# 7 Data-Driven Dynamical Systems

Dynamical systems provide a mathematical framework to describe the world around us, modeling the rich interactions between quantities that co-evolve in time. Formally, dynamical systems concerns the analysis, prediction, and understanding of the behavior of systems of differential equations or iterative mappings that describe the evolution of the state of a system. This formulation is general enough to encompass a staggering range of phenomena, including those observed in classical mechanical systems, electrical circuits, turbulent fluids, climate science, finance, ecology, social systems, neuroscience, epidemiology, and nearly every other system that evolves in time.

Modern dynamical systems began with the seminal work of Poincaré on the chaotic motion of planets. It is rooted in classical mechanics, and may be viewed as the culmination of hundreds of years of mathematical modeling, beginning with Newton and Leibniz. The full history of dynamical systems is too rich for these few pages, having captured the interest and attention of the greatest minds for centuries, and having been applied to countless fields and challenging problems. Dynamical systems provides one of the most complete and well-connected fields of mathematics, bridging diverse topics from linear algebra and differential equations, to topology, numerical analysis, and geometry. Dynamical systems has become central in the modeling and analysis of systems in nearly every field of the engineering, physical, and life sciences.

Modern dynamical systems is currently undergoing a renaissance, with analytical derivations and first principles models giving way to data-driven approaches. The confluence of big data and machine learning is driving a paradigm shift in the analysis and understanding of dynamical systems in science and engineering. Data are abundant, while physical laws or governing equations remain elusive, as is true for problems in climate science, finance, epidemiology, and neuroscience. Even in classical fields, such as optics and turbulence, where governing equations do exist, researchers are increasingly turning toward data-driven analysis. Many critical data-driven problems, such as predicting climate change, understanding cognition from neural recordings, predicting and suppressing the spread of disease, or controlling turbulence for energy efficient power production and transportation, are primed to take advantage of progress in the data-driven discovery of dynamics.

In addition, the classical geometric and statistical perspectives on dynamical systems are being complemented by a third *operator-theoretic* perspective, based on the evolution of measurements of the system. This so-called *Koopman* operator theory is poised to capitalize on the increasing availability of measurement data from complex systems. Moreover, Koopman theory provides a path to identify intrinsic coordinate systems to represent nonlinear dynamics in a linear framework. Obtaining linear representations of

strongly nonlinear systems has the potential to revolutionize our ability to predict and control these systems.

This chapter presents a modern perspective on dynamical systems in the context of current goals and open challenges. Data-driven dynamical systems is a rapidly evolving field, and therefore, we focus on a mix of established and emerging methods that are driving current developments. In particular, we will focus on the key challenges of discovering dynamics from data and finding data-driven representations that make nonlinear systems amenable to linear analysis.

## 7.1 Overview, Motivations, and Challenges

Before summarizing recent developments in data-driven dynamical systems, it is important to first provide a mathematical introduction to the notation and summarize key motivations and open challenges in dynamical systems.

### Dynamical Systems

Throughout this chapter, we will consider dynamical systems of the form:

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{f}(\mathbf{x}(t), t; \boldsymbol{\beta}), \tag{7.1}$$

where $\mathbf{x}$ is the state of the system and $\mathbf{f}$ is a vector field that possibly depends on the state $\mathbf{x}$, time $t$, and a set of parameters $\boldsymbol{\beta}$.

For example, consider the Lorenz equations [345]

$$\dot{x} = \sigma(y - x) \tag{7.2a}$$

$$\dot{y} = x(\rho - z) - y \tag{7.2b}$$

$$\dot{z} = xy - \beta z, \tag{7.2c}$$

with parameters $\sigma = 10$, $\rho = 28$, and $\beta = 8/3$. A trajectory of the Lorenz system is shown in Fig. 7.1. In this case, the state vector is $\mathbf{x} = \begin{bmatrix} x & y & z \end{bmatrix}^T$ and the parameter vector is $\boldsymbol{\beta} = \begin{bmatrix} \sigma & \rho & \beta \end{bmatrix}^T$.

The Lorenz system is among the simplest and most well-studied dynamical systems that exhibits chaos, which is characterized as a sensitive dependence on initial conditions. Two trajectories with nearby initial conditions will rapidly diverge in behavior, and after long times, only statistical statements can be made.

It is simple to simulate dynamical systems, such as the Lorenz system. First, the vector field $\mathbf{f}(\mathbf{x}, t; \boldsymbol{\beta})$ is defined in the function **lorenz**:

```
function dx = lorenz(t,x,Beta)
dx = [
Beta(1)*(x(2)-x(1));
x(1)*(Beta(2)-x(3))-x(2);
x(1)*x(2)-Beta(3)*x(3);
];
```

Next, we define the system parameters $\boldsymbol{\beta}$, initial condition $\mathbf{x}_0$, and time span:

```
Beta = [10; 28; 8/3]; % Lorenz's parameters (chaotic)

x0=[0; 1; 20];  % Initial condition
```

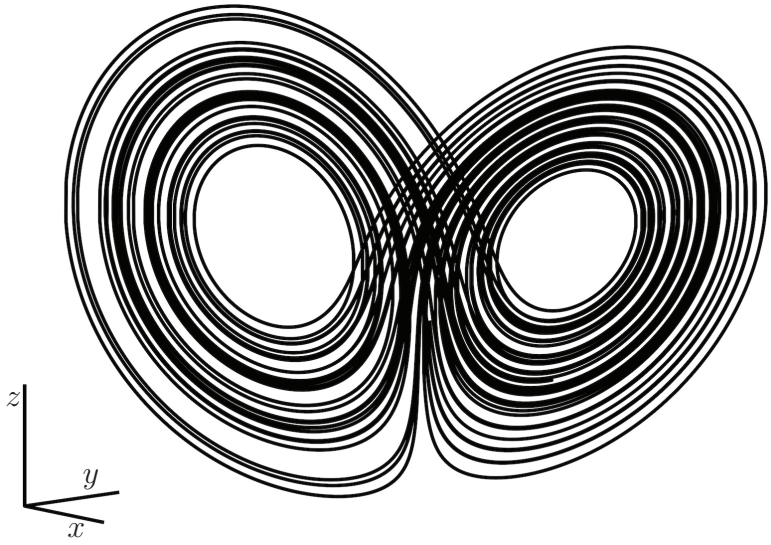


**Figure 7.1** Chaotic trajectory of the Lorenz system from (7.2).

```
dt = 0.001;
tspan=dt:dt:50;
options = odeset('RelTol',1e-12,'AbsTol',1e-12*ones(1,3));
```

Finally, we simulate the equations with **ode45**, which implements a fourth-order Runge Kutta integration scheme with adaptive time step:

```
[t,x]=ode45(@(t,x) lorenz(t,x,Beta),tspan,x0,options);
plot3(x(:,1),x(:,2),x(:,3));
```

We will often consider the simpler case of an autonomous system without time dependence or parameters:

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{f}(\mathbf{x}(t)). \tag{7.3}$$

In general, $\mathbf{x}(t) \in \mathbf{M}$ is an $n$-dimensional state that lives on a smooth manifold $\mathbf{M}$, and $\mathbf{f}$ is an element of the tangent bundle $\mathbf{TM}$ of $\mathbf{M}$ so that $\mathbf{f}(\mathbf{x}(t)) \in \mathbf{T}_{\mathbf{x}(t)}\mathbf{M}$. However, we will typically consider the simpler case where $\mathbf{x}$ is a vector, $\mathbf{M} = \mathbb{R}^n$, and $\mathbf{f}$ is a Lipschitz continuous function, guaranteeing existence and uniqueness of solutions to (7.3). For the more general formulation, see [1].

### *Discrete-Time Systems*

We will also consider the discrete-time dynamical system

$$\mathbf{x}_{k+1} = \mathbf{F}(\mathbf{x}_k). \tag{7.4}$$

Also known as a *map*, the discrete-time dynamics are more general than the continuous-time formulation in (7.3), encompassing discontinuous and hybrid systems as well.

For example, consider the logistic map:

$$x_{k+1} = \beta x_k(1 - x_k). \tag{7.5}$$

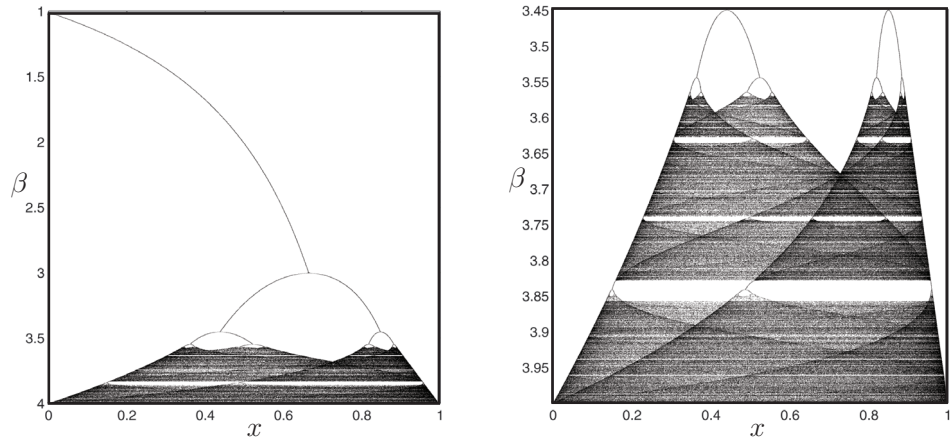


**Figure 7.2** Attracting sets of the logistic map for varying parameter $\beta$.

As the parameter $\beta$ is increased, the attracting set becomes increasingly complex, shown in Fig. 7.2. A series of period-doubling bifurcations occur until the attracting set becomes fractal.

Discrete-time dynamics may be induced from continuous-time dynamics, where $\mathbf{x}_k$ is obtained by sampling the trajectory in (7.3) discretely in time, so that $\mathbf{x}_k = \mathbf{x}(k\Delta t)$. The discrete-time propagator $\mathbf{F}_{\Delta t}$ is now parameterized by the time step $\Delta t$. For an arbitrary time $t$, the *flow map* $\mathbf{F}_t$ is defined as

$$\mathbf{F}_t(\mathbf{x}(t_0)) = \mathbf{x}(t_0) + \int_{t_0}^{t_0+t} \mathbf{f}(\mathbf{x}(\tau))\, d\tau. \tag{7.6}$$

The discrete-time perspective is often more natural when considering experimental data and digital control.

### *Linear Dynamics and Spectral Decomposition*

Whenever possible, it is desirable to work with linear dynamics of the form

$$\frac{d}{dt}\mathbf{x} = \mathbf{A}\mathbf{x}. \tag{7.7}$$

Linear dynamical systems admit closed-form solutions, and there are a wealth of techniques for the analysis, prediction, numerical simulation, estimation, and control of such systems. The solution of (7.7) is given by

$$\mathbf{x}(t_0 + t) = e^{\mathbf{A}t}\mathbf{x}(t_0). \tag{7.8}$$

The dynamics are entirely characterized by the eigenvalues and eigenvectors of the matrix $\mathbf{A}$, given by the *spectral decomposition* (eigen-decomposition) of $\mathbf{A}$:

$$\mathbf{A}\mathbf{T} = \mathbf{T}\boldsymbol{\Lambda}. \tag{7.9}$$

When $\mathbf{A}$ has $n$ distinct eigenvalues, then $\boldsymbol{\Lambda}$ is a diagonal matrix containing the eigenvalues $\lambda_j$ and $\mathbf{T}$ is a matrix whose columns are the linearly independent eigenvectors $\boldsymbol{\xi}_j$ associated

with eigenvalues $\lambda_j$. In this case, it is possible to write $\mathbf{A} = \mathbf{T}\mathbf{\Lambda}\mathbf{T}^{-1}$, and the solution in (7.8) becomes

$$\mathbf{x}(t_0 + t) = \mathbf{T}e^{\mathbf{\Lambda} t}\mathbf{T}^{-1}\mathbf{x}(t_0). \tag{7.10}$$

More generally, in the case of repeated eigenvalues, the matrix $\mathbf{\Lambda}$ will consist of Jordan blocks [427]. See Section 8.2 for a detailed derivation of the above arguments for control systems. Note that the continuous-time system gives rise to a discrete-time dynamical system, with $\mathbf{F}_t$ given by the solution map $\exp(\mathbf{A}t)$ in (7.8). In this case, the discrete-time eigenvalues are given by $e^{\lambda t}$.

The matrix $\mathbf{T}^{-1}$ defines a transformation, $\mathbf{z} = \mathbf{T}^{-1}\mathbf{x}$, into intrinsic eigenvector coordinates, $\mathbf{z}$, where the dynamics become decoupled:

$$\frac{d}{dt}\mathbf{z} = \mathbf{\Lambda}\mathbf{z}. \tag{7.11}$$

In other words, each coordinate, $z_j$, only depends on itself, with simple dynamics given by

$$\frac{d}{dt}z_j = \lambda_j z_j. \tag{7.12}$$

Thus, it is highly desirable to work with linear systems, since it is possible to easily transform the system into eigenvector coordinates where the dynamics become decoupled. No such closed-form solution or simple linear change of coordinates exist in general for nonlinear systems, motivating many of the directions described in this chapter.

### Goals and Challenges in Modern Dynamical Systems

As we generally use dynamical systems to model real-world phenomena, there are a number of high-priority goals associated with the analysis of dynamical systems:

1. **Future state prediction.** In many cases, such as meteorology and climatology, we seek predictions of the future state of a system. Long-time predictions may still be challenging.
2. **Design and optimization.** We may seek to tune the parameters of a system for improved performance or stability, for example through the placement of fins on a rocket.
3. **Estimation and control.** It is often possible to actively control a dynamical system through feedback, using measurements of the system to inform actuation to modify the behavior. In this case, it is often necessary to estimate the full state of the system from limited measurements.
4. **Interpretability and physical understanding.** Perhaps a more fundamental goal of dynamical systems is to provide physical insight and interpretability into a system's behavior through analyzing trajectories and solutions to the governing equations of motion.

Real-world systems are generally nonlinear and exhibit multi-scale behavior in both space and time. It must also be assumed that there is uncertainty in the equations of motion, in the specification of parameters, and in the measurements of the system. Some systems are more sensitive to this uncertainty than others, and probabilistic approaches must be

used. Increasingly, it is also the case that the basic equations of motion are not specified and they might be intractable to derive from first principles.

This chapter will cover recent data-driven techniques to identify and analyze dynamical systems. The majority of this chapter addresses two primary challenges of modern dynamical systems:

1. **Nonlinearity.** Nonlinearity remains a primary challenge in analyzing and controlling dynamical systems, giving rise to complex global dynamics. We saw above that linear systems may be completely characterized in terms of the spectral decomposition (i.e., eigenvalues and eigenvectors) of the matrix $\mathbf{A}$, leading to general procedures for prediction, estimation, and control. No such overarching framework exists for nonlinear systems, and developing this general framework is a mathematical grand challenge of the 21st century.

   The leading perspective on nonlinear dynamical systems considers the geometry of subspaces of local linearizations around fixed points and periodic orbits, global heteroclinic and homoclinic orbits connecting these structures, and more general attractors [252]. This geometric theory, originating with Poincaré, has transformed how we model complex systems, and its success can be largely attributed to theoretical results, such as the Hartman-Grobman theorem, which establish when and where it is possible to approximate a nonlinear system with linear dynamics. Thus, it is often possible to apply the wealth of linear analysis techniques in a small neighborhood of a fixed point or periodic orbit. Although the geometric perspective provides quantitative locally linear models, global analysis has remained largely qualitative and computational, limiting the theory of nonlinear prediction, estimation, and control away from fixed points and periodic orbits.
2. **Unknown dynamics.** Perhaps an even more central challenge arises from the lack of known governing equations for many modern systems of interest. Increasingly, researchers are tackling more complex and realistic systems, such as are found in neuroscience, epidemiology, and ecology. In these fields, there is a basic lack of known *physical laws* that provide first principles from which it is possible to derive equations of motion. Even in systems where we do know the governing equations, such as turbulence, protein folding, and combustion, we struggle to find patterns in these high-dimensional systems to uncover intrinsic coordinates and coarse-grained variables along which the dominant behavior evolves.

   Traditionally, physical systems were analyzed by making ideal approximations and then deriving simple differential equation models via Newton's second law. Dramatic simplifications could often be made by exploiting symmetries and clever coordinate systems, as highlighted by the success of Lagrangian and Hamiltonian dynamics [2, 369]. With increasingly complex systems, the paradigm is shifting from this classical approach to data-driven methods to discover governing equations.

   All models are approximations, and with increasing complexity, these approximations often become suspect. Determining what is the correct model is becoming more subjective, and there is a growing need for automated model discovery techniques that illuminate underlying physical mechanisms. There are also often latent variables that are relevant to the dynamics but may go unmeasured. Uncovering these hidden effects is a major challenge for data-driven methods.

Identifying unknown dynamics from data and learning intrinsic coordinates that enable the linear representation of nonlinear systems are two of the most pressing goals of modern dynamical systems. Overcoming the challenges of unknown dynamics and nonlinearity has the promise of transforming our understanding of complex systems, with tremendous potential benefit to nearly all fields of science and engineering.

Throughout this chapter we will explore these issues in further detail and describe a number of the emerging techniques to address these challenges. In particular, there are two key approaches that are defining modern data-driven dynamical systems:

1. **Operator theoretic representations.** To address the issue of nonlinearity, operator-theoretic approaches to dynamical systems are becoming increasingly used. As we will show, it is possible to represent nonlinear dynamical systems in terms of infinite-dimensional but linear operators, such as the Koopman operator from Section 7.4 that advances measurement functions, and the Perron-Frobenius operator that advances probability densities and ensembles through the dynamics.
2. **Data-driven regression and machine learning.** As data becomes increasingly abundant, and we continue to investigate systems that are not amenable to first-principles analysis, regression and machine learning are becoming vital tools to discover dynamical systems from data. This is the basis of many of the techniques described in this chapter, including the dynamic mode decomposition (DMD) in Section 7.2, the sparse identification of nonlinear dynamics (SINDy) in Section 7.3, the data-driven Koopman methods in Section 7.5, as well as the use of genetic programming to identify dynamics from data [68, 477].

It is important to note that many of the methods and perspectives described in this chapter are interrelated, and continuing to strengthen and uncover these relationships is the subject of ongoing research. It is also worth mentioning that a third major challenge is the high-dimensionality associated with many modern dynamical systems, such as are found in population dynamics, brain simulations, and high-fidelity numerical discretizations of partial differential equations. High-dimensionality is addressed extensively in the subsequent chapters on reduced-order models (ROMs).

## 7.2 Dynamic Mode Decomposition (DMD)

Dynamic mode decomposition was developed by Schmid [474, 472] in the fluid dynamics community to identify spatio-temporal coherent structures from high-dimensional data. DMD is based on proper orthogonal decomposition (POD), which utilizes the computationally efficient singular value decomposition (SVD), so that it scales well to provide effective dimensionality reduction in high-dimensional systems. In contrast to SVD/POD, which results in a hierarchy of modes based entirely on spatial correlation and energy content, while largely ignoring temporal information, DMD provides a modal decomposition where each mode consists of spatially correlated structures that have the same linear behavior in time (e.g., oscillations at a given frequency with growth or decay). Thus, DMD not only provides dimensionality reduction in terms of a reduced set of modes, but also provides a model for how these modes evolve in time.

Soon after the development of the original DMD algorithm [474, 472], Rowley, Mezic, and collaborators established an important connection between DMD and Koopman the-

ory [456] (see Section 7.4). DMD may be formulated as an algorithm to identify the best-fit linear dynamical system that advances high-dimensional measurements forward in time [535]. In this way, DMD approximates the Koopman operator restricted to the set of direct measurements of the state of a high-dimensional system. This connection between the computationally straightforward and linear DMD framework and nonlinear dynamical systems has generated considerable interest in these methods [317].

Within a short amount of time, DMD has become a workhorse algorithm for the data-driven characterization of high-dimensional systems. DMD is equally valid for experimental and numerical data, as it is not based on knowledge of the governing equations, but is instead based purely on measurement data. The DMD algorithm may also be seen as connecting the favorable aspects of the SVD (see Chapter 1) for spatial dimensionality reduction and the FFT (see Chapter 2) for temporal frequency identification [129, 317]. Thus, each DMD mode is associated with a particular *eigenvalue* $\lambda = a + ib$, with a particular frequency of oscillation $b$ and growth or decay rate $a$.

There are many variants of DMD and it is connected to existing techniques from system identification and modal extraction. DMD has become especially popular in recent years in large part due to its simple numerical implementation and strong connections to nonlinear dynamical systems via Koopman spectral theory. Finally, DMD is an extremely flexible platform, both mathematically and numerically, facilitating innovations related to compressed sensing, control theory, and multi-resolution techniques. These connections and extensions will be discussed at the end of this section.

### The DMD Algorithm

Several algorithms have been proposed for DMD, although here we present the *exact* DMD framework developed by Tu et al. [535]. Whereas earlier formulations required uniform sampling of the dynamics in time, the approach presented here works with irregularly sampled data and with concatenated data from several different experiments or numerical simulations. Moreover, the exact formulation of Tu et al. provides a precise mathematical definition of DMD that allows for rigorous theoretical results. Finally, exact DMD is based on the efficient and numerically well-conditioned singular value decomposition, as is the original formulation by Schmid [472].

DMD is inherently data-driven, and the first step is to collect a number of pairs of snapshots of the state of a system as it evolves in time. These snapshot pairs may be denoted by $\{(\mathbf{x}(t_k), \mathbf{x}(t'_k)\}_{k=1}^{m}$, where $t'_k = t_k + \Delta t$, and the timestep $\Delta t$ is sufficiently small to resolve the highest frequencies in the dynamics. As before, a snapshot may be the state of a system, such as a three-dimensional fluid velocity field sampled at a number of discretized locations, that is reshaped into a high-dimensional column vector. These snapshots are then arranged into two data matrices, $\mathbf{X}$ and $\mathbf{X}'$:

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}(t_1) & \mathbf{x}(t_2) & \cdots & \mathbf{x}(t_m) \\ | & | & & | \end{bmatrix} \tag{7.13a}$$

$$\mathbf{X}' = \begin{bmatrix} | & | & & | \\ \mathbf{x}(t'_1) & \mathbf{x}(t'_2) & \cdots & \mathbf{x}(t'_m) \\ | & | & & | \end{bmatrix}. \tag{7.13b}$$

The original formulations of Schmid [472] and Rowley et al. [456] assumed uniform sampling in time, so that $t_k = k\Delta t$ and $t'_k = t_k + \Delta t = t_{k+1}$. If we assume uniform sampling in time, we will adopt the notation $\mathbf{x}_k = \mathbf{x}(k\Delta t)$.

The DMD algorithm seeks the leading spectral decomposition (i.e., eigenvalues and eigenvectors) of the best-fit linear operator $\mathbf{A}$ that relates the two snapshot matrices in time:

$$\mathbf{X}' \approx \mathbf{A}\mathbf{X}. \tag{7.14}$$

The best fit operator $\mathbf{A}$ then establishes a linear dynamical system that best advances snapshot measurements forward in time. If we assume uniform sampling in time, this becomes:

$$\mathbf{x}_{k+1} \approx \mathbf{A}\mathbf{x}_k. \tag{7.15}$$

Mathematically, the best-fit operator $\mathbf{A}$ is defined as

$$\mathbf{A} = \underset{\mathbf{A}}{\operatorname{argmin}} \|\mathbf{X}' - \mathbf{A}\mathbf{X}\|_F = \mathbf{X}'\mathbf{X}^\dagger \tag{7.16}$$

where $\|\cdot\|_F$ is the Frobenius norm and $^\dagger$ denotes the pseudo-inverse. The optimized DMD algorithm generalizes the optimization framework of exact DMD to perform a regression to exponential time dynamics, thus providing an improved computation of the DMD modes and their eigenvalues [20].

It is worth noting at this point that the matrix $\mathbf{A}$ in (7.15) closely resembles the Koopman operator in (7.53), if we choose direct linear measurements of the state, so that $\mathbf{g}(\mathbf{x}) = \mathbf{x}$. This connection was originally established by Rowley, Mezic and collaborators [456], and has sparked considerable interest in both DMD and Koopman theory. These connections will be explored in more depth below.

For a high-dimensional state vector $\mathbf{x} \in \mathbb{R}^n$, the matrix $\mathbf{A}$ has $n^2$ elements, and representing this operator, let alone computing its spectral decomposition, may be intractable. Instead, the DMD algorithm leverages dimensionality reduction to compute the dominant eigenvalues and eigenvectors of $\mathbf{A}$ without requiring any explicit computations using $\mathbf{A}$ directly. In particular, the pseudo-inverse $\mathbf{X}^\dagger$ in (7.16) is computed via the singular value decomposition of the matrix $\mathbf{X}$. Since this matrix typically has far fewer columns than rows, i.e. $m \ll n$, there are at most $m$ nonzero singular values and corresponding singular vectors, and hence the matrix $\mathbf{A}$ will have at most rank $m$. Instead of computing $\mathbf{A}$ directly, we compute the projection of $\mathbf{A}$ onto these leading singular vectors, resulting in a small matrix $\tilde{\mathbf{A}}$ of size at most $m \times m$. A major contribution of Schmid [472] was a procedure to approximate the high-dimensional DMD modes (eigenvectors of $\mathbf{A}$) from the reduced matrix $\tilde{\mathbf{A}}$ and the data matrix $\mathbf{X}$ without ever resorting to computations on the full $\mathbf{A}$. Tu et al. [535] later proved that these approximate modes are in fact exact eigenvectors of the full $\mathbf{A}$ matrix under certain conditions. Thus, the exact DMD algorithm of Tu et al. [535] is given by the following steps:

**Step 1.** Compute the singular value decomposition of $\mathbf{X}$ (see Chapter 1):

$$\mathbf{X} \approx \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^*, \tag{7.17}$$

where $\tilde{\mathbf{U}} \in \mathbb{C}^{n\times r}$, $\tilde{\mathbf{\Sigma}} \in \mathbb{C}^{r\times r}$, and $\tilde{\mathbf{V}} \in \mathbb{C}^{m\times r}$ and $r \leq m$ denotes either the exact or approximate rank of the data matrix $\mathbf{X}$. In practice, choosing the approximate rank $r$ is one of the most important and subjective steps in DMD, and in dimensionality reduction in general. We advocate the principled hard-thresholding algorithm of Gavish and Donoho [200] to determine $r$ from noisy data (see Section 1.7). The columns of the matrix $\tilde{\mathbf{U}}$ are also known as POD modes, and they satisfy $\tilde{\mathbf{U}}^*\tilde{\mathbf{U}} = \mathbf{I}$. Similarly, columns of $\tilde{\mathbf{V}}$ are orthonormal and satisfy $\tilde{\mathbf{V}}^*\tilde{\mathbf{V}} = \mathbf{I}$.

**Step 2.** According to (7.16), the full matrix $\mathbf{A}$ may be obtained by computing the pseudo-inverse of $\mathbf{X}$:

$$\mathbf{A} = \mathbf{X}'\tilde{\mathbf{V}}\tilde{\mathbf{\Sigma}}^{-1}\tilde{\mathbf{U}}^*. \tag{7.18}$$

However, we are only interested in the leading $r$ eigenvalues and eigenvectors of $\mathbf{A}$, and we may thus project $\mathbf{A}$ onto the POD modes in $\mathbf{U}$:

$$\tilde{\mathbf{A}} = \tilde{\mathbf{U}}^*\mathbf{A}\tilde{\mathbf{U}} = \tilde{\mathbf{U}}^*\mathbf{X}'\tilde{\mathbf{V}}\tilde{\mathbf{\Sigma}}^{-1}. \tag{7.19}$$

The key observation here is that the reduced matrix $\tilde{\mathbf{A}}$ has the same nonzero eigenvalues as the full matrix $\mathbf{A}$. Thus, we need only compute the reduced $\tilde{\mathbf{A}}$ directly, without ever working with the high-dimensional $\mathbf{A}$ matrix. The reduced-order matrix $\tilde{\mathbf{A}}$ defines a linear model for the dynamics of the vector of POD coefficients $\tilde{\mathbf{x}}$:

$$\tilde{\mathbf{x}}_{k+1} = \tilde{\mathbf{A}}\tilde{\mathbf{x}}_k. \tag{7.20}$$

Note that the matrix $\tilde{\mathbf{U}}$ provides a map to reconstruct the full state $\mathbf{x}$ from the reduced state $\tilde{\mathbf{x}}$: $\mathbf{x} = \tilde{\mathbf{U}}\tilde{\mathbf{x}}$.

**Step 3.** The spectral decomposition of $\tilde{\mathbf{A}}$ is computed:

$$\tilde{\mathbf{A}}\mathbf{W} = \mathbf{W}\mathbf{\Lambda}. \tag{7.21}$$

The entries of the diagonal matrix $\mathbf{\Lambda}$ are the DMD eigenvalues, which also correspond to eigenvalues of the full $\mathbf{A}$ matrix. The columns of $\mathbf{W}$ are eigenvectors of $\tilde{\mathbf{A}}$, and provide a coordinate transformation that diagonalizes the matrix. These columns may be thought of as linear combinations of POD mode amplitudes that behave linearly with a single temporal pattern given by $\lambda$.

**Step 4.** The high-dimensional DMD modes $\mathbf{\Phi}$ are reconstructed using the eigenvectors $\mathbf{W}$ of the reduced system and the time-shifted snapshot matrix $\mathbf{X}'$ according to:

$$\mathbf{\Phi} = \mathbf{X}'\tilde{\mathbf{V}}\tilde{\mathbf{\Sigma}}^{-1}\mathbf{W}. \tag{7.22}$$

Remarkably, these DMD modes are eigenvectors of the high-dimensional $\mathbf{A}$ matrix corresponding to the eigenvalues in $\mathbf{\Lambda}$, as shown in Tu et al. [535]:

$$\begin{aligned}
\mathbf{A}\mathbf{\Phi} &= (\mathbf{X}'\tilde{\mathbf{V}}\tilde{\mathbf{\Sigma}}^{-1}\underbrace{\tilde{\mathbf{U}}^*)(\mathbf{X}'\tilde{\mathbf{V}}\tilde{\mathbf{\Sigma}}^{-1}}_{\tilde{\mathbf{A}}}\mathbf{W}) \\
&= \mathbf{X}'\tilde{\mathbf{V}}\tilde{\mathbf{\Sigma}}^{-1}\tilde{\mathbf{A}}\mathbf{W} \\
&= \mathbf{X}'\tilde{\mathbf{V}}\tilde{\mathbf{\Sigma}}^{-1}\mathbf{W}\mathbf{\Lambda} \\
&= \mathbf{\Phi}\mathbf{\Lambda}.
\end{aligned}$$

In the original paper by Schmid [472], DMD modes are computed using $\boldsymbol{\Phi} = \tilde{\mathbf{U}}\mathbf{W}$, which are known as *projected modes*; however, these modes are not guaranteed to be exact eigenvectors of $\mathbf{A}$. Because $\mathbf{A}$ is defined as $\mathbf{A} = \mathbf{X}'\mathbf{X}^{\dagger}$, eigenvectors of $\mathbf{A}$ should be in the column space of $\mathbf{X}'$, as in the exact DMD definition, instead of the column space of $\mathbf{X}$ in the original DMD algorithm. In practice, the column spaces of $\mathbf{X}$ and $\mathbf{X}'$ will tend to be nearly identical for dynamical systems with low-rank structure, so that the projected and exact DMD modes often converge.

To find a DMD mode corresponding to a zero eigenvalue, $\lambda = 0$, it is possible to use the exact formulation if $\boldsymbol{\phi} = \mathbf{X}'\tilde{\mathbf{V}}\tilde{\boldsymbol{\Sigma}}^{-1}\mathbf{w} \neq 0$. However, if this expression is null, then the projected mode $\boldsymbol{\phi} = \tilde{\mathbf{U}}\mathbf{w}$ should be used.

### *Historical Perspective*

In the original formulation, the snapshot matrices $\mathbf{X}$ and $\mathbf{X}'$ were formed with a collection of sequential snapshots, evenly spaced in time:

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_m \\ | & | & & | \end{bmatrix} \tag{7.23a}$$

$$\mathbf{X}' = \begin{bmatrix} | & | & & | \\ \mathbf{x}_2 & \mathbf{x}_3 & \cdots & \mathbf{x}_{m+1} \\ | & | & & | \end{bmatrix}. \tag{7.23b}$$

Thus, the matrix $\mathbf{X}$ can be written in terms of iterations of the matrix $\mathbf{A}$ as:

$$\mathbf{X} \approx \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{A}\mathbf{x}_1 & \cdots & \mathbf{A}^{m-1}\mathbf{x}_1 \\ | & | & & | \end{bmatrix}. \tag{7.24}$$

Thus, the columns of the matrix $\mathbf{X}$ belong to a Krylov subspace generated by the propagator $\mathbf{A}$ and the initial condition $\mathbf{x}_1$. In addition, the matrix $\mathbf{X}'$ may be related to $\mathbf{X}$ through the *shift* operator as:

$$\mathbf{X}' = \mathbf{X}\mathbf{S}, \tag{7.25}$$

where $\mathbf{S}$ is defined as

$$\mathbf{S} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & a_1 \\ 1 & 0 & 0 & \cdots & 0 & a_2 \\ 0 & 1 & 0 & \cdots & 0 & a_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & a_m \end{bmatrix}. \tag{7.26}$$

Thus, the first $m-1$ columns of $\mathbf{X}'$ are obtained directly by shifting the corresponding columns of $\mathbf{X}$, and the last column is obtained as a best-fit combination of the $m$ columns of $\mathbf{X}$ that minimizes the residual. In this way, the DMD algorithm resembles an Arnoldi algorithm used to find the dominant eigenvalues and eigenvectors of a matrix $\mathbf{A}$ through iteration. The matrix $\mathbf{S}$ will share eigenvalues with the high-dimensional $\mathbf{A}$ matrix, so that decomposition of $\mathbf{S}$ may be used to obtain dynamic modes and eigenvalues. However, computations based on $\mathbf{S}$ is not as numerically stable as the exact algorithm above.

### *Spectral Decomposition and DMD Expansion*

One of the most important aspects of the DMD is the ability to expand the system state in terms of a data-driven spectral decomposition:

$$\mathbf{x}_k = \sum_{j=1}^{r} \boldsymbol{\phi}_j \lambda_j^{k-1} b_j = \boldsymbol{\Phi} \boldsymbol{\Lambda}^{k-1} \mathbf{b}, \tag{7.27}$$

where $\boldsymbol{\phi}_j$ are DMD modes (eigenvectors of the $\mathbf{A}$ matrix), $\lambda_j$ are DMD eigenvalues (eigenvalues of the $\mathbf{A}$ matrix), and $b_j$ is the mode amplitude. The vector $\mathbf{b}$ of mode amplitudes is generally computed as

$$\mathbf{b} = \boldsymbol{\Phi}^{\dagger} \mathbf{x}_1. \tag{7.28}$$

More principled approaches to select dominant and sparse modes have been considered [129, 270]. However, computing the mode amplitudes is generally quite expensive, even using the straightforward definition in (7.28). Instead, it is possible to compute these amplitudes using POD projected data:

$$\mathbf{x}_1 = \boldsymbol{\Phi} \mathbf{b} \tag{7.29a}$$

$$\implies \tilde{\mathbf{U}} \tilde{\mathbf{x}}_1 = \mathbf{X}' \tilde{\mathbf{V}} \tilde{\boldsymbol{\Sigma}}^{-1} \mathbf{W} \mathbf{b} \tag{7.29b}$$

$$\implies \tilde{\mathbf{x}}_1 = \tilde{\mathbf{U}}^* \mathbf{X}' \tilde{\mathbf{V}} \tilde{\boldsymbol{\Sigma}}^{-1} \mathbf{W} \mathbf{b} \tag{7.29c}$$

$$\implies \tilde{\mathbf{x}}_1 = \tilde{\mathbf{A}} \mathbf{W} \mathbf{b} \tag{7.29d}$$

$$\implies \tilde{\mathbf{x}}_1 = \mathbf{W} \boldsymbol{\Lambda} \mathbf{b} \tag{7.29e}$$

$$\implies \mathbf{b} = (\mathbf{W} \boldsymbol{\Lambda})^{-1} \tilde{\mathbf{x}}_1. \tag{7.29f}$$

The matrices $\mathbf{W}$ and $\boldsymbol{\Lambda}$ are both size $r \times r$, as opposed to the large $\boldsymbol{\Phi}$ matrix that is $n \times r$.

The spectral expansion above may also be written in continuous time by introducing the continuous eigenvalues $\omega = \log(\lambda)/\Delta t$:

$$\mathbf{x}(t) = \sum_{j=1}^{r} \boldsymbol{\phi}_j e^{\omega_j t} b_j = \boldsymbol{\Phi} \exp(\boldsymbol{\Omega} t) \mathbf{b}, \tag{7.30}$$

where $\boldsymbol{\Omega}$ is a diagonal matrix containing the continuous-time eigenvalues $\omega_j$.

### Example and Code

A basic DMD code is provided here:

```
function [Phi, Lambda, b] = DMD(X,Xprime,r)

[U,Sigma,V] = svd(X,'econ');        % Step 1
Ur = U(:,1:r);
Sigmar = Sigma(1:r,1:r);
Vr = V(:,1:r);

Atilde = Ur'*Xprime*Vr/Sigmar;      % Step 2
[W,Lambda] = eig(Atilde);           % Step 3

Phi = Xprime*(Vr/Sigmar)*W;         % Step 4
alpha1 = Sigmar*Vr(1,:)';
b = (W*Lambda)\alpha1;
```

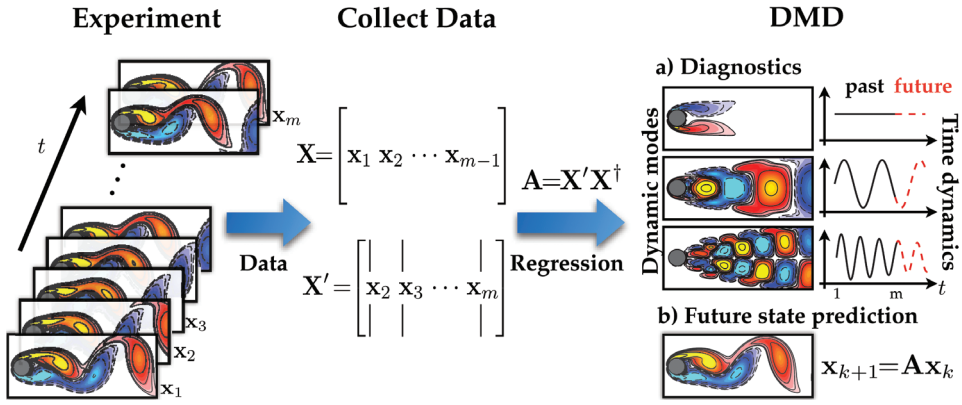


**Figure 7.3** Overview of DMD illustrated on the fluid flow past a circular cylinder at Reynolds number 100. *Reproduced from [317].*

This DMD code is demonstrated in Fig. 7.3 for the fluid flow past a circular cylinder at Reynolds number 100, based on the cylinder diameter. The two-dimensional Navier-Stokes equations are simulated using the immersed boundary projection method (IBPM) solver[1] based on the fast multi-domain method of Taira and Colonius [511, 135]. The data required for this example may be downloaded without running the IBPM code at *dmdbook.com*.

With this data, it is simple to compute the dynamic mode decomposition:

```
% VORTALL contains flow fields reshaped into column vectors
X = VORTALL;
[Phi, Lambda, b] = DMD(X(:,1:end-1),X(:,2:end),21);
```

## Extensions, Applications, and Limitations

One of the major advantages of dynamic mode decomposition is its simple framing in terms of linear regression. DMD does not require knowledge of governing equations. For this reason, DMD has been rapidly extended to include several methodological innovations and has been widely applied beyond fluid dynamics [317], where it originated. Here, we present a number of the leading algorithmic extensions and promising domain applications, and we also present current limitations of the DMD theory that must be addressed in future research.

### *Methodological Extensions*

- **Compression and randomized linear algebra.** DMD was originally designed for high-dimensional data sets in fluid dynamics, such as a fluid velocity or vorticity field, which may contain millions of degrees of freedom. However, the fact that DMD often uncovers low-dimensional structure in these high dimensional data implies that there may be more efficient measurement and computational strategies based on principles of *sparsity* (see Chapter 3). There have been several independent and highly successful extensions and modifications of DMD to exploit low-rank structure and sparsity.

[1] The IBPM code is publicly available at: *https://github.com/cwrowley/ibpm*.

In 2014, Jovanovic et al. [270] used sparsity promoting optimization to identify the fewest DMD modes required to describe a data set, essentially identifying a few dominant DMD mode amplitudes in $\mathbf{b}$. The alternative approach, of testing and comparing all subsets of DMD modes, represents a computationally intractable brute force search.

Another line of work is based on the fact that DMD modes generally admit a sparse representation in Fourier or wavelet bases. Moreover, the time dynamics of each mode are simple pure tone harmonics, which are the definition of sparse in a Fourier basis. This sparsity has facilitated several efficient measurement strategies that reduce the number of measurements required in time [536] and space [96, 225, 174], based on compressed sensing. This has the broad potential to enable high-resolution characterization of systems from under-resolved measurements.

Related to the use of compressed sensing, randomized linear algebra has recently been used to accelerate DMD computations when full-state data is available. Instead of collecting subsampled measurements and using compressed sensing to infer high-dimensional structures, randomized methods start with full data and then randomly project into a lower-dimensional subspace, where computations may be performed more efficiently. Bistrian and Navon [66] have successfully accelerated DMD using a randomized singular value decomposition, and Erichson et al. [175] demonstrates how all of the expensive DMD computations may be performed in a projected subspace.

Finally, libraries of DMD modes have also been used to identify dynamical regimes [308], based on the sparse representation for classification [560] (see Section 3.6), which was used earlier to identify dynamical regimes using libraries of POD modes [80, 98].

- **Inputs and control.** A major strength of DMD is the ability to describe complex and high-dimensional dynamical systems in terms of a small number of dominant modes, which represent spatio-temporal coherent structures. Reducing the dimensionality of the system from $n$ (often millions or billions) to $r$ (tens or hundreds) enables faster and lower-latency prediction and estimation. Lower-latency predictions generally translate directly into controllers with higher performance and robustness. Thus, compact and efficient representations of complex systems such as fluid flows have been long-sought, resulting in the field of reduced order modeling. However, the original DMD algorithm was designed to characterize naturally evolving systems, without accounting for the effect of actuation and control.

  Shortly after the original DMD algorithm, Proctor et al. [434] extended the algorithm to disambiguate between the natural unforced dynamics and the effect of actuation. This essentially amounts to a generalized evolution equation

$$\mathbf{x}_{k+1} \approx \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k, \tag{7.31}$$

  which results in another linear regression problem (see Section 10.1).

  The original motivation for DMD with control (DMDc) was the use of DMD to characterize epidemiological systems (e.g., malaria spreading across a continent), where it is not possible to stop intervention efforts, such as vaccinations and bed nets, in order to characterize the unforced dynamics [433].

Since the original DMDc algorithm, the compressed sensing DMD and DMDc algorithms have been combined, resulting in a new framework for compressive system identification [30]. In this framework, it is possible to collect undersampled measurements of an actuated system and identify an accurate and efficient low-order model, related to DMD and the eigensystem realization algorithm (ERA; see Section 9.3) [272].

DMDc models, based on linear and nonlinear measurements of the system, have recently been used with model predictive control (MPC) for enhanced control of nonlinear systems by Korda and Mezić [302]. Model predictive control using DMDc models were subsequently used as a benchmark comparison for MPC based on fully nonlinear models in the work of Kaiser et al. [277], and the DMDc models performed surprisingly well, even for strongly nonlinear systems.

- **Nonlinear measurements.** Much of the excitement around DMD is due to the strong connection to nonlinear dynamics via the Koopman operator [456]. Indeed, DMD is able to accurately characterize periodic and quasi-periodic behavior, even in nonlinear systems, as long as a sufficient amount of data is collected. However, the basic DMD algorithm uses linear measurements of the system, which are generally not rich enough to characterize truly nonlinear phenomena, such as transients, intermittent phenomena, or broadband frequency cross-talk. In Williams et al. [556], DMD measurements were augmented to include nonlinear measurements of the system, enriching the basis used to represent the Koopman operator. The so-called *extended DMD* (eDMD) algorithm then seeks to obtain a linear model $\mathbf{A_Y}$ advancing nonlinear measurements $\mathbf{y} = \mathbf{g}(\mathbf{x})$:

$$\mathbf{y}_{k+1} \approx \mathbf{A_Y}\mathbf{y}_k. \tag{7.32}$$

For high-dimensional systems, this augmented state $\mathbf{y}$ may be intractably large, motivating the use of kernel methods to approximate the evolution operator $\mathbf{A_Y}$ [557]. This kernel DMD has since been extended to include dictionary learning techniques [332].

It has recently been shown that eDMD is equivalent to the variational approach of conformation dynamics (VAC) [405, 407, 408], first derived by Noé and Nüske in 2013 to simulate molecular dynamics with a broad separation of timescales. Further connections between eDMD and VAC and between DMD and the time lagged independent component analysis (TICA) are explored in a recent review [293]. A key contribution of VAC is a variational score enabling the objective assessment of Koopman models via cross-validation.

Following the extended DMD, it was shown that there are relatively restrictive conditions for obtaining a linear regression model that includes the original state of the system [92]. For nonlinear systems with multiple fixed points, periodic orbits, and other attracting structures, there is no finite-dimensional linear system including the state $\mathbf{x}$ that is topologically conjugate to the nonlinear system. Instead, it is important to identify Koopman invariant subspaces, spanned by eigenfunctions of the Koopman operator; in general, it will not be possible to directly write the state $\mathbf{x}$ in the span of these eigenvectors, although it may be possible to identify $\mathbf{x}$ through a unique inverse. A practical algorithm for identifying eigenfunctions is provided by Kaiser et al. [276].

- **De-noising.** The DMD algorithm is purely data-driven, and is thus equally applicable to experimental and numerical data. When characterizing experimental data with DMD, the effects of sensor noise and stochastic disturbances must be accounted for. The original DMD algorithm is particularly sensitive to noise, and it was shown that significant and systematic biases are introduced to the eigenvalue distribution [164, 28, 147, 241]. Although increased sampling decreases the variance of the eigenvalue distribution, it does not remove the bias [241].

  There are several approaches to correct for the effect of sensor noise and disturbances. Hemati et al. [241] use the total least-squares regression to account for the possibility of noisy measurements and disturbances to the state, replacing the original least-squares regression. Dawson et al. [147] compute DMD on the data in forward and backward time and then average the resulting operator, removing the systematic bias. This work also provides an excellent discussion on the sources of noise and a comparison of various denoising algorithms.

  More recently, Askham and Kutz [20] introduced the optimized DMD algorithm, which uses a variable projection method for nonlinear least squares to compute the DMD for unevenly timed samples, significantly mitigating the bias due to noise. The subspace DMD algorithm of Takeishi et al. [514] also compensates for measurement noise by computing an orthogonal projection of future snapshots onto the space of previous snapshots and then constructing a linear model. Extensions that combine DMD with Bayesian approaches have also been developed [512].
- **Multiresolution.** DMD is often applied to complex, high-dimensional dynamical systems, such as fluid turbulence or epidemiological systems, that exhibit multiscale dynamics in both space and time. Many multiscale systems exhibit transient or intermittent phenomena, such as the El Niño observed in global climate data. These transient dynamics are not captured accurately by DMD, which seeks spatio-temporal modes that are globally coherent across the entire time series of data. To address this challenge, the multiresolution DMD (mrDMD) algorithm was introduced [318], which effectively decomposes the dynamics into different timescales, isolating transient and intermittent patterns. Multiresolution DMD modes were recently shown to be advantageous for sparse sensor placement by Manohar et al. [367].
- **Delay measurements.** Although DMD was developed for high-dimensional data where it is assumed that one has access to the full-state of a system, it is often desirable to characterize spatio-temporal coherent structures for systems with incomplete measurements. As an extreme example, consider a single measurement that oscillates as a sinusoid, $x(t) = \sin(\omega t)$. Although this would appear to be a perfect candidate for DMD, the algorithm incorrectly identifies a real eigenvalue because the data does not have sufficient rank to extract a complex conjugate pair of eigenvalues $\pm i\omega$. This paradox was first explored by Tu et al. [535], where it was discovered that a solution is to stack delayed measurements into a larger matrix to augment the rank of the data matrix and extract phase information. Delay coordinates have been used effectively to extract coherent patterns in neural recordings [90]. The connections between delay DMD and Koopman [91, 18, 144] will be discussed more in Section 7.5.
- **Streaming and parallelized codes.** Because of the computational burden of computing the DMD on high-resolution data, several advances have been made to accel-

erate DMD in streaming applications and with parallelized algorithms. DMD is often used in a streaming setting, where a moving window of snapshots are processed continuously, resulting in redundant computations when new data becomes available. Several algorithms exist for streaming DMD, based on the incremental SVD [242], a streaming method of snapshots SVD [424], and rank-one updates to the DMD matrix [569]. The DMD algorithm is also readily parallelized, as it is based on the SVD. Several parallelized codes are available, based on the QR [466] and SVD [175, 177, 176].

### *Applications*

- **Fluid dynamics.** DMD originated in the fluid dynamics community [472], and has since been applied to a wide range 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 [474, 472], both a cavity flow and a jet were considered. In the original paper of Rowley *et al.* [456], a jet in cross-flow was investigated. It is no surprise that DMD has subsequently been used widely in both cavity flows [472, 350, 481, 43, 42] and jets [473, 49, 483, 475].

  DMD has also been applied to wake flows, including to investigate frequency lock-on [534], the wake past a gurney flap [415], the cylinder wake [28], and dynamic stall [166]. Boundary layers have also been extensively studied with DMD [411, 465, 383]. In acoustics, DMD has been used to capture the near-field and far-field acoustics that result from instabilities observed in shear flows [495]. In combustion, DMD has been used to understand the coherent heat release in turbulent swirl flames [387] and to analyze a rocket combustor [258]. DMD has also been used to analyze non-normal growth mechanisms in thermoacoustic interactions in a Rijke tube. DMD has been compared with POD for reacting flows [459]. DMD has also been used to analyze more exotic flows, including a simulated model of a high-speed train [392]. 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 [222]. DMD has also been used to study self-excited fluctuations in detonation waves [373]. Other problems include identifying hairpin vortices [516], decomposing the flow past a surface mounted cube [393], modeling shallow water equations [65], studying nano fluids past a square cylinder [463], and measuring the growth rate of instabilities in annular liquid sheets [163].
- **Epidemiology.** DMD has recently been applied to investigate epidemiological systems by Proctor and Eckhoff [435]. This is a particularly interpretable application, as modal frequencies often correspond to yearly or seasonal fluctuations. Moreover, the phase of DMD modes gives insight into how disease fronts propagate spatially, potentially informing future intervention efforts. The application of DMD to disease systems also motivated the DMD with control [434], since it is infeasible to stop vaccinations in order to identify the unforced dynamics.
- **Neuroscience.** Complex signals from neural recordings are increasingly high-fidelity and high dimensional, with advances in hardware pushing the frontiers of data collection. DMD has the potential to transform the analysis of such neural recordings, as evidenced in a recent study that identified dynamically relevant features in ECOG data of sleeping patients [90]. Since then, several works

have applied DMD to neural recordings or suggested possible implementation in hardware [3, 85, 520].

- **Video processing.** Separating foreground and background objects in video is a common task in surveillance applications. Real-time separation is a challenge that is only exacerbated by ever increasing video resolutions. DMD provides a flexible platform for video separation, as the background may be approximated by a DMD mode with zero eigenvalue [223, 174, 424].
- **Other applications.** DMD has been applied to an increasingly diverse array of problems, including robotics [56], finance [363], and plasma physics [517]. It is expected that this trend will increase.

### *Challenges*

- **Traveling waves.** DMD is based on the SVD of a data matrix $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$ whose columns are spatial measurements evolving in time. In this case, the SVD is a space-time separation of variables into spatial modes, given by the columns of $\mathbf{U}$, and time dynamics, given by the columns of $\mathbf{V}$. As in POD, DMD thus has limitations for problems that exhibit traveling waves, where separation of variables is known to fail.
- **Transients.** Many systems of interest are characterized by transients and intermittent phenomena. Several methods have been proposed to identify these events, such as the multi-resolution DMD and the use of delay coordinates. However, it is still necessary to formalize the choice of relevant timescales and the window size to compute DMD.
- **Continuous spectrum.** Related to the above, many systems are characterized by broadband frequency content, as opposed to a few distinct and discrete frequencies. This broadband frequency content is also known as a *continuous spectrum*, where every frequency in a continuous range is observed. For example, the simple pendulum exhibits a continuous spectrum, as the system has a natural frequency for small deflections, and this frequency continuously deforms and slows as energy is added to the pendulum. Other systems include nonlinear optics and broadband turbulence. These systems pose a serious challenge for DMD, as they result in a large number of modes, even though the dynamics are likely generated by the nonlinear interactions of a few dominant modes.

  Several data-driven approaches have been recently proposed to handle systems with continuous spectra. Applying DMD to a vector of delayed measurements of a system, the so-called *HAVOK* analysis in Section 7.5, has been shown to approximate the dynamics of chaotic systems, such as the Lorenz system, which exhibits a continuous spectrum. In addition, Lusch et al. [349] showed that it is possible to design a deep learning architecture with an auxiliary network to parameterize the continuous frequency.
- **Strong nonlinearity and choice of measurements.** Although significant progress has been made connecting DMD to nonlinear systems [557], choosing nonlinear measurements to augment the DMD regression is still not an exact science. Identifying measurement subspaces that remain closed under the Koopman operator is an ongoing challenge [92]. Recent progress in deep learning has the potential to enable the representation of extremely complex eigenfunctions from data [550, 368, 513, 564, 412, 349].

## 7.3 Sparse Identification of Nonlinear Dynamics (SINDy)

Discovering dynamical systems models from data is a central challenge in mathematical physics, with a rich history going back at least as far as the time of Kepler and Newton and the discovery of the laws of planetary motion. Historically, this process relied on a combination of high-quality measurements and expert intuition. With vast quantities of data and increasing computational power, the *automated* discovery of governing equations and dynamical systems is a new and exciting scientific paradigm.

Typically, the form of a candidate model is either constrained via prior knowledge of the governing equations, as in Galerkin projection [402, 455, 471, 404, 119, 549, 32, 118] (see Chapter 12), or a handful of heuristic models are tested and parameters are optimized to fit data. Alternatively, best-fit linear models may be obtained using DMD or ERA. Simultaneously identifying the nonlinear structure and parameters of a model from data is considerably more challenging, as there are combinatorially many possible model structures.

The sparse identification of nonlinear dynamics (SINDy) algorithm [95] bypasses the intractable combinatorial search through all possible model structures, leveraging the fact that many dynamical systems

$$\frac{d}{dt}\mathbf{x} = \mathbf{f}(\mathbf{x}) \tag{7.33}$$

have dynamics $\mathbf{f}$ with only a few active terms in the space of possible right-hand side functions; for example, the Lorenz equations in (7.2) only have a few linear and quadratic interaction terms per equation.

We then seek to approximate $\mathbf{f}$ by a generalized linear model

$$\mathbf{f}(\mathbf{x}) \approx \sum_{k=1}^{p} \theta_k(\mathbf{x})\xi_k = \boldsymbol{\Theta}(\mathbf{x})\boldsymbol{\xi}, \tag{7.34}$$

with the fewest nonzero terms in $\boldsymbol{\xi}$ as possible. It is then possible to solve for the relevant terms that are active in the dynamics using sparse regression [518, 573, 236, 264] that penalizes the number of terms in the dynamics and scales well to large problems.

First, time-series data is collected from (7.33) and formed into a data matrix:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}(t_1) & \mathbf{x}(t_2) & \cdots \mathbf{x}(t_m) \end{bmatrix}^T. \tag{7.35}$$

A similar matrix of derivatives is formed:

$$\dot{\mathbf{X}} = \begin{bmatrix} \dot{\mathbf{x}}(t_1) & \dot{\mathbf{x}}(t_2) & \cdots \dot{\mathbf{x}}(t_m) \end{bmatrix}^T. \tag{7.36}$$

In practice, this may be computed directly from the data in $\mathbf{X}$; for noisy data, the total-variation regularized derivative tends to provide numerically robust derivatives [125]. Alternatively, it is possible to formulate the SINDy algorithm for discrete-time systems $\mathbf{x}_{k+1} = \mathbf{F}(\mathbf{x}_k)$, as in the DMD algorithm, and avoid derivatives entirely.

A library of candidate nonlinear functions $\boldsymbol{\Theta}(\mathbf{X})$ may be constructed from the data in $\mathbf{X}$:

$$\boldsymbol{\Theta}(\mathbf{X}) = \begin{bmatrix} \mathbf{1} & \mathbf{X} & \mathbf{X}^2 & \cdots & \mathbf{X}^d & \cdots & \sin(\mathbf{X}) & \cdots \end{bmatrix}. \tag{7.37}$$

Here, the matrix $\mathbf{X}^d$ denotes a matrix with column vectors given by all possible time-series of $d$-th degree polynomials in the state $\mathbf{x}$. In general, this library of candidate functions is only limited by one's imagination.

The dynamical system in (7.33) may now be represented in terms of the data matrices in (7.36) and (7.37) as

$$\dot{\mathbf{X}} = \boldsymbol{\Theta}(\mathbf{X})\boldsymbol{\Xi}. \tag{7.38}$$

Each column $\boldsymbol{\xi}_k$ in $\boldsymbol{\Xi}$ is a vector of coefficients determining the active terms in the $k$-th row in (7.33). A parsimonious model will provide an accurate model fit in (7.38) with as few terms as possible in $\boldsymbol{\Xi}$. Such a model may be identified using a convex $\ell_1$-regularized sparse regression:

$$\boldsymbol{\xi}_k = \text{argmin}_{\boldsymbol{\xi}'_k} \|\dot{\mathbf{X}}_k - \boldsymbol{\Theta}(\mathbf{X})\boldsymbol{\xi}'_k\|_2 + \lambda\|\boldsymbol{\xi}'_k\|_1. \tag{7.39}$$

Here, $\dot{\mathbf{X}}_k$ is the $k$-th column of $\dot{\mathbf{X}}$, and $\lambda$ is a sparsity-promoting knob. Sparse regression, such as the LASSO [518] or the sequential thresholded least-squares (STLS) algorithm used in SINDy [95], improves the numerical robustness of this identification for noisy overdetermined problems, in contrast to earlier methods [548] that used compressed sensing [150, 109, 112, 111, 113, 39, 529]. We advocate the STLS (Code 7.1) to select active terms.

**Code 7.1** Sequentially thresholded least-squares.

```
function Xi = sparsifyDynamics(Theta,dXdt,lambda,n)
% Compute Sparse regression: sequential least squares
Xi = Theta\dXdt;  % Initial guess: Least-squares

% Lambda is our sparsification knob.
for k=1:10
    smallinds = (abs(Xi)<lambda);   % Find small coefficients
    Xi(smallinds)=0;                % and threshold
    for ind = 1:n                   % n is state dimension
        biginds = ~smallinds(:,ind);
% Regress dynamics onto remaining terms to find sparse Xi
        Xi(biginds,ind) = Theta(:,biginds)\dXdt(:,ind);
    end
end
```

The sparse vectors $\boldsymbol{\xi}_k$ may be synthesized into a dynamical system:

$$\dot{x}_k = \boldsymbol{\Theta}(\mathbf{x})\boldsymbol{\xi}_k. \tag{7.40}$$

Note that $x_k$ is the $k$-th element of $\mathbf{x}$ and $\boldsymbol{\Theta}(\mathbf{x})$ is a row vector of symbolic functions of $\mathbf{x}$, as opposed to the data matrix $\boldsymbol{\Theta}(\mathbf{X})$. Fig. 7.4 shows how SINDy may be used to discover the Lorenz equations from data. Code 7.2 generates data and performs the SINDy regression for the Lorenz system.

**Code 7.2** SINDy regression to identify the Lorenz system from data.

```
%% Generate Data
Beta = [10; 28; 8/3]; % Lorenz's parameters (chaotic)
n = 3;
x0=[-8; 8; 27];  % Initial condition
tspan=[.01:.01:50];
options = odeset('RelTol',1e-12,'AbsTol',1e-12*ones(1,n));
[t,x]=ode45(@(t,x) lorenz(t,x,Beta),tspan,x0,options);

%% Compute Derivative
for i=1:length(x)
```

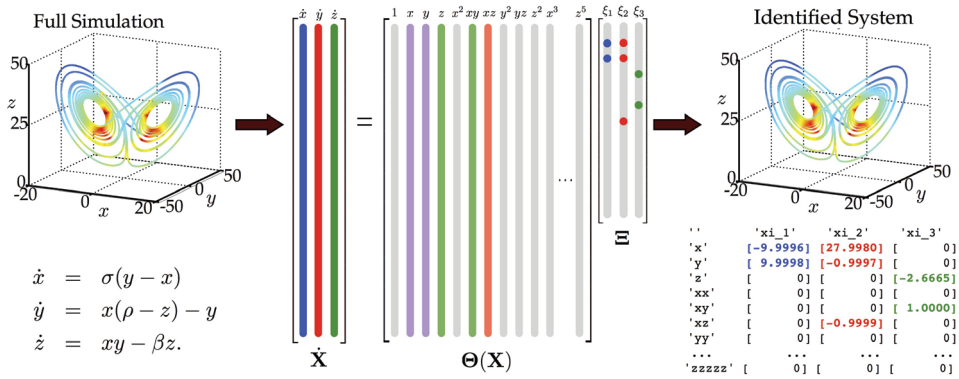


**Figure 7.4** Schematic of the sparse identification of nonlinear dynamics (SINDy) algorithm [95]. Parsimonious models are selected from a library of candidate nonlinear terms using sparse regression. This library $\boldsymbol{\Theta}(\mathbf{X})$ may be constructed purely from measurement data. *Modified from Brunton et al. [95].*

```
    dx(i,:) = lorenz(0,x(i,:),Beta);
end

%% Build library and compute sparse regression
Theta = poolData(x,n,3);  % up to third order polynomials
lambda = 0.025;       % lambda is our sparsification knob.
Xi = sparsifyDynamics(Theta,dx,lambda,n)
```

This code also relies on a function **poolData** that generates the library $\boldsymbol{\Theta}$. In this case, polynomials up to third order are used. This code is available online.

The output of the SINDy algorithm is a sparse matrix of coefficients $\boldsymbol{\Xi}$:

```
    ''        'xdot'          'ydot'          'zdot'
    '1'         [        0]     [       0]     [       0]
    'x'         [-10.0000]     [28.0000]     [       0]
    'y'         [ 10.0000]     [-1.0000]     [       0]
    'z'         [        0]     [       0]     [-2.6667]
    'xx'        [        0]     [       0]     [       0]
    'xy'        [        0]     [       0]     [ 1.0000]
    'xz'        [        0]     [-1.0000]     [       0]
    'yy'        [        0]     [       0]     [       0]
    'yz'        [        0]     [       0]     [       0]
    'zz'        [        0]     [       0]     [       0]
    'xxx'       [        0]     [       0]     [       0]
    'xxy'       [        0]     [       0]     [       0]
    'xxz'       [        0]     [       0]     [       0]
    'xyy'       [        0]     [       0]     [       0]
    'xyz'       [        0]     [       0]     [       0]
    'xzz'       [        0]     [       0]     [       0]
    'yyy'       [        0]     [       0]     [       0]
    'yyz'       [        0]     [       0]     [       0]
    'yzz'       [        0]     [       0]     [       0]
    'zzz'       [        0]     [       0]     [       0]
```

The result of the SINDy regression is a parsimonious model that includes only the most important terms required to explain the observed behavior. The sparse regression procedure used to identify the most parsimonious nonlinear model is a convex procedure.

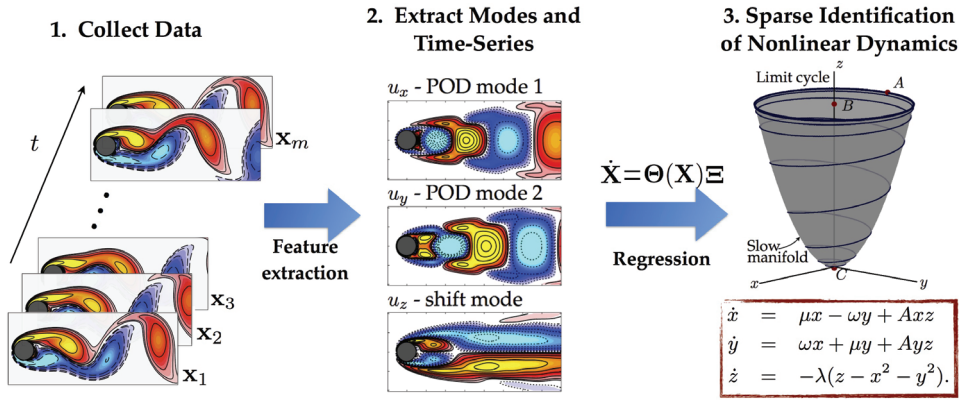


**Figure 7.5** Schematic overview of nonlinear model identification from high-dimensional data using the sparse identification of nonlinear dynamics (SINDy) [95]. This procedure is modular, so that different techniques can be used for the feature extraction and regression steps. In this example of flow past a cylinder, SINDy discovers the model of Noack et al. [402]. *Modified from Brunton et al. [95].*

The alternative approach, which involves regression onto every possible sparse nonlinear structure, constitutes an intractable brute-force search through the combinatorially many candidate model forms. SINDy bypasses this combinatorial search with modern convex optimization and machine learning. It is interesting to note that for discrete-time dynamics, if $\boldsymbol{\Theta}(\mathbf{X})$ consists only of linear terms, and if we remove the sparsity promoting term by setting $\lambda = 0$, then this algorithm reduces to the dynamic mode decomposition [472, 456, 535, 317]. If a least-squares regression is used, as in DMD, then even a small amount of measurement error or numerical round-off will lead to every term in the library being active in the dynamics, which is non-physical. A major benefit of the SINDy architecture is the ability to identify parsimonious models that contain only the required nonlinear terms, resulting in interpretable models that avoid overfitting.

### *Applications, Extensions, and Historical Context*

The SINDy algorithm has recently been applied to identify high-dimensional dynamical systems, such as fluid flows, based on POD coefficients [95, 341, 342]. Fig. 7.5 illustrates the application of SINDy to the flow past a cylinder, where the generalized mean-field model of Noack et al. [402] was discovered from data. SINDy has also been applied to identify models in nonlinear optics [497] and plasma physics [141].

Because SINDy is formulated in terms of linear regression in a nonlinear library, it is highly extensible. The SINDy framework has been recently generalized by Loiseau and Brunton [341] to incorporate known physical constraints and symmetries in the equations by implementing a constrained sequentially thresholded least-squares optimization. In particular, energy preserving constraints on the quadratic nonlinearities in the Navier-Stokes equations were imposed to identify fluid systems [341], where it is known that these constraints promote stability [355, 32, 118]. This work also showed that polynomial libraries are particularly useful for building models of fluid flows in terms of POD coefficients, yielding interpretable models that are related to classical Galerkin projection [95, 341].

Loiseau et al. [342] also demonstrated the ability of SINDy to identify dynamical systems models of high-dimensional systems, such as fluid flows, from a few physical sensor measurements, such as lift and drag measurements on the cylinder in Fig. 7.5. For actuated systems, SINDy has been generalized to include inputs and control [100], and these models are highly effective for model predictive control [277]. It is also possible to extend the SINDy algorithm to identify dynamics with rational function nonlinearities [361], integral terms [469], and based on highly corrupt and incomplete data [522]. SINDy was also recently extended to incorporate information criteria for objective model selection [362], and to identify models with hidden variables using delay coordinates [91]. Finally, the SINDy framework was generalized to include partial derivatives, enabling the identification of partial differential equation models [460, 468]. Several of these recent innovations will be explored in more detail below.

More generally, the use of sparsity-promoting methods in dynamics is quite recent [548, 467, 414, 353, 98, 433, 31, 29, 89, 364, 366]. Other techniques for dynamical system discovery include methods to discover equations from time-series [140], equation-free modeling [288], empirical dynamic modeling [503, 563], modeling emergent behavior [452], the nonlinear autoregressive model with exogenous inputs (NARMAX) [208, 571, 59, 484], and automated inference of dynamics [478, 142, 143]. Broadly speaking, these techniques may be classified as system identification, where methods from statistics and machine learning are used to identify dynamical systems from data. Nearly all methods of system identification involve some form of regression of data onto dynamics, and the main distinction between the various techniques is the degree to which this regression is constrained. For example, the dynamic mode decomposition generates best-fit linear models. Recent nonlinear regression techniques have produced nonlinear dynamic models that preserve physical constraints, such as conservation of energy. A major breakthrough in automated nonlinear system identification was made by Bongard and Lipson [68] and Schmidt and Lipson [477], where they used genetic programming to identify the structure of nonlinear dynamics. These methods are highly flexible and impose very few constraints on the form of the dynamics identified. In addition, SINDy is closely related to NARMAX [59], which identifies the structure of models from time-series data through an orthogonal least squares procedure.

### Discovering Partial Differential Equations

A major extension of the SINDy modeling framework generalized the library to include partial derivatives, enabling the identification of partial differential equations [460, 468]. The resulting algorithm, called the partial differential equation functional identification of nonlinear dynamics (PDE-FIND), has been demonstrated to successfully identify several canonical PDEs from classical physics, purely from noisy data. These PDEs include Navier-Stokes, Kuramoto-Sivashinsky, Schrödinger, reaction diffusion, Burgers, Korteweg–de Vries, and the diffusion equation for Brownian motion [460].

PDE-FIND is similar to SINDy, in that it is based on sparse regression in a library constructed from measurement data. The sparse regression and discovery method is shown in Fig. 7.6. PDE-FIND is outlined below for PDEs in a single variable, although the theory is readily generalized to higher dimensional PDEs. The spatial time-series data is arranged into a single column vector $\Upsilon \in \mathbb{C}^{mn}$, representing data collected over $m$ time points

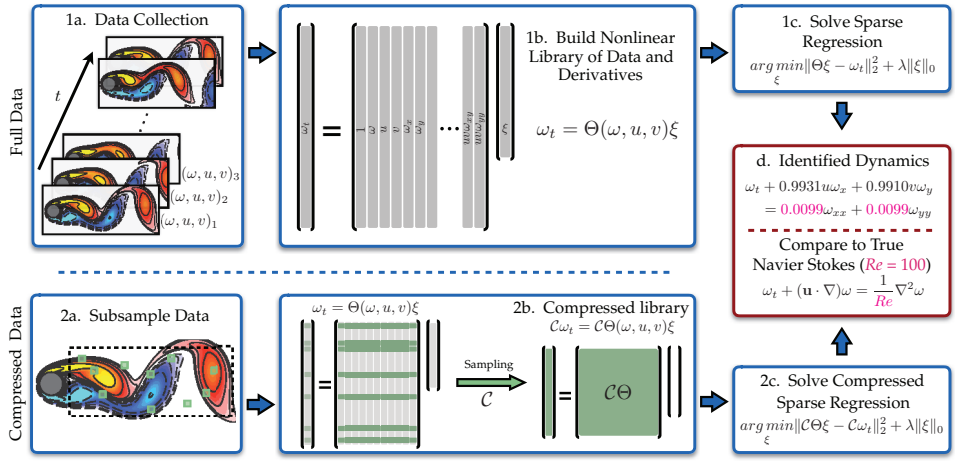


**Figure 7.6** Steps in the PDE functional identification of nonlinear dynamics (PDE-FIND) algorithm, applied to infer the Navier-Stokes equations from data (*reproduced from Rudy et al. [460]*). **1a.** Data is collected as snapshots of a solution to a PDE. **1b.** Numerical derivatives are taken and data is compiled into a large matrix $\boldsymbol{\Theta}$, incorporating candidate terms for the PDE. **1c.** Sparse regressions is used to identify active terms in the PDE. **2a.** For large datasets, sparse sampling may be used to reduce the size of the problem. **2b.** Subsampling the dataset is equivalent to taking a subset of rows from the linear system in (7.42). **2c.** An identical sparse regression problem is formed but with fewer rows. **d.** Active terms in $\boldsymbol{\xi}$ are synthesized into a PDE.

and $n$ spatial locations. Additional inputs, such as a known potential for the Schrödinger equation, or the magnitude of complex data, is arranged into a column vector $\mathbf{Q} \in \mathbb{C}^{mn}$. Next, a library $\boldsymbol{\Theta}(\boldsymbol{\Upsilon}, \mathbf{Q}) \in \mathbb{C}^{mn \times D}$ of $D$ candidate linear and nonlinear terms and partial derivatives for the PDE is constructed. Derivatives are taken either using finite differences for clean data, or when noise is added, with polynomial interpolation. The candidate linear and nonlinear terms and partial derivatives are then combined into a matrix $\boldsymbol{\Theta}(\boldsymbol{\Upsilon}, \mathbf{Q})$ which takes the form:

$$\boldsymbol{\Theta}(\boldsymbol{\Upsilon}, \mathbf{Q}) = \begin{bmatrix} \mathbf{1} & \boldsymbol{\Upsilon} & \boldsymbol{\Upsilon}^2 & \dots & \mathbf{Q} & \dots & \boldsymbol{\Upsilon}_x & \boldsymbol{\Upsilon}\boldsymbol{\Upsilon}_x & \dots \end{bmatrix}. \tag{7.41}$$

Each column of $\boldsymbol{\Theta}$ contains all of the values of a particular candidate function across all of the $mn$ space-time grid points on which data is collected. The time derivative $\boldsymbol{\Upsilon}_t$ is also computed and reshaped into a column vector. Fig. 7.6 demonstrates the data collection and processing. As an example, a column of $\boldsymbol{\Theta}(\boldsymbol{\Upsilon}, \mathbf{Q})$ may be $qu_x^2$.

The PDE evolution can be expressed in this library as follows:

$$\boldsymbol{\Upsilon}_t = \boldsymbol{\Theta}(\boldsymbol{\Upsilon}, \mathbf{Q})\boldsymbol{\xi}. \tag{7.42}$$

Each entry in $\boldsymbol{\xi}$ is a coefficient corresponding to a term in the PDE, and for canonical PDEs, the vector $\boldsymbol{\xi}$ is *sparse*, meaning that only a few terms are active.

If the library $\boldsymbol{\Theta}$ has a sufficiently rich column space that the dynamics are in it's span, then the PDE should be well-represented by (7.42) with a sparse vector of coefficients $\boldsymbol{\xi}$. To identify the few active terms in the dynamics, a sparsity-promoting regression is employed, as in SINDy. Importantly, the regression problem in (7.42) may be poorly conditioned.

**Algorithm 1** STRidge($\boldsymbol{\Theta}$, $\boldsymbol{\Upsilon}_t$, $\lambda$, *tol*, iters)

$\hat{\boldsymbol{\xi}} = arg\,min_{\boldsymbol{\xi}} \|\boldsymbol{\Theta}\boldsymbol{\xi} - \boldsymbol{\Upsilon}_t\|_2^2 + \lambda\|\boldsymbol{\xi}\|_2^2$ % ridge regression
bigcoeffs = $\{j : |\hat{\xi}_j| \geq tol\}$ % select large coefficients
$\hat{\boldsymbol{\xi}}[\sim$ bigcoeffs] = 0 % apply hard threshold
$\hat{\boldsymbol{\xi}}$[bigcoeffs] = STRidge($\boldsymbol{\Theta}$[:, bigcoeffs], $\boldsymbol{\Upsilon}_t$, *tol*, iters $-$ 1)
% recursive call with fewer coefficients
return $\hat{\boldsymbol{\xi}}$

Error in computing the derivatives will be magnified by numerical errors when inverting $\boldsymbol{\Theta}$. Thus a least squares regression radically changes the qualitative nature of the inferred dynamics.

In general, we seek the sparsest vector $\boldsymbol{\xi}$ that satisfies (7.42) with a small residual. Instead of an intractable combinatorial search through all possible sparse vector structures, a common technique is to relax the problem to a convex $\ell_1$ regularized least squares [518]; however, this tends to perform poorly with highly correlated data. Instead, we use ridge regression with hard thresholding, which we call sequential threshold ridge regression (STRidge in Algorithm 1, reproduced from Rudy et al. [460]). For a given tolerance and threshold $\lambda$, this gives a sparse approximation to $\boldsymbol{\xi}$. We iteratively refine the tolerance of Algorithm 1 to find the best predictor based on the selection criteria,

$$\hat{\boldsymbol{\xi}} = \text{argmin}_{\boldsymbol{\xi}} \|\boldsymbol{\Theta}(\boldsymbol{\Upsilon}, \mathbf{Q})\boldsymbol{\xi} - \boldsymbol{\Upsilon}_t\|_2^2 + \epsilon\kappa(\boldsymbol{\Theta}(\boldsymbol{\Upsilon}, \mathbf{Q}))\|\boldsymbol{\xi}\|_0 \tag{7.43}$$

where $\kappa(\boldsymbol{\Theta})$ is the condition number of the matrix $\boldsymbol{\Theta}$, providing stronger regularization for ill-posed problems. Penalizing $\|\boldsymbol{\xi}\|_0$ discourages over fitting by selecting from the optimal position in a Pareto front.

As in the SINDy algorithm, it is important to provide sufficiently rich training data to disambiguate between several different models. For example, Fig. 7.7 illustrates the use of PDE-FIND algorithm identifying the Korteweg–de Vries (KdV) equation. If only a single traveling wave is analyzed, the method incorrectly identifies the standard linear advection equation, as this is the simplest equation that describes a single traveling wave. However, if two traveling waves of different amplitudes are analyzed, the KdV equation is correctly identified, as it describes the different amplitude-dependent wave speeds.

The PDE-FIND algorithm can also be used to identify PDEs based on Lagrangian measurements that follow the path of individual particles. For example, Fig. 7.8 illustrates the identification of the diffusion equation describing Brownian motion of a particle based on a single long time-series measurement of the particle position. In this example, the time series is broken up into several short sequences, and the evolution of the distribution of these positions is used to identify the diffusion equation.

### Extension of SINDy for Rational Function Nonlinearities

Many dynamical systems, such as metabolic and regulatory networks in biology, contain rational function nonlinearities in the dynamics. Often, these rational function nonlinearities arise because of a separation of time scales. Although the original SINDy algorithm is highly flexible in terms of the choice of the library of nonlinearities, it is not straightforward to identify rational functions, since general rational functions are not sparse linear combi-

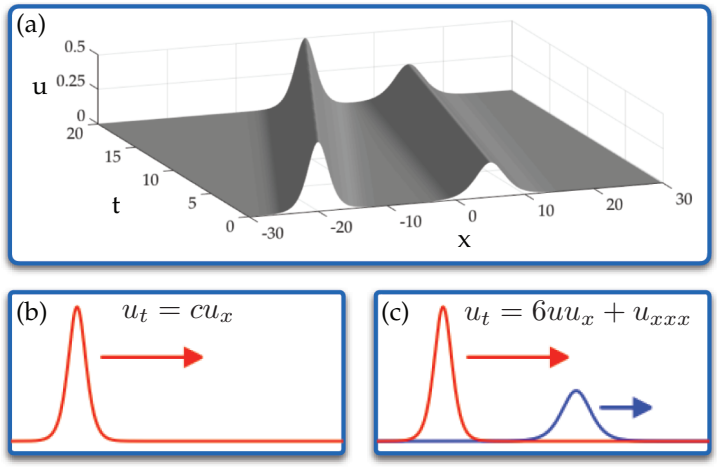


**Figure 7.7** Inferring nonlinearity via observing solutions at multiple amplitudes (*reproduced from Rudy et al. [460]*). (**a**) An example 2-soliton solution to the KdV equation. (**b**) Applying our method to a single soliton solution determines that it solves the standard advection equation. (**c**) Looking at two completely separate solutions reveals nonlinearity.

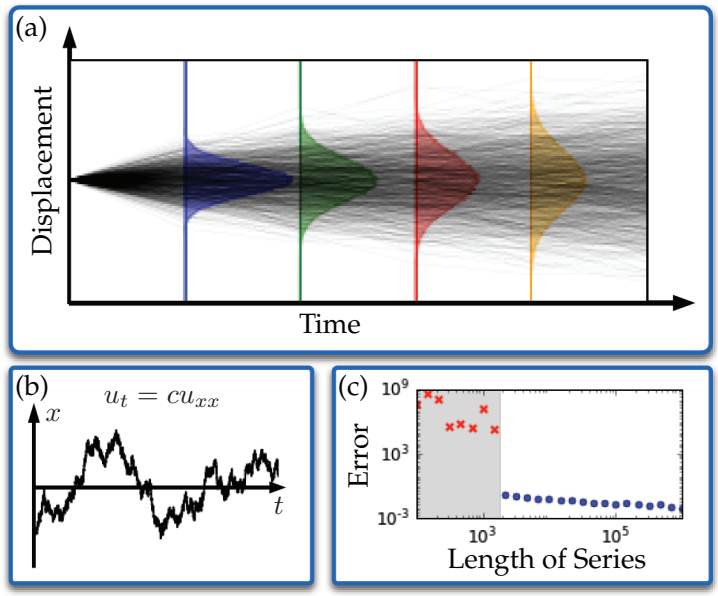


**Figure 7.8** Inferring the diffusion equation from a single Brownian motion (*reproduced from Rudy et al. [460]*). (**a**) Time series is broken into many short random walks that are used to construct histograms of the displacement. (**b**) The Brownian motion trajectory, following the diffusion equation. (**c**) Parameter error ($\|\boldsymbol{\xi}^* - \hat{\boldsymbol{\xi}}\|_1$) vs. length of known time series. Blue symbols correspond to correct identification of the structure of the diffusion model, $u_t = cu_{xx}$.

nations of a few basis functions. Instead, it is necessary to reformulate the dynamics in an *implicit* ordinary differential equation and modify the optimization procedure accordingly, as in Mangan et al. [361].

We consider dynamical systems with rational nonlinearities:

$$\dot{x}_k = \frac{f_N(\mathbf{x})}{f_D(\mathbf{x})} \tag{7.44}$$

where $x_k$ is the $k$-th variable, and $f_N(\mathbf{x})$ and $f_D(\mathbf{x})$ represent numerator and denominator polynomials in the state variable $\mathbf{x}$. For each index $k$, it is possible to multiply both sides by the denominator $f_D$, resulting in the equation:

$$f_N(\mathbf{x}) - f_D(\mathbf{x})\dot{x}_k = 0. \tag{7.45}$$

The implicit form of (7.45) motivates a generalization of the function library $\boldsymbol{\Theta}$ in (7.37) in terms of the state $\mathbf{x}$ and the derivative $\dot{x}_k$:

$$\boldsymbol{\Theta}(\mathbf{X}, \dot{x}_k(\mathbf{t})) = \begin{bmatrix} \boldsymbol{\Theta}_N(\mathbf{X}) & \text{diag}\,(\dot{x}_k(\mathbf{t}))\,\boldsymbol{\Theta}_D(\mathbf{X}) \end{bmatrix}. \tag{7.46}$$

The first term, $\boldsymbol{\Theta}_N(\mathbf{X})$, is the library of numerator monomials in $\mathbf{x}$, as in (7.37). The second term, $\text{diag}\,(\dot{x}_k(\mathbf{t}))\,\boldsymbol{\Theta}_D(\mathbf{X})$, is obtained by multiplying each column of the library of denominator polynomials $\boldsymbol{\Theta}_D(\mathbf{X})$ with the vector $\dot{x}_k(\mathbf{t})$ in an element-wise fashion. For a single variable $x_k$, this would give the following:

$$\text{diag}(\dot{x}_k(\mathbf{t}))\boldsymbol{\Theta}(\mathbf{X}) = \begin{bmatrix} \dot{x}_k(\mathbf{t}) & (\dot{x}_k x_k)(\mathbf{t}) & (\dot{x}_k x_k^2)(\mathbf{t}) & \dots \end{bmatrix}. \tag{7.47}$$

In most cases, we will use the same polynomial degree for both the numerator and denominator library, so that $\boldsymbol{\Theta}_N(\mathbf{X}) = \boldsymbol{\Theta}_D(\mathbf{X})$. Thus, the augmented library in (7.46) is only twice the size of the original library in (7.37).

We may now write the dynamics in (7.45) in terms of the augmented library in (7.46):

$$\boldsymbol{\Theta}(\mathbf{X}, \dot{x}_k(\mathbf{t}))\boldsymbol{\xi}_k = \mathbf{0}. \tag{7.48}$$

The sparse vector of coefficients $\boldsymbol{\xi}_k$ will have nonzero entries for the active terms in the dynamics. However, it is not possible to use the same sparse regression procedure as in SINDy, since the sparsest vector $\boldsymbol{\xi}_k$ that satisfies (7.48) is the trivial zero vector.

Instead, the sparsest nonzero vector $\boldsymbol{\xi}_k$ that satisfies (7.48) is identified as the sparsest vector in the null space of $\boldsymbol{\Theta}$. This is generally a nonconvex problem, although there are recent algorithms developed by Qu et al. [440], based on the alternating directions method (ADM), to identify the sparsest vector in a subspace. Unlike the original SINDy algorithm, this procedure is quite sensitive to noise, as the null-space is numerically approximated as the span of the singular vectors corresponding to small singular value. When noise is added to the data matrix $\mathbf{X}$, and hence to $\boldsymbol{\Theta}$, the noise floor of the singular value decomposition goes up, increasing the rank of the numerical null space.

### *General Formulation for Implicit ODEs*

The optimization procedure above may be generalized to include a larger class of implicit ordinary differential equations, in addition to those containing rational function nonlinearities. The library $\boldsymbol{\Theta}(\mathbf{X}, \dot{x}_k(\mathbf{t}))$ contains a subset of the columns of the library $\boldsymbol{\Theta}(\begin{bmatrix} \mathbf{X} & \dot{\mathbf{X}} \end{bmatrix})$, which is obtained by building nonlinear functions of the state $\mathbf{x}$ and derivative $\dot{\mathbf{x}}$. Identifying the sparsest vector in the null space of $\boldsymbol{\Theta}(\begin{bmatrix} \mathbf{X} & \dot{\mathbf{X}} \end{bmatrix})$ provides more flexibility in identifying

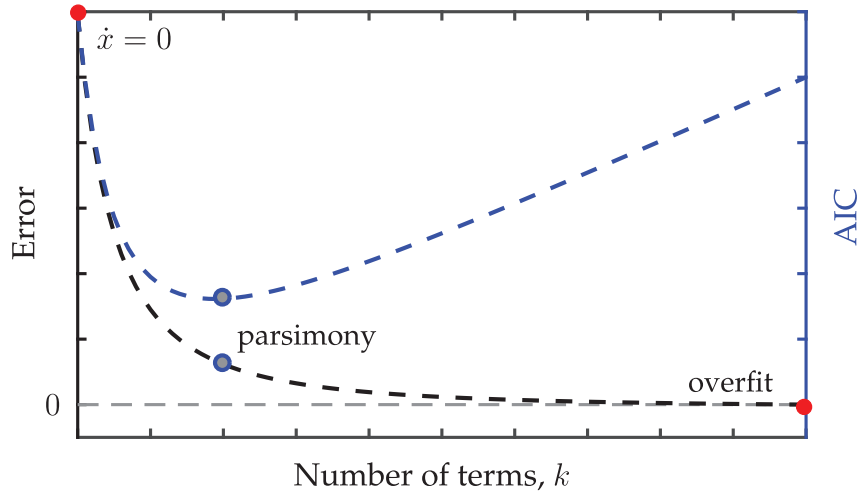


**Figure 7.9** Illustration of model selection using SINDy and information criteria, as in Mangan et al. [362]. The most parsimonious model on the Pareto front is chosen to minimize the AIC score (blue circle), preventing overfitting.

nonlinear equations with mixed terms containing various powers of any combination of derivatives and states. For example, the system given by

$$\dot{x}^2 x^2 - \dot{x}x - x^2 = 0 \tag{7.49}$$

may be represented as a sparse vector in the null space of $\boldsymbol{\Theta}([\mathbf{X} \quad \dot{\mathbf{X}}])$. This formulation may be extended to include higher order derivatives in the library $\boldsymbol{\Theta}$ library, for example to identify second-order implicit differential equations:

$$\boldsymbol{\Theta}\left([\mathbf{X} \quad \dot{\mathbf{X}} \quad \ddot{\mathbf{X}}]\right). \tag{7.50}$$

The generality of this approach enables the identification of many systems of interest, including those systems with rational function nonlinearities.

### Information Criteria for Model Selection

When performing the sparse regression in the SINDy algorithm, the sparsity-promoting parameter $\lambda$ is a free variable. In practice, different values of $\lambda$ will result in different models with various levels of sparsity, ranging from the trivial model $\dot{x} = 0$ for very large $\lambda$ to the simple least-squares solution for $\lambda = 0$. Thus, by varying $\lambda$, it is possible to sweep out a Pareto front, balancing error versus complexity, as in Fig. 7.9. To identify the most parsimonious model, with low error and a reasonable complexity, it is possible to leverage information criteria for model selection, as described in Mangan et al. [362]. In particular, if we compute the Akaike information criterion (AIC) [6, 7], which penalizes the number of terms in the model, then the most parsimonious model minimizes the AIC. This procedure has been applied to several sparse identification problems, and in every case, the true model was correctly identified [362].

## 7.4 Koopman Operator Theory

Koopman operator theory has recently emerged as an alternative perspective for dynamical systems in terms of the evolution of measurements $g(\mathbf{x})$. In 1931, Bernard O. Koopman demonstrated that it is possible to represent a nonlinear dynamical system in terms of an infinite-dimensional linear operator acting on a Hilbert space of measurement functions of the state of the system. This so-called *Koopman operator* is linear, and its spectral decomposition completely characterizes the behavior of a nonlinear system, analogous to (7.7). However, it is also infinite-dimensional, as there are infinitely many degrees of freedom required to describe the space of all possible measurement functions $g$ of the state. This poses new challenges. Obtaining finite-dimensional, matrix approximations of the Koopman operator is the focus of intense research efforts and holds the promise of enabling globally linear representations of nonlinear dynamical systems. Expressing nonlinear dynamics in a linear framework is appealing because of the wealth of optimal estimation and control techniques available for linear systems (see Chapter 8) and the ability to analytically predict the future state of the system. Obtaining a finite-dimensional approximation of the Koopman operator has been challenging in practice, as it involves identifying a subspace spanned by a subset of eigenfunctions of the Koopman operator.

### Mathematical Formulation of Koopman Theory

The Koopman operator advances measurement functions of the state with the flow of the dynamics. We consider real-valued measurement functions $g : \mathbf{M} \to \mathbb{R}$, which are elements of an infinite-dimensional Hilbert space. The functions $g$ are also commonly known as *observables*, although this may be confused with the unrelated *observability* from control theory. Typically, the Hilbert space is given by the Lebesgue square-integrable functions on $\mathbf{M}$; other choices of a measure space are also valid.

The Koopman operator $\mathcal{K}_t$ is an infinite-dimensional linear operator that acts on measurement functions $g$ as:

$$\mathcal{K}_t g = g \circ \mathbf{F}_t \tag{7.51}$$

where $\circ$ is the composition operator. For a discrete-time system with timestep $\Delta t$, this becomes:

$$\mathcal{K}_{\Delta t} g(\mathbf{x}_k) = g(\mathbf{F}_{\Delta t}(\mathbf{x}_k)) = g(\mathbf{x}_{k+1}). \tag{7.52}$$

In other words, the Koopman operator defines an infinite-dimensional linear dynamical system that advances the observation of the state $g_k = g(\mathbf{x}_k)$ to the next time step:

$$g(\mathbf{x}_{k+1}) = \mathcal{K}_{\Delta t} g(\mathbf{x}_k). \tag{7.53}$$

Note that this is true for *any* observable function $g$ and for any state $\mathbf{x}_k$.

The Koopman operator is linear, a property which is inherited from the linearity of the addition operation in function spaces:

$$\mathcal{K}_t \left(\alpha_1 g_1(\mathbf{x}) + \alpha_2 g_2(\mathbf{x})\right) = \alpha_1 g_1 \left(\mathbf{F}_t(\mathbf{x})\right) + \alpha_2 g_2 \left(\mathbf{F}_t(\mathbf{x})\right) \tag{7.54a}$$

$$= \alpha_1 \mathcal{K}_t g_1(\mathbf{x}) + \alpha_2 \mathcal{K}_t g_2(\mathbf{x}). \tag{7.54b}$$

For sufficiently smooth dynamical systems, it is also possible to define the continuous-time analogue of the Koopman dynamical system in (7.53):

$$\frac{d}{dt}g = \mathcal{K}g. \tag{7.55}$$

The operator $\mathcal{K}$ is the infinitesimal generator of the one-parameter family of transformations $\mathcal{K}_t$ [1]. It is defined by its action on an observable function $g$:

$$\mathcal{K}g = \lim_{t\to 0}\frac{\mathcal{K}_{tg} - g}{t} = \lim_{t\to 0}\frac{g\circ \mathbf{F}_t - g}{t}. \tag{7.56}$$

The linear dynamical systems in (7.55) and (7.53) are analogous to the dynamical systems in (7.3) and (7.4), respectively. It is important to note that the original state $\mathbf{x}$ may be the observable, and the infinite-dimensional operator $\mathcal{K}_t$ will advance this function. However, the simple representation of the observable $g = \mathbf{x}$ in a chosen basis for Hilbert space may become arbitrarily complex once iterated through the dynamics. In other words, finding a representation for $\mathcal{K}\mathbf{x}$ may not be simple or straightforward.

***Koopman Eigenfunctions and Intrinsic Coordinates***

The Koopman operator is linear, which is appealing, but is infinite dimensional, posing issues for representation and computation. Instead of capturing the evolution of all measurement functions in a Hilbert space, applied Koopman analysis attempts to identify key measurement functions that evolve linearly with the flow of the dynamics. Eigenfunctions of the Koopman operator provide just such a set of special measurements that behave linearly in time. In fact, a primary motivation to adopt the Koopman framework is the ability to simplify the dynamics through the eigen-decomposition of the operator.

A discrete-time Koopman eigenfunction $\varphi(\mathbf{x})$ corresponding to eigenvalue $\lambda$ satisfies

$$\varphi(\mathbf{x}_{k+1}) = \mathcal{K}_{\Delta t}\varphi(\mathbf{x}_k) = \lambda\varphi(\mathbf{x}_k). \tag{7.57}$$

In continuous-time, a Koopman eigenfunction $\varphi(\mathbf{x})$ satisfies

$$\frac{\mathrm{d}}{\mathrm{d}t}\varphi(\mathbf{x}) = \mathcal{K}\varphi(\mathbf{x}) = \lambda\varphi(\mathbf{x}). \tag{7.58}$$

Obtaining Koopman eigenfunctions from data or from analytic expressions is a central applied challenge in modern dynamical systems. Discovering these eigenfunctions enables globally linear representations of strongly nonlinear systems.

Applying the chain rule to the time derivative of the Koopman eigenfunction $\varphi(\mathbf{x})$ yields

$$\frac{\mathrm{d}}{\mathrm{d}t}\varphi(\mathbf{x}) = \nabla\varphi(\mathbf{x})\cdot\dot{\mathbf{x}} = \nabla\varphi(\mathbf{x})\cdot\mathbf{f}(\mathbf{x}). \tag{7.59}$$

Combined with (7.58), this results in a partial differential equation (PDE) for the eigenfunction $\varphi(\mathbf{x})$:

$$\nabla\varphi(\mathbf{x})\cdot\mathbf{f}(\mathbf{x}) = \lambda\varphi(\mathbf{x}). \tag{7.60}$$

With this nonlinear PDE, it is possible to approximate the eigenfunctions, either by solving for the Laurent series or with data via regression, both of which are explored below. This formulation assumes that the dynamics are both continuous and differentiable. The discrete-time dynamics in (7.4) are more general, although in many examples the continuous-time dynamics have a simpler representation than the discrete-time map for

long times. For example, the simple Lorenz system has a simple continuous-time representation, yet is generally unrepresentable for even moderately long discrete-time updates.

The key takeaway from (7.57) and (7.58) is that the nonlinear dynamics become completely linear in eigenfunction coordinates, given by $\varphi(\mathbf{x})$. As a simple example, any conserved quantity of a dynamical system is a Koopman eigenfunction corresponding to eigenvalue $\lambda = 0$. This establishes a Koopman extension of the famous Noether's theorem [406], implying that any symmetry in the governing equations gives rise to a new Koopman eigenfunction with eigenvalue $\lambda = 0$. For example, the Hamiltonian energy function is a Koopman eigenfunction for a conservative system. In addition, the constant function $\varphi = 1$ is always a trivial eigenfunction corresponding to $\lambda = 0$ for every dynamical system.

*Eigenvalue lattices* Interestingly, a set of Koopman eigenfunctions may be used to generate more eigenfunctions. In discrete time, we find that the product of two eigenfunctions $\varphi_1(\mathbf{x})$ and $\varphi_2(\mathbf{x})$ is also an eigenfunction

$$\mathcal{K}_t\left(\varphi_1(\mathbf{x})\varphi_2(\mathbf{x})\right) = \varphi_1(\mathbf{F}_t(\mathbf{x}))\varphi_2(\mathbf{F}_t(\mathbf{x})) \tag{7.61a}$$

$$= \lambda_1\lambda_2\varphi_1(\mathbf{x})\varphi_2(\mathbf{x}) \tag{7.61b}$$

corresponding to a new eigenvalue $\lambda_1\lambda_2$ given by the product of the two eigenvalues of $\varphi_1(\mathbf{x})$ and $\varphi_2(\mathbf{x})$.

In continuous time, the relationship becomes:

$$\mathcal{K}\left(\varphi_1\varphi_2\right) = \frac{d}{dt}\left(\varphi_1\varphi_2\right) \tag{7.62a}$$

$$= \dot{\varphi}_1\varphi_2 + \varphi_1\dot{\varphi}_2 \tag{7.62b}$$

$$= \lambda_1\varphi_1\varphi_2 + \lambda_2\varphi_1\varphi_2 \tag{7.62c}$$

$$= (\lambda_1 + \lambda_2)\varphi_1\varphi_2. \tag{7.62d}$$

Interestingly, this means that the set of Koopman eigenfunctions establishes a commutative monoid under point-wise multiplication; a monoid has the structure of a group, except that the elements need not have inverses. Thus, depending on the dynamical system, there may be a finite set of *generator* eigenfunction elements that may be used to construct all other eigenfunctions. The corresponding eigenvalues similarly form a lattice, based on the product $\lambda_1\lambda_2$ or sum $\lambda_1 + \lambda_2$, depending on whether the dynamics are in discrete time or continuous time. For example, given a linear system $\dot{x} = \lambda x$, then $\varphi(x) = x$ is an eigenfunction with eigenvalue $\lambda$. Moreover, $\varphi^\alpha = x^\alpha$ is also an eigenfunction with eigenvalue $\alpha\lambda$ for any $\alpha$.

The continuous time and discrete time lattices are related in a simple way. If the continuous-time eigenvalues are given by $\lambda$, then the corresponding discrete-time eigenvalues are given by $e^{\lambda t}$. Thus, the eigenvalue expressions in (7.61b) and (7.62d) are related as:

$$e^{\lambda_1 t}e^{\lambda_2 t}\varphi_1(\mathbf{x})\varphi_2(\mathbf{x}) = e^{(\lambda_1+\lambda_2)t}\varphi_1(\mathbf{x})\varphi_2(\mathbf{x}). \tag{7.63}$$

As another simple demonstration of the relationship between continuous-time and discrete-time eigenvalues, consider the continuous-time definition in (7.56) applied to an eigenfunction:

$$\lim_{t\to 0}\frac{\mathcal{K}_t\varphi(\mathbf{x}) - \varphi(\mathbf{x})}{t} = \lim_{t\to 0}\frac{e^{\lambda t}\varphi(\mathbf{x}) - \varphi(\mathbf{x})}{t} = \lambda\varphi(\mathbf{x}). \tag{7.64}$$

### Koopman Mode Decomposition and Finite Representations

Until now, we have considered scalar measurements of a system, and we uncovered special *eigen*-measurements that evolve linearly in time. However, we often take multiple measurements of a system. In extreme cases, we may measure the entire state of a high-dimensional spatial system, such as an evolving fluid flow. These measurements may then be arranged in a vector $\mathbf{g}$:

$$\mathbf{g}(\mathbf{x}) = \begin{bmatrix} g_1(\mathbf{x}) \\ g_2(\mathbf{x}) \\ \vdots \\ g_p(\mathbf{x}) \end{bmatrix}. \tag{7.65}$$

Each of the individual measurements may be expanded in terms of the eigenfunctions $\varphi_j(\mathbf{x})$, which provide a basis for Hilbert space:

$$g_i(\mathbf{x}) = \sum_{j=1}^{\infty} v_{ij}\varphi_j(\mathbf{x}). \tag{7.66}$$

Thus, the vector of observables, $\mathbf{g}$, may be similarly expanded:

$$\mathbf{g}(\mathbf{x}) = \begin{bmatrix} g_1(\mathbf{x}) \\ g_2(\mathbf{x}) \\ \vdots \\ g_p(\mathbf{x}) \end{bmatrix} = \sum_{j=1}^{\infty} \varphi_j(\mathbf{x})\mathbf{v}_j, \tag{7.67}$$

where $\mathbf{v}_j$ is the $j$-th *Koopman mode* associated with the eigenfunction $\varphi_j$.

For conservative dynamical systems, such as those governed by Hamiltonian dynamics, the Koopman operator is unitary. Thus, the Koopman eigenfunctions are orthonormal for conservative systems, and it is possible to compute the Koopman modes $\mathbf{v}_j$ directly by projection:

$$\mathbf{v}_j = \begin{bmatrix} \langle \varphi_j, g_1 \rangle \\ \langle \varphi_j, g_2 \rangle \\ \vdots \\ \langle \varphi_j, g_p \rangle \end{bmatrix}, \tag{7.68}$$

where $\langle \cdot, \cdot \rangle$ is the standard inner product of functions in Hilbert space. These modes have a physical interpretation in the case of direct spatial measurements of a system, $\mathbf{g}(\mathbf{x}) = \mathbf{x}$, in which case the modes are coherent *spatial* modes that behave linearly with the same temporal dynamics (i.e., oscillations, possibly with linear growth or decay).

Given the decomposition in (7.67), it is possible to represent the dynamics of the measurements $\mathbf{g}$ as follows:

$$\mathbf{g}(\mathbf{x}_k) = \mathcal{K}_{\Delta t}^k \mathbf{g}(\mathbf{x}_0) = \mathcal{K}_{\Delta t}^k \sum_{j=0}^{\infty} \varphi_j(\mathbf{x}_0)\mathbf{v}_j \tag{7.69a}$$

$$= \sum_{j=0}^{\infty} \mathcal{K}_{\Delta t}^k \varphi_j(\mathbf{x}_0)\mathbf{v}_j \tag{7.69b}$$

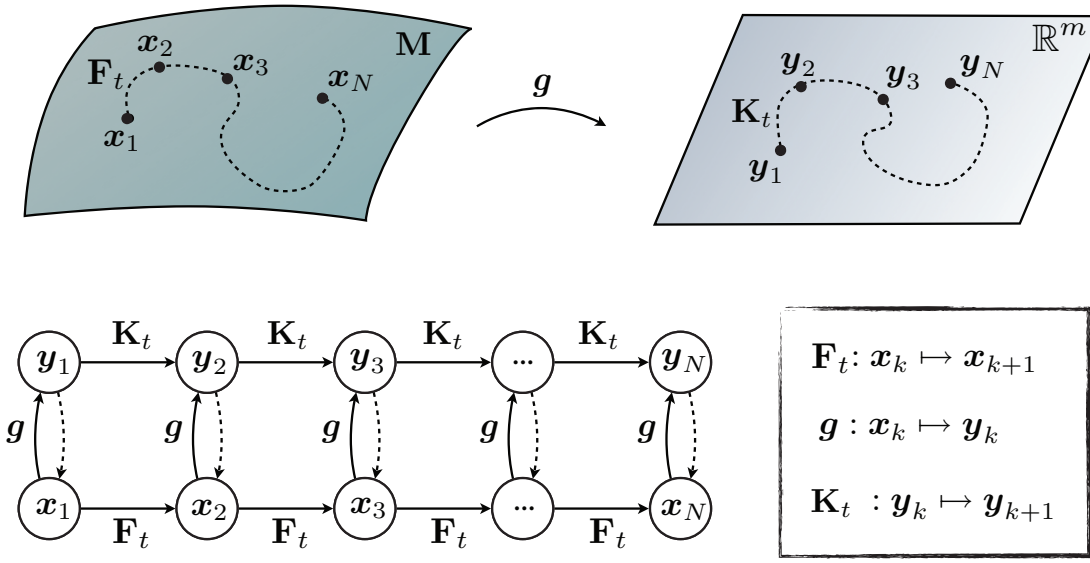


**Figure 7.10** Schematic illustrating the Koopman operator for nonlinear dynamical systems. The dashed lines from $\mathbf{y}_k \rightarrow \mathbf{x}_k$ indicate that we would like to be able to recover the original state.

$$= \sum_{j=0}^{\infty} \lambda_j^k \varphi_j(\mathbf{x}_0)\mathbf{v}_j. \tag{7.69c}$$

This sequence of triples, $\{(\lambda_j, \varphi_j, \mathbf{v}_j)\}_{j=0}^{\infty}$ is known as the *Koopman mode decomposition*, and was introduced by Mezic in 2005 [376]. The Koopman mode decomposition was later connected to data-driven regression via the dynamic mode decomposition [456], which will be discussed in Section 7.2.

### *Invariant Eigenspaces and Finite-Dimensional Models*

Instead of capturing the evolution of all measurement functions in a Hilbert space, applied Koopman analysis approximates the evolution on an invariant subspace spanned by a finite set of measurement functions.

A *Koopman-invariant subspace* is defined as the span of a set of functions $\{g_1, g_2, \cdots, g_p\}$ if all functions $g$ in this subspace

$$g = \alpha_1 g_1 + \alpha_2 g_2 + \cdots + \alpha_p g_p \tag{7.70}$$

remain in this subspace after being acted on by the Koopman operator $\mathcal{K}$:

$$\mathcal{K}g = \beta_1 g_1 + \beta_2 g_2 + \cdots + \beta_p g_p. \tag{7.71}$$

It is possible to obtain a finite-dimensional matrix representation of the Koopman operator by restricting it to an invariant subspace spanned by a finite number of functions $\{g_j\}_{j=0}^{p}$. The matrix representation $\mathbf{K}$ acts on a vector space $\mathbb{R}^p$, with the coordinates given by the values of $g_j(\mathbf{x})$. This induces a finite-dimensional linear system, as in (7.53) and (7.55).

Any finite set of eigenfunctions of the Koopman operator will span an invariant subspace. Discovering these eigenfunction coordinates is, therefore, a central challenge, as

they provide intrinsic coordinates along which the dynamics behave linearly. In practice, it is more likely that we will identify an *approximately* invariant subspace, given by a set of functions $\{g_j\}_{j=0}^{p}$, where each of the functions $g_j$ is well approximated by a finite sum of eigenfunctions: $g_j \approx \sum_{k=0}^{p} \alpha_k \varphi_k$.

## Examples of Koopman Embeddings

### *Nonlinear System with Single Fixed Point and a Slow Manifold*

Here, we consider an example system with a single fixed point, given by:

$$\dot{x}_1 = \mu x_1 \tag{7.72a}$$

$$\dot{x}_2 = \lambda(x_2 - x_1^2). \tag{7.72b}$$

For $\lambda < \mu < 0$, the system exhibits a slow attracting manifold given by $x_2 = x_1^2$. It is possible to augment the state $\mathbf{x}$ with the nonlinear measurement $g = x_1^2$, to define a three-dimensional Koopman invariant subspace. In these coordinates, the dynamics become linear:

$$\frac{d}{dt}\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} \mu & 0 & 0 \\ 0 & \lambda & -\lambda \\ 0 & 0 & 2\mu \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \quad \text{for} \quad \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_1^2 \end{bmatrix}. \tag{7.73a}$$

The full three-dimensional Koopman observable vector space is visualized in Fig. 7.11. Trajectories that start on the invariant manifold $y_3 = y_1^2$, visualized by the blue surface, are constrained to stay on this manifold. There is a *slow* subspace, spanned by the eigenvectors corresponding to the slow eigenvalues $\mu$ and $2\mu$; this subspace is visualized by the green surface. Finally, there is the original asymptotically attracting manifold of the original system, $y_2 = y_1^2$, which is visualized as the red surface. The blue and red parabolic surfaces always intersect in a parabola that is inclined at a 45° angle in the $y_2$-$y_3$ direction. The green surface approaches this 45° inclination as the ratio of fast to slow dynamics become increasingly large. In the full three-dimensional Koopman observable space, the dynamics produce a single stable node, with trajectories rapidly attracting onto the green subspace and then slowly approaching the fixed point.

*Intrinsic coordinates defined by eigenfunctions of the Koopman operator* The left eigenvectors of the Koopman operator yield Koopman eigenfunctions (i.e., eigenobservables). The Koopman eigenfunctions of (7.73a) corresponding to eigenvalues $\mu$ and $\lambda$ are:

$$\varphi_\mu = x_1, \quad \text{and} \quad \varphi_\lambda = x_2 - b x_1^2 \quad \text{with} \quad b = \frac{\lambda}{\lambda - 2\mu}. \tag{7.74}$$

The constant $b$ in $\varphi_\lambda$ captures the fact that for a finite ratio $\lambda/\mu$, the dynamics only shadow the asymptotically attracting slow manifold $x_2 = x_1^2$, but in fact follow neighboring parabolic trajectories. This is illustrated more clearly by the various surfaces in Fig. 7.11 for different ratios $\lambda/\mu$.

In this way, a set of intrinsic coordinates may be determined from the observable functions defined by the left eigenvectors of the Koopman operator on an invariant subspace. Explicitly,

$$\varphi_\alpha(\mathbf{x}) = \boldsymbol{\xi}_\alpha \mathbf{y}(\mathbf{x}), \quad \text{where} \quad \boldsymbol{\xi}_\alpha \mathbf{K} = \alpha \boldsymbol{\xi}_\alpha. \tag{7.75}$$

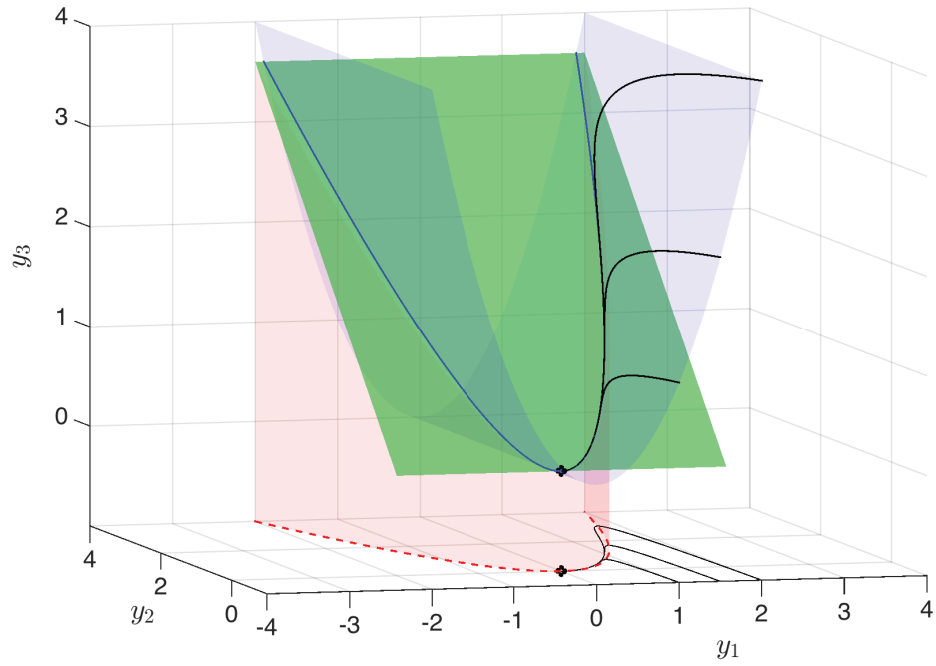


**Figure 7.11** Visualization of three-dimensional linear Koopman system from (7.73a) along with projection of dynamics onto the $x_1$-$x_2$ plane. The attracting slow manifold is shown in red, the constraint $y_3 = y_1^2$ is shown in blue, and the slow unstable subspace of (7.73a) is shown in green. Black trajectories of the linear Koopman system in $\mathbf{y}$ project onto trajectories of the full nonlinear system in $\mathbf{x}$ in the $y_1$-$y_2$ plane. Here, $\mu = -0.05$ and $\lambda = 1$. *Reproduced from Brunton et al. [92].*

These eigen-observables define observable subspaces that remain invariant under the Koopman operator, even after coordinate transformations. As such, they may be regarded as intrinsic coordinates [556] on the Koopman-invariant subspace.

### *Example of Intractable Representation*

Consider the logistic map, given by:

$$x_{k+1} = \beta x_k (1 - x_k). \tag{7.76}$$

Let our observable subspace include $x$ and $x^2$:

$$\mathbf{y}_k = \begin{bmatrix} x \\ x^2 \end{bmatrix}_k \triangleq \begin{bmatrix} x_k \\ x_k^2 \end{bmatrix}. \tag{7.77}$$

Writing out the Koopman operator, the first row equation is simple:

$$\mathbf{y}_{k+1} = \begin{bmatrix} x \\ x^2 \end{bmatrix}_{k+1} = \begin{bmatrix} \beta & -\beta \\ ? & ? \end{bmatrix} \begin{bmatrix} x \\ x^2 \end{bmatrix}_k, \tag{7.78}$$

but the second row is not obvious. To find this expression, expand $x_{k+1}^2$:

$$x_{k+1}^2 = (\beta x_k (1 - x_k))^2 = \beta^2 \left( x_k^2 - 2x_k^3 + x_k^4 \right). \tag{7.79}$$

Thus, cubic and quartic polynomial terms are required to advance $x^2$. Similarly, these terms need polynomials up to sixth and eighth order, respectively, and so on, ad infinitum:

$$\begin{bmatrix} x \\ x^2 \\ x^3 \\ x^4 \\ x^5 \\ \vdots \end{bmatrix}_{k+1} = \begin{bmatrix} x & x^2 & x^3 & x^4 & x^5 & x^6 & x^7 & x^8 & x^9 & x^{10} & \\ \beta & -\beta & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & \beta^2 & -2\beta^2 & r^2 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & \beta^3 & -3\beta^3 & 3\beta^3 & \beta^3 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & \beta^4 & -4\beta^4 & 6\beta^4 & -4\beta^4 & \beta^4 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & \beta^5 & -5\beta^5 & 10\beta^5 & -10\beta^5 & 5\beta^5 & -\beta^5 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} x \\ x^2 \\ x^3 \\ x^4 \\ x^5 \\ \vdots \end{bmatrix}_k .$$

It is interesting to note that the rows of this equation are related to the rows of Pascal's triangle, with the $n$-th row scaled by $r^n$, and with the omission of the first row:

$$\left[x^0\right]_{k+1} = \left[0\right]\left[x^0\right]_k . \tag{7.80}$$

The above representation of the Koopman operator in a polynomial basis is somewhat troubling. Not only is there no closure, but the determinant of any finite-rank truncation is very large for $\beta > 1$. This illustrates a pitfall associated with naive representation of the infinite dimensional Koopman operator for a simple chaotic system. Truncating the system, or performing a least squares fit on an augmented observable vector (i.e., DMD on a nonlinear measurement; see Section 7.5) yields poor results, with the truncated system only agreeing with the true dynamics for a small handful of iterations, as the complexity of the representation grows quickly:

$$\begin{matrix} 1 \\ x \\ x^2 \\ x^3 \\ x^4 \\ x^5 \\ x^6 \\ x^7 \\ x^8 \\ \vdots \end{matrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \end{bmatrix} \overset{\mathcal{K}}{\Longrightarrow} \begin{bmatrix} 0 \\ \beta \\ -\beta \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \end{bmatrix} \overset{\mathcal{K}}{\Longrightarrow} \begin{bmatrix} 0 \\ \beta^2 \\ -\beta^2 - \beta^3 \\ 2\beta^3 \\ -\beta^3 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \end{bmatrix} \overset{\mathcal{K}}{\Longrightarrow} \begin{bmatrix} 0 \\ \beta^3 \\ -\beta^3 - \beta^4 - \beta^5 \\ 2\beta^4 + 2\beta^5 + 2\beta^6 \\ -\beta^4 - \beta^5 - 6\beta^6 - \beta^7 \\ 6\beta^6 + 4\beta^7 \\ -2\beta^6 - 6\beta^7 \\ 4\beta^7 \\ -\beta^7 \\ \vdots \end{bmatrix} . \tag{7.81}$$

### Analytic Series Expansions for Eigenfunctions

Given the dynamics in (7.1), it is possible to solve the PDE in (7.60) using standard techniques, such as recursively solving for the terms in a Taylor or Laurent series. A number of simple examples are explored below.

#### *Linear Dynamics*

Consider the simple linear dynamics

$$\frac{d}{dt}x = x. \tag{7.82}$$

Assuming a Taylor series expansion for $\varphi(x)$:

$$\varphi(x) = c_0 + c_1 x + c_2 x^2 + c_3 x^3 + \cdots$$

then the gradient and directional derivatives are given by:

$$\begin{aligned}\nabla\varphi &= c_1 + 2c_2 x + 3c_3 x^2 + 4c_4 x^3 + \cdots \\ \nabla\varphi \cdot f &= c_1 x + 2c_2 x^2 + 3c_3 x^3 + 4c_4 x^4 + \cdots\end{aligned}$$

Solving for terms in the Koopman eigenfunction PDE (7.60), we see that $c_0 = 0$ must hold. For any positive integer $\lambda$ in (7.60), only one of the coefficients may be nonzero. Specifically, for $\lambda = k \in \mathbb{Z}^+$, then $\varphi(x) = cx^k$ is an eigenfunction for any constant $c$. For instance, if $\lambda = 1$, then $\varphi(x) = x$.

### *Quadratic Nonlinear Dynamics*

Consider a nonlinear dynamical system

$$\frac{d}{dt} = x^2. \tag{7.83}$$

There is no Taylor series that satisfies (7.60), except the trivial solution $\varphi = 0$ for $\lambda = 0$. Instead, we assume a Laurent series:

$$\begin{aligned}\varphi(x) = \cdots + c_{-3}x^{-3} + c_{-2}x^{-2} + c_{-1}x^{-1} + c_0 \\ + c_1 x + c_2 x^2 + c_3 x^3 + \cdots .\end{aligned}$$

The gradient and directional derivatives are given by:

$$\begin{aligned}\nabla\varphi = \cdots - 3c_{-3}x^{-4} - 2c_{-2}x^{-3} - c_{-1}x^{-2} + c_1 + 2c_2 x \\ + 3c_3 x^2 + 4c_4 x^3 + \cdots \\ \nabla\varphi \cdot f = \cdots - 3c_{-3}x^{-2} - 2c_{-2}x^{-1} - c_{-1} + c_1 x^2 + 2c_2 x^3 \\ + 3c_3 x^4 + 4c_4 x^5 + \cdots .\end{aligned}$$

Solving for the coefficients of the Laurent series that satisfy (7.60), we find that all coefficients with positive index are zero, i.e. $c_k = 0$ for all $k \geq 1$. However, the nonpositive index coefficients are given by the recursion $\lambda c_{k+1} = k c_k$, for negative $k \leq -1$. Thus, the Laurent series is

$$\varphi(x) = c_0 \left(1 - \lambda x^{-1} + \frac{\lambda^2}{2}x^{-2} - \frac{\lambda^3}{3}x^{-3} + \cdots\right) = c_0 e^{-\lambda/x}.$$

This holds for all values of $\lambda \in \mathbb{C}$. There are also other Koopman eigenfunctions that can be identified from the Laurent series.

### *Polynomial Nonlinear Dynamics*

For a more general nonlinear dynamical system

$$\frac{d}{dt} = ax^n, \tag{7.84}$$

$\varphi(x) = e^{\frac{\lambda}{(1-n)a}x^{1-n}}$ is an eigenfunction for all $\lambda \in \mathbb{C}$.

As mentioned above, it is also possible to generate new eigenfunctions by taking powers of these primitive eigenfunctions; the resulting eigenvalues generate a *lattice* in the complex plane.

### History and Recent Developments

The original analysis of Koopman in 1931 was introduced to describe the evolution of measurements of Hamiltonian systems [300], and this theory was generalized by Koopman and von Neumann to systems with continuous eigenvalue spectrum in 1932 [301]. In the case of Hamiltonian flows, the Koopman operator $\mathcal{K}_t$ is unitary, and forms a one-parameter family of unitary transformations in Hilbert space. Unitary operators should be familiar by now, as the discrete Fourier transform (DFT) and the singular value decomposition (SVD) both provide unitary coordinate transformations. Unitarity implies that the inner product of any two observable functions remains unchanged through action of the Koopman operator, which is intuitively related to the phase-space volume preserving property of Hamiltonian systems. In the original paper [300], Koopman drew connections between the Koopman eigenvalue spectrum and conserved quantities, integrability, and ergodicity. Interestingly, Koopman's 1931 paper was central in the celebrated proofs of the ergodic theorem by Birkhoff and von Neumann [62, 399, 61, 389].

Koopman analysis has recently gained renewed interest with the pioneering work of Mezic and collaborators [379, 376, 102, 104, 103, 377, 322]. The Koopman operator is also known as the composition operator, which is formally the pull-back operator on the space of scalar observable functions [1], and it is the dual, or left-adjoint, of the Perron-Frobenius operator, or transfer operator, which is the push-forward operator on the space of probability density functions. When a polynomial basis is chosen to represent the Koopman operator, then it is closely related to Carleman linearization [121, 122, 123], which has been used extensively in nonlinear control [500, 305, 38, 509]. Koopman analysis is also connected to the resolvent operator theory from fluid dynamics [487].

Recently, it has been shown that the operator theoretic framework complements the traditional geometric and probabilistic perspectives. For example, level sets of Koopman eigenfunctions form invariant partitions of the state-space of a dynamical system [103]; in particular, eigenfunctions of the Koopman operator may be used to analyze the ergodic partition [380, 102]. Koopman analysis has also been recently shown to generalize the Hartman-Grobman theorem to the entire basin of attraction of a stable or unstable equilibrium point or periodic orbit [322].

At the time of this writing, representing Koopman eigenfunctions for general dynamical systems remains a central unsolved challenge. Significant research efforts are focused on developing data-driven techniques to identify Koopman eigenfunctions and use these for control, which will be discussed in the following sections and chapters. Recently, new work has emerged that attempts to leverage the power of deep learning to discover and represent eigenfunctions from data [550, 368, 513, 564, 412, 349].

## 7.5 Data-Driven Koopman Analysis

Obtaining linear representations for strongly nonlinear systems has the potential to revolutionize our ability to predict and control these systems. The linearization of dynamics near

fixed points or periodic orbits has long been employed for *local* linear representation of the dynamics [252]. The Koopman operator is appealing because it provides a *global* linear representation, valid far away from fixed points and periodic orbits. However, previous attempts to obtain finite-dimensional approximations of the Koopman operator have had limited success. Dynamic mode decomposition [472, 456, 317] seeks to approximate the Koopman operator with a best-fit linear model advancing spatial measurements from one time to the next, although these linear measurements are not rich enough for many nonlinear systems. Augmenting DMD with nonlinear measurements may enrich the model, but there is no guarantee that the resulting models will be closed under the Koopman operator [92]. Here, we describe several approaches for identifying Koopman embeddings and eigenfunctions from data. These methods include the extended dynamic mode decomposition [556], extensions based on SINDy [276], and the use of delay coordinates [91].

### Extended DMD

The extended DMD algorithm [556] is essentially the same as standard DMD [535], except that instead of performing regression on direct measurements of the state, regression is performed on an augmented vector containing nonlinear measurements of the state. As discussed earlier, eDMD is equivalent to the variational approach of conformation dynamics [405, 407, 408], which was developed in 2013 by Noé and Nüske.

Here, we will modify the notation slightly to conform to related methods. In eDMD, an augmented state is constructed:

$$\mathbf{y} = \boldsymbol{\Theta}^T(\mathbf{x}) = \begin{bmatrix} \theta_1(\mathbf{x}) \\ \theta_2(\mathbf{x}) \\ \vdots \\ \theta_p(\mathbf{x}) \end{bmatrix}. \tag{7.85}$$

$\boldsymbol{\Theta}$ may contain the original state $\mathbf{x}$ as well as nonlinear measurements, so often $p \gg n$. Next, two data matrices are constructed, as in DMD:

$$\mathbf{Y} = \begin{bmatrix} | & | & & | \\ \mathbf{y}_1 & \mathbf{y}_2 & \cdots & \mathbf{y}_m \\ | & | & & | \end{bmatrix}, \qquad \mathbf{Y}' = \begin{bmatrix} | & | & & | \\ \mathbf{y}_2 & \mathbf{y}_3 & \cdots & \mathbf{y}_{m+1} \\ | & | & & | \end{bmatrix}. \tag{7.86a}$$

Finally, a best-fit linear operator $\mathbf{A_Y}$ is constructed that maps $\mathbf{Y}$ into $\mathbf{Y}'$:

$$\mathbf{A_Y} = \underset{\mathbf{A_Y}}{\operatorname{argmin}} \|\mathbf{Y}' - \mathbf{A_Y}\mathbf{Y}\| = \mathbf{Y}'\mathbf{Y}^{\dagger}. \tag{7.87}$$

This regression may be written in terms of the data matrices $\boldsymbol{\Theta}(\mathbf{X})$ and $\boldsymbol{\Theta}(\mathbf{X}')$:

$$\mathbf{A_Y} = \underset{\mathbf{A_Y}}{\operatorname{argmin}} \|\boldsymbol{\Theta}^T(\mathbf{X}') - \mathbf{A_Y}\boldsymbol{\Theta}^T(\mathbf{X})\| = \boldsymbol{\Theta}^T(\mathbf{X}') \left(\boldsymbol{\Theta}^T(\mathbf{X})\right)^{\dagger}. \tag{7.88}$$

Because the augmented vector $\mathbf{y}$ may be significantly larger than the state $\mathbf{x}$, kernel methods are often employed to compute this regression [557]. In principle, the enriched library $\boldsymbol{\Theta}$ provides a larger basis in which to approximate the Koopman operator. It has been shown recently that in the limit of infinite snapshots, the extended DMD operator converges to the Koopman operator projected onto the subspace spanned by $\boldsymbol{\Theta}$ [303]. However, if $\boldsymbol{\Theta}$ does not span a Koopman invariant subspace, then the projected operator may not have any

resemblance to the original Koopman operator, as all of the eigenvalues and eigenvectors may be different. In fact, it was shown that the extended DMD operator will have spurious eigenvalues and eigenvectors unless it is represented in terms of a Koopman invariant subspace [92]. Therefore, it is essential to use validation and cross-validation techniques to ensure that eDMD models are not overfit, as discussed below. For example, it was shown that eDMD cannot contain the original state $\mathbf{x}$ as a measurement and represent a system that has multiple fixed points, periodic orbits, or other attractors, because these systems cannot be topologically conjugate to a finite-dimensional linear system [92].

### Approximating Koopman Eigenfunctions from Data

In discrete-time, a Koopman eigenfunction$\varphi(\mathbf{x})$ evaluated at a number of data points in $\mathbf{X}$ will satisfy:

$$\begin{bmatrix} \lambda\varphi(\mathbf{x}_1) \\ \lambda\varphi(\mathbf{x}_2) \\ \vdots \\ \lambda\varphi(\mathbf{x}_m) \end{bmatrix} = \begin{bmatrix} \varphi(\mathbf{x}_2) \\ \varphi(\mathbf{x}_3) \\ \vdots \\ \varphi(\mathbf{x}_{m+1}) \end{bmatrix}. \tag{7.89}$$

It is possible to approximate this eigenfunction as an expansion in terms of a set of candidate functions,

$$\boldsymbol{\Theta}(\mathbf{x}) = \begin{bmatrix} \theta_1(\mathbf{x}) & \theta_2(\mathbf{x}) & \cdots & \theta_p(\mathbf{x}) \end{bmatrix}. \tag{7.90}$$

The Koopman eigenfunctionmay be approximated in this basis as:

$$\varphi(\mathbf{x}) \approx \sum_{k=1}^{p} \theta_k(\mathbf{x})\xi_k = \boldsymbol{\Theta}(\mathbf{x})\boldsymbol{\xi}. \tag{7.91}$$

Writing (7.89) in terms of this expansion yields the matrix system:

$$\left(\lambda\boldsymbol{\Theta}(\mathbf{X}) - \boldsymbol{\Theta}(\mathbf{X}')\right)\boldsymbol{\xi} = \mathbf{0}. \tag{7.92}$$

If we seek the best *least-squares* fit to (7.92), this reduces to the extended DMD [557, 556] formulation:

$$\lambda\boldsymbol{\xi} = \boldsymbol{\Theta}(\mathbf{X})^{\dagger}\boldsymbol{\Theta}(\mathbf{X}')\boldsymbol{\xi}. \tag{7.93}$$

Note that (7.93) is the transpose of (7.88), so that left eigenvectors become right eigenvectors. Thus, eigenvectors $\boldsymbol{\xi}$ of $\boldsymbol{\Theta}^{\dagger}\boldsymbol{\Theta}'$ yield the coefficients of the eigenfunction $\varphi(\mathbf{x})$ represented in the basis $\boldsymbol{\Theta}(\mathbf{x})$. It is absolutely essential to then confirm that predicted eigenfunctions actually behave linearly on trajectories, by comparing them with the predicted dynamics $\varphi_{k+1} = \lambda\varphi_k$, because the regression above will result in spurious eigenvalues and eigenvectors unless the basis elements $\theta_j$ span a Koopman invariant subspace [92].

#### *Sparse Identification of Eigenfunctions*

It is possible to leverage the SINDy regression [95] to identify Koopman eigenfunctions corresponding to a particular eigenvalue $\lambda$, selecting only the few active terms in the library $\boldsymbol{\Theta}(\mathbf{x})$ to avoid overfitting. Given the data matrices, $\mathbf{X}$ and $\dot{\mathbf{X}}$ from above it is possible to construct the library of basis functions $\boldsymbol{\Theta}(\mathbf{X})$ as well as a library of directional derivatives,

representing the possible terms in $\nabla\varphi(\mathbf{x}) \cdot \mathbf{f}(\mathbf{x})$ from (7.60):

$$\boldsymbol{\Gamma}(\mathbf{x}, \dot{\mathbf{x}}) = \begin{bmatrix} \nabla\theta_1(\mathbf{x}) \cdot \dot{\mathbf{x}} & \nabla\theta_2(\mathbf{x}) \cdot \dot{\mathbf{x}} & \cdots & \nabla\theta_p(\mathbf{x}) \cdot \dot{\mathbf{x}} \end{bmatrix}. \tag{7.94}$$

It is then possible to construct $\boldsymbol{\Gamma}$ from data:

$$\boldsymbol{\Gamma}(\mathbf{X}, \dot{\mathbf{X}}) = \begin{bmatrix} \nabla\theta_1(\mathbf{x}_1) \cdot \dot{\mathbf{x}}_1 & \nabla\theta_2(\mathbf{x}_1) \cdot \dot{\mathbf{x}}_1 & \cdots & \nabla\theta_p(\mathbf{x}_1) \cdot \dot{\mathbf{x}}_1 \\ \nabla\theta_1(\mathbf{x}_2) \cdot \dot{\mathbf{x}}_2 & \nabla\theta_2(\mathbf{x}_2) \cdot \dot{\mathbf{x}}_2 & \cdots & \nabla\theta_p(\mathbf{x}_2) \cdot \dot{\mathbf{x}}_2 \\ \vdots & \vdots & \ddots & \vdots \\ \nabla\theta_1(\mathbf{x}_m) \cdot \dot{\mathbf{x}}_m & \nabla\theta_2(\mathbf{x}_m) \cdot \dot{\mathbf{x}}_m & \cdots & \nabla\theta_p(\mathbf{x}_m) \cdot \dot{\mathbf{x}}_m \end{bmatrix}.$$

For a given eigenvalue $\lambda$, the Koopman PDE in (7.60) may be evaluated on data:

$$\left(\lambda\boldsymbol{\Theta}(\mathbf{X}) - \boldsymbol{\Gamma}(\mathbf{X}, \dot{\mathbf{X}})\right)\boldsymbol{\xi} = \mathbf{0}. \tag{7.95}$$

The formulation in (7.95) is implicit, so that $\boldsymbol{\xi}$ will be in the null-space of $\lambda\boldsymbol{\Theta}(\mathbf{X}) - \boldsymbol{\Gamma}(\mathbf{X}, \dot{\mathbf{X}})$. The right null-space of (7.95) for a given $\lambda$ is spanned by the right singular vectors of $\lambda\boldsymbol{\Theta}(\mathbf{X}) - \boldsymbol{\Gamma}(\mathbf{X}, \dot{\mathbf{X}}) = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^*$ (i.e., columns of $\mathbf{V}$) corresponding to zero-valued singular values. It may be possible to identify the few active terms in an eigenfunction by finding the sparsest vector in the null-space [440], as in the implicit-SINDy algorithm [361] described in Section 7.3. In this formulation, the eigenvalues $\lambda$ are not known *a priori*, and must be learned with the approximate eigenfunction. Koopman eigenfuntions and eigenvalues can also be determined as the solution to the eigenvalue problem $\mathbf{A}_\mathbf{Y}\boldsymbol{\xi}_\alpha = \lambda_\alpha\boldsymbol{\xi}_\alpha$, where $\mathbf{A}_\mathbf{Y} = \boldsymbol{\Theta}^\dagger\boldsymbol{\Gamma}$ is obtained via least-squares regression, as in the continuous-time version of eDMD. While many eigenfunctions are spurious, those corresponding to lightly damped eigenvalues can be well approximated.

From a practical standpoint, data in $\mathbf{X}$ does not need to be sampled from full trajectories, but can be obtained using more sophisticated strategies such as latin hypercube sampling or sampling from a distribution over the phase space. Moreover, reproducing kernel Hilbert spaces (RKHS) can be employed to describe $\varphi(\mathbf{x})$ *locally* in patches of state space.

***Example: Duffing System (*Kaiser et al [276]*)***

We demonstrate the sparse identification of Koopman eigenfunctions on the undamped Duffing oscillator:

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ x_1 - x_1^3 \end{bmatrix}$$

where $x_1$ is the position and $x_2$ is the velocity of a particle in a double well potential with equilibria $(0, 0)$ and $(\pm 1, 0)$. This system is conservative, with Hamiltonian $\mathcal{H} = \frac{1}{2}x_2^2 - \frac{1}{2}x_1^2 + \frac{1}{4}x_1^4$. The Hamiltonian, and in general any conserved quantity, is a Koopman eigenfunction with zero eigenvalue.

For the eigenvalue $\lambda = 0$, (7.95) becomes $-\boldsymbol{\Gamma}(\mathbf{X}, \dot{\mathbf{X}})\boldsymbol{\xi} = \mathbf{0}$, and hence a sparse $\boldsymbol{\xi}$ is sought in the null-space of $-\boldsymbol{\Gamma}(\mathbf{X}, \dot{\mathbf{X}})$. A library of candidate functions is constructed from data, employing polynomials up to fourth order:

$$\boldsymbol{\Theta}(\mathbf{X}) = \begin{bmatrix} | & | & | & | & & | \\ x_1(t) & x_2(t) & x_1^2(t) & x_1(t)x_2(t) & \cdots & x_2^4(t) \\ | & | & | & | & & | \end{bmatrix}$$

and

$$\boldsymbol{\Gamma}(\mathbf{X}, \dot{\mathbf{X}}) = \begin{bmatrix} | & | & | & | & & | \\ \dot{x}_1(t) & \dot{x}_2(t) & 2x_1(t)\dot{x}_1(t) & x_2(t)\dot{x}_1(t) + x_1(t) + \dot{x}_2(t) & \cdots & 4x_2(t)^3\dot{x}_2(t) \\ | & | & | & | & & | \end{bmatrix}.$$

A sparse vector of coefficients $\boldsymbol{\xi}$ may be identified, with the few nonzero entries determining the active terms in the Koopman eigenfunction. The identified Koopman eigenfunctionassociated with $\lambda = 0$ is

$$\varphi(\mathbf{x}) = -2/3x_1^2 + 2/3x_2^2 + 1/3x_1^4. \tag{7.96}$$

This eigenfunction matches the Hamiltonian perfectly up to a constant scaling.

### Data-Driven Koopman and Delay Coordinates

Instead of advancing instantaneous linear or nonlinear measurements of the state of a system directly, as in DMD, it may be possible to obtain intrinsic measurement coordinates for Koopman based on time-delayed measurements of the system [506, 91, 18, 144]. This perspective is data-driven, relying on the wealth of information from previous measurements to inform the future. Unlike a linear or weakly nonlinear system, where trajectories may get trapped at fixed points or on periodic orbits, chaotic dynamics are particularly well-suited to this analysis: trajectories evolve to densely fill an attractor, so more data provides more information. The use of delay coordinates may be especially important for systems with long-term memory effects, where the Koopman approach has recently been shown to provide a successful analysis tool [508]. Interestingly, a connection between the Koopman operator and the Takens embedding was explored as early as 2004 [379], where a stochastic Koopman operator is defined and a statistical Takens theorem is proven.

The time-delay measurement scheme is shown schematically in Fig. 7.12, as illustrated on the Lorenz system for a single time-series measurement of the first variable, $x(t)$. The conditions of the Takens embedding theorem are satisfied [515], so it is possible to obtain a diffeomorphism between a delay embedded attractor and the attractor in the original coordinates. We then obtain eigen-time-delay coordinates from a time-series of a single measurement $x(t)$ by taking the SVD of the Hankel matrix $\mathbf{H}$:

$$\mathbf{H} = \begin{bmatrix} x(t_1) & x(t_2) & \cdots & x(t_{m_c}) \\ x(t_2) & x(t_3) & \cdots & x(t_{m_c+1}) \\ \vdots & \vdots & \ddots & \vdots \\ x(t_{m_o}) & x(t_{m_o+1}) & \cdots & x(t_m) \end{bmatrix} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^*. \tag{7.97}$$

The columns of $\mathbf{U}$ and $\mathbf{V}$ from the SVD are arranged hierarchically by their ability to model the columns and rows of $\mathbf{H}$, respectively. Often, $\mathbf{H}$ may admit a low-rank approximation by the first $r$ columns of $\mathbf{U}$ and $\mathbf{V}$. Note that the Hankel matrix in (7.97) is the basis of the eigensystem realization algorithm [272] in linear system identification (see Section 9.3) and singular spectrum analysis (SSA) [88] in climate time-series analysis.

The low-rank approximation to (7.97) provides a *data-driven* measurement system that is approximately invariant to the Koopman operator for states on the attractor. By definition, the dynamics map the attractor into itself, making it *invariant* to the flow. In other words, the columns of $\mathbf{U}$ form a Koopman invariant subspace. We may re-write (7.97) with the

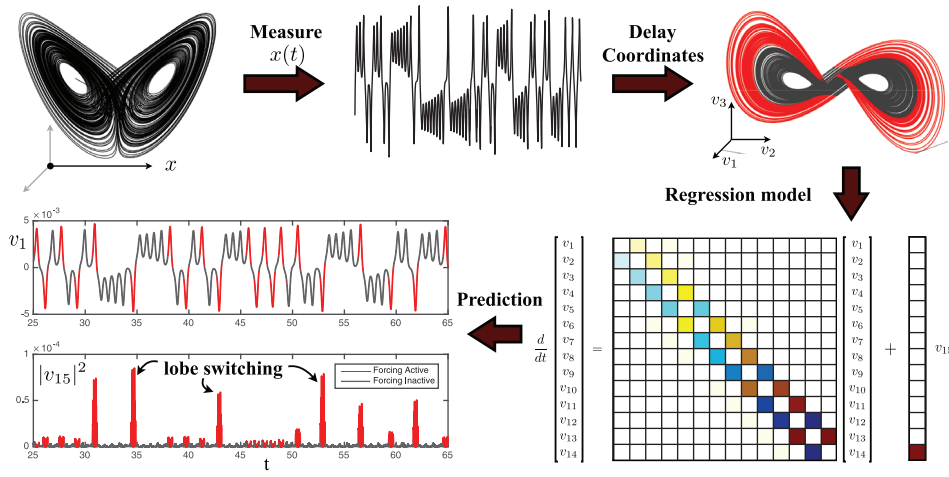


**Figure 7.12** Decomposition of chaos into a linear system with forcing. A time series $x(t)$ is stacked into a Hankel matrix $\mathbf{H}$. The SVD of $\mathbf{H}$ yields a hierarchy of *eigen* time series that produce a delay-embedded attractor. A best-fit linear regression model is obtained on the delay coordinates $\mathbf{v}$; the linear fit for the first $r-1$ variables is excellent, but the last coordinate $v_r$ is not well-modeled as linear. Instead, $v_r$ is an input that forces the first $r-1$ variables. Rare forcing events correspond to lobe switching in the chaotic dynamics. This architecture is called the Hankel alternative view of Koopman (HAVOK) analysis, from [91]. *Figure modified from Brunton et al. [91].*

Koopman operator $\mathcal{K} \triangleq \mathcal{K}_{\Delta t}$:

$$\mathbf{H} = \begin{bmatrix} x(t_1) & \mathcal{K}x(t_1) & \cdots & \mathcal{K}^{m_c-1}x(t_1) \\ \mathcal{K}x(t_1) & \mathcal{K}^2x(t_1) & \cdots & \mathcal{K}^{m_c}x(t_1) \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{K}^{m_o-1}x(t_1) & \mathcal{K}^{m_o}x(t_1) & \cdots & \mathcal{K}^{m-1}x(t_1) \end{bmatrix}. \tag{7.98}$$

The columns of (7.97) are well-approximated by the first $r$ columns of $\mathbf{U}$. The first $r$ columns of $\mathbf{V}$ provide a time series of the magnitude of each of the columns of $\mathbf{U}\Sigma$ in the data. By plotting the first three columns of $\mathbf{V}$, we obtain an embedded attractor for the Lorenz system (See Fig. 7.12).

The connection between eigen-time-delay coordinates from (7.97) and the Koopman operator motivates a linear regression model on the variables in $\mathbf{V}$. Even with an approximately Koopman-invariant measurement system, there remain challenges to identifying a linear model for a chaotic system. A linear model, however detailed, cannot capture multiple fixed points or the unpredictable behavior characteristic of chaos with a positive Lyapunov exponent [92]. Instead of constructing a closed linear model for the first $r$ variables in $\mathbf{V}$, we build a linear model on the first $r-1$ variables and recast the last variable, $v_r$, as a forcing term:

$$\frac{d}{dt}\mathbf{v}(t) = \mathbf{A}\mathbf{v}(t) + \mathbf{B}v_r(t), \tag{7.99}$$

where $\mathbf{v} = \begin{bmatrix} v_1 & v_2 & \cdots & v_{r-1} \end{bmatrix}^T$ is a vector of the first $r-1$ eigen-time-delay coordinates. Other work has investigated the splitting of dynamics into deterministic linear, and chaotic stochastic dynamics [376].

In all of the examples explored in [91], the linear model on the first $r-1$ terms is accurate, while no linear model represents $v_r$. Instead, $v_r$ is an input forcing to the linear dynamics in (7.99), which approximates the nonlinear dynamics. The statistics of $v_r(t)$ are non-Gaussian, with long tails correspond to rare-event forcing that drives lobe switching in the Lorenz system; this is related to rare-event forcing distributions observed and modeled by others [355, 461, 356]. The forced linear system in (7.99) was discovered after applying the SINDy algorithm [95] to delay coordinates of the Lorenz system. Continuing to develop Koopman on delay coordinates has significant promise in the context of closed-loop feedback control, where it may be possible to manipulate the behavior of a chaotic system by treating $v_r$ as a disturbance.

In addition, the use of delay coordinates as intrinsic measurements for Koopman analysis suggests that Koopman theory may also be used to improve spatially distributed sensor technologies. A spatial array of sensors, for example the $\mathcal{O}(100)$ strain sensors on the wings of flying insects, may use phase delay coordinates to provide nearly optimal embeddings to detect and control convective structures (e.g., stall from a gust, leading edge vortex formation and convection, etc.).

### *HAVOK Code for Lorenz System*

Below is the code to generate a HAVOK model for the same Lorenz system data generated in Code 7.2. Here we use $\Delta t = 0.01$, $m_o = 10$, and $r = 10$, although the results would be more accurate for $\Delta t = 0.001$, $m_o = 100$, and $r = 15$.

**Code 7.3** HAVOK code for Lorenz data generated in Section 7.1.

```
%% EIGEN-TIME DELAY COORDINATES
stackmax = 10;    %  Number of shift-stacked rows
r=10;             %  Rank of HAVOK Model
H = zeros(stackmax,size(x,1)-stackmax);
for k=1:stackmax
    H(k,:) = x(k:end-stackmax-1+k,1);
end
[U,S,V] = svd(H,'econ'); % Eigen delay coordinates

%% COMPUTE DERIVATIVES (4TH ORDER CENTRAL DIFFERENCE)
dV = zeros(length(V)-5,r);
for i=3:length(V)-3
    for k=1:r
        dV(i-2,k) = (1/(12*dt))*(-V(i+2,k)+8*V(i+1,k)-8*V(i-1,k)
            +V(i-2,k));
    end
end
% trim first and last two that are lost in derivative
V = V(3:end-3,1:r);

%%  BUILD HAVOK REGRESSION MODEL ON TIME DELAY COORDINATES
Xi = V\dV;
A = Xi(1:r-1,1:r-1)';
B = Xi(end,1:r-1)';
```

### Neural Networks for Koopman Embeddings

Despite the promise of Koopman embeddings, obtaining tractable representations has remained a central challenge. Recall that even for relatively simple dynamical systems, the eigenfunctions of the Koopman operator may be arbitrarily complex. Deep learning, which is well-suited for representing arbitrary functions, has recently emerged as a promising approach for discovering and representing Koopman eigenfunctions [550, 368, 513, 564, 412, 332, 349], providing a data-driven embedding of strongly nonlinear systems into intrinsic linear coordinates. In particular, the Koopman perspective fits naturally with the deep auto-encoder structure discussed in Chapter 6, where a few key latent variables $\mathbf{y} = \boldsymbol{\varphi}(\mathbf{x})$ are discovered to parameterize the dynamics. In a Koopman network, an additional constraint is enforced so that the dynamics must be linear on these latent variables, forcing the functions $\boldsymbol{\varphi}(\mathbf{x})$ to be Koopman eigenfunctions, as illustrated in Fig. 7.13. The constraint of linear dynamics is enforced by the loss function $\|\boldsymbol{\varphi}(\mathbf{x}_{k+1}) - \mathbf{K}\boldsymbol{\varphi}(\mathbf{x}_k)\|$, where $\mathbf{K}$ is a matrix. In general, linearity is enforced over multiple time steps, so that a trajectory is captured by iterating $\mathbf{K}$ on the latent variables. In addition, it is important to be able to map back to physical variables $\mathbf{x}$, which is why the autoencoder structure is favorable [349]. Variational autoencoders are also used for stochastic dynamical systems, such as molecular dynamics, where the map back to physical configuration space from the latent variables is probabilistic [550, 368].

For simple systems with a discrete eigenvalue spectrum, a compact representation may be obtained in terms of a few autoencoder variables. However, dynamical systems with continuous eigenvalue spectra defy low-dimensional representations using many existing neural network or Koopman representations. Continuous spectrum dynamics are ubiquitous, ranging from the simple pendulum to nonlinear optics and broadband turbulence. For example, the classical pendulum, given by

$$\ddot{x} = -\sin(\omega x) \tag{7.100}$$

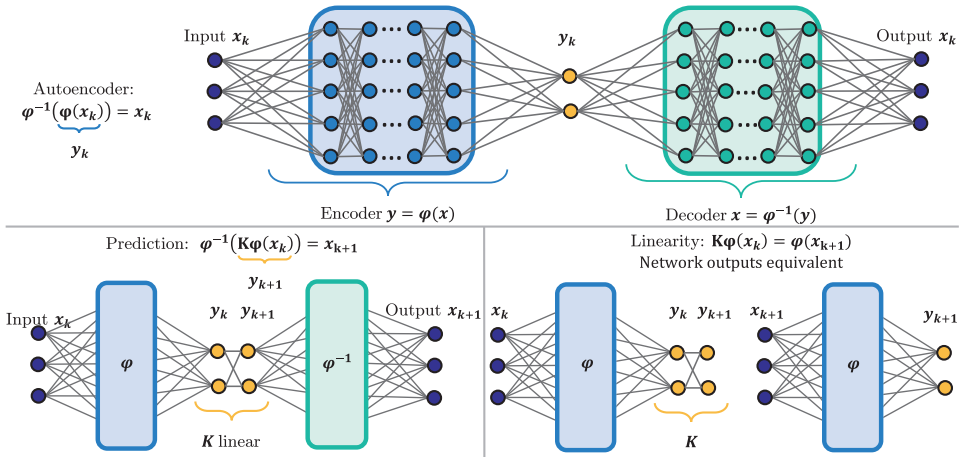


**Figure 7.13** Deep neural network architecture used to identify Koopman eigenfunctions $\boldsymbol{\varphi}(\mathbf{x})$. The network is based on a deep auto-encoder (a), which identifies intrinsic coordinates $\mathbf{y} = \boldsymbol{\varphi}(\mathbf{x})$. Additional loss functions are included to enforce linear dynamics in the auto-encoder variables (b,c). *Reproduced with permission from Lusch et al. [349].*

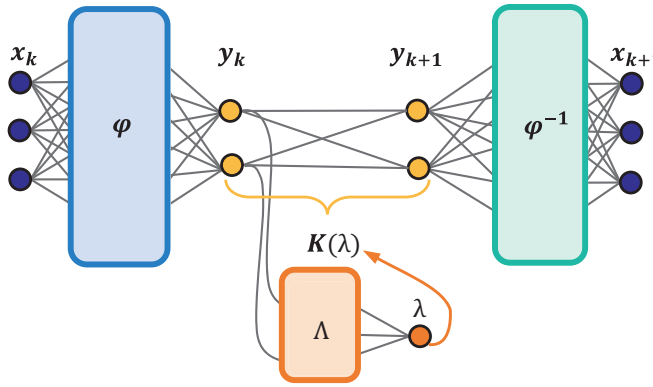


**Figure 7.14** Modified network architecture with auxiliary network to parameterize the continuous eigenvalue spectrum. A continuous eigenvalue $\lambda$ enables aggressive dimensionality reduction in the auto-encoder, avoiding the need for higher harmonics of the fundamental frequency that are generated by the nonlinearity. *Reproduced with permission from Lusch et al. [349].*

exhibits a continuous range of frequencies, from $\omega$ to 0, as the amplitude of the pendulum oscillation is increased. Thus, the continuous spectrum confounds a simple description in terms of a few Koopman eigenfunctions [378]. Indeed, away from the linear regime, an infinite Fourier sum is required to approximate the shift in frequency.

In a recent work by Lusch et al. [349], an auxiliary network is used to parameterize the continuously varying eigenvalue, enabling a network structure that is both parsimonious and interpretable. This parameterized network is depicted schematically in Fig. 7.14 and illustrated on the simple pendulum in Fig. 7.15. In contrast to other network structures, which require a large autoencoder layer to encode the continuous frequency shift with an asymptotic expansion in terms of harmonics of the natural frequency, the parameterized network is able to identify a single complex conjugate pair of eigenfunctions with a varying imaginary eigenvalue pair. If this explicit frequency dependence is unaccounted for, then a high-dimensional network is necessary to account for the shifting frequency and eigenvalues.

It is expected that neural network representations of dynamical systems, and Koopman embeddings in particular, will remain a growing area of interest in data-driven dynamics. Combining the representational power of deep learning with the elegance and simplicity of Koopman embeddings has the potential to transform the analysis and control of complex systems.

## Suggested Reading

### Texts

**(1)** **Nonlinear oscillations, dynamical systems, and bifurcations of vector fields**, by P. Holmes and J. Guckenheimer, 1983 [252].

**(2)** **Dynamic mode decomposition: Data-driven modeling of complex systems**, by J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor, 2016 [317].

**(3)** **Differential equations and dynamical systems**, by L. Perko, 2013 [427].

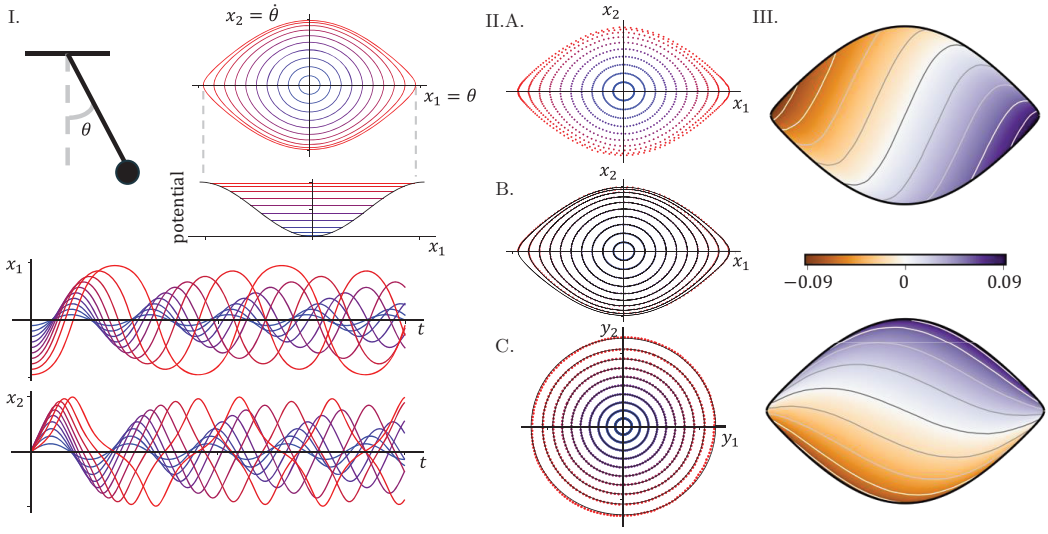


**Figure 7.15** Neural network embedding of the nonlinear pendulum, using the parameterized network in Fig. 7.14. As the pendulum amplitude increases, the frequency continuously changes (I). In the Koopman eigenfunction coordinates (III), the dynamics become linear, given by perfect circles (IIIC). *Reproduced with permission from Lusch et al. [349].*

## Papers and reviews

**(1)** **Distilling free-form natural laws from experimental data**, by M. Schmidt and H. Lipson, *Science*, 2009 [477].

**(2)** **Discovering governing equations from data by sparse identification of nonlinear dynamical systems**, by S. L. Brunton, J. L. Proctor, and J. N. Kutz, *Proceedings of the National Academy of Sciences*, 2016 [95].

**(3)** **On dynamic mode decomposition: theory and applications**, by J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, J. N. Kutz, *Journal of Computational Dynamics*, 2014 [535].

**(4)** **Hamiltonian systems and transformation in Hilbert Space**, by B. O. Koopman, *Proceedings of the National Academy of Sciences*, 1931 [300].

**(5)** **Spectral properties of dynamical systems, model reduction and decompositions**, by I. Mezić, *Nonlinear Dynamics*, 2005 [376].

**(6)** **Data-driven model reduction and transfer operator approximation**, by S. Klus, F. Nuske, P. Koltai, H. Wu, I. Kevrekidis, C. Schutte, and F. Noe, *Journal of Nonlinear Dynamics*, 2018 [293].

**(7)** **Hidden physics models: Machine learning of nonlinear partial differential equations**, by M. Raissi and G. E. Karniadakis, *Journal of Computational Physics*, 2018 [445].