Análisis y Solución de un Problema de Optimización Sin Restricciones

11 de noviembre de 2025

1. Nombre y Grupo

- **Nombre y Apellidos : Darián Santamarina Hernández**
- **Grupo : C-311**

2. Modelo a Analizar

El problema consiste en la optimización sin restricciones de la función objetivo $f(\mathbf{x}) : \mathbb{R}^2 \to \mathbb{R}$, dada por:

$$f(x, y) = \ln\left(e^{(x^2+y^2)} + 10e^x\right)$$

3. Análisis Teórico de los Modelos

3.1. Existencia de Solución

- **Dominio:** El argumento del logaritmo $g(x, y) = e^{(x^2+y^2)} + 10e^x$ es la suma de exponenciales y es siempre positivo ($g(x, y) > 0$). Por lo tanto, el dominio es $\text{Dom}(f) = \mathbb{R}^2$.
- **Existencia de Óptimo:** La función $f(x, y)$ es continua y coercive (tiende a infinito cuando $|\mathbf{x}| \to \infty$).

$$\lim_{|\mathbf{x}|\to\infty} f(x, y) = \infty$$

  Por el Teorema de Weierstrass para funciones coercivas, existe al menos un **mínimo global**.

3.2. Convexidad

La función $f(x, y)$ es de la forma Log-Sum-Exp: $f(\mathbf{x}) = \ln\left(e^{g_1(\mathbf{x})} + e^{g_2(\mathbf{x})}\right)$, donde:

- $g_1(x, y) = x^2 + y^2$
- $g_2(x, y) = x + \ln(10)$

La función Log-Sum-Exp es convexa si sus argumentos $g_i(\mathbf{x})$ son convexos.

- $g_1(x, y) = x^2 + y^2$ es estrictamente convexa (su matriz Hessiana es $2\mathbf{I}$, definida positiva).
- **Cálculo de la Hessiana:**

$$\nabla^2 g_1(x, y) = \begin{pmatrix} \frac{\partial^2 g_1}{\partial x^2} & \frac{\partial^2 g_1}{\partial x \partial y} \\ \frac{\partial^2 g_1}{\partial y \partial x} & \frac{\partial^2 g_1}{\partial y^2} \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$

- **G1 conclusion:** La matriz $\nabla^2 g_1(x, y)$ es una matriz diagonal con valores propios $\lambda_1 = 2$ y $\lambda_2 = 2$. Dado que ambos valores propios son estrictamente positivos ($2 > 0$), la matriz Hessiana es Definida Positiva ($\succ 0$) para todo $\mathbf{x} \in \mathbb{R}^2$.

- $g_2(x, y) = x + \ln(10)$ es convexa (lineal).

Dado que $f(x, y)$ es Log-Sum-Exp de funciones convexas, $f(x, y)$ es una función convexa. Más aún, debido al término $x^2 + y^2$, $f(x, y)$ es estrictamente convexa.

- **Conclusión:** La estricta convexidad garantiza la existencia de un óptimo global único.
- **Extremos Locales:** Al ser estrictamente convexa, no pueden existir otros mínimos locales; el único extremo local es el mínimo global.

3.3. Tipo de Variables

Variables continuas: x,y $\in \mathbb{R}$. No hay variables discretas ni restricciones.

3.4. Cálculo del Gradiente y Hessiana

Sea $u(x, y) = e^{(x^2+y^2)} + 10e^x$. El gradiente de $f(x, y) = \ln(u)$ es $\nabla f = \frac{1}{u}\nabla u$.

3.4.1. Gradiente (∇f)

$$\frac{\partial u}{\partial x} = e^{(x^2+y^2)}(2x) + 10e^x$$
$$\frac{\partial u}{\partial y} = e^{(x^2+y^2)}(2y)$$

El gradiente es:

$$\nabla f(x, y) = \frac{1}{e^{(x^2+y^2)} + 10e^x} \begin{pmatrix} 2xe^{(x^2+y^2)} + 10e^x \\ 2ye^{(x^2+y^2)} \end{pmatrix}$$

3.4.2. Hessiana ($\nabla^2 f$)

Sea la función:

$$f(x, y) = \ln\left(e^{x^2+y^2} + 10e^x\right)$$

La matriz Hessiana se obtiene mediante:

$$\nabla^2 f(x, y) = \frac{1}{S}\begin{pmatrix} S_{xx} & S_{xy} \\ S_{xy} & S_{yy} \end{pmatrix} - \frac{1}{S^2}\begin{pmatrix} S_x^2 & S_x S_y \\ S_x S_y & S_y^2 \end{pmatrix},$$

donde:

$$S = e^{x^2+y^2} + 10e^x,$$
$$S_x = 2xe^{x^2+y^2} + 10e^x,$$
$$S_y = 2ye^{x^2+y^2},$$
$$S_{xx} = (2 + 4x^2)e^{x^2+y^2} + 10e^x,$$
$$S_{xy} = 4xye^{x^2+y^2},$$
$$S_{yy} = (2 + 4y^2)e^{x^2+y^2}.$$

Por tanto, la matriz Hessiana explícita es:

$$\nabla^2 f(x, y) = \frac{1}{e^{x^2+y^2} + 10e^x}\begin{pmatrix} (2 + 4x^2)e^{x^2+y^2} + 10e^x & 4xye^{x^2+y^2} \\ 4xye^{x^2+y^2} & (2 + 4y^2)e^{x^2+y^2} \end{pmatrix} - \frac{1}{\left(e^{x^2+y^2} + 10e^x\right)^2}\begin{pmatrix} (2xe^{x^2+y^2} + 10e^x)^2 \\ (2xe^{x^2+y^2} + 10e^x)(2ye^{x^2+y^2} \end{pmatrix}$$

3.5. Condiciones necesarias y suficientes

Función:

$$f(x, y) = \ln\big(e^{x^2+y^2} + 10e^x\big).$$

Condición necesaria (Gradiente nulo). El gradiente de f es:

$$\nabla f(x,y) = \frac{1}{S}\begin{pmatrix} 2xe^{x^2+y^2} + 10e^x \\ 2ye^{x^2+y^2} \end{pmatrix}, \qquad S = e^{x^2+y^2} + 10e^x.$$

Igualando a cero:

$$\frac{2ye^{x^2+y^2}}{S} = 0 \Rightarrow y = 0,$$

y sustituyendo $y = 0$ en la primera componente:

$$\frac{2xe^{x^2} + 10e^x}{e^{x^2} + 10e^x} = 0 \quad \Longrightarrow \quad 2xe^{x^2} + 10e^x = 0.$$

Dividiendo por $e^x > 0$:

$$2xe^{x^2-x} + 10 = 0 \quad \Longleftrightarrow \quad 2xe^{x(x-1)} = -10.$$

Por tanto, los puntos críticos tienen la forma:

$$(x,y) = (x^*, 0),$$

donde x^* es una solución real de la ecuación $2xe^{x^2} + 10e^x = 0$. Numéricamente existe una única raíz real negativa ($x^* \approx -0{,}9012267233$).

Condición suficiente (segunda derivada / Hessiana). Sea

$$\phi_1(x,y) = x^2 + y^2, \qquad \phi_2(x,y) = x + \ln 10.$$

Entonces

$$f(x,y) = \ln\left(e^{\phi_1(x,y)} + e^{\phi_2(x,y)}\right).$$

Definimos:

$$p_1(x,y) = \frac{e^{\phi_1}}{e^{\phi_1} + e^{\phi_2}}, \qquad p_2(x,y) = \frac{e^{\phi_2}}{e^{\phi_1} + e^{\phi_2}}, \qquad p_1, p_2 > 0,\ p_1 + p_2 = 1.$$

Una identidad útil para la función *log-sum-exp* es:

$$\nabla^2 f(x,y) = \sum_{i=1}^{2} p_i\, \nabla^2\phi_i + \mathrm{Cov}_p(\nabla\phi),$$

donde $\mathrm{Cov}_p(\nabla\phi)$ es la matriz de covarianza de los gradientes $\nabla\phi_i$ bajo las probabilidades p_i. Dado que la covarianza es siempre semidefinida positiva, resulta que:

$$\nabla^2 f(x,y) \succeq \sum_{i=1}^{2} p_i\, \nabla^2\phi_i.$$

En nuestro caso:

$$\nabla^2\phi_1 = 2I_2, \qquad \nabla^2\phi_2 = 0,$$

por lo que:

$$\nabla^2 f(x,y) = 2p_1(x,y)I_2 + \mathrm{Cov}_p(\nabla\phi) \succeq 2p_1(x,y)I_2.$$

Como $p_1(x,y) > 0$ para todo $(x,y) \in \mathbb{R}^2$, la matriz Hessiana es *definida positiva en todo el dominio*. Por tanto, f es **estrictamente convexa**.

Conclusiones.

- La condición necesaria $\nabla f = 0$ se cumple únicamente para $(x,y) = (x^*, 0)$, donde x^* es la raíz real de $2xe^{x^2} + 10e^x = 0$.
- La condición suficiente (Hessiana definida positiva) se cumple en todo $\mathbb{R}^2$.
- El punto $(x^*, 0)$ es por tanto un **mínimo estricto local** y, debido a la convexidad global de f, es además el **mínimo global único**.
- No existen máximos locales ni puntos de silla, pues la estricta convexidad excluye tales casos.

Comprobación numérica. Evaluando la Hessiana en el punto crítico $(x^*, 0)$ con $x^* \approx -0{,}9012267233$, se obtienen autovalores positivos:

$$\lambda_1 \approx 2{,}5161, \qquad \lambda_2 \approx 0{,}71366,$$

confirmando que $\nabla^2 f(x^*, 0)$ es definida positiva.

3.6. Que óptimos puedes encontrar teoricamente?

Para encontrar los óptimos (máximos o mínimos) teóricos, debemos encontrar los puntos críticos de la función. Los puntos críticos son aquellos donde el gradiente es cero, es decir, donde las derivadas parciales de primer orden son iguales a cero El mínimo se encuentra en un punto crítico de la forma $(x^*, 0)$ donde es la solución real de

$$2xe^{x^2-x} + 10 = 0$$

3.7. Existen extremos locales?

Hay exactamente un extremo local, y es un mínimo estricto. No existen máximos locales ni puntos de silla.

Valor numérico del punto crítico. Resolviendo numéricamente

$$2xe^{x^2} + 10e^x = 0$$

se obtiene la raíz real

$$x^* \approx -0{,}9012267233.$$

Por tanto el único punto crítico es aproximadamente

$$(x^*, y^*) \approx \big(-0{,}9012267233,\ 0\big),$$

y ese punto es un mínimo estricto local y, por estricta convexidad, el mínimo global único de f.

Conclusión.

- Existe exactamente un extremo local: un mínimo estricto en $(x^*, 0)$.
- No existen máximos locales ni puntos de silla.
- El mínimo es único y global.

4. Descripción de los algoritmos utilizados

4.1. Algoritmo BFGS para Optimización No Lineal

El algoritmo BFGS (Broyden-Fletcher-Goldfarb-Shanno) es un método quasi-Newton ampliamente utilizado para optimización no lineal sin restricciones. Desarrollado independientemente por cuatro investigadores en 1970, se ha convertido en uno de los métodos más populares y robustos para problemas de optimización de mediana y gran escala.

Historia y Contexto

El método BFGS fue propuesto simultáneamente en 1970 por:

- **C. G. Broyden** - Matemático británico
- **R. Fletcher** - Científico computacional británico
- **D. Goldfarb** - Matemático estadounidense
- **D. F. Shanno** - Matemático canadiense

Este algoritmo surge como una mejora sobre los métodos de gradiente convencionales y los métodos de Newton, combinando la eficiencia computacional con la robustez numérica.

Fundamento Matemático

El algoritmo BFGS aproxima la matriz Hessiana (o su inversa) utilizando únicamente información de primer orden (gradientes), evitando el costoso cálculo de segundas derivadas. La actualización se realiza mediante la fórmula:

$$H_{k+1} = (I - \rho_k s_k y_k^T) H_k (I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T \tag{1}$$

donde:

- $s_k = x_{k+1} - x_k$ es el vector de desplazamiento
- $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$ es la diferencia de gradientes
- $\rho_k = \frac{1}{y_k^T s_k}$ es el factor de escala
- H_k es la aproximación actual de la inversa del Hessiano

Características Principales

Ventajas del Método BFGS

1. **Eficiencia Computacional**: Evita el cálculo explícito del Hessiano, reduciendo significativamente el costo computacional.
2. **Convergencia Superlineal**: Bajo condiciones adecuadas, exhibe convergencia superlineal, más rápida que los métodos de gradiente.
3. **Preservación de Definida Positividad**: La actualización BFGS mantiene la definida positividad de H_k si se satisface la condición de curvatura $y_k^T s_k > 0$.
4. **Robustez**: Funciona bien en una amplia variedad de problemas sin requerir ajustes finos de parámetros.
5. **Escalabilidad**: Es adecuado para problemas de mediana dimensión (decenas a cientos de variables).

Componentes del Algoritmo

El algoritmo implementado consta de tres componentes principales:

1. **Función Objetivo y Gradiente**: Define el problema de optimización a resolver.
2. **Búsqueda de Línea con Armijo**: Garantiza un decrecimiento suficiente en cada iteración mediante la condición:
$$f(x_k + \alpha p_k) \leq f(x_k) + c\alpha \nabla f(x_k)^T p_k \tag{2}$$
3. **Actualización BFGS**: Mejora iterativamente la aproximación del Hessiano inverso.

Implementación y Consideraciones Prácticas

En la implementación mostrada, se incluyen varias salvaguardas numéricas:

- **Condición de Curvatura**: Se verifica $y_k^T s_k > 10^{-10}$ antes de actualizar H_k
- **Criterio de Parada**: Basado en la norma del gradiente $\|\nabla f(x_k)\| < \text{tol}$
- **Límite de Iteraciones**: Evita ciclos infinitos con un máximo de iteraciones
- **Estabilización Numérica**: Uso de log-sum-exp para evitar overflow numérico

Aplicaciones y Casos de Uso

El algoritmo BFGS es particularmente útil en:

- Problemas de aprendizaje automático (regresión logística, SVM)
- Calibración de modelos financieros
- Optimización de parámetros en ingeniería
- Ajuste de curvas en análisis de datos
- Problemas de diseño óptimo

Limitaciones y Alternativas

Aunque BFGS es muy efectivo, presenta algunas limitaciones:

- Requiere almacenar una matriz $n \times n$, lo que puede ser costoso para problemas de muy alta dimensión.
- Para problemas a gran escala ($n > 1000$), se prefieren métodos de memoria limitada como L-BFGS.
- La convergencia a óptimos globales no está garantizada en funciones no convexas.

Conclusión

El algoritmo BFGS representa un equilibrio óptimo entre eficiencia computacional, velocidad de convergencia y facilidad de implementación. Su robustez y versatilidad lo convierten en la elección predilecta para una amplia gama de problemas de optimización no lineal en la práctica ingenieril y científica.

La implementación presentada demuestra cómo combinar la actualización BFGS con búsqueda de línea por condiciones de Armijo para crear un método de optimización completo y numéricamente estable.

4.2. El Algoritmo: Newton Amortiguado

El objetivo del algoritmo es encontrar el vector $\mathbf{x}^*$ que minimiza una función $f(\mathbf{x})$, donde $\mathbf{x}$ es un vector de variables.

Este método es un algoritmo de segundo orden porque utiliza no solo la pendiente (el gradiente, $\nabla f(\mathbf{x})$) sino también la curvatura (el Hessiano, $\mathbf{H}(\mathbf{x})$) para tomar decisiones sobre la dirección y el tamaño del paso.

La Dirección de Newton Original

El Método de Newton puro calcula una dirección de paso $\mathbf{p}_k$ en cada iteración k resolviendo el siguiente sistema de ecuaciones lineales:

$$\mathbf{H}_k \mathbf{p}_k = -\nabla f_k$$

Donde $\mathbf{H}_k$ es el Hessiano (matriz de segundas derivadas) evaluado en el punto actual $\mathbf{x}_k$, y ∇f_k es el gradiente.

- **Ventaja:** Si la función es cuadrática, converge en una sola iteración. Cerca del óptimo, converge cuadráticamente (muy rápido).
- **Problema:** Requiere que el Hessiano $\mathbf{H}_k$ sea Definido Positivo (DP) en cada iteración. Si no lo es (o es singular), la dirección $\mathbf{p}_k$ puede no ser una dirección de descenso, o el sistema no tiene solución única. Esto hace al método puro inestable si el punto inicial está lejos del óptimo.

Estabilización: Damping (Amortiguamiento) El algoritmo de Newton Amortiguado aborda la inestabilidad del método puro añadiendo un término de regularización (o damping) al Hessiano. Esto se hace de forma análoga al método de Levenberg-Marquardt.

El Paso de Newton Amortiguado

En lugar de resolver el sistema de Newton puro, se resuelve el sistema amortiguado, introduciendo un parámetro $\lambda > 0$:

$$(\mathbf{H}_k + \lambda \mathbf{I})\mathbf{p}_k = -\nabla f_k$$

Donde $\mathbf{I}$ es la matriz identidad.

Efecto de λ:

- Si λ es pequeño (cercano a cero), la matriz $\mathbf{H}_k + \lambda \mathbf{I}$ se parece a $\mathbf{H}_k$, y el paso $\mathbf{p}_k$ se acerca al paso de Newton (rápido).
- Si λ es grande, la matriz se parece a $\lambda \mathbf{I}$, y el sistema se convierte en $(\lambda \mathbf{I})\mathbf{p}_k \approx -\nabla f_k$, por lo que $\mathbf{p}_k \approx -\frac{1}{\lambda}\nabla f_k$. Esto se aproxima al paso de Máximo Descenso (más lento, pero garantiza descenso).

El algoritmo ajusta λ dinámicamente:

- Si un paso es exitoso (la función disminuye), se reduce λ (hacia Newton).
- Si el paso falla o la dirección no es de descenso, se aumenta λ (hacia Máximo Descenso, para estabilizar).

Búsqueda de Línea (Armijo Backtracking)

Una vez que se tiene una dirección de búsqueda $\mathbf{p}_k$, el algoritmo necesita decidir cuánto avanzar en esa dirección, es decir, encontrar el tamaño de paso α.

La Búsqueda de Línea de Armijo

La Búsqueda de Línea con backtracking (retroceso) de Armijo es un procedimiento que comienza con un α grande (por ejemplo, $\alpha = 1$) y lo va reduciendo hasta que se cumpla una condición que asegure un "descenso suficiente" en la función.

Condición de Armijo (Suficiente Descenso):

$$f(\mathbf{x}_k + \alpha \mathbf{p}_k) \leq f(\mathbf{x}_k) + c \cdot \alpha \cdot \nabla f(\mathbf{x}_k)^T \mathbf{p}_k$$

El término $c \cdot \alpha \cdot \nabla f(\mathbf{x}_k)^T \mathbf{p}_k$ es un descenso "predicho" linealmente. La condición asegura que el descenso real ($f(\mathbf{x}_k) - f(\mathbf{x}_k + \alpha \mathbf{p}_k)$) es al menos una pequeña fracción (c) del descenso predicho.

Resumen del Flujo

En cada iteración, el algoritmo realiza lo siguiente:

1. **Evaluar:** Calcular el valor de la función f_k, el gradiente ∇f_k y el Hessiano $\mathbf{H}_k$.
2. **Verificar Parada:** Si $||\nabla f_k||$ es menor que la tolerancia, detener (se está cerca de un mínimo).
3. **Calcular Dirección:** Resolver el sistema amortiguado $(\mathbf{H}_k + \lambda \mathbf{I})\mathbf{p}_k = -\nabla f_k$.
4. **Ajustar Damping:** Verificar si $\mathbf{p}_k$ es una dirección de descenso. Si no, aumentar λ y volver al paso 3 (intentar de nuevo).
5. **Buscar Paso (α):** Usar el método de Armijo para encontrar un tamaño de paso α adecuado a lo largo de $\mathbf{p}_k$.
6. **Actualizar:** $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha \mathbf{p}_k$.
7. **Ajustar Damping (Convergencia):** Si el paso fue exitoso, reducir λ para favorecer el comportamiento de Newton.

5. Comparación entre los métodos BFGS y Newton amortiguado

Con el objetivo de evaluar el desempeño de ambos métodos de optimización, se aplicaron sobre un mismo conjunto de 100 puntos iniciales. A continuación se presenta un resumen de los resultados obtenidos.

Cuadro 1: Resumen comparativo de desempeño promedio entre BFGS y Newton amortiguado

Métrica	**BFGS**	**Newton amortiguado**
Iteraciones promedio	8.79	6.02
Norma del gradiente promedio	$2{,}82 \times 10^{-7}$	$1{,}33 \times 10^{-9}$
Valor promedio de f_{opt}	1,8427048713555	1,8427048713555

En la Figura 1 se observa que el método de Newton amortiguado requiere, en promedio, alrededor de 2,8 iteraciones menos que el método BFGS para alcanzar la convergencia. Además, logra una norma del gradiente significativamente menor, lo que indica una mayor precisión numérica en el punto óptimo encontrado.

- Ambos métodos convergen al mismo punto óptimo global, con diferencias en $f(x^*)$ del orden de 10^{-14}.
- El método de Newton presenta una convergencia más rápida y estable frente a distintos puntos iniciales.
- BFGS, aunque más lento, mantiene una robustez general y no requiere el cálculo directo de la Hessiana.

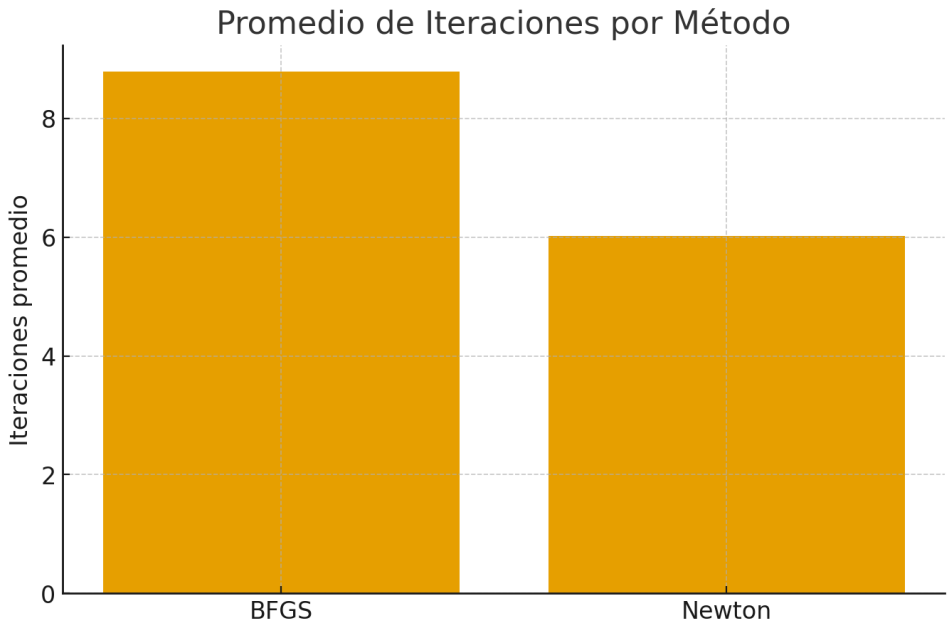

Figura 1: Comparación del número de iteraciones promedio entre BFGS y Newton amortiguado.

En conclusión, el método de **Newton amortiguado (trust region)** demostró un mejor desempeño global en términos de velocidad de convergencia y precisión, mientras que **BFGS** ofrece una alternativa eficiente cuando no se dispone del Hessiano o su cálculo resulta costoso.

6. Graficación del Modelo y de Instancias de los Algoritmos

6.1. Modelo

Lo haremos con un mapa de contorno (contour plot) y una superficie 3D, para observar dónde están los mínimos.

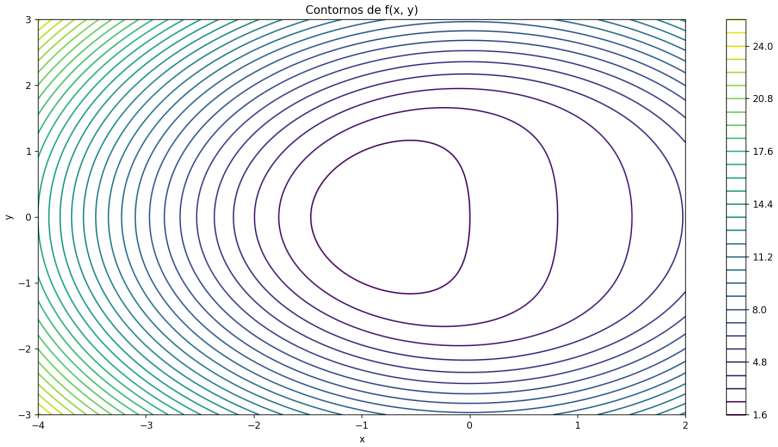

Figura 2: Se mostrará la forma de f(x,y): una superficie suave con un mínimo claro en torno a $x \approx -0{,}9$, $y = 0$

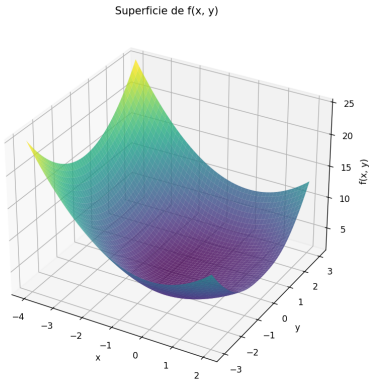

Figura 3: Superficie de f(x,y) 3D

6.2. Instancias de los Algoritmos

Puntos Iniciales (x_0):

Importancia de la Dispersión:

La muestra que los puntos iniciales están altamente dispersos (cubren un amplio rango, como -100 a 100 en x_1 y x_2).Los puntos se toman del archivo $initial_points.json$ generado aleatoriamente

Propósito:

Utilizar puntos iniciales dispersos es una excelente práctica para la optimización.

Significado:

Esto asegura que los algoritmos no se queden atrapados en un mínimo local o en una región de la función que no sea representativa. Al probar con diferentes x_0, se verifica la robustez y la capacidad de convergencia global del método, incluso cuando el punto de partida está muy lejos de la solución óptima.

Resultado Observable:

A pesar de la amplia dispersión de x_0, casi todas las instancias de ambos algoritmos convergen al mismo punto (alrededor de $x_1 \approx -0{,}9012, x_2 \approx 0$), lo que indica que el mínimo global es el único mínimo relevante en la región explorada.

Algoritmo de Newton Amortiguado (Damped Newton Trust)

Al examinar los valores de x_opt y iterations en $results_newton.json$:

Precisión:

Los valores de x_{opt} son extremadamente precisos. El componente x_2 está típicamente en el rango de 10^{-11} a 10^{-15} (muy cercano a cero), y el gradiente ($grad_norm$) es similarmente pequeño.

Eficiencia:

El número de iteraciones (iterations) es notablemente bajo, a menudo entre 5 y 8 iteraciones para alcanzar la convergencia, incluso desde puntos iniciales lejanos.

Conclusión:

La convergencia de este método es de segundo orden (cuadrática), utilizando la matriz Hessiana para encontrar la dirección óptima de manera muy eficiente. Esto lo convierte en el método más rápido y preciso en la región del óptimo.

Algoritmo BFGS

Al examinar los valores de x_opt y iters en $bfgs_results.json$:

Precisión:

La precisión de BFGS es excelente, pero generalmente ligeramente inferior a la de Newton. Los valores de x_2 suelen ser del orden de 10^{-7} a 10^{-9}.

Eficiencia:

El número de iteraciones (iters) es mayor que en Newton, típicamente entre 8 y 12 iteraciones. Esto es normal, ya que BFGS es un método de cuasi-Newton que aproxima la matriz Hessiana.

Robustez (success):

Es fundamental revisar cuántas instancias tienen "success": true. Si el número es alto, significa que BFGS también es un algoritmo muy robusto para esta función, capaz de manejar la alta curvatura de la función.

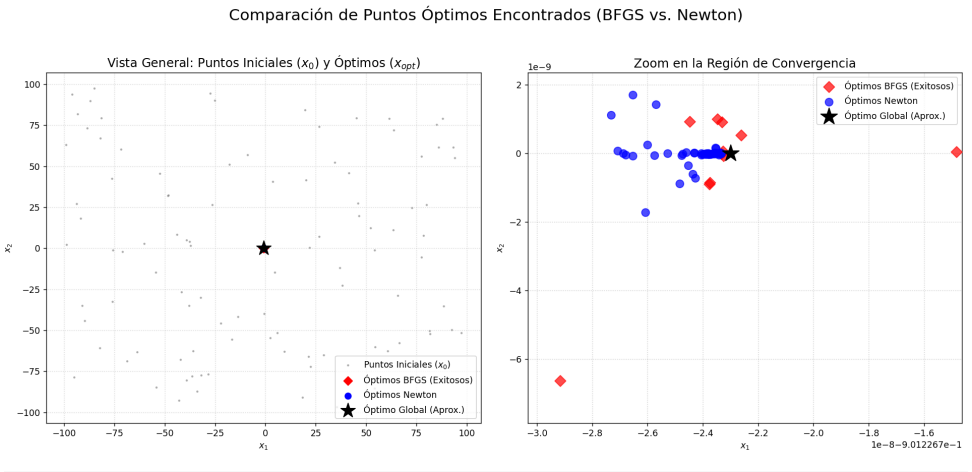

Figura 4: Comparacion de Resultados de instancias de BFGS VS Newton

Conclusión Final

Ambos algoritmos demostraron una excelente robustez al converger de manera consistente al mismo mínimo global, a pesar de que los puntos iniciales estaban distribuidos en un amplio dominio.

El algoritmo Damped Newton Trust fue el más eficiente en términos de número de iteraciones y el más preciso en la localización del mínimo. El algoritmo BFGS fue una alternativa robusta, aunque requirió un poco más de iteraciones y logró una precisión ligeramente menor. En un contexto real, la elección dependería del compromiso entre el costo computacional por iteración (menor en BFGS) y la velocidad de convergencia global (mayor en Newton).