Uppsala University

Modelling Complex Systems 1MA256

Dynamics of a Chaotic Cancer Model

Final Project

Isak Voltaire Edoh

1 Model Description

Cancer is a major health issue accounting for nearly 10 million deaths in 2020 [1]. The complexity of cancer has motivated many researchers to develop models to increase the general comprehension of the disease. One such approach is mathematical modelling with differential equations which allows for modelling of tumor growth dynamics. The main biological mechanism behind these models is the interaction of tumor cells with other healthy cells in the human body. This brief report studies a three-dimensional dynamical cancer growth model, containing the interaction between tumor cells, healthy tissue cells and immune cells (anti-tumor effector cells). The model is chaotic for a given set of parameters described in later sections. The existence of a chaotic attractor is proved by numerical simulations and verified by the computation of the Lyapunov exponents.

This project analyzes the mentioned above model formulated by Itik and Banks in [2], which is described by the following set of equations

$$\frac{dx}{dt} = x(1-x) - a_{12}xy - a_{13}xz, \tag{1}$$

$$\frac{dy}{dt} = r_2 y(1-y) - a_{21}xy, \tag{2}$$

$$\frac{dz}{dt} = \frac{r_3 xz}{x+k_3} - a_{31}xz - d_3 z. \tag{3}$$

For the sake of simplicity the set of equations (1)-(3) have been rescaled and nondimensionalized. Essentially the model consists of three cell populations; x are tumor cells, y are healthy host cells and z are immune cells. This dynamical model captures the competition between these cells, similar to the Lotka-Volterra prey predator model.

The first term of equation (1) is the logistic growth of tumour cells. The second and third term refers to the loss of tumour cells by competition with healthy host cells and the killing by immune cells, respectively. Similarly the first term in equation (2) corresponds to the logistic growth of healthy host cells, the second term refers to the tumor cells competing with healthy host cells. Equation (3) describes the change of the immune cell population. The first term is derived from a complex biological process, put simply, it describes the increase of immune cells depending on the amount of tumor cells. The last terms corresponds to the killing of immune cells by tumor cells and by natural death. All the parameters are defined in table 1.

Parameter	Value	Description
a_{12}	1.0	Rate of tumor cells killed by healthy cells.
a_{13}	2.5	Rate of tumor cells killed by immune cells.
r_2	0.6	Rate of healthy cells growth.
a_{21}	1.5	Rate of healthy cells killed by tumor cells.
r_3	4.5	Rate of immune cells stimulated.
k_3	1.0	
a_{31}	0.2	Rate of immune cells inactivated by tumor cells.
d_3	0.5	Rate of immune cells dying.

Table 1: Model parameters. Values are chosen so that the model becomes chaotic and with regards to being biologically sensible.

With these parameters the model provides the evolution of the tumor growth through the nature of equilibrium points, limit cycles and chaotic (strange) attractors. The most critical parameter is the bifurcation parameter a_{12}, which will be discussed in detail in later sections.

In order to analyze the stability and equilibrium points of the model, the Jacobian matrix is computed as

$$J = \begin{bmatrix} 1 - 2x - a_{12}y - a_{13}z & -a_{12}x & -a_{13}x \\ -a_{21}y & r_2 - 2r_2y - a_{21}x & 0 \\ \dfrac{r_3 z}{x + k_3}\left(1 - \dfrac{x}{x + k_3}\right) - a_{31}z & 0 & \dfrac{r_3 x}{x + k_3} - a_{31}x - d_3 \end{bmatrix}. \quad (4)$$

The system's equilibrium points are obtained by setting

$$\frac{dx}{dt} = \frac{dy}{dt} = \frac{dz}{dt} = 0 \quad (5)$$

which results in the following equations (nullclines)

$$x = 0, \quad x = 1 - a_{12}y - a_{13}z, \quad (6)$$

$$y = 0, \quad y = \frac{1 - a_{21}x}{r_2}, \quad (7)$$

$$z = 0, \quad z = x^2 + x\left(k_3 + \frac{d_3 - r_3}{a_{31}}\right) + \frac{d_3 k_3}{a_{31}}. \quad (8)$$

Hence solving the system of equations (6)-(8) results in eight unique equilibrium points (x_{eq}, y_{eq}, z_{eq}). The equilibrium points are characterized by inserting them into the Jacobian matrix resulting in different eigenvalues. The first three critical equilibrium points are described below.

1. Trivial equilibrium point at $(0, 0, 0)$ with eigenvalues $\lambda_{1,2,3} = (1, r_2, -d_3)$. This is a saddle point because there are two unstable eigenvalues $\lambda_{1,2}$ and one stable λ_3. No cells survive.

2. Non-trivial equilibrium point at $(0, 1, 0)$ with eigenvalues $\lambda_{1,2,3} = (-r_2, -d_3, 1 - a_{12})$. The system's trajectories go towards the healthy cell population, hence no cancer. The stability depends on the bifurcation parameter a_{12}.

3. Non-trivial equilibrium point at $(1, 0, 0)$ with eigenvalues $\lambda_{1,2} = (-1, -r_2 - a_{21})$ and $\lambda_3 = ((r_3 - d_3 - a_{31} - a_{31}k_3 - d_3k_3)/k_3 + 1)$. With the given parameters, $\lambda_{1,2}$ are stable and λ_3 is unstable making this a saddle point. The system goes toward the tumor cell population.

The remaining five equilibrium points are not within the scope of this project due to being biologically irrelevant (negative) and beyond the interest of this project. The relevance of these points are further explained in [2].

2 Simulation Results

The chaotic cancer model is simulated in the Python language with the given set of parameters in table 1. Noteworthy is that these parameters give rise to chaotic dynamics which are confirmed by the following numerical simulations.

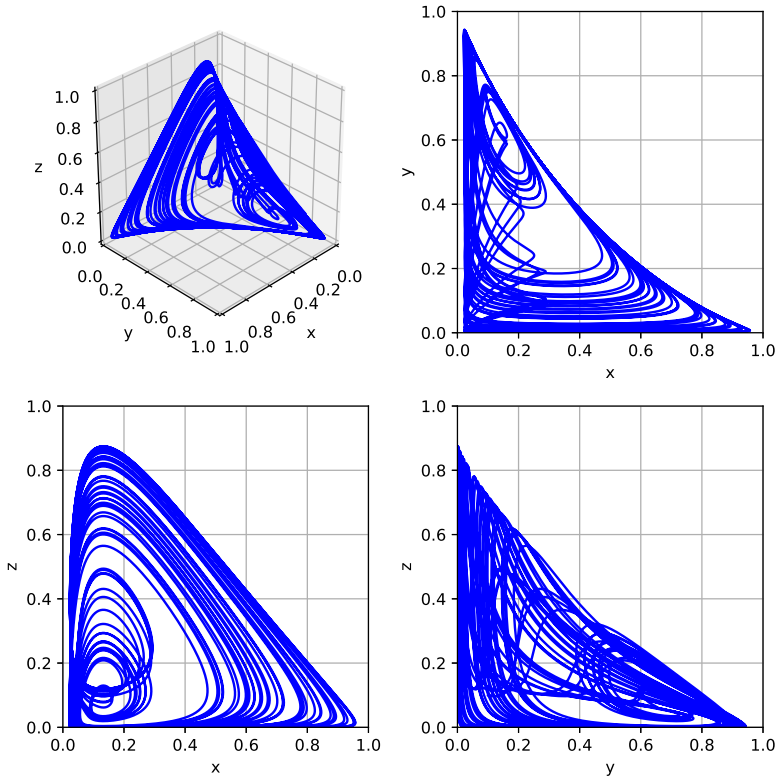

Figure 1: Phase space and corresponding phase planes.

Figure 1 illustrates the obvious chaotic attractor in the phase space and phase planes with the initial states $(x_0,\ y_0,\ z_0) = (0.1,\ 0.1,\ 0.1)$. Also visible is the origin point $(0, 0, 0)$ which is a saddle point, some trajectories head for it while other depart from it. The tumor-free solution in the steady state $(0, 1, 0)$ can also be found in the top-right corner of figure 1. This point may be Lyapunov stable for $a_{12} = 1$ since the eigenvalues are $\lambda_{1,2,3} = (-0.6, -0.5, 0)$. In contrast, $(1, 0, 0)$ represent a dangerous cancer saddle point.

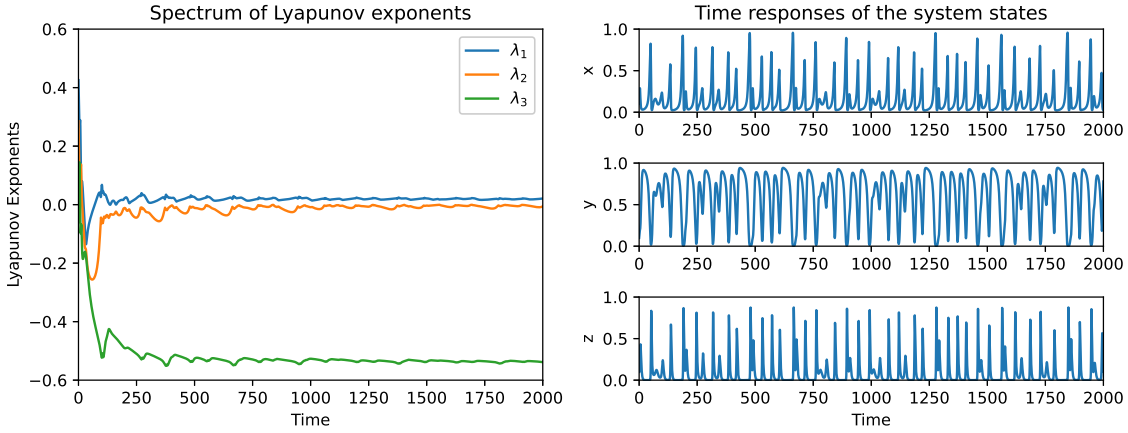

Figure 2: Lyapunov and time

Not only does figure 1 show the chaotic attractor but it is also evident from the state-time responses in figure 2. The solutions of $x(t)$, $y(t)$, $z(t)$ are oscillating and unpredictable but there is some consistency to it as well. For instance when there is a spike in tumor cells, healthy cells drop rapidly. This results in the rapid growth of immune anti-tumor cells, from which the healthy cell population recover and the tumor cell population decay. This process in then repeated indefinitely.

The Lyapunov exponents are computed in order to measure the chaotic behaviour of the system. Because this model is three-dimensional it will have a set of three Lyapunov exponents [3]. Each exponent corresponds to the average stretching or folding of phase space in a direction. It is calculated by assuming an arbitrary trajectory $x(t)$ in the phase space has a nearby trajectory $x(t) + \delta(t)$, where $\delta(t)$ is a infinitesimal vector. The maximal Lyapunov exponent λ is then given by $|\delta(t)| \approx |\delta(0)|e^{\lambda t}$ [4]. Chaos is detected if the largest Lyapunov exponent is positive. A negative largest Lyapunov value indicates non-chaotic behaviour. The Lyapunov spectrum is computed in Matlab and are $\lambda_{1,2,3} = 0.0208, -0.0062, -0.5384$. The largest exponent $\lambda_1 = 0.0208$ is positive proving that the model is chaotic. The Lyapunov spectrum is plotted in figure 2 showing the chaotic dynamics. It is apparent that a_{12} needs to be further studied because of its large impact on the system.

3 Modifying Parameter

One of the more important parameter is the bifurcation parameter a_{12} that determines the rate of tumor cells killed by healthy host cells. The effect of this parameter is investigated further by systemically changing it and studying the system's response. As before the initial states are $(x_0,\, y_0,\, z_0) = (0.1,\, 0.1,\, 0.1)$, ensuring that all cell populations are equal which seems biologically justified in the simulations.

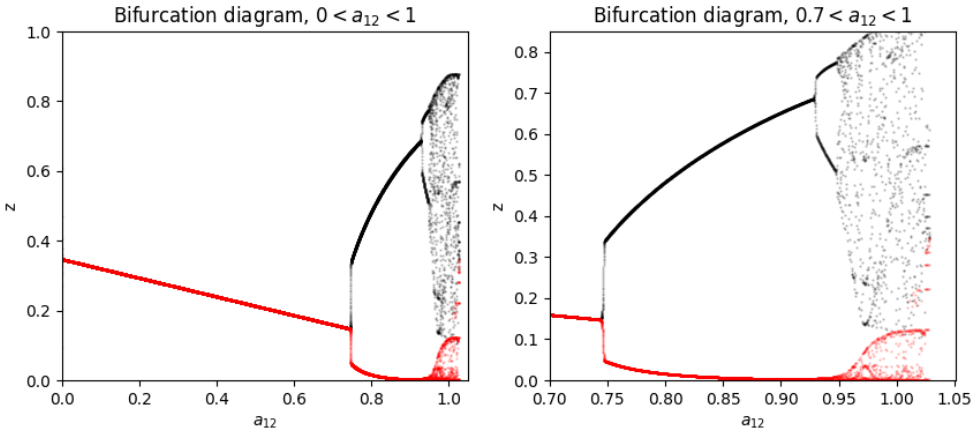

Figure 3: Bifurcation parameter a_{12} for the immune cells $z(t)$ that inactivate tumor cells. Multiple bifurcations resulting in chaos.

The obtained bifurcation diagram in figure 3 shows the different dynamics related to the variation of a_{12}. The figure shows that the system has a one-point attractor until $a_{12} = 0.75$, hence for $0 \leq a_{12} < 0.75$ the system converges to a stable 1-point limit cycle or non-zero tumor cell equilibrium state. This phenomenon is also visible in figures 4.a and 5.a when $a_{12} = 0.50$. Increasing a_{12} further results in a period doubling bifurcation at $a_{12} = 0.75$. The dynamical behaviour is a 2-point limit cycle as shown in 4.b and 5.b. The difference between $a_{12} < 0.75$ and $a_{12} \geq 0.75$ is that the latter case results in periodic solutions. As seen in figure 5.b the solutions of the three different cell populations never converge to a point. While the former case of $a_{12} < 0.75$ the solutions actually do converge to a steady state after some time. Hence proving the coexistence of the different cell populations. The dynamics change again when $a_{12} \geq 0.92$. For $a_{12} = 0.92$ a high-periodic limit cycle is visible and at $a_{12} = 0.95$ the signs of chaos are apparent. The solutions in figure 6 are also of oscillating and chaotic nature. When $a_{12} = 1$ the chaotic attractor is obtained, as studied earlier in the project. This means that two, very near solutions will diverge in an exponential order after a long time.

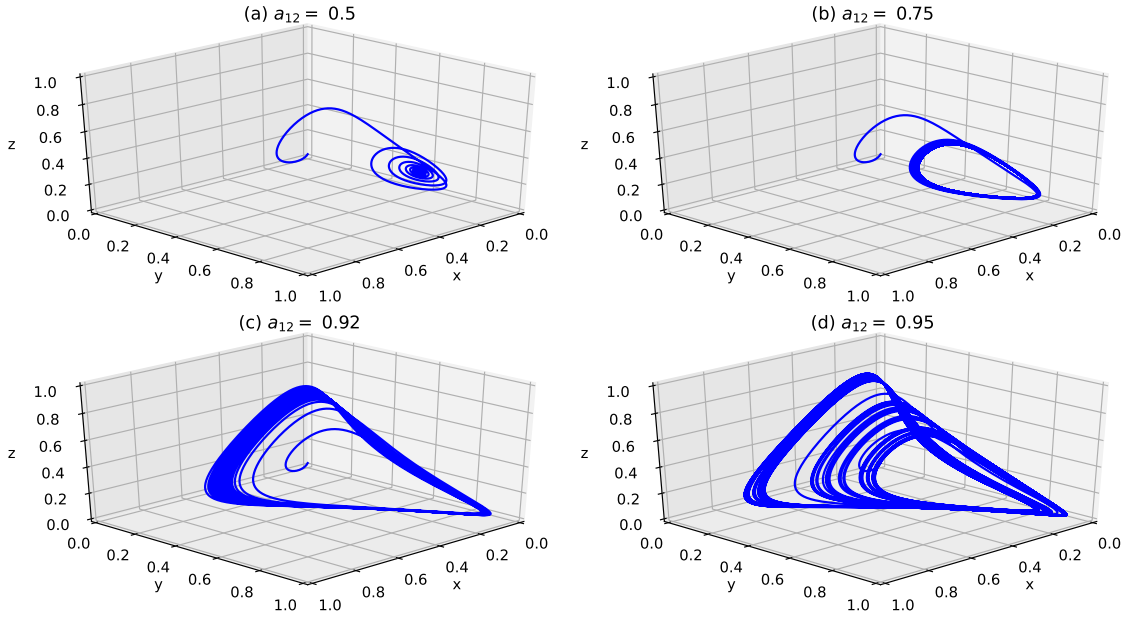

Figure 4: Phase spaces while a_{12} is varied. (a) 1-point limit cycle, convergence. (b) 2-point limit cycle. (c) Chaotic high-periodic limit cycle. (d) Chaotic high-periodic limit cycle.

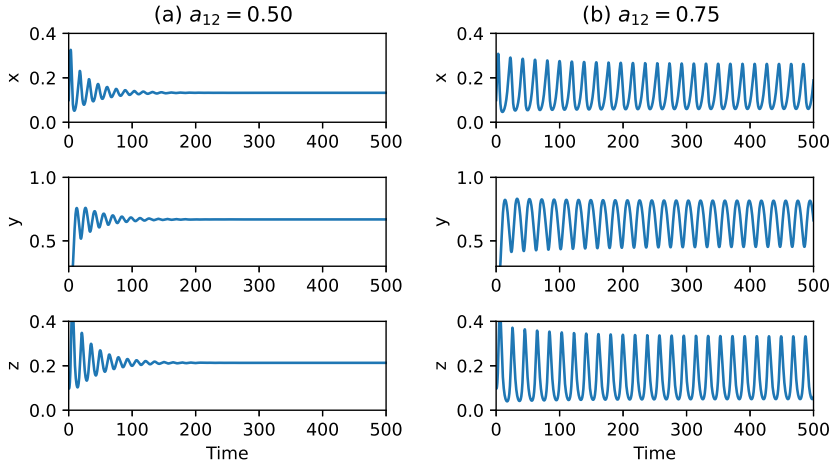

Figure 5: (a) Solutions for $a_{12} = 0.50$. (b) Oscillating solutions for $a_{12} = 0.75$.

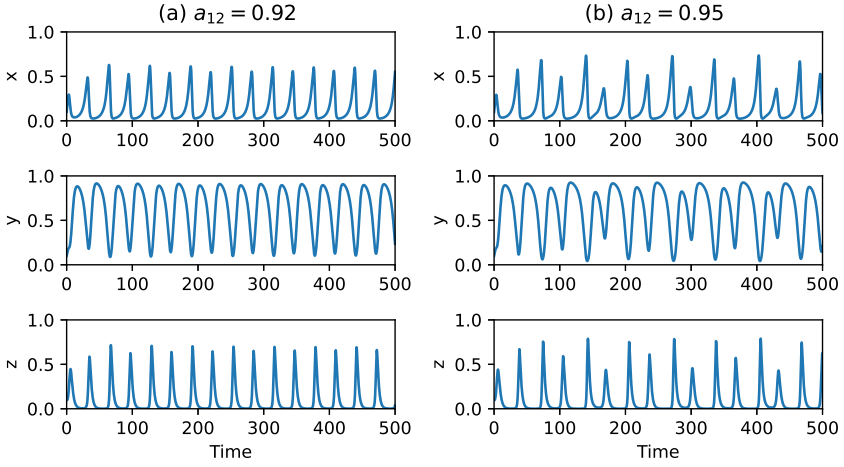

Figure 6: (a) Chaotic solutions for $a_{12} = 0.92$. (b) Chaotic solutions for $a_{12} = 0.95$.

One interesting behavior happens when $a_{12} > 1$ where all system solutions converge to a healthy and tumor-free equilibrium point. This transition is important because tumor growth is avoided which realistically is desirable. Although, as the aim of this project was to study a chaotic model, this transition was initially ignored as it is non-chaotic. Nonetheless it is still of interest in order to understand the different dynamics of the model.

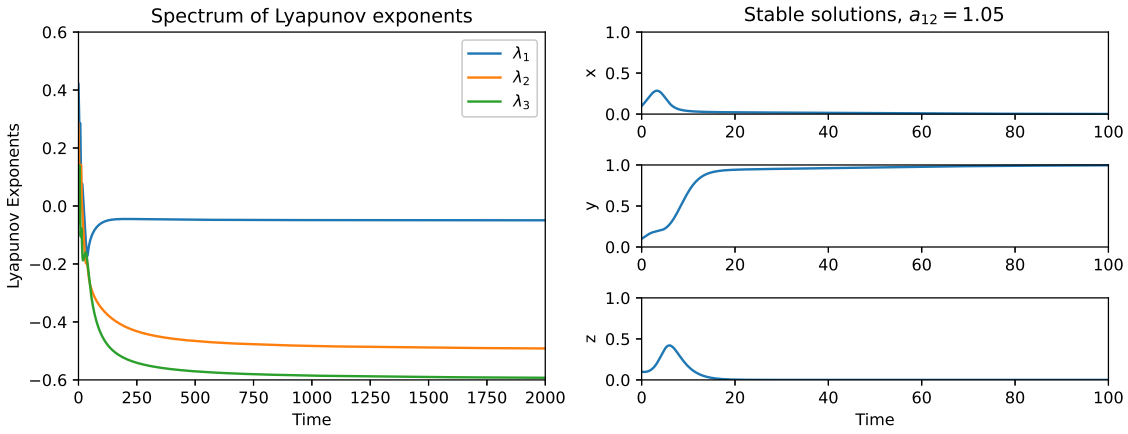

Figure 7: Stable solutions for $a_{12} = 1.05$. Lyapunov spectrum indicates non-chaotic behaviour as do the solutions.

The solutions of $x(t)$, $y(t)$ and $z(t)$ in figure 7 shows convergence to a stable equilibrium point without a tumor cell population. This equilibrium point $(0, 1, 0)$ was mentioned in the first section and the eigenvalues are $\lambda_{1,2,3} = (-r_2, -d_3, 1 - a_{a12}) = (-0.6, -0.5, -0.05)$. The eigenvalues are all negative indicating its stability. The Lyapunov exponents are calculated to be $\lambda_{1,2,3} = -0.0494, -0.4915, -0.5926$ (not to be confused with the eigenvalues of the equilibrium). The largest Lyapunov exponent is negative confirming that it is not chaotic at all.

4 Extension

The tumor growth model in (1)-(3) can also be extended in a way that includes drug therapy. For instance chemotherapy, which destroys cancer cells or slows their growth. Not only is tumor growth modelling important in cancer research but so is

its treatment. This section discusses an extension to the tumor growth model that involves drug therapy, targeting the tumor cell population. This extension is similar to the tumor growth and treatment model proposed in [5]. Although, in this extension, the drug therapy only kills cancer cells and not healthy or immune cells unlike in [5]. Hence it is assumed that the drug therapy does no harm to healthy cells which might not be entirely realistic. Often treatments such as chemotherapy does in fact harm or kill healthy cells.

In order to extend the original tumor growth model, a fourth drug therapy term is added to the cancer cell population in equation (1).

$$\frac{dx}{dt} = x(1-x) - a_{12}xy - a_{13}xz - x(1-e^{-u(t)})a \tag{9}$$

Here $(1-e^{-u(t)})a$ is an exponential function of tumor cell kill depending on the amount of drug $u(t)$ at time t. The kill rate is given by the parameter a. Further, a differential equation (13) for $u(t)$ is added as well. The extended tumor growth model with treatment becomes

$$\frac{dx}{dt} = x(1-x) - a_{12}xy - a_{13}xz - x(1-e^{-u(t)})a, \tag{10}$$

$$\frac{dy}{dt} = r_2y(1-y) - a_{21}xy, \tag{11}$$

$$\frac{dz}{dt} = \frac{r_3xz}{x+k_3} - a_{31}xz - d_3z, \tag{12}$$

$$\frac{du}{dt} = V - d_2u. \tag{13}$$

V is the given dose of the drug for treatment and d_2 determines the decay rate of the injected drug. The general idea is for the drug to suppress the tumor cell population. The cell population in the time series of figure 2 reaches dangerously high levels. When $x(t)$ reaches 1 it means that the tumor cells kills the healthy cells which should be avoided by this drug therapy treatment.

The tumor growth model with treatment is simulated with the following additional parameters

$$a = 0.15, \quad V = 0.01, \quad d_2 = 1.5. \tag{14}$$

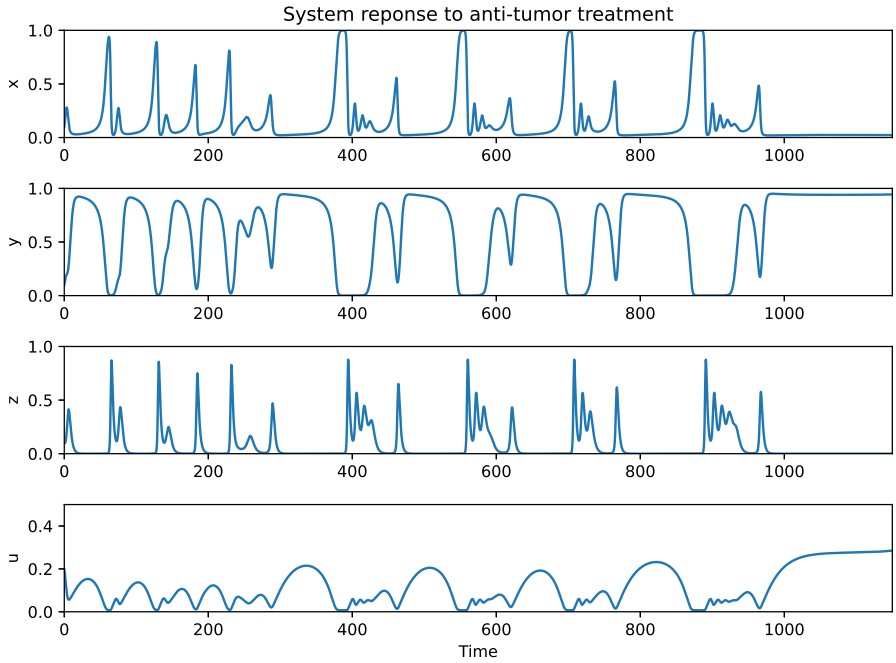

Figure 8: Solutions appear chaotic until the amount of drug $u(t)$ is administered constantly. Importantly the tumor cell population $x(t)$ goes toward 0 at $t = 1000$.

Figure 8 illustrates the responses of the tumor, healthy and immune cell populations with respect to the drug therapy. It is evident that the drug suppresses the tumor cell population. The tumor cell population $x(t)$ rapidly decreases when the amount of drugs $u(t)$ spikes. The drug administration along with the other solutions appears chaotic until the drug is constantly administered. After which the system converges to a tumor-free and healthy solution, which was the desired effect.

5 Conclusions

In this project, a model for the evolution of cancer cells have been analyzed and extended. The tumor growth model initially proposed in [2] is based on competition dynamics. The different behaviours of the model, such as, chaotic attractors, limit cycles and equilibrium points have been carefully studied with an bifurcation analysis. It was concluded that the model is chaotic when the bifurcation parameter is $a_{12} = 1$ by calculating the Lyapunov exponent. For $a_{12} > 1$

the system converges to a tumor-free solution which was not covered in [2]. Despite that it is not chaotic, it is highly relevant because it is tumor-free and the healthy cells prevail. It was also shown that for $a_{12} < 1$ the system has periodic solutions, hence the existence of limit cycles was also confirmed. This project also extended the model in such way to add a treatment equation for the model. The results of the treatment were at first chaotic but then the tumor cells were killed and converged to a stable tumor-free solution. This extension was inspired by [5], however in this case the treatment only accepted the tumor cells and not the healthy cells. While the sophisticated approach in [5] used complex control theory to administer the drug, the presented approach in this projected worked well. One could argue of the realism of such treatment targeting only tumor cells, but this might actually be possible in the future. Hence, it is relevant to this application area. The treatment model could of course be improved to avoid the chaotic behaviour. Noteworthy is that [5] did not analyze the treatment during chaotic behaviour which should have been relevant because it is one of the fundamental dynamics of the system. To conclude, from a biological viewpoint, the study of the chaotic behaviour is this model is highly relevant since it may correspond to the chaotic growth of tumor cells in a human body.

References

[1] Ferlay J, Ervik M, Lam F, Colombet M, Mery L, Piñeros M, et al. "Global Cancer Observatory: Cancer Today". (Lyon: International Agency for Research on Cancer; 2020). June, 2021. `gco.iarc.fr/today`

[2] Itik, M. and S. Banks. "Chaos in a Three-Dimensional Cancer Model." (Int. J. Bifurc. Chaos 20, 2010), 71-79.

[3] J. C. Sprott. "Chaos and Time-Series Analysis". (Oxford University Press, 2003), 116-117.

[4] Strogatz, Steven H. 1994. "Nonlinear dynamics and Chaos: with applications to physics, biology, chemistry, and engineering". (Reading, Mass: Addison-Wesley Pub 1994), 320-325.

[5] L.G De Pillis, A Radunskaya. "The dynamics of an optimally controlled tumor model: A case study". (Mathematical and Computer Modelling, Volume 37, Issue 11, 2003), 1221-1244.

A Python Code

Listing 1: Main model function.

```
from pylab import *
from mpl_toolkits.mplot3d import Axes3D
import seaborn as sns

###### Model
Dt = 0.001 # Size of time step
a12 = 1.05 # Chaotic behaviour if a12 = 1.
a13 = 2.5
a21 = 1.5
a31 = 0.2
r2 = 0.6
r3 = 4.5
k3 = 1
d3 = 0.5


def initialize(x0, y0, z0):
    global x, xresult, y, yresult, z, zresult, t, timesteps
    x = x0
    y = y0
    z = z0
    xresult = [x]
    yresult = [y]
    zresult = [z]
    t = 0.
    timesteps = [t]

def observe():
    global x, xresult, y, yresult, z, zresult, t, timesteps
    xresult.append(x)
    yresult.append(y)
    zresult.append(z)
    timesteps.append(t)

def update():
```

```
    global x, xresult, y, yresult, z, zresult, t, timesteps

    nextx = x + (x - x*x - a12*x*y - a13*x*z) * Dt
    nexty = y + (r2*y - r2*y*y - a21*x*y) * Dt
    nextz = z + ((r3*x*z)/(x+k3) - a31*x*z - d3*z) * Dt

    x, y, z = nextx, nexty, nextz
    t = t + Dt

##### Simulate
#initialize(0.5, 0.1, 0.1)

initialize(0.1, 0.1, 0.1)
while t < 2000:
  update()
  observe()


#### Plotting
fig = plt.figure(figsize=(7,7))
ax = fig.add_subplot(2,2,1, projection='3d')
ax.plot(xresult, yresult, zresult, "b")
ax.view_init(30, 45)
ax.set_xlabel("x")
ax.set_ylabel("y")
ax.set_zlabel("z")
#ax.set_title("Phase Space")
ax.set_xlim(0, 1)
ax.set_ylim(0, 1)
ax.set_zlim(0, 1)
ax1 = fig.add_subplot(2,2,2)
ax1.plot(xresult, yresult, "b")
xlabel("x")
ylabel("y")
plt.xlim([0, 1])
plt.ylim([0, 1])
ax1.grid(True)
ax1 = fig.add_subplot(2,2,3)
```

```
ax1.plot(xresult, zresult, "b")
xlabel("x")
ylabel("z")
plt.xlim([0, 1])
plt.ylim([0, 1])
ax1.grid(True)
ax1 = fig.add_subplot(2,2,4)
ax1.plot(yresult, zresult, "b")
xlabel("y")
ylabel("z")
plt.xlim([0, 1])
plt.ylim([0, 1])
ax1.grid(True)
fig.tight_layout()
#plt.savefig('phases_c.eps', format='eps')
plt.show()
```

Listing 2: Extended model.

```
from pylab import *
from mpl_toolkits.mplot3d import Axes3D
import seaborn as sns

Dt = 0.001 # Size of time step
a12 = 1 # Chaotic behaviour if a12 = 1.
a13 = 2.5
a21 = 1.5
a31 = 0.2
r2 = 0.6
r3 = 4.5
k3 = 1
d3 = 0.5
p = 1.1

def initialize(x0, y0, z0):
    global x, xresult, y, yresult, z, zresult, t, timesteps, u, uresult
    x = x0
    y = y0
    z = z0
```

```
    xresult = [x]
    yresult = [y]
    zresult = [z]

    u = 0.2
    uresult = [u]

    t = 0.
    timesteps = [t]

def observe():
    global x, xresult, y, yresult, z, zresult, t, timesteps, u, uresult
    xresult.append(x)
    yresult.append(y)
    zresult.append(z)
    uresult.append(u)
    timesteps.append(t)

def update():
    global x, xresult, y, yresult, z, zresult, t, timesteps, u, uresult

    nextx = x + (x - x*x - a12*x*y - a13*x*z - x*(0.15*(1-exp(-u)))) * Dt
    nexty = y + (r2*y - r2*y*y - a21*x*y) * Dt
    nextz = z + ((r3*x*z)/(x+k3) - a31*x*z - d3*z) * Dt
    nextu = u + (0.01-1.5*u*x) * Dt

    x, y, z, u = nextx, nexty, nextz, nextu
    t = t + Dt

####### PLOT
initialize(0.1, 0.1, 0.1)
while t < 2000:
  update()
  observe()
fig=plt.figure(figsize=(12,6))
plt.subplot(411)
plt.plot(timesteps, xresult)
plt.xlim([0, 2000])
```

```
plt.ylim([0, 1])
ylabel("x")
plt.title("Time responses of the system states")
plt.subplot(412)
plt.plot(timesteps, yresult)
plt.xlim([0, 2000])
plt.ylim([0, 1])
ylabel("y")
plt.subplot(413)
plt.plot(timesteps, zresult)
plt.xlim([0, 2000])
plt.ylim([0, 1])
xlabel("Time")
ylabel("z")
plt.subplot(414)
plt.plot(timesteps, uresult)
plt.xlim([0, 2000])
plt.ylim([0, 1])
xlabel("Time")
ylabel("u")
plt.tight_layout()
#plt.savefig('more.eps', format='eps')
plt.show()
```

Listing 3: Bifurcation diagram.

```
from pylab import *
from mpl_toolkits.mplot3d import Axes3D
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt

a13 = 2.5
a21 = 1.5
a31 = 0.2
r2 = 0.6
r3 = 4.5
k3 = 1
d3 = 0.5
```

```
14
15 def cancer_model(x, y, z, a12):
16     x_dot = x - x*x - a12*x*y - a13*x*z
17     y_dot = r2*y - r2*y*y - a21*x*y
18     z_dot = (r3*x*z)/(x+k3) - a31*x*z - d3*z
19     return x_dot, y_dot, z_dot
20
21
22 dr = 0.001  # parameter step size
23 a12 = np.arange(0, 3, dr)  # parameter range
24 dt = 0.001  # time step
25 t = np.arange(0, 50, dt)  # time range
26
27 # Solution arrays
28 xs = np.empty(len(t) + 1)
29 ys = np.empty(len(t) + 1)
30 zs = np.empty(len(t) + 1)
31
32 # Initial values
33 xs[0], ys[0], zs[0] = (0.1, 0.1, 0.1)
34
35 z_maxes = []
36 r_mins = []
37 z_mins = []
38
39 for R in a12:
40     for i in range(len(t)):
41         # Approximate solutions
42         x_dot, y_dot, z_dot = cancer_model(xs[i], ys[i], zs[i], R)
43         xs[i + 1] = xs[i] + (x_dot * dt)
44         ys[i + 1] = ys[i] + (y_dot * dt)
45         zs[i + 1] = zs[i] + (z_dot * dt)
46     # Compute z
47     for i in range(1, len(zs) - 1):
48         # Max
49         if zs[i - 1] < zs[i] and zs[i] > zs[i + 1]:
50             r_maxes.append(R)
51             z_maxes.append(zs[i])
```

```
        # Min
        elif zs[i - 1] > zs[i] and zs[i] < zs[i + 1]:
            r_mins.append(R)
            z_mins.append(zs[i])

    # Use final values as next innitial values
    xs[0], ys[0], zs[0] = xs[i], ys[i], zs[i]
# Plot

plt.scatter(r_maxes, z_maxes, color="black", s=0.5, alpha=0.2)
plt.scatter(r_mins, z_mins, color="red", s=0.5, alpha=0.2)
plt.xlim(0, 1)
plt.ylim(0, 1)
plt.show()
```

B Matlab Code

Listing 4: Run function.

```
[T, Res] = lyapunov(3, @cancer, @ode45, 0, 0.5, 1000, [0.1 0.1
   0.1], 10);
plot(T,Res);
% Plot
title('Spectrum of Lyapunov exponents');
xlabel('Time');
ylabel('Lyapunov exponents');
```

Listing 5: Model definition.

```
function f = cancer(t, X)

a12 = 0.73;
a13 = 2.5;
a21 = 1.5;
a31 = 0.2;
r2 = 0.6;
r3 = 4.5;
k3 = 1;
d3 = 0.5;
```

```

x = X(1); y = X(2); z = X(3);
Y = [X(4), X(7), X(10);
    X(5), X(8), X(11);
    X(6), X(9), X(12)];
f = zeros(9,1);

% Model equations
f(1)=x - x*x - a12*x*y - a13*x*z;
f(2)=r2*y - r2*y*y - a21*x*y;
f(3)=(r3*x*z)/(x+k3) - a31*x*z - d3*z;


A = 1-2*x-a12*y-a13*z;
B = -a12*x;
C = -a13*x;
D = -a21*y;
E = r2-2*r2*y-a21*x;
F = (r3*z)/(x+k3)*(1-(x/(x+k3)))-a31*z;
G = (r3*x)/(x+k3)-a31*x-d3;

% Jacobian
 Jac=[A,    B,    C;
      D,    E,    0;
      F,    0,    G];

% Variational form
f(4:12) = Jac * Y;
```

Listing 6: Function to compute Lyapunov exponents.

```
function [Time_exp, Lya_exp] = lyapunov(n, func, ode_45,
   time_start, step_size, time_end, l_start, out);

% Lyapunov calculation
% A. Wolf, J. B. Swift, H. L. Swinney, and J. A. Vastano,
% "Determining Lyapunov Exponents from a Time Series".

n1 = n;
```

```
n2 = n1*(n1+1);

% Number steps
step = round((time_end-time_start)/step_size);

% Allocation
y = zeros(n2,1);
mem = zeros(n1,1);
y0 = y;
gsc = mem;
znorm = mem;

y(1:n) = l_start(:);
for i = 1:n1 y((n1+1)*i) = 1.0; end;
% Start time
t = time_start;

% Main loop
for lya_index = 1:step
  [T,Y] = feval(ode_45,func,[t t+step_size],y);

  t = t+step_size;
  y = Y(size(Y,1),:);
  for i = 1:n1
      for j = 1:n1 y0(n1*i+j)=y(n1*j+i); end;
  end;

% Orthonormal basis, Gram-Schmidt:
  znorm(1) = 0.0;
  for j=1:n1 znorm(1) = znorm(1)+y0(n1*j+1)^2; end;
  znorm(1) = sqrt(znorm(1));
  for j = 1:n1 y0(n1*j+1) = y0(n1*j+1)/znorm(1); end;
  for j = 2:n1
      for k = 1:(j-1)
          gsc(k) = 0.0;
          for l = 1:n1 gsc(k) = gsc(k) + y0(n1*l+j) * y0(n1*l+k);
   end;
      end;
```

```
45
46        for k = 1:n1
47            for l = 1:(j-1)
48                y0(n1*k+j) = y0(n1*k+j) - gsc(l)*y0(n1*k+l);
49            end;
50        end;
51        znorm(j) = 0.0;
52        for k = 1:n1 znorm(j) = znorm(j) + y0(n1*k+j)^2; end;
53        znorm(j) = sqrt(znorm(j));
54        for k=1:n1 y0(n1*k+j) = y0(n1*k+j) / znorm(j); end;
55    end;
56
57  % Update
58    for k=1:n1 mem(k) = mem(k) + log(znorm(k)); end;
59
60  %Normalization of exponent
61    for k=1:n1
62        lp(k) = mem(k)/(t-time_start);
63    end;
64
65  % Output
66    if lya_index == 1
67       Lya_exp = lp;
68       Time_exp = t;
69    else
70       Lya_exp = [Lya_exp; lp];
71       Time_exp = [Time_exp; t];
72    end;
73
74    % Print result
75    if (mod(lya_index,out == 0))
76       fprintf('t=%6.4f',t);
77       for k=1:n1 fprintf(' %10.6f',lp(k)); end;
78       fprintf('\n');
79    end;
80    for i = 1:n1
81        for j = 1:n1
82            y(n1*j+i) = y0(n1*i+j);
```

```
83        end;
84    end;
85 end;
```