SEMINARIO DE SOLUCION DE PROBLEMAS DE INTELIGENCIA ARTIFICIAL I

Vázquez Pérez Ignacio David

218292866

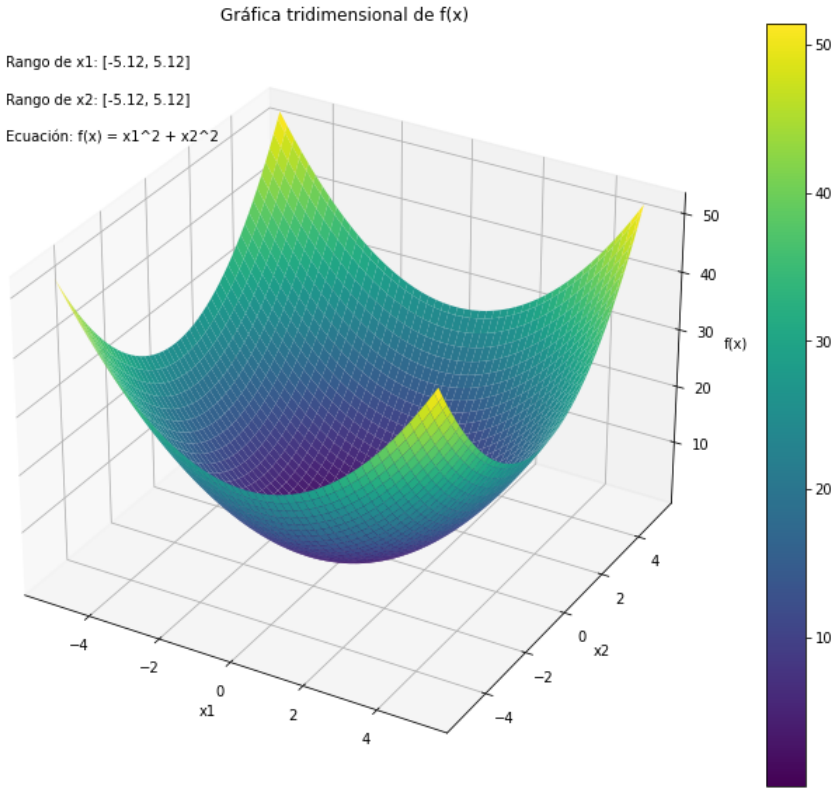
Ingeniería en computación

Funciones para probar algoritmos

Producir con Python las gráficas para las funciones descritas en el archivo adjunto.

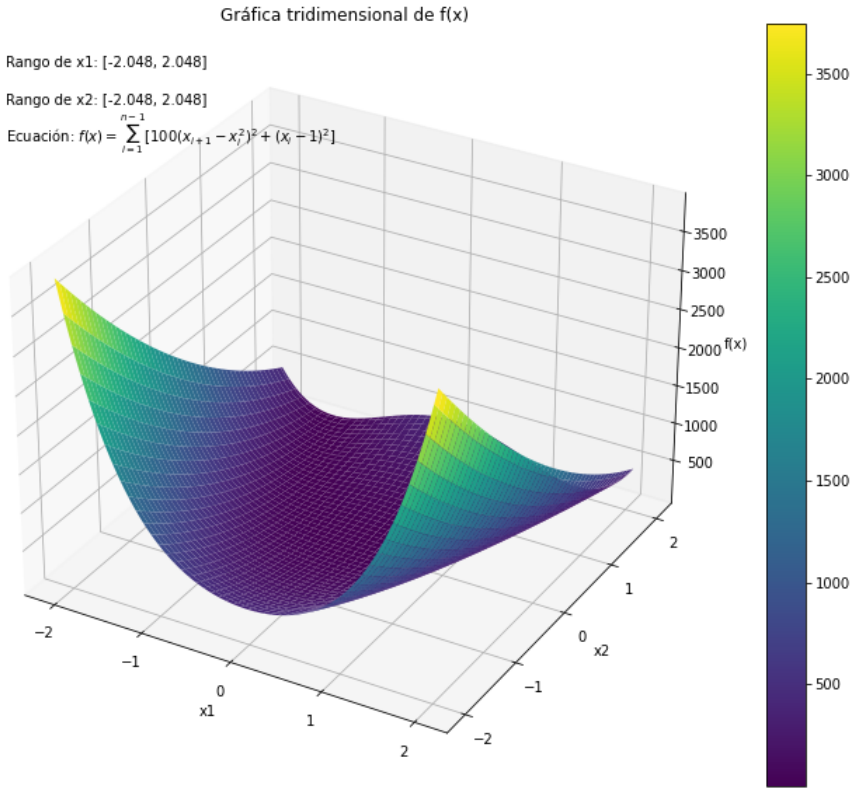
In [ ]:

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

# Definir la función f(x)
def f(x):
    return sum(xi**2 for xi in x)

# Crear una malla de valores para x1 y x2
x1_vals = np.linspace(-5.12, 5.12, 100)
x2_vals = np.linspace(-5.12, 5.12, 100)
x1, x2 = np.meshgrid(x1_vals, x2_vals)

# Calcular los valores de f(x) para cada punto en la malla
z = f([x1, x2])

# Crear la figura tridimensional
fig = plt.figure(figsize=(12, 10))
ax = fig.add_subplot(111, projection='3d')

# Graficar la superficie
surf = ax.plot_surface(x1, x2, z, cmap='viridis')

# Configurar etiquetas y título
ax.set_xlabel('x1')
ax.set_ylabel('x2')
ax.set_zlabel('f(x)')
ax.set_title('Gráfica tridimensional de f(x)')

# Mostrar la barra de colores
fig.colorbar(surf)

# Agregar los rangos de la malla de valores
ax.text2D(0.05, 0.95, f'Rango de x1: [{x1_vals.min():.2f}, {x1_vals.max():.2f}]', transform=ax.transAxes)
ax.text2D(0.05, 0.90, f'Rango de x2: [{x2_vals.min():.2f}, {x2_vals.max():.2f}]', transform=ax.transAxes)

# Agregar la ecuación
ax.text2D(0.05, 0.85, 'Ecuación: f(x) = x1^2 + x2^2', transform=ax.transAxes)

# Mostrar la gráfica
plt.show()
```



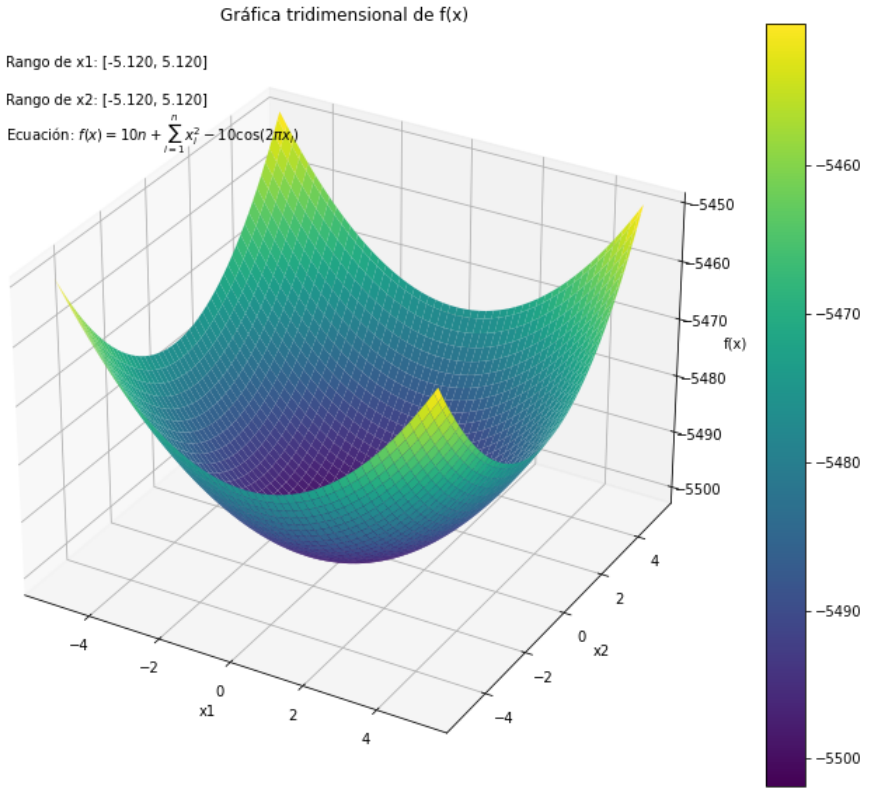
In [ ]:

```python
# Definir la función f(x)
def f(x):
    n = len(x)
    result = 0
    for i in range(n-1):
        term1 = 100 * (x[i+1] - x[i]**2)**2
        term2 = (x[i] - 1)**2
        result += term1 + term2
    return result

# Crear una malla de valores para x1 y x2
x1_vals = np.linspace(-2.048, 2.048, 100)
x2_vals = np.linspace(-2.048, 2.048, 100)
x1, x2 = np.meshgrid(x1_vals, x2_vals)

# Calcular los valores de f(x) para cada punto en la malla
z = f([x1, x2])

# Crear la figura tridimensional
fig = plt.figure(figsize=(12, 10))
ax = fig.add_subplot(111, projection='3d')

# Graficar la superficie
surf = ax.plot_surface(x1, x2, z, cmap='viridis')

# Configurar etiquetas y título
ax.set_xlabel('x1')
ax.set_ylabel('x2')
ax.set_zlabel('f(x)')
ax.set_title('Gráfica tridimensional de f(x)')

# Mostrar la barra de colores
fig.colorbar(surf)

# Agregar los rangos de la malla de valores
ax.text2D(0.05, 0.95, f'Rango de x1: [{x1_vals.min():.3f}, {x1_vals.max():.3f}]', transform=ax.transAxes)
ax.text2D(0.05, 0.90, f'Rango de x2: [{x2_vals.min():.3f}, {x2_vals.max():.3f}]', transform=ax.transAxes)

# Agregar la ecuación
ax.text2D(0.05, 0.85, r'Ecuación: $f(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$', transform=ax.transA

# Mostrar la gráfica
plt.show()
```



In [ ]:

```python
# Definir la función f(x)
def f(x):
    n = len(x)
    sum_term = sum(xi**2 for xi in x)
    cos_term = -10 * np.cos(2 * np.pi * x).sum()
    return 10 * n + sum_term + cos_term

# Crear una malla de valores para x1 y x2
x1_vals = np.linspace(-5.12, 5.12, 100)
x2_vals = np.linspace(-5.12, 5.12, 100)
x1, x2 = np.meshgrid(x1_vals, x2_vals)

# Convertir x1 y x2 en matrices unidimensionales
x1_flat = x1.ravel()
x2_flat = x2.ravel()

# Calcular los valores de f(x) para cada punto en la malla
z = f(np.array([x1_flat, x2_flat]))

# Reshape z para que tenga la misma forma que x1 y x2
z = z.reshape(x1.shape)

# Crear la figura tridimensional
fig = plt.figure(figsize=(12, 10))
ax = fig.add_subplot(111, projection='3d')

# Graficar la superficie
surf = ax.plot_surface(x1, x2, z, cmap='viridis')

# Configurar etiquetas y título
ax.set_xlabel('x1')
ax.set_ylabel('x2')
ax.set_zlabel('f(x)')
ax.set_title('Gráfica tridimensional de f(x)')

# Mostrar la barra de colores
fig.colorbar(surf)

# Agregar los rangos de la malla de valores
ax.text2D(0.05, 0.95, f'Rango de x1: [{x1_vals.min():.3f}, {x1_vals.max():.3f}]', transform=ax.transAxes)
ax.text2D(0.05, 0.90, f'Rango de x2: [{x2_vals.min():.3f}, {x2_vals.max():.3f}]', transform=ax.transAxes)

# Agregar la ecuación
ax.text2D(0.05, 0.85, r'Ecuación: $f(x) = 10n + \sum_{i=1}^{n} x_i^2 - 10 \cos(2\pi x_i)$', transform=ax.transAxes)

# Mostrar la gráfica
plt.show()
```



In [ ]:

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

# Define the function f(x)
def f(x):
    n = len(x[0])  # asumimos que x es una lista de arrays, tomamos la longitud del primer array
    sum_term = np.sum(i * xi**4 for i, xi in enumerate(x, start=1))
    return sum_term

# Crear una malla de valores para x1 y x2
x1_vals = np.linspace(-1.28, 1.28, 100)
x2_vals = np.linspace(-1.28, 1.28, 100)
x1, x2 = np.meshgrid(x1_vals, x2_vals)

# Calcular los valores de f(x) para cada punto en la malla
z = f([x1, x2])

# Crear la gráfica 3D
fig = plt.figure(figsize=(12, 10))
ax = fig.add_subplot(111, projection='3d')

# Graficar la superficie
surf = ax.plot_surface(x1, x2, z, cmap='viridis')

# Configurar etiquetas y título
ax.set_xlabel('x1')
ax.set_ylabel('x2')
ax.set_zlabel('f(x)')
ax.set_title('Gráfico 3D de f(x)')

# Mostrar la ecuación de la función
equation_text = r'$f(x) = \sum_{i=1}^{n} i x_i^4$'
ax.text2D(0.05, 0.95, equation_text, transform=ax.transAxes, fontsize=12)

# Mostrar los valores de los ejes
ax.text2D(0.05, 0.90, f'Rango de x1: [{x1_vals.min():.2f}, {x1_vals.max():.2f}]', transform=ax.transAxes, fontsize=10)
ax.text2D(0.05, 0.85, f'Rango de x2: [{x2_vals.min():.2f}, {x2_vals.max():.2f}]', transform=ax.transAxes, fontsize=10)

# Mostrar la barra de colores
fig.colorbar(surf)

# Mostrar la gráfica
plt.show()
```

```
/home/nacho/.local/lib/python3.6/site-packages/ipykernel_launcher.py:8: DeprecationWarning: Calling np.sum(generator) is deprecated, and in the future will give a different result. Use np.sum(np.fromiter(generator)) or the python sum builtin instead.
```

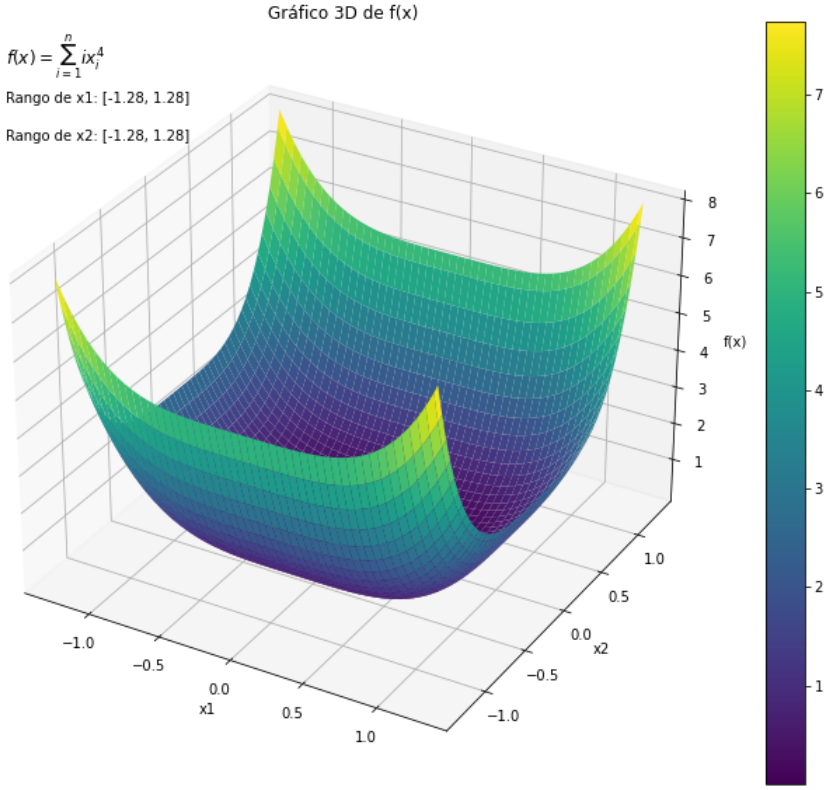


Gráfico 3D de f(x)
$f(x) = \sum_{i=1}^{n} i x_i^4$
Rango de x1: [-1.28, 1.28]
Rango de x2: [-1.28, 1.28]
f(x)
x1
x2

In [ ]: