# EJE2NEWTON

April 2, 2023

## 1 Método de Newton

[19]:
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

[20]:
```
# Definamos la función f y sus derivadas.
#f(x, y) := (e^x^2)(y^2) + (2e^y^2)(x^2) + 4xy + (2x^2) + 4x - 2y
```

[21]:
```
def f(x, y):
    return np.exp(x**2) * y**2 + 2*np.exp(y**2) * x**2 + 4*x*y + 2*x**2 + 4*x -␣
 ↪2*y
```

[22]:
```
def grad_f(x, y):
    df_dx = 2*x*np.exp(x**2)*y**2 + 4*np.exp(y**2)*x + 4*y + 4*x + 4
    df_dy = 2*y*np.exp(x**2)*x**2 + 4*np.exp(y**2)*y + 4*x - 2
    return np.array([df_dx, df_dy])
```

[23]:
```
def hess_f(x, y):
    d2f_dx2 = 2*np.exp(x**2)*y**2 + 4*np.exp(x**2)*x**2 + 4*np.exp(y**2)
    d2f_dy2 = 2*np.exp(x**2)*x**2 + 4*np.exp(y**2)
    d2f_dxdy = 4*x*y*np.exp(x**2)
    return np.array([[d2f_dx2, d2f_dxdy], [d2f_dxdy, d2f_dy2]])
```

[24]:
```
#Ahora definamos el método de Newton.
```

[25]:
```
def newton_method(f, grad_f, hess_f, x0, y0, tol=1e-5, max_iter=100):
    x, y = x0, y0
    for i in range(max_iter):
        grad = grad_f(x, y)
        hess = hess_f(x, y)
        if np.linalg.norm(grad) < tol:
            break
        delta = np.linalg.solve(hess, -grad)
        x += delta[0]
        y += delta[1]
    return x, y
```

[26]:
```
#Para encontrar el punto crítico de nuestra función.
```

[27]:
```
x0, y0 = 25, -30
x_star, y_star = newton_method(f, grad_f, hess_f, x0, y0)

print('Coordenadas del punto crítico: ({:.6f}, {:.6f})'.format(x_star, y_star))
```

```
Coordenadas del punto crítico: (nan, nan)

C:\Users\ASUS\AppData\Local\Temp\ipykernel_13244\3234313047.py:2:
RuntimeWarning: overflow encountered in exp
  df_dx = 2*x*np.exp(x**2)*y**2 + 4*np.exp(y**2)*x + 4*y + 4*x + 4
C:\Users\ASUS\AppData\Local\Temp\ipykernel_13244\3234313047.py:3:
RuntimeWarning: overflow encountered in exp
  df_dy = 2*y*np.exp(x**2)*x**2 + 4*np.exp(y**2)*y + 4*x - 2
```

[28]:
```
#Grafiquemos la función
```

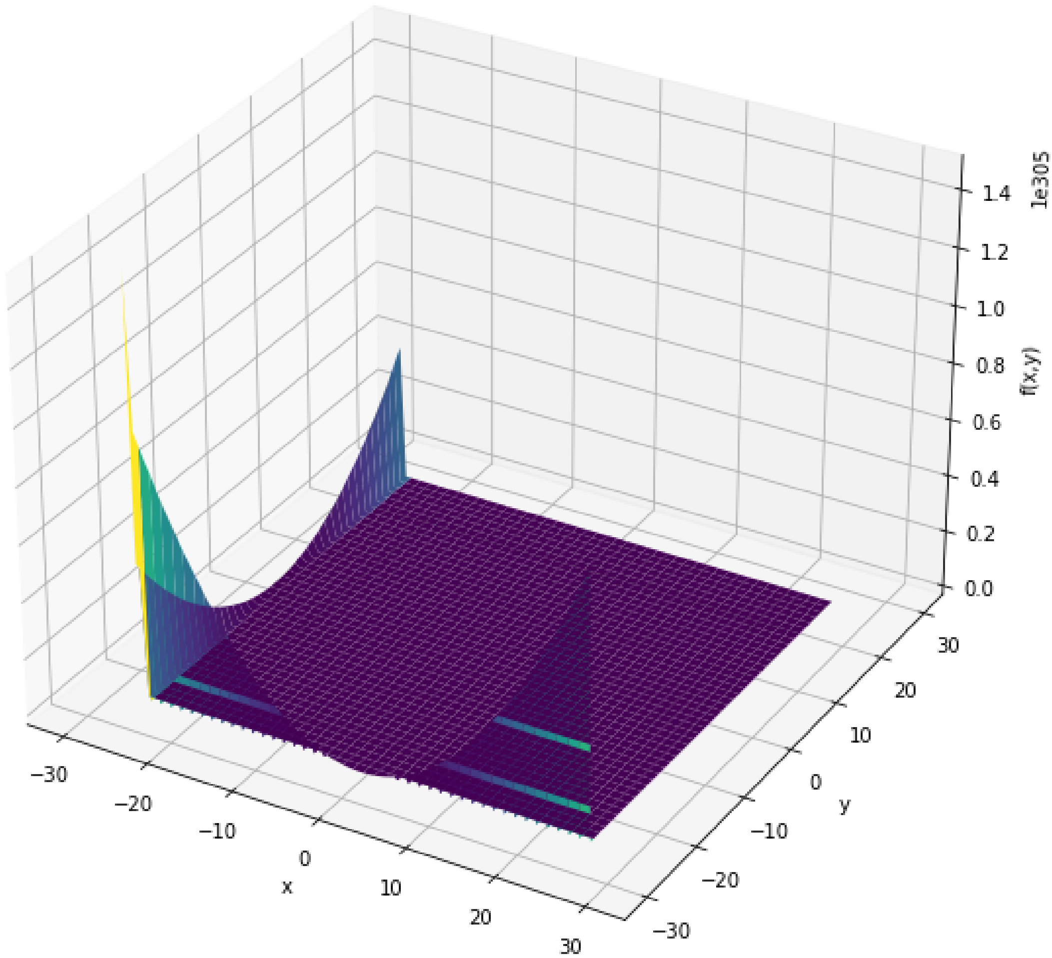
[29]:
```
x_vals = np.linspace(-30, 30, 100)
y_vals = np.linspace(-30, 30, 100)
X, Y = np.meshgrid(x_vals, y_vals)
Z = f(X, Y)

fig = plt.figure(figsize=(10, 10))
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X, Y, Z, cmap='viridis')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('f(x,y)')

# graficar el punto crítico encontrado
ax.scatter(x_star, y_star, f(x_star, y_star), color='red', s=100, marker='*')

plt.show()
```

```
C:\Users\ASUS\AppData\Local\Temp\ipykernel_13244\3691417216.py:2:
RuntimeWarning: overflow encountered in exp
  return np.exp(x**2) * y**2 + 2*np.exp(y**2) * x**2 + 4*x*y + 2*x**2 + 4*x -
2*y
C:\Users\ASUS\anaconda3\lib\site-packages\mpl_toolkits\mplot3d\proj3d.py:109:
RuntimeWarning: invalid value encountered in true_divide
  txs, tys, tzs = vecw[0]/w, vecw[1]/w, vecw[2]/w
```


1e305
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
f(x,y)
−30
−20
−10
0
10
20
30
x
30
20
10
0
−10
−20
−30
y

[ ]: