InstitutoTecnológico y de Estudios Superiores de Occidente

Maestría Ciencia de Datos

Investigación, Desarrollo e Innovación II

Tarea 2: Gradiente descendente

ITESO, Universidad Jesuita de Guadalajara

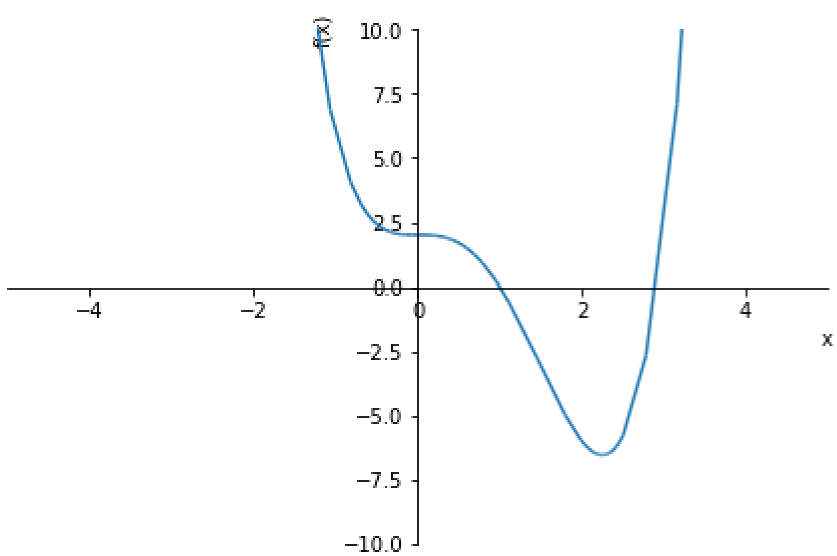
Estudiante: Daniel Nuño
Profesor: Fernando Becerra
25 Septiembre 2021

Realice código en Python que, recibiendo una función f dada, un valor inicial x_0 y una exactitud (error) dado E , use el método de gradiente descendente (ascendente) para encontrar un mínimo (máximo) local de f. Asegúrese que cuenta el número de iteraciones realizadas.

Use su código para encontrar (si existe) los mínimos y máximos locales de cada función con precisión de 4 cifras significativas. y exactitud de 10^3. En todos los casos indique el(los) valor(es) inicial(es) que utilizó y el número de iteraciones que fueron necesarias para alcanzar la respuesta.

1 $f(x) = x^4 - 3x^3 + 3$

In [1]:
```
%matplotlib inline

import sympy as sp
from sympy.plotting.plot import plot_contour

x, y, z = sp.symbols('x y z')

f = x**4-3*x**3+2
f_derivada = f.diff(x,1)

sp.plotting.plot(f,  xlim=(-5, 5), ylim=(-10,10))
f
```



Out[1]: $x^4 - 3x^3 + 2$

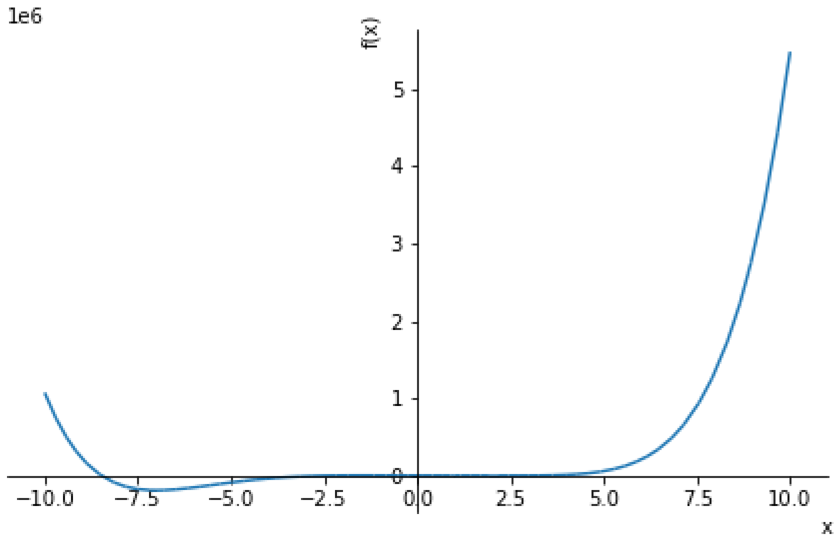
Siendo una función de una variable tenemos la ventaja de poder gráficar y por lo tanto aproximar el mínimo visualmente en 2.

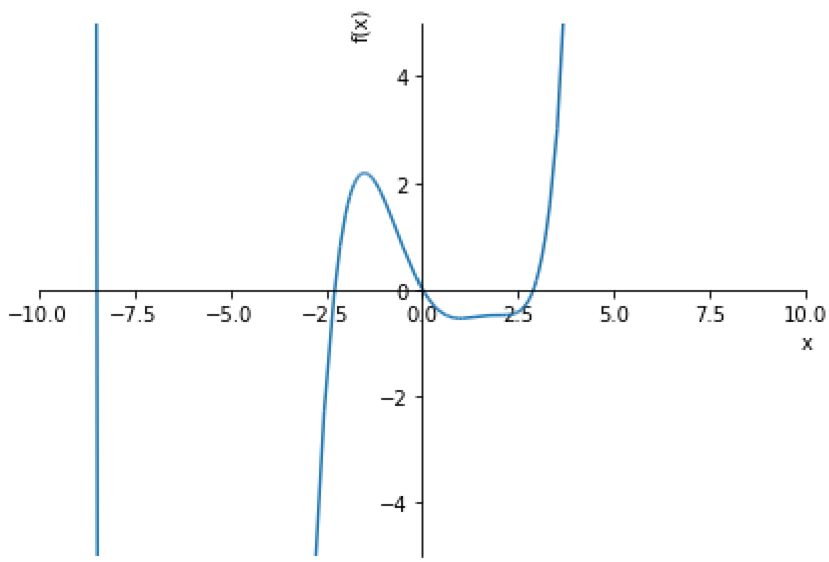
In [2]:
```
xn = 2
exactitud = 10**-3
iteraciones = 0
error = 1
alpha = 0.01
lista_resultados = [xn]

y_derivada = f_derivada.evalf(subs={x: xn})

while error >= exactitud:
    iteraciones += 1
    xn_1 = xn - alpha*y_derivada
    lista_resultados.append(xn_1)
    y_derivada = f_derivada.evalf(subs={x: xn_1})
    error = abs( y_derivada )
    xn = xn_1

print("El mínimo local {} esta en x {} y fue calculado con {} iteraciones.".fo
rmat(sp.N(f.evalf(subs={x: xn}),4),

sp.N(xn,4),

iteraciones))
```

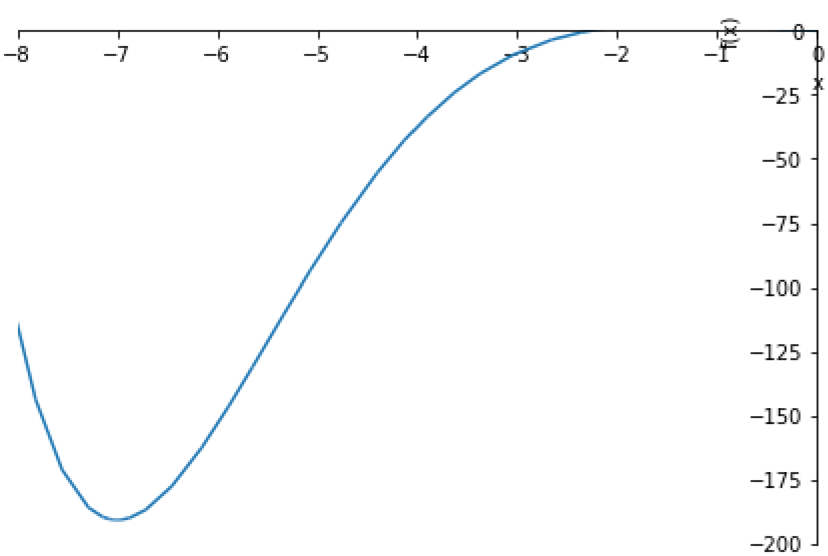
```
El mínimo local -6.543 esta en x 2.250 y fue calculado con 40 iteraciones.
```

2 $f(x) = 5*x^6 + 21*x^5 - 180*x^4 + 115*x^3 + 750*x^2 - 1260*x + 10$

In [3]:
```
f = 5*x**6 + 21*x**5 - 180*x**4 + 115*x**3 + 750*x**2 - 1260*x + 10
f_derivada = f.diff(x,1)
sp.plotting.plot(f)
sp.plotting.plot(f/1000, xlim=(-10, 10), ylim=(-5,5))
sp.plotting.plot(f/1000, xlim=(-8, 0), ylim=(-200,0))
f
```







Out[3]: $5x^6 + 21x^5 - 180x^4 + 115x^3 + 750x^2 - 1260x + 10$

Esta función tiene sabemos que tiene valores de y muy altos, y se puede apreciar en la primera de las tres gráficas que muestra y en la escala $10e6$. Por lo tanto para poder observar visualmente los posibles minimos y maximos dividimos $f(x)$ entre 1000. Tenemos un mínimo cerca de zero, un máximo cerca de -2 y un segundo mínimo cerca de -7.

También vamos a disminuir alpha para evitar saltos muy grandes cuando nos encontramos con derivadas muy inclinadas.

Primer mínimo:

In [4]:
```
xn = 0
exactitud = 10**-3
iteraciones = 0
error = 1
alpha = 0.0001
lista_resultados = [xn]
y_derivada = f_derivada.evalf(subs={x: xn})

while error >= exactitud:
    iteraciones += 1
    xn_1 = xn - alpha*y_derivada
    lista_resultados.append(xn_1)
    y_derivada = f_derivada.evalf(subs={x: xn_1})
    error = abs( y_derivada )
    xn = xn_1

print("El mínimo local {} esta en x {} y fue calculado con {} iteraciones.".fo
rmat(sp.N(f.evalf(subs={x: xn}),4),

sp.N(xn,4),

iteraciones))
```

```
El mínimo local -539.0 esta en x 1.000 y fue calculado con 201 iteraciones.
```

Primer máximo:

```
In [5]: xn = -2
exactitud = 10**-3
iteraciones = 0
error = 1
alpha = 0.0001
lista_resultados = [xn]
y_derivada = f_derivada.evalf(subs={x: xn})

while error >= exactitud:
    iteraciones += 1
    xn_1 = xn + alpha*y_derivada
    lista_resultados.append(xn_1)
    y_derivada = f_derivada.evalf(subs={x: xn_1})
    error = abs( y_derivada )
    xn = xn_1

print("El máximo local {} esta en x {} y fue calculado con {} iteraciones.".fo
rmat(sp.N(f.evalf(subs={x: xn}),4),

sp.N(xn,4),

iteraciones))
```

```
El máximo local 2186 esta en x -1.500 y fue calculado con 20 iteraciones.
```

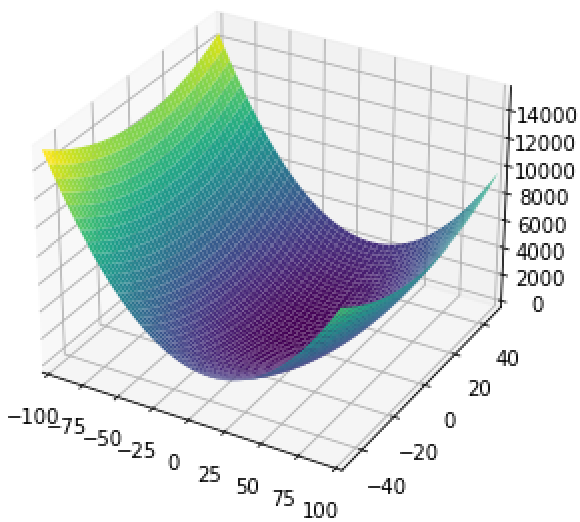
Segundo mínimo:

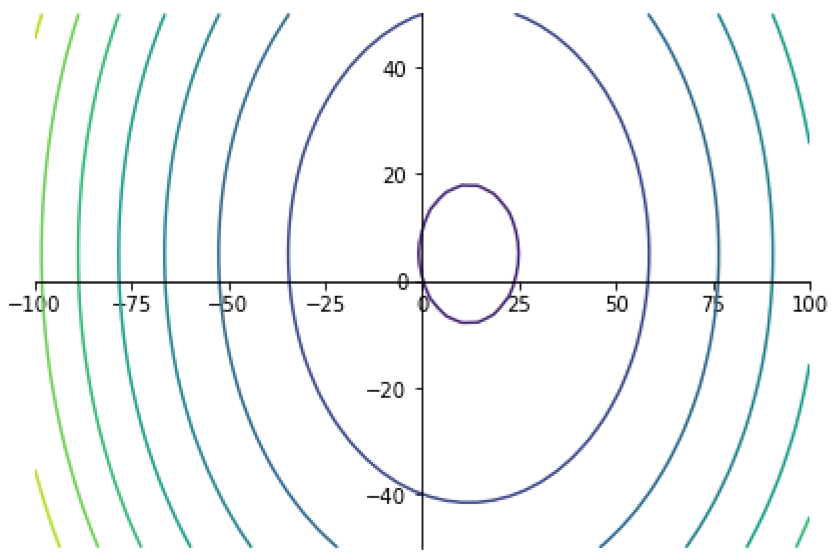
```
In [6]: xn = -6
exactitud = 10**-3
iteraciones = 0
error = 1
alpha = 0.00001
lista_resultados = [xn]
y_derivada = f_derivada.evalf(subs={x: xn})

while error >= exactitud:
    iteraciones += 1
    xn_1 = xn - alpha*y_derivada
    lista_resultados.append(xn_1)
    y_derivada = f_derivada.evalf(subs={x: xn_1})
    error = abs( y_derivada )
    xn = xn_1

print("El máximo local {} esta en x {} y fue calculado con {} iteraciones.".fo
rmat(sp.N(f.evalf(subs={x: xn}),4),

sp.N(xn,4),

iteraciones))
```

```
El máximo local -1.907E+5 esta en x -7.000 y fue calculado con 8 iteraciones.
```

3. $f(x, y) = x^2 - 24x + y^2 - 10y$

In [7]:
```
f = x**2 - 24*x + y**2 - 10*y
f_derivada = sp.Matrix([f.diff(x,1), f.diff(y,1)])
sp.plotting.plot3d(f, (x, -100, 100), (y, -50, 50))
plot_contour(f, (x, -100, 100), (y, -50, 50))
f
```





Out[7]: $x^2 - 24x + y^2 - 10y$

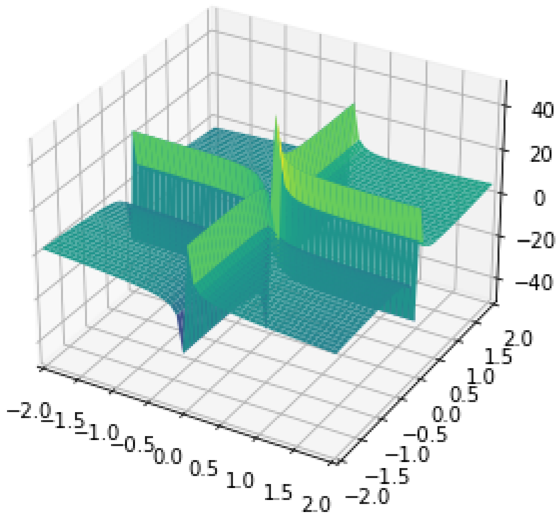
Viendo el gráfica en tres dimensiones, el mapa de curvas de nivel podemos observar, y el gradiende de color de verde a azul, podemos inferir que hay un minimo cerca alrededor de $x = 15$ y $y = 3$.

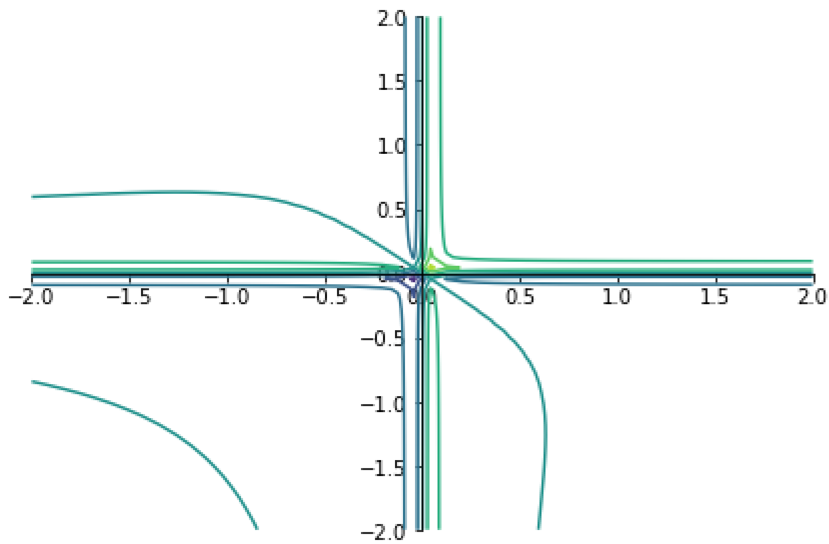
In [8]:
```
exactitud = 10**-3
iteraciones = 0
error = 1
xn=15
yn=3
Xn = sp.Matrix([xn, yn])
lista_X = [xn]
lista_y = [yn]
alpha = 0.1
y_derivada = f_derivada.evalf(subs={x: Xn[0], y: Xn[1]})

while error >= exactitud:
    iteraciones += 1
    Xn_1 = Xn - alpha*y_derivada
    lista_X.append(Xn_1[0])
    lista_y.append(Xn_1[1])
    y_derivada = f_derivada.evalf(subs={x: Xn_1[0], y: Xn_1[1]})
    error = ( y_derivada[0]**2 + y_derivada[1]**2 )**(1/2)
    Xn = Xn_1

print("El minimo local {} esta en x {} e y {} fue calculado con {} iteracione
s.".format(sp.N(f.evalf(subs={x: Xn[0], y: Xn[1]}),4),

sp.N(Xn[0],4),

sp.N(Xn[1],4),

iteraciones))
```

```
El minimo local -169.0 esta en x 12.00 e y 5.000 fue calculado con 40 iteraci
ones.
```

4. $f(x,y) = xy + \frac{1}{x} + \frac{1}{y}$

In [9]:
```
f = x*y + 1/x + 1/y
f_derivada = sp.Matrix([f.diff(x,1), f.diff(y,1)])
sp.plotting.plot3d(f, (x, -2, 2), (y, -2, 2))
plot_contour(f, (x, -2, 2), (y, -2, 2))
f
```





Out[9]: $xy + \frac{1}{y} + \frac{1}{x}$

Es una función y gráfica complicada pero sabemos analiticamente que no existe en $x = 0$ ni $y = 0$; por lo tanto no podemos comenzar por ahí por que nos dara automaticamente un mínimo o máximo. Comenzamos con 1 y 1:

In [10]:
```
exactitud = 10**-3
iteraciones = 0
error = 1
xn=1
yn=1
Xn = sp.Matrix([xn, yn])
lista_X = [xn]
lista_y = [yn]
alpha = 0.01
y_derivada = f_derivada.evalf(subs={x: Xn[0], y: Xn[1]})

while error >= exactitud and iteraciones < 500:
    iteraciones += 1
    Xn_1 = Xn - alpha*y_derivada
    lista_X.append(Xn_1[0])
    lista_y.append(Xn_1[1])
    y_derivada = f_derivada.evalf(subs={x: Xn_1[0], y: Xn_1[1]})
    error = ( y_derivada[0]**2 + y_derivada[1]**2 )**(1/2)
    Xn = Xn_1

print("El minimo local {} esta en x {} e y {} fue calculado con {} iteracione
s.".format(
                                                                    sp.N(f.evalf(subs={x: Xn[0
], y: Xn[1]}),4),

sp.N(Xn[0],4),

sp.N(Xn[1],4),

iteraciones))
```

```
El minimo local 3.000 esta en x 1.000 e y 1.000 fue calculado con 1 iteracion
es.
```

Ahora vamos a buscar en el cuadrante de negativos -2 y -2.

In [11]:
```
exactitud = 10**-3
iteraciones = 0
error = 1
xn=-2
yn=-2
Xn = sp.Matrix([xn, yn])
lista_X = [xn]
lista_y = [yn]
alpha = 0.01
y_derivada = f_derivada.evalf(subs={x: Xn[0], y: Xn[1]})

while error >= exactitud and iteraciones < 500:
    iteraciones += 1
    Xn_1 = Xn - alpha*y_derivada
    lista_X.append(Xn_1[0])
    lista_y.append(Xn_1[1])
    y_derivada = f_derivada.evalf(subs={x: Xn_1[0], y: Xn_1[1]})
    error = ( y_derivada[0]**2 + y_derivada[1]**2 )**(1/2)
    Xn = Xn_1

print("El mínimo local {} esta en x {} e y {} fue calculado con {} iteracione
s.".format(
                                                    sp.N(f.evalf(subs={x: Xn[0
], y: Xn[1]}),4),

sp.N(Xn[0],4),

sp.N(Xn[1],4),

iteraciones))
```

```
El mínimo local 3.000 esta en x 1.000 e y 1.000 fue calculado con 435 iteraci
ones.
```

Obtuvimos el mismo resultado. Ahora buscamos un máximo con limite de 500 iteraciones.

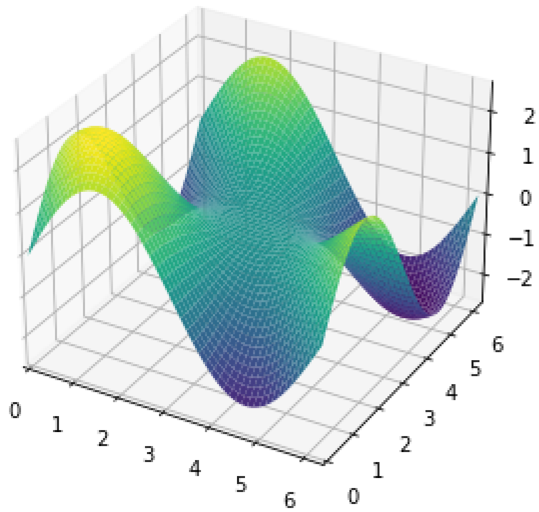
In [12]:
```
exactitud = 10**-3
iteraciones = 0
error = 1
xn = -2
yn = -2
Xn = sp.Matrix([xn, yn])
lista_X = [xn]
lista_y = [yn]
alpha = 0.01
y_derivada = f_derivada.evalf(subs={x: Xn[0], y: Xn[1]})

while error >= exactitud and iteraciones < 500:
    iteraciones += 1
    Xn_1 = Xn + alpha*y_derivada
    lista_X.append(Xn_1[0])
    lista_y.append(Xn_1[1])
    y_derivada = f_derivada.evalf(subs={x: Xn_1[0], y: Xn_1[1]})
    error = ( y_derivada[0]**2 + y_derivada[1]**2 )**(1/2)
    Xn = Xn_1

print("En x {} e y {} fue calculado z {} con {} iteraciones. El último error f
ue {}".format(

sp.N(Xn[0],4),

sp.N(Xn[1],4),

f.evalf(subs={x: Xn[0], y: Xn[1]}),

iteraciones,

error))
```

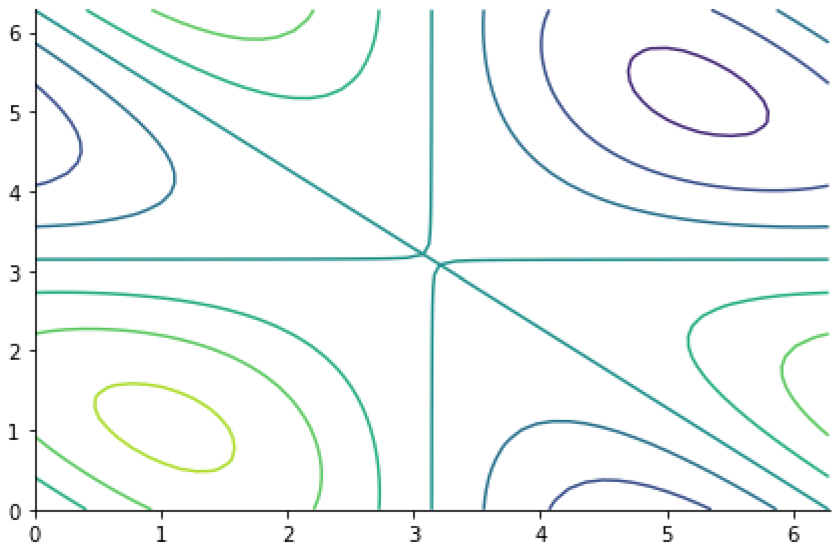
```
En x -301.3 e y -301.3 fue calculado z 90756.0251369336 con 500 iteraciones.
El último error fue 426.042341592042
```

Después de 500 iteraciones el error no convergio a zero y se alejó entonces concluimos que no hay máximo.

4. $f(x,y) = sin(x) + sin(y) + sin(x+y), 0 \leq x \leq 2\pi, 0 \leq y \leq 2\pi$

```
In [13]: f = sp.sin(x)+sp.sin(y)+sp.sin(x+y)
f_derivada = sp.Matrix([f.diff(x,1), f.diff(y,1)])
sp.plotting.plot3d(f, (x, 0, 2*sp.pi), (y, 0, 2*sp.pi))
plot_contour(f, (x, 0, 2*sp.pi), (y, 0, 2*sp.pi))
f
```





Out[13]: $\sin(x) + \sin(y) + \sin(x + y)$

Mirando la gráfica el máximo en la region delimitada esta cerca de 1 y 1 y el mínimo cerca de 6 y 6.

In [14]:
```
exactitud = 10**-3
iteraciones = 0
error = 1
xn=1
yn=1
Xn = sp.Matrix([xn, yn])
lista_X = [xn]
lista_y = [yn]
alpha = 0.01
y_derivada = f_derivada.evalf(subs={x: Xn[0], y: Xn[1]})

while error >= exactitud and iteraciones < 500:
    iteraciones += 1
    Xn_1 = Xn + alpha*y_derivada
    lista_X.append(Xn_1[0])
    lista_y.append(Xn_1[1])
    y_derivada = f_derivada.evalf(subs={x: Xn_1[0], y: Xn_1[1]})
    error = ( y_derivada[0]**2 + y_derivada[1]**2 )**(1/2)
    Xn = Xn_1

print("El máximo local {} esta en x {} e y {} fue calculado con {} iteracione
s.".format(
                                                                    sp.N(f.evalf(subs={x: Xn[0
], y: Xn[1]}),4),

sp.N(Xn[0],4),

sp.N(Xn[1],4),

iteraciones))
```

```
El máximo local 2.598 esta en x 1.047 e y 1.047 fue calculado con 196 iteraci
ones.
```

In [15]:
```
exactitud = 10**-3
iteraciones = 0
error = 1
xn=6
yn=6
Xn = sp.Matrix([xn, yn])
lista_X = [xn]
lista_y = [yn]
alpha = 0.01
y_derivada = f_derivada.evalf(subs={x: Xn[0], y: Xn[1]})

while error >= exactitud and iteraciones < 500:
    iteraciones += 1
    Xn_1 = Xn - alpha*y_derivada
    lista_X.append(Xn_1[0])
    lista_y.append(Xn_1[1])
    y_derivada = f_derivada.evalf(subs={x: Xn_1[0], y: Xn_1[1]})
    error = ( y_derivada[0]**2 + y_derivada[1]**2 )**(1/2)
    Xn = Xn_1

print("El mínimo local {} esta en x {} e y {} fue calculado con {} iteracione
s.".format(
                                                                     sp.N(f.evalf(subs={x: Xn[0
], y: Xn[1]}),4),

sp.N(Xn[0],4),

sp.N(Xn[1],4),

iteraciones))
```

```
El mínimo local -2.598 esta en x 5.236 e y 5.236 fue calculado con 300 iterac
iones.
```

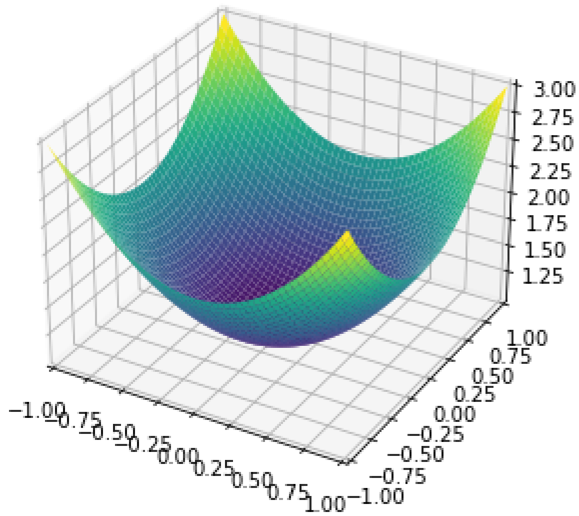
5 $f(x, y, z) = x^2 + y^2 + z^2 + 1$

In [16]:
```
f = x**2 + y**2 + z**2 + 1
f_derivada = sp.Matrix([f.diff(x,1), f.diff(y,1), f.diff(z,1)])
f
```

Out[16]: $x^2 + y^2 + z^2 + 1$

Para intentar verla gráficamente podemos reducir una a tres dimensiones con dos variables independientes haciendo z una constante.

In [17]:
```
f_sin_z = x**2 + y**2 + 1
sp.plotting.plot3d(f_sin_z, (x, -1, 1), (y, -1, 1))
```



Out[17]: <sympy.plotting.plot.Plot at 0x2608c647ac0>

Basados en la gráfica y analiticamente sabemos también que es un paraboloide con dos variables y por lo tanto buscamos un mínimo.

In [18]:
```
exactitud = 10**-3
iteraciones = 0
error = 1
xn = 1
yn = 1
zn = 1
Xn = sp.Matrix([xn, yn, zn])
lista_X = [xn]
lista_y = [yn]
lista_z = [zn]
alpha = 0.1
y_derivada = f_derivada.evalf(subs={x: Xn[0], y: Xn[1], z: Xn[2]})

while error >= exactitud and iteraciones < 500:
    iteraciones += 1
    Xn_1 = Xn - alpha*y_derivada
    lista_X.append(Xn_1[0])
    lista_y.append(Xn_1[1])
    lista_z.append(Xn_1[2])
    y_derivada = f_derivada.evalf(subs={x: Xn_1[0], y: Xn_1[1], z: Xn_1[2]})
    error = ( y_derivada[0]**2 + y_derivada[1]**2 + y_derivada[2]**2 )**(1/2)
    Xn = Xn_1

print("El mínimo local {} esta en ( {}, {}, {} ) fue calculado con {} iteracio
nes.".format(
                                                    sp.N(f.evalf(subs={x: Xn[0
], y: Xn[1], z: Xn[2]}),4),

sp.N(Xn[0],4),

sp.N(Xn[1],4),

sp.N(Xn[2],4),

iteraciones))
```

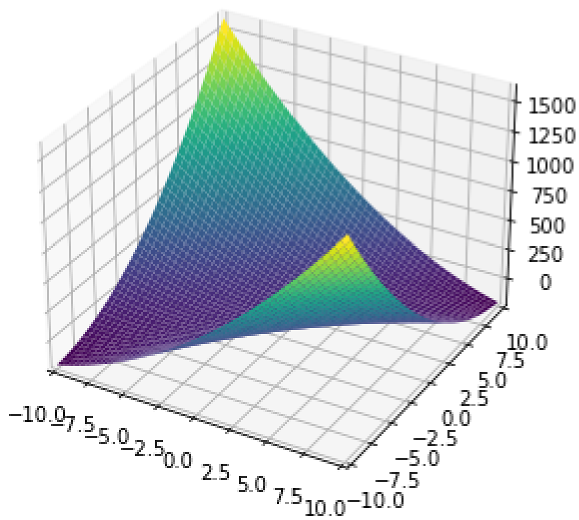
```
El mínimo local 1.000 esta en ( 0.0002596, 0.0002596, 0.0002596 ) fue calcula
do con 37 iteraciones.
```

6 $f(x, y, x) = 3x^2 + 4y^2 + z^2 - 9xyz$

In [19]:
```
f = 3*x**2+4*y**2+z**2-9*x*y*z
f_derivada = sp.Matrix([f.diff(x,1), f.diff(y,1), f.diff(z,1)])
f
```

Out[19]: $3x^2 - 9xyz + 4y^2 + z^2$

In [20]:
```
f_sin_z = 3*x**2+4*y**2+1-9*x*y
sp.plotting.plot3d(f_sin_z, (x, -10, 10), (y, -10, 10))
```



Out[20]:
```
<sympy.plotting.plot.Plot at 0x2608afdf460>
```

In [21]:
```python
exactitud = 10**-3
iteraciones = 0
error = [1]
xn = -1
yn = -1
zn = -1
Xn = sp.Matrix([xn, yn, zn])
lista_X = [xn]
lista_y = [yn]
lista_z = [zn]
alpha = 0.001
y_derivada = f_derivada.evalf(subs={x: Xn[0], y: Xn[1], z: Xn[2]})

while error[-1] >= exactitud and iteraciones < 5000:
    iteraciones += 1
    Xn_1 = Xn - alpha*y_derivada
    lista_X.append(Xn_1[0])
    lista_y.append(Xn_1[1])
    lista_z.append(Xn_1[2])
    y_derivada = f_derivada.evalf(subs={x: Xn_1[0], y: Xn_1[1], z: Xn_1[2]})
    error.append( ( y_derivada[0]**2 + y_derivada[1]**2 + y_derivada[2]**2 )**(1/2) )
    Xn = Xn_1

print("El mínimo local {} esta en ( {}, {}, {} ) fue calculado con {} iteraciones.".format(
                                                    sp.N(f.evalf(subs={x: Xn[0], y: Xn[1], z: Xn[2]}),4),

sp.N(Xn[0],4),

sp.N(Xn[1],4),

sp.N(Xn[2],4),

iteraciones))
```

```
El mínimo local 2.495E-7 esta en ( -3.608E-10, 4.914E-12, -0.0004995 ) fue calculado con 3589 iteraciones.
```

In [22]:
```
exactitud = 10**-3
iteraciones = 0
error = [1]
xn = -1
yn = -1
zn = -1
Xn = sp.Matrix([xn, yn, zn])
lista_X = [xn]
lista_y = [yn]
lista_z = [zn]
alpha = 0.001
y_derivada = f_derivada.evalf(subs={x: Xn[0], y: Xn[1], z: Xn[2]})

while error[-1] >= exactitud and iteraciones < 100:
    iteraciones += 1
    Xn_1 = Xn + alpha*y_derivada
    lista_X.append(Xn_1[0])
    lista_y.append(Xn_1[1])
    lista_z.append(Xn_1[2])
    y_derivada = f_derivada.evalf(subs={x: Xn_1[0], y: Xn_1[1], z: Xn_1[2]})
    error.append( ( y_derivada[0]**2 + y_derivada[1]**2 + y_derivada[2]**2 )**(1/2) )
    Xn = Xn_1
```

Despues de varios intentos parece no tener máximo. Con 100 interaciones el error es $5.99130418528237e + 379$

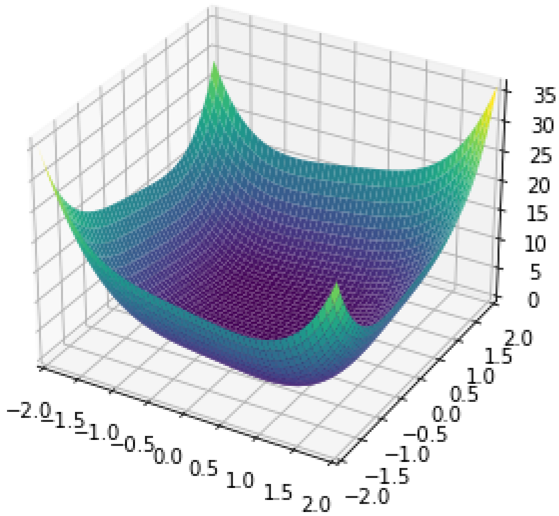
6 $f(x, y, x) = x^4 + y^4 + z^4 + xyz$

In [23]:
```
f = x**4 + y**4 + z**4 + x*y*z
f_derivada = sp.Matrix([f.diff(x,1), f.diff(y,1), f.diff(z,1)])
f
```

Out[23]: $x^4 + xyz + y^4 + z^4$

Para intentar verla gráficamente podemos reducir una a tres dimensiones con dos variables independientes haciendo $z = 0$.

In [24]:
```
f_sin_z = x**4 + y**4 + x*y
sp.plotting.plot3d(f_sin_z, (x, -2, 2), (y, -2, 2))
```



Out[24]:
```
<sympy.plotting.plot.Plot at 0x2608afdd8e0>
```

Puede que podamos encontrar varios minimos entre -2 a 2 en diferentes direcciones.

In [25]:

```
exactitud = 10**-3
iteraciones = 0
error = [1]
xn = -1
yn = -2
zn = -1
Xn = sp.Matrix([xn, yn, zn])
lista_X = [xn]
lista_y = [yn]
lista_z = [zn]
alpha = 0.01
y_derivada = f_derivada.evalf(subs={x: Xn[0], y: Xn[1], z: Xn[2]})

while error[-1] >= exactitud and iteraciones < 5000:
    iteraciones += 1
    Xn_1 = Xn - alpha*y_derivada
    lista_X.append(Xn_1[0])
    lista_y.append(Xn_1[1])
    lista_z.append(Xn_1[2])
    y_derivada = f_derivada.evalf(subs={x: Xn_1[0], y: Xn_1[1], z: Xn_1[2]})
    error.append( ( y_derivada[0]**2 + y_derivada[1]**2 + y_derivada[2]**2 )**
(1/2) )
    Xn = Xn_1

print("El primer mínimo local {} esta en ( {}, {}, {} ) fue calculado con {} i
teraciones.".format(
                                                    sp.N(f.evalf(subs={x: Xn[0
], y: Xn[1], z: Xn[2]}),4),

sp.N(Xn[0],4),

sp.N(Xn[1],4),

sp.N(Xn[2],4),

iteraciones))
```

```
El primer mínimo local -0.003904 esta en ( -0.2523, -0.2523, -0.2523 ) fue ca
lculado con 1474 iteraciones.
```

In [26]:

```
exactitud = 10**-3
iteraciones = 0
error = [1]
xn = -1
yn = 2
zn = 1
Xn = sp.Matrix([xn, yn, zn])
lista_X = [xn]
lista_y = [yn]
lista_z = [zn]
alpha = 0.01
y_derivada = f_derivada.evalf(subs={x: Xn[0], y: Xn[1], z: Xn[2]})

while error[-1] >= exactitud and iteraciones < 5000:
    iteraciones += 1
    Xn_1 = Xn - alpha*y_derivada
    lista_X.append(Xn_1[0])
    lista_y.append(Xn_1[1])
    lista_z.append(Xn_1[2])
    y_derivada = f_derivada.evalf(subs={x: Xn_1[0], y: Xn_1[1], z: Xn_1[2]})
    error.append( ( y_derivada[0]**2 + y_derivada[1]**2 + y_derivada[2]**2 )**
(1/2) )
    Xn = Xn_1

print("El segundo mínimo local {} esta en ( {}, {}, {} ) fue calculado con {}
 iteraciones.".format(
                                                                    sp.N(f.evalf(subs={x: Xn[0
], y: Xn[1], z: Xn[2]}),4),

sp.N(Xn[0],4),

sp.N(Xn[1],4),

sp.N(Xn[2],4),

iteraciones))
```

```
El segundo mínimo local -0.003904 esta en ( -0.2523, 0.2523, 0.2523 ) fue cal
culado con 1474 iteraciones.
```

In [27]:
```
exactitud = 10**-3
iteraciones = 0
error = [1]
xn = 1
yn = -2
zn = -1
Xn = sp.Matrix([xn, yn, zn])
lista_X = [xn]
lista_y = [yn]
lista_z = [zn]
alpha = 0.05
y_derivada = f_derivada.evalf(subs={x: Xn[0], y: Xn[1], z: Xn[2]})

while error[-1] >= exactitud and iteraciones < 5000:
    iteraciones += 1
    Xn_1 = Xn - alpha*y_derivada
    lista_X.append(Xn_1[0])
    lista_y.append(Xn_1[1])
    lista_z.append(Xn_1[2])
    y_derivada = f_derivada.evalf(subs={x: Xn_1[0], y: Xn_1[1], z: Xn_1[2]})
    error.append( ( y_derivada[0]**2 + y_derivada[1]**2 + y_derivada[2]**2 )**
(1/2) )
    Xn = Xn_1

print("El tercer mínimo local {} esta en ( {}, {}, {} ) fue calculado con {} i
teraciones.".format(
                                                    sp.N(f.evalf(subs={x: Xn[0
], y: Xn[1], z: Xn[2]}),4),

sp.N(Xn[0],4),

sp.N(Xn[1],4),

sp.N(Xn[2],4),

iteraciones))
```

```
El tercer mínimo local -0.003904 esta en ( 0.2477, 0.2477, -0.2477 ) fue calc
ulado con 694 iteraciones.
```

In [28]:
```
exactitud = 10**-3
iteraciones = 0
error = [1]
xn = 1
yn = -2
zn = 1
Xn = sp.Matrix([xn, yn, zn])
lista_X = [xn]
lista_y = [yn]
lista_z = [zn]
alpha = 0.05
y_derivada = f_derivada.evalf(subs={x: Xn[0], y: Xn[1], z: Xn[2]})

while error[-1] >= exactitud and iteraciones < 5000:
    iteraciones += 1
    Xn_1 = Xn - alpha*y_derivada
    lista_X.append(Xn_1[0])
    lista_y.append(Xn_1[1])
    lista_z.append(Xn_1[2])
    y_derivada = f_derivada.evalf(subs={x: Xn_1[0], y: Xn_1[1], z: Xn_1[2]})
    error.append( ( y_derivada[0]**2 + y_derivada[1]**2 + y_derivada[2]**2 )**
(1/2) )
    Xn = Xn_1

print("El cuarto mínimo local {} esta en ( {}, {}, {} ) fue calculado con {} i
teraciones.".format(
                                                   sp.N(f.evalf(subs={x: Xn[0
], y: Xn[1], z: Xn[2]}),4),

sp.N(Xn[0],4),

sp.N(Xn[1],4),

sp.N(Xn[2],4),

iteraciones))
```

```
El cuarto mínimo local -0.003904 esta en ( 0.2523, -0.2523, 0.2523 ) fue calc
ulado con 288 iteraciones.
```

Cuatro minimos fueron calculados en $f(x, y, z) \approx -0.003904$ con los mismos valores de x, y, z pero en diferentes direcciones en $x, y, z \approx + - 0.2523$