24_1s_MTM1013T11_Q01_AndreGoncalvesdaSilva

April 8, 2024

MTM1013 - MÉT. NUM. ECOMPUTACIONAIS Prof. Charles R. P. Szinvelski

0.0.1 QUESTÃO (01)

24/1s - UFSM Data: 08/04/24 (Das 8h às 9h)

Acadêmico: André Gonçalves da Silva Matrícula: 202210071
Curso/Turma: 123/11 A tarefa QUESTÃO (01) tem por finalidade a aplicação direta dos aspectos teóricos e práticos dos métodos numéricos abordados na disciplina (Zeros de Funções e recursos de programação em linguagem PYTHON 3 empregados nas respectivas implementações). A produção do arquivo-resposta deverá seguir os moldes da produção dos arquivos respostas (formatos IPYNB e PDF e ambos deverão ser entregues via TAREFA aberta no MOODLE) para a QUESTÃO (01) de 08/04/2024.

OBSERVAÇÕES

- Salienta-se que observância da nomeação dos arquivos IPYNB e PDF (ver abaixo) e informações do cabeçalho (prencher devidamente) serão partes constituinte do escore (divisões sucessivas por 2):
  - 24_1s_MTM1013T11_Q01_NomeSobrenome.ipynb;
  - 24_1s_MTM1013T11_Q01_NomeSobrenome.pdf;
- Os algoritmos que devem ser empregados para as implementações em linguagem PYTHON 3 e *testes de mesas* são os algoritmos dados pelo material didático disponibilizado nos repositórios MEGA e DRIVE e não os sendo as respostas serão desconsideradas. Ressalta-se que os algoritmos do Material Didático são adequações dos algoritmos apresentados em Campos Filho (2018)[ ], os quais podem ser utilizados para a implementação;
- A elaboração das questões buscará caracterizações personalizadas para cada aluno, fato que resultará em um diversificado conjunto de situações-testes para a exploração dos aspectos teóricos e práticos da disciplina em detrimento, eventualmente, da aplicabilidade do tema abordado na questão.

[ ]: Filho, Frederico Ferreira C. Algoritmos Numéricos - Uma Abordagem Moderna de Cálculo Numérico, 3ª edição. Disponível em: Minha Biblioteca, Grupo GEN, 2018. Ainda, há exemplares na BSCCNE.

Obtenção de dados para elaboração das questões.

Ao considerar o meu número de matrícula na UFSM, 3332882, monta-se a tabela

d1	d2	d3	d4	d5	d6	d7	d8	d9	d10
0	0	0	3	3	3	2	8	8	2

e sobre essas informações, colocam-se as seguintes elaborações:

(I) Com as informações da matrícula:

- $aux = |d9 - d10| = |8 - 2| = 6$, e se $aux = 0$, considere $aux = 2$;
- $p = \left\lceil \frac{d1+\ldots+d4}{aux} \right\rceil = \left\lceil \frac{0+0+0+3}{6} \right\rceil = \lceil 0,5 \rceil = 1$;
- $q = \left\lfloor \frac{d5+d6+d7+d8}{aux} \right\rfloor = \left\lfloor \frac{3+3+2+8}{6} \right\rfloor = \lfloor 2,67 \rfloor = 2$. Se $q = 0$ considere $q = 1$;
- $Dados = [aux, p, q] = [6, 1, 2]$;

Montar a função f definida em $(-\infty, +\infty)$:

$$\begin{aligned} f(x) = \exp\left(-p\,|x|\right)\cos\left(q|x|\right) &\overset{DADOS}{\Leftrightarrow} f(x) = \exp\left(-\,|x|\right)\cos\left(2|x|\right). \\ &\Leftrightarrow f(x) = \exp\left(-\,|x|\right)\cos\left(2x\right) \end{aligned}$$

(Gráfico via WolframAlpha e a simplificação da paridade de cos.)

Uma aplicação dessa função está descrita no artigo: Uma revisão teórica sobre funções de Autocorrelação aplicadas a altas e baixas Velocidades do vento (DEGRAZIA,2014)[1].

(II) Para gráficos pode-se usar as páginas tutoriais: MATPLOLIB, SYMPY ou SYMPY(2).

[1]: DEGRAZIA *et al.* Uma revisão teórica sobre funções de Autocorrelação aplicadas a altas e baixas Velocidades do vento. Ciência e Natura, Santa Maria, vol. 36, Ed. Especial, pg. 101-107, março, 2014. Disponível em: < https://periodicos.ufsm.br/cienciaenatura/article/view/13222/pdf_1 >. Acesso em: 14/05/2023.

0.0.2 QUESTÕES.

1) (0,5 ponto) Faça o que se pede, a partir das intruções acima:

1.1) (0,25 ponto [Dados]) Informe sua matrícula ($matricula = [d1, d2, d3, d4, d5, d6, d7, d8, d9, d10]$) e Dados($Dados = [a, b, c]$) como uma lista;

1.2) (0,25 ponto [Dados]) Monte a função f e seu gráfico;

2) (1,0 ponto) A partir de 1.2), aplique dois métodos distintos de obtenção de *Zero de Funções* para obter o primeiro ponto crítico positivo da função f para $x > 0$ e ilustre graficamente essas informações;

OBSERVAÇÃO: Colocar os documentos IPYNB e PDF da resolução no espaço TAREFA do MOODLE aberta, caso queira entregar o arquivo-resposta com esses recursos computacionais. Caso se resolva via Google COLAB, colocar link compartilhado sem restrições de acesso e também o arquivo PDF associado.

```
[ ]: import numpy as np
     import matplotlib.pyplot as plt
     from math import *
     from tabulate import tabulate
```

```
[ ]: d1 = 0
     d2 = 2
     d3 = 0
     d4 = 2
     d5 = 2
     d6 = 1
     d7 = 0
     d8 = 0
     d9 = 7
     d10 = 1
```

Matrícula = Matricula é [0, 2, 0, 2, 2, 1, 0, 0, 7, 1]

```
[ ]: aux = abs(d9 - d10)
     if aux == 0:
         aux = 2

     p = ceil((d1+d2+d3+d4)/aux)
     q = floor((d5+d6+d7+d8)/aux)
     if q == 0:
         q = 1
```

$$x > 0$$
$$f(x) = e^{-x}\cos(x)$$
$$f'(x) = -e^{-x}\cos(x) - e^{-x}\sin(x)$$
$$f''(x) = 2e^{-x}\sin(x)$$

```
[ ]: f = lambda x: exp(-p*abs(x))*cos(q*abs(x))
     derf = lambda x: -exp(-x)*cos(x) - exp(-x)*sin(x)
     der2f = lambda x: 2*exp(-x)*sin(x)
```

```
[ ]: def bissecao(f, a, b, eps, max_iter=100):
         points = []
         # aqui eu estou definindo o x anterior como o próprio a,
         # mas nas proximas definições de x anterior vai ser o xk
         x = a
         error_rel = lambda x, xk: abs(xk - x) / abs(xk)
         error_abs = lambda x, xk: abs(xk - x)
```

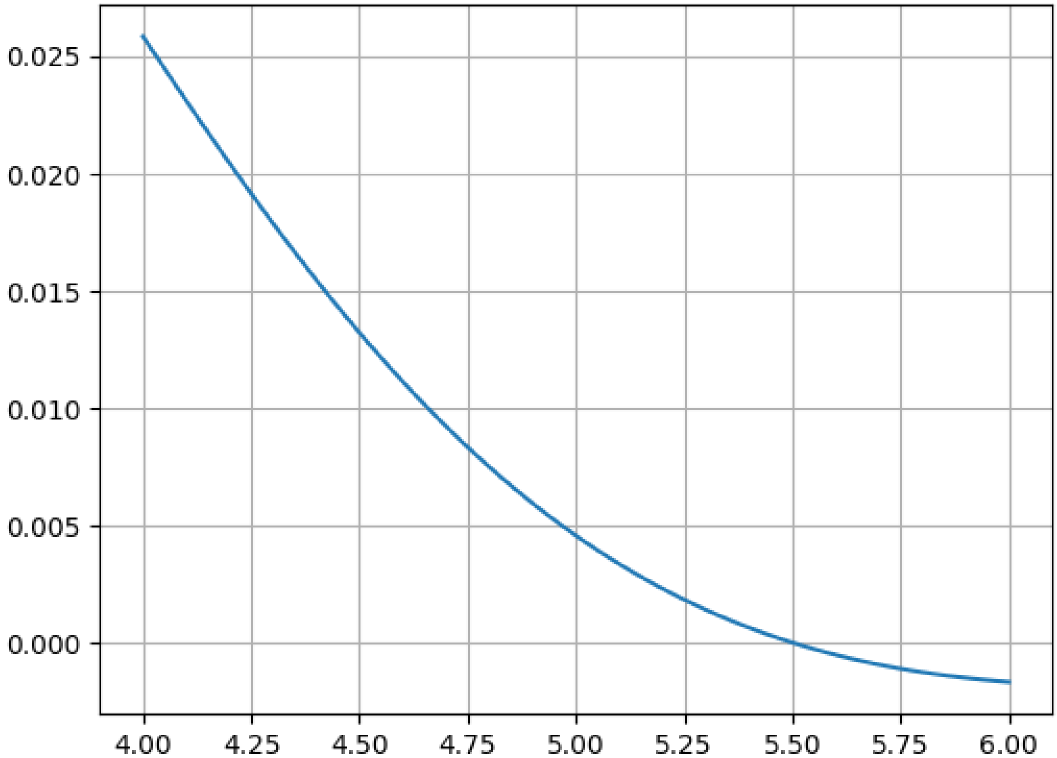
```
    for i in range(max_iter):
        xk = (a + b) / 2

        points.append([i+1, a, b, xk, f(xk), error_abs(x, xk), error_rel(x,␣
↪xk)])
        if error_rel(x, xk) < eps:
            return points

        if f(a)*f(xk) < 0:
            b = xk
        elif f(a)*f(xk) > 0:
            a = xk

        x = xk

    return points
```

[ ]:
```
def newton(f, derf, x0, eps, maxiter=100):
    xk = x0
    xk1 = lambda xk: xk - f(xk)/derf(xk)
    ek = lambda xk1, xk: abs(xk1 - xk)

    points = []
    for k in range(maxiter):
        xk1_val = xk1(xk)
        points.append([k, xk, xk1_val, f(xk1_val), ek(xk1_val, xk)/
↪abs(xk1_val)])

        if ek(xk1_val, xk)/abs(xk1_val) < eps:
            break

        xk = xk1_val

    return points
```

[ ]:
```
headersbiss = ["k", "a", "b", "x", "f(x)", "Erro Absoluto", "Erro Relativo"]
headersnewt = ['k', 'xk', 'xk+1', 'f(xk+1)', 'ek+1/|xk+1|']
```

Graficamente o primeiro ponto crítico está entre 2 e 4.

[ ]:
```
derfvectorized = np.vectorize(derf)

x = np.linspace(4, 6, 100)
y = derfvectorized(x)
plt.plot(x, y)
plt.grid()
plt.show()
```



[ ]:
```
eps = 1e-2
pointsbiss = bissecao(derf, 4.0, 6.0, eps)
pointsnewt = newton(derf, der2f, 5.0, eps)

print("------ METODO DA BISSECAO -------")
print(tabulate(pointsbiss, headers=headersbiss, floatfmt=".8f"))
print()
print("------ METODO DE NEWTON -------")
print(tabulate(pointsnewt, headers=headersnewt, floatfmt=".8f"))
```

```
------ METODO DA BISSECAO -------
  k           a           b           x         f(x)    Erro Absoluto     Erro
Relativo
---  ----------  ----------  ----------  -----------  ---------------
---------------
  1  4.00000000  6.00000000  5.00000000   0.00454988       1.00000000
0.20000000
  2  5.00000000  6.00000000  5.50000000  -0.00001279       0.50000000
0.09090909
  3  5.00000000  5.50000000  5.25000000   0.00182010       0.25000000
0.04761905
  4  5.25000000  5.50000000  5.37500000   0.00080213       0.12500000
```

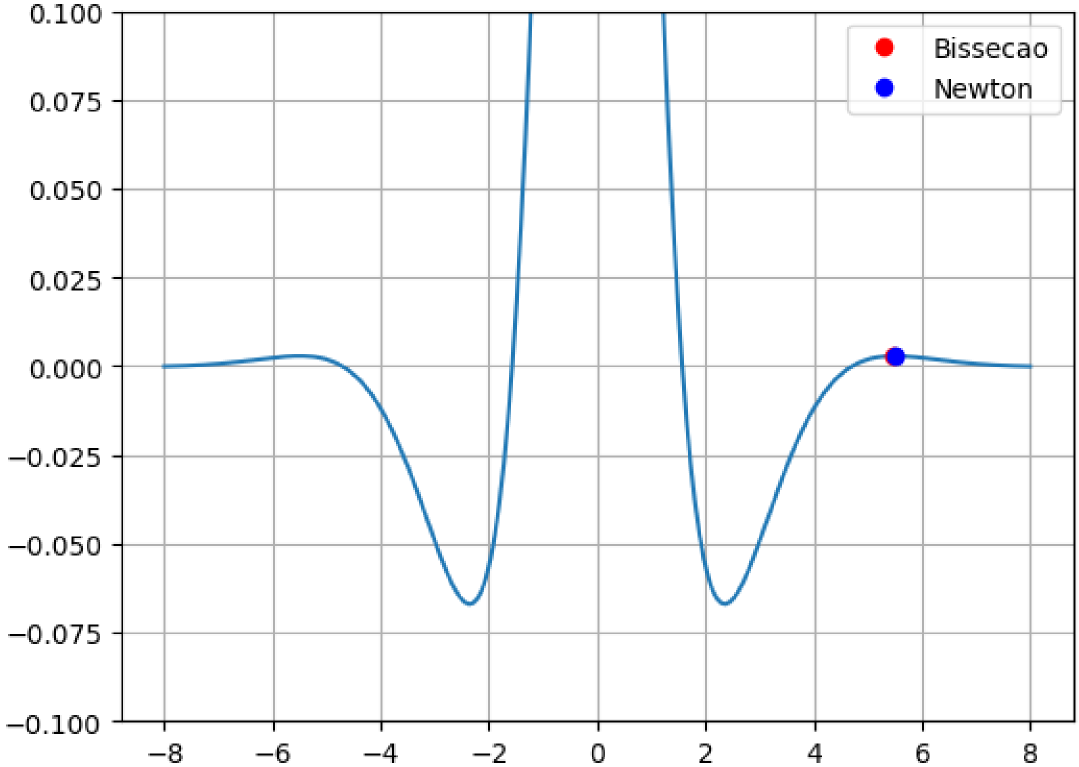
```
0.02325581
  5  5.37500000  5.50000000  5.43750000   0.00037068       0.06250000
0.01149425
  6  5.43750000  5.50000000  5.46875000   0.00017313       0.03125000
0.00571429

------ METODO DE NEWTON -------
  k          xk        xk+1     f(xk+1)    ek+1/|xk+1|
---  ----------  ----------  ----------  -------------
  0  5.00000000  5.35209354  0.00097282     0.06578613
  1  5.35209354  5.48005119  0.00010457     0.02334972
  2  5.48005119  5.49747986  0.00000178     0.00317030
```

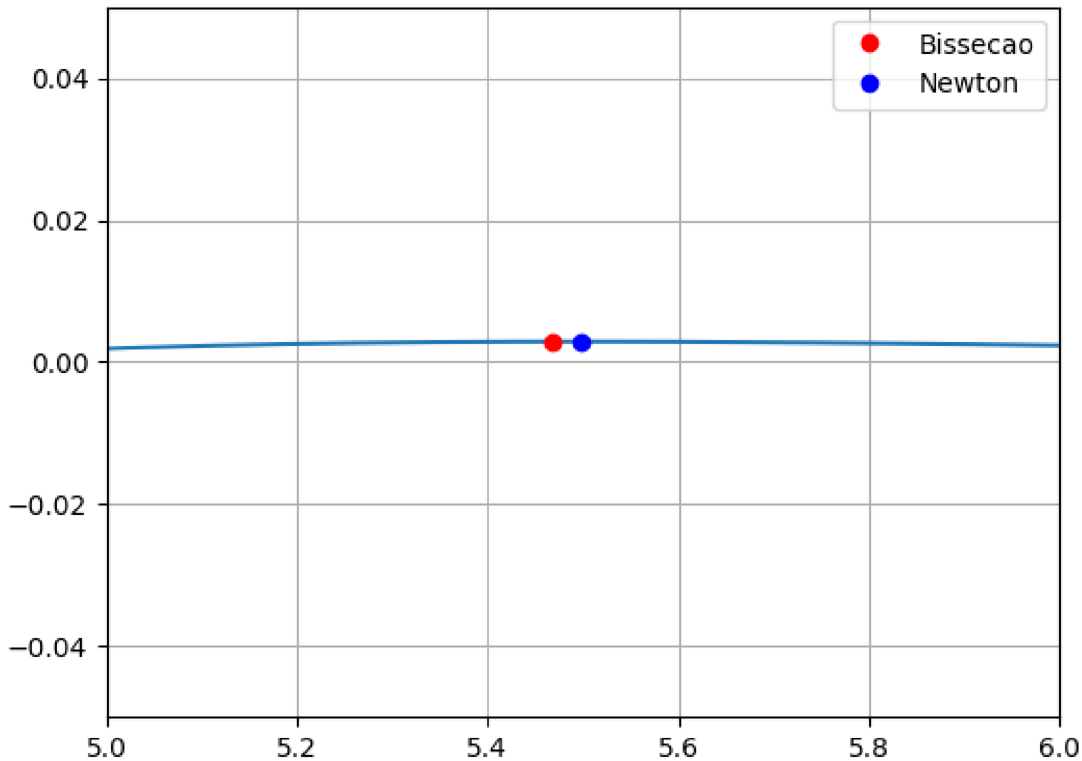
1 GABARITO

[ ]:
```python
print("Matrícula = {}".format([d1, d2, d3, d4, d5, d6, d7, d8, d9, d10]))
print("Dados = {}".format([aux, p, q]))

fvectorized = np.vectorize(f)

x = np.linspace(-8, 8, 200)
y = fvectorized(x)

plt.plot(x, y)
plt.plot(pointsbiss[-1][3], f(pointsbiss[-1][3]), 'o', label='Bissecao',␣
 ↪color='red')
plt.plot(pointsnewt[-1][2], f(pointsnewt[-1][2]), 'o', label='Newton',␣
 ↪color='blue')
plt.ylim(-0.1, 0.1)
plt.legend()
plt.grid()
plt.show()

plt.plot(x, y)
plt.xlim(5, 6)
plt.ylim(-0.05, 0.05)
plt.plot(pointsbiss[-1][3], f(pointsbiss[-1][3]), 'o', label='Bissecao',␣
 ↪color='red')
plt.plot(pointsnewt[-1][2], f(pointsnewt[-1][2]), 'o', label='Newton',␣
 ↪color='blue')
plt.legend()
plt.grid()
plt.show()

print("Primeiro ponto crítico utilizando o método de Bisseção: %.
 ↪4f"%(pointsbiss[-1][3]))
```

```
print("Primeiro ponto crítico utilizando o método de Newton:   %.
 ↪4f"%(pointsnewt[-1][2]))
```

```
Matrícula = [0, 2, 0, 2, 2, 1, 0, 0, 7, 1]
Dados = [6, 1, 1]
```






```
Primeiro ponto crítico utilizando o método de Bisseção: 5.4688
Primeiro ponto crítico utilizando o método de Newton:   5.4975
```

É bom notar que ambos os métodos obtiveram a mesma precisão em erro relativo na convergência.