Week2

September 12, 2024

1 Week 2

Code SIRPLOT (page 69) and make sure you can modify it to make "smooth" functions.

- How small do the timesteps need to be to make the functions S, I, and R smooth? How does this relate to the length of time we want to look at? Will the graphs we create for the following questions be identical if we use different timesteps?
- Write up the "How bad is your epidemic Activity" – you can share answers in Piazza.
- Choose one or more of the Case Studies to write up using code. (optional, make up your own case study).
- 1.2 #24, 25, 28 (Population, and Cooling)
- 2.2 #1 - 7 (logistic equation) pp85-86
- Read 2.3 and use write the code for LENGTH in python (share in Piazza if you have a good solution). page 91
- Do 2.3 #1- 10, calculating the length of a curve. pp95-96

1.1 1: Coding smooth SIR plots

1. How small do the timesteps need to be to make the functions S, I, and R smooth? How does this relate to the length of time we want to look at? Will the graphs we create for the following questions be identical if we use different timesteps?

- *The timesteps need to be approximately equal to or greater than the number of days that the function is evaluated for in order for the curves to appear smooth. If we use different numbers of timesteps (close to half the number of days or less) then the curves appear jagged and have clear joints. The graphs will show similar data and trends, but the curves will not look the same.*

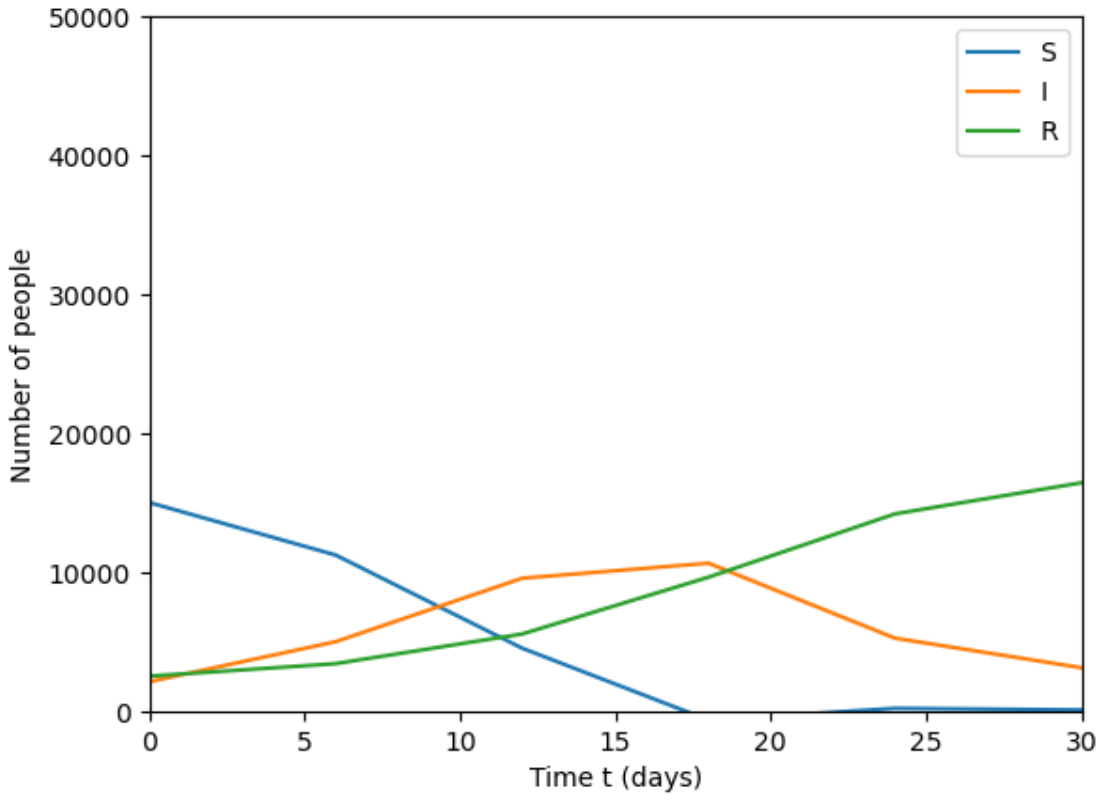
[4]:
```
import matplotlib.pyplot as plt
import numpy as np
```

[5]:
```
t_data = list()
S_data = list()
I_data = list()
R_data = list()


tinitial = 0
tfinal = 30   # How many DAYS!
```

```
numberofsteps = 5 # steps of iteration == line segments

t = tinitial
S = 15000# Susceptible
I = 2100 # Infected
R = 2500 # Recovered

a = .00002 # a = qp  therefore is the effective contact rate, q = faction of␣
 ↪contact that actually lead to new infections, p = contacts with infected␣
 ↪population
b = (1/14) # 14 days to recover

#numberofsteps = 1 # steps of iteration == line segments
deltat = (tfinal - tinitial)/numberofsteps    # Step
for k in range(0, numberofsteps+1, 1):

    # Step1: S, I, R Rate Equations OR Dynamical System
    Sprime = -a * S * I  # decrease population in S per day, Susceptible turned␣
 ↪Infectious
    Iprime = (a * S * I) - (I * b)  # changes in population I per day, balance␣
 ↪of input from Susceptible AND output to Recovered
    Rprime = (I * b)    # increase population in R per day, Infectious turned␣
 ↪Recovered

    # Step2: Central Principle Calculus in 9 words "net change equals rate of␣
 ↪change times elapsed time"
    deltaS = Sprime * deltat   # Xprime = rate of change, deltat = "Step" to o␣
 ↪describe the relation between rates and changes
    deltaI = Iprime * deltat
    deltaR = Rprime * deltat


    # each data point append to list
    t_data.append(t)
    S_data.append(S)
    I_data.append(I)
    R_data.append(R)


    # Step3: simple addition to update the data to update the data
    t = t + deltat
    S = S + deltaS
    I = I + deltaI
    R = R + deltaR
```

```
    #print(math.floor(deltaS), math.floor(deltaI), math.floor(deltaR))



fig = plt.figure()
ax = fig.add_subplot(111, axisbelow=True)
ax.plot(t_data, S_data, label='S')
ax.plot(t_data, I_data, label='I')
ax.plot(t_data, R_data, label='R')
ax.legend()
ax.set_ylim(0, 50000)
ax.set_xlim(0, tfinal)
ax.set_xlabel('Time t (days)')
ax.set_ylabel('Number of people')
```

[5]: Text(0, 0.5, 'Number of people')

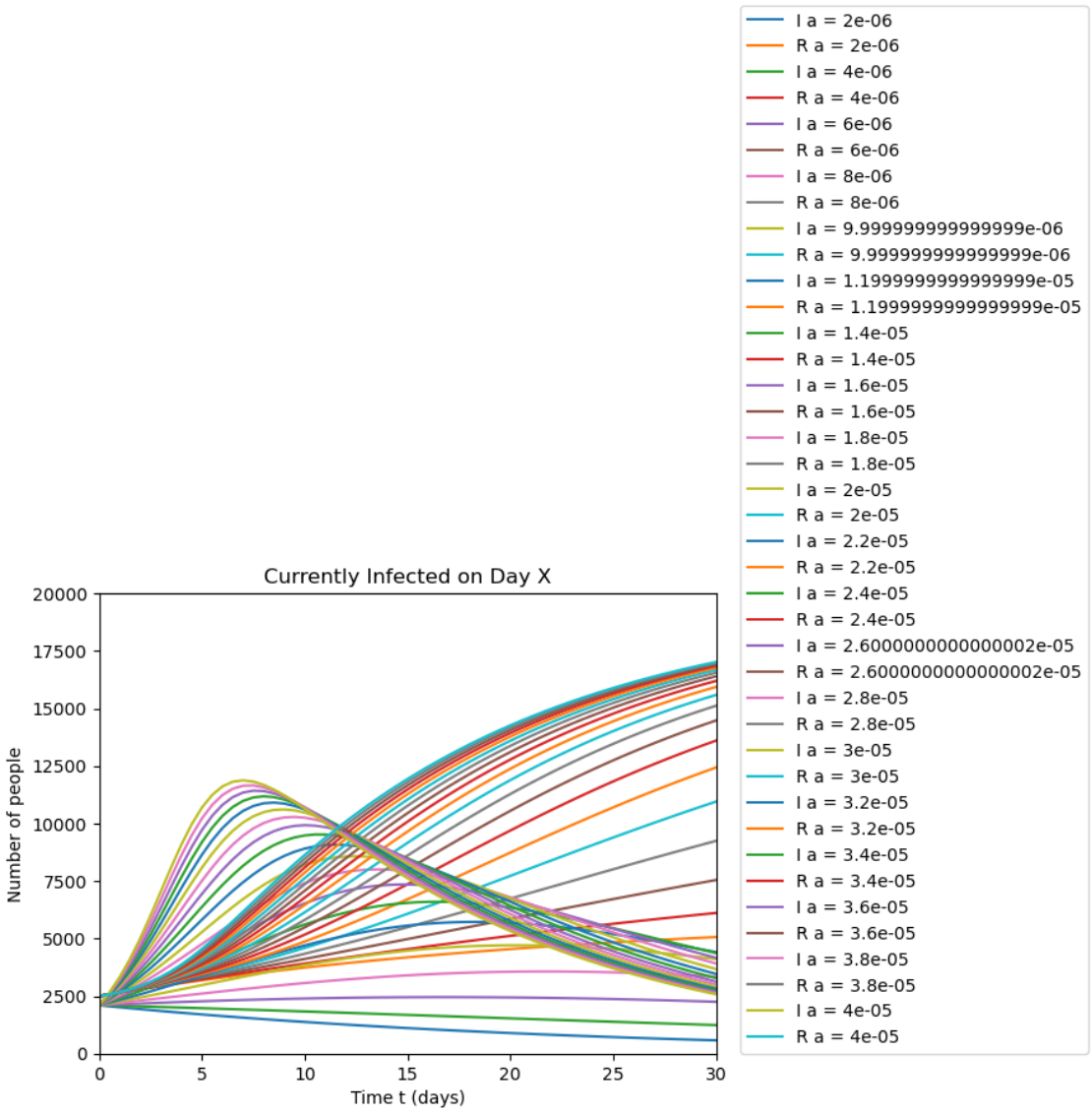


1.2 2: "How Bad Is Your Epidemic?"

[131]:
```
#APLOT
alst = np.linspace(0.000002, 0.00004, 20)
print(a)
b = (1/14) # 14 days to recover

#numberofsteps = 1 # steps of iteration == line segments
deltat = (tfinal - tinitial)/numberofsteps    # Step

fig = plt.figure()
ax = fig.add_subplot(111, axisbelow=True)

for a in alst:
    t_data = list()
    S_data = list()
    I_data = list()
    R_data = list()


    tinitial = 0
    tfinal = 30   # How many DAYS!

    numberofsteps = 60 # steps of iteration == line segments

    t = tinitial
    S = 15000# Susceptible
    I = 2100 # Infected
    R = 2500 # Recovered
    for k in range(0, numberofsteps+1, 1):

        # Step1: S, I, R Rate Equations OR Dynamical System
        Sprime = -a * S * I  # decrease population in S per day, Susceptible␣
↪turned Infectious
        Iprime = (a * S * I) - (I * b)  # changes in population I per day,␣
↪balance of input from Susceptible AND output to Recovered
        Rprime = (I * b)    # increase population in R per day, Infectious␣
↪turned Recovered

        # Step2: Central Principle Calculus in 9 words "net change equals rate␣
↪of change times elapsed time"
        deltaS = Sprime * deltat   # Xprime = rate of change, deltat = "Step"␣
↪to o describe the relation between rates and changes
        deltaI = Iprime * deltat
        deltaR = Rprime * deltat
```

```
        # each data point append to list
        t_data.append(t)
        S_data.append(S)
        I_data.append(I)
        R_data.append(R)


        # Step3: simple addition to update the data to update the data
        t = t + deltat
        S = S + deltaS
        I = I + deltaI
        R = R + deltaR

    ax.plot(t_data, I_data, label=f'I a = {a}')
    ax.plot(t_data, R_data, label=f'R a = {a}')

    #print(math.floor(deltaS), math.floor(deltaI), math.floor(deltaR))




ax.set_title("Currently Infected on Day X")
ax.legend(loc = (1.04, 0))
ax.set_ylim(0, 20000)
ax.set_xlim(0, tfinal)
ax.set_xlabel('Time t (days)')
ax.set_ylabel('Number of people')
plt.show()
```

```
4e-05
```



3. What effect does A have on the curve of currently Infected ? How does A affect when the illness peaks?

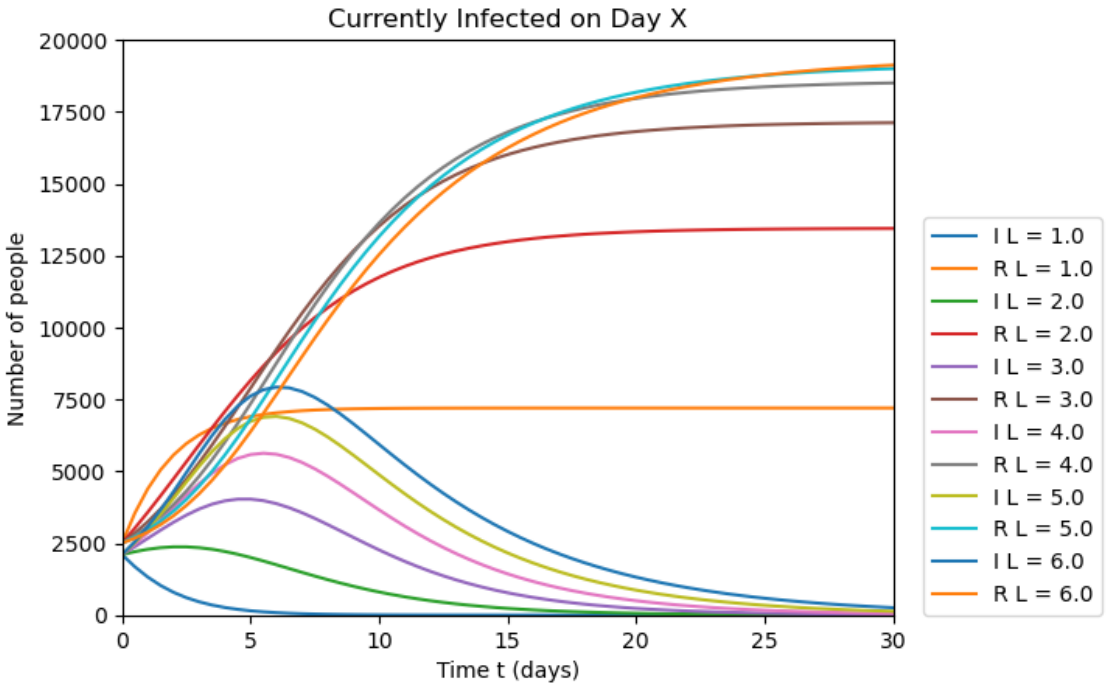
- *A larger value of A causes the initial rise in infected population and subsequent fall-off of the infected population to be steeper. A essentially represents the rate at which susceptible individuals become infected, so it makes sense that it would cause faster initial growth of the infected population.*

4. How does A effect the Total number of infected people by the end of the epidemic, to see this you will need to plot the total recovered (why ?)

- *A larger value of A will increase the total number of people infected by the end of the epidemic. As seen when the number of recovered individuals is also plotted, the highest value of A shows the steepest infection curve and the highest recovery curve.*

[132]:
```
#LPLOT
llst = np.linspace
alst = np.linspace(0.000002, 0.00004, 20)

Lst = np.linspace(1, 6, 6)


#numberofsteps = 1 # steps of iteration == line segments
deltat = (tfinal - tinitial)/numberofsteps    # Step

fig = plt.figure()
ax = fig.add_subplot(111, axisbelow=True)

for L in Lst:
    b = (1/L) # 14 days to recover
    t_data = list()
    S_data = list()
    I_data = list()
    R_data = list()


    tinitial = 0
    tfinal = 30   # How many DAYS!

    numberofsteps = 60 # steps of iteration == line segments

    t = tinitial
    S = 15000# Susceptible
    I = 2100 # Infected
    R = 2500 # Recovered
    for k in range(0, numberofsteps+1, 1):

        # Step1: S, I, R Rate Equations OR Dynamical System
        Sprime = -a * S * I  # decrease population in S per day, Susceptible␣
↪turned Infectious
        Iprime = (a * S * I) - (I * b)  # changes in population I per day,␣
↪balance of input from Susceptible AND output to Recovered
        Rprime = (I * b)    # increase population in R per day, Infectious␣
↪turned Recovered

        # Step2: Central Principle Calculus in 9 words "net change equals rate␣
↪of change times elapsed time"
        deltaS = Sprime * deltat   # Xprime = rate of change, deltat = "Step"␣
↪to o describe the relation between rates and changes
        deltaI = Iprime * deltat
        deltaR = Rprime * deltat
```

```
        # each data point append to list
        t_data.append(t)
        S_data.append(S)
        I_data.append(I)
        R_data.append(R)


        # Step3: simple addition to update the data to update the data
        t = t + deltat
        S = S + deltaS
        I = I + deltaI
        R = R + deltaR

    ax.plot(t_data, I_data, label=f'I L = {L}')
    ax.plot(t_data, R_data, label=f'R L = {L}')

    #print(math.floor(deltaS), math.floor(deltaI), math.floor(deltaR))




ax.set_title("Currently Infected on Day X")
ax.legend(loc = (1.04, 0))
ax.set_ylim(0, 20000)
ax.set_xlim(0, tfinal)
ax.set_xlabel('Time t (days)')
ax.set_ylabel('Number of people')
plt.show()
```



6. How does L affect the curve of currently Infected? How does changing B affect the curve?

- *Increasing L decreases B, and vice versa. A larger value of L means that there is a longer recovery time, meaning that the currently infected curve will be sharper as L increases, and as B decreases.*

7. Compare curves of currently Infected using L=1 to L= 6. Why is there such a big difference? Modify LPLOT to show a series of curves between L= 1, L=6. Exactly where is the big change? How many S would you expect at this point? What do we call this concept ?

- *There is such a big difference because at some point (where L = 2 here) the recovery rate is not greater than the infection rate, which is determined by A, so the number of infected people increases after 2 days. This change would happen exactly where the rate of infection is equal to the rate of recovery. at this point, the number of susceptible people would be 0 because those with the disease would have recovered.*

8. How does L effect the Total infected by the end of the epidemic? (You will need to plot the Recovered again, or can you think of another way to get the info ?)

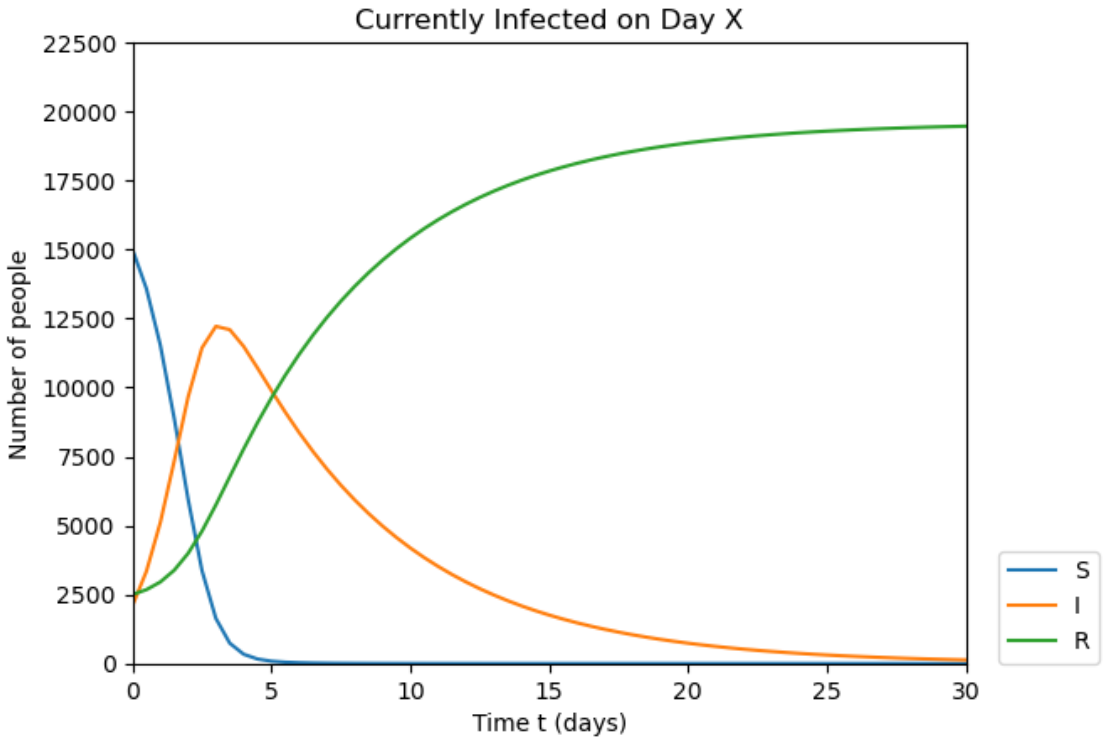
- *A higher value of L means that the recovery time is longer, and a higher number of people will be infected in total.*

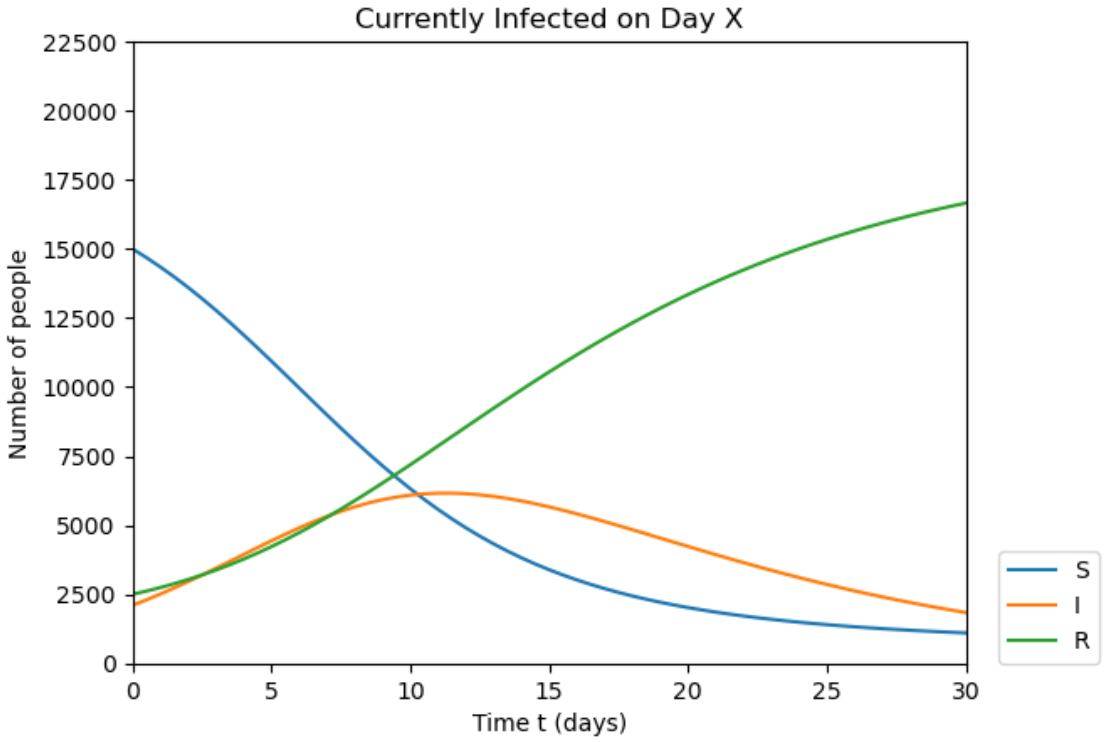
[133]:
```
def SIRPLOT(a, L, Si, Ii, Ri, tf):


    #numberofsteps = 1 # steps of iteration == line segments
      # Step
```

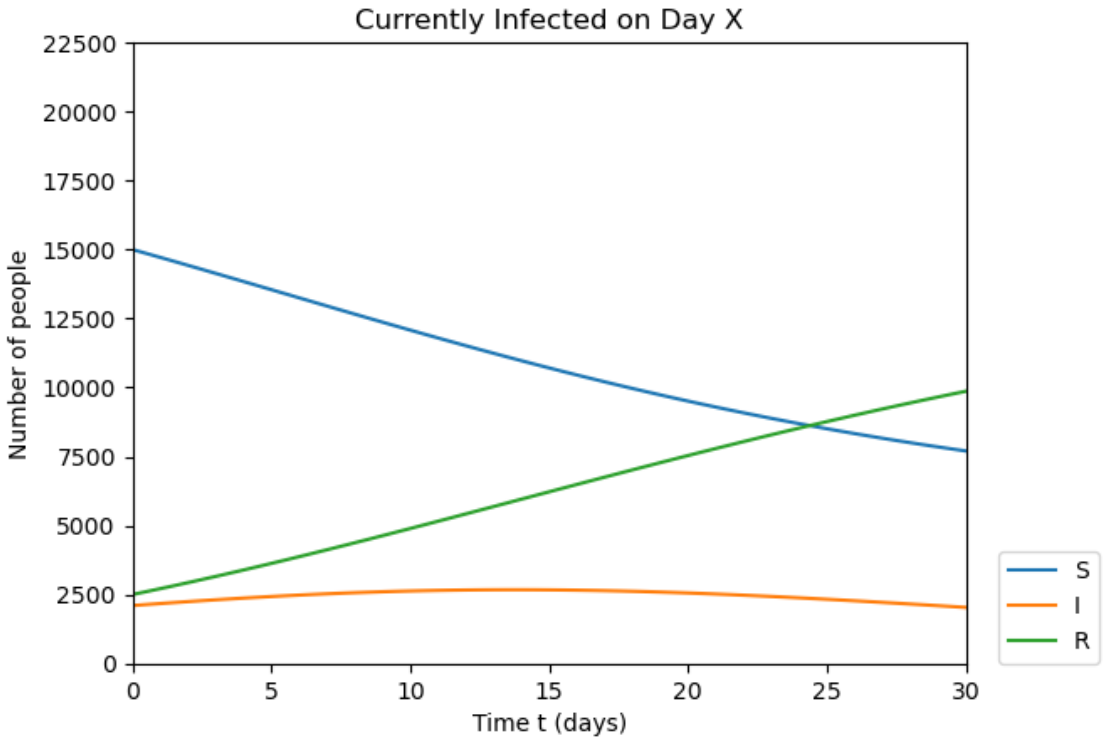
```
  b = (1/L) # 14 days to recover
  t_data = list()
  S_data = list()
  I_data = list()
  R_data = list()


  tinitial = 0
  tfinal = tf   # How many DAYS!

  numberofsteps = 60 # steps of iteration == line segments
  deltat = (tfinal - tinitial)/numberofsteps
  t = tinitial
  yinit = Si
  # Susceptible now modifiable
  S = Si
  I = Ii # Infected
  R = Ri # Recovered
  for k in range(0, numberofsteps+1, 1):

      # Step1: S, I, R Rate Equations OR Dynamical System
      Sprime = -a * S * I  # decrease population in S per day, Susceptible␣
↪turned Infectious
      Iprime = (a * S * I) - (I * b)  # changes in population I per day,␣
↪balance of input from Susceptible AND output to Recovered
      Rprime = (I * b)    # increase population in R per day, Infectious␣
↪turned Recovered

      # Step2: Central Principle Calculus in 9 words "net change equals rate␣
↪of change times elapsed time"
      deltaS = Sprime * deltat   # Xprime = rate of change, deltat = "Step"␣
↪to o describe the relation between rates and changes
      deltaI = Iprime * deltat
      deltaR = Rprime * deltat


      # each data point append to list
      t_data.append(t)
      S_data.append(S)
      I_data.append(I)
      R_data.append(R)


      # Step3: simple addition to update the data to update the data
      t = t + deltat
      S = S + deltaS
```

```
        I = I + deltaI
        R = R + deltaR



        #print(math.floor(deltaS), math.floor(deltaI), math.floor(deltaR))




    fig = plt.figure()
    ax = fig.add_subplot(111, axisbelow=True)
    ax.plot(t_data, S_data, label = "S")
    ax.plot(t_data, I_data, label = "I")
    ax.plot(t_data, R_data, label = "R")
    ax.set_title("Currently Infected on Day X")
    ax.legend(loc = (1.04, 0))
    ax.set_ylim(0, (yinit + yinit/2))
    ax.set_xlim(0, tfinal)
    ax.set_xlabel('Time t (days)')
    ax.set_ylabel('Number of people')
    plt.show()
```

[134]:
```
SIRPLOT(0.00009, 6, 15000, 2100, 2500, 30) #Peaks Soon with many infected
SIRPLOT(0.00002, 9, 15000, 2100, 2500, 30)#peaks later with many infected
SIRPLOT(0.000009, 10, 15000, 2100, 2500, 30) #Peaks later with few infected
```


Currently Infected on Day X
Number of people
Time t (days)
S
I
R


Currently Infected on Day X
Number of people
Time t (days)
S
I
R



11. How do A and L affect the threshold value ? How does B affect the threshold value?

- *The threshold value is the number of susceptible individuals that, if surpassed, will cause the number of infected indifiduals to continue to increase rather than stopping the infection. A faster recovery time (lower L, higher B) causes this number to be lower. On the other hand, having a higher infection rate modeled by A causes this number to become higher.*

12. How would you modify SIRPLOT to show the total new infected per day, rather than the total infected per day? How would you modify it to show the Total infected since day 0.

- To show the total new infected per day, we would need to plot the rate of change (Iprime, etc) rather than adding the value to the number I.

1.3 Case Study - Measles Panic

A small city of 100,000 is hit by a measles epidemic. It is believed to have been started by just one visitor to the town (on day 1). Unfortunately, no one in the town is immune to this unique strain of the measles which is known to last 5 days. However, by day 5 only 10 total are ill (I=10) and 4 total have recovered (R= 4), and on day 6 I= 19, R = 6. Since these number are so small the mayor decides that this epidemic will not be a big problem. Is she right? Make a model and find out. When will the epidemic peak? How many will get ill?

- *To model this, I used the SIRPLOT algorithm above, but modified the initial number of susceptible individuals to be 100000, and set L = 5. We can double check this number by*

looking at the change in recovered individuals between days 5 and 6:

$$R' = 2$$

$$2 = 10 * \frac{1}{5}$$

$$2 = 2$$

To find the infection rate, I looked at the difference between the number of infected individuals on days 5 vs 6. Because $I' = (a * S * I) - (b * I)$, we can plug in the known values to find a:

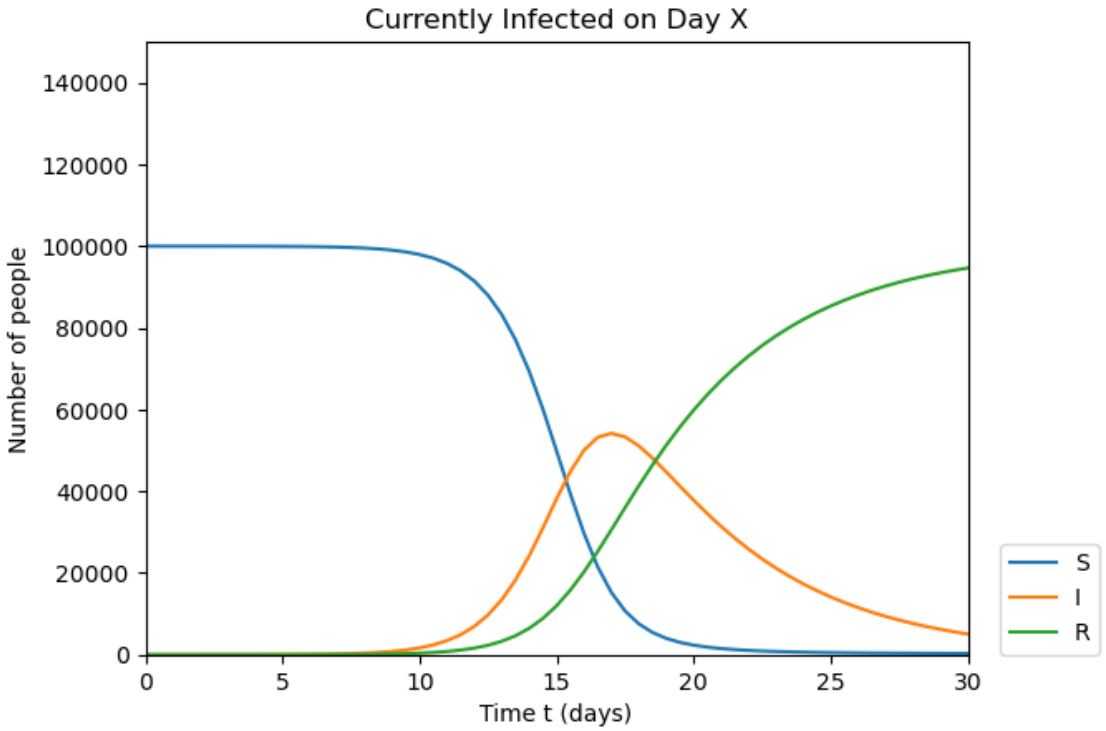
$$9 = (a * (100,000 - 14) * 10) - (\frac{1}{5} * 10)$$

$$9 = (a * 999860) - 2$$

$$\frac{11}{999860} = a$$

$$a \approx 0.000011$$

[135]:
```
SIRPLOT(0.000011, 5, 100000, 1, 0, 30)
```



- *The mayor is wrong, this infection peaks late, around day 17. Although everything looks fine at day 5, around day 10 the number of infected will increase rapidly and eventually everyone in the city will catch measles. At least in this model, everyone recovers!*

1.4 Book Problems

1.4.1 1.2 (24)

In the equation P = k P , above, explain why the units for k are persons per year person . The number k is called the per capita growth rate. ("Per capita" means per person"—"per head", literally.)

The units for P and P' are listed below:

$$P' = \frac{persons}{year}$$

$$P = persons$$

in order for k to multiply P to match the units of P', then the units would need to be $\frac{1}{year}$. In this case, however, it would not make sense for k to have units of 1/t, because P' must be proportional to P. In this case, we can divide the rate of change by the population

$$\frac{P'}{P} = \frac{\text{persons/year}}{persons}$$

but not cancel out the "persons" unit. k must be based on how many people are in the population currently.

1.4.2 1.2 (25)

In 1985 the per capita growth rate in Poland was 9 persons per year per thousand persons. (That is, k = 9/1000 = .009.) In Afghanistan it was 21.6 persons per year per thousand.

- a) Let P denote the population of Poland and A the population of Afghanistan. Write the equations that govern the growth rates of these populations.

$$P' = k_P * P = 0.009 * P$$

$$A' = k_A * A = 0.0216 * A$$

- b) In 1985 the population of Poland was estimated to be 37.5 million persons, that of Afghanistan 15 million. What are the net growth rates P and A (as distinct from the per capita growth rates)? Comment on the following assertion: When comparing two countries, the one with the larger per capita growth rate will have the larger net growth rate.

The net growth rate of Poland is as follows:

$$P' = 0.009 * 37500000 = 337500$$

The net growth rate of Afghanistan is as follows:

$$A' = 0.0216 * 15000000 = 324000$$

The assertion above is not correct in this case - although Afghanistan has a higher per-capita growth rate, it has a lower population in 1995 which causes it's net growth rate to be lower than that of Poland during the same year. The above assertion is only true if the populations are close to equal or if one population or per-capita growth rate is much higher than the other

- c) On the average, how long did it take the population to increase by one person in Poland in 1985? What was the corresponding time interval in Afghanistan?

The net growth rate has units of person per year. To find the corresponding 1 person growth interval we solve for t in the following equations:

$$P' = \frac{337500 \text{ people}}{1 \text{ year}} = \frac{1 \text{ person}}{t_p \text{ years}}$$

$$A' = \frac{324000 \text{ people}}{1 \text{ year}} = \frac{1 \text{ person}}{t_a \text{ years}}$$

$$1/337500 = 0.000003 \text{ years}$$

$$1/324000 = 0.00000308641 \text{ years}$$

$$0.00000296296 \text{ years} = 1.557 \text{ minutes}$$

$$0.00000308641 \text{ years} = 1.622 \text{ minutes}$$

1.4.3 1.2 (28)

Cooling. Suppose a cup of hot coffee is brought into a room at $70 \circ$F. It will cool off, and it will cool off faster when the temperature difference between the coffee and the room is greater. The simplest assumption we can make is that the rate of cooling is proportional to this temperature difference (this is called Newton's law of cooling). Let C denote the temperature of the coffee, in $\circ$F, and C the rate at which it is cooling, in $\circ$F per minute. The new element here is that C is proportional, not to C, but to the difference between C and the room temperature of $70 \circ$F.

- a) Write an equation that relates C and C. It will contain a proportionality constant k. How did you indicate that the coffee is cooling and not heating up?

$$C' = -kC$$

by explicitly showing that the constant k is negative, we denote that this is a decreasing function.

- b) When the coffee is at $180 \circ$F it is cooling at the rate of $9 \circ$F per minute. What is k?

$$-9 = -k * 180$$

$$-9/180 = -k = -0.05$$

$$k = 0.05$$

- c) At what rate is the coffee cooling when its temperature is $120 \circ$F?

$$C' = -0.05 * 120 = 6 \frac{\text{degrees F}}{\text{minute}}$$

- d) Estimate how long it takes the temperature to fall from 180 ◦ F to 120 ◦ F. Then make a better estimate, and explain why it is better. When the temperature is 180, the cooling rate is 9 degrees/minute, and when the temperature is 120 the cooling rate is 6 degrees per minute. We can find the average $(6 + 9)/2 = 7.5$ degrees per minute. If we divide the temperature difference $180 - 120 = 60$ by 7.5, we get an estimate of 8 minutes.

We can also obtain this estimate by finding the cooling rate in intervals of minutes. For example we have the series :

$$-9 = -0.05 * 180$$

$$-8.55 = -0.05 * 171$$

$$-8.122 = -0.05 * 162.45$$

$$-7.716375 = -0.05 * 154.3275$$

$$-7.331 = -0.05 * 146.611$$

$$-6.96 = -0.05 * 139.28$$

$$-6.61 = -0.05 * 132.32$$

$$-6.28 = -0.05 * 125.700$$

We have had to do this incremental calculation 8 times before reaching the rate of 6 at 120 seconds, which means that it will take just over 8 minutes to reach this temperature. This estimate is better because we are using a discrete version of the derivative and calculating a more accurate rate of change.

1.4.4 2.2 (1)

Modify SIRVALUE and SIRPLOT to analyze the population of Poland (see exercise 25 of chapter 1, page 47). We assume the population P (t) satisfies the conditions P = .009 P and P (0) = 37,500,000, where t is years since 1985. We want to know P 100 years into the future; you can assume that P does not exceed 100,000,000.

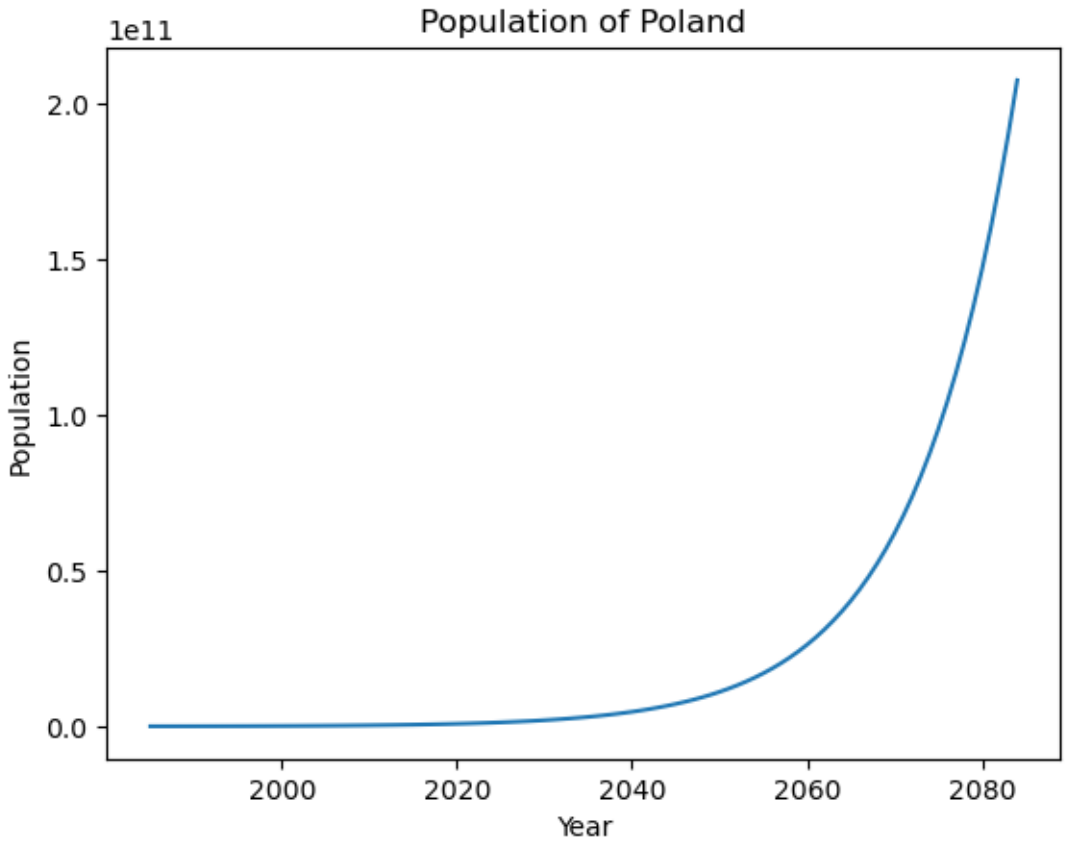
a) Estimate the population in 2085.

b) Sketch the graph that describes this population growth.

[136]:
```
def PolandPlot(k, P0, yf):
    tinit = 1985
    P = P0
    Plist = list()
    tlist = list()
    for step in range(0, (yf-1985)):
        Pprime = k*P
        P = P+Pprime
        Plist.append(P)
        tlist.append(step+1985)

    fig = plt.figure()
    ax = fig.add_subplot(111, axisbelow=True)
    ax.plot(tlist, Plist)
```

```
    ax.set_title("Population of Poland")
    ax.set_xlabel("Year")
    ax.set_ylabel("Population")

    plt.show()

    return P
```

[137]:
```
PolandPlot(0.09, 37500000, 2085)
```



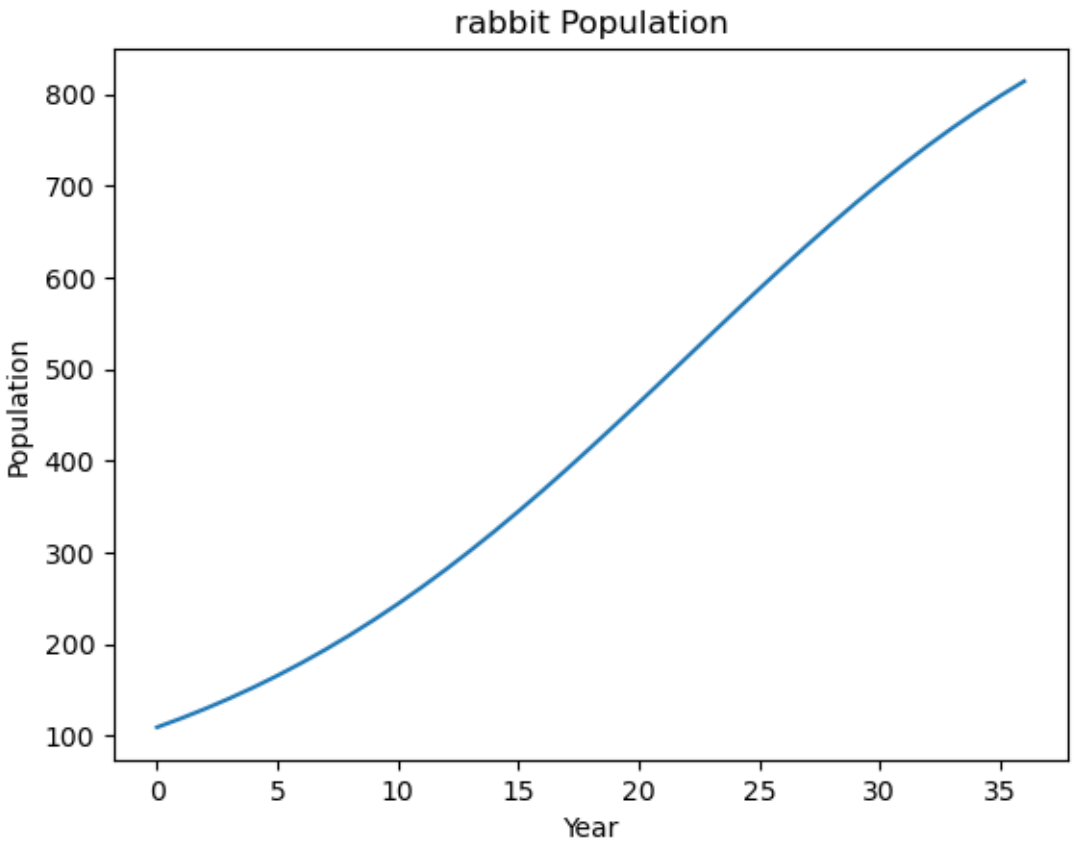
[137]: 207339029693.47043

1.4.5 2.2 (2)

By modifying SIRVALUE in the way we modified SIRPLOT to get SEQUENCE, obtain a sequence of estimates for y(37) that allows you to specify the exact value of y(37) to two decimal places accuracy.

[138]:
```
def RabbitPlot(R0, tf,cc = 1000):
    R = R0
    Rlist = list()
    tlist = list()
    for step in range(0, (tf)):
        Rprime = (0.1 * R) * (1 -(R/cc))
        R = R+Rprime
        Rlist.append(R)
        tlist.append(step)


    fig = plt.figure()
    ax = fig.add_subplot(111, axisbelow=True)
    ax.plot(tlist, Rlist)
    ax.set_title("rabbit Population")
    ax.set_xlabel("Year")
    ax.set_ylabel("Population")

    plt.show()

    return R
```

[139]:
```
y37 = RabbitPlot(100, 37)
print(y37)
```




```
814.00679485056
```

1.4.6 2.2 (3)

- a) Referring to the graph of y(t) obtained in the text on page 82, what can you say about the behavior of y as t gets large? as t gets large, the population y levels out and approaches a constant
- b) Suppose we had started with y(0) = 1000. How would the population have changed over time? Why? If y(0) = 1000, then the population would not have changed at all over the course of time because the second term in the rate equation, $1 - \frac{y}{1000}$ always evaluates to 0.
- c) Suppose we had started with y(0) = 1500. How would the population have changed over time? Why? If y(0) = 1500, then the population would initially decline because the second term in the equation evaluates to a negative number. Once it reached 1000, it would start to level out.
- d) Suppose we had started with y(0) = 0. How would the population have changed over time? Why? If we had started with y0=0, then the population would have immediately started growing exponentially until y gets to 1000 where it levels out.
- The number 1000 in the denominator of the rate equation is called the carrying capacity of

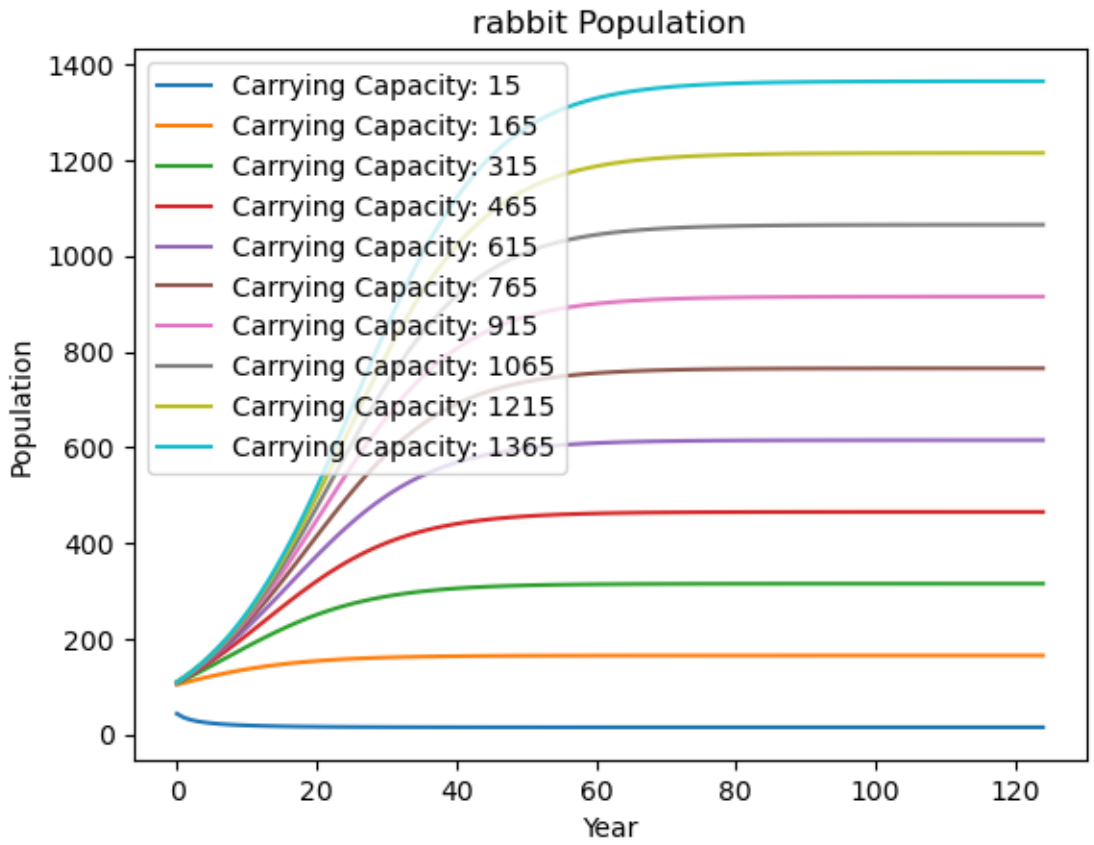
the system. Can you give a physical interpretation for this number? The carrying capacity is the upper limit on the population.

1.4.7 2.2(4)

- Obtain graphical solutions for the rate equation for different values of the carrying capacity. What seems to be happening as the carrying capacity is increased? (Don't restrict yourself to t = 37 here.) In this problem and the next you should sketch the different solutions on the same set of axes

as the carrying capacity increases, the population takes more time to reach the carrying capacity, but not by much. The maximum growth rate of the population is higher when the carrying capacity is larger

[140]:
```
fig = plt.figure()
ax = fig.add_subplot(111, axisbelow=True)
for capacity in range(15, 1500, 150):
    R = 100
    Rlist = list()
    tlist = list()

    for step in range(0, (125)):
        Rprime = (0.1 * R) * (1 -(R/capacity))
        R = R+Rprime
        Rlist.append(R)
        tlist.append(step)

    ax.plot(tlist, Rlist, label = f'Carrying Capacity: {capacity}')



ax.legend()
ax.set_title("rabbit Population")
ax.set_xlabel("Year")
ax.set_ylabel("Population")

plt.show()
```



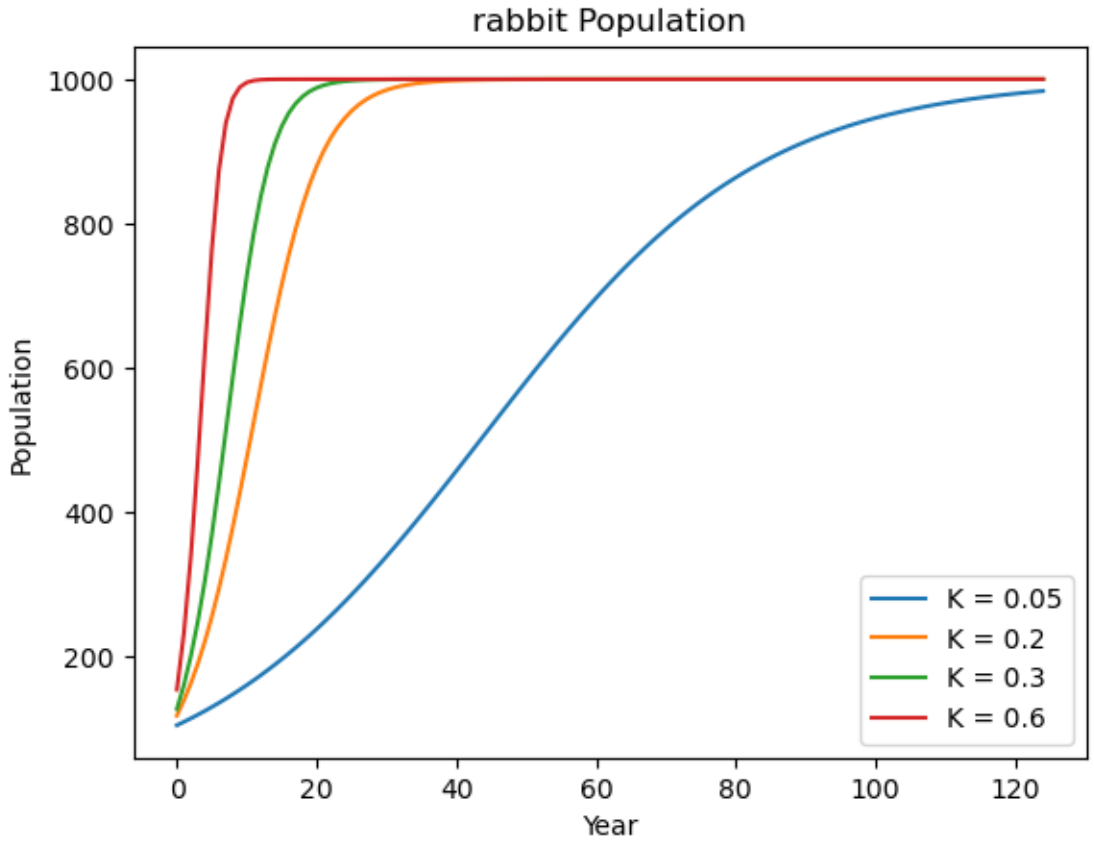
1.4.8 2.2(5)

Keep everything in the original problem unchanged except for the cont stant .1 out front. Obtain graphical solutions with the value of this constant = .05, .2, .3, and .6. How does the behavior of the solution change as this constant changes?

Changing this value causes the maximum growth rate to decrease, and the time before the population reaches carrying capacity to increase.

[141]:
```
fig = plt.figure()
ax = fig.add_subplot(111, axisbelow=True)
for k in [0.05, 0.2, 0.3, 0.6]:
    R = 100
    Rlist = list()
    tlist = list()

    for step in range(0, (125)):
        Rprime = (k * R) * (1 -(R/1000))
        R = R+Rprime
        Rlist.append(R)
        tlist.append(step)
```

```
    ax.plot(tlist, Rlist, label = f'K = {k}')



ax.legend()
ax.set_title("rabbit Population")
ax.set_xlabel("Year")
ax.set_ylabel("Population")

plt.show()
```

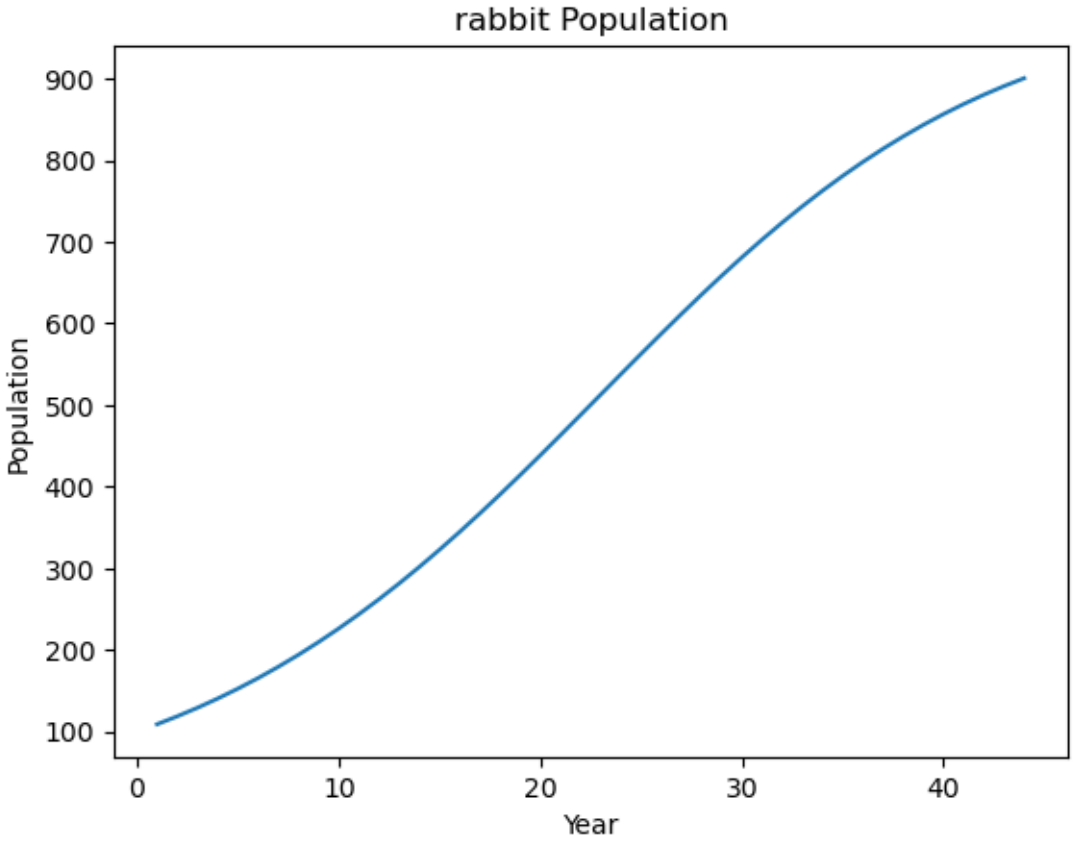


1.4.9 2.2 (6)

Returning to the original logistic equation, modify SIRVALUE or DO–WHILE to find the value for t such that y(t) = 900

[142]:
```
    R = 100
    Rlist = list()
    tlist = list()
```

```
    t = 0
    while R <= 900:
        Rprime = (0.1 * R) * (1 -(R/1000))
        R = R+Rprime
        t = t + 1
        Rlist.append(R)
        tlist.append(t)


    fig = plt.figure()
    ax = fig.add_subplot(111, axisbelow=True)
    ax.plot(tlist, Rlist)
    ax.set_title("rabbit Population")
    ax.set_xlabel("Year")
    ax.set_ylabel("Population")

    plt.show()
    print(t)
```



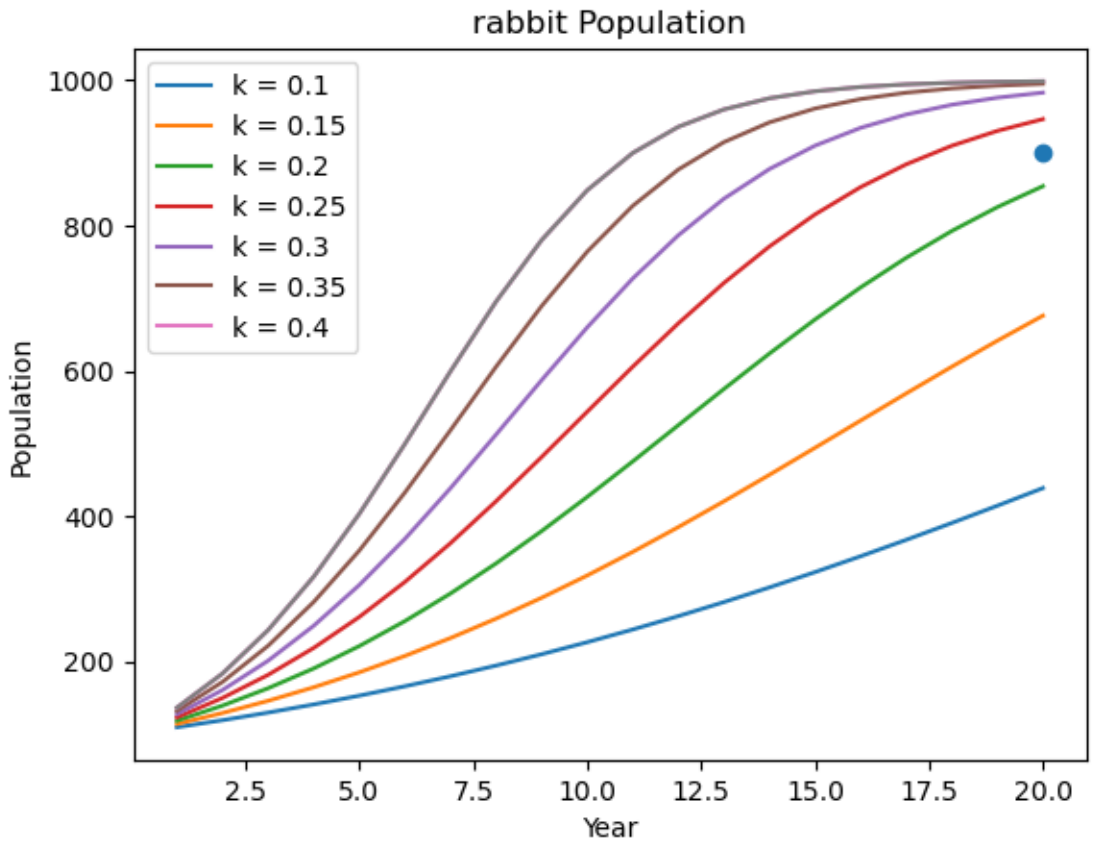
44

1.4.10 2.2 (7)

Suppose we wanted to fit a logistic rate equation to a population, starting with $y(0) = 100$. Suppose further that we were comfortable with the 1000 in the denominator of the equation, but weren't sure about the .1 out front. If we knew that $y(20) = 900$, what should the value for this constant be?

It should be approximately 0.225 rather than 0.1

[143]:
```
fig = plt.figure()
ax = fig.add_subplot(111, axisbelow=True)
for k in [0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4]:
    R = 100
    Rlist = list()
    tlist = list()
    t = 0
    for step in range(t, 20):
        Rprime = (k * R) * (1 -(R/1000))
        R = R+Rprime
        t = t + 1
        Rlist.append(R)
        tlist.append(t)
    ax.plot(tlist, Rlist, label = f'k = {k}')


ax.scatter(20, 900)
ax.legend()
ax.plot(tlist, Rlist)
ax.set_title("rabbit Population")
ax.set_xlabel("Year")
ax.set_ylabel("Population")

plt.show()
print(t)
```



20

[ ]:

1.5 LENGTH

Read 2.3 and use write the code for LENGTH in python (share in Piazza if you have a good solution).

[144]:
```
def Length(func, xi, xf, n):
    numberofsteps = n
    deltax = (xf - xi) / numberofsteps
    total = 0
    for k in range(1, numberofsteps):
        xl = xi + (k - 1) * deltax
        xr = xi + k * deltax
        yl = func(xl)
        yr = func(xr)
        segment = np.sqrt((xr - xl)**2 + (yr - yl)**2)
        total = total + segment
```

```
        #print(k, segment)

    print(numberofsteps, total)
```

[145]:
```
def f1(x):
    return x**2
Length(f1, 0, 8, 4)
```

```
4 36.737412257837796
```

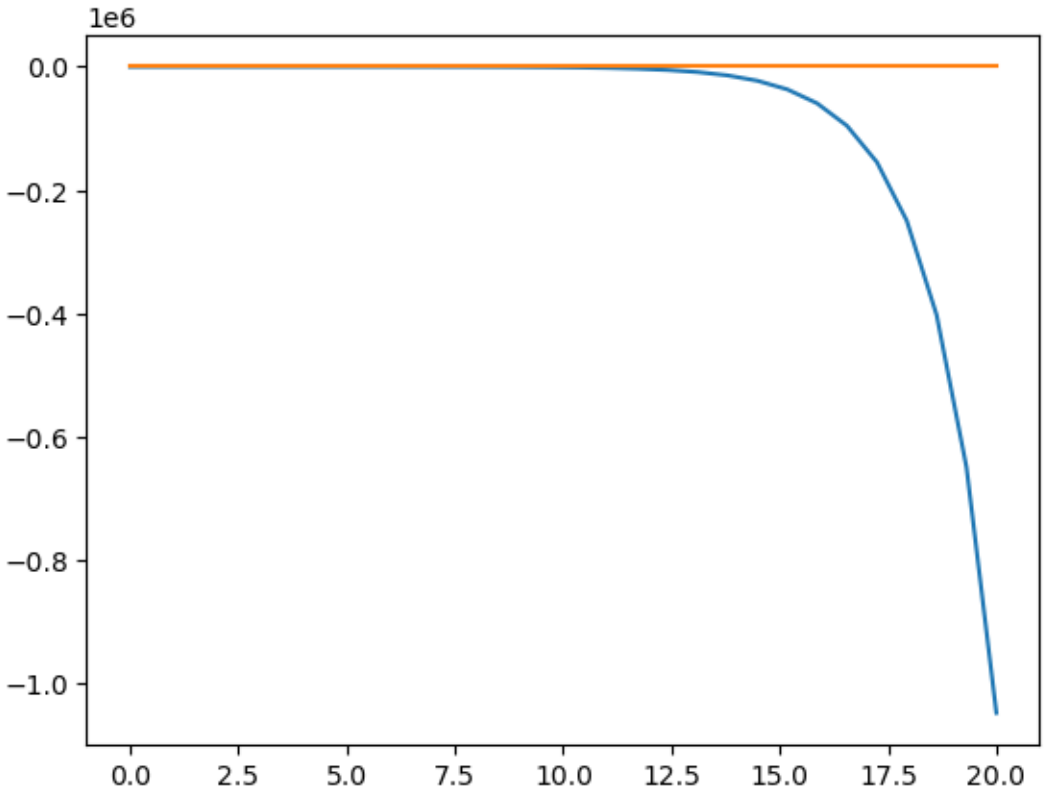
1.6 Book Problems 2.3

1.6.1 2.3(1)

By using a computer to graph y = x2 − 2x, find the solutions of th eequation x2 = 2x to four decimal place accuracy

[146]:
```
X= np.linspace(0, 20, 30)
Y = list()
sol = list()
for x in X:
    i = (x**2) - (2**x)
    Y.append(i)
    if (x**2) == (2**x):
        sol.append(x)

plt.plot(X, Y)
plt.plot(X, np.zeros(30))
print(sol)
```

```
[]
```



1.6.2 2.3 (2):

Run the program LENGTH to verify that it gives the lengths of the individual segments and their total length *See above*

1.6.3 2.3 (3):

What line in the program gives the instruction to work with the function fx) = x2? What line indicates the number of segments to be measured?

Line 1 first creates the function, and it is called in lines 12 and 13 to get values of y, Line 4 specifies the number of steps or segments to create

1.6.4 2.3(4):

Each segment has a left and a right endpoint. What lines in the program designate the x- and y-coordinates of the left endpoint; the right endpoint?

Lines 10-13 designate the lower corner(start) (10, 12) and upper right corner (end point)(11, 13).

1.6.5 2.3 (5):

Where in the program is the length of the k-th segment calculated? The segment is treated as the hypotenuse of a triangle whose length is measured by the Pythagorean theorem. How is the base

of that triangle denoted in the program? How is the altitude of that triangle denoted?

The length of the k-th segment is set as variable segment in line 14. The base of the triangle is the difference between the x values, and the altitude of the triangle is the change in y values

1.6.6 2.3 (6):

Modify the program so that it uses 20 segments to estimate the length of the parabola. What is the estimated value?

See below: Answer is about 1.3692

[147]:
```
Length(f1, 0, 1, 20)
```

```
20 1.3691834770250182
```

2.3 (7) Modify the program so that it estimates the length of the parabola using 200, 2, 000, 20, 000, 200, 000, and 2, 000, 000 segments. Compare your results with those tabulated below. To speed the process up, you will cer- tainly want to delete the PRINT k, segment statement that appears inside the loop. Do you see why?

Yes, the results are much more straightforward without showing the length at every segment. See below for measurements, they match well with book results.

[ ]:
```
for num in [200, 2000, 20000, 200000, 2000000]:
    Length(f1, 0, 1, num)
```

```
200 1.4677830093448685
2000 1.4778250285231551
20000 1.4788310561954459
200000 1.478931677225224
2000000 1.478941739510839
```

1.6.7 2.3 (8)

What is the length of the parabola y = x2 over the interval 0 x 1, correct to 8 decimal places? What is the length, correct to 12 decimal places?

8 Decimal places: 1.47894174, 12 Decimal Places: 1.478941738511

1.7 2.3 (9)

Starting at the origin, and moving along the parabola y = x2, where are you when you've gone a total distance of 10? *See below, x = 1.35*

[ ]:
```
total = 0
while (total < 10):
    X2 = np.linspace(0, 3, 20)
    for x in X2:
        numberofsteps = 200
        deltax = (x - 0) / numberofsteps
        for k in range(1, numberofsteps):
```

```
            xl = 0 + (k - 1) * deltax
            xr = 0 + k * deltax
            yl = f1(xl)
            yr = f1(xr)
            segment = np.sqrt((xr - xl)**2 + (yr - yl)**2)
            total = total + segment
            #print(k, segment)

        print(x, numberofsteps, total)
```

```
0.0 200 0.0
0.15789473684210525 200 0.159653373868766
0.3157894736842105 200 0.49346531563664126
0.47368421052631576 200 1.027372294358257
0.631578947368421 200 1.7949994799926738
0.7894736842105263 200 2.834831155559266
0.9473684210526315 200 4.188424693734495
1.1052631578947367 200 5.8993366710572435
1.263157894736842 200 8.012471780434367
1.4210526315789473 200 10.573676436078305
1.5789473684210527 200 13.629476229729196
1.7368421052631577 200 17.226900500118184
1.894736842105263 200 21.41336142280194
2.052631578947368 200 26.236568333633553
2.2105263157894735 200 31.74446550802286
2.3684210526315788 200 37.98518598131081
2.526315789473684 200 45.00701660824195
2.6842105263157894 200 52.85837116986329
2.8421052631578947 200 61.58776935591508
3.0 200 71.24382011279862
```

1.7.1 2.3(10)

Modify the program to find the length of the curve y = x3 over the interval 0 x 1. Find a value that is correct to 8 decimal places.

1.53212266

[ ]:
```
def func2(x):
    return x**3

Length(func2, 0, 1, 200)
```

```
200 1.5321226566902577
```

[ ]:
```

```