UTM

UNIVERSITI TEKNOLOGI MALAYSIA

SECB4313

Bioinformatics Modelling & Simulation

Section 1

Assignment 1

Dr. Azurah A Samah

Group Members:

Name	Matric No.
Mek Zhi Qing	A20EC0077
Zereen Teo Huey Huey	A20EC0173

Introduction

The simulation model is a tool which can help in understanding the cellular dynamics and the relationships between the cells and tumors. For easier understanding, it is a computational simulate the process of infection and tumor growth in the microenvironment. This enables the researcher to investigate different therapy approaches and circumstances in order to know about the underlying biological mechanisms and available treatment. The example shows the Python Flask web application which define a mathematical model to understand immune response to cancer using the ordinary differential equations (ODEs). By applied the equations to the data input, the result will be visualized through plotted graphs displayed on the webpage.

Objective

The objective of model is to explore how changes in various parameters such as cell growth rates, immune response factors, and treatment efficacy affect the population dynamics of different cell types over time.

Flow of Simulation Model

First, the Flask web application is initialized and defines a mathematical model which using OEDs to reflect the dynamics of cancer progression and therapy effects. The user can use this model through a webpage which provided a set of default value for the parameter. If the user wishes to adjust the parameter, they can directly interact with the form displayed on the homepage by key in the desired value. The inputs will post to the application and the model parameter will be updated based on the value filled. The simulation disease model will integrate these parameters using odeint while Matplotlib is used to visualise the result. After that, the program displays the updated parameters and simulation graph on the homepage so that the users can observe and evaluate the dynamics of the system under various settings. This graph also can be presented separately for a more in-depth analysis on a different results page.

Mathematical Equation

The model has five different variables which are C, H, IL, T, and S. C represents the concentration of cancer cells, H represents the concentration of healthy cells, IL represents the

concentration of interleukins, T represents the concentration of tumour cells while S represents the effect of treatment.

The first equation used is listed as below:

dCdt = rC * C * (1 — (T/K)) * (1 — S) — dC * C

This equation represents the change in the concentration of cancer cells over time. It 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).

The second equation used is listed as below:

dHdt = rH * H

This equation represents the change in concentration of healthy cells over time. It is dependent on the rate of growth of healthy cells (rH).

The third equation used is listed as below:

dILdt = kIL * H

This equation represents the change in concentration of interleukin over time. It is dependent on the rate of production of interleukin (kIL).

The fourth equation used is listed as below:

dTdt = -kCT * C * T

This equation represents the change in the concentration of tumour cells over time. It is dependent 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).

The fifth equation used is listed as below:

dSdt = s * T

This equation represents the change in the effect of treatment over time. It is dependent on the concentration of tumour cells (T) and the effectiveness of the treatment (s).

Python Libraries

The Python libraries used included Flask. Flask is a web framework that provides features and tools for creating web application. It shows great performance and scalability for applications that ranged from small to medium-sized. Next, the second Python libraries used is NumPy. It provides support for multi-dimensional arrays and tools to deal with these arrays. Besides, matplotlib is used for creating graphics and interactive figures for better visualization for results. Lastly, SciPy is used as it provides the ability to solve complex mathematical equations. In this model, odeint function of SciPy is imported to solve the ordinary differential equations.

Inputs of the Simulation Model

The inputs of the model are listed as below:

- y: list of initial values (C, H, IL, T, S)
- t: time

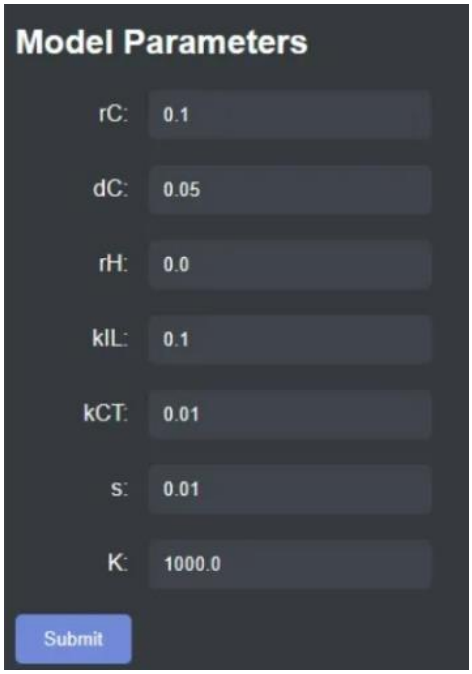
Model Parameters

The parameters of the model are listed as below:

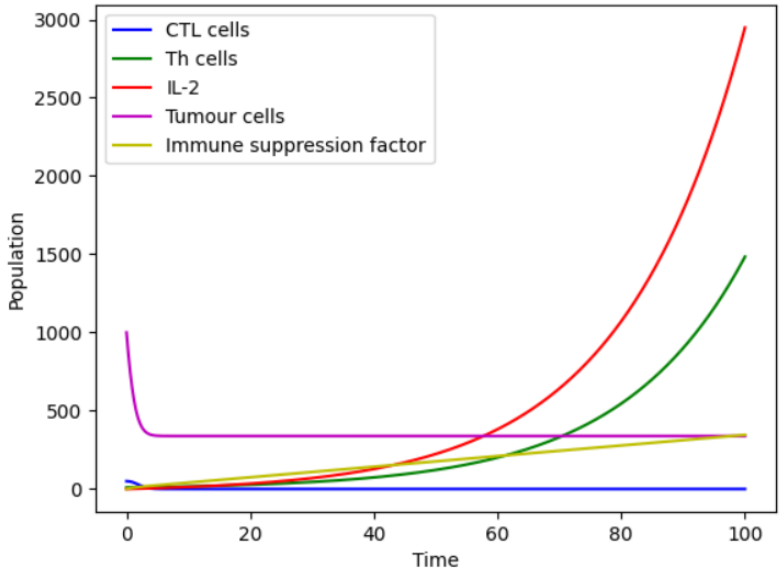
- rC: rate of growth of cancer cells
- dC: death rate of cancer cells
- rH: rate of growth of healthy cells
- kIL: rate of production of interleukin
- kCT: the death rate of tumour cells due to treatment
- s: effectiveness of treatment
- K: maximum population size that the environment can sustain

Simulation Output and Graph Generated

After compiling and running the code, the application will first prompt out an input page with default value to let user inputs the value. Then, the simulation output is presented in the form of graphical where a graph will be generated according to the model parameters input by the user. The input page is shown as the figure below.



The graph below shows the simulation output according to the default model parameters.



Based on the graph, the population size of the tumour cells decreased initially and remained unchanged for the rest of the time. Besides, the population size of IL-2 (interleukin cells) and Th cells increased exponentially. For CTL cells, the population size decreased a bit initially and remained unchanged for the rest of time. Lastly, for the immune suppression factor, it increased slowly from time to time.

Experimentation that can be carried out

The simulation disease model can be used in the parameter sensitivity analysis as it can investigate how the dynamics of cell populations are affected over time by variations in model parameters, such as growth rates, decay rates, immune response components, and treatment efficacy. Besides, it also helps in the immune system interactions such as the dynamics of immune cell populations and their interactions with tumor cells and immuno-suppressive substances. This allows researchers to learn more about the mechanisms behind disease development and possible therapeutic targets.

Appendix

```
!pip install flask

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))

    # Integrate the model with the updated parameters
    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, use_reloader=False)
```