1 Weight Change and Predator-Prey Models

Lab Objective: *We use IVP methods to study two dynamical systems. The first system is a weight change model based on thermodynamics and kinematics. The second model looks at the relationship between a predator and its prey.*

ODE solvers

Inital Value Problems, or IVPs, are a system of one or more ordinary differential equations (ODEs) with initial values. In more basic math courses, these problems could be solved by hand, but in real life that is very much not the case. For more complicated IVPs, numerical solvers are required. We will introduce the basic `ode` method in `scipy.integrate` and work through an example so you can understand how ODE solvers work to solve IVP problems. At the end of this lab you will see a similar method, `odeint`, in the same package.

The `ode` method can use many numerical approximations to solve. We will use the `dopri5` method, which runs the RK4 (Runge-Kutta method of order 4) iterative ODE solver.

Below you will see sample code for solving a differential equation.

Predator-Prey Model

One common problem to solve is the predator-prey model which involves two species, where one species (the prey) is the food source of the other species (the predator). For this example we will consider the predator to be wolves and the prey to be rabbits in Yellowstone National Park. Let `t` represent time, `r(t)` represents the rabbit population at time `t`, and `w(t)` represents the wolf population at time `t`.

We need to make a few assumptions in this model:

- In the absence of wolves,the rabbit population grows at a positive rate proportional to the current population. Thus $dr/dt = ar(t)$ when $w(t) = 0$, where $a > 0$.
- In the absence of rabbits, the wolves die out. Thus $dw/dt = -cw(t)$ when $r(t) = 0$, where $c > 0$.
- The number of encounters between rabbits and wolves is proportional to the product of their populations. Encounters encourage the growth of the wolf population and a decrease in the

rabbit population. The growth rate of rabbits is decreased by some $-\alpha r(t)w(t)$ and the growth rate of wolves is increased by $\gamma r(t)w(t)$

This leads us to the following system of ODEs (for simplicity of notation we will drop off `t`:

$$\begin{aligned} dr/dt &= ar - \alpha rw = r(a - \alpha w) \\ dw/dt &= -cw + \gamma rw = w(-c + \gamma r) \end{aligned} \tag{1.1}$$

Now we will run through this example. Start with the necessary imports.

```
from scipy.integrate import ode
import numpy as np
import matplotlib.pyplot as plt
```

Establish our initial conditions

```
r0 = 5 # Initial rabbit population
w0 = 3 # Initial wolf population

# Define rabbit growth paramters
a = 1.0
alpha = 0.5

# Define wolf growth parameters
c = 0.75
gamma = 0.25

t_f = 20 # How long we want to run the model
y0 = [r0, w0]

# Initialize time and output arrays needed for the ode solver
t = np.linspace(0, t_f, 5*t_f)
y = np.zeros((len(t), len(y0)))
y[0,:] = y0
```

Problem 1. Finish the following code box. You will create the function that takes as input the time, current `r` and `w` values, and all growth parameters.

```
def predator_prey(t, y, a, alpha, c, gamma):
    '''
    Parameters:
    --------------
    t:  time variable.
    y:  an array of length len(y0) representing current wolf and rabbit ↩
        populations at time t.
    a, alpha, c, gamma: growth parameters. These are keyword arguments and↩
         can be of any length.
```

```
    Return:
    --------
    Return a list corresponding to the Predator-Prey model.
    '''
    pass
```

We need to set up our ODE solver to use the RK4 numerical integrator and give it the correct initial values. Also note that the `predator_prey` function above must be turned into a `lambda` function.

```
predator_prey_ode = lambda t, y:predator_prey(t, y, a, alpha, c, gamma)
p_p_solver = ode(predator_prey_ode).set_integrator('dopri5') # set the ↩
    numerical integrator
p_p_solver.set_initial_value(y0, 0) # Set the initial values. The second ↩
    argument is the initial time, which we set to 0
```

The `ode` solver has an `integrate` method which solves one time step at a time. Iterate through the time array and update the `y` values at each step.

```
for j in range(1, len(t)):
    y[j,:] = p_p_solver.integrate((t[j]))
```

The following code can be used to graph the resulting system. See Figure 1.1.

```
plt.plot(t, y[:,0], label='rabbit')
plt.plot(t, y[:,1], label='wolf')
plt.legend()
plt.xlabel('Time')
plt.ylabel('Population')
plt.show()
```

Problem 2. Graph the wolf-rabbit model given in (1.1). It should look like Figure 1.1.

A weight change model

The main idea behind weight change is simple. If a person's energy intake is more than their energy expended, then they gain weight. If their intake is less, then they lose weight. Let the *energy balance* EB be the difference between *energy intake* EI and *energy expenditure* EE, so that

$$EB = EI - EE. \tag{1.2}$$

When the energy intake is greater than the energy expended, the balance is positive and weight is gained. Similarly, the balance is negative and weight is lost if the energy intake is less than the energy expended.

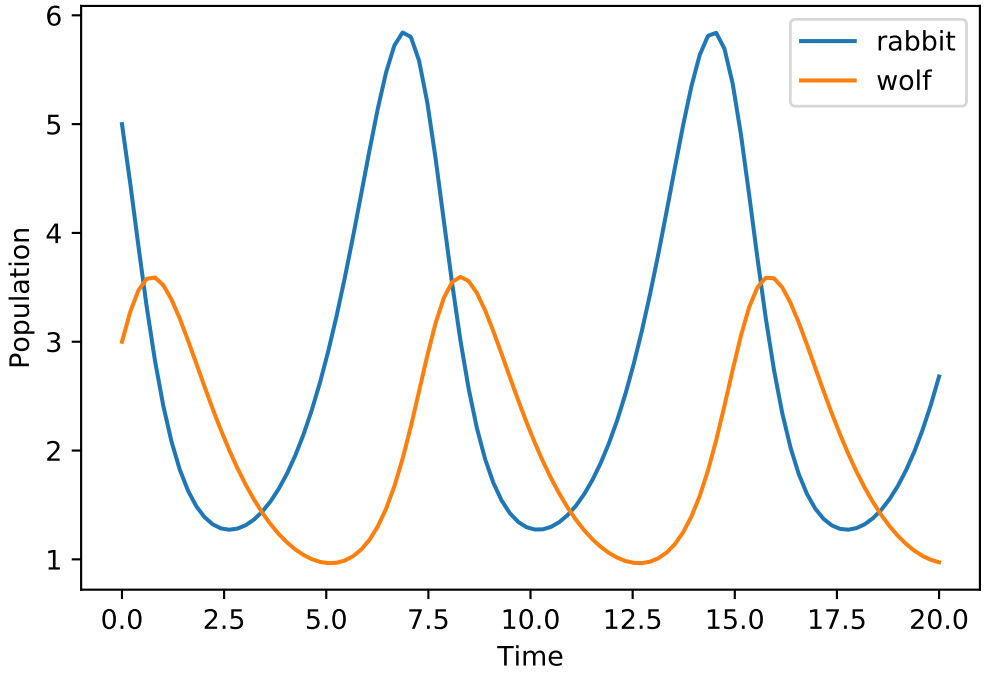

Figure 1.1: The solution to the system found in (1.1)

Body weight at time t is the sum of the weight of fat and lean tissue; that is, $BW(t) = F(t) + L(t)$. These quantities can be described by the compartmental model

$$\begin{aligned} \rho_F \frac{dF(t)}{dt} &= (1 - p(t))EB(t), \\ \rho_L \frac{dL(t)}{dt} &= p(t)EB(t), \end{aligned} \tag{1.3}$$

where $p(t)$ and $1 - p(t)$ represent the proportion of the energy balance ($EB(t)$) that results in a change in the quantity of lean or fatty tissue, respectively. Constants ρ_L and ρ_F represent the energy density of lean and fatty tissue (about 1800 and 9400 kcal/kg).

Next we need to find expressions for $p(t)$ and $EB(t)$ in terms of L and F (the dependent variables), PAL and EI (possibly varying parameters), and other constant parameters.

The proportion $p(t)$ will vary with F and L; from Forbes' Law [1] we have that

$$\frac{dF}{dL} = \frac{F}{10.4}. \tag{1.4}$$

Hence,

$$\frac{F}{10.4} = \frac{dF}{dL} = \frac{dF/dt}{dL/dt} = \frac{\dfrac{(1 - p(t))EB(t)}{\rho_F}}{\dfrac{p(t)EB(t)}{\rho_L}} = \frac{\rho_L}{\rho_F}\frac{1 - p(t)}{p(t)}.$$

[1] *Lean body mass-body fat interrelationships in humans*, Forbes, G.B.; *Nutrition reviews*, pgs 225-231, 1987.

Solving for $p(t)$ gives Forbes' equation

$$p(t) = \frac{C}{C + F(t)} \quad \text{where} \quad C = 10.4\frac{\rho_L}{\rho_F}. \tag{1.5}$$

We will use two expressions for energy expenditure (EE). First, we have the formula

$$EE = PAL \times RMR, \tag{1.6}$$

where PAL is your physical activity level and RMR your resting metabolic rate. Your resting metabolic rate can be determined by using the Mifflin equation. This equation is an estimate based on a population study and is widely used in the literature. It takes into account your gender, age (A) in years, and height (H) in meters:

$$RMR = \begin{cases} 9.99W + 625H + 5A + 5 & \text{if male} \\ 9.99W + 625H + 5A - 161 & \text{if female.} \end{cases} \tag{1.7}$$

Your physical activity level can be determined by using the table below.

1.40–1.69	People who are sedentary and do not exercise regularly, spend most of their time sitting, standing, with little body displacement
1.70–1.99	People who are active, with frequent body displacement throughout the day or who exercise frequently
2.00–2.40	People who engage regularly in strenuous work or exercise for several hours each day

Table 1.1: This is a rough guide for physical activity level (PAL).

The second expression for energy expenditure comes from decomposing more precisely the different ways that energy is expended:

$$EE = \underbrace{\delta BW}_{\substack{\text{physical} \\ \text{activity}}} + \underbrace{\beta_{tef} EI}_{\substack{\text{thermic} \\ \text{effect of} \\ \text{eating}}} + \underbrace{\beta_{at} EI + \gamma_F F + \gamma_L L + \eta_F \frac{dF}{dt} + \eta_L \frac{dL}{dt} + K}_{\text{resting metabolic rate (RMR)}}, \tag{1.8}$$

where $\gamma_F = 3.2$ kcal/kg/d, $\gamma_L = 22$ kcal/kg/d, $\eta_F = 180$ kcal/kg, and $\eta_L = 230$ kcal/kg [2] [3]. Further, we let $\beta_{tef} = 0.10$ and $\beta_{at} = 0.14$ denote the coefficients for the thermic effect of feeding and adaptive thermogenesis, respectively. The parameter δ is the coefficient representing the amount of energy expended from physical activity per kilogram of body mass. Notice that γ_L is significantly larger than γ_F. This means that lean tissue metabolizes energy much faster than fatty tissue. As a result, there are instances where one may want to increase their lean body mass through resistance training so that they are better able to support a higher caloric intake without significant weight gain. Finally, we remark that the constant K can be tuned to an individual's body type directly through RMR and fat measurement, and is assumed to remain constant over time.

Thus, since the input EI is assumed to be known, we can use (1.8) and (1.5) to write (1.3) in terms of F and L, thus allowing us to close the system of ordinary differential equations (ODEs).

[2] *Modeling weight-loss maintenance to help prevent body weight regain*; Hall, K.D. and Jordan, P.N.; *The American journal of clinical nutrition*, pg 1495, 2008

[3] *Quantification of the effect of energy imbalance on bodyweight*; Hall, K.D. et al.; *The Lancet*, pgs 826-837, 2011

Specifically, we have

$$\begin{aligned} RMR(t) = \frac{EE}{PAL} &= K + \gamma_F F(t) + \gamma_L L(t) + \eta_F \frac{dF}{dt} + \eta_L \frac{dL}{dt} + \beta_{at} EI \\ \frac{1}{PAL}(EE - EI + EI) &= K + \gamma_F F(t) + \gamma_L L(t) \\ &\quad + \left(\frac{\eta_F}{\rho_F}(1 - p(t)) + \frac{\eta_L}{\rho_L} p(t)\right) EB(t) + \beta_{at} EI. \\ \left(\frac{1}{PAL} - \beta_{at}\right) EI &= K + \gamma_F F(t) + \gamma_L L(t) \\ &\quad + \left(\frac{\eta_F}{\rho_F}(1 - p(t)) + \frac{\eta_L}{\rho_L} p(t) + \frac{1}{PAL}\right) EB(t). \end{aligned}$$

Solving for $EB(t)$ in the last equation yields

$$EB(t) = \frac{\left(\frac{1}{PAL} - \beta_{at}\right) EI - K - \gamma_F F(t) - \gamma_L L(t)}{\frac{\eta_F}{\rho_F}(1 - p(t)) + \frac{\eta_L}{\rho_L} p(t) + \frac{1}{PAL}}. \tag{1.9}$$

In equilibrium ($EB = 0$), this gives us

$$K = \left(\frac{1}{PAL} - \beta_{at}\right) EI - \gamma_F F - \gamma_L L. \tag{1.10}$$

Thus, for a subject who has maintained the same weight for a while, one can determine K by using (1.10), if they know their average caloric intake and amount of fat (assume $L = BW - F$). The function `weight_odesystem` in the following code implements (1.3).

```
from math import log
# Fixed Constants:
rho_F = 9400.
rho_L = 1800.
gamma_F = 3.2
gamma_L = 22.
eta_F = 180.
eta_L = 230.
C = 10.4  # Forbes constant
beta_AT = 0.14  # Adaptive Thermogenesis
beta_TEF = 0.1   # Thermic Effect of Feeding
K = 0

def forbes(F):
    C1 = C * rho_L / rho_F
    return C1 / (C1 + F)

def energy_balance(F, L, EI, PAL):
    p = forbes(F)
    a1 = (1. / PAL - beta_AT) * EI - K - gamma_F * F - gamma_L * L
    a2 = (1 - p) * eta_F / rho_F + p * eta_L / rho_L + 1. / PAL
    return a1 / a2
```

```
def weight_odesystem(t, y, EI, PAL):
    F, L = y[0], y[1]
    p, EB = forbes(F), energy_balance(F, L, EI, PAL)
    return np.array([(1 - p) * EB / rho_F , p * EB / rho_L])

def fat_mass(BW, age, H, sex):
    BMI = BW / H**2.
    if sex == 'male':
        return BW * (-103.91 + 37.31 * log(BMI) + 0.14 * age) / 100
    else:
        return BW * (-102.01 + 39.96 * log(BMI) + 0.14 * age) / 100
```

Problem 3. Consider the initial value problem

$$\begin{aligned} \rho_F \frac{dF(t)}{dt} &= (1 - p(t))EB(t), \\ \rho_L \frac{dL(t)}{dt} &= p(t)EB(t), \\ F(0) &= F_0, \\ L(0) &= L_0. \end{aligned} \tag{1.11}$$

The ode is given above by the function `weight_odesystem`. To solve this IVP for a specific individual we need initial conditions F_0 and L_0. The function `fat_mass` given earlier calculates F_0 based on an individual's body weight (kg), age, height (meters), and gender. L_0 is then given by $L_0 = BW - F_0$.

Suppose a 38 year old female, standing 5'8" and weighing 160 lbs, reduces her intake from 2143 to 2025 calories/day, and increases her physical activity from little to no exercise (PAL=1.4) to exercising to 2-3 days per week (PAL=1.5). Using `scipy.integrate.ode`, find and graph the solution curve for this single-stage weightloss intervention over a period of 5 years.

Note the provided code requires quantities in metric units (kilograms, meters, days) while our graph is converted to units of pounds and days.

Problem 4. Modify the preceding problem to handle a two stage weightloss intervention: Suppose for the first 16 weeks intake is reduced from 2143 to 1600 calories/day and physical activity is increased from little to no exercise (PAL=1.4) to an hour of exercise 5 days per week (PAL=1.7). The following 16 weeks intake is increased from 1600 to 2025 calories/day, and exercise is limited to only 2-3 days per week (PAL=1.5).

Find and graph the solution curve over a period of 32 weeks.

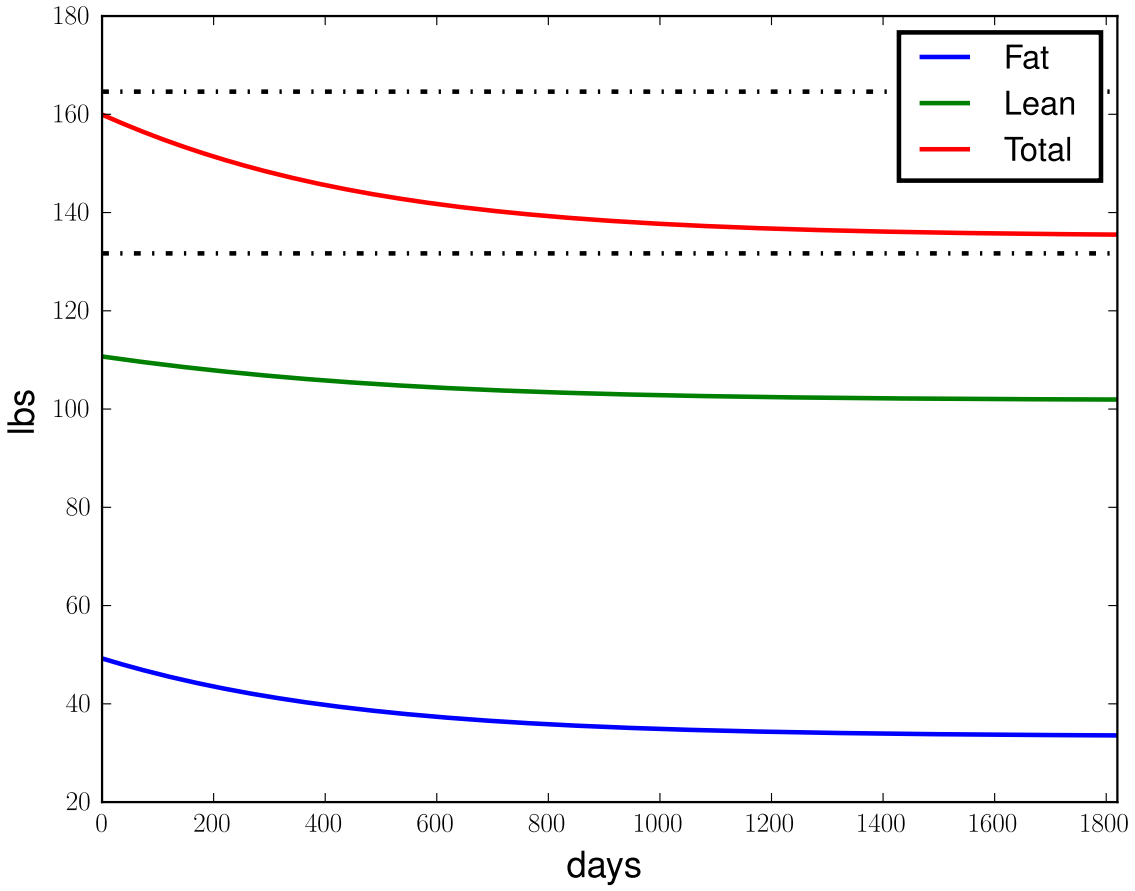

Figure 1.2: The solution of the weight change model for problem 1.

Two Predator-Prey Models

The Lotka-Volterra model

Reconsider (1.1). This type of problem has a special name. The Lotka-Volterra predator-prey model is a well-known system of ODEs given by

$$\frac{du}{dt} = au - buv,$$
$$\frac{dv}{dt} = -cv + duv.$$

u and v represent the prey and predator populations, respectively. Here a represents the rate of growth of the prey, and bu the amount of prey being eaten. Similarly, c represents the rate of natural predator death, and du the growth of the predator population due to the quantity of prey eaten.

Let us look at the dynamics of this system. First we note that there are exactly two equilibria (fixed points): either $(u, v) = (0, 0)$ corresponding to the extinction of both species, or $(u, v) = (\frac{c}{d}, \frac{a}{b})$. Furthermore, from the ODEs we can see that if $v = 0$ (there is an absence of any predators) then the population of prey will grow exponentially.

To get a better idea of the dynamics of this system we will graph its phase portrait. We begin

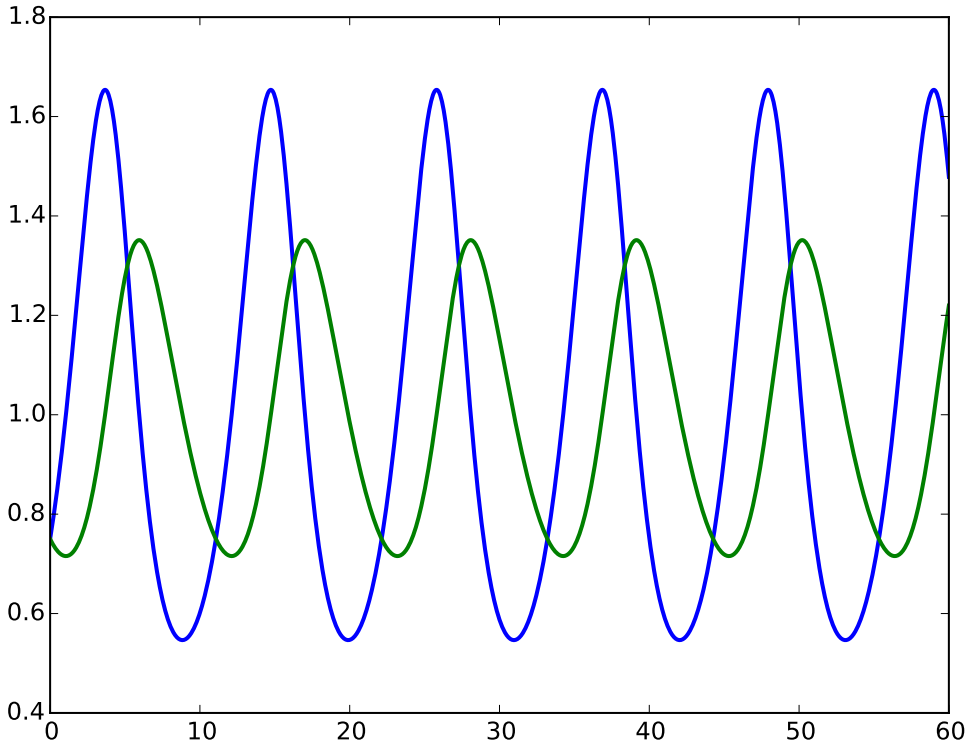

Figure 1.3: The solution of the nondimensionalized Lotka-Volterra predator-prey equations with parameter $\alpha = 1/3$. This solution has initial conditions $(U, V) = (3/4, 3/4)$.

by nondimensionalizing the system to reduce the number of parameters: Let $U = \frac{d}{c}u$, $V = \frac{b}{a}v$, $\bar{t} = at$, and $\alpha = \frac{d}{a}$. Substituting into the original ODEs we obtain the nondimensional system of equations

$$\begin{aligned} \frac{dU}{d\bar{t}} &= U(1 - V), \\ \frac{dV}{d\bar{t}} &= \alpha V(U - 1). \end{aligned} \tag{1.12}$$

In the following code we plot the phase portrait of (1.12) along with a example trajectory, see Figures 1.3 and 1.4. We will use `scipy.integrate.odeint` which acts similar to the `ode` function used earlier but integrates over all the time steps at once. To plot the direction field for the equations we use `numpy`'s `meshgrid` function and `matplotlib`'s `quiver` function.

```
from scipy.integrate import odeint
a, b = 0., 13.                    # (Nondimensional) Time interval for one '↩
    period'
alpha = 1. / 3                    # Nondimensional parameter
dim = 2                           # dimension of the system
y0 = np.array([1 / 2., 1 / 3.])   # initial conditions
```

```
# Note: swapping order of arguments to match the calling convention
# used in the built in IVP solver.
def Lotka_Volterra(y, x):
    return np.array([y[0] * (1. - y[1]), alpha * y[1] * (y[0] - 1.)])

subintervals = 200
# Using the built in ode solver
Y = odeint(Lotka_Volterra, y0, np.linspace(a, b, subintervals))

# Plot the direction field
Y1, Y2 = np.meshgrid(np.arange(0, 4.5, .2), np.arange(0, 4.5, .2), sparse=True,↩
     copy=False)
U, V = Lotka_Volterra((Y1, Y2), 0)
Q = plt.quiver(Y1[::3, ::3], Y2[::3, ::3],  U[::3, ::3],  V[::3, ::3], pivot='↩
    mid', color='b', units='dots',width=3.)
# Plot the 2 Equilibrium points
plt.plot(1, 1, 'ok', markersize=8)
plt.plot(0, 0, 'ok', markersize=8)
# Plot the solution in phase space
plt.plot(Y[:,0], Y[:,1], '-k', linewidth=2.0)
plt.plot(Y[::10,0], Y[::10,1], '*b')

plt.axis([-.5, 4.5, -.5, 4.5])
plt.title("Phase Portrait of the Lotka-Volterra Predator-Prey Model")
plt.xlabel('Prey',fontsize=15)
plt.ylabel('Predators',fontsize=15)
plt.show()
```

Problem 5. Compute the solutions (U, V) of (1.12) for initial conditions $(1/2, 3/4)$, $(1/16, 3/4)$, and $(1/40, 3/4)$. Add these solutions to the phase portrait of the Lotka-Volterra model. Can you see any limitations of this model?

The Logistic model

We have already noticed that in the absence of predators, the Lotka-Volterra equations predict that the prey population will grow exponentially. The logistic predator-prey equations change this dynamic by adding a term to give the prey population a carrying capacity K:

$$\frac{du}{dt} = au\left(1 - \frac{u}{K}\right) - buv,$$
$$\frac{dv}{dt} = -cv + duv.$$

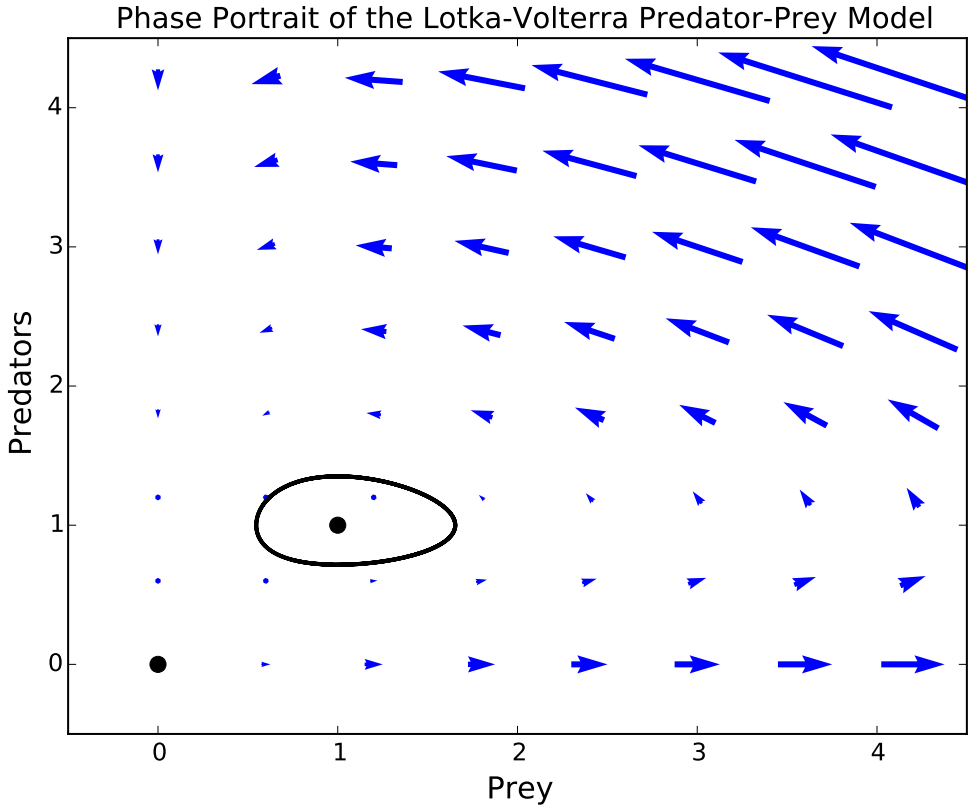

Figure 1.4: The phase portrait for the nondimensionalized Lotka-Volterra predator-prey equations with parameters $\alpha = 1/3$. The portrait includes the direction field, the two equilibrium points, and the graph of the solution with initial conditions $(U, V) = (3/4, 3/4)$.

Let $U = \frac{u}{K}$, $V = \frac{b}{a}v$, $\bar{t} = at$, $\alpha = \frac{dK}{a}$, and $\beta = \frac{c}{dK}$. Then the nondimensional logistic equations are

$$\begin{aligned} \frac{dU}{d\bar{t}} &= U(1 - U - V), \\ \frac{dV}{d\bar{t}} &= \alpha V(U - \beta). \end{aligned} \tag{1.13}$$

Problem 6. Compute the solutions (U, V) of (1.13) for initial conditions $(1/3, 1/3)$ and $(1/2, 1/5)$. Do this for parameter values $\alpha, \beta = 1, .3$ and also for values $\alpha, \beta = 1, 1.1$. Create a phase portrait for the logistic equations using both sets of parameter values. Remember to plot the direction field, all equilibrium points, and the orbits of the solutions.