PMR3401 - Mecânica Computacional
Exercício Programa 1

Luan Gustavo de Brito - NUSP 10410121
Vinícius Araujo da Costa - NUSP 7254069

12 de Junho de 2022

Conteúdo

1 Enunciado e equacionamento **3**

2 Solução **5**
2.1 Método de Runge-Kutta de 4ª Ordem 5
2.2 Item 1. 5
2.3 Item 2. 6

3 Conclusão **7**

4 Código **8**
4.1 main 8

Parte 1 - Método de Runge-Kutta

1 Enunciado e equacionamento

A figura 1 representa um modelo de circuito elétrico que aciona o forno sendo representado pelas seguintes equações:

$$\begin{cases} L_a \frac{di_1}{dt} + R_a(i_1 - i_2) + \frac{q(t)}{C} = e(t) \\ -\frac{q(t)}{C} - R_a(i_1 - i_2) + R_b i_2 + L_b \frac{di_2}{dt} = 0 \\ q(t) = \int_0^t (i_1(t') - i_2(t'))\, dt' + \mathrm{q(0)} \end{cases} \tag{1}$$

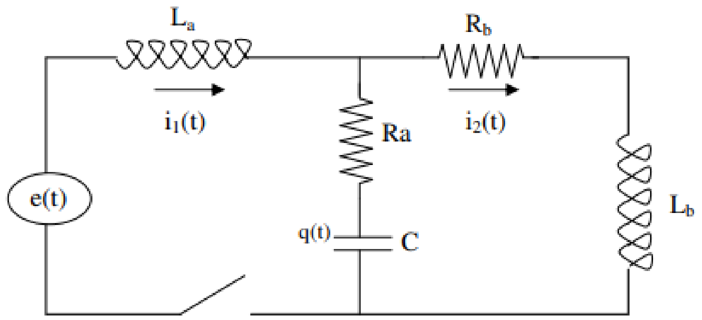

Figura 1: Circuito elétrico

Sendo $i_1(t)$,$i_2(t)$ e $q(t)$ nossas incógnitas pode-se representá-las da seguinte forma:

$$Y(t) = \begin{bmatrix} i_1(t) \\ \\ i_2(t) \\ \\ q(t) \end{bmatrix} \tag{2}$$

Assim, temos a partir de 1 e 2:

$$\dot{Y}(t) = \begin{bmatrix} \dot{i_1}(t) \\ \dot{i_2}(t) \\ \dot{q}(t) \end{bmatrix} = \begin{bmatrix} -\dfrac{R_a(i_1 - i_2) - \dfrac{q(t)}{C} + e(t)}{L_a} \\ \dfrac{\dfrac{q(t)}{C} + R_a(i_1 - i_2) - R_b i_2}{L_b} \\ i_1 - i_2 \end{bmatrix} \tag{3}$$

Pede-se:

1. Resolva as equações para $0 \leq t \leq 0,03s$ utilizando o Método de Runge-Kutta de 4a ordem que deve ser implementado pelo aluno. Verifique a influência do passo "Δt"sobre a solução – escolha três passos (pequeno médio e grande) (em que aspectos você vai verificar essa influência? Por quê?). Para cada valor de passo escolhido plote todos os itens $i_1(t), \dot{i_1}(t), i_2(t), \dot{i_2}(t), q(t)$ num mesmo gráfico, em função de t. Para isso utilize escalas diferentes na plotagem de $i_1(t), \dot{i_1}(t), i_2(t), \dot{i_2}(t), q(t)$, ou seja, p.ex. $i_1(t).10^p, \dot{i_1}(t).10^q, i_2(t).10^r, \dot{i_2}(t).10^s, q(t).10^z$. Encontre valores apropriados de p, q, r, s e z de forma que todos os gráficos apareçam na plotagem;

2. Resolva as equações novamente com passo apropriado considerando:

   (a) Mesmas constantes da figura, mas La=0,1;

   (b) Mesmas constantes da figura, mas Ra=2000;

   Para cada questão, plote todos os itens $i_1(t), \dot{i_1}(t), i_2(t), \dot{i_2}(t), q(t)$ **num mesmo gráfico**, em função de t, como sugerido acima.

2 Solução

2.1 Método de Runge-Kutta de 4ª Ordem

O método a ser utilizado para resolver esse problema é o método de Runge-Kutta de quarta ordem. A solução das EDOs através desse método segue a seguinte forma:

$$k_1 = f(x_i, Y_i) = \frac{dY}{dx} \qquad k_2 = f(x_i + \frac{1}{2}h, Y_i + \frac{1}{2}hk_1) \tag{4}$$

$$k_3 = f(x_i + \frac{1}{2}h, Y_i + \frac{1}{2}hk_2) \qquad k_4 = f(x_i + h, Y_i + hk_3) \tag{5}$$

$$Y_{i+1} = Y_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \qquad Y = [i_1(t), i_2(t), q(t)] \tag{6}$$

Onde x_i são as variáveis independentes do problema (nesse caso o tempo t), Y_i as variáveis dependentes e h o passo utilizado na integração numérica. Além disso, Y é o vetor de estados do problema no instante i e k_1,k_2, k_3, k_4 são funções incremento, que tem como intuito melhorar a estimativa da derivada e, como consequência, a acurácia do método.

Dadas as constantes do problema:

$$\begin{aligned} L_b &= 0,5 \\ R_b &= 20 \\ C &= 0,002 \\ L_a &= 0,01 \\ R_a &= 200 \end{aligned}$$

E a seguinte condição inicial:

$$i_1(0) = i_2(0) = q(0) = 0$$

Por esse método, chega-se nas seguintes soluções:

2.2 Item 1.

Foram selecionados os passos $0,0000005$, $0,0001$ e 0.01 para representarem os passos pequeno, médio e grande, respectivamente. Foram obtidos os seguintes resultados:

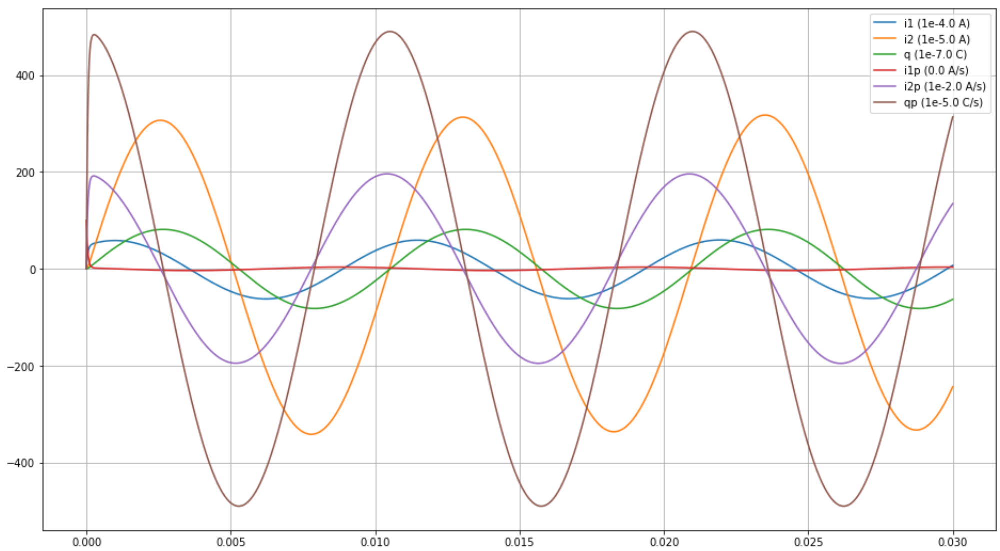

Figura 2: Resultado da resolução com o passo pequeno

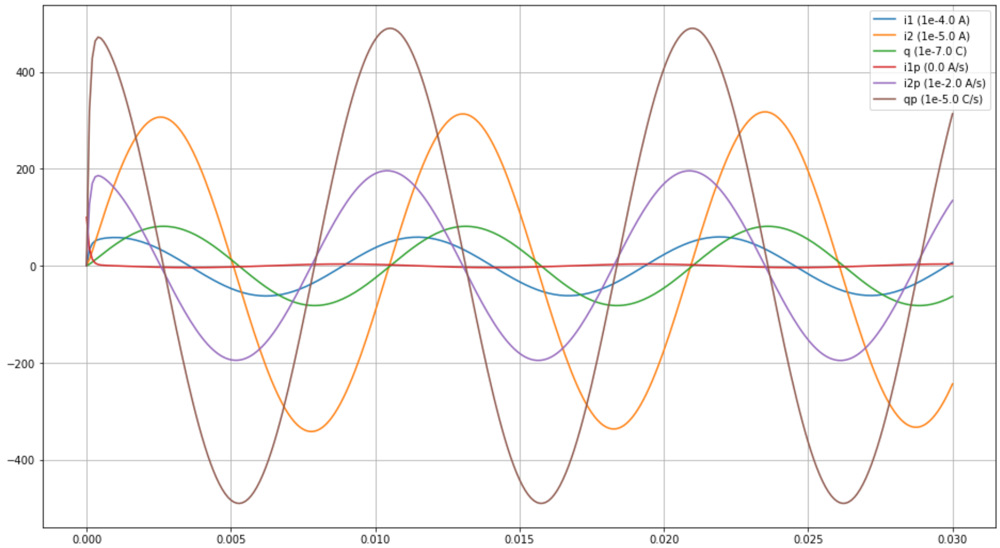

Figura 3: Resultado da resolução com o passo médio

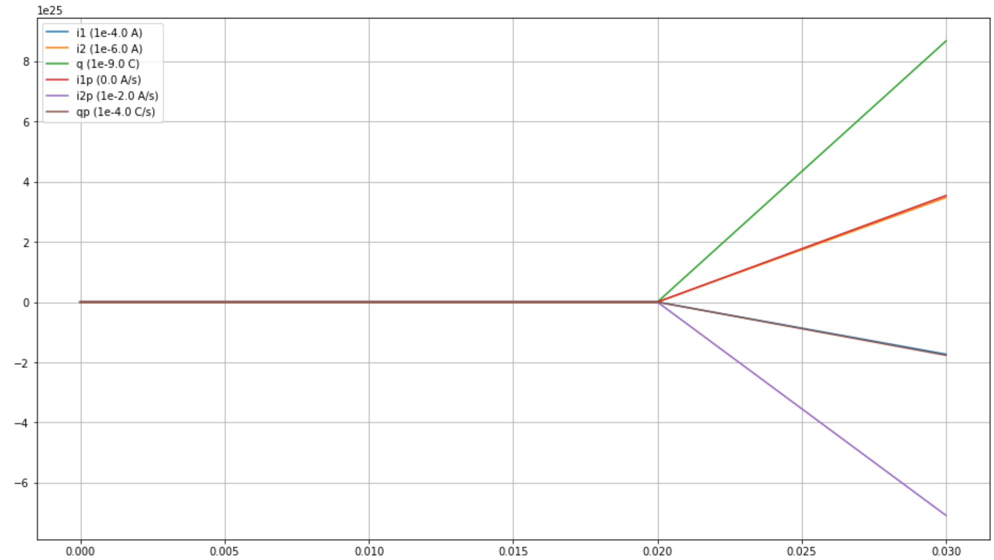

Figura 4: Resultado da resolução com o passo grande

A partir desses resultados obtidos pode-se observar a necessidade de se ter um passo adequado ao se utilizar o método de Runge-Kutta. Passos muito grandes irão produzir resultados imprecisos e com muita perda de informação, enquanto passos muito pequenos acarretarão em um tempo muitp grande de execução, sem a precisão adicional gerada justificar o gasto computacional necessário.

2.3 Item 2.

Para o item 2, será utilizado o passo médio, que se mostrou o mais apropriado no item anterior. Para as mesmas constantes, mas com $L_a = 0.1$, obteve-se:

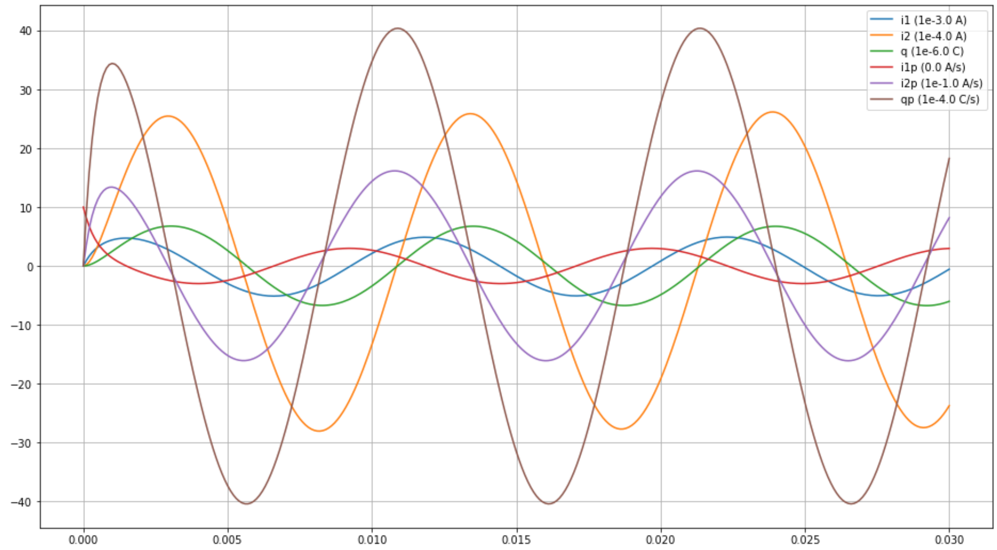

Figura 5: Resultado da resolução com passo médio e $L_a = 0.1$

Já para as mesmas constantes, mas com $R_a = 2000$, obteve-se:

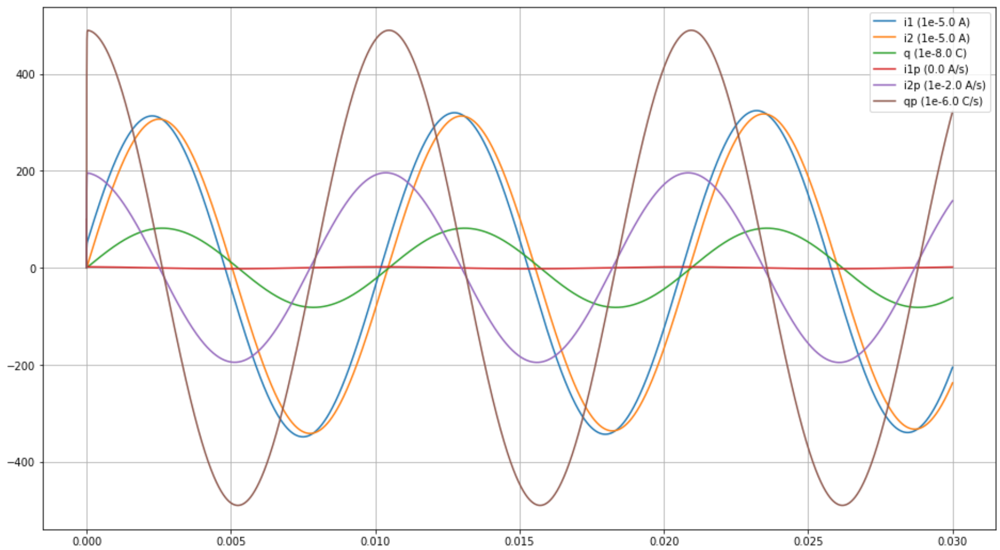

Figura 6: Resultado da resolução com passo médio e $R_a = 2000$

3 Conclusão

Por esse exercício, foi possível observar de forma prática as diferenças causadas pela mudança de passo no método de Runge-Kutta de 4ª ordem. O custo computacional acrescido para passos pequenos e a imprecisão de passos grandes torna essencial se calcular o passo correto a ser utilizado.

4 Código

Em seguida são apresentados os códigos utilizados no programa. Além disso, junto ao relatório foram anexados os arquivos desses programas.

4.1 main

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

def runge_kutta(f,Y0,t0,h,tf=None,n=None):
    """
    f: fun  o a ser aplicada
    Y0: vetor inicial
    t0: tempo inicial
    h: passo de integra  o
    tf: tempo final
    n: n mero de passos
    """
    if tf is None:
        tf = t0 + n*h
    if n is None:
        n = int((tf-t0)/h)
    t = np.linspace(t0,tf,n+1)
    Y = np.zeros((n+1,len(Y0)))
    dY=np.zeros((n+1,len(Y0)))
    Y[0] = Y0
    for i in range(n):
        k1 = f(t[i],Y[i])
        k2 = f(t[i]+h/2,Y[i]+h*k1/2)
        k3 = f(t[i]+h/2,Y[i]+h*k2/2)
        k4 = f(t[i]+h,Y[i]+h*k3)
        Y[i+1] = Y[i] + h*(k1+2*k2+2*k3+k4)/6
        dY[i] = k1
    dY[n]=f(t[n],Y[n])

    return t,Y,dY

def f(t,Y):
    #Constants
    Ra = 200
    Rb = 20
    La = 0.01
    Lb = 0.5
    C = 0.002
    e = lambda t: np.cos(t*600)/La

    #Variables
    i1 = Y[0]
    i2 = Y[1]
    q = Y[2]

    #Calculates the slope vector
    dY = np.zeros(3)
    dY[0] = e(t)+(-Ra*(i1-i2)-q/C)/La
    dY[1] = (+Ra*(i1-i2)-Rb*i2+q/C)/Lb
    dY[2] = i1-i2

    return dY

def plot(t,Y,dY):
    Y = np.concatenate((Y,dY),axis=1)
    OGs = [np.floor(np.log10(abs(x.min()-x.max()))) for x in Y.T]
    scaler = [max(OGs)-x for x in OGs]
    fig = plt.figure(figsize=(16,9))
    ax = fig.add_subplot(1,1,1)
```

```
    ax.plot(t,Y[:,0]*10**scaler[0],label=f'i1 (1e-{scaler[0]} A)')
    ax.plot(t,Y[:,1]*10**scaler[1],label=f'i2 (1e-{scaler[1]} A)')
    ax.plot(t,Y[:,2]*10**scaler[2],label=f'q (1e-{scaler[2]} C)')
    ax.plot(t,dY[:,0]*10**scaler[3],label=f'i1p ({scaler[3]} A/s)')
    ax.plot(t,dY[:,1]*10**scaler[4],label=f'i2p (1e-{scaler[4]} A/s)')
    ax.plot(t,dY[:,2]*10**scaler[5],label=f'qp (1e-{scaler[5]} C/s)')
    ax.legend()
    ax.grid()
    plt.show()

#passo pequeno
t,Y,dY = runge_kutta(f, [0,0,0], 0, (0.000001)/2, tf=0.03)
plot(t,Y,dY)
#passo medio
t,Y,dY = runge_kutta(f, [0,0,0], 0, 0.0001, tf=0.03)
plot(t,Y,dY)
#passo grande
t,Y,dY = runge_kutta(f, [0,0,0], 0, 0.01, tf=0.03)
plot(t,Y,dY)
```

Listing 1: main.py