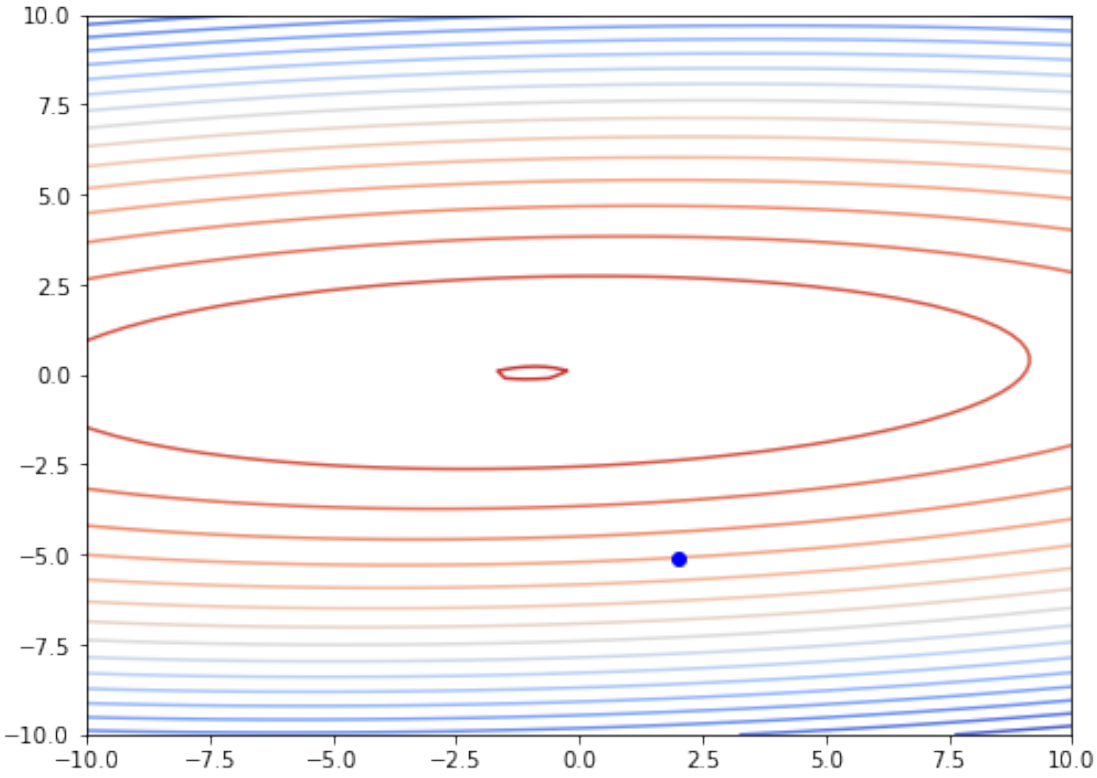
EJE3GRAD

April 3, 2023

1 Método del Gradiente

[ ]:
```python
import numpy as np
import matplotlib.pyplot as plt

# Definamos la función
def f(x, y):
    return -(4*x**2) - (57*y**2) + 4*x*y - 8*x + 10*y - 2

# Definamos el gradiente de la función
def grad_f(x, y):
    return np.array([-8*x + 4*y - 8, -114*y + 4*x + 10])

# Definamos los parámetros del método
point = np.array([2, -5]) # Punto semilla
precision = 0.00001 # Nivel de precisión
step_size = 0.1 # Longitud de paso

# Ahora creamos la figura
fig, ax = plt.subplots(figsize=(8, 6))

# Creamos la malla de puntos para graficar la función
x_range = np.linspace(-10, 10, 100)
y_range = np.linspace(-10, 10, 100)
X, Y = np.meshgrid(x_range, y_range)
Z = f(X, Y)

# Graficamos la función
ax.contour(X, Y, Z, levels=20, cmap='coolwarm')

# Iteramos hasta alcanzar el nivel de precisión
while np.linalg.norm(grad_f(point[0], point[1])) > precision:
    # Calcular el gradiente en el punto actual
    grad = grad_f(point[0], point[1])
    # Calcular la dirección del máximo aumento de la función
    direction = -grad / np.linalg.norm(grad)
    # Actualizar el punto avanzando en la dirección opuesta
```

```
    point = point + step_size * direction
    # Graficar el punto actual
    ax.plot(point[0], point[1], 'bo')
    plt.pause(0.1) # Pausar para mostrar la iteración en la gráfica

# Mostramos el punto crítico en la gráfica
ax.plot(point[0], point[1], 'ro')

# Mostramos la gráfica
plt.show()
```



[ ]: