

Mini relatório de MetComp: Tarefa erro Integral

Aluno: Marcelo Camaran Lucas

Foi pedido para confeccionar um programa que tivesse as características de retornar os valores da integral numérica e erros correspondente ao cálculo. Também criar um gráfico para demonstrar a progressão do erro conforme a diminuição do intervalo de dx. Para realizar este trabalho, foi usado dois métodos, o de trapézio e o Simpson.

Os parâmetros utilizados foram os seguintes:

$$f(x) = \frac{1}{(x^2 + 1)}$$

e

$$x \in (-3, 3)$$

E para o número de fatias N foi criado um aranje que varia conforme

$$N = (2^2, \dots, 2^{20})$$

O método do trapézio é feito separando a área abaixo da curva em N trapézios, assim a precisão é maior que usando retângulos. A largura de cada fatias no intervalo de (a,b) pode ser representada da seguinte maneira:

$$\Delta x = \frac{(b - a)}{N}$$

Assim cada fatia pela altura média dos dois lados da fatia vai resultar na área do trapézio. A Integral pode ser escrita da seguinte forma:

$$\int_a^b f(x) dx \cong \left[\frac{f(x_0) + f(x_N)}{2} + \sum_{i=1}^{N-1} f(x_i) \right] \Delta x$$

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import math
4
5 #Metodo do Trapézio
6
7 def f(x):
8     return 1/(math.pow(x, 2) + 1)
9
10 n = []
11 for i in np.arange(2, 22, 2):
12     n.append(int(math.pow(2, i)))
13 #print (n)
14
15 a = -3    #valor inicial
16 b = 3     #valor final
17
18 def integral_trapezio(f, a, b, n):
```

```

19     dx = (b-a)/n
20     soma_1 = (f(a) + f(b))/2
21     soma_2 = 0
22     for i in range (n):
23         x = a + i*dx
24         soma_2 += f(x)
25     return (soma_1 + soma_2)*dx
26
27 print(integral_trapezio(f, a, b, 10)) #o valor de N=10 é apenas para teste

2.5546898806889313

```

Podemos calcular o erro com a seguinte equação:

$$\varepsilon = |I_{exata} - I(N)|$$

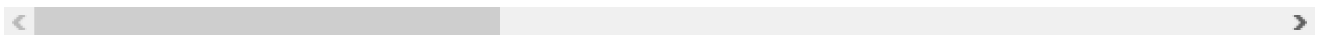
Clique duas vezes (ou pressione "Enter") para editar

```

1 def f_analitica(a, b):
2     return math.atan(b) - math.atan(a)
3
4 print(f_analitica(a, b))
5
6 erro_trapezio = []
7 for i in range(10):
8     erro_trapezio.append(math.fabs(f_analitica(a, b) - integral_trapezio(f, a, b, n[i]))
9 print(erro_trapezio)

2.498091544796509
[0.22498537828041432, 0.036097225861289495, 0.009287121729578907, 0.0023382568842156,

```



```

1 x_grafico = []
2 for i in n:
3     x_grafico.append((b-a)/i)
4 #print(y)
5
6 plt.figure()
7 plt.grid()
8 plt.plot(x_grafico, erro_trapezio, 'r')
9 plt.xscale('log')
10 plt.yscale('log')
11 plt.xlabel('log(Erro)')
12 plt.ylabel('log(Δx)')
13 plt.show()
14
15 def inclinacao(erro, a, b, n):
16     dx1 = (b - a)/n[0]
17     dx2 = (b - a)/n[9]
18     return (math.log(dx2) - math.log(dx1))/(math.log(erro[9]) - math.log(erro[0]))
19
20 print(f'Inclinação da reta do trapézio: {inclinacao(erro_trapezio, a, b, n)}')

```

Na segunda parte da tarefa, foi utilizado o método simpson, este método utiliza de aproximar a função por uma constante para cada fatia.

Para cada par de fatias, é criado uma coleção de parábolas, elas dão os 3 pontos que precisamos para criar uma parábola. A integral pode ser aproximada como:

$$\int_a^b f(x) dx \cong \frac{h}{3} \left[f(a) + f(b) + 4 \sum_{i=1}^{\frac{N}{2}} f(a + (2i-1)h) + 2 \sum_{i=1}^{\frac{N}{2}-1} f(a + 2ih) \right]$$

h é a largura das fatias

```

1 #Método de Simpson
2
3 def integral_simpson(f, a, b, n):
4     soma_par = 0
5     soma_impar = soma_par
6     dx = (b - a)/n
7     for i in range(1, n, 2):
8         soma_impar += f(a + i*dx)
9     for i in range(2, n, 2):
10        soma_par += f(a + i*dx)
11    return (dx/3)*(f(a) + f(b) + 4*soma_impar + 2*soma_par)
12
13 print(integral_simpson(f, a, b, 10))    #o valor de N=10 é apenas para teste
14
15 erro_simpson = []
16 for i in range(10):
17     erro_simpson.append(math.fabs(f_analitica(a, b) - integral_simpson(f, a, b, n[i])))
18 print(erro_simpson)
19
20 plt.figure()
21 plt.grid()
22 plt.plot(x_grafico, erro_simpson, 'r')
23 plt.xscale('log')
24 plt.yscale('log')
25 plt.xlabel('log(Erro)')
26 plt.ylabel('log(Δx)')
27 plt.show()
28
29 print(f'Inclinação da reta de Simpson: {inclinacao(erro_simpson, a, b, n)}')
```

Para concluir, com base na análise dos gráficos e a inclinação, podemos ver que o método Simpson é o que mais tem precisão, julgando a base de erro dos dois métodos para a mesma função.

Podemos notar também que, o valor da inclinação corresponde a taxa de decréscimo do erro em razão da largura das fatias, portanto, novamente mostrando que o método simpson é mais efetivo.

✓ 1s conclusão: 19:14

