Métodos Numéricos Computacionais - APS-AV1 - 2022.2

- Aluno: João Pedro Espechit Silveira - 2019200901
- Professor: Sérgio Assunção Monteiro
- Turma: 145R

O código também pode ser visualizado na íntegra por meio

[DESTE LINK]

Atividade

Questão 1

Dada a equação **f(x) = $x^4 - x^3 - x^2 + x + 8 = 0$, obtenha o valor aproximado da raiz. Utilizando o método da bissecção no intervalo [-2,5; 2,5] com precisão 1E-06.**

```
#import numpy as np


def bissec(f, a, b, epsilon, maxIter = 50):

    Fa = f(a)
    Fb = f(b)


    if(Fa*Fb>0):
      print("Erro! A função não muda de sinal.")
      return (True, None)

    print("k\t  a\t\t  fa\t\t  b\t\t  fb\t\t  x\t\t  fx\t\tintervX")

    intervX = abs(b-a)
    x = (a+b)/2
    Fx = f(x)

    print("-\t%e\t%e\t%e\t%e\t%e\t%e\t%e" % (a, Fa, b, Fb, x, Fx, intervX))

    if(intervX<=epsilon):
      return(False,x)

    k=1
```

```
    while k <= maxIter:
      if Fa*Fx>0:
        a=x
        Fa=Fx
      else:
        b=x
        Fb=Fx
      intervX = abs(b-a)
      x = (a+b)/2
      Fx = f(x)

      print("%d\t%e\t%e\t%e\t%e\t%e\t%e\t%e"%(k, a, Fa, b, Fb, x, Fx, intervX))

      if(intervX<=epsilon):
        return(False,x)

      k = k+1

    print("ERRO! número máximo de iterações atingido.")
    return(True, x)

def f(x):
    #return x**3 - 9*x + 3
    return x**4 - x**3 - x**2 + x + 8
    #return x**4 -  x**3 - x**2 + x
    #return np.log10(x)*x - 1


#test
print(f(1))
```

```
1
```

```
a = -2.5
b = 2.5
#epsilon = 0.001
epsilon = 1e-06
maxIter = 100


(checkError, result) = bissec(f, a, b, epsilon, maxIter)
```

```
Erro! A função não muda de sinal.
```

```
if checkError:
    print("O Método da Bisseção retornou um erro.")
if result is not None:
    print("Raiz encontrada: %s" % result)
```

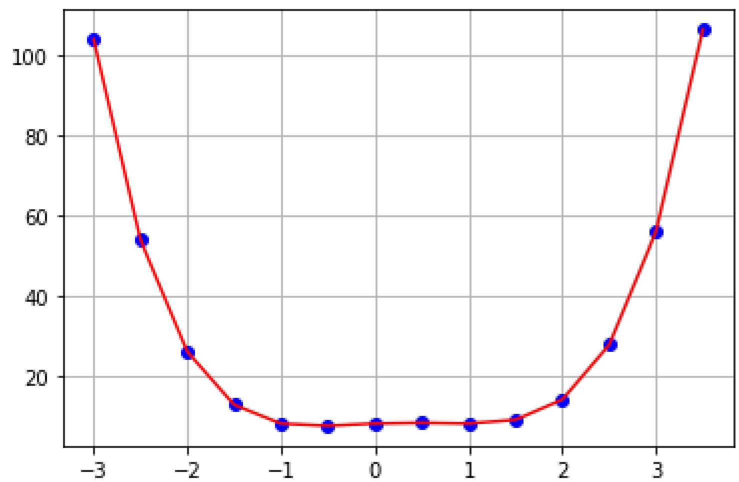
```
O Método da Bisseção retornou um erro.
```

Resposta

Como a função não corta o ponto 0 no eixo y, como demonstrado no gráfico abaixo.

Sendo assim, impossível tirar suas raízes pelo método da bissecção.

```
import matplotlib.pyplot as plt
x = np.arange(-3.0, 4.0, 0.5)
plt.figure()
plt.grid()
plt.plot(x,f(x),'bo',x,f(x),'r-')
plt.show()
```



Questão 2

Sejam x=[1; 3; 5; -6; 6] e sua aproximação x̄=[2; -4; 6; 8; -7]. Calcule o erro absoluto e o erro relativo (usar x como referência)

```
import numpy as np
#import math


def Norm2(x,xLine):
  counter=0
  n2=0
  if len(x)!=len(xLine):
    print("Não é possível calcular a Norma 2. Arrays têm tamanhos diferentes")
    return None
  for counter in range(len(x)):
    n2=n2+(x[counter]-xLine[counter])
    #print(n2)
```

```
  #return(np.sqrt((x[0]-xLine[0]+x[1]-xLine[1]+x[2]-xLine[2]+x[3]-xLine[3]+x[4]-xLine[4])**2)
  return np.sqrt(n2**2)

def Norm(x):
  counter=0
  n=0
  for counter in range(len(x)):
    n=n+x[counter]
    #print(n)
  return np.sqrt(n**2)


def rel_Error(norm2,norm):
  return norm2/norm


def truncate(value,Range):
    temp = str(value)
    for counter in range(len(temp)):
        if temp[counter] == '.':
            try:
                return float(temp[:counter+Range+1])
            except:
                return float(temp)
    return float(temp)



x=[1,3,5,-6,6]
xLine=[2,-4,6,8,-7]
print(" x: ",x)
print(" x̅: ",xLine)
if(Norm2(x,xLine)!=None):
  norm=Norm(x)
  norm2=Norm2(x,xLine)
  relError=rel_Error(norm2,norm)
  print("Erro absoluto: ",norm2)#Norm2(x,xLine))
  print("Erro relativo: ",relError)#rel_Error(Norm2(x,xLine),Norm(x)))
  print("Erro relativo truncado(3 casas decimais)",truncate(relError,3))
  print("Erro relativo arredondado (3 casas decimais): %.3f"%round(rel_Error(Norm2(x,xLine),N
#Norm(x)
#Norm2(x,xLine)
#print(xLine)
```

```
 x:  [1, 3, 5, -6, 6]
 x̅:  [2, -4, 6, 8, -7]
Erro absoluto:  4.0
Erro relativo:  0.4444444444444444
Erro relativo truncado(3 casas decimais) 0.444
Erro relativo arredondado (3 casas decimais): 0.440
```

Resposta

Após a execução dos cálculos, foi possível identificar os seguntes valores:

> Erro Absoluto: 4
>
> Erro Relativo: 0.4444444444444444
>
> Erro Relativo (Truncado): 0.444
>
> Erro Relativo (Arredondado): 0.440