Algorithms and Simulation - Template Notebook

March 7, 2022

1 CS51 Assignment 2: Algorithms and Simulation

This assignment has two distinct parts in addition to a reflection: Part 1 requires you to apply your knowledge of algorithmic thinking and optimization and Part 2 allows you to demonstrate your modeling and coding skills by writing a numerical simulation. Material relevant for Part 1 will be covered in class during weeks 3-5, while material relevant for Part 2 will be covered in class during weeks 7 and 8.

You'll notice several "*Optional Challenge*" problems throughout the assignment to challenge yourself. These will only be scored (4 or 5) if they are completed correctly with thorough explanation. If you attempt an optional challenge but do not succeed, you will not be penalized with a low score. Remember that you must include an explanation and interpretation for optional problems to be scored.

This is an individual assignment. We will be checking for similarities among submissions and will take plagiarism seriously.

You must complete all tasks within this pre-formatted Jupyter notebook. Please follow ALL formatting guidelines and the HC Guidelines in the assignment instructions on Forum (near the top and bottom of the instructions respectively).

1.1 PART 1: OPTIMIZATION

For this section of the assignment, you will select one of the scenarios below and apply #optimization. You must complete all sections. [#optimization, potentially relevant: #modeling, #algorithms, #variables, #utility, #constraints]

1. *Scenario 1*: To prevent the spread of an infectious disease, a vaccine needs to be distributed as quickly and efficiently as possible to the 15 cities that have had major outbreaks. How can you optimize the route between the cities? For this scenario, you should select cities that are relevant to the disease that you will choose for PART 2. It may be helpful to include a map of these cities (either an existing map or create your own).

2. *Scenario 2*: Suppose that a new virus is starting to spread, and many clinics do not have sophisticated diagnostic tools and must be able to determine whether or not a patient has this dangerous virus based solely on easily measured symptoms. You have been collecting information on symptoms (temperature, WBC count, headache severity, and cough severity) and you need to determine which patients have this new disease and which have only a milder illness. Plots that provide an overview of the data are available here. The data can be accessed at this link (1 = Infected, 0 = Not infected).

1.1 Optimization Problem: Describe the optimization problem for your scenario: what is the objective function? What are the decision variables? Are there any constraints? Clearly articulate each component so that it's clear how the objective value would be measured and how the decision variables would impact it (~200 words).

Answer 1.1 For my scenario (1), since the objective is to minimize the length of the route, generating less time delivering the vaccines, the objective function is the function that inputs routes in different orders and outputs the length—the objective value. As decision variables, there is the order of the routes. As real-world constraints, different levels of accessibility to a place, how long vaccines can wait until they go bad, and if some cities need the vaccine first then others may be encountered, but they will not be considered in this scenario. As for algorithmic constraints, since I chose a Genetic Algorithm, having to be able to do crossovers may not make sense in all scenarios. The objective value is measured as the routes are calculated, the objective value being the final route sum. The decision variables can affect the process in the sense that the order of the cities impacts directly the length of the route, an output where paths cross is not ideal, for example.

Word count: 165 words.

1.2 Optimization Technique: What process can be used to find the optimal solution in your scenario? Identify and describe an existing algorithm that could be used to complete this process, including the inputs, outputs, required steps, and the termination condition. Explain the advantages and limitations of this algorithm. In your explanation, you should address whether your algorithm would lead to the global optimum and you may wish to compare your algorithm with other possible optimization techniques. (~200 words)

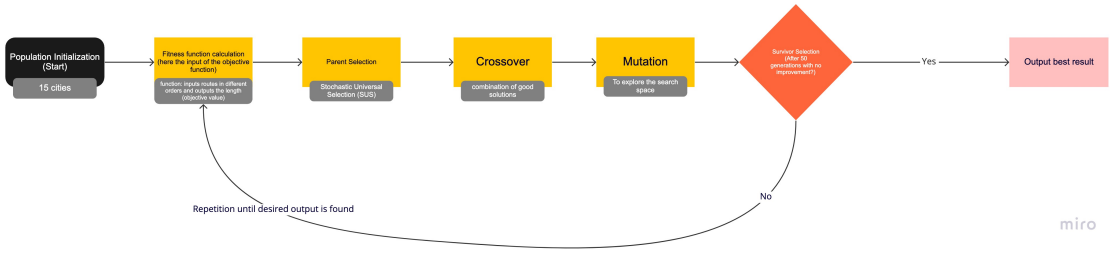
- *Optional*: Draw a flowchart to illustrate the process.
- *Optional challenge*: Create a program in Python to implement this optimization process. Please provide a thorough explanation of how the code works, both holistically and using in-line comments. You must also provide at least one "test case" that demonstrates that your code is properly implemented. This test should be something like, "In this case, it's clear that the only possible solution is X; let's check that with our code. Yes, the code outputs X as well, so we can in principle induce that the implementation is correct."

Answer 1.2 A genetic algorithm can be used to find the optimal solution. GAs have a population of possible solutions that pass through recombinations and "mutations" producing new solutions (children). This process is repeated multiple times each one creating a "generation." The location of the cities will be our input. The output would be the optimized routes, the fitness function, here the same as the objective, serving to assess the fitness of the output. As for steps, the input undergoes a "parent selection", a "crossover", a "mutation", and a "survivor selection" (final output). This algorithm has as an advantage the fact it serves to find optimal or near-optimal solutions to complex problems such as defining the fastest route between 15 cities—it finds a solution in a reasonable time frame. However, if the levels of mutation are set too high the output can become less fit. This happens because the bigger the mutation probability, the more randomness is added to the process. For this algorithm to generate a good output we need to choose strong parents for the next generations. If the algorithm cannot find the exact global optimum, it can get close enough (near-optimal).

Word count: 192 words.

[6]:
```
# flowchart
# to import images

from IPython.display import Image
Image("CS51_algorithms_simulation.jpg") # file name
```

[6]:



FOOTNOTE The flowchart was based in one of the class' readings, that is why the fitness function appears at the beginning. It is understood the function assesses the fitness of an OUTPUT—meaning it should probably be at the end of the flowchart.

1.2 PART 2: SIMULATION

The SIR model of the spread of disease is commonly used to help understand how a disease might move through a population. You were introduced to this with the NetLogo agent-based model in NS50 and will review it again in Weeks 7 and 8 of CS51. Check out the class readings to learn about this model.

For this assignment, you will select one disease of your choice to model. Please choose a disease from THIS LIST to investigate. If you would like to select your own disease to model, you may email your professor with the disease and parameter descriptions for approval. You must select an infectious disease (one that is transmitted from person to person through a viral, bacterial, or parasitic agent), not a genetic or environmental disease.

1.2.1 Part 2.1 Numerical Modeling and Simulation

For this part of the assignment, you'll consider the SIR model described by the set of differential equations below, and the numerical simulation in Python via Euler's method.

$$\frac{dS}{dt} = -\frac{b}{N}S(t)I(t)$$
$$\frac{dI}{dt} = \frac{b}{N}S(t)I(t) - kI(t)$$
$$\frac{dR}{dt} = kI(t)$$

2.1.1 Variables and Parameters (~250 words) [#variables] This section serves to set up an initial analysis of the SIR model. 1. State the disease you selected to model.

2. Identify the relevant **variables** of the model, fully classify what type of variables they are, and explain what they mean in the context of your model. Next, identify appropriate numerical values and units for the **initial values** of the variables. You're encouraged to use empirical data if possible to justify these values. You may also complete a well-reasoned #estimation for any values that are difficult to justify with empirical data. Include APA citations for any external sources used. Note that you can work with population values S, I, R, or proportions, S/N, I/N, R/N, as long as you are consistent.

3. Explain what the relevant **parameters** (b and k) are and what they mean. For your model, identify and justify appropriate numerical values and units for b and k. As above, you may include a well-reasoned estimate using empirical data to support your justification. Further, explain what it would mean for the parameters (b and k) to be smaller or larger. Consider what real-world factors, in the context of the disease you selected, would reduce or increase these parameters.

4. *Optional*: Modify the basic SIR model to add a layer of real-world complexity. A few ideas are listed below. Explain the key features of the extended model, including the modified differential equations and a full description and classification of any new variables and parameters following the steps above.

- Vaccination
- Antibiotic use and/or development of antibiotic resistance
- Variability in population susceptibility (e.g. children and the elderly have different rates of infection compared to young adults).
- Birth and death rates in the population

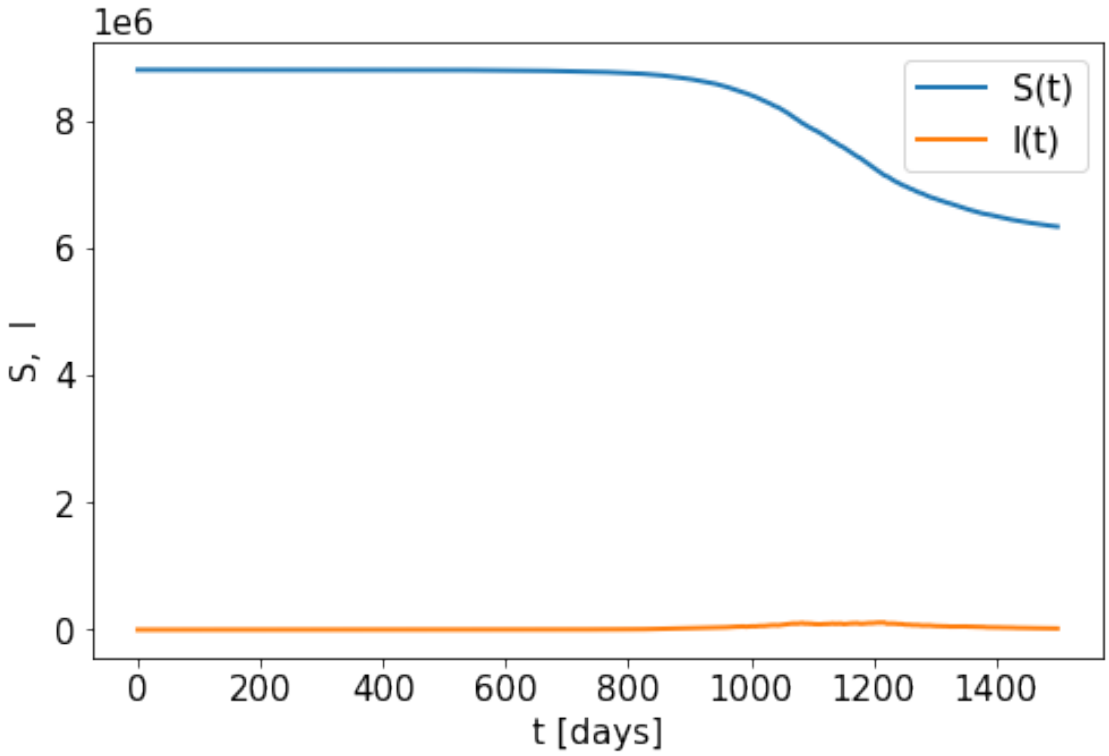
Answer 2.1.1 I selected COVID-19 as the disease to be modeled. For variables, we have the population of New York City—8,804,190 people (United States Census Bureau, 2020), the susceptible portion of the population (S), the infected (I), and the recovered portion (R)—all three quantitative and discrete variables since they are defined by numbers that cannot be represented by floats, only integers. In an initial scenario, S would count as the whole city's population, minus the patient zero (I = 1), and R = 0, since no one would have recovered from the disease. As parameters, there is b: the infection rate and k: the recovery rate. Both are measured with respect to time. b is drawn from the contact between infected and susceptible people, having a value of 0.085 infected/day. k is drawn based on how much time it takes for an infected individual to be cured, having a value of 0.062 recovered/day. If b was bigger, more people would be infected by one person, while if the contrary happened one infected individual would have the power to infect fewer people. If k increases, a person would recover faster, the contrary if it decreases. Factors such as vaccines can decrease b since more people would have a certain immunity to COVID. Vaccines may also result in milder symptoms, probably making an individual recover faster (k). Immune system strength can affect both parameters similarly, and self-isolation interacts directly with b since an isolated infected person cannot infect others.

Word count: 250 words.

Optional As a modification I added vaccination as a real-world complexity. In the model, only a part of the population is vaccinated (simulated by the randomization in the code bellow) and having the vaccine does not make an individual sully immune, but it brings down the infection rate. The equations do not change, but because b gets smaller the breakout takes more days to happen and the total infected people is smaller.

[52]:
```
# optional - vaccine
# code based on CS51's Session 13 notebook

# importing all needed libraries
import numpy as np
import matplotlib.pyplot as plt
from random import randint

# setting the plot "style"
plt.rcParams.update({'font.size': 15})
plt.rcParams["figure.figsize"] = [8,5]


# function that takes b, k and the initial conditions of the scenario
# and calculates outputs using Euler's method
def SIR_Euler_vaccine(b,k,initial_conds):
    t0 = 0 # tstart
    t_end = 1500 # tend

    h = 1 # stepsize
    steps = int((t_end - t0)/h + 1) # number of steps

    # variables:
    t = np.linspace(t0, t_end, steps) # to store t values
    S = np.zeros(steps) # to store S values
    I = np.zeros(steps) # to store I values

    # initial conditions:
    S[0] = initial_conds[0]
    I[0] = initial_conds[1]
    N = S[0]+I[0] # total population



    for n in range(steps-1): # range(start, stop, step)
        # randomization of vaccines
        vacc = randint(0, 4)
        if vacc == 1:
            b = 0.03
        else:
            b = infection_rate
        S[n+1] = S[n] - h*(b*S[n]*I[n]/N) # S past value minus people who␣
↪became infected
        I[n+1] = I[n] + h*(b*S[n]*I[n]/N - k*I[n]) # I past value plus people␣
↪who became infected minus who recovered

    # plotting
```

```
    plt.plot(t,S,linewidth=2,label='S(t)')
    plt.plot(t,I,linewidth=2,label='I(t)')
    plt.xlabel('t [days]')
    plt.ylabel('S,  I')
    plt.legend(loc='best')
    plt.show()

# parameters:
infection_rate = 0.085 # b
recovery_rate = 0.062 # k

# initial conditions:
S0 = 8804189  # NYC population - 1
I0 = 1 # patient zero
initial_vals = [S0,I0]

# call the function to run the simulation
SIR_Euler_vaccine(b=infection_rate, k=recovery_rate, initial_conds=initial_vals)
```



2.1.2 Euler's Method Description (~150 words) [#algorithms] Explain what it means to solve the SIR differential equations and how Euler's method works as an algorithm to achieve a numerical solution via simulation. In your explanation, identify the inputs, outputs, and steps that

the algorithm takes, and consider the role of the step size (`h`) in the algorithm.

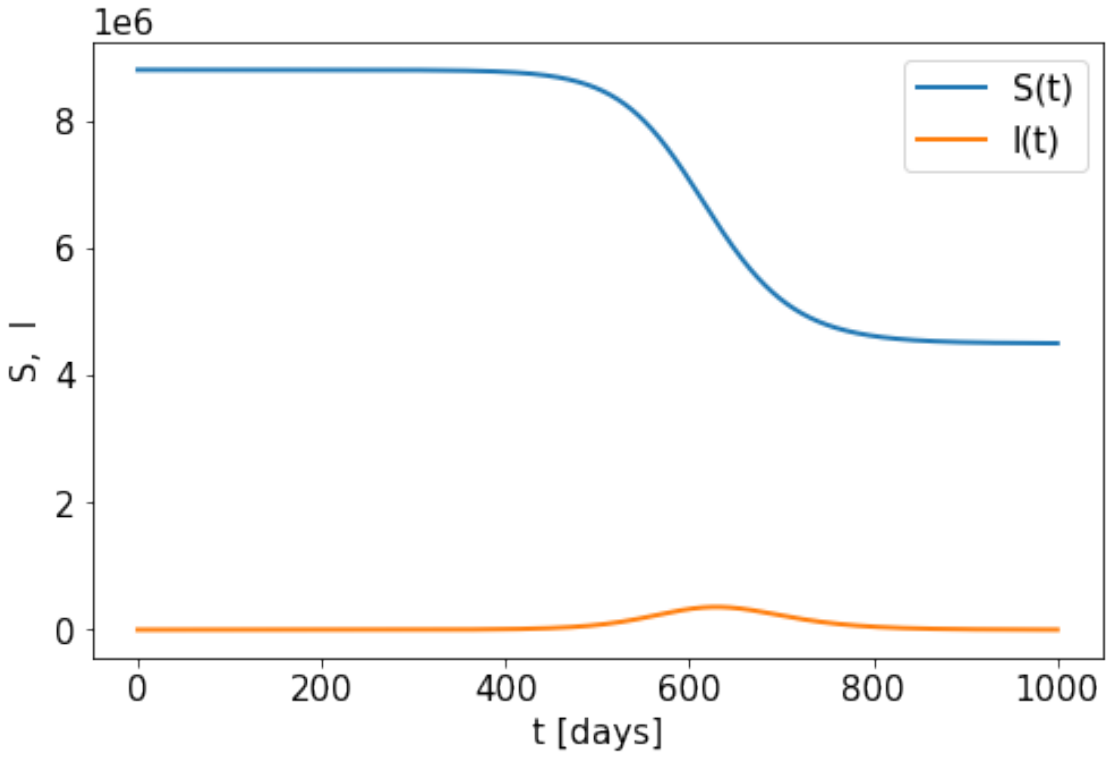
Answer 2.1.2 SIR's differential equations solutions are S(t), I(t), and R(t) functions. Time is their independent variable. S is the number of susceptible people to the disease, I is infected, and R is the recovered—all dependent variables. The differential equations indicate the continuous change of those variables in time, meaning its results cannot be determined easily analytically. Euler's method numerically generates points that approximate the functions' real trend. An approximate behavior of the disease in real life is generated—there is an initial condition and the model is run to get outputs based on it. The algorithm's inputs are the initial condition of the population, the step size, and the functions used to get the outputs. The outputs are the simulated behavior of the population and the number of S, I, and R. In Euler's method from the initial point, a tangent is traced to find another point. An interval from the initial to final condition is defined, being chopped into small subdivisions of length h—the step size. The smaller the h the closer the points are to each other and to the real curve of the functions.

Word count: 188 words [the word count is over 150 words, but my other responses are mostly under the limit].

2.1.3 Euler's Method Implementation [#algorithms, #dataviz] Define a function that implements a numerical simulation to derive the implications of your model using Euler's method in Python. Your simulation must generate at least one relevant visualization of the disease dynamics (see required analysis below), including a descriptive figure legend and caption. You may need to adjust the run-time and step size in your simulation to ensure the visualization is maximally informative. Include thorough comments in your code to convey your understanding of the implementation of Euler's method.

[53]:
```
# code based on CS51's Session 13 notebook

# importing all needed libraries
import numpy as np
import matplotlib.pyplot as plt

# setting the plot "style"
plt.rcParams.update({'font.size': 15})
plt.rcParams["figure.figsize"] = [8,5]


# function that takes b, k and the initial conditions of the scenario
# and calculates outputs using Euler's method
def SIR_Euler(b,k,initial_conds):
    t0 = 0 # tstart
    t_end = 1000 # tend (days)

    h = 1 # stepsize
    steps = int((t_end - t0)/h + 1) # number of steps

    # variables:
    t = np.linspace(t0, t_end, steps) # to store t values
```

```
    S = np.zeros(steps) # to store S values
    I = np.zeros(steps) # to store I values

    # initial conditions:
    S[0] = initial_conds[0]
    I[0] = initial_conds[1]
    N = S[0]+I[0] # total population

    for n in range(steps-1): # range(start, stop, step)
        S[n+1] = S[n] - h*(b*S[n]*I[n]/N) # S past value minus people who␣
↪became infected
        I[n+1] = I[n] + h*(b*S[n]*I[n]/N - k*I[n]) # I past value plus people␣
↪who became infected minus who recovered

    # plotting
    plt.plot(t,S,linewidth=2,label='S(t)')
    plt.plot(t,I,linewidth=2,label='I(t)')
    plt.xlabel('t [days]')
    plt.ylabel('S,  I')
    plt.legend(loc='best')
    plt.show()

# parameters:
infection_rate = 0.085 # b
recovery_rate = 0.062 # k

# initial conditions:
S0 = 8804189  # NYC population - 1
I0 = 1 # patient zero
initial_vals = [S0,I0]

# call the function to run the simulation
SIR_Euler(b=infection_rate, k=recovery_rate, initial_conds=initial_vals)
```

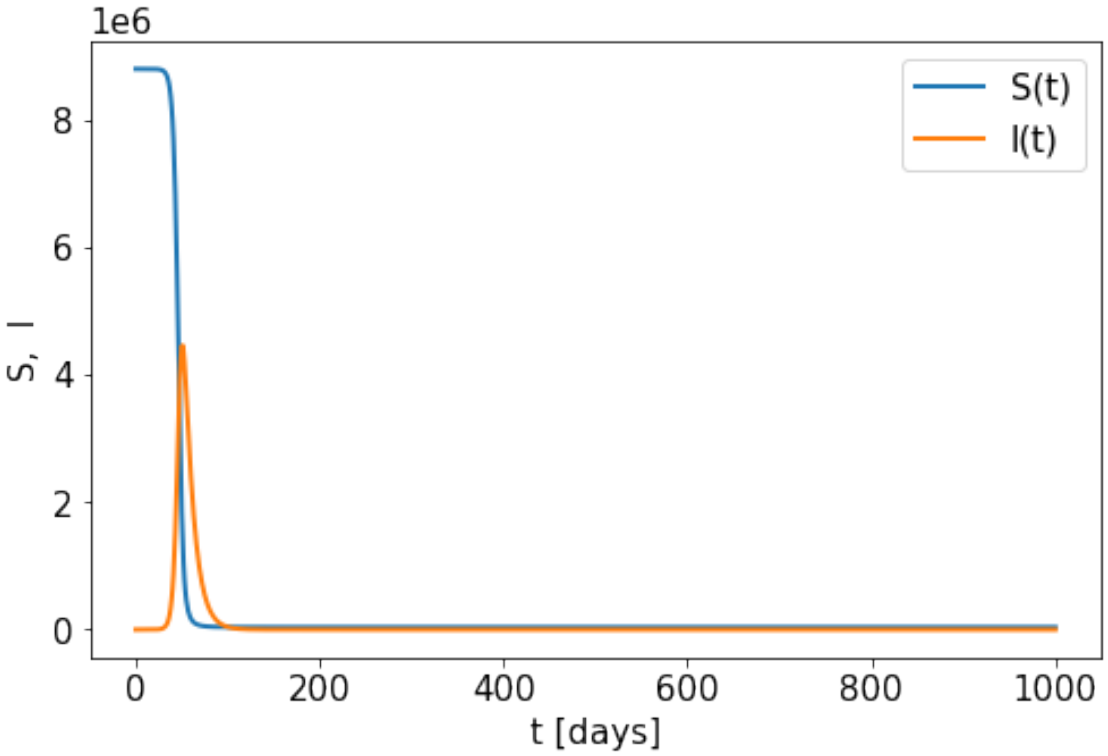


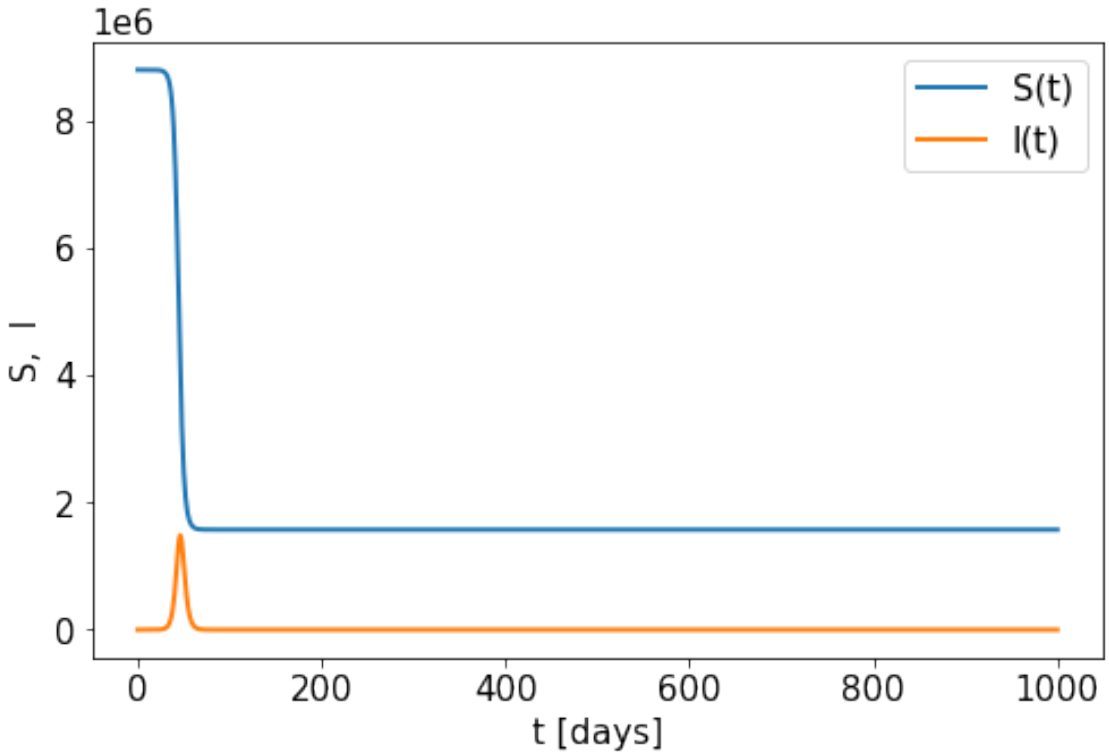
2.1.4 Results and Interpretation (~250 words) [#modeling, #dataviz]

- Interpret the results of the numerical simulation by making reference the output in the visualization(s). To fully interpret the results, you should re-run the simulation above multiple times with varying parameter inputs (b and k) and observe the behavior of your model. Include at least two additional visualizations here to support your answer. Does the behavior align with what you would expect these adjustments to have in reality (given your answer to 2.1.1.3 above)?
- *Optional*: include at least one multidimensional phase space plot and provide a full interpretation of what it shows.
- Explain how useful this model is by considering the following guiding questions: What insights can be gained? How closely do the results match what you'd expect in reality? What are the most notable assumptions of this model and what impact do they have on its usefulness?

[54]:
```
# Additional vizualizations

# new parameters
infection_rate_sec = 0.5
recovery_rate_sec = 0.1

infection_rate_third = 0.8
recovery_rate_third = 0.4
```

```
# call the function to run the simulation
SIR_Euler(b=infection_rate_sec, k=recovery_rate_sec, initial_conds=initial_vals)
SIR_Euler(b=infection_rate_third, k=recovery_rate_third,␣
 ↪initial_conds=initial_vals)
```





Answer 2.1.4 In the visualizations, the blue line represents the number of susceptible people over time, and the orange the number of infected. It can be observed in all visualizations that there is a fall of S(t), this shows the total number of infected people over time, while the orange line shows the number of infected people each day, the area of the curve is the total number. The S(t) line drops because once someone is infected they do not come back to being susceptible. Evaluating the peaks, the first one is similar to what happens in real life. COVID-19 cases had many cases during a limited number of days, and a fall in cases. However, in real life people can not only get reinfected but we also create variants of the virus, causing other peaks of infections. This means the simulation takes as assumptions everyone once infected and recovered cannot get reinfected, which at one point makes the amount of newly infected people reach zero. Even in the case where b is higher than k, what happens is there is a peak faster and higher depending on the defined proportion between the rates, but even so it gets to a point where no new cases are registered.

Footnote Since the different parameters served only for me to further explain how the simulation worked and what each trend meant I used random values.

1.2.2 Part 2.2 Agent-Based Modeling and Simulation (*OPTIONAL*)

This part of the assignment is optional and will only be scored if completed effectively (score of 4 or 5). It is a valuable chance to compare the simulation above with the agent-based simulation implemented in NetLogo.

Note about parameters: this model uses similar variables and parameters as the one above, but the parameters are not defined identically. In particular, the "Infectiousness" parameter in NetLogo

is analogous to, but not equivalent to the infection rate in the SIR model. Thus, they should not be set to the same value in both of your simulations. The infection rate in the SIR model already incorporates the interaction rate of individuals, while the NetLogo simulation sets that rate separately. In other words, the "infectiousness" parameter in NetLogo only dictates the probability of infecting someone if they come close enough, but does not take into account how frequent those interactions occur. Be sure to investigate the meaning of the other parameters as well so that you understand how to set them appropriately.

2.2.1 *Optional*: Simulation Comparison (~250 words) [#modeling] After fully exploring the NetLogo model and running multiple simulations, summarize how it compares to your Python SIR simulation above. Aim to identify the main similarities, the major differences, and at least advantage for each one. Comment on which you believe to be a more realistic representation of nature, justifying your reasoning.

Answer 2.2.1 here

2.2.2 *Optional challenge*: Your own agent-based simulation [#algorithms, #modeling]: Create your own agent-based simulation of the disease dynamics for your chosen disease in Python. You may add in real-world complexities as desired (vaccination, antibiotic use and/or development of antibiotic resistance, variability in population susceptibility). Your work needs to be explained in sufficient detail, including citations to any external sources consulted, in order to receive credit.

- One option: a tree graph can be useful in modeling person-to-person interactions.
- Another option: turtles.

[ ]:
```
# Add code to complete the optional implementation
# Add more cells as needed to explain your work
```

1.3 REFLECTION

In less than 100 words, explain how this unit has enhanced your view of the power of modeling, algorithms, and simulations to describe the natural world around you.

Reflection Modeling real-world situations with algorithms and simulations sometimes do not give us the precise answers or behaviors we see in reality, but they can get close enough and be useful for many areas of expertise. Modeling is important for product development, for understanding how viruses behave and the possible power they may have in a population, and for also assessing the behavior of animals (such as bird flocking). These approximations are important so humans can understand the world around them and act up, fixing and improving what is necessary.

Word Count: 89 words.

1.4 You're done!

You must upload TWO files:

1. A **PDF** of your entire assignment. This is to be submitted as a separate file, NOT simply inside the zipped folder. Email attachments will not be accepted. We encourage students to follow the tips available in this guide, especially the best practices listed at the end.

2. A **zipped folder** containing the .ipynb file and any other relevant files for running the notebook.