Project 12

Kyle Ma

Collaborated with Owen Smith

Executive Summary

In this project, we are exploring the usage of Euler's method and the RK2 method to explore mapping different differential equations. We will try to see how the algorithms work, and the accuracy for both.

Method implementations

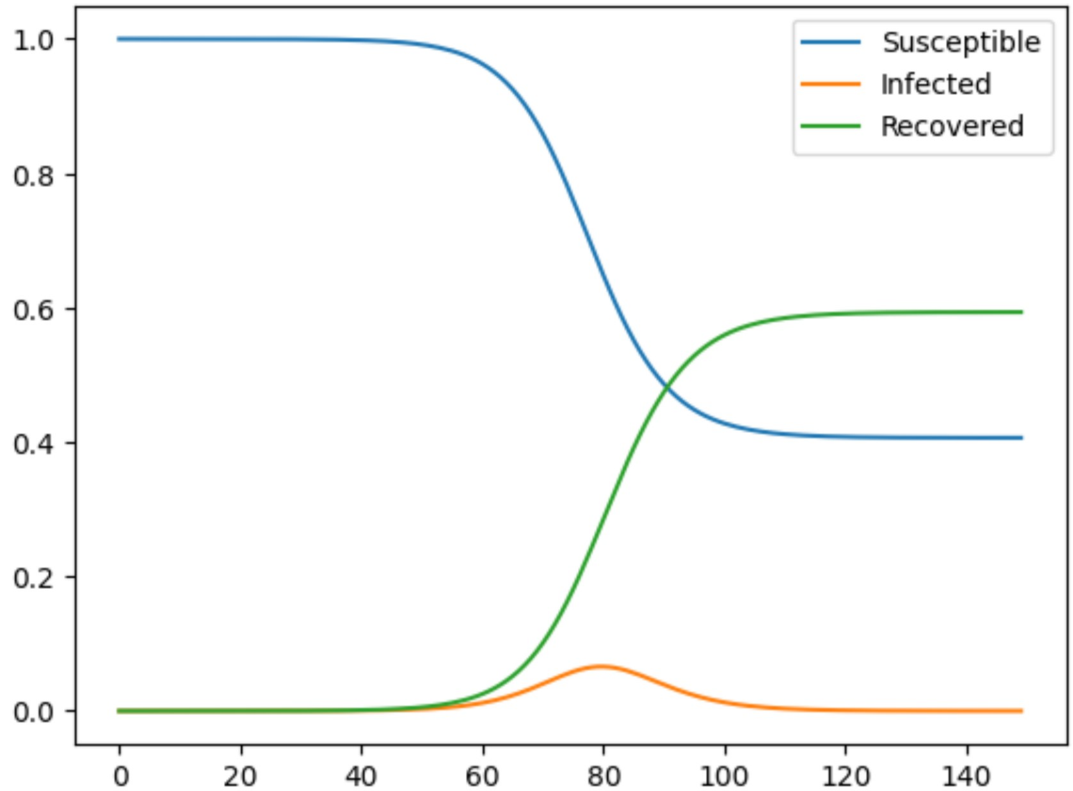
Implement the methods used in the numerical experiment with comments. Comment formatting example: https://numpydoc.readthedocs.io/en/latest/example.html

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
def sDerivative(b, s, i):   #given s derivatives
    return -b*s*i

def iDerivative(b, s, i, k):#given i derivatives
    return b*s*i-i*k

def rDerivative(k, i):      #given r derivatives
    return k * i
```

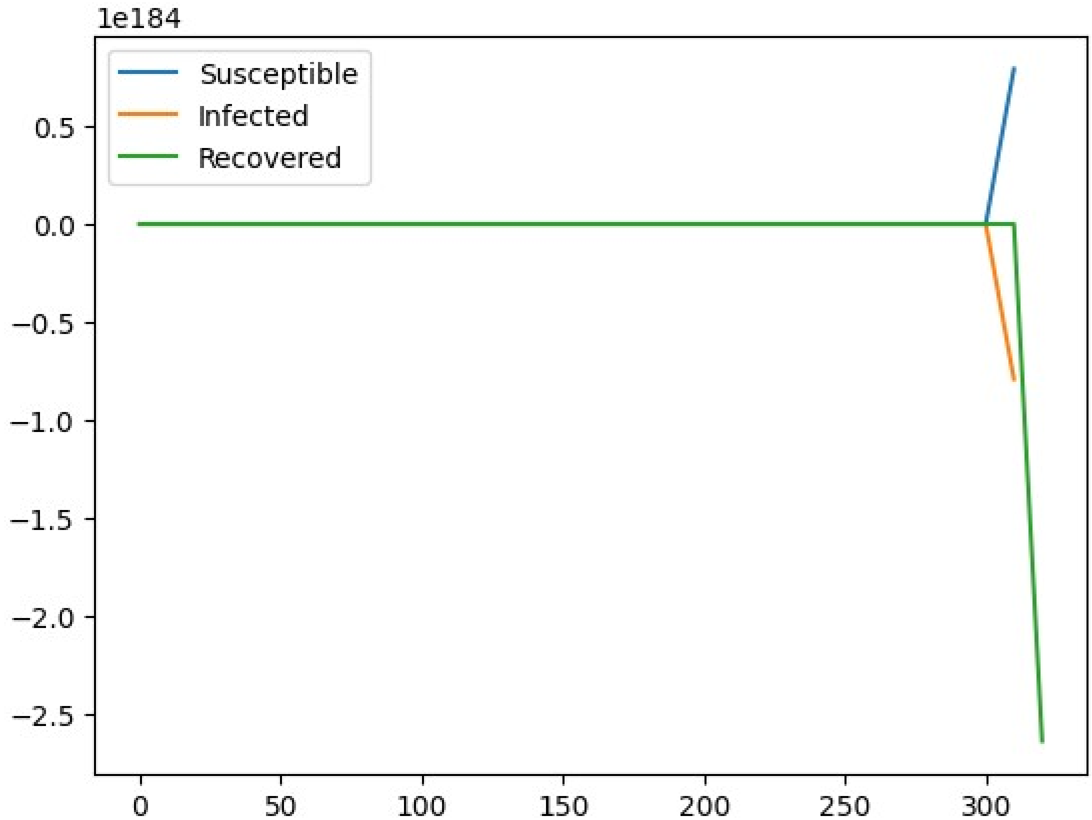
In [3]:
```
s = 1
i = 1.27*10**(-6)
r = 0
k = 1/3
b = 1/2
h = 1



def Euler(s, i, r, k, b, h):
    index = 0
    sVal = []
    iVal = []
    rVal = []
    indexVal = []
    while index < 150*h:
        #save our values
        sVal.append(s)
        iVal.append(i)
        rVal.append(r)
        indexVal.append(index)
        #get the derivative at these points
        sChange = sDerivative(b, s, i)
        iChange = iDerivative(b, s, i, k)
        rChange = rDerivative(k, i)
        #update the values, with teh derivative * our step size
        s += sChange * h
        i += iChange * h
        r += rChange * h
        index += h
    return sVal, iVal, rVal, indexVal


sVal, iVal, rVal, indexVal = Euler(s, i, r, k, b, h)
plt.plot(indexVal, sVal)
plt.plot(indexVal, iVal)
plt.plot(indexVal, rVal)
plt.legend(["Susceptible", "Infected", "Recovered"])
```

Out[3]:
```
<matplotlib.legend.Legend at 0x24d0b2abfa0>
```



In [4]:
```
sVal, iVal, rVal, indexVal = Euler(s, i, r, k, b, 10)
plt.plot(indexVal, sVal)
plt.plot(indexVal, iVal)
plt.plot(indexVal, rVal)
plt.legend(["Susceptible", "Infected", "Recovered"])
```

Out[4]:
```
<matplotlib.legend.Legend at 0x24d0d474d90>
```

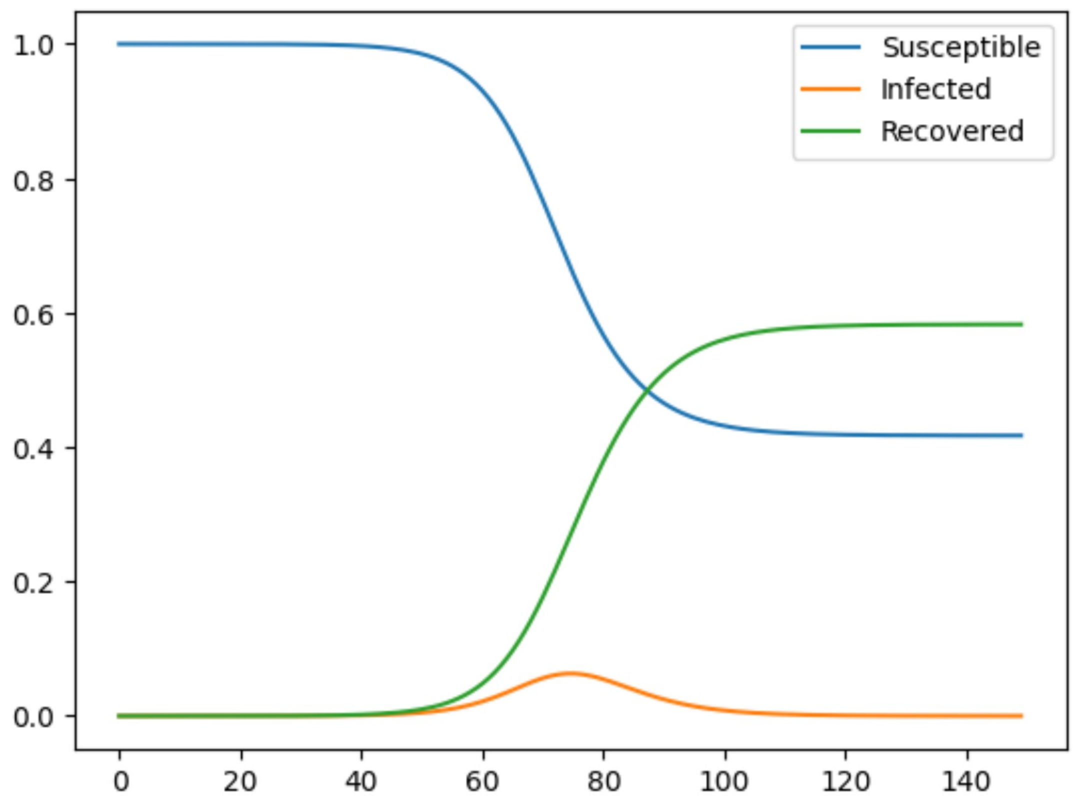

1e184
Susceptible
Infected
Recovered
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
0
50
100
150
200
250
300

In [5]:

```
def RK2(s, i, r, k , b, h):
    index = 0
    sVal = []
    iVal = []
    rVal = []
    indexVal = []
    while index < 150*h:
        #save our values
        sVal.append(s)
        iVal.append(i)
        rVal.append(r)
        indexVal.append(index)
        #finds the derivative at the current points
        sChange = sDerivative(b, s, i)
        iChange = iDerivative(b, s, i, k)
        rChange = rDerivative(k, i)
        #goes "halfway"
        sHalf = sChange * h/2 + s
        iHalf = iChange * h/2 + i
        rHalf = rChange * h/2 + r
        #finds the derivatives at the halfway
        sChange2 = sDerivative(b, sHalf, iHalf)
        iChange2 = iDerivative(b, sHalf, iHalf, k)
        rChange2 = rDerivative(k, iHalf)
        s += sChange2
        i += iChange2
        r += rChange2
        index += h
    return sVal, iVal, rVal, indexVal

sVal, iVal, rVal, indexVal = RK2(s, i, r, k, b, 1)
# print(indexVal)
# print(sVal)
indices = [i for i in range(len(indexVal))]
plt.plot(indices, sVal)
plt.plot(indices, iVal)
plt.plot(indices, rVal)
plt.legend(["Susceptible", "Infected", "Recovered"])
```

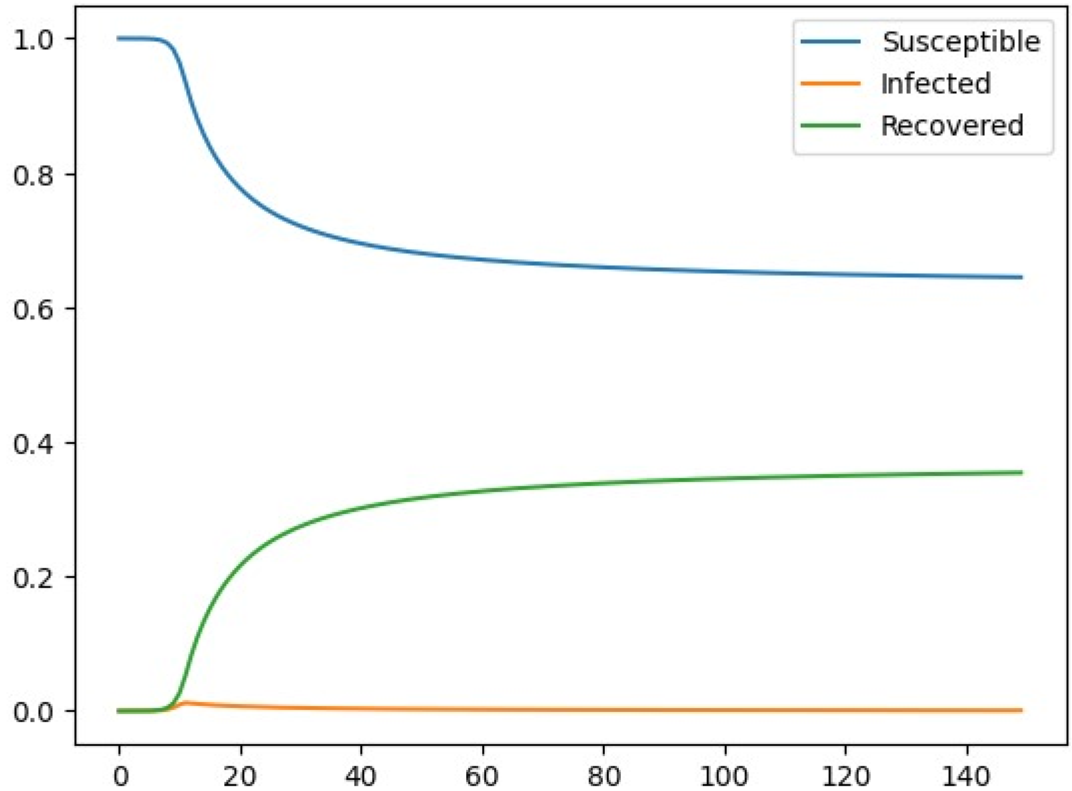
Out[5]:

```
<matplotlib.legend.Legend at 0x24d0b3786a0>
```



In [6]:
```
sVal, iVal, rVal, indexVal = RK2(s, i, r, k, b,101)
# print(indexVal)
# print(sVal)
indices = [i for i in range(len(indexVal))]
plt.plot(indices, sVal)
plt.plot(indices, iVal)
plt.plot(indices, rVal)
plt.legend(["Susceptible", "Infected", "Recovered"])
```

Out[6]:
```
<matplotlib.legend.Legend at 0x24d0b417400>
```



Conclusions

For both methods, I used 1 day as the step size. And for both, it creates a plot that accurately matches the plots that were posted on the website. This choice is the most intuitive, as we would just calculate the change at each day, and change it, then it's only day by day. I tried using a time step of 10 for each. And here, we can see the advantages of RK2. The RK2 plot still sort of looks like the plot on the website, while the plot for Euler looks nothing like the plot online.