Chapter 7
Delay Coordinates, ERA, and Hidden Markov Models

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

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

7.1 ▪ Delay coordinates and shift-stacking data

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

7.1.1 ▪ Example: DMD fails to capture a standing wave

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

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

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

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

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

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

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

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

```
Omega = 0.0255
```

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

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

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

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

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

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

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

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

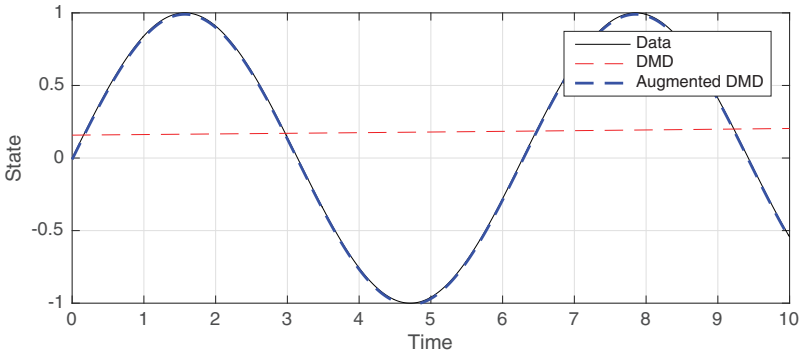


Figure 7.1. *Example of DMD and augmented DMD on a standing wave example.*

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

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

The eigenvalues returned by this code are

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

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

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

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

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

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

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

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

7.1.2 ▪ Time-delay coordinates

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

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

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

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

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

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

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

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

7.2 ▪ Connection to ERA and Hankel matrices

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

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

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

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

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

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

A discrete-time impulse response is given by

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

which results in output measurements

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

7.2.1 ▪ Simple ERA examples

Consider the differential equation

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

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

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

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

However, if the dynamics are more complicated,

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

so that there are two modes,

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

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

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

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

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

```
clear all, close all, clc

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

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

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

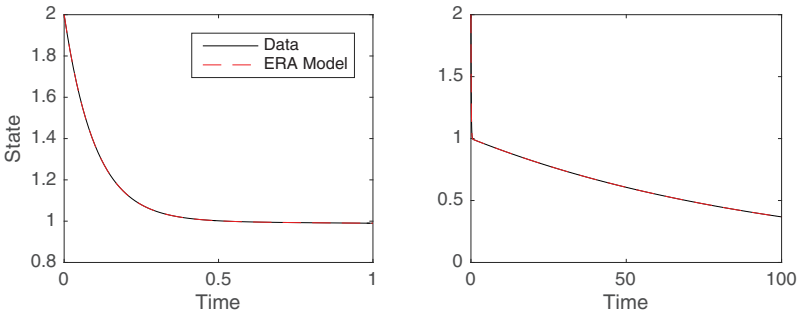


Figure 7.2. *A two-mode ERA model can accurately capture the dynamics of the system from* (7.18) *with slow and fast modes.*

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

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

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

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

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

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

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

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

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

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


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

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

7.3 ▪ HMMs

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

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

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

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

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

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

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

7.3.1 ▪ Weather system example

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

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

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

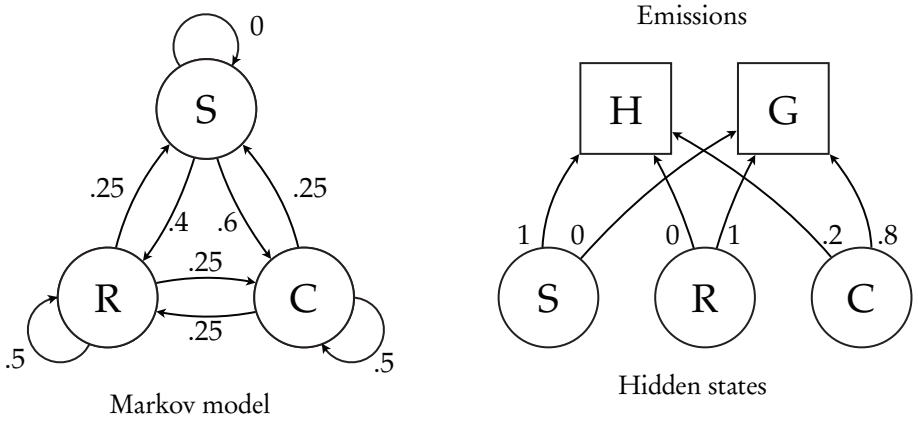
```
clear all, close all, clc
% Weather state [S R C]: sun (S), rain (R), clouds (C)
% if sunny today: 0% sun tomorrow, 40% rain, 60% cloud
% rainy today: 25% sun tomorrow, 50% rain, 25% cloud
% cloudy today: 25% sun tomorrow, 25% rain, 50% cloud
```

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

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

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

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



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

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

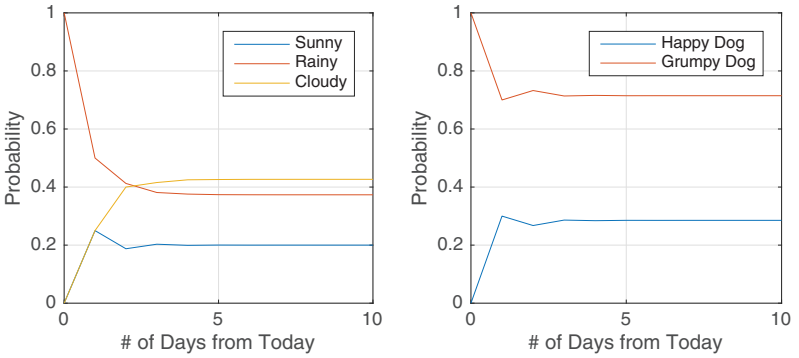


Figure 7.4. *Probability distribution of the hidden state (left) and the emission (right) using a* 10*-day forecast.*

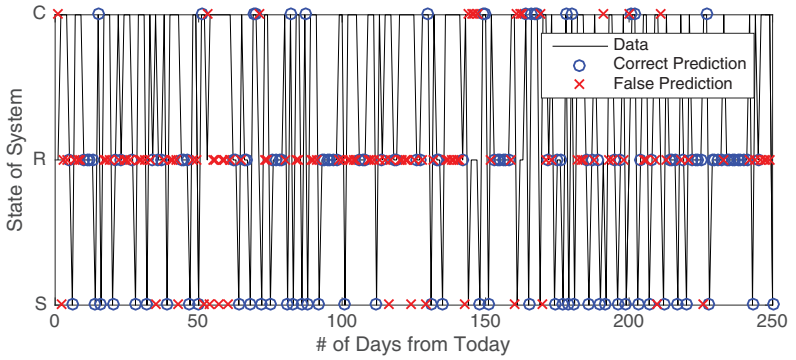


Figure 7.5. *Prediction of the hidden state given the sequence of emissions for* 250*-day simulation.*

```
end
```

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

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

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

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

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

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

Other useful HMM commands include the following:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

The output of the DMD algorithm is

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

which is close to $\mathbf{T}$, up to a permutation on the state labels. The eigenvalues are

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