AMATH 563 – HW2

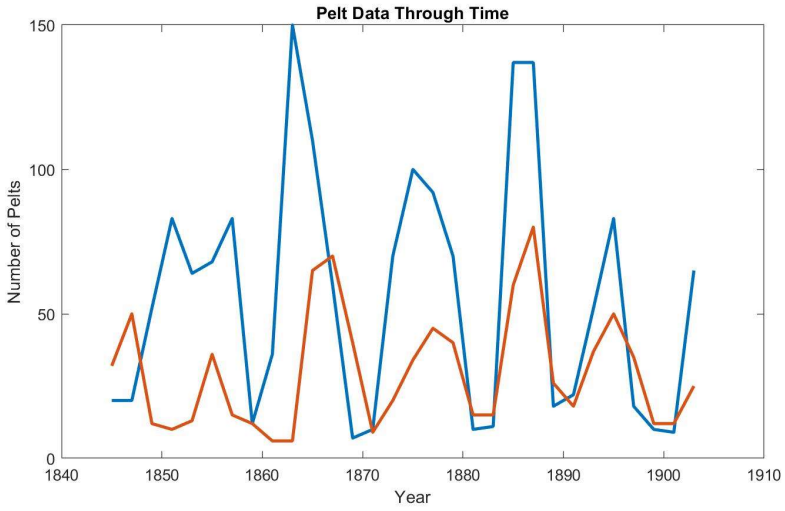
Nick Montana

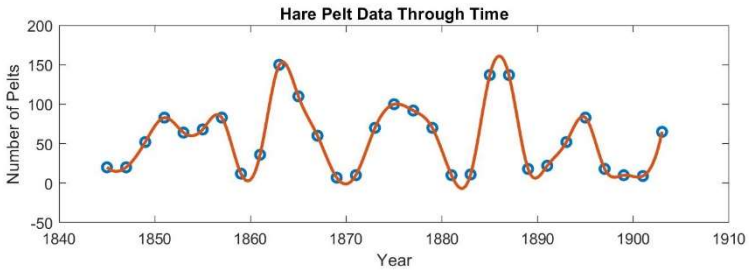
5/3/2019

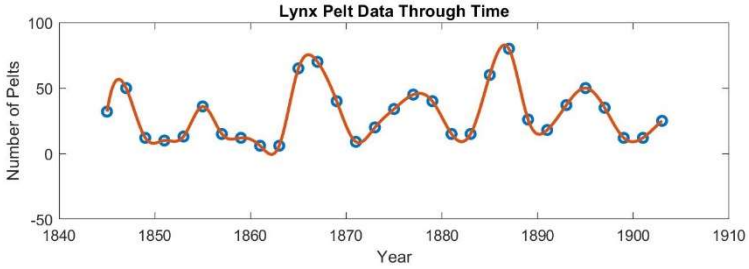
Abstract

This assignment investigates a data set concerning Canadian lynx and snowshoe hare populations from 1845 to 1903, as well as a data set concerning a chemical reaction. We assume that the first data set is modeled after an ordinary differential equation and we use regression methods to determine this equation. We apply a similar approach to the chemical reaction, but assume it is modeled after a partial differential equation.

Sec. I: Introduction and Overview







Sec. II: Theoretical Background

Our goal in this task to turn the problem of solving the differential equation $\frac{dy}{dt} = f(y,t)$ into an Ax = b problem. To do this, we calculate the estimated derivative, y'(t), at each time value t. This is our vector b. We then create a library of potential functions which might model this differential equation, called A. These functions are guesses, but typically polynomials of degree n, constants, and trig functions are good guesses. Solving Ax=b, therefore essentially boils down to trying to find the "weights" or "coefficients", x, for these functions. If one of them is not a guess, then the corresponding weight will be very small (less than .01, for example). If, however, it is essential to modeling the differential equation, then it's corresponding will be large. We solve this Ax=b equation using the regression methods covered in HW1.

Sec. III: Algorithm Implementation and Development

Our code begins by manually entering the pelt data into a 30x3 matrix. We then define variables such as large dt which is the given time change between data points, 2, and the range of t: 1845 to 1903. Our data set is relatively small and there are large gaps in time between them. To compensate for this, we interpolate our data using the spline function to put more points in between them using a dt of .25. We then plot our interpolated data.

Now, we need to calculate an estimate for y'(t). We do this by calculating the slope of the secant between each two data points. We then create 3 libraries of potential functions, Library1 and Library2. Library1 features only polynomials up to degree 3, while Library2 features constant functions, trig functions and polynomials up to degree 5.

Now that we have our b and library A, we implement a regression to find our coefficients. We then use 3 different approaches to find 3 different models. Our first model uses backslash on Library1, our second uses backslash on Library2, and our third uses lasso on Library2.

For each of these models, we threshold our coefficients by changing any whose absolute value is less than .00000000000001. I chose this number after trying many different lengths of 1x10^-n.

After thresholding, we're ready to see how our well our methods worked. We use Euler's method to go from our derivative to a function which models the data.

We then calculate the KL Divergence and AIC/BIC scores of our 3 different models to see which one produces the best results.

Sec. IV: Computational Results

1. **Find the best model.**

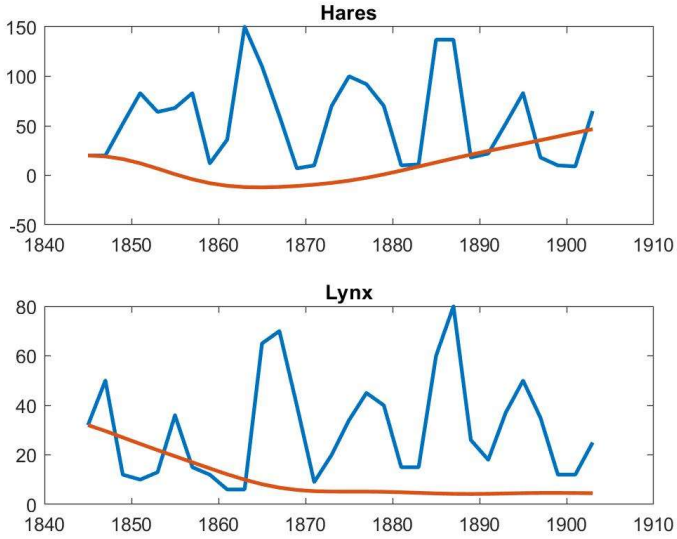


Figure 1: Model 1 Results

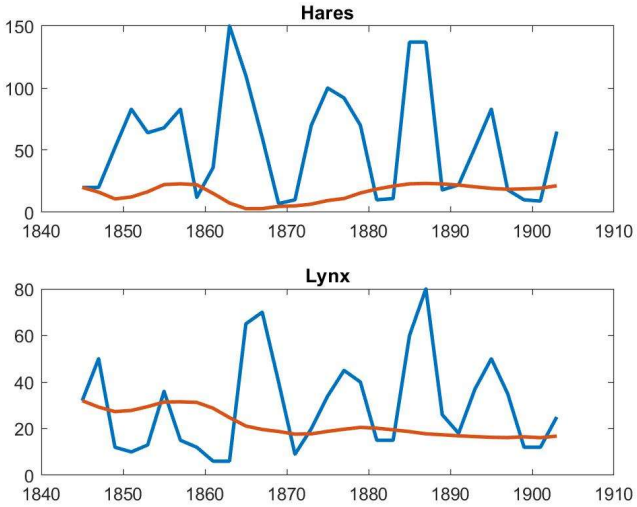


Figure 2: Model 2 Results

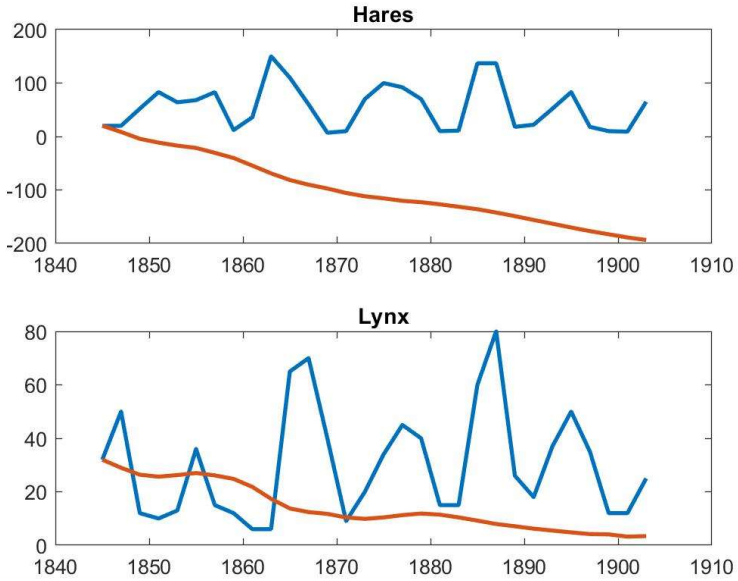


Figure 3: Model 3 Results

2. **Compute the KL divergence of the best model**
    a. **Model 1**
        i. **Hares: 0.7442**
        ii. **Lynx: 1.2125**
    b. **Model 2**
        i. **Hares: 0.7745**
        ii. **Lynx: 1.3684**
    c. **Model 3**
        i. **Hares: 0.7058**
        ii. **Lynx: 1.2795**
3. **Compare the three models AIC and BIC scores**
    a. **Model 1**
        i. **Hares: AIC = 4.3574, BIC = 2.6914**
        ii. **Lynx: AIC = 3.9219, BIC = 2.2560**
    b. **Model 2**
        i. **Hares: AIC = 46.8942, BIC = 40.4387**
        ii. **Lynx: AIC = 53.5815, BIC = 47.1261**
    c. **Model 3**
        i. **Hares: AIC = 39.772, BIC = 33.3238**
        ii. **Lynx: AIC = 49.7116, BIC = 43.2562**

Sec. V: Summary and Conclusion

The models did not produce very good results. Perhaps this is because we are dealing with such a small amount of data points.

Appendix A

Appendix B

```
%AMATH 563 - HW2
%Author: Nick Montana


pelt_data = [1845 20 32;
    1847 20 50;
    1849 52 12;
    1851 83 10;
    1853 64 13;
    1855 68 36;
    1857 83 15;
    1859 12 12;
    1861 36 6;
    1863 150 6;
    1865 110 65;
    1867 60 70;
    1869 7 40;
    1871 10 9;
    1873 70 20;
    1875 100 34;
    1877 92 45;
    1879 70 40;
    1881 10 15;
    1883 11 15;
    1885 137 60;
    1887 137 80;
    1889 18 26;
    1891 22 18;
```

```
    1893 52 37;
    1895 83 50;
    1897 18 35;
    1899 10 12;
    1901 9 12;
    1903 65 25];

large_dt =2;
t=1845:large_dt:1903;

dt = .25;
interpolated_time = 1845:dt:1903;
interpolated_hare = spline(t, pelt_data(:, 2), interpolated_time);
interpolated_lynx = spline(t, pelt_data(:, 3), interpolated_time);

%% Plot the interpolated data
figure(1)
subplot (2, 1, 1);
plot(t,pelt_data(:,2), 'o',interpolated_time, interpolated_hare,'Linewidth',[2]);
title ('Hare Pelt Data Through Time');
xlabel('Year');
ylabel('Number of Pelts')
subplot (2, 1, 2);
plot(t,pelt_data(:,3), 'o',interpolated_time, interpolated_lynx,'Linewidth',[2]);
%Right after pelt_data(:,2), you could put a '', 'r*', to make points.
%Might want to consider to interpolate the data. Check out spline.
title ('Lynx Pelt Data Through Time');
xlabel('Year');
ylabel('Number of Pelts');
```

```
%{
n = length (pelt_data);

x1=pelt_data (:, 2);
x2=pelt_data (:, 3);
%}

%% Setting up data and Libraries
n = length(interpolated_hare);
x1 = interpolated_hare.';
x2 = interpolated_lynx.';

for j=2:n-1
  x1dot(j-1)=(x1(j+1)-x1(j-1))/(2*dt);
  x2dot(j-1)=(x2(j+1)-x2(j-1))/(2*dt);
end

x1s=x1(2:n-1);
x2s=x2(2:n-1);

one = ones(231,1);
two = 2*one;
three = 3*one;

Library1=[x1s x2s x1s.^2 x1s.*x2s x2s.^2 x1s.^3 (x2s.^2).*x1s x2s.^3];
Library2=[one two three sin(x1s) sin(x2s) cos(x1s) cos(x2s) sin(x1s).*cos(x1s) ...
    sin(x1s).*cos(x2s) sin(x2s).*cos(x1s) sin(x2s).*cos(x2s) ...
    x1s x2s x1s.^2 x1s.*x2s x2s.^2 ...
    x1s.^3 (x1s.^2).*x2s (x2s.^2).*x1s x2s.^3 ...
    x1s.^4 (x1s.^3).*x2s (x1s.^2).*(x2s.^2) x1s.*(x2s.^3) x2s.^4 ...
```

```
    x1s.^5 (x1s.^4).*x2s (x1s.^3).*(x2s.^2) (x1s.^2).*(x2s.^3)  ...
    x1s.*(x2s.^4) x2s.^5];

%% Do regression with Backslash for Library1
%{
mu_coeffecients1=Library1\x1dot.';
mu_coeffecients2=Library1\x2dot.';
figure(2)
subplot(2,1,1), bar(mu_coeffecients1)
subplot(2,1,2), bar(mu_coeffecients2)
%}
%% Do regression with Backslash for Library 2
%{
mu_coeffecients1=Library2\x1dot.';
mu_coeffecients2=Library2\x2dot.';
figure(2)
subplot(2,1,1), bar(mu_coeffecients1)
subplot(2,1,2), bar(mu_coeffecients2)
%}
%% Do regression with Lasso for Library 2

mu_coeffecients1=lasso(Library2, x1dot.');
mu_coeffecients2=lasso(Library2,x2dot.');
figure(5)
subplot(2,1,1), bar(mu_coeffecients1)
subplot(2,1,2), bar(mu_coeffecients2)

%% Thresholding
mu1_new = mu_coeffecients1;
for i = 1:length (mu_coeffecients1)
```

```
    if (abs(mu_coeffecients1(i)) < .0000000000001)
        mu1_new(i) = 0;
    else
        mu1_new(i) = mu_coeffecients1 (i);
    end
end

mu2_new = mu_coeffecients2;
for i = 1:length (mu_coeffecients2)
    if (abs(mu_coeffecients2(i)) < .0000000000001)
        mu2_new(i) = 0;
    else
        mu2_new(i) = mu_coeffecients2 (i);
    end
end

%% I don't know what this section is
%I don't think this did anything.
%{
f1x = A2*xi1_new;
xi1_newest=A2\f1x;

figure(14)
bar(xi1_newest)
%}

figure(3)
bar(mu1_new);
f1x = Library2*mu1_new;
f2x = Library2*mu2_new;
```

```
%% Compute the slopes and see if they match up. Euler's Method
x_estimate1 = zeros(30,1);
x_estimate1(1) = x1(1);
for i = 1:29
   x_estimate1 (i + 1) = x_estimate1 (i) + dt*f1x(i);
end

x_estimate2 = zeros(30,1);
x_estimate2(1) = x2(1);
for i = 1:29
   x_estimate2 (i + 1) = x_estimate2 (i) + dt*f2x(i);
end

figure;
subplot(2,1,1); plot(t,pelt_data(:,2),t, x_estimate1,'Linewidth',[2]);
title('Hares');
subplot (2,1,2); plot(t,pelt_data(:,3),t, x_estimate2,'Linewidth',[2]);
title('Lynx');

%% Create a scatter Plot of the Data and Model
figure;
scatter(x_estimate1, x_estimate2, 'filled');
figure;
scatter(pelt_data(:, 2), pelt_data(:,3), 'filled');

%% Trying to use the Euler function
%{
x_estimate1 = Euler (f1x, 1845, 1903, x1(1), 29);
```

```
figure (45);
plot(t,pelt_data(:,2),t, x_estimate1,'Linewidth',[2]);
%}


%% Calculate the KL Divergence
data_range = min(pelt_data(:,2)): .01: max(pelt_data(:,2));
f=hist(pelt_data(:,2),data_range)+0.01;
g1=hist(x_estimate1,data_range)+0.01;
f=f/trapz(data_range,f); % normalize data
g1=g1/trapz(data_range,g1);

plot(data_range,f,data_range,g1,'Linewidth',[2])
Int1=f.*log(f./g1);
I1=trapz(data_range,Int1)

data_range = min(pelt_data(:,3)): .01: max(pelt_data(:,3));
f=hist(pelt_data(:,3),data_range)+0.01;
g1=hist(x_estimate2,data_range)+0.01;
f=f/trapz(data_range,f); % normalize data
g1=g1/trapz(data_range,g1);

plot(data_range,f,data_range,g1,'Linewidth',[2])
Int2=f.*log(f./g1);
I2=trapz(data_range,Int2)

%% This is where ode45 would go if I could figure out how to work it.
%{
%Trying to implement ode45
f1x = A2*xi1;
```

```
tspan=[1845 1903];
y0 = [x1(1) x2(1)];
[r y] =ode45('rhs_test', t, x1(1), x2(1), f1x);

figure;
plot(t,y(1),t,y(2))
%}

%% AIC - K is the number of parameters in your library. Google AIC ,
%Use sum(y-xdot)^2, n is the number of "bins"

K = size(Library2, 2);
RSS1 = 0;
for i = 1:30
RSS1 = (pelt_data(i,2) - x_estimate1(i))^2;
end
AIC1 = 2*K - 2*log(RSS1)
n = 6;
BIC1 = log(n)*K -2*log(RSS1)

RSS2 = 0;
for i = 1:30
RSS2 = (pelt_data(i,3) - x_estimate2(i))^2;
end
AIC2 = 2*K - 2*log(RSS2)
n = 6;
BIC2 = log(n)*K -2*log(RSS2)
```