Lecture 28
DMD and Delay Coordinates

Jason J. Bramburger

In the previous two lectures we introduced the concept of Dynamic Mode Decomposition (DMD) and saw that it produces a linear model that governs our data snapshots. This is awesome because linear models are easy to work with! They're so easy in fact that I already know what the solution to a linear system of ODEs will be: just exponential functions. This allowed us to jump right over providing the dynamical model and just giving the solution. Hence, we can use DMD to forecast systems far intro the future by just picking the time we want to go up until and putting it into the $\mathbf{x}_{DMD}(t)$ solution to the DMD model.

Okay, not everything is perfect here though. Imagine you have have just one measurement as a function of time, given by $x(t) = \sin(t)$. This is an extreme example, but it perfectly captures what we will talk about in this lecture. With a spatial step $\Delta t > 0$ we can define $t_n = (n-1)\Delta t$, with $n = 1, \dots N$, and create the snapshot matrix

$$\mathbf{X} = \begin{bmatrix} x(t_1) & x(t_2) & x(t_3) & \cdots & x(t_N) \end{bmatrix}.$$

Notice that this matrix is of size $1 \times N$, meaning that its rank is at most 1. Then, the Koopman operator/matrix $\mathbf{A}$ is just a real number, i.e. 1×1 matrix. So, what's the problem here? Well, the eigenvalue of a 1×1 real matrix is a real number (don't overthink this - I wrote it more complicated than it needs to be). The problem is then that the snapshots are governed by a sine function, which is the sum of exponential functions with imaginary exponents:

$$\sin(t) = \frac{1}{2\mathrm{i}}\left(\mathrm{e}^{\mathrm{i}t} - \mathrm{e}^{-\mathrm{i}t}\right),$$

meaning that we need complex eigenvalues to get oscillations from DMD. With only one real eigenvalue our DMD model can only grow exponentially or decay exponentially, not oscillate. Therefore, DMD will fail miserably on this super simple example.

The above example was silly and you probably think it's useless, but it emphasizes something very important for DMD. What if I have incomplete information about a system? That is, there may be tons of variables associated to my system, but I can only measure a couple of them? Or, maybe I can only measure a system at a few points in space - maybe near the sides of a pool, but not in the middle. If I try to use DMD on the few variables I can measure then I'll probably be missing something, just like the example above. It turns out that we don't necessarily need to make more observations in order to apply DMD to incomplete information.

Time-Delay Coordinates

Suppose we are given an incomplete set of measurements $\mathbf{x}(t_n)$, $n = 1, \dots, N$. We can artificially inflate the dimension of the system using time-delay coordinate embeddings. We will define our snapshot matrix by

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}(t_1) & \mathbf{x}(t_2) & \cdots & \mathbf{x}(t_{N-d}) \\ \mathbf{x}(t_2) & \mathbf{x}(t_3) & \cdots & \mathbf{x}(t_{N-d+1}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}(t_d) & \mathbf{x}(t_{d+1}) & \cdots & \mathbf{x}(t_N) \end{bmatrix}.$$

Here $d \geq 1$ is the number of time-delays and $\mathbf{X}$ is called a **Hankel matrix**. Notice that $\mathbf{X}$ now has d times the number of components in $\mathbf{x}(t)$ rows and $(N-d)$ columns. The individual rows and columns of $\mathbf{X}$ are successive snapshots of the data, each starting at different times.

We can use the SVD to produce *observables* of the data $\mathbf{x}(t)$ from the Hankel matrix $\mathbf{X}$. The SVD gives

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

so that the columns of $\mathbf{U}$ constitute the new observables and the columns of $\mathbf{V}$ are how these observables evolve in time. Here's the especially great part: we can apply DMD to the $d \geq 1$ columns of $\mathbf{V}$ to arrive at a more accurate

DMD model. We want to take d to be large enough so that the DMD model can truly capture the dynamics of the of the original snapshots $\mathbf{x}(t)$. Remember that with the SVD, if we forecast the columns of $\mathbf{V}$, we can get back to $\mathbf{X}$ by simply multiplying these forecasts with Σ and the observables $\mathbf{U}$ to get back to a forecast of $\mathbf{X}$. Having a forecast of $\mathbf{X}$ immediately gives us a forecast of the original snapshots $\mathbf{x}(t)$ just from the form of the rows.

Return to the Simple Example

Now that we know how to 'artificially' inflate the size of our snapshot matrix, let's see how it does with the simple signal that was introduced at the beginning. We will take $x(t) = \sin(t)$ with $\Delta t = 0.1$ and $N = 201$. The following code synthesizes the data, creates the Hankel matrix, and plots the relevant SVD information.

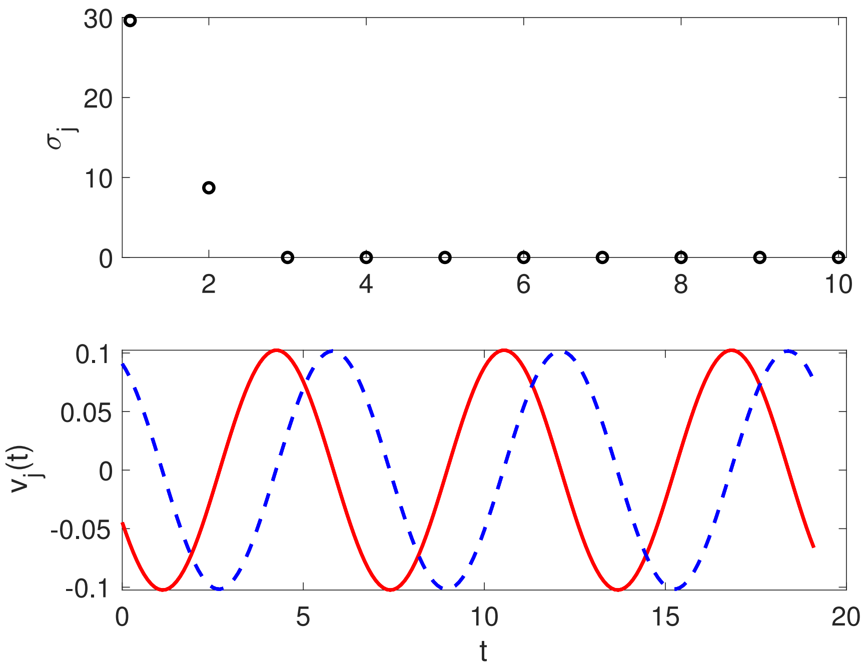
```
% Generate data
dt = 0.1;
t = 0:dt:20;
x = sin(t);

% Hankel matrix
delays = 10;
xd = hankel(x(1:delays),x(delays:end));

% Apply SVD
[U, S, V] = svd(xd); % SVD of delay matrix

% Plot SVD Results
subplot(2,1,1) % Singular values
plot(diag(S),'ko','Linewidth',2)
ylabel('\sigma_j')
set(gca,'Fontsize',16,'Xlim',[0.9 delays+0.1])

subplot(2,1,2) % Right-singular vectors
plot(t(1:end-delays+1),V(:,1),'r','Linewidth',2)
hold on
plot(t(1:end-delays+1),V(:,2),'b--','Linewidth',2)
xlabel('t')
ylabel('v_j(t)')
set(gca,'Fontsize',16)
```



Go ahead and check out the singular values. All but the first two are exactly zero! Not close to zero, actually zero. Furthermore, look at the columns of V. They look like perfect sinusoidal modes. This should remind you a lot of your third assignment - the red curve looks to be the derivative of the blue curve, or equivalently, the blue curve looks to be the negative of the derivative of the red curve. This is certainly a property of sines and cosines, meaning that that time-delay procedure was able to locate a basis where the time dynamics are essentially just sines and cosines. We can then apply DMD to get our linear model for the columns of V. Since only the first two singular values are nonzero, we will only use the first two columns of V since the other columns have no effect on the snapshot matrix X.

```
X1 = V(1:end-1,1:2)';
X2 = V(2:end,1:2)'; % first two columns of V

[U2, S2, V2] = svd(X1,'econ');
Stilde = U2'*X2*V2*diag(1./diag(S2));
[eV, D] = eig(Stilde); % compute eigenvalues + eigenvectors
mu = diag(D); % extract eigenvalues
omega = log(mu)/dt;
Phi = U2*eV;

y0 = Phi\X1(:,1); % pseudoinverse to get initial conditions

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

Check out what the continuous-time eigenvalues ω turn out to be: $\pm$i. That means that the DMD solution is a linear combination of $\mathrm{e}^{\mathrm{i}t}$ and $\mathrm{e}^{-\mathrm{i}t}$, exactly what I said before we needed to make $\sin(t)$. Therefore, with the delay coordinates we were able to inflate the dimension enough to be able to apply DMD and find a linear mode of exponentials to forecast the signal $x(t) = \sin(t)$.

<u>**The Van der Pol Oscillator**</u> Let's look at another, more complex example. We will consider solutions to the Van der Pol oscillator, an ODE given by

$$\begin{aligned}\dot{x} &= y\\ \dot{y} &= -x + 10(1-x^2)y.\end{aligned}$$

Here's some code to simulate and plot the solution in MATLAB.

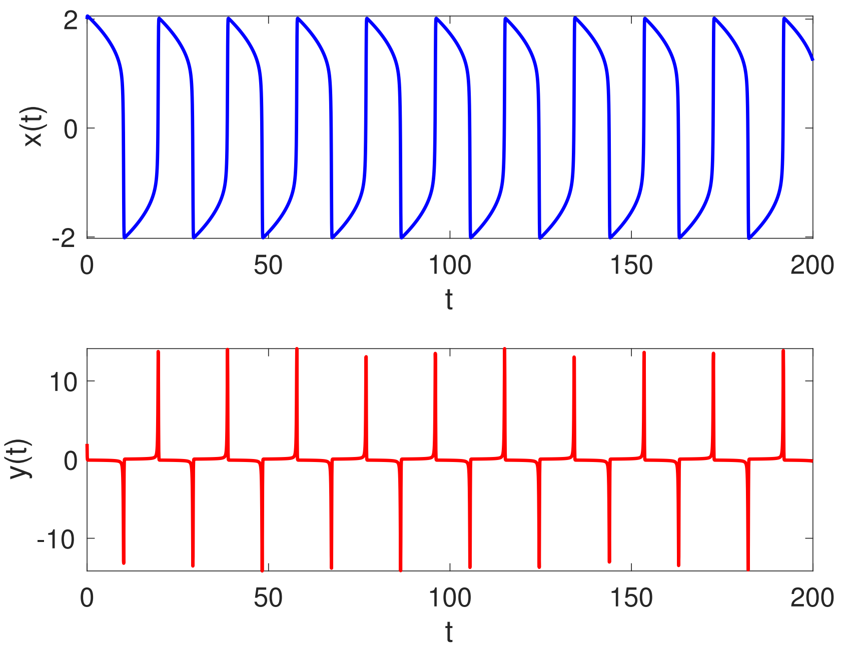
```
% Clean workspace
clear all; close all; clc

% Simulate ODE
dt = 0.05;
t = 0:dt:200;
x0 = [2; 2];
[t, x] = ode45(@(t,x) VdP(t,x),t,x0);

% Plot solution
subplot(2,1,1) % x(t)
plot(t,x(:,1),'b','Linewidth',2)
xlabel('t')
ylabel('x(t)')
set(gca,'Fontsize',16)
axis tight

subplot(2,1,2) % y(t)
plot(t,x(:,2),'r','Linewidth',2)
xlabel('t')
```

```
ylabel('y(t)')
set(gca,'Fontsize',16)
axis tight
```



Using the function:

```
function rhs = VdP(t,x)
    rhs = [x(2); -x(1) + 10*(1 - x(1)^2)*x(2)];
end
```

The solution is completely periodic, but it is *highly* nonlinear. No single sine and cosine combination can be put together to fit that, meaning that even though we have two observations - enough to have oscillatory solutions - there is no way that only two observations is going to be enough to get a good DMD solution for the Van der Pol oscillator. Hence, we will inflate the number of observations using delay coordinates in an effort to get ourselves to a point where we can fit the periodic behaviour using finitely many sines and cosines. I'm only going to use the $x(t)$ component for this demonstration, but you can easily add in $y(t)$ too or just take $y(t)$ on its own and get similar results.

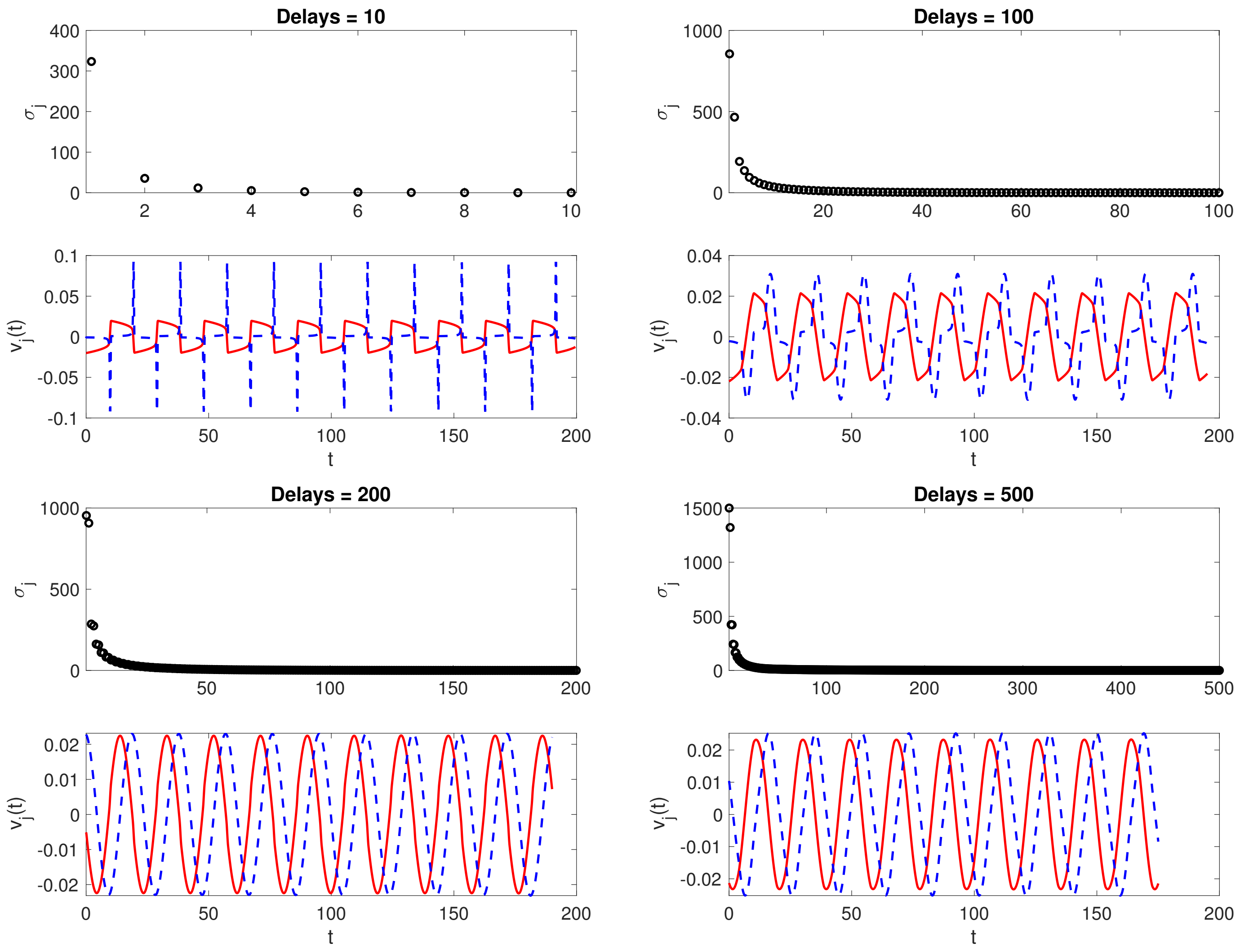
```
% Hankel matrix
delays = 10;
xd = hankel(x(1:delays,1),x(delays:end,1));

% Apply SVD
[U, S, V] = svd(xd); % SVD of delay matrix

% Plot SVD Results
subplot(2,1,1) % Singular values
plot(diag(S),'ko','Linewidth',2)
ylabel('\sigma_j')
set(gca,'Fontsize',16,'Xlim',[0.9 delays+0.1])
title(['Delays = ', num2str(delays)])

subplot(2,1,2) % Right-singular vectors
plot(t(1:end-delays+1),V(:,1),'r','Linewidth',2)
hold on
plot(t(1:end-delays+1),V(:,2),'b--','Linewidth',2)
```

```
xlabel('t')
ylabel('v_j(t)')
set(gca,'Fontsize',16)
```



Look what happens to the first two columns of V as the number of delays is increased. The oscillations become more and more linear, meaning they look more and more like sines and cosines. If you go into the code you will see that the other columns of V are similarly linear for a high number of delays. Hence, we can easily fit a DMD model to the data when we have lots of delays.

So, what's going on? One way to think of it is the following. The original signal was periodic, meaning we can expand it in a Fourier basis. What is a Fourier basis other than just sines and cosines with different frequencies? The problem with this expansion is that it requires infinitely many terms, especially since our signal is so drastically different from a simple sine function. Then, one way to think about what the delay coordinate + SVD procedure is doing is that it is creating a new linear variable whose time dynamics are governed by simple sinusoidal motion. Putting all of these variables together will result in the original highly nonlinear signal, just like adding up all the simple Fourier terms can give you very complicated signals.