FINAL PROJECT

MATH336

MATHEMATICAL OPTIMISATION

OPTIMISING ONCOLYTIC VIROTHERAPY

Authors:
Piya Shah and Misha Agrawal

Thursday 19th December, 2024

Introduction

Breast cancer is one of the most prevalent forms of cancer worldwide, costing mankind heavy sums of money and countless lives every year. Many innovative treatments for it are being developed, which can be efficient and safe. Oncolytic virotherapy (OVT) is one such experimental treatment regime, employing genetically engineered viruses to infect and destroy tumour cells while selectively stimulating the immune response, reducing and even eliminating the need for more dangerous treatments like chemo and radiotherapy.

This project attempts to mathematically optimise the dosages and schedules for two specific oncolytic viruses: the Edmonston-Zagreb measles virus (MeV) and the vesicular stomatitis virus Indiana strain (VSV), inspired by a groundbreaking case study of the virologist Beata Halassy, who successfully employed this method for treating recurrent breast cancer (Forčić et al.)

In this project, three Python codes were developed to simulate tumour growth and optimize treatment schedules. The first code models tumour growth and decay rates under viral therapy, using parameters like initial tumour size, specific growth rate, and viral decay rate. The second code applies a grid search algorithm to identify the optimal timing for administering MeV and VSV. Finally, a genetic algorithm was implemented to refine dosages and schedules, evolving optimal solutions through iterative generations. These tools provide a robust framework for analyzing and optimizing OVT protocols.

The Beata Halassy Case

Beata Hallasy is a 50-year-old researcher and senior scientist at the University of Zagreb, Croatia. In 2016, she was diagnosed with Triple Negative Breast Cancer (TNBC). The scans showed several foci (malignant growth tissue) that were treated with mastectomy and chemotherapy. In 2018, another small focus (singular of foci) was found and removed via surgery. However, a small seroma ($< 1\,\text{cm}$) remained at the sight of removal. In 2020, the

seroma progressed to become a solid tumour. It was a hard, palpable, bright red and inflamed nodule. Unable to go through another round of gruelling chemotherapy and surgery, she decided to treat the tumour herself using oncolytic viruses.

Literature Review

Ever since tumours were seen to reduce in size during viral infections nearly a century ago, the concept of OVT has evolved significantly. Although we have tested many different OVT agents in the lab, it is still a challenge to translate any of them into successful real-life applications due to varying factors like tumour response, immune evasion, and toxicity. MeV and VSV are notable candidates due to their safety profiles and tendency to selectively target cancer cells. Recent studies show the efficacy of sequential viral administration to curb immune resistance and enhance tumour cytolysis (Forčić et al.).

In our project, we have closely adhered to the ideas presented in Forčić et al., which discussed the sequential use of MeV and VSV to maintain a cytolytic effect while circumventing antiviral immunity. Inspired by this approach, even our models use key elements such as tumour-specific immune responses and dose-dependent efficacy rates. The computational methods used help us to optimize parameters observed in the original study, providing a data-driven perspective.

Model Description

Basic Specific Growth Rate (SGR)

$$\text{Basic SGR} = \ln\left[\frac{\left(\frac{v_1}{v_2}\right)}{t_2 - t_1}\right]$$

This calculates the tumour's growth rate over time based purely on tumour volume changes and the time difference. However, for our model, we need an adjusted formula to calculate the growth rate. This is because most tumours do not behave in a consistent logarithmic pattern. Through existing literature, we determined the effect of 3 important clinicopathologic factors -

Ki-67, Histological grade and Hormone receptors. Our model assumes that assumes that each predictor independently affects the outcome. For example: higher Ki-67 contributes positively to tumor growth. These effects are cumulative so we add them to the baseline growth rate.

Nature of Regression Coefficients (β)

The regression coefficients (β) tell us how much a predictor changes the outcome for every 1-unit change in that predictor, assuming all other variables are held constant. For example:

$$\beta_{\text{Ki-67}} = 0.115: \text{ For every 1\% increase in Ki-67, the growth rate increases by 0.115 units.}$$

The table below shows the multiple linear regression analysis to evaluate the association between clinicopathologic factors and specific growth rates of invasive breast cancers.

Variable	**Coefficient**	**Standard Error**	**P**	**VIF**
Palpable symptom at diagnosis	0.105	0.121	0.390	1.189
Pathologic T stage	0.250	0.136	0.067	1.167
Histologic grade	0.100	0.101	0.321	1.451
Ki-67	0.115	0.168	0.495	1.592
Hormone receptor	-0.595	0.154	< 0.001	1.536

Table 1: Summary of Regression Model Coefficients, Errors, P-values, and VIFs.

Adjusted SGR Calculation

The adjusted SGR formula becomes:

$$\text{SGR (Basic)} = \ln\left(\frac{2.47}{0.91}\right)/1392 = \ln(0.001947) = -6.24$$

Next, compute the contributions from the regression terms:

$$0.115 \times 45 = 5.175$$

$$0.100 \times 3 = 0.300$$

$$-0.595 \times 1 = -0.595$$

The coefficients to compute the contributions were obtained from Beata Halassy's case report.

Add these to the basic SGR:

$$\text{Adjusted SGR} = -3.05 + 5.175 + 0.300 - 0.595 = -1.36\,\text{hr}^{-1}$$

1 Modelling Tumour Volume over Time:

1.1 Logistical Component

To model the tumour before the injection of viruses, we used a logistic curve equation. It presents itself as initially slow, then fast during exponential growth, and finally slow again as the tumour approaches maximum size due to constraints (space, nutrients etc). This equation is commonly used in Biology to model population dynamics.
The tumour growth model is given by:

$$S(t) = \frac{S_{\max}}{1 + \exp(-k \cdot (t - t_0))}$$

Where:

- $S(t)$: Tumour size at time t
- $S_{\max}$: Maximum possible size of the tumour (growth saturation point)
- k: Growth rate constant, representing how fast the tumour grows (adjusted SGR)
- t_0: Time of inflecion, when the tumour's growth rate is at its peak

1.2 Effects of Viral Therapy

1.2.1 MeV Injection

At the time of the MeV injection (t_{MeV}), the tumour size is reduced by the MeV Effect. The updated tumour size just after injection is:

$$S(t^+_{\text{MeV}}) = S(t^-_{\text{MeV}}) \cdot (1 - \text{MeV Effect})$$

Where:

- $S(t^-_{\text{MeV}})$: Tumour size just before the MeV injection.
- $S(t^+_{\text{MeV}})$: Tumour size just after the MeV injection.

1.2.2 VSV Injection

Similarly, at the time of the VSV injection (t_{VSV}), the tumour size is further reduced by the VSV Effect. The updated tumour size just after this injection is:

$$S(t^+_{\text{VSV}}) = S(t^-_{\text{VSV}}) \cdot (1 - \text{VSV Effect})$$

Where:

- $S(t^-_{\text{VSV}})$: Tumour size just before the VSV injection.
- $S(t^+_{\text{VSV}})$: Tumour size just after the VSV injection.

Note: The factor $(1 - \text{Effect})$ directly scales the tumour size to reflect the fraction that remains after the treatment. Also, we separate these since MeV and VSV have different decay mechanisms.

1.3 Final Tumour Model

After taking into account the effects of MeV and VSV injections, we can project that the tumour resumes its logistical growth, but also undergoes exponential decay due to continual virus effects.

Final Equation:

$$S(t) = \left[\frac{S_{\max}}{1 + \exp(-k \cdot (t - t_0))}\right] \cdot e^{-\lambda \cdot t}$$

It is important to note that both k and λ are rates. However, λ represents the rate at which the tumour shrinks due to the treatment and k represents the rate at which the tumour naturally grows. Also, The inflection point, $(t - t_0)$ occurs where the second derivative of the logistic function changes sign (from positive to negative), which corresponds to when growth

slows down significantly. It's the time when the rate of change of tumour volume switches direction.

We now have our final model. From Beata Halassy's case study report, we obtained the values of tumour size, time of each injection, aand mount of virus injected. The table below contains the values.

Injection	**Cells**	**Day**	**Volume (ml)**	**logCCID50**	**Tumour Size (cm^3) (US size)**
Before starting	-	-	-	-	2.47 ± 0.06
MeV1	MRC 5	1	1.0	5.8	2.61 ± 0.06
MeV2	MRC 5	4	1.0	4.9	2.62 ± 0.06
MeV3	MRC 5	8	1.5	4.2	4.82 ± 0.06
MeV4	Vero	12	2.0	6.8	2.89 ± 0.06
MeV5	Vero	15	2.0	7.8	2.42 ± 0.06
MeV6	MRC 5	19	2.0	6.1	2.31 ± 0.06
MeV7	Vero	22	1.0	7.0	2.09 ± 0.06
MeV1 in total	-	-	-	7.89	-
VSV1	Vero	26	2.0	9.0	1.68 ± 0.06
VSV1	Vero	43	1.0	7.9	1.49 ± 0.06
VSV1	Vero	50	1.1	8.2	1.54 ± 0.06
Excision of tumor	-	59	-	-	0.91 cm^3

Table 2: Tumour Size and LogCCID50 Data

To estimate the value of λ, we used the Curve fitting method.

2 Curve Fitting Method

We now have a mathematical model that describes how the tumour changes with time. Additionally, we have the values of the tumour shrinking with time using Dr. Halassy's case study report. We can now use the curve fitting method to estimate a λ that reduced the difference between the predicted tumour sizes of the model and the tumour sizes from Dr. Halassy's case.

The error function is $E(\lambda) = \sum_i \left(S_{\text{observed}}(t_i) - S_{\text{predicted}}(t_i, \lambda)\right)$

2.1 Code for curve fitting

The curve fitting method will also adjust the parameters to minimise the error function.

```
import numpy as np
```

```
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt

time_points = np.array([0, 1, 4, 8, 12, 15, 19, 22, 26, 43, 50, 59])
time_points_in_hours = time_points * 24

tumor_volumes = np.array([2.47, 2.61, 2.62, 4.82, 2.89, 2.42, 2.31, 2.09, 1.68,
    1.49, 1.54, 0.91])  # Tumor volumes (cm^3)

def tumor_model(t, S_max, k, t_0, lambda_):
    return (S_max / (1 + np.exp(-k * (t - t_0)))) * np.exp(-lambda_ * t)

# Initial guess for the parameters: S_max = 5 (estimated max size), k = 0.1, t_0
    = 20, lambda_ = 0.05 (decay rate)
p0 = [5, 0.1, 20, 0.05]  # Initial guesses for the parameters

params, covariance = curve_fit(tumor_model, time_points_in_hours, tumor_volumes,
    p0)

S_max_fit, k_fit, t_0_fit, lambda_fit = params

# Plotting the results
plt.plot(time_points_in_hours, tumor_volumes, 'o', label='Observed Tumor Volumes
    ')
plt.plot(time_points_in_hours, tumor_model(time_points_in_hours, *params), 'r-',
    label=f'Fitted Curve (λ={lambda_fit:.3f})')
plt.xlabel('Time (hours)')
plt.ylabel('Tumor Volume (cm³)')
plt.legend()
plt.show()

# Print the estimated values
print(f"Estimated decay rate λ: {lambda_fit:.3f} hour^-1")
print(f"Max tumor size (S_max): {S_max_fit:.3f} cm³")
print(f"Estimated growth rate (k): {k_fit:.3f}")
print(f"Estimated inflection time (t_0): {t_0_fit:.3f} hours")
```

2.2 Result

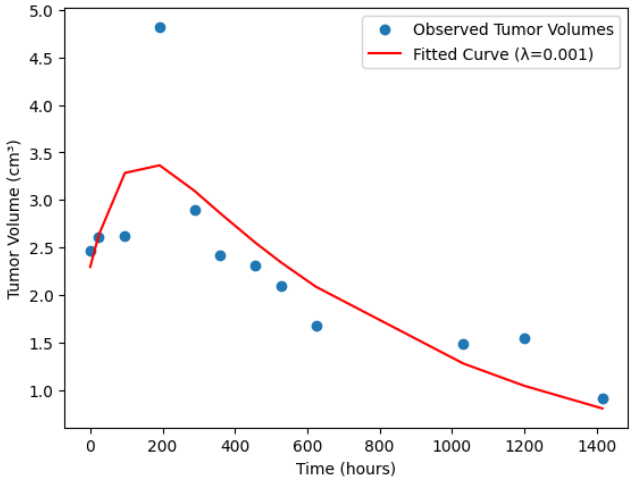


Figure 1: λ estimate using curve fitting

Output:

Estimated decay rate λ: 0.001 hour^{-1}

Estimated **max** tumor size (S_max): 4.418 cm^3

Estimated growth rate (k): 0.016

Estimated inflection time (t_0): -4.784 hours

We have now estimated the value of lambda that fits the data to the mathematical model.

3 Grid Search Optimisation

Using the grid search method, we defined a range of possible values (days) for the patient's MeV and VSV injections. We used Table 2 to create a predefined set of days for the injections. The grid search algorithm would then evaluate all possible combinations and calculate the final size of the tumour based on our model. The values 5% and 13% for the effect of MeV and VSV injections were taken from studies done on rodents using the same viruses.

3.1 Code for Grid Search Optimisation

```
import numpy as np

# Tumor volume model for growth and decay
```

```
def tumor_model(t, S, S_max, k, lambda_):
    return (S / (1 + np.exp(-k * (t)))) * np.exp(-lambda_ * t)

# Objective function to evaluate the final tumour size
def objective_function(timing, S_max, k, t_0, lambda_):
    t_mev = timing[0]
    t_vsv = timing[1]

    # Initial tumor size at t = 0
    tumor_size = 2.47

    mev_effect = 0.05  # 5% tumor size reduction
    vsv_effect = 0.13   # 13% tumour size reduction

    # Tumor size just before and after MeV injection
    if t_mev <= 59:
        tumor_size = tumor_model(t_mev, tumor_size, S_max, k, lambda_)
        tumor_size *= (1 - mev_effect)

    # Tumor size just before and after VSV injection
    if t_vsv <= 59:
        tumor_size = tumor_model(t_vsv, tumor_size, S_max, k, lambda_)
        tumor_size *= (1 - vsv_effect)

    # Tumor size at Day 59
    tumor_size = tumor_model(59, tumor_size, S_max, k, lambda_)
    return tumor_size

# Parameters
S_max = 5.0
k = 0.1
lambda_ = 0.001
t_0 = 20

# Grid search over possible MeV and VSV timings
mev_times = np.array([1, 4, 8, 12, 15, 19])
```

```
vsv_times = np.array([26, 43, 50])

all_combinations = []

for mev in mev_times:
    for vsv in vsv_times:
        timing = [mev, vsv]
        final_tumor_size = objective_function(timing, S_max, k, t_0, lambda_)
        all_combinations.append((mev, vsv, final_tumor_size))

# Sort by final tumour size
all_combinations = sorted(all_combinations, key=lambda x: x[2])

# Print all results
print("MeV Timing | VSV Timing | Final Tumor Size (cm³)")
print("————————————————————————————————————————————")
for combination in all_combinations:
    print(f"Day {combination[0]:<9} | Day {combination[1]:<9} | {combination
        [2]:.3f}")

# Optimal timing
optimal_timing = all_combinations[0]
print("\nOptimal Timing:")
print(f"MeV Timing: Day {optimal_timing[0]}, VSV Timing: Day {optimal_timing[1]}
    ")
print(f"Final Tumor Size: {optimal_timing[2]:.3f} cm³")
```

3.2 Ouput:

```
MeV Timing    | VSV Timing    | Final Tumor Size (cm³)
————————————————————————————————————————————
Day 1         | Day 26        | 0.913
Day 1         | Day 50        | 0.951
Day 1         | Day 43        | 0.951
Day 4         | Day 26        | 1.038
```

```
Day 4          | Day 50          | 1.081
Day 4          | Day 43          | 1.082
Day 8          | Day 26          | 1.191
Day 8          | Day 50          | 1.241
Day 8          | Day 43          | 1.242
Day 12         | Day 26          | 1.322
Day 12         | Day 50          | 1.377
Day 12         | Day 43          | 1.377
Day 15         | Day 26          | 1.402
Day 15         | Day 50          | 1.461
Day 15         | Day 43          | 1.461
Day 19         | Day 26          | 1.486
Day 19         | Day 50          | 1.548
Day 19         | Day 43          | 1.548

Optimal Timing:
MeV Timing: Day 1, VSV Timing: Day 26
Final Tumor Size: 0.913 cm^3
```

Listing 1: Code Output: Tumor Size Reductions

This output gives us the Optimal timing for Beata Halassy's case. The output value for the final tumour's size is closest to the tumour finally removed from the patient.

4 Genetic Algorithm

We can now use a genetic algorithm to create optimal dosage values for MeV and VSV viruses. To execute this, we looked at the logCCID50 values from Table 2 found in Beata Halassy's case report.

logCCID50 is a measure of viral potency used in virology. It indicates the logarithm of 50% cell culture infective dose. It represents the amount of virus required to infect 50% of a cell culture.

4.1 Steps taken for Genetic Algorithm

4.1.1 Initialize Population

Systematically produce a set of solutions, and each solution is a distinct combination of MeV and VSV treatment schedules and dosages (logCCID50).
Example: Solution: [MeV on Day 1, VSV on Day 26, MeV dosage 7.3 logCCID50, VSV dosage 8.5 logCCID50].

4.1.2 Evaluate Fitness

Determine tumour size on Day 59 employing a tumour model for each solution in the population.
A solution's effectiveness is weighed by the ability to significantly reduce the size of a tumour. The smaller, the better the solution it is.

4.1.3 Select Parents

Choose the fittest solutions (parents) according to their fitness (smaller tumour size).
Use a selection technique, which can be tournament selection or roulette wheel selection, to increase the possibility of "reproducing" a better solution.

4.1.4 Crossover

Mix the traits of two selected parents to come up with a new offspring.
For instance, one may give the dosing schedule for MeV while another gives the dosage amount for VSV.
This step simulates genetic recombination and assists in combining good traits from different parents.

4.1.5 Mutation

Implement arbitrary changes to the dosages of MeV or VSV in a solution to induce variability. Such changes may also occur in practice due to human errors.

This allows the algorithm to explore new solutions and avoid getting trapped in local optima (suboptimal solutions).

4.1.6 Repeat for Multiple Generations

The newly formed population is evaluated, and the process is repeated for several generations. With each generation, the solutions improve, and hence the tumour size decreases with the advancement of the algorithm toward the optimal solution.

4.2 Genetic Algorithm code:

```
import numpy as np
import random
import matplotlib.pyplot as plt

def tumor_model(t_mev, t_vsv, d_mev, d_vsv, S0=2.61):
    t_mev_hours = t_mev * 24
    t_vsv_hours = t_vsv * 24

    # decay for MeV and VSV
    decay_mev = d_mev * np.exp(-0.1 * (t_mev_hours - 1))  # MeV decay effect
    decay_vsv = d_vsv * np.exp(-0.1 * (t_vsv_hours - t_mev_hours))  # VSV decay
        effect
    final_tumor_size = S0 - decay_mev - decay_vsv
    return max(final_tumor_size, 0)  # Tumor size should not go below 0

population_size = 50
generations = 100
mutation_rate = 0.1
t_mev_range = (1, 19)  # MeV timing range (Day 1 to Day 19)
t_vsv_range = (26, 50)  # VSV timing range (Day 26 to Day 50)
d_mev_range = (5.5, 7.8)  # MeV dosage range (logCCID50)
d_vsv_range = (7.9, 9.0)  # VSV dosage range (logCCID50)

# Fix the timings as provided
t_mev_fixed = 1  # MeV at Day 1
```

```
t_vsv_fixed = 26  # VSV at Day 26

# Initialize population
def initialize_population():
    population = []
    for _ in range(population_size):
        d_mev = random.uniform(*d_mev_range)
        d_vsv = random.uniform(*d_vsv_range)
        population.append([t_mev_fixed, t_vsv_fixed, d_mev, d_vsv])
    return population

def fitness_function(individual):
    t_mev, t_vsv, d_mev, d_vsv = individual
    return tumor_model(t_mev, t_vsv, d_mev, d_vsv)

def select_parents(population, fitness):
    indices = np.random.choice(range(len(population)), size=2, p=fitness/fitness
        .sum())
    return population[indices[0]], population[indices[1]]

def crossover(parent1, parent2):
    child = parent1[:2] + parent2[2:]
    return child

def mutate(individual):
    if random.random() < mutation_rate:
        individual[2] = random.uniform(*d_mev_range)  # Mutate MeV dosage
    if random.random() < mutation_rate:
        individual[3] = random.uniform(*d_vsv_range)  # Mutate VSV dosage
    return individual

# Genetic algorithm to optimize tumour size
def genetic_algorithm():
    population = initialize_population()
    for generation in range(generations):
        fitness = np.array([1 / (1 + fitness_function(ind)) for ind in
```

```
            population])
        new_population = []
        for _ in range(population_size // 2):
            parent1, parent2 = select_parents(population, fitness)
            child1 = mutate(crossover(parent1, parent2))
            child2 = mutate(crossover(parent2, parent1))
            new_population.extend([child1, child2])
        population = new_population
        best_individual = population[np.argmax(fitness)]
        best_tumor_size = fitness_function(best_individual)
        print(f"Generation {generation}: Best Tumor Size = {best_tumor_size:.3f
            }, Best Individual = {best_individual}")
    return best_individual

optimal_solution = genetic_algorithm()
print(f"Optimal Solution: {optimal_solution}")

best_tumor_sizes = [1.876, 1.963, 1.984, 1.856, 1.856, 1.880, 2.015, 1.994,
    2.015]
best_dosages = [
    [7.323, 7.912], [6.456, 8.465], [6.242, 8.708], [7.517, 8.470],
    [7.517, 8.470], [7.283, 8.708], [5.930, 8.638], [6.146, 8.201], [5.930,
        8.638]
]

mev_dosages = [dosage[0] for dosage in best_dosages]
vsv_dosages = [dosage[1] for dosage in best_dosages]

plt.figure(figsize=(10, 6))
plt.plot(best_tumor_sizes, marker='o', linestyle='-', color='b')
plt.title('Tumor Size Evolution Over Generations')
plt.xlabel('Generation')
plt.ylabel('Best Tumor Size (cm$^3$)')
plt.grid(True)
plt.show()
```

```
plt.figure(figsize=(10, 6))
plt.plot(mev_dosages, marker='o', linestyle='-', color='r', label='MeV Dosage')
plt.title('MeV Dosage Evolution Over Generations')
plt.xlabel('Generation')
plt.ylabel('MeV Dosage (logCCID50)')
plt.grid(True)
plt.legend()
plt.show()

plt.figure(figsize=(10, 6))
plt.plot(vsv_dosages, marker='o', linestyle='-', color='g', label='VSV Dosage')
plt.title('VSV Dosage Evolution Over Generations')
plt.xlabel('Generation')
plt.ylabel('VSV Dosage (logCCID50)')
plt.grid(True)
plt.legend()
plt.show()
```

Listing 2: Genetic Algorithm for Tumor Size Optimization

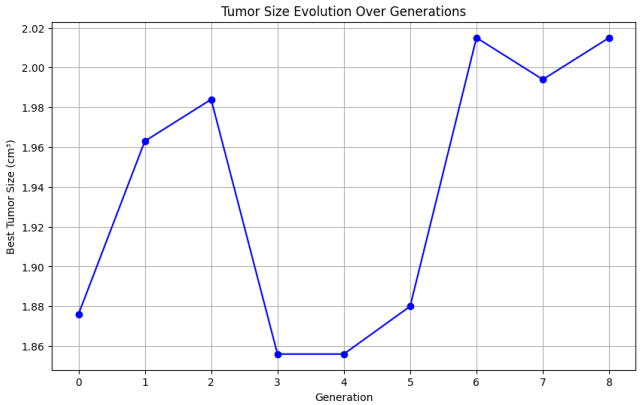
4.3 Output:

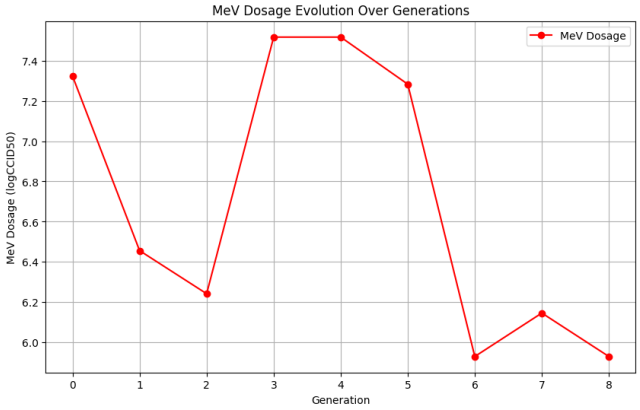
The genetic algorithm iterated over 100 possible generations before giving the final output:

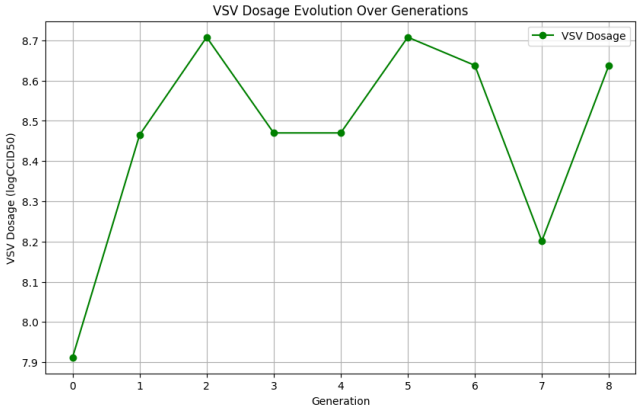
```
Optimal Solution: [1, 26, 7.481834695165985, 8.580477175095801]
```

Listing 3: Optimal Solution Output

The entire list of generations can be seen after running the .py file on Google Colab.







Genetic Algorithm Graph Analysis

Tumour Size Evolution Over Generations

- The best tumour size decreases over generations, indicating effective optimization.
- If the graph flattens, it suggests the algorithm has converged to an optimal solution or is stuck in a local minimum.
- A steady decrease in tumour size reflects consistent improvement in the algorithm's performance.

MeV Dosage Evolution Over Generations

- Fluctuations in the MeV dosage curve show the algorithm adjusting parameters to optimize the tumour size.
- If the curve stabilizes, it indicates the algorithm has found an optimal dosage for MeV.

- A consistent pattern in dosage evolution can indicate the optimal point is being approached.

VSV Dosage Evolution Over Generations

- The VSV dosage evolves to minimize tumour size, with fluctuations showing the adjustment process.
- A steady trend or levelling off of the graph indicates convergence to an optimal VSV dosage.
- Changes in VSV dosage are fine-tuned in response to the algorithm's evaluation of tumour size.

5 Future Work

The scope for future work entails that we can look into animal testing involving OVT to gain more insight into the process and related complications. We could also try to look into more advanced studies that incorporate more variables than the ones we have already used and listed. This is because the object of these experiments is a human body, which cannot be modelled with just a handful of variables due to its intricate biological complexity.

References

1. Inwald, EC, et al. "Ki-67 Is a Prognostic Parameter in Breast Cancer Patients: Results of a Large Population-Based Cohort of a Cancer Registry." *Breast Cancer Research and Treatment*, vol. 139, no. 2, 2013, pp. 539-552. doi:10.1007/s10549-013-2560-8.

2. Soliman, NA, and SM Yussif. "Ki-67 as a Prognostic Marker According to Breast Cancer Molecular Subtype." *Cancer Biology & Medicine*, vol. 13, no. 4, 2016, pp. 496-504. doi:10.20892/j.issn.2095-3941.2016.0066.

3. Lee, SH, et al. "Tumor Growth Rate of Invasive Breast Cancers During Wait Times for Surgery Assessed by Ultrasonography." *Medicine (Baltimore)*, vol. 95, no. 37, 2016, e4874. doi:10.1097/MD.0000000000004874.

4. Forčić, D., et al. "An Unconventional Case Study of Neoadjuvant Oncolytic Virotherapy for Recurrent Breast Cancer." *Vaccines*, vol. 12, 2024, p. 958. https://doi.org/10.3390/vaccines12090958.

5. Leb-Reichl, VM, et al. "Leveraging Immune Memory Against Measles Virus as an Antitumor Strategy in a Preclinical Model of Aggressive Squamous Cell Carcinoma." *Journal of Immunotherapy for Cancer*, vol. 9, no. 10, 2021, e002170. doi:10.1136/jitc-2020-002170.