AMATH 563 Spring 2018 Homework 1
Model Discovery

Tun Sheng Tan

Abstract. This assignment focuses on sparse regression and information theory to build the best model based on the data at hand. Kullback-Leibler divergence and AIC-BIC are used to select models constructed via sparse regression. We apply these techniques to the Canadian lynx-hare population and Belousov-Zhabotinsky reaction and are able to extract a partial model with physically motivated terms.

1. Introduction. In the recent years, data-driven modeling have been the trend in understanding complex system. When dealing with such system, scientists often rely on conjecturing models from toy models. This method often leads to over-fitting of data and lack of predictive power. Therefore, in this assignment, I will be exploring some concepts and techniques used in data-driven model discovery. In section 2, I will introduce some concepts that are used in the assignment, namely sparse regression, Kullback-Leibler(KL) divergence, Akaike information criterion(BIC), Bayesian information criterion(BIC) and time-delayed embedding. In section 3, I will discuss the implementation of algorithms. In section 4, I will attempt to infer the dynamics from two real experimental dataset which have no known 'correct' solution.

2. Theoretical Background.

2.1. Sparse Identification. Given a collection of m time-series measurements, $\mathbf{a}, \mathbf{b}, ..., \mathbf{m}$, with n data points,

$$X = \begin{bmatrix} a(t_1) & b(t_1) & c(t_1) & \dots & m(t_1) \\ a(t_2) & b(t_2) & c(t_2) & \dots & m(t_2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a(t_n) & b(t_n) & c(t_n) & \dots & m(t_n) \end{bmatrix} \tag{2.1}$$

we can write the dynamic of the system in the form

$$\dot{X} = f(X) \tag{2.2}$$

where μ is some parameter for the system. Suppose we know the terms that are relevant for the dynamics, $\Theta(X)$, we can rewrite the expression as a product

$$\dot{X} = f(X) = \Theta(X)\Lambda \tag{2.3}$$

where $\Lambda = [\lambda_1, \lambda_2, ..., \lambda_n]$ are the coefficients for each term.

Suppose that we do not know exactly the correct dynamic, we can try to figure out the relevant by assuming that the underlying dynamics is governed by only a small number of terms. Then, if we have a large collection of terms in our dictionary

$$\Theta(X) = [\ \mathbb{I} \quad X \quad X^2 \quad ... \quad g(X) \quad ...\ ] \tag{2.4}$$

where $g(X)$ is any relevant function for the system, then we can cast this problem as finding the sparsest Λ that fits the data X [1].

$$\arg\min \|\dot{X} - \Theta(X)\Lambda\| + \lambda\|\Lambda\|_1 \tag{2.5}$$

2.2. Model Selection with Information Theory. How do we define the "best" model ? The error between a model and data can be reduced by increasing the complexity of the model. In the case of choosing between polynomial models, high order model has lower error. However, this approach will lead to over-fitting. This implies that the error (typically the residual sum of squares) is not the metric for comparing models.

Definition 2.1. *Suppose that P and Q are some probability distributions. The Kullback-Leibler divergence, D_{KL} is defined to be*

$$D_{KL}(P||Q) = \int p(x) \log \frac{p(x)}{q(x)} dx \tag{2.6}$$

From information theory, Kullback-Leibler divergence, D_{KL} is a measure of 'distance' between two probability distribution. This measure is closely related to the relative entropy. When $D_{KL} = 0$, distribution P is similar to that of Q. Thus, no information is lost when presenting distribution P with Q. So, in selecting the 'best' model, we are minimizing the information lost.

To compute D_{KL}, we need to know the truth P and the model Q. However, for most cases, we do not have full access to the truth P. Akaike showed that we can estimate the information lost by Q using Akaike information criterion.

Definition 2.2. *Akaike Information Criterion(AIC) Suppose that we have a statistical model Q of some data P. Let k be the number of predictors in Q and $\mathcal{L}(Q|P)$ be the maximum likelihood function for Q. Then AIC is defined to be*

$$AIC = 2k - 2\log \mathcal{L}(Q|P) \tag{2.7}$$

AIC penalizes higher complexity models and rewards better goodness of fit. For regression problem, the typical maximum likelihood measure is the sum of squares regression (SSR). SSR is is the sum of the squared differences between the prediction for each observation and the population mean.

Another information criterion is the Bayesian information criterion(BIC) which is formulated based on Bayesian statistics.

Definition 2.3. *Definition Akaike Information Criterion(BIC) Suppose that we have a statistical model Q of some data P. Let k be the number of predictors in Q, n be the number of observations and $\mathcal{L}(Q|P)$ be the maximum likelihood function for Q. Then AIC is defined to be*

$$BIC = k\log n - 2\log \mathcal{L}(Q|P) \tag{2.8}$$

BIC is closely related to AIC. The difference is that the penalty for BIC increases with the number of data points.

Algorithm 3.1 False Nearest Neighbor

```
Define N as the number of data points
Pick a optimal delayed time τ
Initialize number of false neighbors, C = 0
Initialize percentage of false neighbors, P = 1
Initialize the embedded dimension, D = 0
Choose a threshold distance for false neighbors R
Choose a threshold for P
while P > 0 do
  Increase D by 1
  Construct Y(t) := {x(t + τ), x(t + 2τ), . . . , x(t + Dτ)}
  for t ∈ {t_1, ..., t_N} do
    Find t* = arg min_{t*} ||Y(t) − Y(t*)||_2
    Compute r = |x(t)−x(t*)| / ||Y(t)−Y(t*)||_2
    if r > R then
      Increase C by 1
    end if
  end for
  Compute P = C/N
end while
return D
```

2.3. Time delayed Embedding. In most experiments, we do not know what are the observables of the effective dynamics for the system. Our measurements may represent only a partial set of observables required to describe the full dynamic. Whitney (1936) showed by embedding (experimental measurement), a mapping that takes an N-manifold to $2N+1$ Euclidean space, no two independent signals measured from a system (Euclidean space) can be mapped to the same state in the state space of the system [4]. In addition, Takens (1981) showed that with certain conditions satisfied, a time-delayed measurement of a generic signal is sufficient to embed the N-manifold [3]. Thus, one can measure a single quantity instead of $2N+1$ quantities to reconstruct the state space. We will try to infer the latent variables from our data using time-delayed coordinates. There are many algorithms to reconstruct the phase space, for example, false nearest neighbor and HAVOK analysis.

3. Algorithm Implementation and Development.

3.1. Preprocessing Data. Experimental measurements are discrete and non-smooth. When computing the derivatives of such data, the result will be highly unstable especially when the function is oscillating rapidly or the data points are sparse. Thus, we attempt to stabilize the procedure through **spline** and **smoothdata** functions which are available in the MATLAB toolbox [2].

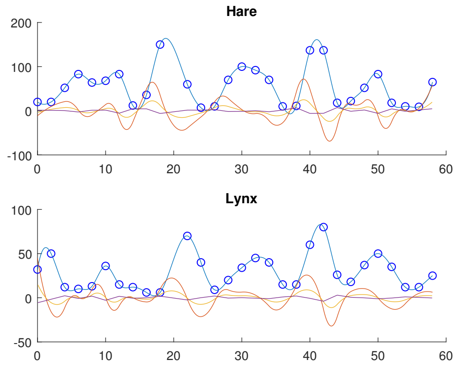

Figure 1. *Population of Canadian lynx and snowshoe hare. Blue dots presents the actual data point. Blue curve represents the spline fitted population. Red curve represents the first order derivative. Purple curve represents the second order derivative.*

3.2. Numerical Differentiation. The differentiation scheme used is the finite difference approximations. The two-point finite difference formula for a function $f(x)$ is given by

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} \tag{3.1}$$

where h is some small number.

3.3. Sparse Regression and Model Selection. The matlab toolbox contains implementation of lasso, elastic-net, pseudo-inverse and QR decomposition which are ready to solve (2.3). As for the model selection, we need to compute the KL divergence using probability distribution from the data and model. We use the normalized histogram of the data and model as our probability distribution.

For AIC and BIC, they are computed directly based on the definition (2.7) and (2.8) using SSR as the maximum likelihood function.

3.4. Time-delayed Embedding for Latent Variable Discovery. We use false nearest neighbor algorithm to estimate the embedded dimension shown in Algorithm 3.1 using the code written by Merve Kizilkaya. By performing a sweep for delayed time τ, we determine the lower and upper bound for the embedded dimension.

4. Computational Results.

4.1. Snowshoe hare and Canadian lynx population. We are looking at the historical dataset for snowshoe hare and Canadian lynx population from 1845 to 1903 shown in Figure 1. The two species are related by the predator-prey relation. It is believed that LotkaVolterra equations can be used to describe the such system.

We infer from the data the dynamics of the system via 10-fold cross validated Lasso regression and perform a sequential thresholding. The best model is selected via the lowest KL divergence score. The coefficients of the top four model for hare is shown in Figure 2. The KL divergence score for the best model is 0.0333. For a comparison, the AIC-BIC scores of the

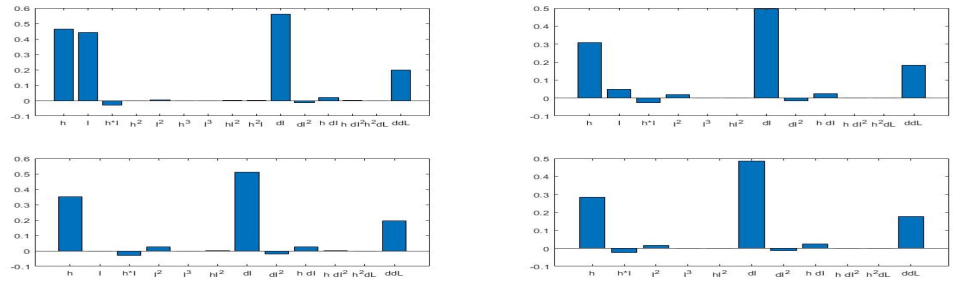

Figure 2. *Coefficients for the top four models for the hare dynamic. The best model is the one on the bottom right.*

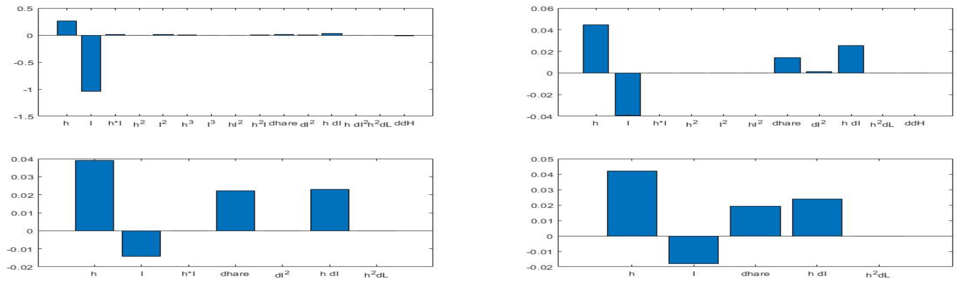

Figure 3. *Coefficients for the top four models for the lynx dynamic. The best model is the one on the bottom right.*

best four models (from top left to bottom right in Figure 2) for the hare dynamics are $AIC = 308.6640,\ 304.9710,\ 302.5643,\ 303.1904$ and $BIC = 327.8061,\ 321.3785, 317.6046,\ 318.2306$.

The coefficients of the top four model for lynx is shown in Figure 3. The KL divergence score for the best model is 0.0333. The AIC-BIC scores of the best four models (from top left to bottom right in Figure 3) for the hare dynamics are $AIC = 212.8857,\ 209.1990,\ 206.9705,\ 206.3612$ and $BIC = 233.3952,\ 218.7701,\ 213.8070,\ 213.1977$.

To summarize, the best model is given by

$$\frac{dH}{dt} = c_1 H + c_2 HL + c_3 L^2 + c_4 \frac{dL}{dt} + c_5 H (\frac{dL}{dt})^2 + c_6 H \frac{dL}{dt} + c_7 \frac{d^2 L}{dt^2} \tag{4.1}$$

$$\frac{dL}{dt} = c_1 H + c_2 L + c_3 \frac{dH}{dt} + c_4 H \frac{dL}{dt} \tag{4.2}$$

where c_i are the coefficients shown in Figure 2 and Figure 3. As shown in Figure 4, equation (4.2) is able to capture the dynamic of the data but equation (4.1) is not able to fully capture the dynamic of the hare population.

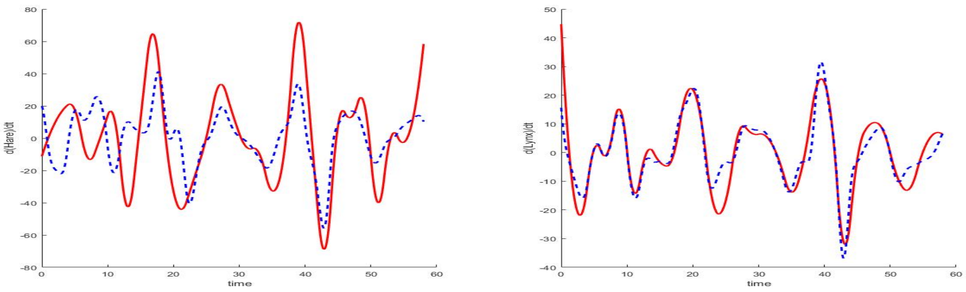

Figure 4. *Coefficients for the top four models for the lynx dynamic. The best model is the one on the bottom right.*

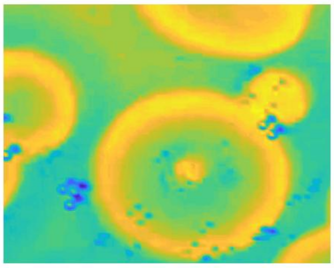

Figure 5. *A snapshot of BZ reaction.*

Next, we will try to time-delay embed the system to discovery any latent variables. By using the false nearest neighbor algorithm, we estimate that the dimension of the phase space for the system is between 2 and 5 latent variables.

4.2. Belousov-Zhabotinsky reaction. Belousov-Zhabotinsky (BZ) reaction describes a non-equilibrium oscillating chemical reactions which arise in macroscopic medium. The first reaction was discovered by Belousov for Ce^{3+}/Ce^{4+} catalyst in citric acid. BZ reaction generates a observable periodic propagation of concentric chemical waves. Here, we will be inferring the dynamics of BZ reaction from an experimental data shown in Figure 5 by taking two 1D slice along the X and Y axes for a ripple. The coefficients of the best four model is shown in Figure 6 which is obtained through Lasso. Due to memory limitation, only 3 cross-validations are performed. The KL divergence score for the best model is -0.0153. To summarize, the best model is

$$\frac{dU}{dt} = c_1 U_x + c_2 U_x^2 + c_3 U_{xx}^2 + c_4 U_x U_{xx} \tag{4.3}$$

where c_i are the coefficients shown in Figure 6.

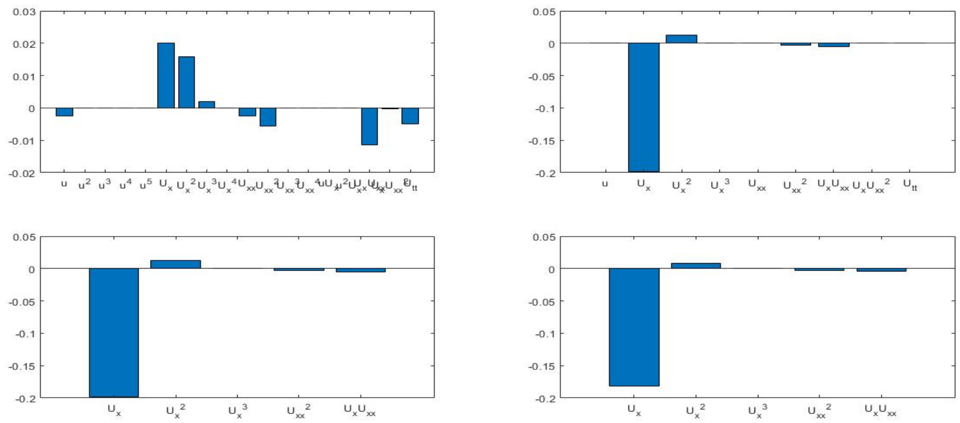

Figure 6. *Coefficients for four top models for BZ oscillation obtained via Lasso. The best model is on the bottom right.*

5. Summary and Conclusions. Interpreting the result from sparse regression works well with toy models like Lorentz system since the solution is known and the date is well defined. In actual application, noises and measurement error makes the regression complicated. We are able to obtain a simple model for the Canadian lynx population but the model for the hare is rather complex. Some terms obtained from the regression model are hard to justify physically (based on our limited knowledge). Similarly, we managed to identify the the dominant feature of BZ reaction (diffusion term U_x). However, with the current analysis it is not sufficient to make any substantial conclusion about those system.

Appendix A. MATLAB functions used and brief implementation explanation.

1. **lasso** - Compute regularized least-squares regression using lasso algorithms.
2. **smoothdata** - Smooth noisy data.
3. **spline** - Compute cubic spline interpolation.
4. **knn_deneme** - Compute embedding dimension for phase-space reconstruction using a geometrical construction based on Phys. Rev. A 45, 3403 (1992) with code by Merve Kizilkaya.

Appendix B. MATLAB codes.

B.0.1. Main Script.

1. Lynx-Hare Population

```
% AMATH 563 Spring 2018
% Homework 1: Question 1 Hare vs Lynx
% By Tun Sheng Tan     4/5/2018

%% Load data
input = load('hw1.mat');
data = input.data;
time = data(:,1);
hare = data(:,2);
lynx = data(:,3);

clear input data;
%% Smoothing the data
time = time - time(1); % Set the zeroth
timeSpline = linspace(time(1), time(end), 1000);
hareNE = spline(time, hare, timeSpline).';
lynxNE = spline(time, lynx, timeSpline).';
dhare = splineD(time, hare, timeSpline).';
dlynx = splineD(time, lynx, timeSpline).';
ddhare = splineD(timeSpline, dhare, timeSpline).';
ddlynx = splineD(timeSpline, dlynx, timeSpline).';

% UNCOMMENT IF USE FINITE DIFF
% [timeFD,dhareFD] = finiteD(timeSpline, hareNE);
% [timeFD,dlynxFD] = finiteD(timeSpline, lynxNE);
% [timeFD,ddhareFD] = finiteD(timeSpline, spline(timeFD, dhareFD,
    timeSpline));
% [timeFD,ddlynxFD] = finiteD(timeSpline, spline(timeFD, dlynxFD,
    timeSpline));

figure
subplot(2,1,1)
hold on
plot(time, hare, 'bo')
plot(timeSpline, hareNE)
plot(timeSpline, dhare)
% plot(timeFD, dhareFD)
% plot(timeFD, ddhareFD)
title('Hare')
hold off
```

```
subplot(2,1,2)
hold on
plot(time, lynx, 'bo')
plot(timeSpline, lynxNE)
plot(timeSpline, dlynx)
% plot(timeFD, dlynxFD)
% plot(timeFD, ddlynxFD)
title('Lynx')
hold off

%% Build Library

% UNCOMMENT IF USING FINITE DIFF
% hareNE = hareNE(2:end-1,1);
% lynxNE = lynxNE(2:end-1,1);
% dhare = dhareFD;
% dlynx = dlynxFD;
% ddhare = ddhareFD;
% ddlynx = ddlynxFD;
% timeSpline = timeFD;

% Udot = dhare;
Udot = dlynx; % UNCOMMENT IF FIT LYNX

% lib = [ hareNE   lynxNE   hareNE.*lynxNE   hareNE.^2   lynxNE.^2  ...
%         hareNE.^3  lynxNE.^3   hareNE.*(lynxNE.^2)   lynxNE.*(hareNE.^2) dlynx  ...
%         dlynx.^2 hareNE.*dlynx hareNE.*dlynx.^2 hareNE.^2.*dlynx ddlynx ...
%         ];
%
% labels = {'h','l','h*l','h^2','l^2',...
%           'h^3','l^3','hl^2','h^2l', 'dl',...
%           'dl^2','h dl','h dl^2','h^2dL','ddL',...
%           };

lib = [ hareNE   lynxNE   hareNE.*lynxNE   hareNE.^2   lynxNE.^2  ...
        hareNE.^3  lynxNE.^3   hareNE.*(lynxNE.^2)   lynxNE.*(hareNE.^2) dhare  ...
        dlynx.^2 hareNE.*dlynx hareNE.*dlynx.^2 hareNE.^2.*dlynx  ddhare ...
        ];

labels = {'h','l','h*l','h^2','l^2',...
          'h^3','l^3','hl^2','h^2l', 'dhare',...
          'dl^2','h dl','h dl^2','h^2dL','ddH',...
          };

M = length(labels);



%% Compare Regression
coeff1 = pinv(lib)*Udot;
```

```
coeff2 = lib\Udot;
[elasticHare, coefElastic] = cvElasticNet(lib, Udot, labels, 0.5);
[lassoHare, coefLasso] = cvLasso(lib, Udot, labels, false);
coeffs = [coeff1, coeff2, coefLasso, coefElastic];
titles = {'PseudoInv','QR','Lasso','ElasticNet'};
plotCoeff(coeffs, labels, titles);

%% Performing Thresholding for Lasso
[lassoHare, coefLasso] = cvLasso(lib, Udot, labels, false);
coefficientsLasso{1} = coefLasso;
labelsLasso{1} = labels;
lassoFitted{1} = lassoHare;
libLasso{1} = lib;
for i=2:10
%     [lib2, labels2, lassoHare2, coefLasso2]= lassoThreshold(lib, labels
    , Udot, coefLasso);
    [libLasso{i}, labelsLasso{i}, lassoFitted{i}, coefficientsLasso{i}]
    =...
        lassoThreshold(libLasso{i-1}, labelsLasso{i-1}, Udot,...
        coefficientsLasso{i-1});
end
% coefficientsLasso = {coefLasso, coefLasso2, coefLasso3, coefLasso4,
    coefLasso5};
% labelsLasso = {labels, labels2, labels3, labels4, labels5};
% lassoFitted = {lassoHare, lassoHare2, lassoHare3, lassoHare4,
    lassoHare5};
%%
figure();
barTot = length(labelsLasso);
for i =1:1
    subplot(2,2,1)
%     title(""+num2str(i));
    bar(coefficientsLasso{1})
    set(gca, 'xticklabel', labelsLasso{1},'Xtick',1:1:length(labelsLasso
    {1}));
    subplot(2,2,2)
    bar(coefficientsLasso{2})
    set(gca, 'xticklabel', labelsLasso{2},'Xtick',1:1:length(labelsLasso
    {2}));
    subplot(2,2,3)
    bar(coefficientsLasso{3})
    set(gca, 'xticklabel', labelsLasso{3},'Xtick',1:1:length(labelsLasso
    {3}));
    subplot(2,2,4)
    bar(coefficientsLasso{4})
    set(gca, 'xticklabel', labelsLasso{4},'Xtick',1:1:length(labelsLasso
    {4}));
end

%% Visual Inspection
figure();
hold on
plot(timeSpline, Udot, 'black-','LineWidth',2)
plot(timeSpline, lassoHare,'b-','LineWidth',2)
```

```
plot(timeSpline, lassoHare2,'r-','LineWidth',2)
plot(timeSpline, lassoHare3,'g-','LineWidth',2)
plot(timeSpline, lassoHare4,'c-','LineWidth',2)
legend({'Truth','1', '2','3','4'})
xlabel('time')
ylabel('hare'' ')
hold off
%% Select Model Based on KL Divergence and AIC BIC Scores
for i = 1:barTot
    koeff = coefficientsLasso{i};
    kld(i) = KLDiv(Udot, koeff);
    [AIC(i), BIC(i)] = aicbicRSS(Udot, lassoFitted{i}, length(time),...
                length(koeff(abs(koeff)>0) ));
end
disp(kld);
disp(AIC);
disp(BIC);
%% Time delay embedding with false nearest neighbor
%tao = 53; % Time delay
mmax = 10; % maximum embedding dimension
rtol = 15;
atol = 2;
flagPlot = false;
Ntao = length(x);
dim = zeros(100,1);
for tao = 1:100
    k = knn_deneme(x,tao,mmax,rtol,atol,flagPlot);
     dim(tao) = find(k==0,1);
end
figure();
plot(dim);
xlabel("Delay Time");
ylabel("Embedding Dimension");
%% Time delayed Embedding
% x = cat(1,hareNE,lynxNE).';
figure()
x = hareNE.';
H= hank(x, 100);
[u,s,v]=svd(H,'econ');
loglog(diag(s)/(sum(diag(s))),'Linewidth',2)
title("Singular Value")
xlabel("Rank of Singular value")
%% Hankel Matrix
function h = hank(X, n)
    h = [];
    m = length(X);
    Y = reshape(X, m, 1);
    for i = 1:(m-n+1)
        h = cat(2, h, Y(i:i+(n-1)));
    end
end
%% Compute Elastic Net with 10 Cross validation
function [elasticHare, coefElastic]= cvElasticNet(lib, Udot, labels, alph
    )
```

```
    [Bnet, info4] = lasso(lib, Udot, 'CV',10,'Alpha',alph);
    lassoPlot(Bnet, info4, 'PlotType','CV');
    legend('show');
    idxLambda1SE = info4.Index1SE;
    coefElastic = Bnet(:,idxLambda1SE);
    interceptNet = info4.Intercept(idxLambda1SE);
    elasticHare = lib*coefElastic + interceptNet;
    lassoPlot(Bnet, info4,'PlotType','Lambda','XScale','log','
    PredictorNames',labels);
    %legend('show');
end
%% Compute Lasso with 10 cross validation
function [lassoHare, coefLasso]=cvLasso(lib, Udot, labels, flagPlot)
    [Blasso, info3] = lasso(lib, Udot, 'CV',10);
    if flagPlot
        lassoPlot(Blasso, info3, 'PlotType','CV');
        legend('show');
    end
    idxLambda1SE = info3.Index1SE;
    coefLasso = Blasso(:,idxLambda1SE);
    interceptLasso = info3.Intercept(idxLambda1SE);
    lassoHare = lib*coefLasso+ interceptLasso;
    if flagPlot
    lassoPlot(Blasso, info3,'PlotType','Lambda','XScale','log','
    PredictorNames',labels);
    % show('legend');
    end
end
%% Compute Lasso by thresholding
function [lib2, labels2, lassoHare2, coefLasso2] = lassoThreshold(lib,
    labels, Udot, coefLasso)
    lib2 = lib;
    labels2 = labels;
    % Threshold very small coefficients
    index = find(abs(coefLasso) < 0.0001);
    lib2(:,index) = [];
    labels2(index) = [];

    % Compute Lasso
    [lassoHare2, coefLasso2] = cvLasso(lib2, Udot, labels2, false);
end
%% Calculating KL Divergence
function kld = KLDiv(hareRef, modelBest)
%     hareRef = interp1(timeSpline, Udot, time);
    hareRef = hareRef/sum(hareRef);
%     hareFitted = lib*koeff/sum(lib*koeff);
%     modelBest =  spline(timeSpline, hareFitted, time);
    modelBest = modelBest/sum(modelBest);

    % generate PDFs
    x = linspace(min(hareRef),max(hareRef),10);
    f = hist(hareRef,x);
    g = hist(modelBest,x);

```

```
    % normalize
    f = f/trapz(x,f);
    g = g/trapz(x,g);
    %plot(x,f,'r',x,g,'b');
    Int = f.*log(f./g);

    % use if needed
    Int(isinf(Int))=0; Int(isnan(Int))=0;

    kld = trapz(x,Int);

end
%% Compute AIC BIC with RSS as likelihood
function [AIC, BIC] = aicbicRSS(hareRef, modelBest, m, k)
    % m number of data points
    % k number of predictors
%     hareRef = hareRef/sum(hareRef);
%     modelBest  = modelBest/sum(modelBest);
    RSS = sum((hareRef - modelBest).^2); % Likelihood function

%     m = length(time);
%     k = length(coeff2(abs(coeff2)>1E-4));
    AIC = m*log(RSS/m) + 2*k;
    BIC = m*log(RSS/m) + k*log(m);
%     disp(AIC);
%     disp(BIC);
end
%AICcorrect = AIC + (2*(k+1)*(k+2))/(m-k-2);
%disp(AICcorrect);
```

2. BZ Reaction

```
% AMATH 563 Spring 2018
% Homework 1: Question 2 BZ Oscillation
% By Tun Sheng Tan    4/5/2018

%% Load Data
data = load('BZ_medium.mat');
BZ_tensor = data.BZ_tensor;
% data = load('BZ.mat');
% BZ_tensor = data.BZ_tensor;
[m,n,k] = size(BZ_tensor);
clear data;
%% Check
figure();
for j=1:1
   A=BZ_tensor(:,:,j);
   pcolor(A), shading interp, pause(0.01)
end
ax = gca
ax.Visible = 'off'
%% Extract X and Y wave
X = BZ_tensor(1:100,97,:);
Y = BZ_tensor(50, 51:150, :);
X = reshape(X, [100,k]);
```

```
Y = reshape(Y, [100,k]);
% Smoothing the data
Xsmooth = smoothdata(X, 'sgolay');
Ysmooth = smoothdata(Y, 'sgolay');
Xsmooth = Xsmooth(50:end,:);
Ysmooth = Ysmooth(50:end,:);
for j=1:1
    A = BZ_tensor(:,:,j);
    A(1:100,98,:) = 0;
    A(50,51:150,:) = 0;
    subplot(3,1,1);
    plot(Xsmooth(:,j));
%     ylim([-50 50]);
    ylim([0 140]);
    subplot(3,1,2);
    plot(Ysmooth(:,j));
    subplot(3,1,3);
%     plot(Xsmooth(50,j),Ysmooth(50,j))
    pcolor(A), shading interp;
    pause(0.05)
end

%% Smoothing the data
[a,b] = size(Xsmooth);
time = linspace(0, b-1, b);
t = linspace(0, b-1, 5*b);
Xsmooth = sqrt(Xsmooth.^2 + Ysmooth.^2); % Take the radial component

Xdot=zeros(a,b-2);
Xdotdot = zeros(a,b-2);
dt = 1;
for jj=1:a  % walk through rows (space)
    for j=2:b-1  % walk through time
        Xdot(jj,j-1)=( Xsmooth(jj,j+1)-Xsmooth(jj,j-1) )/(2*dt);
        Xdotdot(jj,j-1)=( Xsmooth(jj,j+1)+Xsmooth(jj,j-1)-2*Xsmooth(jj,j))/(dt);
    end
end

% derv matrices
dx=1;

D=zeros(a,a); D2=zeros(a,a);
for j=1:a-1
  D(j,j+1)=1;
  D(j+1,j)=-1;
%
  D2(j,j+1)=1;
  D2(j+1,j)=1;
  D2(j,j)=-2;
end
D(a,1)=1;
D(1,a)=-1;
D=(1/(2*dx))*D;
```

```
%
D2(a,a)=-2;
D2(a,1)=1;
D2(1,a)=1;
D2=D2/(dx^2);

u=reshape(Xsmooth(:,2:end-1).',(b-2)*a ,1);
ux = zeros(b-2,a);
uxx = zeros(b-2,a);
for jj=2:b-1
    ux(jj-1,:)=(D*Xsmooth(:,jj)); % u_x
    uxx(jj-1,:)=(D2*Xsmooth(:,jj)); % u_xx
end

Ux=reshape(ux,(b-2)*a,1);
Uxx=reshape(uxx,(b-2)*a,1);
Udotdot=reshape(Xdotdot, (b-2)*a,1);

I = ones(size(Ux));

lib=[ u  u.^2 u.^3 u.^4 u.^5 ...
     Ux Ux.^2 Ux.^3 Ux.^4 ...
     Uxx Uxx.^2 Uxx.^3 Uxx.^4 ...
     Ux.*u Ux.*(u.^2) Ux.*Uxx ...
     Ux.*(Uxx.^2) Udotdot];

labels = {'u' 'u^2'  'u^3' 'u^4' 'u^5' ...
          'U_x' '{U_x}^2' '{U_x}^3' '{U_x}^4'...
          'U_{xx}' '{U_{xx}}^2' '{U_{xx}}^3' '{U_{xx}}^4'...
          'uU_{x}' 'u^2U_{x}' 'U_{x}U_{xx}'...
          'U_{x}{U_{xx}}^2' 'U_{tt}' ...
          };


M = length(labels);

Udot=reshape((Xdot.'),(b-2)*a,1);
%%
[lassoHare, coefLasso] = cvLasso(lib, Udot, labels, false);
coefficientsLasso{1} = coefLasso;
labelsLasso{1} = labels;
lassoFitted{1} = lassoHare;
libLasso{1} = lib;
for i=2:6
    [libLasso{2}, labelsLasso{i}, lassoFitted{i}, coefficientsLasso{i}]
     =...
        lassoThreshold(libLasso{1}, labelsLasso{i-1}, Udot,...
        coefficientsLasso{i-1});
    libLasso{1} = libLasso{2};
end
%%
figure();
barTot = length(labelsLasso);
for i =1:1
```

```
    subplot(2,2,1)
%      title(""+num2str(i));
    bar(coefficientsLasso{1})
    set(gca, 'xticklabel', labelsLasso{1},'Xtick',1:1:length(labelsLasso
    {1}));
    subplot(2,2,2)
    bar(coefficientsLasso{2})
    set(gca, 'xticklabel', labelsLasso{2},'Xtick',1:1:length(labelsLasso
    {2}));
    subplot(2,2,3)
    bar(coefficientsLasso{3})
    set(gca, 'xticklabel', labelsLasso{3},'Xtick',1:1:length(labelsLasso
    {3}));
    subplot(2,2,4)
    bar(coefficientsLasso{4})
    set(gca, 'xticklabel', labelsLasso{4},'Xtick',1:1:length(labelsLasso
    {4}));
end
%% Select Model Based on KL Divergence and AIC BIC Scores
for i = 1:barTot
    koeff = coefficientsLasso{i};
    kld(i) = KLDiv(Udot, koeff);
    [AIC(i), BIC(i)] = aicbicRSS(Udot, lassoFitted{i}, length(time),...
                length(koeff(abs(koeff)>0) ));
end
disp(kld);
disp(AIC);
disp(BIC);
%% Custom Functions
% Compute Lasso with 3 cross validation
function [lassoHare, coefLasso]=cvLasso(lib, Udot, labels, flagPlot)
    [Blasso, info3] = lasso(lib, Udot, 'CV',3);
    if flagPlot
        lassoPlot(Blasso, info3, 'PlotType','CV');
        legend('show');
    end
    idxLambda1SE = info3.Index1SE;
    coefLasso = Blasso(:,idxLambda1SE);
    interceptLasso = info3.Intercept(idxLambda1SE);
    lassoHare = lib*coefLasso+ interceptLasso;
    if flagPlot
    lassoPlot(Blasso, info3,'PlotType','Lambda','XScale','log','
    PredictorNames',labels);
    % show('legend');
    end
end

% Compute Lasso by thresholding
function [lib2, labels2, lassoHare2, coefLasso2] = lassoThreshold(lib,
    labels, Udot, coefLasso)
    lib2 = lib;
    labels2 = labels;
    % Threshold very small coefficients
    index = find(abs(coefLasso) < 0.0001);
```

```
    lib2(:,index) = [];
    labels2(index) = [];

    % Compute Lasso
    [lassoHare2, coefLasso2] = cvLasso(lib2, Udot, labels2, false);
end

% Calculating KL Divergence
function kld = KLDiv(hareRef, modelBest)
%     hareRef = interp1(timeSpline, Udot, time);
    hareRef = hareRef/sum(hareRef);
%     hareFitted = lib*koeff/sum(lib*koeff);
%     modelBest =  spline(timeSpline, hareFitted, time);
    modelBest = modelBest/sum(modelBest);

    % generate PDFs
    x = linspace(min(hareRef),max(hareRef),10);
    f = hist(hareRef,x);
    g = hist(modelBest,x);

    % normalize
    f = f/trapz(x,f);
    g = g/trapz(x,g);
    %plot(x,f,'r',x,g,'b');
    Int = f.*log(f./g);

    % use if needed
    Int(isinf(Int))=0; Int(isnan(Int))=0;

    kld = trapz(x,Int);

end

% Compute AIC BIC with RSS as likelihood
function [AIC, BIC] = aicbicRSS(hareRef, modelBest, m, k)
    % m number of data points
    % k number of predictors

    RSS = sum((hareRef - modelBest).^2); % Likelihood function
    AIC = m*log(RSS/m) + 2*k;
    BIC = m*log(RSS/m) + k*log(m);
end
```

B.1. Custom Coefficients Plotting Script.

1. Plotting Coefficients lstinputlisting[language=matlab]Codes/plotCoeff.m
2. Spline Data

```
% Spline difference Derivative
% return array with length N
function y = splineD(time,X,range)
%     tt = linspace(time(1),time(end),N);
    Xspline = spline(time,X);
    p_Xspline = fnder(Xspline, 1);
    y = ppval(p_Xspline, range);
```

```
end
```

B.2. Codes From Matlab Central.

1. False Nearest Neighbor Script by Merve Kizilkaya

```
function [FNN] = knn_deneme(x,tao,mmax,rtol,atol, flagPlot)
%x : time series
%tao : time delay
%mmax : maximum embedding dimension
%reference:M. B. Kennel, R. Brown, and H. D. I. Abarbanel, Determining
%embedding dimension for phase-space reconstruction using a geometrical
%construction, Phys. Rev. A 45, 3403 (1992).
%author:"Merve Kizilkaya"
%rtol=15
%atol=2;
N=length(x);
Ra=std(x,1);

for m=1:mmax
    M=N-m*tao;
    Y=psr_deneme(x,m,tao,M);
    FNN(m,1)=0;
    for n=1:M
        y0=ones(M,1)*Y(n,:);
        distance=sqrt(sum((Y-y0).^2,2));
        [neardis nearpos]=sort(distance);

        D=abs(x(n+m*tao)-x(nearpos(2)+m*tao));
        R=sqrt(D.^2+neardis(2).^2);
        if D/neardis(2) > rtol || R/Ra > atol
            FNN(m,1)=FNN(m,1)+1;
        end
    end
end
FNN=(FNN./FNN(1,1))*100;
if (flagPlot)
    figure
    plot(1:length(FNN),FNN)
    grid on;
    title('Minimum embedding dimension with false nearest neighbours')
    xlabel('Embedding dimension')
    ylabel('The percentage of false nearest neighbours')
end

function Y=psr_deneme(x,m,tao,npoint)
%Phase space reconstruction
%x : time series
%m : embedding dimension
%tao : time delay
%npoint : total number of reconstructed vectors
%Y : M x m matrix
% author:"Merve Kizilkaya"
N=length(x);
if nargin == 4
```

```
    M=npoint;
else
    M=N-(m-1)*tao;
end

Y=zeros(M,m);

for i=1:m
    Y(:,i)=x((1:M)+(i-1)*tao)';
end
```

REFERENCES

[1] S. L. Brunton, J. L. Proctor, and J. N. Kutz, *Discovering governing equations from data by sparse identification of nonlinear dynamical systems*, Proceedings of the National Academy of Sciences, 113 (2016), pp. 3932–3937, https://doi.org/10.1073/pnas.1517384113, http://www.pnas.org/content/113/15/3932, https://arxiv.org/abs/http://www.pnas.org/content/113/15/3932.full.pdf.

[2] MATLAB and S. Toolbox, 9.3.0.713579 (R2017b).

[3] F. Takens, *Detecting strange attractors in turbulence*, Lecture Notes in Mathematics, Berlin Springer Verlag, 898 (1981), p. 366, https://doi.org/10.1007/BFb0091924.

[4] H. Whitney, *Differentiable manifolds*, Annals of Mathematics, 37 (1936), pp. 645–680, http://www.jstor.org/stable/1968482.