SECB4313 – BIOINFORMATICS MODELING AND SIMULATION

SECTION 01

DR. AZURAH BINTI A SAMAD

ASSIGNMENT 1

NAME	MATRIC NO
KESHINIY A/P MOGAN	A20EC0057
PHANG CHENG YI	A20EC0131

1. Description of the simulation model (Introduction and objective of model)

Introduction:

By protecting the body from infections, including cancer cells, the immune system is essential to preserving health. Effective immunotherapies for the treatment of cancer require a thorough understanding of the dynamics of interactions between immune cells, tumour cells, and signalling molecules. Researchers can investigate a variety of scenarios and hypotheses by using computational models, which are an invaluable tool for examining these intricate biological processes in a controlled setting. If enough information is available about the dynamics of the system under study to enable a high-fidelity simulation and a mapping of the system's behaviour to the output to be expected, then simulation is the recommended approach.

Objectives:

Investigating the dynamics of interactions between immune cells, tumour cells, and signalling chemicals inside a tumour microenvironment is the aim of the simulation model. The model's objective is to simulate these components' behaviour throughout time to:

- Examine Immune Cell-Cancer-Cell Interactions: Examine the intricate relationships that exist between cancer cells and immune cells (such as helper T cells and cytotoxic T lymphocytes) in the context of the tumour microenvironment.
- Analyse the impact of treatment on the growth of tumours: Examine the effects of different treatments on the dynamics of tumour development, including immunotherapy, chemotherapy, and targeted therapy.
- Examine How Drug Resistance Develops: Examine how various treatment modalities cause cancer cells to acquire drug resistance mechanisms. This includes looking into the development of resistant cell populations, modifications to signalling pathways, and possible countermeasures or preventative measures for drug resistance.

2. Flow of the simulation model (how the codes are constructed)

The code started with initialising and setting up the important and required libraries like Flask, NumPy, SciPy and Matplotlib. It then continued with defining the model like the current state y, time t, and various parameters (rC, dC, rH, kIL, kCT, s, K) as inputs and returns the rates of change of the state variables. The flow continues with defining the route for the homepage. '/' route is defined to handle both GET and POST requests. It also sets the default values for the model parameters. If a POST request is received (i.e., form submission), the parameters are updated with the values provided in the form. Odeint is utilised to solve the differential equations, and Matplotlib is employed to plot the obtained results. The results are plotted and saved as an image in a static folder and rendered in the homepage template along with the current parameter values. The '/results' route is defined to display the simulation results. This route renders a separate results template, which can be accessed independently from the homepage. Finally, the Flask application is run using 'app.run(debug=True)' to start the development server.

3. List of mathematical equations used and its descriptions.

There are around 5 mathematical equations used in this simulation model. These mathematical equations are used to describe the dynamics of interactions between various components of immune systems and various cells within the tumour microenvironment.

1. $dCdt = rC * C * (1 - (T/K)) * (1 - S) - dC * C$

   The change in the concentration of cancer cells over time, represented by dCdt, is influenced by the rate of growth of cancer cells (rC), the competition of other cells for resources (1-(T/K)), the impact of treatment (1-S), and the death rate of cancer cells (dC).

2. $dHdt = rH * H$

   The change in the concentration of healthy cells is represented by dHdt which depends on the growth rate of healthy cells (rH).

3. $dILdt = kIL * H$

The rate of the concentration of interleukins is represented by dILdt which depends on the production rate of interleukins (kIL).

4. dTdt = -kCT * C * T

   The change in the concentration of tumour cells over time is represented by dTdt which depends on the competition for resources with other cells, the death rate of tumour cells due to treatment (kCT), and the concentration of tumour cells themselves (T).

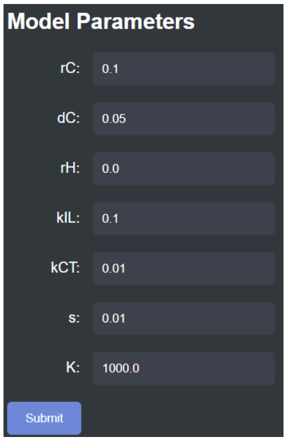
5. dSdt = s * T

   The rate of the effect of treatment depends on the concentration of tumour cells (T) and the effectiveness of the treatment (s).

4. Python libraries used

Library	Description
flask	A lightweight web framework for building web applications in Python.
numpy	A fundamental library for numerical computations in Python.
matplotlib	A plotting library for creating visualisations in Python.
scipy.integrate	An open-source library for numerical integration.

5. Input of the simulation model



Model Parameters

rC: 0.1

dC: 0.05

rH: 0.0

kIL: 0.1

kCT: 0.01

s: 0.01

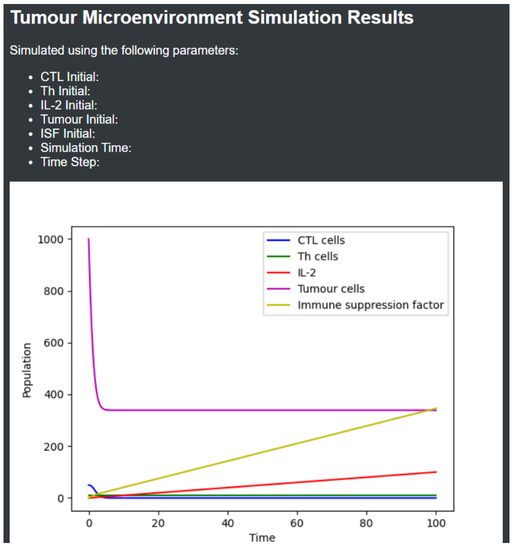
K: 1000.0

These parameters control the model's behaviour throughout time by dictating the changes and interactions between several cell types (Th cells, IL-2, tumour cells, CTL cells, and the immune suppression factor). We can observe how changes affect the system's dynamic by altering the value of the parameters through the form.

6. Model parameters

Parameters	Description
rC	Rate of CTL (Cytotoxic T Lymphocyte) cell proliferation.
dC	Rate of CTL cell death.
rH	Rate of Th (Helper T) cell proliferation.
kIL	Rate constant for IL-2 (Interleukin-2) production by Th cells.
kCT	Rate constant for Tumour cell destruction by CTL cells.
s	Rate of immune suppression factor (S) production.
K	Carrying capacity of Tumour cells (T), representing the maximum sustainable population of tumour cells.

7. Description of simulation output and the generated graph



Tumour Microenvironment Simulation Results

Simulated using the following parameters:

- CTL Initial:
- Th Initial:
- IL-2 Initial:
- Tumour Initial:
- ISF Initial:
- Simulation Time:
- Time Step:

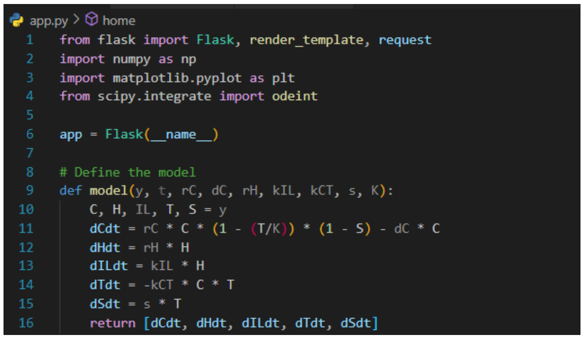
The graph above illustrates the evolution of the cell types over time. The y-axis represents the population of cells while the x-axis represents time. Initially, the count of the population of the tumour cells is high, hovering around 1000. However, it experiences a sharp decline within a short time, almost reaching 350, and then stabilises. The CTL cells count exhibits a slight decrease from Time 0 and remains constant as time progresses while the population of Th cells always stay constant from Time 0. This phenomenon is due to the gradually rising population of immune suppression factors. The IL-2 also slightly increases over time.

8. Experimentation that can be carried out using the simulation model

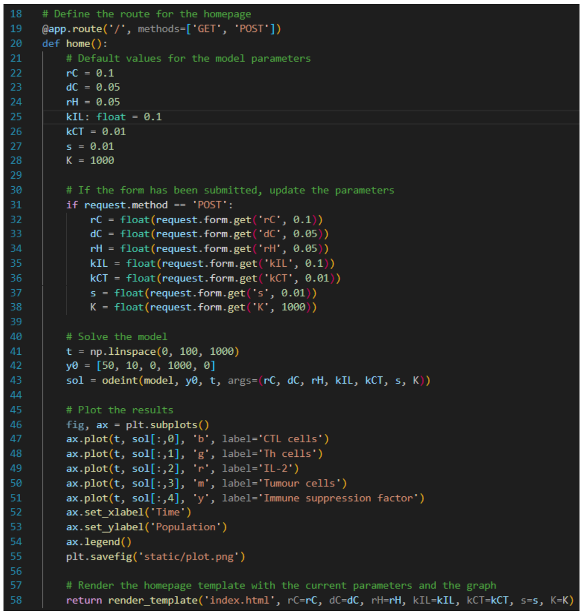
The simulation model can be used to simulate the development of various therapies by adjusting the parameters to study the effect of immunotherapy in combination with other treatments.

Appendix

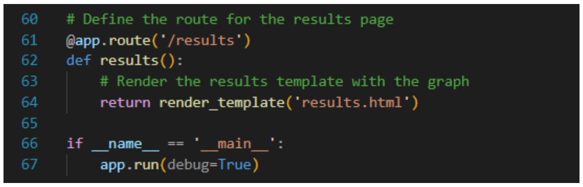
a. app.py



```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]
```

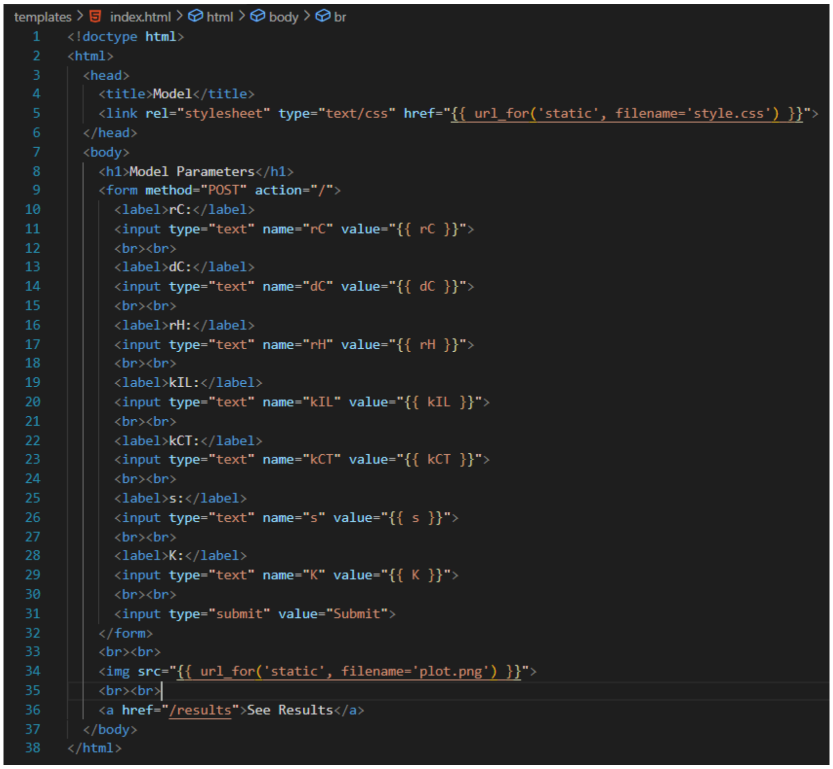


```python
# Define the route for the homepage
@app.route('/', methods=['GET', 'POST'])
def home():
    # Default values for the model parameters
    rC = 0.1
    dC = 0.05
    rH = 0.05
    kIL: float = 0.1
    kCT = 0.01
    s = 0.01
    K = 1000

    # 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)
```



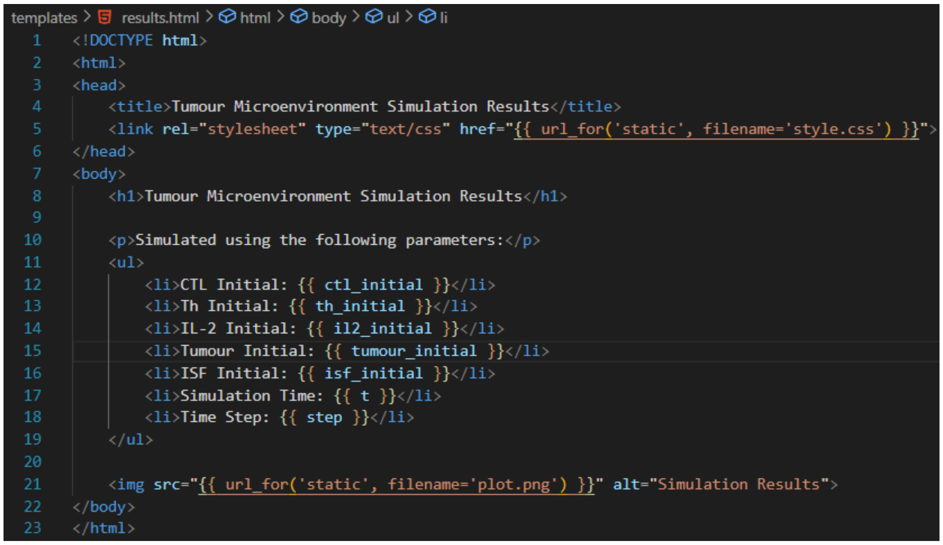
```python
# Define the route for the results page
@app.route('/results')
def results():
    # Render the results template with the graph
    return render_template('results.html')

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

b. index.html



```html
<!doctype html>
<html>
  <head>
    <title>Model</title>
    <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='style.css') }}">
  </head>
  <body>
    <h1>Model Parameters</h1>
    <form method="POST" action="/">
      <label>rC:</label>
      <input type="text" name="rC" value="{{ rC }}">
      <br><br>
      <label>dC:</label>
      <input type="text" name="dC" value="{{ dC }}">
      <br><br>
      <label>rH:</label>
      <input type="text" name="rH" value="{{ rH }}">
      <br><br>
      <label>kIL:</label>
      <input type="text" name="kIL" value="{{ kIL }}">
      <br><br>
      <label>kCT:</label>
      <input type="text" name="kCT" value="{{ kCT }}">
      <br><br>
      <label>s:</label>
      <input type="text" name="s" value="{{ s }}">
      <br><br>
      <label>K:</label>
      <input type="text" name="K" value="{{ K }}">
      <br><br>
      <input type="submit" value="Submit">
    </form>
    <br><br>
    <img src="{{ url_for('static', filename='plot.png') }}">
    <br><br>
    <a href="/results">See Results</a>
  </body>
</html>
```

c. results.html



```html
<!DOCTYPE html>
<html>
<head>
    <title>Tumour Microenvironment Simulation Results</title>
    <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
    <h1>Tumour Microenvironment Simulation Results</h1>

    <p>Simulated using the following parameters:</p>
    <ul>
        <li>CTL Initial: {{ ctl_initial }}</li>
        <li>Th Initial: {{ th_initial }}</li>
        <li>IL-2 Initial: {{ il2_initial }}</li>
        <li>Tumour Initial: {{ tumour_initial }}</li>
        <li>ISF Initial: {{ isf_initial }}</li>
        <li>Simulation Time: {{ t }}</li>
        <li>Time Step: {{ step }}</li>
    </ul>

    <img src="{{ url_for('static', filename='plot.png') }}" alt="Simulation Results">
</body>
</html>
```