Lecture 23: Dynamic Mode Decomposition

MEGR 7080/8090: Dynamic System Learning and Estimation

Artur Wolek

Department of Mechanical Engineering and Engineering Science
UNC Charlotte

Fall 2022

System ID via Dynamic Mode Decomposition

Goal: find a discrete-time linear system

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k \tag{1}$$

that approximates the data obtained from the true dynamical system

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k) \ . \tag{2}$$

Training data consists of m regularly spaced (in time) "snapshots"

$$\mathcal{T} = \{\boldsymbol{x}(t_k), \boldsymbol{x}(t'_k)\}_{k=1}^{m} \tag{3}$$

where $t'_k = t_k + \Delta t = t_{k+1}$. We assume the data $\boldsymbol{x}(t_k) \in \mathbb{R}^{n\times 1}$ is a column vector, but any data can usually be manipulated to satisfy this requirement.

The process begins by arranging the data (3) into two matrices

$$\boldsymbol{X} = \begin{bmatrix} | & | & & | \\ \boldsymbol{x}(t_1) & \boldsymbol{x}(t_2) & \cdots & \boldsymbol{x}(t_m) \\ | & | & & | \end{bmatrix} \tag{4}$$

and

$$\boldsymbol{X}' = \begin{bmatrix} | & | & & | \\ \boldsymbol{x}(t'_1) & \boldsymbol{x}(t'_2) & \cdots & \boldsymbol{x}(t'_m) \\ | & | & & | \end{bmatrix} \tag{5}$$

$\boldsymbol{X} \in \mathbb{R}^{n\times m}$: the ensemble of snapshots of initial system states
$\boldsymbol{X}' \in \mathbb{R}^{n\times m}$: corresponding snapshots of final states (after a time Δt)

The DMD algorithm seeks the best-fit linear operator $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ that relates these two before/after snapshot matrices in time:

$$\boldsymbol{X}' \approx \boldsymbol{A}\boldsymbol{X} \tag{6}$$

$$\begin{bmatrix} | & | & & | \\ \boldsymbol{x}(t_1) & \boldsymbol{x}(t_2) & \cdots & \boldsymbol{x}(t_m) \\ | & | & & | \end{bmatrix} \approx \begin{bmatrix} | & | & & | \\ \boldsymbol{A}\boldsymbol{x}(t'_1) & \boldsymbol{A}\boldsymbol{x}(t'_2) & \cdots & \boldsymbol{A}\boldsymbol{x}(t'_m) \\ | & | & & | \end{bmatrix} \tag{7}$$

Optimization problem: find the best-fit operator $\boldsymbol{A}$ that minimizes the difference between the actual snapshot matrix $\boldsymbol{X}'$ and the predicted snapshot matrix $\boldsymbol{A}\boldsymbol{X}$ according to the *Frobenius norm*:

$$\boldsymbol{A}^* = \underset{\boldsymbol{A}}{\operatorname{argmin}} ||\boldsymbol{X}' - \boldsymbol{A}\boldsymbol{X}||_F \,. \tag{8}$$

Frobenius Norm

The Frobenius norm applies to the set of square matrices. Suppose $\boldsymbol{A}$ is a $n \times n$ matrix then the Frobenius norm of $\boldsymbol{A}$ is

$$||\boldsymbol{A}||_F = \sqrt{\sum_{i,j=1}^{n} |a_{ij}|^2} \, . \tag{9}$$

That is, it is the square root of the sum of squared matrix elements. For example, if

$$\boldsymbol{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \tag{10}$$

then (also in MATLAB, `n = norm(X,"fro")`)

$$||\boldsymbol{A}||_F = \sqrt{1 + 2^2 + 3^2 + 4^2} = \sqrt{30} \tag{11}$$

Pseudoinverse Solution: Accurate but Expensive

We state here (without proof) that matrix which minimizes (8) is

$$\boldsymbol{A}^* = \boldsymbol{X}'\boldsymbol{X}^+ \tag{12}$$

where $\boldsymbol{X}^+ = (\boldsymbol{X}^*\boldsymbol{X})^{-1}\boldsymbol{X}^*$ is the pseudoinverse of $\boldsymbol{X}$.

- Seems simple but not practical for large n
- Example: grayscale camera image in standard definition (SD) which has $n = 852 \times 480 = 408,960$ state elements.
- The $n \times n$ matrix $\boldsymbol{A}$ would then have $167,248,281,600$ elements
- Storing this matrix is not possible on my laptop, let alone computing the pseudoinverse.

DMD Solution: Tradeoff accuracy and computation

- Key Idea: Dimensionality reduction
- Dominant eigenvalues and eigenvectors use to approximate $\boldsymbol{A}$
- Implicitly assume the system has a small number of dominant "modes"
- Need some additional linear algebra machinery first:
  - Eigendecomposition: $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{-1}$
  - Singular value decomposition (SVD): $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$
  - Optimal rank-r matrix approximation

Eigendecomposition (Spectral Decomposition).

Suppose $\boldsymbol{A}$ is a square $n \times n$ matrix with n linearly independent eigenvectors $\boldsymbol{q}_i$ for $i = 1, \ldots, n$ and corresponding eigenvalues $\{\lambda_1, \ldots, \lambda_n\}$. The eigenvalue equation states that

$$\boldsymbol{A}\boldsymbol{q}_i = \lambda_i \boldsymbol{q}_i \tag{13}$$

Now, if we arrange the eigenvectors as columns of a $n \times n$ matrix $\boldsymbol{Q} = [\boldsymbol{q}_1^* \ \boldsymbol{q}_2^* \cdots \boldsymbol{q}_n^*]^*$ (where * denotes the complex conjugate) then

$$\begin{aligned} \boldsymbol{A}\boldsymbol{Q} &= \boldsymbol{A} \begin{bmatrix} | & | & & | \\ \boldsymbol{q}_1 & \boldsymbol{q}_2 & \cdots & \boldsymbol{q}_n \\ | & | & & | \end{bmatrix} \\ &= \begin{bmatrix} | & | & & | \\ \boldsymbol{A}\boldsymbol{q}_1 & \boldsymbol{A}\boldsymbol{q}_2 & \cdots & \boldsymbol{A}\boldsymbol{q}_n \\ | & | & & | \end{bmatrix} \end{aligned}$$

$$\boldsymbol{A}\boldsymbol{Q} = \begin{bmatrix} | & | & & | \\ \lambda_1 \boldsymbol{q}_1 & \lambda_2 \boldsymbol{q}_2 & \cdots & \lambda_n \boldsymbol{q}_n \\ | & | & & | \end{bmatrix}$$

which can be factored as

$$\boldsymbol{A}\boldsymbol{Q} = \begin{bmatrix} | & | & & | \\ \boldsymbol{q}_1 & \boldsymbol{q}_2 & \cdots & \boldsymbol{q}_n \\ | & | & & | \end{bmatrix} \underbrace{\begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix}}_{\boldsymbol{\Lambda}}$$

$$\implies \boldsymbol{A}\boldsymbol{Q} = \boldsymbol{Q}\boldsymbol{\Lambda}$$

where $\boldsymbol{\Lambda} = \mathrm{diag}([\lambda_1, \cdots, \lambda_n]^*)$.

Now right-multiply both sides of the equation by $\boldsymbol{Q}^{-1}$ to obtain

$$\boldsymbol{A}\boldsymbol{Q}\boldsymbol{Q}^{-1} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{-1} \tag{14}$$

$$\implies \boldsymbol{A} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{-1} \tag{15}$$

where $\boldsymbol{Q}$ is called the *modal matrix* of eigenvectors and $\boldsymbol{\Lambda}$ is the *spectral matrix* containing diagonal matrix of eigenvalues. The right-hand-side of (15) is called the *spectral decomposition* or *eigendecomposition* of $\boldsymbol{A}$.

Example: Eigendecomposition

```
A =

     1     2     3
     4     5     6
     7     8     9
```

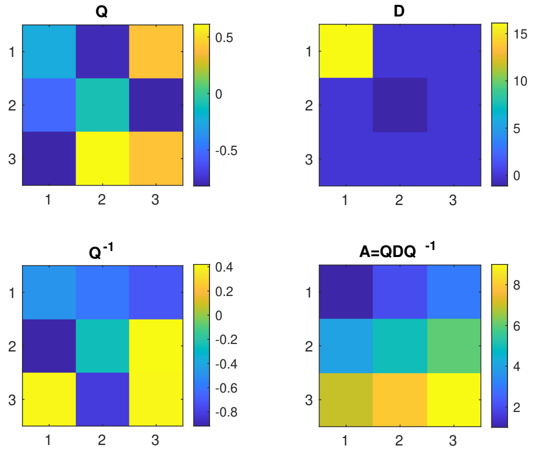
In MATLAB `[Q,D] = eig(A)` .
The eigenvectors (each column)

```
Q =

   -0.2320   -0.7858    0.4082
   -0.5253   -0.0868   -0.8165
   -0.8187    0.6123    0.4082
```

and eigenvalues on the diagonal

```
D =

   16.1168         0         0
         0   -1.1168         0
         0         0   -0.0000
```

Example (cont'd)



Singular Value Decomposition (SVD).

Another type of matrix decomposition is *singular value decomposition*. Any real $n \times m$ matrix $\boldsymbol{X}$ can be decomposed as

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^* \tag{16}$$

where

- $\boldsymbol{U}$ is a $n \times n$ matrix whose columns are orthonormal (i.e., $\boldsymbol{U}^*\boldsymbol{U} = \mathbf{1}_{n\times n}$).
- $\boldsymbol{V}$ is a $m \times m$ matrix whose columns are orthonormal (i.e., $\boldsymbol{V}^*\boldsymbol{V} = \mathbf{1}_{m\times m}$).
- $\boldsymbol{\Sigma}$ is a $n \times m$ matrix containing the $r = \min(n, m)$ *singular values* $\sigma_i \geq 0$ on the main diagonal and zeros elsewhere.

Note that because of their orthonormal properties:

- $\boldsymbol{U}$ and $\boldsymbol{V}$ are called *unitary* matrices.
- The columns of $\boldsymbol{U}$ are called the *left singular vectors* and the columns of $\boldsymbol{V}$ are called the *right singular vectors*.

Recall that eigenvalues/eigenvectors of a matrix $\boldsymbol{A}$ satisfy

$$\boldsymbol{A}\boldsymbol{q} = \lambda \boldsymbol{q} \ . \tag{17}$$

Similarly, the singular values and singular vectors of a matrix $\boldsymbol{A}$ are a triplet $(\sigma, \boldsymbol{u}, \boldsymbol{v})$ that satisfy:

$$\boldsymbol{A}\boldsymbol{v} = \sigma \boldsymbol{u} \tag{18}$$

$$\boldsymbol{A}^* \boldsymbol{u} = \sigma \boldsymbol{v} \tag{19}$$

- Eigenvalues are the characteristic values of a square $n \times n$ matrix that map vectors from one vector space onto itself
- Singular values are important for non-square matrices (e.g., size $n \times m$).
- They map a m dimensional vector space onto a n dimensional one
- Singular values relate to the distance between a matrix and the set of singular (i.e., non-invertible) matrices.

Since $\mathbf{\Sigma}$ is a $n \times m$ matrix with the r singular values on the diagonal and zeros elsewhere,

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \ddots & 0 & \cdots & 0 \\ 0 & 0 & 0 & \sigma_r & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & \vdots & \vdots & \vdots & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \tag{20}$$

it can be rewritten as

$$\mathbf{\Sigma} = \begin{bmatrix} \hat{\mathbf{\Sigma}} & \mathbf{0} \end{bmatrix} \tag{21}$$

where $\hat{\mathbf{\Sigma}} \in \mathbb{R}^{n \times r}$ assuming $m \geq n$ (i.e., there are more columns than rows $\mathbf{\Sigma}$ has at most n nonzero elements in the first n columns).

It is convention to order the singular values in $\hat{\mathbf{\Sigma}}$ from largest to smallest and adjust the singular vectors appropriately. If we also partition $\boldsymbol{V} = [\boldsymbol{V}_1; \boldsymbol{V}_2]$ then we can re-write the

$$
\begin{aligned}
\boldsymbol{X} &= \boldsymbol{U}\mathbf{\Sigma}\boldsymbol{V}^* && (22)\\
&= \boldsymbol{U}\begin{bmatrix} \hat{\mathbf{\Sigma}} & \mathbf{0} \end{bmatrix}\begin{bmatrix} \boldsymbol{V}_1 \\ \boldsymbol{V}_2 \end{bmatrix} && (23)\\
&= \begin{bmatrix} \boldsymbol{U}\hat{\mathbf{\Sigma}} & \mathbf{0} \end{bmatrix}\begin{bmatrix} \boldsymbol{V}_1 \\ \boldsymbol{V}_2 \end{bmatrix} && (24)\\
&= \boldsymbol{U}\hat{\mathbf{\Sigma}}\boldsymbol{V}_1 && (25)
\end{aligned}
$$

which is called the compact or economy form of the SVD (Note: no information is lost!)

Matrix Approximation.

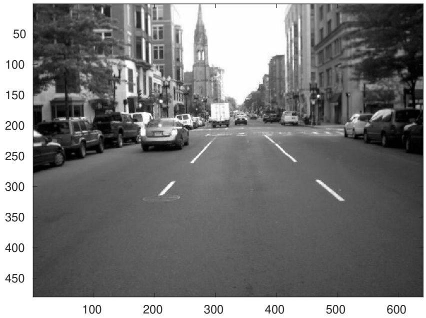
A theorem by Eckart-Young (1937) showed that "optimal rank-r" approximation to a matrix $\boldsymbol{X} \in \mathbb{R}^{n\times n}$ can be obtained by considering the first leading (largest) eigenvalues and corresponding eigenvectors in the SVD.

$$\underset{\tilde{\boldsymbol{X}} \text{ s.t. } \operatorname{rank}(\tilde{\boldsymbol{X}})=r}{\operatorname{argmin}} ||\boldsymbol{X} - \tilde{\boldsymbol{X}}||_F = \tilde{\boldsymbol{U}}\tilde{\Sigma}\tilde{\boldsymbol{V}}^* \tag{26}$$

where $\tilde{\boldsymbol{U}} \in \mathbb{R}^{n\times r}$ and $\tilde{\boldsymbol{V}} \in \mathbb{R}^{m\times r}$ denote the first leading r columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ in (16) and $\tilde{\boldsymbol{\Sigma}} \in \mathbb{R}^{r\times r}$ is the leading $r \times r$ subblock of $\boldsymbol{\Sigma}$ (or equivalently $\hat{\boldsymbol{\Sigma}} = [\tilde{\boldsymbol{\Sigma}}^* \ \mathbf{0}]^*$). r is user defined.

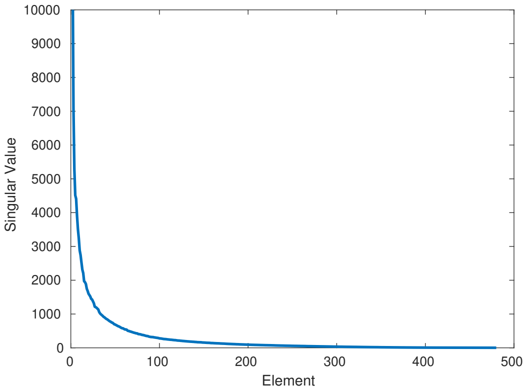
Example: City Street Image

```
A = imread('street1.jpg'); % grab image
A = double(rgb2gray(A)); % convert to grayscale / double format
imagesc(A) % plot
```



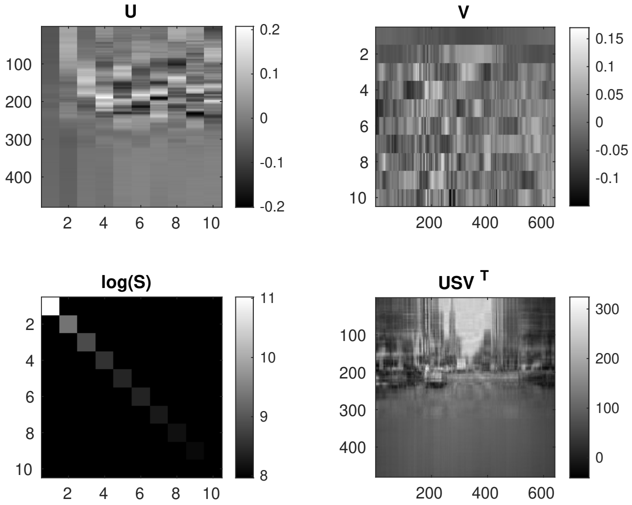
Example: Singular Values

```
[U,S,V] = svd(A); % A = U*S*V'
figure;
plot(diag(S),'linewidth',2)
```

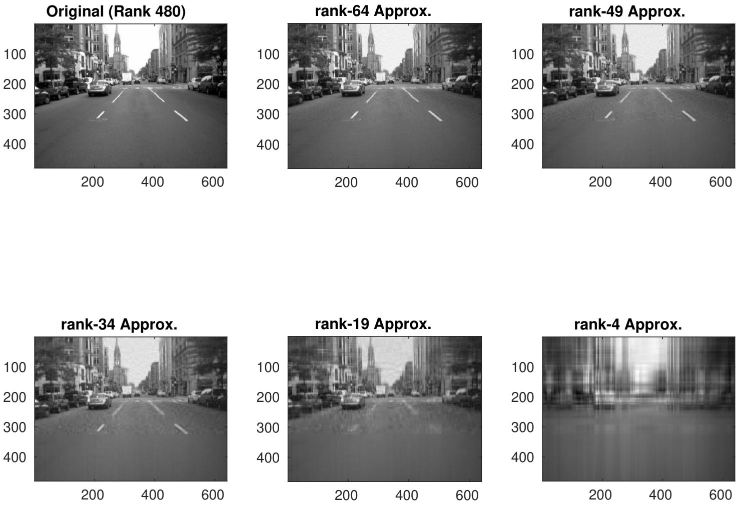


Example: rank-r approximation

```
r = 10;
Ut = U(:,1:r);
Vt = V(:,1:r);
St = S(1:r,1:r);
```



Example: rank-r approximation



DMD Algorithm

Step 1. Compute optimal rank-r approx. to the initial-time snapshot matrix (4):

$$\tilde{\boldsymbol{X}} \approx \tilde{\boldsymbol{U}}\tilde{\Sigma}\tilde{\boldsymbol{V}}^* \tag{27}$$

where r is a user-chosen approximation parameter

Step 2. Compute the $\boldsymbol{A}$ matrix in the linear system (6)

$$\begin{aligned} \boldsymbol{A} &= \boldsymbol{X}'\tilde{\boldsymbol{X}}^+ && (28)\\ &= \boldsymbol{X}'\tilde{\boldsymbol{V}}\tilde{\Sigma}^{-1}\tilde{\boldsymbol{U}}^* && (29) \end{aligned}$$

that minimizes the difference between the predicted final snapshot $\boldsymbol{A}\tilde{\boldsymbol{X}}$ (mapped from Step 1) and the actual final snapshot $\boldsymbol{X}'$

Since we are interested in only the r leading eigenvalues and eigenvectors of $\boldsymbol{A}$ we can consider the system evolving over a new (reduced-order state $\boldsymbol{z}$ that is related to the original state $\boldsymbol{x}$ by

$$\boldsymbol{x} = \tilde{\boldsymbol{U}}\boldsymbol{z} \implies \boldsymbol{z} = \tilde{\boldsymbol{U}}^*\boldsymbol{x} \tag{30}$$

which is a projection of $\boldsymbol{z}$ through $\tilde{\boldsymbol{U}}$ onto $\boldsymbol{x}$. We can now re-write the original system (1) in reduced order form as

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k \tag{31}$$

$$\tilde{\boldsymbol{U}}\boldsymbol{z}_{k+1} = \boldsymbol{A}\tilde{\boldsymbol{U}}\boldsymbol{z}_k \tag{32}$$

then pre-multiply both sides by $\tilde{\boldsymbol{U}}^*$ and use the unitary property of $\tilde{\boldsymbol{U}}$:

$$\tilde{\boldsymbol{U}}^*\tilde{\boldsymbol{U}}\boldsymbol{z}_{k+1} = \tilde{\boldsymbol{U}}^*\boldsymbol{A}\tilde{\boldsymbol{U}}\boldsymbol{z}_k \tag{33}$$

$$\boldsymbol{z}_{k+1} = \underbrace{\tilde{\boldsymbol{U}}^*\boldsymbol{A}\tilde{\boldsymbol{U}}}_{=\tilde{\boldsymbol{A}}}\boldsymbol{z}_k \tag{34}$$

$$\implies \boldsymbol{z}_{k+1} = \tilde{\boldsymbol{A}}\boldsymbol{z}_k \tag{35}$$

The z coordinates are the amplitudes of each mode that sum together to form the response and are all we need to model the system.
Step 3 (Optional: Eigenmode analysis). The matrix $\tilde{\boldsymbol{A}} \in \mathbb{R}^{r\times r}$ can be simplified as

$$\begin{aligned} \tilde{\boldsymbol{A}} &= \tilde{\boldsymbol{U}}^* \boldsymbol{A} \tilde{\boldsymbol{U}} && (36) \\ &= \tilde{\boldsymbol{U}}^* (\tilde{\boldsymbol{X}}' \tilde{\boldsymbol{V}} \tilde{\Sigma}^{-1} \tilde{\boldsymbol{U}}^*) \tilde{\boldsymbol{U}} && (37) \\ \implies \tilde{\boldsymbol{A}} &= \tilde{\boldsymbol{U}}^* \tilde{\boldsymbol{X}}' \tilde{\boldsymbol{V}} \tilde{\Sigma}^{-1} && (38) \end{aligned}$$

Compute the spectral decomposition of the reduced-order matrix $\tilde{\boldsymbol{A}}$

$$\tilde{\boldsymbol{A}} = \boldsymbol{Q} \tilde{\boldsymbol{\Lambda}} \boldsymbol{Q}^{-1} \qquad (39)$$

$\tilde{\boldsymbol{\Lambda}}$ is the diagonal matrix of eigenvalues for $\tilde{\boldsymbol{A}}$ and $\boldsymbol{Q}$ is the eigenvector matrix.

Also consider the spectral decomposition of the full matrix $\boldsymbol{A}$:

$$\boldsymbol{A} = \boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{\Phi}^{-1} \tag{40}$$

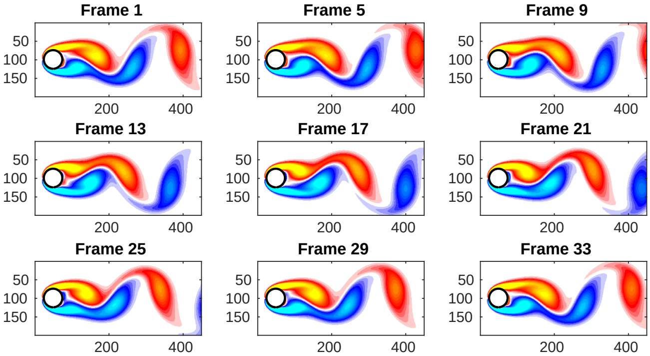
where $\boldsymbol{\Lambda}$ is the diagonal matrix of eigenvalues for $\boldsymbol{A}$ and $\boldsymbol{\Phi}$ is the eigenvector matrix. The high-dimensional eigenvector matrix $\boldsymbol{\Phi}$ can be obtained from the low-dimensional eigenvector matrix $\boldsymbol{Q}$ in a manner analogous to (29) as:

$$\boldsymbol{\Phi} = \tilde{\boldsymbol{X}}'\tilde{\boldsymbol{V}}\tilde{\Sigma}^{-1}\boldsymbol{Q} \tag{41}$$

The above eigenmodes are for the original system x coordinates.

Example: Flow over a Cylinder (Adapted from Brunton/Kutz)

Dataset: 151 snapshots of data. Each snapshot is an image (449 x 199 pixels) and is reshaped to be a long column vector. Each pixel corresponds to the vorticity Γ. Video: `play_Cylinder.m`

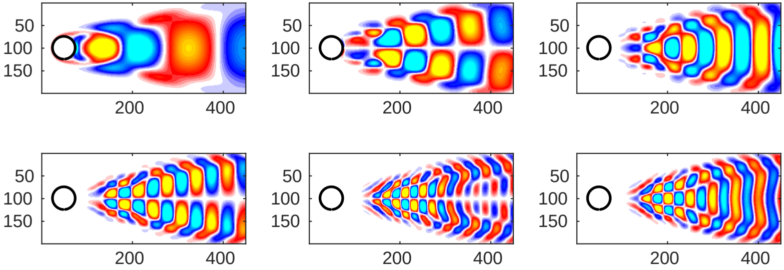


In this example the DMD algorithm is applied using just a few lines of code:

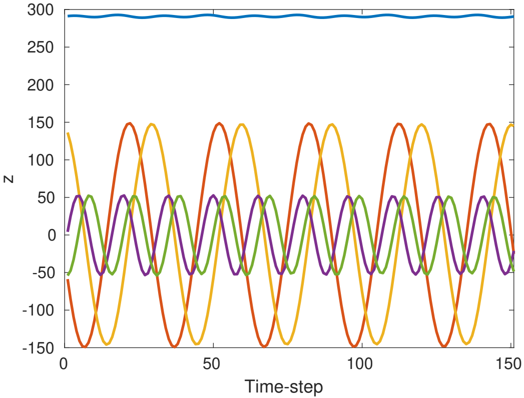
```
load CYLINDER_ALL.mat; % loads data
X = VORTALL(:,1:end-1); % creates initial snapshots
X2 = VORTALL(:,2:end); % creates final snapshots
[U,S,V] = svd(X,'econ'); % (Step 1) singular value decomposition
r = 21;  % truncate at 21 modes
Ur = U(:,1:r);  Sr = S(1:r,1:r); Vr = V(:,1:r);
Atilde = Ur'*X2*Vr*inv(Sr); % (Step 2)  reduced order system
zhist = zeros(r,M);
zhist(:,1) = Ur'*x1;
for k = 2:M % simulate linear z dynamics
    zhist(:,k) = Atilde*zhist(:,k-1);
end
xhist = Ur*zhist; % project back to x coords
```

The matrix `Atilde` computed above corresponds to the reduced order linear system. The spectral decomposition of this matrix reveals the dominant modes that describe the flow. A select number of these modes are plotted below.

```
[Q,eigs] = eig(Atilde); % eigendecomp
Phi = X2*V*inv(S)*; % eigenvectors of   original system
```



$r = 5$ time evolution of z



Full solution: `play_modes.m`

References

- Murphy: Sec. 7.5
- Brunton and Kutz: Secs. 1.1, 1.2, 7.2