

Jhonatan Rodrigues Biller

Simulação temporal do crescimento tumoral avascular utilizando Python

Rio Grande, Rio Grande do Sul, Brasil

Dezembro, 2023

Jhonatan Rodrigues Biller

Simulação temporal do crescimento tumoral avascular utilizando Python

Trabalho de Conclusão de Curso, Matemática Aplicada Bacharelado, submetido por Jhonatan Rodrigues Biller junto ao Instituto de Matemática, Estatística e Física da Universidade Federal do Rio Grande.

Universidade Federal do Rio Grande - FURG

Instituto de Matemática, Estatística e Física - IMEF

Curso de Matemática Aplicada Bacharelado

Orientador: Dra. Bárbara Denicol do Amaral Rodriguez

Coorientador: Dra. Cristiana Andrade Poffal

Rio Grande, Rio Grande do Sul, Brasil


Dezembro, 2023

Jhonatan Rodrigues Biller


Simulação temporal do crescimento tumoral avascular utilizando Python

Trabalho de Conclusão de Curso, Matemática Aplicada Bacharelado, submetido por Jhonatan Rodrigues Biller junto ao Instituto de Matemática, Estatística e Física da Universidade Federal do Rio Grande.


Trabalho aprovado. Rio Grande, 14 de dezembro de 2023.

Documento assinado digitalmente
 **BARBARA DENICOL DO AMARAL RODRIGUEZ**
Data: 19/12/2023 16:05:00-0300
Verifique em <https://validar.iti.gov.br>


**Dra. Bárbara Denicol do Amaral
Rodriguez**
(Orientador - FURG)

Documento assinado digitalmente
 **CRISTIANA ANDRADE POFFAL**
Data: 19/12/2023 17:32:50-0300
Verifique em <https://validar.iti.gov.br>

Dra. Cristiana Andrade Poffal
(Coorientadora - FURG)

Documento assinado digitalmente
 **DARCI LUIZ SAVICKI**
Data: 05/01/2024 22:02:59-0300
Verifique em <https://validar.iti.gov.br>

Dr. Darci Luiz Savicki
(Avaliador - FURG)

Documento assinado digitalmente
 **JULIANA DA SILVA RICARDO NUNES**
Data: 08/01/2024 16:13:39-0300
Verifique em <https://validar.iti.gov.br>

Dra. Juliana da Silva Ricardo Nunes
(Avaliador - FURG)

Rio Grande, Rio Grande do Sul, Brasil
Dezembro, 2023

Dedico este trabalho à beleza única da abstração matemática, onde conceitos aparentemente distantes se entrelaçam em um tecido fascinante de padrões e relações.

Agradecimentos

Reconheço a Deus, guia supremo e fonte de toda sabedoria, por permitir-me compreender uma fração de Sua criação.

Estendo minha gratidão aos meus pais, Andrielle Rodrigues Lopes e Tiago Theodoro Santos Lopes, pela educação e apoio em tempo integral.

Ao meu tio Leandro Pereira Rodrigues, agradeço pela sugestão e encorajamento para cursar o bacharelado em Matemática Aplicada.

Expresso também minha gratidão aos meus avós, Maria de Fátima Santos Lopes e Pedro Barbosa Lopes, por suas orações e por me acompanharem na primeira viagem à Universidade Federal do Rio Grande. Um agradecimento especial à minha avó materna, Maria do Carmo Pereira Rodrigues, que igualmente tem orado por mim.

Estendo meus agradecimentos às professoras e orientadoras Bárbara Denicol do Amaral Rodriguez e Cristiana Andrade Poffal, que têm sido presenças fundamentais em minha formação desde o primeiro ano de curso, inicialmente como aluno e, posteriormente, como bolsista em projetos e autor de monografia. Em especial, atribuo todas as minhas conquistas no Ensino Superior ao apoio da professora e orientadora Bárbara Denicol do Amaral Rodriguez, responsável pela minha instrução acadêmica e evolução pessoal.

Agradeço ainda pela formação complementar, preparação e orientação para a pós-graduação, obtidas nos Seminários de Iniciação Científica promovidos pelos professores Adilson da Silva Nunes e Juliana da Silva Ricardo Nunes, e pelos colegas Alana Baldez, Luís Fernandes e Marina Schenque.

Por fim, minha gratidão se estende a todos os amigos que compartilharam essa jornada durante a graduação. Em especial, agradeço ao meu amigo Miguel Cunha pelo apoio constante.

“Dê-me, Senhor, agudeza para entender, capacidade para reter, método e faculdade para aprender, sutileza para interpretar, graça e abundância para falar, acerto ao começar, direção ao progredir e perfeição ao concluir...”
(São Tomás de Aquino)

Resumo

Este trabalho estuda o crescimento de um tumor avascular, desconsiderando fatores de tratamento, por meio da aplicação de três modelos populacionais: Verhulst, Gompertz e Gatenby. Para garantir uma análise comparativa consistente, consideram-se os mesmos parâmetros, bem como uma quantidade inicial idêntica de células cancerosas no problema de valor inicial. Para a obtenção da solução, são empregados três métodos numéricos: Euler, Euler Modificado e Runge-Kutta de 4ª ordem, com passos de 0,1, 0,25 e 0,5, objetivando identificar o método de solução mais preciso. A implementação dos algoritmos é realizada por meio da linguagem de programação Python, com o suporte do NumPy. Para a visualização gráfica é utilizada a biblioteca Matplotlib. Os resultados revelam que o método de Runge-Kutta de 4ª ordem é superior ao método de Euler Modificado, que, por sua vez, supera o método de Euler. Adicionalmente, ao analisar os modelos, observa-se que o de Gompertz apresenta uma taxa de crescimento do câncer significativamente mais acentuada desde os primeiros estágios, quando comparado aos demais. Por outro lado, os modelos de Verhulst e Gatenby demonstram crescimentos semelhantes nos estágios iniciais, mas em estágios mais avançados, Verhulst indica um crescimento mais rápido em comparação ao de Gatenby.

Palavras-chave: Dinâmica populacional. Tumor Avascular. Métodos Numéricos. Python. NumPy. Matplotlib. Análise Numérica.

Abstract

This study investigates the growth of an avascular tumor, disregarding treatment factors, through the application of three popular population models: Verhulst, Gompertz, and Gatenby. To ensure a consistent comparative analysis, the same parameters are considered, along with an identical initial quantity of cancer cells in the initial value problem. Three numerical methods, namely Euler, Modified Euler, and 4th-order Runge-Kutta, are employed with step sizes of 0.1, 0.25, and 0.5 to obtain the solution, aiming to identify the most accurate solution method. The algorithms are implemented using the Python programming language with the support of NumPy. Matplotlib is utilized for graphical visualization. Results reveal that the 4th-order Runge-Kutta method outperforms the Modified Euler method, which, in turn, surpasses the Euler method. Additionally, upon analyzing the models, it is observed that the Gompertz model exhibits a significantly more pronounced cancer growth rate from the early stages compared to the others. On the other hand, the Verhulst and Gatenby models demonstrate similar growth in the initial stages, but in more advanced stages, Verhulst indicates a faster growth compared to Gatenby.

Key-words: Population Dynamics. Avascular Tumor. Numerical Methods. Python. NumPy. Matplotlib. Numerical Analysis.

Lista de ilustrações

Figura 1 – Processo de interpretação Python	37
Figura 2 – Declaração de variáveis	39
Figura 3 – Declaração de variáveis	40
Figura 4 – Estrutura de uma função	41
Figura 5 – Laço de repetição for	41
Figura 6 – Comando de impressão print	42
Figura 7 – Importação e uso de bibliotecas	42
Figura 8 – Exemplos de NumPy arrays	43
Figura 9 – Parâmetro dtype	43
Figura 10 – Função len	44
Figura 11 – Função append	44
Figura 12 – Função linspace	45
Figura 13 – Função reshape	45
Figura 14 – Acessando elementos de um array	46
Figura 15 – Acessando elementos de um array bidimensional	46
Figura 16 – Biblioteca Matplotlib	47
Figura 17 – Exemplos de parâmetros da função plot	47
Figura 18 – Exemplo de plotagem de duas séries de dados	48
Figura 19 – Alteração de escala dos eixos ordenados do comando plot	48
Figura 20 – Modularização da implementação computacional	50
Figura 21 – Estrutura da implementação dos métodos numéricos	52
Figura 22 – Implementação do método de Euler	53
Figura 23 – Implementação do método de Euler Modificado	53
Figura 24 – Implementação do método Runge-Kutta de 4ª ordem	54
Figura 25 – Estrutura da implementação dos modelos populacionais	55
Figura 26 – Implementação do modelo de Verhulst	55
Figura 27 – Implementação do modelo de Gompertz	56
Figura 28 – Implementação do modelo de Gatenby	56
Figura 29 – Implementação das soluções analíticas	57
Figura 30 – Cálculo dos pontos de malha	58
Figura 31 – Arquivo resumo_de_dados	58
Figura 32 – Bibliotecas, módulos e variáveis necessárias para realizar as simulações	61
Figura 33 – Obtendo soluções para o modelo de Verhulst	62
Figura 34 – Soluções para o modelo de Verhulst com passo de 0,1	63
Figura 35 – Obtendo soluções para o modelo de Gompertz	67
Figura 36 – Simulação segundo o modelo de Gompertz (RK4 com $h = 0,1$)	67

Figura 37 – Obtendo soluções para o modelo de Gatenby	70
Figura 38 – Simulação segundo o modelo de Gatenby (RK4 com $h = 0,1$)	71
Figura 39 – Simulações (RK4 com $h = 0,1$)	74
Figura 40 – Relação entre o tempo e a evolução do câncer	76

Lista de tabelas

Tabela 1 – Operações matemáticas em Python	38
Tabela 2 – Ordem de precedência em Python	39
Tabela 3 – Funções matemáticas	47
Tabela 4 – Parâmetros para simulação de câncer humano	60
Tabela 5 – Relação passo e quantidade de pontos de malha	60
Tabela 6 – Resultado da simulação para o modelo de Verhulst utilizando $h = 0,1$.	64
Tabela 7 – Resultado da simulação para o modelo de Verhulst utilizando $h = 0,25$.	64
Tabela 8 – Resultado da simulação para o modelo de Verhulst utilizando $h = 0,5$.	64
Tabela 9 – Erro numérico percentual para o modelo de Verhulst solucionado pelo método de Euler	65
Tabela 10 – Erro numérico percentual para o modelo de Verhulst solucionado pelo método de Euler Modificado	65
Tabela 11 – Erro numérico percentual para o modelo de Verhulst solucionado pelo método de Runge-Kutta de 4ª ordem	66
Tabela 12 – Resultado da simulação para o modelo de Gompertz utilizando $h = 0,1$.	68
Tabela 13 – Resultado da simulação para o modelo de Gompertz utilizando $h = 0,25$.	68
Tabela 14 – Resultado da simulação para o modelo de Gompertz utilizando $h = 0,5$.	68
Tabela 15 – Erro numérico percentual para o modelo de Gompertz solucionado pelo método de Euler	69
Tabela 16 – Erro numérico percentual para o modelo de Gompertz solucionado pelo método de Euler Modificado	69
Tabela 17 – Erro numérico percentual para o modelo de Gompertz solucionado pelo método Runge-Kutta de 4ª ordem	70
Tabela 18 – Resultado da simulação segundo o modelo de Gatenby para células tumorais, utilizando $h = 0,1$	71
Tabela 19 – Resultado da simulação segundo o modelo de Gatenby para células tumorais, utilizando $h = 0,25$	72
Tabela 20 – Resultado da simulação segundo o modelo de Gatenby para células tumorais, utilizando $h = 0,5$	72
Tabela 21 – Resultado da simulação segundo o modelo de Gatenby para células normais, utilizando $h = 0,1$	73
Tabela 22 – Resultado da simulação segundo o modelo de Gatenby para células normais, utilizando $h = 0,25$	73
Tabela 23 – Resultado da simulação segundo o modelo de Gatenby para células normais, utilizando $h = 0,5$	74
Tabela 24 – Resultado da simulação (RK4 com $h = 0,1$)	75

Lista de símbolos e abreviações

α_1	Coeficiente de competição entre as células do câncer e as células normais.
α_2	Coeficiente de competição entre as células normais e cancerígenas.
h	Passo de discretização.
K_1	Capacidade suporte do tumor.
K_2	Capacidade suporte das células normais.
N_0	Número inicial de células tumorais.
N_1	Número de células cancerosas.
N_2	Número de células saudáveis.
N_1^0	Número inicial de células tumorais.
N_2^0	Número inicial de células normais.
r_1	Taxa de crescimento do câncer.
r_2	Taxa de crescimento das células normais.
t	Variável real temporal.
t_i	Instante presente de tempo.
t_j	Instante futuro de tempo.
DNA	Ácido desoxirribonucleico.
ED	Equações diferencial.
EDO's	Equações diferenciais ordinárias.
PVI	Problema do valor inicial.
RK4	Runge-Kutta de 4ª ordem.

Sumário

Introdução	14
1 OBJETIVOS	19
1.1 Objetivo geral	19
1.2 Objetivos específicos	19
2 BIOLOGIA DO CÂNCER	20
2.1 Biologia celular	20
2.1.1 Divisão celular	20
2.1.2 Anormalidades na divisão celular e câncer	20
2.2 Tumores avasculares	21
2.3 Colônias de células cancerosas	21
2.4 Exames de imagem na detecção do câncer	21
2.5 Falência de órgãos	22
2.6 Interação entre células normais e cancerosas	22
3 FUNDAMENTAÇÃO MATEMÁTICA	23
3.1 Conceitos preliminares no \mathbb{R}^n	23
3.2 Limites e continuidade de funções reais	23
3.3 Derivadas de funções reais	24
3.4 Teste da concavidade	25
3.5 Pontos de inflexão	25
3.6 Série de Taylor	26
3.7 Equações diferenciais ordinárias	26
3.8 Problemas do valor inicial	27
3.9 Métodos numéricos de solução de EDO's	27
3.9.1 Tipos de Erros em Processos Numéricos	28
3.9.1.1 Erro percentual absoluto	28
3.9.1.2 Erro de truncamento local	28
3.9.1.3 Erro de Arredondamento	28
3.9.1.4 Erro de truncamento global	29
3.9.2 Métodos de passo único	29
3.9.3 Método de Euler	30
3.9.4 Método de Euler Modificado	31
3.9.5 Método de Runge-Kutta de 4ª ordem	32

3.10	Modelos matemáticos de dinâmica populacional aplicados ao crescimento tumoral avascular	33
3.10.1	Modelo de Verhulst	34
3.10.2	Modelo de Gompertz	34
3.10.3	Modelo de Gatenby	35
4	LINGUAGEM DE PROGRAMAÇÃO PYTHON	37
4.1	Linhas de comando e sintaxe	38
4.2	Operações matemáticas e ordem de precedência	38
4.3	Declaração de variáveis e funções	39
4.4	Laço de repetição for	40
4.5	Comando de impressão print	41
4.6	Importação de bibliotecas e módulos	41
4.7	Biblioteca NumPy e manipulação de vetores	42
4.7.1	Função array	43
4.7.2	Função append	44
4.7.3	Função linspace	44
4.7.4	Função reshape	45
4.7.5	Acesso a elementos de um array	45
4.7.6	Funções matemáticas	46
4.8	Biblioteca Matplotlib e visualização de gráficos	46
5	METODOLOGIA	50
5.1	Estrutura de organização dos arquivos Python	50
5.2	Implementação dos métodos numéricos de passo único	51
5.2.1	Método de Euler	52
5.2.2	Método de Euler Modificado	52
5.2.3	Método Runge-Kutta de 4ª ordem	53
5.3	Implementação dos modelos populacionais	54
5.3.1	Modelo de Verhulst	54
5.3.2	Modelo de Gompertz	55
5.3.3	Modelo de Gatenby	56
5.4	Implementação das soluções analíticas	56
5.5	Funções auxiliares	57
6	RESULTADOS E SIMULAÇÕES	59
6.1	Definição do problema	59
6.2	Simulações	60
6.2.1	Modelo de Verhulst	62
6.2.2	Modelo de Gompertz	66

6.2.3	Modelo de Gatenby	69
6.3	Avaliação dos métodos numéricos	71
6.4	Avaliação dos modelos populacionais	73
7	CONCLUSÃO	77
	REFERÊNCIAS	79

Introdução

O câncer é uma condição caracterizada pelo crescimento descontrolado e anormal de células no corpo (MAYWORM, 2014). Essas células cancerosas têm a capacidade de invadir tecidos circundantes e, eventualmente, se espalhar para outras partes do corpo, formando metástases. Existem diversos tipos de câncer, cada um com características específicas, mas todos compartilham o denominador comum do crescimento celular desordenado.

A relação do câncer com a idade e as condições ambientais é amplamente reconhecida, como evidenciado por Hoffmann (HOFF, 2013) e pelo Instituto Nacional de Câncer - INCA (2022). Esse desequilíbrio no sistema de divisão celular, resultando no crescimento descontrolado de células anormais e no subsequente aumento de volume de tecido corporal, frequentemente afetam do órgãos, é influenciado por diversas causas, tanto internas, como hormônios, condições imunológicas e mutações genéticas (INCA, 2022), quanto externas, relacionadas ao ambiente. Entre esses fatores externos, a industrialização dos alimentos e a presença elevada de poluentes atmosféricos expõem os indivíduos a elementos que podem acentuar o desenvolvimento do câncer.

Uma das principais maneiras de classificar o câncer é em relação à sua malignidade (MAYWORM, 2014). Os tumores malignos, também conhecidos como cânceres malignos, são caracterizados por sua capacidade de invadir tecidos circundantes e se disseminar para outras partes do corpo, tornando-se potencialmente fatais se não forem tratados adequadamente. No contexto da pesquisa em oncologia, a compreensão do crescimento tumoral desempenha um papel crucial na área da oncologia e na investigação de terapias contra o câncer (ARAUJO; ELWAIN, 2004), permitindo uma melhor compreensão dos processos biológicos subjacentes, o desenvolvimento de terapias personalizadas, detecção precoce, prevenção, monitoramento e prognóstico mais precisos, pesquisa de novos tratamentos e aprimoramento da qualidade de vida dos pacientes.

O interesse pelo estudo de modelos que descrevam crescimento de tumores, segundo Saute (2006), continua sendo um desafio para médicos e cientistas, seja na busca de cura, seja na atenuação dos efeitos da doença. Além disso, a dificuldade na obtenção de dados experimentais e as diferenças entre pacientes faz com que a utilização da modelagem matemática seja muito importante no planejamento de tratamentos (SAUTE, 2006). Para Cabella (2012), a elaboração de um modelo que seja capaz de reproduzir os resultados de um sistema reveste-se de importância fundamental sempre que o objetivo seja a compreensão dos mecanismos que conduzem esse sistema a manifestar um comportamento específico. Essa modelagem desempenha um papel crucial, permitindo que se

façam previsões e proporcionando um maior controle sobre os processos envolvidos. Adicionalmente, a modelagem pode revelar padrões de comportamento e, em alguns casos, até mesmo estabelecer conexões entre diversas áreas do conhecimento, unificando-as sob um único formalismo. A aplicação de um mesmo modelo em diferentes contextos pode, assim, culminar na formação de uma teoria fundamental unificadora, que expõe aspectos simples e distintos da natureza.

Técnicas e modelos, para diminuição do agravamento do câncer, são constantemente estudados e aprimorados com o auxílio de diversas áreas do conhecimento. Ao longo de muitos anos, pesquisadores têm utilizado a modelagem matemática como uma ferramenta para descrever o crescimento de entidades biológicas. Particularmente, no âmbito da Biomatemática, os modelos de dinâmica populacional podem ser empregados para a descrição do crescimento tumoral.

Malthus (1798) foi um dos pioneiros na formulação de modelos matemáticos, escrevendo o que se tornou conhecido como o modelo Malthusiano: aborda a relação entre o crescimento populacional e a capacidade de sustentação dos recursos disponíveis. Malthus argumentou que a população humana tinha o potencial de crescer exponencialmente, enquanto a produção de alimentos aumentava em um ritmo mais lento, em uma progressão aritmética. Ele previu que, se o crescimento populacional não fosse controlado, haveria uma escassez de alimentos e, como resultado, ocorreriam crises como a fome e a miséria.

Posteriormente, em contraste com o proposto por Malthus, pesquisadores formularam modelos populacionais com limitantes de crescimento, como o de Verhulst (1838). Esses modelos reconhecem que o crescimento populacional ilimitado, como previsto pelo Malthusiano, raramente ocorre na realidade, uma vez que os recursos são finitos e a densidade populacional pode influenciar o próprio crescimento da população humana.

O princípio de limitação do crescimento populacional serviu de base para a elaboração do modelo de Gompertz (1825). Inicialmente formulado para analisar a mortalidade humana, tem como objetivo explicar e modelar a taxa de mortalidade ao longo da vida das populações, incorporando uma série de fatores que influenciam a mortalidade, como o envelhecimento, a vulnerabilidade à doença e outros riscos que variam com a idade. De forma pioneira, o modelo de Gompertz permitiu a Laird (1964) ajustar com sucesso dados de crescimento tumoral, marcando um avanço importante na aplicação desse modelo à compreensão das dinâmicas do crescimento tumoral. Nesse trabalho o autor discute que raramente o crescimento exponencial de tumores é observado, apenas por curtos períodos de tempo e que os tumores crescem cada vez mais lentamente à medida que o tumor fica maior.

Além da utilização de modelos populacionais clássicos para descrever o crescimento tumoral, ao longo do tempo, pesquisadores dedicaram-se a formular equações capazes de abordar esse fenômeno. Gatenby e Gawlinski (1996) conduziram um estudo que fornece

uma estrutura matemática para a hipótese de mediação ácida e suas implicações no contexto do câncer. No âmbito desse estudo, um sistema de equações de reação-difusão foi formulado, permitindo previsões detalhadas sobre a estrutura e dinâmica da interface entre o tumor e o hospedeiro. Esse modelo baseia-se em um conjunto limitado de parâmetros celulares e subcelulares, demonstrando que alterações nesses parâmetros podem resultar na transição de um tumor benigno para um altamente invasivo. Essa abordagem matemática é consistentemente respaldada por observações clínicas e experimentais de transformações de adenoma ou carcinoma *in situ* em câncer invasivo.

Rodrigues, Pinho e Mancera (2011) conduzem uma revisão abrangente da modelagem matemática na área de dinâmica populacional, com um foco especial nas aplicações relacionadas ao câncer e à farmacologia. No decorrer do trabalho, os autores investigam uma variedade de modelos populacionais. A abordagem consiste em realizar simulações nas quais os mesmos parâmetros são utilizados para avaliar as características distintas fornecidas por cada modelo, considerando e desconsiderando fatores de tratamento para o câncer. Essa análise comparativa oferece uma visão das diferentes dinâmicas populacionais em contextos relacionados ao câncer e à farmacologia.

Considerando o contexto em que a obtenção de soluções exatas para modelos de dinâmica tumoral desempenha um papel crucial, Bortuli, Freire e Maidana (2021) apresentam soluções exatas alcançadas por meio da aplicação da técnica de simetrias de Lie. O modelo matemático em questão trata da invasão tumoral e é composto por um sistema não linear de equações diferenciais parciais que descreve a dinâmica das interações entre a densidade de células tumorais, a densidade da matriz extracelular, a concentração de enzimas degradantes da matriz e a concentração de oxigênio.

Outro trabalho que sublinha a relevância das soluções analíticas no contexto da validação de abordagens numéricas é a pesquisa conduzida por Maganin et al. (2020). No foco desse estudo, encontra-se a simulação e análise de um modelo matemático não linear relacionado ao crescimento tumoral, especificamente no âmbito do câncer de mama, sem adentrar em considerações referentes a tratamentos. Essa investigação abrange uma descrição espacial detalhada da geometria da mama e do desenvolvimento do tumor nos tecidos. A equipe de pesquisadores optou por empregar o método de diferenças finitas para resolver um conjunto de quatro Equações Diferenciais Parciais. No que concerne aos termos não lineares inerentes ao modelo, esses foram linearizados mediante a expansão da série de Taylor.

Fenômenos representados por equações diferenciais, como os modelos de Verhulst (1838), Gompertz (1825) e Gatenby (1996), frequentemente, não possuem soluções analíticas, principalmente quando fatores de tratamento são incorporados, resultando em equações mais complexas. Nesse caso, recorre-se à aplicação de métodos numéricos para obter resultados aproximados e resolver tais equações.

O estudo conduzido por César (2019) engloba a aplicação de métodos numéricos para resolver problemas relacionados ao crescimento tumoral modelados por meio de equações diferenciais. O autor adota como base o modelo de crescimento tumoral formulado por equações diferenciais parciais para investigar a dinâmica do desenvolvimento de tumores invasivos avasculares. Essa escolha de modelo se mostra relevante, visto que concentra sua descrição no crescimento tumoral em contextos nos quais a vascularização ainda não se estabeleceu. A utilização desse modelo proporciona uma perspectiva sobre os mecanismos subjacentes ao crescimento tumoral nessas condições específicas, contribuindo significativamente para uma melhor compreensão dessa área de pesquisa essencial nas esferas médica e científica. O autor emprega o método de diferenças finitas para realizar simulações do crescimento tumoral, conduzindo ainda uma comparação entre o método explícito e o método de dois estágios.

No contexto da modelagem matemática, os métodos numéricos desempenham um papel fundamental na resolução de problemas complexos que muitas vezes carecem de soluções analíticas diretas (CHAPRA; CANALE, 2011) para equações diferenciais. Um aspecto essencial desses métodos é o processo iterativo, no qual cálculos são realizados repetidamente para se aproximar da solução desejada. Essa iteração é uma característica central na resolução de equações diferenciais e sistemas de equações que descrevem fenômenos dinâmicos, como o crescimento tumoral. Essas técnicas, são geralmente implementadas em computadores, aproveitando o poder computacional para lidar com problemas numéricos desafiadores e realizar simulações de alta precisão. Dentre as linguagens de programação atuais que podem ser utilizadas para a implementação de métodos numéricos, encontra-se o Python.

O Python é uma das linguagens de programação mais proeminentes na atualidade, conhecida por sua versatilidade e facilidade de uso (KIUSALAAS, 2005). Sua popularidade se deve, em grande parte, à ampla disponibilidade de bibliotecas que simplificam o desenvolvimento de aplicações em diversas áreas, incluindo a Matemática Aplicada. Um exemplo notável é o NumPy, uma biblioteca especializada em matemática e computação científica, que oferece ferramentas robustas para realizar cálculos numéricos de maneira eficiente. Além disso, o Python conta com o Matplotlib, uma biblioteca que permite a criação de visualizações gráficas de dados.

Nesse contexto, esta pesquisa concentra-se na aplicação dos modelos populacionais de Verhulst (1838), Gompertz (1825) e Gatenby (1996) para a descrição do crescimento tumoral invasivo, sob os mesmos parâmetros. A resolução das equações diferenciais que representam matematicamente o problema é realizada por meio de técnicas numéricas, com o auxílio das bibliotecas NumPy e Matplotlib em Python. Os métodos numéricos empregados incluem Euler, Euler Modificado e Runge-Kutta de 4ª ordem. A relevância desse tema é evidente, especialmente no contexto clínico, considerando a importância de

compreender o crescimento inicial dos tumores e o desenvolvimento de estratégias terapêuticas eficazes (BRASIL, 2011). A pesquisa contribuirá com uma análise comparativa dos modelos e métodos numéricos, fornecendo *insights* sobre como tais modelos se comportam no contexto do crescimento tumoral avascular e como as escolhas dos métodos numéricos podem afetar os resultados. Além disso, ao destacar as discrepâncias entre os modelos, espera-se fornecer dados valiosos para a tomada de decisões clínicas e a formulação de hipóteses para pesquisas futuras.

A estrutura deste trabalho está dividida em sete capítulos. Após esta introdução, o Capítulo 1 detalha os objetivos da pesquisa. No Capítulo 2 aborda-se a biologia do câncer, aprofundando os aspectos biológicos e as características do crescimento tumoral. Em seguida, no Capítulo 3, são discutidos os fundamentos matemáticos, incluindo os modelos de dinâmica populacional e os métodos numéricos para a solução de equações diferenciais. No Capítulo 4, apresenta-se uma introdução à linguagem de programação Python e os comandos utilizados na implementação das soluções. O Capítulo 5 fornece detalhes da metodologia, descrevendo as implementações e os cálculos auxiliares. É importante ressaltar que as soluções obtidas pelos métodos numéricos serão comparadas com soluções analíticas, especialmente no caso dos modelos de Verhulst e Gompertz. No Capítulo 6, são apresentados os resultados e simulações, incluindo gráficos e tabelas para a comparação dos modelos, juntamente com discussões sobre suas diferenças. Por fim, o Capítulo 7 traz as conclusões deste trabalho.

1 Objetivos

1.1 Objetivo geral

Aplicar os modelos populacionais de Verhulst, Gompertz e Gatenby na descrição do número de células cancerosas ao longo do tempo, desconsiderando fatores de tratamento para o câncer, e obter a solução do problema discretizado pelo métodos numéricos de Euler, Euler-Modificado e Runge-Kutta de 4ª ordem.

1.2 Objetivos específicos

- Aprofundar os estudos acerca de Equações Diferenciais Ordinárias (EDO's) e métodos numéricos de solução;
- Estudar os modelos populacionais de Verhulst, Gompertz e Gatenby;
- Implementar, na linguagem de programação Python, os algoritmos correspondentes aos métodos numéricos de Euler, Euler modificado e Runge-Kutta de 4ª ordem;
- Validar os algoritmos implementados;
- Investigar o comportamento de cada modelo de dinâmica populacional, sob os mesmos parâmetros, quando aplicados na solução do problema para a descrição do volume que uma colônia de células cancerosas adquire ao longo do tempo com base no seu tamanho inicial;
- Estabelecer um estudo comparativo a respeito da acurácia dos métodos numéricos de Euler, Euler modificado e Runge-Kutta de 4ª ordem, quando aplicados na solução dos modelos de dinâmica populacional de Verhulst, Gompertz e Gatenby para a descrição do volume de células cancerosas ao longo do tempo.

2 Biologia do Câncer

Neste capítulo, apresentam-se conceitos teóricos fundamentais que sustentam a compreensão do câncer, com ênfase nos tumores avasculares. Descrevem-se temas relacionados à biologia celular (MAYWORM, 2014), que abrange divisão celular e o surgimento do câncer, colônias de células cancerosas (WEINBERG, 2014), falência dos órgãos (CEDERVALL; ZHANG; OLSSON, 2016), a interação entre células normais e cancerosas (WEINBERG, 2014) e uso de exames de imagem na detecção e acompanhamento de tumores (INCA, 2021).

2.1 Biologia celular

A biologia celular é essencial para a compreensão do desenvolvimento do câncer, uma vez que a maioria dos cânceres tem origem em anormalidades na divisão celular e no funcionamento das células. Nesta seção, exploram-se conceitos com ênfase na divisão celular e no acúmulo de anormalidades celulares que podem levar à formação de tumores.

2.1.1 Divisão celular

A divisão celular é um processo fundamental para o crescimento, reparação e manutenção de tecidos em organismos multicelulares. Ela ocorre em duas etapas principais: mitose e citocinese. Na mitose, o núcleo da célula-mãe se divide em dois núcleos filhos, garantindo que cada célula filha contenha a mesma informação genética que a célula-mãe. A citocinese é a divisão do citoplasma e de outros componentes celulares, resultando na formação de duas células filhas independentes.

2.1.2 Anormalidades na divisão celular e câncer

O câncer frequentemente é resultado de anormalidades no processo de divisão celular. A replicação do DNA (ácido desoxirribonucleico) durante a divisão celular é rigorosamente regulada por mecanismos de controle, tais como *checkpoints* celulares e reparo de danos no DNA. Contudo, o acúmulo de mutações genéticas ou a falha nesses mecanismos de controle pode levar a anormalidades na divisão celular, incluindo mutações genéticas, falhas nos *checkpoints* celulares e anormalidades nos processos de reparo de danos no DNA.

O acúmulo de anormalidades celulares, particularmente na regulação da divisão celular e no reparo do DNA, pode resultar na formação de tumores. Tumores podem ser

benignos (não cancerosos), mas se as células continuarem a se dividir de forma descontrolada e a invadir tecidos circundantes, um tumor maligno (câncer) pode se desenvolver.

2.2 Tumores avasculares

O câncer é uma condição caracterizada pelo crescimento descontrolado e anormal de células no corpo. Esse crescimento desenfreado resulta na formação de massas de tecido chamadas tumores, que podem ser benignas ou malignas.

Tumores avasculares referem-se a tumores que ainda não desenvolveram um suprimento sanguíneo próprio, ou seja, não possuem vasos sanguíneos. Geralmente representam estágios iniciais do crescimento tumoral, nos quais as células cancerosas se multiplicam localmente, sem invadir estruturas circundantes ou desenvolver sistemas vasculares para obter nutrientes.

Tumores avasculares desempenham um papel crítico na progressão do câncer, uma vez que são o ponto de partida para o desenvolvimento de tumores vascularizados. O estudo de tumores avasculares é fundamental para compreender as fases iniciais do crescimento tumoral e a competição por recursos no microambiente tumoral.

2.3 Colônias de células cancerosas

Colônias de células cancerosas referem-se a grupos de células tumorais que se multiplicam e formam aglomerados dentro do tecido. Essas colônias desempenham um papel significativo na disseminação e crescimento do câncer.

As colônias de células cancerosas são cruciais na invasão de tecidos circundantes e na formação de metástases. Elas colaboram para o aumento da massa tumoral e desencadeiam uma série de interações no microambiente tumoral.

2.4 Exames de imagem na detecção do câncer

Exames de imagem desempenham um papel fundamental na detecção do câncer. Eles são procedimentos diagnósticos que permitem aos médicos visualizar o interior do corpo, identificar tumores, determinar sua localização e avaliar seu estágio. Esses exames são aplicados quando há suspeita de câncer devido a sintomas clínicos, como dor, inchaço, ou quando o paciente está em um programa de rastreamento para detecção precoce. A importância dos exames de imagem reside na detecção precoce do câncer, o que pode aumentar significativamente as chances de tratamento bem-sucedido. Além disso, eles auxiliam os médicos na determinação do tipo de câncer, seu estágio e na formulação de um plano de tratamento adequado.

2.5 Falência de órgãos

Dentro do contexto do câncer, a falência de órgãos ocorre quando o tumor maligno cresce a ponto de comprometer severamente a função de um órgão vital. Isso pode ocorrer quando o tumor invade, comprime ou bloqueia estruturas vitais, levando à disfunção do órgão afetado. A falência do órgão é uma complicação séria do câncer e pode ser uma das principais causas de mortalidade em pacientes com câncer avançado. A detecção precoce, o tratamento do câncer e a gestão dos sintomas são fundamentais para reduzir o risco de falência de órgãos em pacientes oncológicos.

2.6 Interação entre células normais e cancerosas

O microambiente tumoral refere-se ao ambiente que envolve um tumor ou massa de células cancerosas dentro do corpo. Nesse ambiente, ocorre uma complexa interação entre as células normais e as células cancerosas que compõem o tumor. As células normais presentes no microambiente tumoral incluem células saudáveis do tecido circundante.

Uma característica marcante do microambiente tumoral é a competição por recursos essenciais, como nutrientes e oxigênio. As células normais e cancerosas disputam esses recursos vitais para seu crescimento e sobrevivência. Essa competição tem um impacto significativo no comportamento das células envolvidas.

O microambiente tumoral é reconhecido como um fator crítico na progressão do câncer. As interações que ocorrem nesse ambiente complexo podem influenciar o crescimento do tumor, a capacidade de invasão das células cancerosas em tecidos circundantes e até mesmo a resposta a tratamentos médicos. Compreender a dinâmica do microambiente tumoral é fundamental para desenvolver estratégias terapêuticas mais eficazes no combate ao câncer, uma vez que ele desempenha um papel central na evolução e disseminação da doença.

3 Fundamentação Matemática

Nesta seção, apresenta-se a fundamentação matemática necessária para a realização deste trabalho. Os problemas envolvem equações diferenciais (CHAPRA; CANALE, 2011), sistemas de equações diferenciais (BURDEN; FAIRES, 2006) e a implementação de métodos numéricos para problemas de valor inicial (CHAPRA; CANALE, 2011). Inicialmente, conceitos preliminares (LIMA, 2016) são abordados. Para a análise de estabilidade dos resultados das soluções analíticas das equações diferenciais, utilizam-se conceitos de pontos de acumulação (LIMA, 2016), limites e continuidade para funções de uma (STEWART, 2006) e várias variáveis reais. Além disso, apresentam-se derivadas de primeira e segunda ordem, o teste da concavidade (STEWART, 2006) e série de Taylor (CHAPRA; CANALE, 2011). As demonstrações dos teoremas não são incluídas neste trabalho, mas estão de acordo com as referências (LIMA, 2000), (LIMA, 2016), (BURDEN; FAIRES, 2006), (STEWART, 2006) e (CHAPRA; CANALE, 2011).

3.1 Conceitos preliminares no \mathbb{R}^n

Definição 3.1.1. Seja n um número natural. O espaço euclidiano n -dimensional \mathbb{R}^n é o produto cartesiano de n fatores iguais a \mathbb{R} : $\mathbb{R}^n = \mathbb{R} \times \mathbb{R} \times \dots \times \mathbb{R}$. Seus elementos, portanto, são sequências de n -termos reais $x = (x_1, \dots, x_n)$.

Definição 3.1.2. Uma relação $f : X \rightarrow \mathbb{R}^n$, definida no conjunto $X \subset \mathbb{R}^n$, associa a cada ponto $x \in X$ sua imagem $f(x) = (f_1(x), \dots, f_n(x))$.

Definição 3.1.3. Dados o ponto $a \in \mathbb{R}^n$ e o número real $\xi > 0$, a bola aberta de centro a e raio ξ é o conjunto $B(a, \xi)$ dos pontos $x \in \mathbb{R}^n$ cuja distância ao ponto a é menor que ξ .

Definição 3.1.4. Diz-se que $a \in \mathbb{R}^n$ é ponto de acumulação do conjunto $X \subset \mathbb{R}^n$ quando toda bola aberta de centro a contém algum ponto de X diferente de a .

3.2 Limites e continuidade de funções reais

Definição 3.2.1. Sejam $f : X \rightarrow \mathbb{R}^n$ definida no conjunto $X \subset \mathbb{R}^m$ e $a \in \mathbb{R}^n$ um ponto de acumulação de X . Diz-se que $b \in \mathbb{R}^n$ é o limite de $f(x)$ quando x tende para a e escreve-se $\lim_{x \rightarrow a} f(x) = b$ quando a seguinte condição é válida: Para todo $\varepsilon > 0$ existe $\delta > 0$ tal que $x \in X$ e $0 < |x - a| < \delta$ implicam $|f(x) - b| < \varepsilon$.

Definição 3.2.2. Diz-se que f é contínua no ponto $a \in X$ quando, para cada $\varepsilon > 0$ arbitrariamente dado, pode-se obter $\delta > 0$ tal que

$$x \in X, |x - a| < \delta \Rightarrow |f(x) - f(a)| < \varepsilon. \quad (3.1)$$

Definição 3.2.3. A função $f(x, y_1, \dots, y_m)$, definida no conjunto

$$D = \{(x, u_1, \dots, u_m) \mid a \leq x \leq b \text{ e } -\infty < u_i < \infty, \text{ para cada } i = 1, 2, \dots, m\} \quad (3.2)$$

satisfaz a condição de Lipschitz em D nas variáveis u_1, u_2, \dots, u_m se existe uma constante $L > 0$ tal que

$$|f(x, u_1, \dots, u_m) - f(x, z_1, \dots, z_m)| \leq L \sum_{j=1}^m |u_j - z_j|, \quad (3.3)$$

para todo (x, u_1, \dots, u_m) e (x, z_1, \dots, z_m) em D .

Em particular, 3.2.4 e 3.2.5 apresentam as notações e definições de limites e continuidade para funções reais de uma variável.

Definição 3.2.4. Escreve-se

$$\lim_{x \rightarrow a} f(x) = b \quad (3.4)$$

e diz-se que o limite de f quando x tende a a é igual a b se for possível tornar os valores de f arbitrariamente próximos de b , tomando-se x suficientemente próximo de a .

Definição 3.2.5. Uma função f é contínua em um número a se

$$\lim_{x \rightarrow a} f(x) = f(a). \quad (3.5)$$

3.3 Derivadas de funções reais

Definição 3.3.1. Seja $f : U \rightarrow \mathbb{R}$ uma função definida no aberto $U \subset \mathbb{R}^n$. Para cada $i = 1, \dots, n$, a i -ésima derivada parcial de f no ponto $a = (a_1, \dots, a_n) \in U$ é o número

$$\frac{\partial f}{\partial x_i}(a) = \lim_{\theta \rightarrow 0} \frac{f(a + \theta e_i) - f(a)}{\theta} = \lim_{\theta \rightarrow 0} \frac{f(a_1, \dots, a_i + \theta, \dots, a_n) - f(a)}{\theta}, \quad (3.6)$$

caso este limite exista.

Particularmente, para funções reais de uma variável, a derivada representa a taxa de variação instantânea da função em relação a essa única variável, conforme define 3.3.2

Definição 3.3.2. A derivada de uma função f , em relação à variável x , é definida por

$$f'(x) = \lim_{\theta \rightarrow 0} \frac{f(x + \theta) - f(x)}{\theta}, \quad (3.7)$$

dado um número x para o qual esse limite existe. Se $y = f(x)$ então algumas notações alternativas para a derivada são

$$f'(x) = y' = \frac{dy}{dx} = \frac{df}{dx} = \frac{d}{dx}f(x). \quad (3.8)$$

Definição 3.3.3. Uma função f é diferenciável em a se $f'(a)$ existir. É diferenciável em um intervalo aberto (a, b) se for diferenciável em cada número do intervalo.

Definição 3.3.4. Se f for uma função diferenciável, então sua derivada (3.7) também é uma função, logo f pode ter sua própria derivada, denotada por $(f')' = f''$. Essa nova função f'' é chamada de derivada segunda de f , pois é derivada da derivada de f . Usando a notação de Leibniz escreve-se a derivada segunda de $y = f(x)$ como

$$\frac{d}{dx} \left(\frac{dy}{dx} \right) = \frac{d^2y}{dx^2}. \quad (3.9)$$

A derivada terceira f''' é a derivada da derivada segunda f'' : $f''' = (f'')'$. O processo pode ser continuado. Em geral, a derivada n -ésima de f é denotada por $f^{(n)}$ e é obtida diferenciando-se f , n vezes. Se $y = f(x)$, escreve-se

$$y^{(n)} = f^{(n)}(x) = \frac{d^ny}{dx^n}. \quad (3.10)$$

3.4 Teste da concavidade

Para identificar se uma função é côncava para cima ou para baixo em um intervalo específico, utiliza-se o Teste da Concavidade, o qual envolve a análise do sinal da segunda derivada da função dentro desse intervalo. Veja o Teorema 3.4.1.

Teorema 3.4.1. Seja I um intervalo e f uma função de uma variável, então vale:

1. Se $f'' > 0$ para todo x em I , então o gráfico de f é côncavo para cima (ou convexo) em I .
2. Se $f'' < 0$ para todo x em I , então o gráfico de f é côncavo para baixo (ou concavo) em I .

3.5 Pontos de inflexão

O ponto sobre uma curva onde há troca de concavidade é chamado de Ponto de Inflexão, conforme descreve a Definição 3.5.1.

Definição 3.5.1. Seja $x \in \mathbb{R}$, um ponto P na curva $y = f(x)$ é conhecido como ponto de inflexão, se f é contínua no ponto e a curva mudar de côncava para cima para côncava para baixo ou vice-versa em P .

3.6 Série de Taylor

A série de Taylor (3.11) é uma ferramenta essencial utilizada na matemática e nas ciências para reescrever funções por meio de polinômios mais simples. Ela é construída a partir da expansão de uma função em uma soma infinita de termos, onde cada termo é uma função polinomial derivada da função original em um ponto específico c , ou seja,

$$f(x) = f(c) + \sum_{n=1}^{\infty} \frac{f^{(n)}(c)}{n!} (x - c)^n, \quad (3.11)$$

onde $f(x)$ representa a função original a ser aproximada, c é o ponto de interesse e $f^{(n)}(c)$ denota a n -ésima derivada da função avaliada em c .

A série de Taylor pressupõe que as derivadas da função estejam definidas em c , o que requer que a função seja suficientemente diferenciável na vizinhança desse ponto. O polinômio $T_n(x)$, formado pelos n primeiros termos da série de Taylor, é chamado de polinômio de Taylor de grau n e é dado por:

$$T_n(x) = f(c) + \frac{f'(c)}{1!} (x - c) + \frac{f''(c)}{2!} (x - c)^2 + \frac{f'''(c)}{3!} (x - c)^3 + \cdots + \frac{f^{(n)}(c)}{n!} (x - c)^n. \quad (3.12)$$

Esse polinômio é obtido ao truncar a série de Taylor após os n termos, o que é comumente feito para simplificar a representação da função e obter aproximações práticas.

3.7 Equações diferenciais ordinárias

Definição 3.7.1. Uma equação que contém as derivadas (ou diferenciais) de uma ou mais funções não conhecidas (ou variáveis dependentes), em relação a uma ou mais variáveis independentes é chamada de equação diferencial (ED).

Definição 3.7.2. A ordem de uma equação diferencial é a ordem da maior derivada na equação.

Definição 3.7.3. Se uma equação diferencial contiver somente derivadas ordinárias de uma ou mais funções não conhecidas com relação a uma única variável independente, ela será chamada de equação diferencial ordinária (EDO). Pode-se expressar uma EDO de ordem n em uma variável dependente na forma geral

$$F(x, y, y', \dots, y^{(n)}) = 0, \quad (3.13)$$

onde F é uma função de valores reais de $n+2$ variáveis, $x, y, y', \dots, y^{(n)}$, e $y^{(n)} = d^n y / dx^n$.

Definição 3.7.4. A equação diferencial

$$\frac{d^n f}{dx^n} = f(x, y, y', \dots, y^{(n-1)}), \quad (3.14)$$

onde f é uma função contínua de valores reais, é conhecida por forma normal de (3.13).

Definição 3.7.5. Toda função ϕ , definida em um intervalo I que tem pelo menos n derivadas contínuas em I , as quais quando substituídas em uma equação diferencial ordinária de ordem n reduzem a equação a uma identidade, é denominada uma solução da equação diferencial no intervalo.

3.8 Problemas do valor inicial

Definição 3.8.1. Em um intervalo $I = [a, b]$ o problema de resolver uma equação diferencial de n -ésima ordem sujeita a n condições de contorno especificadas em a :

$$\text{Resolver: } \frac{d^n y}{dx^n} = f(x, y, y', \dots, y^{(n-1)}) \quad (3.15)$$

$$\text{Sujeito à: } y(a) = \omega_1, \ y'(a) = \omega_2, \dots, \ y^{(n-1)}(a) = \omega_n,$$

onde $\omega_1, \omega_2, \dots, \omega_n$ são constantes reais especificadas, é chamado de problema de valor inicial (PVI). Os valores de $f(x)$ e suas $n - 1$ derivadas em um único ponto a : $f(a) = \omega_1$, $f'(a) = \omega_2, \dots, f^{(n-1)}(a) = \omega_n$ são chamadas de condições iniciais.

Definição 3.8.2. Um sistema de m problemas de valor inicial de primeira ordem tem a forma

$$\begin{aligned} \frac{du_1}{dx} &= f_1(x, u_1, u_2, \dots, u_m), \\ \frac{du_2}{dx} &= f_2(x, u_1, u_2, \dots, u_m), \\ &\vdots \\ \frac{du_m}{dx} &= f_m(x, u_1, u_2, \dots, u_m), \end{aligned} \quad (3.16)$$

para $a \leq x \leq b$, com as condições iniciais

$$u_1(a) = \omega_1, \ u_2(a) = \omega_2, \dots, \ u_m(a) = \omega_m. \quad (3.17)$$

3.9 Métodos numéricos de solução de EDO's

O amplo campo das técnicas de soluções exatas e numéricas de equações diferenciais é extremamente diversificado, abrangendo tanto métodos analíticos quanto métodos numéricos (ANDUTTA, 2003). Quando a obtenção de uma solução analítica não é viável, é possível recorrer a soluções numéricas de passo único para problemas de valor inicial.

Esses métodos envolvem a discretização das equações diferenciais para estimar o valor da solução analítica em todo o domínio. De acordo com Carneiro (2020), ferramentas como a modelagem de equações diferenciais, a aplicação de métodos de solução numérica e a implementação computacional desses problemas desempenham um papel fundamental no estudo do comportamento dos modelos. Neste capítulo, exploram-se diferentes tipos de erros numéricos (CHAPRA; CANALE, 2011), métodos de passo único (CHAPRA; CANALE, 2011), incluindo a análise de convergência dessas técnicas (SAUER, 2012), e os métodos numéricos de solução, como Euler, Euler modificado e Runge-Kutta de 4ª ordem (CHAPRA; CANALE, 2011).

3.9.1 Tipos de Erros em Processos Numéricos

Nas simulações matemáticas e cálculos numéricos, é importante considerar a presença de erros, uma vez que os métodos computacionais envolvem aproximações e arredondamentos de valores. Esses erros podem afetar a precisão e a confiabilidade dos resultados obtidos.

3.9.1.1 Erro percentual absoluto

O erro percentual absoluto é uma métrica que quantifica a discrepância entre um valor calculado numericamente S_n e o valor exato S_a . É calculado como a diferença absoluta entre o valor numérico e o valor analítico, dividida pelo valor correto, e expressa em termos percentuais, ou seja:

$$E = \left| \frac{S_a - S_n}{S_a} \right| \times 100. \quad (3.18)$$

Esse erro é uma medida útil para avaliar a precisão de cálculos numéricos e verificar a proximidade dos resultados às soluções teóricas.

3.9.1.2 Erro de truncamento local

O erro de truncamento local é uma medida da discrepância entre o valor real da solução de uma equação diferencial em um determinado ponto e o valor calculado por um método numérico nesse mesmo ponto. Ele ocorre devido à aproximação realizada pelo método numérico e representa a diferença entre a solução exata e a solução aproximada em um único passo de computação.

3.9.1.3 Erro de Arredondamento

O erro de arredondamento ocorre devido à limitação da representação numérica de valores em sistemas computacionais. Uma vez que os computadores utilizam representações finitas para números reais, valores fracionários ou irracionais podem ser arredondados

para se ajustarem à capacidade de armazenamento. Esse processo de arredondamento pode levar a pequenas discrepâncias entre os resultados numéricos e os valores reais. É essencial estar ciente desses erros de arredondamento ao realizar cálculos em ambientes computacionais, pois podem se acumular e impactar a precisão das simulações. A compreensão desses erros e a implementação de estratégias para mitigá-los são aspectos fundamentais em simulações numéricas precisas.

3.9.1.4 Erro de truncamento global

O erro de truncamento global é uma medida que quantifica a discrepância entre a solução exata de um problema matemático e a solução aproximada obtida por métodos numéricos, considerando todas as etapas do cálculo. Ele abrange todas as fontes de erro ao longo do processo computacional, incluindo a discretização de equações, arredondamento de números, simplificação de algoritmos e outros. Avaliar e minimizar o erro de truncamento global é fundamental na análise numérica para garantir que as soluções calculadas se aproximem o máximo possível das soluções reais, proporcionando resultados precisos e confiáveis.

3.9.2 Métodos de passo único

As técnicas de Euler, Euler Modificado e Runge-Kutta de 4ª ordem solucionam problemas de valor inicial para equações diferenciais ordinárias na forma normal (3.13) com $n = 1$. Esses métodos são utilizados para resolver equações diferenciais ordinárias (EDO's), que calculam um novo valor y_j ($j = i + 1$) com base no valor anterior y_i , inclinação φ e um passo h . Ou seja,

$$y_j = y_i + \varphi h. \quad (3.19)$$

O passo h é uma constante usada para dividir o domínio da variável independente t em segmentos menores. O passo baseia-se no valor da variável independente atual t_i para calcular o próximo valor t_j , dado por

$$t_j = t_i + h. \quad (3.20)$$

Cada segmento representa um intervalo de tempo e a equação diferencial é resolvida em cada ponto do intervalo usando a aproximação discreta.

Os métodos que podem ser escritos na forma geral (3.19) são chamados de métodos de passo único. Essa abordagem iterativa é fundamental para a resolução de EDO's, permitindo uma estimativa progressiva da solução ao longo do intervalo de interesse. Cada um desses métodos utiliza variações na maneira como a inclinação é calculada e usada para aproximar a solução, resultando em diferentes níveis de precisão e complexidade

computacional. No entanto, todos eles compartilham o mesmo princípio fundamental de avançar na direção da inclinação para estimar a próxima iteração da solução da EDO. A aplicação dos métodos de passo únicos neste trabalho são análogos para problemas do valor inicial de uma ou várias variáveis. Se o problema de valor inicial for expresso por um sistema de equações diferenciais ordinárias, a variável y_j representará um vetor coluna com um número de linhas equivalente à ordem m do sistema.

No que diz respeito à convergência dos métodos de passo único, o Teorema 3.9.1 assume um papel fundamental na análise da convergência de solucionadores de equações diferenciais de um passo. A análise da dependência do erro global em relação a h sugere que à medida que h diminui, é possível esperar uma redução no erro, pelo menos em cenários de aritmética exata, onde o erro pode ser reduzido conforme necessário. No entanto, um ponto crucial a ser destacado é a dependência exponencial do erro global em relação a b . À medida que o tempo aumenta, o erro global pode crescer significativamente. Em situações em que os valores de t_i são grandes, o tamanho do passo h necessário para manter o erro global sob controle pode se tornar tão pequeno a ponto de se tornar impraticável.

Teorema 3.9.1. Suponha que $f(t, y)$ possui uma constante de Lipschitz L para a variável y e que o valor y_i da solução do problema de valor inicial (3.15) em t_i seja aproximado por w_i a partir de um solucionador de equação diferencial de um passo com erro de truncamento local $e_i \leq Ch^{k+1}$, para alguma constante C e $k \geq 0$. Então, para cada $a < t_i < b$, o solucionador possui o erro de truncamento global definido por

$$g_i = |w_i - y_i| \leq \frac{Ch^k}{L} \left(e^{L(t_i-a)} - 1 \right). \quad (3.21)$$

Se um solucionador de equações diferenciais satisfaz (3.21) à medida que $h \rightarrow 0$, diz-se que o solucionador tem ordem k . Um método de um passo é dito consistente se a equação de diferença do método se aproxima da equação diferencial à medida que o tamanho do passo tende a zero.

3.9.3 Método de Euler

Esta técnica utiliza, a cada iteração, a inclinação dy/dt para aproximar a solução da equação diferencial ordinária (EDO). O cálculo da próxima estimativa da solução é realizado por meio de um avanço (3.20) na direção da inclinação atual definida por

$$y_j = y_i + f(t_i, y_i)h. \quad (3.22)$$

Esse processo é repetido iterativamente ao longo do intervalo de interesse, permitindo uma aproximação da solução da EDO em incrementos sucessivos. O método de

Euler é um método de passo único que deriva da expansão de Taylor, onde a inclinação é aproximada pela derivada da função no ponto atual t_i e multiplicada pelo tamanho do passo h .

3.9.4 Método de Euler Modificado

A técnica descrita na subseção 3.9.3 apresenta uma fonte significativa de erro no método, que reside na suposição de que a derivada dy/dt no início do intervalo pode ser aplicada em todo o intervalo. Uma maneira de aprimorar a estimativa da inclinação envolve a determinação de duas derivadas: uma, no ponto inicial, e outra, no ponto final do intervalo. Em seguida, calcula-se a média dessas duas derivadas para obter uma estimativa melhorada da inclinação ao longo de todo o intervalo. Essa abordagem é conhecida como o método de Euler Modificado e é dada por

$$y_j^0 = y_i + f(t_i, y_i)h, \quad (3.23)$$

$$y_j = f(t_j, y_j^0). \quad (3.24)$$

No Método de Euler, a inclinação no início de um intervalo é usada para extrapolar linearmente y_j . A Equação (3.23) é conhecida como equação preditora, que fornece uma estimativa de (3.24) que permite o cálculo de uma estimativa da inclinação na extremidade final do intervalo.

Assim, pode-se combinar as inclinações (3.23) e (3.24) para obter uma inclinação média no intervalo, que é utilizada para extrapolar linearmente de y_i a y_j , usando o Método de Euler, ou seja:

$$y_j = y_i + \frac{h}{2} (f(t_i, y_i) + f(t_j, y_j^0)). \quad (3.25)$$

A Equação (3.25) é chamada de corretora.

Por fim, é possível representar o Método de Euler Modificado de forma concisa por

$$\begin{aligned} \text{Preditor: } y_j^0 &= y_i + f(t_i, y_i)h. \\ \text{Corretor: } y_j &= y_i + \frac{h}{2} (f(t_i, y_i) + f(t_j, y_j^0)). \end{aligned} \quad (3.26)$$

Nesse método, a equação corretora depende da preditora a cada iteração, permitindo uma abordagem de dois passos para a aproximação da solução do problema de valor inicial. Isso significa que, em cada passo, são obtidas duas estimativas da inclinação da

função, uma no início e outra no final do intervalo, e a média dessas inclinações é utilizada para aprimorar a estimativa da solução ao longo do intervalo.

3.9.5 Método de Runge-Kutta de 4ª ordem

O método de Runge-Kutta é uma família de métodos numéricos amplamente utilizada para resolver equações diferenciais ordinárias e sistemas de EDO's. Esse método tem forma geral dada por:

$$y_j = y_i + \varphi(t_i, y_i, h)h, \quad (3.27)$$

onde a função $\varphi(t_i, y_i, h)$ é chamada função incremento e pode ser escrita como:

$$\varphi = a_1 k_1 + a_2 k_2 + \cdots + a_n k_n. \quad (3.28)$$

Na equação (3.28), os termos a_1, a_2, \dots, a_n são constantes, enquanto os termos k_1, k_2, \dots, k_n representam relações de recorrência. Note que k_1 é parte da equação de k_2 , que, por sua vez, contribui para a equação de k_3 , e assim por diante, conforme

$$\begin{aligned} k_1 &= f(t_i, N_i), \\ k_2 &= f(t_i + p_1 h, N_i + q_{11} k_1 h), \\ k_3 &= f(t_i + p_2 h, N_i + q_{21} k_1 h + q_{22} k_2 h), \\ &\vdots \\ k_n &= f(t_i + p_{n-1} h, N_i + q_{n-1,1} k_1 h + q_{n-1,2} k_2 h + \cdots + q_{n-1,n-1} k_{n-1} h), \end{aligned} \quad (3.29)$$

onde os termos p_1, p_2, \dots, p_{n-1} e $q_{n-1,1}, q_{n-1,2}, \dots, q_{n-1,n-1}$ são considerados constantes.

Em particular, o Runge-Kutta de 4ª ordem é uma extensão do método de Euler e pertence à família de métodos de Runge-Kutta, com o número de termos na função incremento igual a 4. O próximo valor y_j da solução da equação diferencial ordinária, em um ponto subsequente ao valor atual y_i , é uma estimativa que considera uma série de incrementos ponderados, calculados com base na taxa de variação da função desconhecida e em passos de tamanho apropriado ao longo do intervalo de interesse. A quantidade y_j é calculada com base em:

$$y_j = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)h. \quad (3.30)$$

O cálculo de y_j depende das constantes k_1, k_2, k_3 e k_4 , as quais são determinadas por:

$$\begin{aligned}
k_1 &= f(t_i, y_i), \\
k_2 &= f\left(t_i + \frac{h}{2}, y_i + \frac{h}{2}k_1\right), \\
k_3 &= f\left(t_i + \frac{h}{2}, y_i + \frac{h}{2}k_2\right), \\
k_4 &= f(t_i + h, y_i + hk_3).
\end{aligned} \tag{3.31}$$

3.10 Modelos matemáticos de dinâmica populacional aplicados ao crescimento tumoral avascular

Segundo Coelho (2019), muitos pesquisadores, que atuam na dinâmica de crescimento populacional, seja ela animal ou vegetal, procuram compreender a natureza por meio das investigações científicas e da utilização da matemática na descrição, modelagem e previsão do mundo em todos os seus aspectos. Para o autor, a compreensão do crescimento, ou declínio das populações na natureza, e a luta das espécies para predominar uma sobre as outras têm sido assuntos de interesse por um longo período. Com o passar do tempo, na busca por modelos que melhor descrevessem a dinâmica populacional, considerando fatores como restrições para o crescimento, reprodução, migração e mortalidade, os pesquisadores passaram a adaptar e melhorar tais os modelos (COELHO, 2019). No âmbito biológico, podem ser utilizados na descrição e ajuste de dados referentes a crescimento de tumores (CABELLA, 2012).

Para Coelho (2019), considerando que as células tumorais competem entre si por recursos vitais e oxigênio, verifica-se que o modelo de Verhulst é um modelo de dinâmica populacional que contempla tal interação. Em alguns casos, pode ser utilizado para descrever o crescimento da colônia de células cancerosas (COELHO, 2019).

Dentre os modelos de crescimento populacional, os mais notórios são os de Gatenby e Gompertz (COELHO, 2019). O primeiro, é fundamentado em teoria da competição e possui parâmetros tanto para as células cancerosas quanto para as saudáveis. Já o último, é amplamente aplicado em ciências biológicas e é utilizado para descrever o crescimento tumoral.

A seguir, são apresentadas as equações diferenciais que descrevem os modelos de Verhulst, Gompertz e Gatenby, bem como a solução analítica para os modelos de Verhulst (BACAER, 2010) e Gompertz (COELHO, 2019). Vale ressaltar que a obtenção de soluções analíticas não faz parte do escopo deste trabalho.

3.10.1 Modelo de Verhulst

Fundamentado nos estudos de Malthus e Quetelet (BACAËR, 2010) a respeito de dinâmica populacional, em 1838 Pierre François Verhulst publicou seu estudo intitulado “Notas a Respeito da Lei de Crescimento Populacional”. Neste estudo, ele propõe que o aumento populacional, segundo o modelo de Malthus, deve estar sujeito a um limite, determinado pelo tamanho e fertilidade do país no qual a população é objeto de estudo.

Com base nisto, Verhulst propôs um modelo, que mais tarde no artigo “Investigações Matemáticas sobre a Lei de Crescimento Populacional”, foi nomeado de Curva (ou Função) Logística. Tal modelo, é utilizado em diversas áreas do conhecimento, na medicina, por exemplo, para modelar crescimento de tumores (CABELLA, 2012). Ele é dado por:

$$\begin{cases} N_1'(t) &= r_1 N_1 \left(1 - \frac{N_1}{K_1}\right), \\ N_1(0) &= N_0, \end{cases} \quad (3.32)$$

onde $N_1(t)$ representa o tamanho do tumor no instante t , N_0 é o tamanho do tumor no instante $t = 0$, r_1 é a taxa intrínseca de crescimento e K_1 é o tamanho máximo alcançado pelo tumor.

A solução analítica para (3.32) (BACAËR, 2010) é dada por:

$$N_1(t) = \frac{K_1 N_0 e^{r_1 t}}{K_1 + N_0 (e^{r_1 t} - 1)}. \quad (3.33)$$

3.10.2 Modelo de Gompertz

Em 1825, o matemático inglês Benjamin Gompertz sugeriu e aplicou pela primeira vez seu modelo, inicialmente intitulado de “Lei Teorética da Mortalidade de Gompertz”, que relaciona o aumento da taxa de mortalidade e a idade (TJØRVE, 2017). Neste modelo, Gompertz propôs que a taxa de crescimento é inicialmente alta, alterando rapidamente para um crescimento mais lento (COELHO, 2019).

A partir da década de 90, Gompertz e sua teoria ganharam notoriedade em outros campos de estudo, tais como economia, logística e biologia. A biologia é, especialmente, abrangida pelo modelo de Gompertz. Por volta de 1960, A. K. Laird (LAIRD, 1964) ajustou, com sucesso, dados relacionados ao crescimento tumoral ao modelo. A formulação de Gompertz (NASCIMENTO, 2012a) para o crescimento tumoral é:

$$\begin{cases} N_1'(t) &= r_1 N_1 \ln \left(\frac{K_1}{N_1}\right), \\ N_1(0) &= N_0, \end{cases} \quad (3.34)$$

onde $N_1(t)$ representa o tamanho do tumor no instante t , N_0 é o tamanho do tumor no instante $t = 0$, r_1 é a taxa intrínseca de crescimento e K_1 é o tamanho máximo alcançado pelo tumor.

O modelo (3.34) possui solução analítica (COELHO, 2019) representada por:

$$N_1(t) = K_1 e^{\ln(N_0/K_1)e^{-r_1 t}}, \quad (3.35)$$

K_1 pode ser escrito como:

$$K_1 = \lim_{t \rightarrow \infty} N_1(t). \quad (3.36)$$

A Equação (3.35) possui um ponto de variação máximo em $N_{inf} = K_1/e$. A curva (3.35) é côncava para cima se $N_1 < K_1/e$ e côncava para baixo se $N_1 > K_1/e$. O modelo (3.34) é frequentemente utilizado para descrever o crescimento de tumores sólidos (NASCIMENTO, 2012b).

3.10.3 Modelo de Gatenby

Estudos consideram a teoria da competição para a formulação de modelos que descrevam o crescimento tumoral com maior precisão. Um exemplo notável é o modelo proposto por Robert Gatenby, conhecido como o modelo de Gatenby (RODRIGUES; PINHO; MANCERA, 2011), que não leva em consideração o tratamento no contexto do crescimento da colônia de células cancerosas. O modelo de Gatenby (RODRIGUES; PINHO; MANCERA, 2011) é definido da seguinte forma:

$$\begin{cases} N_1'(t) = r_1 N_1 \left(1 - \frac{N_1}{K_1} - \frac{\alpha_1 N_2}{K_1} \right), \\ N_2'(t) = r_2 N_2 \left(1 - \frac{N_2}{K_2} - \frac{\alpha_2 N_1}{K_2} \right), \\ N_1(0) = N_1^0, \\ N_2(0) = N_2^0, \end{cases} \quad (3.37)$$

onde N_1 representa a população de células tumorais e N_2 a população de células normais no instante t , K_1 é a capacidade suporte do tumor, K_2 é a capacidade suporte das células normais determinada pelo tamanho do órgão, α_1 é o coeficiente de competição entre as células do câncer e as células normais, α_2 é o coeficiente de competição entre as células normais e as cancerígenas, r_1 a taxa de crescimento do câncer e r_2 é a taxa de crescimento da célula normal. As constantes N_1^0 e N_2^0 dizem a respeito do número inicial de células tumorais e normais, respectivamente.

O capítulo a seguir apresenta a linguagem de programação Python e as principais funcionalidades utilizadas para implementar as técnicas de solução numérica e os modelos dinâmicos comentados neste capítulo.

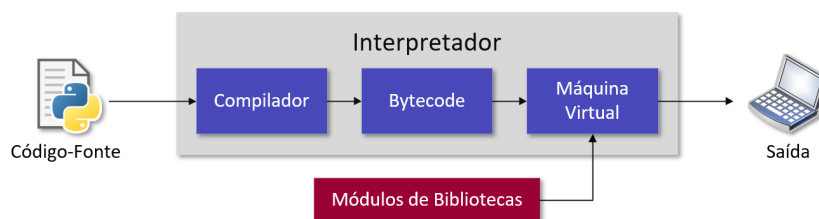
4 Linguagem de programação Python

Python foi criado no final dos anos oitenta (FREITAS, 2019) por Guido Van Rossum no Centro de Matemática e Tecnologia da Informação (*Centrum Wiskunde & Informatica* - CWI), na Holanda, como sucessor da linguagem de programação ABC, que tinha como foco usuários engenheiros e físicos. É dita interpretada, cujo principal objetivo é apresentar uma sintaxe simples de modo a facilitar a leitura do código (KIUSALAAS, 2005). Além disso, é um software de código aberto. Uma de suas características é a tipagem dinâmica e forte, isso significa que o próprio interpretador do Python infere o tipo dos dados que uma variável recebe, sem a necessidade que o usuário da linguagem declare de que tipo determinada variável é.

Outra propriedade dessa linguagem é a variedade de bibliotecas que podem ser utilizadas em diversas áreas do conhecimento. Dentre esses conjunto de módulos estão o NumPy (NUMPY, 2022) e o Matplotlib (MATPLOTLIB, 2022). O NumPy fornece matrizes multi-dimensionais, juntamente com uma coleção de funções matemáticas para operar sobre estas matrizes. Já, o Matplotlib é uma biblioteca de software para criação de visualizações estáticas, animadas e iterativas em Python.

O processo de interpretação (PYTHON, 2023) segue um fluxo simples (Figura 1): o código-fonte é compilado pelo interpretador Python, o qual inclui a importação de módulos de bibliotecas durante essa fase, resultando em bytecode, uma representação intermediária do código. Esse bytecode é então salvo em arquivos com extensão pyc. A execução ocorre na Máquina Virtual Python, que interpreta o bytecode e lida com aspectos como a importação dinâmica de módulos, tornando o código independente da arquitetura do sistema.

Figura 1 – Processo de interpretação Python



Fonte: O autor.

A seguir, serão apresentados os principais tópicos relacionados às funcionalidades em Python utilizadas neste trabalho. A exposição começa com uma discussão sobre linhas de comando e sintaxe (ROMANO, 2015), seguida por operações matemáticas e ordem de precedência (PYTHON, 2023), declaração de variáveis e funções (ROMANO, 2015), laços de repetição (PYTHON, 2023), comandos de impressão (PYTHON, 2023),

importação de bibliotecas e módulos (ROMANO, 2015), além de uma análise mais detalhada das bibliotecas NumPy para manipulação de vetores (NUMPY, 2022) e Matplotlib para visualização de gráficos (MATPLOTLIB, 2022).

4.1 Linhas de comando e sintaxe

Dentro do contexto das linguagens de programação, a sintaxe se refere a um conjunto de estruturas que estabelecem como um algoritmo deve ser formalmente escrito, de modo a ser compreensível e executável pelo computador. Essas regras estabelecem as diretrizes que moldam as linhas de código (ou instruções) que representam operações a serem realizadas pelo computador.

A sintaxe em Python é flexível e permite espaços arbitrários entre operações e palavras reservadas da linguagem. Além disso, é permitida a inclusão de linhas em branco entre as linhas de código. Outra característica notável é a utilização opcional do ponto e vírgula para indicar o fim de uma única linha de instrução; no entanto, seu uso é obrigatório quando se deseja segmentar vários comandos em uma única linha. Além disso, utiliza a indentação (espaços ou tabulações no início das linhas) para estabelecer a hierarquia e a estrutura do código. Por último, os comentários, que são textos adicionados para explicar o código, são precedidos pelo símbolo de cerquilha e não têm efeito na execução do código.

4.2 Operações matemáticas e ordem de precedência

Os cálculos matemáticos entre dois valores em uma linha de comando Python é composta, respectivamente, por três elementos: o primeiro valor, o caractere que representa uma determinada operação e o segundo valor. A Tabela 1 exibe os caracteres utilizados para representar as operações de adição, divisão, subtração, multiplicação e exponenciação.

Tabela 1 – Operações matemáticas em Python

Operação	Caractere
Adição	+
Divisão	/
Subtração	-
Multiplicação	*
Exponenciação	**

Fonte: O autor.

As operações matemáticas em Python estão sujeitas à ordem de precedência, que é um conjunto de regras que definem a prioridade de execução das operações quando

uma expressão contém múltiplos operadores. Essas regras determinam qual operação é realizada antes de outra, estabelecendo uma hierarquia de execução. A Tabela 2 exibe, simplificada, a ordem de precedência em Python, do mais alto para o mais baixo.

Tabela 2 – Ordem de precedência em Python

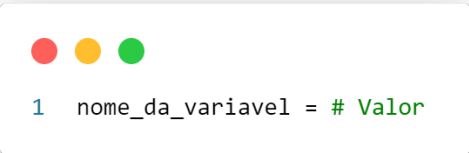
Precedência	Categoria de Operações
1	Parênteses
2	Exponenciação
3	Multiplicação e Divisão
4	Adição e Subtração

Fonte: O autor.

4.3 Declaração de variáveis e funções

Uma variável é um nome empregado para representar e guardar informações na memória do computador. A declaração de variáveis em Python consiste em três elementos essenciais: o nome da variável, o operador de atribuição “=”, e o valor que se deseja atribuir à variável, nessa ordem (Figura 2).

Figura 2 – Declaração de variáveis

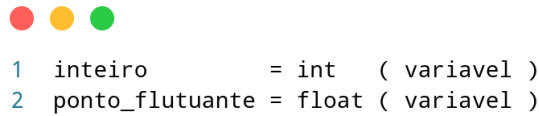


```
1 nome_da_variavel = # Valor
```

Fonte: O autor.

Em Python, é possível realizar a conversão entre tipos de variáveis utilizando as palavras-chave `int` para torná-las números inteiros ou `float` para transformá-las em números do tipo ponto flutuante. A Figura 3 exemplifica essas conversões. Na primeira linha, ocorre a transformação de uma variável para um número inteiro, enquanto na segunda, a variável é convertida para um número de ponto flutuante.

Figura 3 – Declaração de variáveis

A code snippet in a light gray box with a subtle drop shadow. At the top left of the box are three colored circles: red, yellow, and green. Below them are two lines of Python code. The first line is '1 inteiro = int (variavel)' and the second line is '2 ponto_flutuante = float (variavel)'. The numbers 1 and 2 are in blue, and the rest of the code is in black.

```
1 inteiro = int ( variavel )
2 ponto_flutuante = float ( variavel )
```

Fonte: O autor.

Assim como as variáveis são utilizadas para armazenar informações na memória do computador, as funções desempenham um papel fundamental na organização e reutilização de blocos de código em Python. Uma função em Python é uma estrutura que encapsula um conjunto de instruções para realizar uma tarefa específica. Da mesma forma que na declaração de variáveis, uma função também possui três elementos essenciais: o nome da função, parâmetros (se houver) e o bloco de código que define o comportamento da função. O retorno, quando presente, é a informação que a função pode fornecer como resultado de sua execução, tornando-a uma ferramenta poderosa para abstrair e reutilizar lógica em seu código. Os parâmetros personalizam o comportamento da função, e o retorno permite que ela forneça resultados específicos quando chamada. Ambas as variáveis e funções desempenham papéis vitais na construção de programas em Python, contribuindo para sua modularidade e organização (ROMANO, 2015).

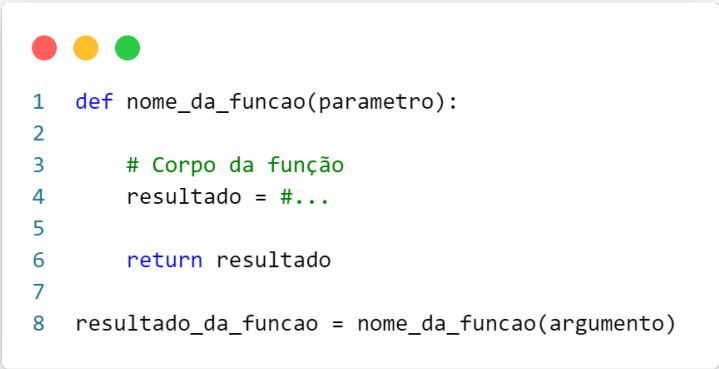
A sintaxe de uma função em Python (Figura 4) inicia-se com a palavra-chave `def`, seguida pelo nome da função. Entre parênteses, listam-se os parâmetros separados por vírgula. Após a declaração dos parâmetros, inserem-se dois pontos para indicar o início do corpo da função, que é indentado com quatro espaços. Dentro do corpo da função, a instrução `return` é utilizada para especificar o valor que a função deve retornar como resultado.

Para executar uma função em Python, digita-se o nome da função seguido dos valores que devem ser fornecidos à função separados por vírgulas (argumentos). Se a função possui retorno, pode-se armazenar esse resultado em uma variável, conforme exemplifica a linha 8 da Figura 4.

4.4 Laço de repetição for

O laço de repetição `for` em Python é uma construção de programação que possibilita a iteração por meio de uma sequência de elementos. O bloco de código dentro do `for` é indentado com quatro espaços e contém as instruções que serão executadas repetidamente para cada elemento na sequência, conforme mostra a Figura 5.

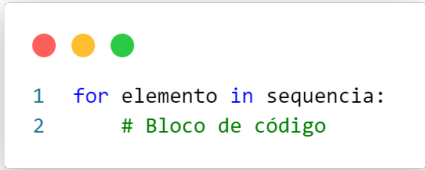
Figura 4 – Estrutura de uma função

A screenshot of a Python code editor window with a white background and three colored window control buttons (red, yellow, green) in the top-left corner. The code is as follows:

```
1 def nome_da_funcao(parametro):  
2  
3     # Corpo da função  
4     resultado = #...  
5  
6     return resultado  
7  
8 resultado_da_funcao = nome_da_funcao(argumento)
```

Fonte: O autor.

Figura 5 – Laço de repetição for

A screenshot of a Python code editor window with a white background and three colored window control buttons (red, yellow, green) in the top-left corner. The code is as follows:

```
1 for elemento in sequencia:  
2     # Bloco de código
```

Fonte: O autor.

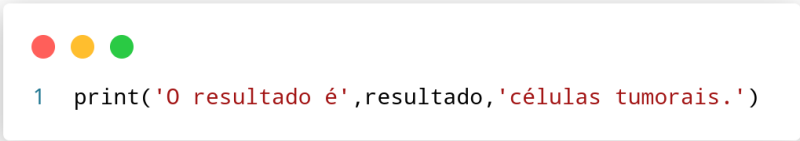
4.5 Comando de impressão print

A função `print` é uma das operações fundamentais em Python, utilizada para exibir informações na saída padrão, comumente no console. O console é um ambiente de linha de comando que permite a interação dos usuários com um sistema de computador ou programa. Com o comando `print`, é possível mostrar texto, variáveis, resultados de cálculos e outros dados na tela. A Figura 6 exemplifica o uso desse comando: inserem-se textos entre aspas e, separados por vírgulas, incluem-se as variáveis cujos resultados desejam-se apresentar ao lado dos textos. O comando `print` coleta essas informações e as converte em um único texto que exibe as informações na ordem em que foram fornecidas.

4.6 Importação de bibliotecas e módulos

Módulos em Python são blocos de código organizados em arquivos individuais que podem conter variáveis, funções e classes. Bibliotecas, por outro lado, são conjuntos de módulos relacionados que oferecem funcionalidades específicas para tarefas comuns de programação. Elas podem ser criadas pela comunidade de desenvolvedores ou fornecidas

Figura 6 – Comando de impressão print

A terminal window with a white background and three colored dots (red, yellow, green) in the top left corner. It contains a single line of Python code: `1 print('O resultado é', resultado, 'células tumorais.')`

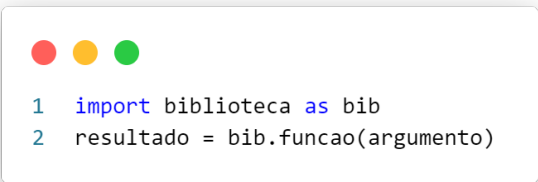
```
1 print('O resultado é', resultado, 'células tumorais.')
```

Fonte: O autor.

como parte da biblioteca padrão do Python.

Embora sejam conceitos distintos, a ação de carregar uma biblioteca ou módulo em Python é realizada por meio da mesma instrução em ambos os casos: o comando `import`. Essa instrução permite que as funções e variáveis definidas no módulo se tornem acessíveis no programa principal. Além disso, a instrução `import` pode ser acompanhada da palavra-chave `as`, que tem o propósito de atribuir um nome alternativo à biblioteca ou módulo. A Figura 7 exemplifica o comando `import` acompanhado da palavra `as` para importar e nomear uma biblioteca fictícia. Neste caso, para executar uma função pertencente a biblioteca ou módulo deve-se escrever o nome atual da biblioteca (renomeada pelo `as`) seguido de ponto, o nome da função e parênteses. Se houver, os argumentos são escritos entre os parênteses.

Figura 7 – Importação e uso de bibliotecas

A terminal window with a white background and three colored dots (red, yellow, green) in the top left corner. It contains two lines of Python code: `1 import biblioteca as bib` and `2 resultado = bib.funcao(argumento)`

```
1 import biblioteca as bib
2 resultado = bib.funcao(argumento)
```

Fonte: O autor.

4.7 Biblioteca NumPy e manipulação de vetores

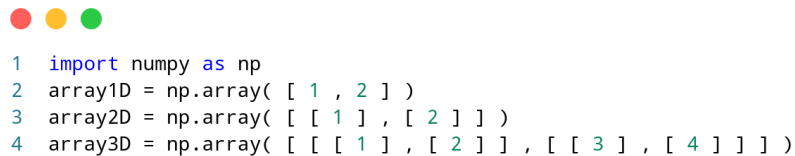
A biblioteca NumPy é um dos pilares fundamentais da computação numérica e científica em Python. Ela desempenha um papel crucial ao permitir o uso de estruturas de dados chamadas arrays que podem conter múltiplas dimensões (também conhecidos como tensores). Além disso, o NumPy oferece um arsenal de funções matemáticas otimizadas, proporcionando uma maneira eficaz de executar cálculos complexos com grande eficiência computacional.

4.7.1 Função array

Um array NumPy é um tipo de estrutura de dados no Python que é especialmente projetado para armazenar e operar eficientemente em dados multidimensionais. Esses arrays podem ser classificados como unidimensional, bidimensional e multidimensional.

Um array unidimensional em NumPy é uma sequência linear de elementos com apenas uma dimensão (linha 2 da Figura 8). Por outro lado, um array bidimensional em NumPy é uma tabela retangular de elementos com duas dimensões (linha 3 da Figura 8). Em último lugar, um array multidimensional são arrays de dimensão superior aos bidimensionais (linha 4 da Figura 8).

Figura 8 – Exemplos de NumPy arrays



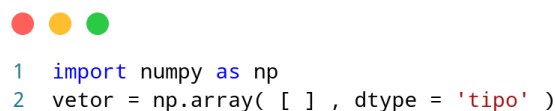
```
1 import numpy as np
2 array1D = np.array( [ 1 , 2 ] )
3 array2D = np.array( [ [ 1 ] , [ 2 ] ] )
4 array3D = np.array( [ [ [ 1 ] , [ 2 ] ] , [ [ 3 ] , [ 4 ] ] ] )
```

Fonte: O autor.

As dimensões dos arrays em Python estão diretamente associadas à estrutura dos dados. Na Figura 8, um vetor unidimensional contém elementos com apenas um “nível” (dimensão), delimitados por dois colchetes. Em contrapartida, um array bidimensional possui elementos com dois níveis, e, por fim, arrays multidimensionais apresentam três ou mais níveis de elementos, organizados em múltiplos colchetes.

A função array possibilita a atribuição de tipos de dados específicos aos seus elementos por meio do parâmetro dtype (Figura 9). Os tipos de dados podem variar, incluindo int para inteiros, float para números de ponto flutuante, bool para valores binários e muitos outros. Isso permite um controle preciso sobre o tipo de dados contidos no array.

Figura 9 – Parâmetro dtype



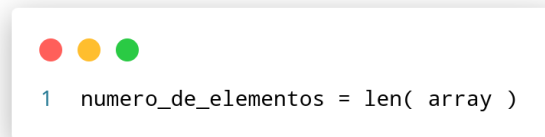
```
1 import numpy as np
2 vetor = np.array( [ ] , dtype = 'tipo' )
```

Fonte: O autor.

O número de elementos em um vetor pode ser determinado com o uso da função len,

como ilustrado na Figura 10. Essa função, quando aplicada a um vetor unidimensional, retorna a contagem total de elementos de primeiro nível presentes no vetor.

Figura 10 – Função len



```
1 numero_de_elementos = len( array )
```

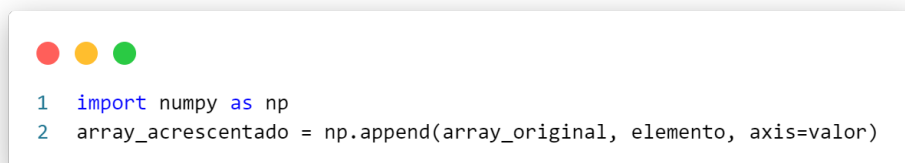
Fonte: O autor.

4.7.2 Função append

A função `append` do NumPy é empregada para inserir elementos em um array NumPy já existente. Os elementos adicionados devem possuir dimensões compatíveis com as do array no qual a adição está ocorrendo.

A Figura 11 mostra a sintaxe do `append`. Essa função apresenta três parâmetros principais: o array no qual ocorre a adição, o elemento a ser acrescentado e um valor binário (0 ou 1) que determina o eixo ao longo do qual a operação de adição é realizada, 0 para adicionar em linhas e 1 para acrescentar em colunas.

Figura 11 – Função append



```
1 import numpy as np
2 array_acrescentado = np.append(array_original, elemento, axis=valor)
```

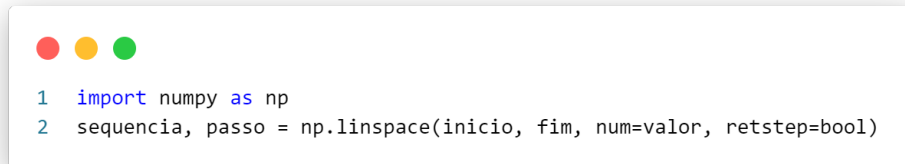
Fonte: O autor.

4.7.3 Função linspace

A função `linspace` do NumPy é usada para criar um array numérico espaçado uniformemente dentro de um intervalo especificado. Os quatro principais parâmetros do `linspace` são: início, fim, número de pontos (`num`) e retornar o espaço entre os elementos (`retstep`) (Figura 12). O início e fim são valores numéricos que indicam o valor inicial e final da sequência. Por sua vez, o valor atribuído ao número de pontos determina a quantidade de elementos na sequência. E, por fim, quando atribui-se o valor `True` ao `retstep`, a função `linspace` retorna dois elementos. O primeiro elemento é o array contendo a sequência de números, e o segundo elemento é o valor do espaçamento entre os elementos

dessa sequência. Caso o valor atribuído ao `retstep` seja `False`, o `linspace` retorna apenas um elemento contendo a sequência.

Figura 12 – Função `linspace`

A code block with a white background and a light gray border. It contains two lines of Python code. The first line is `1 import numpy as np` and the second line is `2 sequencia, passo = np.linspace(inicio, fim, num=valor, retstep=bool)`. Above the code, there are three colored circles: red, yellow, and green.

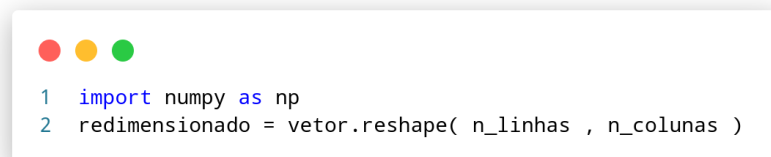
```
1 import numpy as np
2 sequencia, passo = np.linspace(inicio, fim, num=valor, retstep=bool)
```

Fonte: O autor.

4.7.4 Função `reshape`

A função `reshape` do NumPy é uma ferramenta útil para reconfigurar a forma (shape) de um array multidimensional, permitindo que você redimensione o array de forma flexível, mantendo os mesmos elementos, mas organizando-os de uma maneira diferente. A sintaxe básica dessa função é mostrada na Figura 13. Uma característica útil é que o NumPy pode calcular automaticamente o número de linhas ao atribuir -1 ao número de linhas, tornando o processo de redimensionamento mais simples.

Figura 13 – Função `reshape`

A code block with a white background and a light gray border. It contains two lines of Python code. The first line is `1 import numpy as np` and the second line is `2 redimensionado = vetor.reshape(n_linhas , n_colunas)`. Above the code, there are three colored circles: red, yellow, and green.

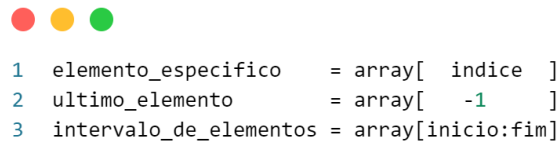
```
1 import numpy as np
2 redimensionado = vetor.reshape( n_linhas , n_colunas )
```

Fonte: O autor.

4.7.5 Acesso a elementos de um array

Em Python os índices de array são usados para acessar elementos individuais dentro de um array. Os índices são baseados em zero, o que significa que o primeiro elemento do array tem o índice 0, o segundo elemento tem o índice 1, o terceiro tem o índice 2 e assim por diante. Um elemento em específico é acessado usando o operador de colchetes seguido do índice desejado, conforme a linha 1 de Figura 14. Ainda, índices negativos são utilizados para contar a partir do final do array, portanto, o índice -1 traz o último elemento de um array (linha 2 da Figura 14). Por fim, pode-se extrair uma parte do array usando uma técnica chamada fatiamento. Ela envolve a especificação de um intervalo de índices, inicial e final, conforme mostra a linha 3 da Figura 14.

Figura 14 – Acessando elementos de um array

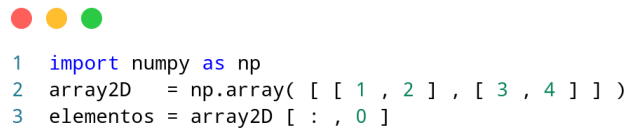
A code snippet in a white box with a red, yellow, and green header bar. It shows three lines of Python code for accessing elements of a NumPy array.

```
1 elemento_especifico = array[ indice ]  
2 ultimo_elemento    = array[ -1     ]  
3 intervalo_de_elementos = array[inicio:fim]
```

Fonte: O autor.

Especificamente, ao lidar com elementos em um vetor NumPy contidos em um array bidimensional, pode-se acessar os primeiros elementos do vetor indicando, entre colchetes, o índice do elemento do primeiro nível e o índice do elemento do segundo nível. Esses índices são separados por uma vírgula. A Figura 15 ilustra esse caso, onde todos os elementos da primeira coluna de cada linha são acessados. A linha 3 atribui à variável `elementos` um vetor unidimensional com dois elementos: 1 e 3.

Figura 15 – Acessando elementos de um array bidimensional

A code snippet in a white box with a red, yellow, and green header bar. It shows three lines of Python code for creating and accessing a 2D NumPy array.

```
1 import numpy as np  
2 array2D = np.array( [ [ 1 , 2 ] , [ 3 , 4 ] ] )  
3 elementos = array2D [ : , 0 ]
```

Fonte: O autor.

4.7.6 Funções matemáticas

A biblioteca NumPy facilita a obtenção de valores de funções matemáticas. A Tabela 3 exibe algumas funções matemáticas e suas equivalências em Python, onde a variável “valor” representa um valor numérico. Vale ressaltar que para utilizar essas funções da maneira como exposta na Tabela 3 deve-se, primeiramente, carregar a biblioteca NumPy como `np`.

4.8 Biblioteca Matplotlib e visualização de gráficos

A biblioteca Matplotlib é uma biblioteca em Python para a criação de gráficos e visualizações de dados. Essa biblioteca é capaz de gerar gráficos tanto em duas dimensões (2D) quanto em três dimensões (3D), permitindo a criação de visualizações mais complexas quando necessário.

Tabela 3 – Funções matemáticas

Função Matemática	Equivalente em Python
Exponencial	<code>np.exp(valor)</code>
Logarítmica (base e)	<code>np.log(valor)</code>
Raiz Quadrada	<code>np.sqrt(valor)</code>
Seno	<code>np.sin(valor)</code>
Cosseno	<code>np.cos(valor)</code>
Módulo	<code>np.abs(valor)</code>

Fonte: O autor.

A função `plot` é fundamental para criar gráficos de linha em duas dimensões. É utilizada em conjunto com a função `show`, que é empregada para exibir a figura desenhada pelo `plot`, ainda, a função não exige parâmetros (linha 3 da Figura 16). A sintaxe da função `plot` (linha 2 da Figura 16) possui três elementos principais, o vetor de pontos do eixo x , do eixo y e a enumeração dos parâmetros, separados por vírgula.

Figura 16 – Biblioteca Matplotlib

```
1 import matplotlib.pyplot as plt
2 plt.plot(x, y, parametros)
3 plt.show()
```

Fonte: O autor.

O `plot` possui diversos parâmetros com finalidade de controlar detalhes como cores, tamanhos, rótulos, legendas, eixos e muito mais para criar gráficos que se adéquem às necessidades específicas. A Figura 17 exibe o comando `plot` com alguns parâmetros.

Figura 17 – Exemplos de parâmetros da função `plot`

```
1 import matplotlib.pyplot as plt
2 plt.plot(x, y, 'cor', linestyle='estilo', label='legenda')
3 plt.legend()
4 plt.show()
```


Fonte: O autor.

O argumento `cor` especifica a cor da linha ou dos marcadores em um gráfico. Neste

trabalho, utilizam-se cores hexadecimais. Por sua vez, o parâmetro `linestyle` é usado para especificar o estilo da linha que conecta os pontos em um gráfico de linha. Pode-se atribuir a esse parâmetro os estilos `solid`, para uma desenhar uma curva contínua, e `dotted`, para um obter um gráfico pontilhado. Em último lugar, `legend` é utilizado para gerar um rótulo para uma determinada curva e atribui-se à legenda um texto personalizado. A legenda é adicionada ao gráfico criado com o comando `plot` por meio do comando `legend` (linha 4 da Figura 17).

Outra funcionalidade oferecida pelo Matplotlib é a capacidade de plotar várias séries de dados no mesmo gráfico chamando a função `plot` várias vezes antes da função `show`. A Figura 18 exemplifica essa funcionalidade, onde utiliza-se o `plot` com finalidade de exibir duas séries de dados.

Figura 18 – Exemplo de plotagem de duas séries de dados




```
1 import matplotlib.pyplot as plt
2 plt.plot(x, y, 'cor', linestyle='estilo', label='legenda1')
3 plt.plot(z, w, 'cor', linestyle='estilo', label='legenda2')
4 plt.legend()
5 plt.show()
```

Fonte: O autor.

Por fim, pode-se alterar a escala dos eixos x e y para logarítmica, atribuindo a palavra `log` ao parâmetro das funções `xscale` (escala do eixo x) e `yscale` (escala do eixo y), conforme exemplifica a Figura 19.

Figura 19 – Alteração de escala dos eixos ordenados do comando `plot`



```
1 import matplotlib.pyplot as plt
2 plt.plot(x, y, 'cor', linestyle='estilo', label='legenda')
3 plt.xscale('log')
4 plt.yscale('log')
5 plt.legend()
6 plt.show()
```

Fonte: O autor.

No capítulo seguinte, detalha-se a metodologia adotada neste trabalho, onde apresentam-se a estrutura de organização dos arquivos Python, a implementação dos métodos numéricos de passo único, a incorporação dos modelos populacionais e a criação de funções pertinentes a este estudo.

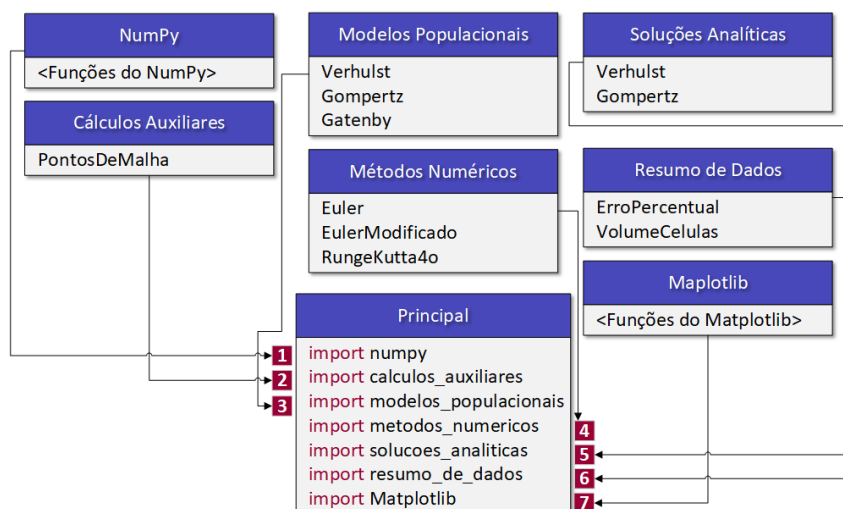
5 Metodologia

O objetivo deste trabalho é implementar os métodos numéricos de Euler, Euler modificado e Runge-Kutta de 4ª ordem, por meio da linguagem de programação Python, para a obtenção da solução dos problemas de valor inicial para os modelos de Verhulst, Gompertz e Gatenby aplicados ao crescimento tumoral avascular. Os modelos de Verhulst e Gompertz carregam as informações com relação à doença, tais como capacidade do meio biológico e taxa intrínseca de crescimento. O modelo de Gatenby, por sua vez, tem como parâmetros não só as informações relacionadas ao tumor, mas também considera a capacidade de defesa das células normais. Neste capítulo, apresenta-se a estrutura adotada para a implementação computacional, na linguagem Python, dos algoritmos para os modelos de Verhulst, Gompertz e Gatenby, considerando os métodos numéricos de Euler, Euler modificado e Runge-Kutta de 4ª ordem.

5.1 Estrutura de organização dos arquivos Python

O projeto em Python é dividido em seis arquivos principais, além das bibliotecas NumPy e Matplotlib, cada um com um propósito específico. Esta abordagem de divisão facilita o desenvolvimento, a depuração e a expansão do código. Os seis arquivos são nomeados como `metodos_numericos`, `modelos_populacionais`, `solucoes_analiticas`, `resumo_de_dados`, `calculos_auxiliares` e `principal` (Figura 20).

Figura 20 – Modularização da implementação computacional



Fonte: O autor.

O arquivo `metodos_numericos` contém a implementação dos métodos numéricos utilizados para resolver os problemas específicos deste trabalho. Aqui, definem-se os

algoritmos para o método de Euler, o método de Euler modificado e o método de Runge-Kutta de 4ª ordem. Isso separa claramente a lógica numérica do restante do código, promovendo uma abordagem modular. Por sua vez, o arquivo `calculos_auxiliares` realiza o cálculo dos pontos de malha para um vetor composto por passos de discretização.

No arquivo `solucoes_analiticas` são fornecidas as soluções analíticas para os modelos que estudados. Isso permite uma comparação direta entre as soluções numéricas e as soluções exatas, auxiliando na avaliação da precisão dos métodos.

O módulo `modelos_populacionais`, por sua vez, contém a definição das funções que representam os modelos matemáticos que descrevem o crescimento populacional.

Em `resumo_de_dados`, realiza-se a avaliação da precisão dos métodos numéricos. Calculam-se erros e comparam-se resultados. Esta separação facilita o acompanhamento e a análise da qualidade das soluções.

Por fim, o arquivo principal onde reúnem-se todas as partes para executar a simulação completa. Aqui, importam-se os módulos necessários, configuram-se os parâmetros e executam-se as simulações.

5.2 Implementação dos métodos numéricos de passo único

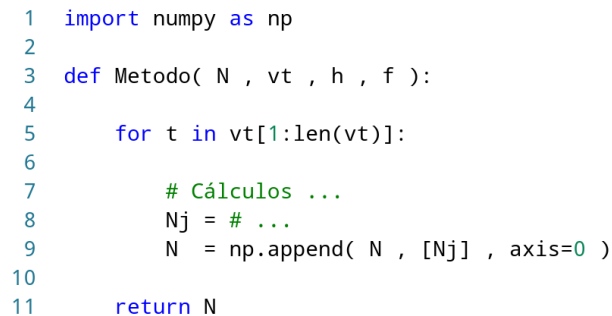
Nesta seção, explora-se a implementação computacional dos métodos numéricos de passo único, que desempenham um papel crucial na resolução de equações diferenciais neste trabalho. É importante notar que, ao implementar computacionalmente essas técnicas de solução, os parâmetros fundamentais e os processos iterativos permanecem consistentes em todos os métodos. A diferença principal reside nos cálculos realizados dentro do laço de repetição `for`, que é adaptado de acordo com a metodologia específica de cada método numérico.

Independentemente do método escolhido, os seguintes elementos são comuns a todos: parâmetros iniciais, processo iterativo, cálculos iterativos e armazenamento dos resultados. Assim, a estrutura dos métodos de Euler, Euler Modificado e Runge-Kutta de 4ª ordem têm a estrutura apresentada na Figura 21.

Inicia-se cada método numérico definindo os parâmetros iniciais necessários para a resolução do problema. Isso inclui a especificação da condição inicial `N`, o vetor tempo `vt`, `h` e `f`. Respectivamente, esses parâmetros são: um array `n`-dimensional NumPy, uma sequência de elementos igualmente espaçados gerados pela função `linspace`, o passo retornado pelo `retstep` e uma função em Python (linha 3 da Figura 21).

Por sua vez, utiliza-se um laço de repetição `for` para realizar iterações sucessivas. Esse laço percorre os elementos do vetor tempo, com exceção do elemento de índice 0 que está associado à condição inicial (linha 5 da Figura 21), onde a cada iteração, avança-se

Figura 21 – Estrutura da implementação dos métodos numéricos

A code editor window with a white background and three colored window control buttons (red, yellow, green) in the top-left corner. The code is written in Python and is as follows:

```
1 import numpy as np
2
3 def Metodo( N , vt , h , f ):
4
5     for t in vt[1:len(vt)]:
6
7         # Cálculos ...
8         Nj = # ...
9         N = np.append( N , [Nj] , axis=0 )
10
11     return N
```

Fonte: O autor.

um pequeno passo na direção do próximo ponto no domínio da variável independente. Ainda, dentro do laço de repetição, realizam-se cálculos específicos para atualizar as variáveis dependentes de acordo com o método numérico escolhido. Esses cálculos são projetados para aproximar a solução da EDO em incrementos sucessivos. À medida que os cálculos avançam, armazenam-se os resultados intermediários e finais (linhas 7, 8, 9 e 11 da Figura 21), dependendo das necessidades do método. Isso permite analisar e visualizar posteriormente os resultados obtidos.

A seguir, são apresentadas as implementações para as técnicas numéricas de solução abordadas neste trabalho.

5.2.1 Método de Euler

O Método de Euler é uma abordagem numérica amplamente utilizada para resolver equações diferenciais ordinárias (EDO's). Sua implementação segue os princípios fundamentais estabelecidos em (3.22). A Figura 22 apresenta a estrutura geral de implementação deste método. Ele se destaca pela simplicidade, dividindo o intervalo de interesse em pequenos incrementos e utilizando a inclinação da função no ponto atual para avançar para o próximo ponto na solução da EDO.

5.2.2 Método de Euler Modificado

A aplicação do método de Euler Modificado segue os princípios fundamentais definidos em (3.26). A Figura 23 ilustra a estrutura geral, em que $N0j$ representa a equação preditora e Nj , a equação corretora.

Figura 22 – Implementação do método de Euler

```
Simulacao - metodos_numericos.py

1 import numpy as np
2
3 def Euler( N , vt , h , f ):
4
5     for t in vt[1:len(vt)]:
6
7         Nj = N[-1] + h * f( t , N[-1] )
8         N = np.append( N , [Nj] , axis=0 )
9
10    return N
```

Fonte: O autor.

Figura 23 – Implementação do método de Euler Modificado

```
Simulacao - metodos_numericos.py

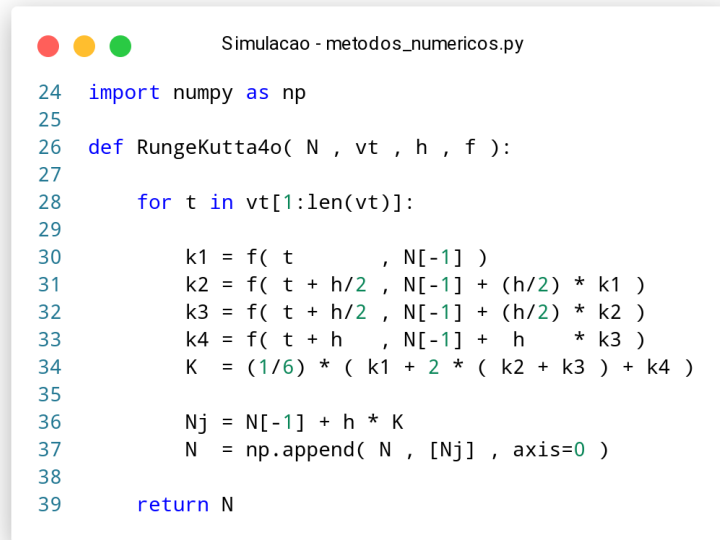
12 import numpy as np
13
14 def EulerModificado( N , vt , h , f ):
15
16     for t in vt[1:len(vt)]:
17
18         N0j = N[-1] + h * f( t , N[-1] )
19         Nj = N[-1] + (h/2) * ( f( t , N[-1] ) + f( t + h , N0j ) )
20         N = np.append( N , [Nj] , axis=0 )
21
22    return N
```

Fonte: O autor.

5.2.3 Método Runge-Kutta de 4ª ordem

A implementação do método Runge-Kutta de 4ª ordem segue os princípios fundamentais que foram estabelecidos com base em (3.30) e pode ser visualizada na Figura 24.

Figura 24 – Implementação do método Runge-Kutta de 4ª ordem



```
24 import numpy as np
25
26 def RungeKutta4o( N , vt , h , f ):
27
28     for t in vt[1:len(vt)]:
29
30         k1 = f( t , N[-1] )
31         k2 = f( t + h/2 , N[-1] + (h/2) * k1 )
32         k3 = f( t + h/2 , N[-1] + (h/2) * k2 )
33         k4 = f( t + h , N[-1] + h * k3 )
34         K = (1/6) * ( k1 + 2 * ( k2 + k3 ) + k4 )
35
36         Nj = N[-1] + h * K
37         N = np.append( N , [Nj] , axis=0 )
38
39     return N
```

Fonte: O autor.

5.3 Implementação dos modelos populacionais

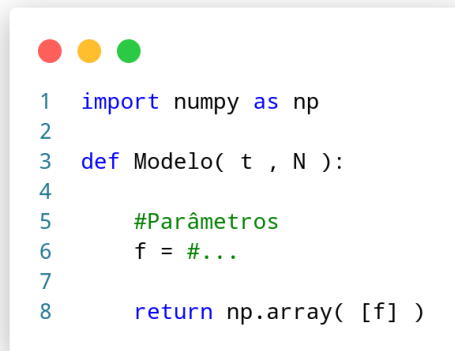
Nesta seção, aplicam-se os métodos numéricos previamente discutidos para traduzir os modelos matemáticos de crescimento populacional em implementações computacionais. Isso consiste em traduzir equações e conceitos abstratos em código real, permitindo simular e analisar o comportamento das populações ao longo do tempo.

Em cada uma dessas implementações, encontra-se uma estrutura comum (Figura 25) que inicia com o comando `def`, seguido por cálculos realizados com base nas equações para cada modelo. Cada uma dessas funções, tem como parâmetros um vetor contendo a quantidade de uma determinada população N em um determinado instante de tempo t . No corpo da função, encontram-se os parâmetros populacionais do modelo e o cálculo da taxa de variação da população de acordo com a forma normal (linha 6 da Figura 25). Como retorno, tem-se um array unidimensional ou bidimensional contendo essas taxas calculadas (linha 8 da Figura 25).

5.3.1 Modelo de Verhulst

O Modelo de Verhulst, como definido anteriormente neste trabalho, é regido pelo PVI dado por (3.32), que descreve o crescimento populacional considerando fatores limitantes. A implementação do Modelo de Verhulst segue estritamente tal equação, conforme representado na Figura 26.

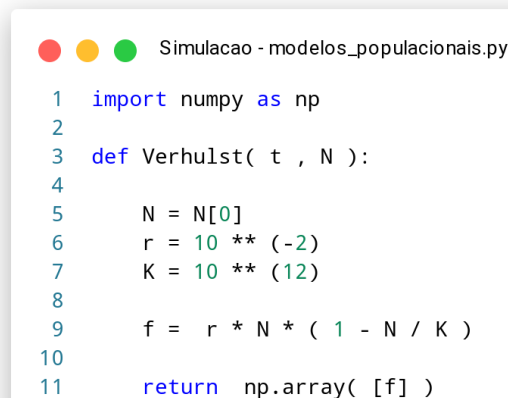
Figura 25 – Estrutura da implementação dos modelos populacionais



```
1 import numpy as np
2
3 def Modelo( t , N ):
4
5     #Parâmetros
6     f = #...
7
8     return np.array( [f] )
```

Fonte: O autor.

Figura 26 – Implementação do modelo de Verhulst



```
Simulacao - modelos_populacionais.py
1 import numpy as np
2
3 def Verhulst( t , N ):
4
5     N = N[0]
6     r = 10 ** (-2)
7     K = 10 ** (12)
8
9     f = r * N * ( 1 - N / K )
10
11     return np.array( [f] )
```

Fonte: O autor.

5.3.2 Modelo de Gompertz

A Figura 27 fornece uma representação visual da estrutura da implementação do modelo de Gompertz. Assim como na subseção 5.3.1, o cálculo apresentado na linha 9, da Figura 27, segue a forma normal do PVI para o modelo de Gompertz (3.34). Em contraste com os demais modelos, este modelo possui duas equações, logo, o parâmetro N da função Gatenby, em Python, é um array bidimensional. Ainda, essa função retorna um vetor também um vetor bidimensional, de elementos f_1 e f_2 os quais são calculados segundo o modelo de Gatenby.

Figura 27 – Implementação do modelo de Gompertz

```
Simulacao - modelos_populacionais.py

13 import numpy as np
14
15 def Gompertz( t , N ):
16
17     N = N[0]
18     r = 10 ** (-2)
19     K = 10 ** (12)
20
21     f = r * N * np.log( K / N )
22
23     return np.array( [f] )
```

Fonte: O autor.

5.3.3 Modelo de Gatenby

A Figura 28 apresenta a implementação do Modelo de Gatenby.

Figura 28 – Implementação do modelo de Gatenby

```
Simulacao - modelos_populacionais.py

25 import numpy as np
26
27 def Gatenby( t , N ):
28
29     r1 = 10 ** (-2) ; r2 = 10 ** (-3)
30     a1 = 9 * 10 ** (-2) ; a2 = 9 * 10 ** (-2)
31     k1 = 10 ** (12) ; k2 = 10 ** (12)
32     N1 = N[0] ; N2 = N[1]
33
34     f1 = r1 * N1 * ( 1 - ( N1 / k1 ) - ( a1 * N2 ) / k1 )
35     f2 = r2 * N2 * ( 1 - ( N2 / k2 ) - ( a2 * N1 ) / k2 )
36
37     return np.array( [f1,f2] )
```

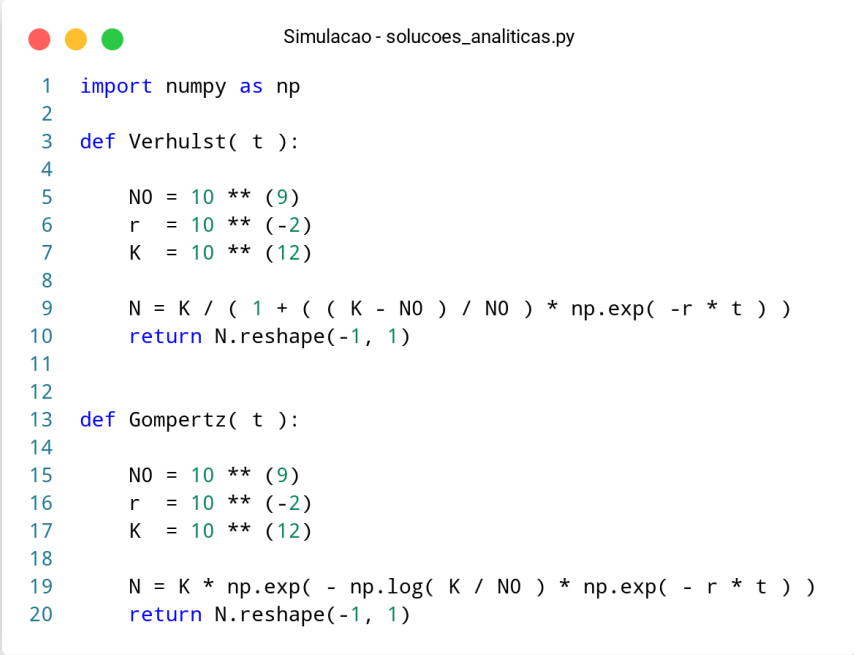
Fonte: O autor.

5.4 Implementação das soluções analíticas

A implementação das soluções analíticas (Figura 29) desempenha um papel fundamental no estudo, permitindo a comparação da precisão das técnicas de solução numérica. Ao desenvolver e incorporar as soluções analíticas no código Python, estabelece-se uma base sólida para avaliar o desempenho dos diferentes métodos numéricos. Essas implementações analíticas servem como referência para verificar a exatidão das soluções obtidas por

meio dos métodos de Euler, Euler Modificado e Runge-Kutta de 4ª ordem. Dessa forma, é possível analisar de forma abrangente a eficácia de cada técnica, identificando quando e como os erros percentuais variam ao longo do tempo, à medida que os modelos de crescimento tumoral são aplicados.

Figura 29 – Implementação das soluções analíticas



```
1 import numpy as np
2
3 def Verhulst( t ):
4
5     N0 = 10 ** (9)
6     r = 10 ** (-2)
7     K = 10 ** (12)
8
9     N = K / ( 1 + ( ( K - N0 ) / N0 ) * np.exp( -r * t ) )
10     return N.reshape(-1, 1)
11
12
13 def Gompertz( t ):
14
15     N0 = 10 ** (9)
16     r = 10 ** (-2)
17     K = 10 ** (12)
18
19     N = K * np.exp( - np.log( K / N0 ) * np.exp( - r * t ) )
20     return N.reshape(-1, 1)
```

Fonte: O autor.

5.5 Funções auxiliares

Este trabalho requer a implementação de algumas funções essenciais para atender aos objetivos da pesquisa. Uma dessas funções é a PontosDeMalha (representada na Figura 30), a qual está contida no arquivo calculos_auxiliares. A função PontosDeMalha recebe como parâmetros o instante inicial t_i e final t_f da simulação, bem como um vetor que contém os passos de discretização vh que deseja-se analisar. Essa função retorna um vetor que indica a quantidade de pontos na malha para cada passo presente em vh .

Outras funções estão implementadas no arquivo resumo_de_dados, tal como a função ErroPercentual (Figura 31) que recebe como argumentos a solução analítica S_a e a solução numérica S_n , calcula e retorna o erro percentual absoluto entre esses dois vetores.

Outra função nesse mesmo arquivo é a VolumeCelulas (ilustrada na Figura 31). Ela aceita como parâmetros um vetor contendo dias específicos como elementos, juntamente

Figura 30 – Cálculo dos pontos de malha

```
Simulacao - calculos_auxiliares.py

1 import numpy as np
2
3 def PontosDeMalha( ti , tf , vh ):
4
5     P = np.array( [ ] , dtype = 'int' )
6
7     for h in vh:
8
9         Ph = ( tf - ti ) / h + 1
10        P = np.append( P , int( Ph ) )
11
12    return P
```

Fonte: O autor.

Figura 31 – Arquivo resumo_de_dados

```
Simulacao - resumo_de_dados.py

1 import numpy as np
2
3 def ErroPercentual( Sa , Sn ):
4
5     return np.abs ( ( ( Sa - Sn ) / Sa ) * 100 )
6
7 def VolumeCelulas( vi , N ):
8
9     Ni = np.array( [ ] )
10
11    for i in vi:
12
13        Ni = np.append( Ni , N[i] )
14
15    return Ni
```

Fonte: O autor.

com uma solução. Essa função tem como finalidade calcular o número de células em momentos de tempo específicos e retorná-los.

A seguir, este trabalho expõe os resultados e discussões, começando pela definição do problema e prosseguindo com a apresentação das simulações dos modelos de dinâmica populacional abordados neste estudo, bem como a avaliação dos métodos numéricos e dos modelos populacionais.

6 Resultados e Simulações

Neste capítulo, aborda-se a resolução de um problema de valor inicial que descreve o crescimento tumoral avascular com base nos modelos de Verhulst, Gompertz e Gatenby. São realizados estudos comparativos entre os métodos de solução utilizados, explorando as características únicas de cada curva de crescimento. Analisa-se o comportamento das soluções para diferentes tamanhos de passos de discretização e apresentam-se gráficos elucidativos dessas soluções ao longo do tempo. Além disso, avalia-se o erro percentual entre as soluções numéricas e as soluções analíticas, proporcionando uma visão abrangente e crítica dos resultados obtidos nas simulações. Essa análise contribui para uma compreensão mais profunda do crescimento tumoral e dos métodos de solução aplicados.

6.1 Definição do problema

Neste estudo, o principal objetivo da simulação de crescimento tumoral é prever o desenvolvimento de um tumor avascular com base em dados iniciais clinicamente observados e na consideração de diversos parâmetros essenciais. As condições iniciais representam o tamanho da colônia de células cancerosas $N_1(0)$ e normais $N_2(0)$. Denota-se, neste trabalho, o volume inicial de células tumorais igual a 10^9 e saudáveis equivalente a 10^{12} . Por meio dos modelos de Verhulst, Gompertz e Gatenby, aborda-se o crescimento tumoral em um contexto onde tratamentos não estão envolvidos, permitindo-se analisar a evolução natural do tumor ao longo do tempo. A simulação se estende ao longo de 1500 dias, a partir do exame de imagem utilizado para verificar o tamanho do tumor. Para alcançar esse propósito, consideram-se fatores cruciais (Tabela 4), como o coeficiente de competição entre células cancerosas e normais α_1 , a competição entre células normais e cancerosas α_2 , a taxa de crescimento do câncer em relação aos tecidos adjacentes r_1 , a taxa de crescimento das células normais r_2 e a capacidade de suporte das células normais K_1 e das células anormais K_2 . Ao empregar essas constantes aos modelos, estima-se o crescimento tumoral futuro com base em um valor inicial clinicamente observado, fornecendo informações sobre a progressão do tumor em cenários onde o tratamento não é considerado.

Discretiza-se o intervalo de simulação em passos de 0,5, 0,25 e 0,1. Esta subdivisão do intervalo permitirá uma análise detalhada do crescimento tumoral em diferentes resoluções temporais. Os resultados desse processo de discretização, que definem os instantes em que as simulações serão avaliadas, estão resumidos na Tabela 5. Essa tabela proporciona uma visão geral das datas específicas em que as simulações serão realizadas, refletindo a evolução temporal da colônia de células tumorais ao longo do estudo.

Tabela 4 – Parâmetros para simulação de câncer humano

Parâmetro	Valor	Unidade
r_1	1×10^{-2}	dia ⁻¹
r_2	1×10^{-3}	dia ⁻¹
α_1	9×10^{-2}	-
α_2	9×10^{-2}	-
K_1	1×10^{12}	células
K_2	1×10^{12}	células

Fonte: Rodrigues, Pinho e Mancera (2011).

Tabela 5 – Relação passo e quantidade de pontos de malha

Passo	Pontos de Malha
0,10	15001
0,25	6001
0,50	3001

Fonte: O autor.

A seguir, é apresentado o processo de obtenção das soluções numéricas para o problema de crescimento tumoral segundo as abordagens de Verhulst, Gompertz e Gatenby.

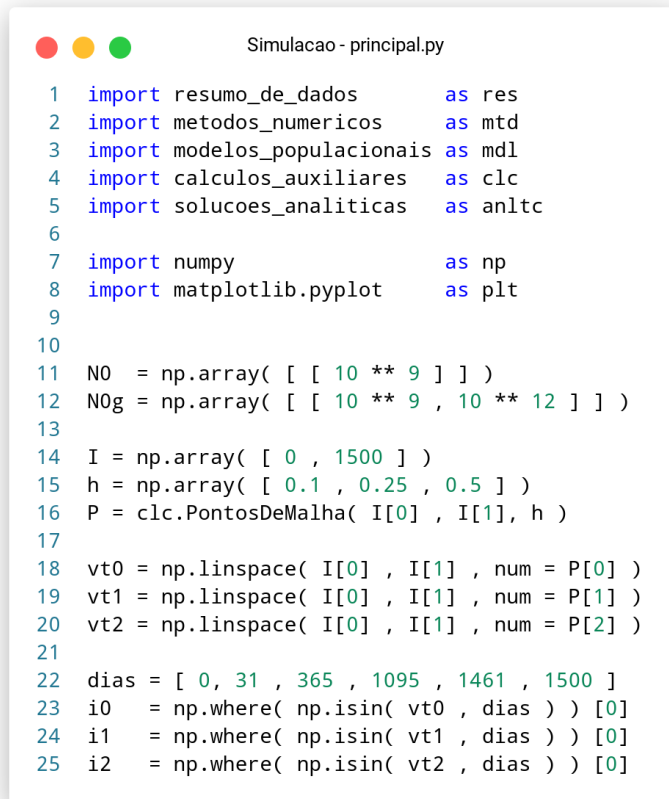
6.2 Simulações

Inicialmente, apresentam-se as etapas que precedem o início das simulações de crescimento tumoral em Python. No arquivo principal, carregam-se os módulos previamente criados neste trabalho (linhas 1 a 6 da Figura 32), em conjunto com as bibliotecas NumPy e Matplotlib (linhas 7 e 8 da Figura 32), antes de executar a função responsável pelas simulações numéricas de crescimento tumoral. Esta ação reveste-se de extrema relevância, uma vez que os módulos personalizados contêm as implementações específicas dos modelos de Gompertz, Gatenby e Verhulst desenvolvidos para este estudo.

Uma vez que as bibliotecas e módulos essenciais são carregados (linhas 1 a 8 da Figura 32), o próximo passo envolve a declaração das condições iniciais dos modelos de dinâmica populacional. As linhas 11 e 12 especificam os valores iniciais necessários para cada modelo, em que N_0 representa a condição inicial para os modelos de Verhulst e Gompertz, e N_{0g} representa a condição inicial para o modelo de Gatenby. Em seguida, na linha 14, declara-se um vetor I com dois elementos: o início e o fim do intervalo de simulação. Já nas linhas 15 e 16, realiza-se o cálculo dos pontos de malha, atribuindo o instante inicial e final da simulação, juntamente com o vetor de passos, à função `PontosDeMalha`, cujo retorno é atribuído à variável P .

Em seguida, são obtidos os vetores de tempo (representados como vt) e os res-

Figura 32 – Bibliotecas, módulos e variáveis necessárias para realizar as simulações



```

1 import resumo_de_dados      as res
2 import metodos_numericos    as mtd
3 import modelos_populacionais as mdl
4 import calculos_auxiliares   as clc
5 import solucoes_analiticas   as anltc
6
7 import numpy                 as np
8 import matplotlib.pyplot     as plt
9
10
11 N0 = np.array( [ [ 10 ** 9 ] ] )
12 N0g = np.array( [ [ 10 ** 9 , 10 ** 12 ] ] )
13
14 I = np.array( [ 0 , 1500 ] )
15 h = np.array( [ 0.1 , 0.25 , 0.5 ] )
16 P = clc.PontosDeMalha( I[0] , I[1], h )
17
18 vt0 = np.linspace( I[0] , I[1] , num = P[0] )
19 vt1 = np.linspace( I[0] , I[1] , num = P[1] )
20 vt2 = np.linspace( I[0] , I[1] , num = P[2] )
21
22 dias = [ 0 , 31 , 365 , 1095 , 1461 , 1500 ]
23 i0 = np.where( np.isin( vt0 , dias ) ) [0]
24 i1 = np.where( np.isin( vt1 , dias ) ) [0]
25 i2 = np.where( np.isin( vt2 , dias ) ) [0]

```

Fonte: O autor.

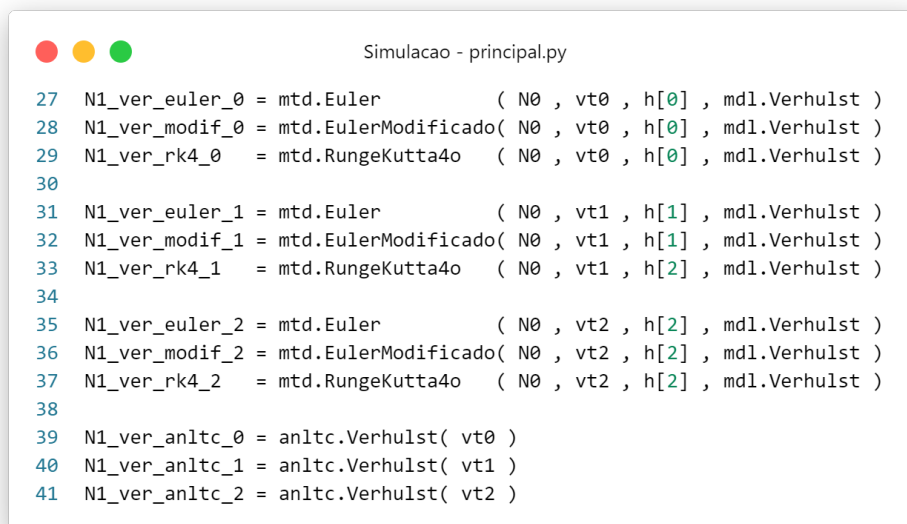
pectivos passos h correspondentes para os pontos da malha P (linhas 18 a 20 da Figura 32). Assim, o vetor $vt0$ abrange o intervalo de simulação com um passo de 0,5, $vt1$ é discretizado com um passo de 0,25, e $vt2$ é criado com um passo de 0,1. Por fim, as linhas 22 a 25 são responsáveis por armazenar os índices nos vetores de tempo que correspondem aos instantes de tempo de 0, 31, 365, 1095, 1461 e 1500 dias. Esses instantes representam, respectivamente, 0, 0,08, 1, 3, 4 e 4,11 anos-calendário.

A seguir, é apresentado o processo de obtenção das soluções numéricas por meio dos métodos de Euler, Euler Modificado e Runge-Kutta de 4ª ordem. Para cada um desses métodos, é exibido um gráfico que ilustra o crescimento tumoral ao longo do período de 1500 dias. Cada gráfico proporciona uma visão detalhada da evolução da população de células cancerosas e normais, permitindo uma análise comparativa dos resultados obtidos. Além disso, é realizada uma análise aprofundada das características e tendências observadas nos gráficos gerados por cada método. Por fim, é apresentado um resumo abrangente que compara os resultados de todos os modelos, fornecendo uma visão global das diferenças e semelhanças entre os modelos de Verhulst, Gompertz e Gatenby no contexto do crescimento tumoral.

6.2.1 Modelo de Verhulst

A Figura 33 exibe os comandos utilizados para realizar simulações do modelo de Verhulst. No contexto dessas simulações, as variáveis que contêm informações temporais seguem um padrão específico, composto por vários elementos separados por sublinhados. Isso inclui a variável matemática que representa o número de células cancerosas N_1 , os primeiros caracteres do nome do modelo, o método numérico utilizado e um dígito que indica o vetor de tempo associado. Esse padrão é aplicado de forma semelhante nas seções subsequentes. Nas linhas 27 a 29, são realizadas simulações com passos de 0,1 unidade de tempo para os métodos de Euler, Euler Modificado e Runge-Kutta de 4ª ordem, respectivamente. As linhas 31 a 33 repetem as simulações com um passo de 0,25, e as linhas 35 a 37 descrevem as simulações com um passo de 0,5 unidade de tempo. Por fim, nas linhas 39 a 41, são obtidas soluções analíticas para cada vetor de tempo.

Figura 33 – Obtendo soluções para o modelo de Verhulst



```

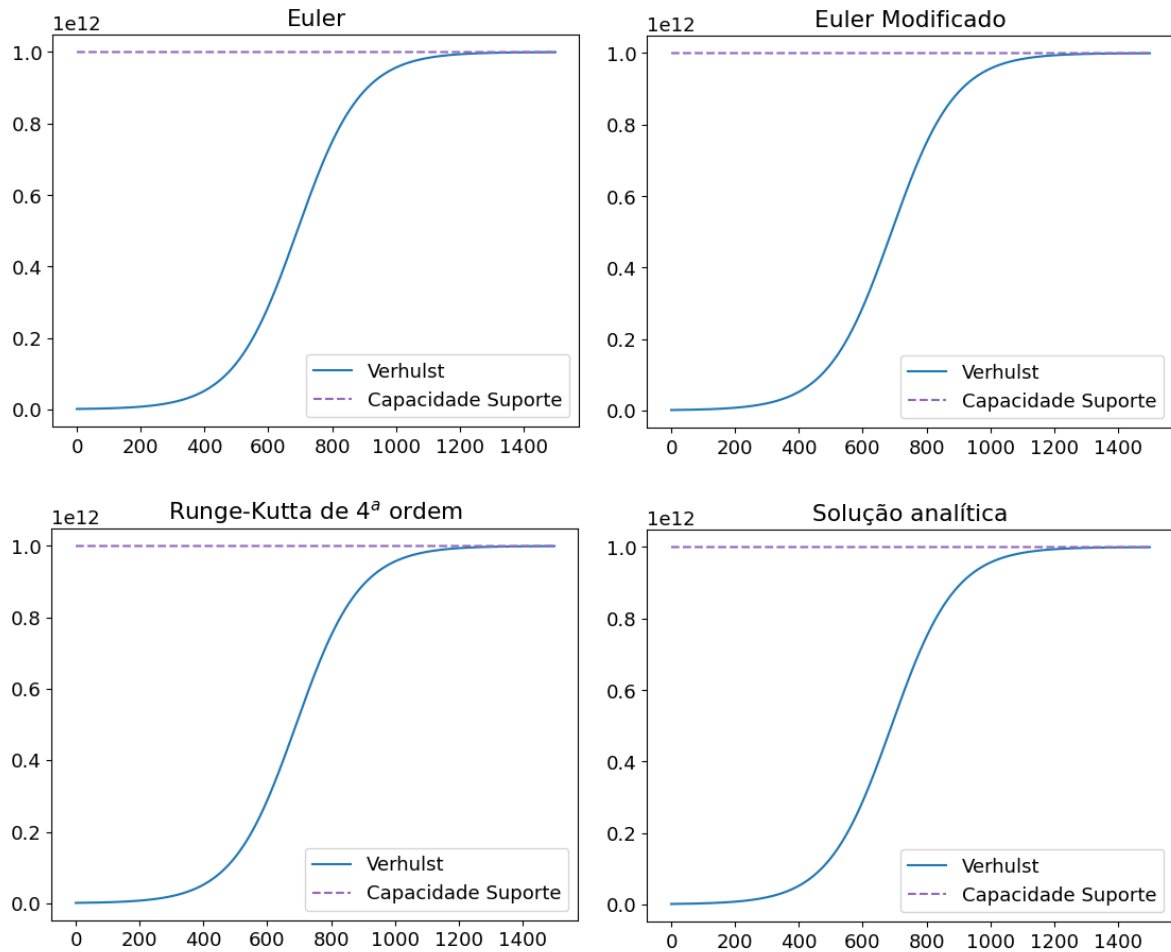
27 N1_ver_euler_0 = mtd.Euler          ( N0 , vt0 , h[0] , mdl.Verhulst )
28 N1_ver_modif_0 = mtd.EulerModificado( N0 , vt0 , h[0] , mdl.Verhulst )
29 N1_ver_rk4_0   = mtd.RungeKutta4o   ( N0 , vt0 , h[0] , mdl.Verhulst )
30
31 N1_ver_euler_1 = mtd.Euler          ( N0 , vt1 , h[1] , mdl.Verhulst )
32 N1_ver_modif_1 = mtd.EulerModificado( N0 , vt1 , h[1] , mdl.Verhulst )
33 N1_ver_rk4_1   = mtd.RungeKutta4o   ( N0 , vt1 , h[2] , mdl.Verhulst )
34
35 N1_ver_euler_2 = mtd.Euler          ( N0 , vt2 , h[2] , mdl.Verhulst )
36 N1_ver_modif_2 = mtd.EulerModificado( N0 , vt2 , h[2] , mdl.Verhulst )
37 N1_ver_rk4_2   = mtd.RungeKutta4o   ( N0 , vt2 , h[2] , mdl.Verhulst )
38
39 N1_ver_anltc_0 = anltc.Verhulst( vt0 )
40 N1_ver_anltc_1 = anltc.Verhulst( vt1 )
41 N1_ver_anltc_2 = anltc.Verhulst( vt2 )

```

Fonte: O autor.

Após obter os resultados utilizando diferentes técnicas de resolução, é possível representá-los graficamente em relação ao tempo. A Figura 34 exibe os gráficos resultantes das simulações, representando quatro abordagens de resolução distintas, com um passo de 0,1 unidade de tempo. Essas representações gráficas são equivalentes quando observadas. Independentemente do método ou passo utilizados neste trabalho, a visualização gráfica não é útil para comparar as soluções geradas para o modelo de Verhulst. Todos os modelos tratados neste trabalho compartilham dessa característica. Portanto, nas seções a seguir, que abordam os modelos de Gompertz e Gatenby, apresenta-se apenas a solução gerada pelo método de Runge-Kutta de 4ª ordem.

Figura 34 – Soluções para o modelo de Verhulst com passo de 0,1



Fonte: O autor.

Embora as soluções numéricas ainda não tenham sido avaliadas, cada solução apresenta características semelhantes do modelo de Verhulst. É evidente que o crescimento do câncer é mais lento nos instantes de tempo em que o volume de células cancerosas se aproxima da condição inicial e da capacidade de suporte do órgão. Por outro lado, nos instantes de tempo próximos ao ponto de inflexão, em aproximadamente, 650 dias, observa-se um crescimento mais rápido em comparação com os outros momentos. Inicialmente, quando os recursos necessários para o crescimento tumoral são abundantes no órgão, a colônia de células cancerosas experimenta um crescimento aproximadamente exponencial. No entanto, à medida que o volume se aproxima da capacidade do órgão, os recursos se tornam mais escassos, o que resulta em uma desaceleração do crescimento da colônia.

Nesta etapa, para avaliar as técnicas de solução, é fundamental obter os valores numéricos da quantidade de células cancerosas em momentos específicos para cada técnica e passo de simulação. A função `VolumeCelulas` é empregada para calcular essas quantidades de células nos momentos desejados. As Tabelas 6, 7 e 8 exibem o número de

células tumorais nos instantes de 0, 31, 365, 1095, 1461 e 1500 dias após o exame clínico, para cada técnica de solução. A Tabela 6 apresenta esses resultados para um passo de discretização de 0,1, a Tabela 7 para 0,25 e a Tabela 8 para 0,5.

Tabela 6 – Resultado da simulação para o modelo de Verhulst utilizando $h = 0,1$

Dias	Técnica de Solução			
	Euler	Euler Modificado	RK4	Analítico
0	$1,00000000 \times 10^0$	$1,00000000 \times 10^0$	$1,00000000 \times 10^0$	$1,00000000 \times 10^0$
31	$1,36271948 \times 10^0$	$1,36292972 \times 10^0$	$1,36292979 \times 10^0$	$1,36292979 \times 10^0$
365	$3,70211572 \times 10^1$	$3,70849000 \times 10^1$	$3,70849210 \times 10^1$	$3,70849210 \times 10^1$
1095	$9,82737908 \times 10^{11}$	$9,82761884 \times 10^{11}$	$9,82761907 \times 10^{11}$	$9,82761907 \times 10^{11}$
1461	$9,99549023 \times 10^{11}$	$9,99548843 \times 10^{11}$	$9,99548844 \times 10^{11}$	$9,99548844 \times 10^{11}$
1500	$9,99694678 \times 10^{11}$	$9,99694496 \times 10^{11}$	$9,99694497 \times 10^{11}$	$9,99694497 \times 10^{11}$

Fonte: O autor.

Tabela 7 – Resultado da simulação para o modelo de Verhulst utilizando $h = 0,25$

Dias	Técnica de Solução			
	Euler	Euler Modificado	RK4	Analítico
0	$1,00000000 \times 10^0$	$1,00000000 \times 10^0$	$1,00000000 \times 10^0$	$1,00000000 \times 10^0$
31	$1,36240459 \times 10^0$	$1,36292935 \times 10^0$	$1,36292979 \times 10^0$	$1,36292979 \times 10^0$
365	$3,69258573 \times 10^1$	$3,70847895 \times 10^1$	$3,70849210 \times 10^{11}$	$3,70849210 \times 10^{10}$
1095	$9,82701832 \times 10^{11}$	$9,82761766 \times 10^{11}$	$9,82761907 \times 10^{11}$	$9,82761907 \times 10^{11}$
1461	$9,99549294 \times 10^{11}$	$9,99548838 \times 10^{11}$	$9,99548844 \times 10^{12}$	$9,99548844 \times 10^{11}$
1500	$9,99694951 \times 10^{11}$	$9,99694493 \times 10^{11}$	$9,99694497 \times 10^{12}$	$9,99694497 \times 10^{11}$

Fonte: O autor.

Tabela 8 – Resultado da simulação para o modelo de Verhulst utilizando $h = 0,5$

Dias	Técnica de Solução			
	Euler	Euler Modificado	RK4	Analítico
0	$1,00000000 \times 10^0$	$1,00000000 \times 10^0$	$1,00000000 \times 10^0$	$1,00000000 \times 10^0$
31	$1,36188132 \times 10^0$	$1,36292805 \times 10^0$	$1,36292979 \times 10^0$	$1,36292979 \times 10^0$
365	$3,67679406 \times 10^1$	$3,70843960 \times 10^{10}$	$3,70849210 \times 10^{10}$	$3,70849210 \times 10^{10}$
1095	$9,82641496 \times 10^{11}$	$9,82761342 \times 10^{11}$	$9,82761907 \times 10^{11}$	$9,82761907 \times 10^{11}$
1461	$9,99549750 \times 10^{11}$	$9,99548822 \times 10^{11}$	$9,99548844 \times 10^{11}$	$9,99548844 \times 10^{11}$
1500	$9,99695408 \times 10^{11}$	$9,99694482 \times 10^{11}$	$9,99694497 \times 10^{11}$	$9,99694497 \times 10^{11}$

Fonte: O autor.

Os resultados apresentados nas Tabelas 6, 7 e 8 destacam de forma nítida o padrão de crescimento tumoral de acordo com o modelo de Verhulst. Durante o primeiro mês,

observa-se que a colônia de células cancerosas não exibe um crescimento tão vigoroso em comparação com o período que abrange de um mês a três anos. Esse crescimento mais gradual, que é caracterizado nos primeiros meses, se repete aproximadamente entre o terceiro e o quarto ano.

Quanto ao erro numérico resultante das técnicas de solução, as Tabelas 9, 10 e 11 exibem o erro percentual absoluto para os passos de 0,1, 0,25 e 0,5 unidades de tempo nos momentos 0, 31, 365, 1095, 1461 e 1500 dias. Na Tabela 9 são apresentados os resultados para o método de Euler. A Tabela 10 contém os resultados para o método de Euler Modificado. Por fim, a Tabela 11 apresenta os erros para o método Runge-Kutta de 4ª ordem.

Tabela 9 – Erro numérico percentual para o modelo de Verhulst solucionado pelo método de Euler

Dias	Passo de discretização		
	0,1	0,25	0,5
0	0	0	0
31	$1,54 \times 10^{-2}$	$3,85 \times 10^{-2}$	$7,69 \times 10^{-2}$
365	$1,71 \times 10^{-1}$	$4,29 \times 10^{-1}$	$8,55 \times 10^{-1}$
1095	$2,44 \times 10^{-3}$	$6,11 \times 10^{-3}$	$1,22 \times 10^{-2}$
1461	$1,80 \times 10^{-5}$	$4,51 \times 10^{-5}$	$9,06 \times 10^{-5}$
1500	$1,81 \times 10^{-5}$	$4,54 \times 10^{-5}$	$9,12 \times 10^{-5}$

Fonte: O autor.

Tabela 10 – Erro numérico percentual para o modelo de Verhulst solucionado pelo método de Euler Modificado

Dias	Passo de discretização		
	0,1	0,25	0,5
0	0	0	0
31	$5,14 \times 10^{-6}$	$3,21 \times 10^{-5}$	$1,28 \times 10^{-4}$
365	$5,68 \times 10^{-5}$	$3,54 \times 10^{-4}$	$1,41 \times 10^{-3}$
1095	$2,30 \times 10^{-6}$	$1,44 \times 10^{-5}$	$5,74 \times 10^{-5}$
1461	$8,73 \times 10^{-8}$	$5,46 \times 10^{-7}$	$2,18 \times 10^{-6}$
1500	$6,11 \times 10^{-8}$	$3,82 \times 10^{-7}$	$1,53 \times 10^{-6}$

Fonte: O autor.

A análise das tabelas de erros numéricos revela que à medida que o passo de discretização diminui, o erro numérico percentual também diminui. Isso é evidente quando observa-se a ordem do erro para cada passo em relação aos dias de crescimento tumoral. Embora um passo menor resulte em uma solução mais próxima da solução exata, é importante notar que, dada a escala do problema, os erros associados a um passo maior, como

Tabela 11 – Erro numérico percentual para o modelo de Verhulst solucionado pelo método de Runge-Kutta de 4ª ordem

Dias	Passo de discretização		
	0,1	0,25	0,5
0	0	0	0
31	$3,50 \times 10^{-13}$	$1,00 \times 10^{-11}$	$1,59 \times 10^{-10}$
365	$3,25 \times 10^{-12}$	$1,08 \times 10^{-10}$	$1,73 \times 10^{-09}$
1095	$1,37 \times 10^{-13}$	$3,84 \times 10^{-12}$	$6,16 \times 10^{-11}$
1461	$1,10 \times 10^{-13}$	$1,59 \times 10^{-13}$	$2,48 \times 10^{-12}$
1500	$7,33 \times 10^{-14}$	$1,22 \times 10^{-13}$	$1,76 \times 10^{-12}$

Fonte: O autor.

0,5, são próximos de zero. Isso significa que o problema pode ser resolvido com menos pontos na malha, economizando recursos computacionais. Uma comparação mais detalhada entre as técnicas numéricas de solução é realizada posteriormente neste trabalho.

Observa-se que os erros são menores para o 1461º e 1500º dias de crescimento do tumor. Essa ocorrência se deve ao fato de que esses pontos estão mais próximos da capacidade suporte do órgão, onde ocorre o comprometimento extenso do órgão pelo câncer e a curva apresenta taxas de variação cada vez menores. Isso torna a técnica numérica de solução mais precisa nesses pontos, resultando em erros percentuais menores.

6.2.2 Modelo de Gompertz

Para a simulação do crescimento tumoral avascular utilizando o modelo de Gompertz, empregam-se as mesmas estruturas de código previamente utilizadas na simulação de acordo com o modelo de Verhulst, conforme demonstrado na subseção 6.2.1. A Figura 35 apresenta os comandos relevantes para essa simulação.

A Figura 36 exibe a representação gráfica da solução para o modelo de Gompertz. Nessa simulação, inicialmente o tumor encontra-se em um ambiente rico em recursos e espaço, o que confere um crescimento mais rápido e desimpedido. À medida que o tempo avança, o crescimento no modelo de Gompertz começa a desacelerar. Isso ocorre porque, à medida que o tumor cresce, ele consome mais recursos e espaço, tornando-se mais competitivo e enfrentando restrições crescentes.

As Tabelas 12, 13 e 14 apresentam uma análise numérica das simulações realizadas em instantes de tempo específicos, bem como os intervalos de discretização utilizados nesses experimentos. Esses resultados destacam o notável crescimento inicial, onde em apenas um ano, a população de células tumorais já alcança a magnitude de 10^{11} . Após esse ponto, observa-se um estágio de crescimento mais moderado, devido às crescentes

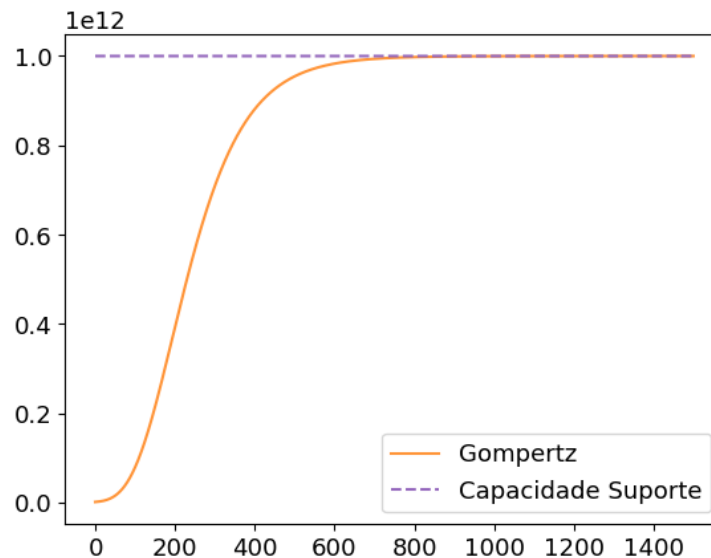
Figura 35 – Obtendo soluções para o modelo de Gompertz

```

Simulacao - principal.py
43 N1_gom_euler_0 = mtd.Euler          ( N0 , vt0 , h[0] , mdl.Gompertz )
44 N1_gom_modif_0 = mtd.EulerModificado( N0 , vt0 , h[0] , mdl.Gompertz )
45 N1_gom_rk4_0   = mtd.RungeKutta4o   ( N0 , vt0 , h[0] , mdl.Gompertz )
46
47 N1_gom_euler_1 = mtd.Euler          ( N0 , vt1 , h[1] , mdl.Gompertz )
48 N1_gom_modif_1 = mtd.EulerModificado( N0 , vt1 , h[1] , mdl.Gompertz )
49 N1_gom_rk4_1   = mtd.RungeKutta4o   ( N0 , vt1 , h[1] , mdl.Gompertz )
50
51 N1_gom_euler_2 = mtd.Euler          ( N0 , vt2 , h[2] , mdl.Gompertz )
52 N1_gom_modif_2 = mtd.EulerModificado( N0 , vt2 , h[2] , mdl.Gompertz )
53 N1_gom_rk4_2   = mtd.RungeKutta4o   ( N0 , vt2 , h[2] , mdl.Gompertz )
54
55 N1_gom_anltc_0 = anltc.Verhulst( vt0 )
56 N1_gom_anltc_1 = anltc.Verhulst( vt1 )
57 N1_gom_anltc_2 = anltc.Verhulst( vt2 )

```

Fonte: O autor.

Figura 36 – Simulação segundo o modelo de Gompertz (RK4 com $h = 0,1$)

Fonte: O autor.

restrições de expansão.

Os erros percentuais ao resolver o problema de crescimento tumoral com base no modelo de Gompertz estão registrados nas Tabelas [15](#), [16](#) e [17](#).

Especificamente, o modelo de Gompertz gera um erro percentual diferente de zero no primeiro dia (dia inicial). A solução analítica, conforme escrita em Python, leva em consideração a condição inicial do problema. No entanto, essa implementação utiliza funções do NumPy para cálculos exponenciais e logarítmicos, o que torna as operações suscetíveis a erros de arredondamento. A diferença, em módulo, entre a quantidade inicial

Tabela 12 – Resultado da simulação para o modelo de Gompertz utilizando $h = 0,1$

Dias	Técnica de Solução			
	Euler	Euler Modificado	RK4	Analítico
0	$1,00000000 \times 10^{09}$	$1,00000000 \times 10^{09}$	$1,00000000 \times 10^{09}$	$1,00000000 \times 10^{09}$
31	$6,28026974 \times 10^{09}$	$6,30457769 \times 10^{09}$	$6,30462314 \times 10^{09}$	$6,30462314 \times 10^{09}$
365	$8,35423158 \times 10^{11}$	$8,35653792 \times 10^{11}$	$8,35654226 \times 10^{11}$	$8,35654226 \times 10^{11}$
1095	$9,99878966 \times 10^{11}$	$9,99878720 \times 10^{11}$	$9,99878721 \times 10^{11}$	$9,99878721 \times 10^{11}$
1461	$9,99996891 \times 10^{11}$	$9,99996879 \times 10^{11}$	$9,99996879 \times 10^{11}$	$9,99996879 \times 10^{11}$
1500	$9,99997895 \times 10^{11}$	$9,99997887 \times 10^{11}$	$9,99997887 \times 10^{11}$	$9,99997887 \times 10^{11}$

Fonte: O autor.

Tabela 13 – Resultado da simulação para o modelo de Gompertz utilizando $h = 0,25$

Dias	Técnica de Solução			
	Euler	Euler Modificado	RK4	Analítico
0	$1,00000000 \times 10^{09}$	$1,00000000 \times 10^{09}$	$1,00000000 \times 10^{09}$	$1,00000000 \times 10^{09}$
31	$6,24412180 \times 10^{09}$	$6,30434088 \times 10^{09}$	$6,30462313 \times 10^{09}$	$6,30462314 \times 10^{09}$
365	$8,35075996 \times 10^{11}$	$8,35651527 \times 10^{11}$	$8,35654226 \times 10^{11}$	$8,35654226 \times 10^{10}$
1095	$9,99879334 \times 10^{11}$	$9,99878718 \times 10^{11}$	$9,99878721 \times 10^{11}$	$9,99878721 \times 10^{11}$
1461	$9,99996909 \times 10^{11}$	$9,99996879 \times 10^{11}$	$9,99996879 \times 10^{11}$	$9,99996879 \times 10^{11}$
1500	$9,99997908 \times 10^{11}$	$9,99997887 \times 10^{11}$	$9,99997887 \times 10^{11}$	$9,99997887 \times 10^{11}$

Fonte: O autor.

Tabela 14 – Resultado da simulação para o modelo de Gompertz utilizando $h = 0,5$

Dias	Técnica de Solução			
	Euler	Euler Modificado	RK4	Analítico
0	$1,00000000 \times 10^{09}$	$1,00000000 \times 10^{09}$	$1,00000000 \times 10^{09}$	$1,00000000 \times 10^{09}$
31	$6,18487543 \times 10^{09}$	$6,30350586 \times 10^{09}$	$6,30462309 \times 10^{09}$	$6,30462314 \times 10^{09}$
365	$8,34495885 \times 10^{11}$	$8,35643514 \times 10^{11}$	$8,35654225 \times 10^{11}$	$8,35654226 \times 10^{10}$
1095	$9,99879947 \times 10^{11}$	$9,99878709 \times 10^{11}$	$9,99878721 \times 10^{11}$	$9,99878721 \times 10^{11}$
1461	$9,99996939 \times 10^{11}$	$9,99996879 \times 10^{11}$	$9,99996879 \times 10^{11}$	$9,99996879 \times 10^{11}$
1500	$9,99997929 \times 10^{11}$	$9,99997887 \times 10^{11}$	$9,99997887 \times 10^{11}$	$9,99997887 \times 10^{11}$

Fonte: O autor.

de células tumorais, 10^{09} , e o valor inicial da solução para o problema de Gompertz é de $2,38418579^{-07}$, evidenciando o erro de arredondamento resultante do encadeamento de funções exponenciais e logarítmicas no NumPy.

Os erros percentuais exibem um comportamento semelhante ao apresentado no modelo de Verhulst. Conforme a curva se torna mais suave, os erros tendem a diminuir. Ainda, possuem menores erros para passos menores em cada método de solução. Nota-

Tabela 15 – Erro numérico percentual para o modelo de Gompertz solucionado pelo método de Euler

Dias	Passo de discretização		
	0,1	0,25	0,5
0	$2,38 \times 10^{-14}$	$2,38 \times 10^{-14}$	$1,90 \times 10^{-14}$
31	$3,87 \times 10^{-01}$	$9,60 \times 10^{-01}$	$1,39 \times 10^{+00}$
365	$2,76 \times 10^{-02}$	$6,92 \times 10^{-02}$	$8,55 \times 10^{-01}$
1095	$2,45 \times 10^{-05}$	$6,13 \times 10^{-05}$	$1,23 \times 10^{-04}$
1461	$1,20 \times 10^{-06}$	$2,80 \times 10^{-06}$	$5,98 \times 10^{-06}$
1500	$8,54 \times 10^{-07}$	$2,13 \times 10^{-06}$	$4,25 \times 10^{-06}$

Fonte: O autor.

Tabela 16 – Erro numérico percentual para o modelo de Gompertz solucionado pelo método de Euler Modificado

Dias	Passo de discretização		
	0,1	0,25	0,5
0	$2,38 \times 10^{-14}$	$2,38 \times 10^{-14}$	$2,38 \times 10^{-14}$
31	$7,21 \times 10^{-04}$	$4,48 \times 10^{-03}$	$1,77 \times 10^{-02}$
365	$5,19 \times 10^{-05}$	$3,23 \times 10^{-04}$	$1,28 \times 10^{-03}$
1095	$4,93 \times 10^{-08}$	$3,07 \times 10^{-07}$	$1,22 \times 10^{-06}$
1461	$1,46 \times 10^{-09}$	$9,10 \times 10^{-09}$	$3,62 \times 10^{-08}$
1500	$1,00 \times 10^{-09}$	$6,25 \times 10^{-09}$	$2,49 \times 10^{-08}$

Fonte: O autor.

velmente, o método de Runge-Kutta com um passo de discretização de 0,1 exibe erros muito pequenos em 1 095 dias. Até esse ponto, o método RK4 superestima os valores da solução analítica. No entanto, em 1461 e 1500 dias, ele começa a subestimar a solução numérica em relação a exata, indicando que, em algum momento, a solução exata e a solução analítica se tornam muito próximas.

6.2.3 Modelo de Gatenby

Assim como realizado para os modelos de Verhulst e Gompertz, inicialmente, são obtidas as simulações para o modelo de Gatenby, conforme ilustrado na Figura 37. O resultado gráfico dessas simulações pode ser visualizado na Figura 38.

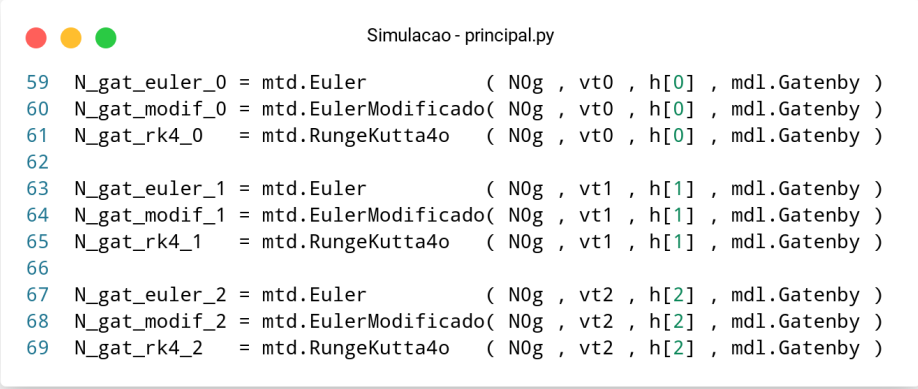
Nesse modelo, a dinâmica populacional de células cancerosas e normais é influenciada pela disponibilidade desses recursos. À medida que as células cancerosas crescem, elas consomem recursos, o que, por sua vez, afeta as células normais e as torna menos competitivas, fazendo com que diminuam de volume em comparação as células cancerosas. Embora neste trabalho inevitavelmente as células anormais cresçam, a resistência

Tabela 17 – Erro numérico percentual para o modelo de Gompertz solucionado pelo método Runge-Kutta de 4ª ordem

Dias	Passo de discretização		
	0,1	0,25	0,5
0	$2,38 \times 10^{-14}$	$2,38 \times 10^{-14}$	$2,38 \times 10^{-14}$
31	$1,14 \times 10^{-09}$	$4,41 \times 10^{-08}$	$6,98 \times 10^{-07}$
365	$5,37 \times 10^{-11}$	$2,09 \times 10^{-09}$	$3,30 \times 10^{-08}$
1095	$3,66 \times 10^{-14}$	$1,39 \times 10^{-12}$	$2,27 \times 10^{-11}$
1461	$1,10 \times 10^{-13}$	$1,10 \times 10^{-13}$	$6,22 \times 10^{-13}$
1500	$1,10 \times 10^{-13}$	$4,88 \times 10^{-14}$	$4,15 \times 10^{-13}$

Fonte: O autor.

Figura 37 – Obtendo soluções para o modelo de Gatenby



```

Simulacao - principal.py
59 N_gat_euler_0 = mtd.Euler          ( N0g , vt0 , h[0] , mdl.Gatenby )
60 N_gat_modif_0 = mtd.EulerModificado( N0g , vt0 , h[0] , mdl.Gatenby )
61 N_gat_rk4_0   = mtd.RungeKutta4o   ( N0g , vt0 , h[0] , mdl.Gatenby )
62
63 N_gat_euler_1 = mtd.Euler          ( N0g , vt1 , h[1] , mdl.Gatenby )
64 N_gat_modif_1 = mtd.EulerModificado( N0g , vt1 , h[1] , mdl.Gatenby )
65 N_gat_rk4_1   = mtd.RungeKutta4o   ( N0g , vt1 , h[1] , mdl.Gatenby )
66
67 N_gat_euler_2 = mtd.Euler          ( N0g , vt2 , h[2] , mdl.Gatenby )
68 N_gat_modif_2 = mtd.EulerModificado( N0g , vt2 , h[2] , mdl.Gatenby )
69 N_gat_rk4_2   = mtd.RungeKutta4o   ( N0g , vt2 , h[2] , mdl.Gatenby )

```

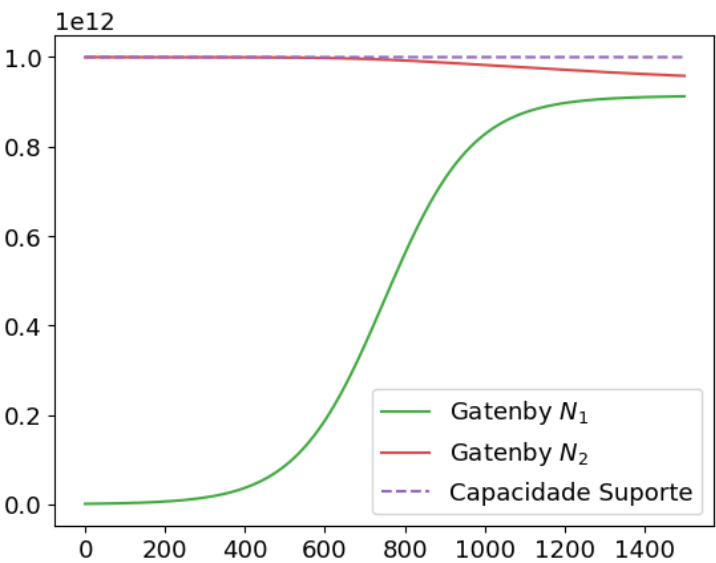
Fonte: O autor.

das células saudáveis atrasa a disseminação do tumor ao longo do órgão.

No início da simulação, representada na Figura 38, o crescimento das células tumorais é notavelmente lento. Entretanto, após aproximadamente um ano e meio, há um aumento acentuado na taxa de crescimento. Por outro lado, as células normais exibem um crescimento moderado em todos os intervalos de tempo, em comparação com as células cancerosas. É importante destacar que, embora a simulação abranja um período de 1500 dias, ambas as colônias de células convergem para um limiar comum. Essa convergência ocorre a um nível diferente da capacidade suporte do órgão, indicando que, em condições específicas, as células normais superarão as células cancerosas em números absolutos.

Numericamente, as Tabelas 18, 19 e 20 fornecem o número de células cancerosas em momentos específicos, considerando cada técnica de solução. A Tabela 18 mostra as simulações com um passo de 0,1, enquanto a Tabela 19 aborda o passo de 0,25, e a Tabela 20 engloba o passo de 0,5. Correspondentemente, as Tabelas 21, 22 e 23 contêm

Figura 38 – Simulação segundo o modelo de Gatenby (RK4 com $h = 0,1$)



Fonte: O autor.

as mesmas informações, mas referentes às células normais.

Tabela 18 – Resultado da simulação segundo o modelo de Gatenby para células tumorais, utilizando $h = 0,1$

Dias	Técnica de Solução		
	Euler	Euler Modificado	RK4
0	$1,00000000 \times 10^9$	$1,00000000 \times 10^9$	$1,00000000 \times 10^9$
31	$1,32526733 \times 10^9$	$1,32543661 \times 10^9$	$1,32543666 \times 10^9$
365	$2,68740158 \times 10^{10}$	$2,69127388 \times 10^{10}$	$2,69127505 \times 10^{10}$
1095	$8,74228141 \times 10^{11}$	$8,74287855 \times 10^{11}$	$8,74287891 \times 10^{11}$
1461	$9,11818899 \times 10^{11}$	$9,11820138 \times 10^{11}$	$9,11820140 \times 10^{11}$
1500	$9,12384086 \times 10^{11}$	$9,12385061 \times 10^{11}$	$9,12385063 \times 10^{11}$

Fonte: O autor.

6.3 Avaliação dos métodos numéricos

Conforme era esperado, os resultados obtidos na seção de simulações (consulte a seção 6.2) demonstram que o método de Runge-Kutta de 4ª ordem é mais preciso do que o método de Euler Modificado, que, por sua vez, é mais preciso do que o método de Euler. Isso ocorre devido ao fato de que o RK4 utiliza um processo de quatro etapas para calcular a solução, permitindo assim a captura de curvas de função mais complexas com maior precisão. Por outro lado, o método de Euler Modificado leva em consideração a inclinação média entre dois pontos para fazer uma estimativa mais precisa da solução. Por último, o método de Euler utiliza uma aproximação linear para calcular as soluções,

Tabela 19 – Resultado da simulação segundo o modelo de Gatenby para células tumorais, utilizando $h = 0,25$

Dias	Técnica de Solução		
	Euler	Euler Modificado	RK4
0	$1,00000000 \times 10^0$	$1,00000000 \times 10^0$	$1,00000000 \times 10^0$
31	$1,32501375 \times 10^0$	$1,32543634 \times 10^0$	$1,32543666 \times 10^0$
365	$2,68160982 \times 10^1$	$2,69126777 \times 10^1$	$2,69127505 \times 10^1$
1095	$8,74138301 \times 10^2$	$8,74287670 \times 10^2$	$8,74287891 \times 10^2$
1461	$9,11817039 \times 10^2$	$9,11820126 \times 10^2$	$9,11820140 \times 10^2$
1500	$9,12382621 \times 10^2$	$9,12385052 \times 10^2$	$9,12385063 \times 10^2$

Fonte: O autor.

Tabela 20 – Resultado da simulação segundo o modelo de Gatenby para células tumorais, utilizando $h = 0,5$

Dias	Técnica de Solução		
	Euler	Euler Modificado	RK4
0	$1,00000000 \times 10^0$	$1,00000000 \times 10^0$	$1,00000000 \times 10^0$
31	$1,32459224 \times 10^0$	$1,32543538 \times 10^0$	$1,32543666 \times 10^0$
365	$2,67200578 \times 10^1$	$2,69124598 \times 10^1$	$2,69127505 \times 10^1$
1095	$8,73987988 \times 10^2$	$8,74287007 \times 10^2$	$8,74287891 \times 10^2$
1461	$9,11813945 \times 10^2$	$9,11820083 \times 10^2$	$9,11820140 \times 10^2$
1500	$9,12380185 \times 10^2$	$9,12385020 \times 10^2$	$9,12385063 \times 10^2$

Fonte: O autor.

tornando-o menos preciso, especialmente quando os passos de discretização são maiores. A aproximação linear resulta em erros maiores, reduzindo sua precisão.

É evidente que o comportamento das funções afeta os erros percentuais. Conforme discutido anteriormente, passos menores geram soluções numéricas melhores. No entanto, como esses métodos são técnicas de aproximação, eles podem subestimar ou superestimar a solução exata, resultando em erros muito pequenos em pontos específicos de interseção entre a solução numérica e a solução exata.

Por um lado, o método de Runge-Kutta gera soluções mais precisas do que os métodos de Euler. No entanto, as soluções geradas pelo método de Euler exigem menos recursos computacionais. É importante destacar que, dado o tamanho da escala do problema, todos os erros apresentados são aceitáveis. Portanto, pode-se optar por abordar a resolução do problema de forma mais simplificada, utilizando o método de Euler, ou escolher o método de Runge-Kutta com um intervalo de discretização maior para solucionar os Problemas de Valor Inicial em estudo, o que resultaria na redução do número de cálculos necessários a serem executados pelo computador.

Tabela 21 – Resultado da simulação segundo o modelo de Gatenby para células normais, utilizando $h = 0,1$

Dias	Técnica de Solução		
	Euler	Euler Modificado	RK4
0	$1,00000000 \times 10^{12}$	$1,00000000 \times 10^{12}$	$1,00000000 \times 10^{12}$
31	$9,99996826 \times 10^{11}$	$9,99996824 \times 10^{11}$	$9,99996824 \times 10^{11}$
365	$9,99763331 \times 10^{11}$	$9,99762975 \times 10^{11}$	$9,99762975 \times 10^{11}$
1095	$9,76993822 \times 10^{11}$	$9,76978602 \times 10^{11}$	$9,76978598 \times 10^{11}$
1461	$9,59632579 \times 10^{11}$	$9,59622340 \times 10^{11}$	$9,59622337 \times 10^{11}$
1500	$9,58101062 \times 10^{11}$	$9,58091284 \times 10^{11}$	$9,58091281 \times 10^{11}$

Fonte: O autor.

Tabela 22 – Resultado da simulação segundo o modelo de Gatenby para células normais, utilizando $h = 0,25$

Dias	Técnica de Solução		
	Euler	Euler Modificado	RK4
0	$1,00000000 \times 10^{12}$	$1,00000000 \times 10^{12}$	$1,00000000 \times 10^{12}$
31	$9,99996828 \times 10^{11}$	$9,99996824 \times 10^{11}$	$9,99996824 \times 10^{11}$
365	$9,99763864 \times 10^{11}$	$9,99762976 \times 10^{11}$	$9,99762975 \times 10^{11}$
1095	$9,77016662 \times 10^{11}$	$9,76978624 \times 10^{11}$	$9,76978598 \times 10^{11}$
1461	$9,59647945 \times 10^{11}$	$9,59622357 \times 10^{11}$	$9,59622337 \times 10^{11}$
1500	$9,58115737 \times 10^{11}$	$9,58091300 \times 10^{11}$	$9,58091281 \times 10^{11}$

Fonte: O autor.

Considerando que o método de Runge-Kutta gera soluções com erros percentuais menores, a análise dos modelos populacionais a seguir utiliza os resultados desse método com um passo de discretização de 0,1 dias.

6.4 Avaliação dos modelos populacionais

Conforme ilustrado na seção [6.2](#), os modelos populacionais abordados neste estudo compartilham uma característica comum: a quantidade de células tumorais atinge um limite, ou seja, não ultrapassa um determinado volume. No entanto, a diferença fundamental entre eles reside na velocidade com que o tumor alcança esse limite. A Figura [39](#) apresenta uma representação gráfica do crescimento tumoral e saudável (para o modelo de Gatenby) para os diferentes modelos examinados neste trabalho.

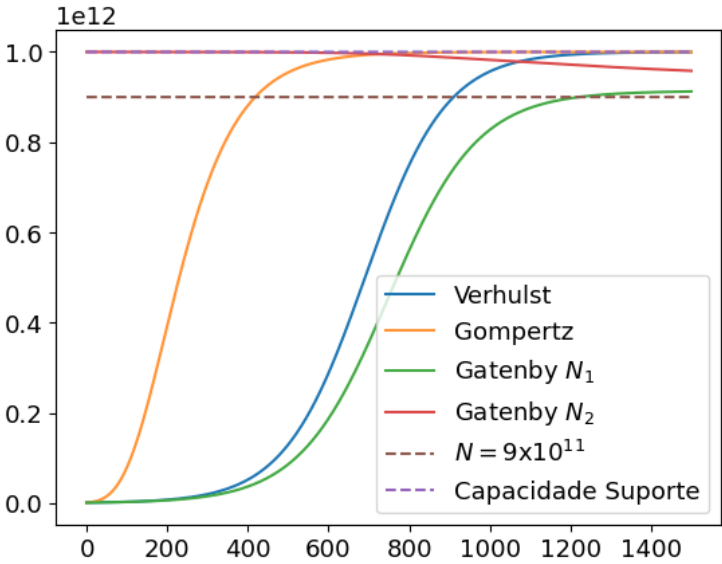
Inicialmente, o número de células tumorais, de acordo com o modelo de Gompertz, apresenta uma taxa de variação muito alta, o que o diferencia das curvas dos modelos de Verhulst e Gatenby (para células tumorais). Essas curvas descrevem a quantidade de

Tabela 23 – Resultado da simulação segundo o modelo de Gatenby para células normais, utilizando $h = 0,5$

Dias	Técnica de Solução		
	Euler	Euler Modificado	RK4
0	$1,00000000 \times 10^{12}$	$1,00000000 \times 10^{12}$	$1,00000000 \times 10^{12}$
31	$9,99996832 \times 10^{11}$	$9,99996824 \times 10^{11}$	$9,99996824 \times 10^{11}$
365	$9,99764748 \times 10^{11}$	$9,99762978 \times 10^{11}$	$9,99762975 \times 10^{11}$
1095	$9,77054735 \times 10^{11}$	$9,76978705 \times 10^{11}$	$9,76978598 \times 10^{11}$
1461	$9,59673562 \times 10^{11}$	$9,59622418 \times 10^{11}$	$9,59622337 \times 10^{11}$
1500	$9,58140201 \times 10^{11}$	$9,58091359 \times 10^{11}$	$9,58091281 \times 10^{11}$

Fonte: O autor.

Figura 39 – Simulações (RK4 com $h = 0,1$)



Fonte: O autor.

células tumorais de forma semelhante até cerca de 300 dias.

As diferentes taxas de crescimento tumoral dos modelos analisados neste estudo influenciam diretamente a velocidade com que cada curva de solução atinge seu limiar. No caso do modelo de Gompertz, a curva atinge rapidamente a capacidade suporte do órgão em aproximadamente 700 dias. Em contraste, a curva do modelo de Verhulst alcança a capacidade suporte do órgão em cerca de 1200 dias, enquanto o modelo de Gatenby estabiliza suas células saudáveis em um período que ultrapassa os 1500 dias.

As simulações do crescimento tumoral conduzidas neste estudo se estendem até 1500 dias. No entanto, é importante observar que, na prática, um paciente dificilmente sobreviveria até esse ponto devido às complicações associadas a um tumor com um volume da ordem de 9×10^{11} células cancerosas (RODRIGUES; PINHO; MANCERA, 2011), equivalente a aproximadamente 900 gramas. Nesse contexto, o modelo de Gompertz

atinge essa marca em cerca de 350 dias, o modelo de Verhulst em 850 dias e o modelo de Gatenby em 1200 dias.

Observando a taxa de crescimento tumoral na curva de Verhulst, nota-se como a resistência das células saudáveis influencia a taxa de crescimento da curva de Gompertz. Essa taxa é menor do que a do modelo de Verhulst e atinge um limiar diferente em relação à capacidade de suporte do órgão, determinada pela resistência das células saudáveis.

Conforme a literatura, o modelo de Gatenby incorpora parâmetros relevantes para descrever o crescimento tumoral. No entanto, é notável que o modelo de Verhulst oferece uma representação mais precisa em comparação com o modelo de Gatenby, que apresenta um comportamento distinto em relação aos outros dois modelos. Além disso, ao analisar o comportamento das curvas de solução, pode-se concluir que, nos primeiros 300 dias, o modelo de Verhulst é aproximadamente equivalente ao modelo de Gatenby para o crescimento de células tumorais. Isso sugere que nos estágios iniciais da doença, antes do início do tratamento, os modelos de Verhulst ou Gatenby podem ser adequados para descrever o crescimento tumoral. A escolha entre esses modelos dependerá da agressividade do tumor em um órgão específico.

Tabela 24 – Resultado da simulação (RK4 com $h = 0,1$)

Dias	Técnica de Solução			
	Verhulst	Gompertz	Gatenby N_1	Gatenby N_2
0	$1,00000000 \times 10^0$	$1,00000000 \times 10^0$	$1,00000000 \times 10^0$	$1,00000000 \times 10^0$
31	$1,36292979 \times 10^0$	$6,30462314 \times 10^0$	$1,32543666 \times 10^0$	$9,99996824 \times 10^0$
365	$3,70849210 \times 10^0$	$8,35651527 \times 10^0$	$2,69127505 \times 10^0$	$9,99762975 \times 10^0$
1095	$9,82761907 \times 10^0$	$9,99878718 \times 10^0$	$8,74287891 \times 10^0$	$9,76978598 \times 10^0$
1461	$9,99548844 \times 10^0$	$9,99996879 \times 10^0$	$9,11820140 \times 10^0$	$9,59622337 \times 10^0$
1500	$9,99694497 \times 10^0$	$9,99997887 \times 10^0$	$9,12385063 \times 10^0$	$9,58091281 \times 10^0$

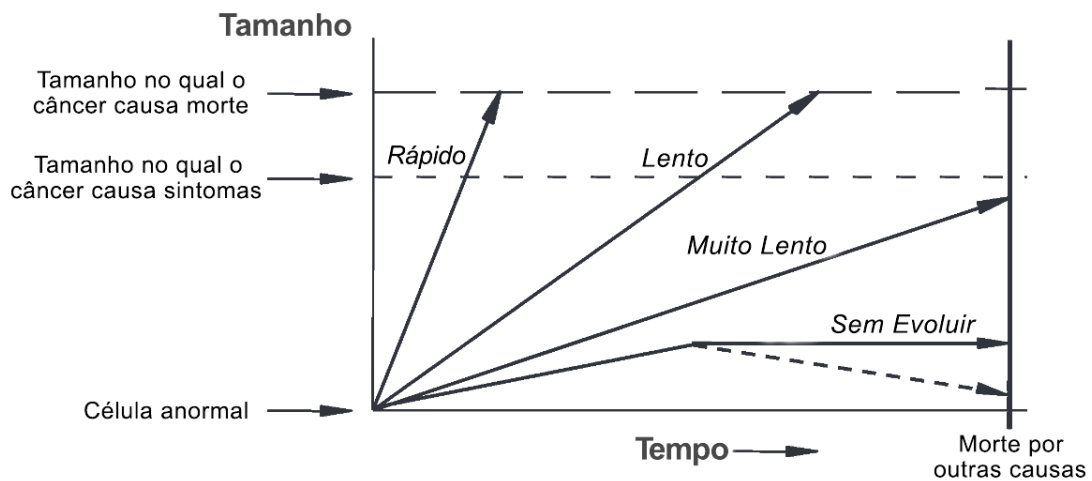
Fonte: O autor.

Numericamente, a Tabela 24 mostra as simulações numéricas para 0, 31, 365, 1095, 1461 e 1500 dias para os modelos de Verhulst, Gompertz e Gatenby com as células N_1 e N_2 . O resultado registrado na tabela reforça o resultado gráfico, o modelo de Gompertz tem crescimento veloz de 0 a 365 dias, e atinge estabilidade em 1095 dias em diante.

Os modelos para células tumorais de Verhulst e Gatenby apresentam valores muito próximos até os primeiros 31 dias. Após esse período, os valores simulados mantêm a mesma ordem de grandeza, mas o modelo de Gatenby gera valores menores em comparação ao modelo de Verhulst. Essa diferença é atribuída à influência da resistência das células saudáveis, que atua de forma mais pronunciada no modelo de Gatenby. Além disso, as células normais experimentam uma redução mais acentuada entre os dias 1095 e 1461 de crescimento tumoral.

A Figura 40 ilustra a classificação da evolução do câncer quanto à sua velocidade de crescimento. A partir da simulação apresentada na Figura 39, é possível observar que a solução de Gompertz representa um crescimento rápido do câncer, caracterizado por um desenvolvimento acelerado que leva rapidamente ao surgimento de sintomas e, conseqüentemente, à morte (WELCH; BLACK, 2010). Por outro lado, as soluções de Verhulst e Gatenby indicam um crescimento mais lento, sugerindo uma doença que manifesta sintomas e resulta em morte somente após muitos anos. Portanto, a escolha criteriosa de um modelo para a simulação de longo prazo é crucial para compreender e prever adequadamente a dinâmica do câncer, permitindo abordagens mais eficazes no tratamento e na gestão dessa condição complexa.

Figura 40 – Relação entre o tempo e a evolução do câncer



Fonte: Adaptado de Welch e Black (2010).

7 Conclusão

Em resumo, este estudo abordou o crescimento tumoral avascular, desconsiderando fatores de tratamento, por meio da aplicação de três modelos populacionais: Verhulst, Gompertz e Gatenby. Utilizou-se uma abordagem numérica com métodos como o Euler, Euler Modificado e Runge-Kutta de 4ª ordem para resolver o problema de valor inicial. Os resultados revelaram que o método de Runge-Kutta se destacou como o mais eficaz em comparação com os métodos de Euler e Euler Modificado, quando confrontados com a solução exata do problema.

Além disso, ao analisar os modelos, observou-se que o modelo de Gompertz demonstrou uma taxa de crescimento do câncer mais acentuada em seus estágios iniciais, enquanto os modelos de Verhulst e Gatenby apresentaram crescimentos iniciais semelhantes. Entretanto, em estágios mais avançados, o modelo de Verhulst revelou um crescimento mais rápido do câncer em comparação ao modelo de Gatenby.

A implementação dos métodos numéricos foi realizada utilizando a linguagem de programação Python, com o suporte da biblioteca NumPy. Essa biblioteca se destaca como uma ferramenta indispensável na implementação de métodos numéricos. Sua capacidade de lidar eficientemente com arrays multidimensionais, ampla gama de funções matemáticas, eficiência computacional e compatibilidade com outras bibliotecas o tornam uma escolha preferencial para cientistas, pesquisadores e engenheiros. A documentação extensa, a ativa comunidade de desenvolvedores e a versatilidade do NumPy o tornam um recurso valioso para a resolução de problemas complexos envolvendo cálculos numéricos. Portanto, o NumPy desempenha um papel fundamental na simplificação da implementação de métodos numéricos e na capacitação da comunidade científica a abordar desafios complexos em diversas disciplinas.

Esses resultados oferecem uma compreensão valiosa sobre o comportamento do crescimento tumoral avascular e a eficácia dos métodos numéricos na resolução desses modelos. A confirmação de que o método de Runge-Kutta é o mais preciso tem implicações significativas para estudos futuros e aplicações clínicas. No entanto, é importante ressaltar que o estudo abordou apenas o crescimento tumoral avascular, e a inclusão de fatores de tratamento pode fornecer *insights* adicionais no futuro.

Este trabalho contribui para o campo da modelagem matemática em dinâmica populacional, especialmente na área de oncologia, e demonstra a importância de abordagens numéricas na resolução de problemas complexos como o crescimento tumoral.

Como perspectiva para trabalhos futuros, sugere-se conduzir uma análise comparativa entre os solucionadores implementados na biblioteca SciPy em Python e as soluções

analíticas dos modelos de dinâmica populacional abordados neste estudo. Além disso, seria relevante explorar o aspecto do tempo computacional, avaliando a eficiência das bibliotecas SciPy em comparação com os métodos implementados em NumPy. Essas investigações contribuiriam para uma compreensão mais abrangente das vantagens e desvantagens das diferentes abordagens de implementação e solução de modelos de crescimento tumoral, aprimorando assim o estado da arte nessa área de pesquisa interdisciplinar.

Referências

- ANDUTTA, F. *Equações Diferenciais: Métodos Analíticos e Numéricos*. Dissertação (Mestrado), 12 2003. Citado na página [27](#).
- ARAÚJO, R. P.; ELWAIN, D. L. S. M. C. A history of the study of solid tumour growth: The contribution of mathematical modelling. *Bulletin of Mathematical Biology*, n. 66, p. 1039–1091, 2004. Citado na página [14](#).
- BACAËR, N. *A Short History of Mathematical Population Dynamics*. France: Institut de Recherche pour le Développement, 2010. Citado 2 vezes nas páginas [33](#) e [34](#).
- BORTULI, A.; FREIRE, I. L.; MAIDANA, N. A. Soluções exatas de um modelo matemático de invasão tumoral. *Proceeding Series of the Brazilian Society of Computational and Applied Mathematics*, v. 8, n. 1, 2021. Citado na página [16](#).
- BRASIL. *ABC do câncer : abordagens básicas para o controle do câncer*. Rio de Janeiro: Instituto Nacional de Câncer, 2011. Citado na página [18](#).
- BURDEN, R. L.; FAIRES, J. D. *Análise Numérica*. [S.l.]: Cengage Learning, 2006. Citado na página [23](#).
- CABELLA, B. C. T. *Modelos Aplicados ao Crescimento e Tratamento de Tumores e à Disseminação da Dengue e Tuberculose*. Dissertação (Mestrado) — Universidade de São Paulo, Ribeirão Preto, 2012. Disponível em: <https://www.teses.usp.br/teses/disponiveis/59/59135/tde-01082012-110701/publico/BrennoCabella.pdf>. Acesso em: 22.05.2022. Citado 3 vezes nas páginas [14](#), [33](#) e [34](#).
- CARNEIRO, D. *Estudo do Câncer via Equações diferenciais ordinárias*. 2020. Researchgate. Disponível em: <https://www.researchgate.net>. Acesso em: 01.08.2022. Citado na página [28](#).
- CEDERVALL, J.; ZHANG, Y.; OLSSON, A.-K. Tumor-Induced NETosis as a Risk Factor for Metastasis and Organ Failure. *Cancer Research*, v. 76, n. 15, p. 4311–4315, 07 2016. ISSN 0008-5472. Disponível em: <https://doi.org/10.1158/0008-5472.CAN-15-3051>. Citado na página [20](#).
- CHAPRA, S. C.; CANALE, R. P. *Métodos Numéricos para Engenharia*. São Paulo: MGH, 2011. Citado 3 vezes nas páginas [17](#), [23](#) e [28](#).
- COELHO, J. C. *Modelo Matemático de Crescimento de Tumor Avascular Invasivo*. Dissertação (Mestrado) — Universidade Estadual de Londrina, Londrina, 2019. Citado 4 vezes nas páginas [17](#), [33](#), [34](#) e [35](#).
- FREITAS, E. K. *Equações Diferenciais Ordinárias Lineares de Primeira Ordem: aplicações na Economia*. Dissertação (Mestrado) — Universidade Federal do Rio Grande, Rio Grande, 2019. Disponível em: https://imef.furg.br/images/stories/Monografias/Matematica_aplicada/2019/2019-2ElisandraFreitas.pdf. Acesso em: 02.08.2022. Citado na página [37](#).

GATENBY, R. A.; GAWLINSKI, E. T. A reaction-diffusion model of cancer invasion. *Cancer Res*, v. 15, n. 56, p. 5745–5753, 1996. Citado 3 vezes nas páginas 15, 16 e 17.

GOMPERTZ, B. On the nature of the function expressive of the law of human mortality, and on a new mode of determining the value of life contingencies. *Philosophical Transactions of the Royal Society of London*, v. 115, p. 513–583, 1825. Citado 3 vezes nas páginas 15, 16 e 17.

HOFF, A. K. P. M. G. *Tratado de oncologia*. São Paulo: Atheneu, 2013. Citado na página 14.

INCA. *Deteção precoce do câncer*. Rio de Janeiro: [s.n.], 2021. Ministério da Saúde. Instituto Nacional de Câncer José de Alencar Gomes da Silva. Disponível em: <<https://www.inca.gov.br/sites/ufu.sti.inca.local/files/media/document/deteccao-precoce-do-cancer.pdf>>. Acesso em: 28.10.2023. Citado na página 20.

INCA. *Tipos de cânceres*. Rio de Janeiro: [s.n.], 2022. Ministério da Saúde. Instituto Nacional de Câncer José de Alencar Gomes da Silva. Disponível em: <<https://www.inca.gov.br/o-que-e-cancer>>. Acesso em: 11.07.2022. Citado na página 14.

KIUSALAAS, J. *Numerical Methods in Engineering with Python*. The Edinburgh Building, Cambridge, CB2 2RU, UK: Cambridge University Press, 2005. Citado 2 vezes nas páginas 17 e 37.

LAIRD, A. K. Dynamics of tumour growth. *British Journal of Cancer*, v. 18, n. 3, p. 490, 1964. Nature Publishing Group. Citado 2 vezes nas páginas 15 e 34.

LIMA, E. L. *Curso de Análise*. Rio de Janeiro: SBM/IMPA, 2000. v. 1. ISBN 85-244-0047-1. Citado na página 23.

LIMA, E. L. *Análise Real*. 6ª. ed. Rio de Janeiro: IMPA, 2016. v. 2. ISBN 978-85-244-0221-0. Citado na página 23.

MAGANIN, J. et al. Simulação de um modelo matemático de crescimento tumoral utilizando diferenças finitas. *Brazilian Journal of Development*, n. 87696, 2020. Citado na página 16.

MALTHUS, T. R. *An Essay on the Principle of Population*. London: J. Johnson, 1798. Citado na página 15.

MATPLOTLIB. *Matplotlib: Visualização com Python*. 2022. Página inicial do Matplotlib. Disponível em: <<https://matplotlib.org/>>. Acesso em: 30.07.2022. Citado 2 vezes nas páginas 37 e 38.

MAYWORM, S. H. *Introdução à Biologia Celular*. Rio de Janeiro: Editora Universidade Estácio de Sá, 2014. ISSN 978-85-60923-15-1. Citado 2 vezes nas páginas 14 e 20.

NASCIMENTO, A. do. *Introdução à Dinâmica de Crescimentos Populacionais*. Dissertação (Mestrado) — Universidade Estadual de Maringá, Maringá, 2012. Citado na página 34.

NASCIMENTO, A. do. *Introdução à Dinâmica de Crescimentos Populacionais*. Dissertação (Mestrado) — Universidade Estadual de Maringá, Maringá, 2012. Disponível em: <http://site.dfi.uem.br/wp-content/uploads/2016/12/ADRIANE-DO-NASCIMENTO.pdf>. Acesso em: 22.06.2022. Citado na página 35.

NUMPY. *O que é o Numpy?* 2022. O que é o Numpy? Disponível em: <https://numpy.org/doc/stable/user/whatisnumpy.html>. Acesso em: 30.07.2022. Citado 2 vezes nas páginas 37 e 38.

PYTHON. *Página de Documentação do Python*. 2023. Disponível em: <https://docs.python.org/3/download.html>. Acesso em: 28.10.2023. Citado na página 37.

RODRIGUES, D.; PINHO, S.; MANCERA, P. Um modelo matemático em quimioterapia. *Notas em Matemática Aplicada*, v. 58, n. 13, p. 1–74, 2011. Disponível em: https://proceedings.science/series/23/proceedings_non_indexed/63. Citado 4 vezes nas páginas 16, 35, 60 e 74.

ROMANO, F. *Curso de Análise*. Birmingham: Packt Publishing Ltd., 2015. v. 1. ISBN 978-1-78355-171-2. Citado 3 vezes nas páginas 37, 38 e 40.

SAUER, T. *Calculus*. New York: Pearson Education, Inc., 2012. ISBN 978-0-321-78367-7. Citado na página 28.

SAUTE, G. S. *Modelagem Matemática da Dinâmica e Controle do Crescimento de Tumores*. Dissertação (Mestrado) — Universidade Regional do Noroeste do Estado do Rio Grande do Sul, Ijuí, 2006. Citado na página 14.

STEWART, J. *Cálculo*. São Paulo: Thomson Learning, 2006. v. 1. Citado na página 23.

TJØRVE, K. M. C. *The use of Gompertz models in growth analyses, and new Gompertz-model approach: An addition to the Unified-Richards family*. 2017. PLOS ONE - RESEARCH ARTICLE. Disponível em: <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0178691#sec002>. Acesso em: 11.06.2022. Citado na página 34.

VERHULST, P.-F. A note on the law of population growth. *Correspondence Mathématique et Physique Publiée par A. Quetelet*, n. 10, p. 113–117, 1838. Citado 3 vezes nas páginas 15, 16 e 17.

WEINBERG, R. A. *The biology of cancer*. New York: Garland Science, Taylor & Francis Group, LLC, 2014. ISBN 978-0-8153-4220-5. Citado na página 20.

WELCH, H. G.; BLACK, W. C. Overdiagnosis in cancer. *Journal of the National Cancer Institute*, v. 102, n. 9, p. 605–13, 2010. Citado na página 76.