Lab Assignment 4

Predator-Prey Dynamics in an Uncertain World

Written by

Prof. Erin N. Bodine

Prepared for

Math 214: Discrete Math Modeling

Rhodes College

Last Edited: May 1, 2018

In the reference readings we examined a model for a predator-prey system,

$$x_{t+1} = rx_t - \alpha x_t y_t \qquad y_{t+1} = sy_t + \beta x_t y_t, \tag{1}$$

where x_t is the density of the prey, y_t is the density of the predators, $r > 1$ is the growth rate of the prey in the absence of the predator population, $0 < s < 1$ is the survival rate of the predator population in the absence of its prey source, $\alpha > 0$ is the consumption rate of the predators (where αx_t is the average number of prey eaten per predator in time step t), and $\beta > 0$ is the growth rate of the predator population due to the consumption of prey. Recall that the equilibrium points of this system were

$$(0, 0) \quad \text{and} \quad \left(\frac{1-s}{\beta}, \frac{r-1}{\alpha}\right).$$

The first equilibrium point is also known as the *trivial equilibrium* or *mutual extinction equilibrium*; the second equilibrium point is known as the *coexistence equilibrium.*

The goals of this lab assignment are to:

- Visualize the relationship between equilibrium points and time-series solutions of a predator-prey model.
- Determine how stochasticity in the model parameters (r, s, α, and β) impact the dynamics of the populations.

In this lab, we will be using MATLAB to help achieve these goals. Thus, as a part of this lab you will learn how to do the following in MATLAB:

- Construct a phase-portraits.
- Use histograms to visualize variance in stochastic simulations.

MATLAB Skills

After completing Labs 1–3 you have most of the skill necessary to complete this lab project. In previous labs you have constructed and plotted the time-series solutions for a single difference equation. Note, a time-series solution refers to the sequence of values generated by a difference equation, and a time-series plot is a graph of that sequence of values over time (i.e., the horizontal axis represents the sequence index t or n, and the vertical axis represents the value of the terms in the sequence x_t or x_n). In this lab, you will construct and plot time-series solutions for systems of difference equations, which simply means we will plot both the x_t sequence and the y_t sequence over time on the same graph.

In addition to the time-series solutions, you will also plot phase portraits. A phase-portrait shows the time-series solution in the state space of a system of difference equations. If, for example, your system of difference equations generates sequences for x_t and y_t, then the state space of the system is in the xy-plane, and the phase portrait would plot the time-series solution (x_t, y_t) in the xy–plane.

Plots of Systems of Difference Equations

1. Download and script file `L04Ex01.m` from Moodle and open in MATLAB .

L04Ex01.m

```
% Lab 4 Example 1 by Prof Bodine, Simulates Predator-Prey Model
clear all

%% Parameters
r = 1.0015;
s = 0.9994;
a = 0.0006;
b = 0.00025;
x0 = 2;
y0 = 3;

years = 100;
T = 365*years; % Each time step represents 1 day
5
%Print the location of each equilibrium
fprintf('EQ 1 is (0,0)\n');
fprintf('EQ 2 is (%.2f,%.2f)\n',(1-s)/b,(r-1)/a);

%% Generate Solutions over T years
x(1) = x0; y(1) = y0;
for t = 1:T
  x(t+1) = r*x(t) - a*x(t)*y(t);
  y(t+1) = s*y(t) + b*x(t)*y(t);
end

%% Graphs of Solutions
% Plot time-series solution over time
time = 0:years/T:years;
subplot(1,2,1); plot(time,x,'r-',time,y,'b-')
xlabel('Time (years)'); ylabel('Population Density');
legend('Prey (x)','Predators (y)')

% Plot phase portrait in xy plane
subplot(1,2,2); plot(x,y,'c'); hold on;
plot(x0,y0,'kh');            % Plot Initial Condition
plot((1-s)/b,(r-1)/a,'mp') ; % Plot Non-trivial equilibrium
hold off;
xlabel('Prey density (x)'); ylabel('Predator density (y)');
```

2. Now, let us take a moment to understand everything contained within this script.

(a) The first two line give a description of the file in comments and use the `clear` command to erase the values of all variables and parameters from the workspace.

(b) Lines 4–10 define the values of the parameters of the model. Notice that in lines 9 and 10 we define the initial conditions of the system for each of the two state variables, x and y.

(c) Lines 12–13 define the number of time steps in the simulation. Since each time step in the model will represent 1 day, the total number of time steps is 365 times the number of years over which we wish to simulate.

(d) Lines 15–17 print the values of the two equilibrium points for the model using `fprintf` commands.

(e) In lines 19–24, the solution of the system of difference equations is generated. Notice that each pass through the loop generates both an x_n value and a y_n value using the equations from System (1).

Note that line 20 sets the initial condition for both the x sequence and the y sequence. If you wish to include two commands on one line, you must place a semi-colon after the first command.

(f) Lines 27–31 creates a graph of the time-series solution over time. Recall that each time step represents one day. If we used the command

```
plot(0:T,x,'r-', 0:T,y,'b-')
```

we would be plotting over $[0, 36500]$. This makes it difficult for someone looking at the graph to interpret the results, so we rescale the horizontal axis to represent years in line 28. The code `0:years/T:years` creates an array from 0 to `years` (which was defined to be 100 in line 12) in step sizes of `years/T` (which calculates to $1/365$). Note, this array has 36501 elements, which is the same length as the arrays `x` and `y`.

Notice in line 29 that the plot command is preceded by the command `subplot(1,2,1)`. This subplot command creates 1 column and 2 rows of plots. The last third entry of 1 indicates that the following plot command will be placed in the first position of the subplots (i.e. the top row).

(g) Lines 33–38 creates the second subplot (in the bottom row) which is a phase portrait of the solution. Line 34 generates the plot of the sequences of `x` and `y` in the xy–plane in cyan. Line 35 plots a black hexagonal star at the initial point (x_0, y_0), and line 36 plots a magenta pentagonal star at the non-trivial equilibrium of the system. Recall from the reference readings that the predatory-prey system defined by System (1) had two equilibria: $(0, 0)$ the trivial equilibrium, and $\left(\frac{1-s}{\beta}, \frac{r-1}{\alpha}\right)$ the only non-trivial equilibrium.

(h) Take another moment to review `L04Ex01.m`. **Do you understand what each line of code is doing?** If not, now is the time to stop and ask.

3. Run the script file. The resulting graphs should look like Figure 1.

Notice that each population is oscillating over time, which results in the phase portrait producing an egg-like shape that is traced over and over each time the oscillation repeats. Additionally, notice that in the phase portrait the egg-like shape is "centered" about the non-trivial equilibrium. We can describe this equilibrium point as being a ***center*** since it is the center of some stable oscillation. This would also imply that the other equilibrium, the trivial equilibrium, is unstable.

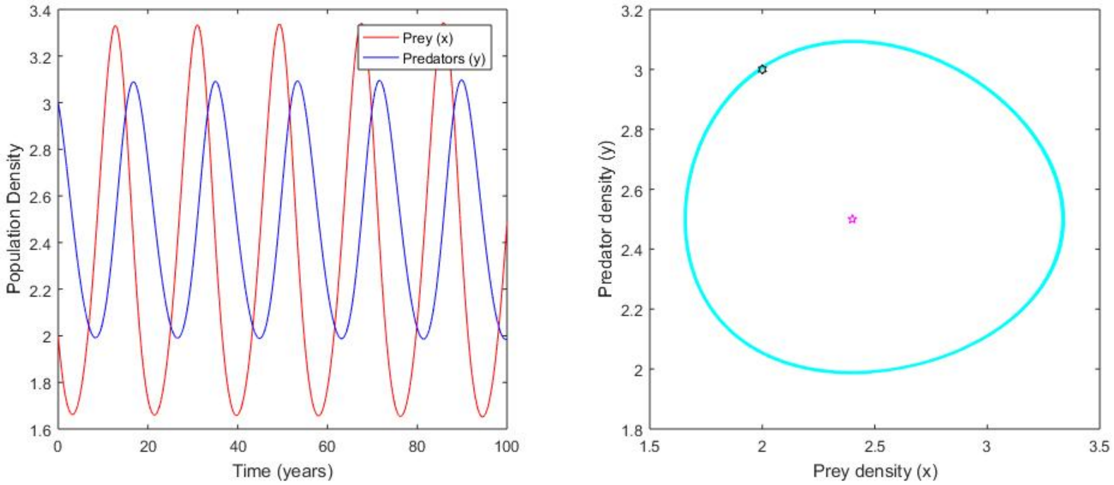


Figure 1: Plot produced from `L04Ex01.m` script.

Systems of Difference Equations with Stochastic Parameters

1. From Moodle, download the script file `L04Ex02.m`. This file is similar to `L04Ex01.m`, but allows for the parameters to be randomly drawn from uniform distribution.

`L04Ex02.m`

```
% Lab 4 Example 2 by Prof Bodine, Simulates Stochastic Predator-Prey Model
clear all; clf;

%% Parameters
r = 1.0015;
s = 0.9994;
a = 0.0006;
b = 0.00025;
x0 = 2;
y0 = 3;
years = 100;
T = 365*years; % Each time step represents 1 day

%Print the location of each equilibrium
xstar = (1-s)/b; ystar = (r-1)/a;
fprintf('EQ 1 is (0,0)\n');
fprintf('EQ 2 is (%.2f,%.2f)\n',xstar,ystar);

%% Generate 1 solution over T years with no stochasticity
x(1) = x0; y(1) = y0;
```

```
for t = 1:T
        x(t+1) = r*x(t) - a*x(t)*y(t);
        y(t+1) = s*y(t) + b *x(t)*y(t);
end

%% Generate N solutions over T years with stochastic parameters
N = 100;
for i=1:N
  X(i,1)=x0; Y(i,1)=y0;
  for t=1:T
    % Generate random parameter values
    R = r + 0.001*(2*rand-1);
    S = s + 0.001*(2*rand-1);
    A = a + 0.001*(2*rand-1);
    B = b + 0.001*(2*rand-1);

    % Generate terms in the sequence
    X(i,t+1) = R*X(i,t) - A*X(i,t)*Y(i,t);
    Y(i,t+1) = S*Y(i,t) + B*X(i,t)*Y(i,t);
  end
end

%% Graphs of Solutions
% Plot time-series solution over time
time = 0:years/T:years;
subplot(2,2,1); plot(time,x,'r:',time,y,'b:');
hold on; plot(time,X(N,:),'r-',time,Y(N,:),'b-'); hold off;
xlabel('Time (years)'); ylabel('Population Density');
legend('Prey (x)','Predators (y)')

% Plot phase portrait in xy plane
subplot(2,2,2); plot(x,y,'c'); hold on; plot(X(N,:),Y(N,:),'k');
plot(x0,y0,'ko');             % Plot Initial Condition
plot((1-s)/b,(r-1)/a,'mp') ; % Plot Non-trivial equilibrium
xlabel('x'); ylabel('y');
hold off;

% Histogram of the population densities at T+1 over all N simulations
subplot(2,2,3:4); hold on;
histogram(X(:,T+1),10,'facecolor','r');
histogram(Y(:,T+1),10,'facecolor','b');
plot([xstar,xstar],[0,40],'r-.',[ystar,ystar],[0,40],'b-.');
```

```
63  xlabel('Population Density');
64  legend('Prey (x)','Predators (y)','x^*','y^*'); hold off;
65
66  % Print the variance of the population densities at T+1 over all N simulations
67  fprintf('X(T+1) standard deviation is %.3f\n',std(X(:,T+1)))
68  fprintf('Y(T+1) standard deviation is %.3f\n',std(Y(:,T+1)))
```

2. Now, let us take a moment to understand everything contained within this script.

   (a) Notice that lines 1–24 of `L04Ex02.m` are essentially the same as from `L04Ex01.m`. The `clf` command on line 2, clears the figure window, which is necessary for the histograms to plot corrected if running the code more than once.

   (b) Lines 26–41 contain a set of nested for-loops. The inner loop generates sequences for x_t and y_t for T times steps. At each time step, the values of the parameters are randomly selected over uniform distributions. The outer loop repeats this simulation N times.

   Each of the parameter values is sampled over a uniform distribution centered around the corresponding parameter value defined in lines 5–8. For example, in line 5 the value of r is set to 1.0015. In line 33, we set the value of R (the stochastic version of r) by sampling from the uniform distribution $r \pm 0.001 = \mathcal{U}[1.0015 - 0.001, 1.0015 + 0.001] = \mathcal{U}[1.0005, 1.0025]$. Notice, the code `2*rand-1` generates random numbers in the range $[-1, 1]$, and thus the code `0.001*(2*rand-1)` generates random numbers in the range $[-0.001, 0.001]$. Since any value from this range is equally likely to be selected, the values are sampled from a uniform distribution.

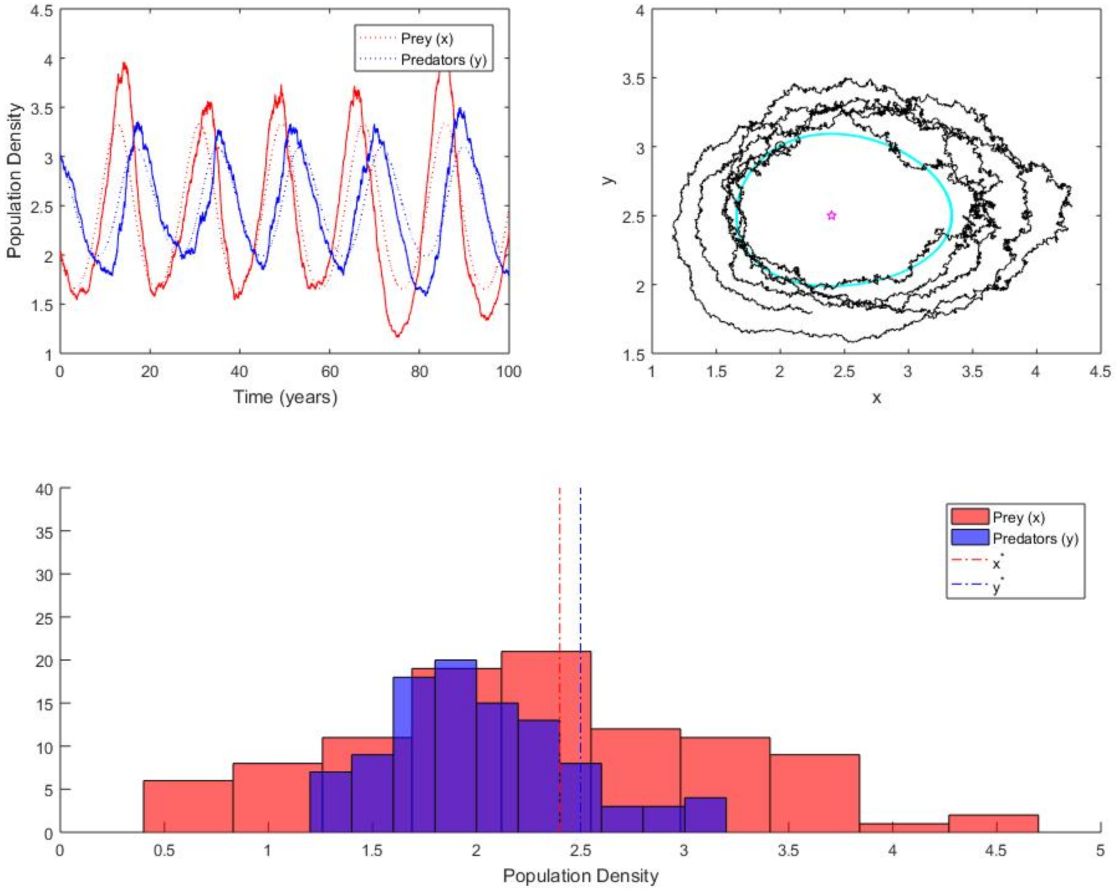
   Recall from Lab01 that the Matlab coding language is case sensitive and thus `x` and `X` represent different quantities in Matlab. The arrays `x` and `y` are one dimension and have length $T + 1$. The arrays `X` and `Y` are two-dimensional where each row (indicated by index i) represents a single simulation and each column (indicated by index t) represents a single time step. Thus, if we look across a row, we see all the values generated for a single simulation, but if we look down a column, we see the values over all N simulations for a single time step. The term `X(i,t)` in line 38 is referencing the element in the i^{th} row and t^{th} column of the two-dimensional array `X`.

   (c) Lines 44–49 generate a time series plot where the non-stochastic sequences of x_t and y_t are plotted with dotted lines, and just the last simulation (i.e., the N^{th} simulation) of the stochastic sequences are shown with solid lines.

   (d) Lines 51–56 generate a phase portrait where the non-stochastic solution is plotted in cyan, and just the last simulation (i.e., the N^{th} simulation) of the stochastic solutions is shown in black. Note the non-trivial equilibrium is also plotted on the phase portrait as a magenta star.

   (e) To get an idea of how much variation there is between the different stochastic simulations at the final time step (i.e. at $T + 1$), lines 58–64 generate histograms of the values of each sequence at the final time step across all N simulations. In Matlab, `histogram` is a built-in function. The general structure is

```
histogram(X,bins)
```



Command Window

```
>> L04Ex02
EQ 1 is (0,0)
EQ 2 is (2.40,2.50)
X(T+1) standard deviation is 0.883
Y(T+1) standard deviation is 0.523
```

Figure 2: Plots produced from `L04Ex02.m` script and corresponding output in Command Window.

where `X` is a one-dimensional array and bins is the number of divisions (or bars) in which you wish to divide the data within the array `X`.

For the histogram generated in line 60, we use the one-dimensional array `X(:,T+1)` which contains all the rows of column $T + 1$, and uses 10 bins. Note the colon in the first argument is used to indicate that all rows should be included. If we used a colon in the second argument, it would be to indicate that all columns should be included.

To include both the histogram for `X` and the histogram for `Y` on the same graph, we use the `hold on` command. Additionally, to distinguish one histogram from the other, we use the `'facecolor'` option in each `histogram` command along with the `xlabel` function on line 63 and the `legend` function on line 64.

One line 62, we plot vertical dashed lines to indicate where values of x^* and y^* are. Notice the colors for each line corresponds to the colors used for the histograms.

(f) Lastly, to have a measure of how much variation there is between the different stochastic simulation at the final time step (i.e. $T + 1$), lines 66–68 calculate and print the standard deviation of the one-dimensional arrays `X(:,T+1)` and `Y(:,T+1)`. Note, the more spread out the histograms generated in lines 60 and 61, the larger the standard deviations will be (i.e. a higher standard deviation at the final time means that the stochasticity has lead to a wider range of population densities at the final time).

(g) Take another moment to review `L04Ex02.m`. **Do you understand what each line of code is doing?** If not, now is the time to stop and ask.

3. Run the script file. The resulting graphs should look similar to Figure 2.

Notice that the stochastic simulation shown is not smooth, though the general pattern of oscillations can still be seen. The histograms on the bottom along with the output from the Command Window reveal that, at the final time, there is more variation in the prey population density than in the predator population density, and that there is overlap in the distribution of the population densities. More over, the predator density at the final density is more likely to be below y^* than above (this can be seen from most of the blue histogram bars being to the left of the dashed blue line). However, the distribution of prey densities at the final time seem centered around x^* (this can be seen from the red histograms bars having roughly 5 bars to the left of the dashed red line and 5 bars to the right).

Lab Group Discussion

Week 1:

1. Suppose you want to set the variable `c` to the value 4.2, and variable `w` to the array containing the integers from 0 to 10 in ascending order. Write one line of MATLAB code which accomplishes both of these tasks.
2. Write a line of Matlab code to generate 1000 samples from the uniform distribution 0.837 ± 0.05. Store the array of values in a variable called `P`.
3. Using the array `P` generated in the question above, write the MATLAB code necessary to generate a histogram of the sampled values.
    (a) Try the histogram with 10, 20, and 50 bins. Describe how the appearance of the histogram changes when the number of bins changes.
    (b) Do the samples appear "uniformly" distributed?
4. Recall the Beverton-Holt model from the Reference Readings and from Lab03. Suppose, that the prey population, in the absence of the predator population, could be modeled with the Beverton-Holt equation. Thus, consider the system of difference equations

$$x_{t+1} = \frac{rx_t}{1 + \frac{r-1}{K}x_t} - \alpha x_t y_t \qquad y_{t+1} = sy_t + \beta x_t y_t, \tag{2}$$

   where x_t is the prey density, y_t is the predator density, $r > 1$ is the intrinsic growth rate of the prey, $0 < s < 1$ is the survival rate of the predator population, K is the carrying capacity of the prey population in the absence of the predator population, $\alpha > 0$ is the consumption rate of the predator, and $\beta > 0$ is the growth rate of the predator population due to the consumption of prey. This system has three equilibria: the trivial equilibrium, one of the form $(x^*, 0)$ (the predator extinction equilibrium), and one of the form (x^*, y^*) (the coexistence equilibrium).
    (a) What is x^* in the predator extinction equilibrium?
    (b) Can x^* ever be negative? Why or why not?
    (c) What are x^* and y^* in the coexistence equilibrium?
    (d) Can either x^* or y^* ever by negative? Why or why not?
    (e) What happens if $K = x^*$ from the coexistence equilibrium?

Week 2:

1. In `L04Ex02.m` how are arrays `x` and `X` different in terms of their dimensions?
2. Why do we need two x and two y arrays in `L04Ex02.m`, i.e. why have `x` and `X`, and `y` and `Y`?
3. Describe the array that line 45 in `L04Ex02.m` creates.
4. In `L04Ex02.m`, change the number of bins of each histogram to 20 and run the code. How does this change the appearance (and perhaps the perception) of the distribution of prey and predator densities at the final time? Are you still able to discern the same general trends?

5. Notice that each time your run `L04Ex02.m`, the standard deviation value reported in the Command Window changes. Why does this happen?

6. Execute `L04Ex02.m` at least three times. On the board, each student present at the lab tutorial should write down the x and y standard deviation from two executions. Calculate the range in the standard deviations for x and y.

7. In `L04Ex02.m`, increase the number of simulations to 500. Execute the code at least three times. On the board, each student present at the lab tutorial should write down the x and y standard deviation from two executions. Calculate the range in the standard deviations for x and y. Are the ranges smaller or larger than when only 100 simulations were run? Why?

8. When running stochastic simulations, what are the benefits and drawbacks of running more simulations?

Checkpoint Exercise

1. Write a MATLAB script similar to `L04Ex01.m` to simulate the model in System (2) for 500 years. Save your script file as `PLab04_YourLastName.m`. Use the same parameter values for r, s, α, β, x_0, and y_0 as in `L04Ex01.m`, and additionally use $K = 2.6$.

   The script should print the locations of all three equilibrium points, generate a time series plot of the predator and prey sequences generated by System 2, and generate a phase portrait in the xy-plane with marking indicating the initial point and the locations of the nontrivial equilibria.

   Run the script. Save the figure produced from this simulation as `PLab04_YourLastName_A.jpg`.

   (a) Which equilibrium point are these simulations approaching?

2. Change the carrying capacity for the prey population to a density of $K = 2.1$. Run the script. Save the figure produced from this simulation as `PLab04_YourLastName_B.jpg`.

   Notice that in Part 1, K is larger than the x^* of the coexistence equilibrium, but in this part (Part 2) the value of K has been lowered to be smaller than the x^* of the coexistence equilibrium.

   (a) Which equilibrium point are the simulations in this part approaching?

   (b) How has the change in K impacted the ultimate "fate" of each population?

   (c) What is the location of the coexistence equilibrium point? Is this biologically reasonable?

3. Change the carrying capacity for the prey population to a density of $K = 2.4$. Run the script. Save the figure produced from this simulation as `PLab04_YourLastName_C.jpg`.

   (a) Which equilibrium point are the simulations in this part approaching?

   (b) What is the location of the coexistence equilibrium point? Is this biologically reasonable?

4. CHECKPOINT: Verify that your code is running correctly by showing your three saved images to the lab tutor. Additionally, if you unclear about how to answer the questions posed in Parts 1–3, please ask your lab tutor to go over those questions with you.

Lab Assignment

Suppose that the parameters r, s, α, and β in System (2) varied from year to year due to stochastic environmental conditions.

1. Write a MATLAB script similar to `L04Ex02.m` to simulate the model in System (2) for 500 years. Save your script file as `Lab04_YourLastName.m`. Use the same base parameter values for r, s, α, β, x_0, and y_0 as in `L04Ex02.m`, and additionally use $K = 2.6$. Note, for the stochastic simulations, the parameters values are taken from the following uniform distributions

   - $r = 1.0015 \pm 0.002$
   - $s = 0.9994 \pm 0.0002$
   - $\alpha = 0.0006 \pm 0.005$
   - $\beta = 0.00025 \pm 0.005$
   - $K = 2.6 \pm 0.5$

   The script should print the locations of all three equilibrium points, generate a time series plot of the predator and prey sequences generated by System 2 (the non-stochastic time series and the last generated stochastic time series), and generate a phase portrait in the xy-plane with marking indicating the initial point and the locations of the nontrivial equilibria. The phase portrait should plot the non-stochastic output, and the last generated stochastic output. Additionally, the script should generate a histogram of the prey and predator densities at the last time step across all simulations, and print out the standard deviation of the prey and predator densities at the last time step across all simulations.

   Run the script at least 3 times to see how the stochasticity changes the output each time. Note it may take a few moments for the script to fully run (remember that it is running 100 simulations of a model with 500×365 times steps each). For one run of the script, save the figure produced as `Lab04_YourLastName_A.jpg`.

   CHECKPOINT: Verify that your code is producing reasonable results by showing the tutor the figure you just generated.

2. Change the carrying capacity for the prey population to a density of $K = 2.1$. Run the script at least 3 times to see how the stochasticity changes the output each time. For one run of the script, save the figure produced as `Lab04_YourLastName_B.jpg`.

3. Change the carrying capacity for the prey population to a density of $K = 2.4$. Run the script at least 3 times to see how the stochasticity changes the output each time. For one run of the script, save the figure produced as `Lab04_YourLastName_C.jpg`.

4. **Submit:**

   - 1 script file (`Lab04_YourLastName.m`)
   - 1 Word document (`Lab04_YourLastName.m`) summarizing the results of your simulations. Use the template provided on Moodle. Your image files will be included in this Word document and should NOT be submitted separately.