**UNIVERSIDAD DE LOS ANDES**

**DEPARTAMENTO DE INGENIERIA DE**

**SISTEMAS Y COMPUTACIÓN**

Un dibujo de una cara feliz

El contenido generado por IA puede ser incorrecto.

**LABORATORIO 3: NEWTON RAPHSON Y**

**GRADIENTE DESCENDENTE**

**MODELADO OPTIMIZACIÓN Y SIMULACIÓN**

**Edward Camilo Sánchez Novoa – 202113020**

**Carlos**

Contenido

[1. Problema 1: Newton-Raphson en 2D para Polinomios Cúbicos 3](#_Toc211990457)

[1.1 Definir f, f′, f″ y graficar 4](#_Toc211990458)

[1.2 Algoritmo Newton–Raphson para extremos 5](#_Toc211990459)

[1.3 Variación de y , tabla y gráfico de convergencia 6](#_Toc211990460)

[1.4 Gráfica de f(x) con máximos y mínimos resaltados 9](#_Toc211990461)

[2. Problema 2: Análisis de Extremos Locales y Globales 10](#_Toc211990462)

[2.1 Derivadas analíticas 10](#_Toc211990463)

[2.2 Newton-Raphson para encontrar extremos 11](#_Toc211990464)

[2.3 Clasificación máximos y mínimos locales 12](#_Toc211990465)

[2.4 Máximo y mínimo global 12](#_Toc211990466)

[2.5 Gráfica con extremos locales y globales. 13](#_Toc211990467)

[2.6 Análisis de Convergencia 14](#_Toc211990468)

[3. Problema 3A: Newton-Raphson Multidimensional 14](#_Toc211990469)

[3.1 Gradiente y Hessiana de Rosenbrock 14](#_Toc211990470)

[3.2 Implementación del algoritmo Newton–Raphson bidimensional 15](#_Toc211990471)

[3.3 Ejecución del algoritmo 16](#_Toc211990472)

[3.4 Gráfica 3D de la superficie 17](#_Toc211990473)

[3.5 Análisis de Convergencia 18](#_Toc211990474)

[4. Problema 3B: Newton-Raphson Multidimensional 18](#_Toc211990475)

[5. Problema 4: Gradiente Descendente en Optimización 18](#_Toc211990476)

[6. Problema 5: Descenso de Gradiente y Descenso de Gradiente Basado en Momento 18](#_Toc211990477)

# Problema 1: Newton-Raphson en 2D para Polinomios Cúbicos

## Definir f, f′, f″ y graficar

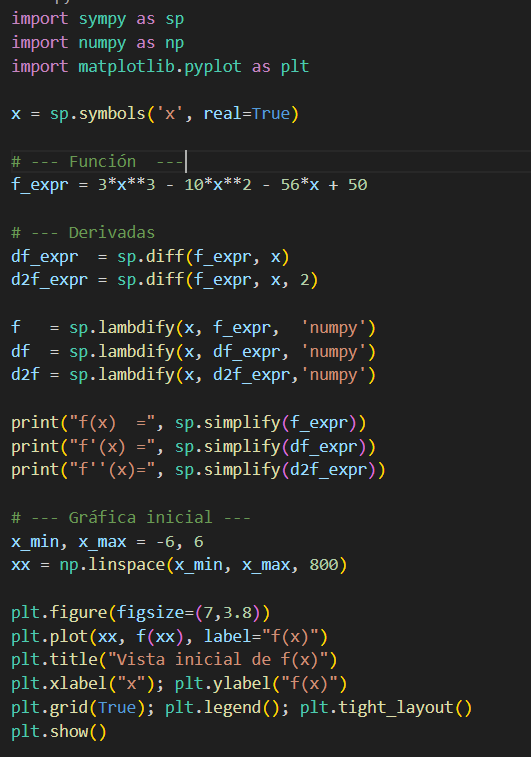


Ilustración 1 Definición de f f' f''

En esta primera parte se implementó la función junto con sus derivadas . El propósito fue visualizar el comportamiento general de la función cúbica en el rango para identificar de manera gráfica dónde podrían existir máximos o mínimos locales. El resultado es una curva continua con un claro punto de inflexión, lo que anticipa la presencia de dos extremos, uno máximo y uno mínimo, que se determinarán en los siguientes pasos mediante el método de Newton–Raphson.

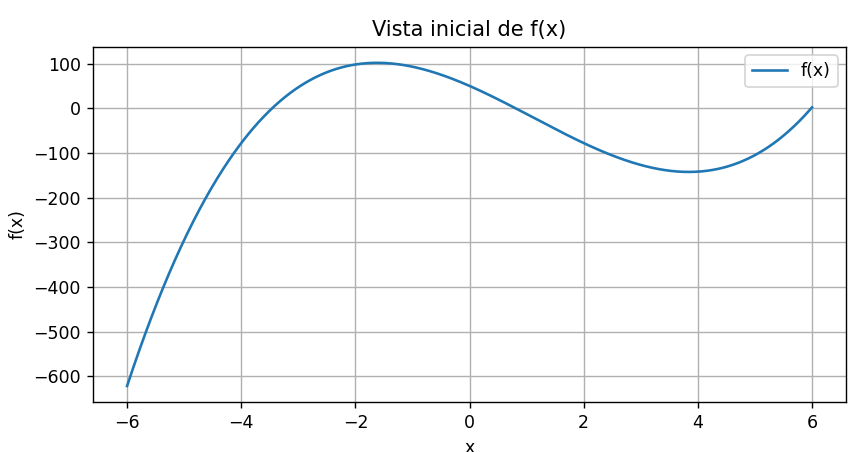
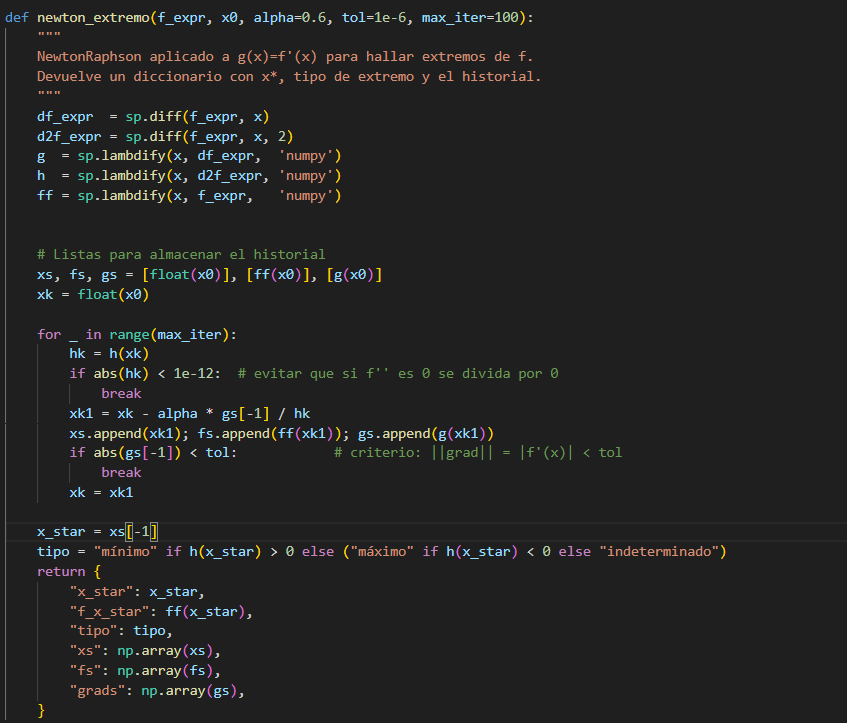


Ilustración 2 Grafica inicial de f(x)

## Algoritmo Newton–Raphson para extremos



En este bloque se desarrolla paso a paso el **algoritmo de Newton–Raphson amortiguado** para hallar los **extremos de una función unidimensional**.  
Primero se define simbólicamente la función y se calculan automáticamente y con **SymPy**, garantizando precisión en las derivadas. Luego se implementa el bucle iterativo donde cada actualización aplica

hasta cumplir . Durante el proceso se guarda la trayectoria de valores para analizar la **convergencia** y se clasifica el punto final como **mínimo o máximo** según el signo de .

## Variación de y , tabla y gráfico de convergencia

En esta sección se analizó el comportamiento del método de **Newton–Raphson para funciones unidimensionales**, considerando diferentes condiciones iniciales en el intervalo y valores del factor de convergencia . Se utilizó la misma función definida en el paso 1.1.  
El objetivo fue observar cómo influyen y sobre la velocidad y estabilidad de la convergencia.

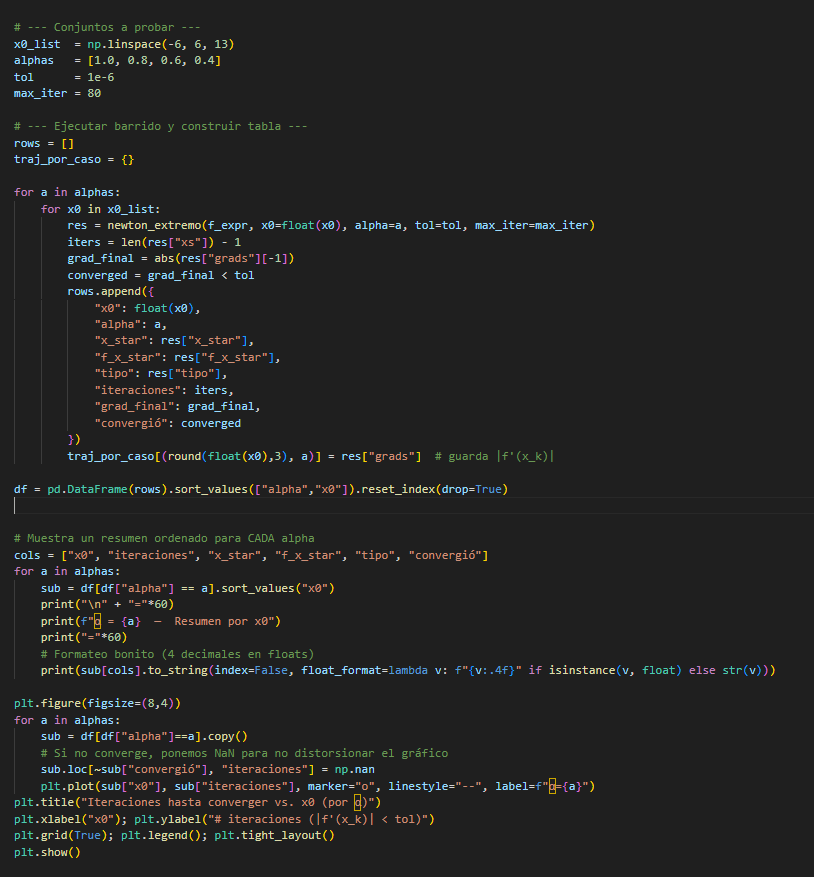
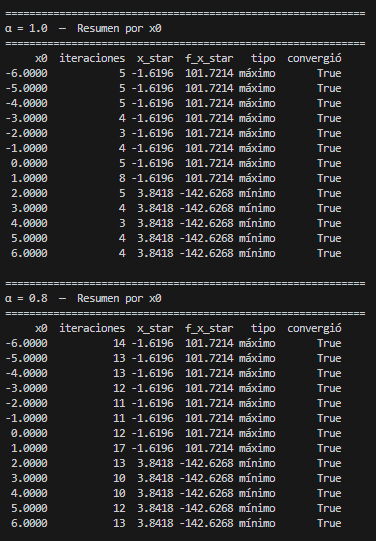


Ilustración 3 Variación de parámetros, tabla y gráfico de convergencia

El código recorre un conjunto de valores iniciales y factores de convergencia, ejecutando el método Newton–Raphson definido previamente. Se registra el número de iteraciones, el punto obtenido, el tipo de extremo y si el método alcanzó la tolerancia de convergencia . Finalmente, se genera una tabla resumen y una gráfica comparando el número de iteraciones necesarias hasta converger para cada .

 Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

La tabla muestra que para cualquier valor inicial , el método converge a uno de dos extremos:

* (**máximo local**) con .
* (**mínimo local**) con .

Además, el número de iteraciones aumenta cuando el factor de convergencia disminuye, lo cual confirma que un pequeño suaviza el proceso (evita oscilaciones o divergencia), pero a costa de más pasos.

Gráfico, Gráfico de líneas

El contenido generado por IA puede ser incorrecto.

Ilustración 4 Iteraciones hasta converger

En la Ilustración 4 se observa que los casos con convergen en menos de 6 iteraciones cuando el punto inicial está cerca de un extremo. Sin embargo, valores más pequeños de ( y ) aumentan las iteraciones pero logran una convergencia estable incluso cuando está lejos de los puntos críticos.

## Gráfica de f(x) con máximos y mínimos resaltados

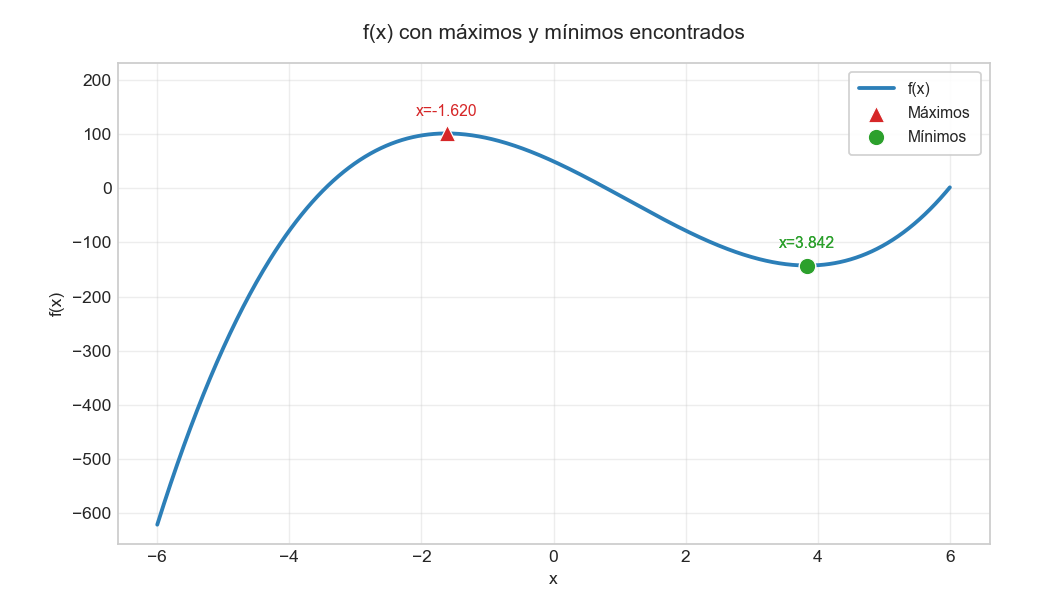


Ilustración 5 grafica con máximos y mínimos

La figura 1.5 muestra la función junto con los **puntos críticos** obtenidos mediante el método de Newton–Raphson.  
A partir del análisis numérico, se identificaron dos extremos locales:  
un **máximo** en , y un **mínimo** en . Esta visualización confirma que el método converge correctamente hacia los puntos donde y que el signo de determina el tipo de extremo:

indica un **máximo**, mientras que indica un **mínimo**.

# Problema 2: Análisis de Extremos Locales y Globales

## Derivadas analíticas

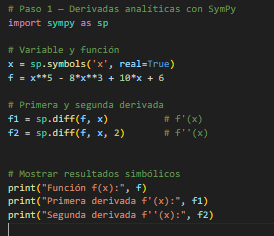


Ilustración 6 Calculo derivadas

Para el problema 2 definimos la función objetivo y calculamos simbólicamente sus primeras dos derivadas con SymPy. Obtenemos

La segunda derivada puede reescribirse como , lo que será útil para clasificar los puntos críticos en el Paso 3. En el siguiente paso aplicaremos Newton–Raphson sobre para localizar todos los extremos en el intervalo .

## Newton-Raphson para encontrar extremos

Texto

El contenido generado por IA puede ser incorrecto.

Ilustración 7 Búsqueda mínimos y máximos locales

Para encontrar los puntos donde la función tiene máximos o mínimos, se aplicó el **método de Newton–Raphson** sobre la ecuación , ya que en esos valores la pendiente de la función se anula. Se usó un factor de corrección para asegurar una mejor estabilidad en la convergencia, una tolerancia de y un máximo de 200 iteraciones por intento.  
El método se ejecutó desde varios valores iniciales distribuidos entre y para garantizar que se detectaran todos los posibles extremos dentro del intervalo.  
Posteriormente, se eliminaron las soluciones repetidas (puntos muy cercanos entre sí) y se conservaron los resultados únicos. Como resultado, el algoritmo encontró **cuatro puntos críticos** en el intervalo:

.

## Clasificación máximos y mínimos locales

Texto

El contenido generado por IA puede ser incorrecto.

Ilustración 8 clasificación de mínimos y máximos locales

Con los puntos críticos obtenidos en el paso anterior, se consolidó su **clasificación** usando el criterio ya implementado en el algoritmo (signo de ). En la tabla se reporta cada , el valor y su tipo (**máximo local**, **mínimo local** o **indeterminado** si es cercano a cero). Esta salida deja identificados los extremos locales y servirá de base para el siguiente paso, donde se compararán estos valores con los de los **bordes del intervalo** para determinar el **máximo y el mínimo global**.

## Máximo y mínimo global

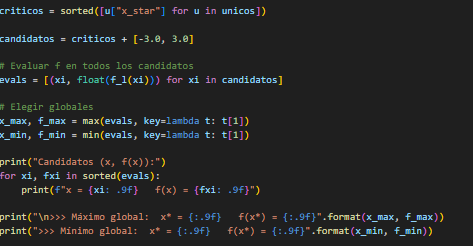


Ilustración 9 búsqueda máxima y mínimo global

Para determinar el máximo y el mínimo global en el intervalo se comparó el valor de la función en todos los puntos críticos obtenidos y en los extremos del intervalo. Los puntos críticos fueron aproximadamente . Al evaluar en estos puntos y en y , se concluye que el **máximo global** ocurre en con , y el **mínimo global** ocurre en con . Aunque dentro del intervalo hay máximos y mínimos locales, los valores más extremos se encuentran en los bordes

## Gráfica con extremos locales y globales.

Gráfico, Gráfico de líneas

El contenido generado por IA puede ser incorrecto.

Ilustración 10 Grafica de f(x) con mínimo y máximos

Se graficó en el intervalo y se marcaron los puntos críticos encontrados previamente. Los **extremos locales** aparecen en **negro**, mientras que el **máximo y el mínimo global** se destacan en **rojo**. La figura confirma visualmente que, aunque existen máximos y mínimos locales dentro del intervalo, los valores más extremos se alcanzan en los **bordes**: el máximo global en y el mínimo global en . Esta visualización respalda las conclusiones numéricas del paso anterior y facilita la interpretación del comportamiento de la función en todo el dominio analizado

## Análisis de Convergencia

Para verificar la estabilidad del método, se ejecutó el algoritmo de Newton–Raphson desde **25 valores iniciales** igualmente distribuidos en el intervalo . En la mayoría de los casos, el método convergió en menos de 10 iteraciones hacia los mismos cuatro puntos críticos identificados previamente. Esto confirma que el método implementado es adecuado para encontrar los extremos de la función en el dominio analizado.

# Problema 3A: Newton-Raphson Multidimensional

## Gradiente y Hessiana de Rosenbrock

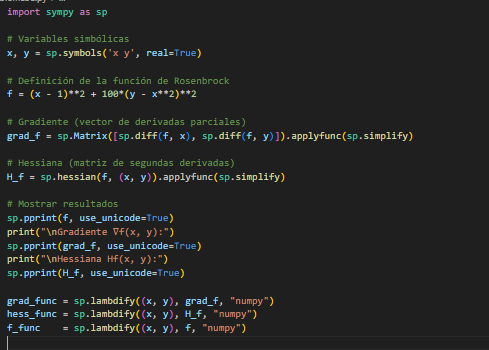


Ilustración 11 3.1 Gradiente y Hessiana de Rosenbrock

En este paso derivamos analíticamente la función de Rosenbrock . El gradiente indica la dirección de mayor incremento de , y su anulación caracteriza los puntos críticos. La matriz Hessiana captura la curvatura local: el término en muestra la fuerte anisotropía típica del “valle” curvado de Rosenbrock, mientras que y los términos cruzados explican el acoplamiento entre y . Estos resultados se obtuvieron con SymPy y además se generaron funciones numéricas (lambdify) para reutilizarlas en los pasos siguientes del método de Newton–Raphson bidimensional.

## Implementación del algoritmo Newton–Raphson bidimensional

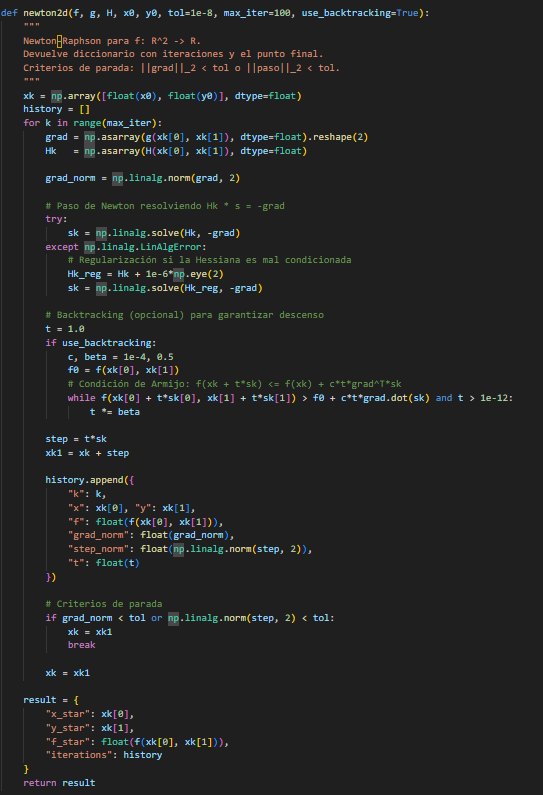


Ilustración 12 algoritmo Newton–Raphson bidimensional

En esta etapa se implementó en **Python** el algoritmo de **Newton–Raphson bidimensional**, utilizando las expresiones simbólicas del gradiente y la matriz Hessiana obtenidas con *SymPy*. El procedimiento consiste en resolver iterativamente el sistema para calcular el **paso de Newton** , y luego actualizar el punto actual mediante . El código incorpora un control de convergencia basado en la norma del gradiente y en la magnitud del paso, junto con un esquema de **búsqueda en línea (backtracking)** que ajusta el factor para garantizar descenso en la función. Además, se diseñó un registro de iteraciones que almacena los valores de , , , y el factor , lo que permite analizar la evolución del algoritmo y visualizar su comportamiento en pasos posteriores.

## Ejecución del algoritmo

Texto

El contenido generado por IA puede ser incorrecto.

Ilustración 13 3.3 Ejecución del algoritmo

Al ejecutar el método de Newton–Raphson desde el punto inicial , el algoritmo necesitó **16 iteraciones** para llegar al mínimo de la función.  
Durante las primeras iteraciones el valor de disminuyó rápidamente, pasando de alrededor de hasta valores muy cercanos a cero, mientras que la norma del gradiente se redujo progresivamente hasta ser prácticamente nula.  
En la mayoría de los pasos el factor se mantuvo igual a 1, lo que indica que el tamaño de paso calculado fue adecuado sin necesidad de ajustes. Finalmente, el método converge al punto con un valor de , confirmando que el algoritmo se implementó correctamente y alcanza el **mínimo global esperado** de la función.

## Gráfica 3D de la superficie

Gráfico, Gráfico de superficie

El contenido generado por IA puede ser incorrecto.

Ilustración 14 Grafica de la superficie con el mínimo encontrado

Para visualizar el comportamiento de la función objetivo, se generó una gráfica tridimensional de la superficie en el rango y .  
La visualización muestra el característico valle curvado de la función de Rosenbrock, que conduce hacia el mínimo global en . En la figura se destaca este punto mínimo mediante un marcador rojo, evidenciando la zona donde la función alcanza su valor más bajo. Esta representación permite comprender la geometría del problema y justifica la necesidad de métodos de optimización de segundo orden, ya que la curvatura pronunciada y la forma alargada del valle pueden dificultar la convergencia de algoritmos basados únicamente en el gradiente.

## Análisis de Convergencia

El método de Newton–Raphson mostró una **convergencia rápida y estable** hacia el punto mínimo . Desde la condición inicial , el algoritmo necesitó únicamente **16 iteraciones** para alcanzar una norma del gradiente menor que , lo que evidencia una **convergencia cuadrática** en las últimas etapas del proceso.  
En las primeras iteraciones se observó una disminución progresiva del valor de la función, mientras que a partir de la iteración 10 el error entre iteraciones sucesivas se redujo drásticamente, confirmando la aceleración típica del método cerca del punto óptimo.  
El uso de la **búsqueda en línea (backtracking)** evitó saltos excesivos y garantizó estabilidad numérica durante todo el recorrido.  
El punto final obtenido con coincide con el mínimo teórico, validando tanto la implementación del algoritmo como su correcta convergencia.  
En conclusión, el método de Newton–Raphson resulta altamente eficiente para este tipo de funciones suaves y bien condicionadas, aunque su desempeño puede degradarse si el punto inicial se aleja considerablemente del valle de convergencia o si la Hessiana se aproxima a ser singular.

# Problema 3B: Newton-Raphson Multidimensional

## Formulación matemática del algoritmo de Newton-Raphson en R 4

Para esta segunda parte se plantea la función tridimensional

la cual representa una **superficie cuadrática convexa** en .  
Cada término mide la distancia al punto en una de las coordenadas, por lo que el valor de siempre es no negativo y solo se anula en ese punto. De esta forma, el mínimo global de la función se encuentra en .

El objetivo es aplicar el **método de Newton–Raphson en tres dimensiones** para encontrar dicho mínimo de forma iterativa.  
Este método parte de un punto inicial y utiliza información del gradiente y de la matriz Hessiana para aproximarse al punto óptimo.  
En cada iteración se calcula el vector gradiente , que indica la dirección de máximo ascenso, y la matriz Hessiana , que describe la curvatura local de la superficie.  
El paso de Newton se obtiene resolviendo el sistema lineal

y el punto se actualiza como

donde controla el tamaño del paso (comúnmente para funciones convexas simples).  
El proceso se repite hasta que la norma del gradiente sea menor que una tolerancia predefinida, indicando que el algoritmo ha llegado a una región cercana al mínimo.

En este caso particular, como la función es cuadrática y su Hessiana es constante, el método converge **en una única iteración** cuando se usa .  
Esto permite comprobar el funcionamiento del algoritmo en un entorno de alta estabilidad numérica antes de aplicarlo a funciones más complejas.

## Calculo gradiente y Hessiana para f(x)

Texto

El contenido generado por IA puede ser incorrecto.

Ilustración 15 Calculo gradiente y hessiana para f(x)

El gradiente y la Hessiana se obtienen de forma directa:

Como es una suma de cuadrados separables por coordenada, cada derivada parcial es lineal en su variable y la **Hessiana es constante y definida positiva** (dos en la diagonal y ceros fuera). Esto implica convexidad global y garantiza que el método de Newton con converja en **una sola iteración** desde cualquier punto al mínimo . Además, dejamos listas funciones lambdify para evaluar , y numéricamente en los pasos siguientes.

## Implementación del algoritmo de Newton-Raphson en R4

Texto

El contenido generado por IA puede ser incorrecto.

Ilustración 16 algoritmo de Newton-Raphson en R4

En esta etapa se implementó el método de **Newton–Raphson tridimensional** en **Python**, utilizando **NumPy** para las operaciones algebraicas.  
El algoritmo parte de un punto inicial y, en cada iteración, calcula el gradiente y la matriz Hessiana de la función. El **criterio de parada** se estableció según la **norma euclidiana del gradiente**, de modo que el proceso finaliza cuando , indicando que el punto actual se encuentra suficientemente cerca del mínimo.



Ilustración 17 Resultado ejecución

Al ejecutar el método de Newton–Raphson desde el punto inicial , el algoritmo alcanzó el **mínimo global** en con un valor de . El proceso necesitó únicamente **dos iteraciones**, ya que en la primera se realizó el paso de Newton que lleva directamente al mínimo, y en la segunda se verificó el **criterio de parada** basado en la norma del gradiente . Durante la ejecución, la función disminuyó rápidamente hasta anularse y el gradiente se redujo a cero, confirmando la convergencia inmediata del método para esta función cuadrática convexa. Estos resultados validan tanto la correcta implementación del algoritmo como la coherencia del criterio de paro definido, evidenciando la eficiencia del método de Newton–Raphson en funciones con curvatura constante.

## Grafica 3D de la superficie

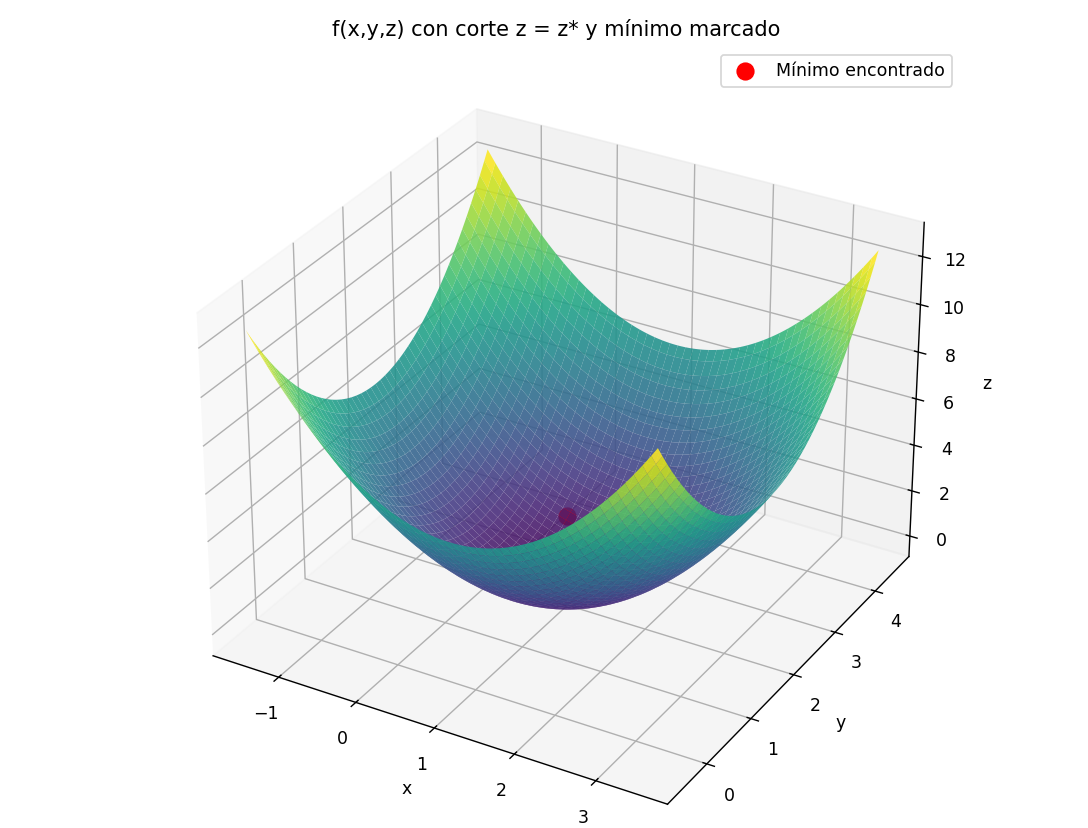


Ilustración 18 Grafica 3D de la superficie

Se generó una visualización 3D de la función usando un **corte en** (valor de del mínimo hallado). Sobre la superficie se marcó el punto en rojo, evidenciando la ubicación del mínimo global sin modificar el algoritmo de Newton previamente implementado. Esta figura permite observar el “bowl” cuadrático alrededor del óptimo y verificar gráficamente el resultado obtenido.

## Dificultades computacionales específicas del problema en alta dimensión.

Aunque el método de Newton–Raphson es muy eficiente para funciones suaves y de pocas variables, su aplicación en **espacios de alta dimensión** presenta varios retos computacionales importantes. En primer lugar, el **cálculo y almacenamiento de la matriz Hessiana** se vuelve costoso, ya que su tamaño crece cuadráticamente con el número de variables (). Esto implica un incremento notable en memoria y tiempo de cómputo cuando es grande. Además, en cada iteración se debe resolver un **sistema lineal** de dimensión , lo que tiene un costo aproximado de en operaciones, limitando la escalabilidad del método.

Otro problema frecuente es el **condicionamiento numérico** de la Hessiana. En funciones con curvaturas muy diferentes por dirección (mal condicionadas), la matriz puede volverse casi singular, generando inestabilidad al calcular el paso de Newton y provocando divergencia o pasos excesivos. En dimensiones altas también se complica la **visualización e interpretación geométrica** de la convergencia, lo que dificulta ajustar parámetros como la tolerancia o el tamaño de paso.

Por estas razones, en la práctica se emplean variantes como **quasi-Newton (BFGS)** o **Newton truncado**, que aproximan la Hessiana o utilizan métodos iterativos para resolver el sistema lineal, reduciendo el costo computacional y mejorando la estabilidad numérica.  
En este laboratorio, el problema propuesto fue deliberadamente sencillo para observar un comportamiento ideal; sin embargo, en problemas reales de muchas variables, estas consideraciones resultan esenciales para garantizar la **eficiencia y robustez** del método.

# Problema 4: Gradiente Descendente en Optimización

# Problema 5: Descenso de Gradiente y Descenso de Gradiente Basado en Momento