Tumor Growth Model

Introduction

Cancer growth and treatment dynamics can be described using ordinary differential equations (ODEs), which model how tumor cells proliferate, respond to chemotherapy, and die over time. Such models are widely used in pharmacometrics and oncology simulation research because they provide mechanistic tools for predicting tumor burden under different treatment strategies.

In this project we will use a 3-phase system to model how tumor cells interact with a general chemotherapeutic drug. The first phase represents the population of viable tumor cells, which grow exponentially. The second phase consists of viable tumor cells being hit/weakened by the drug. This phase is supposed to represent the delay between drug exposure and cell death. The 3rd phase is where tumor cells are dying/dead and are then removed from the system over time.

This main reference for this model is *A Review of Mathematical Models for Tumor Dynamics and Treatment Resistance Evolution of Solid Tumors*, a report published in 2019. The ODE system and constant values used will be based off the ones used in this study. The graphs in the study will be used as a reference point to ensure accuracy of the model and the generated graphs.

Ordinary Differential Equations

$$\begin{cases} \frac{dS_1}{dt} = f(S_1) - k_d \,\text{Exposure}\, S_1 \\ \frac{dS_2}{dt} = k_d \text{Exposure} S_1 - k_d S_2 \\ \frac{dD}{dt} = k_d \,\text{Exposure}\, S_2 - d\, D \\ T = S_1 + S_2 \end{cases}$$

Variables:

- T(t): Total tumor size
- S_1(t): Viable (alive) tumor cells
- S_2(t): Tumor cells affected by the drug
- D(t): Dying/dead cells that have been hit by the drug but not yet cleared from the tumor
- f(S_1): Growth function, either exponential or logistic growth
- k_g: Tumor growth rate constant, found in f(S_1)
- k_d: Drug kill coefficient, Controls how strongly the drug converts viable cells S into damaged cells D
- Exposure: Drug exposure or effect over time, can be changed to vary over time
- d: Rate at which damaged/dead cells D are removed from the tumor volume

Drug exposure will be calculated using Hill's equation found in the tumor dyanmics study. Hills Equation:

$$Exposure = E_{max} \cdot \frac{t^{0.5}}{10^{0.5} + t^{0.5}}$$

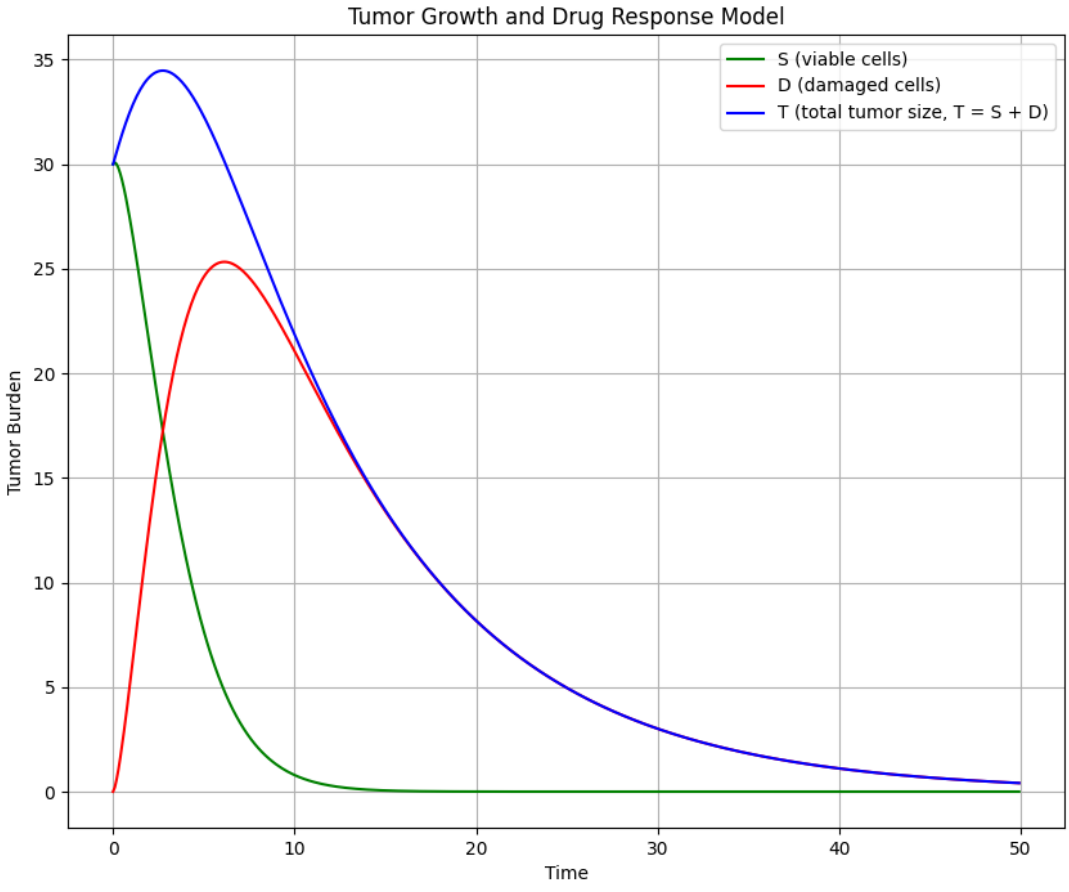
Implementation

All numerical calculations and visualizations for this project are implemented in C++. The system of ordinary differential equations is solved using the fourth-order Runge–Kutta (RK4) method, which provides a stable and accurate approach for integrating nonlinear biological models. Visualization of the results is performed using the matplotlibcpp library, allowing the computed tumor and drug-response trajectories to be plotted directly from the C++ environment using Python's Matplotlib. The libary can be found here, https://github.com/lava/matplotlib-cpp The program will be tested under several conditions to verify that the model behaves as expected. Under no drug exposure the tumor should follow pure logistic growth, increasing smoothly until it approaches the carrying capacity. Under constant drug exposure, viable cells (S_1) should decrease almost instantly while damaged cells (D) initially rise and then decay as they clear, leading to a drop in total tumor size. Additionally we can add varying drug exposure to see how the model performs.

Demo

Before building the final model, the demo used this system of ODEs and used constant variables $k_g = 0.1$, $k_d = 0.04$, $d = 0.1$. Initial S (viable cells) is set to 30.0. Time step is 0.1. $E_{max} = 30$. The demo uses an exponential growth function, $f(S_1) = k_g S$

$$\begin{cases} \dfrac{dS}{dt} = k_g S - k_d \,\text{Exposure}\, S \\ \dfrac{dD}{dt} = k_d \,\text{Exposure}\, S - d\, D \\ T = S + D \end{cases}$$

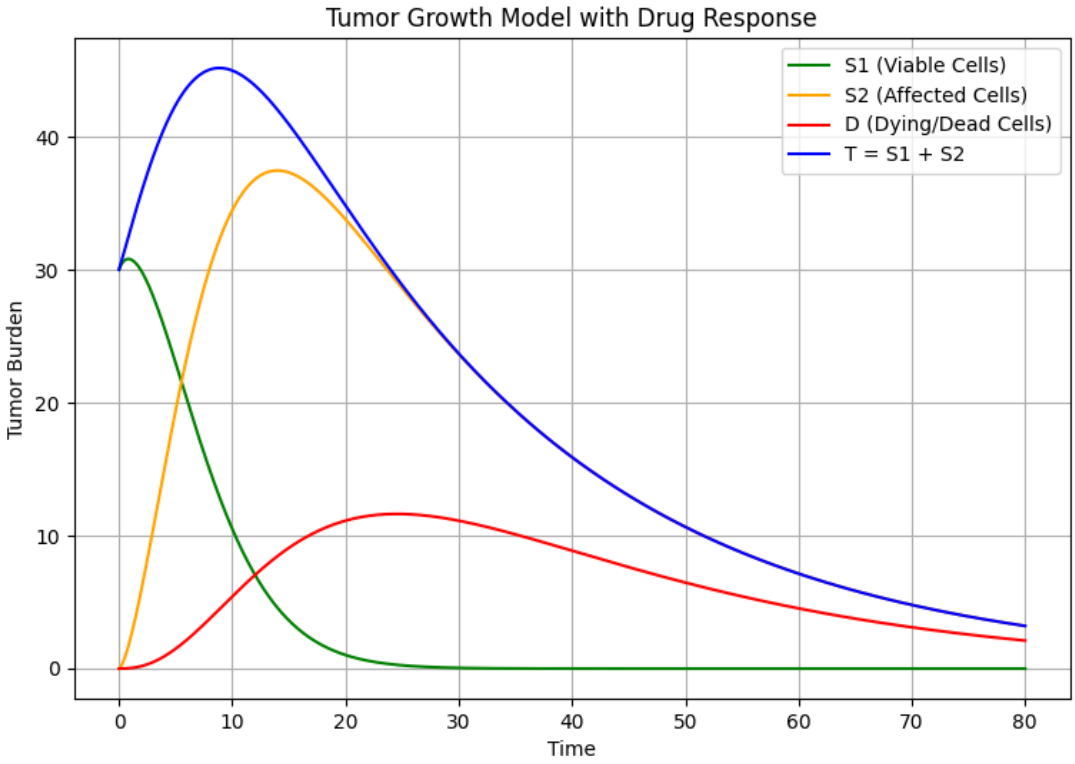


The total tumor size curve matches the one found in Figure 2, model 4 of the Tumor Dynamics study

Main Model

Below is the graph of the model using the system of ODEs first mentioned. All constant values remain the same from the demo, but tumor growth is calculated using logistic growth as opposed to exponential growth. Logistic growth more accurately reflects how tumors actually grow. The logistic growth function is defined below. T_{max} is the carrying capacity, with $T_{max} = 120.0$.

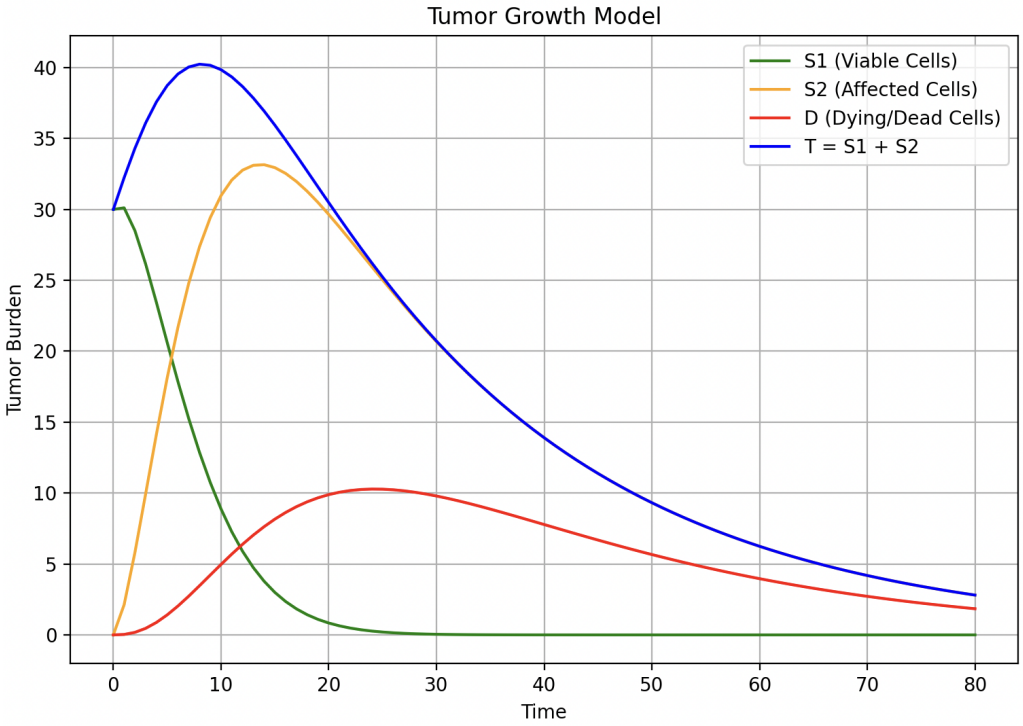
$$f(S_1) = k_g S_1 \left(1 - \frac{S_1}{T_{\max}}\right)$$



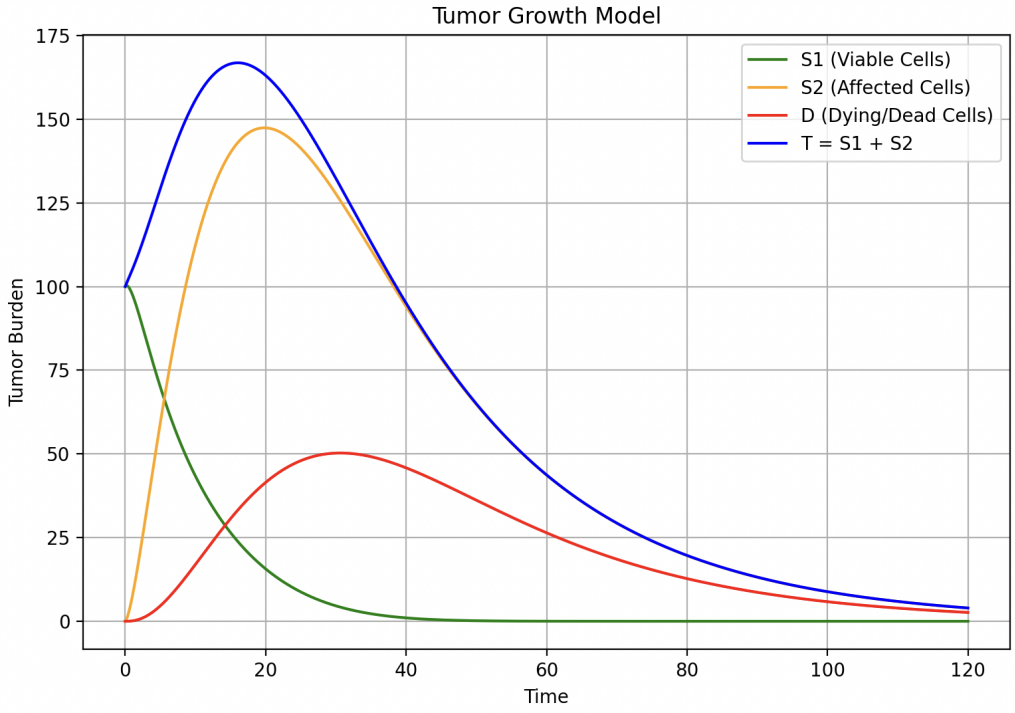
At t = 0, S_1 viable cells begins to grow but quickly declines as exposure also increase with time. At the same time S_2 affected cells increases quickly, indicating the drugs effect. S_2 reaches a maximum when enough viable cells have been converted, but also are near depletion. We can see how the T and S_2 converge together because there are no more viable cells, and all tumor cells have been hit by the drug. Dead cells rise steadily and peak after S_2 peaks, then steadily fall off, showing the dead cells slowly leaving the system. Near the end of the simulation, all the curves are nearly flat or are flat, reflecting the diminishing rates of cell kill and clearance as the tumor population becomes very small.

Tests & Exploration

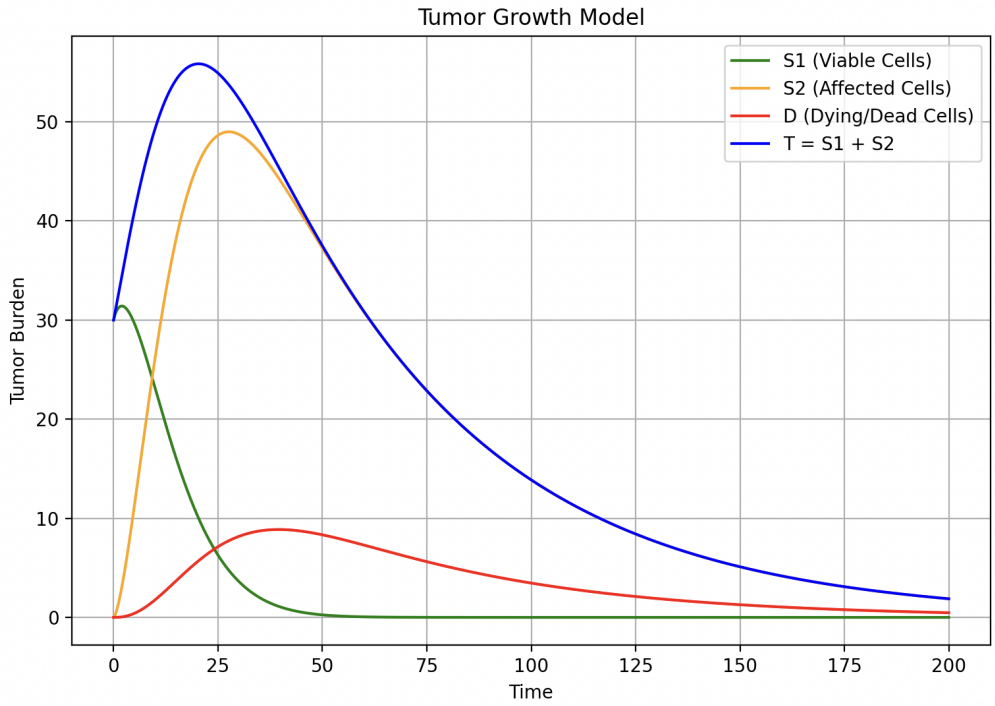
To double check our model, lets set the time step to 1 from 0.1. The graph is generally the exact same which is, expected as our values are not changing very quickly. Although some detail was lost on the S_1 at the start. S_1 rises very briefly and quickly at the start but that cannot be shown here as clearly.



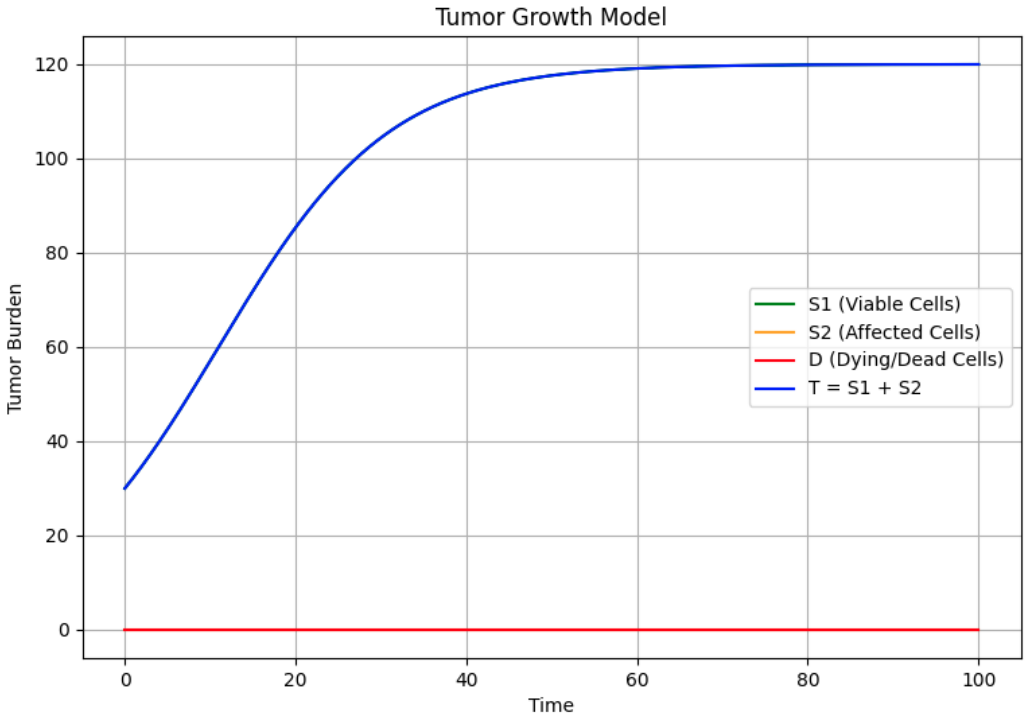
Here we set the initial S_1 to 100 as opposed to 30 and increase the growth rate constant k_g to 0.3, to ensure our model follows the same behavior. As shown it is still very similar to our original graph. The total tumor size reaches a much larger maximum value and it takes slightly longer for it to begin decreasing.



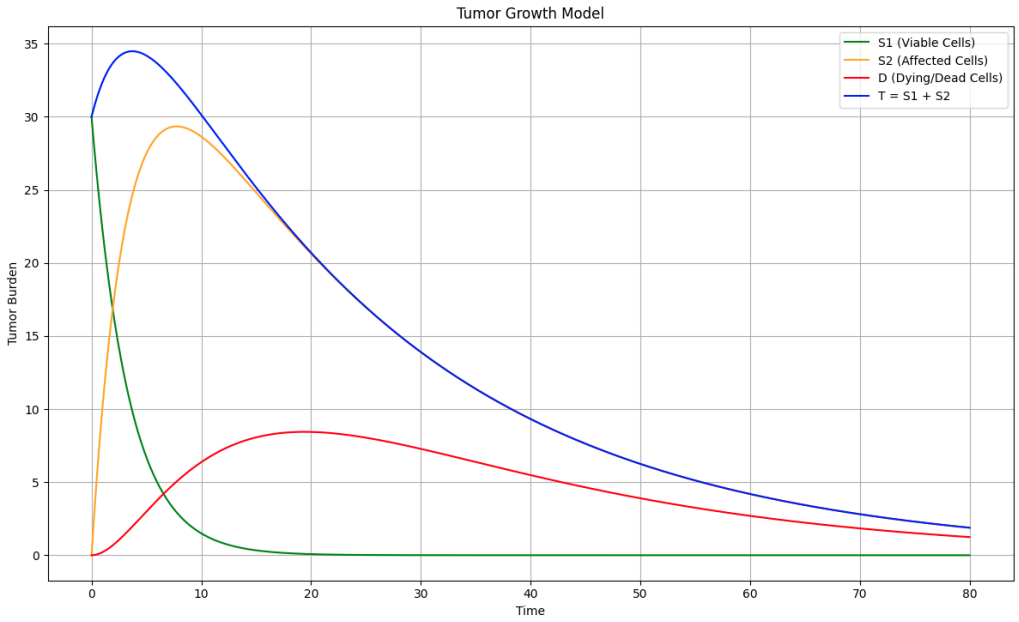
Below we halve the drug kill coefficient, k_d from 0.4 to 0.2. The bounds for the x-axis must be increased to 200 to show the full behavior. The viable cells are still affected very quickly, but it takes much longer to convert them into dying cells. This makes sense as we did not change exposure but only the strength of the drug.



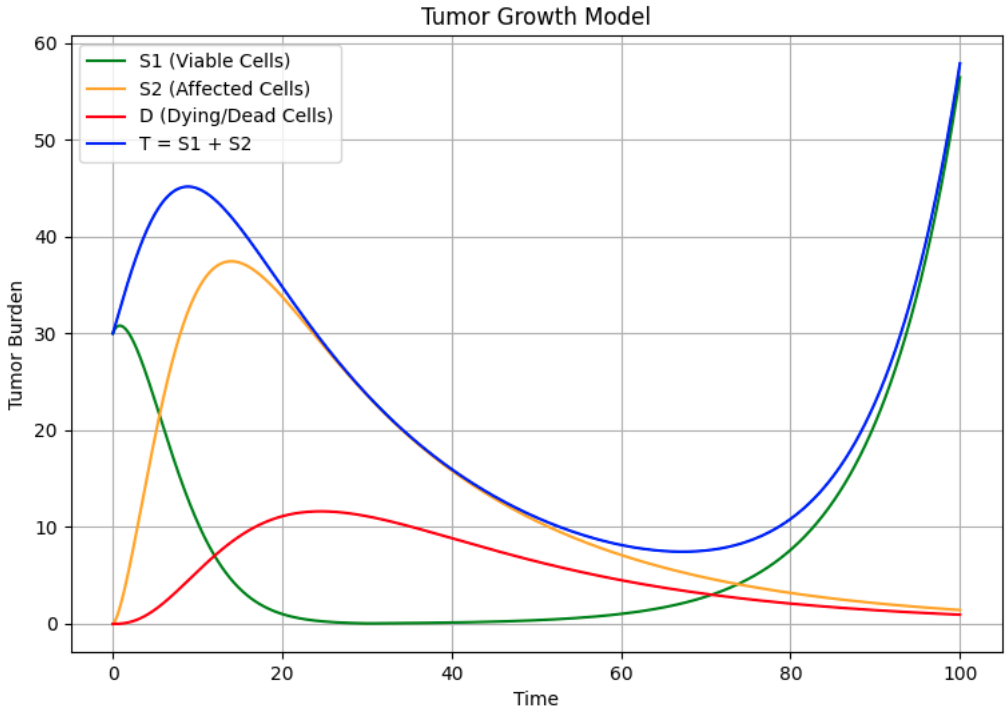
Below is how the model behaves if we set drug exposure to 0. There will be no affected or dying cell populations and S_1 and T will overlap. Without exposure the model simulates standard logistic growth. Only our first equation will have changing values and the other ones will remain 0. Since for our logistic growth function, $T_{max} = 120.0$, the growth will approach the limit 120.



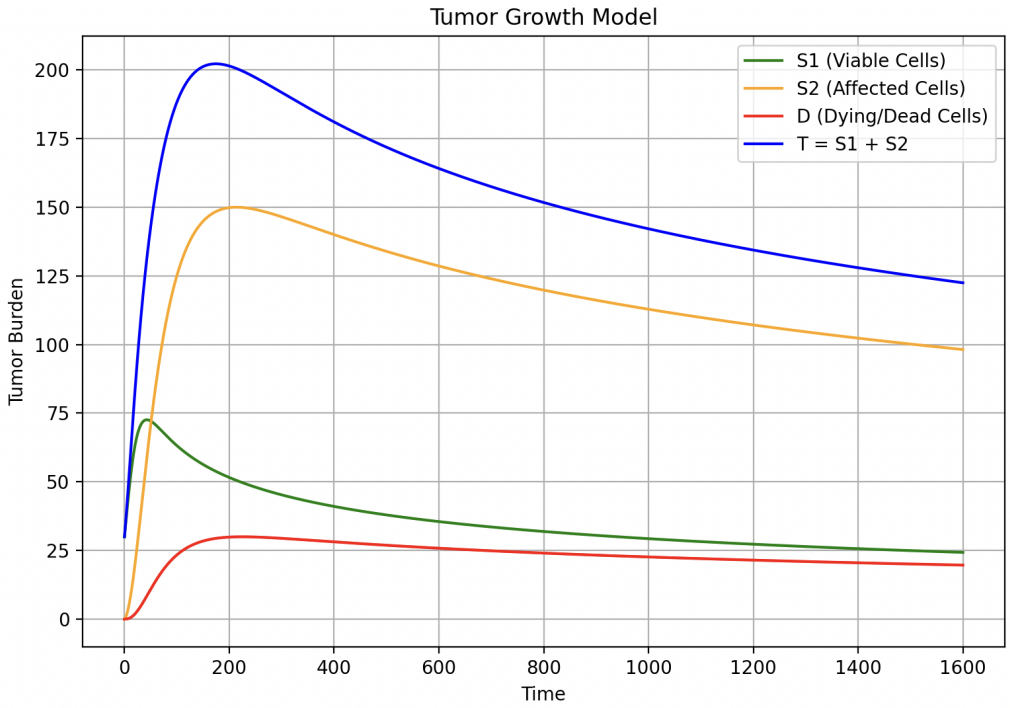
Below is how the model behaves when drug exposure is set to a constant value of 10. We can see how the viable cells are dealt with much quicker and the overall tumor size begins to decrease much quicker. This is the expected result. It's important to note that constant drug exposure is not practical. Real treatments will vary levels of drug doses and it is biologically incorrect to ignore drug absorption, metabolism and many other factors.



Now we will simulate cutting off treatment by setting the drug exposure to 0.0 at $t = 30$. We can see how the population of viable cells begins to increase again after $t = 30$, and the population of affected cells and dead cells begins to approach 0. Near $t = 100$, the population of viable cells and total tumor size begin to converge.



Below is a scenario where the treatment might fail. The drug kill coefficient, k_d is reduced to 0.2 and the max drug exposure, E_{max}, is reduced to 5.0. The drug strength and exposure is not enough to counteract the rate at which the the tumor grows, and all curves appear to reach a limit. Although they are still steadily decreasing, it is going to take too much time to and we can conclude this treatment will not be effective.



Model Accuracy

The scope of the model is very simple and therefore misses a lot of important factors impacting tumor growth. The pipeline of viable cells to affected cells to dead cells is heavily simplified. The model also ignores factors such as the immune system, drug resistance and assumes all cells behave identically. There is a possibility for an affected cell to resist the drug and continue to live. Real tumor treatment also has a more significant delay, and dosage and drug strength is heavily dependent on specific scenarios. In the main model, we use logistic growth, but other growth functions, such as Gompertz growth or a combination of linear and exponential growth, may also be explored to see if they represent real tumor growth more accurately. It is also important to note tumor cell behaviour has a degree of randomness to it, which we do not explore here. These simplifications mean that while the model captures relative behavior, quantitative predictions may differ from reality.

Improvements

Several features and variables can be implemented to improve the models performance for various scenarios and increase real world accuracy. First, a subset of differential equations could be introduced to model drug behavior by accounting for absorption, metabolism and clearance. Another addition could be splitting tumor cells into subpopulation where some are more sensitive and some are more resistant. This greater emphasizes how tumors actually work. If available, real world patient data could be used to validate the model and improve it. This can be done automatically if implemented with C++ to automatically parse data and compare it with the mode.

References

Yin, Anyue. A Review of Mathematical Models for Tumor Dynamics and Treatment Resistance Evolution of Solid Tumors. 9 August 2019. National Library Of Medicine, https://pmc.ncbi.nlm.nih.gov/articles/PMC6813171/.

Yonsei Medical Journal. A Review of Modeling Approaches to Predict Drug Response in Clinical Oncology. 7 November 2016. National Library of Medicine, https://pmc.ncbi.nlm.nih.gov/articles/PMC5122624/.

main.cpp

```
#include <cmath>
#include <iostream>
#include <vector>

#include "matplotlibcpp.h"

namespace plt = matplotlibcpp;

// Model constants
const double k_g = 0.1;   // Tumor growth rate constant
const double k_d = 0.04;   // drug kill coefficient
const double d   = 0.10;   // dead-cell clearance rate
const double e_max = 30.0; // maximum exposure level
const double T_max = 120.0; // carrying capacity
const bool simulaute_treatment_end = false;
const double treatment_end_time = 30.0;

// Exposure function
double exposure(double t) {
    if (simulaute_treatment_end && t >= treatment_end_time){
        return 0.0;
    }
    //hills equation
    return e_max * std::pow(t, 0.5) /
           (std::pow(100.0, 0.5) + std::pow(t, 0.5));
}

struct State {
    double S1;
    double S2;
    double D;
};

double logisticGrowth(double S1) {
    return k_g * S1 * (1.0 - S1 / T_max);
}
// Exponential growth
double exponentialGrowth(double S1) {
    return k_g * S1;
}

// ODEs
State derivatives(double t, const State& y) {
    double S1 = y.S1;
    double S2 = y.S2;
    double D  = y.D;
    double exp = exposure(t);

    State dydt;
    //dydt.S1 = logisticGrowth(S1) - k_d * exp * S1;
    dydt.S1 = logisticGrowth(S1) - k_d * exp * S1;

    dydt.S2 = k_d * exp * S1 - k_d * S2;

    dydt.D  = k_d * S2 - d * D;
```

```
    return dydt;
}

// Runge-Kutta 4th order step
State rk4_step(double t, double dt, const State& y) {
    State k1 = derivatives(t, y);

    State k2 = derivatives(t + dt/2.0, {
        y.S1 + dt * k1.S1 / 2.0,
        y.S2 + dt * k1.S2 / 2.0,
        y.D  + dt * k1.D  / 2.0
    });

    State k3 = derivatives(t + dt/2.0, {
        y.S1 + dt * k2.S1 / 2.0,
        y.S2 + dt * k2.S2 / 2.0,
        y.D  + dt * k2.D  / 2.0
    });

    State k4 = derivatives(t + dt, {
        y.S1 + dt * k3.S1,
        y.S2 + dt * k3.S2,
        y.D  + dt * k3.D
    });

    State next;
    next.S1 = y.S1 + (dt/6.0) * (k1.S1 + 2*k2.S1 + 2*k3.S1 + k4.S1);
    next.S2 = y.S2 + (dt/6.0) * (k1.S2 + 2*k2.S2 + 2*k3.S2 + k4.S2);
    next.D  = y.D  + (dt/6.0) * (k1.D  + 2*k2.D  + 2*k3.D  + k4.D );

    return next;
}

int main() {
    double t0 = 0.0, t_end = 100.0, dt = 0.1;

    // Initial conditions
    State y;
    y.S1 = 30.0;
    y.S2 = 0.0;
    y.D  = 0.0;

    std::vector<double> time_values, S1_values, S2_values, D_values, T_values;

    for (double t = t0; t <= t_end; t += dt) {
        double T = y.S1 + y.S2;

        time_values.push_back(t);
        S1_values.push_back(y.S1);
        S2_values.push_back(y.S2);
        D_values.push_back(y.D);
        T_values.push_back(T);

        y = rk4_step(t, dt, y);
    }

    // Plotting
    plt::figure_size(900, 600);
```

```
    plt::plot(time_values, S1_values, {{"label", "S1 (Viable Cells)"}, { "color", "green"}});
    plt::plot(time_values, S2_values, {{"label", "S2 (Affected Cells)"}, { "color", "orange"}});
    plt::plot(time_values, D_values,  {{"label", "D (Dying/Dead Cells)"}, { "color", "red"}});
    plt::plot(time_values, T_values,  {{"label", "T = S1 + S2"}, { "color", "blue"}});

    plt::xlabel("Time");
    plt::ylabel("Tumor Burden");
    plt::title("Tumor Growth Model");
    plt::legend();
    plt::grid(true);

    plt::show();

    return 0;
}
```