INE5202 - Cálculo Numérico em Computadores

Relatório EP3

Alunos: Jéssica R. dos Santos e Marcos Machado

Professor: Priscila Cardoso Calegari

Florianópolis, 2024

Implementação do Método Runge Kutta para solução de um SIR

Objetivo: Implementar e analisar resultados do método de Runge Kutta para ordem 4 para solucionar um PVI que descreve um modelo SIR.

Introdução

O modelo matemático SIR (Susceptible, Infected, Removed) é uma forma muito popular e simples de representar o comportamento de uma doença em um instante de tempo através de funções baseadas numa variável 't' (quantidade de tempo decorrido).

S(t) descreve a quantidade de pessoas suscetíveis à doença, mas ainda não infectadas, em um instante de tempo t.

I(t) descreve a quantidade de pessoas infectadas com a doença em um instante de tempo t.

R(t) descreve a quantidade de pessoas removidas (recuperadas da doença ou falecidas) em um instante de tempo t (o modelo presume que, uma vez que um indivíduo for recuperado, se torna imune, e não pode mais retornar ao grupo de suscetíveis ou infectados).

Essas funções são descritas por meio de um sistema de Equações Diferenciais Ordinárias (EDOs), da forma:

$$dS/dt = -\beta SI$$
$$dI/dt = \beta SI - \gamma I$$
$$dR/dt = \gamma I$$

Em que β é a taxa de infecção (taxa com que indivíduos suscetíveis adquirem a condição) e γ a taxa de recuperação (taxa com que indivíduos são recuperados da doença e se tornam imunes).

Nossa tarefa era, dado os valores iniciais de 49 indivíduos suscetíveis, 1 infectado e nenhum removido, e parâmetros para ambas as taxas, realizar uma simulação para a proliferação da doença, usando o método de Runge Kutta de ordem 4 para resolver Problemas de Valor Inicial (PVI), ensinado ao longo do curso. Após isso, deveríamos também resolver o problema assumindo a possibilidade de um distanciamento social, que reduziria a taxa de virulência.

$$\begin{cases} y_0 = y(t_0), \\ y_{k+1} = y_k + \Delta t \left(\dfrac{k_1 + 2k_2 + 2k_3 + k_4}{6} \right) \end{cases}$$

com $k_1 = f(t_k, y_k)$, $k_2 = f\left(t_k + \dfrac{\Delta t}{2}, y_k + \dfrac{\Delta t}{2} k_1\right)$ e $k_3 = f\left(t_k + \dfrac{\Delta t}{2}, y_k + \dfrac{\Delta t}{2} k_2\right)$ e $k_4 = f\left(t_k + \Delta t, y_k + \Delta t k_3\right)$.

Algoritmo fornecido pela professora do método de Runge Kutta de ordem 4 (aqui, considerando apenas uma função y(t) para o sistema: fizemos as modificações adequadas para a solução com 3 funções).

Implementação do Runge Kutta

Nossa solução contém 2 programas, uma para cada simulação: com e sem distanciamento social. A diferença única entre elas é a existência de uma nova variável na simulação 2, `beta_fifth`, representando uma redução da taxa de virulência em 5 vezes, por conta do distanciamento.

```python
# Definindo novo valor de beta reduzido em cinco vezes
beta_fifth = beta / 5
```

Primeiramente, cada programa conta com 3 funções auxiliares que implementam o sistema de EDOs do SIR (demonstrados na introdução acima).

```python
# Definindo as EDOs do modelo SIR
def dS_dt(S, I, R, t):
    return -beta * S * I

def dI_dt(S, I, R, t):
    return beta * S * I - gamma * I

def dR_dt(S, I, R, t):
    return gamma * I
```

Estas são usadas na função principal do código, `runge_kutta()`, que usa o método Runge_Kutta de quarta ordem para achar os valores de S, I e R. Ele calcula os valores k1, k2, k3 e k4 para cada variável, e calcula o próximo termo baseado nestes valores (ver algoritmo na introdução do relatório). Vale ressaltar que, por conta de como funciona o sistema de montagem de gráficos da biblioteca matplotlib do Python, implementamos S, I e R como um vetor com todos os termos gerados do método, a fim de obtermos gráficos das nossas soluções).

```python
def runge_kutta(S0, I0, R0, t0, tn, h):
    t = np.arange(t0, tn+h, h)
    S = np.zeros(len(t))
    I = np.zeros(len(t))
    R = np.zeros(len(t))

    S[0], I[0], R[0] = S0, I0, R0

    for i in range(1, len(t)):
        k1_S = dS_dt(S[i-1], I[i-1], R[i-1], t[i-1])
        k1_I = dI_dt(S[i-1], I[i-1], R[i-1], t[i-1])
        k1_R = dR_dt(S[i-1], I[i-1], R[i-1], t[i-1])

        k2_S = dS_dt(S[i-1] + k1_S * h / 2, I[i-1] + k1_I * h / 2, R[i-1] + k1_R * h / 2, t[i-1] + h / 2)
        k2_I = dI_dt(S[i-1] + k1_S * h / 2, I[i-1] + k1_I * h / 2, R[i-1] + k1_R * h / 2, t[i-1] + h / 2)
        k2_R = dR_dt(S[i-1] + k1_S * h / 2, I[i-1] + k1_I * h / 2, R[i-1] + k1_R * h / 2, t[i-1] + h / 2)

        k3_S = dS_dt(S[i-1] + k2_S * h / 2, I[i-1] + k2_I * h / 2, R[i-1] + k2_R * h / 2, t[i-1] + h / 2)
        k3_I = dI_dt(S[i-1] + k2_S * h / 2, I[i-1] + k2_I * h / 2, R[i-1] + k2_R * h / 2, t[i-1] + h / 2)
        k3_R = dR_dt(S[i-1] + k2_S * h / 2, I[i-1] + k2_I * h / 2, R[i-1] + k2_R * h / 2, t[i-1] + h / 2)

        k4_S = dS_dt(S[i-1] + k3_S * h, I[i-1] + k3_I * h, R[i-1] + k3_R * h, t[i-1] + h)
        k4_I = dI_dt(S[i-1] + k3_S * h, I[i-1] + k3_I * h, R[i-1] + k3_R * h, t[i-1] + h)
        k4_R = dR_dt(S[i-1] + k3_S * h, I[i-1] + k3_I * h, R[i-1] + k3_R * h, t[i-1] + h)

        S[i] = S[i-1] + (h / 6) * (k1_S + 2*k2_S + 2*k3_S + k4_S)
        I[i] = I[i-1] + (h / 6) * (k1_I + 2*k2_I + 2*k3_I + k4_I)
        R[i] = R[i-1] + (h / 6) * (k1_R + 2*k2_R + 2*k3_R + k4_R)

    return t, S, I, R
```

Experimentação e Análise dos Resultados

Como demonstrado nos gráficos abaixo, a primeira e segunda simulação retornam resultados muito diferentes.

Na primeira simulação, por falta de medidas protetivas como o distanciamento social, o número de infectados, $I(t)$, subiu rapidamente, atingindo um ápice preocupante. Entretanto, isso também ocasionou num fim rápido para a doença, visto, que, o aumento no número de infectados levou a um aumento no número de removidos, fazendo com que I e S caíssem rapidamente. Na simulação 1, a doença já está quase extinta.

Na simulação 2, vemos um estrago muito menor, com a quantidade de infectados se mantendo baixa durante todo o período. Entretanto, por conta disso, a doença ainda está se transmitindo: menos pessoas infectadas levam a menos pessoas se tornando imunes.

Interessantemente, é possível realizar uma analogia com o mundo real através desses resultados: o distanciamento social desacelera a extinção da doença, mas também diminui o seu impacto e impede um crescimento súbito e acentuado no número de infectados.

(É importante lembrar que essa simulação é muito simples, desconsiderando fatores como: vacinas, nascimentos, falecimentos por outras causas, pessoas naturalmente imunes, entre muitos outros fatores de um contexto real. Por exemplo, se considerássemos tratamento como vacinas - ou seja, formas de suscetíveis se tornarem removidos sem passar pelo estágio de infectados - através de uma taxa de vacinação φ, representando a taxa com que pessoas são vacinadas, poderíamos alterar as equações do sistema para $dS/dt = -\beta SI - \varphi S$, $dI/dt = \beta SI - \gamma I$

e dR/dt = γI + φS. Esse é um modelo muito interessante que pode ser alterado de várias formas para produzir resultados diferentes.)

Gráficos

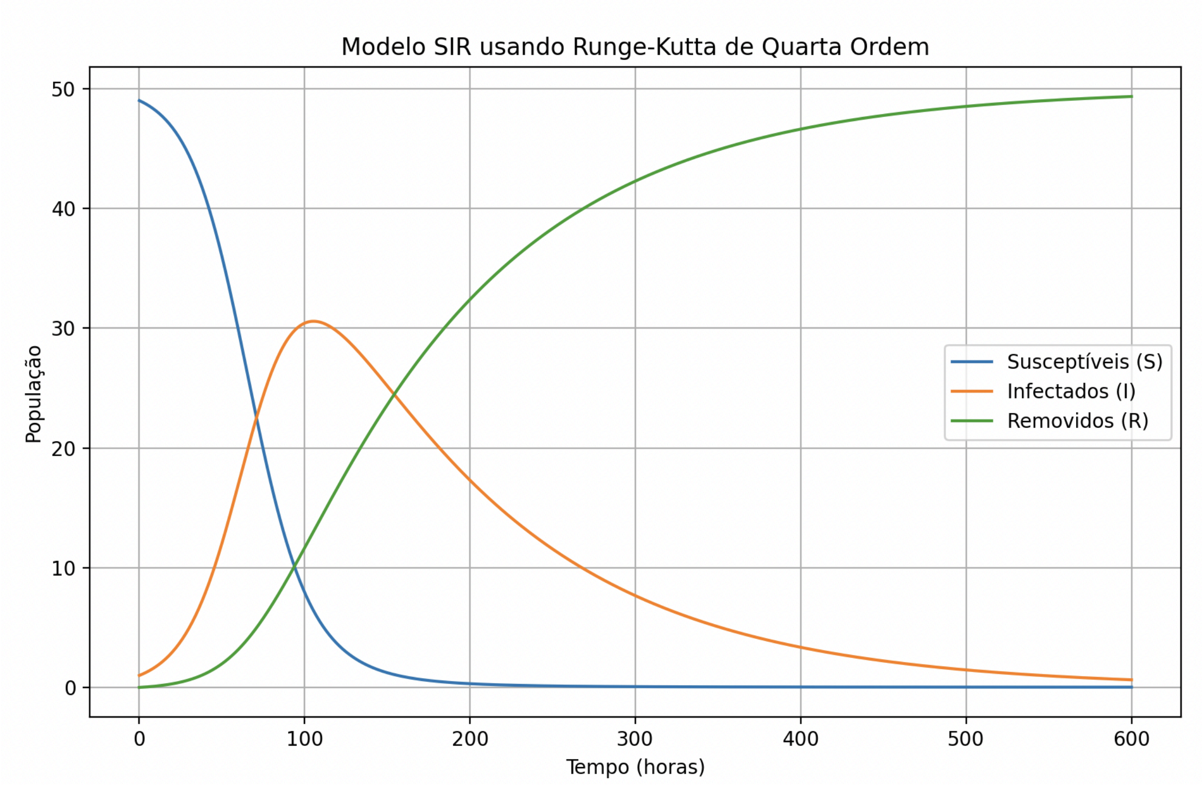

Gráfico da Simulação 1

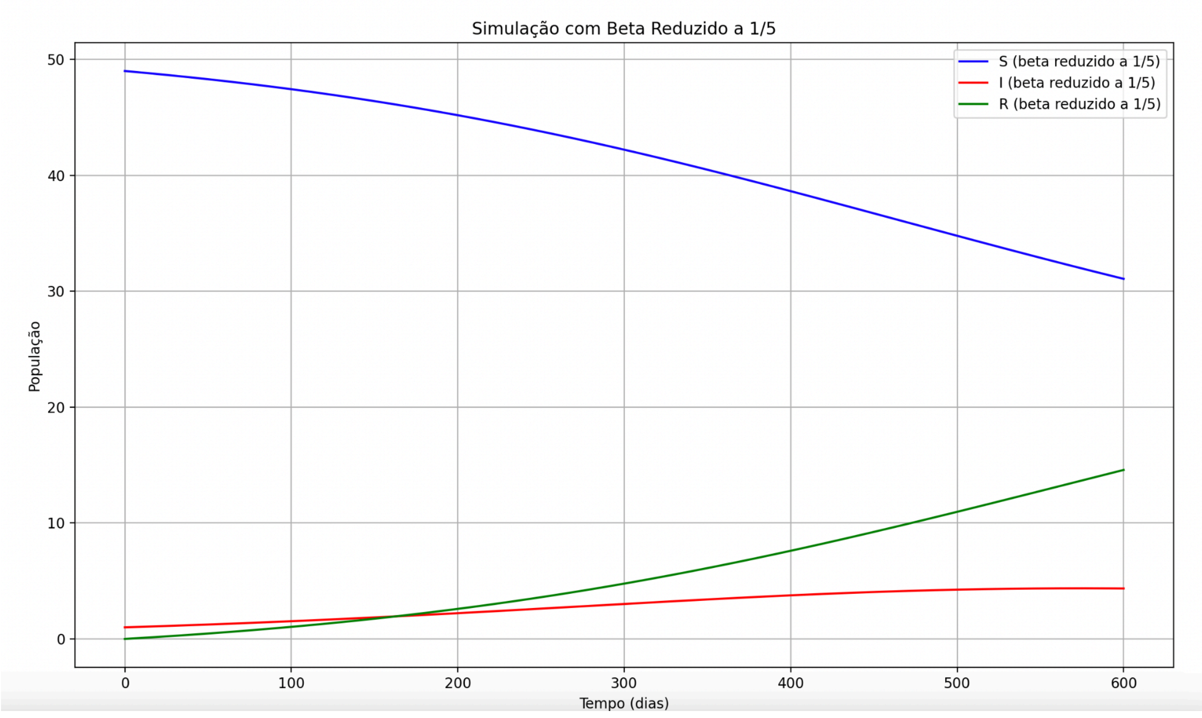

Gráfico da Simulação 2

Tabelas

- 5 dias = 120 horas;
- 10 dias = 240 horas;
- 15 dias = 360 horas;
- 20 dias = 480 horas;
- 25 dias = 600 horas;

	Valores Padrão	**Resultado com distanciamento social**
S(120)	3.60	47.04
I(120)	29.69	1.65
R(120)	16.71	1.31
S(240)	0.14	44.08
I(240)	12.53	2.53
R(240)	37.33	3.38
S(360)	0.04	40.12
I(360)	4.66	3.48
R(360)	45.29	6.40
S(480)	0.03	35.55
I(480)	1.72	4.18
R(480)	48.25	10.26
S(600)	0.02	31.07
I(600)	0.64	4.35
R(600)	49.34	14.58

Problemas/Dificuldades Encontradas

Durante o desenvolvimento do projeto, encontramos algumas dificuldades:

- Implementação do método: Alterar o método de Runge-Kutta de quarta ordem de uma única função y(t) para um sistema de três funções (S, I, R) exigiu entendimento cuidadoso das interações entre as equações. Foi necessário garantir que os cálculos de k1, k2, k3 e k4 para cada função fossem aplicados de maneira complementar.
- Implementação com distanciamento social: A introdução de uma nova variável para representar a redução da taxa de infecção devido ao distanciamento social adicionou um pouco mais de complexidade, principalmente, garantir que essa nova variável fosse corretamente integrada ao sistema de EDOs e ajustada conforme necessário.
- Visualização dos resultados: Gerar gráficos claros e informativos com a biblioteca matplotlib exigiu várias iterações. Tivemos que ajustar os eixos, rótulos, legendas e cores para garantir que os gráficos fossem fáceis de interpretar e comparáveis entre as duas simulações.

Agradecimento

Foi ótimo ser teu aluno esse semestre :) Foi muito boa a experiência com a realização dos EPs ->,esse foi o que teve o contexto mais interessante e atrativo, aliás! Boas férias, Professora Priscila!

- Marcos

Foi muito bom ser sua aluna nesse semestre, professora Priscila, você é uma querida! Concordo com o Marcos, esse EP foi especialmente interessante devido ao seu contexto prático/social. Também agradeço ao Marcos por ser meu parceiro ao longo do semestre, será um profissional incrível. Boas férias a todos!

- Jéssica