**Trabalho Computacional 2**

**Entrega:** 12/09/2019

**PARTE 1**

**Exercício 1 -** *A temperatura no interior de um material com fontes de calor no seu interior é obtida pela solução da equação:* ***exp(-(½)t)\*[cosh-1(exp((1/2)t))] = sqrt(k/2)***

*Dado que k=0.67, encontre a temperatura t. Justifique a escolha do método.*

R: O método escolhido foi o Método das Secantes que, apesar da convergência ser menos eficaz em comparação ao Método de Newton, não é necessário a realização do cálculo das derivadas a cada iteração, sendo então um método mais rápido.

**Código implementado:**

| import numpy as np  import math as mt  def func(t):  return np.exp(-(1/2)\*t)\*(1/np.cosh(np.exp((1/2)\*t)))-mt.sqrt(0.67/2)  x2 = 0.0  x1 = 0.5  x0 = 1.0  cont = 0  while abs(func(x2)) >= (10\*\*(-6)):    x2 = ((x0 \* func(x1))-(x1\*func(x0)))/(func(x1)-func(x0))    cont += 1  x0 = x1  x1 = x2    print("t = ", x2)  print("Número de interações: ", cont)  print("Aproximação: ", np.exp(-(1/2)\*x2)\*(1/np.cosh(np.exp((1/2)\*x2))))  print("Resultado esperado: ", mt.sqrt(0.67/2))  **RESULTADO:**  *t = 0.12561755129157506*  *Número de interações: 5*  *Aproximação: 0.5787918449299634*  *Resultado esperado: 0.5787918451395113* |
| --- |

**Exercício 2 -** *Resolva o sistema não-linear a seguir usando o Método de Newton para sistemas não lineares:* ***f1(x1,x2)=(x1)3+3(x2)2-21=0***

***f2(x1x2)=(x1)2+2(x2)+2=0***

*Use x0=(1,-1) e critério de parada: max\_i{abs(𝚫xi)}<10^(-6) ou max\_j{fj(x1,x2)}<10^(-6) .*

**Código implementado:**

| import numpy as np  x = 1  y = -1  for i in range(100):  f1 = (x\*\*3)+3\*(y\*\*2)-21  f2 = (x\*\*2)+(2\*y)+2  J = np.array([[3\*(x\*\*2), 6\*y],[2\*x, 2]])  solucao = np.array([-f1, -f2])  s = np.linalg.solve(J, solucao)    if((abs(s[0]) < 10\*\*(-6) and abs(s[1]) < 10\*\*(-6)) or ((abs(f1) < 10\*\*(-6) and abs(f2) < 10\*\*(-6)))):  break    x = s[0] + x  y = s[1] + y  print("x: ", x)  print("y: ", y)  **RESULTADO:**  *x: 1.643038059561554*  *y: -2.3497870234302036* |
| --- |

**Exercício 3 -** *Use o método de newton para encontrar duas soluções próximas à origem para o sistema não linear:* ***x^2+x-y^2=1***

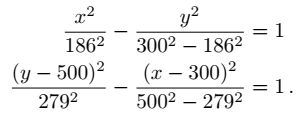
***y-sin(x^2)=0***

*Critério de parada max\_i{abs(𝚫xi)}<10^(-6) ou max\_j{fj(x1,x2)}<10^(-6).*

**Código implementado:**

| import numpy as np  x = 0  y = 0  for i in range(100):  f1 = x\*\*2+x-y\*\*2-1  f2 = y-np.sin(x\*\*2)  J = np.array([[2\*x+1, -2\*y],[-2\*x\*np.cos(x\*\*2), 1]])  solucao = np.array([-f1, -f2])  s = np.linalg.solve(J, solucao)    if((abs(s[0]) < 10\*\*(-6) and abs(s[1]) < 10\*\*(-6)) or ((abs(f1) < 10\*\*(-6) and abs(f2) < 10\*\*(-6)))):  break    x = s[0] + x  y = s[1] + y  print("x: ", x)  print("y: ", y)  **RESULTADO**:  *x: 0.725951607092812*  *y: 0.5029470803538852* |
| --- |

**Exercício 4 -** *O sistema LORAN (LOng RAnge Navigation) calcula a posição de um barco no mar usando sinais de transmissores fixos. Das diferenças de tempo de chegada de sinais emitidos, o barco obtém as diferenças de distâncias aos transmissores. Isso resulta em duas equações definidas pelas diferenças de distância de dois pontos (focos), conforme o exemplo abaixo:*



*Determine pelo menos duas soluções desse sistema usando o método de newton com precisão de 10^(-6).*

R: Para encontrar as duas soluções, foi necessário alterar os valores iniciais de x e y para que os resultados convergissem em pontos distintos. Na primeira iteração foi utilizado x = 1 e y = -1. Na segunda iteração foi utilizado x = 1000 e y = 1500. Os códigos implementados se encontram abaixo com seus respectivos resultados.

**Código implementado (I):**

| import numpy as np  x = 1  y = -1  for i in range(1000):  f1 = ((x\*\*2)/(186\*\*2))-((y\*\*2)/((300\*\*2)-(186\*\*2)))-1  f2 = (((y-500)\*\*2)/(279\*\*2))-((((x-300)\*\*2))/((500\*\*2)-(279\*\*2)))  J = np.array([[(x/17298), (-y/27702)],[((-2\*x+600)/172159), ((2\*y-1000)/77841)]])  solucao = np.array([-f1, -f2])  s = np.linalg.solve(J, solucao)    if((abs(s[0]) < 10\*\*(-6) and abs(s[1]) < 10\*\*(-6)) or ((abs(f1) < 10\*\*(-6) and abs(f2) < 10\*\*(-6)))):  break    x = s[0] + x  y = s[1] + y  print("x: ", x)  print("y: ", y)  **RESULTADO**:  *x = 392.69477220028733*  *y = 437.6703656409695* |
| --- |

**Código implementado (II):**

| import numpy as np  x = 1000  y = 1500  for i in range(1000):  f1 = ((x\*\*2)/(186\*\*2))-((y\*\*2)/((300\*\*2)-(186\*\*2)))-1  f2 = (((y-500)\*\*2)/(279\*\*2))-((((x-300)\*\*2))/((500\*\*2)-(279\*\*2)))  J = np.array([[(x/17298), (-y/27702)],[((-2\*x+600)/172159), ((2\*y-1000)/77841)]])  solucao = np.array([-f1, -f2])  s = np.linalg.solve(J, solucao)    if((abs(s[0]) < 10\*\*(-6) and abs(s[1]) < 10\*\*(-6)) or ((abs(f1) < 10\*\*(-6) and abs(f2) < 10\*\*(-6)))):  break    x = s[0] + x  y = s[1] + y  print("x: ", x)  print("y: ", y)  **RESULTADO**:  *x: 569.5652607721909*  *y: 681.2605498621871* |
| --- |

**PARTE 2**

**Exercício 1 -** *Escreva um programa, em Python para resolver um sistema de n equações lineares e n incógnitas usando o método de eliminação gaussiana. Os dados de entrada devem ser: a) o número de equações n; b) a matriz aumentada [A b]. .A saída deve ser: a) a matriz triangular superior [U]. b) o vetor solução [x].*

*Teste seu programa para resolver o seguinte sistema linear de equações Ax=b, onde*

*A = [ 1 -1 0 5; 3 -2 1 -1; 1 1 9 4; 1 -7 2 3 ]*

*x = [ x1 x2 x2 x4 ] t*

*b = [ 18 8 47 32 ] t*

**Código implementado:**

| import numpy as np  m=4  n=4  matriz=np.array([[1, -1, 0, 5],[3, -2, 1, -1],[1, 1, 9, 4],[1, -7, 2, 3]])  vetor=np.array([18, 8, 47, 32])  x=np.array([0, 0, 0, 0])  print ('Matriz dada:')  print(matriz)  type(m)  for k in range(m):  for r in range(k+1,m):  f=(matriz[r,k]/matriz[k,k])  vetor[r]=vetor[r]-(f\*vetor[k])  for c in range(0,n):  matriz[r,c]=matriz[r,c]-(f\*matriz[k,c])  #substituição para trás  x[m-1]=vetor[m-1]/matriz[m-1,m-1]  for r in range(m-2,-1,-1):  suma=0  for c in range(0,n):  suma=suma+matriz[r,c]\*x[c]  x[r]=(vetor[r]-suma)/matriz[r,r]  print ('\nA Matriz triangular superior:')  print(matriz)  print ('\nO Vetor solução:')  print(vetor)  print ('\nResultados da resolução do sistema linear: ')  print(x)  **RESULTADO**:  *Matriz dada:*  *[[ 1 -1 0 5]*  *[ 3 -2 1 -1]*  *[ 1 1 9 4]*  *[ 1 -7 2 3]]*  *A Matriz triangular superior:*  *[[ 1 -1 0 5]*  *[ 0 1 1 -16]*  *[ 0 0 7 31]*  *[ 0 0 0 -133]]*  *O Vetor solução:*  *[ 18 -46 121 -400]*  *Resultados da resolução do sistema linear:*  *[ 1 -2 4 3]* |
| --- |

**Exercício 2 -** *Uma empresa fabrica três tipos de móveis: cadeiras, mesas e armários. Cada móvel requer uma quantidade de madeira, plástico e alumínio, conforme a tabela abaixo. A empresa tem em estoque 400 unidades de madeira, 600 unidades de plástico e 1500 unidades de alumínio. Por ser final de temporada, a empresa quer usar todo o seu estoque. Para fazer isso, quantas cadeiras, mesas e armários ela precisa fabricar?*

|  | ***Madeira*** | ***Plástico*** | ***Alumínio*** |
| --- | --- | --- | --- |
| ***Cadeira*** | *1 unidade* | *1 unidade* | *2 unidades* |
| ***Mesa*** | *1 unidade* | *1 unidade* | *3 unidades* |
| ***Armário*** | *1 unidade* | *2 unidades* | *5 unidades* |

**Código implementado:**

| import numpy as np  m=3  n=3  matriz=np.array([[1, 1, 1],[2, 3, 5],[1, 1, 2]], dtype = 'f')  vetor=np.array([400, 1500, 600], dtype = 'f')  x=np.array([0, 0, 0], dtype = 'f')  print ('Matriz dada:')  print(matriz)  type(m)  for k in range(m):  for r in range(k+1,m):  f=(matriz[r,k]/matriz[k,k])  vetor[r]=vetor[r]-(f\*vetor[k])  for c in range(0,n):  matriz[r,c]=matriz[r,c]-(f\*matriz[k,c])  #substituição para trás  x[m-1]=vetor[m-1]/matriz[m-1,m-1]  for r in range(m-2,-1,-1):  suma=0  for c in range(0,n):  suma=suma+matriz[r,c]\*x[c]  x[r]=(vetor[r]-suma)/matriz[r,r]  print ('\nA Matriz triangular superior:')  print(matriz)  print ('\nO Vetor solução:')  print(vetor)  print ('\nResultados da resolução do sistema linear: ')  print(x) |
| --- |

| **RESULTADO**:  *Matriz dada:*  *[[1. 1. 1.]*  *[2. 3. 5.]*  *[1. 1. 2.]]*  *A Matriz triangular superior:*  *[[1. 1. 1.]*  *[0. 1. 3.]*  *[0. 0. 1.]]*  *O Vetor solução:*  *[400. 700. 200.]*  *Resultados da resolução do sistema linear:*  *[100. 100. 200.]* |
| --- |

Assim, verifica-se que ele precisa fabricar 100 cadeiras, 100 armários e 200 mesas.