MEMAD-T03

ALEJANDRO ZARATE MACIAS

8 de Septiembre 2025

Introducción

Para la tarea de esta semana se busca trabajar distintos problemas enfocados en el uso de gradientes y su aplicación en métodos de optimización. El objetivo es poner en práctica lo aprendido en los videos proporcionados como material de estudio, así como en los libros sugeridos para el curso. Todo esto con el fin de aplicar:

- Propiedades de convexidad.
- Teorema de Taylor.
- Gradientes: dirección, tamaño de paso, suficiente descenso, etc.
- Condiciones de Armijo y Wolfe
- Metodos: Newton, Quasi-Newton y Steepest Descent (descenso mas pronunciado)

1 Problema 1

1.1 Enunciado

Supóngase que $f(\mathbf{x}) = \mathbf{x}^T Q\mathbf{x}$, donde $Q \in \mathbb{R}^{n\times n}$, $Q \geq 0$, $Q = Q^T$. Demuestre que $f(\mathbf{x})$ es convexa $\forall\, \mathbf{x} \in \mathbb{R}^n$.

1.2 Metodología

Para demostrar la convexidad de $f(\mathbf{x})$, utilizaremos el criterio de la matriz hessiana. Una función es convexa si y solo si su matriz hessiana es semidefinida positiva. Por ende, se deben de realizar los siguientes pasos:

1. Calcular el gradiente de $f(\mathbf{x})$.
2. Calcular la matriz hessiana $\nabla^2 f(\mathbf{x})$.
3. Demostrar que la hessiana es semidefinida positiva utilizando las propiedades dadas de Q.

1.3 Resultados

Comenzamos calculando el gradiente de $f(\mathbf{x}) = \mathbf{x}^T Q\mathbf{x}$. Para una función cuadrática de la forma $\mathbf{x}^T Q\mathbf{x}$, el gradiente está dado por:

$$\nabla f(\mathbf{x}) = (Q + Q^T)\mathbf{x} \tag{1}$$

Dado que $Q = Q^T$ (la matriz es simétrica), tenemos:

$$\nabla f(\mathbf{x}) = (Q + Q)\mathbf{x} \tag{2}$$
$$= 2Q\mathbf{x} \tag{3}$$

Ahora calculamos la matriz hessiana. La hessiana es la matriz de segundas derivadas parciales:

$$\nabla^2 f(\mathbf{x}) = \frac{\partial}{\partial \mathbf{x}}(\nabla f(\mathbf{x})) \tag{4}$$
$$= \frac{\partial}{\partial \mathbf{x}}(2Q\mathbf{x}) \tag{5}$$
$$= 2Q \tag{6}$$

Para que $f(\mathbf{x})$ sea convexa, necesitamos que $\nabla^2 f(\mathbf{x}) \geq 0$ (semidefinida positiva). Dado que $\nabla^2 f(\mathbf{x}) = 2Q$ y sabemos por hipótesis que $Q \geq 0$ (semidefinida positiva), entonces se confirma que:

$$f(\mathbf{x}) = \mathbf{x}^T Q\mathbf{x} \quad \text{Es convexa} \quad \forall\, \mathbf{x} \in \mathbb{R}^n$$

1.4 Discusión

El resultado demuestra que la convexidad de $f(\mathbf{x}) = \mathbf{x}^T Q\mathbf{x}$ está directamente relacionada con las propiedades de la matriz Q. La clave está en que:

- La matriz hessiana resulta ser constante: $\nabla^2 f(\mathbf{x}) = 2Q$
- Como $Q \geq 0$ por hipótesis, multiplicar por el escalar positivo 2 preserva la propiedad semidefinida positiva
- Al ser la hessiana semidefinida positiva en todo punto, la función es convexa globalmente

Este resultado es fundamental en optimización, ya que las funciones cuadráticas con matrices semidefinidas positivas aparecen frecuentemente en problemas de optimización convexa.

1.5 Conclusión

Se ha demostrado que $f(\mathbf{x}) = \mathbf{x}^T Q\mathbf{x}$ es convexa para toda $\mathbf{x} \in \mathbb{R}^n$ cuando $Q \geq 0$ y $Q = Q^T$. La demostración se basa en que la matriz hessiana $\nabla^2 f(\mathbf{x}) = 2Q$ es semidefinida positiva, lo cual es una condición suficiente para la convexidad.

2 Problema 2

2.1 Enunciado

Sea $A \in \mathbb{R}^{n\times n}$, $A = A^T$. Considere

$$B = A + \alpha\mathbb{I},$$

donde $\mathbb{I} \in \mathbb{R}^{n\times n}$ denota la matriz identidad y $\alpha \in \mathbb{R}^+$. Demuestre que $B > 0$ para valores suficientemente grandes de α.

2.2 Metodología

Para demostrar que B es definida positiva para valores suficientemente grandes de α, se pueden utilizar las propiedades de los valores propios:

1. Analizar los valores propios de A al ser simétrica.
2. Determinar cómo afecta la adición de $\alpha\mathbb{I}$ a los valores propios.
3. Encontrar el valor mínimo de α que garantice que todos los valores propios de B sean positivos.

2.3 Resultados

Sea A una matriz simétrica, por el teorema espectral, se puede escribir como:

$$A = Q\Lambda Q^T \tag{1}$$

donde Q es una matriz ortogonal y $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$. Además, podemos reescribir a $\mathbb{I}$ como QQ^T (por las propiedades de la matriz identidad). Cuando formamos $B = A + \alpha\mathbb{I}$, obtenemos:

$$\begin{aligned} B &= A + \alpha\mathbb{I} && (2)\\ &= Q\Lambda Q^T + \alpha QQ^T && (3)\\ &= Q(\Lambda + \alpha\mathbb{I})Q^T && (4)\\ &= Q\text{diag}(\lambda_1 + \alpha, \lambda_2 + \alpha, \ldots, \lambda_n + \alpha)Q^T && (5) \end{aligned}$$

Por lo tanto, los valores propios de B son:

$$\mu_i = \lambda_i + \alpha, \quad i = 1, 2, \ldots, n \tag{6}$$

Para que B sea definida positiva ($B > 0$), todos sus valores propios deben ser estrictamente positivos:

$$\lambda_i + \alpha > 0 \quad \forall\, i = 1, 2, \ldots, n \tag{7}$$

Sea $\lambda_{\min} = \min\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ el menor valor propio de A. Entonces:

$$\alpha > -\lambda_{\min} \tag{8}$$

Lo que garantiza que $B > 0$.

2.4 Discusión

El resultado muestra que siempre es posible hacer que una matriz simétrica sea definida positiva agregando un múltiplo suficientemente grande de la matriz identidad. Esto es posible porque la adición de $\alpha\mathbb{I}$ desplaza todos los valores propios por la misma cantidad α, preservando los vectores propios.

2.5 Conclusión

Se ha demostrado que para cualquier matriz simétrica A, la matriz $B = A + \alpha\mathbb{I}$ es definida positiva cuando $\alpha > -\lambda_{\min}$, donde $\lambda_{\min}$ es el menor valor propio de A. Esta técnica es fundamental en métodos de optimización para regularizar matrices hessianas.

3 Problema 3

3.1 Enunciado

Escriba un script en Python para generar una matriz simétrica aleatoria $A \in (-0.5, 0.5)^{10\times10}$. Use la idea del Problema 2 para encontrar un valor adecuado de α y construir B tal que $B > 0$. Explique por qué la "estructura interna" de ambas matrices A y B permanece igual al inspeccionar su espectro.

3.2 Metodología

Para resolver este problema, se puede realizar un pequeño script con lo siguiente:

1. Generar una matriz aleatoria $A \in \mathbb{R}^{10\times10}$ con valores entre $(-0.5, 0.5)$ usando numpy.
2. Hacer que sea simétrica usando la operación $A = \frac{A+A^T}{2}$.
3. Calcular los valores propios de A utilizando lo que nos ofrece la libreria.
4. Determinar $\alpha = |\lambda_{\min}| + \epsilon$ donde $\epsilon = 10^{-3}$ para garantizar que $B > 0$.
5. Construir $B = A + \alpha\mathbb{I}$ y verifica su definida positividad.

3.3 Resultados

Los resultados se pueden encontrar en el archivo "`t03_alejandro_zarate_macias.ipynb`".

3.4 Discusión

La "estructura interna" de las matrices A y B permanece igual al inspeccionar su espectro porque:

1. Vectores propios preservados: Al agregar $\alpha\mathbb{I}$ a una matriz simétrica A, los vectores propios de B son idénticos a los de A. Esto se debe a que:

$$A\mathbf{v}_i = \lambda_i \mathbf{v}_i \implies (A + \alpha\mathbb{I})\mathbf{v}_i = (\lambda_i + \alpha)\mathbf{v}_i \tag{1}$$

2. Desplazamiento espectral uniforme: Los valores propios de B son simplemente los de A desplazados por α:

$$\text{eig}(B) = \text{eig}(A) + \alpha \tag{2}$$

3.5 Conclusión

Se implementó exitosamente un script de pocas líneas que transforma una matriz simétrica en una matriz definida positiva mediante regularización espectral. La técnica preserva la estructura de la matriz original, modificando únicamente la magnitud de los valores propios. Este resultado confirma la teoría del Problema 2.

4 Problema 4

4.1 Enunciado

Explique por qué la idea del Problema 2 falla si $A \neq A^T$.

4.2 Metodología

Para explicar por qué la técnica falla con matrices no simétricas, se debe analizar:

1. Las diferencias en las propiedades espectrales entre matrices simétricas y no simétricas.
2. Cómo la condición de definida positiva se relaciona con la simetría.
3. Proporcionar un ejemplo específico que lo pruebe.

4.3 Resultados

La técnica del Problema 2 falla cuando $A \neq A^T$ por una razón simple: el concepto de matriz "definida positiva" solo se aplica a matrices simétricas.

Cuando A no es simétrica, $B = A + \alpha\mathbb{I}$ tampoco será simétrica. Aunque se pueda hacer que $\mathbf{x}^T B\mathbf{x} > 0$, esto no da las propiedades que se necesitan.

Ejemplo:

$$A = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} \alpha & 1 \\ -1 & \alpha \end{pmatrix} \tag{1}$$

Para cualquier vector $\mathbf{x} = (x_1, x_2)^T$:

$$\mathbf{x}^T B\mathbf{x} = \alpha(x_1^2 + x_2^2) \tag{2}$$

Aunque esto es positivo para $\alpha > 0$, la matriz B sigue sin ser simétrica. El problema es que sin simetría:

- No podemos garantizar valores propios reales
- La descomposición espectral no funciona igual
- Las funciones cuadráticas asociadas pueden no ser convexas

4.4 Discusión

En resumen, la técnica falla porque necesitamos simetría para tener control sobre los valores propios. Sin simetría, agregar $\alpha\mathbb{I}$ no nos garantiza que la matriz resultante tenga las propiedades que buscamos en optimización.

4.5 Conclusión

La idea del Problema 2 no funciona para matrices no simétricas porque el concepto de "definida positiva" requiere simetría. Sin esta propiedad, perdemos las garantías necesarias para optimización convexa.

Importante

Para los problemas 5 – 8 considere las siguientes funciones $f : \mathbb{R}^n \to \mathbb{R}$:

- Función de esfera trasladada:

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

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

- Función de Rosenbrock:

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

- Función de Perm n, β:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \left(\sum_{j=1}^{n} (j^{\,i} + \beta) \left(\left(\frac{x_j}{j} \right)^{i} - 1 \right) \right)^2, \quad \text{para un determinado (fijo)} \quad \beta \in \mathbb{R}. \tag{3}$$

  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

Calcule analíticamente $\nabla f(\mathbf{x})$ para las funciones (1)–(3).

5.2 Metodología

Para calcular los gradientes analíticamente se procederá función por función:

1. Función de esfera trasladada: aplicar la regla de la cadena directamente.
2. Función de Rosenbrock: usar la regla del producto y la cadena para cada término.
3. Función de Perm: descomponer las sumas anidadas y aplicar derivación paso a paso.

5.3 Resultados

1. Función de esfera trasladada:

$$f(\mathbf{x}) = \sum_{i=1}^{n} (x_i - c_i)^2 \tag{1}$$

Para calcular $\frac{\partial f}{\partial x_j}$:

$$\frac{\partial f}{\partial x_j} = \frac{\partial}{\partial x_j} \sum_{i=1}^{n} (x_i - c_i)^2 \tag{2}$$

$$= \sum_{i=1}^{n} \frac{\partial}{\partial x_j} (x_i - c_i)^2 \tag{3}$$

$$= 2(x_j - c_j) \tag{4}$$

Por lo tanto:

$$\nabla f(\mathbf{x}) = 2(\mathbf{x} - \mathbf{c}) \tag{5}$$

2. Función de Rosenbrock:

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \right] \tag{6}$$

Para $j = 1$:

$$\frac{\partial f}{\partial x_1} = 100 \cdot 2(x_2 - x_1^2) \cdot (-2x_1) + 2(x_1 - 1) \tag{7}$$

$$= -400x_1(x_2 - x_1^2) + 2(x_1 - 1) \tag{8}$$

Para $1 < j < n-1$:

$$\frac{\partial f}{\partial x_j} = 100 \cdot 2(x_j - x_{j-1}^2) + 100 \cdot 2(x_{j+1} - x_j^2) \cdot (-2x_j) + 2(x_j - 1) \tag{9}$$

$$= 200(x_j - x_{j-1}^2) - 400x_j(x_{j+1} - x_j^2) + 2(x_j - 1) \tag{10}$$

Para $j = n$:

$$\frac{\partial f}{\partial x_n} = 100 \cdot 2(x_n - x_{n-1}^2) \tag{11}$$

$$= 200(x_n - x_{n-1}^2) \tag{12}$$

3. Función de Perm n, β:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \left(\sum_{j=1}^{n} (j^{\,i} + \beta) \left(\left(\frac{x_j}{j} \right)^i - 1 \right) \right)^2 \tag{13}$$

Esta función es más compleja debido a las sumas anidadas. Para simplificar el análisis, definimos:

$$S_i = \sum_{j=1}^{n} (j^{\,i} + \beta) \left(\left(\frac{x_j}{j} \right)^i - 1 \right) \tag{14}$$

Entonces la función se puede escribir como:

$$f(\mathbf{x}) = \sum_{i=1}^{n} S_i^2 \tag{15}$$

Para calcular $\frac{\partial f}{\partial x_k}$, aplicamos la regla de la cadena:

$$\frac{\partial f}{\partial x_k} = \sum_{i=1}^{n} \frac{\partial}{\partial x_k}(S_i^2) = \sum_{i=1}^{n} 2S_i \frac{\partial S_i}{\partial x_k} \tag{16}$$

Ahora necesitamos calcular $\frac{\partial S_i}{\partial x_k}$. En la suma S_i, solo el término con $j = k$ depende de x_k:

$$\frac{\partial S_i}{\partial x_k} = \frac{\partial}{\partial x_k} \left[(k^i + \beta) \left(\left(\frac{x_k}{k} \right)^i - 1 \right) \right] \tag{17}$$

$$= (k^i + \beta) \frac{\partial}{\partial x_k} \left(\frac{x_k}{k} \right)^i \tag{18}$$

$$= (k^i + \beta) \cdot i \cdot \left(\frac{x_k}{k} \right)^{i-1} \cdot \frac{1}{k} \tag{19}$$

$$= (k^i + \beta) \cdot \frac{i}{k} \cdot \left(\frac{x_k}{k} \right)^{i-1} \tag{20}$$

Sustituyendo de vuelta:

$$\frac{\partial f}{\partial x_k} = 2\sum_{i=1}^{n} S_i \cdot (k^i + \beta) \cdot \frac{i}{k} \cdot \left(\frac{x_k}{k}\right)^{i-1} \tag{21}$$

Forma completa del gradiente de la función Perm:
El k-ésimo componente del gradiente es:

$$\frac{\partial f}{\partial x_k} = 2\sum_{i=1}^{n} \left[\sum_{j=1}^{n} (j^{\,i} + \beta)\left(\left(\frac{x_j}{j}\right)^{i} - 1\right)\right] \cdot (k^i + \beta) \cdot \frac{i}{k} \cdot \left(\frac{x_k}{k}\right)^{i-1} \tag{22}$$

donde $k = 1, 2, \ldots, n$.

5.4 Discusión

Los gradientes calculados muestran diferentes niveles de complejidad. La función esfera trasladada tiene el gradiente más simple (lineal), mientras que Rosenbrock y Perm tienen gradientes no lineales más complejos. Esto afectará directamente la velocidad de convergencia de los algoritmos de optimización.

5.5 Conclusión

Se han calculado analíticamente los gradientes de las tres funciones. La función esfera trasladada tiene gradiente lineal, Rosenbrock tiene términos cuadráticos y cúbicos, y la función Perm tiene la expresión más compleja con potencias variables dependiendo de los parámetros i y k.

6 Problema 6

6.1 Enunciado

Implemente en Python el algoritmo de descenso más pronunciado (SD) para las funciones (1) − (3). Use un paso de longitud fija y el gradiente analítico. Muestre gráficas del número de iteraciones contra el valor de la función.

6.2 Metodología

Se implementará el algoritmo de descenso más pronunciado en Python con las siguientes características:

1. **Clase SteepestDescent**: Algoritmo SD con paso fijo que incluye parámetros configurables (tolerancia, máximo de iteraciones, tamaño de paso) y registro del historial de convergencia.

2. **Funciones objetivo**: Implementación de las tres funciones según el Problema 5 con $n = 5$, $\mathbf{c} = (1, 1, 1, 1, 1)$ y $\beta = 1$.

3. **Gradientes analíticos**: Uso de las derivadas calculadas en el Problema 5 para obtener direcciones de descenso exactas.

4. **Configuración experimental**: Punto inicial $\mathbf{x}_0 = (0.5, 0.5, 0.5, 0.5, 0.5)$ y paso fijo optimizado empíricamente para cada función.

6.3 Resultados

Las implementaciones del algoritmo de descenso más pronunciado se ejecutaron para las tres funciones objetivo. Los resultados se presentan en las siguientes gráficas que muestran la evolución del valor de la función respecto al número de iteraciones:

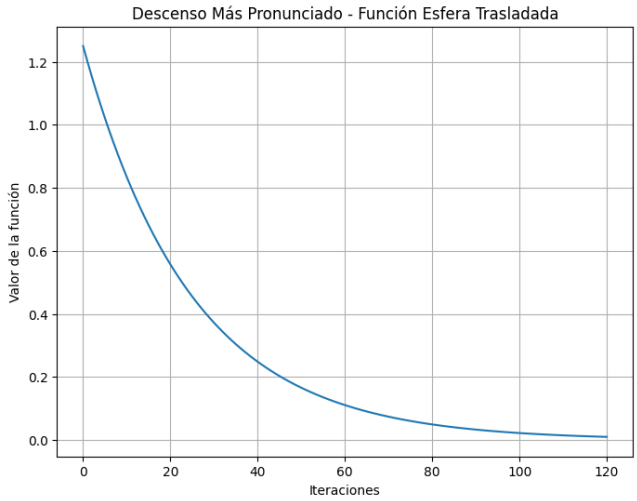

Figure 1: Convergencia del algoritmo SD para la función de esfera trasladada. La función converge rápidamente debido a su naturaleza convexa cuadrática.

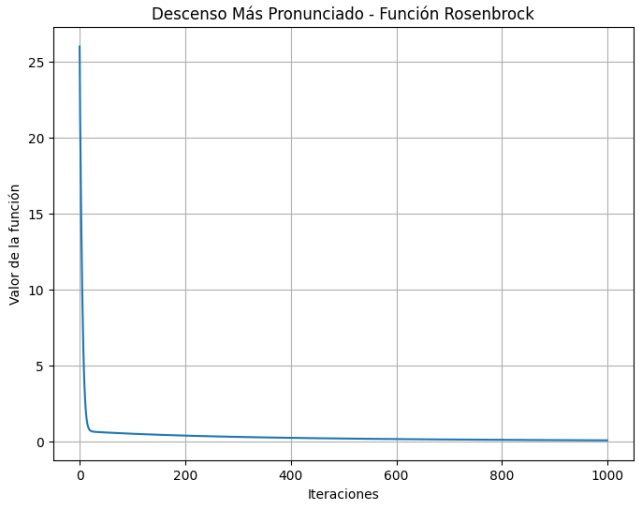

Figure 2: Convergencia del algoritmo SD para la función de Rosenbrock. Se observa una convergencia más lenta debido a la naturaleza.

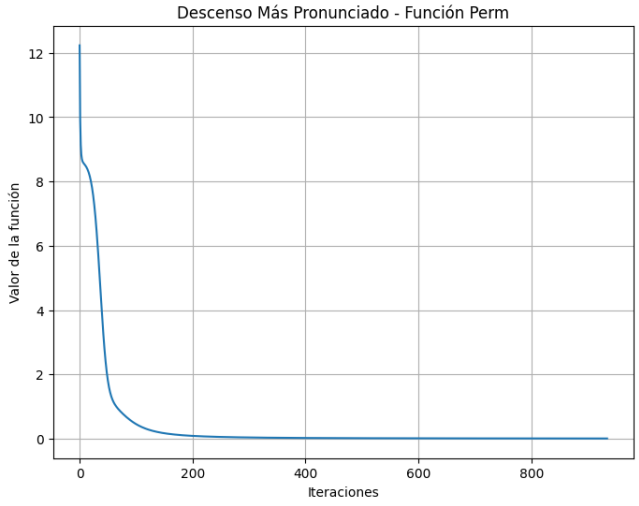

Figure 3: Convergencia del algoritmo SD para la función de Perm. La convergencia presenta comportamiento irregular debido a la complejidad de la función.

Los resultados detallados y la implementación completa del código se encuentran en el archivo `t03_alejandro_zarate_macias.ipynb`.

6.4 Discusión

Los resultados confirman las características de optimización vistas en el Problema 5:

- **Función de esfera**: Como se predijo, resultó la más sencilla de optimizar. Su naturaleza convexa cuadrática permite una convergencia rápida hacia el mínimo global. El gradiente analítico simple $\nabla f(\mathbf{x}) = 2(\mathbf{x} - \mathbf{c})$ facilita direcciones de descenso eficientes.
- **Función de Rosenbrock**: Representa un verdadero reto de optimización debido a su curvatura pronunciada. La convergencia es notablemente más lenta, requiriendo un ajuste cuidadoso del tamaño de paso para evitar oscilaciones o que los valores incrementen en lugar de acercarse a 0. El gradiente analítico, aunque complejo, proporciona direcciones precisas pero la geometría de la función limita la eficiencia del método.
- **Función de Perm**: Confirma ser un desafío significativo con convergencia irregular. La complejidad de su gradiente analítico con sumas anidadas y potencias variables requiere evaluaciones computacionalmente más costosas.

6.5 Conclusión

Se implementó exitosamente el algoritmo de descenso más pronunciado con gradientes analíticos para las tres funciones objetivo. Los resultados demuestran la efectividad del método para funciones convexas simples como la esfera, mientras revelan las limitaciones del método para funciones más complejas como Rosenbrock y Perm. El gradiente analítico garantiza precisión en las direcciones de descenso, aunque requiere un esfuerzo previo considerable en la derivación matemática.

7 Problema 7

7.1 Enunciado

Mejore su script anterior usando un gradiente numérico en lugar del gradiente analítico. A continuación, resuelva el problema de minimización para las funciones (1)-(3) usando el mismo número de iteraciones que antes. Realice una comparación de las soluciones obtenidas con las del Problema 6.

7.2 Metodología

Se modifica la implementación del Problema 6 reemplazando los gradientes analíticos por gradientes numéricos:

1. **Gradiente numérico**: Implementación usando diferencias finitas centrales:

$$\frac{\partial f}{\partial x_i} \approx \frac{f(\mathbf{x} + h\mathbf{e}_i) - f(\mathbf{x} - h\mathbf{e}_i)}{2h} \tag{1}$$

   donde $h = 10^{-5}$ y $\mathbf{e}_i$ es el vector unitario en la dirección i.

2. **Misma configuración experimental**: Se mantienen los mismos parámetros del Problema 6 (punto inicial, número de iteraciones, tamaño de paso) para permitir comparación directa.

3. **Análisis comparativo**: Evaluación de convergencia, precisión y costo computacional entre ambos métodos.

7.3 Resultados

Los resultados del algoritmo de descenso más pronunciado con gradiente numérico se presentan en las siguientes gráficas:

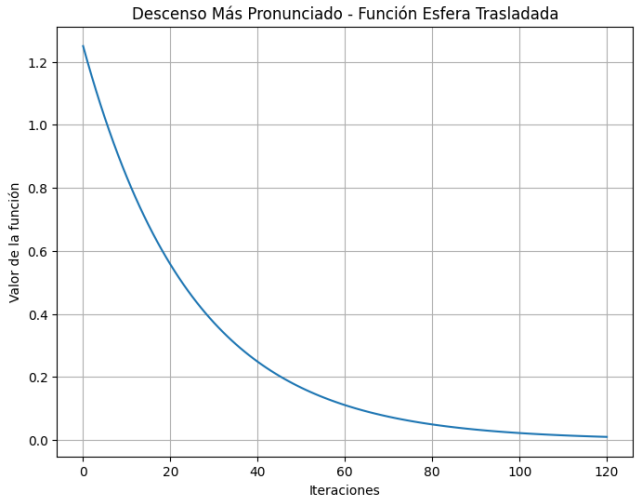

Figure 4: Convergencia del algoritmo SD con gradiente numérico para la función de esfera trasladada. El comportamiento es prácticamente idéntico al gradiente analítico.

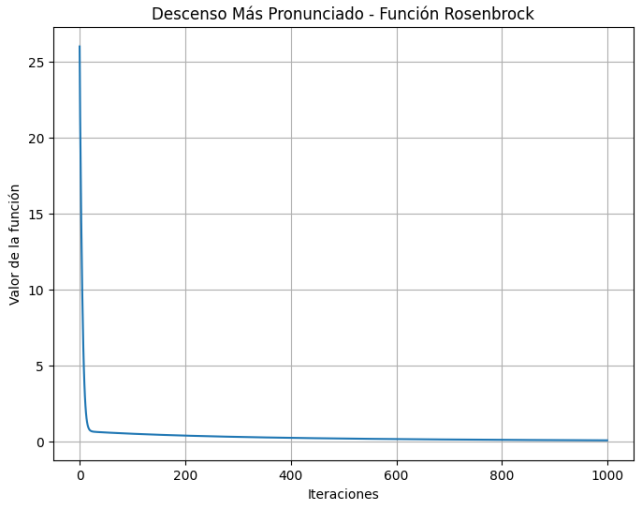

Figure 5: Convergencia del algoritmo SD con gradiente numérico para la función de Rosenbrock. Se observa convergencia similar al caso analítico con ligeras variaciones numéricas.

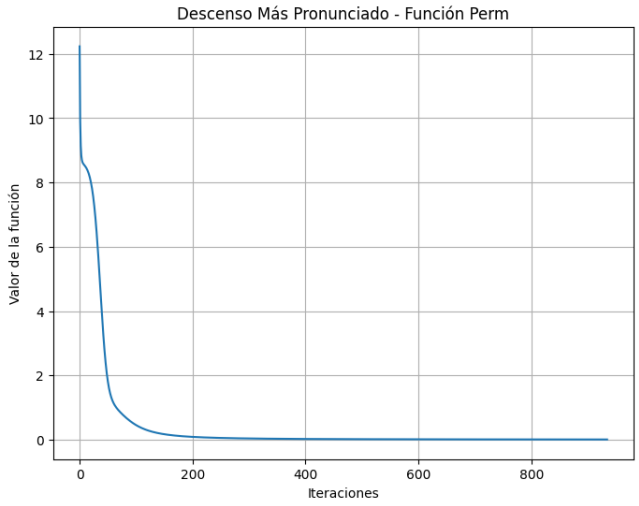

Figure 6: Convergencia del algoritmo SD con gradiente numérico para la función de Perm. El comportamiento irregular se mantiene consistente con el gradiente analítico.

Los resultados detallados y la implementación completa del código se encuentran en el archivo `t03_alejandro_zarate_macias.ipynb`.

7.4 Discusión

La comparación entre gradientes analíticos y numéricos revela resultados notablemente similares:

- **Comportamiento de convergencia**: Las tres funciones exhiben patrones de convergencia prácticamente idénticos entre ambos métodos. Las trayectorias de optimización son casi indistinguibles, confirmando la precisión del gradiente numérico implementado.
- **Facilidad de implementación**: El gradiente numérico ofrece una ventaja significativa en términos de implementación: no requiere derivación manual ni implementación específica para cada función. Un solo método genérico puede calcular gradientes para cualquier función diferenciable.
- **Robustez**: El método numérico elimina la posibilidad de errores en la derivación manual, especialmente relevante para funciones complejas como la de Perm donde el gradiente analítico involucra múltiples sumas anidadas.

7.5 Conclusión

Se demostró que el gradiente numérico produce resultados prácticamente idénticos al gradiente analítico para las tres funciones estudiadas, con la ventaja adicional de simplificar significativamente la implementación. La facilidad de no requerir derivación manual hace que el gradiente numérico sea una alternativa muy atractiva para problemas de optimización, especialmente cuando las funciones objetivo son complejas o cuando se necesita un enfoque genérico aplicable a diferentes tipos de funciones.

8 Problema 8

8.1 Enunciado

Mejore aún más su script añadiendo longitudes de paso de tipo decreciente lineal, adaptativa e inteligente (i.e. condiciones de Armijo o de Wolfe). Puede usar un gradiente analítico o numérico. Use algunas gráficas para comparar los resultados obtenidos con cada una de las cuatro estrategias de longitud de paso.

8.2 Metodología

Se implementan cuatro estrategias diferentes para el tamaño de paso en el algoritmo de descenso más pronunciado:

1. **Paso fijo**: Tamaño de paso constante durante toda la optimización (implementado en problemas anteriores).
2. **Paso decreciente lineal**: Reducción lineal del tamaño de paso conforme avanzan las iteraciones.

3. **Paso adaptativo**: Ajuste dinámico del tamaño de paso basado en el progreso de la función objetivo, incrementando cuando hay mejora y reduciendo cuando no la hay.

4. **Condición de Armijo**: Búsqueda de línea que satisface la condición de descenso suficiente mediante backtracking.

Se compararán estas estrategias usando las mismas tres funciones objetivo (esfera trasladada, Rosenbrock y Perm) con configuración experimental idéntica para evaluar su efectividad relativa.

8.3 Resultados

Los resultados de las cuatro estrategias de tamaño de paso se presentan en las siguientes gráficas de convergencia:

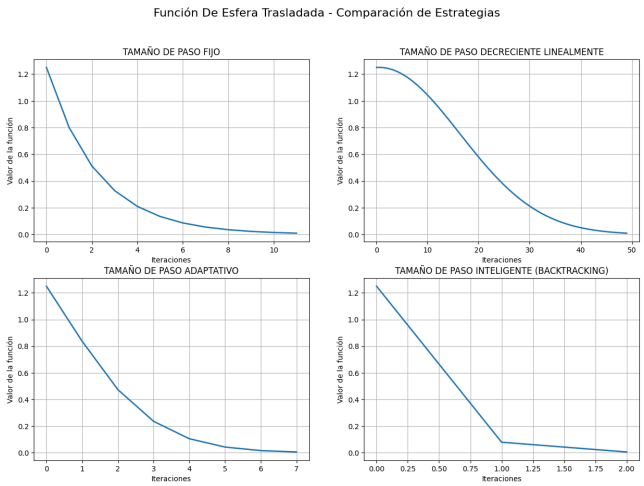

Figure 7: Comparación de estrategias de tamaño de paso para la función de esfera trasladada. Se observa que todas las estrategias convergen efectivamente, con Armijo mostrando la convergencia más estable.

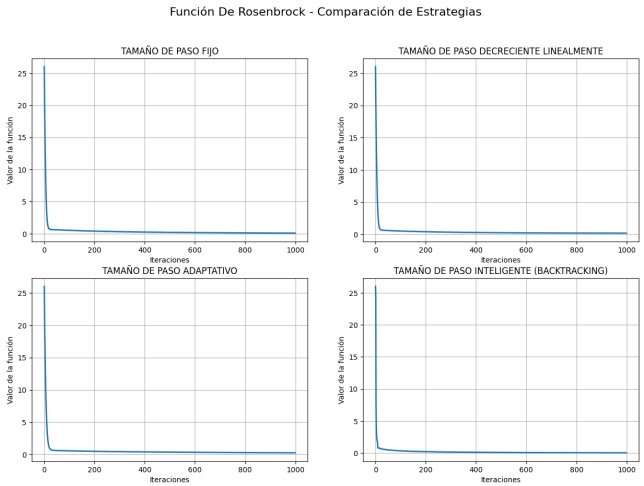

Figure 8: Comparación de estrategias de tamaño de paso para la función de Rosenbrock. Las estrategias adaptativas muestran ventajas significativas sobre el paso fijo en esta función desafiante.

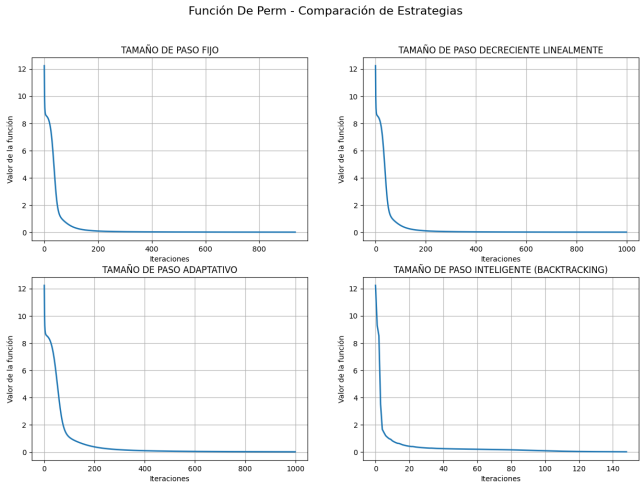

Figure 9: Comparación de estrategias de tamaño de paso para la función de Perm. La condición de Armijo demuestra mayor robustez en el manejo de la complejidad de esta función.

Adicionalmente, se presenta la evolución del tamaño de paso para cada estrategia:

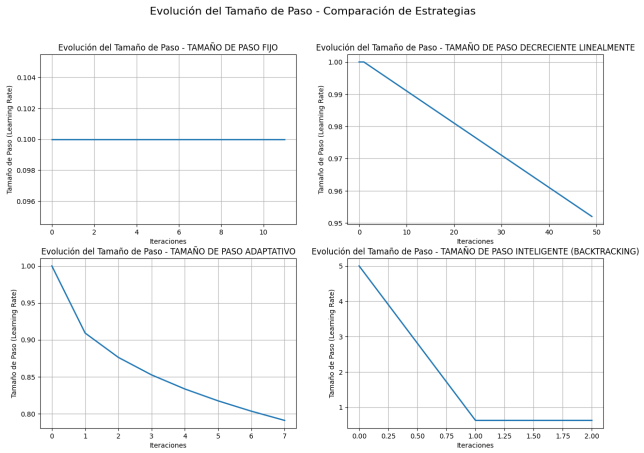

Figure 10: Evolución del tamaño de paso (learning rate) para la función de esfera trasladada. Se observa cómo cada estrategia ajusta dinámicamente el paso según su metodología específica.

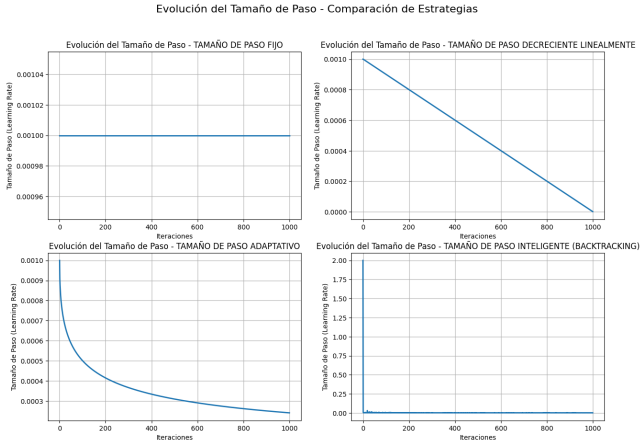

Figure 11: Evolución del tamaño de paso para la función de Rosenbrock. El método adaptativo y Armijo muestran ajustes más sofisticados que mejoran la convergencia.

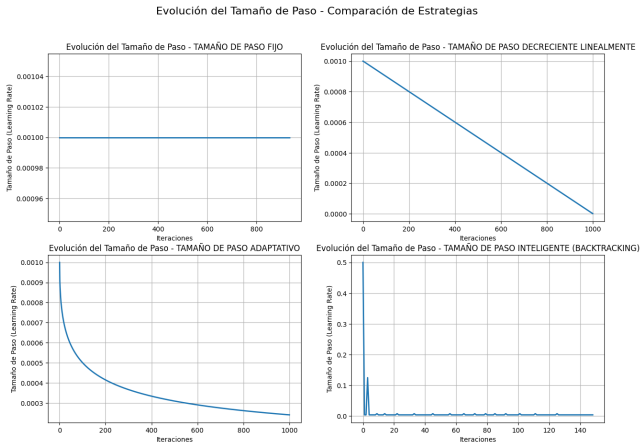

Figure 12: Evolución del tamaño de paso para la función de Perm. Se evidencia la importancia del control dinámico del paso en funciones complejas.

Los resultados detallados y la implementación completa del código se encuentran en el archivo `t03_alejandro_zarate_macias.ipynb`.

8.4 Discusión

La comparación de las cuatro estrategias revela patrones importantes sobre el comportamiento del tamaño de paso:

- **Paso fijo**: Proporciona comportamiento predecible pero puede ser subóptimo. Para la función esfera funciona bien, pero para Rosenbrock y Perm puede ser demasiado agresivo o conservador según la región del espacio de búsqueda.
- **Paso decreciente lineal**: Ofrece un balance entre exploración inicial (paso grande) y refinamiento final (paso pequeño). Funciona especialmente bien cuando se tiene una estimación del número total de iteraciones necesarias.
- **Paso adaptativo**: Demuestra excelente flexibilidad al ajustarse automáticamente según el progreso. Incrementa el paso cuando hay mejora (acelerando convergencia) y lo reduce cuando hay retroceso (evitando divergencia). Particularmente efectivo para Rosenbrock donde la curvatura varía significativamente.
- **Condición de Armijo**: Proporciona la estrategia más robusta con garantías teóricas de convergencia. Aunque computacionalmente más costosa (requiere múltiples evaluaciones de función por iteración), ofrece la convergencia más estable y confiable, especialmente para funciones complejas como Perm.

8.5 Conclusión

Se implementaron exitosamente cuatro estrategias diferentes de tamaño de paso, cada una con características distintivas. Los resultados demuestran que la elección de la estrategia de paso es crucial para el rendimiento del algoritmo, especialmente en funciones no convexas. La condición de Armijo se mostró como la más robusta para funciones complejas, mientras que el paso adaptativo ofreció un excelente balance entre simplicidad de implementación y efectividad. El paso decreciente lineal se mostró como una alternativa práctica cuando se conoce aproximadamente el horizonte de optimización.

9 Problema 9

9.1 Enunciado

Considere la siguiente función

$$f(x_1, x_2) = 10^9 x_1^2 + x_2^2. \tag{4}$$

Considere $\mathbf{x}_0 = (1.5, 1.5)$ como punto inicial. Resuelva el problema de minimización usando su script de SD. ¿Cuántas iteraciones necesita su implementación de SD para alcanzar un valor de la función menor que $1e^{-4}$? A continuación, escale las variables de (4). ¿Cuántas iteraciones necesita su implementación de SD para alcanzar un valor menor que $1e^{-4}$ en esta versión escalada de (4)?

9.2 Metodología

Se analiza el problema de optimización de la función mal condicionada $f(x_1, x_2) = 10^9 x_1^2 + x_2^2$ utilizando dos enfoques:

1. **Función original**: Aplicación directa del algoritmo SD con las cuatro estrategias de tamaño de paso implementadas en el Problema 8.

2. **Función escalada**: Normalización de variables para equilibrar los coeficientes mediante transformación $x_1' = \frac{x_1}{\sqrt{10^9}}$ y $x_2' = x_2$, resultando en una función mejor condicionada.

3. **Análisis comparativo**: Evaluación del número de iteraciones requeridas para alcanzar $f(\mathbf{x}) < 10^{-4}$ en ambos casos.

9.3 Resultados

Los resultados de optimización revelan diferencias dramáticas entre la función original y la escalada:

Función original ($f(x_1, x_2) = 10^9 x_1^2 + x_2^2$):

Todas las estrategias de tamaño de paso experimentaron convergencia prematura, quedando estancadas en valores cercanos a $f(\mathbf{x}) \approx 2$. Ninguna estrategia logró alcanzar el criterio de parada $f(\mathbf{x}) < 10^{-4}$, incluso después de aumentar significativamente el número máximo de iteraciones.

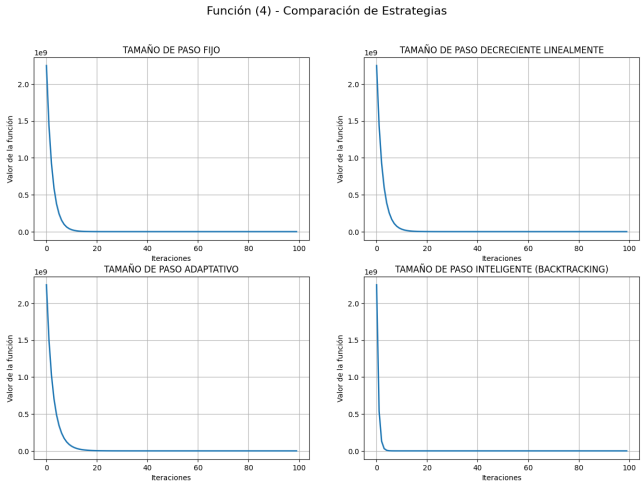

Figure 13: Convergencia del algoritmo SD para la función original mal condicionada. Todas las estrategias se estancan en valores cercanos a 2 sin alcanzar el criterio de convergencia.

Función escalada:

La normalización de variables permitió que todas las estrategias alcanzaran exitosamente el criterio de convergencia:

- **Paso fijo**: 26 iteraciones
- **Paso decreciente lineal**: 30 iteraciones
- **Paso adaptativo**: 4 iteraciones
- **Paso inteligente (Armijo)**: 5 iteraciones

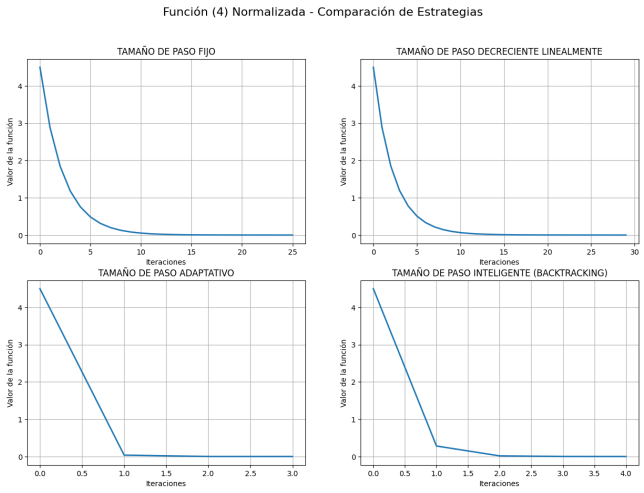

Figure 14: Convergencia del algoritmo SD para la función escalada. Todas las estrategias convergen exitosamente, con las estrategias adaptativas mostrando convergencia más rápida.

Los resultados detallados y gráficas comparativas se encuentran en el archivo `t03_alejandro_zarate_macias.ipynb`.

9.4 Discusión

Los resultados ilustran claramente el impacto devastador del mal condicionamiento en algoritmos de optimización:

- **Problema de condicionamiento**: La función original presenta un número de condición extremadamente alto ($\kappa \approx 10^9$) debido a la disparidad entre coeficientes. Esto crea un paisaje de optimización con forma de valle muy estrecho y alargado.
- **Estancamiento en función original**: El gradiente de la función mal condicionada produce direcciones de descenso que oscilan entre las dos variables sin hacer progreso efectivo hacia el mínimo. El algoritmo queda atrapado en un comportamiento de "zigzag" que impide la convergencia.
- **Efectividad del escalado**: La normalización equilibra los gradientes de ambas variables, permitiendo que el algoritmo encuentre direcciones de descenso más eficientes. El número de condición se reduce drásticamente, mejorando la geometría del problema.

9.5 Conclusión

Este problema demuestra la importancia crítica del preprocesamiento en optimización numérica. El escalado de variables transforma un problema intratable

en uno que converge eficientemente con todas las estrategias probadas. Los resultados indican que el mal condicionamiento puede ser más limitante que la elección del algoritmo de optimización, y que técnicas simples de normalización pueden producir mejoras dramáticas en el rendimiento. Las estrategias adaptativas e inteligentes emergieron como las más robustas, especialmente en el contexto de la función escalada.

10 Problema 10

10.1 Enunciado

La diferenciación analítica y numérica no son las únicas formas de calcular derivadas. Investigue un poco sobre la diferenciación automática. Luego, considere la función de Himmelblau:

$$f(x, y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2. \tag{5}$$

Calcule las derivadas parciales de (5) en el punto $(1, -1)$ usando diferenciación automática (dibuje en papel las gráficas correspondientes y muestre su procedimiento). ¿Qué ventajas y desventajas tienen la diferenciación analítica, numérica y automática al compararlas entre sí?

10.2 Metodología

Para aplicar diferenciación automática se seguirá el modo directo (forward mode):

1. Descomponer la función en operaciones elementales.
2. Construir el grafo correspondiente.
3. Evaluar simultáneamente la función y sus derivadas usando la regla de la cadena.
4. Comparar los tres métodos de diferenciación.

10.3 Resultados

La función de Himmelblau se puede descomponer como:

$$f(x, y) = u_1^2 + u_2^2 \tag{1}$$

donde $u_1 = x^2 + y - 11$ y $u_2 = x + y^2 - 7$.

El grafo computacional correspondiente se muestra en la Figura 15:

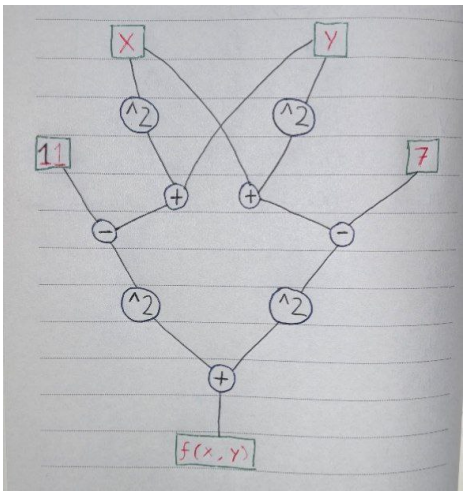

Figure 15: Grafo computacional para la función de Himmelblau usando diferenciación automática.

Usando el grafo dibujado, evaluamos en el punto $(1, -1)$:
Para $\frac{\partial f}{\partial x}$**:** Inicializamos con $\dot{x} = 1, \dot{y} = 0$.

Operación	**Valor**	**Derivada**	**Cálculo**
x	1	1	Entrada inicial
y	-1	0	Entrada inicial
$v_1 = x^2$	1	2	$\dot{v_1} = 2x\dot{x} = 2(1)(1) = 2$
$v_2 = y^2$	1	0	$\dot{v_2} = 2y\dot{y} = 2(-1)(0) = 0$
$v_3 = v_1 + y$	0	2	$\dot{v_3} = \dot{v_1} + \dot{y} = 2 + 0 = 2$
$v_4 = x + v_2$	2	1	$\dot{v_4} = \dot{x} + \dot{v_2} = 1 + 0 = 1$
$v_5 = v_3 - 11$	-11	2	$\dot{v_5} = \dot{v_3} = 2$
$v_6 = v_4 - 7$	-5	1	$\dot{v_6} = \dot{v_4} = 1$
$v_7 = v_5^2$	121	-44	$\dot{v_7} = 2v_5\dot{v_5} = 2(-11)(2) = -44$
$v_8 = v_6^2$	25	-10	$\dot{v_8} = 2v_6\dot{v_6} = 2(-5)(1) = -10$
$f = v_7 + v_8$	146	-54	$\dot{f} = \dot{v_7} + \dot{v_8} = -44 + (-10) = -54$

Table 1: Evaluación forward para $\frac{\partial f}{\partial x}$ en $(1, -1)$

Para $\frac{\partial f}{\partial y}$**:** Inicializamos con $\dot{x} = 0, \dot{y} = 1$.
Proceso paso a paso:
1. Inicialización**:** Se establecen las variables de entrada $(x, y) = (1, -1)$ y las semillas direccionales $(\dot{x}, \dot{y})$ según la derivada que se quiera calcular.
2. Operaciones elementales**:** Cada operación en el grafo se evalúa propagando tanto el valor como su derivada usando las reglas:

- Potenciación: $\frac{d}{dx}(u^n) = nu^{n-1}\frac{du}{dx}$

Operación	**Valor**	**Derivada**	**Cálculo**
x	1	0	Entrada inicial
y	-1	1	Entrada inicial
$v_1 = x^2$	1	0	$\dot{v_1} = 2x\dot{x} = 2(1)(0) = 0$
$v_2 = y^2$	1	-2	$\dot{v_2} = 2y\dot{y} = 2(-1)(1) = -2$
$v_3 = v_1 + y$	0	1	$\dot{v_3} = \dot{v_1} + \dot{y} = 0 + 1 = 1$
$v_4 = x + v_2$	2	-2	$\dot{v_4} = \dot{x} + \dot{v_2} = 0 + (-2) = -2$
$v_5 = v_3 - 11$	-11	1	$\dot{v_5} = \dot{v_3} = 1$
$v_6 = v_4 - 7$	-5	-2	$\dot{v_6} = \dot{v_4} = -2$
$v_7 = v_5^2$	121	-22	$\dot{v_7} = 2v_5\dot{v_5} = 2(-11)(1) = -22$
$v_8 = v_6^2$	25	20	$\dot{v_8} = 2v_6\dot{v_6} = 2(-5)(-2) = 20$
$f = v_7 + v_8$	146	-2	$\dot{f} = \dot{v_7} + \dot{v_8} = -22 + 20 = -2$

Table 2: Evaluación forward para $\frac{\partial f}{\partial y}$ en $(1, -1)$

- Suma: $\frac{d}{dx}(u + v) = \frac{du}{dx} + \frac{dv}{dx}$
- Constante: $\frac{d}{dx}(u + c) = \frac{du}{dx}$

3. Propagación**:** Los valores y derivadas se propagan hacia adelante en el grafo hasta llegar a la función objetivo.

Resultados en $(1, -1)$:

$$f(1, -1) = 146 \tag{2}$$

$$\left.\frac{\partial f}{\partial x}\right|_{(1,-1)} = -54 \tag{3}$$

$$\left.\frac{\partial f}{\partial y}\right|_{(1,-1)} = -2 \tag{4}$$

10.4 Discusión

Comparación de métodos de diferenciación:

- Diferenciación Analítica:
  - Ventajas: Exacta, expresión simbólica, eficiente para evaluaciones múltiples
  - Desventajas: Requiere derivación manual, propensa a errores, compleja para funciones complicadas
- Diferenciación Numérica:
  - Ventajas: Fácil de implementar, aplicable a cualquier función
  - Desventajas: Errores de truncamiento y redondeo, inestable numéricamente, costosa computacionalmente

- Diferenciación Automática:
  - Ventajas: Exacta hasta precisión de máquina, automática, eficiente
  - Desventajas: Requiere software especializado, overhead de memoria en modo reverso

10.5 Conclusión

La diferenciación automática combina las ventajas de los métodos analítico y numérico: proporciona derivadas exactas sin requerir derivación manual. En el ejemplo de la función de Himmelblau, se obtuvieron las derivadas parciales $\frac{\partial f}{\partial x} = -54$ y $\frac{\partial f}{\partial y} = -2$ en $(1, -1)$ de manera sistemática y precisa usando el grafo computacional.