Header:

Author: Crissy Hendrickson

Title: Lotka-Volterra Predator-Prey System

Group Members: Erik Herslebs

Analysis of the Problem:

The interaction between predators and prey in an area can be modeled by the Lotka-Volterra predator-prey system. According to the model, the rate of change of the population of predators is dependent upon the mass action between the predators and the prey and the predator death. At the same time, the rate of change of the population of prey is dependent upon their growth and death, and the mass action between the prey and the predators. In python, the system of differential equations can be solved by using odeint from SciPy or by writing integrators such as the Euler method or Runge-Kutta-2. A graph of the predator and prey populations versus time can be expected to be composed of oscillations because of the negative feedback with a time delay that impacts the population numbers. However, at certain fixed points the predator and prey populations can be expected to level out and not oscillate because the rates of change of the populations are zero. When the populations oscillate, the phase portrait, which shows the rates of change of the predator and prey populations at different population numbers, can be expected to be a spiral which rotates clockwise outward from the fixed point. But when the populations do not oscillate and approach a fixed point, the phase portrait will be mostly composed of lines pointing in the same direction. A phase diagram is a summary of the phase portraits and shows for which parameter values the system oscillates and for which values it approaches a fixed point.

Model Design:

Computational models are simplified representations of phenomena, systems, or processes. They are used to help us study and understand different aspects of life and can lead to predictions involving responses in un-tested scenarios. Often because of certain assumptions and simplifications, they are much easier to work with than the systems they represent and can point to new insight and comprehension. To translate a problem into a computational model there are five main steps: analysis of a problem, model development and formulation, model implementation, model verification, and model interpretation. The first step, analysis of a problem, involves determining what the question and unknowns are and what the answer would look like. The second step, model development and formulation, includes gathering relevant data, making assumptions, determining variable relationships and units, and writing down specific equations or rules. Model implementation involves writing a program in a computer language to solve the problem while model verification involves testing special cases whose answers are known to confirm the solution is accurate and the program does what it is supposed to do. Finally, model interpretation includes answering the main question sometimes with figures or tables and discussing more about the solution: if it was expected, what it means and how it could change if the assumptions are relaxed.

In the model for the Lotka-Volterra predator-prey system, several assumptions were made before the model was implemented. The model was assumed to be dynamic since it depends on time and deterministic since there is no element of chance and the solution is expected to be the same under the same conditions. It is also assumed to be a point model since the solution does not depend on the location in space. Because the individual predators and prey will be interacting, dying, and reproducing at different times and taking up all available space, time and space are assumed to be continuous in the model. Furthermore, the number of predators and prey is assumed to be continuous because the amount of each population can include all points on the number line. In the model, it is also assumed that the predators only eat the prey and that the populations reflect the number of females so sexual reproduction can be accounted for. The initial number of predators was chosen to be 20 x 10,000 females and the initial number of prey was chosen to be 20 x 10,000 females so that even if the solution involved a fraction, the fraction would still represent a whole organism. Additionally, the initial number of each

population is assumed to be large enough so that the predators and prey will interact, die, and reproduce even if the period of time is small. The rate of interaction between the predators and the prey was arbitrarily chosen to be .01 day^{-1}x100000 $females^{-1}$ while the rate of interaction between the prey and predators was arbitrarily chosen to be .05 day^{-1}x10000 $females^{-1}$. Furthermore, the death rate of the prey was arbitrarily chosen to be .1 day^{-1}, the death rate of the predators was picked to be .2 day^{-1}, and the growth rate of the prey was chosen to be .7 day^{-1}. The carrying capacity of the prey was picked to be 10000 x10000 females and the simulation time was chosen to be 100 days with a step size of .1 day so the simulation would run long enough for trends on the graph to be determined.

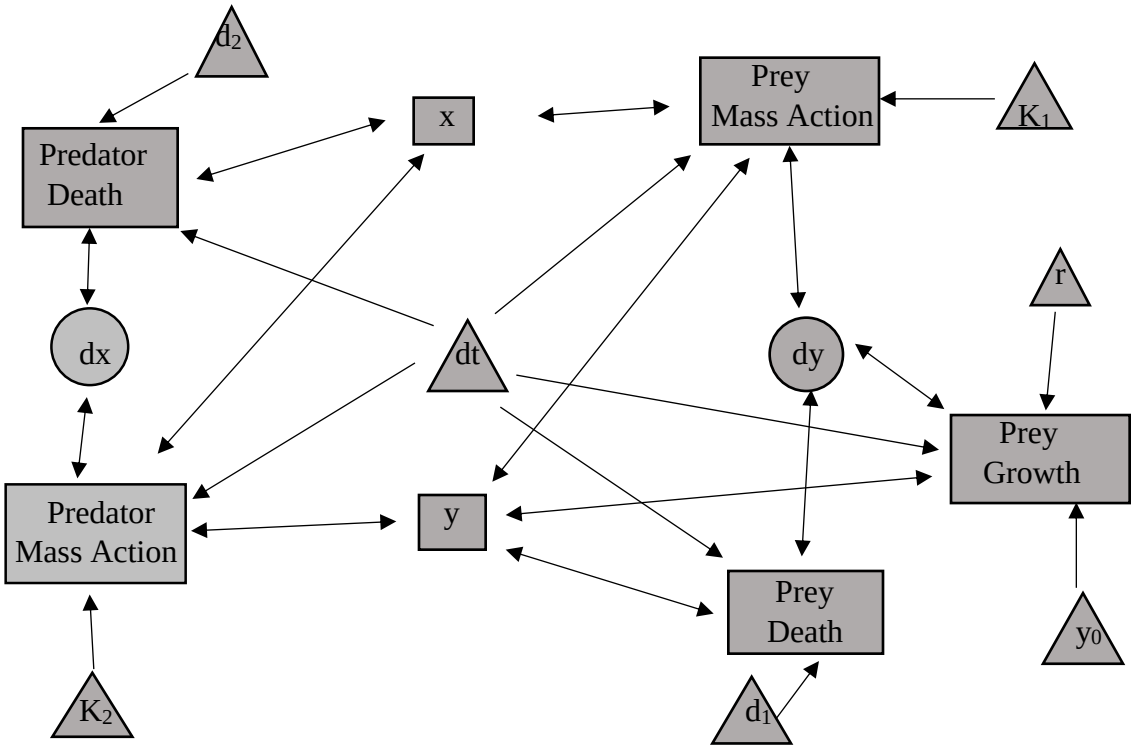
In the model development, several variables and equations were used to help find the solution. The rate of interaction between the predators and the prey, K_2 day^{-1}x100000 $females^{-1}$, the death rate of the predators, d_2 day^{-1}, and the step size, dt day, are constants. Additional constants include the rate of interaction between the prey and predators, K_1 day^{-1}x10000 $females^{-1}$, the death rate of the prey, d_1 day^{-1}, the growth rate of the prey, r day^{-1}, and the carrying capacity of the prey, y_0 x10000 females. The number of predators, x x10000 females, is a container variable as is the number of prey, y x10000 females. Furthermore, the change in number of predators, dx x10000 females and the change in number of prey, dy x10000 females are dependent variables. Using the previously described variables, a system of differential equations can be used to represent the rate of change of the predator and prey populations over time. The first equation

$$\frac{dx}{dt} = K_2xy - d_2x \qquad (1)$$

describes the rate of change of the predator population where K_2xy is the mass action of the predators and d_2x is the death of the predators. The second equation

$$\frac{dy}{dt} = ry\left(1 - \frac{y}{y_o}\right) - K_1xy - d_1y \qquad (2)$$

describes the rate of change of the prey population where $ry(1-\frac{y}{y_o})$ is the growth of the prey, K_1xy is the mass action of the prey, and d_1y is the death of the prey. The two equations can then be solved numerically in Python to give the predator and prey populations at different times. Overall, the variables and equations are represented by the following diagram.

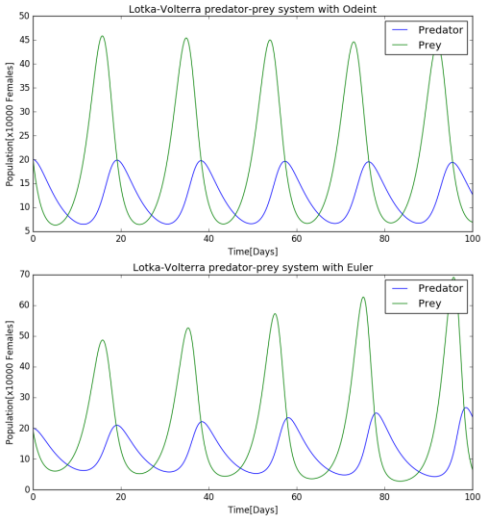


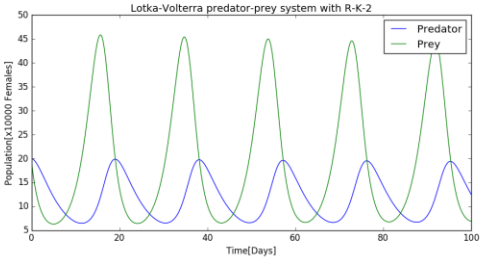
Model Solution:

To find the solution, the code at the end of the report was run in python. First, initial values were defined. Then a method was written which used odeint from SciPy to solve Eq.(1) and (2) numerically and return values of time and the corresponding populations. Additionally, methods were written that used R-K-2 and Euler to solve Eq.(1) and (2) numerically and return values of time and the corresponding populations. The R-K-2 and Euler methods both used for loops to find the populations at each time in the model. Graphs were then made using all three of these methods so the solutions could be compared. Once the graphs of the populations versus time were made, a phase portrait of the system was created using plt.quiver. Specifically, a method was written that returned an array of x and y values and the vectors at each point so plt.quiver could be used. Numerous phase portraits and the corresponding population graphs were then created using plt.quiver and the odeint method for varying K_2 and r values. Then using the phase portraits a phase diagram was created using plt.scatter that showed for which parameter values the system oscillated and for which values it approached a fixed point. To verify the solution, plots were created using the odeint method with different initial values and constants. They were then compared with the original plot that used the odeint method of populations versus time to see if the new plots behaved as expected.

Results, Verification, and Conclusion:

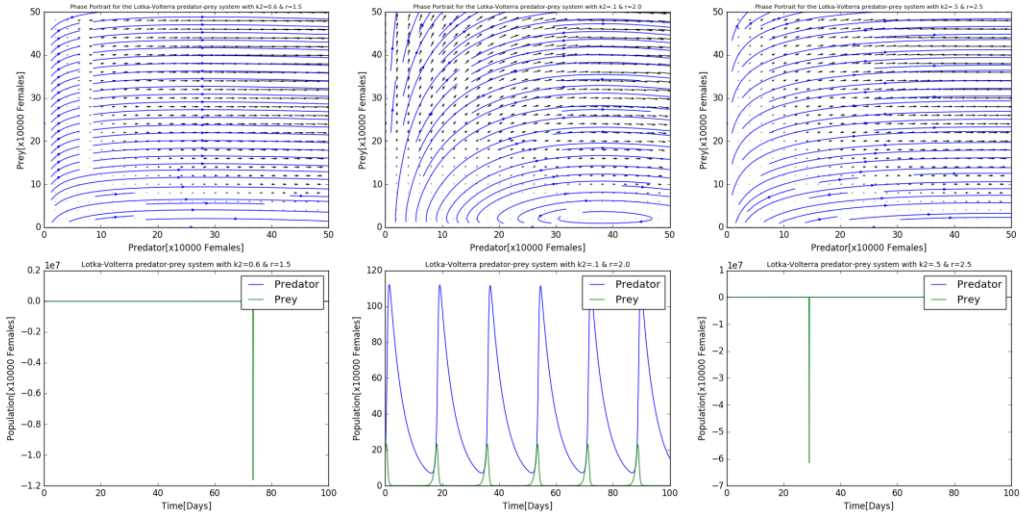
The following graphs show the predator and prey population numbers over time. The first graph was created using the odeint method, the second graph was created using R-K-2 and the third graph was created using the Euler method.





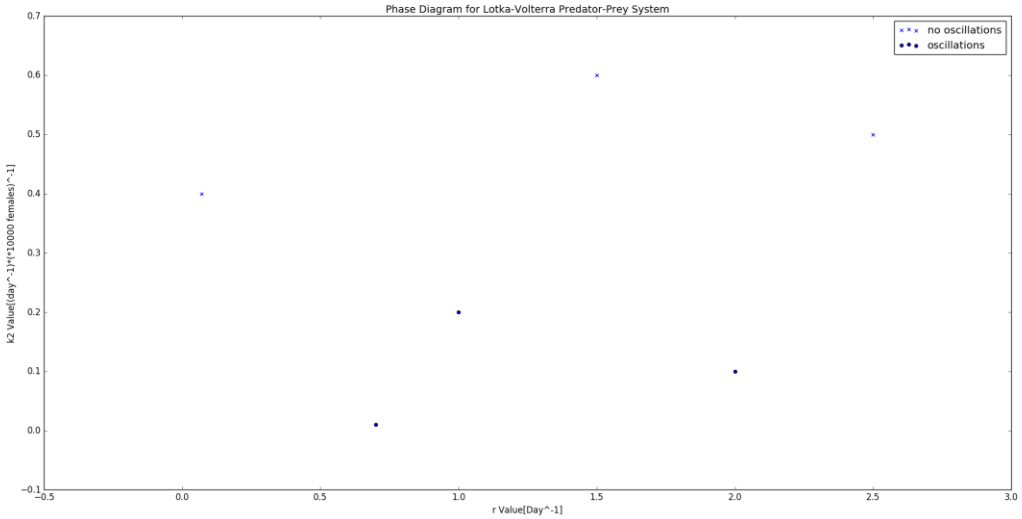
The data produced results that were expected because the populations oscillate with time because of the negative feedback with a time delay that impacts the population numbers. Additionally, all three integrators produce the same general trends, but in the graph that uses the Euler method the predator and prey populations are slightly larger at larger time values. However, according to the model the populations should oscillate around the same values and should not generally increase over time. The larger error then occurs in the third graph because when the Euler method is used error has a linear relationship with step size. Therefore, it is not as accurate as R-K-2 or the odeint which can have errors that decrease by the square of the step size depending on whether the higher order derivatives of the function exist. Next time, to improve results, all the graphs could use a smaller step size, dt day, so there are more data points and each error is even smaller than before.

Using various K_2 and r values, phase portraits and the corresponding population graphs were also created to see how the system behaved under different conditions. The variables K_2 day^{-1}x100000 females^{-1}and r day^{-1} were varied because they were determined to have a large effect on how the system behaved.



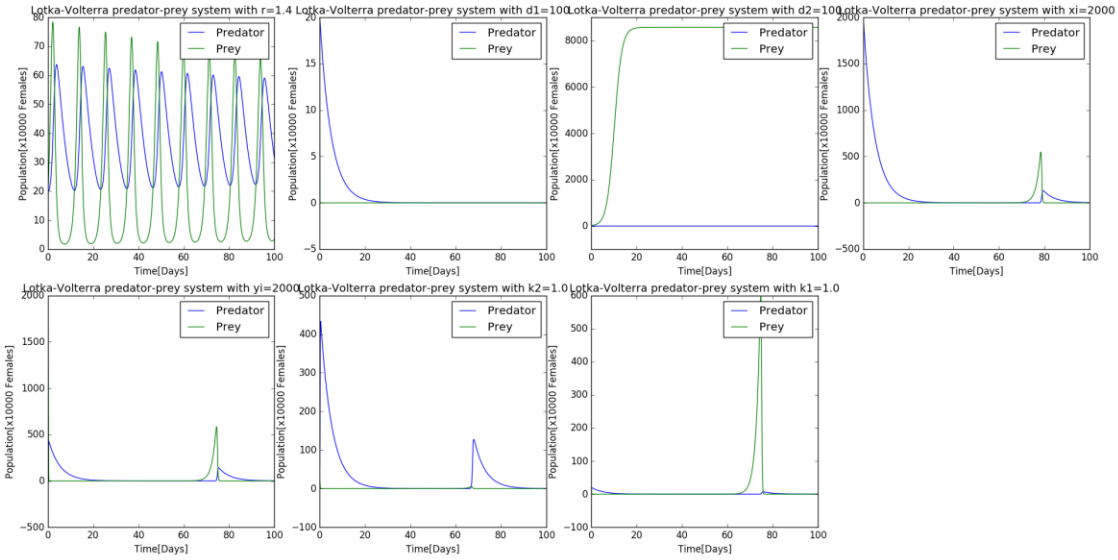
As shown by the graphs for certain K_2 and r values, the system oscillates while at other values it approaches a fixed point which occurs when the derivatives are equal to zero. Setting Eq.(1) and (2) equal to zero show the fixed points should occur when x is either zero or approximately r/K_1 and y is either zero or d_2/K_2. The phase portraits then behave as expected because when the populations oscillate, they create a spiral which rotates clockwise outward from the fixed point. Additionally, when the populations do not oscillate and approach a fixed point, the phase portrait is mostly composed of lines pointing in the same direction.

Using the data from the phase portraits and the corresponding population graphs, a phase diagram was created to show the values at which the system oscillates and the values at which it approaches a fixed point.



As shown by the graph when r =.7, 1.0, and 2.0 day^{-1} and k = .01, .2, and .1 day^{-1}x100000 $females^{-1}$ the system oscillates. But when r = .07, 1.5, and 2.5 day^{-1} and k = .4, .6, and .5 day^{-1}x100000 $females^{-1}$ the system does not oscillate. At some unknown value between the points that oscillate and the points that do no the system transitions between the fluctuations and approaching a fixed point.

To verify the solution, plots were created with different initial values and constants and compared with the original plot to see if the system behaved as expected.



When the growth rate of the prey is doubled, the prey population can be expected to also approximately double and as a result the predator population can be expected to increase. If the death rate of the prey is increased to 100 day^{-1} it can be expected that with time both the predator and prey populations will approach zero. Additionally, if the death rate of the predator is increased to 100 day^{-1} it can be expected that the predator population will approach zero over time will the prey population will approach the carrying capacity. When the initial predator population, xi x10000 females, is multiplied by 100, both the predators and the prey can be expected by approach zero over time. Furthermore, when the initial prey population, yi x10000 females, multiplied by 100 it can also be expected that over time

both the populations will approach zero. Finally, if the rate of interaction between the predators and prey, K_2 day^{-1}x10000 females^{-1}, is increased to 1.0 it can be expected that both the populations over time will approach zero. And if the rate of interaction between the prey and predators, K_1 day^{-1}x10000 females^{-1}, is increased to 1.0 it can also be expected that both the populations will approach zero over time. Since all the plots match what was expected the solution is verified.

Code:

```
# -*- coding: utf-8 -*-
"""
Created on Sun Feb 19 16:05:57 2017

@author: Crissy
"""
#%%
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint
#in the simulation prey is grass *10000 females,
#and predator is rabbits*10000 females
k2 = 0.01#the rate the predators interact with the prey
#in (day^-1)*(*10000 females)^-1
k1 = 0.05#the rate the prey interact with the predators
#in (day^-1)*(*10000 females)^-1
d1 = 0.1#death rate of the prey in day^-1
d2 = 0.2#death rate of the predator in day^-1
r = 0.7#growth rate of the prey in day^-1
y0 = 10000#carrying capacity of the prey in *10000 females
dt=.1#step size, days
xi=20#initial predator population, *10000 females
yi=20#initial prey population, *10000 females
Ti=0#model start time in days
Tf=100#model end time in days
#%%
#Odeint
#solves the system of differential equations numerically using Odeint
#returns the time and the populations
def Odeint(k1, k2, d1, d2, r, y0, xi, yi, Ti, Tf, dt):
    #defines a function Odeint
    t=np.arange(Ti,Tf +dt, dt)#creates an array of the values of time
     to use
    initial_conditions = [xi,yi]#defines the initinal conditions
    def dX_dt(X_odeint, t=0):#creates an array of the populations
        return np.array([(k2*X_odeint[0]*X_odeint[1])-\
        (d2*X_odeint[0]), \
        ((r*X_odeint[1])*(1-(X_odeint[1]/y0)))-\
        (k1*X_odeint[0]*X_odeint[1])-(d1*X_odeint[1])])
    X_odeint, infodict =odeint(dX_dt,initial_conditions,t,
    full_output=True)
```

```
    return [t, X_odeint]

t, X_odeint = Odeint(k1, k2, d1, d2, r, y0, xi, yi, Ti, Tf, dt)
plt.subplot(2,2,1)
plt.plot(t,X_odeint)#plots the populations v. time
plt.legend(["Predator", "Prey"])
plt.xlabel('Time[Days]', size=12)
plt.ylabel('Population[x10000 Females]', size=12)
plt.title('Lotka-Volterra predator-prey system with Odeint', size=14)

#%%
#R-K-2
#solves the system of differential equations numerically using R-K-2
#returns the time and the populations
def RK2(k1, k2, d1, d2, r, y0, xi, yi, Ti, Tf, dt):
#defines a R-K-2 function
    X_RK2 = []
    #creates an empty array to fill with the values of the predator
#population
    T_RK2 = np.arange(Ti,Tf+dt,dt)
    #creates an array of the values of time to use
    Y_RK2= []
    #creates an empty array to fill with the values of the prey
#population
    x_RK2 = xi#defines the initinal predator population
    y_RK2 = yi#defines the initinal prey population
    for t in np.arange(Ti,Tf+dt,dt):#implements R-K-2
        dx=((k2*x_RK2*y_RK2)-(d2*x_RK2))*dt
        dy=(((r*y_RK2)*(1-(y_RK2/y0))) - (k1*x_RK2*y_RK2)-\
       (d1*y_RK2))*dt
        xguess=x_RK2+dx
        yguess=y_RK2+dy
        fxdx = ((k2*xguess*yguess)-(d2*xguess))*dt
        fydy = (((r*yguess)*(1-(yguess/y0))) - (k1*xguess*yguess)-\
        (d1*yguess))*dt
        x_RK2= x_RK2 + 0.5*(dx+fxdx)
        y_RK2= y_RK2 + 0.5*(dy+fydy)
        X_RK2.append(x_RK2)
        Y_RK2.append(y_RK2)
    return [T_RK2, X_RK2, Y_RK2]

plt.subplot(2,2,2)
T_RK2, X_RK2, Y_RK2 = RK2(k1, k2, d1, d2, r, y0, xi, yi, Ti, Tf, dt)
plt.plot(T_RK2,X_RK2, label='Predator')
#plots the predator population v. time
plt.plot(T_RK2,Y_RK2, label='Prey')#plots the prey population v. time
plt.xlabel('Time[Days]', size=12)
```

```
plt.ylabel('Population[x10000 Females]', size=12)
plt.title('Lotka-Volterra predator-prey system with R-K-2', size=14)
plt.legend()

#%%
#Euler Method
#solves the system of differential equations numerically using Euler
#returns the time and the populations
def Euler(k1, k2, d1, d2, r, y0, xi, yi, Ti, Tf, dt):
#defines a Euler function
    X_Euler = []
    #creates an empty array to fill with the values of the predator
#population
    T_Euler = np.arange(Ti,Tf+dt,dt)
    #creates an array of the values of time to use
    Y_Euler= []
    #creates an empty array to fill with the values of the prey
#population
    x_Euler=xi#defines the initinal predator population
    y_Euler=yi#defines the initinal prey population
    for t in np.arange(Ti,Tf+dt,dt):#implements Euler
        dx=((k2*x_Euler*y_Euler)-(d2*x_Euler))*dt
        dy=(((r*y_Euler)*(1-(y_Euler/y0))) - \
        (k1*x_Euler*y_Euler)-(d1*y_Euler))*dt
        x_Euler=x_Euler+dx
        y_Euler=y_Euler+dy
        X_Euler.append(x_Euler)
        Y_Euler.append(y_Euler)
    return[T_Euler,X_Euler,Y_Euler]
T_Euler,X_Euler,Y_Euler=Euler(k1, k2, d1, d2, r, y0, xi, yi, Ti, Tf,
dt)
plt.subplot(2,2,3)
plt.plot(T_Euler,X_Euler, label="Predator")
#plots the predator population v. time
plt.plot(T_Euler,Y_Euler, label="Prey")
#plots the prey population v. time
plt.xlabel('Time[Days]', size=12)
plt.ylabel('Population[x10000 Females]', size=12)
plt.title('Lotka-Volterra predator-prey system with Euler', size=14)
plt.legend()
#%%
#makes a phase portrait of the system
def Phase_Portrait(k1, k2, d1, d2, r, y0):
    coords=np.linspace(0, 50, 26)
    xe, ye = np.meshgrid(coords, coords)
    dx=((k2*xe*ye)-(d2*xe))
    dy=(((r*ye)*(1-(ye/y0))) - (k1*xe*ye)-(d1*ye))
```

```
    Vectordxdt, Vectordydt= dx, dy
    return [xe, ye, Vectordxdt, Vectordydt]
xe, ye, Vectordxdt, Vectordydt = Phase_Portrait(k1, k2, d1, d2, r,
y0)
plt.figure()
plt.quiver(xe, ye, Vectordxdt, Vectordydt)#plots the phase portrait
plt.title('Phase Portrait for the Lotka-Volterra predator-prey
system', size=14)
plt.xlabel('Predator[x10000 Females]', size=12)
plt.ylabel('Prey[x10000 Females]', size=12)
plt.streamplot(xe, ye, Vectordxdt, Vectordydt)
plt.xlim(0,50)
plt.ylim(0,50)
#%%
#Phase Portrait and corresponding population graphs
#for various parameter values
#k2=.4, r=.07
xe, ye, Vectordxdt, Vectordydt = Phase_Portrait(k1, .4, d1, d2, .07,
y0)
plt.subplot(2,3,1)
plt.quiver(xe, ye, Vectordxdt, Vectordydt)
plt.title('Phase Portrait for the Lotka-Volterra predator-prey system
with k2=.4 & r=.07', size=10)
plt.xlabel('Predator[x10000 Females]', size=12)
plt.ylabel('Prey[x10000 Females]', size=12)
plt.streamplot(xe, ye, Vectordxdt, Vectordydt)
plt.xlim(0,50)
plt.ylim(0,50)
t, X_odeint = Odeint(k1, .4, d1, d2, .07, y0, xi, yi, Ti, Tf, dt)
plt.subplot(2,3,4)
plt.plot(t,X_odeint)
plt.legend(["Predator", "Prey"])
plt.xlabel('Time{Days]', size=12)
plt.ylabel('Population[x10000 Females]', size=12)
plt.title('Lotka-Volterra predator-prey system with k2=.4 & r=.07',
size=12)
#k2=.01, r=.7
xe, ye, Vectordxdt, Vectordydt = Phase_Portrait(k1, .01, d1, d2, .7,
y0)
plt.subplot(2,3,2)
plt.quiver(xe, ye, Vectordxdt, Vectordydt)
plt.title('Phase Portrait for the Lotka-Volterra predator-prey system
with k1=.01 & r=.7', size=10)
plt.xlabel('Predator[x10000 Females]', size=12)
plt.ylabel('Prey[x10000 Females]', size=12)
plt.streamplot(xe, ye, Vectordxdt, Vectordydt)
plt.xlim(0,50)
```

```
plt.ylim(0,50)
t, X_odeint = Odeint(.01, k2, d1, d2, .7, y0, xi, yi, Ti, Tf, dt)
plt.subplot(2,3,5)
plt.plot(t,X_odeint)
plt.legend(["Predator", "Prey"])
plt.xlabel('Time[Days]', size=12)
plt.ylabel('Population[x10000 Females]', size=12)
plt.title('Lotka-Volterra predator-prey system with k2=.01 & r=.7',
size=12)
#k2=.2, r=1.0
xe, ye, Vectordxdt, Vectordydt = Phase_Portrait(k1, 0.2, d1, d2, 1.0,
y0)
plt.subplot(2,3,3)
plt.quiver(xe, ye, Vectordxdt, Vectordydt)
plt.title('Phase Portrait for the Lotka-Volterra predator-prey system
with k2=0.2 & r=1.0', size=8)
plt.xlabel('Predator[x10000 Females]', size=12)
plt.ylabel('Prey[x10000 Females]', size=12)
plt.streamplot(xe, ye, Vectordxdt, Vectordydt)
plt.xlim(0,50)
plt.ylim(0,50)
t, X_odeint = Odeint(k1, .2, d1, d2, 1.0, y0, xi, yi, Ti, Tf, dt)
plt.subplot(2,3,6)
plt.plot(t,X_odeint)
plt.legend(["Predator", "Prey"])
plt.xlabel('Time[Days]', size=12)
plt.ylabel('Population[x10000 Females]', size=12)
plt.title('Lotka-Volterra predator-prey system with k2=0.2 & r=1.0',
size=10)
#k2=.6, r=1.5
xe, ye, Vectordxdt, Vectordydt = Phase_Portrait(k1, 0.6, d1, d2, 1.5,
y0)
plt.figure()
plt.subplot(2,3,1)
plt.quiver(xe, ye, Vectordxdt, Vectordydt)
plt.title('Phase Portrait for the Lotka-Volterra predator-prey system
with k2=0.6 & r=1.5', size=8)
plt.xlabel('Predator[x10000 Females]', size=12)
plt.ylabel('Prey[x10000 Females]', size=12)
plt.streamplot(xe, ye, Vectordxdt, Vectordydt)
plt.xlim(0,50)
plt.ylim(0,50)
t, X_odeint = Odeint(k1, 0.6, d1, d2, 1.5, y0, xi, yi, Ti, Tf, dt)
plt.subplot(2,3,4)
plt.plot(t,X_odeint)
plt.legend(["Predator", "Prey"])
plt.xlabel('Time[Days]', size=12)
```

```
plt.ylabel('Population[x10000 Females]', size=12)
plt.title('Lotka-Volterra predator-prey system with k2=0.6 & r=1.5',
size=10)
#k2=.1, r=2.0
xe, ye, Vectordxdt, Vectordydt = Phase_Portrait(k1, .1, d1, d2, 2.0,
y0)
plt.subplot(2,3,2)
plt.quiver(xe, ye, Vectordxdt, Vectordydt)
plt.title('Phase Portrait for the Lotka-Volterra predator-prey system
with k2=.1 & r=2.0', size=8)
plt.xlabel('Predator[x10000 Females]', size=12)
plt.ylabel('Prey[x10000 Females]', size=12)
plt.streamplot(xe, ye, Vectordxdt, Vectordydt)
plt.xlim(0,50)
plt.ylim(0,50)
t, X_odeint = Odeint(k1, .1, d1, d2, 2.0, y0, xi, yi, Ti, Tf, dt)
plt.subplot(2,3,5)
plt.plot(t,X_odeint)
plt.legend(["Predator", "Prey"])
plt.xlabel('Time[Days]', size=12)
plt.ylabel('Population[x10000 Females]', size=12)
plt.title('Lotka-Volterra predator-prey system with k2=.1 & r=2.0',
size=10)
#k2=.5,r=2.5
xe, ye, Vectordxdt, Vectordydt = Phase_Portrait(k1, .5, d1, d2, 2.5,
y0)
plt.subplot(2,3,3)
plt.quiver(xe, ye, Vectordxdt, Vectordydt)
plt.title('Phase Portrait for the Lotka-Volterra predator-prey system
with k2=.5 & r=2.5', size=8)
plt.xlabel('Predator[x10000 Females]', size=12)
plt.ylabel('Prey[x10000 Females]', size=12)
plt.streamplot(xe, ye, Vectordxdt, Vectordydt)
plt.xlim(0,50)
plt.ylim(0,50)
t, X_odeint = Odeint(k1, .5, d1, d2, 2.5, y0, xi, yi, Ti, Tf, dt)
plt.subplot(2,3,6)
plt.plot(t,X_odeint)
plt.legend(["Predator", "Prey"])
plt.xlabel('Time[Days]', size=12)
plt.ylabel('Population[x10000 Females]', size=12)
plt.title('Lotka-Volterra predator-prey system with k2=.5 & r=2.5',
size=10)
#%%
#Phase Diagram for different k2 and r values
plt.figure()
plt.scatter(.07,.4, marker='x', label='no oscillations')
```

```
plt.scatter(.7,.01, marker='o', label='oscillations')
plt.scatter(1.0,.2, marker='o')
plt.scatter(1.5,0.6, marker='x')
plt.scatter(2.0,.1, marker='o')
plt.scatter(2.5,.5, marker='x')
plt.legend()
plt.xlabel('r Value[Day^-1]', size=12)
plt.ylabel('k2 Value[(day^-1)*(*10000 females)^-1]', size=12)
plt.title('Phase Diagram for Lotka-Volterra Predator-Prey System',
size=14)
#%%
#Verification of the Model
plt.figure()
#r=1.4
t, X_odeint = Odeint(k1, k2, d1, d2, 2, y0, xi, yi, Ti, Tf, dt)
plt.subplot(2,4,1)
plt.plot(t,X_odeint)
plt.legend(["Predator", "Prey"])
plt.xlabel('Time[Days]', size=12)
plt.ylabel('Population[x10000 Females]', size=12)
plt.title('Lotka-Volterra predator-prey system with r=1.4', size=14)
#d1=100
t, X_odeint = Odeint(k1, k2, 100, d2, r, y0, xi, yi, Ti, Tf, dt)
plt.subplot(2,4,2)
plt.plot(t,X_odeint)
plt.legend(["Predator", "Prey"])
plt.xlabel('Time[Days]', size=12)
plt.ylabel('Population[x10000 Females]', size=12)
plt.title('Lotka-Volterra predator-prey system with d1=100', size=14)
#d2=100
t, X_odeint = Odeint(k1, k2, d1, 100, r, y0, xi, yi, Ti, Tf, dt)
plt.subplot(2,4,3)
plt.plot(t,X_odeint)
plt.legend(["Predator", "Prey"])
plt.xlabel('Time[Days]', size=12)
plt.ylabel('Population[x10000 Females]', size=12)
plt.title('Lotka-Volterra predator-prey system with d2=100', size=14)
#xi=2000
t, X_odeint = Odeint(k1, k2, d1, d2, r, y0, 2000, yi, Ti, Tf, dt)
plt.subplot(2,4,4)
plt.plot(t,X_odeint)
plt.legend(["Predator", "Prey"])
plt.xlabel('Time[Days]', size=12)
plt.ylabel('Population[x10000 Females]', size=12)
plt.title('Lotka-Volterra predator-prey system with xi=2000',
size=14)
#yi=2000
```

```
t, X_odeint = Odeint(k1, k2, d1, d2, r, y0, xi, 2000, Ti, Tf, dt)
plt.subplot(2,4,5)
plt.plot(t,X_odeint)
plt.legend(["Predator", "Prey"])
plt.xlabel('Time[Days]', size=12)
plt.ylabel('Population[x10000 Females]', size=12)
plt.title('Lotka-Volterra predator-prey system with yi=2000',
size=14)
#k2=1.0
t, X_odeint = Odeint(k1, 1.0, d1, d2, r, y0, xi, yi, Ti, Tf, dt)
plt.subplot(2,4,6)
plt.plot(t,X_odeint)
plt.legend(["Predator", "Prey"])
plt.xlabel('Time[Days]', size=12)
plt.ylabel('Population[x10000 Females]', size=12)
plt.title('Lotka-Volterra predator-prey system with k2=1.0', size=14)
#k1=1.0
t, X_odeint = Odeint(1.0, k2, d1, d2, r, y0, xi, yi, Ti, Tf, dt)
plt.subplot(2,4,7)
plt.plot(t,X_odeint)
plt.legend(["Predator", "Prey"])
plt.xlabel('Time[Days]', size=12)
plt.ylabel('Population[x10000 Females]', size=12)
plt.title('Lotka-Volterra predator-prey system with k1=1.0', size=14)
```