DMD and Model Discovery

Echo Liu

May 6, 2020

https://github.com/EchoRLiu/DMD-Model-Discovery

Abstract

Dynamical system is a good framework explaining how certain quantity changes with time. Typically, when dealing with such systems, we hope to explain the behaviour and also make predictions for the future. However, normally it would not be possible to directly observe certain variables, i.e. latent variables, and the systems are often non-linear and high-dimensional. Dynamic mode decomposition (DMD) is a great way to deal with such problems. Based on singular value decomposition (SVD), DMD is able to provide dimension reduction, and also temporal data compared to SVD. With time-embedding, it's possible to reconstruct a linear system and discover latent variables. Two datasets presented here are used to demonstrate how we can use DMD to explain and conduct forecasting. KL-divergence, AIC and BIC are also used to check the model behavior.

1 Introduction and Overview

Here, we illustrate how DMD modes are constructed to reconstruct the data and make predictions for the future.

1.1 Problem Description

Canadian lynx and snowshoe hare population data shown in figure 1 is a good representation of Predator-Prey model. We are only observing two population variables, and if there is any latent variables influencing the system, we are not directly observing them. Another information from the plotting is that the system is clearly non-linear.

Another dataset is about Belousov-Zhabotinsky phenomenon. Figure 2 shows a snapshot of what the data represents. We also hope to reconstruct this and make future prediction.

1.2 General Approach

Two versions of DMD are used here first. To better fit a model for the population data, we can use interpolation to obtain more data points. For a simple DMD, we just need to consider a dynamic system such that $dy/dx = Ax$. For a length of t, we pick first $t-1$ columns and columns from second to last as x_1, x_2, and based on the SVD result on rank information, we conduct dimension truncation for u, s, v, based on which, we are able to get A and Φ, DMD modes. Using DMD modes and initial conditions, we are able to reconstruct the data. For time-embedding DMD, we would be needing Hankel matrix, which contains time-shifted data. We then conduct similar DMD steps to reconstruct the data or make prediction.

Since the population data is related to predator-preyer model as

$$x' = (b - py)x, y' = (rx - d)y \tag{1}$$

we can also consider building models through model discovery: we fit the differential equations with a library matrix containing possible variables

$$[variable\ library\ matrix]xi = Data' \tag{2}$$

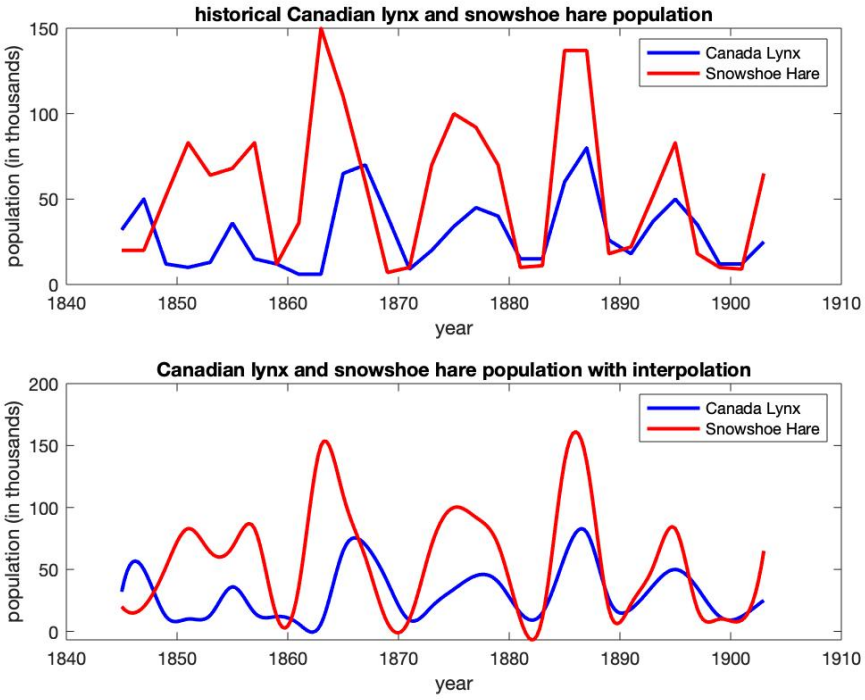

Figure 1: Population data, original and interpolated

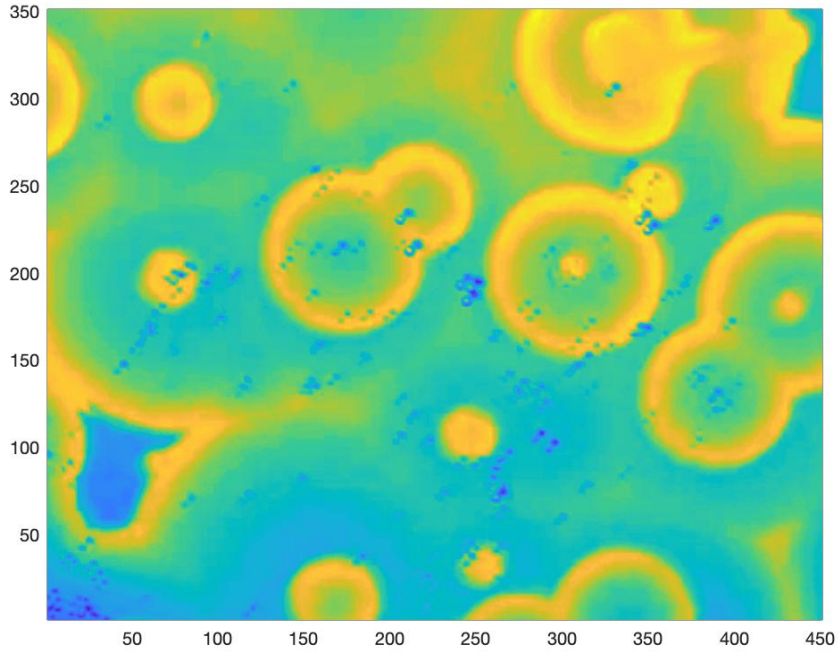

Figure 2: Snapshot of BZ

which could be solved with back-slash or lasso to promote sparsity. We then can use this model to reconstruct our data or even make prediction.

For the population data, to compare which model behaves better, we would be using KL-divergence, AIC, BIC to evaluate those models.

For the BZ data set, we would be using the same DMD and embedding approach.

2 Theoretical Background

As from Brunton and Kutz [1], DMD can be used to approximate nonlinear dynamic system, such as $\frac{dx}{dt} = f(x, t, u)$, with linear system, $\frac{dx}{dt} = Ax$, with which we can get

$$x = ve^{\lambda t}, \lambda v = Av \tag{3}$$

with the eigenvalue and eigenfunction, we can get the result

$$\sum_{j=1}^{n} b_j \Phi_j e^{\lambda_j t} \tag{4}$$

This gives us the idea for building DMD. Given data of measurements $\mathcal{X}_\infty, \mathcal{X}_\in$, we can conduct SVD on $\mathcal{X}_\infty$ to get U, Σ, V^*, which we would take r-rank truncation on. Then we can get

$$\begin{aligned} \tilde{A} &= U^* \mathcal{X}_2 V \Sigma^{-1} \\ \Phi &= \mathcal{X}' V \Sigma^{-1} \end{aligned} \tag{5}$$

where W is the eigenvectors, and finally we have

$$\sum_{k=1}^{r} \Phi_k e^{w_k t} b_k \tag{6}$$

One thing we need to be aware is that the variables we observe might not directly influence the system, meaning we need to find latent variables to help us better explain the system. Time-embedding DMD can be useful on this. Through time-shifted data, we are able to transfer the non-linear system to a system that can be constructed with sine and cosine series, which is easier to handle.

Model discovery is also a good way to help us understand the dynamics of the data. Observing certain related variables, we can construct a library matrix A containing them, and solve a simple $Ax = b$ system, either using built in back-slash or lasso to promote sparsity.

Here we use accuracy to evaluate how good a model is to label each image in the testing dataset. Some other ways to select better model are based on Kullback-Liebler Divergence, Akaike Information Criteria and Bayesian Information Criteria. By checking KL Divergence as defined in equation 7, we can see which model can produce closer result to the actual data. AIC and BIC, as in equation 8, check the log likelihood of a model with different ways to penalize on terms in the model.

$$I(f, g) = \int f(A, \beta) log[\frac{f(A, \beta)}{g(A, \mu)}] dA \tag{7}$$

$$\begin{aligned} AIC &= 2K - 2log[L(\hat{\mu}|x)] \\ BIC &= log(n)K - 2log[L(\hat{\mu}|x)] \end{aligned} \tag{8}$$

3 Algorithm Implementation and Development

The general approach is implemented in the following way.

1. population data is loaded into `cl2` and `sh2`, which are then being interpolated to get more data points and stored in `cl` and `sh`, which can be combined into `pop`. This can also gives us the histogram of the data to help us analyzing KL-divergence later;

2. conduct SVD on `pop` to get an idea of how the rank would look like for this data;

3. use algorithm 1 to reconstruct the data, get the histogram of the reconstructed data for later model analysis;

Algorithm 1: Dynamical Mode Decomposition

conduct SVD and r-rank truncation on the results U, S, V
get Atilde as $U' * x_2 * V/S$
get [W, D] = eig(Atilde)
get DMD modes as $x_2 * V/S * W$
get $\omega = log(diag(D))/dt$
using initial conditions u_0 to get $y0 = Phi\ u0$
for $i = 1 : length(t)$ **do**
  update the result as u modes $y0. * exp(omega * t(i)))$
end for
obtain final result $u_dmd = Phi * umodes$

4 Computational Results

1. Figure 3 shows the SVD result on population directly, we can see the first mode of v mimics the dynamics of the population data. The third subplot in figure **??** shows the two modes of Φ, we can see one of them is very close to the result in figure 3, while another one is not.

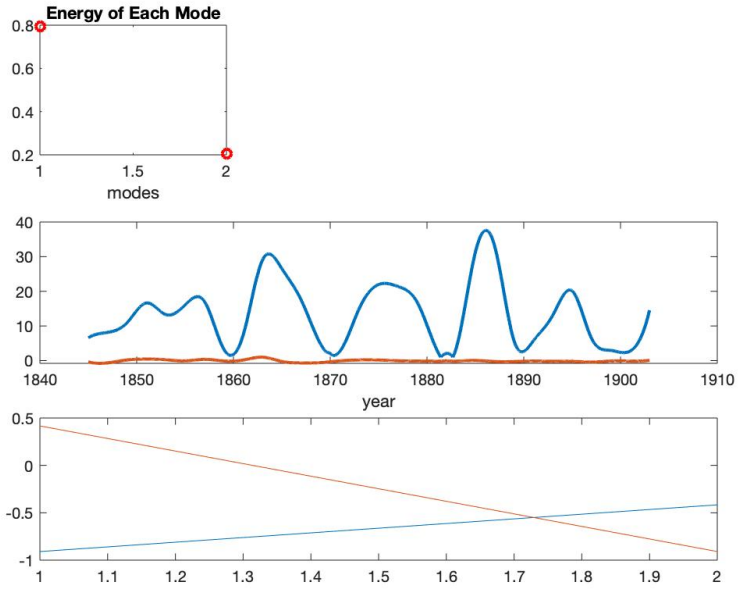

Figure 3: SVD of population data

Figure 5 shows the result through DMD and the prediction. This is clearly not a very model explaining the data.

2. with time-embedding, the rank grows, but through figure 6, we can see the dynamic system is transferring to a linear system.

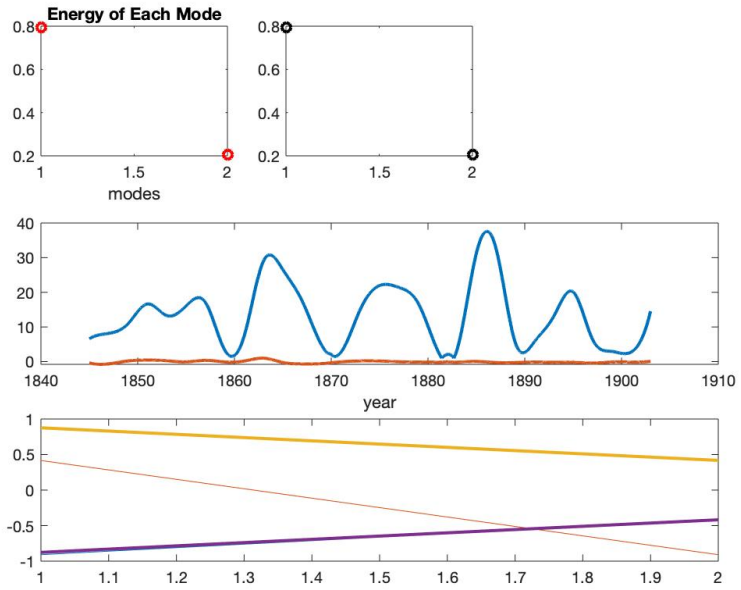

Figure 4: SVD of population data

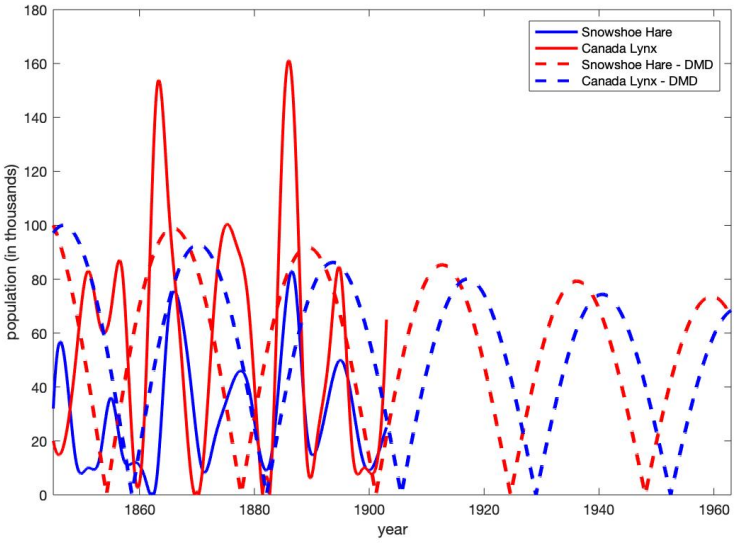

Figure 5: population data reconstruction and prediction- DMD

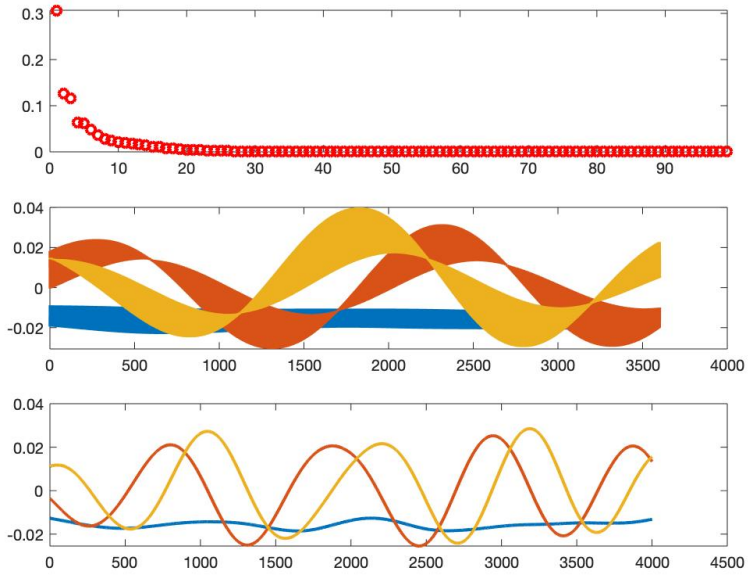

Figure 6: SVD on Hankel matrix - DMD time embedding

Figure 7 shows the result through DMD with time-embedding, the result seems to grow with time, the opposite of the result we got from the last part, which does not seem to capture the dynamics of the data either.

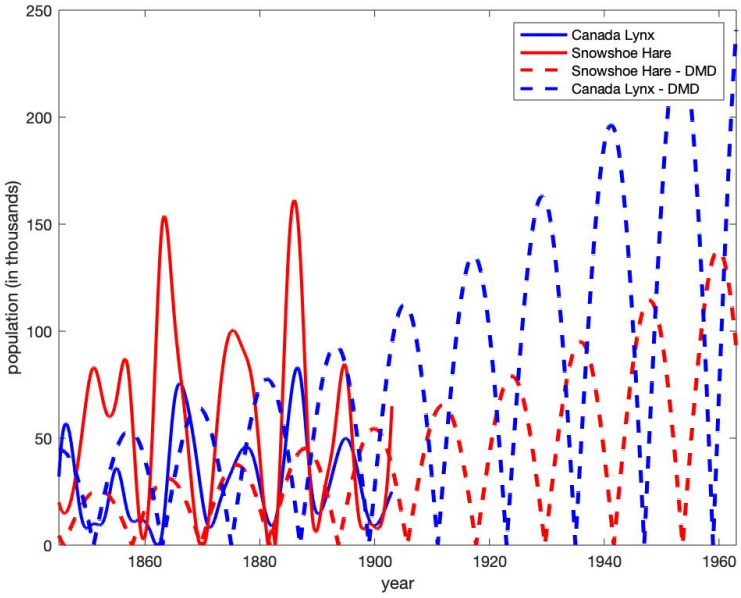

Figure 7: population data reconstruction and prediction-DMD time embedding

3. through building model discovery, using back-slash, we can rebuild the model as

$$x' = .28x - .29y, y' = .1x - .29y \tag{9}$$

4. using lasso, we can rebuild one version of the model as

$$x' = .1521x - .4129y, y' = .004xy - .09078y \tag{10}$$

another as

$$x' = .1067x - .4634y, y' = .01429x + .0002432x^2 + .001657xy; \tag{11}$$

5. doing the same thing for BZ dataset, we actually get a better result. Figure 8 and figure 9 shows the comparison between the model and the real data.

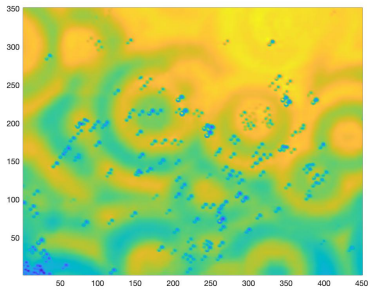

Figure 8: DMD result of BZ

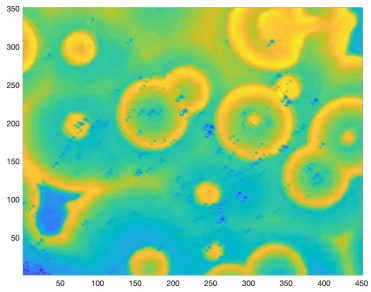

Figure 9: Visualization for BZ snapshot

5 Summary and Conclusions

Through these two examples, we can see how we can use DMD to help reconstruct non-linear dynamical systems and also time-embedding to discover latent variables. However, in the first example, DMD does not seem to give a very good result, while in the second example, DMD gives a reasonable result. For different datasets, we might need to do further digging before simply applying the DMD models.

References

[1] Steven L. Brunton and J. Nathan Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019.

Appendix A MATLAB Code

```
close all; clear all; clc

cl2 = [32 50 12 10 13 36 15 12 6 6 65 70 40 9 20 34 45 40 15 15 60 80 26 18 37 50 35 12 12
sh2 = [20 20 52 83 64 68 83 12 36 150 110 60 7 10 70 100 92 70 10 11 137 137 18 22 52 83 18
t2 = 1845:2:1903;

t = 1845:.01:1903;
cl = abs(spline(t2, cl2, t));
sh = abs(spline(t2, sh2, t));
pop = [sh; cl];

f = hist(pop, t);

figure(1)
subplot(2,1,1), plot(t2, cl2, 'b-', t2, sh2, 'r-', 'Linewidth', [2]),
```

```
xlabel('year'), ylabel('population (in thousands)'),
legend('Canada Lynx', 'Snowshoe Hare'),
title('historical Canadian lynx and snowshoe hare population');
subplot(2,1,2), plot(t, cl, 'b-', t, sh, 'r-', 'Linewidth', [2]),
xlabel('year'), ylabel('population (in thousands)'),
legend('Canada Lynx', 'Snowshoe Hare'),
title('Canadian lynx and snowshoe hare population with interpolation');

%%

% PCA decomposition.
[u,s,v] = svd(pop);

figure(2);
subplot(3,3,1)
plot(diag(s)/sum(diag(s)), 'ro', 'LineWidth', [2]), xlabel('modes'),
title('Energy of Each Mode');

subplot(3,1,2), plot(t,v(:,1)/max(v(:,1)), t,v(:,2)/max(v(:,2)),'LineWidth',[2]),
xlabel('year');
% We actually only need the first column.

subplot(3,1,3), plot(u(:,1:2));

%%

% Develop DMD model to forecast future population states.

% DMD J-Tu decomposition.
X1 = pop(:, 1:(end-1)); X2 = pop(:, 2:end);
r = 2;
[U2, S2, V2] = svd(X1); U = U2(:,1:r); S = S2(1:r,1:r); V = V2(:,1:r);

Atilde = U'*X2*V/S;
[W,D] = eig(Atilde);
Phi = X2*(V/S)*W; % DMD modes.

dt = t(2)-t(1);
mu = diag(D);
omega = log(mu)/dt;

figure(2)
subplot(3,3,2), plot(diag(S2)/sum(diag(S2)), 'ko', 'Linewidth', [2]);
subplot(3,1,3), hold on, plot(1:1:2,real(-Phi(:,1)), 'Linewidth', [2]);
subplot(3,1,3), plot(1:1:2,real(Phi(:,2)), 'Linewidth', [2]);

u0 = pop(:,1);
y0 = Phi\u0;
tt = linspace(1845, 1963, 5801); % going twice of the time into the future.
u_modes = zeros(r, length(tt));
for i = 1:length(tt)
    u_modes(:, i) = (y0.*exp(omega*tt(i)));
end
```

```
u_dmd = Phi*u_modes;
%%

g1 = hist(100*abs(real(u_dmd(1:2,1:length(t)))),t);

%%

figure(3)
plot(t, cl, 'b-', t, sh, 'r-', 'Linewidth', [2]), hold on,
plot(tt, 100*abs(real(u_dmd(1,:))),'r--', tt, 100*abs(real(u_dmd(2,:))), 'b--', 'Linewidth
legend('Canada Lynx', 'Snowshoe Hare', 'Snowshoe Hare - DMD','Canada Lynx - DMD'),
xlabel('year'), ylabel('population (in thousands)'),
xlim([1845 1963]);

%%

x1 = pop(1,:); x2 = pop(2,:);

H1 = [x1(1:5700)
     x2(1:5700)
     x1(2:5701)
     x2(2:5701)
     x1(3:5702)
     x2(3:5702)
     x1(4:5703)
     x2(4:5703)
     x1(5:5704)
     x2(5:5704)
     x1(6:5705)
     x2(6:5705)];

H2 = [x1(1:5700)
     x2(1:5700)
     x1(2:5701)
     x2(2:5701)
     x1(3:5702)
     x2(3:5702)
     x1(4:5703)
     x2(4:5703)
     x1(5:5704)
     x2(5:5704)
     x1(6:5705)
     x2(6:5705)
     x1(7:5706)
     x2(7:5706)
     x1(8:5707)
     x2(8:5707)
     x1(9:5708)
     x2(9:5708)
     x1(10:5709)
     x2(10:5709)];

figure(4)
[u,s,v] = svd(H1,'econ');
```

```
subplot(2,1,1), plot(diag(s)/sum(diag(s)), 'ro', 'Linewidth', [2]);
figure(5)
subplot(2,1,1), plot(u(:,1:3), 'Linewidth',[2])
subplot(2,1,2), plot(v(:,1:3), 'Linewidth',[2])
figure(7), subplot(2,1,1), plot(v(:,1:2), 'Linewidth', [2]); hold on

figure(4)
[u,s,v]=svd(H2,'econ');
subplot(2,1,2), plot(diag(s)/sum(diag(s)), 'ro', 'Linewidth', [2]);
figure(6)
subplot(2,1,1), plot(u(:,1:3), 'Linewidth',[2])
subplot(2,1,2), plot(v(:,1:3), 'Linewidth',[2])
figure(7), subplot(2,1,2), plot(v(:,1:2)), legend('H2-1','H2-2');
% Time embedding does not change that much here. But it is non-linear.

%%

H3 = [];
for j = 1:1800
    H3 = [H3; pop(:,j:(4000+j))];
end

%%

figure(8)
[u,s,v] = svd(H3,'econ');
subplot(3,1,1), plot(diag(s)/sum(diag(s)), 'ro', 'Linewidth', [2]);
subplot(3,1,2), plot(u(:,1:3), 'Linewidth',[2])
subplot(3,1,3), plot(v(:,1:3), 'Linewidth',[2])

%%

H1 = [];
for j = 1:1800
    H1 = [H1; pop(:,j:(4000+j))];
end
H2 = [];
for j = 2:1801
    H2 = [H2; pop(:,j:(4000+j))];
end

%%

r = 10; [u,s,v]=svd(H1, 'econ');
U = u(:,1:r); S = s(1:r,1:r); V = v(:,1:r);

Atilde = U'*H2*V/S;
[W,D] = eig(Atilde);
Phi = H2*(V/S)*W; % DMD modes.

dt = t(2)-t(1);
mu = diag(D);
omega = log(mu)/dt;
```

```
u0 = H1(:,1);
y0 = Phi\u0;

tt = linspace(1845, 1963, 5801); % going twice of the time into the future.
u_modes = zeros(r, length(tt));
for i = 1:length(tt)
    u_modes(:, i) = (y0.*exp(omega*tt(i)));
end

u_dmd = Phi*u_modes;
g2 = hist(abs(real(u_dmd(1:2,1:length(t))))/10^12, t);

%%

figure(9)

plot(t, cl, 'b-', t, sh, 'r-', 'Linewidth', [2]), hold on,
plot(tt, abs(real(u_dmd(1,:)))/10^12,'r--', tt, abs(real(u_dmd(2,:)))/10^12, 'b--', 'Linew
legend('Canada Lynx', 'Snowshoe Hare', 'Snowshoe Hare - DMD','Canada Lynx - DMD'),
xlabel('year'), ylabel('population (in thousands)'),
xlim([1845 1963]);


%%

n = length(t);
xdot = zeros(2, n-2);
for i = 1:2
    for j = 2:n-1
        xdot(i, j-1) = (pop(i,j+1)-pop(i, j-1))/(2*dt);
    end
end

xs = pop(1,2:n-1).';
ys = pop(2,2:n-1).';

A = [xs xs.^2 xs.^3 ys ys.^2 ys.^3 ys.*xs ys.*xs.^2 ys.^2.*xs];

xi = A\xdot(1,:).';
yi = A\xdot(2,:).';

figure(10)
subplot(2,1,1), bar(xi),
subplot(2,1,2), bar(yi);

%%

dx = @(xs, ys) .28*xs - .29*ys;
dy = @(xs, ys) .1*xs - .29*ys;

pop1 = zeros(2, length(tt));
pop1(1:2,1) = pop(1:2,1);

for j = 2:length(tt)
```

```
    pop1(1, j) = pop1(1, j-1) + dx(pop1(1, j-1), pop1(2, j-1));
    pop1(2, j) = pop1(2, j-1) + dy(pop1(1, j-1), pop1(2, j-1));
end

g3 = hist(real(pop1(:,1:length(t))), t);

figure(10)

plot(t, cl, 'b-', t, sh, 'r-', 'Linewidth', [2]), hold on,
plot(tt, abs(real(pop1(1,:))),'r--', tt, abs(real(pop1(2,:))), 'b--', 'Linewidth', [2]),
legend('Canada Lynx', 'Snowshoe Hare', 'Snowshoe Hare - DMD','Canada Lynx - DMD'),
xlabel('year'), ylabel('population (in thousands)'),
xlim([1845 1963]);


%%

xi = lasso(A, xdot(1,:).', 'Lambda', .1);
yi = lasso(A, xdot(2,:).', 'Lambda', .1);

figure(10)
subplot(2,1,1), bar(xi),
subplot(2,1,2), bar(yi);

%%

dx = @(xs, ys) .1521*xs - .4129*ys;
dy = @(xs, ys) .004*xs.*ys - .09078*ys;

pop1 = zeros(2, length(tt));
pop1(1:2,1) = pop(1:2,1);

for j = 2:length(tt)
    pop1(1, j) = pop1(1, j-1) + dx(pop1(1, j-1), pop1(2, j-1));
    pop1(2, j) = pop1(2, j-1) + dy(pop1(1, j-1), pop1(2, j-1));
end

g4 = hist(real(pop1(:,1:length(t))), t);

figure(10)

plot(t, cl, 'b-', t, sh, 'r-', 'Linewidth', [2]), hold on,
plot(tt, abs(real(pop1(1,:))),'r--', tt, abs(real(pop1(2,:))), 'b--', 'Linewidth', [2]),
legend('Canada Lynx', 'Snowshoe Hare', 'Snowshoe Hare - DMD','Canada Lynx - DMD'),
xlabel('year'), ylabel('population (in thousands)'),
xlim([1845 1963]);

%%

xi = lasso(A, xdot(1,:).', 'Lambda', .1, 'Alpha', .8);
yi = lasso(A, xdot(2,:).', 'Lambda', .1, 'Alpha', .8);

figure(10)
subplot(2,1,1), bar(xi),
```

```
subplot(2,1,2), bar(yi);

%%

dx = @(xs, ys) .1067*xs - .4634*ys;
dy = @(xs, ys) .01429*xs + .0002432*xs.^2 + .001657*xs.*ys;

pop1 = zeros(2, length(tt));
pop1(1:2,1) = pop(1:2,1);

for j = 2:length(tt)
    pop1(1, j) = pop1(1, j-1) + dx(pop1(1, j-1), pop1(2, j-1));
    pop1(2, j) = pop1(2, j-1) + dy(pop1(1, j-1), pop1(2, j-1));
end

g5 = hist(real(pop1(:,1:length(t))), t);

figure(10)

plot(t, cl, 'b-', t, sh, 'r-', 'Linewidth', [2]), hold on,
plot(tt, abs(real(pop1(1,:))),'r--', tt, abs(real(pop1(2,:))), 'b--', 'Linewidth', [2]),
legend('Canada Lynx', 'Snowshoe Hare', 'Snowshoe Hare - DMD','Canada Lynx - DMD'),
xlabel('year'), ylabel('population (in thousands)'),
xlim([1845 1963]);


%%

% KL-divergence.

% f = f/trapz(f);
% g1 = g1/trapz(g1); g2 = g2/trapz(g2); g3 = g3/trapz(g3);
% g4 = g4/trapz(g4); g5 = g5/trapz(g5);
%
%
% %%
% plot(t,f,t,g1,t,g2,t,g3,t,g4,t,g5,'Linewidth',[2]);




%%

close all; clear all; clc

bz = load("/Users/yuhongliu/Downloads/BZ.mat");
BZ_tensor = bz.BZ_tensor;

%%

% Visualize the data and reshape each snapshot into a vector.
[m,n,k]=size(BZ_tensor); % x vs y vs time data.
BZ_matrix = zeros(m*n, k);
```

```
figure(1);
%pcolor(BZ_tensor(:,:,1)), shading interp;
for j=1:k
    A=BZ_tensor(:,:,j);
    %pcolor(A), shading interp, pause(0.01);
    BZ_matrix(:,j) = reshape(A, m*n, 1);
end

%%

% Develop DMD model to forecast future population states.

% PCA decomposition.
[u,s,v] = svd(BZ_matrix,'econ');

%%

figure(2);
subplot(2,1,1)
plot(diag(s)/sum(diag(s)), 'ro', 'LineWidth', [2]), xlim([0 30]), xlabel('modes');
subplot(2,1,2), plot(v(:,1)/max(v(:,1)), 'LineWidth', [2]), hold on,
subplot(2,1,2), plot(v(:,2)/max(v(:,2)), 'LineWidth', [2]),
xlabel('t'), legend('1st_mode', '2nd_mode');
figure(3)
subplot(2,1,1), pcolor(reshape(u(:,1), m, n)), shading interp;
subplot(2,1,2), pcolor(reshape(u(:,2), m, n)), shading interp;

%%

% DMD J-Tu decomposition.
X1 = BZ_matrix(:, 1:(end-1)); X2 = BZ_matrix(:, 2:end);
r = 5; % Based on the first subplot we see.
[U2, S2, V2] = svd(X1, 'econ'); U = U2(:,1:r); S = S2(1:r,1:r); V = V2(:,1:r);

Atilde = U'*X2*V/S;
[W,D] = eig(Atilde);
Phi = X2*V/S*W;

mu = diag(D);
omega = log(mu);

u0 = BZ_matrix(:,1);
y0 = Phi\u0;
u_modes = zeros(r, k);
for i = 1:k
    u_modes(:, i) = (y0.*exp(omega*i));
end

u_dmd = Phi*u_modes;

%%

figure(4)
```

```
for j=1:k
    pcolor(reshape(real(u_dmd(:,j)), m, n)), shading interp, pause(0.01);
end

%%

% Time-Embedding.

H1 = [];
for j = 1:4
    H1 = [H1; BZ_matrix(:,j:(1195+j))];
end
H2 = [];
for j = 2:5
    H2 = [H2; BZ_matrix(:,j:(1195+j))];
end

%%

[u,s,v]=svd(H1, 'econ');

%%
r = 10;
U = u(:,1:r); S = s(1:r,1:r); V = v(:,1:r);

%%

Atilde = U'*H2*V/S;
[W,D] = eig(Atilde);
Phi = H2*(V/S)*W; % DMD modes.

dt = t(2)-t(1);
mu = diag(D);
omega = log(mu)/dt;

u0 = H1(:,1);
y0 = Phi\u0;

tt = linspace(1845, 1963, 5801); % going twice of the time into the future.
u_modes = zeros(r, length(tt));
for i = 1:length(tt)
    u_modes(:, i) = (y0.*exp(omega*tt(i)));
end

u_dmd = Phi*u_modes;
g2 = hist(abs(real(u_dmd(1:2,1:length(t))))/10^12, t);

%%

figure(9)

plot(t, cl, 'b-', t, sh, 'r-', 'Linewidth', [2]), hold on,
plot(tt, abs(real(u_dmd(1,:)))/10^12,'r--', tt, abs(real(u_dmd(2,:)))/10^12, 'b--', 'Linew
legend('Canada Lynx', 'Snowshoe Hare', 'Snowshoe Hare - DMD','Canada Lynx - DMD'),
```

```
xlabel('year'), ylabel('population (in thousands)'),
xlim([1845  1963]);
```