FACULTY OF COMPUTING

SECB4313-01
BIOINFORMATICS MODELING AND SIMULATION
ASSIGNMENT 1

LECTURER:
DR. AZURAH BTE A SAMAH

GROUP MEMBERS:
MARNISHA BINTI MUSTAFA KAMAL (A20EC0075)
SAYANG ELYIANA AMIERA BINTI HELMEY (A20EC0143)

TABLE OF CONTENT

1.0 DESCRIPTION OF SIMULATION MODEL **3**

1.1 Introduction 3

1.2 Objectives of the Simulation Model 3

2.0 FLOW OF SIMULATION MODEL **4**

2.1 Model Flowchart 4

3.0 MATHEMATICAL EQUATIONS **5**

3.1 List of Mathematical Equations Used 5

3.1.1 Equation 1 *5*

3.1.2 Equation 2 *5*

3.1.3 Equation 3 *5*

3.1.4 Equation 4 *5*

3.1.5 Equation 5 *5*

4.0 PYTHON LIBRARIES USED **6**

5.0 INPUT OF THE SIMULATION MODEL **7**

6.0 MODEL PARAMETERS **8**

7.0 OUTPUT OF SIMULATION MODEL **9**

7.1 The Generated Graph 9

7.2 Description of Simulation Output 10

8.0 APPENDIX **11**

8.1 Python Codes 11

8.2 HTML Codes 13

8.2.1 index.html *13*

8.2.2 results.html *14*

1.0 DESCRIPTION OF SIMULATION MODEL

1.1 Introduction

This report will provide the summarization of a microenvironment tumor simulation model, according to the article titled 'Modeling a Disease-model in Python' written by Anoop Johny. The model is created in order to simulate the dynamic interaction between a variety of cell populations among tumors which includes CTL cells, Th cells, IL-2, Tumor cells and ISF.

- CTL cells (Cytotoxic T lymphocytes): Immune cells that are capable of killing tumor cells
- Th cells (T helper cells): Immune cells that are capable of activating CTL cells
- IL-2 (Interleukin-2): A molecule that induces the growth and multiplication of T cells
- Tumor cells: The cancerous cells that are targeted by the immune system
- ISF (Immune Suppressive Factor): A factor generated by the tumor microenvironment that is capable to inhibit immune response

The models are designed with the intention of providing insights on the intricate relationships between these elements and how they influence the development of tumors.

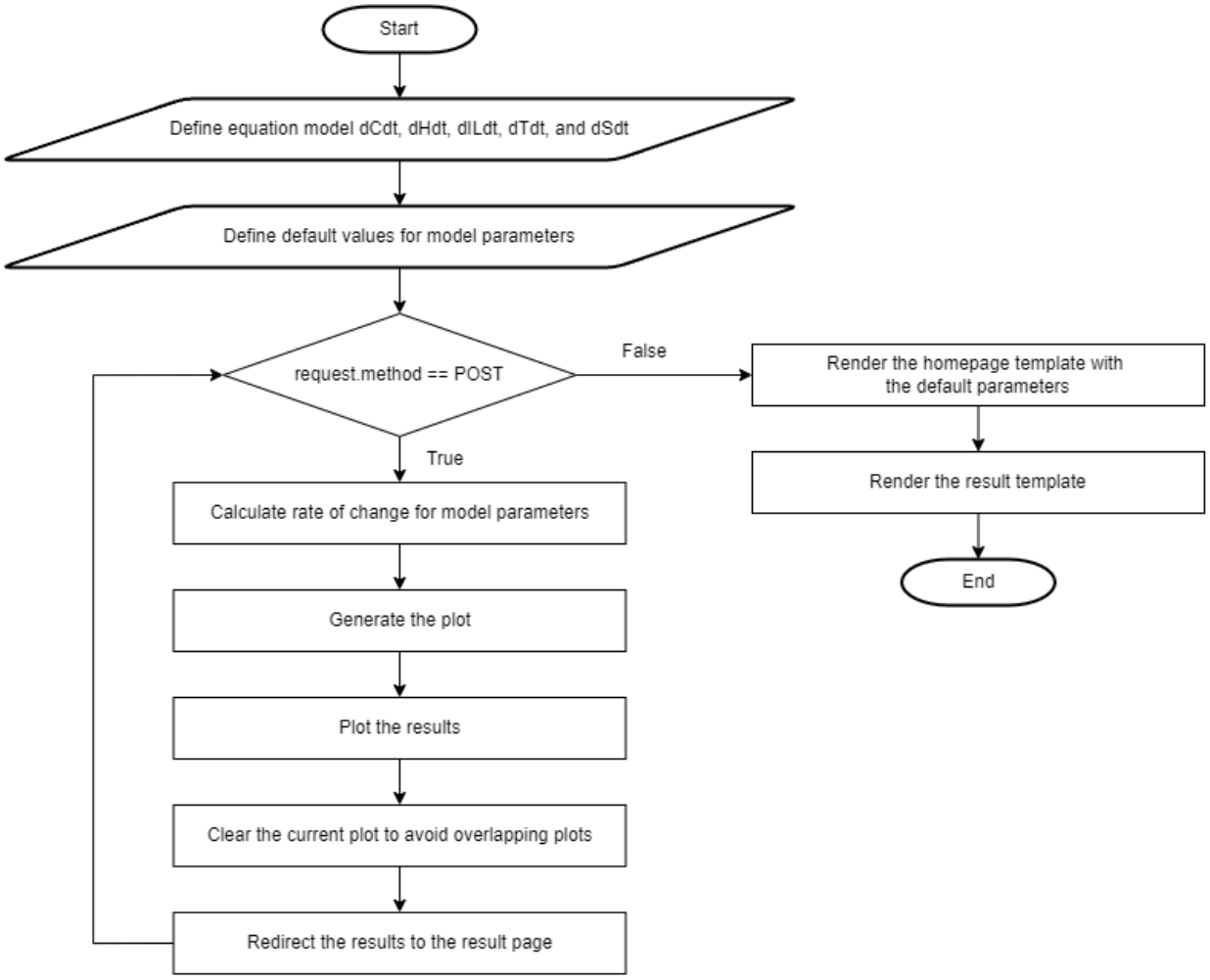
1.2 Objectives of the Simulation Model

The objectives of this simulation model are portrayed below.

- To understand the dynamics and intricacies of the tumor microenvironment
- To assess the impact of various different factors on tumor advancement

2.0 FLOW OF SIMULATION MODEL

2.1 Model Flowchart



3.0 MATHEMATICAL EQUATIONS

3.1 List of Mathematical Equations Used

3.1.1 Equation 1

$$\frac{dC}{dt} = rC \times C \times (1 - \frac{T}{K}) \times (1 - S) - dC \times C$$

This mathematical expression describes the variation in the level of cancerous cells as time progresses. It is considered as the growth rate of cancer cells ('rC'), the rivalry for resources among cells (1-T/K), the impact of therapeutic interventions (1-S), and the death rate of cancer cells ('dC).

3.1.2 Equation 2

$$\frac{dH}{dt} = rH \times H$$

This mathematical expression demonstrates the alterations in the concentration of healthy cells over time. The growth rate of healthy cells (rH) is the sole determinant for this change.

3.1.3 Equation 3

$$\frac{dIL}{dt} = kIL \times H$$

This mathematical expression shows the alterations in the concentration of interleukins over time. The synthesis rate of interleukins (kIL) is dependent on the concentration of healthy cells (H).

3.1.4 Equation 4

$$\frac{dT}{dt} = -kCT \times C \times T$$

This mathematical expression illustrates the variation in the level of neoplastic cells as time progresses. It considers the rivalry for resources among other cells, the death rate of tumor cells from therapy (kCT), and the concentration of tumor cells themselves (T).

3.1.5 Equation 5

$$\frac{dS}{dt} = s \times T$$

This mathematical expression illustrates the variation across different time intervals. It is determined by the level of tumor cells (T) and the efficacy of the treatment (s).

4.0 PYTHON LIBRARIES USED

Python Library	**Usage**
Flask	Flask is used to determine routes, handle the HTTP requests, and render HTML templates.
render_template	This function is used to render HTML templates to allow dynamic web pages creation by inserting data into placeholders defined in the HTML file.
request	This function is used for handling HTTP requests in Flask. It is used to access the data from the submitted form by the user in the homepage.
redirect	This function is used to redirect the user to a different URL. It is used to navigate the user to the results page after they submit the form in the homepage.
url_for	This function is used to generate URLs dynamically. It is used to generate the URL for the results page when redirecting the user.
numpy (np)	NumPy is a Python library used for numerical computing. It is used to generate numerical data for the model simulation.
matplotlib.pyplot (plt)	Matplotlib is a library for Python that is used to plot quality figures. It is used to create plots of the model simulation results.
scipy.integrate.odeint	SciPy is a scientific computing library for Python. It is used to solve the differential equations defined by the model.

5.0 INPUT OF THE SIMULATION MODEL

The inputs are primarily handled through a web form on the homepage ('/'). Users can input values for parameters such as 'rC', 'dC', 'rH', 'kIL', 'kCT', 's', and 'K'. These inputs are obtained via the request.form.get() method.

The initial conditions for the simulation are set as default values ('y0 = [50, 10, 0, 1000, 0]'). These values represent the initial populations of CTL cells, Th cells, IL-2, tumor cells, and immune suppression factor, respectively.

6.0 MODEL PARAMETERS

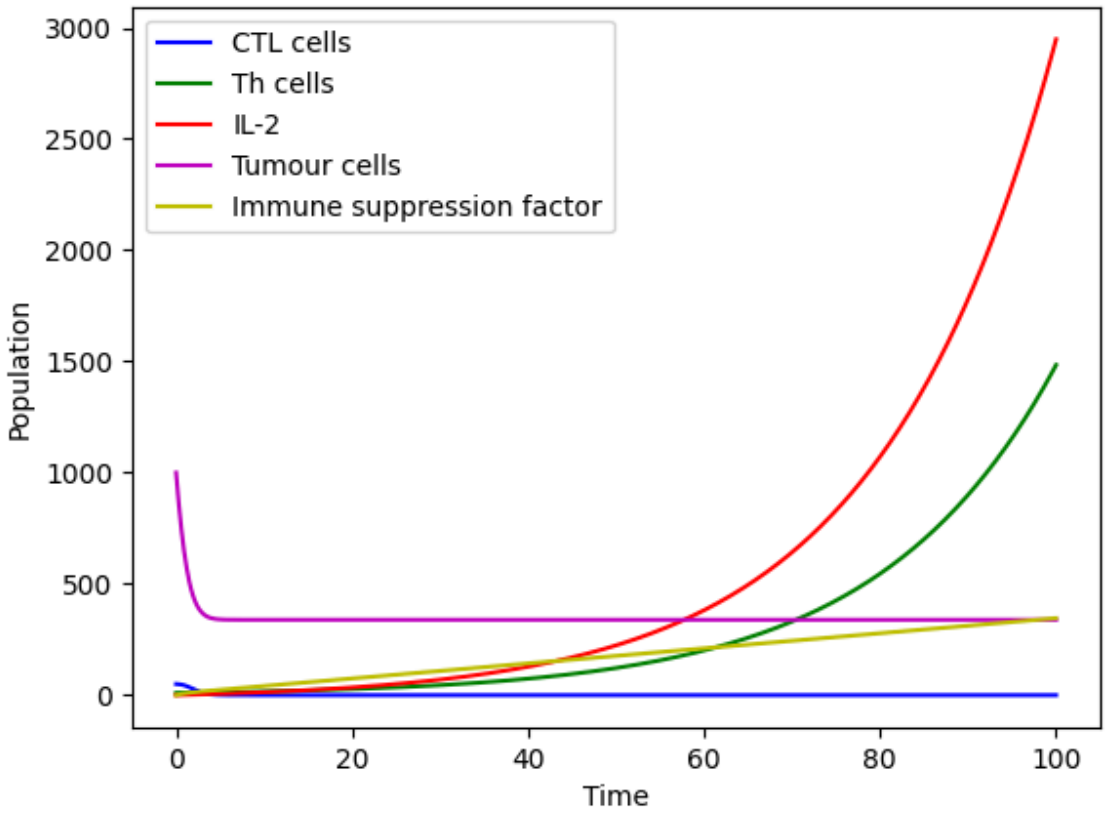
Parameters are the constants used in the mathematical model. They are defined within the model() function and include:

Input Parameters	**Acronym**	**Parameters Values**
The rate of CTL cell proliferation	rC	0.1
The rate of CTL cell death	dC	0.05
The rate of Th cell proliferation	rH	0.05
The rate of IL-2 production by Th cells	kIL	0.1
The rate of killing of tumor cells by CTL cells	kCT	0.01
The rate of increase in immune suppression due to tumor presence	s	0.01
The carrying capacity or maximum sustainable population size for tumor cells	K	1000

These parameters influence the behavior of the differential equations that describe the system dynamics. The code then integrates the differential equations using the odeint() function from SciPy to simulate the system's behavior over time. Finally, it generates a plot showing the population dynamics of CTL cells, Th cells, IL-2, tumor cells, and immune suppression factor over time, which is saved as 'static/plot.png'.

7.0 OUTPUT OF SIMULATION MODEL

7.1 The Generated Graph



By observing the trends and interactions between these components on the graph, one can gain insights into how changes in the parameters affect the immune response against tumor growth. For example, increasing the rate of CTL cell proliferation or decreasing the rate of tumor cell proliferation may result in more effective suppression of tumor growth over time. Conversely, alterations in other parameters could lead to different outcomes in terms of immune response and tumor progression.

7.2 Description of Simulation Output

The plot illustrates the complex interactions among various cell populations, including immune cells and tumor cells over time. Observing the changes in the population sizes reveals and highlights the adaptive responses to the presence of tumor cells and the corresponding adaptations of tumor cells under immune surveillance.

The fluctuations observed in the plot indicate the variations in the quantities of immune cells and tumor cells, reflecting the dynamic aspect of the immune response. Peaks in CTL cells (cytotoxic T lymphocytes) might be corresponsive to active stages of tumor cell destruction, while troughs could indicate periods of immune inactivity or suppression. Changes in Th cells (helper T lymphocyte) and IL-2 (interleukin-2) levels signify the stimulation and regulation of the immune response, influencing the proliferation and functionality of other immune cells. Changes in the population of tumor cells reveal the effectiveness of the immune response in managing tumor growth or the appearance of tumor escape mechanisms.

Model parameters have effects that can be observed by modulating variables such as rC (proliferation rate of CTL cells), dC (death rate of CTL cells), rH (proliferation rate of Th cells), kIL (production rate of IL-2), kCT (cytotoxicity rate), s (suppression rate), and K (carrying capacity) can lead to distinct patterns in the plot. Conducting a sensitivity analysis on these model parameters is important as it can help identify critical factors that influence the dynamics of the immune-tumor interaction, thus providing guidance for experimental design and potential therapeutic interventions.

8.0 APPENDIX

8.1 Python Codes

```
from flask import Flask, render_template, request, redirect, url_for
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 = 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))

        # Generate the plot
        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')

        # Clear the current plot to avoid overlapping plots
        plt.clf()

        # Redirect to the results page
        return redirect(url_for('results'))

    # Render the homepage template with the default parameters
    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
    return render_template('results.html')

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

8.2 HTML Codes

8.2.1 index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Simulation Parameters</title>
    <!-- Bootstrap CSS -->
    <link
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
rel="stylesheet">
    <style>
        /* Adjust margin and padding for the container */
        .container-custom {
            padding: 20px;
            background-color: #f8f9fa; /* Grey color */
        }
    </style>
</head>
<body>
    <div class="container container-custom">
        <h1 class="text-center mb-4">Simulation Parameters</h1>
        <form action="/" method="post" autocomplete="off">
            <div class="form-group">
                <label for="rC">rC:</label>
                <input type="text" class="form-control" name="rC" value="{{ rC }}">
            </div>

            <div class="form-group">
                <label for="dC">dC:</label>
                <input type="text" class="form-control" name="dC" value="{{ dC }}">
            </div>

            <div class="form-group">
                <label for="rH">rH:</label>
                <input type="text" class="form-control" name="rH" value="{{ rH }}">
            </div>

            <div class="form-group">
                <label for="kIL">kIL:</label>
                <input type="text" class="form-control" name="kIL" value="{{ kIL }}">
            </div>

            <div class="form-group">
                <label for="kCT">kCT:</label>
                <input type="text" class="form-control" name="kCT" value="{{ kCT }}">
            </div>

            <div class="form-group">
                <label for="s">s:</label>
                <input type="text" class="form-control" name="s" value="{{ s }}">
            </div>
```

```
            <div class="form-group">
                <label for="K">K:</label>
                <input type="text" class="form-control" name="K" value="{{ K }}">
            </div>

            <button type="submit" class="btn btn-primary">Submit</button>
        </form>
    </div>

    <!-- Bootstrap JS (Optional, for Bootstrap features like modal) -->
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
</html>
```

8.2.2 results.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Simulation Results</title>
</head>
<body>
    <h1>Simulation Results</h1>
    <img src="{{ url_for('static', filename='plot.png') }}" alt="Plot">
</body>
</html>
```