XTick', linspace(1,numel(xi),3)),
set(gca, 'Xticklabel',{'-10', '0', '10'});

subplot(2,2,2);
surfl(real(f2));
shading interp; colormap(gray); view(-20,60);
set(gca, 'YTick', numel(t)/4 * (0:4)),
set(gca, 'Yticklabel',{'0','\pi','2\pi','3\pi','4\pi'});
set(gca, 'XTick', linspace(1,numel(xi),3)),
set(gca, 'Xticklabel',{'-10', '0', '10'});

subplot(2,2,3);
surfl(real(f));
shading interp; colormap(gray); view(-20,60);
```

```
set(gca, 'YTick', numel(t)/4 * (0:4)),
set(gca, 'Yticklabel',{'0','\pi','2\pi','3\pi','4\pi'});
set(gca, 'XTick', linspace(1,numel(xi),3)),
set(gca, 'Xticklabel',{'-10', '0', '10'});
```

This code produces the data matrix **X** of (1.10). **X** is the starting point for the decomposition algorithm of DMD, PCA, and ICA. The following code implements DMD following the steps outlined in the last subsection. In this particular case, we also perform a rank-2 decomposition to illustrate the low-dimensional nature of the decomposition.

ALGORITHM 1.3. Perform DMD on data.

```
%% Create DMD data matrices
X1 = X(:, 1:end-1);
X2 = X(:, 2:end);

%% SVD and rank-2 truncation
r = 2; % rank truncation
[U, S, V] = svd(X1, 'econ');
Ur = U(:, 1:r);
Sr = S(1:r, 1:r);
Vr = V(:, 1:r);

%% Build Atilde and DMD Modes
Atilde = Ur'*X2*Vr/Sr;
[W, D] = eig(Atilde);
Phi = X2*Vr/Sr*W;  % DMD Modes

%% DMD Spectra
lambda = diag(D);
omega = log(lambda)/dt;

%% Compute DMD Solution
x1 = X(:, 1);
b = Phi\x1;
time_dynamics = zeros(r,length(t));
for iter = 1:length(t),
    time_dynamics(:,iter) = (b.*exp(omega*t(iter)));
end;
X_dmd = Phi*time_dynamics;

subplot(2,2,4);
surfl(real(X_dmd'));
shading interp; colormap(gray); view(-20,60);
set(gca, 'YTick', numel(t)/4 * (0:4)),
set(gca, 'Yticklabel',{'0','\pi','2\pi','3\pi','4\pi'});
set(gca, 'XTick', linspace(1,numel(xi),3)),
set(gca, 'Xticklabel',{'-10', '0', '10'});
```

This code computes several important diagnostic features of the data, including the singular values of the data matrix **X**, the DMD modes $\mathbf{\Phi}$, and the continuous-time DMD eigenvalues ω. The eigenvalues ω match the underlying frequencies exactly, where their imaginary components correspond to the frequencies of oscillation:

```
omega =
   0.0000 + 2.8000i
   0.0000 + 2.3000i
```

Following (1.24), a rank-2 DMD reconstruction $\mathbf{X}_{\text{DMD}}$ is possible, separating the data $\mathbf{X}$ into a sum of coupled spatiotemporal modes. This reconstruction is shown in Figure 1.2(d). Figure 1.3(a) shows the decay of singular values of $\mathbf{X}$, which reveals that the data may be appropriately represented as rank $r = 2$. Figure 1.3(a) compares the temporal and spatial modes as extracted by DMD with the true modes; these DMD modes almost exactly match the true solution modes $f_1(x,t)$ and $f_2(x,t)$.

To built intuition for DMD, we compare it against two commonly used modal decomposition techniques, PCA and ICA. The following code computes the PCA modes and their temporal evolution.

ALGORITHM **1.4. PCA computed by SVD.**

```
[U, S, V] = svd(X);
pc1 = U(:, 1); % first PCA mode
pc2 = U(:, 2); % second PCA mode
time_pc1 = V(:, 1); % temporal evolution of pc1
time_pc2 = V(:, 2); % temporal evolution of pc2
```

In a similar fashion, we can also perform an ICA decomposition. There exist several implementations of ICA; here we use `fastica` from a MATLAB package [105].

ALGORITHM **1.5. Fast ICA.**

```
[IC, ICt, ~] = fastica(real(X)');
ic1 = IC(1, :); % first ICA mode
ic2 = IC(2, :); % second ICA mode
time_ic1 = ICt(:, 1); % temporal evolution of ic1
time_ic2 = ICt(:, 2); % temporal evolution of ic2
```

PCA and ICA extract modal structures contained in the data matrix $\mathbf{X}$; Figure 1.3 shows these modes as compared to DMD modes and the true solution. PCA has no mechanism to separate the independent signals $f_1(x,t)$ and $f_2(x,t)$. It does, however, produce the correct rank-2 structure. The PCA modes produced in a rank-2 decomposition are chosen to maximize the variance in the data. They mix the two spatiotemporal modes, as shown in Figure 1.3(e) and (f). The time dynamics of the PCA modes are also shown in Figure 1.3(c) and (d). The ICA results are much better than the PCA results since ICA attempts to account for the independent nature of the two signals [166]. The ICA modes and time dynamics are also shown in Figure 1.3(c)–(f).

To make quantitative comparisons among DMD, PCA, and ICA, Figure 1.3(b) shows the ℓ_2-norm of the difference between the two true spatial modes and modes extracted by the three algorithms. The PCA modes have the largest error, while the DMD modes are accurate to nearly machine precision. The ICA error is less than half the error associated with PCA. This comparison shows the relative merits of DMD and its efficiency in decomposing spatiotemporal data.

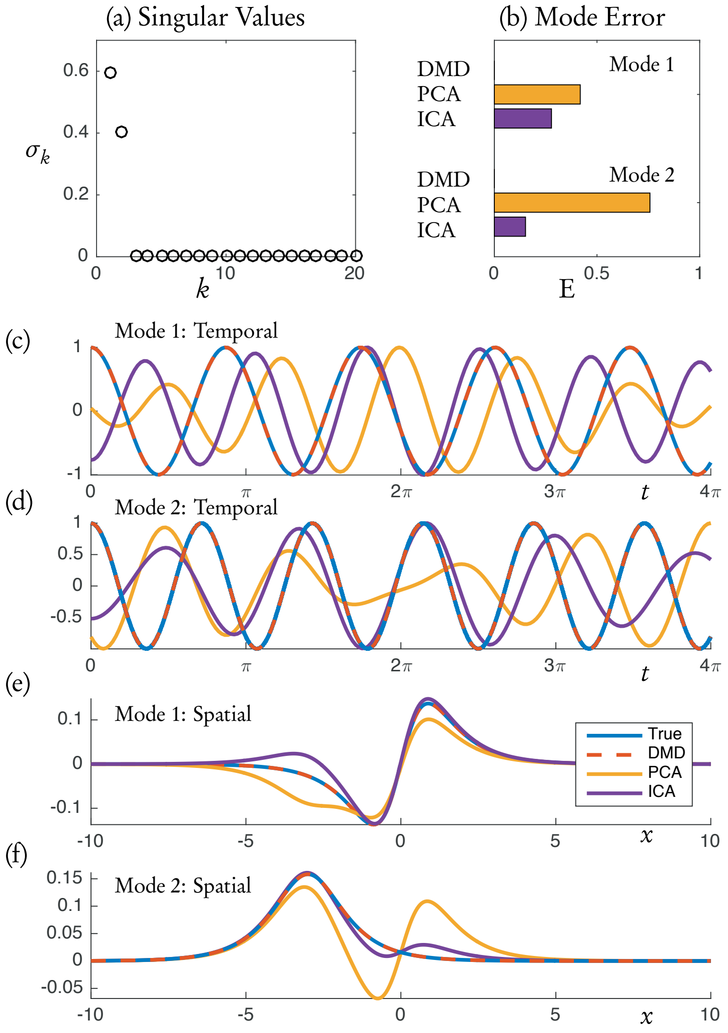

Figure 1.3. *A comparison of DMD with PCA and ICA as applied to example data generated with* (1.33)*. The singular values in* (a) *show that a rank-2 truncation is appropriate. Panels* (c)–(f) *compare the temporal and spatial aspects of the two modes, where the true modes are plotted along with modes extracted by DMD, PCA, and ICA from the data. DMD produces results that are exactly (to numerical precision) aligned with the true solution. ICA modes approximate the true modes better than PCA does, but both deviate from the true solution. The ℓ_2-norm difference between the two true spatial modes and modes extracted by DMD, PCA, and ICA are shown in* (b)*. DMD modes match the true modes exactly and have zero error.*

1.5 ▪ Limitations of the DMD method

The DMD method, much like PCA, is based on an underlying SVD that extracts correlated patterns in the data. It is well known that a fundamental weakness of such

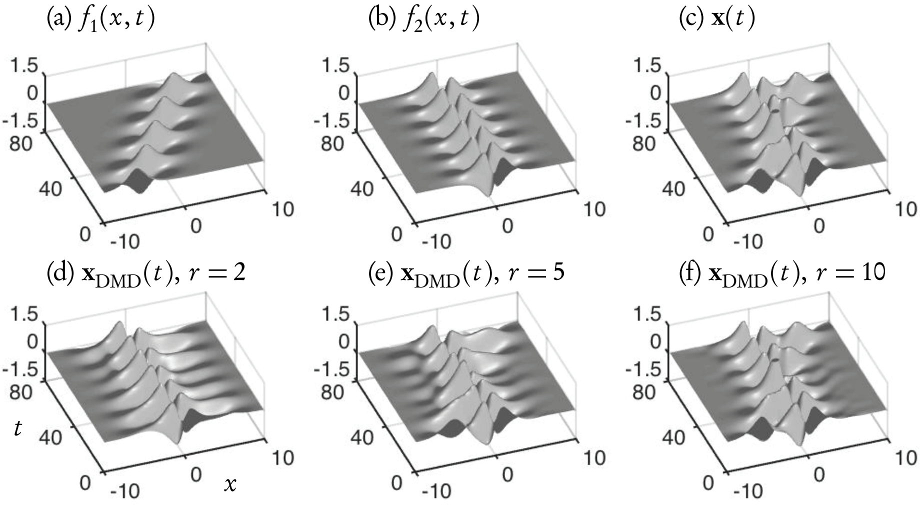

Figure 1.4. *An example of spatiotemporal dynamics with translation* (1.34). *Panels* (a)–(c) *illustrate the real parts of the two signals and their mixture. Panels* (d)–(f) *show reconstructions of the signal* $\mathbf{x}_{DMD}(t)$ *using rank-*2*, rank-*5*, and rank-*10 *approximations. Although the dynamics are constructed from a two-mode interaction, the reconstruction now requires approximately* 10 *modes to get the right dynamics.*

SVD-based approaches is the inability to efficiently handle invariances in the data. Specifically, translational and/or rotational invariances of low-rank objects embedded in the data are not well captured. Moreover, transient time phenomena are also not well characterized by such methods. This will be illustrated in various examples that follow.

1.5.1 ▪ Translational and rotational invariances

To demonstrate the DMD algorithm and some of its limitations, we once again consider the simple example of two mixed spatiotemporal signals. In this case, however, one of the signals is translating at a constant velocity across the spatial domain. The two signals of interest are

$$\begin{aligned} f(x,t) &= f_1(x,t)+f_2(x,t) \\ &= \text{sech}(x+6-t)\exp(i2.3t)+2\text{sech}(x)\tanh(x)\exp(i2.8t). \end{aligned} \tag{1.34}$$

As before, the two frequencies present are $\omega_1 = 2.3$ and $\omega_2 = 2.8$, with given corresponding spatial structures. The individual spatiotemporal signals $f_1(x,t)$ and $f_2(x,t)$, along with the mixed solution $f(x,t) = f_1(x,t)+f_2(x,t)$, are illustrated in Figure 1.4(a)–(c), respectively. In this case, the MATLAB code is reevaluated using rank-2, rank-5, and rank-10 reconstruction. The rank-2 reconstruction is no longer able to characterize the dynamics due to the translation. Indeed, it now takes nearly 10 modes to produce an accurate reconstruction. This artificial inflation of the dimension is a result of the inability of SVD to capture translational invariances and correlate across snapshots of time.

To visualize this artificial inflation in dimension, Figure 1.5(a) shows the singular values of $\mathbf{X}$, which are used in both DMD and PCA. Although there are still only two

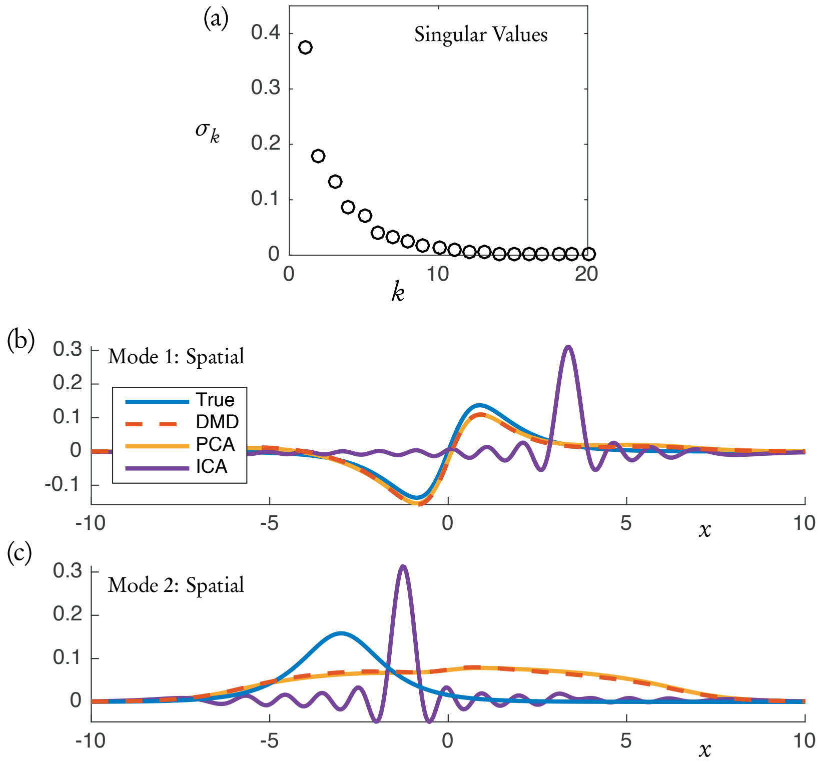

Figure 1.5. *Comparison of DMD with PCA and ICA for spatiotemporal dynamics with translation. The singular value decay in* (a) *shows that a rank-2 truncation is no longer appropriate due to the translation of one of the modes. The* (b) *first and* (c) *second modes (normalized by the maximum value) generated from all three methods are compared to the true solution. None are aligned with the true solution.*

true modes, the SVD suggests that at least 10 modes are required to converge to an accurate solution. Figure 1.5(c) and (d) illustrate the first two modes generated from DMD, PCA, and ICA. Note the significantly different modal structures generated by each algorithm in this case. Ultimately, none of the methods handle the translation in any viable way. Any continuous symmetry in a data set will produce similarly poor low-rank characterizations. The discussion of multiresolution DMD (mrDMD) in Chapter 5 includes one way to handle this issue in practice.

1.5.2 ▪ Transient time behavior

Transient phenomena are also difficult for DMD, PCA, and/or ICA methods to handle. To demonstrate the impact of the transient time dynamics, we once again consider the simple example of two mixed spatiotemporal signals, with one turning on and off during the time domain. In this case, the signals are

$$\begin{aligned} f(x,t) &= f_1(x,t)+f_2(x,t)\\ &= 0.5\,\text{sech}(x+5)\exp(i2.3t)[\tanh(t-\pi)-\tanh(t-3\pi)]\\ &\quad +2\text{sech}(x)\tanh(x)\exp(i2.8t). \end{aligned} \tag{1.35}$$

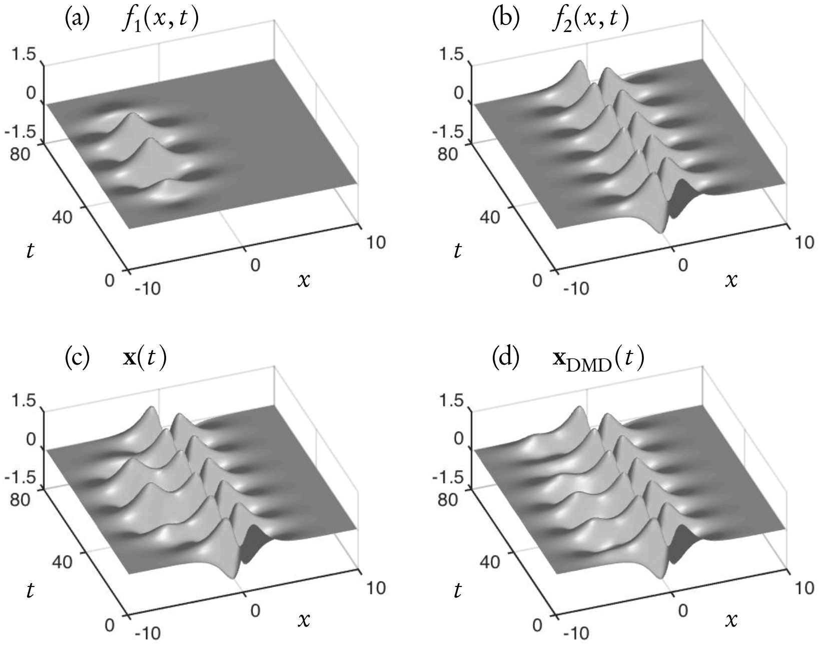

Figure 1.6. *Example spatiotemporal dynamics (real part) of two signals of* (1.35) *that have transient dynamics. Panel* (d) *illustrates the reconstruction of the signal* $\mathbf{x}_{DMD}(t)$ *from a rank-*10 *approximation* (1.24)*. The reconstruction fails to capture the on-off nature of the mode.*

As before, the two frequencies present are $\omega_1 = 2.3$ and $\omega_2 = 2.8$, with given corresponding spatial structures. The individual spatiotemporal signals $f_1(x,t)$ and $f_2(x,t)$, along with the mixed solution $f(x,t)$, are illustrated in Figure 1.6(a)–(c), respectively. We can once again try to reconstruct the solution using DMD. Interestingly, regardless of the rank ($r = 10$ in the figure), DMD is incapable of correctly getting the right turn-on and turn-off behavior of the first mode. The result shown is the same for any rank approximation above two. Unlike the translational or rotational invariance, which simply created an artificially high dimension in DMD, in this case DMD completely fails to characterize the correct dynamics. mrDMD in Chapter 5 presents a potential strategy to handle this issue.

1.6 ▪ Broader context of equation-free methods

One may consider DMD as an equation-free way to approximate a nonlinear dynamical system. Often, (1.1) arises as a discretization of a partial differential equation

$$\mathbf{q}_t = \mathbf{N}(\mathbf{q}, \mathbf{q}_\xi, \mathbf{q}_{\xi\xi}, \ldots, \xi, t; \mu), \tag{1.36}$$

where $\mathbf{q}(\xi,t)$ represents a vector of the variables of interest, the subscripts denote partial differentiation with respect to either time or space, and $\mathbf{N}(\cdot)$ determines the nonlinear governing evolution equations. The evolution equations depend generically upon time, t, and the spatial variable, ξ, and derivatives of the quantity of interest $\mathbf{q}$. For instance, the vector $\mathbf{q}(\xi,t)$ might represent the velocity, temperature, and pressure fields in a complex, three-dimensional fluid flow system. Thus, there would be a set of five coupled partial differential equations governing the overall dynamics of the system. The vector μ determines the values of one or more parameters in the complex system; as these parameters vary, the system may exhibit drastically different behaviors, characterized by bifurcations. In the fluid example above, the bifurcation parameter μ would represent critical quantities like the Reynolds number.

Equation (1.36) provides a very general model framework for studying complex systems. In traditional dynamical systems modeling and scientific computing, a critical assumption is made about (1.36). Namely, we know the governing equations, and thus the nonlinear function $\mathbf{N}(\cdot)$. However, it is almost always the case that the right-hand side is approximated using asymptotic arguments or neglecting what are thought to be unimportant physical terms. For instance, we typically neglect the effects of air friction when modeling a simple pendulum in our differential equations courses. For sufficiently complex systems, constructing an accurate set of governing equations and function $\mathbf{N}(\cdot)$ is extremely challenging and confounded by multiscale physics effects. Indeed, understanding the interaction of microscale and macroscale phenomena in many engineering, physical, and biological systems is at the forefront of current research methods. DMD reverses this process and instead uses the abundance of data collected to reverse-engineer the governing equations directly from data.

The rise of equation-free modeling techniques challenges the foundational notion that $\mathbf{N}(\cdot)$ is known. In particular, equation-free strategies seek to relax the assumption concerning knowledge of $\mathbf{N}(\cdot)$ by exploiting sampling and data collection in the complex system. Such data-driven strategies are capable of transformative impact, especially in systems where traditional modeling methods fail due to lack of information and insight into the evolution equation (1.36). As a specific example, atmospheric sciences and weather forecasting have seen tremendous performance gains in prediction by using data-assimilation techniques, e.g., ensemble Kalman filtering. Interestingly, the field of weather forecasting has a long history of model development. Thus the continued development of traditional first-principles modeling techniques, especially those that model the myriad of multiscale physics effects, are aimed at improving our construction of $\mathbf{N}(\cdot)$ in (1.36). In emerging fields such as computational neuroscience, it is extremely difficult at this time to construct first-principles equations (1.36). Such systems are known to be both dynamical and highly networked, with only limited knowledge of the network architecture currently available. Thus, for such a system, constructing an accurate $\mathbf{N}(\cdot)$ is a challenging proposition. Both of these examples highlight the crucial need to integrate data-driven strategies in modeling efforts. Moreover, given their sensitivity to initial conditions, even having an accurate $\mathbf{N}(\mathbf{q})$ poses significant problems in accurate future-state prediction, thus again highlighting the need to integrate data measurements within the modeling architecture.

Data-driven techniques for dealing with complex systems (1.36) are of growing interest in the mathematical sciences community. Moreover, they are having significant impact across the engineering, physical, and biological sciences. DMD is only one of a handful of different techniques available for data-driven modeling. As such, we highlight a number of example techniques that are at the forefront of mathematical developments for modeling and computing high-dimensional complex systems (1.36). The methods highlighted are illustrative of techniques and philosophies, and this is by no means an exhaustive list.

1.6.1 ▪ Equation-free, multiscale projections

Kevrekidis and co-workers [107, 161, 260, 160] have developed an equation-free architecture with the moniker *equation-free* methods (EFMs) that capitalizes on inherent multiscale phenomena. The central theme of the method is the fact that the macroscale evolution dynamics and/or variables are unknown. Thus, evolving (1.36) to future states is not viable given the ambiguity in the macroscale dynamics. However, the method also assumes that the microscale physics driving the larger macroscopic evolution, which is often stochastic in nature, is in fact known. The fundamental question arises as to how to connect the microscale dynamics to the larger macroscale. Thus, the term *equation-free* refers to the unknown macroscale evolution dynamics.

The EFM relies on a multistep procedure of *lifting* and *restriction*, which are methods for moving from microscale to macroscale variables and vice versa. Mathematically, the problem formulation results in an attempt to computationally construct the macroscopic, or coarse-grained, dynamical evolution (1.36), where the governing equations determined by $\mathbf{N}(\cdot)$ in (1.36) are unknown. Importantly, one needs only a computational approximation to $\mathbf{N}(\cdot)$ to compute a future state. In particular, an Euler time-stepping algorithm simply makes use of a slope formula and the definition of derivative to compute the future state t_{k+1} from state t_k:

$$\mathbf{q}(\xi, t_{k+1}) = \mathbf{q}(\xi, t_k) + \Delta t \tilde{\mathbf{N}}(\mathbf{q}(\xi, t_k), \xi, t_k), \tag{1.37}$$

where $\Delta t = t_{k+1} - t_k$ and $\tilde{\mathbf{N}}$ is a macroscale, typically low-dimensional, approximation of the full nonlinearity $\mathbf{N}(\cdot)$. As such, the Euler method essentially provides a template, or protocol, for a future-state estimation.

The algorithm begins with microscale simulation or data collection of the system of interest. The restriction step looks for methods that can perform a dimensionality reduction of the data through, for example, an SVD. It is assumed in the EFM that the unknown governing equations (1.36) have an underlying slow manifold that all the dynamics collapse onto. The goal of the dimensionality reduction step is to represent the dynamics on this low-rank attractor in the natural variables of the system (e.g., POD modes). Once the microscale dynamics have collapsed to the slow manifold, then the reduced variables are used to take a time step represented by (1.37) with a computed value of $\tilde{\mathbf{N}}$ that is evaluated on the low-rank attractor. This time step is in the macroscopic variables and is much larger than any time step allowed in the microscale dynamics. After taking a large macroscale time step, the lifting procedure is used to reinitialize a microscale simulation based on the state of the time-advanced macroscopic evolution. This algorithm advances the solution by alternating between the microscopic and macroscopic variables through restriction and lifting steps.

1.6.2 ▪ Reduced-order models

Reduced-order models (ROMs) are playing an increasingly important role in simulation strategies and future-state prediction for high-dimensional, complex systems [228]. Although not inherently equation free, ROMs aim to capitalize on low-dimensional structures in the dynamics, much as the DMD and EFM strategies do. Specifically, ROMs look to construct representations of the solution $\mathbf{q}(\xi, t)$ in some optimal basis (e.g., POD modes). Similarly to both DMD and EFM, snapshots of the evolving system are acquired and the low-rank subspace on which $\mathbf{q}(\xi, t)$ evolves is extracted. Often this strategy of data acquisition is done in an *off-line* manner given how computationally expensive it can be to sample the high-dimensional simulation space. The low-rank subspaces, once computed, can then be used in an *online* fashion to compute future states.

The ROM architecture can easily be combined with the EFM [260] and with DMD [300]. Indeed, the methods partner naturally since they both aim to exploit low-rank dynamics. ROMs and EFMs can also take advantage of limited measurements [108] to advance the solution in time. Interestingly, the state-of-the-art ROM techniques make a point to capitalize on limited measurements. The gappy POD [92], for example, constructs accurate Galerkin-POD approximations of complex systems [133] from limited, incomplete, or corrupted measurements. First introduced by Everson and Sirovich [92], the gappy method did not advocate a principled sensor placement algorithm until the works of Willcox [298], Karniadakis [309], Sorenson [63], and their co-workers. More recently, the low-rank POD approximations were combined with compressed sensing to perform sensor tasks and ROM computations [36, 49, 242]. Sensor placement based on alternative criteria, such as providing accurate classification or clustering of data [280, 304, 68, 43, 14], may also be important in modeling efforts. We will also consider the role of limited measurements in the DMD method in Chapter 9, as it is clear that such techniques can be critically enabling in many applications.

1.6.3 ▪ Stochastic-statistical dynamical systems methods

DMD and EFMs attempt to provide approximations to (1.36) by either a best-fit regression to a linear dynamical system through the sampled data or a time-scale separation algorithm that alternates between lifting and restriction to numerically construct $\mathbf{N}(\cdot)$, respectively. Another emerging set of techniques that acknowledge from the start the lack of an accurate governing set of equations revolve around the concept of stochastic-statistical dynamical systems [188, 35]. The idea of such methods is to use multimodel ensemble forecasts to improve the statistical accuracy of imperfect predictions through combining information from a collection of ROMs.

This information-theoretic framework capitalizes on a collection of imperfect, potentially low-fidelity models, such as ROM-type models, to improve future-state predictions in a statistical way. Innovations in this area center around understanding uncertainty quantification metrics and their role in weighting the various ROM predictions. Interestingly, the philosophical leanings of this approach are closely related to the developments of the machine learning method of boosting. In particular, Kearns and Valiant [155, 156] formulated the question: Can a set of weak learners create a single strong learner? A weak learner is defined to be a classifier that is only slightly correlated with the true classification (it can label examples better than random guessing). In contrast, a strong learner is a classifier that is arbitrarily well correlated with the true classification. In 1990, Schapire [246] showed that this was indeed the case. Not only were there significant ramifications in machine learning and statistics, but also this seems to have important implications for complex dynamical systems.

From the viewpoint of complex systems, it is appropriate to restate an insightful quote from a review article of A. Majda concerning climate modeling [188]:

> *The central difficulty in climate change science is that the dynamical equations for the actual climate are unknown. All that is available from the true climate in nature are some coarse-grained observations of functions such as mean or variance of temperature, tracer greenhouse gases such as carbon dioxide, or the large-scale horizontal winds. Thus, climate change science must*

> *cope with predicting the coarse-grained dynamic changes of an extremely complex system only partially observed from a suite of imperfect models for the climate.*

1.7 ▪ Interdisciplinary connections of DMD

The underlying ideas of the DMD algorithm have a strong intuitive appeal. Indeed, the Koopman construction concept dates back to the 1930s and provides a simple framework for understanding the flow of measurements on the state of a system into the future. Given the long history of the idea, it is not surprising that many fields have developed similar ideas for specific applications of interest. In what follows, we illustrate a few example applications that are closely related to DMD.

1.7.1 ▪ ARIMA: Autoregressive integrated moving average

In statistics and econometrics, and in particular in time-series analysis, autoregressive moving average (ARMA) models and the generalized autoregressive integrated moving average (ARIMA) models [32] are commonly used. These models are fitted to time-series data either to better understand the data or to predict future points in the series (forecasting). ARIMA models are often applied to data that shows evidence of non-stationarity, where an initial differencing step (corresponding to the "integrated" part of the model) can be applied to reduce the nonstationarity. Such models are characterized by a number of key parameters, one of them being the number of past time points used to forecast a future point. This is similar to what DMD accomplishes. However, DMD links each time snapshot directly to the previous time snapshot.

A number of variations on the ARIMA model are commonly employed and allow us to connect the method more closely to DMD. If multiple time series are used, which would be the case for DMD, then ARIMA can be generalized to a vector framework, i.e., the VARIMA (vector ARIMA) model. Sometimes a seasonal effect is exemplified in the time series, in which case oscillatory behavior should be modeled. For such cases, it is generally common to use the SARIMA (seasonal ARIMA) model instead of increasing the order of the AR or MA part of the model. In the DMD architecture, seasonal variations are automatically included. Moreover, if the mean of the data matrix $\mathbf{X}$ is subtracted, then DMD has been shown to be equivalent to a Fourier decomposition of the vector field in time [64]. DMD can be thought of as taking advantage of both the vector nature of the data and any oscillatory (seasonal) variations. Further, the real part of the DMD spectrum allows one to automatically capture exponential trends in the data.

1.7.2 ▪ LIM: Linear inverse modeling

Linear inverse modeling (LIM) was developed in the climate science community. LIM is essentially identical to the DMD architecture under certain assumptions [290]. By construction, LIM relies on low-rank modeling, like DMD, using the empirical orthogonal functions (EOFs) first introduced in 1956 by Lorenz [181]. Used in the climate science literature, EOFs are the result of applying PCA to meteorological data [215]; thus, EOFs are essentially POD modes. Nearly two decades before the first DMD papers appeared, Penland [215] derived a method to compute a linear dynamical system that approximates data from a stochastic linear Markov system; these systems will be revisited in Chapter 7. This method later came to be known as LIM [216].

Under certain circumstances, DMD and LIM may be considered as equivalent algorithms. LIM is also performed in the low-dimensional space of EOF/POD coefficients computed on the first $m-1$ snapshots in $\mathbf{X}_1^{m-1}$. Since LIM considers sequential snapshot data where $\mathbf{X}_1^{m-1}$ and $\mathbf{X}_2^m$ only differ by a single column, it often makes little difference which matrix is used to compute EOFs [216]. Given these assumptions, the equivalence of projected DMD and LIM gives us yet another way to interpret DMD analysis. If the data mean is removed, then the low-rank map that generates the DMD eigenvalues and eigenvectors is simply the one that yields the statistically most likely state in the future. This is the case for both exact and projected DMD, as both are built on the same low-order linear map. LIM is typically performed for data where the mean has been subtracted, whereas DMD is valid with or without mean subtraction. Regardless, there is a strong enough similarity between the two methods that the DMD community may benefit from further investigating the use of LIM in the climate science literature. Similarly, climate science may benefit from recent algorithmic developments and extensions to DMD.

1.7.3 ▪ PCR: Principal component regression

In statistics, principal component regression (PCR) is a regression analysis technique that is once again based on PCA. PCR regresses the outcome (also known as the response or the dependent variable) on the principal components of a set of covariates (also known as predictors or explanatory variables or independent variables) based on a standard linear regression model.

In PCR, which predates DMD by almost three decades [148], instead of regressing the dependent variable on the explanatory variables directly, the principal components of the explanatory variables are used as regressors. One typically uses only a subset of all the principal components for regression, thus making PCR a regularized procedure. Often the principal components with larger variances are selected as regressors; these principal components correspond to eigenvectors with larger eigenvalues of the sample variance-covariance matrix of the explanatory variables. However, for the purpose of predicting the outcome, principal components with low variances may also be important, in some cases even more important [148, 143].

Unlike the DMD/LIM literature, which has traditionally been steeped in dynamics, the statistics literature is often concerned with regression of static data. PCR is a linear regression that finds relationships of the output variables in terms of the PCA of the input variables. Typically PCR is not specifically applied to time-series data but is instead a general regression procedure that may or may not be applied to data from a dynamical system. In some sense, the first two steps of the DMD algorithm may be viewed as performing PCR on snapshot data from a high-dimensional dynamical system. However, PCR does not include the additional steps of eigendecomposition of the matrix $\tilde{\mathbf{A}}$ or the reconstruction of high-dimensional coherent modes. This last step is what relates DMD to the Koopman operator, connecting data analysis to nonlinear dynamics.

1.7.4 ▪ System identification methods

System identification methods from the control community are intimately connected to DMD. Previous work has established the connection between DMD and the eigensystem realization algorithm (ERA) [151, 290]. Subsequently, DMD with control (DMDc) was also linked to ERA and the observer Kalman filter identification methods (OKID) [222], as will be discussed in Chapters 6 and 7. ERA and OKID construct input-output models using impulse response data and continuous disturbance data, respectively [151, 152, 220, 218, 150]. Both of these methods were developed to construct input-output state-space models for aerospace applications where the systems tend to have higher rank than the number of sensor measurements [132, 151]. In contrast, DMD and DMDc were developed for systems with a large number of measurements and low-rank dynamics. Further, the DMD methods have been connected to the analysis of *nonlinear* complex systems [162, 196, 235, 195].

ERA and OKID have also been categorized as subspace identification methods [226]. A number of significant connections have been identified between DMDc and other prominent subspace identification methods [222]. These methods include the numerical algorithms for subspace system identification (N4SID), multivariable output error state space (MOESP), and canonical variate analysis (CVA) [292, 293, 154, 226]. Algorithmically, these methods involve regression, model reduction, and parameter estimation steps similar to DMDc. There are important contrasts regarding the projection scaling between all of these methods [222]. Similar research has combined system identification methods and balanced truncation to construct ROMs of nonlinear input-output systems [201, 123].

Recent advances in machine learning, compressed sensing, and data science have resulted in powerful nonlinear techniques in system identification. Genetic programming [165], also known as symbolic regression, has provided a method of determining nonlinear dynamical systems from measurement data [31, 252]. It is also possible to use an elastic net for fast symbolic regression [193] to identify nonlinear dynamics [227]. Alternatively, compressed sensing may be used to reconstruct dynamical systems from data to predict catastrophes [295]. The sparse identification of nonlinear dynamics (SINDy) algorithm [47] uses sparse regression [280] in a nonlinear function space to determine the relevant terms in the dynamics. This method has recently been connected to DMD and the Koopman operator [45]. This may be thought of as a generalization of earlier methods that employ symbolic regression (i.e., genetic programming) to identify dynamics. This follows a growing trend to exploit sparsity in dynamics [211, 245, 186] and dynamical systems [15, 221, 49]. Other methods analyze causality in dynamical systems using time-delay embedding [307, 269], which is related to the delay coordinates introduced in Chapter 7. The connection of DMD to these system identification methods is an exciting future direction of research for complex nonlinear systems.