In [1]:

```
import pandas as pd
import math
import numpy as np
import matplotlib.pyplot as plt
from GetModelParameters import *
from scipy.integrate import solve_ivp
from geneticalgorithm import geneticalgorithm as ga
```

In [2]:

```
def thetaModel(t, y, N, gammas, p):
S, E, I, I_u, Hr, Hd = y
actual = math.floor(t)
gamma_E, gamma_I, gamma_Iu, gamma_Hr, gamma_Hd, gamma_Q = gammas.__fields__()
omega, tau1, tau2, omega_u = p[actual,0], p[actual,1], p[actual,2], p[actual,3]
theta, rho = p[actual,4], p[actual,5]
beta_e, beta_I, beta_Iu, beta_hr, beta_hd = p[actual,6], p[actual,7], p[actual,8], p[ac
betas_mul = beta_e*E + beta_I*I + beta_Iu*I_u + beta_hr*Hr + beta_hd*Hd
newE = E/gamma_E
newI = I/gamma_I
newIu = I_u/gamma_Iu
newHr = Hr/gamma_Hr
newHd = Hd/gamma_Hd
dSdt = -(S/N) * betas_mul
dEdt = (S/N) * betas_mul - newE + tau1 - tau2
dIdt = newE - newI
dIudt = (1 - theta - omega_u) * newI - newIu
dHrdt = rho*(theta - omega) * newI - newHr
dHddt = omega * newI - newHd
return dSdt, dEdt, dIdt, dIudt, dHrdt, dHddt
```

In [3]:

```
class OptimizeModel:
def __init__(self, path, t0, tMAX, lambdas, ms_val):
self.data = Data(path)
self.tspan = np.arange(t0, tMAX)
self.dates = get_lambdas(lambdas, self.data)
self.N = self.data.population[t0]
self.u0 = [
self.data.susceptible[t0],
self.data.exposed[t0],
self.data.infec[t0],
self.data.infec_u[t0],
self.data.hospitalized[t0],
self.data.hospitalized[t0]*0.3,
]
self.ms_val = ms_val
def distance(self, X):
gamma_E, gamma_I, gamma_Iu, gamma_Hr, gamma_Hd, gamma_Q, c3, c5, omega_u0, max_omega, m
beta_I0_min, beta_e0 = c_E * beta_I0, c_u * beta_I0
gammas = Gammas(gamma_E, gamma_I, gamma_Iu, gamma_Hr, gamma_Hd, gamma_Q)
delays = Delays(gammas)
times = Times(self.tspan, self.data, delays)
ms = get_Ms(
self.tspan[0], self.dates, self.ms_val,
np.array([k2]),
np.array([c3, c5])
)
p = parameters_list(times, self.data, gammas, delays, ms, self.dates, max_omega, min_om
```

```
sol = solve_ivp(thetaModel, self.tspan, self.u0, args=(self.N, gammas, p), dense_output
u = sol.sol(self.tspan)
distance = math.sqrt(np.sum(list(
map(
lambda x: x**2,
u.T[:,2] - self.data.infec
)
)))
#print(distance)
return distance
def genetic_alg(self, bounds, params, timeOut=10.0):
model = ga(
function=self.distance,
dimension=bounds.shape[0],
variable_type='real',
variable_boundaries=bounds,
algorithm_parameters=params,
function_timeout=timeOut
)
return model
```

In [4]:
```
from ThetaModel import *
```

In [5]:
```
W = 15
H = 10
```

In [6]:
```
path = "D:\\Code\\[Servicio Social]\\Datos\\Casos_Modelo_Theta_v3.csv"
data = Data(path)
```

In [7]:
```
lambdas = [
"2020-03-20",
"2020-03-23",
"2020-03-30",
"2020-04-21",
"2020-06-01"
]
dates = get_lambdas(lambdas, data)
```

In [8]:
```
tspan = np.arange(0, 446)
```

In [9]:
```
N = data.population[tspan[0]]
u0 = [
data.susceptible[tspan[0]],
data.exposed[tspan[0]],
data.infec[tspan[0]],
data.infec_u[tspan[0]],
data.hospitalized[tspan[0]],
data.hospitalized[tspan[0]]*0.3,
]
ms_val = np.array([1.0,1.0, 0.0, 0.0, 0.0, 0.0])
```

In [10]:

```
gammas = Gammas(5.5, 5.0, 9.0, 14.2729, 5.0, 36.0450)
beta_I0, c_E, c_u, rho0, omega_u0, k2, c3, c5 = 0.4992, 0.3806, 0.3293, 0.7382, 0.42, 0
beta_I0_min, beta_e0 = c_E * beta_I0, c_u * beta_I0
max_omega, min_omega = 0.804699241804244, 0.12785018214405364
#region
delays = Delays(gammas)
times = Times(tspan, data, delays)
ms = get_Ms(
tspan[0], dates, ms_val,
np.array([k2]),
np.array([c3, c5])
)
```

In [11]:

```
p = parameters_list(times, data, gammas, delays, ms, dates, max_omega, min_omega, omega
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
d:\Code\[Servicio Social]\Python\ThetaModel\main.py in <module>
----> 1 p = parameters_list(times, data, gammas, delays, ms, dates, max_omega, min_omeg
a, omega_u0, rho0, beta_I0, beta_I0_min, beta_e0)

TypeError: parameters_list() takes 12 positional arguments but 13 were given
```

In [12]:

```
p = parameters_list(times, data, gammas, delays, ms, dates, omega_u0, rho0, beta_I0, be
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
d:\Code\[Servicio Social]\Python\ThetaModel\main.py in <module>
----> 1 p = parameters_list(times, data, gammas, delays, ms, dates, omega_u0, rho0, bet
a_I0, beta_I0_min, beta_e0)

TypeError: parameters_list() missing 1 required positional argument: 'beta_e0'
```

In [13]:

```
gammas = Gammas(5.5, 5.0, 9.0, 14.2729, 5.0, 36.0450)
beta_I0, c_E, c_u, rho0, omega_u0, k2, c3, c5 = 0.4992, 0.3806, 0.3293, 0.7382, 0.42, 0
beta_I0_min, beta_e0 = c_E * beta_I0, c_u * beta_I0
omega = 0.014555
max_omega, min_omega = 0.804699241804244, 0.12785018214405364
#region
delays = Delays(gammas)
times = Times(tspan, data, delays)
ms = get_Ms(
tspan[0], dates, ms_val,
np.array([k2]),
np.array([c3, c5])
)
```

In [14]:

```
p = parameters_list(times, data, gammas, delays, ms, dates, omega, omega_u0, rho0, beta
```

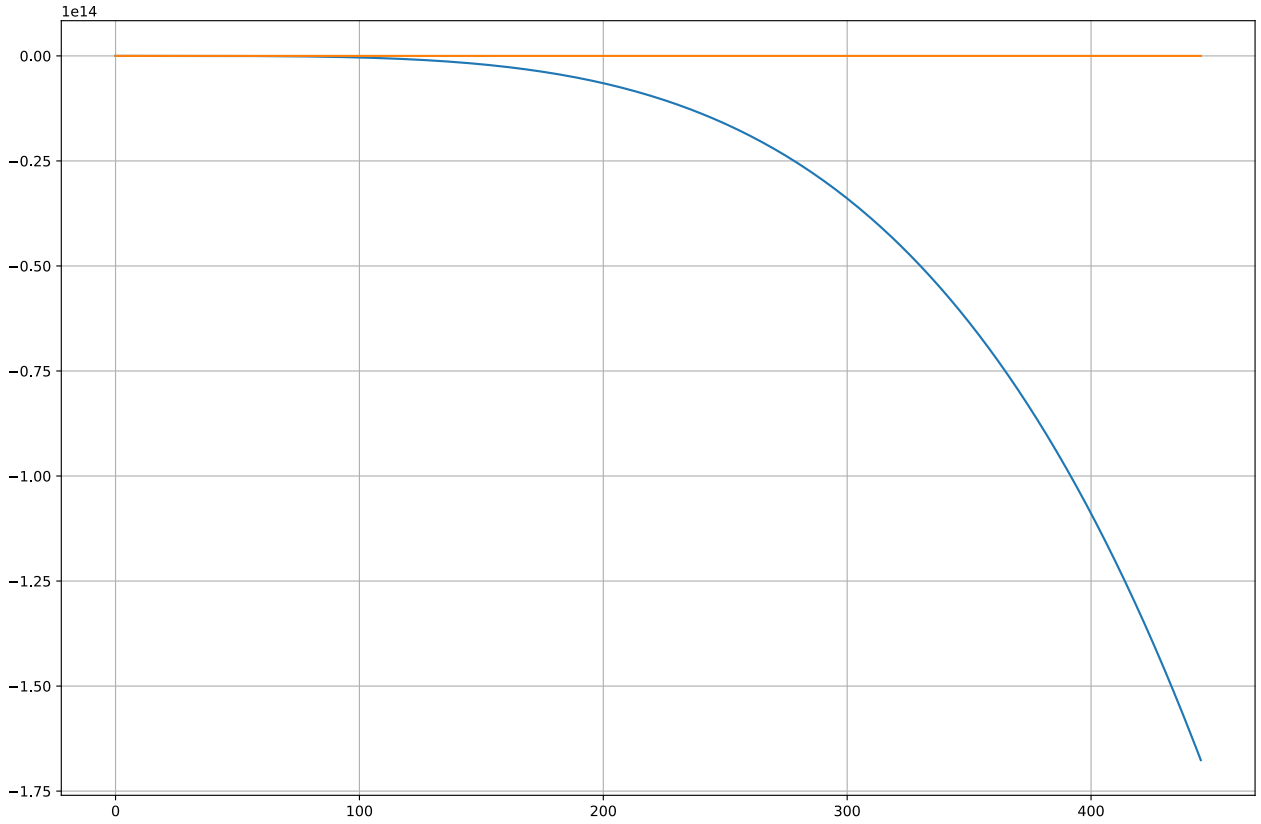
In [15]:

```
sol = solve_ivp(thetaModel, tspan, u0, args=(N, gammas, p), dense_output=True)
u = sol.sol(tspan)
```

In [16]:

```
fig = plt.figure()
fig.set_figwidth(W)
```

```
fig.set_figheight(H)
plt.plot(tspan, u.T[:,2])
plt.plot(tspan, data.infec)
plt.grid()
plt.show()
#endregion
```



In [17]:
```
bounds_gammas = np.array([[0, 200]] * 6)
bounds_01 = np.array([[0, 1]] * 7)
bounds_0inf = np.array([[0, 500]] * 3)
bounds = np.vstack((bounds_gammas, bounds_01, bounds_0inf))
```

In [18]:
```
parameters= {
'max_num_iteration': 1000,
'population_size':100,
'mutation_probability':0.3,
'elit_ratio': 0.01,
'crossover_probability': 0.5,
'parents_portion': 0.3,
'crossover_type':'uniform',
'max_iteration_without_improv':200
}
```

In [19]:
```
model = OptimizeModel(path, 0, 446, lambdas, ms_val)
```

In [20]:
```
genetic_opt = model.genetic_alg(bounds, parameters, 100.0)
```

In [21]:

```
genetic_opt.run()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
d:\Code\[Servicio Social]\Python\ThetaModel\main.py in <module>
----> 1 genetic_opt.run()

~\miniconda3\lib\site-packages\geneticalgorithm\geneticalgorithm.py in run(self)
    301
    302
--> 303             obj=self.sim(var)
    304             solo[self.dim]=obj
    305             pop[p]=solo.copy()

~\miniconda3\lib\site-packages\geneticalgorithm\geneticalgorithm.py in sim(self, X)
    540         obj=None
    541         try:
--> 542             obj=func_timeout(self.funtimeout,self.evaluate)
    543         except FunctionTimedOut:
    544             print("given function is not applicable")

~\miniconda3\lib\site-packages\func_timeout\dafunc.py in func_timeout(timeout, func, arg
s, kwargs)
    106
    107     if exception:
--> 108         raise_exception(exception)
    109
    110     if ret:

~\miniconda3\lib\site-packages\func_timeout\py3_raise.py in raise_exception(exception)
      5 # Only available in python3.3+
      6 def raise_exception(exception):
----> 7     raise exception[0] from None

~\miniconda3\lib\site-packages\geneticalgorithm\geneticalgorithm.py in evaluate(self)
    534 ###############################################################################
    535     def evaluate(self):
--> 536         return self.f(self.temp)
    537 ###############################################################################
    538     def sim(self,X):

d:\Code\[Servicio Social]\Python\ThetaModel\ThetaModel.py in distance(self, X)
     81         )
     82
---> 83         p = parameters_list(times, self.data, gammas, delays, ms, self.dates, m
ax_omega, min_omega, omega_u0, rho0, beta_I0, beta_I0_min, beta_e0)
     84
     85         sol = solve_ivp(thetaModel, self.tspan, self.u0, args=(self.N, gammas,
 p), dense_output=True)

TypeError: parameters_list() takes 12 positional arguments but 13 were given
```