Segura_Guillermo_Tarea8

April 29, 2024

1 Tarea 8. Optimizacion

Guillermo Segura Gomez

1.1 Ejercicio 1

Programar el método de BFGS modificado descrito en el Algoritmo 2 de la Clase 23.

1. Programe la función que implementa el algoritmo. Debe recibir como parámetros

- El punto inicial $\mathbf{x}_0$
- La matriz $\mathbf{H}_0$
- La función f
- El gradiente $\nabla f(\mathbf{x})$
- La tolerancia τ
- El número máximo de iteraciones N
- Los paramétros ρ, c_1, N_b del algoritmo de backtracking

[1]:
```
# Librerias
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
# Funcion de Backtracking
def Backtracking_DescSuf(alpha_0, rho, c1, xk, fk, gk, pk, nMax):

    for i in range(nMax):

        comp1 = fk(xk + alpha_0*pk)
        comp2 = fk(xk) + c1*alpha_0* np.dot(gk, pk)

        if (comp1 <= comp2):
            return alpha_0, i

        alpha_0 = alpha_0*rho

    return alpha_0, i
```

[3]:
```
# Método de Broyden, FLetcher, Goldfarb y Shannon
def BFGS_mod(f, gradf, x0, tau, HessAprox, nMax, alpha_0, rho, c1, nBack):
```

```
    # Valores iniciales
    dim = len(x0)
    xk = np.array(x0)
    sequence = []

    # Inicializamos gk y Hk
    gk = gradf(xk)
    Hk = HessAprox(xk)

    for k in range(nMax):

        # Convergencia
        if np.linalg.norm(gk) < tau:
            return xk, k, gk, True, sequence

        # Calculamos la direccion pk = -Hk * gk
        pk = np.dot(-Hk, gk)

        # Condición de parada
        if pk @ gk > 0:

            # Calculamos lambda
            lambda1 = (10**-5) + (pk.T @ gk) / (gk.T @ gk)

            # Actualizamos Hk
            Hk = Hk + lambda1 * np.eye(dim)

            # Redefinimos p
            # pk = -lambda1 * gk
            pk = np.dot(-Hk, gk)

        # Calculamos el tamaño de paso
        alphak, _ = Backtracking_DescSuf(alpha_0, rho, c1, xk, f, gk, pk, nBack)

        # Calculamos los valores siguientes
        xk_next = xk + alphak * pk
        sk = xk_next - xk
        yk = gradf(xk_next) - gk

        # Calculamos Hk_next
        if (yk.T @ sk) <= 0:

            if np.dot(yk, yk) > 1e-10:  # Asegura que el denominador es␣
↪suficientemente grande
                # Calculamos lambda
                lambda2 = 10**-5 - (yk.T @ sk) / (yk.T @ yk)
```

```
            else:
                lambda2 = 10**-5

            Hk_next = Hk + lambda2 * np.eye(dim)

        else:
            rhok = 1 / (yk.T @ sk) # Calculamos rho_k
            I = np.eye(dim)
            Hk_next = (I - rhok * np.outer(sk, yk)) @ Hk @ (I - rhok * np.
↪outer(yk, sk)) + rhok * np.outer(sk, sk)

        # Actualizamos los valores
        xk = xk_next
        Hk = Hk_next
        gk = gradf(xk)

        # Almacenar puntos solo para visualización en 2D
        if len(x0) == 2:
            sequence.append(xk.tolist())

    return xk, nMax, gk, False, sequence
```

2. Pruebe el algoritmo para minimizar las siguientes funciones usando los parámetros $N = 5000$, $\tau = \sqrt{n}\epsilon_m^{1/3}$, donde n es la dimensión de la variable $\mathbf{x}$, $\mathbf{H}_0$ como la matriz identidad y ϵ_m es el épsilon máquina. Para backtracking use $\rho = 0.5$, $c_1 = 0.001$ y el número máximo de iteraciones $N_b = 500$.

   En cada caso imprima los siguientes datos:

- la dimensión n,
- $f(\mathbf{x}_0)$,
- el número k de iteraciones realizadas,
- $f(\mathbf{x}_k)$,
- las primeras y últimas 4 entradas del punto $\mathbf{x}_k$ que devuelve el algoritmo,
- la norma del vector gradiente $\mathbf{g}_k$,
- la variable *res* que indica si el algoritmo terminó porque se cumplió el criterio de la tolerancia o terminó por iteraciones.

Función de cuadrática 1: Para $\mathbf{x} = (x_1, x_2, ..., x_n)$

- $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{A}_1 \mathbf{x} - \mathbf{b}_1^\top \mathbf{x}$, donde $\mathbf{A}_1$ y $\mathbf{b}_1$ están definidas por

$$\mathbf{A}_1 = n\mathbf{I} + \mathbf{1} = \begin{bmatrix} n & 0 & \cdots & 0 \\ 0 & n & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & n \end{bmatrix} + \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}, \qquad \mathbf{b}_1 = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix},$$

donde $\mathbf{I}$ es la matriz identidad y $\mathbf{1}$ es la matriz llena de 1's, ambas de tamaño n, usando los puntos iniciales
- $\mathbf{x}_0 = (0, ..., 0) \in \mathbb{R}^{10}$ - $\mathbf{x}_0 = (0, ..., 0) \in \mathbb{R}^{100}$ - $\mathbf{x}_0 = (0, ..., 0) \in \mathbb{R}^{1000}$

Función de cuadrática 2: Para $\mathbf{x} = (x_1, x_2, ..., x_n)$

- $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{A}_2 \mathbf{x} - \mathbf{b}_2^\top \mathbf{x}$, donde $\mathbf{A}_2 = [a_{ij}]$ y $\mathbf{b}_2$ están definidas por

$$a_{ij} = exp\left(-0.25(i-j)^2\right), \qquad \mathbf{b}_2 = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

usando los puntos iniciales: - $\mathbf{x}_0 = (0, ..., 0) \in \mathbb{R}^{10}$ - $\mathbf{x}_0 = (0, ..., 0) \in \mathbb{R}^{100}$ - $\mathbf{x}_0 = (0, ..., 0) \in \mathbb{R}^{1000}$

Función de Beale : Para $\mathbf{x} = (x_1, x_2)$

$$f(\mathbf{x}) = (1.5 - x_1 + x_1 x_2)^2 + (2.25 - x_1 + x_1 x_2^2)^2 + (2.625 - x_1 + x_1 x_2^3)^2.$$

- $\mathbf{x}_0 = (2, 3)$

Función de Himmelblau: Para $\mathbf{x} = (x_1, x_2)$

$$f(\mathbf{x}) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2.$$

- $\mathbf{x}_0 = (2, 4)$

Función de Rosenbrock: Para $\mathbf{x} = (x_1, x_2, ..., x_n)$

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2\right] \quad n \geq 2.$$

- $\mathbf{x}_0 = (-1.2, 1.0) \in \mathbb{R}^2$
- $\mathbf{x}_0 = (-1.2, 1.0, ..., -1.2, 1.0) \in \mathbb{R}^{20}$
- $\mathbf{x}_0 = (-1.2, 1.0, ..., -1.2, 1.0) \in \mathbb{R}^{40}$

[4]:
```
def himmelblau(x):
    return (x[0]**2 + x[1] - 11)**2 + (x[0] + x[1]**2 - 7)**2

def grad_himmelblau(x):
    df_dx1 = 4 * x[0] * (x[0]**2 + x[1] - 11) + 2 * (x[0] + x[1]**2 - 7)
    df_dx2 = 2 * (x[0]**2 + x[1] - 11) + 4 * x[1] * (x[0] + x[1]**2 - 7)
    return np.array([df_dx1, df_dx2])
```

[5]:
```
def beale(x):
    return ((1.5 - x[0] + x[0]*x[1])**2 +
            (2.25 - x[0] + x[0]*x[1]**2)**2 +
            (2.625 - x[0] + x[0]*x[1]**3)**2)
```

```
def grad_beale(x):
    x1, x2 = x
    df_dx1 = 2*(1.5 - x1 + x1*x2)*(-1 + x2) + 2*(2.25 - x1 + x1*x2**2)*(-1 +␣
 ↪x2**2) + 2*(2.625 - x1 + x1*x2**3)*(-1 + x2**3)
    df_dx2 = 2*(1.5 - x1 + x1*x2)*x1 + 2*(2.25 - x1 + x1*x2**2)*2*x1*x2 + 2*(2.
 ↪625 - x1 + x1*x2**3)*3*x1*x2**2
    return np.array([df_dx1, df_dx2])
```

[6]:
```
def rosenbrock(x):
    return sum(100*(x[1:] - x[:-1]**2)**2 + (1 - x[:-1])**2)

def grad_rosenbrock(x):
    df_dx = np.zeros_like(x)
    n = len(x)
    df_dx[:-1] += -400 * x[:-1] * (x[1:] - x[:-1]**2) + 2 * (x[:-1] - 1)  #␣
 ↪Derivadas parciales para x_i donde i < n
    df_dx[1:] += 200 * (x[1:] - x[:-1]**2)  # Derivadas parciales para x_{i+1}␣
 ↪donde i < n
    return df_dx
```

[7]:
```
# Función para visualizar los contornos de nivel de función en 2D
def contornosFnc2D(fncf, xleft, xright, ybottom, ytop, levels, secuencia=None):
    ax = np.linspace(xleft, xright, 250)
    ay = np.linspace(ybottom, ytop, 200)
    mX, mY = np.meshgrid(ax, ay)
    mZ = np.array([[fncf(np.array([x, y])) for x in ax] for y in ay])

    fig, ax = plt.subplots()
    CS = ax.contour(mX, mY, mZ, levels, cmap='viridis')
    plt.colorbar(CS, ax=ax)
    ax.set_xlabel('$x_1$')
    ax.set_ylabel('$x_2$')

    # Graficar la secuencia de puntos
    if secuencia is not None:
        secuencia = np.array(secuencia)
        ax.plot(secuencia[:, 0], secuencia[:, 1], 'r.-')  # 'r.-' para puntos␣
 ↪rojos conectados por líneas
        ax.plot(secuencia[0, 0], secuencia[0, 1], 'go')  # Punto de inicio en␣
 ↪verde
        ax.plot(secuencia[-1, 0], secuencia[-1, 1], 'bo')  # Punto final en azul

    plt.show()
```

Probamos las funciones cuadraticas

[8]:
```python
# Función para generar A1 y b1
def generate_A1_b1(n):
    A1 = n * np.eye(n) + np.ones((n, n))
    b1 = np.ones(n)
    return A1, b1

# Función para generar A2 y b2
def generate_A2_b2(n):
    A2 = np.array([[np.exp(-0.25 * (i - j) ** 2) for j in range(n)] for i in␣
 ↪range(n)])
    b2 = np.ones(n)
    return A2, b2
```

[9]:
```python
def fG(x, A, b):
    return 0.5 * np.dot(x.T, np.dot(A, x)) - np.dot(b.T, x)

def gradfG(x, A, b):
    return np.dot(A, x) - b
```

[10]:
```python
# Epsilon de la máquina
epsilon_m = np.finfo(float).eps

# Configuración de tolerancia
tau = lambda n: np.sqrt(n) * epsilon_m**(1/3)

# Parámetros iniciales
alpha_0 = 1
rho = 0.5
c1 = 0.001

# Número máximo de iteraciones para el descenso máximo y la sección dorada
NMax = 5000
NBack = 500

# Función para probar el algoritmo de newton con diferentes funciones
def probar_algoritmo(func, grad_func, hess_func, puntos_iniciales):
    for x0 in puntos_iniciales:
        xk, k, gk, convergio, secuencia = BFGS_mod(func, grad_func, x0,␣
 ↪tau(len(x0)), hess_func, NMax, alpha_0, rho, c1, NBack)
        valor_final = func(xk)
        print(f"Resultado para x0 = {x0}, f(x0) = {func(x0)}:")
        print(f"xk = {xk}, k = {k}, f(xk) = {valor_final}, convergió:␣
 ↪{convergio}")
        if len(x0) == 2 and secuencia:
            contornosFnc2D(func, xleft=-5.5, xright=5.5, ybottom=-5.5, ytop=5.
 ↪5, levels=[0.5, 5, 10, 25, 50, 100, 150, 250, 400], secuencia=secuencia)
        print()
```

```python
# Definimos la hessiana como la matriz identidad
def HessfG(x, A, b):
    return np.eye(len(x))


# Función para probar las funciones cuadráticas
def test_quadratic(n, generate):

    # Puntos iniciales
    x0 = [np.zeros(n)]

    # Generar A, b
    A, b = generate(n)

    # Generamos la función y su gradiente
    f = lambda x: fG(x, A, b)
    gradf = lambda x: gradfG(x, A, b)
    hessf = lambda x: HessfG(x, A, b)

    probar_algoritmo(f, gradf, hessf, x0)
```

[11]:
```python
# Prueba del algoritmo para f1
for n in [10, 100, 1000]:
    test_quadratic(n, generate_A1_b1)
```

```
Resultado para x0 = [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.], f(x0) = 0.0:
xk = [0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05], k = 2, f(xk) = -0.25,
convergió: True

Resultado para x0 = [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0.], f(x0) = 0.0:
xk = [0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005
 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005
 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005
 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005
 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005
 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005
 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005
 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005
 0.005 0.005 0.005 0.005], k = 2, f(xk) = -0.25, convergió: True

Resultado para x0 = [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0.
```

```
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.], f(x0) = 0.0:
xk = [0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
```

0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005

```
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005
 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005], k = 2,
f(xk) = -0.249999999999999, convergió: True
```

[12]:
```python
# Prueba del algoritmo para f2
for n in [10, 100, 1000]:
    test_quadratic(n, generate_A2_b2)
```

```
Resultado para x0 = [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.], f(x0) = 0.0:
xk = [ 1.36910165 -1.16637731  1.60908339 -0.61339229  0.59500721  0.59500721
 -0.61339229  1.60908339 -1.16637731  1.36910165], k = 18, f(xk) =
-1.7934208025210774, convergió: True

Resultado para x0 = [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0.], f(x0) = 0.0:
xk = [ 1.44628123 -1.41633442  2.11047122 -1.42499584  1.75922764 -0.94218914
  1.27161336 -0.50579374  0.90377927 -0.20568947  0.6634142  -0.01520447
  0.5133468   0.10266653  0.42086291  0.17526727  0.36377549  0.22028886
  0.32812059  0.24866329  0.30541531  0.2669251   0.29068462  0.278796
  0.28120576  0.2861968   0.27567114  0.28999556  0.27351311  0.29060975
  0.27432959  0.28853122  0.2774374   0.28469198  0.28163621  0.28054577
  0.28530736  0.27774789  0.28692149  0.27750144  0.28577981  0.2798531
  0.28257262  0.28341919  0.27922847  0.28598165  0.27790017  0.28582123
  0.27953187  0.28300641  0.28300686  0.27953356  0.28582332  0.27790045
  0.28597989  0.27922662  0.28341828  0.28257274  0.2798546   0.28578197
  0.27750236  0.2869205   0.27774613  0.28530617  0.28054581  0.28163744
  0.2846937   0.27743844  0.28853056  0.27432738  0.29060782  0.27351294
  0.28999669  0.27567272  0.28619854  0.2812063   0.27879419  0.29068244
  0.26692492  0.30541621  0.24866412  0.32812216  0.22028987  0.36377344
  0.17526427  0.42086251  0.10266744  0.51334733 -0.01520242  0.66341665
 -0.20569075  0.90377612 -0.50579414  1.27161411 -0.94218941  1.75922935
 -1.4249928   2.11047032 -1.41633792  1.44628086], k = 139, f(xk) =
-14.494330069283095, convergió: True

Resultado para x0 = [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
```

```
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.], f(x0) = 0.0:
xk = [ 1.44628094 -1.41635864  2.11049968 -1.42504235  1.75929773 -0.94229003
  1.27174691 -0.50596459  0.90399897 -0.20597266  0.66376779 -0.01564678
  0.51390271  0.1019896   0.42170171  0.17424909  0.36499026  0.21884764
  0.32980818  0.24673174  0.30760251  0.26449906  0.29332536  0.27598108
  0.28414194  0.28321336  0.27862435  0.2871614   0.27614624  0.28826973
  0.2763037   0.28699162  0.27847732  0.284203    0.28154955  0.28123149
  0.284053    0.27949677  0.28481076  0.27976836  0.28363767  0.28156431
  0.28161087  0.28337787  0.28039637  0.28375565  0.28089649  0.28255365
  0.28243201  0.28112857  0.28334983  0.28096785  0.28274292  0.28209603
  0.28149136  0.28304088  0.28117817  0.28264293  0.28209433  0.2815808
  0.28287619  0.28139008  0.28242958  0.28224822  0.28154823  0.28275623
  0.28163525  0.28214817  0.28244698  0.28153156  0.28256653  0.28195353
  0.28185365  0.28256753  0.28165826  0.2822488   0.28229384  0.28167984
  0.28247304  0.28197503  0.28189222  0.28247558  0.28178514  0.2821429
  0.28232547  0.28174598  0.28232082  0.28213904  0.28182652  0.28239059
  0.28197664  0.28195504  0.28237964  0.28188322  0.28208136  0.28231005
  0.281845    0.28218367  0.28223196  0.28185256  0.28224343  0.28215092
  0.28188507  0.28228057  0.28209086  0.28192167  0.28228661  0.28204657
  0.28196386  0.28228692  0.28201698  0.28199028  0.2822756   0.28200557
```

0.28201734 0.28226425 0.28199559 0.28202736 0.28225551 0.28200262
0.28203519 0.28224282 0.28200513 0.28203413 0.2822397 0.28202077
0.28202606 0.28222722 0.28203539 0.28201984 0.28222286 0.28205581
0.28200367 0.28220773 0.28208297 0.28199799 0.28219187 0.28210649
0.28198679 0.28216921 0.28214002 0.28198992 0.28213586 0.28216064
0.28199947 0.28210476 0.28218323 0.28202033 0.28206244 0.28218678
0.28205764 0.2820355 0.28217643 0.28209329 0.28201199 0.28215102
0.28213914 0.28201401 0.28210564 0.28216185 0.28203664 0.28206773
0.28217257 0.2820741 0.28202854 0.28215054 0.2821248 0.28202612
0.28210867 0.28215184 0.282044 0.28206542 0.28216302 0.28208979
0.28203066 0.28212938 0.28213608 0.28204129 0.28208541 0.28215277
0.2820723 0.28204524 0.28214041 0.28212564 0.28203843 0.28208784
0.28214898 0.28207728 0.2820524 0.28213247 0.28211966 0.28204422
0.28208871 0.28214906 0.28208357 0.28204557 0.28211983 0.28213158
0.28205944 0.28207448 0.28213676 0.28209768 0.28204935 0.28210666
0.2821397 0.28207212 0.28205452 0.28212387 0.28212597 0.2820619
0.28207556 0.2821302 0.28210109 0.28205694 0.28209977 0.2821346
0.28208169 0.28205443 0.28211247 0.28213304 0.2820749 0.28206157
0.28211628 0.28212088 0.28207076 0.28207604 0.28212258 0.28210767
0.28206261 0.28208507 0.28213039 0.28210362 0.28205751 0.28208514
0.28213109 0.28210508 0.28206117 0.28208462 0.28212447 0.28210273
0.28206797 0.28209004 0.28212045 0.28209599 0.28206732 0.28209482
0.2821253 0.28209636 0.28206124 0.28208815 0.28212858 0.28210769
0.28206387 0.28207616 0.28211979 0.28211623 0.28207628 0.28207334
0.28210788 0.28211414 0.28208434 0.28207713 0.28210455 0.28211246
0.28208496 0.28207442 0.28210288 0.28211907 0.28209113 0.28206678
0.28209058 0.28212367 0.28210868 0.28206894 0.28207082 0.28211244
0.28212599 0.28208919 0.28206118 0.28208625 0.28212396 0.28211554
0.28207589 0.28206676 0.28209991 0.28212294 0.28210273 0.28207283
0.28207772 0.28210664 0.28211523 0.28209357 0.28207633 0.28208753
0.28210762 0.28210695 0.28208919 0.2820816 0.28209351 0.2821057
0.28210119 0.28208842 0.28208602 0.2820958 0.28210285 0.28209818
0.28208977 0.2820892 0.28209581 0.28209984 0.28209671 0.28209188
0.28209173 0.28209528 0.28209717 0.28209551 0.28209343 0.28209378
0.28209539 0.28209563 0.28209436 0.28209372 0.28209474 0.28209594
0.28209557 0.28209406 0.28209337 0.28209437 0.28209583 0.28209605
0.28209486 0.28209365 0.28209368 0.28209477 0.28209573 0.28209568
0.2820948 0.28209395 0.28209386 0.28209452 0.28209534 0.28209562
0.28209511 0.28209424 0.28209376 0.28209414 0.28209507 0.28209572
0.28209544 0.28209445 0.28209365 0.28209381 0.28209483 0.28209578
0.28209576 0.28209476 0.28209369 0.28209355 0.28209449 0.28209569
0.28209606 0.28209523 0.28209393 0.28209334 0.282094 0.28209538
0.28209625 0.28209582 0.28209444 0.28209329 0.28209342 0.28209474
0.28209614 0.28209638 0.28209522 0.28209362 0.28209295 0.28209383
0.28209554 0.28209665 0.28209617 0.28209448 0.28209295 0.2820929
0.2820944 0.28209625 0.28209691 0.28209581 0.28209381 0.28209252
0.28209302 0.28209495 0.2820968 0.28209707 0.28209552 0.28209334
0.28209224 0.28209311 0.28209532 0.28209718 0.28209725 0.28209545

0.2820931 0.28209199 0.28209295 0.28209532 0.28209735 0.28209754
0.28209574 0.2820932 0.28209175 0.28209244 0.28209478 0.28209716
0.28209791 0.28209649 0.28209385 0.28209179 0.28209168 0.28209361
0.28209631 0.28209801 0.28209759 0.28209532 0.28209265 0.28209125
0.28209202 0.28209447 0.28209709 0.28209831 0.28209737 0.28209486
0.28209225 0.28209107 0.282092 0.28209449 0.28209714 0.28209847
0.28209775 0.28209539 0.28209267 0.28209105 0.28209136 0.28209344
0.28209619 0.28209821 0.2820985 0.28209693 0.28209428 0.28209184
0.28209076 0.28209153 0.28209377 0.28209645 0.28209836 0.28209865
0.28209722 0.28209472 0.2820922 0.28209073 0.28209089 0.2820926
0.28209516 0.28209755 0.28209887 0.28209862 0.28209693 0.28209444
0.28209206 0.28209064 0.28209065 0.28209208 0.28209442 0.28209689
0.2820987 0.28209927 0.28209846 0.28209652 0.28209407 0.28209182
0.28209043 0.28209028 0.28209138 0.28209343 0.28209584 0.28209799
0.2820993 0.28209947 0.28209845 0.28209652 0.28209415 0.28209192
0.28209035 0.2820898 0.2820904 0.28209199 0.2820942 0.28209653
0.28209846 0.28209954 0.28209954 0.28209846 0.28209653 0.2820942
0.28209199 0.2820904 0.2820898 0.28209035 0.28209192 0.28209415
0.28209652 0.28209845 0.28209947 0.2820993 0.28209799 0.28209584
0.28209343 0.28209138 0.28209028 0.28209043 0.28209182 0.28209407
0.28209652 0.28209846 0.28209927 0.2820987 0.28209689 0.28209442
0.28209208 0.28209065 0.28209064 0.28209206 0.28209444 0.28209693
0.28209862 0.28209887 0.28209755 0.28209516 0.2820926 0.28209089
0.28209073 0.2820922 0.28209472 0.28209722 0.28209865 0.28209836
0.28209645 0.28209377 0.28209153 0.28209076 0.28209184 0.28209428
0.28209693 0.2820985 0.28209821 0.28209619 0.28209344 0.28209136
0.28209105 0.28209267 0.28209539 0.28209775 0.28209847 0.28209714
0.28209449 0.282092 0.28209107 0.28209225 0.28209486 0.28209737
0.28209831 0.28209709 0.28209447 0.28209202 0.28209125 0.28209265
0.28209532 0.28209759 0.28209801 0.28209631 0.28209361 0.28209168
0.28209179 0.28209385 0.28209649 0.28209791 0.28209716 0.28209478
0.28209244 0.28209175 0.2820932 0.28209574 0.28209754 0.28209735
0.28209532 0.28209295 0.28209199 0.2820931 0.28209545 0.28209725
0.28209718 0.28209532 0.28209311 0.28209224 0.28209334 0.28209552
0.28209707 0.2820968 0.28209495 0.28209302 0.28209252 0.28209381
0.28209581 0.28209691 0.28209625 0.2820944 0.2820929 0.28209295
0.28209448 0.28209617 0.28209665 0.28209554 0.28209383 0.28209295
0.28209362 0.28209522 0.28209638 0.28209614 0.28209474 0.28209342
0.28209329 0.28209444 0.28209582 0.28209625 0.28209538 0.282094
0.28209334 0.28209393 0.28209523 0.28209606 0.28209569 0.28209449
0.28209355 0.28209369 0.28209476 0.28209576 0.28209578 0.28209483
0.28209381 0.28209365 0.28209445 0.28209544 0.28209572 0.28209507
0.28209414 0.28209376 0.28209424 0.28209511 0.28209562 0.28209534
0.28209452 0.28209386 0.28209395 0.2820948 0.28209568 0.28209573
0.28209477 0.28209368 0.28209365 0.28209486 0.28209605 0.28209583
0.28209437 0.28209337 0.28209406 0.28209557 0.28209594 0.28209474
0.28209372 0.28209436 0.28209563 0.28209539 0.28209378 0.28209343
0.28209551 0.28209717 0.28209528 0.28209173 0.28209188 0.28209671

0.28209984 0.28209581 0.2820892 0.28208977 0.28209818 0.28210285
0.2820958 0.28208602 0.28208842 0.28210119 0.2821057 0.28209351
0.2820816 0.28208919 0.28210695 0.28210762 0.28208753 0.28207633
0.28209357 0.28211523 0.28210664 0.28207772 0.28207283 0.28210273
0.28212294 0.28209991 0.28206676 0.28207589 0.28211554 0.28212396
0.28208625 0.28206118 0.28208919 0.28212599 0.28211244 0.28207082
0.28206894 0.28210868 0.28212367 0.28209058 0.28206678 0.28209113
0.28211907 0.28210288 0.28207442 0.28208496 0.28211246 0.28210455
0.28207713 0.28208434 0.28211414 0.28210788 0.28207334 0.28207628
0.28211623 0.28211979 0.28207616 0.28206387 0.28210769 0.28212858
0.28208815 0.28206124 0.28209636 0.2821253 0.28209482 0.28206732
0.28209599 0.28212045 0.28209004 0.28206797 0.28210273 0.28212447
0.28208462 0.28206117 0.28210508 0.28213109 0.28208514 0.28205751
0.28210362 0.28213039 0.28208507 0.28206261 0.28210767 0.28212258
0.28207604 0.28207076 0.28212088 0.28211628 0.28206157 0.2820749
0.28213304 0.28211247 0.28205443 0.28208169 0.2821346 0.28209977
0.28205694 0.28210109 0.2821302 0.28207556 0.2820619 0.28212597
0.28212387 0.28205452 0.28207212 0.2821397 0.28210666 0.28204935
0.28209768 0.28213676 0.28207448 0.28205944 0.28213158 0.28211983
0.28204557 0.28208357 0.28214906 0.28208871 0.28204422 0.28211966
0.28213247 0.2820524 0.28207728 0.28214898 0.28208784 0.28203843
0.28212564 0.28214041 0.28204524 0.2820723 0.28215277 0.28208541
0.28204129 0.28213608 0.28212938 0.28203066 0.28208979 0.28216302
0.28206542 0.282044 0.28215184 0.28210867 0.28202612 0.2821248
0.28215054 0.28202854 0.2820741 0.28217257 0.28206773 0.28203664
0.28216185 0.28210564 0.28201401 0.28213914 0.28215102 0.28201199
0.28209329 0.28217643 0.2820355 0.28205764 0.28218678 0.28206244
0.28202033 0.28218323 0.28210476 0.28199947 0.28216064 0.28213586
0.28198992 0.28214002 0.28216921 0.28198679 0.28210649 0.28219187
0.28199799 0.28208297 0.28220773 0.28200367 0.28205581 0.28222286
0.28201984 0.28203539 0.28222722 0.28202606 0.28202077 0.2822397
0.28203413 0.28200513 0.28224282 0.28203519 0.28200262 0.28225551
0.28202736 0.28199559 0.28226425 0.28201734 0.28200557 0.2822756
0.28199028 0.28201698 0.28228692 0.28196386 0.28204657 0.28228661
0.28192167 0.28209086 0.28228057 0.28188507 0.28215092 0.28224343
0.28185256 0.28223196 0.28218367 0.281845 0.28231005 0.28208136
0.28188322 0.28237964 0.28195504 0.28197664 0.28239059 0.28182652
0.28213904 0.28232082 0.28174598 0.28232547 0.2821429 0.28178514
0.28247558 0.28189222 0.28197503 0.28247304 0.28167984 0.28229384
0.2822488 0.28165826 0.28256753 0.28185365 0.28195353 0.28256653
0.28153156 0.28244698 0.28214817 0.28163525 0.28275623 0.28154823
0.28224822 0.28242958 0.28139008 0.28287619 0.2815808 0.28209433
0.28264293 0.28117817 0.28304088 0.28149136 0.28209603 0.28274292
0.28096785 0.28334983 0.28112857 0.28243201 0.28255365 0.28089649
0.28375565 0.28039637 0.28337787 0.28161087 0.28156431 0.28363767
0.27976836 0.28481076 0.27949677 0.284053 0.28123149 0.28154955
0.284203 0.27847732 0.28699162 0.2763037 0.28826973 0.27614624
0.2871614 0.27862435 0.28321336 0.28414194 0.27598108 0.29332536

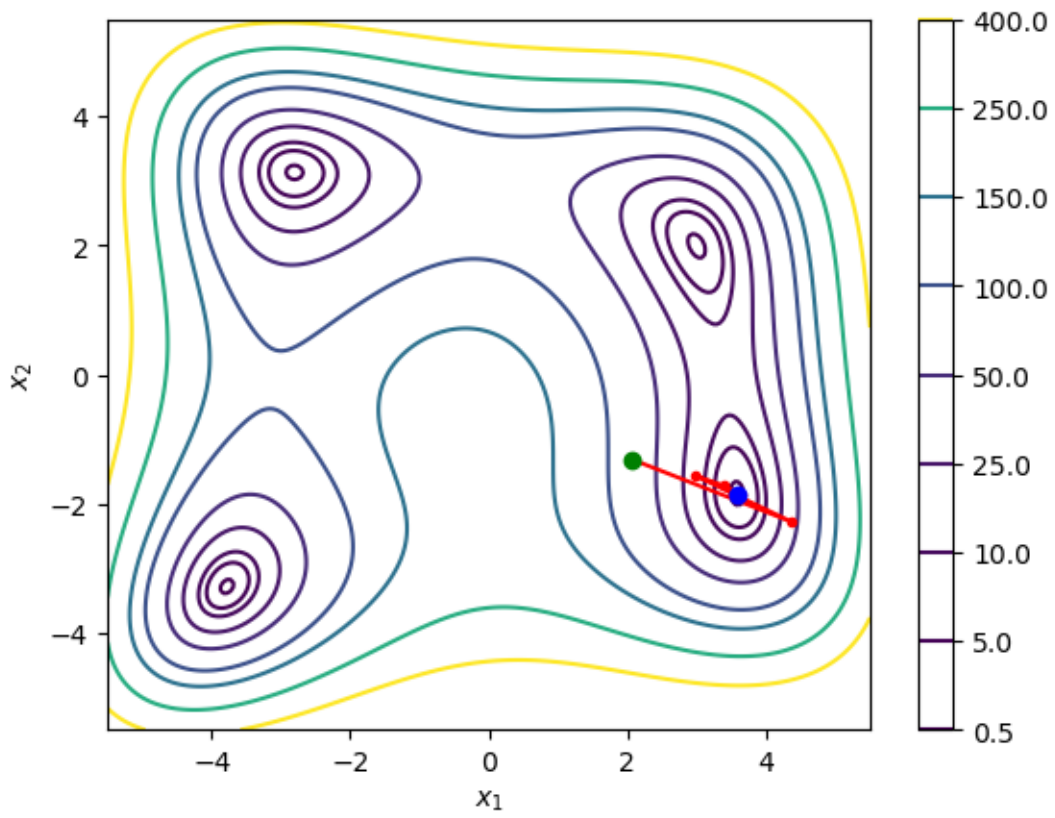
```
  0.26449906  0.30760251  0.24673174  0.32980818  0.21884764  0.36499026
  0.17424909  0.42170171  0.1019896   0.51390271 -0.01564678  0.66376779
 -0.20597266  0.90399897 -0.50596459  1.27174691 -0.94229003  1.75929773
 -1.42504235  2.11049968 -1.41635864  1.44628094], k = 257, f(xk) =
-141.43698680561425, convergió: True
```

Probamos las otras funciones

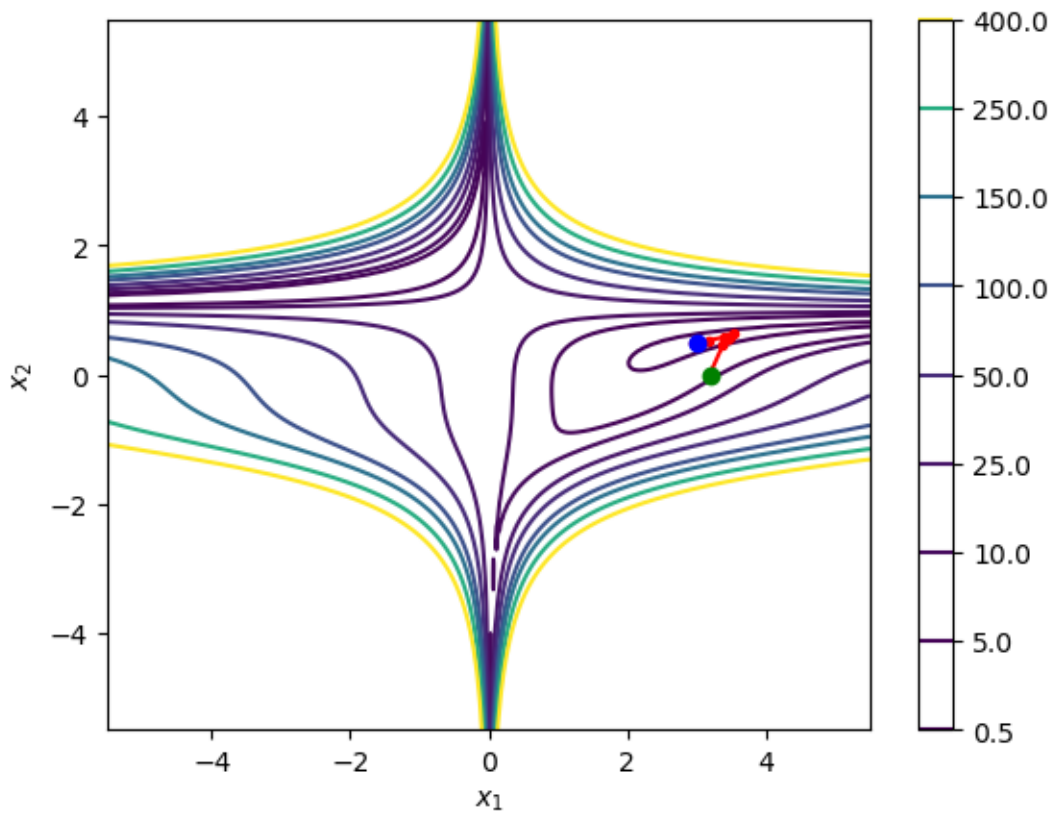
[13]:
```
# Puntos iniciales para la función de Himmelblau
puntos_iniciales_himmelblau = [np.array([2.0, 4.0])]

# Puntos iniciales para la función de Beale
puntos_iniciales_beale = [np.array([0.0, 0.0]), np.array([2.0, 3.0])]

# Puntos iniciales para la función de Rosenbrock
puntos_iniciales_rosenbrock = [
    np.array([-1.2, 1.0]),
    np.array([-1.2 if i % 2 == 0 else 1.0 for i in range(20)]),
    np.array([-1.2 if i % 2 == 0 else 1.0 for i in range(50)])
]

# Epsilon de la máquina
epsilon_m = np.finfo(float).eps

# Configuración de tolerancia
tau = lambda n: np.sqrt(n) * epsilon_m**(1/3)

# Parámetros iniciales
alpha_0 = 1
rho = 0.5
c1 = 0.001

# Número máximo de iteraciones para el descenso máximo y la sección dorada
NMax = 5000
NBack = 500

# Función para probar el algoritmo de newton con diferentes funciones
def probar_algoritmo(func, grad_func, hess_func, puntos_iniciales):
    for x0 in puntos_iniciales:
        xk, k, gk, convergio, secuencia = BFGS_mod(func, grad_func, x0,␣
 ↪tau(len(x0)), hess_func, NMax, alpha_0, rho, c1, NBack)
        valor_final = func(xk)
        print(f"Resultado para x0 = {x0}, f(x0) = {func(x0)}:")
        print(f"xk = {xk}, k = {k}, f(xk) = {valor_final}, convergió:␣
 ↪{convergio}")
        if len(x0) == 2 and secuencia:
```

```
            contornosFnc2D(func, xleft=-5.5, xright=5.5, ybottom=-5.5, ytop=5.
 ↪5, levels=[0.5, 5, 10, 25, 50, 100, 150, 250, 400], secuencia=secuencia)
        print()

# Probamos como hessiana la funcion identidad
def hessian_identidad(x):
    return np.eye(len(x))

# Probar con la función de Himmelblau
print("Función de Himmelblau:")
probar_algoritmo(himmelblau, grad_himmelblau, hessian_identidad,␣
 ↪puntos_iniciales_himmelblau)

# Probar con la función de BealeNewtonTruncado(func, grad_func, hess_func, x0,␣
 ↪tau(len(x0)), NMax, alpha_0, rho, c1, NBack)
print("Función de Beale:")
probar_algoritmo(beale, grad_beale, hessian_identidad, puntos_iniciales_beale)

# Probar con la función de Rosenbrock
print("Función de Rosenbrock:")
probar_algoritmo(rosenbrock, grad_rosenbrock, hessian_identidad,␣
 ↪puntos_iniciales_rosenbrock)
```

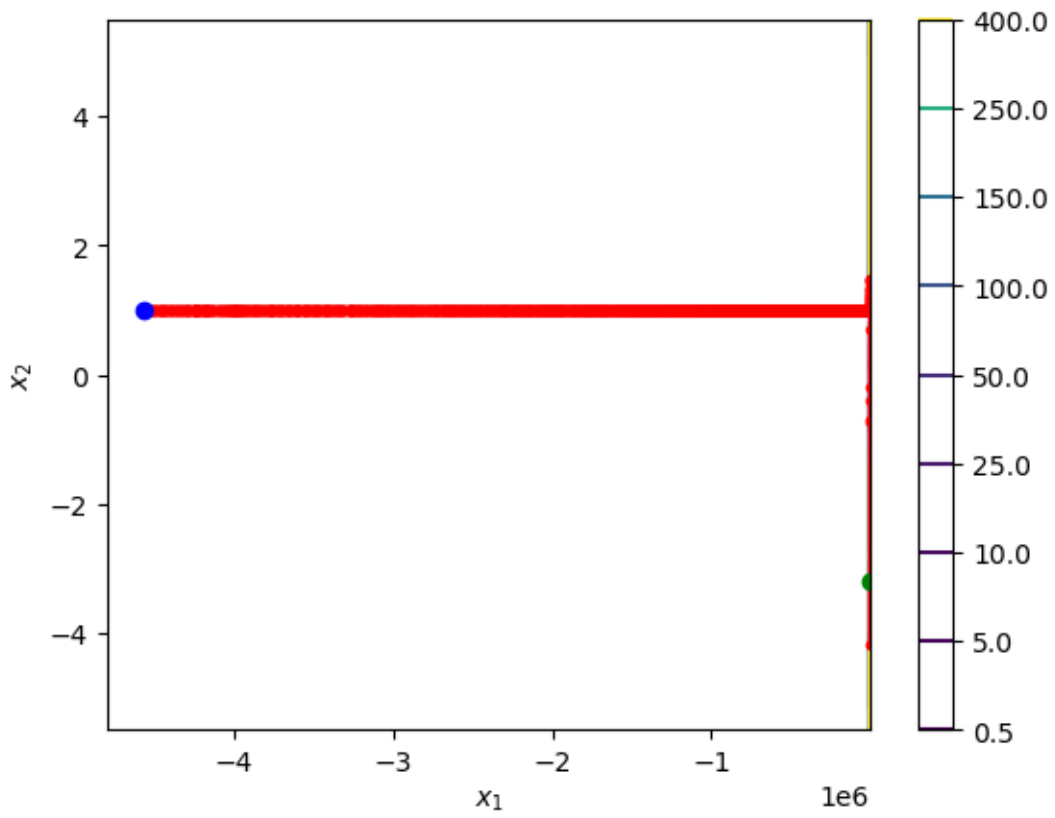
```
Función de Himmelblau:
Resultado para x0 = [2. 4.], f(x0) = 130.0:
xk = [ 3.58442834 -1.84812653], k = 10, f(xk) = 9.834452856641356e-16,
convergió: True
```



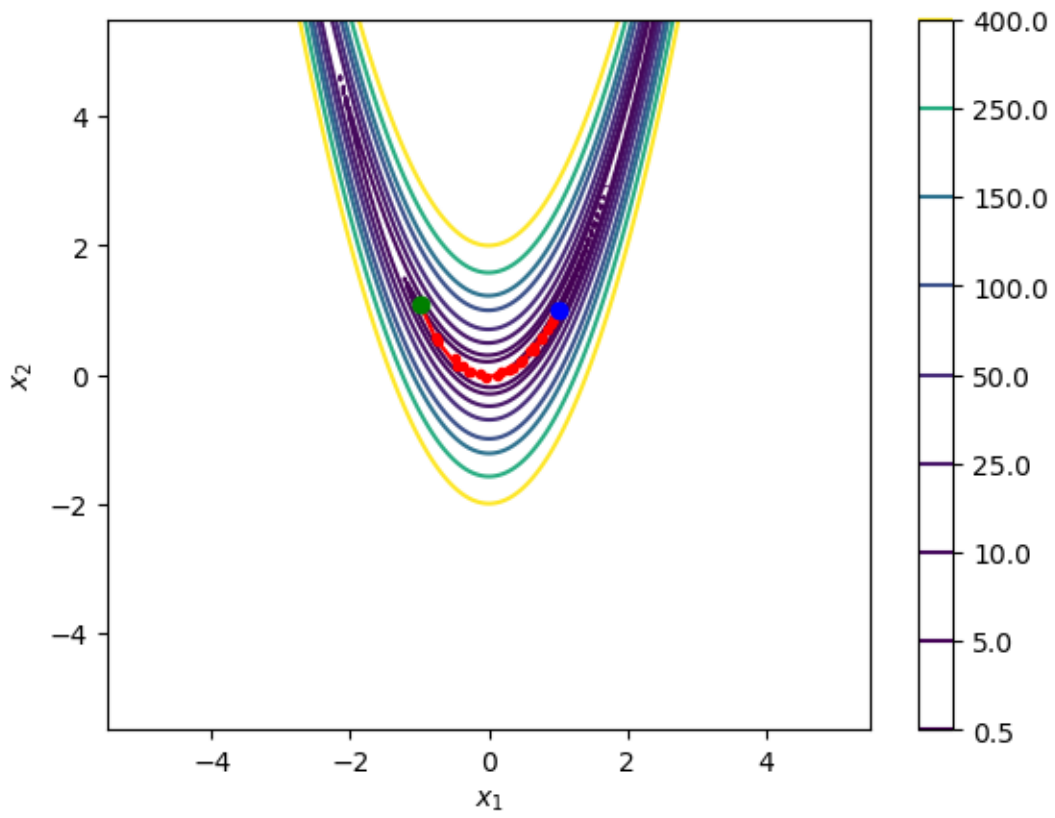

```
Función de Beale:
Resultado para x0 = [0. 0.], f(x0) = 14.203125:
xk = [3.0000002  0.50000006], k = 18, f(xk) = 8.665292270724936e-15, convergió:
True
```




```
Resultado para x0 = [2. 3.], f(x0) = 3347.203125:
xk = [-4.57030483e+06  1.00000022e+00], k = 5000, f(xk) = 0.45200926135300823,
convergió: False
```




```
Función de Rosenbrock:
Resultado para x0 = [-1.2  1. ], f(x0) = 24.199999999999996:
xk = [1.         0.99999999], k = 34, f(xk) = 2.745636868826416e-17, convergió:
True
```




```
Resultado para x0 = [-1.2  1.  -1.2  1.  -1.2  1.  -1.2  1.  -1.2  1.  -1.2  1.
-1.2  1.
 -1.2  1.  -1.2  1.  -1.2  1. ], f(x0) = 4597.999999999999:
xk = [1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         0.99999999 0.99999998
 0.99999998 0.99999995], k = 129, f(xk) = 1.808629645082156e-14, convergió: True

Resultado para x0 = [-1.2  1.  -1.2  1.  -1.2  1.  -1.2  1.  -1.2  1.  -1.2  1.
-1.2  1.
 -1.2  1.  -1.2  1.  -1.2  1.  -1.2  1.  -1.2  1.  -1.2  1.  -1.2  1.
 -1.2  1.  -1.2  1.  -1.2  1.  -1.2  1.  -1.2  1.  -1.2  1.  -1.2  1.
 -1.2  1.  -1.2  1.  -1.2  1.  -1.2  1. ], f(x0) = 12221.000000000004:
xk = [1.         1.         1.         1.         0.99999999 1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.00000001 1.
 1.         1.         0.99999999 1.         0.99999999 1.
```

```
0.99999999 1.         1.         1.         1.         0.99999999
0.99999999 0.99999998], k = 272, f(xk) = 8.271715705521489e-14, convergió: True
```

Discusion

El método BFGS modificado convergio en general para todas las funciones, solo tuvo problemas con la función de Beale desde el punto inicial [2.0, 3.0], en donde el metodo no convergio. Agregue el punto inicial [0.0, 0.0] y el algoritmo convergio. Este comportamiento puede atribuirse a la sensibilidad del método a la aproximación inicial de la matriz Hessiana inversa (H_0), crucial para su rendimiento. Dado que la función de Beale presenta valles estrechos y curvaturas complicadas, un punto inicial inadecuado puede llevar a una mala estimación de H_0, afectando la dirección de búsqueda y la convergencia. Esto nos lleva a considerar la importancia de una elección cuidadosa del punto inicial y de H_0 en métodos cuasi-Newton, especialmente en funciones con complejidades topológicas significativas.