UNIVERSIDADE FEDERAL DE SÃO CARLOS
Centro de Ciências Exatas e de Tecnologia
Departamento de Física

Trabalho 3

por

Vitor de Souza Barboza/791446 - Curso: Engenharia Física

Departamento de Física
São Carlos - SP, 4 de março de 2023

Ministrante:
Prof. Leonardo Kleber Castelano

1 Introdução

Este trabalho consiste na resolução do "Trabalho 3"da disciplina de Física Computacional 1. Todos os arquivos utilizados na prática podem ser acessados no repositório do GitHub [1]

2 Resolução

- **Questão 1**

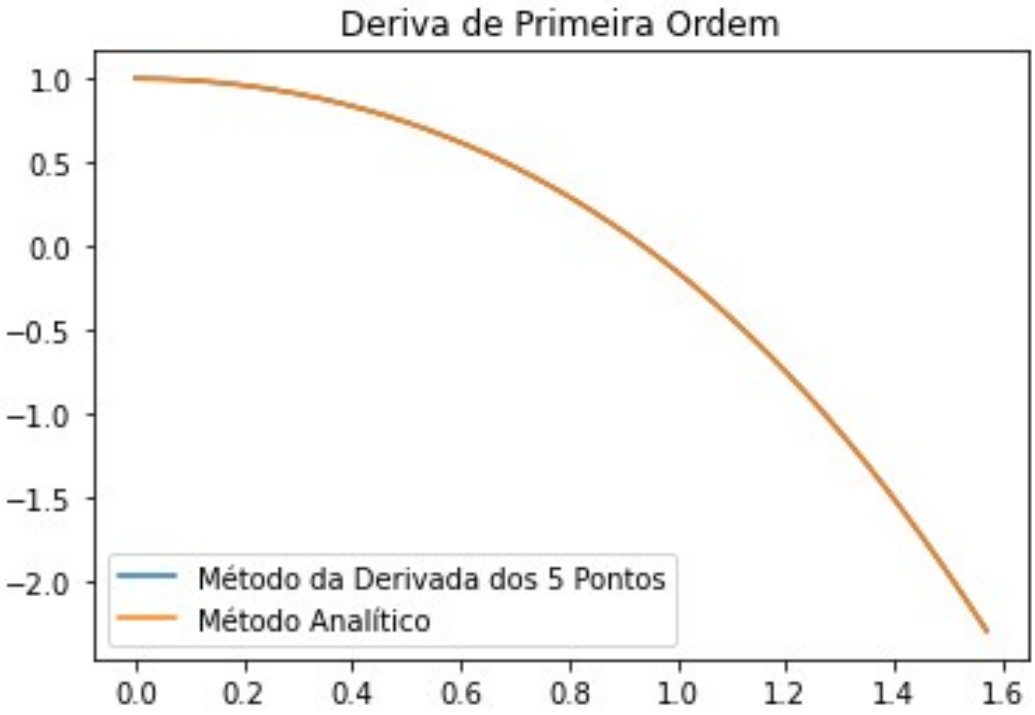
  O código para resolução da questão 1 pode ser visto abaixo:

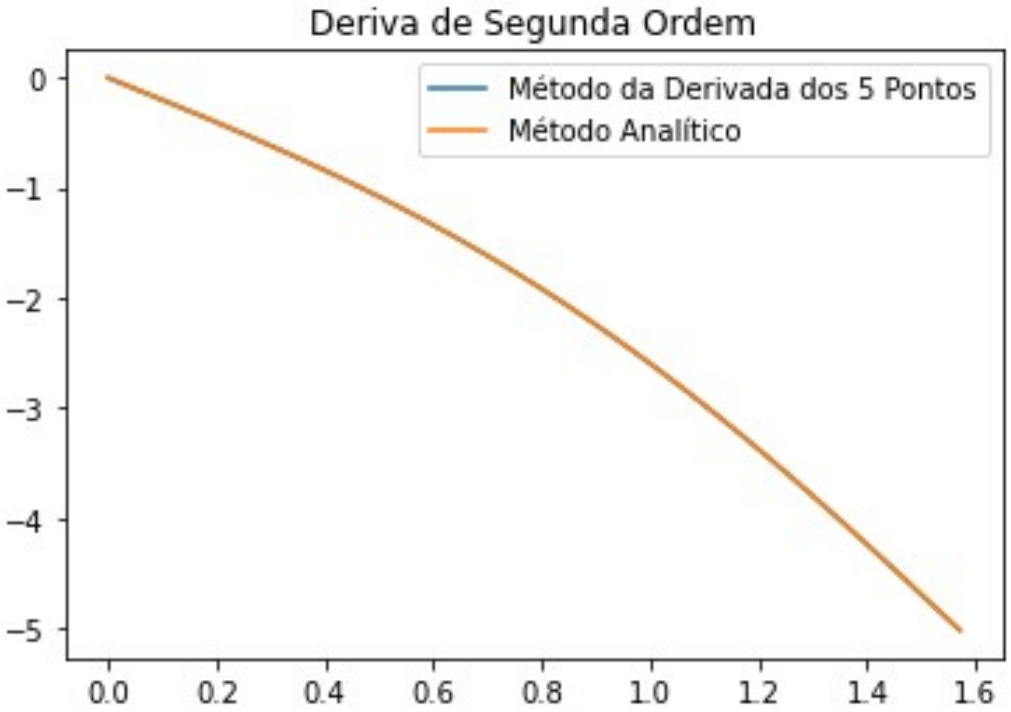
```
[1] import numpy as np
    import matplotlib.pyplot as plt
```

```
[3] # Questão 1

    # Derivada de Primeira Ordem

    def f(x):
      y = np.cos(x)*np.sinh(x)
      return y

    def f1_analitic(x):
      y = -1*np.sin(x)*np.sinh(x) + np.cosh(x)*np.cos(x)
      return y

    f1_analitico = []
    y = []
    x = np.linspace(0,np.pi/2,1001)
    for i in range(1001):
      y.append(f(x[i]))
      f1_analitico.append(f1_analitic(x[i]))


    h = x[1] - x[0]

    f1 = []
    for i in range(1001):
      if i == 0 or i == 1:
        a = (y[i+1]-y[i])/(h)
        f1.append(a)
      elif i == 999 or i == 1000:
        a = (y[i]-y[i-1])/(h)
        f1.append(a)
      else:
        a = (y[i-2]-(8*y[i-1])+(8*y[i+1])-y[i+2])/(12*h)
        f1.append(a)
```

```
plt.plot(x,f1, label = "Método da Derivada dos 5 Pontos")
plt.plot(x,f1_analitico, label = "Método Analítico")
plt.title("Deriva de Primeira Ordem")
plt.legend()
plt.show()

# Derivada de Segunda Ordem

def f2_analitic(x):
  y = -2*np.sin(x)*np.cosh(x)
  return y


y1 = []
f2_analitico = []
for i in range(1001):
  y1.append(f1_analitic(x[i]))
  f2_analitico.append(f2_analitic(x[i]))

f2 = []

for i in range(1001):
  if i == 0 or i == 1:
    a = (y1[i+1]-y1[i])/(h)
    f2.append(a)
  elif i == 999 or i == 1000:
    a = (y1[i]-y1[i-1])/(h)
    f2.append(a)
  else:
    a = (y1[i-2]-(8*y1[i-1])+(8*y1[i+1])-y1[i+2])/(12*h)
    f2.append(a)


plt.plot(x,f2, label = "Método da Derivada dos 5 Pontos")
plt.plot(x,f2_analitico, label = "Método Analítico")
plt.title("Deriva de Segunda Ordem")
plt.legend()
plt.show()
```

O código acima apresentou saída (resultado):





- **Questão 2**

  O código para resolução da questão 2 pode ser visto abaixo:

```
# Questão 2
b = 1
a = 0
N = 10000
h = (b - a)/N

def f(x):
    y = 4/(1+(x**2))
    return y

x = np.linspace(0,1,10000)
y = []
for i in range(10000):
    y.append(f(x[i]))

# Método do Trapézio
res_trap = 0
for i in range(N):
        if i == 0 or i == N-1:
            res_trap += y[i]
        else:
            res_trap += 2*y[i]
res_trap = h*res_trap/2
print("O resultado pelo método do trapézio: ", res_trap)

# Método de Simpson 1/3
res_terço = 0
for i in range(N):
    if i == 0 or i == N-1:
        res_terço += y[i]
    elif i % 2 == 0:
        res_terço += 2*y[i]
    elif i % 2 == 1:
        res_terço += 4*y[i]
res_terço = h*res_terço/3
print("O resultado pelo método de Simpson 1/3: ", res_terço)

# Método de Simpson 3/8

res_oitavo = 0
for i in range(N):
    if i == 0 or i == N-1:
        res_oitavo += y[i]
    elif i % 3 == 0:
        res_oitavo += 2*y[i]
    elif i % 3 == 1 or i % 3 == 2:
        res_oitavo += 3*y[i]
res_oitavo = 3*h*res_oitavo/8
print("O resultado pelo método de Simpson 3/8: ", res_oitavo)
```

O código acima apresentou saída (resultado):

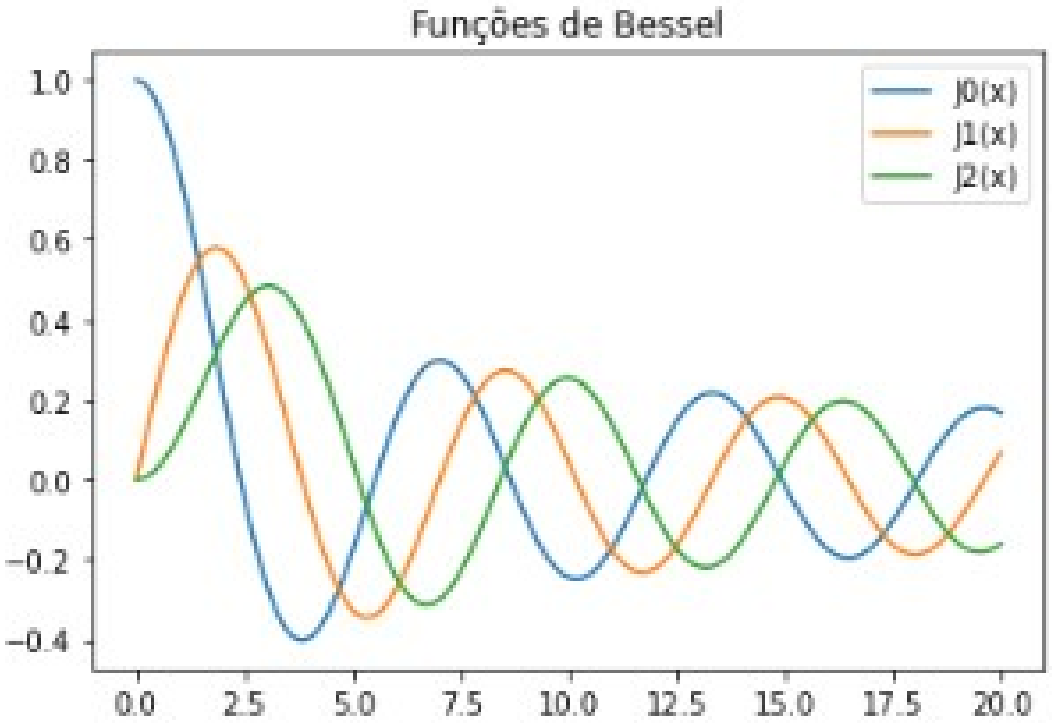
```
O resultado pelo método do trapézio:  3.141278492657605
O resultado pelo método de Simpson 1/3:  3.1412118243241083
O resultado pelo método de Simpson 3/8:  3.141278494324436
```

- **Questão 3**

  O código para resolução da questão 3-a) pode ser visto abaixo:

```python
# Questão 3 - a)

def Bessel(m,x1):
    a = 0
    b = np.pi
    N = 1000
    h = (b-a)/N
    x1 = x1
    m = m
    x = np.linspace(a,b,N)
    y = []
    def func(theta):
        y = np.cos(m*theta - x1*np.sin(theta))
        return y
    for i in range(N):
        y.append(func(x[i]))
    res_bessel = 0
    for i in range(N):
        if i == 0 or i == N-1:
            res_bessel += y[i]
        elif i % 2 == 0:
            res_bessel += 2*y[i]
        elif i % 2 == 1:
            res_bessel += 4*y[i]
    res_bessel = h*res_bessel/(3*np.pi)
    return res_bessel

x = np.linspace(0,20,100)
y0 = []
y1 = []
y2 = []
for i in x:
    y0.append(Bessel(0,i))
    y1.append(Bessel(1,i))
    y2.append(Bessel(2,i))

plt.plot(x,y0, label = "J0(x)")
plt.plot(x,y1, label = "J1(x)")
plt.plot(x,y2, label = "J2(x)")
plt.title("Funções de Bessel")
plt.legend()
plt.show()
```

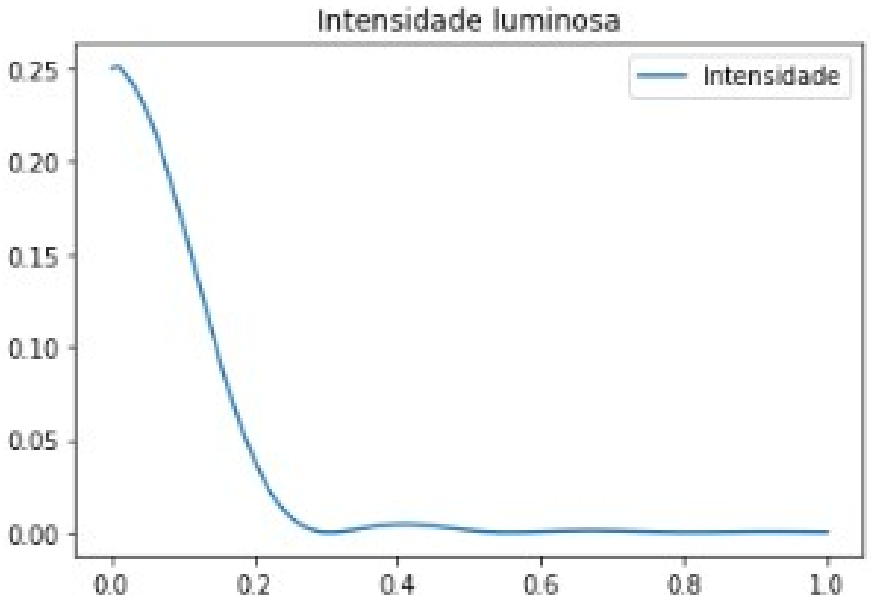
O código acima apresentou saída (resultado):



O código para resolução da questão 3-b) pode ser visto abaixo:

```
# Questão 3 - b)
comp_onda = 0.5 #(micrometros)
k = 2*np.pi/comp_onda
def I(r):
    y = (Bessel(1,k*r)/(k*r))**2
    return y
x = np.linspace(0,1,95)
y = []
for i in range(95):
    if i == 0:
        y.append(0.25)
    else:
        y.append(I(x[i]))

plt.plot(x,y, label = "Intensidade")
plt.title("Intensidade luminosa")
plt.legend()
plt.show()
```

O código acima apresentou saída (resultado):



Bibliografia

[1] https://github.com/vitorsbarboza/FisComp1