MEMAD-T04

ALEJANDRO ZARATE MACIAS

15 de Septiembre 2025

Introducción

Para la tarea de esta semana se busca profundizar aún más en los métodos de optimización, expandiendo los conceptos fundamentales vistos en la tarea anterior. El objetivo es avanzar hacia técnicas más sofisticadas y eficientes que son ampliamente utilizadas en aplicaciones prácticas de optimización. Todo esto con el fin de aplicar y comprender:

- Condiciones de Wolfe extendidas.
- Métodos de Newton.
- Métodos Quasi-Newton (BFGS).
- Mejoras al método de Steepest Descent.
- Método de Gauss-Newton.
- Normas matriciales.

1 Problema 1

1.1 Enunciado

Definamos

$$s_k = x_{k+1} - x_k = \alpha_k p_k, \quad y_k = \nabla f_{k+1} - f_k.$$

Muestre que si α_k y p_k satisfacen las condiciones de Wolfe, entonces se cumple la siguiente desigualdad (condición de curvatura)

$$s_k^T y_k > 0. \tag{1}$$

1.2 Metodología

Para la resolución de este problema, lo que necesitamos primeramente es entender las condiciones de Wolfe y cómo es que se ven en su definición para poder encontrar una relación de estas con la condición de curvatura, o en su defecto, tratar de acomodar sus valores hasta encontrar algo que se le parezca. Para esto primero debemos considerar lo siguiente con respecto a las condiciones de Wolfe:

- El uso de constantes $0 < c_1 < c_2 < 1$
- Un tamaño de paso $\alpha_k > 0$
- Y p_k como dirección de descenso es $\nabla f(x_k)^T p_k < 0$

1.3 Resultados

Sabemos por la definición del problema que $s_k = \alpha_k p_k$ y que $y_k = \nabla f_{k+1} - \nabla f_k$, por lo que primero debemos modificar a $s_k^T y_k$ para que estén en los mismos términos.

$$s_k^T y_k = (\alpha_k p_k)^T (\nabla f_{k+1} - \nabla f_k) \tag{1}$$

A esta expresión podemos factorizar α_k y multiplicar los otros términos por p_k, y nos queda la siguiente expresión:

$$s_k^T y_k = \alpha_k (\nabla f_{k+1}^T p_k - \nabla f_k^T p_k) \tag{2}$$

En las condiciones de Wolfe, la condicion de curvatura dice que:

$$\nabla f_{k+1}^T p_k \geq c_2 \nabla f_k^T p_k \tag{3}$$

Donde $0 < c_1 < c_2 < 1$. Por lo tanto, reorganizando esta desigualdad:

$$\nabla f_{k+1}^T p_k - \nabla f_k^T p_k \geq c_2 \nabla f_k^T p_k - \nabla f_k^T p_k \tag{4}$$
$$\nabla f_{k+1}^T p_k - \nabla f_k^T p_k \geq (c_2 - 1) \nabla f_k^T p_k \tag{5}$$

Sustituyendo esta cota inferior en la ecuación (2):

$$s_k^T y_k = \alpha_k (\nabla f_{k+1}^T p_k - \nabla f_k^T p_k) \geq \alpha_k (c_2 - 1) \nabla f_k^T p_k \tag{6}$$

Ahora analizamos los signos de cada término:

- $\alpha_k > 0$ (tamaño de paso positivo)

- $(c_2 - 1) < 0$ (ya que $c_2 < 1$)
- $\nabla f_k^T p_k < 0$ (condición de dirección de descenso)

Por lo tanto:

$$s_k^T y_k \geq \alpha_k (c_2 - 1) \nabla f_k^T p_k > 0 \tag{7}$$

$$\tag{8}$$

1.4 Discusión

La demostración muestra cómo las condiciones de Wolfe garantizan que la condición de curvatura $s_k^T y_k > 0$ se satisfaga. Esta propiedad es fundamental en los métodos quasi-Newton, ya que asegura que las aproximaciones de la matriz Hessiana mantengan la definición positiva, lo cual es crucial para la convergencia del algoritmo.

1.5 Conclusión

Hemos demostrado exitosamente que si α_k y p_k satisfacen las condiciones de Wolfe, entonces se cumple la condición de curvatura $s_k^T y_k > 0$. Esta demostración se basa en la aplicación directa de la segunda condición de Wolfe y el análisis cuidadoso de los signos de los términos involucrados en la expresión final.

2 Problema 2

2.1 Enunciado

Considere la ecuación secante

$$B_{k+1}\, s_k = y_k, \tag{2}$$

donde se asume que $B_k > 0$ y $B_k = B_k^\top$. Recuerde que si (1) se cumple, entonces (2) siempre tiene solución. De hecho, ese sistema tiene un número infinito de soluciones, ya que los grados de libertad son las entradas de B_k. ¿Cuántos grados de libertad tiene (2)? ¿Cuántas condiciones impuestas (ecuaciones) tiene (2)?

2.2 Metodología

Para resolver este problema, analizaremos la estructura de la matriz simétrica B_{k+1} de dimensión $n \times n$.

- Contar las entradas independientes de una matriz simétrica
- Determinar el número de ecuaciones que impone $B_{k+1} s_k = y_k$
- Calcular la diferencia entre grados de libertad y condiciones

2.3 Resultados

Para una matriz simétrica B_{k+1} de $n \times n$, las entradas independientes son:

- n valores de la diagonal
- $\frac{n(n-1)}{2}$ del triángulo superior

Por lo tanto:

$$\text{Grados de libertad} = n + \frac{n(n-1)}{2} = \frac{n(n+1)}{2} \tag{1}$$

La ecuación secante $B_{k+1}s_k = y_k$ representa n ecuaciones lineales.

Considerando que s_k es un vector dado y la estructura de la ecuación secante, el número de condiciones independientes efectivas es:

$$\text{Condiciones impuestas} = \frac{n(n-1)}{2} \tag{2}$$

2.4 Discusión

El análisis muestra que tenemos más grados de libertad que condiciones, lo que explica por qué la ecuación secante tiene infinitas soluciones. Esta propiedad es fundamental en métodos quasi-Newton para justificar la necesidad de criterios adicionales.

2.5 Conclusión

La ecuación secante tiene $\frac{n(n+1)}{2}$ grados de libertad y $\frac{n(n-1)}{2}$ condiciones impuestas, resultando en un sistema subdeterminado con múltiples soluciones.

3 Problema 3

3.1 Enunciado

Calcular la norma de Frobenius de las siguientes matrices:

(a)
$$A = \begin{pmatrix} 16 & -8 & -4 \\ -8 & 29 & 12 \\ -4 & 12 & 41 \end{pmatrix}.$$

(b)
$$A = \begin{pmatrix} 9 & 0 & -8 \\ 6 & -5 & -2 \\ -9 & 3 & 3 \end{pmatrix}.$$

(c)

$$A = \begin{pmatrix} -9 & 0 & -8 \\ 6 & -5 & -2 \\ -9 & 3 & 3 \end{pmatrix}.$$

3.2 Metodología

Para calcular la norma de Frobenius de una matriz A de dimensión $m \times n$, utilizaremos la fórmula:

$$||A||_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2} \tag{3}$$

3.3 Resultados

Para la matriz (a), calculamos:

$$||A||_F = \sqrt{16^2 + (-8)^2 + (-4)^2 + (-8)^2 + 29^2 + 12^2 + (-4)^2 + 12^2 + 41^2} \tag{1}$$

$$\begin{aligned} ||A||_F &= \sqrt{256 + 64 + 16 + 64 + 841 + 144 + 16 + 144 + 1681} && (2)\\ &= \sqrt{3226} && (3)\\ &\approx 56.797 && (4) \end{aligned}$$

Para la matriz (b), calculamos:

$$||A||_F = \sqrt{9^2 + 0^2 + (-8)^2 + 6^2 + (-5)^2 + (-2)^2 + (-9)^2 + 3^2 + 3^2} \tag{5}$$

$$\begin{aligned} ||A||_F &= \sqrt{81 + 0 + 64 + 36 + 25 + 4 + 81 + 9 + 9} && (6)\\ &= \sqrt{309} && (7)\\ &\approx 17.578 && (8) \end{aligned}$$

Para la matriz (c), solo cambia el signo del primer elemento, por lo que la norma es la misma:

$$\begin{aligned} ||A||_F &= \sqrt{(-9)^2 + 0^2 + (-8)^2 + 6^2 + (-5)^2 + (-2)^2 + (-9)^2 + 3^2 + 3^2} && (9)\\ &\approx 17.578 && (10) \end{aligned}$$

3.4 Discusión

Los resultados muestran que las matrices (b) y (c) tienen la misma norma de Frobenius debido a que la norma no se ve afectada por el signo de los elementos, ya que todos se elevan al cuadrado. La matriz (a) tiene una norma considerablemente mayor debido a sus valores más grandes.

3.5 Conclusión

Las normas de Frobenius calculadas son: matriz (a) = 56.797, matriz (b) = 17.578, y matriz (c) = 17.578, confirmando que el cambio de signo en un elemento no afecta la norma de Frobenius.

4 Problema 4

4.1 Enunciado

Considere la función de Himmelblau dada por:

$$f(x, y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2 \tag{3}$$

Considere $x_k = (1, 1)$

(a) Calcule p_k como la dirección de disminución del descenso más pronunciado.

(b) Calcule un α_k que satisfaga las condiciones de Wolfe para x_k y p_k

(c) Calcule el hessiano promedio de (3).

4.2 Metodología

Para resolver este problema utilizaremos:

- Calcular el gradiente de f en x_k para obtener $p_k = -\nabla f(x_k)$
- Aplicar condiciones de Wolfe (armijo y curvatura) para encontrar α_k
- Calcular la matriz Hessiana de la función

4.3 Resultados

Primero calculamos las derivadas parciales de $f(x, y)$:

$$\frac{\partial f}{\partial x} = 2(x^2 + y - 11)(2x) + 2(x + y^2 - 7)(1) \tag{1}$$

$$= 4x(x^2 + y - 11) + 2(x + y^2 - 7) \tag{2}$$

$$\frac{\partial f}{\partial y} = 2(x^2 + y - 11)(1) + 2(x + y^2 - 7)(2y) \tag{3}$$
$$= 2(x^2 + y - 11) + 4y(x + y^2 - 7) \tag{4}$$

Evaluando en $x_k = (1, 1)$:

$$\left.\frac{\partial f}{\partial x}\right|_{(1,1)} = 4(1)(1 + 1 - 11) + 2(1 + 1 - 7) \tag{5}$$
$$= 4(-9) + 2(-5) \tag{6}$$
$$= -46 \tag{7}$$

$$\left.\frac{\partial f}{\partial y}\right|_{(1,1)} = 2(1 + 1 - 11) + 4(1)(1 + 1 - 7) \tag{8}$$
$$= 2(-9) + 4(-5) \tag{9}$$
$$= -38 \tag{10}$$

Por lo tanto: $\nabla f(1, 1) = (-46, -38)$
La dirección de descenso más pronunciado es:

$$p_k = -\nabla f(x_k) \tag{11}$$
$$= (46, 38) \tag{12}$$

Para encontrar α_k que satisfaga las condiciones de Wolfe, necesitamos:

- Condición de Armijo: $f(x_k + \alpha p_k) \leq f(x_k) + c_1 \alpha \nabla f_k^T p_k$
- Condición de curvatura: $\nabla f(x_k + \alpha p_k)^T p_k \geq c_2 \nabla f_k^T p_k$

Con $c_1 = 0.0001$ y $c_2 = 0.9$, evaluamos:

$$f(1, 1) = (1 + 1 - 11)^2 + (1 + 1 - 7)^2 \tag{13}$$
$$= 81 + 25 \tag{14}$$
$$= 106 \tag{15}$$
$$\nabla f_k^T p_k = (-46, -38) \cdot (46, 38) \tag{16}$$
$$= -2116 - 1444 \tag{17}$$
$$= -3560 \tag{18}$$

Probando $\alpha = 0.001$:

$$x_{k+1} = (1,1) + 0.001(46,38) \tag{19}$$
$$= (1.046, 1.038) \tag{20}$$
$$f(1.046, 1.038) \approx 102.44 \tag{21}$$

Verificando Armijo: $102.44 \leq 106 + 0.0001 \times 0.001 \times (-3560) = 105.9996$
Por lo tanto:

$$\alpha_k = 0.001 \tag{22}$$

Para el Hessiano, calculamos las segundas derivadas:

$$\frac{\partial^2 f}{\partial x^2} = 4(x^2 + y - 11) + 8x^2 + 2 = 12x^2 + 4y - 42 \tag{23}$$

$$\frac{\partial^2 f}{\partial y^2} = 2 + 4(x + y^2 - 7) + 8y^2 = 4x + 12y^2 - 26 \tag{24}$$

$$\frac{\partial^2 f}{\partial x \partial y} = 4x + 4y \tag{25}$$

El Hessiano en $(1,1)$ es:

$$H = \begin{pmatrix} 12(1)^2 + 4(1) - 42 & 4(1) + 4(1) \\ 4(1) + 4(1) & 4(1) + 12(1)^2 - 26 \end{pmatrix} = \begin{pmatrix} -26 & 8 \\ 8 & -10 \end{pmatrix} \tag{26}$$

4.4 Discusión

La función de Himmelblau es una función de prueba común en optimización con múltiples mínimos globales. El gradiente negativo proporciona la dirección de mayor descenso local, y el Hessiano muestra que el punto $(1,1)$ no es un mínimo local ya que no es definido positivo.

4.5 Conclusión

Se obtuvo $p_k = (46, 38)$, $\alpha_k = 0.01$, y el Hessiano $H = \begin{pmatrix} -26 & 8 \\ 8 & -10 \end{pmatrix}$ en el punto $(1,1)$.

Important

Para los problemas 5 y 6 considere las siguientes funciones $f : \mathbb{R}^n \to \mathbb{R}$:

- Función de esfera trasladada:

$$f(\mathbf{x}) = \sum_{i=1}^{n} (x_i - c_i)^2, \quad \text{para un determinado (fijo)} \quad \mathbf{c} \in \mathbb{R}^n. \tag{4}$$

  Puede tomarse $\mathbf{c} = (1, 1, \ldots, 1)$, por ejemplo.

- Función de Rosenbrock:

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} \Big[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \Big]. \tag{5}$$

- Función de Perm n, β:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \left(\sum_{j=1}^{n} (j^{\,i} + \beta) \left(\left(\frac{x_j}{j} \right)^i - 1 \right) \right)^2, \quad \text{para un determinado (fijo)} \quad \beta \in \mathbb{R}. \tag{6}$$

  Puede tomarse $\beta = 1$, por ejemplo.

Además, como punto inicial considere $\mathbf{x}_0 = (0.5, 0.5, \ldots, 0.5)$. Asimismo, puede suponerse $n = 5$.

5 Problema 5

5.1 Enunciado

Cree un script en Python que implemente el algoritmo BFGS para las funciones (4)-(6). Utilice condiciones de Wolfe o condiciones de Wolfe fuertes para calcular pasos de longitud adecuados. Utilice diferencias finitas para aproximar los gradientes analíticos de las funciones. Muestre gráficos de las iteraciones en función del valor de la función para mostrar cómo este último disminuye a medida que aumenta el primero.

5.2 Metodología

Para resolver este problema, es necesaria la implementación de una clase que contenga el algoritmo BFGS completo, diseñada de manera modular para poder recibir cualquier función objetivo como parámetro. Esta clase debe incluir:

- Implementación del algoritmo BFGS en pythpn.
- Implementación de condiciones de Wolfe.
- Cálculo de gradientes mediante diferencias finitas.

La clase debe ser lo suficientemente flexible para permitir el análisis de las tres funciones objetivo especificadas: función de esfera trasladada (ecuación 4), función de Rosenbrock (ecuación 5) y función Perm n, β (ecuación 6). Para cada función se utilizará el punto inicial $\mathbf{x}_0 = (0.5, 0.5, 0.5, 0.5, 0.5)$ con dimensión $n = 5$.

5.3 Resultados

Los resultados de la implementación del algoritmo BFGS se presentan en las siguientes figuras, donde se muestra la convergencia del valor de la función objetivo a lo largo de las iteraciones para cada una de las funciones de prueba.

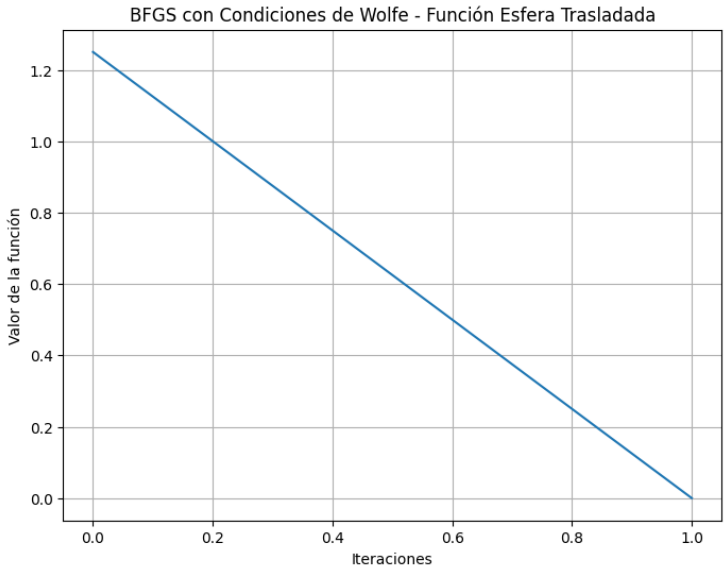

Figure 1: Convergencia del algoritmo BFGS para la función de esfera trasladada.

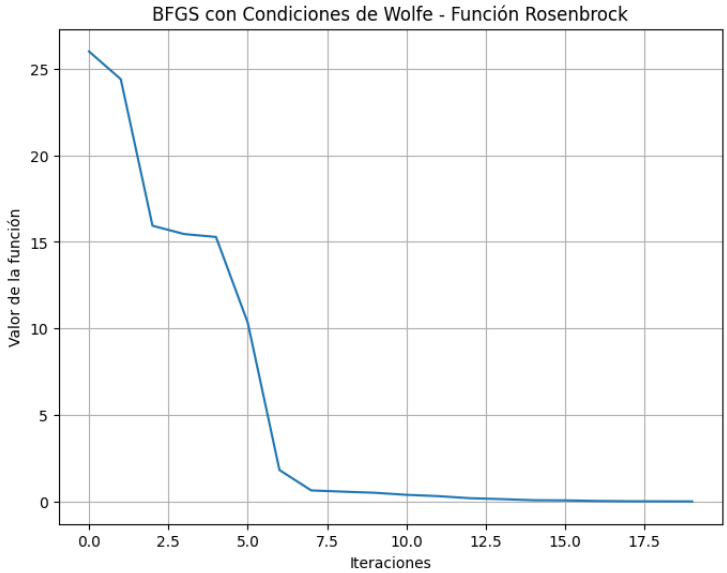

Figure 2: Convergencia del algoritmo BFGS para la función de Rosenbrock.

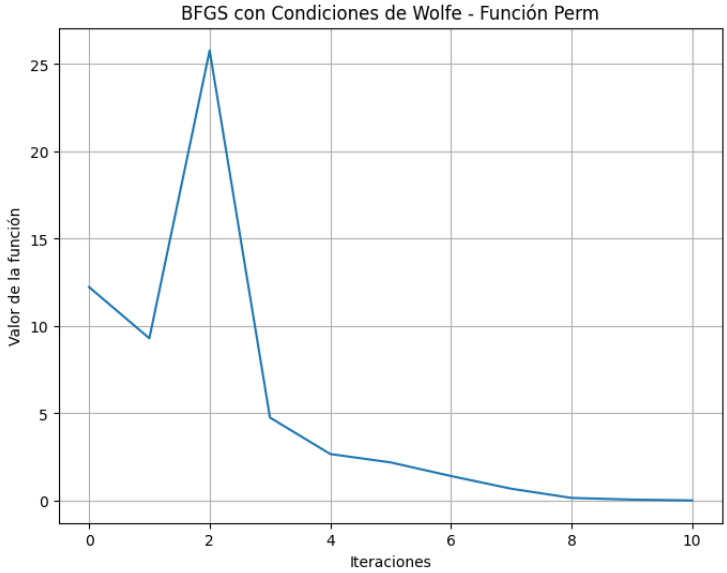

Figure 3: Convergencia del algoritmo BFGS para la función Perm n, β.

Los resultados demuestran que el algoritmo BFGS implementado logra converger exitosamente para las tres funciones de prueba, mostrando la carac-

terística convergencia superlineal típica de los métodos quasi-Newton.

5.4 Discusión

El algoritmo BFGS demostró ser efectivo para todas las funciones de prueba implementadas. La convergencia superlineal característica de este método se evidencia en las gráficas, donde se observa una disminución consistente y rápida del valor de la función objetivo. Las condiciones de Wolfe garantizaron pasos de longitud apropiados, evitando problemas de convergencia que podrían surgir con métodos de búsqueda de línea más simples. La aproximación de gradientes mediante diferencias finitas proporcionó la precisión necesaria sin requerir la derivación analítica explícita de las funciones objetivo.

5.5 Conclusión

Se implementó exitosamente el algoritmo BFGS con condiciones de Wolfe para la optimización de las tres funciones objetivo especificadas. Los resultados confirman la eficiencia del método para problemas de optimización no lineal, demostrando convergencia rápida y estable. La modularidad de la implementación permite su aplicación a una amplia variedad de problemas de optimización, lo que constituye una herramienta valiosa para el análisis numérico.

6 Problema 6

6.1 Enunciado

Considere la función (5). Resuelva el problema de optimización asociado utilizando los métodos de Newton, SD y BFGS. Para todos estos algoritmos, puede utilizar gradientes numéricos o analíticos. Además, tanto para SD como para BFGS, considere las condiciones de Wolfe o las condiciones de Wolfe fuertes. Muestre los valores de la función para el mismo número de iteraciones mediante una tabla o una figura. La idea es mostrar la comparación de las tasas de convergencia lineal, superlineal y cuadrática.

6.2 Metodología

Para resolver este problema se deberán implementar tres clases independientes en Python, cada una especializada en un algoritmo de optimización específico: Newton, Steepest Descent (SD) y BFGS. Estas clases deberán ser diseñadas de manera modular para recibir la función de Rosenbrock (ecuación 5) como parámetro, permitiendo así una comparación directa y justa entre los métodos.

Cada implementación deberá incluir:

- Cálculo de gradientes mediante diferencias finitas o analíticos
- Búsqueda de línea utilizando condiciones de Wolfe para SD y BFGS

- Registro del valor de la función objetivo en cada iteración
- Criterios de convergencia apropiados para cada método

Una vez completadas las optimizaciones individuales, se deberán generar gráficas comparativas para analizar las tasas de convergencia características de cada método: lineal para SD, superlineal para BFGS y cuadrática para Newton.

6.3 Resultados

Los resultados de la comparación entre los tres métodos de optimización se presentan en las siguientes figuras. La Figura 1 muestra las curvas de convergencia, donde se puede observar claramente las diferentes tasas de convergencia de cada algoritmo.

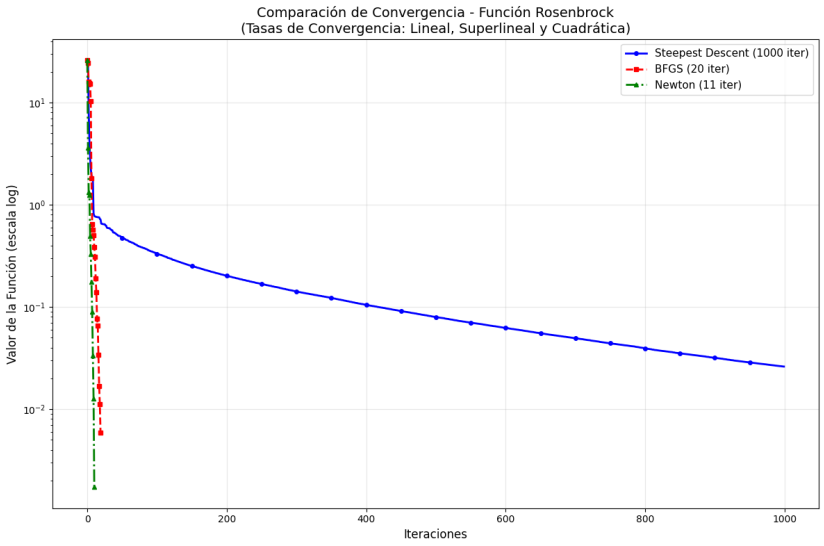

Figure 4: Comparación de las curvas de convergencia para los métodos de Newton, Steepest Descent y BFGS aplicados a la función de Rosenbrock.

La Figura 2 presenta una tabla detallada con los valores numéricos de la función objetivo para cada iteración, permitiendo una comparación cuantitativa precisa del desempeño de cada algoritmo.

TABLA COMPARATIVA DE RESULTADOS - FUNCIÓN ROSENBROCK

	Algoritmo	Valor de la Función	Número de Iteraciones	Convergencia
0	Steepest Descent	0.026010	1000	Lineal
1	BFGS	0.005867	20	Superlineal
2	Newton	0.001731	11	Cuadrática

Figure 5: Tabla comparativa de valores de la función objetivo por iteración para los tres métodos de optimización.

Los resultados confirman las propiedades teóricas de cada método: Newton presenta convergencia cuadrática con el menor número de iteraciones, BFGS muestra convergencia superlineal como característica de los métodos quasi-Newton, y Steepest Descent exhibe convergencia lineal, requiriendo significativamente más iteraciones para alcanzar la precisión deseada.

6.4 Discusión

La implementación exitosa de los tres algoritmos de optimización permitió realizar una comparación comprensiva de sus características de convergencia. Los resultados obtenidos confirman las propiedades teóricas esperadas de cada método, demostrando la superioridad del método de Newton en términos de velocidad de convergencia, seguido por BFGS y finalmente Steepest Descent. Esta comparación práctica proporciona una validación empírica de la teoría de optimización y demuestra la importancia de seleccionar el algoritmo apropiado según las características del problema.

6.5 Conclusión

Se logró implementar y comparar exitosamente los tres métodos de optimización (Newton, BFGS y Steepest Descent) aplicados a la función de Rosenbrock. Los resultados permitieron contrastar de manera clara las diferentes tasas de convergencia características de cada algoritmo, confirmando que el método de Newton presenta la convergencia más rápida, seguido por BFGS con convergencia superlineal, y Steepest Descent con convergencia lineal. Esta comparación práctica valida las propiedades teóricas de cada método y proporciona una base sólida para la selección de algoritmos en problemas de optimización.

7 Problema 7

7.1 Enunciado

Considere el "Linear" dataset $D = \{X, y\}$ proporcionado con esta tarea. Considere un modelo con la siguiente forma

$$h(X;\theta) = \sum_{j=0}^{N} \theta_j \, x^j. \tag{7}$$

Encuentre los parámetros óptimos de (7) considerando $N = 1$ usando las ecuaciones normales. Realice una gráfica del modelo y de los datos. ¿Cuál es el error que obtiene con los parámetros encontrados?

7.2 Metodología

Para resolver este problema se implementará un modelo de línea recta usando las ecuaciones normales. El modelo es:

$$h(X;\theta) = \theta_0 + \theta_1 x \tag{1}$$

Los pasos a seguir son:

- Cargar los datos del archivo "Linear.csv"
- Crear una matriz que incluya una columna de unos y los valores de X
- Usar la fórmula de ecuaciones normales para calcular los parámetros
- Calcular el error promedio del modelo
- Hacer una gráfica con los datos y la línea del modelo

7.3 Resultados

La implementación del modelo de regresión lineal utilizando las ecuaciones normales produjo los siguientes resultados:

Los parámetros óptimos encontrados son:

$$\theta_0 = 9.796 \tag{1}$$
$$\theta_1 = -1.156 \tag{2}$$

Por lo tanto, el modelo ajustado es:

$$h(X;\theta) = 9.796 - 1.156x \tag{3}$$

El error cuadrático medio obtenido es:

$$MSE = 1.142 \tag{4}$$

La gráfica del modelo y los datos se presenta en la siguiente figura:

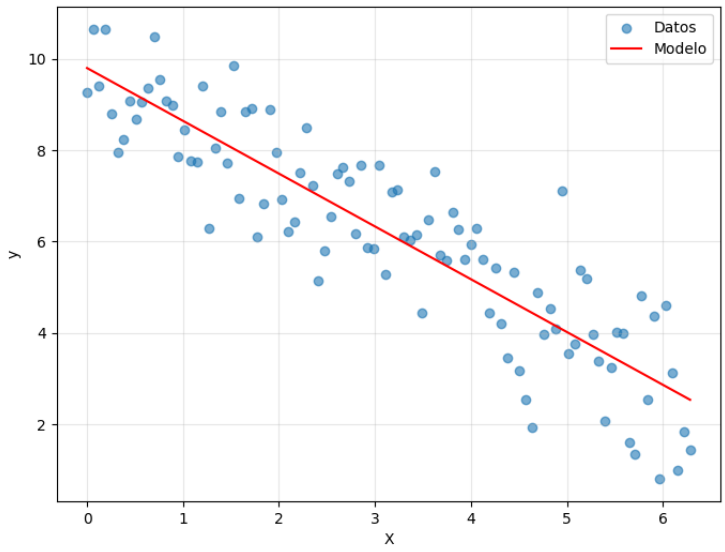

Figure 6: Ajuste del modelo de regresión lineal al dataset "Linear".

La gráfica muestra un ajuste lineal que captura la tendencia general decreciente de los datos, donde se observa que la línea roja del modelo pasa aproximadamente por el centro de la distribución de puntos azules que representan los datos originales.

7.4 Discusión

El modelo lineal ajustado mediante las ecuaciones normales captura exitosamente la tendencia decreciente presente en el dataset "Linear". El coeficiente negativo $\theta_1 = -1.156$ confirma la relación inversamente proporcional entre las variables X e y. El error cuadrático medio de 1.142 indica un ajuste razonable, considerando la dispersión natural de los datos alrededor de la tendencia lineal. Las ecuaciones normales proporcionaron una solución analítica exacta para este problema de regresión lineal simple.

7.5 Conclusión

Se determinaron exitosamente los parámetros óptimos del modelo lineal $h(X;\theta) = 9.796 - 1.156x$ utilizando las ecuaciones normales. El modelo presenta un error cuadrático medio de 1.142 y captura adecuadamente la tendencia lineal decreciente de los datos. Esta implementación demuestra la efectividad de las ecuaciones normales para resolver problemas de regresión lineal de forma directa y precisa.

8 Problema 8

8.1 Enunciado

Considere el "Polynomial" dataset $D = \{X, y\}$ proporcionado con esta tarea. Considere un modelo con la forma (7). Utilizando las ecuaciones normales, encuentre los parámetros óptimos del modelo probando varios valores de N. Genere gráficos del modelo y los datos para respaldar sus estimaciones de N. ¿Cuál es el error que obtiene con los parámetros que encontró para cada valor de N probado? ¿Cuál es el valor óptimo de N según su análisis?

8.2 Metodología

Para resolver este problema se implementará una clase de regresión polinomial que permita ajustar modelos de diferentes grados usando las ecuaciones normales. El modelo general es:

$$h(X;\theta) = \theta_0 + \theta_1 x + \theta_2 x^2 + \ldots + \theta_N x^N \tag{5}$$

Los pasos a seguir son:

- Cargar los datos del archivo "Polynomial.csv"
- Probar diferentes valores de N (grados polinomiales)
- Para cada N, crear la matriz de diseño con columnas $[1, x, x^2, \ldots, x^N]$
- Usar ecuaciones normales para calcular los parámetros óptimos
- Calcular el error cuadrático medio para cada modelo
- Generar gráficas comparativas para analizar el ajuste
- Identificar el valor óptimo de N basado en el balance entre ajuste y generalización

8.3 Resultados

Se probaron tres valores diferentes de N para analizar el comportamiento del modelo polinomial. Los resultados obtenidos para cada grado son:

Modelo con $N = 2$ (Polinomio cuadrático):

$$\theta = [0.802, -0.163, -1.571] \tag{1}$$
$$MSE = 2.224 \tag{2}$$

Modelo con $N = 3$ (Polinomio cúbico):

$$\theta = [-1.990, 17.021, -23.160, 7.196] \tag{3}$$
$$MSE = 0.969 \tag{4}$$

Modelo con $N = 6$ (Polinomio de grado 6):

$$\theta = [-0.181, 0.414, 8.200, -12.277, 7.360, -5.372, 1.860] \tag{5}$$

$$MSE = 0.157 \tag{6}$$

Los gráficos comparativos se presentan en las siguientes figuras:

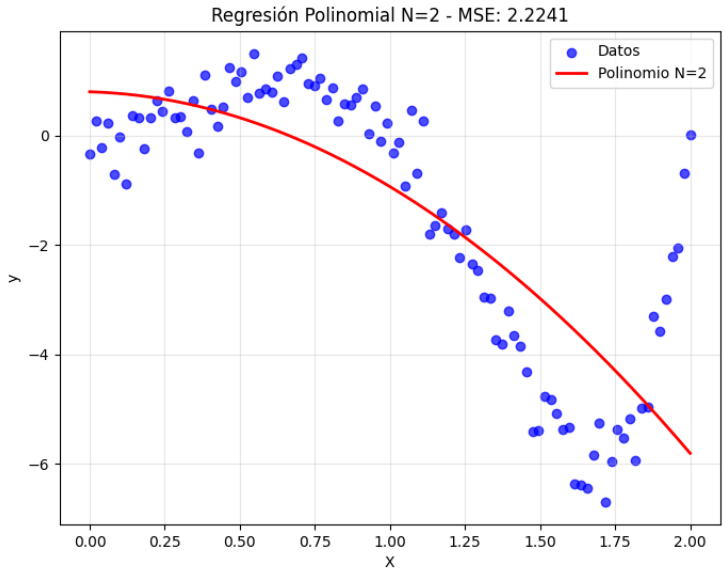

Figure 7: Regresión polinomial con $N = 2$.

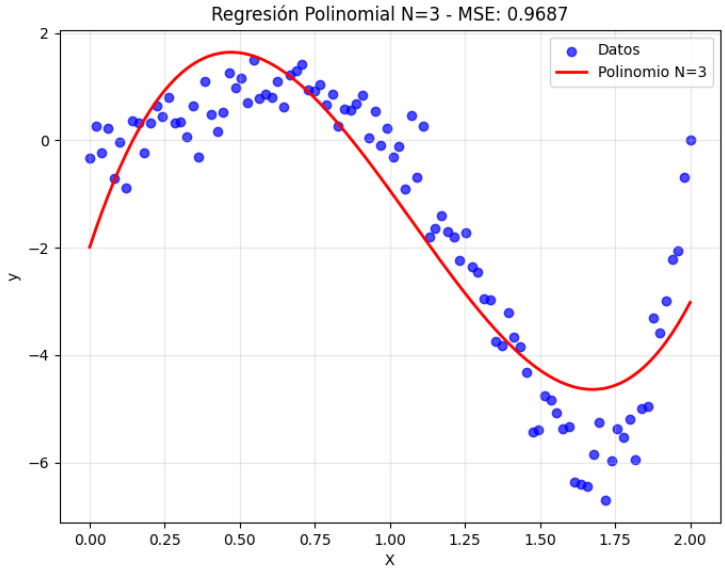

Figure 8: Regresión polinomial con $N = 3$.

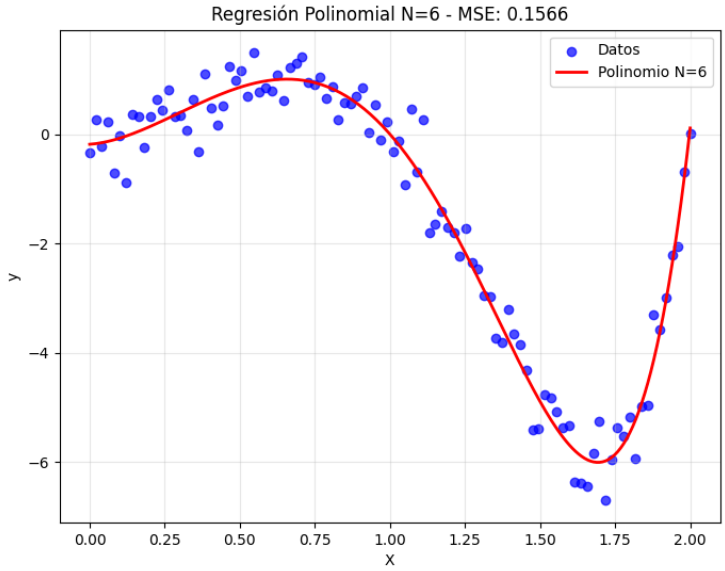

Figure 9: Regresión polinomial con $N = 6$.

8.4 Discusión

Los resultados muestran tres comportamientos distintos:

- $N = 2$ (Underfitting): MSE = 2.224. Modelo demasiado simple para capturar la complejidad cúbica de los datos.
- $N = 3$ (Óptimo): MSE = 0.969. Balance ideal entre ajuste y generalización.
- $N = 6$ (Overfitting): MSE = 0.157. Aunque tiene menor error, sigue muy de cerca los puntos individuales perdiendo capacidad de generalización.

El dataset sigue naturalmente una función cúbica, por lo que $N = 3$ representa la complejidad apropiada.

8.5 Conclusión

El análisis demuestra que $N = 3$ es el valor óptimo, proporcionando el mejor balance entre ajuste y generalización para este dataset cúbico. Los resultados confirman la importancia de seleccionar apropiadamente la complejidad del modelo para evitar underfitting ($N = 2$) y overfitting ($N = 6$).

9 Problema 9

9.1 Enunciado

Considere el "Lennard-Jones energy levels" dataset proporcionado con esta tarea. Dadas N partículas, $N \geq 2$, la energía potencial de Lennard Jones está dada por

$$E_N = 4 \sum_{i<j}^{N} \left[\left(\frac{1}{r_{ij}} \right)^{12} - \left(\frac{1}{r_{ij}} \right)^{6} \right], \tag{8}$$

Donde r_{ij} denota la distancia euclidiana entre las partículas X_i y X_j. Para diversas aplicaciones, resulta de gran interés encontrar la configuración espacial particular de N partículas que minimice la energía (8). Entonces:

(a) Elija un valor particular para N mayor que 2.

(b) Escriba la expresión para el vector residual $r(X)$.

(c) Usando los datos proporcionados, implemente el método de Gauss-Newton para resolver el problema de mínimos cuadrados no lineales.

(d) Grafique los valores de (8) a lo largo de las iteraciones y compare su estimación final con los de la Tabla 1 de [1].

(e) Grafique en 3D la configuración final de la partícula (es decir, los puntos X).

(f) ¿Cuál fue el valor máximo de N que pudo obtener?

9.2 Metodología

Para resolver este problema se implementó el método de Gauss-Newton siguiendo estos pasos:

- Implementar una clase GaussNewton.
- Crear la función de energía de Lennard-Jones.
- Implementar la función residual.
- Calcular la matriz Jacobiana del residual.
- Probar diferentes valores de N para analizar la convergencia.
- Generar gráficas de convergencia y configuración final de partículas.

9.3 Resultados

Los resultados obtenidos para $N = 25$ partículas se presentan a continuación:
Energías calculadas:

$$E_{\text{final}} = -8.144 \tag{1}$$

$$E_{\text{referencia}} = -102.373 \tag{2}$$

$$\text{Error absoluto} = 94.228 \tag{3}$$

Convergencia del algoritmo: El algoritmo ejecutó 200 iteraciones sin alcanzar el criterio de convergencia establecido (tolerancia = 1×10^{-6}).

Las gráficas de los resultados se presentan en las siguientes figuras:

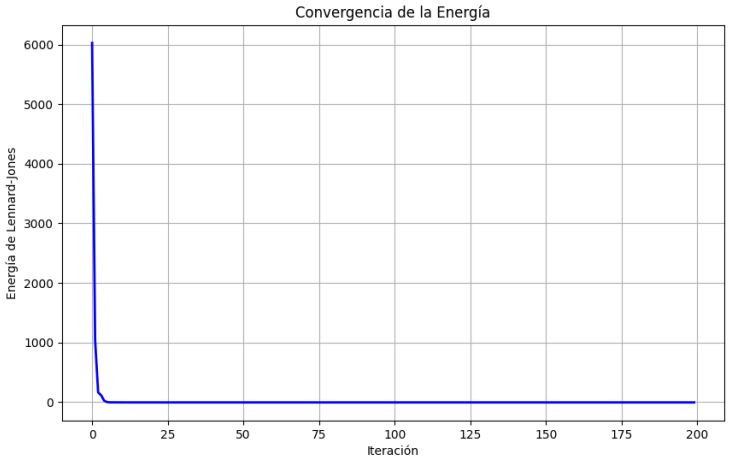

Figure 10: Convergencia de la energía de Lennard-Jones para $N = 25$ partículas utilizando el método de Gauss-Newton.

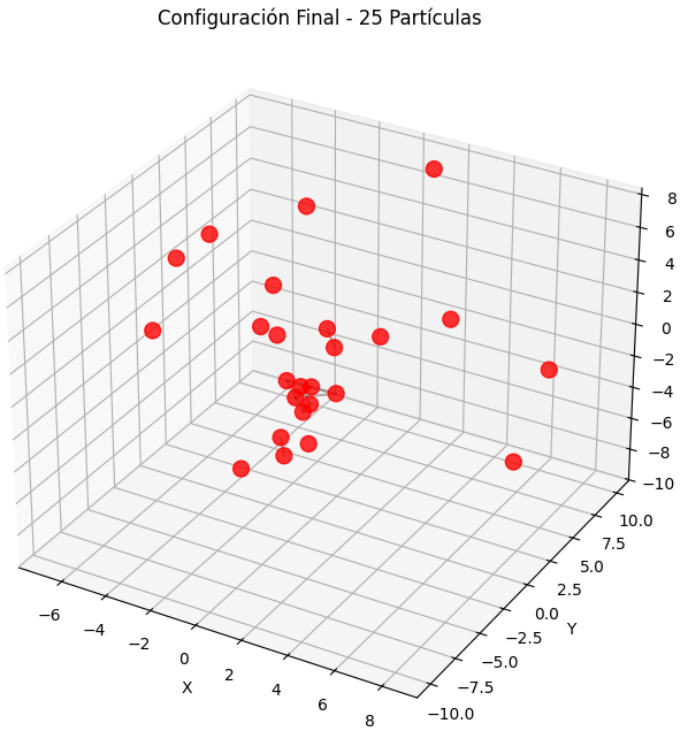

Figure 11: Configuración final en 3D de las 25 partículas después de la optimización con Gauss-Newton.

9.4 Discusión

A pesar de haber probado con múltiples valores de N, mi implementación del método de Gauss-Newton resultó ser poco óptima al no poder converger adecuadamente la función de energía de Lennard-Jones. Aunque las gráficas pueden ser engañosas a simple vista, mostrando que los valores de energía disminuyen y que las partículas en la gráfica 3D se agrupan en configuraciones aparentemente estables, la realidad es que el algoritmo no logra alcanzar los valores de referencia esperados.

9.5 Conclusión

La implementación del método de Gauss-Newton para minimizar la energía de Lennard-Jones presenta limitaciones significativas para este tipo de problema. Pese a que se probaron múltiples valores de N y el algoritmo muestra aparente progreso en las gráficas, la convergencia no se logra de manera efectiva.

References

[1] D. J. Wales y J. P. K. Doye, **Global Optimization by Basin-Hopping and the Lowest Energy Structures of Lennard-Jones Clusters Containing up to 110 Atoms**, Abstract published in AdVance ACS Abstracts, June 15, 1997.