Relatório de Cálculo Numérico
Implementação e resultado do exercício de avaliação 3

Kaio César de Oliveira Barreto

CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

João Pessoa, 2024

Sumário

1 QUESTÃO 1

1.1 Enunciado

Usando os métodos de Euler e Runge-Kutta de 3ª ordem com $h = 0,2$. Calcule $y(1)$ sabendo que $y(x)$ é solução de

$$2x + yy' = y, y(0) = 1$$

Sabendo que a solução exata do PVI acima é $y(x) = \sqrt{2x+1}$, calcule para os dois métodos o erro absoluto cometido na aproximação de $y(1)$.

1.2 Metodologia

Para resolver o problema proposto, aplicam-se os métodos numéricos de Euler e Runge-Kutta de 3ª ordem a fim de aproximar a solução da equação diferencial ordinária (EDO) e comparar os resultados com a solução exata fornecida.

A equação diferencial dada é:

$$2x + y\frac{dy}{dx} = y^2$$

Rearranjando para a forma padrão, temos:

$$\frac{dy}{dx} = \frac{y^2 - 2x}{y}$$

A condição inicial fornecida é $y(0) = 1$, o que significa que para $x = 0$, $y = 1$. Esse valor será utilizado como ponto de partida para os métodos numéricos.

A solução exata fornecida para o problema é:

$$y(x) = \sqrt{2x+1}$$

Assim, para $x = 1$, a solução exata é:

$$y(1) = \sqrt{2(1)+1} = \sqrt{3}$$

O método de Euler é um método numérico de passo simples que utiliza a seguinte fórmula iterativa:

$$y_{n+1} = y_n + hf(x_n, y_n)$$

Onde $f(x,y)$ é a função derivada $\frac{y^2-2x}{y}$, e h é o tamanho do passo dado como $h = 0,2$. Inicializa-se com $y_0 = 1$ e $x_0 = 0$, iterando até $x = 1$, ou seja, realizando 5 passos de 0,2 cada.

O método de Runge-Kutta de 3ª ordem usa uma média ponderada de inclinações para melhorar a aproximação, conforme a fórmula:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 4k_2 + k_3)$$

Onde:

$$k_1 = f(x_n, y_n)$$

$$k_2 = f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right)$$

$$k_3 = f(x_n + h, y_n - hk_1 + 2hk_2)$$

Da mesma forma, o método é inicializado com $y_0 = 1$ e $x_0 = 0$, avançando até $x = 1$.

Após obter as aproximações de $y(1)$ para ambos os métodos, o erro absoluto é calculado comparando o valor aproximado com a solução exata $y(1) = \sqrt{3}$ pela fórmula:

$$ErroAbsoluto = |y_{aproximado}(1) - y_{exato}(1)|$$

Dessa forma, os resultados numéricos obtidos podem ser avaliados em relação à solução exata.

1.3 Implementação

```
import numpy as np

# Definindo a função f(x, y) = y' = (y^2 - 2x) / y
def f(x, y):
    return (y**2 - 2*x) / y

# Método de Euler
def euler_method(f, x0, y0, h, n):
    x = x0
    y = y0
```

```
    for i in range(n):
        y = y + h * f(x, y)
        x = x + h
    return y

# Método de Runge-Kutta de 3ª ordem
def runge_kutta_3(f, x0, y0, h, n):
    x = x0
    y = y0
    for i in range(n):
        k1 = f(x, y)
        k2 = f(x + h/2, y + h/2 * k1)
        k3 = f(x + h, y - h * k1 + 2 * h * k2)
        y = y + (h/6) * (k1 + 4*k2 + k3)
        x = x + h
    return y

# Solução exata
def exact_solution(x):
    return np.sqrt(2*x + 1)

# Parâmetros iniciais
x0 = 0
y0 = 1
h = 0.2
n = int(1 / h)  # Número de passos para x = 1

# Aproximação pelo método de Euler
y_euler = euler_method(f, x0, y0, h, n)

# Aproximação pelo método de Runge-Kutta de 3ª ordem
y_runge_kutta = runge_kutta_3(f, x0, y0, h, n)

# Valor exato de y(1)
y_exact = exact_solution(1)

# Cálculo do erro absoluto
error_euler = abs(y_exact - y_euler)
error_runge_kutta = abs(y_exact - y_runge_kutta)
```

```
# Exibição dos resultados
print(f"Aproximação de y(1) com Euler: {y_euler:.5f}")
print(f"Aproximação de y(1) com Runge-Kutta de 3ª ordem: {y_runge_kutta:.5f}")
print(f"Valor exato de y(1): {y_exact:.5f}")
print(f"Erro absoluto de Euler: {error_euler:.5f}")
print(f"Erro absoluto de Runge-Kutta de 3ª ordem:{error_runge_kutta:.5f}")
```

1.4 Resultados

Aproximação de y(1) com de Euler: 1.82695
Aproximação de y(1) com de Runge-Kutta de 3ª ordem: 1.73247
Valor exato de y(1): 1.73205
Erro absolutode Euler: 0.09490
Erro absoluto de Runge-Kutta de 3ª ordem: 0.00042

2 QUESTÃO 2

2.1 Enunciado

Um projetil de massa $m = 0.11kg$, lançado verticalmente para acima com velocidade inicial $v(0) = 8m/s$, é detido pela força gravitacional $Fg = mg$ e a resistência do ar $Fr = -kv|v|$ onde $g = -9.8m/s$, $k = 0.002kg/m$. A equação¸ diferencial para a velocidade é dada por

$$mv' = mg–kv|v|$$

(a) Encontre a velocidade depois de 0.1s, 0.2s, ..., 1s.
(b) Numericamente, encontre o tempo no qual o projetil começa a cair.

2.2 Metodologia

A equação diferencial que descreve o movimento de um projétil verticalmente lançado para cima, sob a ação da gravidade e da resistência do ar, é dada por:

$$m\frac{dv}{dt} = mg - kv|v|$$

Onde:

- $m = 0.11\,kg$ é a massa do projétil,
- $g = -9.8\,m/s^2$ é a aceleração gravitacional,
- $k = 0.002\,kg/m$ é a constante de resistência do ar,
- $v(t)$ é a velocidade do projétil em função do tempo.

Rearranjando a equação, obtemos:

$$\frac{dv}{dt} = g - \frac{k}{m}v|v|$$

A condição inicial dada é:

$$v(0) = 8\,m/s$$

Isso significa que no tempo $t = 0$, a velocidade inicial do projétil é de 8 m/s.

Para resolver essa equação diferencial ordinária (EDO), utilizamos métodos numéricos, como o método de Euler ou Runge-Kutta, que permitem calcular a velocidade do projétil em intervalos de tempo discretos.

Para a parte (a), devemos calcular a velocidade do projétil após intervalos de 0.1 segundo, ou seja, para $t = 0.1, 0.2, \ldots, 1\,s$. O método de Euler é aplicado usando a fórmula iterativa:

$$v_{n+1} = v_n + h \cdot \left(g - \frac{k}{m} v_n |v_n|\right)$$

Onde h é o passo de tempo, neste caso $h = 0.1\,s$, e v_n é a velocidade no instante t_n.

O projétil começa a cair quando sua velocidade se torna zero ou negativa, o que significa que precisamos encontrar o tempo t_c em que $v(t_c) = 0$. Utilizando o método numérico, iteramos até que a velocidade passe de positiva para negativa, indicando o início da queda do projétil.

2.3 Implementação

```
import numpy as np

# Função que define a EDO dv/dt
def dv_dt(v):
    return g - (k / m) * v * abs(v)

# Método de Euler para resolver a EDO
def euler_method(v0, h, t_final):
    t_values = np.arange(0, t_final + h, h)
    v_values = np.zeros(len(t_values))
    v_values[0] = v0

    for i in range(1, len(t_values)):
        v_values[i] = v_values[i-1] + h * dv_dt(v_values[i-1])

    return t_values, v_values

# Encontrar o tempo em que o projétil começa a cair
def find_time_to_fall(v0, h):
    t = 0
    v = v0
    while v > 0:
        v = v + h * dv_dt(v)
```

```
        t += h
    return t

# Constantes
m = 0.11  # massa (kg)
g = -9.8  # gravidade (m/s^2)
k = 0.002  # coeficiente de resistência do ar (kg/m)
v0 = 8.0  # velocidade inicial (m/s)
h = 0.1  # intervalo de tempo (s)
t_final = 1.0  # tempo final (s)

# Resolver a EDO
t_values, v_values = euler_method(v0, h, t_final)

# Exibir os resultados
for t, v in zip(t_values, v_values):
    print(f"Tempo: {t:.1f}s, Velocidade: {v:.5f} m/s")

# Tempo em que o projétil começa a cair
time_to_fall = find_time_to_fall(v0, h)
print(f"O projetil começa a cair após {time_to_fall:.2f} segundos")
```

2.4 Resultados

a)

Tempo (s)	**Velocidade (m/s)**
0.0	8.00000
0.1	6.90364
0.2	5.83698
0.3	4.79504
0.4	3.77323
0.5	2.76735
0.6	1.77342
0.7	0.78770
0.8	-0.19343
0.9	-1.17336
1.0	-2.15085

Tabela 1: Velocidade ao longo do tempo

b) O projetil começa a cair entre 0.7 e 0.8 segundos.

3 QUESTÃO 3

3.1 Enunciado

Seja $P(t)$ o número de indivíduos de uma certa população medido em anos. Se a taxa de nascimentos é constante b e a taxa de mortalidade d é proporcional ao tamanho da população, então o crescimento da população é dado pela equação logística

$\frac{dP(t)}{dt} = bP(t) - k(P(t))^2$

Onde $d = kP(t)$. Suponha que $P(0) = 50976$, $b = 2.9 \times 10^{-2}$ e $k = 1.4 \times 10^{-7}$. Encontre a população estimada depois de 5 anos utilizando Runge-Kutta de ordem 4.

3.2 Metodologia

A equação diferencial que descreve o crescimento de uma população é dada por:

$$\frac{dP(t)}{dt} = bP(t) - k(P(t))^2$$

Onde:

- $P(t)$ é o número de indivíduos da população no tempo t,
- b é a taxa de nascimentos constante,
- k é a constante proporcional à taxa de mortalidade.

A condição inicial é $P(0) = 50976$, e os parâmetros fornecidos são:

- $b = 2.9 \times 10^{-2}$,
- $k = 1.4 \times 10^{-7}$.

A equação diferencial pode ser reescrita como:

$$\frac{dP}{dt} = bP - kP^2$$

Esta é uma equação diferencial não linear de primeira ordem.

O método de Runge-Kutta de 4ª ordem é um método numérico que fornece uma aproximação precisa da solução da equação diferencial. A fórmula geral para a atualização de P usando este método é:

$$P_{n+1} = P_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

Onde:

$$k_1 = f(t_n, P_n)$$

$$k_2 = f\left(t_n + \frac{h}{2}, P_n + \frac{h}{2}k_1\right)$$

$$k_3 = f\left(t_n + \frac{h}{2}, P_n + \frac{h}{2}k_2\right)$$

$$k_4 = f(t_n + h, P_n + hk_3)$$

Aqui, $f(t, P)$ é a função derivada da população:

$$f(t, P) = bP - kP^2$$

Para encontrar a população estimada após 5 anos, você deve integrar a equação diferencial de $t = 0$ até $t = 5$ anos. Utilizando o método de Runge-Kutta de 4ª ordem, o cálculo é realizado em passos de h, atualizando o valor de P a cada passo.

3.3 Implementação

```
import numpy as np

# Constantes do problema
b = 2.9e-2  # taxa de nascimento
k = 1.4e-7  # constante de mortalidade
P0 = 50976  # população inicial
t_final = 5  # anos
h = 0.1  # passo (tamanho do intervalo)

# Função que define a EDO dP/dt
def dP_dt(t, P):
    return b * P - k * P**2

# Método de Runge-Kutta de 4ª ordem
def runge_kutta_4(dP_dt, P0, t_final, h):
    t_values = np.arange(0, t_final + h, h)
```

```
    P_values = np.zeros(len(t_values))
    P_values[0] = P0

    for i in range(1, len(t_values)):
        t = t_values[i-1]
        P = P_values[i-1]

        k1 = h * dP_dt(t, P)
        k2 = h * dP_dt(t + h/2, P + k1/2)
        k3 = h * dP_dt(t + h/2, P + k2/2)
        k4 = h * dP_dt(t + h, P + k3)

        P_values[i] = P + (k1 + 2*k2 + 2*k3 + k4) / 6

    return t_values, P_values

# Resolver a EDO
t_values, P_values = runge_kutta_4(dP_dt, P0, t_final, h)

# Exibir a população após 5 anos
populacao_final = P_values[-1]
print(f"População estimada após 5 anos: {populacao_final:.2f} indivíduos")
```

3.4 Resultados

População estimada após 5 anos: 56751.04 indivíduos.

4 Links dos Códigos

- Código 1: https://github.com/kaiocesarb15/Calculo_Numerico/blob/main/Prova3/Questao1.py
- Código 2: https://github.com/kaiocesarb15/Calculo_Numerico/blob/main/Prova3/Questao2.py
- Código 3: https://github.com/kaiocesarb15/Calculo_Numerico/blob/main/Prova3/Questao3.py