Project 1 </ins>

Rabbit versus Sheep

We are going to investage the Lotka-Volterra model of competition. Here the two species are rabbits and sheep. Both species compete for the same limited food supply (grass). Before using this model, we must make some assumptions:

- In absense of the competing species, each species would grow according to its growth rate to its carring capacity
- Ignore all other confounding variables like other animals, weather, etc
- When a rabbit and sheep are trying to eat in the same area of grass, usually the sheep nudges the rabbit away and gets the food

Suppose that both species are competing for the same food supply (grass) and the amount available is limited. Furthermore, ignore all other complications, like predators, seasonal effects, and other sources of food. Then there are two main effects we should consider:

1. Eachspecieswouldgrowtoitscarryingcapacityintheabsenceofthe other. This can be modeled by assuming logistic growth for each spe- cies (recall Section 2.3). Rabbits have a legendary ability to repro- duce, so perhaps we should assign them a higher intrinsic growth rate

1. When rabbits and sheep encounter each other, trouble starts. Sometimes the rabbit gets to eat, but more usually the sheep nudges the rabbit aside and starts nibbling (on the grass, that is). We'll assume that these conflicts occur at a rate proportional to the size of each popula- tion. (If there were twice as many sheep, the odds of a rabbit encoun- tering a sheep would be twice as great.) Furthermore, we assume that the

conflicts reduce the growth rate for each species, but the effect is more severe for the rabbits.

$$\dot{x} = x(3 - x - 2y)$$
$$\dot{y} = y(2 - x - y)$$
$$\text{where x(t) = population of rabbits, y(t) population of sheep with } x, y > 0$$

To find the fixed points for the system, we solve x 0 and y 0 simultane- ously. Four fixed points are obtained: (0,0), (0,2), (3,0), and (1,1). To classify them, we compute the Jacobian:

Imports

In [313…
```python
import matplotlib.pyplot as plt
import numpy as np
from scipy.integrate import solve_ivp
from matplotlib.colors import Normalize
import matplotlib.cm as cm
```

First, build all functions needed to preform numerical integration

In [314…
```python
#fuction to give our ODEs
def model_eqn(x, y, alpha=3, beta=2):
    xdot, ydot = [x*(alpha - x - beta*y), y*(beta -x -y )]
    return xdot, ydot

#function to build phase plot
def model_phase(X, YX, alpha, beta):
    xdot, ydot = np.zeros(X.shape), np.zeros(YX.shape)
    Xlim, Ylim = X.shape
    for i in range(Xlim):
        for j in range(Ylim):
            xloc = X[i, j]
            yloc = YX[i, j]
            xdot[i,j], ydot[i,j] = model1_eqn(xloc, yloc,alpha ,beta )
    return xdot, ydot

#function to put in form for solve ivp
def model_solve_ivp(t,curr_vals, alpha, beta):
    x, y = curr_vals
    xdot, ydot = model1_eqn(x,y, alpha, beta)
    return xdot,ydot

# Set conditions and call solve_ivp to numerically integrate
alpha=3
beta=2
tmax = 30
dt = 0.01
tspan = (0,tmax)
t = np.arange(0,tmax,dt)
initial_condition = [4,4]
solved = solve_ivp(model_solve_ivp,tspan,initial_condition,t_eval = t,
                   args = (alpha, beta,),method="RK45", rtol=1e-8 , atol=1e-8,
```

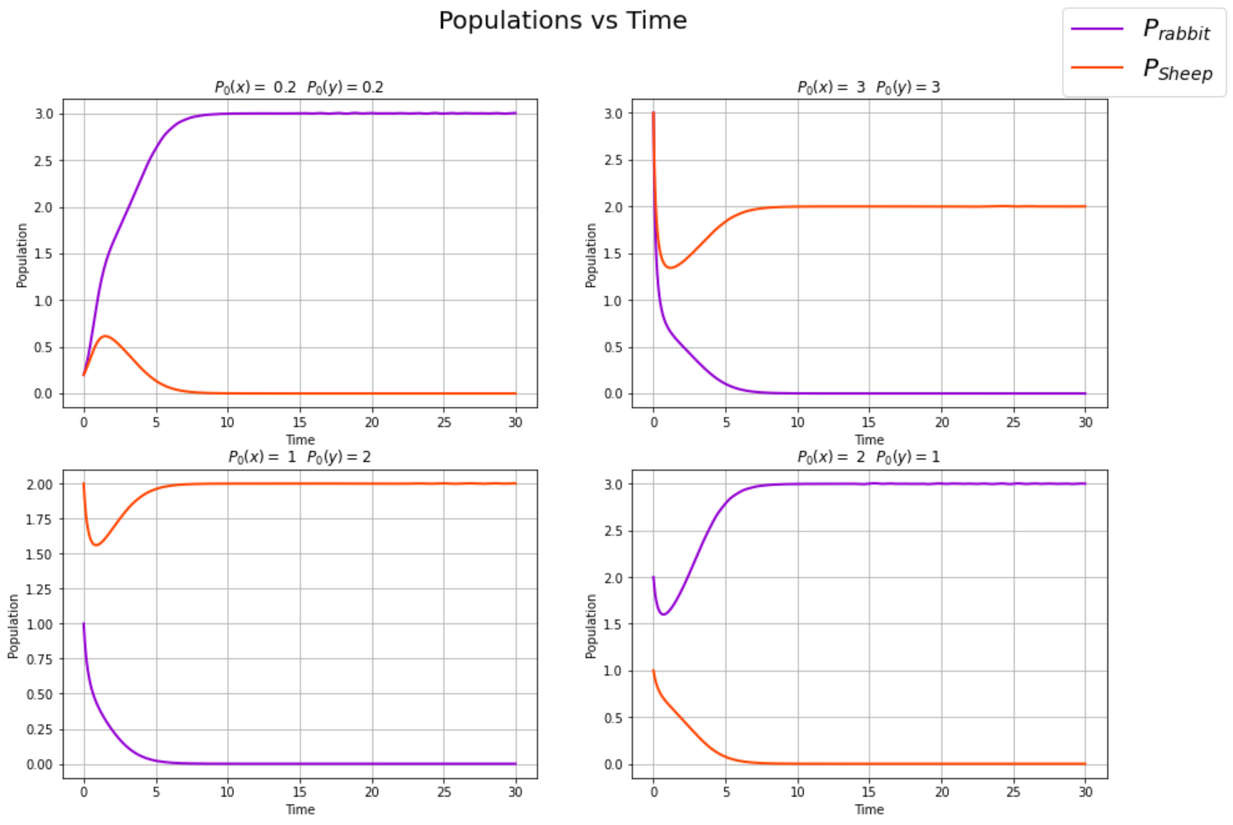
Now look at P_{rabbit} and P_{sheep} vs time

In [315…

```
#This function solves the ODE for certain initial conditions
#and plots rabbit pop and sheep pop vs time
def plot_pop_v_t(axis_number, x0, y0):
    initial_condition= [x0,y0]
    #numerically integrate
    solved = solve_ivp(model_solve_ivp,tspan,initial_condition, t_eval=t, args=

    #plot P rabbit
    axs[axis_number].plot(t, solved.y[0], color="darkviolet", label = r'$P_{rab
    #plot P sheep
    axs[axis_number].plot(t, solved.y[1], label = r'$P_{Sheep} $ ', color= 'ora

    axs[axis_number].set_xlabel("Time")
    axs[axis_number].set_ylabel("Population")
    axs[axis_number].set_title(r'$P_0(x)= $ ' + str(x0)+ r'  $P_0(y)= $' + str(
    axs[axis_number].grid()
```

In [316…

```
#call above function for various starting populations
fig, axs = plt.subplots(2,2, figsize=(15, 10) )
axs = axs.flat

plot_pop_v_t(0, x0= 0.2, y0= 0.2)
plot_pop_v_t(1, x0= 3, y0= 3)
plot_pop_v_t(2, x0= 1, y0= 2)
plot_pop_v_t(3, x0= 2, y0= 1)

handles, labels = axs[0].get_legend_handles_labels() #get legend from a plot
fig.legend(handles, labels, fontsize=20, loc= 'upper right') #add as figure leg

plt.suptitle("Populations vs Time ", size=20);
```



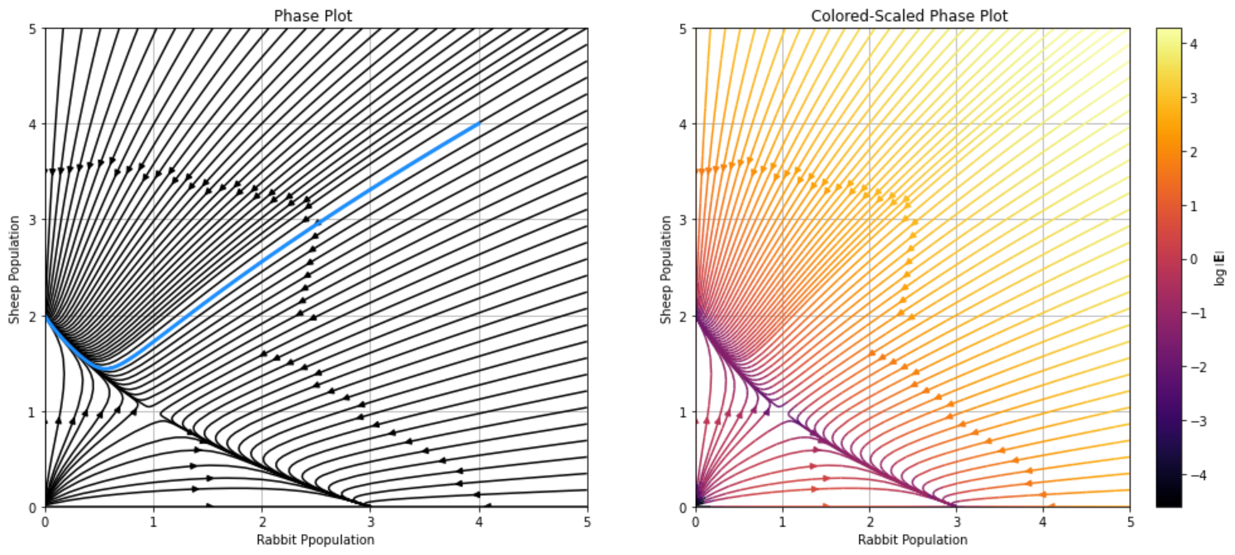
Findings

These plots show us how each population eventually reaches a steady equilibrium. One species will reach a maximum population and the other will go extint.
We can also see that when the populations are far away from 0, they quickly shoot down before approaching equilibrium. Whereas when they start near 0, they increase or decrease more slowly to equilibrium.

Now loook at some phase portraits

In [319…
```
# Plotting stuff
N = 40
x = np.linspace(0, 5., N)
y= np.linspace(0., 5., N)
X, Y = np.meshgrid(x, y)
xdot, ydot = model_phase(X, Y, alpha, beta)

fig, axs = plt.subplots(1,2, figsize=(17, 7) )
axs = axs.flat

axs[0].streamplot(X, Y, xdot, ydot, color='k' , broken_streamlines = False)
axs[0].plot(solved.y[0],solved.y[1],lw = 3,c = 'dodgerblue') # plot trajectory
axs[0].grid()
axs[0].set_xlim((0, 5))
axs[0].set_ylim((0, 5))
axs[0].set_xlabel('Rabbit Ppopulation')
axs[0].set_ylabel('Sheep Population')
```

```
axs[0].set_title('Phase Plot ')


magnitude = np.sqrt(xdot**2 + ydot**2)
colors = np.log(magnitude+0.01) #have colors to represent length of vectors
Q=axs[1].streamplot(X, Y, xdot, ydot, broken_streamlines = False,
                    color=colors, cmap = cm.inferno)
axs[1].grid()
axs[1].set_xlim((0, 5))
axs[1].set_ylim((0, 5))
axs[1].set_xlabel('Rabbit Population')
axs[1].set_ylabel('Sheep Population')
axs[1].set_title('Colored-Scaled Phase Plot ')
cbar = plt.colorbar(Q.lines)
cbar.set_label(r'$\log{\left|\mathbf{E}\right|}$')
```



Findings

At Critical Points:

The phase plots above shows there are two stable nodes of the solutions at $(0, 2)$ and $(3, 0)$.
They represent stable equilibriums of the populations; so at these points, the populations of rabbits and sheep will stay constant. We can see that when this happens, one population decreases to zero while the other increases to maximum population- it's *carrying capacity*.
This makes sense because if there are any animals of the competing species present, they will cause the growth rate of the orginal species to decrease and it will not be able to reach full carrying capacity.'

On the phase portraits, we also see an unstable node at $(0, 0)$. This is marked by the stream of vectors rushing out of the origin. Therefore $(0, 0)$ is an unstable equilibrium of the populations. This makes sense because when $P_{rabbits}$ and P_{sheep} are both 0, neither will change. But with anything just above $(0, 0)$, the populations of both will start to grow according to their growth rate before reaching an equilibrium.

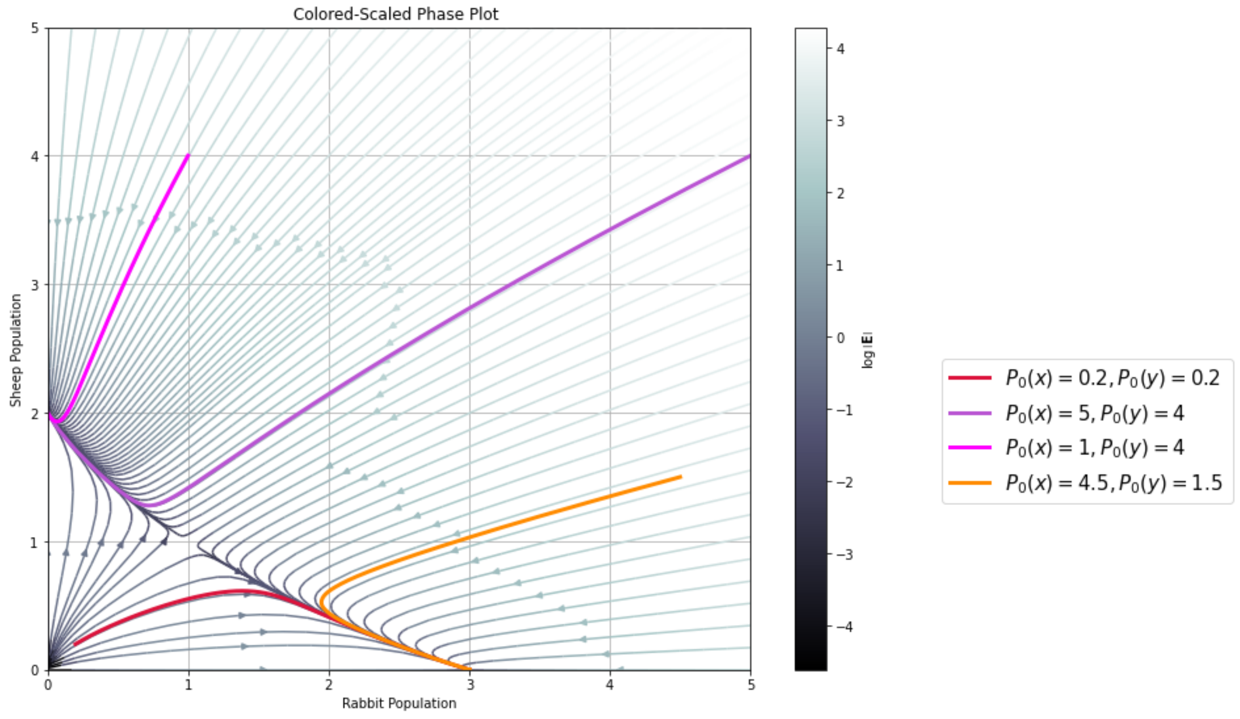
Lastly we can see a stadle point at $(1,1)$. This is marked by solutions rushing toward it in one direction and away from it in another. This means there exisits one solution that goes staight into $(1,1)$, and therefore it is technically possible for the two species to coexisit forever. But reality will never meet these conditions exactly, so the two species will never stay constant infinetly.

Vector Length:
By looking at the colored-scaled portrait, we can see how the magnitudes of the solution vectors change. The vectors get longer as we stray further from our critical points in the positive direction. This means that at high values, solutions rush toward critical points, ie the populations quickly decrease toward equilibrium.
We also can note that vector magnitudes are quite small around $(0,0)$ and also on the trajectory from $(0,2)$ to $(3,0)$. Therefore populations slowly move towards equilibiurm in these places.

Looking at more trajectories on the phase portrait

In [351…
```
initial_condition= [0.2,0.2]
solved1 = solve_ivp(model_solve_ivp,tspan,initial_condition, t_eval=t, args=(al

initial_condition= [5,4]
solved2 = solve_ivp(model_solve_ivp,tspan,initial_condition, t_eval=t, args=(al

initial_condition= [1,4]
solved3 = solve_ivp(model_solve_ivp,tspan,initial_condition, t_eval=t, args=(al

initial_condition= [4.5,1.5]
solved4 = solve_ivp(model_solve_ivp,tspan,initial_condition, t_eval=t, args=(al
```

In [373…
```
plt.figure(figsize=(12,9))

magnitude = np.sqrt(xdot**2 + ydot**2)
colors = np.log(magnitude+0.01) #have colors to represent length of vectors
Q2 =plt.streamplot(X, Y, xdot, ydot, broken_streamlines = False,
                   color=colors, cmap = 'bone')
plt.plot(solved1.y[0],solved1.y[1],lw = 3,c = 'crimson', label= r'$P_0(x)=0.2 ,
plt.plot(solved2.y[0],solved2.y[1],lw = 3,c = 'mediumorchid', label= r'$P_0(x)=
plt.plot(solved3.y[0],solved3.y[1],lw = 3,c = 'magenta', label= r'$P_0(x)=1 , P
plt.plot(solved4.y[0],solved4.y[1],lw = 3,c = 'darkorange', label= r'$P_0(x)=4.

plt.grid()
plt.xlim((0, 5))
plt.ylim((0, 5))
plt.xlabel('Rabbit Population')
plt.ylabel('Sheep Population')
plt.title('Colored-Scaled Phase Plot ')
plt.legend(bbox_to_anchor=[1.7, 0.5], fontsize=15)
cbar = plt.colorbar(Q2.lines)
cbar.set_label(r'$\log{\left|\mathbf{E}\right|}$')
```



This further shows how one species always goes extinct while the other reaches carrying capacity.

```
In [ ]:
```