Algebra Linear Computacional - Lista 04

Mayara Aragão

Exercicio 1)

$$f(x) = log(cosh(x\sqrt{gk})) - 50$$

$$g = 9.806, k = 0.00341$$

Gráfico da função utilizando a biblioteca matplotlib do python entre os pontos $x = 250$ e $x = 300$

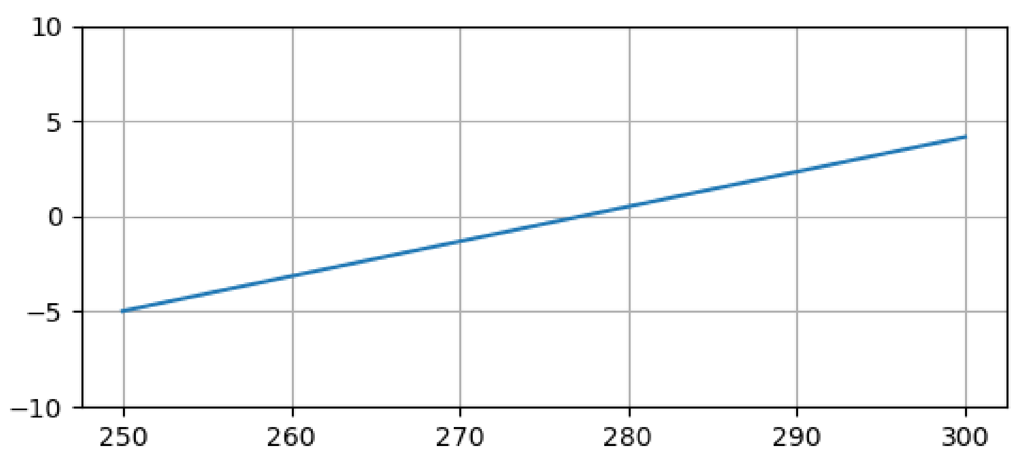
In [1]:
```
%matplotlib notebook
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:
```
def f(x):
    g = 9.806
    k = 0.00341
    y = np.log(np.cosh(x * np.sqrt(g*k)))-50
    return y

X = np.linspace(250, 300, 100)

F = np.vectorize(f)

plt.plot(X, F(X))
plt.ylim(-10, 10)
plt.grid()
plt.show()
```

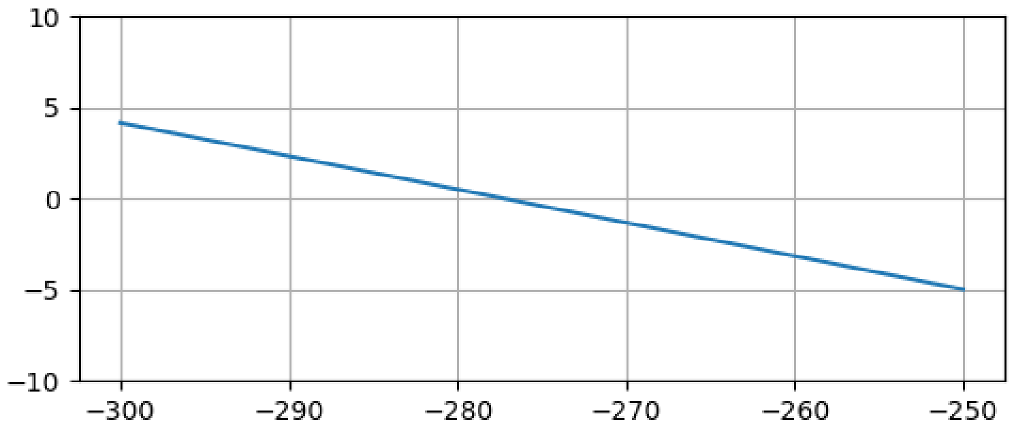


A função é simétrica em relação ao eixo y, portando, plotando o gráfico da função para $x = -300$ e $x = -250$:

In [3]:
```python
X = np.linspace(-300,-250, 100)

F = np.vectorize(f)

plt.plot(X, F(X))
plt.ylim(-10, 10)
plt.grid()
plt.show()
```



Método da Bisseção:

In [4]:
```python
tol = 0.0001
```

In [5]:
```python
def ordena(a, b):
    if a < b:
        return a, b
    return b, a
```

In [6]:
```python
def bissecao(a, b, tol):
    a, b = ordena(a, b)
    err = abs(b-a)
    aux = a
    it = 0

    while err >= tol:
        x = (a+b)/2.0
        print("ITER ", it, "|a = %.5f" % a, "|b = %.5f" % b, "|x = %.5f" % x,
              "|f(x) = %.4f" % f(x), "| Err = %.4f" % err)
        if f(x)*f(a) < 0:
            b = x
        elif f(x)*f(a) > 0:
            a = x
        else:
            return x
        err = abs(x-aux)
        aux = x
        it += 1
    print("ITER ", it, "|a = %.5f" % a, "|b = %.5f" % b, "|x = %.5f" % x,
          "|f(x) = %.4f" % f(x), "| Err = %.4f" % err)
    return x
```

In [7]:
```python
a = 250
b = 300
x = bissecao(a, b, tol)
print("\nA raiz exata é x= %.5f" % x)
```

```
ITER  2 |a = 275.00000 |b = 287.50000 |x = 281.25000 |f(x) = 0.7368 | Err = 12.5000
ITER  3 |a = 275.00000 |b = 281.25000 |x = 278.12500 |f(x) = 0.1653 | Err = 6.2500
ITER  4 |a = 275.00000 |b = 278.12500 |x = 276.56250 |f(x) = -0.1204 | Err = 3.1250
ITER  5 |a = 276.56250 |b = 278.12500 |x = 277.34375 |f(x) = 0.0224 | Err = 1.5625
ITER  6 |a = 276.56250 |b = 277.34375 |x = 276.95312 |f(x) = -0.0490 | Err = 0.7812
ITER  7 |a = 276.95312 |b = 277.34375 |x = 277.14844 |f(x) = -0.0133 | Err = 0.3906
ITER  8 |a = 277.14844 |b = 277.34375 |x = 277.24609 |f(x) = 0.0046 | Err = 0.1953
ITER  9 |a = 277.14844 |b = 277.24609 |x = 277.19727 |f(x) = -0.0043 | Err = 0.0977
ITER  10 |a = 277.19727 |b = 277.24609 |x = 277.22168 |f(x) = 0.0001 | Err = 0.0488
ITER  11 |a = 277.19727 |b = 277.22168 |x = 277.20947 |f(x) = -0.0021 | Err = 0.024
4
ITER  12 |a = 277.20947 |b = 277.22168 |x = 277.21558 |f(x) = -0.0010 | Err = 0.012
2
ITER  13 |a = 277.21558 |b = 277.22168 |x = 277.21863 |f(x) = -0.0004 | Err = 0.006
1
ITER  14 |a = 277.21863 |b = 277.22168 |x = 277.22015 |f(x) = -0.0002 | Err = 0.003
1
ITER  15 |a = 277.22015 |b = 277.22168 |x = 277.22092 |f(x) = -0.0000 | Err = 0.001
5
```

Para raiz negativa:

In [8]:
```
a = -250
b = -300
x = bissecao(a, b, tol)
print("\nA raiz exata é x= %.5f" % x)
```

```
ITER  0 |a = -300.00000 |b = -250.00000 |x = -275.00000 |f(x) = -0.4061 | Err = 50.00
00
ITER  1 |a = -300.00000 |b = -275.00000 |x = -287.50000 |f(x) = 1.8796 | Err = 25.000
0
ITER  2 |a = -287.50000 |b = -275.00000 |x = -281.25000 |f(x) = 0.7368 | Err = 12.500
0
ITER  3 |a = -281.25000 |b = -275.00000 |x = -278.12500 |f(x) = 0.1653 | Err = 6.2500
ITER  4 |a = -278.12500 |b = -275.00000 |x = -276.56250 |f(x) = -0.1204 | Err = 3.125
0
ITER  5 |a = -278.12500 |b = -276.56250 |x = -277.34375 |f(x) = 0.0224 | Err = 1.5625
ITER  6 |a = -277.34375 |b = -276.56250 |x = -276.95312 |f(x) = -0.0490 | Err = 0.781
2
ITER  7 |a = -277.34375 |b = -276.95312 |x = -277.14844 |f(x) = -0.0133 | Err = 0.390
6
ITER  8 |a = -277.34375 |b = -277.14844 |x = -277.24609 |f(x) = 0.0046 | Err = 0.1953
ITER  9 |a = -277.24609 |b = -277.14844 |x = -277.19727 |f(x) = -0.0043 | Err = 0.097
7
ITER  10 |a = -277.24609 |b = -277.19727 |x = -277.22168 |f(x) = 0.0001 | Err = 0.048
8
ITER  11 |a = -277.22168 |b = -277.19727 |x = -277.20947 |f(x) = -0.0021 | Err = 0.02
44
ITER  12 |a = -277.22168 |b = -277.20947 |x = -277.21558 |f(x) = -0.0010 | Err = 0.01
22
ITER  13 |a = -277.22168 |b = -277.21558 |x = -277.21863 |f(x) = -0.0004 | Err = 0.00
61
ITER  14 |a = -277.22168 |b = -277.21863 |x = -277.22015 |f(x) = -0.0002 | Err = 0.00
31
ITER  15 |a = -277.22168 |b = -277.22015 |x = -277.22092 |f(x) = -0.0000 | Err = 0.00
15
ITER  16 |a = -277.22168 |b = -277.22092 |x = -277.22130 |f(x) = 0.0001 | Err = 0.000
8
ITER  17 |a = -277.22130 |b = -277.22092 |x = -277.22111 |f(x) = 0.0000 | Err = 0.000
4
ITER  18 |a = -277.22111 |b = -277.22092 |x = -277.22101 |f(x) = 0.0000 | Err = 0.000
2
ITER  19 |a = -277.22101 |b = -277.22092 |x = -277.22101 |f(x) = 0.0000 | Err = 0.000
1

A raiz exata é x= -277.22101
```

Método de Newton original:

Definindo a derivada de $f(x)$

In [9]:
```
def fder(x):
    cte = (9.806*0.00341)**0.5
    return (cte*np.sinh(x * cte)/np.cosh(x * cte))
```

In [10]:
```
def newton(x0, tol, it_max):
    it = 0
    err = 10

    while (err >= tol) and (it < it_max):
        x = x0 - f(x0)/fder(x0)
        err = abs(x-x0)
        print("ITER ", it, "|x = %.3f" % x0, "|f(x) = %.4f" %
              f(x0), "| f'(x) = %.4f" % fder(x0), "| Err = %.5f" % err)
        x0 = x
        it += 1

    return (x, it)
```

In [11]:
```
x0 = 10
it_max = 100
tol = 0.0001
```

In [12]:
```
x, i = newton(x0, tol, it_max)
if i == it_max:
    print("O método não convergiu")
print("\nRaiz encontrada x= %.5f" % x)
```

```
ITER  0 |x = 10.000 |f(x) = -48.8391 | f'(x) = 0.1737 | Err = 281.23015
ITER  1 |x = 291.230 |f(x) = 2.5617 | f'(x) = 0.1829 | Err = 14.00916
ITER  2 |x = 277.221 |f(x) = 0.0000 | f'(x) = 0.1829 | Err = 0.00000

Raiz encontrada x= 277.22100
```

Para raiz negativa:

In [13]:
```
x0 = -10
x, i = newton(x0, tol, it_max)
if i == it_max:
    print("O método não convergiu")
print("\nRaiz encontrada x= %.5f" % x)
```

```
ITER  0 |x = -10.000 |f(x) = -48.8391 | f'(x) = -0.1737 | Err = 281.23015
ITER  1 |x = -291.230 |f(x) = 2.5617 | f'(x) = -0.1829 | Err = 14.00916
ITER  2 |x = -277.221 |f(x) = 0.0000 | f'(x) = -0.1829 | Err = 0.00000

Raiz encontrada x= -277.22100
```

Método da Secante:

In [14]:
```python
def secante(x0, tol, it_max):
    delta = 0.001
    x1 = x0 + delta
    fa = f(x0)
    it = 0
    err = 10
    while (err >= tol) and (it < it_max):
        fi = f(x1)
        x2 = x1 - (fi*(x1-x0)/(fi-fa))
        err = abs(x2-x1)
        print("ITER ", it, "|x", it, " = %.3f" % x0, "|x", it+1, " = %.3f" % x1,
              "|x", it+2, " = %.4f" %x2, "| Err = %.5f" % err)
        x0 = x1
        x1 = x2

        fa = fi
        it += 1

    return (x1, it)
```

In [15]:
```python
x0 = 10
x, i = secante(x0, tol, it_max)
if i == it_max:
    print("O método não convergiu")
print("\nRaiz encontrada x= %.5f" % x)
```

```
ITER  0 |x 0  = 10.000 |x 1  = 10.001 |x 2  = 291.2275 | Err = 281.22650
ITER  1 |x 1  = 10.001 |x 2  = 291.227 |x 3  = 277.2141 | Err = 14.01344
ITER  2 |x 2  = 291.227 |x 3  = 277.214 |x 4  = 277.2210 | Err = 0.00694
ITER  3 |x 3  = 277.214 |x 4  = 277.221 |x 5  = 277.2210 | Err = 0.00000

Raiz encontrada x= 277.22100
```

Para raiz negativa:

In [16]:
```python
x0 = -10
x, i = secante(x0, tol, it_max)
if i == it_max:
    print("O método não convergiu")
print("\nRaiz encontrada x= %.5f" % x)
```

```
ITER  0 |x 0  = -10.000 |x 1  = -9.999 |x 2  = -291.2328 | Err = 281.23381
ITER  1 |x 1  = -9.999 |x 2  = -291.233 |x 3  = -277.2140 | Err = 14.01876
ITER  2 |x 2  = -291.233 |x 3  = -277.214 |x 4  = -277.2210 | Err = 0.00695
ITER  3 |x 3  = -277.214 |x 4  = -277.221 |x 5  = -277.2210 | Err = 0.00000

Raiz encontrada x= -277.22100
```

Método da Interpolação inversa:

In [17]:
```python
x = [200, 250, 300]
```

In [18]:
```
def maior_indice(y):
    if y[0] > y[1]:
        maior = 0
    else:
        maior = 1
    if maior > y[2]:
        return maior
    return 2
```

In [19]:
```
def interpolacao_inversa(x):
    it = 0
    err = 10
    x = sorted(x)
    x0 = 10**10
    y = [0, 0, 0]
    y[0] = f(x[0])
    y[1] = f(x[1])
    y[2] = f(x[2])

    while (err >= tol) and (it < it_max):
        phi0 = ((y[1]*y[2])/((y[0]-y[1])*(y[0]-y[2])))
        phi1 = ((y[0]*y[2])/((y[1]-y[0])*(y[1]-y[2])))
        phi2 = ((y[0]*y[1])/((y[2]-y[0])*(y[2]-y[1])))
        aux = phi0*x[0] + phi1*x[1] + phi2*x[2]

        err = abs(aux - x0)

        print("ITER ", it, "|x", it, " = %.3f" % x[0],
              "|x", it+1, " = %.3f" % x[1], "|x", it+2, " = %.3f" %
              x[2], "|y", it, " = %.3f" % y[0],
              "|y", it+1, " = %.3f" % y[1], "|y", it+2, " = %.4f" %
              y[2], "| x* = %.3f" % aux, "\n        Err = %.5f" % err)

        i = maior_indice(y)
        x[i] = aux
        y[i] = f(aux)
        x = sorted(x)
        y = sorted(y)
        x0 = aux
        it += 1

    return (aux, it)
```

In [20]:
```
aux, i = interpolacao_inversa(x)
if i == it_max:
    print("O método não convergiu")
print("\nRaiz encontrada x= %.5f" % aux)
```

```
ITER  0 |x 0  = 200.000 |x 1  = 250.000 |x 2  = 300.000 |y 0  = -14.121 |y 1  = -4.97
8 |y 2  = 4.1654 | x* = 277.221
        Err = 9999999722.77900
ITER  1 |x 1  = 200.000 |x 2  = 250.000 |x 3  = 277.221 |y 1  = -14.121 |y 2  = -4.97
8 |y 3  = 0.0000 | x* = 277.221
        Err = 0.00000

Raiz encontrada x= 277.22100
```

Para raiz negativa:

In [21]:
```python
x = [-300, -285, -260]
```

In [22]:
```python
aux, i = interpolacao_inversa(x)
if i == it_max:
    print("O método não convergiu")
print("\nRaiz encontrada x= %.5f" % aux)
```

```
ITER  0 |x 0  = -300.000 |x 1  = -285.000 |x 2  = -260.000 |y 0  = 4.165 |y 1  = 1.42
2 |y 2  = -3.1491 | x* = -277.221
        Err = 10000000277.22100
ITER  1 |x 1  = -285.000 |x 2  = -277.221 |x 3  = -260.000 |y 1  = -3.149 |y 2  = 0.0
00 |y 3  = 1.4225 | x* = -277.221
        Err = 0.00000

Raiz encontrada x= -277.22100
```

Exercício2)

$$f(x) = 4cos(x) - e^{2x}$$

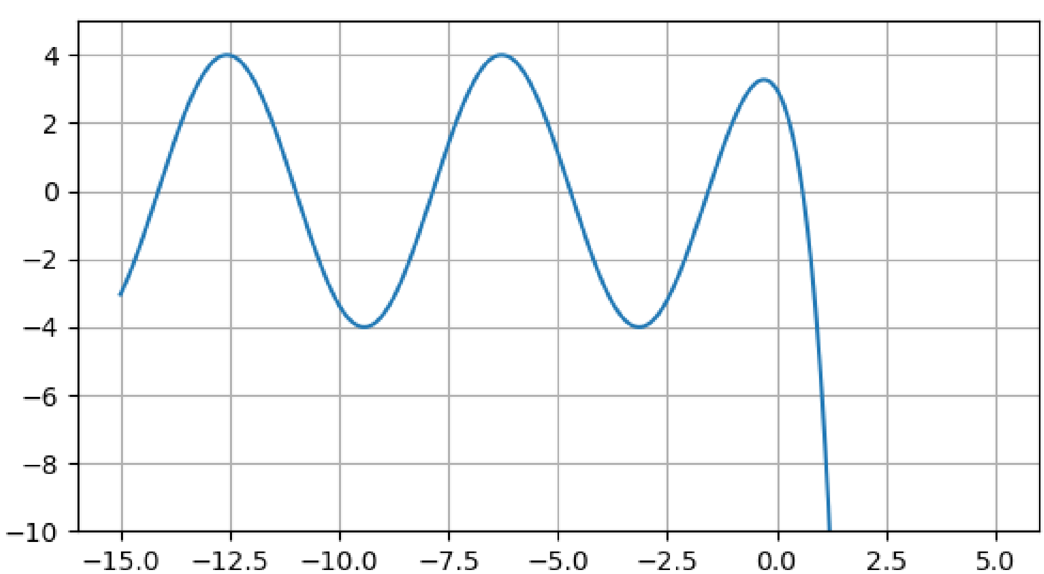
Gráfico da função entre os pontos $x = -15$ e $x = 5$

In [23]:
```python
def f(x): return 4*np.cos(x) - np.exp(2*x)
```

In [24]:
```python
X = np.linspace(-15, 5, 300)

F = np.vectorize(f)

plt.plot(X, F(X))
plt.ylim(-10, 5)
plt.grid()
plt.show()
```



Método da Bisseção:

Para $x > 0$, a função possui apenas uma raiz e para $x < 0$ a função tem comportamento oscilante. Definindo condições iniciais para $x > 0$

In [25]:
```python
x = bissecao(0, 2, tol)
print("\nA raiz exata é x= %.5f" % x)
```

```
ITER  0 |a = 0.00000 |b = 2.00000 |x = 1.00000 |f(x) = -5.2278 | Err = 2.0000
ITER  1 |a = 0.00000 |b = 1.00000 |x = 0.50000 |f(x) = 0.7920 | Err = 1.0000
ITER  2 |a = 0.50000 |b = 1.00000 |x = 0.75000 |f(x) = -1.5549 | Err = 0.5000
ITER  3 |a = 0.50000 |b = 0.75000 |x = 0.62500 |f(x) = -0.2465 | Err = 0.2500
ITER  4 |a = 0.50000 |b = 0.62500 |x = 0.56250 |f(x) = 0.3035 | Err = 0.1250
ITER  5 |a = 0.56250 |b = 0.62500 |x = 0.59375 |f(x) = 0.0365 | Err = 0.0625
ITER  6 |a = 0.59375 |b = 0.62500 |x = 0.60938 |f(x) = -0.1029 | Err = 0.0312
ITER  7 |a = 0.59375 |b = 0.60938 |x = 0.60156 |f(x) = -0.0327 | Err = 0.0156
ITER  8 |a = 0.59375 |b = 0.60156 |x = 0.59766 |f(x) = 0.0020 | Err = 0.0078
ITER  9 |a = 0.59766 |b = 0.60156 |x = 0.59961 |f(x) = -0.0153 | Err = 0.0039
ITER  10 |a = 0.59766 |b = 0.59961 |x = 0.59863 |f(x) = -0.0066 | Err = 0.0020
ITER  11 |a = 0.59766 |b = 0.59863 |x = 0.59814 |f(x) = -0.0023 | Err = 0.0010
ITER  12 |a = 0.59766 |b = 0.59814 |x = 0.59790 |f(x) = -0.0001 | Err = 0.0005
ITER  13 |a = 0.59766 |b = 0.59790 |x = 0.59778 |f(x) = 0.0010 | Err = 0.0002
ITER  14 |a = 0.59778 |b = 0.59790 |x = 0.59784 |f(x) = 0.0004 | Err = 0.0001
ITER  15 |a = 0.59784 |b = 0.59790 |x = 0.59784 |f(x) = 0.0004 | Err = 0.0001

A raiz exata é x= 0.59784
```

Definindo condições iniciais para $x < 0$

In [26]:
```python
x = bissecao(-2, 0, tol)
print("\nA raiz exata é x= %.5f" % x)
```

```
ITER  0 |a = -2.00000 |b = 0.00000 |x = -1.00000 |f(x) = 2.0259 | Err = 2.0000
ITER  1 |a = -2.00000 |b = -1.00000 |x = -1.50000 |f(x) = 0.2332 | Err = 1.0000
ITER  2 |a = -2.00000 |b = -1.50000 |x = -1.75000 |f(x) = -0.7432 | Err = 0.5000
ITER  3 |a = -1.75000 |b = -1.50000 |x = -1.62500 |f(x) = -0.2555 | Err = 0.2500
ITER  4 |a = -1.62500 |b = -1.50000 |x = -1.56250 |f(x) = -0.0108 | Err = 0.1250
ITER  5 |a = -1.56250 |b = -1.50000 |x = -1.53125 |f(x) = 0.1114 | Err = 0.0625
ITER  6 |a = -1.56250 |b = -1.53125 |x = -1.54688 |f(x) = 0.0503 | Err = 0.0312
ITER  7 |a = -1.56250 |b = -1.54688 |x = -1.55469 |f(x) = 0.0198 | Err = 0.0156
ITER  8 |a = -1.56250 |b = -1.55469 |x = -1.55859 |f(x) = 0.0045 | Err = 0.0078
ITER  9 |a = -1.56250 |b = -1.55859 |x = -1.56055 |f(x) = -0.0031 | Err = 0.0039
ITER  10 |a = -1.56055 |b = -1.55859 |x = -1.55957 |f(x) = 0.0007 | Err = 0.0020
ITER  11 |a = -1.56055 |b = -1.55957 |x = -1.56006 |f(x) = -0.0012 | Err = 0.0010
ITER  12 |a = -1.56006 |b = -1.55957 |x = -1.55981 |f(x) = -0.0002 | Err = 0.0005
ITER  13 |a = -1.55981 |b = -1.55957 |x = -1.55969 |f(x) = 0.0002 | Err = 0.0002
ITER  14 |a = -1.55981 |b = -1.55969 |x = -1.55975 |f(x) = -0.0000 | Err = 0.0001
ITER  15 |a = -1.55975 |b = -1.55969 |x = -1.55975 |f(x) = -0.0000 | Err = 0.0001

A raiz exata é x= -1.55975
```

Método de Newton original:

Definindo a derivada da função $f(x)$

In [27]:
```python
def fder(x): return (-4*np.sin(x) - np.exp(2*x)*2)
```

In [28]:
```python
x0 = 1
x, i = newton(x0, tol, it_max)
if i == it_max:
    print("O método não convergiu")
print("\nRaiz encontrada x= %.5f" % x)
```

```
ITER  0 |x = 1.000 |f(x) = -5.2278 | f'(x) = -18.1440 | Err = 0.28813
ITER  1 |x = 0.712 |f(x) = -1.1240 | f'(x) = -10.9182 | Err = 0.10295
ITER  2 |x = 0.609 |f(x) = -0.0988 | f'(x) = -9.0476 | Err = 0.01092
ITER  3 |x = 0.598 |f(x) = -0.0010 | f'(x) = -8.8657 | Err = 0.00011
ITER  4 |x = 0.598 |f(x) = -0.0000 | f'(x) = -8.8638 | Err = 0.00000

Raiz encontrada x= 0.59789
```

Comprovando característica oscilatória da função para $x < 0$

In [29]:
```python
x, i = newton(-2, tol, it_max)
if i == it_max:
    print("O método não convergiu")
print("\nRaiz encontrada x= %.5f" % x)
```

```
ITER  0 |x = -2.000 |f(x) = -1.6829 | f'(x) = 3.6006 | Err = 0.46740
ITER  1 |x = -1.533 |f(x) = 0.1061 | f'(x) = 3.9038 | Err = 0.02718
ITER  2 |x = -1.560 |f(x) = -0.0001 | f'(x) = 3.9114 | Err = 0.00003

Raiz encontrada x= -1.55975
```

In [30]:
```python
x, i = newton(-4, tol, it_max)
if i == it_max:
    print("O método não convergiu")
print("\nRaiz encontrada x= %.5f" % x)
```

```
ITER  0 |x = -4.000 |f(x) = -2.6149 | f'(x) = -3.0279 | Err = 0.86361
ITER  1 |x = -4.864 |f(x) = 0.6025 | f'(x) = -3.9545 | Err = 0.15237
ITER  2 |x = -4.711 |f(x) = -0.0047 | f'(x) = -4.0002 | Err = 0.00116
ITER  3 |x = -4.712 |f(x) = 0.0000 | f'(x) = -4.0002 | Err = 0.00000

Raiz encontrada x= -4.71241
```

Método da Secante:

In [31]:
```python
x, i = secante(x0, tol, it_max)
if i == it_max:
    print("O método não convergiu")
print("Raiz encontrada x= %.5f" % x)
```

```
ITER  0 |x 0  = 1.000 |x 1  = 1.001 |x 2  = 0.7121 | Err = 0.28888
ITER  1 |x 1  = 1.001 |x 2  = 0.712 |x 3  = 0.6331 | Err = 0.07902
ITER  2 |x 2  = 0.712 |x 3  = 0.633 |x 4  = 0.6014 | Err = 0.03169
ITER  3 |x 3  = 0.633 |x 4  = 0.601 |x 5  = 0.5980 | Err = 0.00341
ITER  4 |x 4  = 0.601 |x 5  = 0.598 |x 6  = 0.5979 | Err = 0.00011
ITER  5 |x 5  = 0.598 |x 6  = 0.598 |x 7  = 0.5979 | Err = 0.00000
Raiz encontrada x= 0.59789
```

In [32]:
```
x, i = secante(-2, tol, it_max)
if i == it_max:
    print("O método não convergiu")
print("Raiz encontrada x= %.5f" % x)
```

```
ITER  0 |x 0  = -2.000 |x 1  = -1.999 |x 2  = -1.5327 | Err = 0.46630
ITER  1 |x 1  = -1.999 |x 2  = -1.533 |x 3  = -1.5603 | Err = 0.02761
ITER  2 |x 2  = -1.533 |x 3  = -1.560 |x 4  = -1.5598 | Err = 0.00056
ITER  3 |x 3  = -1.560 |x 4  = -1.560 |x 5  = -1.5598 | Err = 0.00000
Raiz encontrada x= -1.55975
```

Método da Interpolação inversa:

In [33]:
```
x = [0.2, 0.46, 0.77]
```

In [34]:
```
aux, i = interpolacao_inversa(x)
if i == it_max:
    print("O método não convergiu")
print("Raiz encontrada x= %.5f" % aux)
```

```
ITER  0 |x 0  = 0.200 |x 1  = 0.460 |x 2  = 0.770 |y 0  = 2.428 |y 1  = 1.075 |y 2
= -1.7929 | x* = 0.615
        Err = 9999999999.38546
ITER  1 |x 1  = 0.460 |x 2  = 0.615 |x 3  = 0.770 |y 1  = -1.793 |y 2  = -0.150 |y
3  = 1.0749 | x* = 0.632
        Err = 0.01718
ITER  2 |x 2  = 0.460 |x 3  = 0.615 |x 4  = 0.632 |y 2  = -1.793 |y 3  = -0.310 |y
4  = -0.1499 | x* = 0.648
        Err = 0.01624
ITER  3 |x 3  = 0.460 |x 4  = 0.632 |x 5  = 0.648 |y 3  = -1.793 |y 4  = -0.465 |y
5  = -0.1499 | x* = 0.652
        Err = 0.00442
ITER  4 |x 4  = 0.460 |x 5  = 0.648 |x 6  = 0.652 |y 4  = -1.793 |y 5  = -0.508 |y
6  = -0.1499 | x* = 0.648
        Err = 0.00436
ITER  5 |x 5  = 0.460 |x 6  = 0.648 |x 7  = 0.652 |y 5  = -1.793 |y 6  = -0.466 |y
7  = -0.1499 | x* = 0.649
        Err = 0.00100
ITER  6 |x 6  = 0.460 |x 7  = 0.649 |x 8  = 0.652 |y 6  = -1.793 |y 7  = -0.475 |y
8  = -0.1499 | x* = 0.648
        Err = 0.00087
ITER  7 |x 7  = 0.460 |x 8  = 0.648 |x 9  = 0.652 |y 7  = -1.793 |y 8  = -0.467 |y
9  = -0.1499 | x* = 0.649
        Err = 0.00075
ITER  8 |x 8  = 0.460 |x 9  = 0.649 |x 10  = 0.652 |y 8  = -1.793 |y 9  = -0.474 |y
10  = -0.1499 | x* = 0.648
        Err = 0.00065
ITER  9 |x 9  = 0.460 |x 10  = 0.648 |x 11  = 0.652 |y 9  = -1.793 |y 10  = -0.468
|y 11  = -0.1499 | x* = 0.649
        Err = 0.00056
ITER  10 |x 10  = 0.460 |x 11  = 0.649 |x 12  = 0.652 |y 10  = -1.793 |y 11  = -0.4
73 |y 12  = -0.1499 | x* = 0.648
        Err = 0.00049
ITER  11 |x 11  = 0.460 |x 12  = 0.648 |x 13  = 0.652 |y 11  = -1.793 |y 12  = -0.4
69 |y 13  = -0.1499 | x* = 0.649
        Err = 0.00042
ITER  12 |x 12  = 0.460 |x 13  = 0.649 |x 14  = 0.652 |y 12  = -1.793 |y 13  = -0.4
73 |y 14  = -0.1499 | x* = 0.648
        Err = 0.00036
ITER  13 |x 13  = 0.460 |x 14  = 0.648 |x 15  = 0.652 |y 13  = -1.793 |y 14  = -0.4
69 |y 15  = -0.1499 | x* = 0.649
        Err = 0.00031
ITER  14 |x 14  = 0.460 |x 15  = 0.649 |x 16  = 0.652 |y 14  = -1.793 |y 15  = -0.4
72 |y 16  = -0.1499 | x* = 0.648
        Err = 0.00027
ITER  15 |x 15  = 0.460 |x 16  = 0.648 |x 17  = 0.652 |y 15  = -1.793 |y 16  = -0.4
70 |y 17  = -0.1499 | x* = 0.649
        Err = 0.00023
ITER  16 |x 16  = 0.460 |x 17  = 0.649 |x 18  = 0.652 |y 16  = -1.793 |y 17  = -0.4
72 |y 18  = -0.1499 | x* = 0.648
        Err = 0.00020
ITER  17 |x 17  = 0.460 |x 18  = 0.648 |x 19  = 0.652 |y 17  = -1.793 |y 18  = -0.4
70 |y 19  = -0.1499 | x* = 0.649
        Err = 0.00018
ITER  18 |x 18  = 0.460 |x 19  = 0.649 |x 20  = 0.652 |y 18  = -1.793 |y 19  = -0.4
72 |y 20  = -0.1499 | x* = 0.648
```

```
        Err = 0.00015
ITER  19 |x 19  = 0.460 |x 20  = 0.648 |x 21  = 0.652 |y 19  = -1.793 |y 20  = -0.4
70 |y 21  = -0.1499 | x* = 0.649
        Err = 0.00013
ITER  20 |x 20  = 0.460 |x 21  = 0.649 |x 22  = 0.652 |y 20  = -1.793 |y 21  = -0.4
72 |y 22  = -0.1499 | x* = 0.649
        Err = 0.00011
ITER  21 |x 21  = 0.460 |x 22  = 0.649 |x 23  = 0.652 |y 21  = -1.793 |y 22  = -0.4
70 |y 23  = -0.1499 | x* = 0.649
        Err = 0.00010
Raiz encontrada x= 0.64860
```

In [35]:
```
x = [-3, -2, 0.5]
```

In [36]:
```
aux, i = interpolacao_inversa(x)
if i == it_max:
    print("O método não convergiu")
print("Raiz encontrada x= %.5f" % aux)
```

```
ITER  0 |x 0  = -3.000 |x 1  = -2.000 |x 2  = 0.500 |y 0  = -3.962 |y 1  = -1.683 |y
2  = 0.7920 | x* = -0.460
        Err = 10000000000.46027
ITER  1 |x 1  = -3.000 |x 2  = -0.460 |x 3  = 0.500 |y 1  = -3.962 |y 2  = 0.792 |y 3
= 3.1854 | x* = -0.825
        Err = 0.36471
ITER  2 |x 2  = -3.000 |x 3  = -0.825 |x 4  = -0.460 |y 2  = -3.962 |y 3  = 0.792 |y
4  = 2.5222 | x* = -1.068
        Err = 0.24295
ITER  3 |x 3  = -3.000 |x 4  = -1.068 |x 5  = -0.825 |y 3  = -3.962 |y 4  = 0.792 |y
5  = 1.8096 | x* = -1.299
        Err = 0.23073
ITER  4 |x 4  = -3.000 |x 5  = -1.299 |x 6  = -1.068 |y 4  = -3.962 |y 5  = 0.792 |y
6  = 1.0007 | x* = -2.055
        Err = 0.75653
ITER  5 |x 5  = -3.000 |x 6  = -2.055 |x 7  = -1.299 |y 5  = -3.962 |y 6  = -1.879 |y
7  = 0.7920 | x* = -1.470
        Err = 0.58551
ITER  6 |x 6  = -3.000 |x 7  = -1.470 |x 8  = -1.299 |y 6  = -3.962 |y 7  = 0.351 |y
8  = 0.7920 | x* = -1.604
        Err = 0.13408
ITER  7 |x 7  = -3.000 |x 8  = -1.604 |x 9  = -1.299 |y 7  = -3.962 |y 8  = -0.172 |y
9  = 0.7920 | x* = -1.548
        Err = 0.05601
ITER  8 |x 8  = -3.000 |x 9  = -1.548 |x 10  = -1.299 |y 8  = -3.962 |y 9  = 0.047 |y
10  = 0.7920 | x* = -1.564
        Err = 0.01589
ITER  9 |x 9  = -3.000 |x 10  = -1.564 |x 11  = -1.299 |y 9  = -3.962 |y 10  = -0.015
|y 11  = 0.7920 | x* = -1.559
        Err = 0.00510
ITER  10 |x 10  = -3.000 |x 11  = -1.559 |x 12  = -1.299 |y 10  = -3.962 |y 11  = 0.0
05 |y 12  = 0.7920 | x* = -1.560
        Err = 0.00157
ITER  11 |x 11  = -3.000 |x 12  = -1.560 |x 13  = -1.299 |y 11  = -3.962 |y 12  = -0.
001 |y 13  = 0.7920 | x* = -1.560
        Err = 0.00049
ITER  12 |x 12  = -3.000 |x 13  = -1.560 |x 14  = -1.299 |y 12  = -3.962 |y 13  = 0.0
00 |y 14  = 0.7920 | x* = -1.560
        Err = 0.00015
ITER  13 |x 13  = -3.000 |x 14  = -1.560 |x 15  = -1.299 |y 13  = -3.962 |y 14  = -0.
000 |y 15  = 0.7920 | x* = -1.560
        Err = 0.00005
Raiz encontrada x= -1.55974
```

O resultado para raiz positiva é proximo ao real, mas não exato, após algumas tentativas de vetor inicial. Acredito que isso se deve as características da função, que tende a ir a $-\infty$ para $x > 0$

Exercício 3)

$$16x^4 + 16y^4 + z^4 = 16$$

$$x^2 + y^2 + z^2 = 3$$

$$x^3 - y + z = 1$$

Método de Newton:

In [37]:
```
def f(V):
    x, y, z = V
    S = np.array([16*x**4+16*y**4+z**4-16,
                  x**2+y**2+z**2-3,
                  x**3-y+z-1
                  ])
    return S
```

In [38]:
```
def j(V):
    x, y, z = V
    J = np.array([[64*x**3, 64*y**3, 4*z**3],
                  [2*x, 2*y, 2*z],
                  [3*x**2, -1, 1]
                  ])
    return J
```

In [39]:
```
def newton(V, tol, it_max):
    it = 0
    err = 10

    while (err >= tol) and (it < it_max):
        J = j(V)
        F = f(V)
        inv_J = np.linalg.inv(J)

        delta = - inv_J.dot(F)
        V = V + delta
        err = np.linalg.norm(delta)/np.linalg.norm(V)

        X = np.round(V, 3)
        print("ITER ", it, "|X = ", X, " | Err = %.5f" % err)

        it += 1

    return (V, it)
```

Definindo as condições iniciais

In [40]:
```
V = np.array([1, 1, 1])
tol = 0.0001
it_max = 100
```

In [41]:
```
X, i = newton(V, tol, it_max)
X = np.round(X, 3)
if i == it_max:
    print("O método não convergiu")
print("\nSolução X = ",  X)
```

```
ITER  0 |X =  [0.858 0.858 1.283]  | Err = 0.19644
ITER  1 |X =  [0.8   0.811 1.307]  | Err = 0.04551
ITER  2 |X =  [0.791 0.807 1.313]  | Err = 0.00678
ITER  3 |X =  [0.79  0.807 1.313]  | Err = 0.00012
ITER  4 |X =  [0.79  0.807 1.313]  | Err = 0.00000

Solução X =  [0.79  0.807 1.313]
```

Método de Broyden:

In [42]:
```
def broyden(X, B):
    it = 0
    err = 10
    while (err >= tol) and (it < it_max):
        J = np.copy(B)

        F = f(X)
        F = np.reshape(F, (-1, 1))

        delta = np.linalg.solve(J, -F)

        X = np.reshape(X, (-1, 1)) + delta
        X = X + delta

        Y = f(X)
        Y = np.reshape(Y, (-1, 1))
        Y = Y - F
        err = np.linalg.norm(delta)/np.linalg.norm(X)

        print("ITER ", it, "|X = ", np.reshape(np.round(X, 3), (1, -1)), " | Err = %.5f

        aux = (Y - np.dot(B, delta))
        numerador = np.dot(aux, np.transpose(delta))
        denominador = np.dot(np.transpose(delta), delta)
        B = B + np.divide(numerador, denominador)
        it += 1

    return (X, it)
```

Para matriz B inicial sendo o jacobiano do vetor X:

In [43]:
```
X = np.array([1, 1, 1])
B = j(X)
```

In [44]:
```
print(B)
```

```
[[64 64  4]
 [ 2  2  2]
 [ 3 -1  1]]
```

In [45]:
```python
X, i = broyden(X, B)
if i == it_max:
    print("O método não convergiu")
print("\nSolução X = ",  np.reshape(np.round(X, 3), (1, -1)))
```

```
ITER  0 |X =  [[0.717 0.717 1.567]]  | Err = 0.18597
ITER  1 |X =  [[0.748 0.734 1.237]]  | Err = 0.10227
ITER  2 |X =  [[0.753 0.808 1.356]]  | Err = 0.04008
ITER  3 |X =  [[0.784 0.803 1.332]]  | Err = 0.01149
ITER  4 |X =  [[0.788 0.81  1.314]]  | Err = 0.00550
ITER  5 |X =  [[0.791 0.806 1.312]]  | Err = 0.00157
ITER  6 |X =  [[0.79  0.807 1.313]]  | Err = 0.00050
ITER  7 |X =  [[0.79  0.807 1.313]]  | Err = 0.00002

Solução X =  [[0.79  0.807 1.313]]
```

Para matriz B identidade:

In [46]:
```python
X = np.array([1, 1, 1])
B = np.eye(3)
```

In [47]:
```python
X, i = broyden(X, B)
if i == it_max:
    print("O método não convergiu")
print("\nSolução X = ",  np.reshape(np.round(X, 3), (1, -1)))
```

```
ITER  0 |X =  [[-33.   1.   1.]]  | Err = 0.51468
ITER  1 |X =  [[1.    1.002 0.936]]  | Err = 10.01856
ITER  2 |X =  [[1.    1.245 1.004]]  | Err = 0.06699
ITER  3 |X =  [[1.    1.017 1.072]]  | Err = 0.06666
ITER  4 |X =  [[1.    1.014 0.855]]  | Err = 0.06530
ITER  5 |X =  [[1.    1.014 0.98 ]]  | Err = 0.03607
ITER  6 |X =  [[1.    1.017 0.983]]  | Err = 0.00127
ITER  7 |X =  [[1.    1.017 0.983]]  | Err = 0.00012
ITER  8 |X =  [[1.    1.017 0.983]]  | Err = 0.00003

Solução X =  [[1.    1.017 0.983]]
```

Percebemos que são soluções diferentes, contudo, ambas são soluções do sistema de equações definido.

Exercício 4)

$$2c_3^2 + c_2^2 + 6c_4^2 = 1$$

$$8c_3^3 + 6c_3c_2^2 + 36c_3c_2c_4 + 108c_3c_4^2 = \theta_1$$

$$60c_3^4 + 60c_3^2c_2^2 + 576c_3^2c_2c_4 + 2232c_3^2c_4^2 + 252c_4^2c_2^2 + 1296c_4^3c_2 + 3348c_4^4 + 24c_2^3c_4 + 3c_2 = \theta_2$$

```
In [48]: def f(V):
             x, y, z = V
             S = np.array([2*(y**2) + x**2 + 6*(z**2) - 1,
                           8*(y**3) + 6*y*(x**2) + 36*y*x*z + 108*y*(z**2) - theta1,
                           60*y**4 + 60*(y**2)*(x**2) + 576*(y**2)*x*z + 2232*(y**2)*(z**2) +
                               252*(z**2)*(x**2)  + 1296*(z**3)*x + 3348*(z**4) + 24*(x**3)*z +
                           ])
             return S
```

```
In [49]: def j(V):
             x, y, z = V
             S = np.array( [[2*x, 4*y, 12*z],
                           [12*y*x + 36*y*z, 24*y**2+6*x**2+36*x*z+108*z**2, 36*y*x+216*y*z],
                           [120*y**2*x+576*y**2*z+504*z**2*x+1296*z**3+72*x**2*z+3,
                                   240*y**3+120*y*x**2+2*576*y*x*z+4464*y*z**2,
                                   576*y**2*x+4464*y**2*z+504*z*x**2+3*1296*z**2*x+4*3348*z**3+2
                           ])
             return S
```

a)

$$\theta_1 = 0$$
$$\theta_2 = 3$$

```
In [50]: theta1 = 0
         theta2 = 3
```

```
In [51]: V = np.array([0.5, 0.1, 1])
         tol = 0.0001
         it_max = 100
```

In [52]:
```
X, i = newton(V, tol, it_max)
X = np.round(X, 3)
if i == it_max:
    print("O método não convergiu")
print("\nSolução X = ",  X)
```

```
ITER  0 |X =  [10.14  -0.09  -0.236]  | Err = 0.95836
ITER  1 |X =  [ 5.075 -0.107 -0.431]  | Err = 0.99518
ITER  2 |X =  [5.635 0.63  5.61 ]  | Err = 0.76626
ITER  3 |X =  [-22.01    0.738   6.96 ]  | Err = 1.19838
ITER  4 |X =  [-8.853  0.518  4.621]  | Err = 1.33645
ITER  5 |X =  [-1.988  0.245  3.108]  | Err = 1.90289
ITER  6 |X =  [2.594 0.082 1.964]  | Err = 1.45182
ITER  7 |X =  [-3.3    0.072  2.036]  | Err = 1.51986
ITER  8 |X =  [-0.503  0.032  1.369]  | Err = 1.97124
ITER  9 |X =  [1.494 0.01  0.852]  | Err = 1.19954
ITER  10 |X =  [-0.111  0.007  0.774]  | Err = 2.05330
ITER  11 |X =  [0.645 0.003 0.512]  | Err = 0.97239
ITER  12 |X =  [0.489 0.001 0.384]  | Err = 0.32465
ITER  13 |X =  [1.279 0.    0.189]  | Err = 0.62935
ITER  14 |X =  [0.992 0.    0.138]  | Err = 0.29139
ITER  15 |X =  [0.995 0.    0.075]  | Err = 0.06323
ITER  16 |X =  [1.003 0.    0.031]  | Err = 0.04492
ITER  17 |X =  [1.001 0.    0.008]  | Err = 0.02361
ITER  18 |X =  [1.    0.    0.001]  | Err = 0.00711
ITER  19 |X =  [1. 0. 0.]  | Err = 0.00058
ITER  20 |X =  [1. 0. 0.]  | Err = 0.00000

Solução X =  [1. 0. 0.]
```

Buscando outra solução para o sistema:

In [53]:
```
V = np.array([0.5, 10, 11])
tol = 0.0001
it_max = 100
```

In [54]:
```
X, i = newton(V, tol, it_max)
X = np.round(X, 3)
if i == it_max:
    print("O método não convergiu")
print("\nSolução X = ",  X)
```

```
ITER  0 |X =  [31.766 -1.041  7.099]  | Err = 1.02520
ITER  1 |X =  [11.598 -0.732  6.747]  | Err = 1.50135
ITER  2 |X =  [-5.503 -0.581  6.616]  | Err = 1.98279
ITER  3 |X =  [ 3.962 -0.207  4.254]  | Err = 1.67840
ITER  4 |X =  [-20.728  -0.269   5.669]  | Err = 1.15074
ITER  5 |X =  [-9.002 -0.218  3.678]  | Err = 1.22277
ITER  6 |X =  [-2.995 -0.121  2.476]  | Err = 1.57596
ITER  7 |X =  [ 0.329 -0.049  1.64 ]  | Err = 2.04880
ITER  8 |X =  [ 5.799e+00 -1.000e-03  6.820e-01]  | Err = 0.95101
ITER  9 |X =  [ 2.74e+00 -1.00e-03  6.90e-01]  | Err = 1.08283
ITER  10 |X =  [ 1.13  -0.     0.624]  | Err = 1.24721
ITER  11 |X =  [ 0.284 -0.     0.531]  | Err = 1.41563
ITER  12 |X =  [-1.038 -0.     0.528]  | Err = 1.13522
ITER  13 |X =  [-0.622 -0.     0.388]  | Err = 0.59863
ITER  14 |X =  [-0.657 -0.     0.316]  | Err = 0.10882
ITER  15 |X =  [-0.74  -0.     0.279]  | Err = 0.11606
ITER  16 |X =  [-0.763 -0.     0.264]  | Err = 0.03353
ITER  17 |X =  [-0.766 -0.     0.263]  | Err = 0.00401
ITER  18 |X =  [-0.766 -0.     0.262]  | Err = 0.00006

Solução X =  [-0.766 -0.     0.262]
```

b)

$$\theta_1 = 0.75$$
$$\theta_2 = 6.5$$

In [55]:
```
theta1 = 0.75
theta2 = 6.5
```

In [56]:
```
V = np.array([0.4, 0.1, 0.8])
X, i = newton(V, tol, it_max)
X = np.round(X, 3)
if i == it_max:
    print("O método não convergiu")
print("\nSolução X = ",  X)
```

```
ITER  0 |X =  [ 6.032 -0.044  0.022]  | Err = 0.94282
ITER  1 |X =  [ 3.098 -0.038  0.031]  | Err = 0.94681
ITER  2 |X =  [ 1.709 -0.02   0.042]  | Err = 0.81280
ITER  3 |X =  [1.143 0.025 0.057]  | Err = 0.49660
ITER  4 |X =  [0.995 0.076 0.066]  | Err = 0.15633
ITER  5 |X =  [0.98  0.088 0.063]  | Err = 0.01941
ITER  6 |X =  [0.98  0.088 0.063]  | Err = 0.00076
ITER  7 |X =  [0.98  0.088 0.063]  | Err = 0.00000

Solução X =  [0.98  0.088 0.063]
```

Buscando outra solução para o sistema:

In [57]:
```
V = np.array([0.8, 0.1, 0.6])
X, i = newton(V, tol, it_max)
X = np.round(X, 3)
if i == it_max:
    print("O método não convergiu")
print("\nSolução X = ",  X)
```

```
ITER  0 |X =  [0.091 0.074 0.506]  | Err = 1.37798
ITER  1 |X =  [-0.4    0.071  0.429]  | Err = 0.84025
ITER  2 |X =  [-0.548  0.088  0.352]  | Err = 0.25537
ITER  3 |X =  [-0.672  0.127  0.302]  | Err = 0.18608
ITER  4 |X =  [-0.709  0.167  0.274]  | Err = 0.07874
ITER  5 |X =  [-0.715  0.182  0.265]  | Err = 0.02348
ITER  6 |X =  [-0.716  0.183  0.265]  | Err = 0.00175
ITER  7 |X =  [-0.716  0.183  0.265]  | Err = 0.00001

Solução X =  [-0.716  0.183  0.265]
```

c)

$$\theta_1 = 0$$
$$\theta_2 = 11.667$$

In [58]:
```
theta1 = 0
theta2 = 11.667
```

In [59]:
```
V = np.array([0.4, 0.1, 0.8])
X, i = newton(V, tol, it_max)
X = np.round(X, 3)
if i == it_max:
    print("O método não convergiu")
print("\nSolução X = ",  X)
```

```
ITER  0 |X =  [ 6.067 -0.054  0.02 ]  | Err = 0.94318
ITER  1 |X =  [ 3.115 -0.052  0.028]  | Err = 0.94743
ITER  2 |X =  [ 1.716 -0.043  0.044]  | Err = 0.81501
ITER  3 |X =  [ 1.14  -0.021  0.083]  | Err = 0.50528
ITER  4 |X =  [ 0.975 -0.001  0.119]  | Err = 0.17310
ITER  5 |X =  [ 0.955 -0.     0.121]  | Err = 0.02048
ITER  6 |X =  [ 0.955 -0.     0.121]  | Err = 0.00025
ITER  7 |X =  [ 0.955 -0.     0.121]  | Err = 0.00000

Solução X =  [ 0.955 -0.     0.121]
```

Buscando outra solução para o sistema:

In [60]:
```python
V = np.array([1, 1, 1])
X, i = newton(V, tol, it_max)
X = np.round(X, 3)
if i == it_max:
    print("O método não convergiu")
print("\nSolução X = ",  X)
```

```
ITER  0 |X =  [-5.717  1.559  1.267]  | Err = 1.11319
ITER  1 |X =  [-2.493  1.402  0.719]  | Err = 1.11041
ITER  2 |X =  [-0.902  0.953  0.511]  | Err = 1.18350
ITER  3 |X =  [-0.54   0.251  0.533]  | Err = 0.98964
ITER  4 |X =  [-0.34   0.104  0.414]  | Err = 0.50312
ITER  5 |X =  [-0.549  0.05   0.357]  | Err = 0.34055
ITER  6 |X =  [-0.63   0.015  0.321]  | Err = 0.13489
ITER  7 |X =  [-0.653  0.002  0.31 ]  | Err = 0.03988
ITER  8 |X =  [-0.655  0.     0.309]  | Err = 0.00359
ITER  9 |X =  [-0.655  0.     0.309]  | Err = 0.00003

Solução X =  [-0.655  0.     0.309]
```

Exercício 5)

$$f(x) = b_0 + b_1 x^{b_2}$$

$$x = [1, 2, 3]$$
$$y = [1, 2, 9]$$

Mínimos quadrados

Rotina para definir a funcao $f(x)$ e definir o jacobiano $j(x)$ de f .

In [61]:
```python
def f(B):
    X = np.array([1, 2, 3])
    Y = np.array([1, 2, 9])
    b0, b1, b2 = B

    S = np.array([b0 + b1*(X[0]**b2) - Y[0],
                  b0 + b1*(X[1]**b2) - Y[1],
                  b0 + b1*(X[2]**b2) - Y[2]
                  ])
    return S
```

In [62]:
```python
def j(B):
    b0, b1, b2 = B

    J = np.array([[1,  1,  0],
                  [1,  2**b2,  b1*(2**b2)*np.log(2)],
                  [1,  3**b2,  b1*(3**b2)*np.log(3)]
                  ])
    return J
```

In [63]:
```python
def minimos(B, tol, it_max):
    it = 0
    err = 10

    while (err >= tol) and (it < it_max):
        J = j(B)
        F = f(B)
        delta = np.linalg.solve(
            np.dot(np.transpose(J), J),  np.dot(np.transpose(J), -F))
        B = B + delta
        err = np.linalg.norm(delta)/np.linalg.norm(B)
        x = np.round(B, 3)
        print("ITER ", it, "|X = ",  x, " | Err = %.5f" % err)
        it += 1

    return (B, it)
```

Definindo valores iniciais:

In [64]:
```python
B = np.array([0, 1, 2])
tol = 0.0001
it_max = 100
```

In [65]:
```python
V, i = minimos(B, tol, it_max)
V = np.round(V, 4)
if i == it_max:
    print("O método não convergiu")
print("\nSolução X = ",  V)
```

```
ITER  0 |X =  [ 2.643 -1.643  4.139]  | Err = 0.83165
ITER  1 |X =  [0.99  0.01  4.097]  | Err = 0.55480
ITER  2 |X =  [9.920e-01 8.000e-03 1.133e+01]  | Err = 0.63594
ITER  3 |X =  [9.9900e-01 1.0000e-03 1.1214e+01]  | Err = 0.01030
ITER  4 |X =  [9.9900e-01 1.0000e-03 1.0263e+01]  | Err = 0.09224
ITER  5 |X =  [9.980e-01 2.000e-03 8.634e+00]  | Err = 0.18748
ITER  6 |X =  [9.940e-01 6.000e-03 6.303e+00]  | Err = 0.36530
ITER  7 |X =  [0.978 0.022 4.163]  | Err = 0.50031
ITER  8 |X =  [0.988 0.012 7.159]  | Err = 0.41454
ITER  9 |X =  [0.986 0.014 6.31 ]  | Err = 0.13301
ITER  10 |X =  [0.978 0.022 5.418]  | Err = 0.16194
ITER  11 |X =  [0.97  0.03  5.051]  | Err = 0.07150
ITER  12 |X =  [0.969 0.031 5.064]  | Err = 0.00260
ITER  13 |X =  [0.969 0.031 5.063]  | Err = 0.00014
ITER  14 |X =  [0.969 0.031 5.063]  | Err = 0.00000

Solução X =  [0.9692 0.0308 5.0631]
```

Definindo novos valores iniciais:

In [66]:
```python
B = np.array([1, 0.5, 2])
tol = 0.0001
it_max = 100
```

In [67]:
```python
V, i = minimos(B, tol, it_max)
V = np.round(V, 4)
if i == it_max:
    print("O método não convergiu")
print("\nSolução X = ",  V)
```

```
ITER  0 |X =  [ 2.643 -1.643  6.277]  | Err = 0.72196
ITER  1 |X =  [0.978 0.022 6.285]  | Err = 0.37019
ITER  2 |X =  [0.978 0.022 5.713]  | Err = 0.09859
ITER  3 |X =  [0.973 0.027 5.204]  | Err = 0.09620
ITER  4 |X =  [0.969 0.031 5.064]  | Err = 0.02715
ITER  5 |X =  [0.969 0.031 5.063]  | Err = 0.00018
ITER  6 |X =  [0.969 0.031 5.063]  | Err = 0.00000

Solução X =  [0.9692 0.0308 5.0631]
```