Laboratório de Matemática e Estatística

1ª Frequência

29.03.2019

Resolução

1.

Seja $y = f(x)$ uma função com o domínio D e $y = g(x)$ a sua inversa.

Pretende-se esboçar numa figura os gráficos das funções $f(.)$ e de $g(.)$.

Na mesma figura deve ser colocado o gráfico da função $y = x$ no intervalo D (a bissetriz do primeiro e terceiro quadrantes) na forma de uma linha de cor "gray".

Tarefas:

- esboçar os gráficos pela função "plot" de "matplotlib.pyplot"
- colocar o título da figura "Função e a sua inversa"
- colocar os rótulos dos eixos "eixo x" e "eixo y"
- colocar as legendas para f(x) e g(x)
- colocar na figura uma anotação que aponte na bissetriz

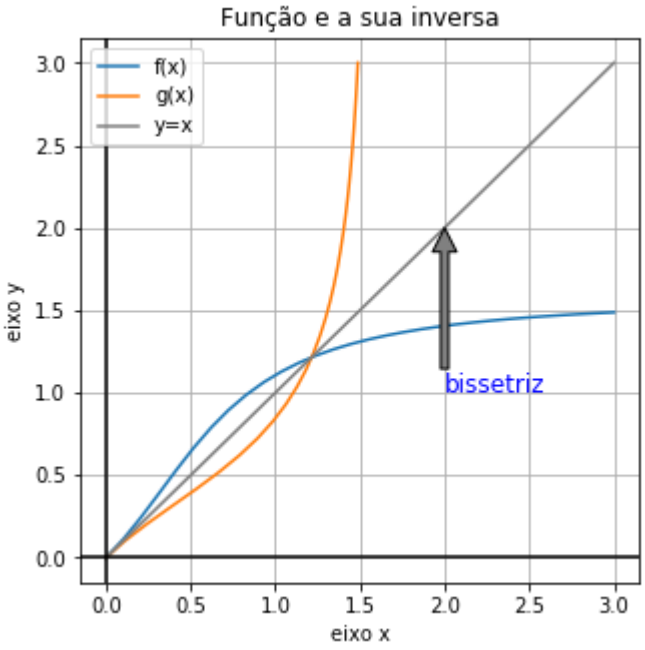
==========

[1.1] Pergunta 1. Versão 1

$y = arctan(x^2 + x)$, domínio $D = [0, 3]$

In [2]:

```
### 1.1 ###
import numpy as np
import matplotlib.pyplot as plt

def f(x):
    return np.arctan(x**2 + x)

xx = np.linspace(0, 3, 30)
yy = f(xx)

fig = plt.figure(figsize=(5,5))
plt.plot(xx, yy, label='f(x)')
plt.plot(yy, xx, label='g(x)')
plt.plot(xx, xx, label='y=x', c='gray')
plt.title('Função e a sua inversa')
plt.xlabel('eixo x')
plt.ylabel('eixo y')
plt.grid()
plt.legend()
# Eixos x e y
plt.axhline(y=0, color='k')
plt.axvline(x=0, color='k')
plt.annotate('bissetriz', xy=(2,2), xytext=(2, 1), color='b',
             fontsize=12, arrowprops=dict(facecolor='gray'))
plt.show()
```



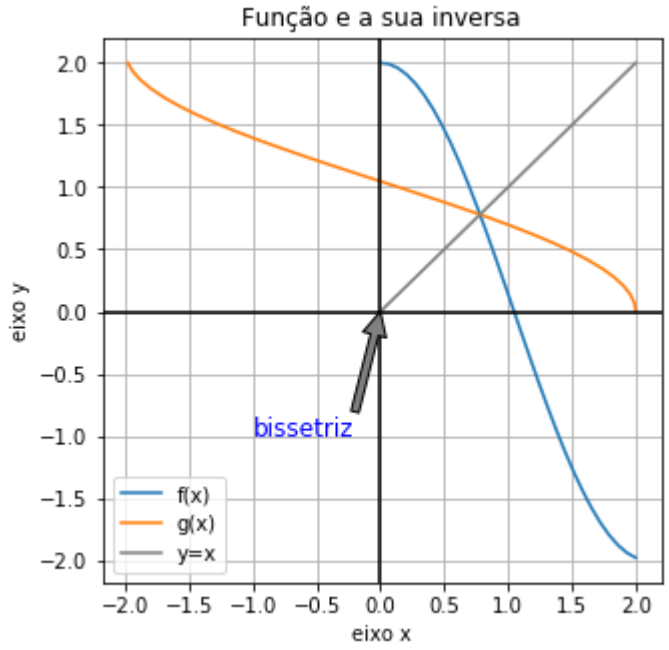
==========

[1.2] Pergunta 1. Versão 2

$y = 2cos(0.5x + x)$, domínio $D = [0, 2]$

In [3]:

```
### 1.2 ###
import numpy as np
import matplotlib.pyplot as plt

def f(x):
    return 2*np.cos(0.5*x + x)

xx = np.linspace(0, 2, 30)
yy = f(xx)

fig = plt.figure(figsize=(5,5))
plt.plot(xx, yy, label='f(x)')
plt.plot(yy, xx, label='g(x)')
plt.plot(xx, xx, label='y=x', c='gray')
plt.title('Função e a sua inversa')
plt.xlabel('eixo x')
plt.ylabel('eixo y')
plt.grid()
plt.legend()
# Eixos x e y
plt.axhline(y=0, color='k')
plt.axvline(x=0, color='k')
plt.annotate('bissetriz', xy=(0,0), xytext=(-1, -1), color='b',
             fontsize=12, arrowprops=dict(facecolor='gray'))
plt.show()
```



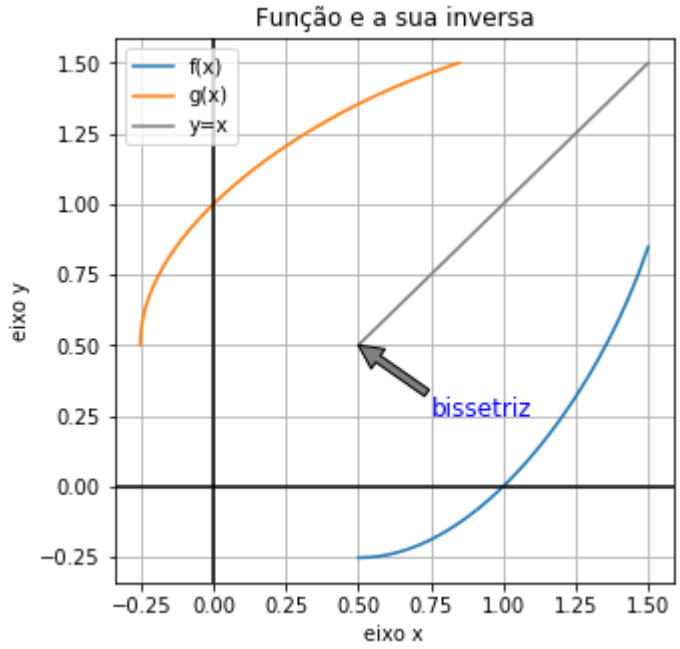
==========

[1.3] Pergunta 1. Versão 3

$y = arcsin(x^2 - x)$, domínio $D = [0.5, 1.5]$

In [4]:

```
### 1.3 ###
import numpy as np
import matplotlib.pyplot as plt

def f(x):
    return np.arcsin(x**2 - x)

xx = np.linspace(0.5, 1.5, 30)
yy = f(xx)

fig = plt.figure(figsize=(5,5))
plt.plot(xx, yy, label='f(x)')
plt.plot(yy, xx, label='g(x)')
plt.plot(xx, xx, label='y=x', c='gray')
plt.title('Função e a sua inversa')
plt.xlabel('eixo x')
plt.ylabel('eixo y')
plt.grid()
# Eixos x e y
plt.axhline(y=0, color='k')
plt.axvline(x=0, color='k')
plt.annotate('bissetriz', xy=(0.5,0.5), xytext=(0.75, 0.25), color='b',
             fontsize=12, arrowprops=dict(facecolor='gray'))

plt.legend()
plt.show()
```



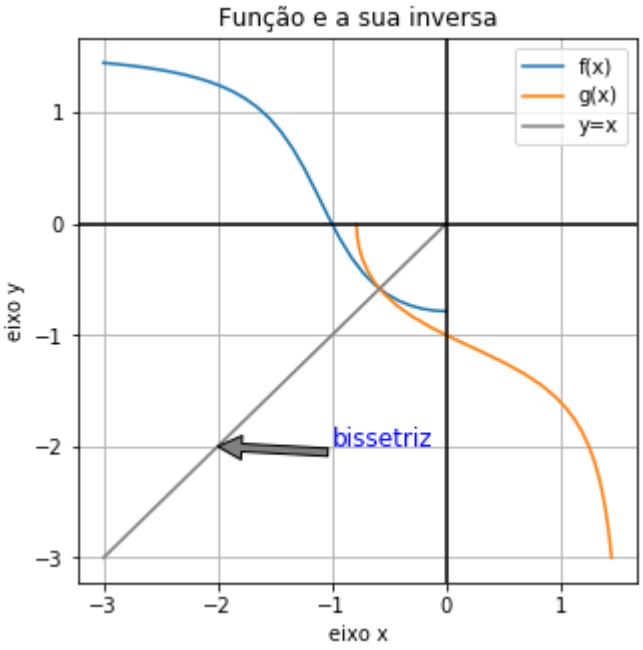
==========

[1.4] Pergunta 1. Versão 4

$y = arctan(x^2 - 1)$, domínio $D = [-3, 0]$

In [5]:

```
### 1.4 ###
import numpy as np
import matplotlib.pyplot as plt

def f(x):
    return np.arctan(x**2 - 1)

xx = np.linspace(-3, 0, 30)
yy = f(xx)

fig = plt.figure(figsize=(5,5))
plt.plot(xx, yy, label='f(x)')
plt.plot(yy, xx, label='g(x)')
plt.plot(xx, xx, label='y=x', c='gray')
plt.title('Função e a sua inversa')
plt.xlabel('eixo x')
plt.ylabel('eixo y')
plt.grid()
plt.legend()
# Eixos x e y
plt.axhline(y=0, color='k')
plt.axvline(x=0, color='k')
plt.annotate('bissetriz', xy=(-2,-2), xytext=(-1, -2), color='b',
             fontsize=12, arrowprops=dict(facecolor='gray'))
plt.show()
```



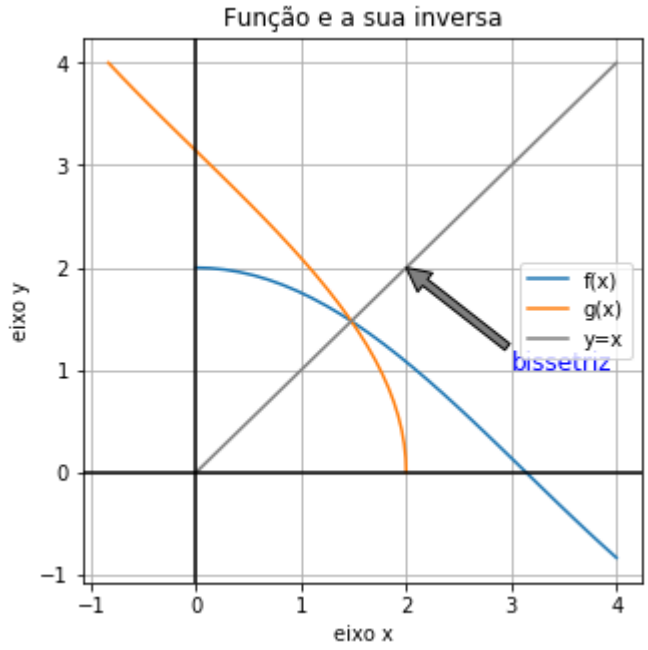
==========

[1.5] Pergunta 1. Versão 5

$y = 2cos(0.5x - x)$, domínio $D = [0, 4]$

In [6]:

```
### 1.5 ###
import numpy as np
import matplotlib.pyplot as plt

def f(x):
    return 2*np.cos(0.5*x - x)

xx = np.linspace(0, 4, 30)
yy = f(xx)

fig = plt.figure(figsize=(5,5))
plt.plot(xx, yy, label='f(x)')
plt.plot(yy, xx, label='g(x)')
plt.plot(xx, xx, label='y=x', c='gray')
plt.title('Função e a sua inversa')
plt.xlabel('eixo x')
plt.ylabel('eixo y')
plt.axhline(y=0, color='k')
plt.axvline(x=0, color='k')
plt.grid()
plt.legend()
plt.annotate('bissetriz', xy=(2,2), xytext=(3, 1), color='b',
             fontsize=12, arrowprops=dict(facecolor='gray'))
plt.show()
```



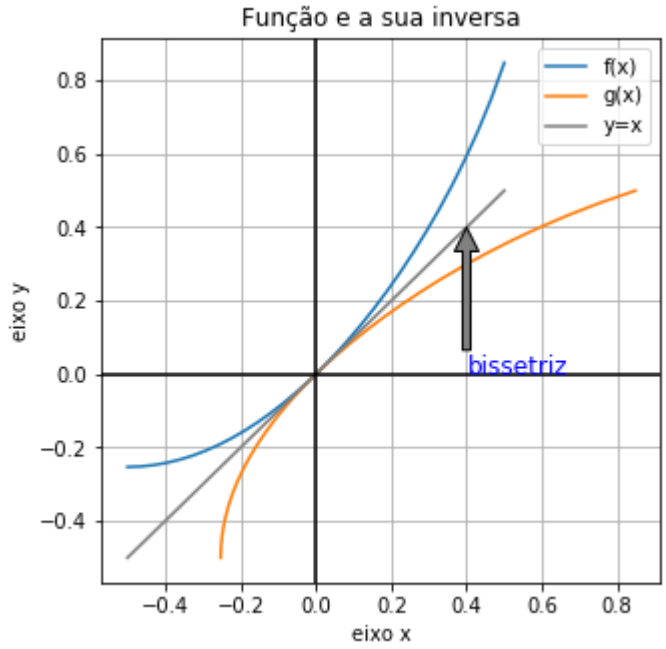
==========

[1.6] Pergunta 1. Versão 6

$y = arcsin(x^2 + x)$, domínio $D = [-0.5, 0.5]$

In [7]:

```
### 1.6 ###
import numpy as np
import matplotlib.pyplot as plt

def f(x):
    return np.arcsin(x**2 + x)

xx = np.linspace(-0.5, 0.5, 30)
yy = f(xx)

fig = plt.figure(figsize=(5,5))
plt.plot(xx, yy, label='f(x)')
plt.plot(yy, xx, label='g(x)')
plt.plot(xx, xx, label='y=x', c='gray')
plt.title('Função e a sua inversa')
plt.xlabel('eixo x')
plt.ylabel('eixo y')
plt.grid()
plt.legend()
# Eixos x e y
plt.axhline(y=0, color='k')
plt.axvline(x=0, color='k')
plt.annotate('bissetriz', xy=(0.4,0.4), xytext=(0.4, 0), color='b',
             fontsize=12, arrowprops=dict(facecolor='gray'))

plt.show()
```



2.

Método da falsa posição

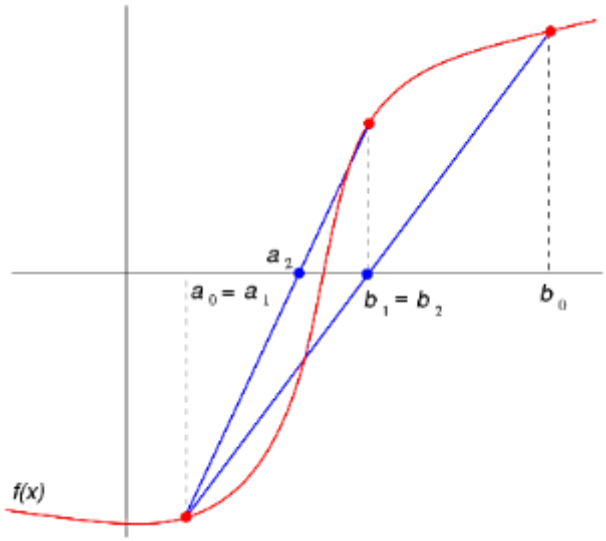
Partimos de um intervalo inicial $[a_0, b_0]$ com $f(a_0)$, e $f(b_0)$ de sinais opostos, assegurando que no interior existe pelo menos uma raiz, de acordo com o Teorema de Bolzano. O objetivo do algoritmo é obter em cada passo um menor intervalo $[a_k, b_k]$ que ainda contenha uma raiz da função f. Na iteração de número k, calcula-se c_k :

$$c_k = b_k - \frac{f(b_k)(b_k - a_k)}{f(b_k) - f(a_k)}$$

onde c_k é a raiz da equação linear corespondente à reta que passa pelos pontos $(a_k, f(a_k))$ e $(b_k, f(b_k))$.

Se $f(a_k)$ e $f(c_k)$ têm o mesmo sinal, então escolhemos $a_{k+1} = c_k$ e $b_{k+1} = b_k$, caso contrário, vamos definir $a_{k+1} = a_k$ e $b_{k+1} = c_k$. Este processo é repetido até que seja encontrada uma raiz aproximada, suficientemente compatível com o erro máximo estimado.

A única diferença entre o método da falsa posição e o método da bissecção é que o último usa $c_k = (a_k + b_k)/2$.



Tarefas:

- Elaborar a função `falsa_pos(fun, a0, b0, eps)` que implemente o método da falsa posição, onde
  - fun é uma função da variável real fun(x)
  - a0, b0 - os extremos do intervalo que contenha uma raiz da equação fun(x)
  - eps - a precisão requerida do resultado (o erro máximo permitido)
- Aplicar a função desenvolvida para calcular a raiz da equação $f(x) = 0$ no intervalo $[a, b]$ com um erro inferior a 0.01

==========

[2.1] Pergunta 2. Versão 1

$f(x) = sin(x) + x - 2$ $\qquad$ $[a, b] = [1, 2]$

In [9]:

```
### 2.1 ###
from math import sin, log, fabs, cos

def myfun(x):
    return  sin(x) + x - 2

def falsa_pos(fun, a, b, eps=0.001):
    ############################
    c = b - fun(b)*(b-a)/(fun(b)-fun(a))
    ############################
    if fun(a)*fun(b) >= 0:
        print('Intervalo [a,b] não verifica o Teorema de Bolzano!')
        return None
    err = (b - a)/2
    for k in range(1,101):
        if fabs(err) <= eps:
            return c
        print('Iter={:2d}, x={:.4f}, a={:.4f}, b={:.4f}, erro={:.4f}'.format(k,c,a,b,err))
        if fabs(fun(c)) < 0.0001:
            return c
        elif fun(a)*fun(c) > 0:
            a = c
        elif fun(c)*fun(b) > 0:
            b = c
        else:
            print('Erro !!!')
            return None
        ############################
        c = b - fun(b)*(b-a)/(fun(b)-fun(a))
        ############################
        err = (b - a)/2
    return c

a = 1
b = 2
res = falsa_pos(myfun, a, b, 0.01)
if res!=None:
    print ('Resposta: x={a:.4f}, f(x)={b:.4f}'.format(a=res, b=myfun(res)))
```

```
Iter= 1, x=1.1485, a=1.0000, b=2.0000, erro=0.5000
Iter= 2, x=1.1074, a=1.0000, b=1.1485, erro=0.0742
Iter= 3, x=1.1061, a=1.0000, b=1.1074, erro=0.0537
Resposta: x=1.1061, f(x)=0.0001
```

==========

[2.2] Pergunta 2. Versão 2

$f(x) = cos(x) - x, \qquad [a, b] = [0, 1]$

In [10]:

```
### 2.2 ###
from math import sin, log, fabs, cos

def myfun(x):
    return  cos(x) - x

def falsa_pos(fun, a, b, eps=0.001):
    ############################
    c = b - fun(b)*(b-a)/(fun(b)-fun(a))
    ############################
    if fun(a)*fun(b) >= 0:
        print('Intervalo [a,b] não verifica o Teorema de Bolzano!')
        return None
    err = (b - a)/2
    for k in range(1,101):
        if fabs(err) <= eps:
            return c
        print('Iter={:2d}, x={:.4f}, a={:.4f}, b={:.4f}, erro={:.4f}'.format(k,c,a,b,err))
        if fabs(fun(c)) < 0.0001:
            return c
        elif fun(a)*fun(c) > 0:
            a = c
        elif fun(c)*fun(b) > 0:
            b = c
        else:
            print('Erro !!!')
            return None
        ############################
        c = b - fun(b)*(b-a)/(fun(b)- fun(a))
        ############################
        err = (b - a)/2
    return c

a = 0
b = 1
res = falsa_pos(myfun, a, b, 0.01)
if res!=None:
    print ('Resposta: x={a:.4f}, f(x)={b:.4f}'.format(a=res, b=myfun(res)))
```

```
Iter= 1, x=0.6851, a=0.0000, b=1.0000, erro=0.5000
Iter= 2, x=0.7363, a=0.6851, b=1.0000, erro=0.1575
Iter= 3, x=0.7389, a=0.7363, b=1.0000, erro=0.1319
Iter= 4, x=0.7391, a=0.7389, b=1.0000, erro=0.1305
Resposta: x=0.7391, f(x)=0.0000
```

==========

[2.3] Pergunta 2. Versão 3

$f(x) = log(x) - x^2 + 5 \qquad [a, b] = [2, 3]$

In [11]:

```
### 2.3 ###
from math import sin, log, fabs, cos

def myfun(x):
    return  log(x) - x**2 + 5

def falsa_pos(fun, a, b, eps=0.001):
    ############################
    c = b - fun(b)*(b-a)/(fun(b)-fun(a))
    ############################
    if fun(a)*fun(b) >= 0:
        print('Intervalo [a,b] não verifica o Teorema de Bolzano!')
        return None
    err = (b - a)/2
    for k in range(1,101):
        if fabs(err) <= eps:
            return c
        print('Iter={:2d}, x={:.4f}, a={:.4f}, b={:.4f}, erro={:.4f}'.format(k,c,a,b,err))
        if fabs(fun(c)) < 0.0001:
            return c
        elif fun(a)*fun(c) > 0:
            a = c
        elif fun(c)*fun(b) > 0:
            b = c
        else:
            print('Erro !!!')
            return None
        ############################
        c = b - fun(b)*(b-a)/(fun(b)- fun(a))
        ############################
        err = (b - a)/2
    return c

a = 2
b = 3
res = falsa_pos(myfun, a, b, 0.01)
if res!=None:
    print ('Resposta: x={a:.4f}, f(x)={b:.4f}'.format(a=res, b=myfun(res)))
```

```
Iter= 1, x=2.3685, a=2.0000, b=3.0000, erro=0.5000
Iter= 2, x=2.4191, a=2.3685, b=3.0000, erro=0.3157
Iter= 3, x=2.4253, a=2.4191, b=3.0000, erro=0.2905
Iter= 4, x=2.4261, a=2.4253, b=3.0000, erro=0.2873
Iter= 5, x=2.4262, a=2.4261, b=3.0000, erro=0.2870
Resposta: x=2.4262, f(x)=0.0001
```

==========

[2.4] Pergunta 2. Versão 4

$f(x) = sin(x)cos(x) + x - 1, \qquad [a, b] = [0, 1]$

In [12]:

```
### 2.4 ###
from math import sin, log, fabs, cos

def myfun(x):
    return  sin(x)*cos(x) + x - 1

def falsa_pos(fun, a, b, eps=0.001):
    ############################
    c = b - fun(b)*(b-a)/(fun(b)-fun(a))
    ############################
    if fun(a)*fun(b) >= 0:
        print('Intervalo [a,b] não verifica o Teorema de Bolzano!')
        return None
    err = (b - a)/2
    for k in range(1,101):
        if fabs(err) <= eps:
            return c
        print('Iter={:2d}, x={:.4f}, a={:.4f}, b={:.4f}, erro={:.4f}'.format(k,c,a,b,err))
        if fabs(fun(c)) < 0.0001:
            return c
        elif fun(a)*fun(c) > 0:
            a = c
        elif fun(c)*fun(b) > 0:
            b = c
        else:
            print('Erro !!!')
            return None
        ############################
        c = b - fun(b)*(b-a)/(fun(b)- fun(a))
        ############################
        err = (b - a)/2
    return c

a = 0
b = 1
res = falsa_pos(myfun, a, b, 0.01)
if res!=None:
    print ('Resposta: x={a:.4f}, f(x)={b:.4f}'.format(a=res, b=myfun(res)))
```

```
Iter= 1, x=0.6875, a=0.0000, b=1.0000, erro=0.5000
Iter= 2, x=0.5836, a=0.0000, b=0.6875, erro=0.3437
Iter= 3, x=0.5593, a=0.0000, b=0.5836, erro=0.2918
Iter= 4, x=0.5543, a=0.0000, b=0.5593, erro=0.2797
Iter= 5, x=0.5533, a=0.0000, b=0.5543, erro=0.2771
Iter= 6, x=0.5531, a=0.0000, b=0.5533, erro=0.2766
Resposta: x=0.5531, f(x)=0.0001
```

==========

[2.5] Pergunta 2. Versão 5

$f(x) = sin(x) + cos(x) + 1, \qquad [a,b] = [3,4]$

In [13]:

```
### 2.5 ###
from math import sin, log, fabs, cos

def myfun(x):
    return  sin(x) + cos(x) + 1

def falsa_pos(fun, a, b, eps=0.001):
    ############################
    c = b - fun(b)*(b-a)/(fun(b)-fun(a))
    ############################
    if fun(a)*fun(b) >= 0:
        print('Intervalo [a,b] não verifica o Teorema de Bolzano!')
        return None
    err = (b - a)/2
    for k in range(1,101):
        if fabs(err) <= eps:
            return c
        print('Iter={:2d}, x={:.4f}, a={:.4f}, b={:.4f}, erro={:.4f}'.format(k,c,a,b,err))
        if fabs(fun(c)) < 0.0001:
            return c
        elif fun(a)*fun(c) > 0:
            a = c
        elif fun(c)*fun(b) > 0:
            b = c
        else:
            print('Erro !!!')
            return None
        ############################
        c = b - fun(b)*(b-a)/(fun(b)-fun(a))
        ############################
        err = (b - a)/2
    return c

a = 3
b = 4
res = falsa_pos(myfun, a, b, 0.01)
if res!=None:
    print ('Resposta: x={a:.4f}, f(x)={b:.4f}'.format(a=res, b=myfun(res)))
```

```
Iter= 1, x=3.2691, a=3.0000, b=4.0000, erro=0.5000
Iter= 2, x=3.1505, a=3.0000, b=3.2691, erro=0.1346
Iter= 3, x=3.1422, a=3.0000, b=3.1505, erro=0.0753
Iter= 4, x=3.1416, a=3.0000, b=3.1422, erro=0.0711
Resposta: x=3.1416, f(x)=-0.0000
```

==========

[2.6] Pergunta 2. Versão 6

$f(x) = sin(x)e^x, \qquad [a,b] = [3,4]$

In [14]:

```
### 2.6 ###
from math import sin, log, fabs, cos, exp

def myfun(x):
    return  sin(x)*exp(x)

def falsa_pos(fun, a, b, eps=0.001):
    ############################
    c = b - fun(b)*(b-a)/(fun(b)-fun(a))
    ############################
    if fun(a)*fun(b) >= 0:
        print('Intervalo [a,b] não verifica o Teorema de Bolzano!')
        return None
    err = (b - a)/2
    for k in range(1,101):
        if fabs(err) <= eps:
            return c
        print('Iter={:2d}, x={:.4f}, a={:.4f}, b={:.4f}, erro={:.4f}'.format(k,c,a,b,err))
        if fabs(fun(c)) < 0.0001:
            return c
        elif fun(a)*fun(c) > 0:
            a = c
        elif fun(c)*fun(b) > 0:
            b = c
        else:
            print('Erro !!!')
            return None
        ############################
        c = b - fun(b)*(b-a)/(fun(b)- fun(a))
        ############################
        err = (b - a)/2
    return c

a = 3
b = 4
res = falsa_pos(myfun, a, b, 0.01)
if res!=None:
    print ('Resposta: x={a:.4f}, f(x)={b:.4f}'.format(a=res, b=myfun(res)))
```

```
Iter= 1, x=3.0642, a=3.0000, b=4.0000, erro=0.5000
Iter= 2, x=3.1003, a=3.0642, b=4.0000, erro=0.4679
Iter= 3, x=3.1198, a=3.1003, b=4.0000, erro=0.4499
Iter= 4, x=3.1302, a=3.1198, b=4.0000, erro=0.4401
Iter= 5, x=3.1356, a=3.1302, b=4.0000, erro=0.4349
Iter= 6, x=3.1385, a=3.1356, b=4.0000, erro=0.4322
Iter= 7, x=3.1400, a=3.1385, b=4.0000, erro=0.4308
Iter= 8, x=3.1408, a=3.1400, b=4.0000, erro=0.4300
Iter= 9, x=3.1412, a=3.1408, b=4.0000, erro=0.4296
Iter=10, x=3.1414, a=3.1412, b=4.0000, erro=0.4294
Iter=11, x=3.1415, a=3.1414, b=4.0000, erro=0.4293
Iter=12, x=3.1415, a=3.1415, b=4.0000, erro=0.4293
Iter=13, x=3.1416, a=3.1415, b=4.0000, erro=0.4292
Iter=14, x=3.1416, a=3.1416, b=4.0000, erro=0.4292
Iter=15, x=3.1416, a=3.1416, b=4.0000, erro=0.4292
Iter=16, x=3.1416, a=3.1416, b=4.0000, erro=0.4292
Iter=17, x=3.1416, a=3.1416, b=4.0000, erro=0.4292
Resposta: x=3.1416, f(x)=0.0001
```

3.

Pretende-se resolver os sistemas lineares:

(I).

$$\begin{aligned} 2x_1 - x_3 + 2x_4 &= 11 \\ x_2 + x_3 &= 0 \\ 2x_1 - 3x_2 + x_3 + x_4 &= 3 \\ x_1 - x_2 + x_3 - 2x_4 &= -6 \end{aligned}$$

(II).

$$\begin{aligned} x_2 - x_3 + 2x_4 &= 8 \\ x_1 + x_2 + x_4 &= 6 \\ x_1 - x_2 + x_3 &= 0 \\ 2x_1 + x_3 - 2x_4 &= -3 \end{aligned}$$

Tarefas:

- Representar o sistema linear da sua versão na forma matricial $Ax = b$
- Resolve-lo pelo método indicado na sua versão

==========

[3.1] Pergunta 3. Versão 1

Sistema (I), resolver pela inversão da matriz A, i.e. $x = A^{-1}b$ (usar `numpy.linalg`)

In [15]:

```
### 3.1 ###   np.array
import numpy as np
import numpy.linalg as la

A = np.array([[2,0,-1,2],[0,1,1,0],[2,-3,1,1],[1,-1,1,-2]])
b = np.array([11, 0, 3, -6])

A1 = la.inv(A)
x = np.dot(A1,b)
print(x)
# Verificação da resposta  Ax-b (não é obrigatório):
np.dot(A,x) - b
```

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

Out[15]:

```
array([ 0.0000000e+00, -4.4408921e-16,  8.8817842e-16, -8.8817842e-16])
```

In [16]:

```
### 3.1 ###   np.matrix
import numpy as np
import numpy.linalg as la

A = np.matrix([[2,0,-1,2],[0,1,1,0],[2,-3,1,1],[1,-1,1,-2]])
b = np.matrix([11, 0, 3, -6])

A1 = la.inv(A)
x = A1 * b.T
print(x)
# Verificação da resposta  Ax-b (não é obrigatório):
A*x - b.T
```

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

Out[16]:

```
matrix([[ 0.0000000e+00],
        [-4.4408921e-16],
        [ 8.8817842e-16],
        [-8.8817842e-16]])
```

==========

[3.2] Pergunta 3. Versão 2

Sistema (II), resolver pela inversão da matriz A, i.e. $x = A^{-1}b$ (usar `numpy.linalg`)

In [17]:

```
### 3.2 ###   np.array
import numpy as np
import numpy.linalg as la

A = np.array([[0,1,-1,2],[1,1,0,1],[1,-1,1,0],[2,0,1,-2]])
b = np.array([8,6,0,-3])

A1 = la.inv(A)
x = np.dot(A1,b)
print(x)
# Verificação da resposta  Ax-b (não é obrigatório):
np.dot(A,x) - b
```

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

Out[17]:

```
array([0., 0., 0., 0.])
```

In [18]:

```
### 3.2 ###   np.matrix
import numpy as np
import numpy.linalg as la

A = np.matrix([[0,1,-1,2],[1,1,0,1],[1,-1,1,0],[2,0,1,-2]])
b = np.matrix([8,6,0,-3])

A1 = la.inv(A)
x = A1 * b.T
print(x)
# Verificação da resposta  Ax-b (não é obrigatório):
A*x - b.T
```

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

Out[18]:

```
matrix([[0.],
        [0.],
        [0.],
        [0.]])
```

==========

[3.3] Pergunta 3. Versão 3

Sistema (I), resolver pela inversão da matriz A, i.e. $x = A^{-1}b$ (usar `sympy`)

In [19]:

```
### 3.3 ###
import sympy
sympy.init_printing()

A = sympy.Matrix([[2,0,-1,2],[0,1,1,0],[2,-3,1,1],[1,-1,1,-2]])
b = sympy.Matrix([11, 0, 3, -6])
A1 = A.inv()
A1
```

Out[19]:

$$\begin{bmatrix} \frac{2}{5} & \frac{3}{25} & -\frac{2}{25} & \frac{9}{25} \\ \frac{1}{5} & \frac{9}{25} & -\frac{6}{25} & \frac{2}{25} \\ -\frac{1}{5} & \frac{16}{25} & \frac{6}{25} & -\frac{2}{25} \\ 0 & \frac{1}{5} & \frac{1}{5} & -\frac{2}{5} \end{bmatrix}$$

In [20]:

```
x = A1 * b
x
```

Out[20]:

$$\begin{bmatrix} 2 \\ 1 \\ -1 \\ 3 \end{bmatrix}$$

In [21]:

```
# Verificação da resposta  Ax-b (não é obrigatório):
A*x - b
```

Out[21]:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

==========

[3.4] Pergunta 3. Versão 4

Sistema (II), resolver pela inversão da matriz A, i.e. $x = A^{-1}b$ (usar `sympy`)

In [22]:

```
### 3.4 ###
import sympy
sympy.init_printing()

A = sympy.Matrix([[0,1,-1,2],[1,1,0,1],[1,-1,1,0],[2,0,1,-2]])
b = sympy.Matrix([8,6,0,-3])

A1 = A.inv()
A1
```

Out[22]:

$$\begin{bmatrix} 1 & -\frac{2}{3} & \frac{1}{3} & \frac{2}{3} \\ -1 & \frac{4}{3} & -\frac{2}{3} & -\frac{1}{3} \\ -2 & 2 & 0 & -1 \\ 0 & \frac{1}{3} & \frac{1}{3} & -\frac{1}{3} \end{bmatrix}$$

In [23]:

```
x = A1 * b
x
```

Out[23]:

$$\begin{bmatrix} 2 \\ 1 \\ -1 \\ 3 \end{bmatrix}$$

In [24]:

```
# Verificação da resposta  Ax-b (não é obrigatório):
A*x - b
```

Out[24]:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

==========

[3.5] Pergunta 3. Versão 5

Sistema (I), resolver pela função `solve` (os dados na forma `array`) de `numpy.linalg`

In [25]:

```
### 3.5 ###
import numpy as np
import numpy.linalg as la

A = np.array([[2,0,-1,2],[0,1,1,0],[2,-3,1,1],[1,-1,1,-2]])
b = np.array([11, 0, 3, -6])

x = la.solve(A,b)
print(x)
# Verificação da resposta  Ax-b (não é obrigatório):
np.dot(A,x) - b
```

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

Out[25]:

```
array([0., 0., 0., 0.])
```

==========

[3.6] Pergunta 3. Versão 6

Sistema (II), resolver pela função `solve` (os dados na forma `array`) de `numpy.linalg`

In [26]:

```
### 3.6 ###
import numpy as np
import numpy.linalg as la

A = np.array([[0,1,-1,2], [1, 1, 0, 1], [1, -1, 1, 0], [2, 0, 1, -2]])
b = np.array([8, 6, 0, -3])

x = la.solve(A,b)
print(x)
# Verificação da resposta  Ax-b (não é obrigatório):
np.dot(A,x) - b
```

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

Out[26]:

```
array([0., 0., 0., 0.])
```

4.

Seja $f(x, y)$ uma função diferenciável.

Tarefas:

- Encontre analiticamente (`sympy`) o vetor gradiente $\nabla f(x, y)$ e a matriz hessiana $H(x, y)$ para a função dada.
- Encontre o valor numérico de ∇f e H no ponto $p = (x, y)$ dado.
- A função $f(x, y)$ possui um ponto estacionário em p ? (justifique a resposta!)
- Aplicando o critério de Sylvester, determine o tipo deste extremo (máximo/mínimo) caso existir ou o tipo de convexidade/concavidade da função no ponto $p = (x, y)$.

==========

[4.1] Pergunta 4. Versão 1

$f(x, y) = 2x^2 - 8x + y^4 - 12y^3 + 54y^2 - 108y + 7$ ponto $p : (x, y) = (2, 3)$

In [125]:

```
### 4.1 ###
from sympy import *
init_printing()

var ('x,y')

f = 2*x**2 - 8*x + y**4 - 12*y**3 + 54*y**2 - 108*y + 7
p = {x:2, y:3}
f, p
```

Out[125]:

$$\left(2x^2 - 8x + y^4 - 12y^3 + 54y^2 - 108y + 7, \quad \{x : 2, \quad y : 3\}\right)$$

In [126]:

```
vg = Matrix([diff(f,x),diff(f,y)])
vg
```

Out[126]:

$$\begin{bmatrix} 4x - 8 \\ 4y^3 - 36y^2 + 108y - 108 \end{bmatrix}$$

In [127]:

```
H = Matrix([[diff(f,x,x), diff(f,x,y)], [diff(f,y,x), diff(f,y,y)]])
H
```

Out[127]:

$$\begin{bmatrix} 4 & 0 \\ 0 & 12\left(y^2 - 6y + 9\right) \end{bmatrix}$$

In [128]:

```
vg.subs(p)
```

Out[128]:

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

In [129]:

```
Hp = H.subs(p)
Hp
```

Out[129]:

$$\begin{bmatrix} 4 & 0 \\ 0 & 0 \end{bmatrix}$$

In [130]:

```
D1 = det(Hp[:1,:1])
D2 = det(Hp[:2,:2])
D1, D2
```

Out[130]:

$$(4, \quad 0)$$

In [34]:

```
# O gradiente é igual a 0 no ponto p, portanto p satisfaz
# a condição necessária do ponto estacionário.
# D1>0, D2=0, o critério de Sylvester não permite identificar
# o tipo do ponto estacionário
```

==========

[4.2] Pergunta 4. Versão 2

$f(x,y) = -x^2 + 2x - 2y^4 + 16y^3 - 48y^2 + 64y + 5$ ponto $p : (x,y) = (1,2)$

In [119]:

```
### 4.2 ###
from sympy import *
init_printing()

var ('x,y')

f = -x**2 + 2*x - 2*y**4 + 16*y**3 - 48*y**2 + 64*y + 5
p = {x:1, y:2}
f,p
```

Out[119]:

$$\left(-x^2 + 2x - 2y^4 + 16y^3 - 48y^2 + 64y + 5, \quad \{x : 1, \quad y : 2\}\right)$$

In [120]:

```
vg = Matrix([diff(f,x),diff(f,y)])
vg
```

Out[120]:

$$\begin{bmatrix} -2x + 2 \\ -8y^3 + 48y^2 - 96y + 64 \end{bmatrix}$$

In [121]:

```
H = Matrix([[diff(f,x,x),diff(f,x,y)],[diff(f,y,x),diff(f,y,y)]])
H
```

Out[121]:

$$\begin{bmatrix} -2 & 0 \\ 0 & 24\left(-y^2+4y-4\right) \end{bmatrix}$$

In [122]:

```
vg.subs(p)
```

Out[122]:

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

In [123]:

```
Hp = H.subs(p)
Hp
```

Out[123]:

$$\begin{bmatrix} -2 & 0 \\ 0 & 0 \end{bmatrix}$$

In [124]:

```
D1 = det(Hp[:1,:1])
D2 = det(Hp[:2,:2])
D1, D2
```

Out[124]:

$$(-2,\quad 0)$$

In [40]:

```
# O gradiente é igual a 0 no ponto p, portanto p satisfaz
# a condição necessária do ponto estacionário.
# D1<0, D2=0, o critério de Sylvester não permite identificar
# o tipo do ponto estacionário
```

==========

[4.3] Pergunta 4. Versão 3

$f(x,y) = x^4 - 4x^3 + 6x^2 - 4x - 2y^2 + 4y - 1$ ponto $p : (x,y) = (1,1)$

In [113]:

```
### 4.3 ###
from sympy import *
init_printing()

var ('x,y')

f = x**4 - 4*x**3 + 6*x**2 - 4*x -2*y**2 + 4*y - 1
p = {x:1, y:1}
f,p
```

Out[113]:

$$\left(x^4 - 4x^3 + 6x^2 - 4x - 2y^2 + 4y - 1, \quad \{x : 1, \quad y : 1\}\right)$$

In [114]:

```
vg = Matrix([diff(f,x), diff(f,y)])
vg
```

Out[114]:

$$\begin{bmatrix} 4x^3 - 12x^2 + 12x - 4 \\ -4y + 4 \end{bmatrix}$$

In [115]:

```
H = Matrix([[diff(f,x,x), diff(f,x,y)], [diff(f,y,x), diff(f,y,y)]])
H
```

Out[115]:

$$\begin{bmatrix} 12\left(x^2 - 2x + 1\right) & 0 \\ 0 & -4 \end{bmatrix}$$

In [116]:

```
vg.subs(p)
```

Out[116]:

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

In [117]:

```
Hp = H.subs(p)
Hp
```

Out[117]:

$$\begin{bmatrix} 0 & 0 \\ 0 & -4 \end{bmatrix}$$

In [118]:

```
D1 = det(Hp[:1,:1])
D2 = det(Hp[:2,:2])
D1, D2
```

Out[118]:

$$(0, \quad 0)$$

In [65]:

```
# O gradiente é igual a 0 no ponto p, portanto p satisfaz
# a condição necessária do ponto estacionário.
# D1=0, D2=0, o critério de Sylvester não permite identificar
# o tipo do ponto estacionário
```

==========

[4.4] Pergunta 4. Versão 4

$f(x, y) = 2x^3 - 12x^2 + 24x + y^2 - 6y - 7$ ponto $p : (x, y) = (2, 3)$

In [106]:

```
### 4.4 ###
from sympy import *
init_printing()

var ('x,y')

f = 2*x**3 - 12*x**2 + 24*x + y**2 - 6*y - 7
p = {x:2, y:3}
f,p
```

Out[106]:

$$\left(2x^3 - 12x^2 + 24x + y^2 - 6y - 7, \quad \{x : 2, \quad y : 3\}\right)$$

In [107]:

```
vg = Matrix([diff(f,x),diff(f,y)])
vg
```

Out[107]:

$$\begin{bmatrix} 6x^2 - 24x + 24 \\ 2y - 6 \end{bmatrix}$$

In [108]:

```
H = Matrix([[diff(f,x,x),diff(f,x,y)],[diff(f,y,x),diff(f,y,y)]])
H
```

Out[108]:

$$\begin{bmatrix} 12\,(x-2) & 0 \\ 0 & 2 \end{bmatrix}$$

In [109]:

```
vg.subs(p)
```

Out[109]:

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

In [110]:

```
H.subs(p)
```

Out[110]:

$$\begin{bmatrix} 0 & 0 \\ 0 & 2 \end{bmatrix}$$

In [111]:

```
Hp = H.subs(p)
Hp
```

Out[111]:

$$\begin{bmatrix} 0 & 0 \\ 0 & 2 \end{bmatrix}$$

In [112]:

```
D1 = det(Hp[:1,:1])
D2 = det(Hp[:2,:2])
D1, D2
```

Out[112]:

$$(0,\quad 0)$$

In [55]:

```
# O gradiente é igual a 0 no ponto p, portanto p saisfaz
# a condição necessária do ponto estacionário.
# D1=0, D2=0, o critério de Sylvester não permite identificar
# o tipo do ponto estacionário
```

==========

[4.5] Pergunta 4. Versão 5

$f(x,y) = 2x^2 - 4x + y^4 + 12y^3 + 54y^2 + 108y + 5$ ponto $p : (x,y) = (1,-3)$

In [94]:

```
### 4.5 ###
from sympy import *
init_printing()

var ('x,y')
f = 2*x**2 - 4*x + y**4 + 12*y**3 + 54*y**2 + 108*y + 5
p = {x:1, y:-3}
f,p
```

Out[94]:

$$\left(2x^2 - 4x + y^4 + 12y^3 + 54y^2 + 108y + 5, \quad \{x : 1, \quad y : -3\}\right)$$

In [95]:

```
vg = Matrix([diff(f,x),diff(f,y)])
vg
```

Out[95]:

$$\begin{bmatrix} 4x - 4 \\ 4y^3 + 36y^2 + 108y + 108 \end{bmatrix}$$

In [96]:

```
H = Matrix([[diff(f,x,x),diff(f,x,y)],[diff(f,y,x),diff(f,y,y)]])
H
```

Out[96]:

$$\begin{bmatrix} 4 & 0 \\ 0 & 12\left(y^2 + 6y + 9\right) \end{bmatrix}$$

In [97]:

```
vg.subs(p)
```

Out[97]:

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

In [98]:

```
Hp = H.subs(p)
Hp
```

Out[98]:

$$\begin{bmatrix} 4 & 0 \\ 0 & 0 \end{bmatrix}$$

In [99]:

```
D1 = det(Hp[:1,:1])
D2 = det(Hp[:2,:2])
D1, D2
```

Out[99]:

$$(4, \quad 0)$$

In [61]:

```
# O gradiente é nulo no ponto p, portanto p saisfaz
# a condição necessária do ponto estacionário.
# D1>0, D2=0, o critério de Sylvester não permite identificar
# o tipo de concavidade
```

==========

[4.6] Pergunta 4. Versão 6

$f(x,y) = -2x^2 - 4x - y^4 + 12y^3 - 54y^2 + 108y - 3$ ponto $p : (x,y) = (-1,3)$

In [100]:

```
### 4.6 ###
from sympy import *
init_printing()

var ('x,y')

f = -2*x**2 - 4*x - y**4 + 12*y**3 - 54*y**2 + 108*y - 3
p = {x:-1, y:3}
f,p
```

Out[100]:

$$\left(-2x^2 - 4x - y^4 + 12y^3 - 54y^2 + 108y - 3, \quad \{x : -1, \quad y : 3\}\right)$$

In [101]:

```
vg = Matrix([diff(f,x),diff(f,y)])
vg
```

Out[101]:

$$\begin{bmatrix} -4x - 4 \\ -4y^3 + 36y^2 - 108y + 108 \end{bmatrix}$$

In [102]:

```
H = Matrix([[diff(f,x,x),diff(f,x,y)],[diff(f,y,x),diff(f,y,y)]])
H
```

Out[102]:

$$\begin{bmatrix} -4 & 0 \\ 0 & 12\left(-y^2+6y-9\right) \end{bmatrix}$$

In [103]:

```
vg.subs(p)
```

Out[103]:

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

In [104]:

```
Hp = H.subs(p)
Hp
```

Out[104]:

$$\begin{bmatrix} -4 & 0 \\ 0 & 0 \end{bmatrix}$$

In [105]:

```
D1 = det(Hp[:1,:1])
D2 = det(Hp[:2,:2])
D1, D2
```

Out[105]:

$$(-4,\quad 0)$$

In [67]:

```
# O gradiente é igual a 0 no ponto p, portanto p satisfaz
# a condição necessária do ponto estacionário
# D1<0, D2=0, o critério de Sylvester não permite identificar
# o tipo do ponto estacionário
```

5.

Pretende-se visualizar a função de duas variáveis $f(x, y)$.

Tarefas:

- Fig1: Utilize as funções `contour()` e `clabel()` de `matplotlib.pyplot` para esboçar as curvas (conjuntos) de nível.
- Fig2: Utilize a função `contour()` no modo de "projection='3d' para visualizar a função em '3D'.
- Fig3: Utilize a função `plot_surface()` para visualizar a função em '3D'

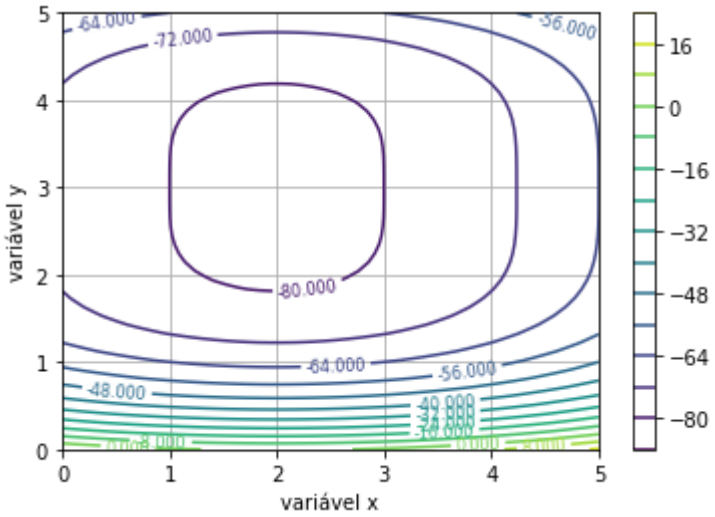
==========

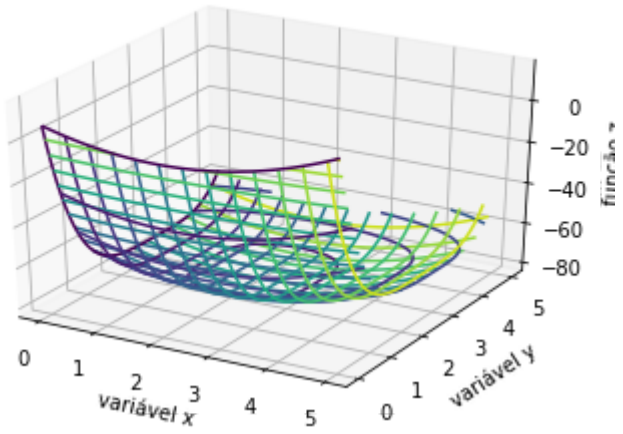
[5.1] Pergunta 5. Versão 1

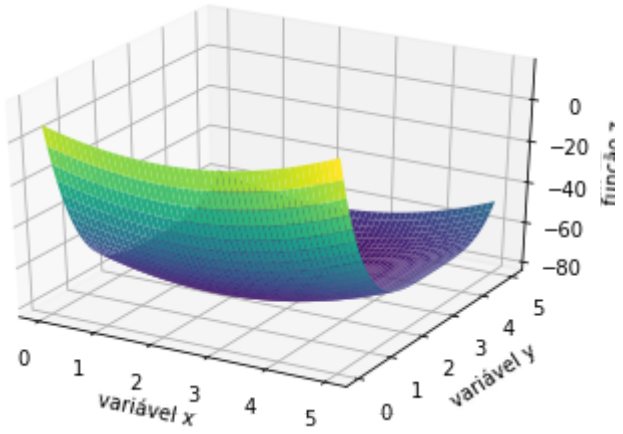
$f(x,y) = 2x^2 - 8x + y^4 - 12y^3 + 54y^2 - 108y + 7; (x_{min}, x_{max}) = (0;5);\ (y_{min}, y_{max}) = (0,5)$
cmap='viridis'

In [142]:

```
### 5.1 ###
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D

def fun(x,y):
    return 2*x**2 - 8*x + y**4 - 12*y**3 + 54*y**2 - 108*y + 7

xx = np.linspace(0, 5, 50)
yy = np.linspace(0, 5, 50)
X,Y = np.meshgrid(xx,yy)
Z = fun(X,Y)

fig1 = plt.figure()
cs = plt.contour(X,Y,Z, 15,cmap='viridis')
plt.clabel(cs, inline=1, fontsize=8)
plt.colorbar(cs)
plt.xlabel('variável x')
plt.ylabel('variável y')
plt.grid()
plt.show()

fig2 = plt.figure()
ax = plt.axes(projection='3d') #projeção 3d
ax.contour(X, Y, Z, 15, zdir='x',cmap='viridis')
ax.contour(X, Y, Z, 15, zdir='y',cmap='viridis')
ax.contour(X, Y, Z, 15, zdir='z',cmap='viridis')
ax.set_xlabel('variável x')
ax.set_ylabel('variável y')
ax.set_zlabel('função z')
plt.show()

fig3 = plt.figure()
ax = plt.axes(projection='3d') #projeção 3d
ax.plot_surface(X, Y, Z, cmap='viridis')
ax.set_xlabel('variável x')
ax.set_ylabel('variável y')
ax.set_zlabel('função z')
plt.show()
```







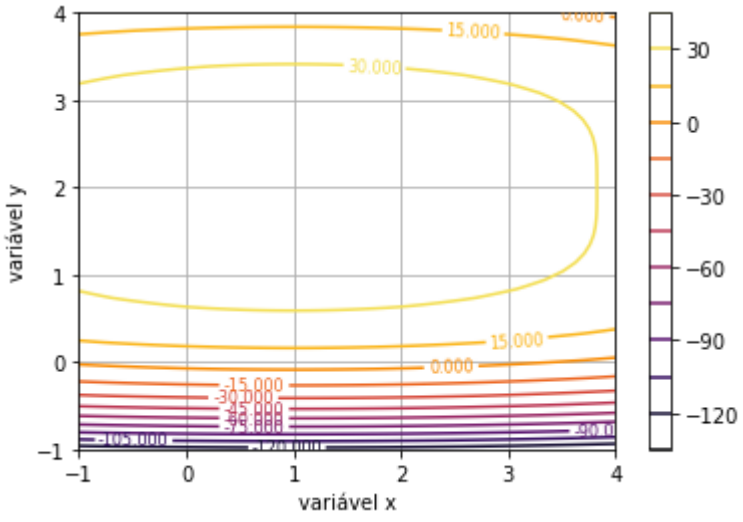
==========

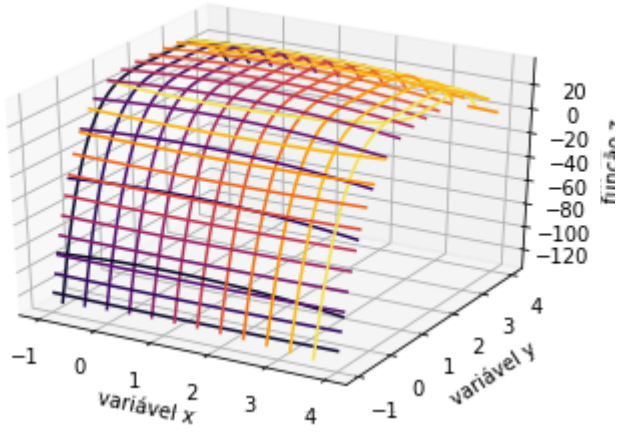
[5.2] Pergunta 5. Versão 2

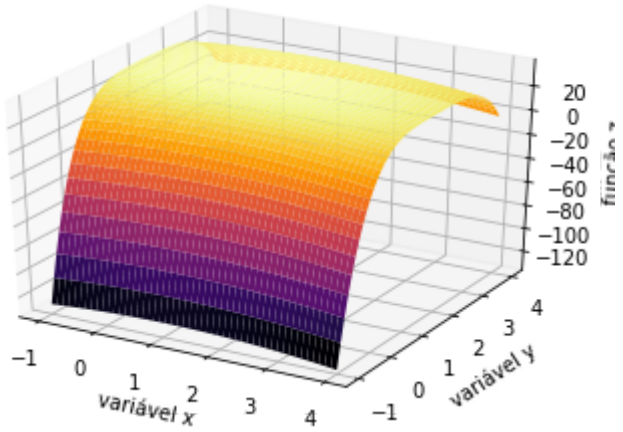
$f(x,y) = -x^2 + 2x - 2y^4 + 16y^3 - 48y^2 + 64y + 5 \quad (x_{min}, x_{max}) = (-1,4);\ (y_{min}, y_{max}) = (-1,4)$
cmap='inferno'

In [143]:

```
### 5.2 ###
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D

def fun(x,y):
    return -x**2 + 2*x - 2*y**4 + 16*y**3 - 48*y**2 + 64*y + 5

xx = np.linspace(-1, 4, 50)
yy = np.linspace(-1, 4, 50)
X,Y = np.meshgrid(xx,yy)
Z = fun(X,Y)

fig1 = plt.figure()
cs = plt.contour(X,Y,Z, 15, cmap='inferno')
plt.clabel(cs, inline=1, fontsize=8)
plt.colorbar(cs)
plt.xlabel('variável x')
plt.ylabel('variável y')
plt.grid()
plt.show()

fig2 = plt.figure()
ax = plt.axes(projection='3d') #projeção 3d
ax.contour(X, Y, Z, 15, zdir='x',cmap='inferno')
ax.contour(X, Y, Z, 15, zdir='y',cmap='inferno')
ax.contour(X, Y, Z, 15, zdir='z',cmap='inferno')
ax.set_xlabel('variável x')
ax.set_ylabel('variável y')
ax.set_zlabel('função z')
plt.show()

fig3 = plt.figure()
ax = plt.axes(projection='3d') #projeção 3d
ax.plot_surface(X, Y, Z,cmap='inferno')
ax.set_xlabel('variável x')
ax.set_ylabel('variável y')
ax.set_zlabel('função z')
plt.show()
```







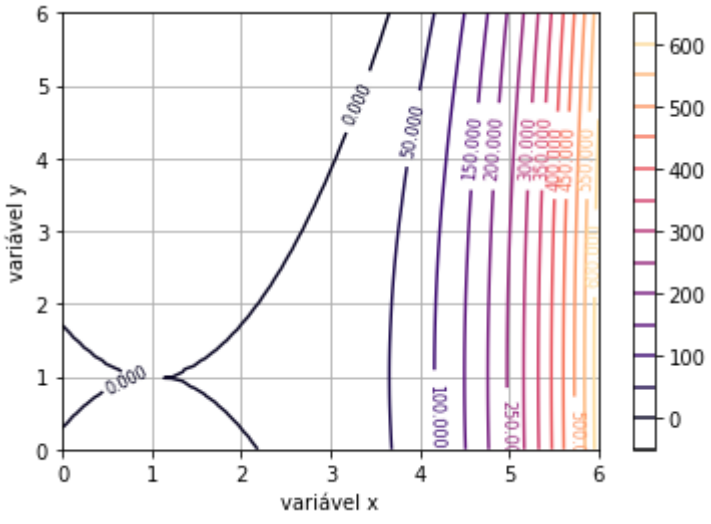
==========

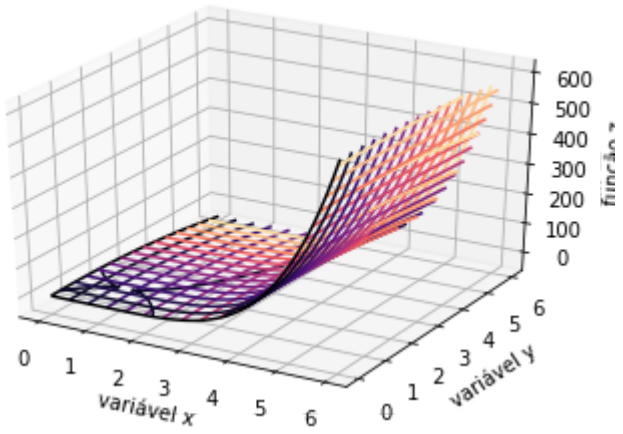
[5.3] Pergunta 5. Versão 3

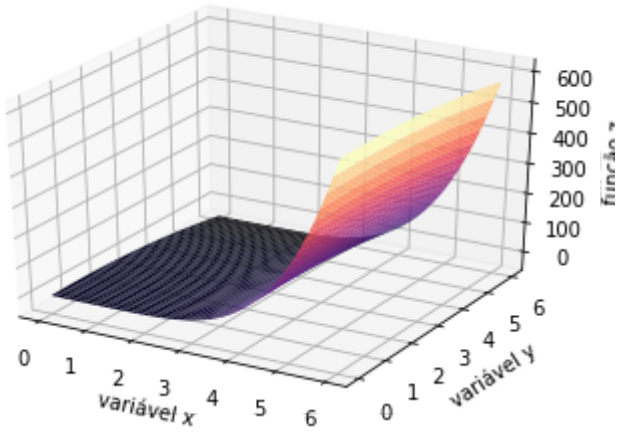
$f(x,y) = x^4 - 4x^3 + 6x^2 - 4x - 2y^2 + 4y - 1$ $\quad (x_{min}, x_{max}) = (0,6);\ (y_{min}, y_{max}) = (0,6)$;
cmap='magma'

In [152]:

```
### 5.3 ###
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D

def fun(x,y):
    return x**4 - 4*x**3 + 6*x**2 -4*x - 2*y**2 + 4*y - 1

xx = np.linspace(0, 6, 50)
yy = np.linspace(0, 6, 50)
X,Y = np.meshgrid(xx,yy)
Z = fun(X,Y)

fig1 = plt.figure()
cs = plt.contour(X,Y,Z, 15, cmap='magma')
plt.clabel(cs, inline=1, fontsize=8)
plt.colorbar(cs)
plt.xlabel('variável x')
plt.ylabel('variável y')
plt.grid()
plt.show()

fig2 = plt.figure()
ax = plt.axes(projection='3d') #projeção 3d
ax.contour(X, Y, Z, 15, zdir='x', cmap='magma')
ax.contour(X, Y, Z, 15, zdir='y', cmap='magma')
ax.contour(X, Y, Z, 15, zdir='z', cmap='magma')
ax.set_xlabel('variável x')
ax.set_ylabel('variável y')
ax.set_zlabel('função z')
plt.show()

fig3 = plt.figure()
ax = plt.axes(projection='3d') #projeção 3d
ax.plot_surface(X, Y, Z, cmap='magma')
ax.set_xlabel('variável x')
ax.set_ylabel('variável y')
ax.set_zlabel('função z')
plt.show()
```







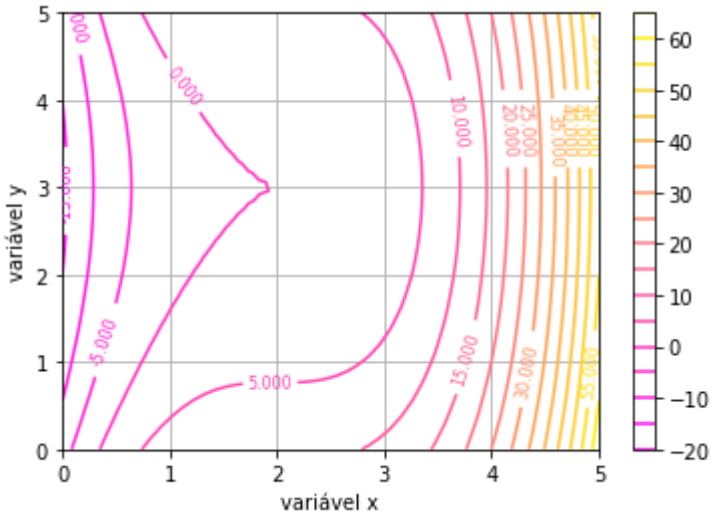
==========

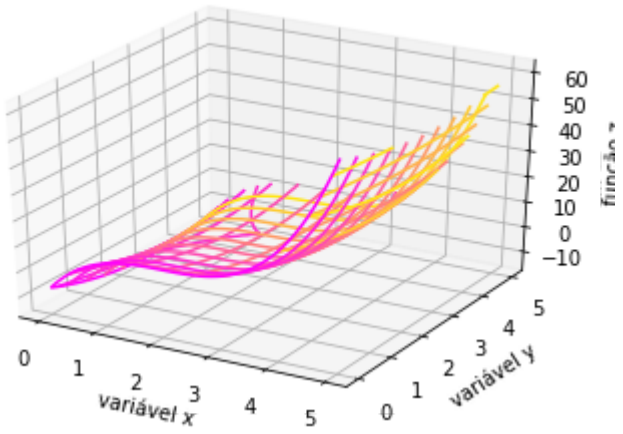
[5.4] Pergunta 5. Versão 4

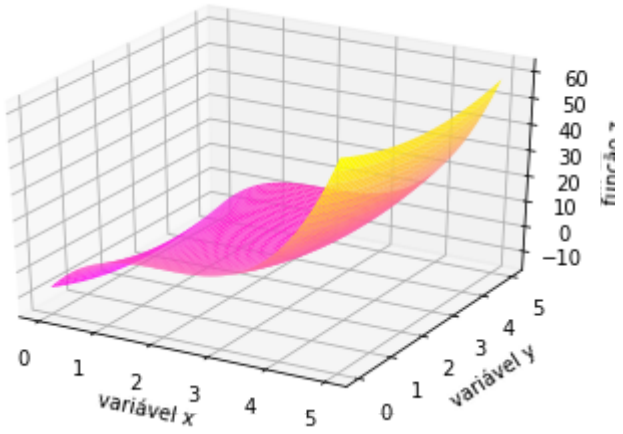
$f(x,y) = 2x^3 - 12x^2 + 24x + y^2 - 6y - 7$ $(x_{min}, x_{max}) = (0,5);\ (y_{min}, y_{max}) = (0,5);$ cmap='spring'

In [145]:

```
### 5.4 ###
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D

def fun(x,y):
    return 2*x**3 - 12*x**2 + 24*x + y**2 - 6*y - 7

xx = np.linspace(0, 5, 50)
yy = np.linspace(0, 5, 50)
X,Y = np.meshgrid(xx,yy)
Z = fun(X,Y)

fig1 = plt.figure()
cs = plt.contour(X,Y,Z,15,cmap='spring')
plt.clabel(cs, inline=1, fontsize=8)
plt.colorbar(cs)
plt.xlabel('variável x')
plt.ylabel('variável y')
plt.grid()
plt.show()

fig2 = plt.figure()
ax = plt.axes(projection='3d') #projeção 3d
ax.contour(X, Y, Z, 10, zdir='x',cmap='spring')
ax.contour(X, Y, Z, 10, zdir='y',cmap='spring')
ax.contour(X, Y, Z, 10, zdir='z',cmap='spring')
ax.set_xlabel('variável x')
ax.set_ylabel('variável y')
ax.set_zlabel('função z')
plt.show()

fig3 = plt.figure()
ax = plt.axes(projection='3d') #projeção 3d
ax.plot_surface(X, Y, Z, cmap='spring')
ax.set_xlabel('variável x')
ax.set_ylabel('variável y')
ax.set_zlabel('função z')
plt.show()
```







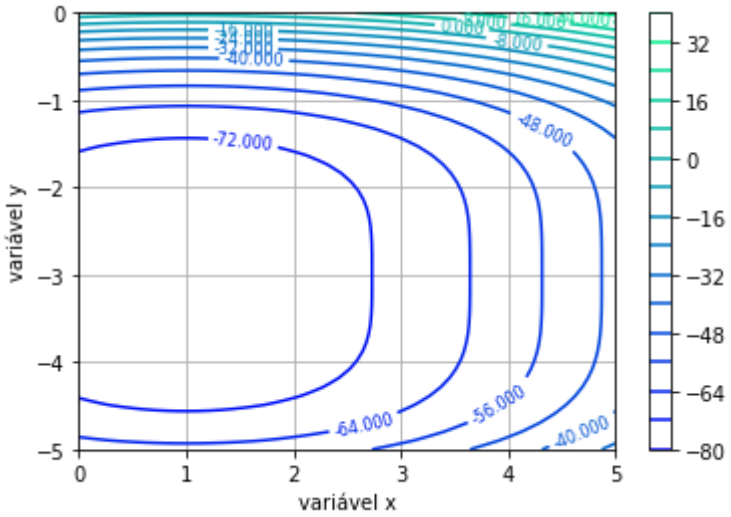
==========

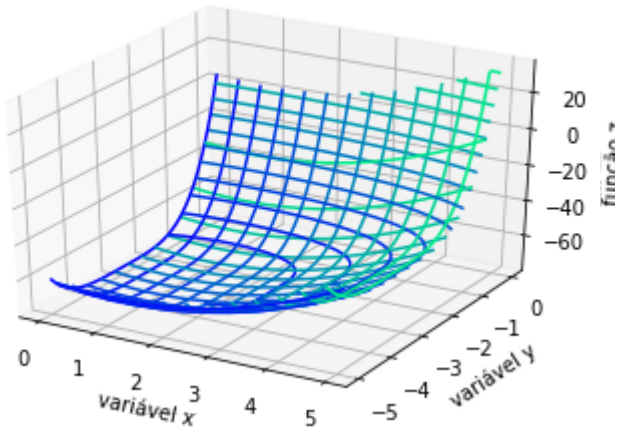
[5.5] Pergunta 5. Versão 5

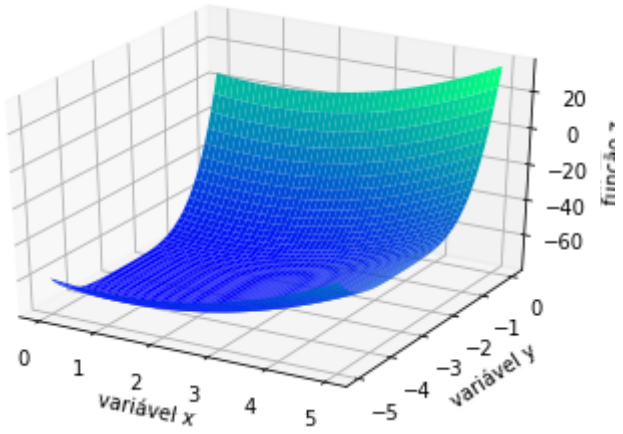
$f(x,y) = 2x^2 - 4x + y^4 + 12y^3 + 54y^2 + 108y + 5 \quad (x_{min}, x_{max}) = (0,5);\ (y_{min}, y_{max}) = (-5,0)$
cmap='winter'

In [147]:

```
### 5.5 ###
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D

def fun(x,y):
    return 2*x**2 - 4*x + y**4 + 12*y**3 + 54*y**2 + 108*y + 5

xx = np.linspace(0,  5, 50)
yy = np.linspace(-5, 0, 50)
X,Y = np.meshgrid(xx,yy)
Z = fun(X,Y)

fig1 = plt.figure()
cs = plt.contour(X,Y,Z,15,cmap='winter')
plt.clabel(cs, inline=1, fontsize=8)
plt.colorbar(cs)
plt.xlabel('variável x')
plt.ylabel('variável y')
plt.grid()
plt.show()

fig2 = plt.figure()
ax = plt.axes(projection='3d') #projeção 3d
ax.contour(X, Y, Z, 15, zdir='x',cmap='winter')
ax.contour(X, Y, Z, 15, zdir='y',cmap='winter')
ax.contour(X, Y, Z, 15, zdir='z',cmap='winter')
ax.set_xlabel('variável x')
ax.set_ylabel('variável y')
ax.set_zlabel('função z')
plt.show()

fig3 = plt.figure()
ax = plt.axes(projection='3d') #projeção 3d
ax.plot_surface(X, Y, Z,cmap='winter')
ax.set_xlabel('variável x')
ax.set_ylabel('variável y')
ax.set_zlabel('função z')
plt.show()
```







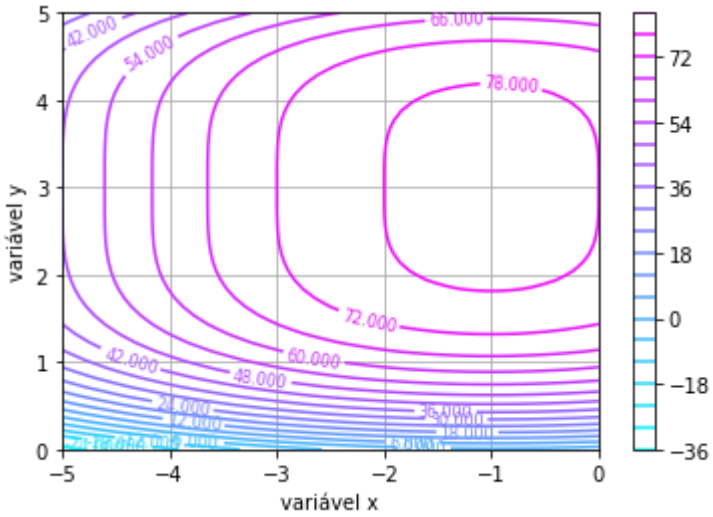
==========

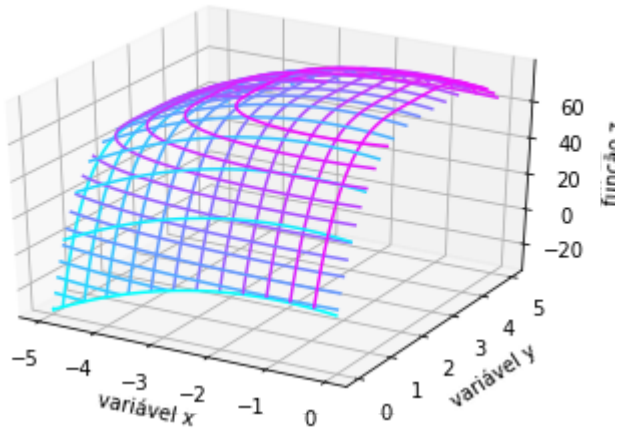
[5.6] Pergunta 5. Versão 6

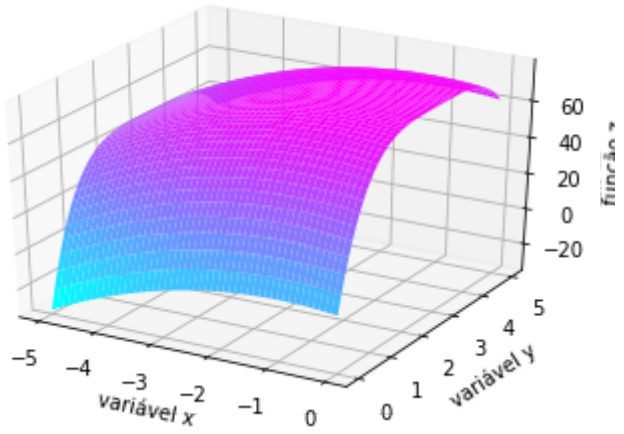
$f(x,y) = -2x^2 - 4x - y^4 + 12y^3 - 54y^2 + 108y - 3 \quad (x_{min}, x_{max}) = (-5, 0);\ (y_{min}, y_{max}) = (0, 5)$
cmap='cool'

In [149]:

```
### 5.6 ###
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D

def fun(x,y):
    return -2*x**2 -4*x - y**4 + 12*y**3 - 54*y**2 + 108*y - 3

xx = np.linspace(-5, 0, 50)
yy = np.linspace(0, 5, 50)
X,Y = np.meshgrid(xx,yy)
Z = fun(X,Y)

fig1 = plt.figure()
cs = plt.contour(X,Y,Z,20,cmap='cool')
plt.clabel(cs, inline=1, fontsize=8)
plt.colorbar(cs)
plt.xlabel('variável x')
plt.ylabel('variável y')
plt.grid()
plt.show()

fig2 = plt.figure()
ax = plt.axes(projection='3d') #projeção 3d
ax.contour(X, Y, Z, 15, zdir='x',cmap='cool')
ax.contour(X, Y, Z, 15, zdir='y',cmap='cool')
ax.contour(X, Y, Z, 15, zdir='z',cmap='cool')
ax.set_xlabel('variável x')
ax.set_ylabel('variável y')
ax.set_zlabel('função z')
plt.show()

fig3 = plt.figure()
ax = plt.axes(projection='3d') #projeção 3d
ax.plot_surface(X, Y, Z,cmap='cool')
ax.set_xlabel('variável x')
ax.set_ylabel('variável y')
ax.set_zlabel('função z')
plt.show()
```







6.

Tarefas:

Calcule analiticamente usando as funções de SymPy.

==========

[6.1] Pergunta 6. Versão 1

A. Avaliar a expressão $\frac{1+x}{\sqrt{x}}$ no ponto $x = 5$ pelas funções `.subs()` e `.n()`.

B. Calcular $\int_0^3 \frac{1+x}{\sqrt{x}}\, dx$

C. Calcular $\lim_{x \to -1} \frac{\cos(x+1) - 1}{x^3 + x^2 - x - 1}$

D. Encontrar o polinómio de Taylor de ordem 8 para $f(x) = x\, cos^2(x)$ na vizinhança do ponto $a = \pi$

In [75]:

```
### 6.1-A ###
from sympy import *
init_printing()

var('x')
expr = (x+1)/sqrt(x)
expr
```

Out[75]:

$$\frac{x+1}{\sqrt{x}}$$

In [76]:

```
exprx = expr.subs({x:5})
exprx
```

Out[76]:

$$\frac{6\sqrt{5}}{5}$$

In [77]:

```
exprx.n(5)
```

Out[77]:

$$2.6833$$

In [78]:

```
### 6.1-B ###
integrate(expr, x)
```

Out[78]:

$$\begin{cases} \frac{2\sqrt{x}(x+1)}{3} + \frac{4\sqrt{x}}{3} & \text{for } |x+1| > 1 \\ \frac{2i\sqrt{-x}(x+1)}{3} + \frac{4i\sqrt{-x}}{3} & \text{otherwise} \end{cases}$$

In [79]:

```
integrate(expr, (x,0,3))
```

Out[79]:

$$4\sqrt{3}$$

In [80]:

```
integrate(expr, (x,0,3)).n(5)
```

Out[80]:

$$6.9282$$

In [81]:

```
### 6.1-C ###
expr2 = (cos(x+1)-1)/(x**3+x**2-x-1)
expr2
```

Out[81]:

$$\frac{\cos(x+1)-1}{x^3+x^2-x-1}$$

In [82]:

```
limit(expr2,x,-1)
```

Out[82]:

$$\frac{1}{4}$$

In [83]:

```
### 6.1-D ###
expr3 = x*(cos(x))**2
expr3
```

Out[83]:

$$x\cos^2(x)$$

In [84]:

```
series(expr3,x,pi,9)
```

Out[84]:

$$-\pi(x-\pi)^2-(x-\pi)^3+\frac{\pi(x-\pi)^4}{3}+\frac{(x-\pi)^5}{3}-\frac{2\pi(x-\pi)^6}{45}-\frac{2(x-\pi)^7}{45}+\frac{\pi(x-}{31}$$

In [86]:

```
series(expr3,x,pi,9).n(5)
```

Out[86]:

$$-3.1416(x-3.1416)^2-(x-3.1416)^3+1.0472(x-3.1416)^4+0.33333(x-3.1416)^5\cdot$$
$$+0.0099733(x-3.1416)^8+x+O\left((x-\pi)^9;\right.$$

==========

[6.2] Pergunta 6. Versão 2

A. Avaliar a expressão $\frac{3x+2}{(x-2)^2(x+2)}$ $x=0$ pelas funções `.subs()` e `.n()` .

B. Calcular $\int \frac{3x+2}{(x-2)^2(x+2)}\,dx$

C. Calcular $\lim_{x\to\frac{\pi}{2}} \frac{\ln(\sin(x))}{(\pi-2x)^2}$

D. Encontrar o polinómio de Taylor de ordem 7 para $f(x)=sin(x^3)$ na vizinhança do ponto $a=\pi/2$

In [87]:

```
### 6.2-A ###
from sympy import *
init_printing()

var('x')
expr = (3*x+2)/(x-2)**2/(x+2)
expr
```

Out[87]:

$$\frac{3x+2}{(x-2)^2\,(x+2)}$$

In [88]:

```
exprx = expr.subs({x:0})
exprx
```

Out[88]:

$$\frac{1}{4}$$

In [89]:

```
exprx.n(5)
```

Out[89]:

$$0.25$$

In [90]:

```
### 6.2-B ###
integrate(expr, x)
```

Out[90]:

$$\frac{\log(x-2)}{4} - \frac{\log(x+2)}{4} - \frac{2}{x-2}$$

In [91]:

```
### 6.2-C ###
expr2 = (log(sin(x)))/(pi-2*x)**2
expr2
```

Out[91]:

$$\frac{\log\left(\sin\left(x\right)\right)}{\left(-2x+\pi\right)^{2}}$$

In [92]:

```
limit(expr2,x,pi/2)
```

Out[92]:

$$-\frac{1}{8}$$

In [93]:

```
### 6.2-D ###
expr3 = sin(x**3)
expr3
```

Out[93]:

$$\sin\left(x^{3}\right)$$

In [94]:

```
series(expr3,x,pi/2,8)
```

Out[94]:

$$\sin\left(\frac{\pi^3}{8}\right) + \frac{3\pi^2\left(x-\frac{\pi}{2}\right)\cos\left(\frac{\pi^3}{8}\right)}{4} + \left(x-\frac{\pi}{2}\right)^2\left(\frac{3\pi\cos\left(\right.\right.}{2}$$

$$+\left(x-\frac{\pi}{2}\right)^3\left(\cos\left(\frac{\pi^3}{8}\right) - \frac{9\pi^3\sin\left(\frac{\pi^3}{8}\right)}{8} - \frac{9\pi^6\cos\left(\frac{\pi^3}{8}\right)}{128}\right) + \left(x-\frac{\pi}{2}\right)^4\left(\frac{27\pi^8\,si}{20}\right.$$

$$+\left(x-\frac{\pi}{2}\right)^5\left(\frac{27\pi^7\sin\left(\frac{\pi^3}{8}\right)}{256} + \frac{81\pi^{10}\cos\left(\frac{\pi^3}{8}\right)}{40960} - \frac{3\pi\sin\left(\right.\right.}{2}$$

$$+\left(x-\frac{\pi}{2}\right)^6\left(\frac{81\pi^9\cos\left(\frac{\pi^3}{8}\right)}{4096} + \frac{99\pi^6\sin\left(\frac{\pi^3}{8}\right)}{256} - \frac{\sin\left(\frac{\pi^3}{8}\right)}{2} - \frac{27\pi^{}}{}\right.$$

$$+\left(x-\frac{\pi}{2}\right)^7\left(\frac{189\pi^8\cos\left(\frac{\pi^3}{8}\right)}{2048} + \frac{27\pi^5\sin\left(\frac{\pi^3}{8}\right)}{32} - \frac{3\pi^2\cos\left(\frac{\pi^3}{8}\right)}{2} - \frac{243\pi^{14}\cos\left(\frac{\pi^3}{8}\right)}{9175040}\right.$$

In [95]:

```
series(expr3,x,pi/2,8).n(4)
```

Out[95]:

$$7.962 + 14.86(x-1.571)^2 + 72.81(x-1.571)^3 + 24.43(x-1.571)^4 - 266.4(x-1.571)$$

$$-5.495x + O\left(\left(x-\frac{\pi}{2}\right)^8; x\to\frac{\pi}{2}\right)$$

==========

[6.3] Pergunta 6. Versão 3

A. Avaliar a expressão $\frac{1}{x^2+4x+4}$ no ponto $x=4$ pelas funções `.subs()` e `.n()`.

B. Calcular $\int_0^1 \frac{dx}{x^2+4x+4}$

C. Calcular $\lim_{x\to 1}\frac{\ln(x)}{x-\sqrt{x}}$

D. Encontrar o polinómio de Taylor de ordem 10 para $f(x) = arctan(x+1)$ na vizinhança do ponto $a=-1$

In [96]:

```
### 6.3-A ###
from sympy import *
init_printing()

var('x')
expr = 1/(x**2+4*x+4)
expr
```

Out[96]:

$$\frac{1}{x^2 + 4x + 4}$$

In [97]:

```
exprx = expr.subs({x:4})
exprx
```

Out[97]:

$$\frac{1}{36}$$

In [98]:

```
exprx.n(5)
```

Out[98]:

$$0.027778$$

In [99]:

```
### 6.3-B ###
integrate(expr, x)
```

Out[99]:

$$-\frac{1}{x + 2}$$

In [100]:

```
integrate(expr, (x,0,1))
```

Out[100]:

$$\frac{1}{6}$$

In [101]:

```
### 6.3-C ###
expr2 = log(x)/(x-sqrt(x))
expr2
```

Out[101]:

$$\frac{\log(x)}{-\sqrt{x}+x}$$

In [102]:

```
limit(expr2,x,1)
```

Out[102]:

$$2$$

In [103]:

```
### 6.3-D ###
expr3 = atan(x+1)
expr3
```

Out[103]:

$$\operatorname{atan}(x+1)$$

In [104]:

```
series(expr3,x,-1,11)
```

Out[104]:

$$1-\frac{(x+1)^3}{3}+\frac{(x+1)^5}{5}-\frac{(x+1)^7}{7}+\frac{(x+1)^9}{9}+x+O\left((x+1)^{11};x\to -1\right)$$

==========

[6.4] Pergunta 6. Versão 4

A. Avaliar a expressão $\sin(\ln(x))$ no ponto $x=5$ pelas funções `.subs()` e `.n()`.

B. Calcular $\int \sin(\ln(x))dx$

C. Calcular $\lim\limits_{x\to-\infty} \frac{\ln(1+\frac{1}{x})}{\sin(\frac{1}{x})}$

D. Encontrar o polinómio de Taylor de ordem 5 para $f(x)=arcsin(x^2)$ na vizinhança do ponto $a=0$

In [105]:

```
### 6.4-A ###
from sympy import *
init_printing()

var('x')
expr = sin(log(x))
expr
```

Out[105]:

$$\sin\left(\log\left(x\right)\right)$$

In [106]:

```
exprx = expr.subs({x:5})
exprx
```

Out[106]:

$$\sin\left(\log\left(5\right)\right)$$

In [107]:

```
exprx.n(5)
```

Out[107]:

$$0.99925$$

In [109]:

```
### 6.4-B ###
integrate(expr, x)
```

Out[109]:

$$\frac{x \sin\left(\log\left(x\right)\right)}{2} - \frac{x \cos\left(\log\left(x\right)\right)}{2}$$

In [110]:

```
### 6.4-C ###
expr2 = log(1+1/x)/sin(1/x)
expr2
```

Out[110]:

$$\frac{\log\left(1 + \frac{1}{x}\right)}{\sin\left(\frac{1}{x}\right)}$$

In [111]:

```
limit(expr2,x,-oo)
```

Out[111]:

$$1$$

In [112]:

```
### 6.4-D ###
expr3 = asin(x**2)
expr3
```

Out[112]:

$$\operatorname{asin}\left(x^{2}\right)$$

In [114]:

```
series(expr3,x,0,6)
```

Out[114]:

$$x^{2} + O\left(x^{6}\right)$$

==========

[6.5] Pergunta 6. Versão 5

A. Avaliar a expressão $\sqrt{e^x - 1}$ no ponto $x = 2$ pelas funções `.subs()` e `.n()`.

B. Calcular $\int_0^{\ln(5)} \sqrt{e^x - 1}\, dx$

C. Calcular $\lim_{x \to 0} \dfrac{x \cos x - \sin x}{x^3}$

D. Encontrar o polinómio de Taylor de ordem 6 para $f(x) = cos(x) + sin(x^2)$ na vizinhança do ponto $a = \pi/2$

In [115]:

```
### 6.5-A ###
from sympy import *
init_printing()

var('x')
expr = sqrt(exp(x)-1)
expr
```

Out[115]:

$$\sqrt{e^{x} - 1}$$

In [116]:

```
exprx = expr.subs({x:2})
exprx
```

Out[116]:

$$\sqrt{-1+e^2}$$

In [117]:

```
exprx.n(5)
```

Out[117]:

$$2.5277$$

In [118]:

```
### 6.5-B ###
integrate(expr, x)
```

Out[118]:

$$2\sqrt{e^x-1}-2\operatorname{atan}\left(\sqrt{e^x-1}\right)$$

In [119]:

```
integrate(expr, (x,0,3))
```

Out[119]:

$$-2\operatorname{atan}\left(\sqrt{-1+e^3}\right)+2\sqrt{-1+e^3}$$

In [120]:

```
integrate(expr, (x,0,3)).n(5)
```

Out[120]:

$$6.0459$$

In [121]:

```
### 6.5-C ###
expr2 = (x*cos(x)-sin(x))/x**3
expr2
limit(expr2,x,0)
```

Out[121]:

$$-\frac{1}{3}$$

In [122]:

```
### 6.5-D ###
expr3 = cos(x)+sin(x**2)
expr3
```

Out[122]:

$$\sin\left(x^2\right) + \cos\left(x\right)$$

In [125]:

```
series(expr3,x,pi/2,7)
```

Out[125]:

$$\sin\left(\frac{\pi^2}{4}\right) + \left(x - \frac{\pi}{2}\right)\left(\pi\cos\left(\frac{\pi^2}{4}\right) - 1\right) + \left(x - \frac{\pi}{2}\right)^2\left(-\frac{\pi^2\sin\left(\frac{\pi^2}{4}\right)}{2} + \cos\left(\frac{\pi^2}{4}\right)\right.$$
$$+ \left(x - \frac{\pi}{2}\right)^4\left(-\frac{\sin\left(\frac{\pi^2}{4}\right)}{2} + \frac{\pi^4\sin\left(\frac{\pi^2}{4}\right)}{24} - \frac{\pi^2\cos\left(\frac{\pi^2}{4}\right)}{2}\right) + \left(x - \frac{\pi}{2}\right)^5\left(\frac{\pi^5\cos\right.$$
$$+ \left(x - \frac{\pi}{2}\right)^6\left(\frac{\pi^4\cos\left(\frac{\pi^2}{4}\right)}{24} - \frac{\pi^6\sin\left(\frac{\pi^2}{4}\right)}{720} - \frac{\cos\left(\frac{\pi^2}{4}\right)}{6} + \frac{\pi^2\sin\left(\right.\right.$$

In [126]:

```
series(expr3,x,pi/2,7).n(5)
```

Out[126]:

$$6.0502 - 3.8618(x - 1.5708)^2 + 2.2425(x - 1.5708)^3 + 6.0767(x - 1.5708)^4 + 2.4526(x$$
$$+ O\left(\left(x - \frac{\pi}{2}\right)^7; x \to \frac{\pi}{2}\right)$$

==========

[6.6] Pergunta 6. Versão 6

A. Avaliar a expressão $\frac{1}{x(1+\ln^2 x)}$ no ponto $x = 2$ pelas funções `.subs()` e `.n()`.

B. Calcular $\int \frac{1}{x(1+\ln^2 x)}\,dx$

C. Calcular $\lim_{x\to 0}(\frac{1}{x} - \frac{1}{e^x - 1})$

D. Encontrar o polinómio de Taylor de ordem 7 para $f(x) = sin(x^3)cos(x)$ na vizinhança do ponto $a = 0$

$$\frac{1}{x(1+\ln^2 x)}$$

In [127]:

```
### 6.6-A ###
from sympy import *
init_printing()

var('x')
expr = 1/(x*(1 + (log(x))**2))
expr
```

Out[127]:

$$\frac{1}{x\left(\log\left(x\right)^2+1\right)}$$

In [128]:

```
exprx = expr.subs({x:2})
exprx
```

Out[128]:

$$\frac{1}{2\left(\log\left(2\right)^2+1\right)}$$

In [129]:

```
exprx.n(5)
```

Out[129]:

$$0.33773$$

In [130]:

```
### 6.6-B ###
integrate(expr, x)
```

Out[130]:

$$\mathrm{RootSum}\left(4z^2+1, \left(i \mapsto i\log\left(2i+\log\left(x\right)\right)\right)\right)$$

In [131]:

```
### 6.6-C ###
expr2 = 1/x -1 /(exp(x)-1)
expr2
```

Out[131]:

$$-\frac{1}{e^x-1}+\frac{1}{x}$$

In [132]:

```
limit(expr2,x,0)
```

Out[132]:

$$\frac{1}{2}$$

In [133]:

```
### 6.6-D ###
expr3 = sin(x**3)*cos(x)
expr3
```

Out[133]:

$$\sin\left(x^3\right)\cos\left(x\right)$$

In [134]:

```
series(expr3,x,0,8)
```

Out[134]:

$$x^3 - \frac{x^5}{2} + \frac{x^7}{24} + O\left(x^8\right)$$

In [135]:

```
series(expr3,x,0,8).n(5)
```

Out[135]:

$$x^3 - 0.5x^5 + 0.041667x^7 + O\left(x^8\right)$$

FIM
