Quantitative Macroeconomics
Homework 2

Lorenzo De Carolis
(In collaboration with Victor Caballero Cordero and Santiago Albarracin)

October 13, 2020

Question 1: Computing Transitions in a Representative Agent Economy

(1.a) Compute the steady-state. Choose z to match an annual capital-output ratio of 4, and an investment-output ratio of .25.

For finding the Steady-State (SS), we look for an analytical solution that is obtaining solving the maximization problem by the following Lagrangian function:

$$L = E_0\left[\sum_{t=0}^{\infty}\beta^t u(c_t)\right] + \sum_{t=0}^{\infty}[\lambda_t(k_t^{1-\theta}(zh_t)^{\theta} + (1-\delta)k_t - k_{t+1} - c_t)] \tag{1}$$

From which we obtain the following First Order Conditions (FOC):

$$\frac{\vartheta L}{\vartheta c_t}; \beta^t\frac{1}{c_t} = \lambda_t \tag{2}$$

$$\frac{\vartheta L}{\vartheta k_{t+1}}; \lambda_{t+1}[(1-\theta)(zh_{t+1})^{\theta}k_{t+1}^{-\theta} + (1-\delta)] = \lambda_t \tag{3}$$

After obtaining the FOC for both of the studied variables, we use the Euler equation for the first derivation and after replacing the equality of λ we achieved the following equation:

$$\frac{1}{\beta}\frac{c_{t+1}}{c_t} = (1-\theta)(h_{t+1}z)^{\theta}k_{t+1}^{-\theta} - \delta + 1 \tag{4}$$

Continuing with the the solution we obtain the value of k taking into consideration that in the SS we equalice all time dependant variables to their respective previous or future values. Meaning that in the case of k we conclude that $k_t = k_{t+1} = k_{t+2}...$, obtaining:

$$\frac{1}{\beta} = (1-\theta)(hz)^{\theta}k^{\theta} - \delta + 1 \tag{5}$$

That in the SS of k_{ss} is equal to:

$$k_{ss} = \left[\frac{\frac{1}{\beta} + \delta - 1}{(1-\theta)(hz)^{\theta}}\right]^{\frac{1}{\theta}} \tag{6}$$

Once we achieve the SS for the k_{ss} we have to normalize the value of $y = 1$ so we can simplify the calculation an use the capital-output ratio of 4, and an investment-output ratio of .25 given in the exercise for ending up with:

$$z = \left(\frac{y}{k^{1-\theta}}\right)^{\frac{1}{\theta}} \frac{1}{h} \tag{7}$$

Finally by plugging in everything back into the k_{ss}, we can obtain the solution of β:

$$\beta = \frac{1}{(1-\theta)(hz)^{\theta}k^{-\theta} - \delta + 1} \tag{8}$$

To complete the analysis we apply all the obtained inforamtion into a python code so we can obtain the estimated values of the previous equation.

The code used is the following:

Listing 1: Insert code directly in your document

```
import math
import numpy as np
from matplotlib import pyplot as plt
from scipy.optimize import fsolve
import pandas as pd


#QUESTION 1

# PART (a)

# Parameters
theta = 0.67
h = 0.31
y_1=1 #normalization assumption

# Given the ratios information in HW2:
i_1 =0.25*y_1
k_1=4*y_1

# Computations to obtain the values of the rest of the variables in the SS
c_1=y_1-i_1
z_1=(y_1/(k_1**(1-theta)))**(1/theta)/h
delta=i_1/k_1
beta=1/((1-theta)*(h*z_1)**theta*k_1**(-theta)-delta+1)

# Output of the first SS:
SS_1={'Variable_name': ['y_1', 'k_1', 'c_1', 'i_1', 'z_1', 'h_1', 'beta', 'theta', 'delta'], 'Values_at
data_SS_1 = pd.DataFrame(SS_1)
print(data_SS_1)
```

Then, after making the mathematical calculation with the help of Python we obtain the following console print of result. We can observe how taking into considerations the parameters given in the exercise we can conclude that the estimations of c_1, i_1, h_1 and β give the subsequent values:

Listing 2: Insert code directly in your document

```
Variable    name        Values at SS_1
0            y_1           1.000000
1            k_1           4.000000
2            c_1           0.750000
3            i_1           0.250000
4            z_1           1.629676
5            h_1           0.310000
6           beta           0.980392
7          theta           0.670000
8          delta           0.062500
```

(1.b) Double permanently the productivity parameter z and solve for the new steady state.

In the following Python console table we obtain both result: i) the values of the SS from the previous exercise with the normal z estimation and ii) the values obtained when we double the value of the z varible. We can observe how a change in the capital output can affect the SS obteined for the growth economy. In the second case, all the values that are derived from z are doubled like in the case of y, k, c and i.

Listing 3: Insert code directly in your document

```
Variable    name        Values at SS_1    Values at SS_2
0              y           1.000000          2.000000
1              k           4.000000          8.000000
2              c           0.750000          1.500000
3              i           0.250000          0.500000
4              z           1.629676          3.259352
5              h           0.310000          0.310000
6           beta           0.980392          0.980392
7          theta           0.670000          0.670000
8          delta           0.062500          0.062500
```

(1.c) Compute the transition from the first to the second steady state and report the time-path for savings, consumption, labor and output.

In order to compute the transition, we necessitate dynamic equations explaining the behaviors of all the variables that change with time (k, c, i, y).

The equation explaining k is derived from the budget constraints:

$$k_{t+1} = k_t^{(1-\theta)}(zh_t)^\theta + (1-\delta)k_t - c_t \tag{9}$$

The equation explaining c is the same as equation (4), with c_{t+1} explicitated.

The equation explaining y is:

$$y_t = k_t^{(1-\theta)}(zh_t)^\theta \tag{10}$$

The equatin explaining i is:

$$i_t = y_t - c_t \tag{11}$$

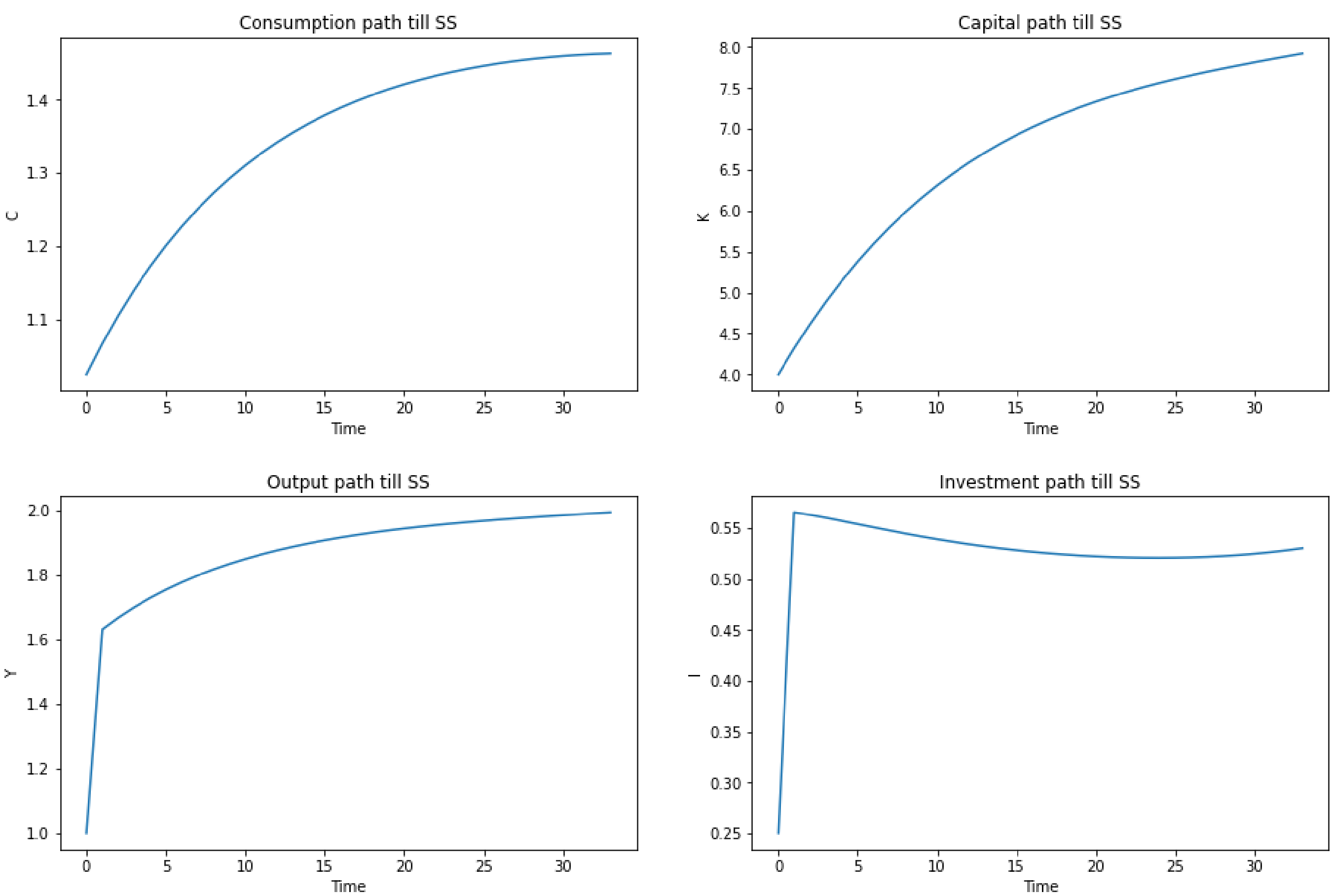
To calculate the values for each period of time, set initial values for all the variables equal to their values at the first SS (as in Listing 1). The only variable that directly reacts when z doubles is consumption, which is set to $c = 1.024$.
Please note that this initial guess reflects many trials. The logic behind consumption's initial value is that by letting $c = c_1$ as for the other variables, we would have reached the SS in less than $t = 10$, so we wouldn't have been able to go on with the exercise.
Then, implement the loop computing the transition paths till capital reaches the SS k_2.

The python code is the following:

```
# PART (c) #————————————————————————————————————————

# Values of the variables at the initial time (when shock hits):
c = 1.024
t = 0
c_t = []
c_t.append(c)
i_t = []
i_t.append(i_1)
y_t = []
y_t.append(y_1)
k_t = []
k_t.append(k_1)

# Computing the transitioni loop until k fids its SS (k=k_2):
while k_2 - k_t[t] > 0.1:
    k_t.append(k_t[t]**(1-theta)*(z_2*h)**theta + (1-delta)*k_t[t]- c_t[t])
    c_t.append(((1-theta)*(h*z_2)**theta*k_t[t+1]**(-theta) + (1 - delta))*c_t[t]*beta)
    y_t.append(k_t[t+1]**(1-theta) * (z_2*h)**theta)
    i_t.append(y_t[t+1] - c_t[t+1])
    t += 1

plt.figure()
```

```
plt.subplot(221)
plt.plot(c_t)
plt.title('Consumption path till SS')
plt.ylabel('C')
plt.xlabel('Time')
plt.subplot(222)
plt.plot(k_t)
plt.title('Capital path till SS')
plt.ylabel('K')
plt.xlabel('Time')
plt.subplot(223)
plt.plot(y_t)
plt.title('Output path till SS')
plt.ylabel('Y')
plt.xlabel('Time')
plt.subplot(224)
plt.plot(i_t)
plt.title('Investment path till SS')
plt.ylabel('I')
plt.xlabel('Time')
plt.subplots_adjust(top=2, bottom=0.08, left=0, right=2, hspace=0.3, wspace=0.2)
plt.show()
```

Outputs are showed in the following graphs:



As the graphs show, the SS is reached after 33 periods.

Capital and consumption have a relatively similar transition path, but consumption converges faster. Output and investment have both a rapid increase in the period right after the initial shock. Investment jumps right above its SS value, while output doesn't initially reach y_2, but firstly reaches 1.65 and then converges slowly.

(1.d) Unexpected shocks. Let the agents believe productivity z_t doubles once and for all periods.
However, after 10 periods, surprise the economy by cutting the productivity z_t back to its original value. Compute the transition for savings, consumption, labor and output.

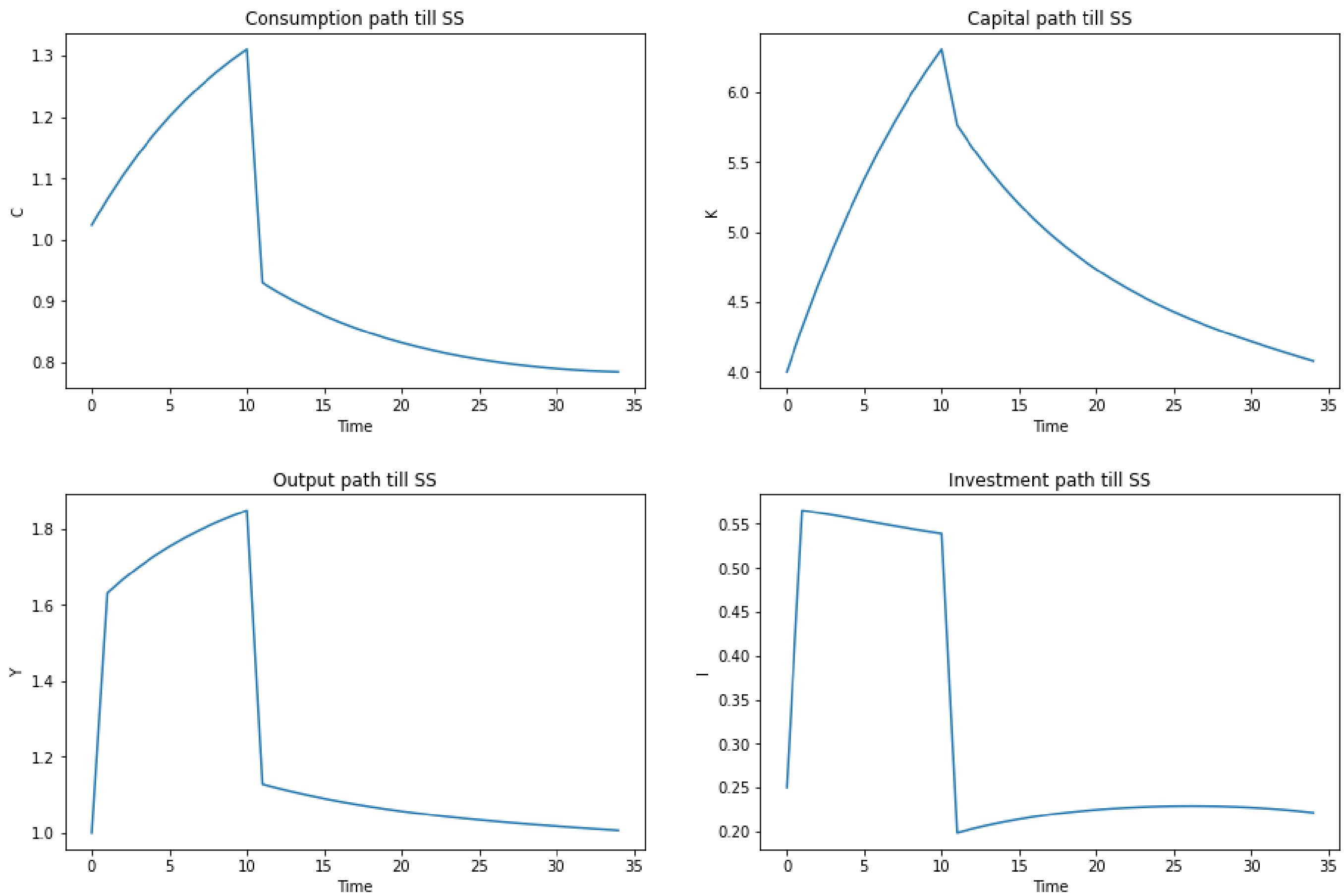
In order to compute the transition paths with such an unexpected shock, the same loop as in section (1.c) has to be implemented. The difference is that since the unexpected shock happens in $t = 10$, two loops have to be created:
The first one is exactly the same as before, but stopping at $t = 9$ included.
The second one defines the shock at $t = 10$, and then computes the transition paths of the variables back to the first SS $(k = k_1)$.

The Python code used is the following:

```
# PART (d) #-------------------------------------------------------------------

# Values of the variables at the initial time (when first shock hits):
c = 1.024
t = 0
c_t = []
c_t.append(c)
i_t = []
i_t.append(i_1)
y_t = []
y_t.append(y_1)
k_t = []
k_t.append(k_1)

# Two loops:
    # The fisrt one explaining how the economy behaves when not expecting
        #another shock;
    # The second one explainig how the economy behaves when in period t=10 the
        #new unexpected shock hits.

while (k_2 - k_t[t] > 0.1) and (t < 10):
    k_t.append(k_t[t]**(1-theta)*(z_2*h)**theta + (1-delta)*k_t[t]- c_t[t])
    c_t.append(((1-theta)*(h*z_2)**theta*k_t[t+1]**(-theta) + (1 - delta))*c_t[t]*beta)
    y_t.append(k_t[t+1]**(1-theta) * (z_2*h)**theta)
    i_t.append(y_t[t+1] - c_t[t+1])
    t += 1
while (k_t[t] - k_1 > 0.1) and (t > 9):
    if t == 10:
        c_t.append(0.93) # ?????????????????????????????????????????????????
        k_t.append(k_t[t]**(1-theta)*(z_1*h)**theta + (1-delta)*k_t[t]- c_t[t])
```

```
            y_t.append(k_t[t+1]**(1-theta) * (z_1*h)**theta)
            i_t.append(y_t[t+1] - c_t[t+1])
            t += 1
        else:
            k_t.append(k_t[t]**(1-theta)*(z_1*h)**theta + (1-delta)*k_t[t]- c_t[t])
            c_t.append(((1-theta)*(h*z_1)**theta*k_t[t+1]**(-theta) + (1 - delta))*c_t[t]*beta)
            y_t.append(k_t[t+1]**(1-theta) * (z_1*h)**theta)
            i_t.append(y_t[t+1] - c_t[t+1])
            t += 1

# Plot time paths:
plt.figure()
plt.subplot(221)
plt.plot(c_t)
plt.title('Consumption path till SS')
plt.ylabel('C')
plt.xlabel('Time')
plt.subplot(222)
plt.plot(k_t)
plt.title('Capital path till SS')
plt.ylabel('K')
plt.xlabel('Time')
plt.subplot(223)
plt.plot(y_t)
plt.title('Output path till SS')
plt.ylabel('Y')
plt.xlabel('Time')
plt.subplot(224)
plt.plot(i_t)
plt.title('Investment path till SS')
plt.ylabel('I')
plt.xlabel('Time')
plt.subplots_adjust(top=2, bottom=0.08, left=0, right=2, hspace=0.3, wspace=0.2)
plt.show()
```

The outputs are showed in the following graphs:



The economy reaches its SS after 34 periods.

The graphs visually show what was expected: the economy was firstly behaving as in point (1.c), because the shock at $t = 10$ is unexpected. Then, when the unexpected shock hits, it converges back to the initial SS (as if the initial jump in z never happened).

Question 2: Solve the optimal COVID-19 lockdown model posed in the slides.

(2.a) Show your results for a continuum of combinations of the β in [0; 1] parameter (vertical axis) and the c(TW) in [0; 1] parameter (hztal axis). That is, plot for each pair of β and c(TW) the optimal allocations of H, Hf , Hnf , Hf/H, output, welfare, amount of infections and deaths. Note that if H = N there is no lockdown, so pay attention to the potential non-binding constraint H ¡ N. Discuss your results.
You may want to use the following parameters: Af = Anf = 1; $\rho = 1:1$, $k_f = k_{nf} = 0:2$, $\omega = 20$,$\gamma = 0.9$, $i_0 = 0.2$ **and N = 1.**

Define all the parameters of the model:

```
#Importing packages
import pandas as pd
import numpy as np
import mpmath as mp
import sympy
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from scipy.optimize import LinearConstraint, minimize, fsolve
import seaborn as sb

#Parameters
A=1
rho=1.1
k=0.2
w=20
gamma=0.9
i_0 =0.2
N=1
```

Create the necessary grids in order to run the maximization:

```
x_points, y_points= np.meshgrid(np.arange(0,101,1), np.arange(0,101,1), indexing="xy")
x_points = x_points*0.01
y_points = y_points*0.01
grid = np.array([x_points, y_points])
grid_2 = grid[0]

results = grid.copy()
output = grid_2.copy()
wel = grid_2.copy()
infec = grid_2.copy()
det = grid_2.copy()
```

Compute the maximization for H_f and H_{nf} for each c(TW) and β(HC). To do so, we used the command "minimize" from the package "scipy.minimize". It requires the function to minimize and an initial point (as main inputs). The function to minimize is the "total welfare function" times -1. The initial guess has been set to $x_0 = (0.5, 0.5)$.
The code is the following:

```
for z in np.arange(1,101,1):
    for y in np.arange(0,101,1):
        c_tw, B = grid[:, z, y]
        Y = lambda H_f, H_nf:(A * (H_f**((rho-1)/rho))+c_tw*A*(H_nf**((rho-1)/rho)))**(rho/(rho-1))
        I = lambda H_f: B*((i_0*H_f)/N)*H_f
        D = lambda H_f: (1-gamma)*I(H_f)

        def solution_func(x):
            H_f = x[0]
            H_nf = x[1]
            return -1*(Y(H_f, H_nf)- k * H_f - k * H_nf - w * D(H_f))

        #1st guess
        x0 = np.array([0.5, 0.5])

        #Such that (s.t.)
        g_1 = np.array([-1, -1])
        g_2 = np.array([-N])
        s_t = [(0, 1), (0, 1)]

        #Solving
        const={"type" : "ineq", "fun" : lambda x: g_1 @ x - g_2}

        solution = minimize(solution_func, x0, method="SLSQP", bounds=s_t, constraints=const)
        results[:, z, y] = solution.x
```

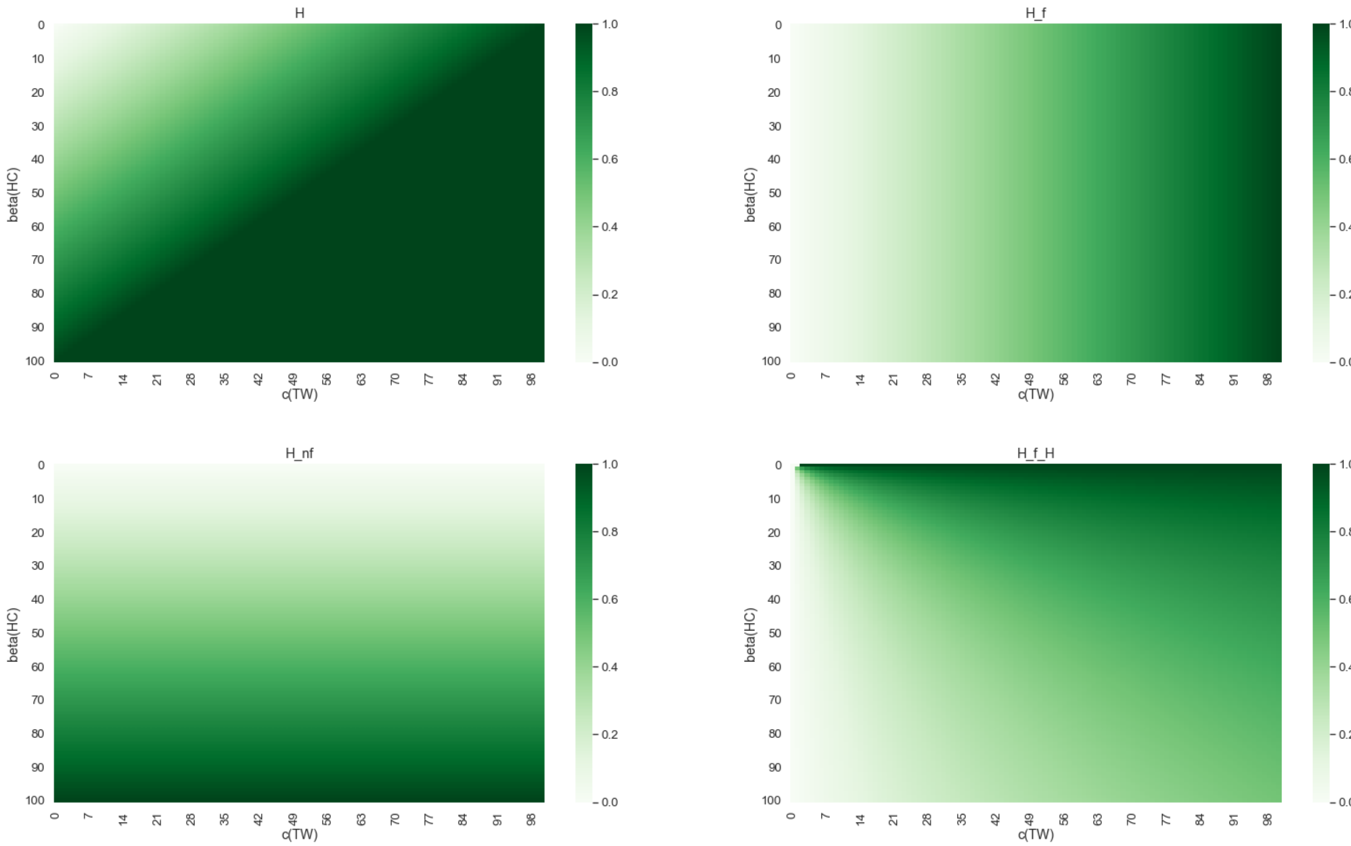
In order to plot the outcomes, we used the following Python code:

```
        #In order to plot the required figures
        output[z, y] = (A * (results[0, z, y] * ((rho - 1) / rho)) + c_tw * A * (results[1, z, y] * ((r
        infec[z, y] = B * ((i_0*results[0, z, y]) / N) * results[0, z, y]
        det[z, y] = (1 - gamma) * infec[z, y]
        wel[z, y] = output[z, y] - k * results[0, z, y] - k * results[1, z, y] - w * det[z, y]

        H_f = results[0]
        H_nf = results[1]

        H = np.array(H_f + H_nf)
        H_f_H = np.array(H_f/H)

        #Ploting the results
        plt.figure(figsize=(10,6))
        plt.subplot(221)
        sb.set(font_scale=1.2)
        sb.heatmap(H,cmap="Greens"   , cbar_kws={'label': ''}, vmin=0, vmax=1)
        plt.xlabel('c(TW)')
        plt.ylabel('beta(HC)')
        plt.title('Optimal allocation of H')
```

```
plt.subplot(222)
sb.set(font_scale=1.2)
sb.heatmap(H_f,cmap="Greens"    , cbar_kws={'label': ''}, vmin=0, vmax=1)
plt.xlabel('c(TW)')
plt.ylabel('beta(HC)')
plt.title('Optimal allocation of H_f')

plt.subplot(223)
sb.set(font_scale=1.2)
sb.heatmap(H_nf,cmap="Greens"    , cbar_kws={'label': ''}, vmin=0, vmax=1)
plt.xlabel('c(TW)')
plt.ylabel('beta(HC)')
plt.title('Optimal allocation of H_nf')

plt.subplot(224)
sb.set(font_scale=1.2)
sb.heatmap(H_f_H,cmap="Greens"    , cbar_kws={'label': ''}, vmin=0, vmax=1)
plt.xlabel('c(TW)')
plt.ylabel('beta(HC)')
plt.title('Optimal allocation of H_f_H')

plt.subplots_adjust(top=2, bottom=0.08, left=0, right=2, hspace=0.3, wspace=0.2)
plt.show()
```

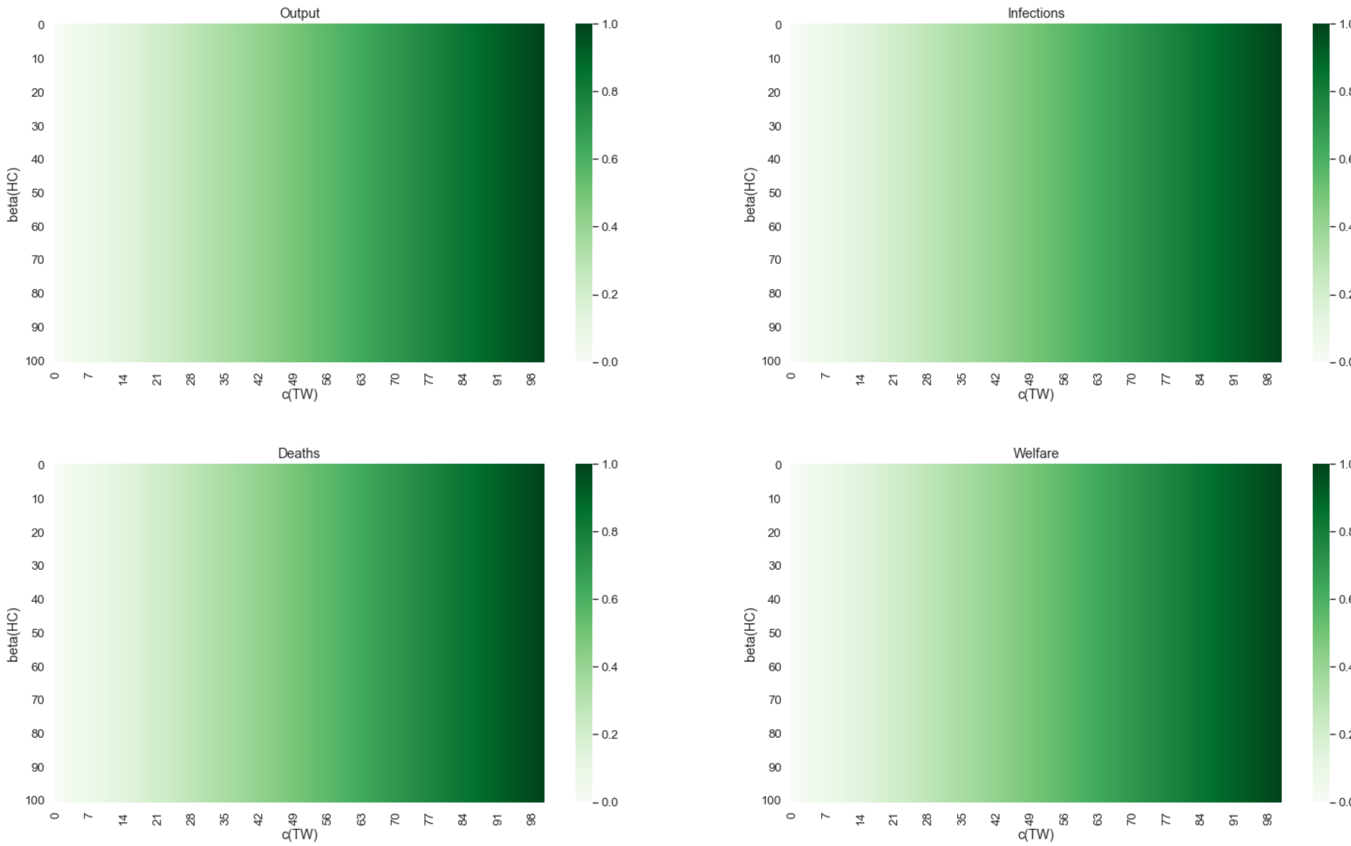
Outputs are shown in the following two sets of graphs.

The first set of graphs show the optimal allocations of H, Hf , Hnf and Hf/H for any possible pair of β and c(TW):



The outcomes for H_{nf} and H_f/H are in line with the theory behind them.
In particular, we see that H_{nf} is at the highest for higher values of β(HC). This reflects the effect of a high degree of HC (human contact) that is present at the workplace. When this variable is at its highest, it is logical to see Hours Teleworking increase, since the social planner tries to limit the spread of the virus. Following the same logic, when β(HC) is at its lowest, we see H_f/H increase, reflecting the decision of sending more people to the workplace if HC is not relevant.

The second set of graphs show the optimal allocations of output, welfare, amount of infections and deaths for any possible pair of β and c(TW):



The outcomes showed for output, welfare, amount of infections and deaths are not reflecting theory. The main two reasons are that they all increase/decrease in the same exact way with c(TW), while they do not respond to changes in β(HC).