1 Exercícios (23/11/2017)

Use o método de integração numérica de Runge-Kutta para obter a aproximação da solução dos seguintes problemas de Cauchy

1.

$$R'(t) = U(t), \quad U'(t) = \frac{35\,(P_v - P_l) - 7U(t)^2\,(5P_l + 5\epsilon_l + 4P_v + 4\epsilon_v)}{7R(t)\,(5P_l + 5\epsilon_l + P_v + \epsilon_v)}, \quad R(0) = 1, \quad U(0) = 0.0001$$

2.

$$R'(t) = U(t), \quad U'(t) = \frac{35\,(P_v - P_l) - 7U(t)^2\,(5P_l + 5\epsilon_l + 4P_v + 4\epsilon_v)}{7R(t)\,(5P_l + 5\epsilon_l + P_v + \epsilon_v)}, \quad R(0) = 0.5, \quad U(0) = 0.146446$$

Adote os seguintes valores para os parâmetros: $P_v = 0.166391$, $P_l = 0.195091$, $\epsilon_v = 1.79634$ e $\epsilon_l = 4.34802$. Note que U é a primeira derivada de R com relação a t.

2 Resolução

O método de Runge-Kutta foi implementado em MATLAB e Python

2.1 Exercício 1

No exercício 1, o valor do passo escolhido foi $h = 0.1$ e o intervalo de simulação foi definido como $t = [0, 12.5]$, pois após este limite superior, o resultado iria pra infinito.

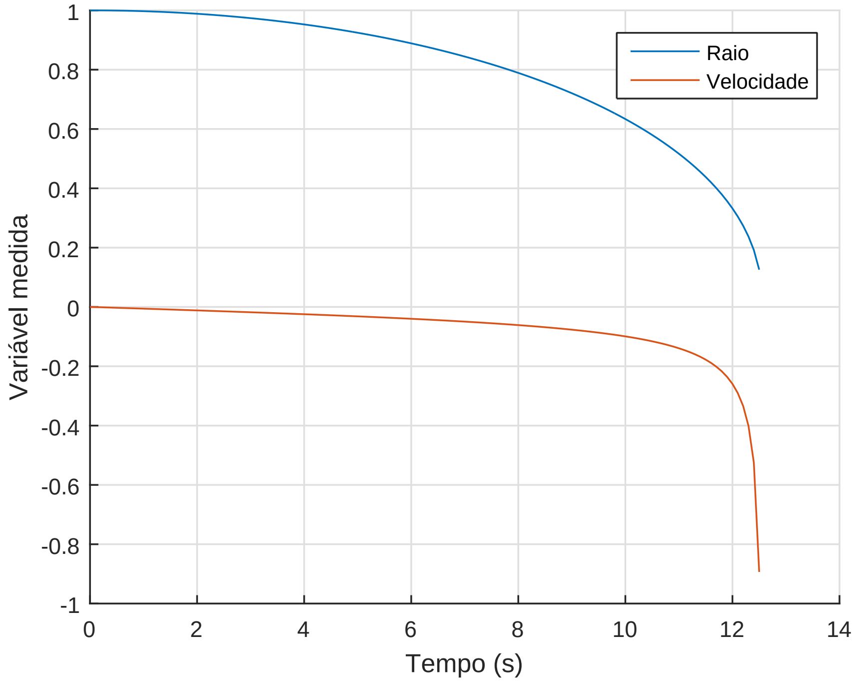

Figura 1: Variáveis medidas no exercício 1.

2.2 Exercício 2

No exercício 2, o valor do passo escolhido foi $h = 0.2$ e o intervalo de simulação foi definido como $t = [0, 22]$, pois após este limite superior, o resultado iria pra infinito.

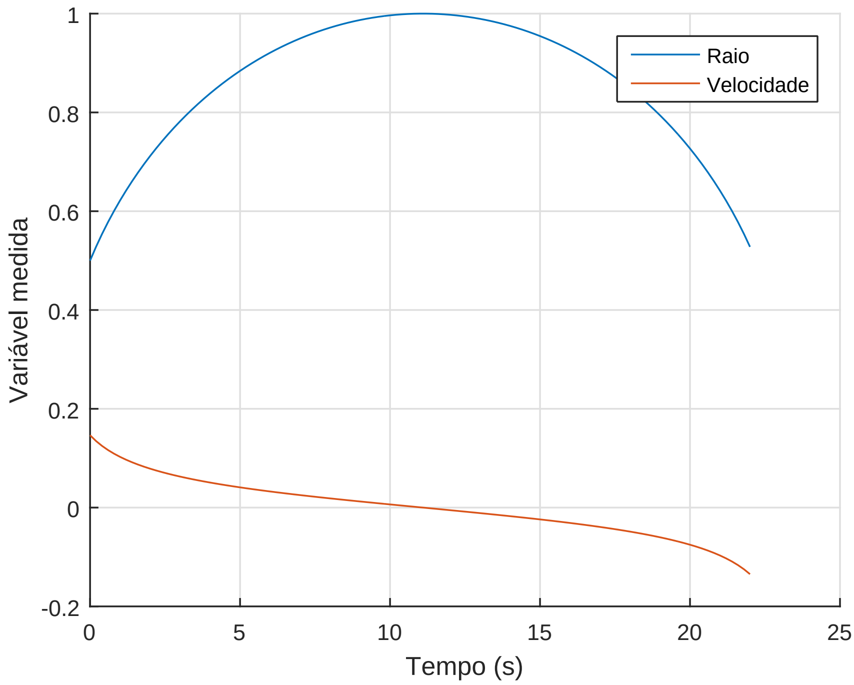

Figura 2: Variáveis medidas no exercício 2.

2.3 Código em MATLAB

```matlab
clear
clc
format long

%% Parâmetros iniciais da equação da bolha
Pl = 0.195091;
Pv = 0.166391;
el = 4.34802;
ev = 1.79634;

f = @(t,X) [X(2); (35*(Pv-Pl) - 7*X(2)^2*(5*Pl + 5*el + 4*Pv + 4*ev))/(7*X(1)*(5*Pl+5*el+Pv+ev))]; % [R;U]

h = 0.3;                         % Tamanho do passo
d = 2;                           % Quantidade de equações no sistema
t = 0:h:12.5;                    % Intervalo da simulação
x = zeros(d, length(t));         % Pré-alocação da matriz de resultados
x(:,1) = [1; 0.0001];            % Chute inicial

%% Encontra soluções do sistema de eq. diferenciais utilizando o método de Runge-Kutta de quarta ordem.
for i = 1:length(t)-1
    k1 = h * f(t(i), x(:, i));
    k2 = h * f(t(i) + h/2, x(:, i) + k1/2);
    k3 = h * f(t(i) + h/2, x(:, i) + k2/2);
    k4 = h * f(t(i) + h, x(:, i) + k3);
    x(:, i+1) = x(:, i) + (k1 + 2*k2 + 2*k3 + k4)/6;
end

%% Plota as variáveis analisadas
hold on
plot(t, x(1,:));
plot(t, x(2,:));
xlabel('Tempo (s)');
ylabel('Variável medida');
grid on;
legend('Raio', 'Velocidade');
hold off
```

2.4 Código em Python

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

# Parâmetros iniciais da equação da bolha
Pl = 0.195091
Pv = 0.166391
el = 4.34802
ev = 1.79634

f = lambda t,X: np.array([X[1], (35*(Pv-Pl) - 7*X[1]**2*(5*Pl + 5*el + 4*Pv + 4*ev))/(7*X[0]*(5*Pl+5*el+Pv+ev))])

h = 0.2                                 # Tamanho do passo
d = 2                                   # Quantidade de equações no sistema
t = np.arange(0, 22.2, 0.2)             # Intervalo da simulação
x = np.zeros((d, len(t)))               # Pré-alocação da matriz de resultados
x[:, 0] = np.array([0.5, 0.146446])     # Chute inicial

# Encontra soluções do sistema de eq. diferenciais utilizando o método de Runge-Kutta de quarta ordem.
for i in range(len(t)-1):
    k1 = h * f(t[i], x[:, i])
    k2 = h * f(t[i] + h/2, x[:, i] + k1.T/2)
    k3 = h * f(t[i] + h/2, x[:, i] + k2.T/2)
    k4 = h * f(t[i] + h, x[:, i] + k3.T)

    x[:, i+1] = x[:, i] + (k1 + 2*k2 + 2*k3 + k4)/6

# Plota as variáveis analisadas
plt.plot(t, x[0, :], 'b', label='Raio')
plt.plot(t, x[1, :], 'r', label='Velocidade')
plt.grid()
plt.legend()
plt.xlabel('Tempo (s)');
plt.ylabel('Variável medida');
plt.show()
```