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 multi-scale phenomena in both space and time. However complex, many of these system evolve on a low-dimensional attractor that may be characterized by spatio-temporal coherent structures. In this chapter, we will introduce the topic of this book, the 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 spatio-temporal coherent structures. Schmid and Sesterhenn [251] and Schmid [248] 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 spatio-temporal 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.* [197, 195, 196]

and Rowley *et al.* [236] showed that it is connected to the underlying nonlinear dynamics through Koopman operator theory [164] and is readily interpretable using standard dynamical systems techniques.

The development of dynamic mode decomposition 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. Throughout this chapter, we will introduce the core DMD algorithm, provide a broader historical context, and lay the mathematical foundation for future innovations and applications of DMD in the remaining chapters.

1.1 ▪ Dynamic mode decomposition

The dynamic mode decomposition 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 $k = 1,2,3,\cdots,m$. As we will show, the 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 $k = 1,2,3,\cdots,m-1$ snapshots. The advantage of this method is 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 [248, 236]. In particular, the algorithm extracts key low-rank spatio-temporal features of many high-dimensional systems, allowing for physically interpretable results in terms of spatial structures and their associated temporal re-

[1] The POD is often computed using the singular value decomposition (SVD). It is also known as the principal components analysis (PCA) in statistics [214, 149], empirical orthogonal functions (EOF) in climate and meteorology [183], the discrete Karhunen-Loeve transform, and the Hotelling transform [135, 136].

sponses. 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 spatio-temporal 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 the 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 the DMD is the fact that the underlying dynamics can exhibit multi-scale 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 the 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 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 of the above objectives. In some applications, it is only reasonable at this point in

time to use it 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 towards 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 $k = 1, 2, \cdots, m$ 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, it is 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), and 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 whose well-known solution [33] is

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

where $\boldsymbol{\phi}_k$ and ω_k are the eigenvectors and eigenvalues of the matrix $\mathbf{A}$, 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 $\mathbf{A}$:

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

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 eigen-decomposition (1.8) of the matrix $\mathbf{A}$ that optimally fits the measured trajectory $\mathbf{x}_k$ for $k = 1, 2, \cdots, m$ 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 $k = 1, 2, \cdots, m-1$. 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 to 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, both 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 $k = 1, 2, \cdots, m$, 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 $\mathbf{A}$ matrix is given by

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

where † is the Moore-Penrose pseudo-inverse. 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 $\mathbf{A}$. 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 $\mathbf{X}$ are high-dimensional so that the matrix is *tall-and-skinny*, meaning that the size of a snapshot n is larger than the number of snapshots $m - 1$. The matrix $\mathbf{A}$ may be high-dimensional; if $n = 10^6$, then $\mathbf{A}$ has 10^{12}

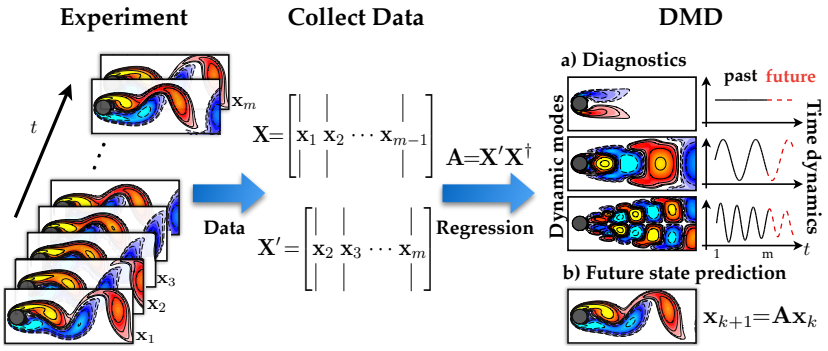

Figure 1.1. *Schematic overview of dynamic mode decomposition on a fluid flow example that will be explored further in Chapter 2. Note that the regression step does not typically construct* $\mathbf{A}$*, but instead constructs* $\tilde{\mathbf{A}}$ *which evolves the dynamics on a low-rank subspace,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

elements, so that it may be difficult to represent or decompose. However, the rank of $\mathbf{A}$ 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 $\mathbf{A}$ 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 non-zero eigenvalues and eigenvectors of the full-dimensional operator $\mathbf{A}$ without ever explicitly computing $\mathbf{A}$. This is discussed in § 1.3 below.

1.2.1 ▪ Defining DMD

The DMD method provides a spatio-temporal 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 Fig. 1.1. The mathematics underlying the extraction of dynamic information from time-resolved snapshots is closely related to the idea of the Arnoldi algorithm [248], one of the workhorses of fast computational solvers. The data collection process involves two parameters:

$$n = \text{number of spatial points saved per time snapshot}$$
$$m = \text{number of snapshots taken}$$

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 temporal [290] collection of data as well as irregularly spaced collection times. Indeed, Tu *et al.* [291] provides the most modern definition of the DMD method and algorithm.

Definition: Dynamic Mode Decomposition (Tu *et al.* 2014 [291]): *Suppose we have a dynamical system (1.1) and two sets of data*

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_{m-1} \\ | & | & & | \end{bmatrix} \tag{1.16a}$$

$$\mathbf{X}' = \begin{bmatrix} | & | & & | \\ \mathbf{x}'_1 & \mathbf{x}'_2 & \cdots & \mathbf{x}'_{m-1} \\ | & | & & | \end{bmatrix} \tag{1.16b}$$

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* $\mathbf{X}' \approx \mathbf{A}\mathbf{X}$*:*

$$\mathbf{A} = \mathbf{X}'\mathbf{X}^{\dagger}. \tag{1.17}$$

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 [236, 196]. 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 $\mathscr{K}$ is an infinite dimensional linear operator that acts on the Hilbert space $\mathscr{H}$ 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 $\mathbf{F}$:

$$\mathcal{K} g = g \circ \mathbf{F}, \tag{1.18a}$$

$$\implies \mathcal{K} g(\mathbf{x}_k) = g(\mathbf{F}(\mathbf{x}_k)) = g(\mathbf{x}_{k+1}). \tag{1.18b}$$

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 this 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 Code 1.3 as a function, proceeds as follows [291]:

1. First, take the singular value decomposition (SVD) of $\mathbf{X}$ [284]:

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*, \tag{1.19}$$

where $*$ denotes the conjugate transpose, $\mathbf{U} \in \mathbb{C}^{n \times r}$, $\mathbf{\Sigma} \in \mathbb{C}^{r \times r}$ 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 (1.19) could also be 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 $\mathbf{\Sigma}$ 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 [107], as discussed in § 8.2.

2. The matrix $\mathbf{A}$ from (1.17) may be obtained by using the pseudo-inverse of $\mathbf{X}$ obtained via the SVD:

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

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:

$$\tilde{\mathbf{A}} = \mathbf{U}^*\mathbf{A}\mathbf{U} = \mathbf{U}^*\mathbf{X}'\mathbf{V}\boldsymbol{\Sigma}^{-1}. \tag{1.21}$$

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

$$\tilde{\mathbf{x}}_{k+1} = \tilde{\mathbf{A}}\tilde{\mathbf{x}}_k. \tag{1.22}$$

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

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

$$\tilde{\mathbf{A}}\mathbf{W} = \mathbf{W}\boldsymbol{\Lambda}, \tag{1.23}$$

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 $\boldsymbol{\Phi}$:

$$\boldsymbol{\Phi} = \mathbf{X}'\mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{W}. \tag{1.24}$$

Note that (1.24) from [291] differs from the formula $\boldsymbol{\Phi} = \mathbf{U}\mathbf{W}$ in [248], although these will tend to converge if $\mathbf{X}$ and $\mathbf{X}'$ have the same column spaces. The modes in (1.24) are often called *exact DMD modes*, because it was proven in Tu *et al.* [291] 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*. To find a dynamic mode of $\mathbf{A}$ associated with a zero eigenvalue $\lambda_k = 0$, the exact formulation in (1.24) may be used if $\phi = \mathbf{X}'\mathbf{V}\boldsymbol{\Sigma}^{-1}w \neq 0$. Otherwise, the projected DMD formulation $\phi = Uw$ should be utilized [291].

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

$$\mathbf{x}(t) \approx \sum_{k=1}^{r} \boldsymbol{\phi}_k \exp(\omega_k t) b_k = \boldsymbol{\Phi}\exp(\boldsymbol{\Omega}t)\mathbf{b} \tag{1.25}$$

where b_k is the initial amplitude of each mode, $\mathbf{\Phi}$ is the matrix whose columns are the DMD eigenvectors $\boldsymbol{\phi}_k$, and $\mathbf{\Omega} = \text{diag}(\omega)$ 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 (1.25) 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 (1.25) gives $\mathbf{x}_1 = \mathbf{\Phi}\mathbf{b}$. The matrix of eigenvectors $\mathbf{\Phi}$ is generically not a square matrix so that the initial conditions

$$\mathbf{b} = \mathbf{\Phi}^\dagger \mathbf{x}_1 \tag{1.26}$$

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

A MATLAB implementation of the DMD algorithm (`DMD.m`) is provided in Code 1.3 below. The DMD algorithm presented here takes advantage of low dimensionality in the data in order 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 outside of plugging in the desired future time into (1.25). 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: multi-resolution (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 reconstrcted 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 DMD algorithm

Historically, the original DMD algorithm [248] was formulated in 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 [248]. To illustrate the algorithm, consider the regularly spaced sampling in time:

$$\text{data collection times}: \quad t_{k+1} = t_k + \Delta t \tag{1.27}$$

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

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}(t_1) & \mathbf{x}(t_2) & \cdots & \mathbf{x}(t_m) \\ | & | & & | \end{bmatrix} \tag{1.28}$$

where the vector $\mathbf{x}$ are 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:

$$\mathbf{X}_j^k = \begin{bmatrix} | & | & & | \\ \mathbf{x}(t_j) & \mathbf{x}(t_{j+1}) & \cdots & \mathbf{x}(t_k) \\ | & | & & | \end{bmatrix} \tag{1.29}$$

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

To construct the appropriate Koopman operator that best represents the data collected, the matrix $\mathbf{X}_1^{m+1}$ is considered:

$$\mathbf{X}_1^m = \begin{bmatrix} | & | & & | \\ \mathbf{x}(t_1) & \mathbf{x}(t_2) & \cdots & \mathbf{x}(t_m) \\ | & | & & | \end{bmatrix}. \tag{1.30}$$

Making use of (1.6), this matrix reduces to

$$\mathbf{X}_1^m = \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{A}\mathbf{x}_1 & \cdots & \mathbf{A}^{m-1}\mathbf{x}_1 \\ | & | & & | \end{bmatrix}. \tag{1.31}$$

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

$$\mathbf{x}_m = \sum_{k=1}^{m-1} b_k \mathbf{x}_k + \mathbf{r} \tag{1.32}$$

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 pseudo-inverse.

In this notation, we have the key result (compare to the definition (1.17)):

$$\mathbf{X}_2^m = \mathbf{A}\mathbf{X}_1^{m-1} \tag{1.33}$$

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 timestep, 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 $\mathbf{X} \to \mathbf{X}_1^{m-1}$, $\mathbf{X}' \to \mathbf{X}_2^m$ and the formula $\mathbf{A}\mathbf{X}_1^{m-1} = \mathbf{X}_2^m$ equivalent to (1.17). 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 spatio-temporal 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

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

The individual spatio-temporal signals $f_1(x,t)$ and $f_2(x,t)$ are illustrated in Fig. 1.2(a)–(b). The two frequencies present are $\omega_1 = 2.3$ and $\omega_2 = 2.8$ with distinct spatial structures. The mixed signal $f(x,t) = f_1(x,t) +$

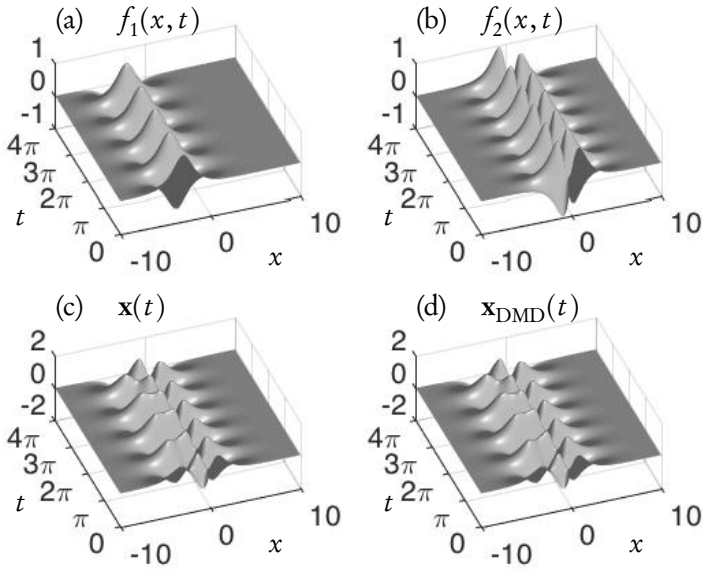

Figure 1.2. *An example of spatio-temporal dynamics of two signals (a) $f_1(x,t)$ and (b) $f_2(x,t)$ of (1.34) that are mixed in (c) $f(x,t) = f_1(x,t) + f_2(x,t)$. The function $f(x,t)$ can be represented instead by $\mathbf{x}(t)$. DMD of $\mathbf{x}$ was computed and a rank-2 approximate reconstruction (1.25) 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 the signals $f_1(x,t)$ and $f_2(x,t)$.*

$f_2(x,t)$ is illustrated in Fig. 1.2(c). The following code in MATLAB constructs the spatio-temporal signals.

ALGORITHM 1.2. **Mixing of two spatio-temporal 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 spatio-temporal 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
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 $\mathbf{X}$ of (1.10). $\mathbf{X}$ is the starting point for the decomposition algorithm of DMD, PCA and ICA. The following code implements DMD follows 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 $\mathbf{X}$, the DMD modes $\mathbf{\Phi}$, and the continuous-time DMD eigenvalues ω. These 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 (1.25), a rank-2 DMD reconstruction $\mathbf{X}_{\text{DMD}}$ is possible, separating the data $\mathbf{X}$ into a sum of coupled spatio-temporal modes. This reconstruction is shown in Fig. 1.2(d). Fig. 1.3(a) shows the decay of singular values of $\mathbf{X}$, which reveals the data may be appropriately

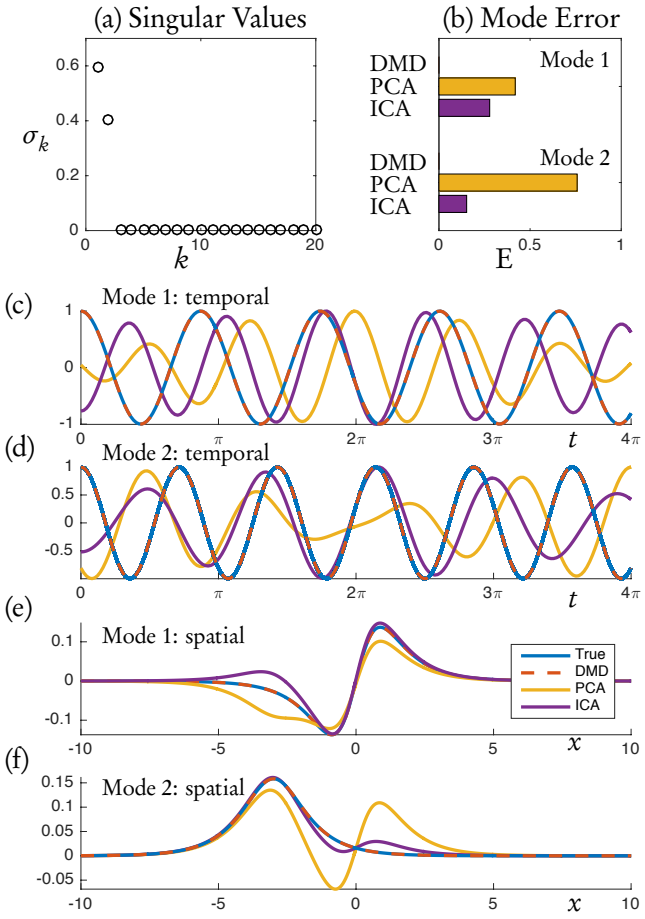

Figure 1.3. *A comparison of DMD with PCA and ICA as applied to example data generated with* (1.34)*. The singular values in (a) show that a rank-2 truncation is appropriate. Panels (c)–(f) compare the temporal and spatial aspects of the 2 modes, where the true modes are plotted along with modes extracted 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 2 true spatial modes and modes extracted by DMD, PCA and ICA are show in (b). DMD modes match the true modes exactly and have zero error..*

represented as rank $r = 2$. Fig. 1.3(a) compare 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 its against two commonly used modal decomposition techniques, PCA and ICA. The following code produces 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 exists several implementations of ICA; here we use `fastica` from a MATLAB package [106].

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
```

The PCA and ICA extract modal structures contained in the data matrix $\mathbf{X}$; Fig. 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 spatio-temporal modes as shown in Figs. 1.3(e) and (f). The time dynamics of the PCA modes are also shown in Fig. 1.3(c) and (d). The ICA results are much better than the PCA results since it attempts to account for the independent nature of the two signals [168]. The ICA modes and time dynamics are also shown in Figs. 1.3(c)–(f).

To make quantitative comparisons amongst DMD, PCA and ICA, Fig. 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 as-

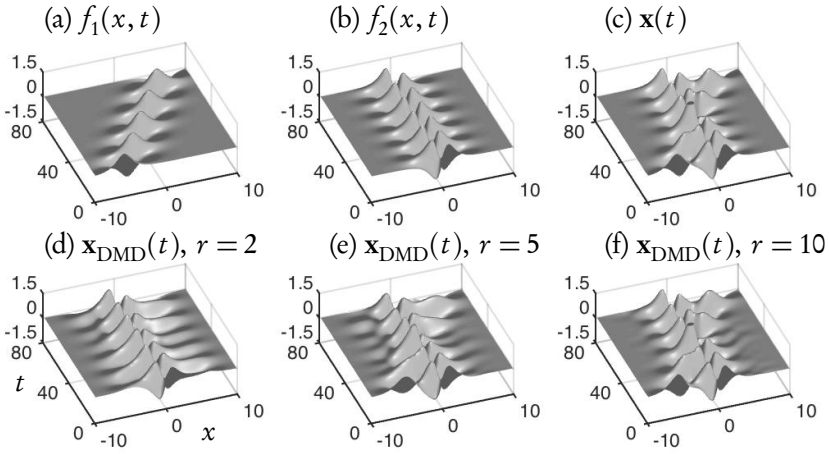

Figure 1.4. *An example of spatio-temporal dynamics with translation* (1.35) . *Panels (a)–(c) illustrate the real parts of the two signals and their mixture. Panels (d)–(f) show reconstructions of the signal* $\mathbf{x}_{DMD}(t)$ *using rank-2, rank-5 and rank-10 approximations. Although the dynamics are constructed from a two-mode interaction, the reconstruction now requires approximately 10 modes to get the right dynamics.*

sociated with PCA. This comparison shows the relative merits of DMD and its efficiency in decomposing spatio-temporal data.

1.5 ▪ Limitations of the DMD method

The DMD method, much like PCA, is based on an underlying singular value decomposition that extracts correlated patterns in the data. It is well known that a fundamental weakness of such 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 spatio-temporal signals. In this case, however, one of the signals is translating at a con-