
UTM
UNIVERSITI TEKNOLOGI MALAYSIA

# SECB 4313 BIOINFORMATICS MODELING AND SIMULATION

## ASSIGNMENT 1

## LECTURER: DR. AZURAH BINTI A SAMAH

## SECTION 1

## TEAM MEMBERS:

| NO. | NAME | MATRIC NUMBER |
|---|---|---|
| 1. | FELICIA CHIN HUI FEN | A20EC0037 |
| 2. | GOH YITIAN | A20EC0038 |

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

Disease modeling simulates the behaviour of cells and tumors in a computational environment, allowing researchers to study the dynamics of immune cells, cancer cells, and their interactions. This approach helps explore different treatment options and understand underlying biological mechanisms. Models ranging from simple differential equations to complex agent-based simulations can be used to study tumor growth, immune response, and drug resistance. By calibrating these models with experimental data, researchers can test hypotheses and refine their understanding, contributing to the development of new therapies and treatment strategies.

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

First, import the Python library.

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

After that, we define the model function.

```
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]
```

The model function is used to define the differential equations for the system.
The model variables are defined as:

| Model Variables | Description |
|---|---|
| *C* | represents the concentration of cancer cells. |
| *H* | represents the concentration of healthy cells. |
| *IL* | represents the concentration of interleukins (proteins involved in immune response) |
| *T* | represents the concentration of tumor cells. |
| *S* | represents the effect of treatment on cancer cell growth. |

Then, set the 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
```

Next, the code will prompt the user to enter the model parameters in the index.html form.

Lastly, apply the parameters and plot the result:

```
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()
```

t = np.linspace(0, 100, 1000). This creates an array of 1000 evenly spaced time points between 0 and 100. This array represents the time over which the differential equations will be solved.

y0 = [50, 10, 0, 1000, 0]. This is the initial condition for the system. It contains initial values for the variables being modeled.

sol = odeint(model, y0, t, args=(rC, dC, rH, kIL, kCT, s, K)). The odeint function from the scipy.integrate library is used to solve the differential equation for the system.

y0: The initial conditions for the differential equations system.

t: The array of time points.

args=(rC, dC, rH, kIL, kCT, s, K): Additional parameters that are passed to the model function.

The detailed result will be shown in the result.html when the user clicks on the "see results" link.

3. List of mathematical equations used and its descriptions

   a. Equation 1: dCdt = rC * C * (1 - (T/K)) * (1 - S) - dC * C

      dCdt describes how the concentration of cancer cells changes over time. This change is influenced by several factors: 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). T/K is the ratio of tumor cells to the carrying capacity of the environment. When T/K is close to 1, the environment is at capacity and there is limited space for further tumor growth.

   b. Equation 2: dHdt = rH * H

      dHdt represents the changes in the concentration of healthy cells over time. It is dependent on the growth rate of healthy cells (rH)

c. Equation 3: DILdT = kIL * H

   DILdT represents the changes in the concentration of interleukins over time. It is dependent on the production rate of interleukins (kIL).

d. Equation 4: dTdt = -kCT * C * T

   dTdt represents the change in the concentration of tumor cells over time. It 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).

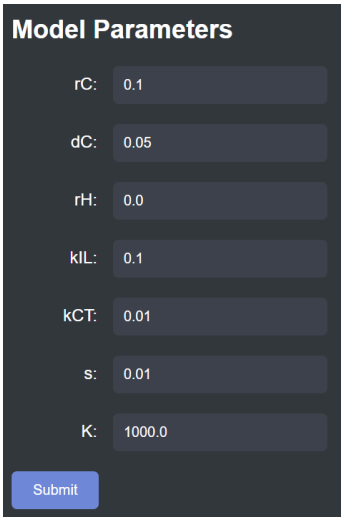
e. Equation 5: dSdt = s * T

   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)

4. Python libraries used

| **Library** | **Description** |
|---|---|
| numpy | Python library used for working with arrays. |
| matplotlib | A cross-platform, data visualisation and graphical plotting library. |
| scipy.integrate | A Python library for numerical integration. |
| flask | A python library used for web development. |

5. Input of the simulation model

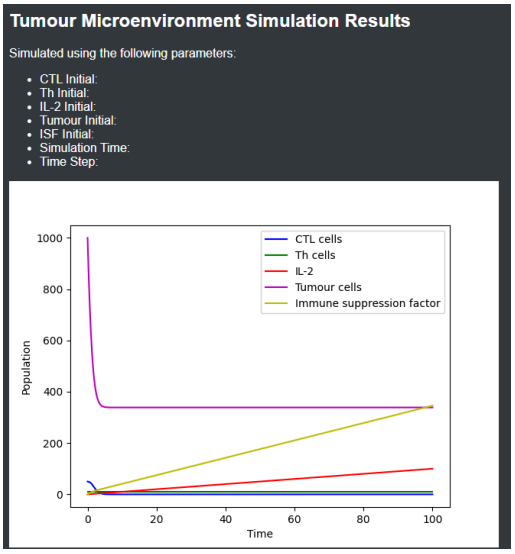



The values for each parameter are entered and submitted. The model parameters will then be passed to the function and calculate the result and define the simulation results.

6. Model parameters

| Model Parameters | Description |
|---|---|
| *rC* | rate of growth of cancer cells |
| *dC* | the death rate of cancer cells |
| *rH* | the growth rate of healthy cells |
| *kIL* | the production rate of interleukins |
| *kCT* | the death rate of tumor cells due to treatment |
| *s* | the effectiveness of the treatment |
| *K* | the number of tumor cells in the environment. |

7. Description of simulation output and the generated graph.




From the graph above, initially, the population of the tumor cell shows a sharp decrease from 1000 to around 350 over a short time and decreases slowly over time. The immune suppression factor shows a steady increase over time until it crosses with the tumor cell.

The CTL and Th cells remain constant over the time. This is because the immune suppression factor suppresses the immune response. The IL-2 slightly increases over time.

The simulation output shows modeling the simulation of tumor cells over time, tracking the population of CTL, Th and IL-2 cells, immune suppression factor and tumor cell over the time.

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

Creation of novel therapies and treatment plans by using the simulation model to observe the predicted effect on the tumor and the immune response over time.

## Appendix

a. App py code:

```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
```

```python
def home():
    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))
```

```python
def home():
    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)
```

b. Index html file:

```html
<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>
```

c. Result html file:

```
<!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>
```