20 Dynamic Mode Decomposition

Exploiting low dimensionality in complex systems has already been demonstrated to be an effective method for either computationally or theoretically reducing a given system to a more tractable form. In what has been proposed so far with POD reductions, we have used *model-based* algorithms. Thus a given set of governing equations dictate the dynamics of the underlying system. These equations construe the *model*. However, an alternative to this approach exists when either the model equations are beyond their range of validity or the governing equations simply are not known or well-formulated. In this alternative approach, experimental data itself drives the understanding of the system, without model equations being prescribed. Thus *data-based* algorithms can be constructed to understand, mimic and control the complex system. *Dynamic mode decomposition* (DMD) is a relatively new data-based algorithm that requires no underlying governing equations, rather snapshots of experimental measurements are used to predict and control a given system [64, 65, 66, 67]. In atmospheric sciences, a version of the DMD method is used to fit a set of data to an underlying (best statistical fit) linear stochastic model. This is known as *linear inverse models* (LIMs) [68, 69, 70]. Thus the methodology presented here uses data alone as the source for informing us of the state and dimensionality of the system.

20.1 Theory of Dynamic Mode Decomposition (DMD)

Exploitation of low dimensionality: that is the overarching goal of data-driven modeling methods. In the case of the technique of dynamic mode decomposition, it is the aim of the method to take advantage of low dimensionality in the experimental data itself without having to rely on a given set of governing equations.

The DMD method provides a decomposition of experimental data into a set of dynamic modes that are derived from snapshots of the data in time. The mathematics underlying the extraction

of dynamic information from time-resolved snapshots is closely related to the idea of the Arnoldi algorithm, one of the workhorses of fast computational solvers. To set the stage, we begin with the data collection process. To important parameters are required:

$$N = \text{number of spatial points saved per time snapshot} \tag{20.1.1a}$$

$$M = \text{number of snapshots taken.} \tag{20.1.1b}$$

In this case, it is imperative to collect data at regularly spaced intervals of time

$$\text{data collection times}: \quad t_{m+1} = t_m + \Delta t \tag{20.1.2}$$

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

The data can then be arranged into an $N \times M$ matrix

$$\mathbf{X} = \left[U(\mathbf{x}, t_1)\ U(\mathbf{x}, t_2)\ U(\mathbf{x}, t_3)\ \cdots\ U(\mathbf{x}, t_M)\right] \tag{20.1.3}$$

where $\mathbf{x}$ is a vector of data collection points of length N. A limitation of the DMD technique, at least applied in a straightforward manner, is apparent from the start: sampling must be done in equally spaced time intervals. Be that as it may, the objective is to mine the data matrix $\mathbf{X}$ for important dynamical information. For the purposes of the DMD method, the following matrix is also defined:

$$\mathbf{X}_j^k = \left[U(\mathbf{x}, t_j)\ U(\mathbf{x}, t_{j+1})\ \cdots\ U(\mathbf{x}, t_k)\right]. \tag{20.1.4}$$

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

The DMD method approximates the modes of the so-called *Koopman operator*. The Koopman operator is a linear, infinite dimensional operator that represents nonlinear, infinite dimensional dynamics without linearization [71, 72], and is the adjoint of the Perron–Frobenius operator. The method can be viewed as computing, from the experimental data, the eigenvalues and eigenvectors (low dimensional modes) of a linear model that approximates the underlying dynamics, even if the dynamics is nonlinear. Since the model is assumed to be linear, the decomposition gives the growth rates and frequencies associated with each mode. If the underlying model is linear, then the DMD method recovers the leading eigenvalues and eigenvectors normally computed using standard solution methods for linear differential equations.

Mathematically, the Koopman operator $\mathbf{A}$ is a linear, time-independent operator $\mathbf{A}$ such that

$$\mathbf{x}_{j+1} = \mathbf{A}\mathbf{x}_j \tag{20.1.5}$$

where j indicates the specific data collection time and $\mathbf{A}$ is the linear operator that maps the data from time t_j to t_{j+1}. The vector $\mathbf{x}_j$ is an N-dimensional vector of the data points collected at time j. The computation of the Koopman operator is at the heart of the DMD methodology. As already stated, the mapping over Δ is linear even though the underlying dynamics that generated $\mathbf{x}_j$ may be nonlinear. It should be noted that this is different from linearizing the dynamics.

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

$$\mathbf{X}_1^{M-1} = [\mathbf{x}_1\ \mathbf{x}_2\ \mathbf{x}_3\ \cdots\ \mathbf{x}_{M-1}]\,. \tag{20.1.6}$$

Making use of (20.1.5), this matrix reduces to

$$\mathbf{X}_1^{M-1} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{A}\mathbf{x}_1 & \mathbf{A}^2\mathbf{x}_1 & \cdots & \mathbf{A}^{M-2}\mathbf{x}_1 \end{bmatrix}. \tag{20.1.7}$$

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

$$\mathbf{x}_M = \sum_{m=1}^{M-1} b_m \mathbf{x}_m + \mathbf{r} \tag{20.1.8}$$

where the b_m are the coefficients of the Krylov space vectors and $\mathbf{r}$ is the residual (or error) that lies outside (orthogonal to) the Krylov space. Ultimately, this best fit to the data using this DMD procedure will be done in an L^2 sense using a pseudo-inverse.

Before proceeding further, it is at this point that the data matrix $\mathbf{X}_1^{M-1}$ in (20.1.7) should be considered further. In particular, our dimensionality reduction methods look to take advantage of any low dimensional structures in the data. To exploit this, the SVD of (20.1.7) is computed:

$$\mathbf{X}_1^{M-1} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^* \tag{20.1.9}$$

where $*$ denotes the conjugate transpose, $\mathbf{U} \in \mathbb{C}^{N\times K}$, $\mathbf{\Sigma} \in \mathbb{C}^{K\times K}$ and $\mathbf{V} \in \mathbb{C}^{M-1\times K}$. Here K is the reduced SVD's approximation to the rank of $\mathbf{X}_1^{M-1}$. If the data matrix is full rank and the data have no suitable low dimensional structure, then the DMD method fails immediately. However, if the data matrix can be approximated by a low-rank matrix, then DMD can take advantage of this low dimensional structure to project a future state of the system. Thus once again, the SVD plays the critical role in the methodology.

Armed with the reduction (20.1.9) to (20.1.7), we can return to the results of the Koopman operator and Krylov basis (20.1.8). Specifically, generalizing (20.1.5) to its matrix form yields

$$\mathbf{A}\mathbf{X}_1^{M-1} = \mathbf{X}_2^M. \tag{20.1.10}$$

But by using (20.1.8), the right-hand side of this equation can be written in the form

$$\mathbf{X}_2^M = \mathbf{X}_1^{M-1}\mathbf{S} + \mathbf{r}e_{M-1}^* \tag{20.1.11}$$

where e_{M-1} is the $(M-1)$th unit vector and

$$\mathbf{S} = \begin{bmatrix} 0 & \cdots & 0 & b_1 \\ 1 & \ddots & 0 & b_2 \\ 0 & \ddots & \ddots & \vdots \\ & \ddots & \ddots & 0 \quad b_{M-2} \\ 0 & \cdots & 0 & 1 \quad b_{M-1} \end{bmatrix}. \tag{20.1.12}$$

Recall that the b_j are the unknown coefficients in (20.1.8).

The key idea now is the observation that the eigenvalues of $\mathbf{S}$ approximate some of the eigenvalues of the unknown Koopman operator $\mathbf{A}$, making the DMD method similar to the Arnoldi algorithm and its approximations to the Ritz eigenvalues. Schmid [64] showed that rather than computing the matrix $\mathbf{S}$ directly, we can instead compute the *lower-rank* matrix

$$\tilde{\mathbf{S}} = \mathbf{U}^*\mathbf{X}_2^M\mathbf{V}\mathbf{\Sigma}^{-1} \tag{20.1.13}$$

which is related to $\mathbf{S}$ via a similarity transformation. Recall that the matrices $\mathbf{U}$, $\mathbf{\Sigma}$ and $\mathbf{V}$ arise from the SVD reduction of $\mathbf{X}_1^{M-1}$ in (20.1.9).

Consider then the eigenvalue problem associated with $\tilde{\mathbf{S}}$:

$$\tilde{\mathbf{S}}\mathbf{y}_k = \mu_k\mathbf{y}_k \qquad k = 1, 2, \cdots, K \tag{20.1.14}$$

where K is the rank of the approximation we are choosing to make. The eigenvalues μ_k capture the time dynamics of the discrete Koopman map $\mathbf{A}$ as a Δt step is taken forward in time. These eigenvalues and eigenvectors can be related back to the similarity transformed original eigenvalues and eigenvectors of $\mathbf{S}$ in order to construct the DMD modes:

$$\psi_k = \mathbf{U}\mathbf{y}_k. \tag{20.1.15}$$

With the low-rank approximations of both the eigenvalues and eigenvectors in hand, the projected future solution can be constructed for all time in the future. By first rewriting for convenience $\omega_k = \ln(\mu_k)/\Delta t$ (recall that the Koopman operator time dynamics is linear), then the approximate solution at all future times, $\mathbf{x}_{DMD}(t)$, is given by

$$\mathbf{x}_{DMD}(t) = \sum_{k=1}^{K} b_k(0)\psi_k(\mathbf{x}\exp(\omega t)) = \mathbf{\Psi}\mathrm{diag}(\exp(\omega t)\mathbf{b} \tag{20.1.16}$$

where $b_k(0)$ is the initial amplitude of each mode, $\mathbf{\Psi}$ is the matrix whose columns are the eigenvectors ψ_k, $\mathrm{diag}(\omega t)$ is a diagonal matrix whose entries are the eigenvalues $\exp(\omega_k t)$, and $\mathbf{b}$ is a vector of the coefficients b_k.

It only remains to compute the initial coefficient values $b_k(0)$. If we consider the initial snapshot ($\mathbf{x}_1$) at time zero, let's say, then (20.1.16) gives $\mathbf{x}_1 = \mathbf{\Psi}\mathbf{b}$. This generically is not a square matrix so that its solution

$$\mathbf{b} = \mathbf{\Psi}^+\mathbf{x}_1 \tag{20.1.17}$$

can be found using a pseudo-inverse. Indeed, $\mathbf{\Psi}^+$ denotes the Moore–Penrose pseudo-inverse that can be accessed in MATLAB via the **pinv** command. As already discussed in the compressive sensing chapter, the pseudo-inverse is equivalent to finding the best solution $\mathbf{b}$ in the least-squares (best fit) sense. This is equivalent to how DMD modes were derived originally.

Overall then, the DMD algorithm presented here takes advantage of low dimensionality in the data in order to make a low-rank approximation of the linear mapping that best approximates the nonlinear dynamics of the data collected for the system. Once this is done, a prediction of the future state of the system is achieved for all time. Unlike the POD method, which requires solving a low-rank set of dynamical quantities to predict the future state, no additional work is required for the future state prediction outside of plugging in the desired future time into (20.1.16). Thus

the advantages of DMD revolve around the fact that (i) no equations are needed, and (ii) the future state is known for all time (of course, provided the DMD approximation holds).

The algorithm is as follows:

(i) Sample data at N prescribed locations M times. The data snapshots should be evenly spaced in time by a fixed Δt. This gives the data matrix $\mathbf{X}$.

(ii) From the data matrix $\mathbf{X}$, construct the two submatrices $\mathbf{X}_1^{M-1}$ and $\mathbf{X}_2^{M}$.

(iii) Compute the SVD decomposition of $\mathbf{X}_1^{M-1}$.

(iv) The matrix $\tilde{\mathbf{S}}$ can then be computed and its eigenvalues and eigenvectors found.

(v) Project the initial state of the system onto the DMD modes using the pseudo-inverse.

(vi) Compute the solution at any future time using the DMD modes along with their projection to the initial conditions and the time dynamics computed using the eigenvalue of $\tilde{\mathbf{S}}$.

20.2 Dynamics of DMD Versus POD

To illustrate the application of the DMD method, a simple example is chosen in order to highlight the implementation of the algorithm advocated at the end of the last section. In fact, it is the same example used to illustrate the POD reduction technique, namely, the $N = 2$ soliton dynamics of the nonlinear Schrödinger (NLS) equation (see Section 18.3). For completeness, the NLS will be given here again:

$$iu_t + \frac{1}{2}u_{xx} + |u|^2 u = 0 \tag{20.2.1}$$

with initial conditions $N\text{sech}(x)$. Thus the data to be used will be generated through simulations of this equation. The PDE solution can be computed using FFTs with the following code. First, the initialization of the space and time discretization is computed:

```
% space
L=40; n=512;
x2=linspace(-L/2,L/2,n+1); x=x2(1:n);
k=(2*pi/L)*[0:n/2-1 -n/2:-1].';
% time
slices=20;
t=linspace(0,2*pi,slices+1); dt=t(2)-t(1);
```

Note that the parameter **slices** will be critical in what follows ($M =$ **slices**$+1$). Specifically, this determines the number of data samples taken over the time interval $t \in [0, 2\pi]$. The initial conditions and time-stepping routines are as follows


```
u=2*sech(x).';   % initial conditions
ut=fft(u);
[t,utsol]=ode45('nls_rhs',t,ut,[],k);
for j=1:length(t)
    usol(j,:)=ifft(utsol(j,:));  % bring back to space
end
```

This implementation of the solver was also demonstrated in Section 18.3. Note that a call is made to the right-hand side function **nls_rhs.m** which is the following

```
function nls_rhs=nls_rhs(z,ut,dummy,k)
u=ifft(ut);
nls_rhs= -(i/2)*(k.^2).*ut + i*fft( (abs(u).^2).*u );
```

Having executed the above code, the matrix **usol** results which arranges the simulation data into a matrix with M time slices and N data points at each slice. Thus the matrix **usol** is the desired data matrix $\mathbf{X}$ required in the first step of the DMD algorithm.

The top left panel of Fig. 20.1 shows the result of the simulation of the NLS equation over the interval $t \in [0, 2\pi]$ with $M = 21$ and $N = 512$. Our goal is to sample from this set of data in order

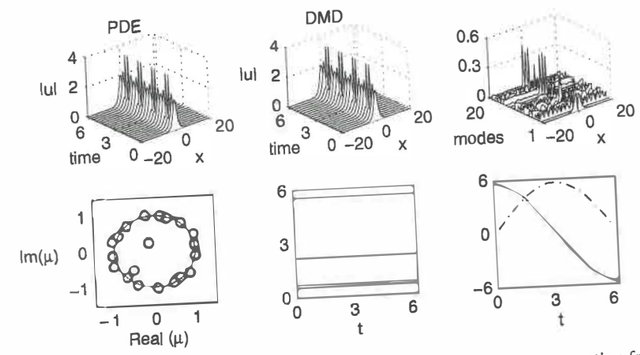

Figure 20.1: The top left panels represent the full PDE simulation and DMD reconstruction from sampling for $t \in [0, 2\pi]$ and $M = 21$ slices of data. The DMD modes used for the linear reconstruction are shown in the top right panel with its eigenvalue distribution μ_k shown in the bottom left panel. The solid line of the bottom left panel represents the unit circle for which eigenvalues outside of it are growth modes. The bottom right panels show the absolute value of the time dynamics (bottom middle) for each of the 20 DMD modes and the real (solid) and imaginary (dotted) amplitude of the time evolution of the first mode (bottom right).

to (i) find a low dimensional framework to exploit, and (ii) project the future state of the system in time without relying on computations of the PDE.

Our first objective is to then construct the data matrices in the second part of the algorithm: $\mathbf{X}_1^{M-1}$ and $\mathbf{X}_2^M$. Thus we have

```
X = usol.'; X1 = X(:,1:end-1); X2 = X(:,2:end);
```

Armed with these subsets of the full data matrix, the SVD decomposition of $\mathbf{X}_1^{M-1}$ can be computed and the eigenvalues and eigenvectors of $\tilde{\mathbf{S}}$ found. The following code performs these operations.

```
[U,Sigma,V] = svd(X1, 'econ');
S = U'*X2*V*diag(1./diag(Sigma));
[eV,D] = eig(S);
mu = diag(D);
omega = log(mu)/(dt);
Phi = U*eV;
```

Note that the vector **vals** contains the eigenvalues μ of interest to the DMD dynamics and **Phi** contains the DMD modes which are the basis function used to predict the future state of the system.

The final steps in the DMD algorithm both project the initial conditions, via the pseudo-inverse, to the DMD modes and then predicts the future state using the DMD modes and their linear time dynamics dictated by the μ_k. The following gives a reconstruction of the dynamics over the same interval as used in collecting the data.

```
y0 = Phi\u;  % pseudo-inverse initial conditions

u_modes = zeros(size(vl,2),length(t));
for iter = 1:length(t)
    u_modes(:,iter) =(y0.*exp(omega*t(iter)));
end
u_dmd = Phi*u_modes;
```

Figure 20.1 gives a summary of the DMD method, including demonstrating the PDE versus DMD dynamics. In fact, in the top panels of this figure, the PDE (exact solution) and DMD reconstruction are shown. In the top right panel, the 20 DMD modes constructed from the data reduction are illustrated. Thus these modes are used as the basis for reconstructing the best

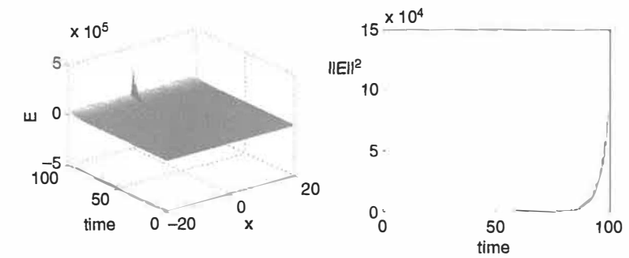

Figure 20.2: Evolution of the error, i.e. the absolute value of the difference between the DMD solution and the PDE solution, over the time interval $t \in [0, 100]$ (left panel) and the evolution of the norm of the error (right panel). Due to the DMD approximation having eigenvalues outside the unit circle, growth modes appear which blow-up for long time projections of the dynamics. However, the error remains small for times less than $t \approx 50$, giving an upper range for when the DMD approximation holds.

possible linear fit to the nonlinear evolution. Their time evolution is determined by the eigenvalues μ_k which are given in the bottom left panel. Also plotted on this panel is the unit circle. Those eigenvalues that lie outside the unit circle represent growth modes in the system. The absolute value of the time dynamics for each mode is given in the bottom middle panel while the first mode time dynamics (the real and imaginary parts) is given in the bottom right panel.

Although the $N = 2$ soliton dynamics is oscillatory (nonlinear) in time, the DMD reduction shows that a few of the eigenvalues μ_k sit slightly outside the unit circle when $M = 21$. This will ultimately lead to a long-term blow-up of the predicted solution using the DMD method. Figure 20.2 shows the error as a function of time. Here, a comparison is made between the DMD predicted solution, using data only in $t \in [0, 2\pi]$, and the full numerical solution for the interval $t \in [0, 100]$. The agreement is quite good until about $t \approx 40$ when the solution starts to diverge exponentially from the true solution. This is simply a result of sampling error as will be shown in what follows. Either way, it is important to recognize that how one samples and how frequently one samples can make a big impact on the DMD method. Moreover, any noise in the system and/or data can push an eigenvalue outside of the unit circle and limit the range (in time) of the DMD algorithm and prediction. Thus to continue to use it in this context, resampling must occur in order to re-project the data and avoid eventual (unphysical) blow-up of the DMD predictions.

As already stated, the computation of the eigenvalues μ_k changes as a function of the sampling rate. To illustrate this, two new figures are produced. Figure 20.3 shows the eigenvalue distribution μ_k for $M = 16, 21, 31$ and 41 when sampling the same PDE dynamics over the interval $t \in [0, 2\pi]$. Note that as the sampling rate gets higher, the eigenvalues outside the unit circle draw closer to the unit circle, thus producing DMD predictions that are valid for a longer window in time since the growth rates of these eigenvalues is smaller. The modes ψ_k associated with these eigenvalues μ_k are shown in Fig. 20.4 for the corresponding number of samplings. These modes are used in linear combination with their time dynamics given by the μ_k to give the approximation to the true PDE dynamics.

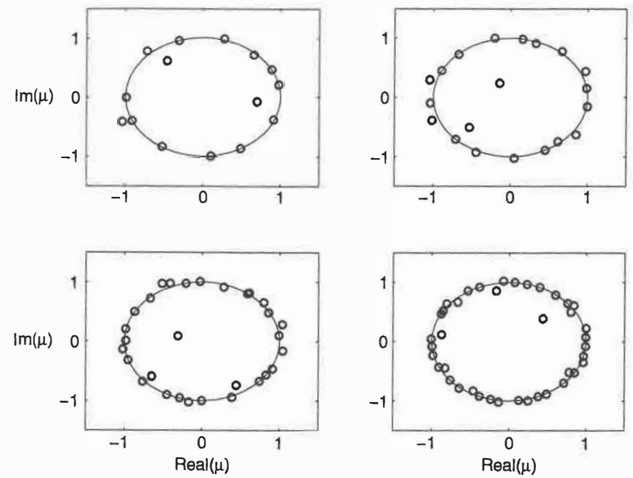

Figure 20.3: Spectral distribution of the eigenvalues μ_k as a function of the sampling rate over the interval $t \in [0, 2\pi]$. From top left to bottom right, the values are $M = 16, 21, 31$ and 41. Note that for higher sampling, the eigenvalues outside of the unit circle collapse to the unit circle, allowing for longer time dynamical predictions using the DMD modes. The corresponding eigenvectors are shown in Fig. 20.4.

An advantage of the DMD algorithm is that it actually can easily identify growth modes of a given physical system. In the example above, the growth modes were spurious. However, in many physical systems, there may indeed be a growth mode that one wishes to suppress. Thus the DMD framework is a natural way to apply the ideas of control theory, i.e. the growth modes are identified and a control algorithm is placed on the complex physical system with the aim of suppressing the pernicious growth mode(s). Such an application might be for the control of turbulent flow fields, for instance, where full simulations not only disagree with experimental observations quantitatively, but where such simulations are prohibitively expensive to do in real time. With the DMD framework, direct measurements can be performed in order to understand the underlying dynamics and growth modes for short windows of time into the future, thus allowing for control and manipulation of the flow field.

Finally, a comment should be made to conclude the section about the DMD method in comparison with the POD method. In the POD method, a low dimensional basis is selected from the data matrix $\mathbf{X}$ by use of the SVD. The dynamics of the governing equations are then projected onto this reduced basis and the resulting nonlinear dynamical system for the mode amplitudes evolved forward in time. With DMD, the dynamics are assumed to be linear so that an exact solution can be constructed. Thus there is no need to evolve a dynamical system forward in time for the mode amplitudes, rather, once the reduction is done and the DMD modes and their eigenvalues computed, then the future state of the system can be predicted without any addition work.

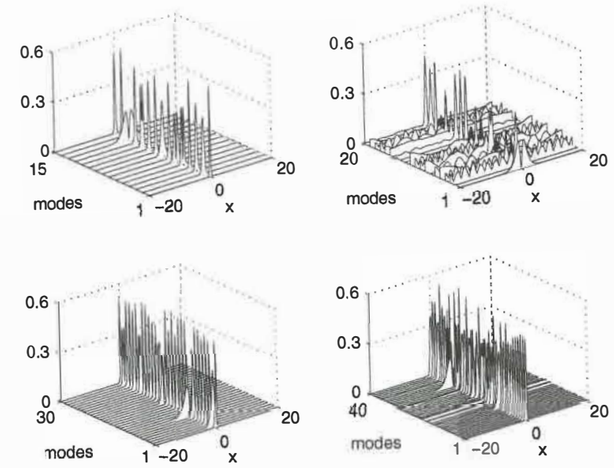

Figure 20.4: Eigenvectors associated with the eigenvalue distribution μ_k of Fig. 20.3. For $M = 16, 21, 31$ and 41, there are 15, 20, 30 and 40 eigenvectors generated, respectively, for reconstruction or approximation of the PDE solution using these DMD basis modes.

Thus the POD method retains the *nonlinear dynamics* of the system through its projection of the low dimensional modes onto the governing equations. DMD tries to construct the Koopman operator that best approximates the nonlinear dynamics directly without the need of a governing equation or additional evolution of any equation. Both have strengths and weaknesses, so that the method of choice usually boils down to taking maximal advantage of specific aspects of the problem at hand. But if the equations are unknown, hard to parameterize, overly approximated, etc, then DMD provides a viable method for extracting and potentially controlling some underlying complex system.

20.3 Applications of DMD

The example given in the previous section is nice in the sense that the computations for the PDE can be performed quite easily and an intuitive understanding of the dynamics is already present. Thus the comparison of the POD, DMD and full PDE solution techniques can be made and each of their advantages and disadvantages compared. But just like the POD method considered in Section 18.3, the primary goal is to apply the DMD method to complex physical systems where full advantage can be taken of its strengths. In this section, two examples will be given for how one might use the DMD method for the modeling of physical and/or computational systems.