In [ ]:
```
"""
Assignment 2
Vladislav Monakhov
32492928
"""
```

In [ ]:
```
"""
Part 1
"""
```

In [7]:
```
"""
I have chosen the SIR model to model the spread of disease. The question that I am posin
of infected people affect the spread of the virus? I have chosen this question because r
mortality rates, and the death of an infectious person means that they are no longer abl
believe that the mortality rate of a disease will significantly impact the dynamics of i
inspiration for this modification is the video game 'Plague Inc.', which I used to play
Seeing the model reminded me of this game, where you have to evolve a virus with the goa
infecting and eliminating the whole world. I remembered how if you made the virus deadly
of its carriers faster than it could spread, wiping itself out and causing you to lose t

In the base model, there are three states: Susceptible, Infected, and Recovered. Suscept
become Infected, getting other Susceptible people sick until they become Recovered. Ther
the disease which we will call 'i' and a recovery rate which we will call 'r', as well a
These states interact with each other over discrete time as such:
S(t+1) = F(S(t), I(t))
I(t+1) = G(S(t), I(t))
R(t+1) = H(I(t))

S and I affect each other because infection happens when the two groups interact, while
In terms of the rates of change:
dS/dt = -iSI/N
dI/dt = iSI/N - rI
dR/dt = rI

To incorporate the death of infected people, I will be adding a fourth state: Deceased.
will die from the disease at rate d, and will no longer be able to infect anyone. The mo
S(t+1) = F(S(t), I(t))
I(t+1) = G(S(t), I(t))
D(t+1) = J(I(t))
R(t+1) = H(I(t))

In terms of the rates of change:
dS/dt = -iSI/N
dI/dt = iSI/N - dI - rI
dD/dt = dI
dR/dt = rI

"""
```

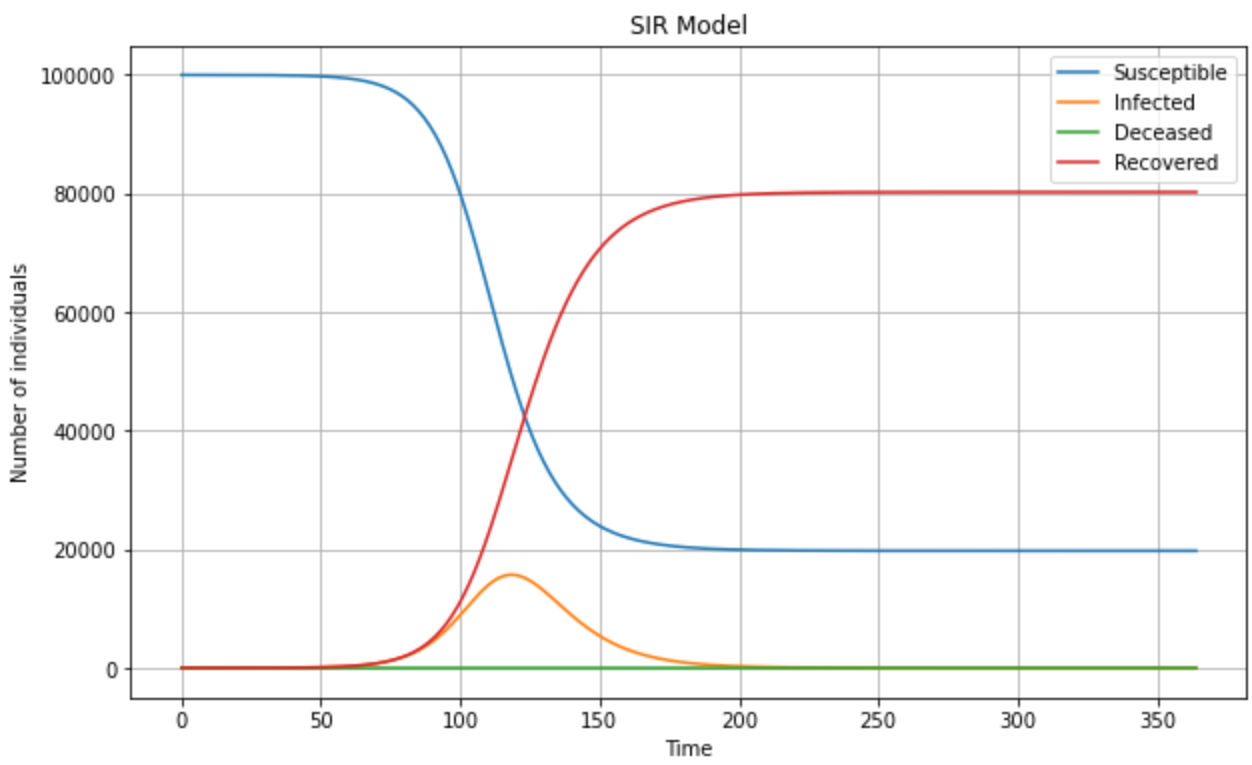
Out[7]:
```
'\n'
```

In [1]:
```
"""
Part 2 - discrete time analysis
S(t+1) = S(t) - i*S(t)*I(t)/N
I(t+1) = I(t) + i*S(t)*I(t)/N - rI(t) - dI(t)
R(t+1) = R(t) + rI(t)
D(t+1) = D(t) + dI(t)

To avoid complicating the plots, the population will always be 100,000 and the simulatio
treating one unit of time as a day.
"""
```

```
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt
def discrete(num,inf,i,d,r,t):
    """
    Inputs
    num: population
    i: infection rate
    d: death rate
    r:recovery rate
    t: number of iterations
    """
    rec = 0
    sus = num-inf
    dead = 0
    #Initialising states
    S = np.zeros(t)
    I = np.zeros(t)
    D = np.zeros(t)
    R = np.zeros(t)

    S[0] = sus
    I[0] = inf
    D[0] = dead
    R[0] = rec
    #Equations
    for a in range(1,t):
        S[a] = S[a-1] - i*S[a-1]*I[a-1]/num
        I[a] = I[a-1] + i*S[a-1]*I[a-1]/num -d*I[a-1] -r*I[a-1]
        D[a] = D[a-1] +d*I[a-1]
        R[a] = R[a-1] + r*I[a-1]
    #Plotting
    time = np.arange(t)
    plt.figure(figsize=(10, 6))
    plt.plot(time, S, label='Susceptible')
    plt.plot(time, I, label='Infected')
    plt.plot(time,D, label="Deceased")
    plt.plot(time, R, label='Recovered')
    plt.xlabel('Time')
    plt.ylabel('Number of individuals')
    plt.title('SIR Model')
    plt.legend()
    plt.grid(True)
    plt.show()
    print("Susceptible at end: " + str(S[-1]))
    print("Infected at end: " + str(I[-1]))
    print("Deceased at end: " + str(D[-1]))
    print("Recovered at end: " + str(R[-1]))
```

In [2]:

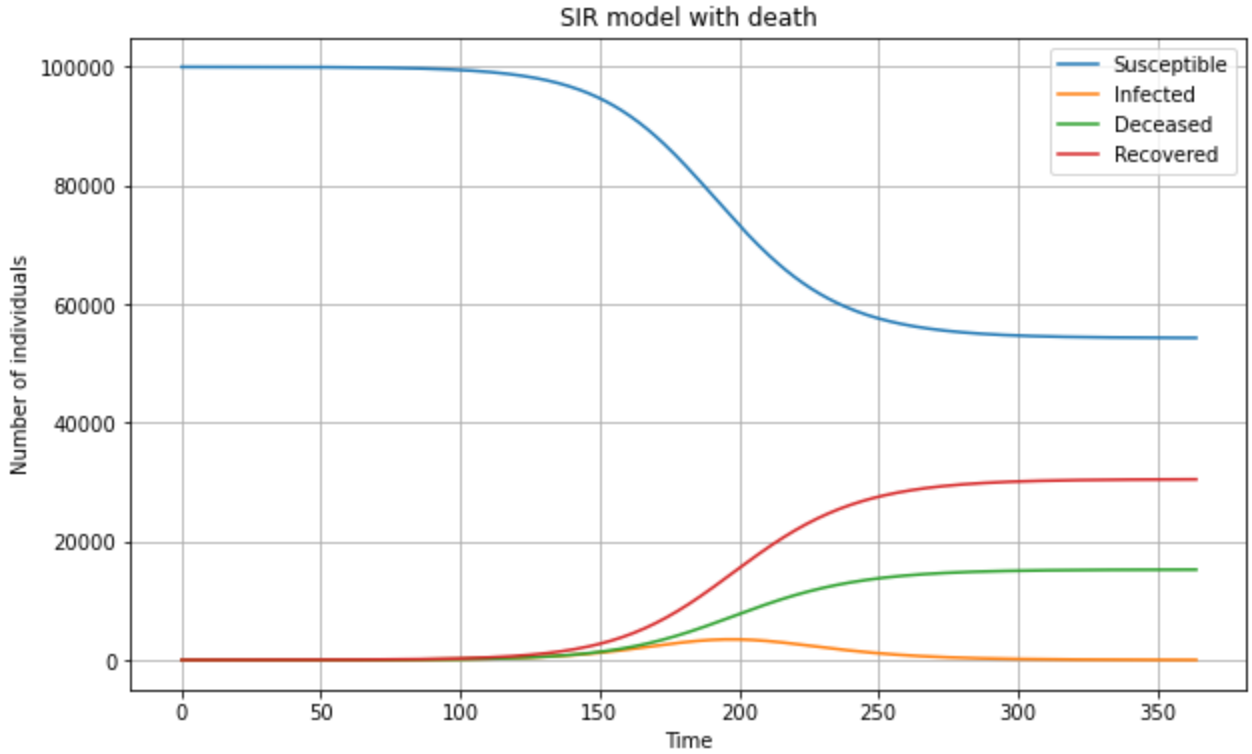
```
"""
We will begin by setting the death rate to zero to establish a baseline and setting arbi
else.
"""
discrete(100000,1,0.20,0,0.1, 365)
```




```
Susceptible at end: 19751.20835353376
Infected at end: 0.00988428065926223
Deceased at end: 0.0
Recovered at end: 80248.78176218543
```

In [126…

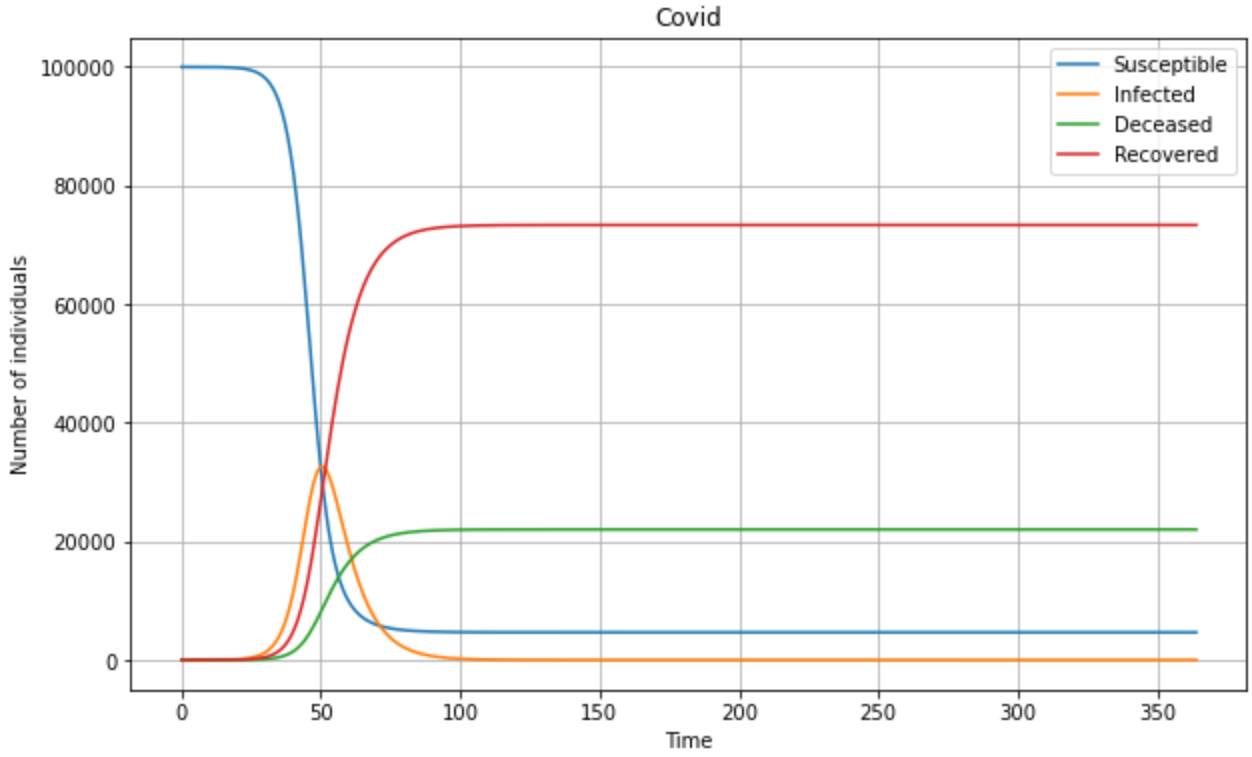
```
"""
We can see that after around 200 iterations, there is a stabilisation as there are enoug
the interaction of S and I is lower than the recovery rate, leading to about 80% of the
with 20% of the population avoiding the disease. Now, we will add the possibility of dea
"""
discrete(100000,1,0.20,0.05,0.1, 365)
```




```
Susceptible at end: 54305.85676059492
Infected at end: 10.73264269945694
Deceased at end: 15227.803532235244
Recovered at end: 30455.60706447049
```

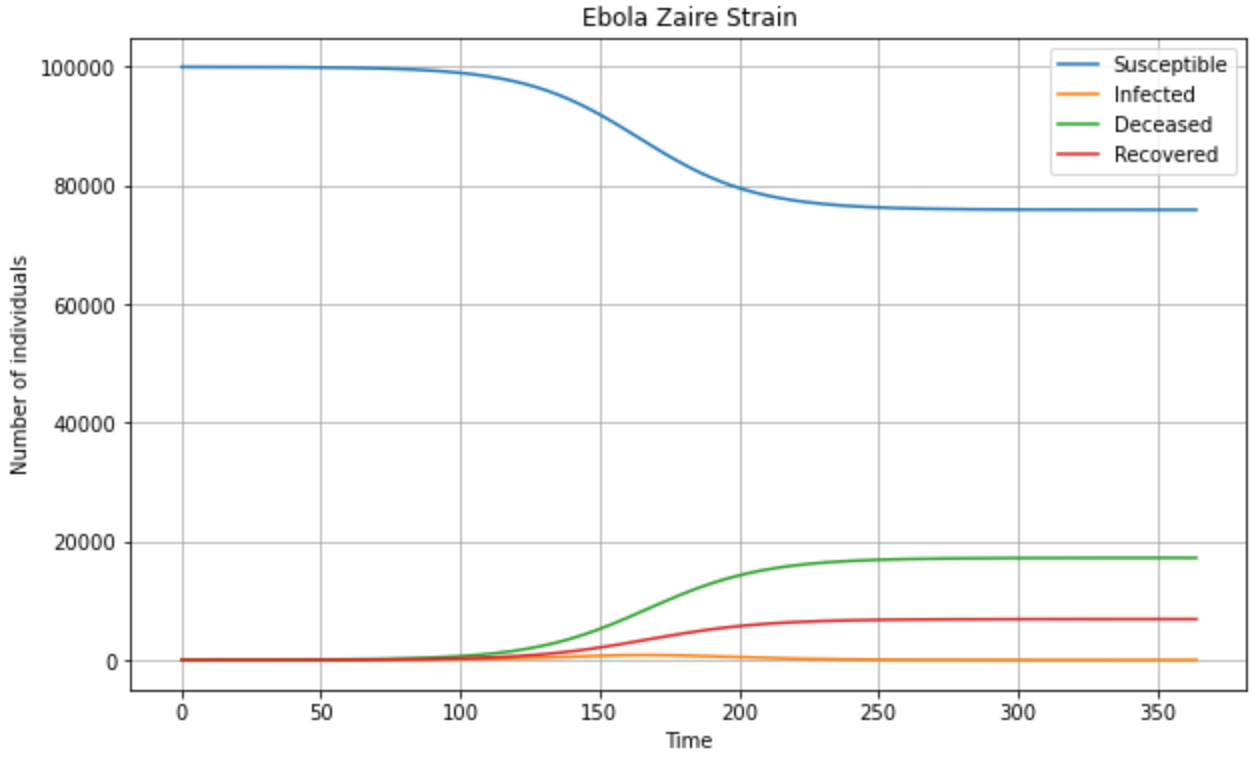
In [ ]:

In [122…
```
"""
Let us now explore how the strength of the death coefficient affects model spread. In re
have high death rates often end up causing less deaths overall than ones which have low
The model can illustrate how a low-mortality virus such as Covid can end up causing more
a much higher death rate.
"""
#A highly contagious virus with low death rate
discrete(100000,1,0.4,0.03,0.1,365)
```




```
Susceptible at end: 4653.644383222592
Infected at end: 5.7236951330594715e-12
Deceased at end: 22003.005142333262
Recovered at end: 73343.3504744441
```

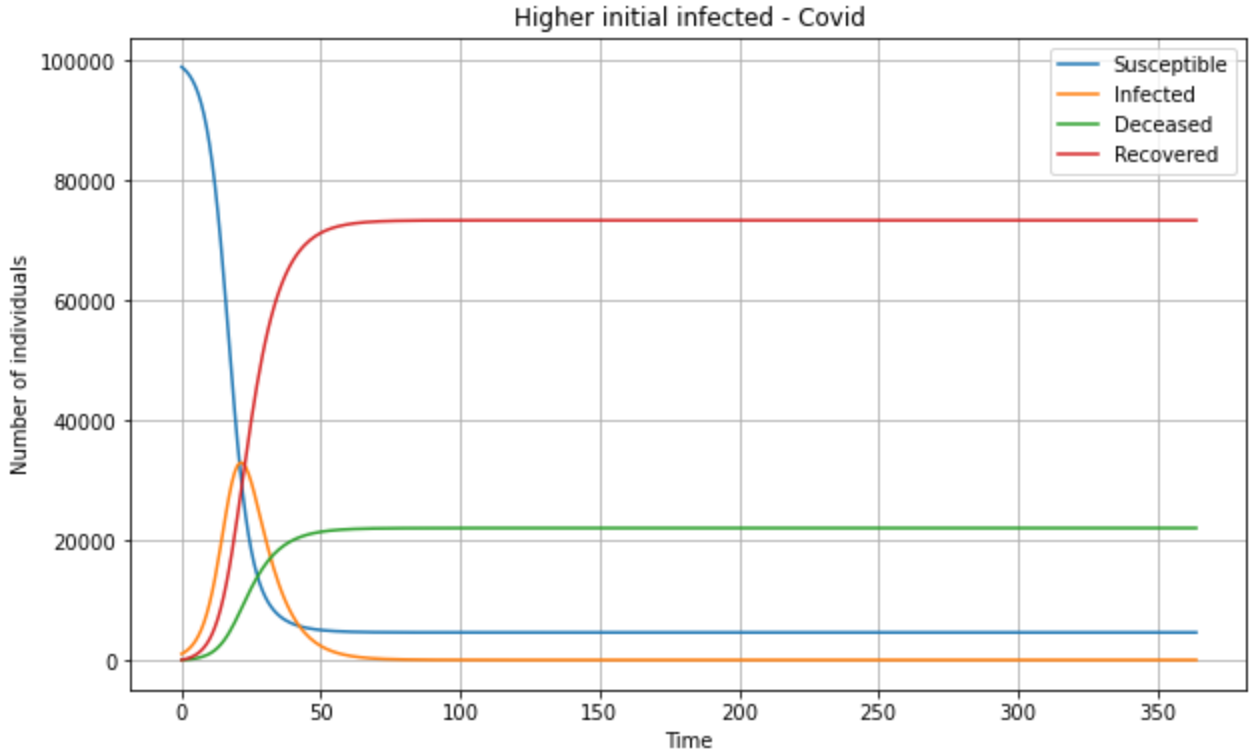
In [110…
```
#An equally contagious virus with high death rate
discrete(100000,1,0.4,0.25,0.1,365)
```




```
Susceptible at end: 75907.34707111707
Infected at end: 0.27275017168037785
Deceased at end: 17208.842984793784
Recovered at end: 6883.537193917516
```

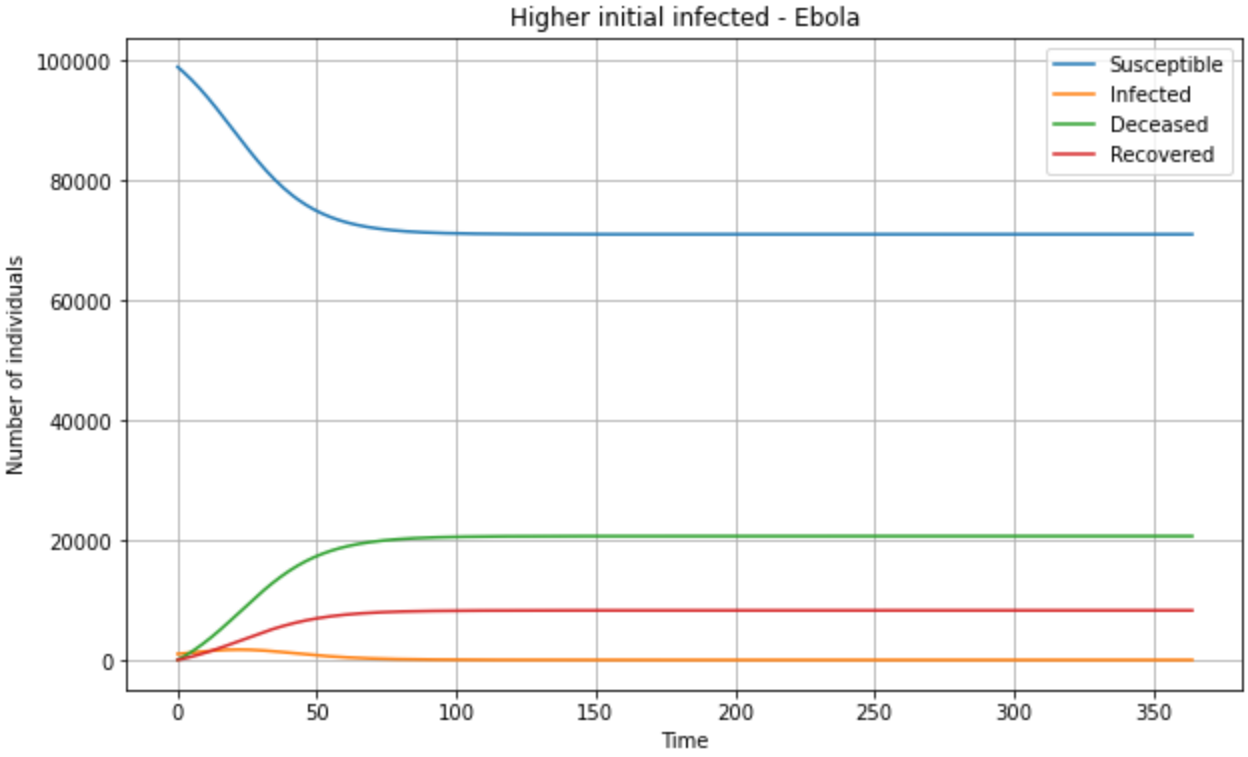
In [130…

```
"""
Here, we see that having a lower death rate can actually lead to more deaths overall - t
reaches the majority of the population. Finally, let's explore what happens when the ini
larger
"""
discrete(100000,1000,0.4,0.03,0.1,365)
```




```
Susceptible at end: 4587.294699511096
Infected at end: 1.6817153473338772e-13
Deceased at end: 22018.316607805144
Recovered at end: 73394.3886926837
```

In [207…
```
discrete(100000,1000,0.4,0.25,0.1,365)
```



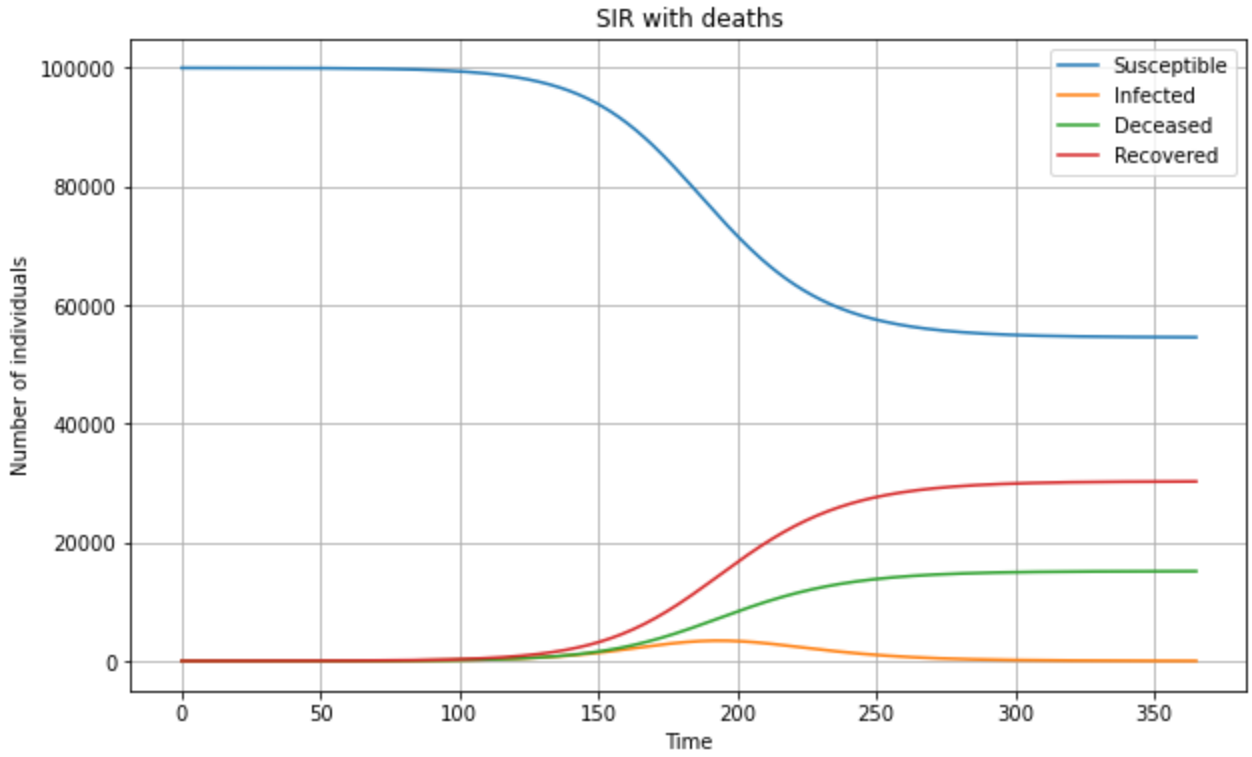

```
Susceptible at end: 71061.43468988613
Infected at end: 5.300213885788701e-07
Deceased at end: 20670.403792559988
Recovered at end: 8268.161517023998
```

In [ ]:
```
"""
Comparing the same two models but with an initial outbreak of 1000 people, we can see th
infection doesn't affect our "Covid" strain, which ends up with almost identical statist
person sick initially. However, the deadly strain has approximately 17% more deaths - th
it to spread much faster initially before reaching a plateau.
"""
```

In [175…
```
"""
Part 3 - Continuous time
Now we will implement the model in continuous time:

dS/dt = -iSI/N
dI/dt = iSI/N - dI - rI
dD/dt = dI
dR/dt = rI

"""
# Solving with odeint
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt
def sir(y,t,num,i,d,r):
    S,I,D,R = y
    dSdt = -i*S*I/num
    dIdt = i*S*I/num - d*I - r*I
    dDdt = d*I
    dRdt = r*I
    return dSdt,dIdt,dDdt,dRdt

def continuous_odeint(num,inf,i,d,r,t):
    rec = 0
    sus = num-inf
    dead = 0
```

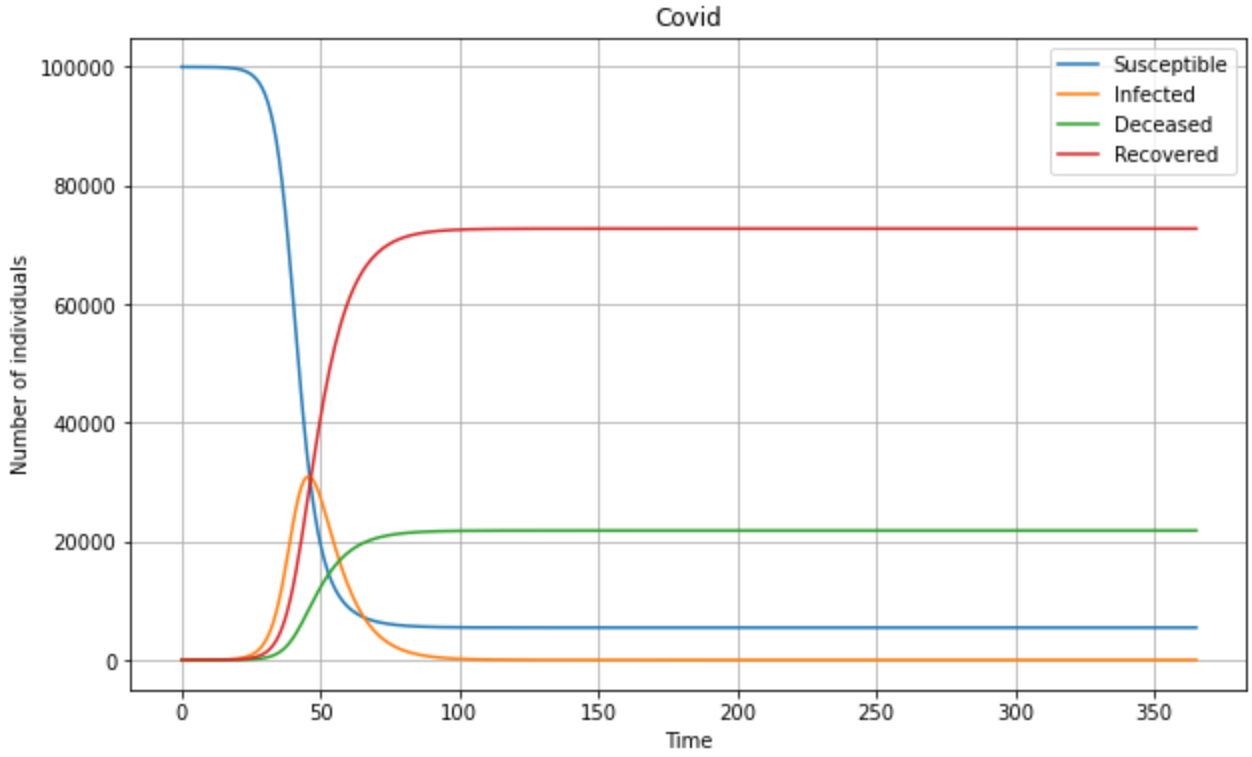
```
    t = np.linspace(0, t, 1000)

    y0 = sus,inf,dead,rec
    res = odeint(sir,y0,t,args=(num,i,d,r))
    S,I,D,R = res.T
    #plotting
    plt.figure(figsize=(10, 6))
    plt.plot(t, S, label='Susceptible')
    plt.plot(t, I, label='Infected')
    plt.plot(t, D, label='Deceased')
    plt.plot(t, R, label='Recovered')
    plt.xlabel('Time')
    plt.ylabel('Number of individuals')
    plt.title('Higher initial infected - Ebola')
    plt.legend()
    plt.grid(True)
    plt.show()
    print("Susceptible at end: " + str(S[-1]))
    print("Infected at end: " + str(I[-1]))
    print("Deceased at end: " + str(D[-1]))
    print("Recovered at end: " + str(R[-1]))
```

In [167…
```
continuous_odeint(100000,1,0.20,0.05,0.1, 365)
```



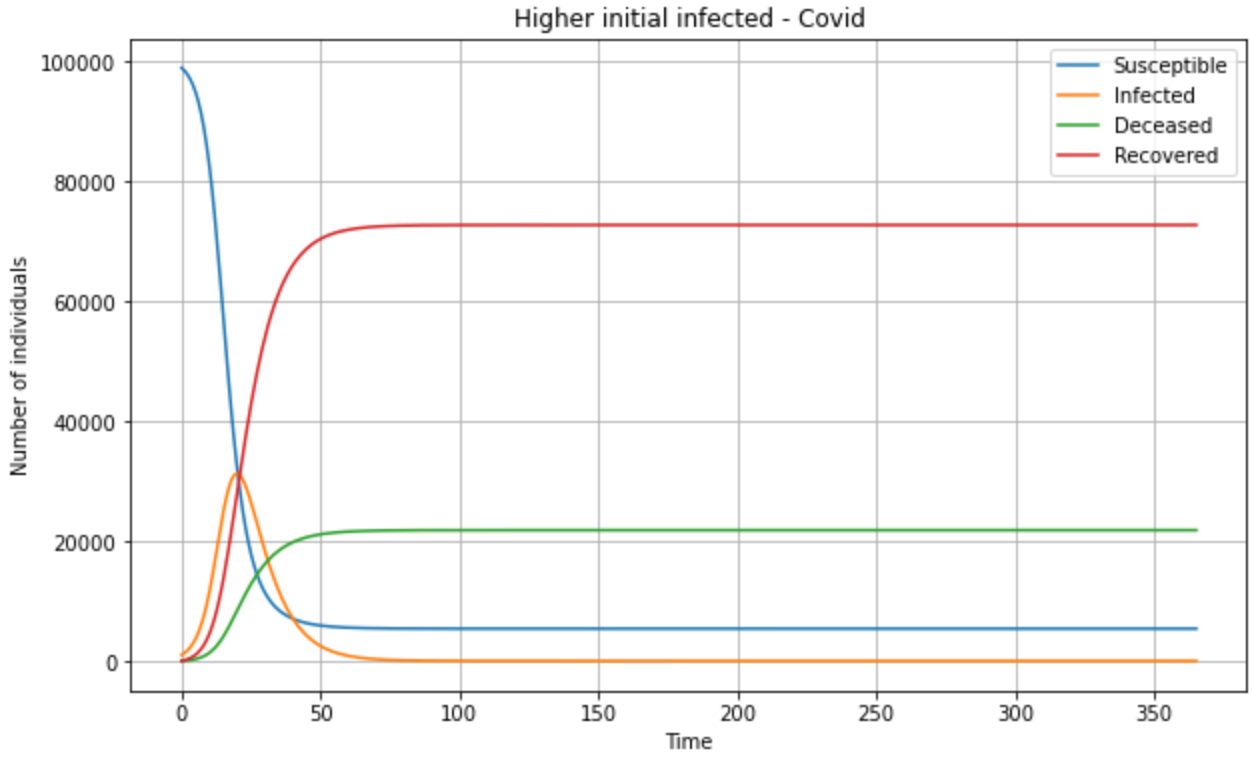

```
Susceptible at end: 54586.822870170115
Infected at end: 10.601804957418306
Deceased at end: 15134.191774957519
Recovered at end: 30268.383549915037
```

In [171…
```
continuous_odeint(100000,1,0.4,0.03,0.1,365)
```



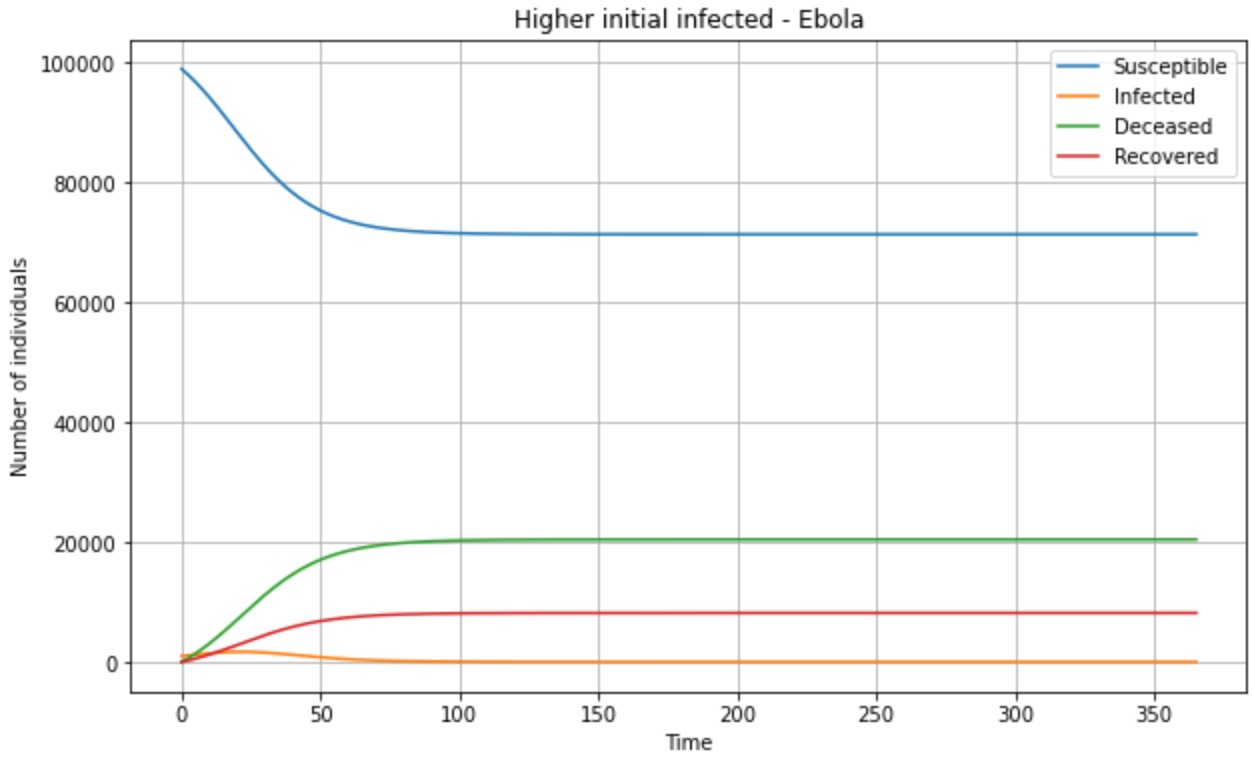

```
Susceptible at end: 5452.055121437934
Infected at end: -4.6468086086532625e-10
Deceased at end: 21818.756510437517
Recovered at end: 72729.18836812506
```

In [165…
```
continuous_odeint(100000,1000,0.4,0.03,0.1,365)
```




```
Susceptible at end: 5386.754418724702
Infected at end: -4.042092251328226e-10
Deceased at end: 21833.825903371297
Recovered at end: 72779.41967790441
```

In [176…
```
continuous_odeint(100000,1000,0.4,0.25,0.1,365)
```



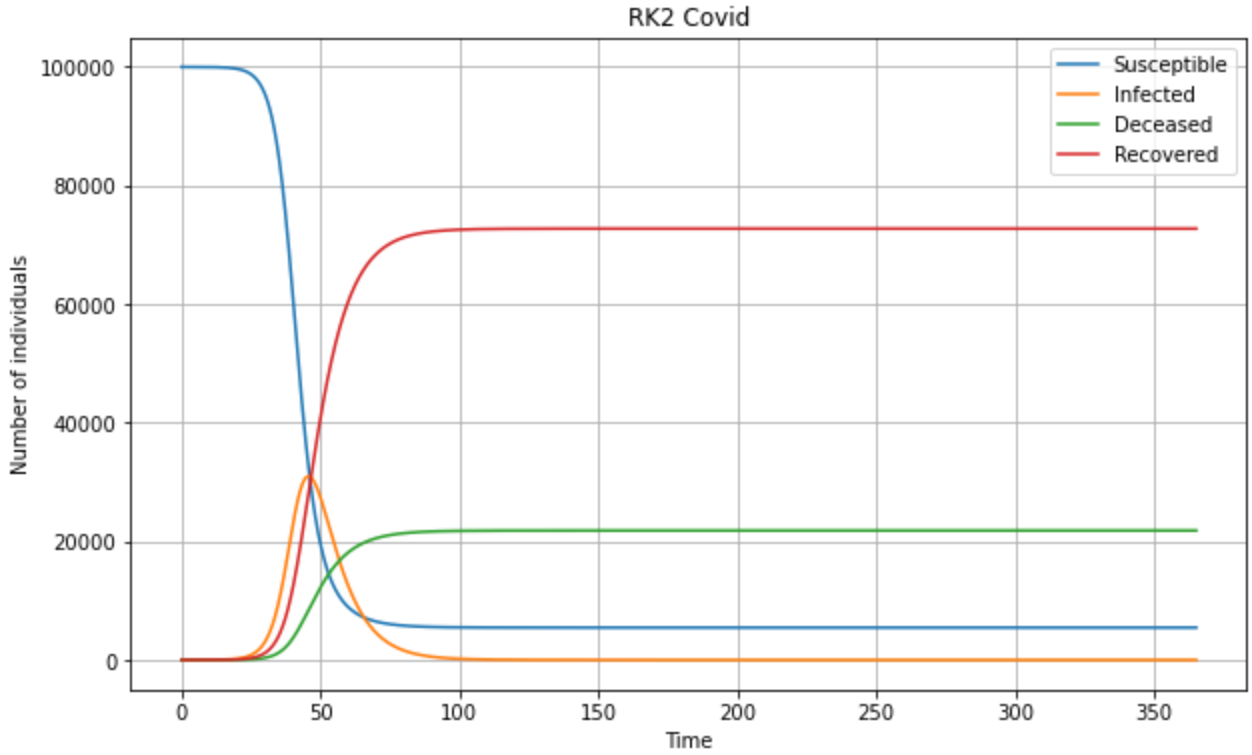

```
Susceptible at end: 71386.36219166768
Infected at end: 1.5042055349738328e-06
Deceased at end: 20438.31271916289
Recovered at end: 8175.325087665152
```

In [233…

```
"""
The continuous odeint model has the exact same results as the discrete time model. Let's
"""
def sir(t, y, N, i, d, r):
    S, I, D, R = y
    dSdt = -i*S*I/N
    dIdt = i*S*I/N - d*I - r*I
    dDdt = d*I
    dRdt = r*I
    return [dSdt, dIdt, dDdt, dRdt]
def rk2(f, t0, y0, time, h, args=()):
    t = t0
    y = y0
    t_values = [t0]
    y_values = [y0]
    while t < time:
        k1 = [h*value for value in f(t, y, *args)]
        k2 = [h*value for value in f(t+h, [y[i]+k1[i] for i in range(len(y))], *args)]
        y = [y[i]+(k1[i]+k2[i])/2 for i in range(len(y))]
        t = t+h
        t_values.append(t)
        y_values.append(y)
    return t_values, y_values
#Conditions
def continuous_rk(num,inf,i,d,r,t):
    S0 = num-1
    I0 = 1
    D0 = 0
    R0 = 0
    y0 = [S0, I0, D0, R0]
    N = num
    t0 = 0
    time = t
    h = 0.1
```

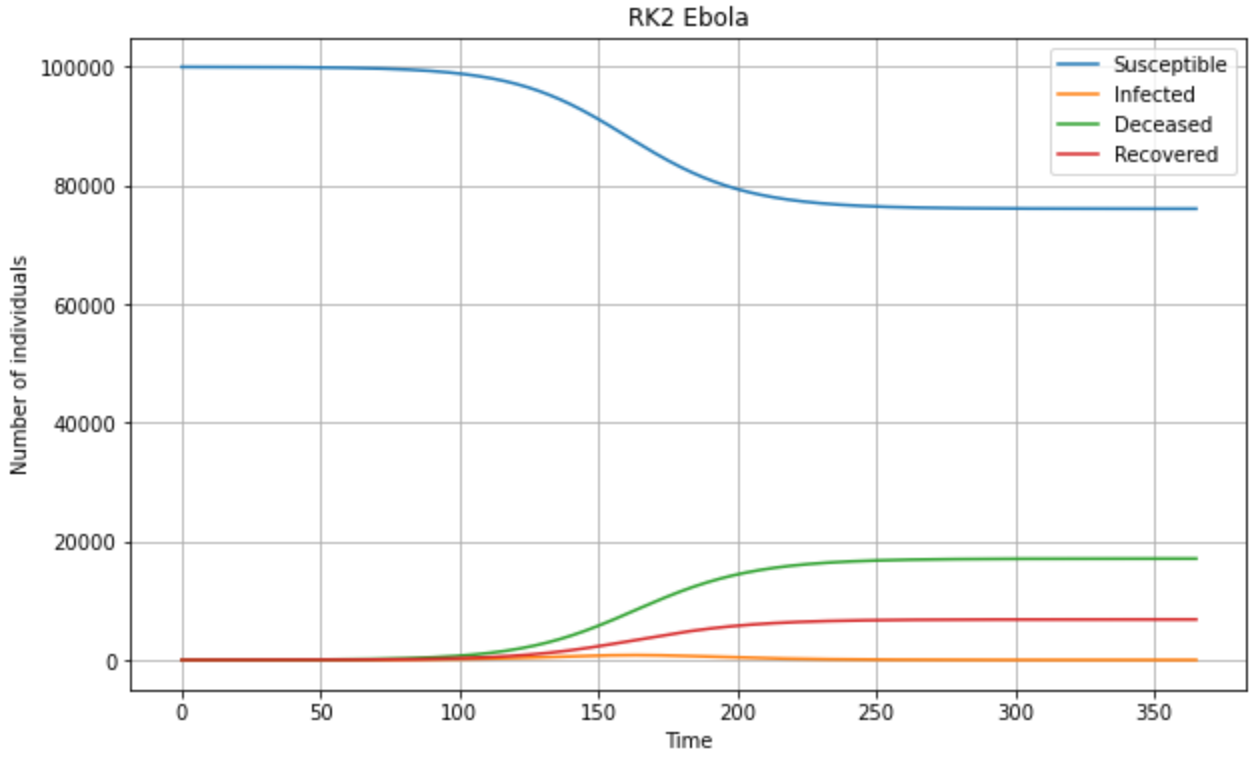
```
    # Solving
    t_values, y_values = rk2(sir, t0, y0, time, h, args=(N, i, d, r))
    S = [y[0] for y in y_values]
    I = [y[1] for y in y_values]
    D = [y[2] for y in y_values]
    R = [y[3] for y in y_values]
    # Plotting
    plt.figure(figsize=(10, 6))
    plt.plot(t_values, S, label='Susceptible')
    plt.plot(t_values, I, label='Infected')
    plt.plot(t_values, D, label='Deceased')
    plt.plot(t_values, R, label='Recovered')
    plt.xlabel('Time')
    plt.ylabel('Number of individuals')
    plt.title('RK2 Ebola')
    plt.legend()
    plt.grid(True)
    plt.show()
    print("Susceptible at end: " + str(S[-1]))
    print("Infected at end: " + str(I[-1]))
    print("Deceased at end: " + str(D[-1]))
    print("Recovered at end: " + str(R[-1]))
```

In [187…
```
continuous_rk(100000,1,0.4,0.03,0.1,365)
```




```
Susceptible at end: 5452.308502905326
Infected at end: 6.515638759458337e-11
Deceased at end: 21818.698037790993
Recovered at end: 72728.99345930314
```

In [189…
```
continuous_rk(100000,1,0.4,0.25,0.1,365)
```



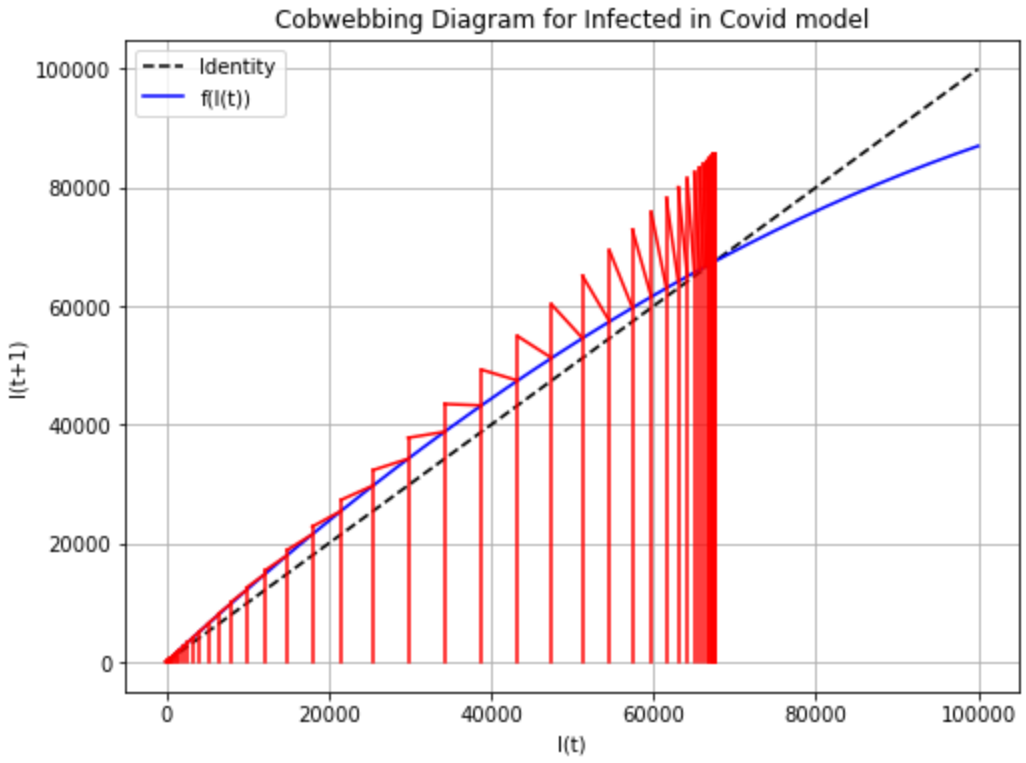

```
Susceptible at end: 76083.95935722107
Infected at end: 0.3018035406011474
Deceased at end: 17082.670599455876
Recovered at end: 6833.068239782365
```

In [ ]:
```
"""
All of the models provide the same results, meaning that discrete vs continuous calculat
this SIR model. This makes sense because the discrete implementation is a simplification
application of the model - people get sick and recover throughout a day, and the discret
changes at the end of a day while the continuous can be interpreted as tracking changes
"""
```

In [ ]:
```
"""
Part 4 - Steady state analysis
A steady state in a model occurs when there is no longer any change in its variables ove
occurs when the infection rate falls back to zero (or close enough to be inconsequential
it is guaranteed to occur after a certain amount of time for any given initial condition
there is a limited number of susceptible people. The infection rate climbs initially but
number of Susceptible people goes down due to infection and the number of infected peopl
and recovery. By its nature, the base SIR model and our modified model are not cyclical.
to not reach a steady state. It is worth nothing that the model technically can have inc
in its values because we are not rounding to the nearest integer to avoid rounding error
its application this does not qualify as real changes - you can't have 0.01 of a person.

"""
```

In [12]:
```
import matplotlib.pyplot as plt
import numpy as np
#initialise variables
i = 0.4
r = 0.1
d = 0.03
num = 100000
S0 = num-1
I0 = 1
R0 = 0
D0 = 0

# Define the function
def f(I, S):
```

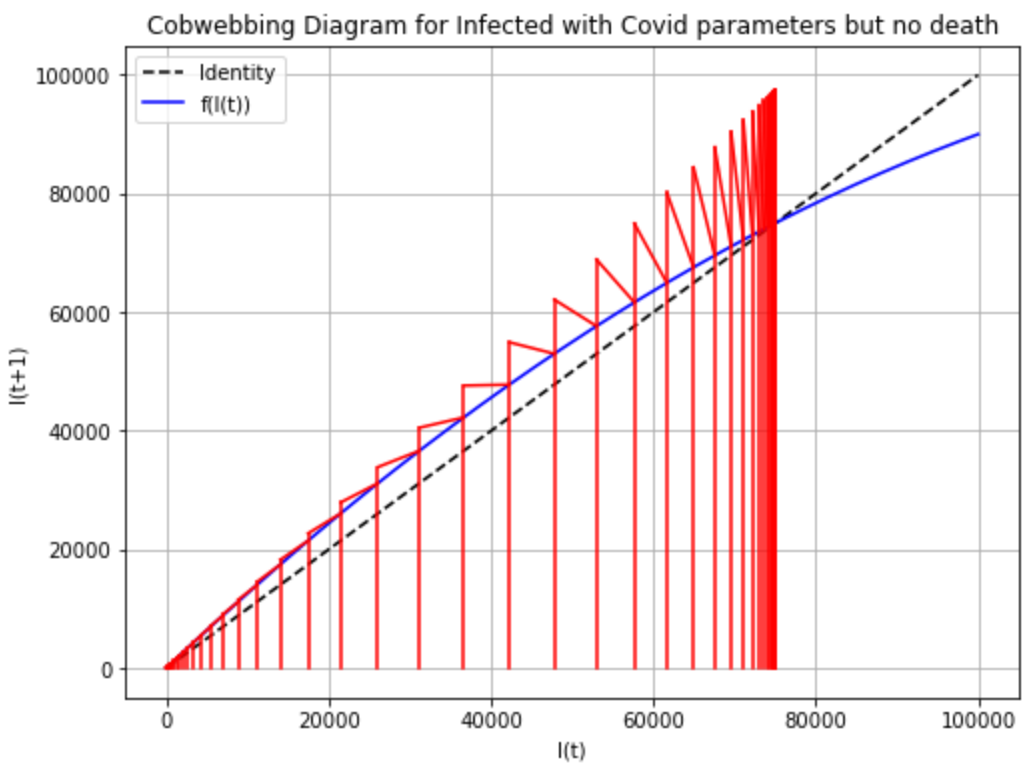
```
    return I+i*S*I/num - r*I - d*I
I_range = np.linspace(0, num, 1000)

# Compute equations
S = S0 - I_range
f_values = f(I_range, S)


# Plotting
plt.figure(figsize=(8, 6))
plt.plot(I_range, I_range, 'k--', label='Identity')
plt.plot(I_range, f_values, 'b-', label='f(I(t))')

# Add the cobweb diagram
I_current = I0
plt.plot([I_current, I_current], [0, f(I_current, S0)], 'r-')  # Vertical line
plt.plot([I_current, f(I_current, S0)], [f(I_current, S0), f(I_current, S0)], 'r-')  # H
I_next = f(I_current, S0)

# Iterate the difference equation for a fixed number of steps
num_iterations = 75
for _ in range(num_iterations):
    plt.plot([I_current, I_current], [0, f(I_current, S0)], 'r-')
    plt.plot([I_current, I_next], [f(I_current, S0), f(I_next, S0 - I_next)], 'r-')
    I_current = I_next
    I_next = f(I_current, S0 - I_current)

plt.xlabel('I(t)')
plt.ylabel('I(t+1)')
plt.title('Cobwebbing Diagram for Infected in Covid model')
plt.legend()
plt.grid(True)
plt.show()
# Acknowledgement: Generative AI was used to help with the implementation of the cobwebb
```



In [ ]:
```
"""
We can see that the Infection rate trends towards a steady state. Given the previously d
SIR model, this trend must hold for all of the different initial conditions and paramete
analysis by making one final comparison between a model with and without death.
"""
```

In [14]:

```
import matplotlib.pyplot as plt
import numpy as np
#initialise variables
i = 0.4
r = 0.1
d = 0.03
num = 100000
S0 = num-1
I0 = 1
R0 = 0
D0 = 0

# Define the function
def f(I, S):
    return I+i*S*I/num - r*I
I_range = np.linspace(0, num, 1000)

# Compute equations
S = S0 - I_range
f_values = f(I_range, S)


# Plotting
plt.figure(figsize=(8, 6))
plt.plot(I_range, I_range, 'k--', label='Identity')
plt.plot(I_range, f_values, 'b-', label='f(I(t))')

# Add the cobweb diagram
I_current = I0
plt.plot([I_current, I_current], [0, f(I_current, S0)], 'r-')  # Vertical line
plt.plot([I_current, f(I_current, S0)], [f(I_current, S0), f(I_current, S0)], 'r-')  # H
I_next = f(I_current, S0)

# Iterate the difference equation for a fixed number of steps
num_iterations = 75
for _ in range(num_iterations):
    plt.plot([I_current, I_current], [0, f(I_current, S0)], 'r-')
    plt.plot([I_current, I_next], [f(I_current, S0), f(I_next, S0 - I_next)], 'r-')
    I_current = I_next
    I_next = f(I_current, S0 - I_current)

plt.xlabel('I(t)')
plt.ylabel('I(t+1)')
plt.title('Cobwebbing Diagram for Infected with Covid parameters but no death')
plt.legend()
plt.grid(True)
plt.show()
```



In [ ]:

```
"""
Once again, a steady state is reached, however slightly later, reflecting that more peop
model without death. To conclude, the SIR model with death is guaranteed to reach a comp
the number of infected people falls sufficiently. The point at which this happens depend
disease is and how fast people recover or die from the infection.
"""
```