UTM

UNIVERSITI TEKNOLOGI MALAYSIA

FACULTY OF COMPUTING

SCHOOL OF COMPUTING

SECB4313-01

BIOINFORMATICS MODELING AND SIMULATION

ASSIGNMENT 1 - MODELING A DISEASE MODEL IN PYTHON

LECTURER:

DR. AZURAH BINTI A SAMAH

GROUP MEMBERS:

SHAHRIL BIN SAIFUL BAHRI (A20EC0144)

MUHAMMAD AIMAN BIN ABDUL RAZAK (A20EC0082)

NAVINTHRA RAO A/L VENKATAKUMAR (A20EC0104)

TABLE OF CONTENTS

1) Introduction and objective of the model

Introduction

The website emphasizes the value of simulating the spread of sickness, particularly when it comes to infectious diseases. It discusses the critical role mathematical models play in comprehending the mechanisms of illness spread and forecasting its future trajectory. The website highlights how important it is to use Python for modeling because of its ease of use and widespread acceptance among scholars and data scientists.

Objective

The objective of the model analyzed in the article is to replicate the gradual dispersion of an illness among a community. The model specifically attempts to reflect the dynamics of disease transmission by taking into account variables like the size of the vulnerable population, the rate of infection, and the rate of recovery. Through the simulation of these variables, the model aims to offer insights into the ways in which various interventions, including immunization or social distancing, can influence the disease's transmission. The ultimate objective is to create a tool that will help public health professionals and legislators make well-informed decisions to lessen the effects of infectious illnesses.

2) Simulation Model Flow

The coding begins by importing essential libraries such as numpy, matplotlib, and scipy. “odeint” will be imported from “scipy.integrate”. This function is important for numerical simulations in scientific and engineering contexts. Following the imports, the model itself is defined. This function encapsulates the system's dynamics, receiving variables representing concentrations of various substances (C, H, IL, T, S), alongside time and parameters characterizing the system's behavior (rC, dC, rH, kIL, kCT, s, K). It returns derivatives of these variables, dictating their rates of change over time.

Afterwards, the script establishes routes for web application functionality. The home route, triggered upon accessing the homepage, handles both GET and POST requests. Initially, it sets default parameter values, serving as fallbacks if no user input is provided. Upon form submission, the function updates parameters with values from the POST request, retrieving them using request.form.get() based on their keys. It then proceeds to solve the system of differential equations using the modified parameters and renders the homepage template, showcasing adjusted parameters and a graph visualizing simulation outcomes.

Lastly, the results route renders a template displaying simulation results alongside a saved graph. The condition, if name == 'main’, ensures that when the script is executed as the main program, it starts the Flask development server in debug mode, facilitating testing and debugging. Overall, this script integrates scientific modeling with web application functionality, enabling dynamic simulations and interactive exploration of system behavior.

3) List of Mathematical Equations

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

The expression of dCdt represents the change in the amount of cancer cells over time. It depends on variables like the pace at which cancer cells proliferate (rC), how they compete with other cells for resources (1 — (T/K)), how treatment affects them (1 — S), and the rate at which cancer cells die, dC.

Equation 2: dHdt = rH * H

Equation 3: dILdt = kIL * H

According to the mentioned equation, changes in the levels of interleukins and healthy cells over time are indicated by dHdt and dILdt, respectively. The rate of interleukin synthesis (kIL) and the growth rate of healthy cells (rH) determine these alterations.

Equation 4: dTdt = -kCT * C * T

dTdt represents how the concentration of tumor cells changes over time. This change depends on variables such as the competition between cells for resources, the treatment-induced tumor cell death rate, kCT, and the current tumor cell concentration, T.

Equation 5: dSdt = s * T

The concentration of tumor cells, T, and treatment efficiency, s, both affect the rate of change in the impact of treatment over time, or dSdt.

4) Python Libraries Used

Libraries	Description
Flask	Flask is a lightweight web application framework for Python. It's designed to make it easy to build web applications quickly and with minimal boilerplate code.
render_template	A function provided by Flask for rendering HTML templates. It allows you to dynamically generate HTML pages by combining static HTML with dynamic content from your Python code.
request	An object provided by Flask that contains information about the current HTTP request, such as form data, cookies, and headers. It allows you to access and process data sent by the client.
redirect	A function provided by Flask for redirecting the client to a different URL. It's commonly used after processing a form submission to redirect the user to a different page.
url_for	A function provided by Flask for generating URLs for endpoints defined in your application. It's used to ensure that your application remains flexible and

	robust to changes in URL structure.
numpy	A fundamental package for numerical computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently.
matplotlib	A Python library for creating static, animated, and interactive visualizations. It's commonly used to generate plots and charts from numerical data.
matplotlib.pyplot	A module within Matplotlib that provides a MATLAB-like interface for creating plots. It simplifies the process of generating various types of plots, such as line charts, scatter plots, and histograms.
scipy	A scientific computing library for Python that builds on top of NumPy. It provides additional functionality for optimization, integration, interpolation, and more.
scipy.integrate	A module within Scipy that provides functions for numerical integration. It includes tools for solving ordinary differential equations (ODEs) and integrating mathematical functions.
odeint	A function provided by Scipy for solving

	systems of ordinary differential equations (ODEs). It's particularly useful for simulating dynamic systems and modeling physical phenomena.

5) Simulation Model Input

The simulation model actually consists of a system of ordinary equations (ODEs) that describe the dynamics of the populations of cells over time. The equations regulate how the populations of CTL cells, Th cells, IL-2, tumor cells and suppression factors change with control over time.

The model's input variables or initial conditions consist of inputs such as (C, H, IL, T and S). 'C' is the initial population of CTL (Cytotoxic T Lymphocyte) cells and 'H' is the initial population of Th (helper T) cells. Meanwhile, 'IL' is the initial concentration of IL-2 (Interleukin-2). 'T' is the initial population of tumor cells. 'S' is the initial immune suppression factor. However, there is one variable for time such as 't' that represents time which is continuous and spans from 0 to 100 units.

6) Model Parameters

The model has several parameters such as (rC, dC, rH, kIL, kCT, s and K). The parameters consisting of 'rC' is the growth rate of a CTL cell, 'dC' is the CTL cell death rate, 'rH' is the cell growth rate. Other than that, parameter 'kIL" is the rate of IL-2 production by Th cells, parameter "kCT" is the rate of tumor cell destruction by CTL cells, parameter 's' is the immune suppression factor production rate. Finally, the parameter 'K' is the carrying capacity of the tumor cells which means the maximum number of tumor cells that a specific environment (such as a tumor's microenvironment within a body) can support.

The model's parameter is set to certain values such as parameter (rC)'s value is 0.1, parameter (dC)'s value is 0.05, parameter (rH)'s value is 0.05, parameter (kIL)'s is set to float with values 0.1, parameter (kCT)'s value is 0.01. Other than that, parameter (s)'s value is set to 0.01 and parameter (K)'s value is 1000.

7) Simulation Model Output

7.1) Model Output Description

The model represents an immune response to tumor growth. The simulation computes the population size of each cell type over a time span (t) of 100 units. The initial conditions are set within "yo" as shown below:

- CTL cells (CTL) : 50
- Th cells (H) : 10
- IL-2 (IL) : 0
- Tumor cells (T) : 10
- Immune suppression factor (S) : 0

7.2) Generated Graph Description

The simulation model results are interpreted in the form of a plot which shows the line graph of the changes in a population of various types of immune cells over time.

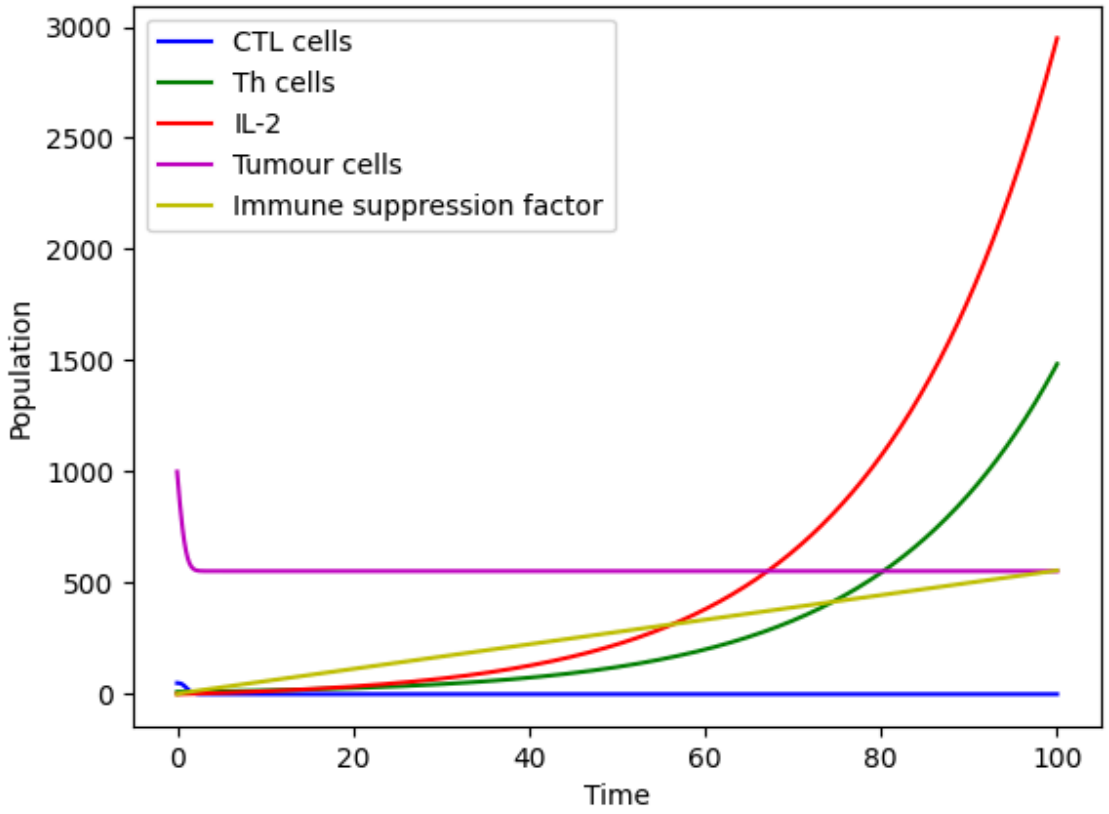

Figure 1.1: The Plot Graph

From the Figure 1.1 above, the y-axis shows the population of each cell and the x-axis shows time. Here's a possible interpretation of the graph. The graph can be interpreted in following conditions:

- The y-axis shows the number of cells.
- The x-axis shows time.
- The five lines represent different cell types and factor: CTL cells (blue), Th cells (green), IL-2 (red), tumor cells (magenta), and an immune suppression factor (yellow).

Finally, the plot graph suggests that the immune system is responding to the presence of tumor cells. The Th cells and IL-2 are increasing in population, which suggests that the Th cells are attacking the tumor cells and IL-2 is involved in stimulating the proliferation of

T-lymphocytes which includes CTL and Th cells. The tumor cells are decreasing in population, which suggests that the immune system is having an effect. The immune suppression factor is linear, which suggests that it is having a major impact on the immune response.Other than that, the CTL cells are relatively constant over the time spanning to 100 units.

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

Some of the experiments that can be conducted are parameter optimization and population dynamics. Parameter optimization uses optimization algorithms to find the parameter values that maximize or minimize specific outcomes. For instance, it is necessary to optimize the IL-2 production rate (kIL) to achieve the highest tumor clearance while minimizing immune suppression.

Besides that, the population dynamics involves simulating the population dynamics over time using different initial conditions and parameter sets. Therefore, we can observe how the carrying capacity (K) affects the equilibrium population sizes of CTL cells, Th cells, and tumor cells.

9) Appendix

```
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)
```

To refer more, do refer the link stated below:
https://github.com/shahril2011/eportfolio-SECB4313-01/tree/master/projects/assignment1