SECB 4313-01

BIOINFORMATICS MODELING AND SIMULATION

ASSIGNMENT 1: DISEASE MODELING

Lecturer:

Dr. Azurah bte A Samah

Group Members:

Chong Kah Wei (A20EC0027)

Heong Yi Qing (A20EC0043)

Semester 2 2023/2024

a) Description of the simulation model (Introduction and objective of model)

Disease modeling serves as a crucial tool that provides a better understanding of cellular dynamics and interaction between the cells and tumors. By simulating infection spread and tumor growth computationally, researchers can explore various treatments and conditions without the need for real-life experiments. This approach is particularly effective for studying complex cellular systems, including the immune cells and cancer. In this context, the given example is a Python Flask web application that models the immune response to cancer using ODEs. The application allows the changing of different parameters to observe the impacts on the predefined key variables through a multiple-line graph. Generally, the main objective of this disease modeling is to simulate the cell behaviors using ODEs to enhance the therapies and treatment plans for cancer.

b) Flow of the simulation model (how the codes are constructed)

The simulation model provides a dynamic representation of cancer progression by employing a system of equations to monitor the five key variables: the concentrations of cancer cells (C), healthy cells (H), interleukins (IL), tumor cells (T), and the effect of treatment (S). In this context, the model initializes these variables with specific initial values and these initial values serve as a baseline, indicating the initial status of a cancer patient. Throughout the simulation, these initial values remain fixed, while different parameters are employed to observe their effects on the variables. By altering parameters such as growth rates, death rates, treatment effectiveness, and environmental factors, the simulation provides insights into how these variables may change over time and how various interventions may impact the progression of cancer. The interaction of the disease model components is as illustrated in Figure 1 for a better understanding of the dynamics of cancer progression and therapy.

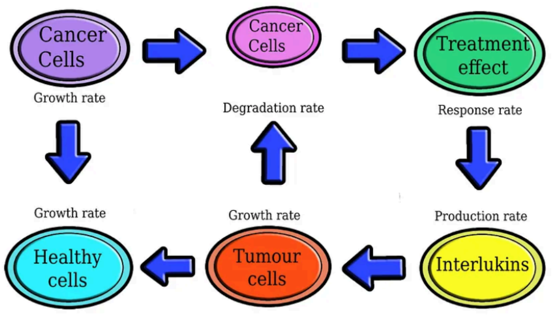


Figure 1 Potential Interaction Map of the Equations

In the code implementation, the simulation model initializes the key variables with C = 50, H = 10, IL = 0, T = 1000, and S = 0. The default value of the parameters, including rC, dC, rH, kIL, kCT, s and K are defined within the code, but the system also allows the manipulation of these parameters from the web interface to observe their effects on key variables over time. Using the provided equations as shown in (c), the simulation model will calculate the rate of changes for each variable based on the input parameters and return the value of dCdt, dHdt, dILdt, dTdt and dSdt. Once the calculations are complete, the system will generate a multi-line graph to visualize the rate of change in the key variables using Matplotlib and display in the web interface.

c) List of mathematical equations used and its descriptions.

The simulation model includes five equations to calculate the rate of changes of the key variables C, H, IL, T and S. The equations and respective descriptions are as shown in Table 1.

Table 1 Equation and Description

Equation	Description
dCdt = rC * C * (1 - (T/K)) * (1 - S) - dC * C	dCdt represents the change in the concentration of cancer cells over time and is dependent on the rate of growth of cancer cells (rC), the competition for resources with other cells (1-(T/K)), the effect of treatment (1-S), and the death rate of cancer cells (dC).
dHdt = rH * H	dHdt represent the changes in the concentration of healthy cells over time, and are dependent on the growth rate of healthy cells (rH)
dILdt = kIL * H	dILdt represent the changes in the concentration of interleukins over time, and are dependent on the production rate of interleukins (kIL)
dTdt = -kCT * C * T	dTdt represents the change in the concentration of tumor cells over time and is dependent on the competition for resources with other cells, the death rate of tumor cells due to treatment (kCT), and the concentration of tumor cells themselves (T).
dSdt = s * T	dSdt represents the change in the effect of treatment over time and is dependent on the concentration of tumor cells (T) and the effectiveness of the treatment (s)

d) Python libraries used.

The first library used is the Flask library, which is often used in disease modeling for building web applications that enable dynamic interactions with disease models. Flask is used to create web interfaces that allow users to input parameters and run simulations of disease models. Users can modify parameters such as growth rates, death rates, treatment effectiveness, and environmental factors in this simulation model. It also serves as the backend for interactive dashboards that display the complex data generated by the disease modeling which can be hard to interpret without visualization.

Next NumPy library has been imported into disease modeling. NumPy is a fundamental package for numerical computing in Python. It is essential for efficiently handling and processing large datasets, performing numerical computations, and facilitating complex mathematical operations, which are common tasks in disease modeling. For instance, the NumPy library function, numpy.linspace is used to return evenly spaced numbers over a specified interval.

Moreover, Matplotlib.pyplot is a versatile plotting library in Python that is widely used in disease modeling to visualize data and model outcomes. When used in conjunction with Flask, matplotlib.pyplot enables the development of Flask applications that can generate and display static, dynamic, or interactive charts and graphs directly in the web browser. By embedding plots generated by matplotlib.pyplot into a Flask web application, users can view updates in real-time, interact with the data through parameters adjustment, and better understand the effects of different scenarios in disease modeling.

Lastly, odeint from scipy.integrate is extensively utilized for solving the ordinary differential equations (ODEs).In the realm of epidemiological studies, ODEs are used to describe how diseases spread over time, accounting for various factors such as transmission rates, recovery rates, and immunity. The odeint function provides a robust and efficient numerical method for integrating these equations, allowing researchers to simulate disease dynamics under different scenarios and parameters.

e) Input of simulation model

The first input of the simulation model is the list of initial values, y, which includes C, H, IL, T and S. Other than the initial values, the time, t is also the input of the simulation model. Followed by the time, the model parameters are the other inputs of the model, which are rC, dC, rH, kIL, kCT, s and K.

f) Model parameters.

There are a total of 7 model parameters which are rC, dC, rH, klL, kCT, s and K. The description of each parameter is in Table 2.

Table 2 Parameters and Description

Parameters	Description
rC	Rate of growth of cancer cells
dC	Death rate of cancer cells
rH	Growth rate of healthy cells
klL	Production rate of Interleukins

kCT	Death rate of tumor cells due to treatment
s	Effectiveness of treatment
K	Carrying capacity of the environment

g) Description of simulation output and generated graph.

Using the default parameters, where rC = 0.1, dC = 0.05, rH = 0.05, kIL = 0.1, kCT = 0.01, s = 0.01 and K = 1000, the simulation output is presented as a multi-line graph as shown in Figure 2.

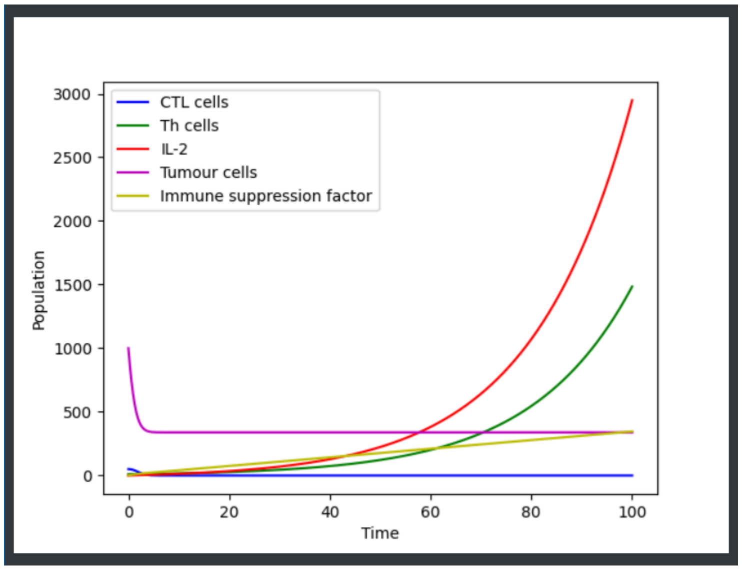


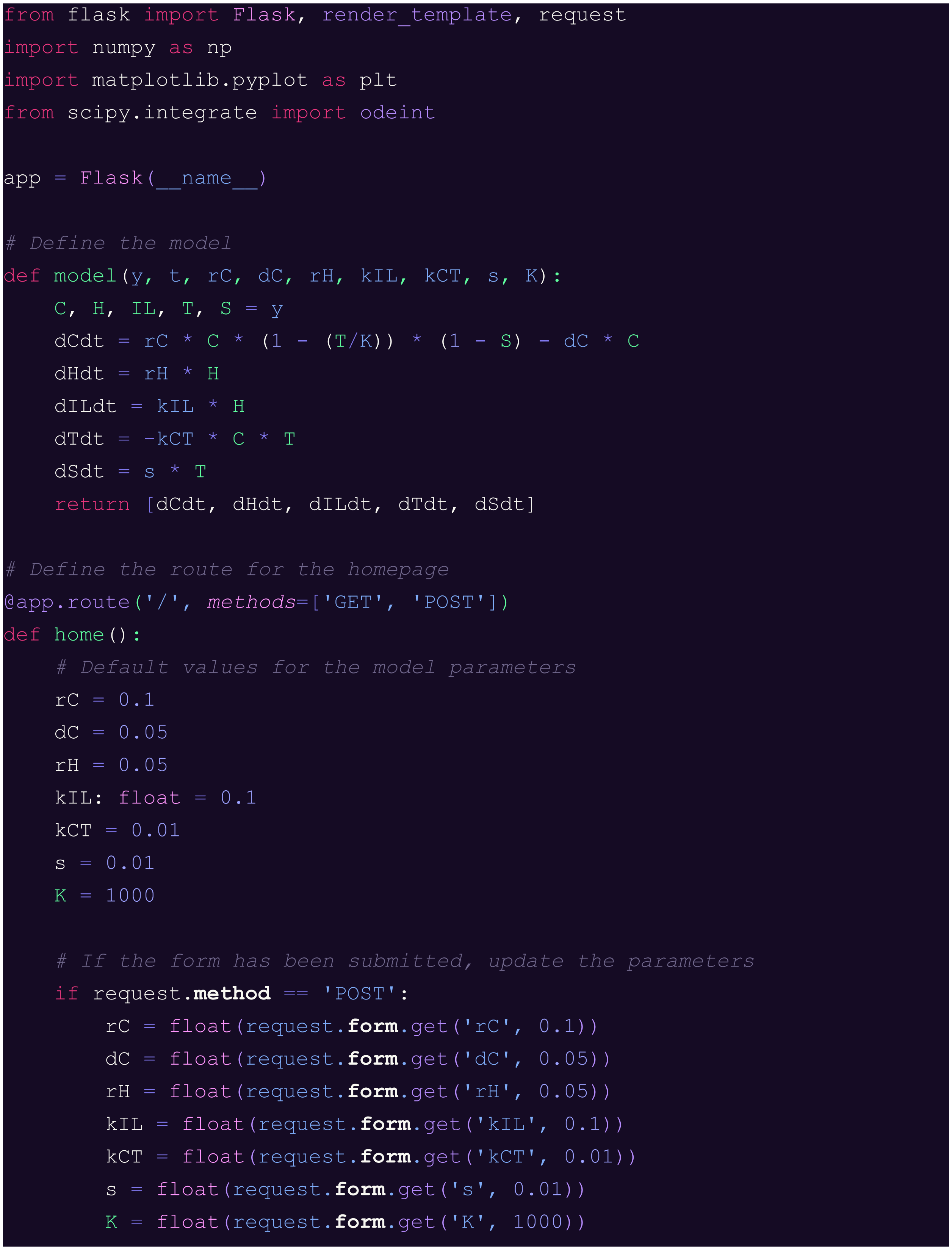
Figure 2 Simulation Output

The generated multi-line graph shows the changes in population of CTL cells, Th cells, IL-2, tumor cells as well as the immune suppression factors over the time. Based on the generated graph, the CTL cells decrease initially and then stabilize over the time. Meanwhile, Th cells and IL-2 show exponential growth, with Th cells surpassing IL-2. Tumor cells undergo a significant initial decrease followed by stability. Conversely, the immune suppression factor exhibits a linear increase.

h) Experimentation that can be carried out using the simulation model.

Cellular interaction analysis can be carried out to explore how tumor cells interact with surrounding healthy cells, including immune cells. This can help in understanding the mechanisms of metastasis, immune evasion and the impact of the immune system on tumor growth and response to therapies. Other than that, drug resistance development also can be done to simulate how and when the tumor cells develop resistance to various treatments. This can help in developing strategies to prevent resistance, such as drug cycling or combination therapies that target multiple pathways simultaneously.

APPENDIX



```python
from flask import Flask, render_template, request
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint

app = Flask(__name__)

# Define the model
def model(y, t, rC, dC, rH, kIL, kCT, s, K):
    C, H, IL, T, S = y
    dCdt = rC * C * (1 - (T/K)) * (1 - S) - dC * C
    dHdt = rH * H
    dILdt = kIL * H
    dTdt = -kCT * C * T
    dSdt = s * T
    return [dCdt, dHdt, dILdt, dTdt, dSdt]

# Define the route for the homepage
@app.route('/', methods=['GET', 'POST'])
def home():
    # Default values for the model parameters
    rC = 0.1
    dC = 0.05
    rH = 0.05
    kIL: float = 0.1
    kCT = 0.01
    s = 0.01
    K = 1000

    # If the form has been submitted, update the parameters
    if request.method == 'POST':
        rC = float(request.form.get('rC', 0.1))
        dC = float(request.form.get('dC', 0.05))
        rH = float(request.form.get('rH', 0.05))
        kIL = float(request.form.get('kIL', 0.1))
        kCT = float(request.form.get('kCT', 0.01))
        s = float(request.form.get('s', 0.01))
        K = float(request.form.get('K', 1000))
    # Solve the model
    t = np.linspace(0, 100, 1000)
    y0 = [50, 10, 0, 1000, 0]
    sol = odeint(model, y0, t, args=(rC, dC, rH, kIL, kCT, s, K))

    # Plot the results
    fig, ax = plt.subplots()
    ax.plot(t, sol[:,0], 'b', label='CTL cells')
    ax.plot(t, sol[:,1], 'g', label='Th cells')
    ax.plot(t, sol[:,2], 'r', label='IL-2')
    ax.plot(t, sol[:,3], 'm', label='Tumour cells')
    ax.plot(t, sol[:,4], 'y', label='Immune suppression factor')
    ax.set_xlabel('Time')
    ax.set_ylabel('Population')
    ax.legend()
    plt.savefig('static/plot.png')

    # Render the homepage template with the current parameters and the graph
    return render_template('index.html', rC=rC, dC=dC, rH=rH, kIL=kIL, kCT=kCT, s=s, K=K)

# Define the route for the results page
@app.route('/results')
def results():
    # Render the results template with the graph
    return render_template('results.html')

if __name__ == '__main__':
    app.run(debug=True)
```