UNIVERSIDADE FEDERAL DE SÃO CARLOS
Centro de Ciências Exatas e de Tecnologia
Departamento de Física

Trabalho 5

por

Vitor de Souza Barboza/791446 - Curso: Engenharia Física

Departamento de Física
São Carlos - SP, 20 de março de 2023

Ministrante:
Prof. Leonardo Kleber Castelano

1 Introdução

Este trabalho consiste na resolução do "Trabalho 5"da disciplina de Física Computacional 1. Todos os arquivos utilizados na prática podem ser acessados no repositório do GitHub [1]

2 Resolução

- **Questão 1**

  O código para resolução da questão 1 pode ser visto abaixo:

```
1
2  import numpy as np
3
4  n = 3
5  a = np.array([ [3., -2., 5.], [6., -9., 12.], [-5., 0.,
       2.] ])
6  b = np.array([20., 51., 1.])
7
8  # Matriz triangular superior :
9  for k in range(n - 1) :
10   for i in range(k + 1, n) :
11     m = -a[i,k] / a[k,k]
12     a[i, k:n]  = m * a[k, k:n] + a[i, k:n]
13     b[i] = m * b[k] + b[i]
14
15 x = np.zeros(n)
16 for i in range(n - 1, -1, -1) :
17   soma = 0
18   for j in range(i + 1, n) :
19     soma += a[i,j] * x[j]
20   x[i] = (b[i] - soma) / a[i,i]
21
22 print(x)
```

Listing 1: Questão 1

O código acima apresentou saída (resultado):

```
1
2  [ 1. -1.  3.]
```

Listing 2: Saída da Questão 1

- **Questão 2**

  O código para resolução da questão 2 pode ser visto abaixo:

```
1
2  import numpy as np
3
4  n = 4
5  a = np.array([ [2., -1., 3., 5.], [6., -3., 12., 11.],
       [4., -1., 10., 8.], [0., -2., -8., 10. ] ])
6  b = np.array([-7., 4., 4., -60.])
7
8  # Matriz triangular superior :
9  for k in range(n - 1) :
10   # Pivoteamento parcial - procura da linha pivo :
11   pivo = a[k,k]
12   i_pivo = k
13   for i  in range(k + 1, n) :
14     if abs(a[i,k]) > pivo :
15       pivo = a[i,k]
16       i_pivo = i
17   # Pivoteamento parcial - Troca das linhas :
18   if i_pivo != k :
19     for j in range(k, n) :
20       aux = a[k,j]
21       a[k,j] = a[i_pivo, j]
22       a[i_pivo, j] = aux
23     aux = b[k]
24     b[k] = b[i_pivo]
25     b[i_pivo] = aux
26
27   for i in range(k + 1, n) :
28     m = -a[i,k] / a[k,k]
29     a[i, k:n]  = m * a[k, k:n] + a[i, k:n]
30     b[i] = m * b[k] + b[i]
31
32 x = np.zeros(n)
33 for i in range(n - 1, -1, -1) :
34   soma = 0
35   for j in range(i + 1, n) :
36     soma += a[i,j] * x[j]
37   x[i] = (b[i] - soma) / a[i,i]
38
```

```
print(x)
```

Listing 3: Questão 2

O código acima apresentou saída (resultado):

```

[ 1. -2.  3. -4.]
```

Listing 4: Saída da Questão 2

- **Questão 3**

O código para resolução da questão 3 pode ser visto abaixo:

```

import numpy as np

def Gauss(a, b, n) :
    # Matriz triangular superior :
    for k in range(n - 1) :
        # Pivoteamento parcial - procura da linha pivo :
        pivo = a[k,k]
        i_pivo = k
        for i  in range(k + 1, n) :
            if abs(a[i,k]) > pivo :
                pivo = a[i,k]
                i_pivo = i
        # Pivoteamento parcial - Troca das linhas :
        if i_pivo != k :
            for j in range(k, n) :
                aux = a[k,j]
                a[k,j] = a[i_pivo, j]
                a[i_pivo, j] = aux
            aux = b[k]
            b[k] = b[i_pivo]
            b[i_pivo] = aux

        for i in range(k + 1, n) :
            m = -a[i,k] / a[k,k]
            a[i, k:n]  = m * a[k, k:n] + a[i, k:n] # m +
               linha do pivo + linha i - pd ser [k,:] ou
               [k,k:n]
            b[i] = m * b[k] + b[i]

```

```
    x = np.zeros(n)
    for i in range(n - 1, -1, -1) :
        soma = 0
        for j in range(i + 1, n) :
            soma += a[i,j] * x[j]
        x[i] = (b[i] - soma) / a[i,i]

    return x
```

Listing 5: Questão 3

- **Questão 4** O código para resolução da questão 3 pode ser visto abaixo:

```

import numpy as np

V = 1.
n = 7
a = np.array([ [3., 0., -1., -1., 0., 0., 0.], [0., 3.,
   0., -1., -1., 0., 0.], [1., 1., 0., -4., 0., 1., 1.],
   [0., 0., 1., 1., 0., -3., 0.], [0., 0., 0., 1., 1.,
   0., -3.], [-1., 0., 2., 0., 0., -1., 0.], [0., -1.,
   0., 0., 2., 0., -1.] ])
b = np.array([V, V, 0., 0., 0., 0., 0.])
x = Gauss(a, b, n)
print(f"As tensoes Vb, Vc, Vd, Ve, Vf, Vg, Vh sao,
   respectivamente, {x}.")

i1 = (V - x[0] ) / 2
i2 = (V - x[1]) / 2
i = i1 + i2

R_equivalente = V / i

print(f"A resistencia efetiva para todas as resistencias
   iguais a 2 Ohms e a tensao inicial igual a 1 V: {round
   (R_equivalente)} Ohms.")
```

Listing 6: Questão 4

O código acima apresentou saída (resultado):

```

As tensoes Vb, Vc, Vd, Ve, Vf, Vg, Vh sao,
   respectivamente, [0.66666667 0.66666667 0.5        0.5
```

```
    0.5 0.33333333 0.33333333].
A resistencia efetiva para todas as resistencias iguais a
    2 Ohms e a tensao inicial igual a 1 V: 3 Ohms.
```

Listing 7: Saída da Questão 4

Bibliografia

[1] https://github.com/vitorsbarboza/FisComp1