Derivadas

In [2]:
```
# Para imprimir todas las líneas
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
# Solo la última
#InteractiveShell.ast_node_interactivity = "last_expr"
```

Importación de las librerías necesarias

In [3]:
```
import numpy as np
from sympy import * # Librería de Calculo
from sympy.plotting import plot as symplot # Librería para las gráficas
from sympy.abc import x, y, h # Carga de un simbolico "x"
from sympy.plotting.pygletplot import PygletPlot as Plot # Librería para la
s gráficas
```

Límites en Python

El concepto de límite es la base del Cálculo Diferencial.

Recordad que:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h)-f(x)}{h}$$

Podemos calcular límites en Python utilizando la expresión `limit(funcion, variable, punto, lateral)` proporcionada por la librería Sympy.

In [12]:
```
# Ejemplo
# x = symbols('x')
limit(sin(x)/x, x, 0)
limit(1/x, x, 0, '+')
limit(x**2, x, 0, '-')
```

Out[12]: 1

Out[12]: ∞

Out[12]: 0

In [13]:
```
limit((cos(x + h) - cos(x))/h, h, 0) # Cálculo de la derivada a partir de s
u definición
```

Out[13]: $-\sin(x)$

En este caso, se calcula el límite cuando h tiende a cero.

Derivada

Cálculo de la derivada

Para calcular la derivada sin utilizar su definición, se usará `diff("funcion")` .

Ejemplo:

In [14]:
```
diff(cos(x)) # Calcula la derivada que coincide con la obtenida a partir de su definición
```

Out[14]: $-\sin(x)$

In [20]:
```
d1 = diff(x**2, x) # Calcula la derivada de una expresión
d1
diff(x**2)
diff(x**2, y)
```

Out[20]: $2x$

Out[20]: $2x$

Out[20]: 0

In [16]:
```
d1 = (x**2).diff(x) # Otra sintaxis para calcular la derivada
d1
```

Out[16]: $2x$

In [17]:
```
d2 = Derivative(x**2,x) # Expresión matemática de la derivada
d2
```

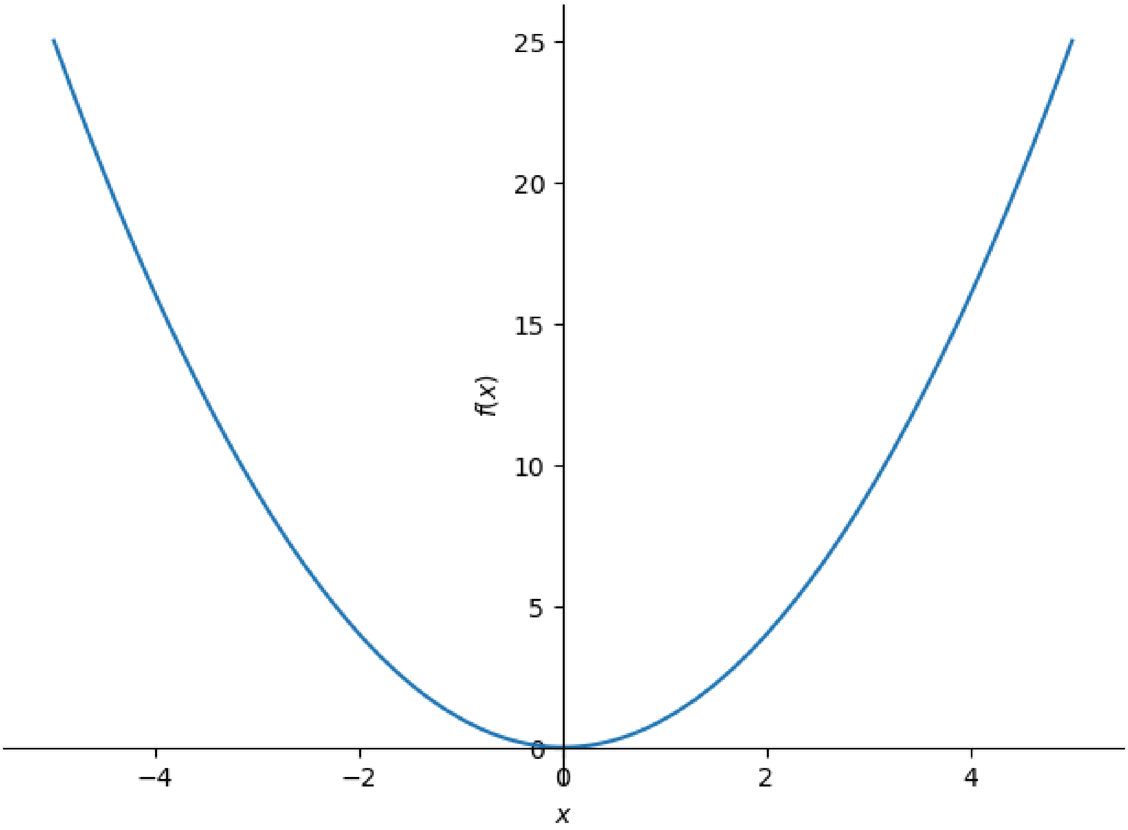
Out[17]: $\frac{d}{dx}x^2$

In [18]:
```
d2.doit() # Calcula la derivada a partir de su expresión matemática
```

Out[18]: $2x$

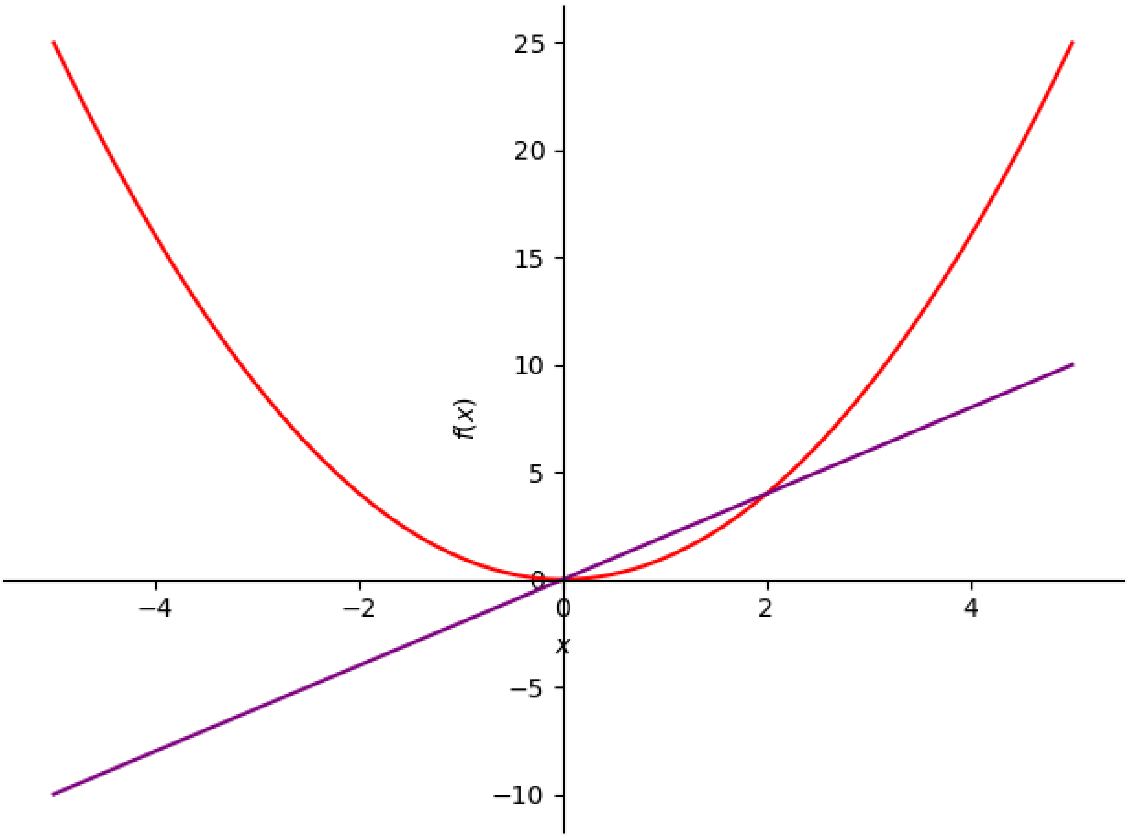
Representación gráfica

In [25]:
```
p = symplot(x**2,(x, -5, 5)) # Representación gráfica de una función matemá
tica
```



Representación gráfica de dos curvas en el mismo gráfico

In [26]:
```
p = symplot(x**2,d1,(x, -5, 5),show=False) # Se crea el gráfico con las dos funciones, el rango de representación y se desactiva la funcionalidad de re presentación directa
p[0].line_color = 'red' # Cambia el color de la primera gráfica
p[1].line_color = 'purple' # Cambia el color de la primera gráfica
p.show() # Mostrar el gráfico
```



Sustitución

Cuando se definen las funciones de manera simbólica, es posible calcular el valor de dicha función para un determinado valor de *x* como sigue:

In [28]:
```
d1.evalf(subs={x:10})
```

Out[28]: 20.0

o

In [29]:
```
d1.subs(x,10)
```

Out[29]: 20

Derivadas Parciales

In [30]:
```
g = x**2 + y**3
```

In [31]:
```
diff(g,x) # Primera derivada parcial con respecto a x
```
Out[31]: $2x$

In [32]:
```
diff(g,y) # Primera derivada parcial con respecto a y
```
Out[32]: $3y^2$

In [33]:
```
# Segunda derivada parcial con respecto a x
diff(g,x,2) # o diff(g,x,x)
```
Out[33]: 2

In [34]:
```
diff(g,y,y) # Segunda derivada parcial con respecto a y
```
Out[34]: $6y$

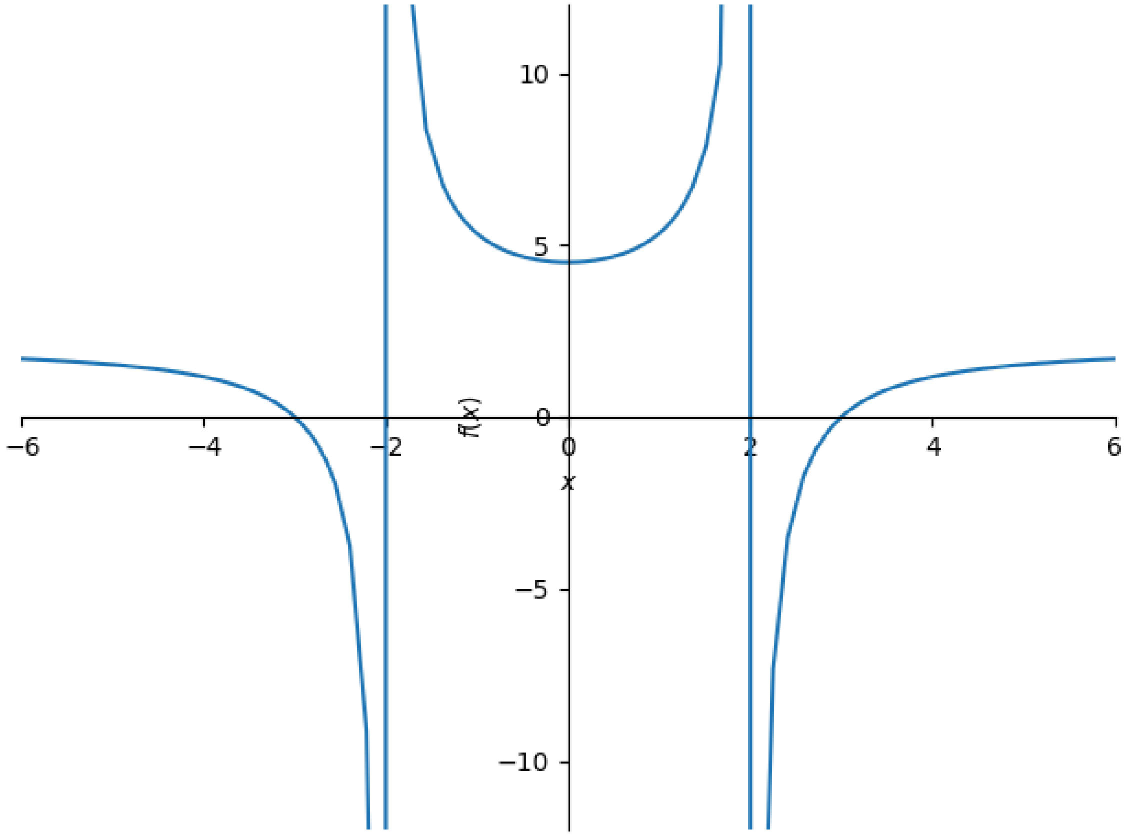
Análisis de funciones

Definición de una función

In [35]:
```
num = 2 * (x**2 - 9)
den = (x**2 - 4)
```

In [36]:
```
f = num / den
f
```
Out[36]: $\frac{2x^2 - 18}{x^2 - 4}$

In [37]:
```
p = symplot(f, xlim=(-6,6), ylim=(-12,12))
```



Asíntotas Horizontales

In [38]:
```
limit(f,x,oo) # oo representa infinito
```

Out[38]: 2

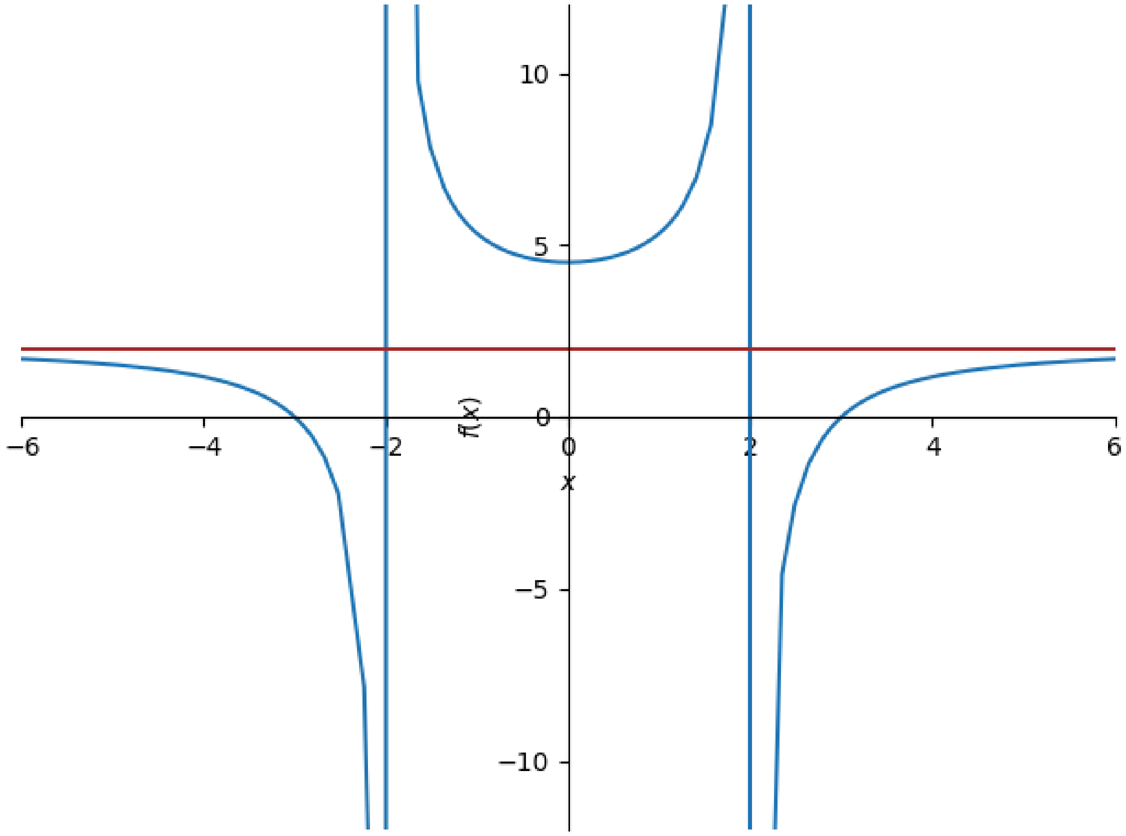
Asíntotas Verticales

Se iguala el denominador a cero y se almacenan las raíces en una variable.

In [39]:
```
r = solve(den, x)
r
```

Out[39]: [-2, 2]

In [46]:
```
# Grafica con la asíntota horizontal
p = symplot(f,2, xlim=(-6,6), ylim=(-12,12),show=false)
p[1].line_color = "brown"
p.show()
```



Puntos críticos

Calculamos la primera derivada e igualamos a cero.

In [47]:
```
f1 = diff(f)
f1
```

Out[47]:

$$\frac{4x}{x^2-4} - \frac{2x\left(2x^2-18\right)}{\left(x^2-4\right)^2}$$

In [48]:
```
criticosX = list(solveset(f1, x))
criticosX
```

Out[48]: [0]

Así pues, en este ejemplo, se tiene un único punto crítico en x = 0

In [49]:
```
criticosY = [f.subs(x, a) for a in criticosX]
criticosY
```

Out[49]: [9/2]

Para ver si es un máximo o un mínimo, se analizará el signo de la segunda derivada.

Segunda derivada

In [50]:
```
f2 = diff (f, x, 2) # Calculamos la segunda derivada
f2
```

Out[50]:

$$\frac{4\left(-\frac{4x^2}{x^2-4}+\frac{(x^2-9)\left(\frac{4x^2}{x^2-4}-1\right)}{x^2-4}+1\right)}{x^2-4}$$

Sustituimos el/los puntos críticos en la segunda derivada

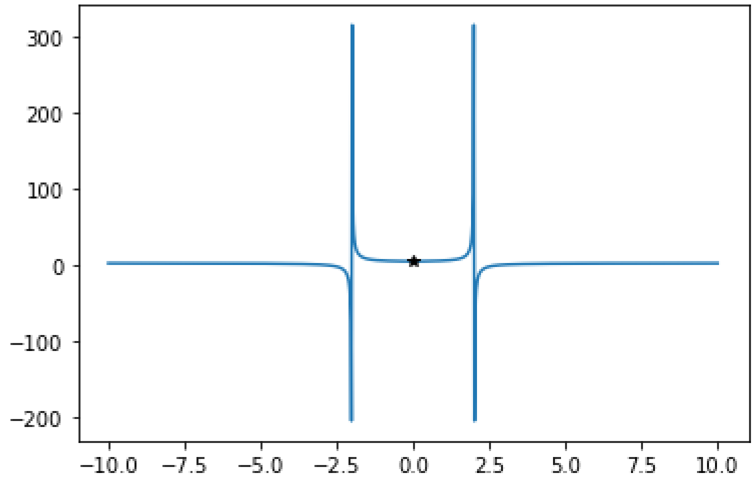
In [51]:
```
segderiv = [f2.subs(x, cr) for cr in criticosX]
segderiv
```

Out[51]: [5/4]

Es positivo, luego hay un mínimo relativo en x = 0

Ahora se representan los puntos críticos

In [30]:
```
import matplotlib.pyplot as plt
xx = np.linspace(-10, 10, 1000)
yy = lambdify(x, f)(xx)
plt.plot(xx, yy)
plt.plot(criticosX, criticosY, 'k*')
plt.show()
```



Optimización

Para resolver un problema de optimización, normalmente se siguen los siguientes pasos:

- Identificar las variables del problema
- Encontrar la función a optimizar y, para el caso de dos variables (x,y), reemplazar `y = f(x)`
- Reducir la función a una única variable independiente
- Comprobar el dominio de admisión de las soluciones y descartar las absurdas
- Calcular el máximo o mínimo de la función objetivo según los requisitos del problema

Ejemplo:

Queremos construir una caja cuya longitud sea tres veces su anchura.
El material usado para construir la tapa y la base cuesta 10 euros por metro cuadrado, mientras que el material usado para construir los lados cuesta 6 euros por metro cuadrado.
Si la caja tiene que tener un volumen de 50 metros cúbicos, determina las dimensiones que minimizan el coste de construir la caja

In [31]:
```
# Hacemos un dibujo del problema
#    +---------+
#   /         /|
#  /         / |  altura
# +---------+  |
# |         |  |
# |         |  +
# |         | /
# |         |/  longitud
# +---------+
#   ancho -> x
```

In [3]:
```
# Queremos construir una caja cuya longitud sea tres veces su anchura.
ancho = x;
longitud = 3*ancho;
```

In [4]:
```
# Si la caja tiene que tener un volumen de 50 metros cúbicos
# Volumen = ancho * longitud * altura
# 50 = ancho * longitud * altura -> altura = 50/(ancho * longitud)
altura = 50/(ancho*longitud)
tapa = longitud*ancho
base = longitud*ancho
Coste1 = (tapa+base)*10
lateral1 = ancho*altura # el lado de la frente
lateral2 = 3*ancho*altura # el lado del lado
Coste2 = (lateral1*2 + lateral2*2 ) * 6 # 2 lados frontales y 2 lados later
ales con un coste de 6
Costetotal = Coste1+Coste2
print('Coste total:', Costetotal)

# Calculamos la primera derivada
derivada=diff(Costetotal)

# Obtenemos los puntos criticos (primera derivada == 0
criticosX = list(N(solveset(derivada, x, domain=S.Reals))) # domain=S.Reals
elimina números complejos del cálculo
criticosX = [cri for cri in criticosX if cri >= 0] # Solo las dimensiones p
ositivas tienen sentido
print("Puntos criticos:", criticosX)

# Resultado del problema
for cri in criticosX:
  print(f'Las dimensiones que minimizan el coste son: Ancho = {cri} m, Long
itud = {longitud.subs(x,cri)} m, Alto = {altura.subs(x,cri)} m')
  print(f'Y su coste total es {Costetotal.subs(x,cri)} €')
```

```
Coste total: 60*x**2 + 800/x
Puntos criticos: [1.88207205776206]
Las dimensiones que minimizan el coste son: Ancho = 1.88207205776206 m, Lon
gitud = 5.64621617328617 m, Alto = 4.70518014440514 m
Y su coste total es 637.595141509567 €
```

Optimización por búsqueda

La librería `scipy` contiene funciones que ayudan a buscar los mínimos de manera automática. No obstante, esta librería y `sympy` son incompatibles, así pues para hacer uso de `scipy` es necesario crear funciones estándar.

In [5]:
```
from scipy import optimize

# en este caso, sympy no puede utilizarse y, por lo tanto, se tiene que def
inir una función estándar manualmente
def f(x):
    return 60*x**2 + 800/x

# Esta expresión calcula el mínimo de una función f a partir de un punto in
icial
# en este caso, "1" es la estimativa inicial
optimize.fmin(f, 1)
```

```
Optimization terminated successfully.
         Current function value: 637.595142
         Iterations: 16
         Function evaluations: 32
```

Out[5]:
```
array([1.88203125])
```

In [35]:
```
optimize.fminbound(f, -1, 2) # Igual que la anterior pero se tienen que ind
icar los puntos inicial y final de búsqueda
```

Out[35]:
```
1.8820717255379549
```

Ejercicios

Ejercicio 1

Calcula el límite de las siguiente funciones y dibuja las mismas:

$\lim_{x \to 0} \frac{x^3+5}{x^4+7}$

$\lim_{x \to 3+} \frac{x-3}{|x-3|}$

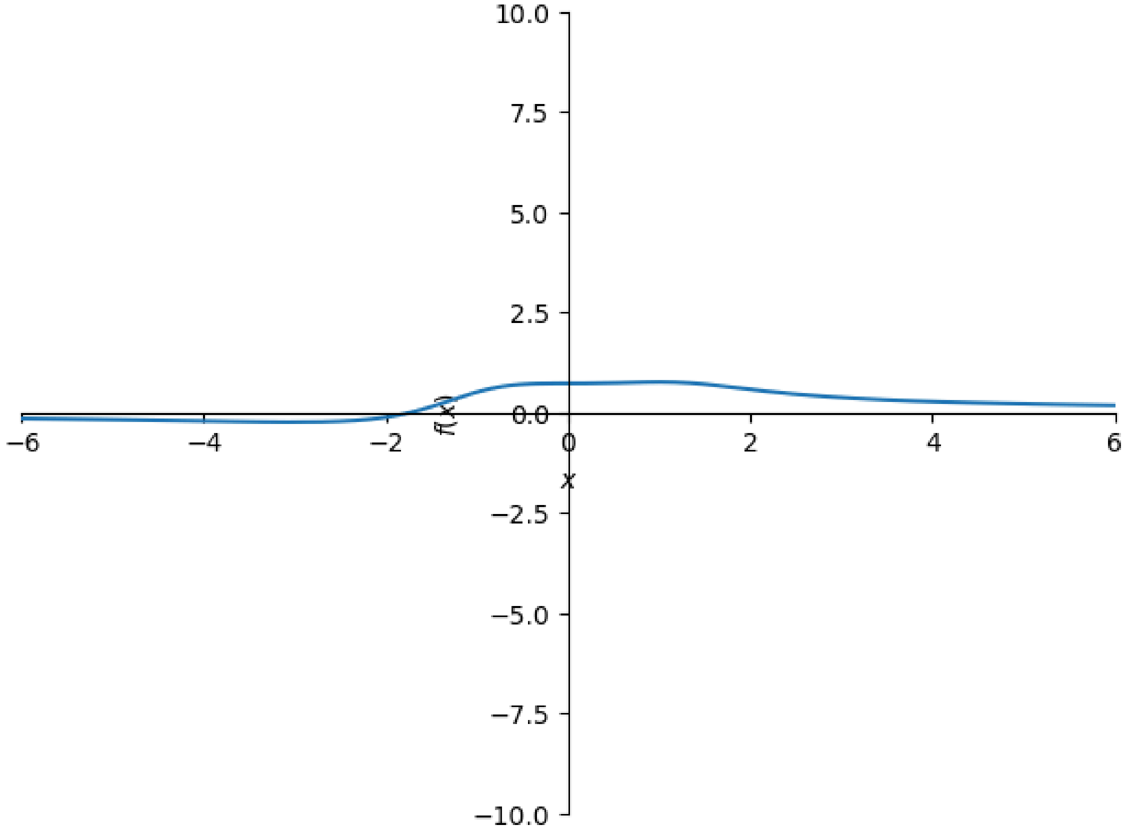
$\lim_{x \to 3-} \frac{x-3}{|x-3|}$

In [21]:
```
f1_ej1 = (x**3 + 5)/(x**4+7)
f2_ej1 = (x-3)/abs((x-3))
f3_ej1 = (x-3)/abs((x-3))

limit(f1_ej1,x,0)
limit(f2_ej1,x,3,'+')
limit(f3_ej1,x,3,'-')
# p = symplot(f,2, xlim=(-6,6), ylim=(-12,12),show=false)
symplot(f1_ej1, xlim=(-6, 6), ylim=(-10, 10))
symplot(f2_ej1, xlim=(-6, 6), ylim=(-10, 10))
```

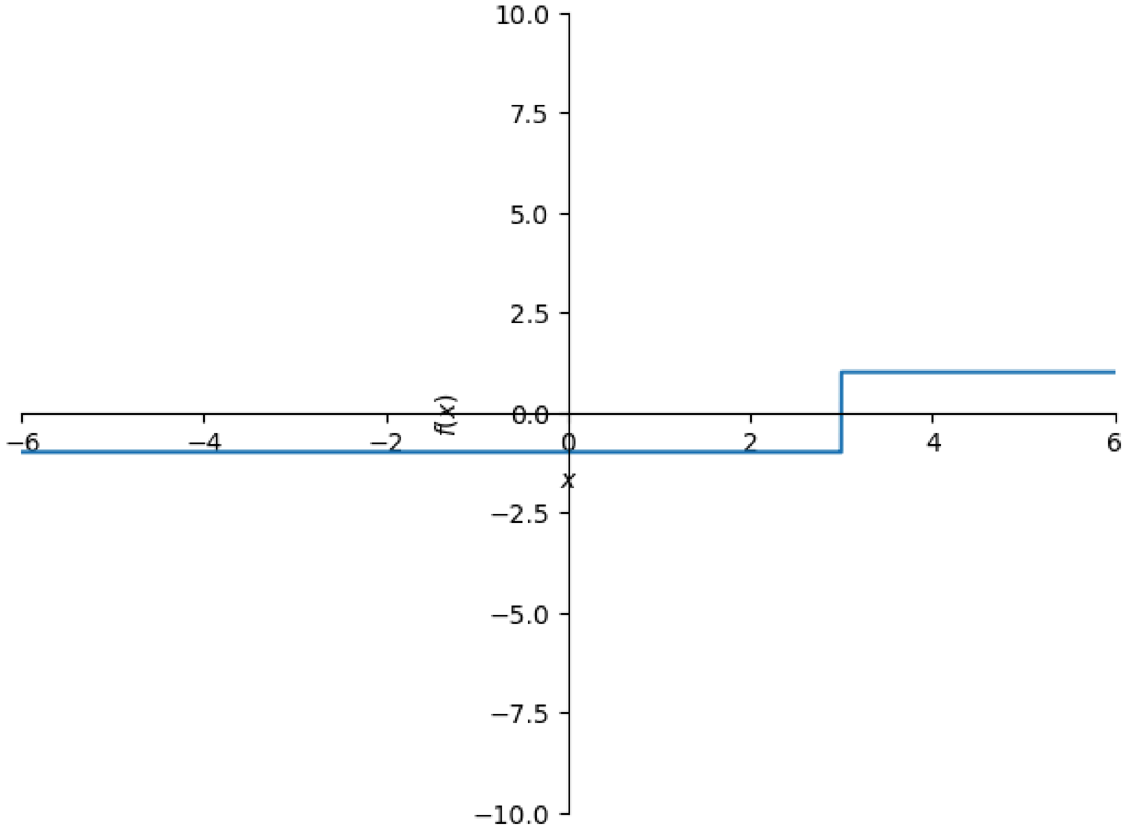
Out[21]: $\frac{5}{7}$

Out[21]: 1

Out[21]: -1



Out[21]: <sympy.plotting.plot.Plot at 0x2086eccf010>

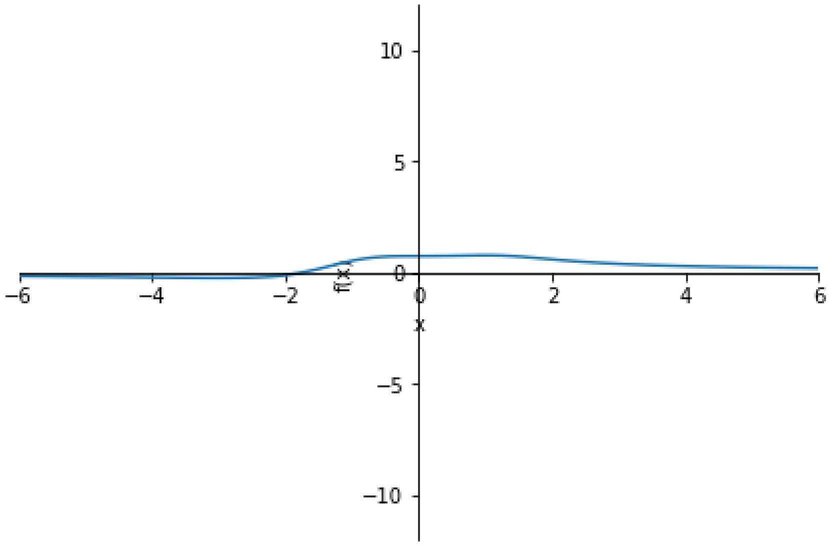


Out[21]: <sympy.plotting.plot.Plot at 0x2086ed122d0>

In [36]:

Out[36]: $\frac{5}{7}$

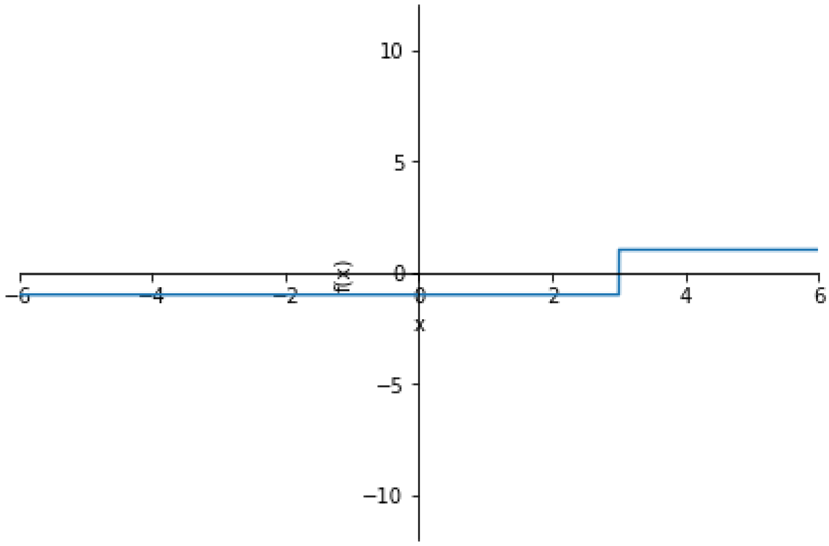
In [37]:



In [38]:

Out[38]: 1

In [39]:



In [40]:

Out[40]: -1

Ejercicio 2

Sabiendo que $f(x) = \frac{3x+5}{x-3}$ y $g(x) = x^2 + 1$

Calcula los siguientes límites:

$$l1 = \lim_{x \to 4} \ f(x)$$

$$l2 = \lim_{x \to 4} \ g(x)$$

$$lsum = \lim_{x \to 4} \ (f(x) + g(x))$$

$$lrest = \lim_{x \to 4} \ (f(x) - g(x))$$

$$lprod = \lim_{x \to 4} \ (f(x) * g(x))$$

$$ldiv = \lim_{x \to 4} \ \frac{f(x)}{g(x)}$$

In [41]:
```
#l1
```

Out[41]: 17

In [26]:
```
f_ej2 = (3*x+5)/(x-3)
g_ej2 = x**2 + 1

l1_ej2 = limit(f_ej2, x, 4)
l1_ej2
```

Out[26]: 17

In [42]:
```
#l2
```

Out[42]: 17

In [27]:
```
l2_ej2 = limit(g_ej2, x, 4)
l2_ej2
```

Out[27]: 17

In [43]:
```
#lsum
```

Out[43]: 34

In [29]:
```
lsum_ej2 = limit(f_ej2 + g_ej2, x, 4)
lsum_ej2
```

Out[29]: 34

In [44]:
```
#lrest
```

Out[44]: 0

In [30]:
```
lrest_ej2 = limit(f_ej2 - g_ej2, x, 4)
lrest_ej2
```
Out[30]: 0

In [45]:
```
#lprod
```
Out[45]: 289

In [31]:
```
lprod_ej2 = limit(f_ej2 * g_ej2, x, 4)
lprod_ej2
```
Out[31]: 289

In [46]:
```
#ldiv
```
Out[46]: 1

In [32]:
```
ldiv_ej2 = limit(f_ej2 / g_ej2, x, 4)
ldiv_ej2
```
Out[32]: 1

Ejercicio 3

Calcula los límites iterados de las siguientes funciones:

$$\lim_{(x,y)\to(4,3)} \frac{x^2-1}{3x+y}$$

$$\lim_{(x,y)\to(2,2)} \frac{x+3}{xy-4}$$

In [47]:
```
# Primera
#lxy
```
Out[47]: 1

In [33]:
```
f1_ej3 = (x**2-1)/(3*x+y)
limit(limit(f1_ej3, x, 4), y, 3)
```
Out[33]: 1

In [48]:
```
#lyx
```
Out[48]: 1

In [34]:
```
limit(limit(f1_ej3, y, 3), x, 4)
```
Out[34]: 1

In [49]:
```
# Segunda
#lxy
```
Out[49]: ∞

In [35]:
```
f2_ej3 = (x+3)/(x*y-4)
limit(limit(f2_ej3, x, 2), y, 2)
```

Out[35]: ∞

In [50]:
```
#Lyx
```

Out[50]: ∞

In [36]:
```
limit(limit(f2_ej3, y, 2), x, 2)
```

Out[36]: ∞

Ejercicio 4

Calcula los límites iterados de la siguiente función y dibuja la función

$$\lim_{(x,y)\to(0,0)} \frac{xy}{x^2+y^2}$$

In [9]:
```
f_ej4 = (x*y)/(x**2+y**2)
```

In [51]:
```
#Lxy
```

Out[51]: 0

In [39]:
```
limit(limit(f_ej4, x, 0), y, 0)
```

Out[39]: 0

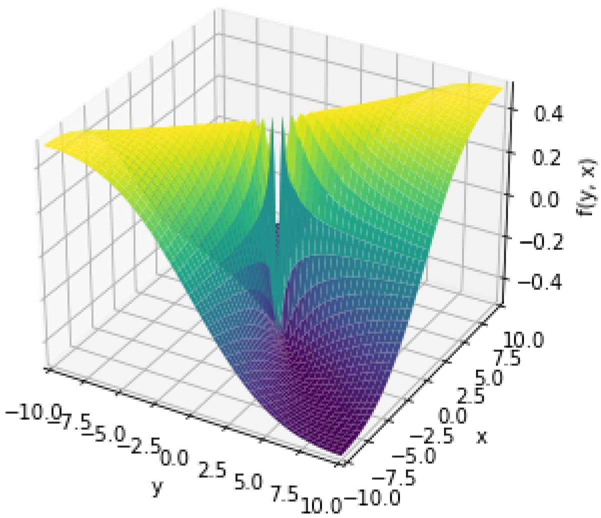
In [52]:
```
#Lyx
```

Out[52]: 0

In [40]:
```
limit(limit(f_ej4, y, 0), x, 0)
```

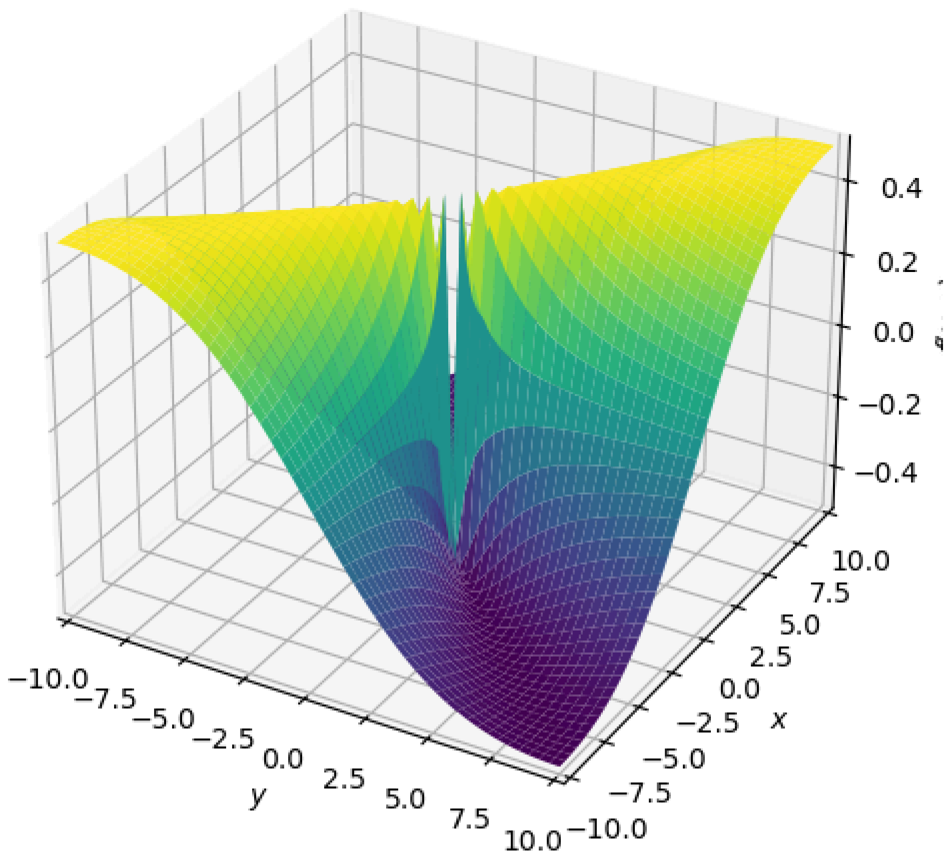
Out[40]: 0

In [53]:
```
#gráfica
```



Out[53]: <sympy.plotting.plot.Plot at 0x7f534cd9d420>

In [10]:
```
from sympy.plotting import plot3d
plot3d(f_ej4)
```



Out[10]: <sympy.plotting.plot.Plot at 0x2c3b0b84290>

Ejercicio 5

Obtén la derivada de la función $sin(x)$ recurriendo a la definición de derivada y utilizando el comando limit

In [54]:

Out[54]: $\cos(x)$

In [12]:
```
limit((sin(x + h)-sin(x))/h, h, 0)
```

Out[12]: $\cos(x)$

Ejercicio 6

Obtén la derivada de la matriz:

$$\begin{bmatrix} cos(4x) & 3x \\ x & sen(5x) \end{bmatrix}$$

Verfica información sobre matrices en [https://docs.sympy.org/latest/modules/matrices/matrices.html#creating-matrices (https://docs.sympy.org/latest/modules/matrices/matrices.html#creating-matrices)]

In [22]:
```
matriz_ej6 = Matrix([[cos(4*x), 3*x],[x, sin(5*x)]])
```

In [55]:

Out[55]: $\begin{bmatrix} -4\sin(4x) & 3 \\ 1 & 5\cos(5x) \end{bmatrix}$

In [23]:
```
diff(matriz_ej6)
```

Out[23]: $\begin{bmatrix} -4\sin(4x) & 3 \\ 1 & 5\cos(5x) \end{bmatrix}$

Ejercicio 7

Calcula la primera derivada parcial respecto a x de las siguientes expresiones:

$tan(x + y)$

$ay + bx + cz$

$x^{0.5} - 3y$

In [34]:
```
a = symbols('a')
b = symbols('b')
c = symbols('c')
z = symbols('z')

f1_ej7 = tan(x + y)
f2_ej7 = a*y + b*x + c*z
f3_ej7 = x**0.5 -3*y
```

In [56]:
```
```

Out[56]: $\tan^2(x+y)+1$

In [31]:
```
diff(f1_ej7, x)
```

Out[31]: $\tan^2(x+y)+1$

In [57]:
```
```

Out[57]: b

In [32]:
```
diff(f2_ej7, x)
```

Out[32]: b

In [58]:
```
```

Out[58]: $\frac{0.5}{x^{0.5}}$

In [35]:
```
diff(f3_ej7, x)
```

Out[35]: $\frac{0.5}{x^{0.5}}$

Ejercicio 8

Analiza las siguientes expresiones. Crea una función que al introducir las expresiones cree un gráfico con la función junto con sus raíces, asíntotas y puntos críticos.

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

$\frac{log(x)}{x}$

$\frac{x+1}{\sqrt{x-1}-5}$

$\frac{x^3}{(x-1)^2}-8$

In [3]:
```
f1_ej8 = (2*x)/(x**2+1)
f2_ej8 = log(x)/x
f3_ej8 = (x+1)/(sqrt(x-1)-5)
f4_ej8 = x**3/(x-1)**2 -8
```

In [32]:
```
import matplotlib.pyplot as plt

def ejercicio8(funcion):
    print("Original:")
    display(funcion)

    asintotas_horizontales = limit(funcion, x, oo)
    print(f"Asíntotas horizontales: {asintotas_horizontales}")

    asintotas_verticales = list(N(solveset(1/funcion, x, domain=S.Reals)))
    print(f"Asíntotas verticales: {asintotas_verticales}")
    print("Derivada:")
    display(diff(funcion))
    criticosX = solve(diff(funcion))
    criticosY = [funcion.subs(x, a) for a in criticosX]
    print(f"Puntos críticos: {criticosX}")
    minimos = [elem for elem in criticosX if diff(funcion, x, 2).subs(x, el
em) > 0]

    for elem in criticosX:
        print(f"Hay un minimo en ({elem},{funcion.subs(x, elem)})")

    xx = np.linspace(-5, 5, 1000)
    yy = lambdify(x, funcion)(xx)
    plt.plot(0, 0, 'ro', markersize=10)
    plt.plot(xx, yy)

    for asintota_vertical in asintotas_verticales:
        plt.axvline(x=asintota_vertical, color='r', linestyle='--', label
='Asíntotas verticales')

    # Asegúrate de que asintotas_horizontales es un número real antes de tr
azarla
    if asintotas_horizontales.is_real:
        plt.axhline(y=asintotas_horizontales, color='g', linestyle='-', lab
el='Asíntotas horizontales')

    plt.plot(criticosX, criticosY, 'k*')
    plt.ylim(float(min(criticosY)-1),float(max(criticosY)+1))
    plt.show()

ejercicio8(f1_ej8)
```

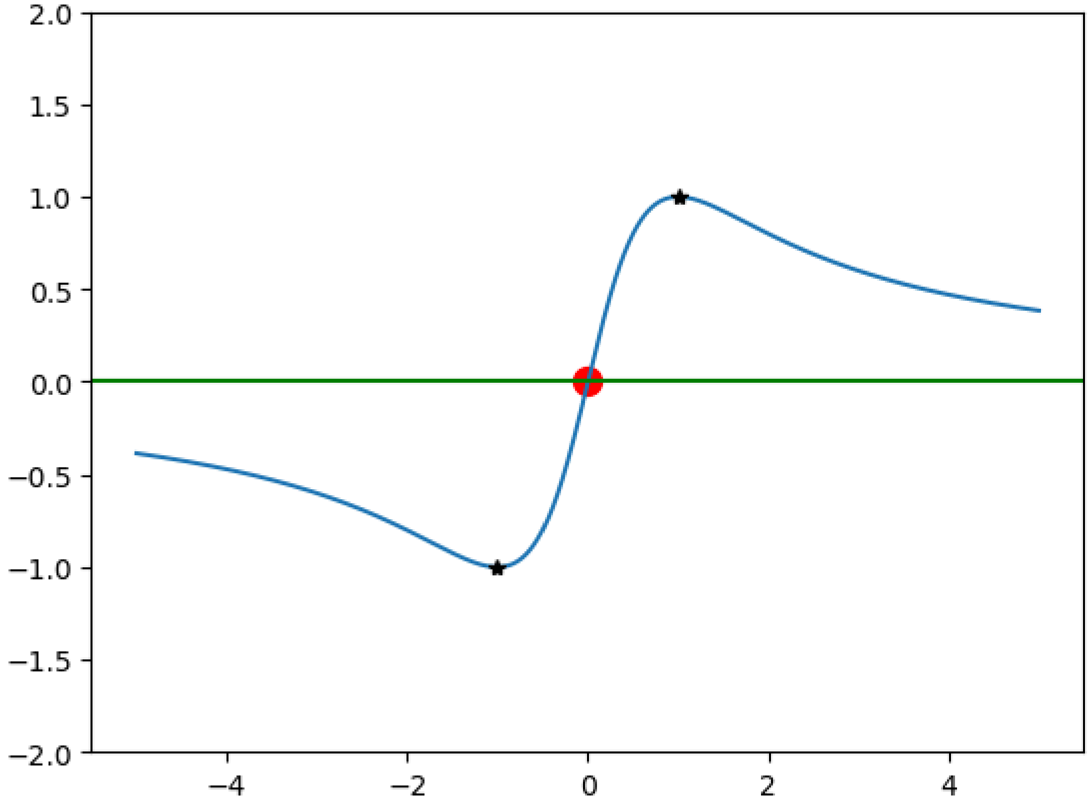
```
Original:
```

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

```
Asíntotas horizontales: 0
Asíntotas verticales: []
Derivada:
```

$$-\frac{4x^2}{(x^2 + 1)^2} + \frac{2}{x^2 + 1}$$

```
Puntos críticos: [-1, 1]
Hay un minimo en (-1,-1)
Hay un minimo en (1,1)
```



In [60]:

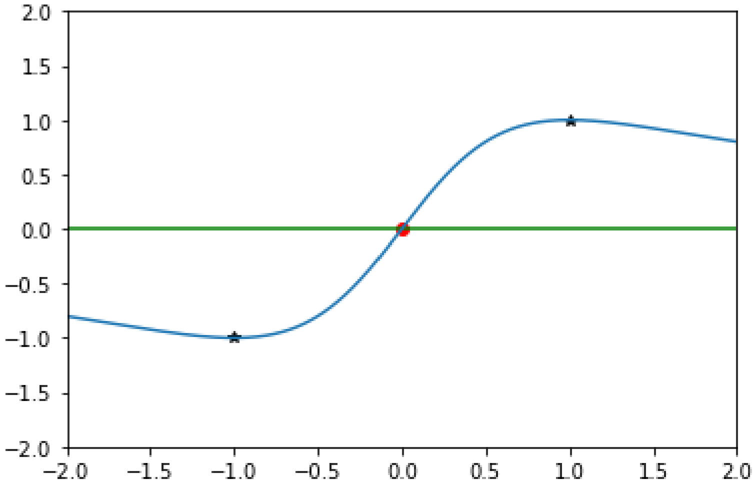
```
Original:
```

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

```
Asíntotas horizontales: 0
Asíntotas verticales: []
Derivada:
```

$$-\frac{4x^2}{\left(x^2 + 1\right)^2} + \frac{2}{x^2 + 1}$$

```
Puntos críticos: [-1, 1]
Hay un mínimo en (-1,-1)
Hay un mínimo en (1,1)
```



In [33]:
```
ejercicio8(f2_ej8)
```

```
Original:
```

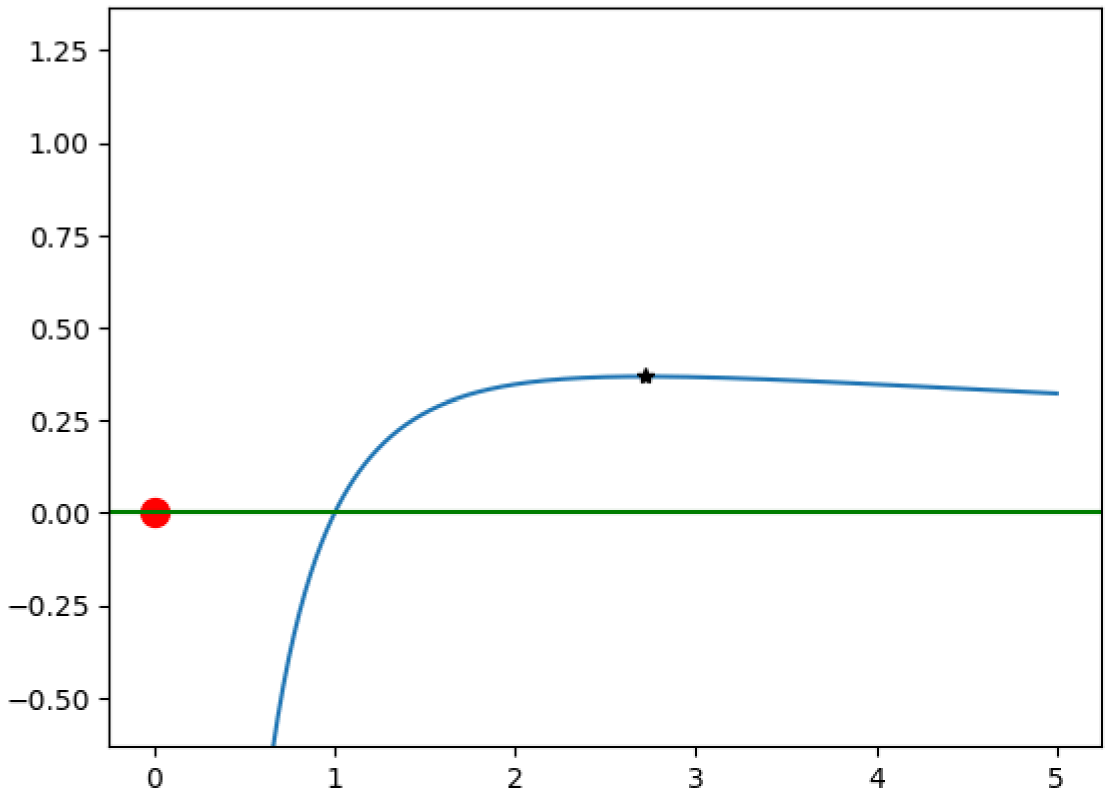
$$\frac{\log(x)}{x}$$

```
Asíntotas horizontales: 0
Asíntotas verticales: []
Derivada:
```

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

```
Puntos críticos: [E]
Hay un minimo en (E,exp(-1))
```

```
<lambdifygenerated-14>:2: RuntimeWarning: invalid value encountered in log
  return log(x)/x
```



In [61]:

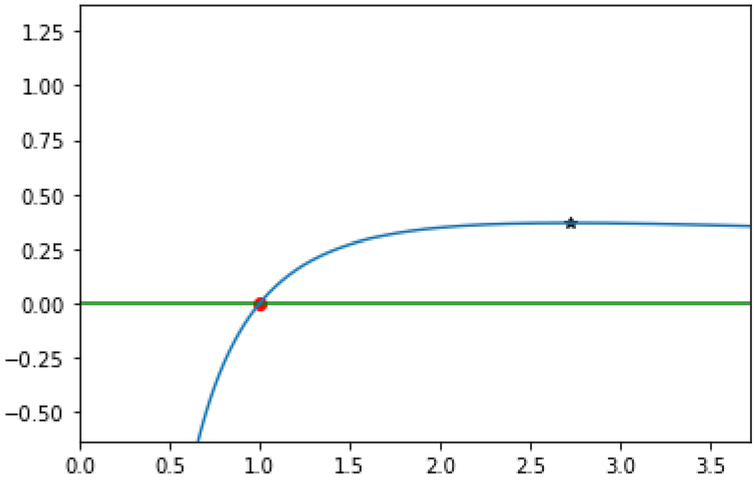
```
Original:
```

$$\frac{\log(x)}{x}$$

```
Asíntotas horizontales: 0
Asíntotas verticales: [0]
Derivada:
```

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

```
Puntos críticos: [E]
Hay un mínimo en (E,exp(-1))
```



In [35]:
```
ejercicio8(f3_ej8)
```

Original:

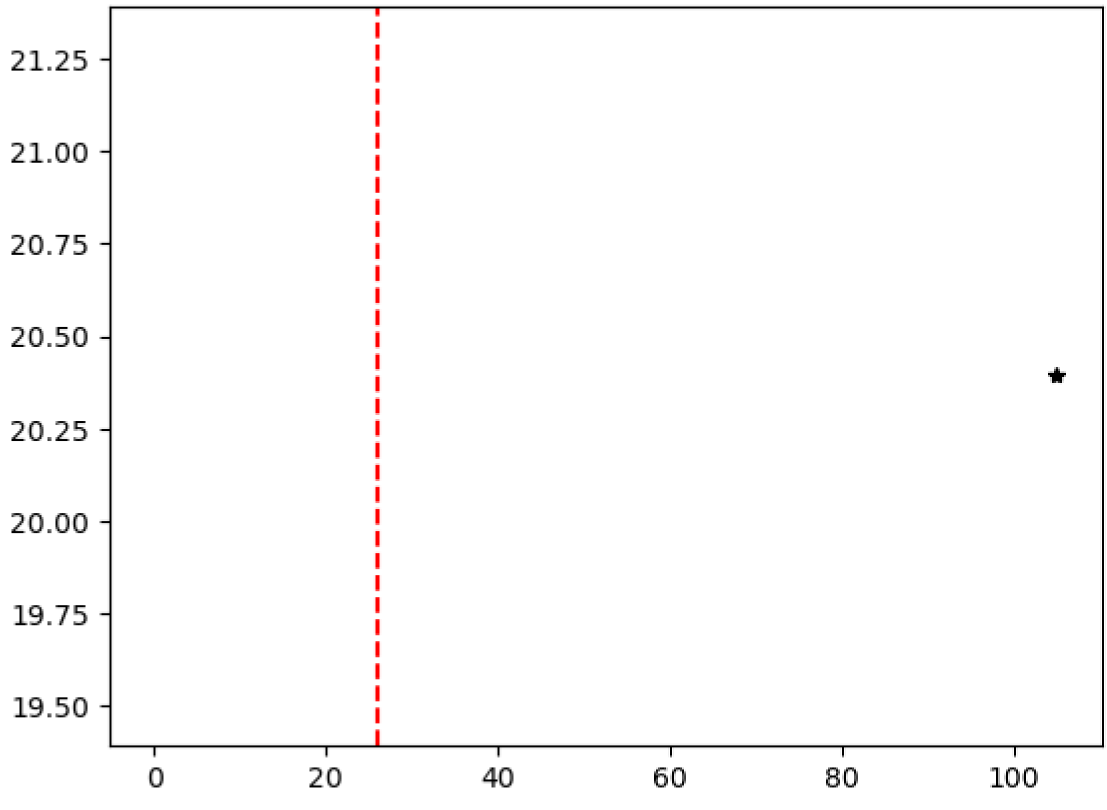
$$\frac{x+1}{\sqrt{x-1}-5}$$

```
Asíntotas horizontales: oo
Asíntotas verticales: [26.0000000000000]
Derivada:
```

$$\frac{1}{\sqrt{x-1}-5} - \frac{x+1}{2\sqrt{x-1}\left(\sqrt{x-1}-5\right)^{2}}$$

```
Puntos críticos: [30*sqrt(3) + 53]
Hay un minimo en (30*sqrt(3) + 53,(30*sqrt(3) + 54)/(-5 + sqrt(30*sqrt(3) + 52)))
```

```
<lambdifygenerated-16>:2: RuntimeWarning: invalid value encountered in sqrt
  return (x + 1)/(sqrt(x - 1) - 5)
```



In [62]:

```
Original:
```

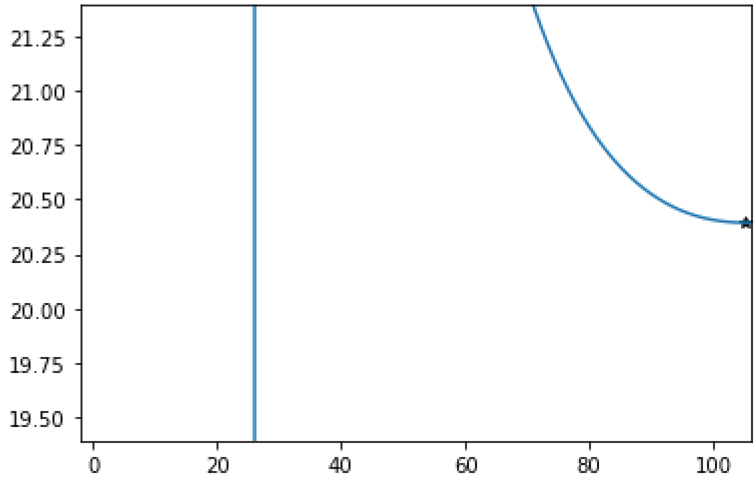
$$\frac{x+1}{\sqrt{x-1}-5}$$

```
Asíntotas horizontales: oo
Asíntotas verticales: [26]
Derivada:
```

$$\frac{1}{\sqrt{x-1}-5} - \frac{x+1}{2\sqrt{x-1}\left(\sqrt{x-1}-5\right)^2}$$

```
Puntos críticos: [30*sqrt(3) + 53]
Hay un máximo en (30*sqrt(3) + 53,(30*sqrt(3) + 54)/(-5 + sqrt(30*sqrt(3) + 52)))
```

```
<lambdifygenerated-4>:2: RuntimeWarning: invalid value encountered in sqrt
  return (x + 1)/(sqrt(x - 1) - 5)
```



In [36]:
```
ejercicio8(f4_ej8)
```

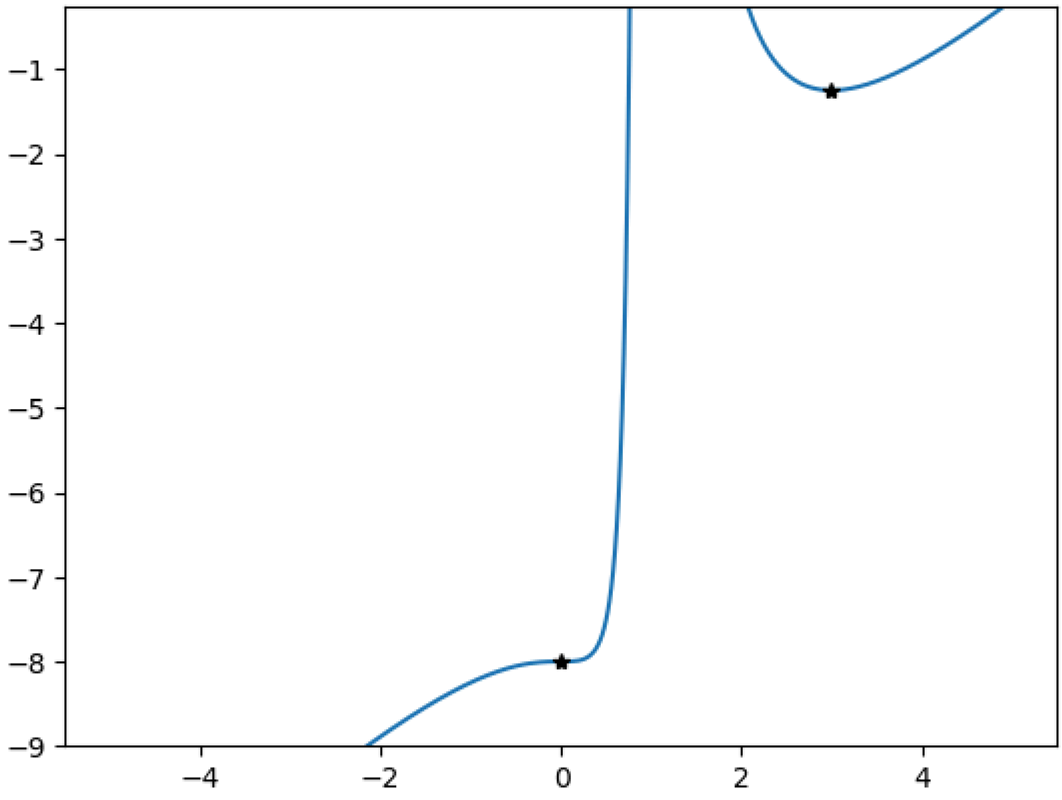
```
Original:
```

$$\frac{x^3}{(x-1)^2} - 8$$

```
Asíntotas horizontales: oo
Asíntotas verticales: []
Derivada:
```

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

```
Puntos críticos: [0, 3]
Hay un minimo en (0,-8)
Hay un minimo en (3,-5/4)
```



In [63]:

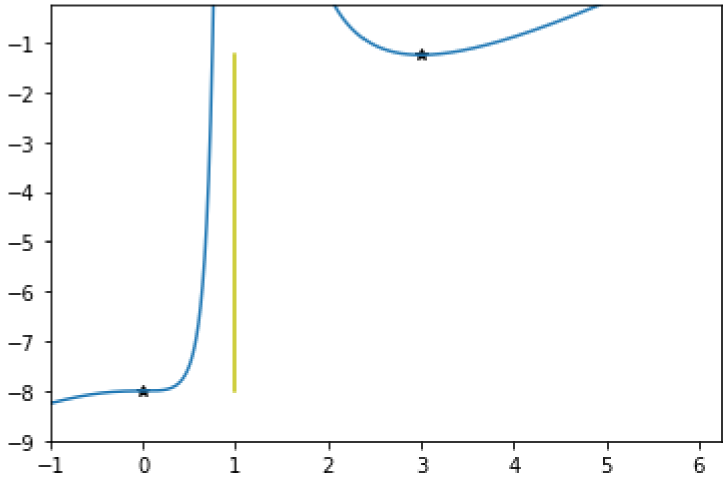
```
Original:
```

$$\frac{x^3}{(x-1)^2} - 8$$

```
Asíntotas horizontales: oo
Asíntotas verticales: [1]
Derivada:
```

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

```
Puntos críticos: [0, 3]
Hay un mínimo en (0,-8)
Hay un máximo en (3,-5/4)
```



Ejercicio 9

Una ventana se construye en su parte superior con un semicírculo y en su parte inferior con un rectángulo. Si hay 12m de materiales, ¿cuáles serán las dimensiones de la ventana para que entre la mayor cantidad de luz?

In [43]:
```
# radio es x
# Perimetro semicirculo
Perimetro_semi = x * (2 + pi)
# Perimetro rectangulo; y es la altura
Perimetro_rect = ((x * 2) + y) * 2

# El perimetro total es 12, así que hay que resolver los perimetros para es
e valor
# para no borrar el y (que es simbolico) se crea una nueva variable
# Hay que restar la parte de bajo del semicirculo y la de arriba del rectan
gulo (porque están unidas) -> x*2
nuevo_y = solve((Perimetro_semi - x * 2) + (Perimetro_rect - x*2) - 12, y)
[0]
print("nuevo_y:")
display(nuevo_y)

# Area de semicirculo
Area_semi = (pi * (x ** 2))/2
# area rectangulo
Area_rectangulo = (x * 2) * nuevo_y
# Total de area que se va a minimizar
Area_total = Area_rectangulo + Area_semi
# Hallar la derivada
derivada_Area = diff(Area_total,x)
# Hallar los criticos -> correspondiente al radio (x)
criticos = solve(derivada_Area)
# Hallar los correspondientes valores de y de los criticos
criticosY = [Area_total.subs(x,crit) for crit in criticos]
# Dibujar
import matplotlib.pyplot as plt
plt.plot(criticos, criticosY, 'k*')
plt.plot(np.linspace(float(min(criticos))-1, float(max(criticos))+1, 100),
[Area_total.subs(x,crit) for crit in np.linspace(float(min(criticos))-1, fl
oat(max(criticos))+1, 100)])

print("Puntos criticos:", criticos)
print("Anchura optima:", criticos[0]*2)
print("Altura optima:", nuevo_y.subs(x,criticos[0]))
```

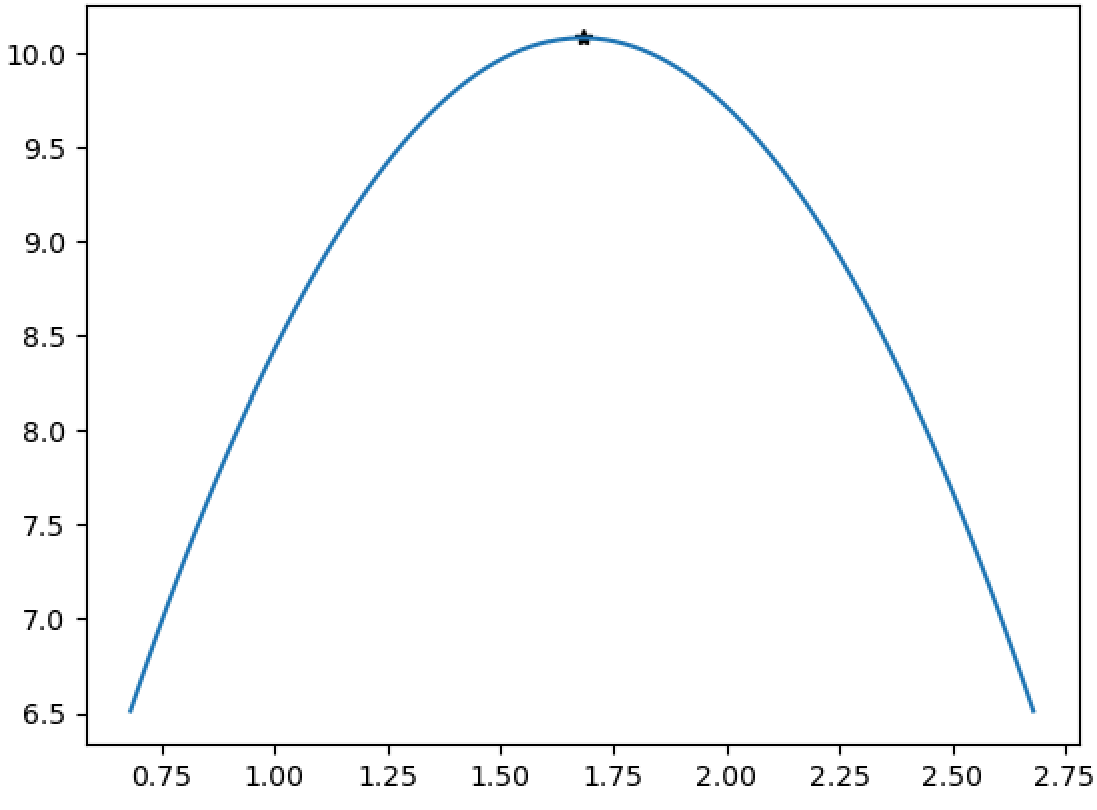
nuevo_y:

$$-\frac{\pi x}{2} - x + 6$$

Out[43]:
```
[<matplotlib.lines.Line2D at 0x2904ecc3c50>]
```

Out[43]:
```
[<matplotlib.lines.Line2D at 0x2904eafd510>]
```

```
Puntos criticos: [12/(pi + 4)]
Anchura optima: 24/(pi + 4)
Altura optima: -6*pi/(pi + 4) - 12/(pi + 4) + 6
```

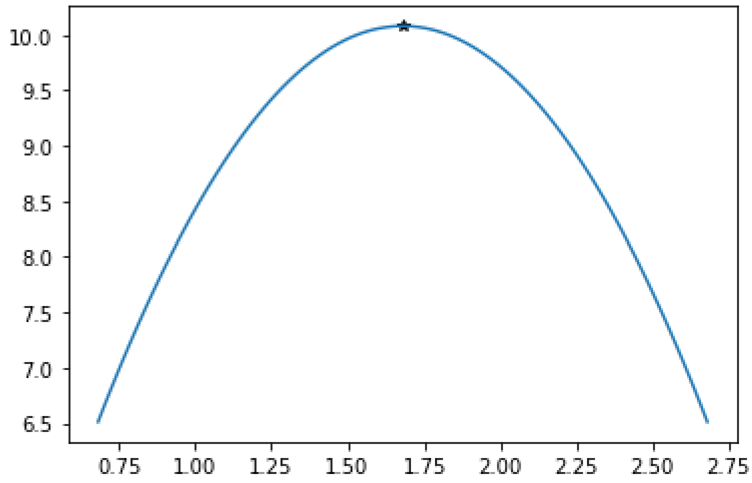


```
In [64]:
```

nuevo_y:

$$-\frac{\pi x}{2} - x + 6$$

```
Puntos criticos: [12/(pi + 4)]
Anchura optima: 24/(pi + 4)
Altura optima: -6*pi/(pi + 4) - 12/(pi + 4) + 6
```



Ejercicio 10

Determina los puntos sobre $y = x^2 + 1$ más cercanos a $(0, 2)$

In [44]:
```
# Versión Manual
puntox = 0
puntoy = 2
f = x**2+1
# Distancia euclidiana
distancia = sqrt((x - puntox)**2 + (f - puntoy)**2)
distancia
derivada = diff(distancia)
criticos = solve(derivada)
print("Puntos críticos:", criticos)
funcion_y = [f.subs(x,puntos) for puntos in np.linspace(-1,3,100)]
plt.plot(np.linspace(-1,3,100), funcion_y)
criticos_y = [f.subs(x,puntos) for puntos in criticos]
plt.plot(criticos, criticos_y, 'ro')
plt.plot(0,2,'ob')
```

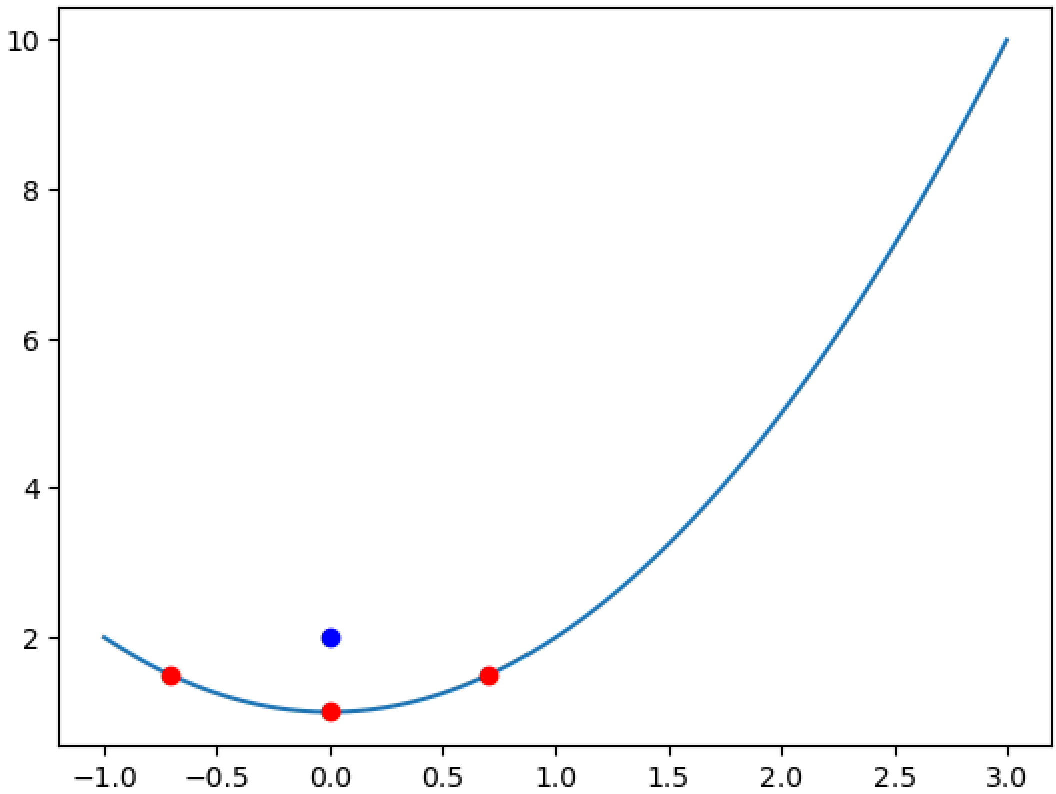
Out[44]: $\sqrt{x^2 + (x^2 - 1)^2}$

```
Puntos críticos: [0, -sqrt(2)/2, sqrt(2)/2]
```

Out[44]: [<matplotlib.lines.Line2D at 0x29049e7abd0>]

Out[44]: [<matplotlib.lines.Line2D at 0x2904ea7c2d0>]

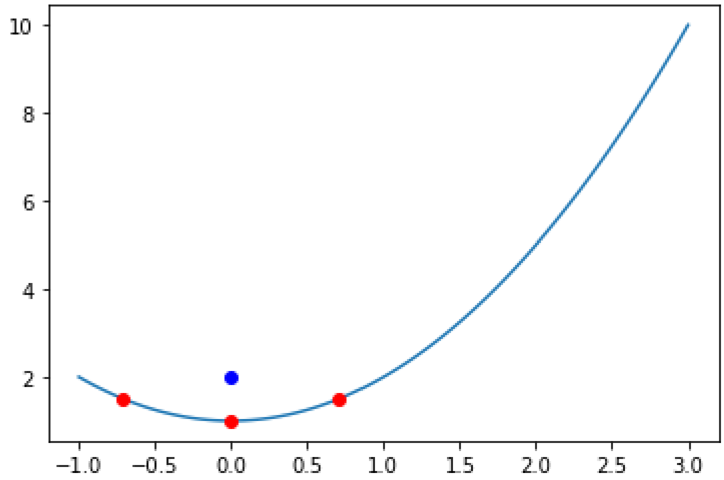
Out[44]: [<matplotlib.lines.Line2D at 0x2904e8590d0>]



In [65]:

```
Puntos críticos: [0, -sqrt(2)/2, sqrt(2)/2]
```

Out[65]: `[<matplotlib.lines.Line2D at 0x7f534830c2e0>]`



Ejercicio 11

Calcula la matriz Jacobiana de $g(x, y, z) = (e^x, cos(y), sen(z))$

Verifica cómo calcular la jacobiana en [https://docs.sympy.org/latest/modules/matrices/matrices.html (https://docs.sympy.org/latest/modules/matrices/matrices.html)]

In [11]:

```python
import math
z = symbols('z')

g_ejercicio11 = [exp(x), cos(y), sin(z)]
Matrix([[diff(elem, x) for elem in g_ejercicio11], [diff(elem, y) for elem in g_ejercicio11], [diff(elem, z) for elem in g_ejercicio11]])

# ó

Matrix([exp(x), cos(y), sin(z)]).jacobian(Matrix([x,y,z]))
```

Out[11]:

$$\begin{bmatrix} e^x & 0 & 0 \\ 0 & -\sin(y) & 0 \\ 0 & 0 & \cos(z) \end{bmatrix}$$

Out[11]:

$$\begin{bmatrix} e^x & 0 & 0 \\ 0 & -\sin(y) & 0 \\ 0 & 0 & \cos(z) \end{bmatrix}$$

In [66]:

Out[66]:

$$\begin{bmatrix} e^x & 0 & 0 \\ 0 & -\sin(y) & 0 \\ 0 & 0 & \cos(z) \end{bmatrix}$$

Ejercicio 12

Calcula la matriz Hessiana de la función $f(x, y) = xy + 2zx$

Verifica cómo calcular la Hessiana en [https://docs.sympy.org/latest/modules/matrices/matrices.html (https://docs.sympy.org/latest/modules/matrices/matrices.html)]

In [16]:
```
from sympy import hessian

f_ej12 = x*y + 2*z*x
hessian(f_ej12, (x, y, z))
```

Out[16]:

$$\begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 0 \\ 2 & 0 & 0 \end{bmatrix}$$

In [67]:

Out[67]:

$$\begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 0 \\ 2 & 0 & 0 \end{bmatrix}$$

Ejercicio 13

Representa las siguientes funciones y calcula su gradiente

$f(x, y) = x^2y^3$

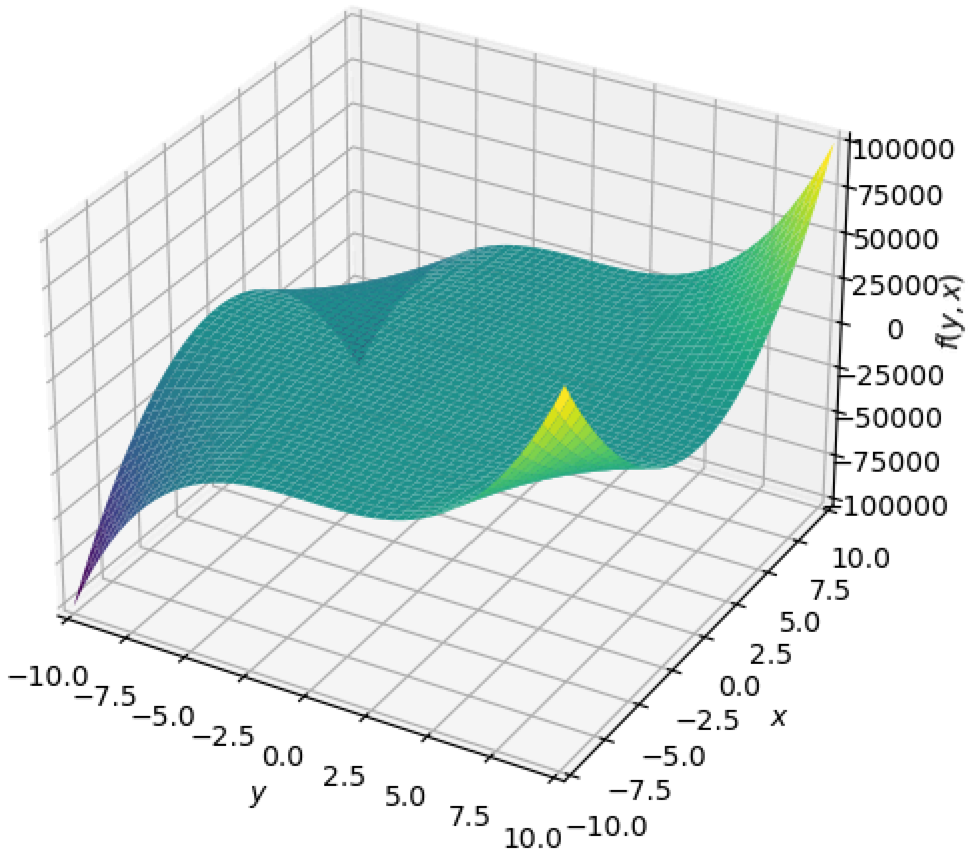
$g(x, y) = xe^{-x^2-y^2}$

Recuerda que el **gradiente** es el vector formado por las derivadas parciales de una función **escalar**.

La matriz **jacobiana** es la matriz formada por las derivadas parciales de una función vectorial. Sus vectores son los **gradientes** de los respectivos componentes de la función.

In [17]:
```
from sympy.plotting import plot3d

f_ej13 = x**2*y**3
g_ej13 = x*exp(-x**2-y**2)
```

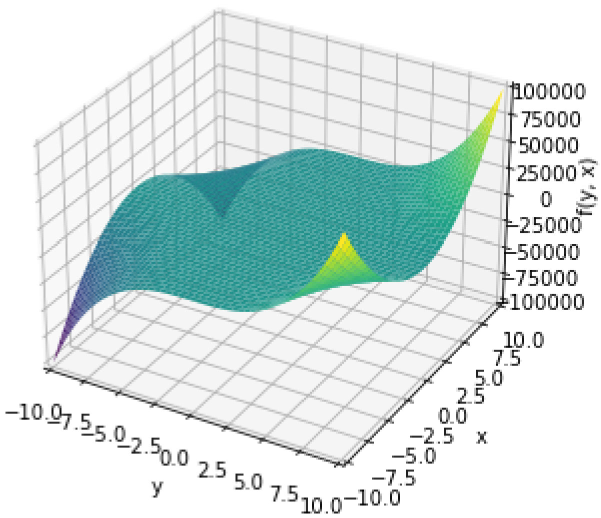
In [19]:
```
plot3d(f_ej13)

[diff(f_ej13,x), diff(f_ej13,y)]
```



Out[19]: `<sympy.plotting.plot.Plot at 0x15d9a383e10>`

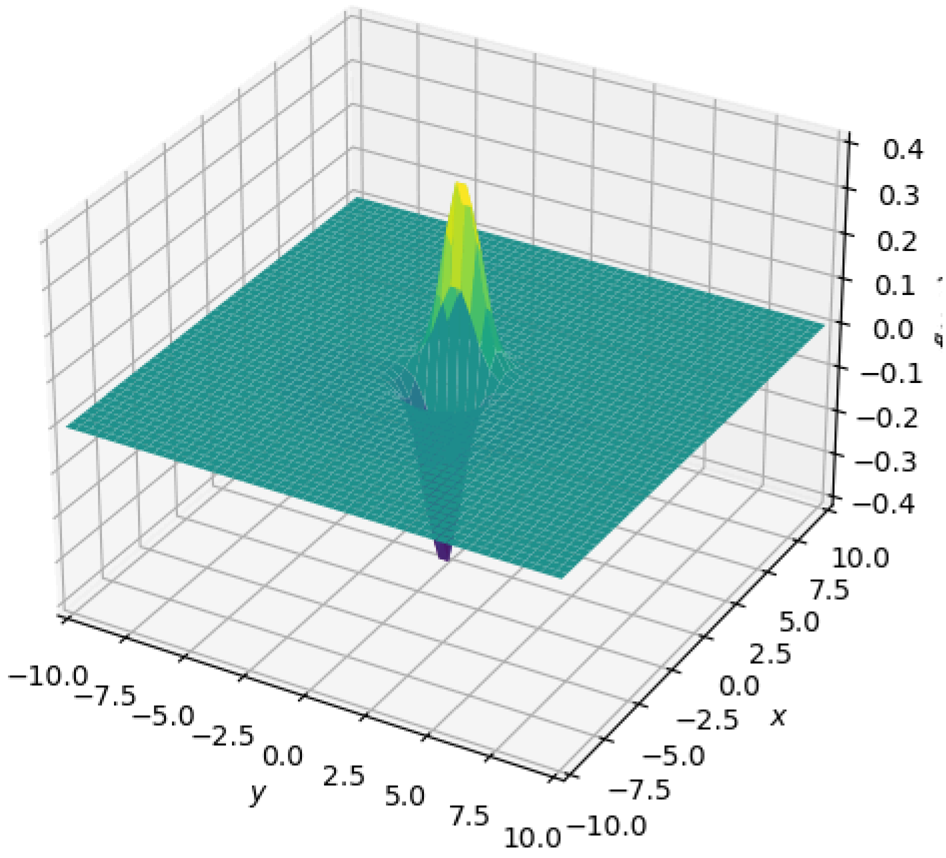
Out[19]: `[2*x*y**3, 3*x**2*y**2]`

In [68]:
```

```



$$\left[2xy^3 \quad 3x^2y^2 \right]$$

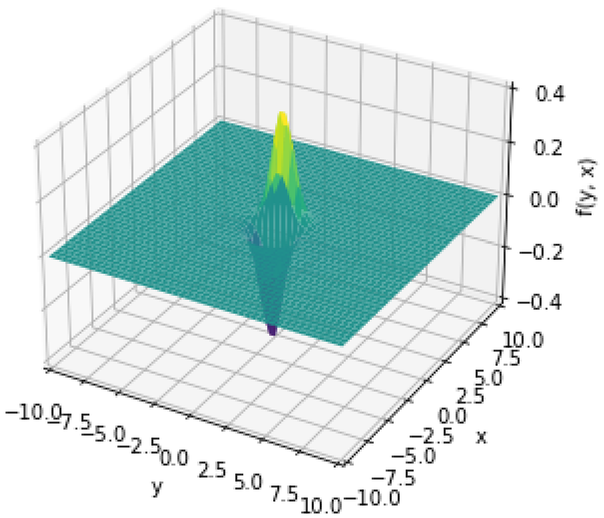
In [20]:
```
plot3d(g_ej13)

[diff(g_ej13,x), diff(g_ej13,y)]
```



Out[20]: <sympy.plotting.plot.Plot at 0x15d99598850>

Out[20]: [-2*x**2*exp(-x**2 - y**2) + exp(-x**2 - y**2), -2*x*y*exp(-x**2 - y**2)]

In [69]:



Out[69]: $\left[-2x^2e^{-x^2-y^2} + e^{-x^2-y^2} \quad -2xye^{-x^2-y^2} \right]$