/ /

D S T Q Q S S

Trabalho Aplicado

i $x^3 + 5\,\text{sen}(x) = -2x - 4 \equiv x^3 + 5\,\text{sen}(x) + 2x + 4$

$f(x) = x^3 + 5\,\text{sen}(x) + 2x + 4$

$f(-1) = -1^3 + 5\,\text{sen}(-1) + 2 \cdot -1 + 4$

$f(-1) \cong -3,207//$

$f(0) = 0 + 0 + 0 + 4$

$= 4//$

Como os dois pontos têm valores com sinais opostos, podemos afirmar que existe pelo menos uma solução neste intervalo //

```python
import math                             #Bibilioteca para equações
import matplotlib.pyplot as grph        #Biblioteca para construção do gráfico

def MaiorMenor(x,y):                    #Define qual dos valores dados é maior, para assim organizar na hora de efetuar as operação
    if x>y:
        z = y
        y = x
        x = z
    return x,y
def Equacao1(x):                        #Resolve o problema pela primeira vez e retorna seu resultado
    sen = math.sin(x)
    resultado = ((x * x) * x) + (5 * sen) + (2 * x) + 4
    return resultado
def TrueOrFake(x,y):                    #Verifica se resultado das equações tem sinais opostos
    if x < 0 and y > 0:
        return 1
    else:
        return 0
def ReduzirEquacao(x,y):                #Função que reduz o tamanho das equações em 0.05 a cada passagem pelo loop,
    v1,v2 = [], []                      #e armazena todos os valores tanto de resultado quanto entrada em dois vetores
    c = 1                               #após isso retorna os últimos valores (valores com diferença de 0.1) e os vetores com todos os valores
    v1.append(x)
    sen1 = math.sin(x)
    resultado1 = ((x * x) * x) + (5 * sen1) + (2 * x) + 4
    v2.append(resultado1)
    sen2 = math.sin(y)
    resultado2 = ((y * y) * y) + (5 * sen2) + (2 * y) + 4
    while resultado1 < -0.05:
        resultado1 = ((x * x) * x) + (5 * sen1) + (2 * x) + 4
        v2.append(resultado1)
    v1.append(y)
    v2.append(resultado2)
    while resultado2 > 0.05:
        y -= 0.05
        v1.insert(len(v1)-c,y)
        sen2 = math.sin(y)
        resultado2 = ((y * y) * y) + (5 * sen2) + (2 * y) + 4
        v2.insert(len(v2)-c,resultado2)
        c += 1
    return x,y,v1,v2
def Main():                             #Função Main, que executa todas as outras funções de maneira ordenada
    ptsxy, rsltsxy = [], []
    x1 = Equacao1(x)
    x2 = Equacao1(y)
    if TrueOrFake(x1,x2):
        xnovo1,xnovo2, ptsxy, rsltsxy = ReduzirEquacao(x,y)
        print(f"\nA equação tem pelo menos uma solução neste intervalo\nIntervalo esse que fica entre {xnovo1:.2f} e {xnovo2:.2f}")
        grph.plot(ptsxy,rsltsxy)        #Utilizando a biblioteca para o gráfico, insiro os dois vetores assim os armazenando
        grph.show()                     #Exibe o gráfico com os valores dos vetores informados
    else:
        print("\nNão é possível afirmar que existe solução neste intervalo, tente outros dois números\n")
while True:                             #Laço para continuar executando -- 1- para continuar e 2-- para sair
    ask = int(input("Digite 1 para fazer uma operação, 2 para sair: "))
    if ask == 1:
        x,y = input("\nDigite seus números(ambos menores que 1000): ").split()
        x = float(x)
        y = float(y)
        x,y = MaiorMenor(x,y)
        if x < 1000 and y < 1000:
            Main()
```

```
63              Main()
64          print()
65      elif ask == 2:
66          break
67      else:
68          print("\nOpção não encontrada, tente novamente\n")
69
```