

# Optimización de la función

$$f(x, y) = x^4 - 4x^3 + 4x + y^2$$

Nombre: Ruben Martinez Rojas  
Grupo: C311

November 27, 2025

## 1 Modelos a analizar

La función a analizar es:

$$f(x, y) = x^4 - 4x^3 + 4x + y^2$$

## 2 Análisis de los modelos

### 2.1 Existencia de solución

Para estudiar la existencia de puntos óptimos, se calcularán los ceros del vector gradiente de la función, dado que todo mínimo (o máximo) local diferenciable debe satisfacer

$$\nabla f(x, y) = 0$$

. El gradiente está dado por el vector de derivadas parciales:

$$\nabla f(x, y) = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} 4x^3 - 12x^2 + 4 \\ 2y \end{bmatrix}$$

Para encontrarlos, igualamos cada componente del gradiente a cero:

$$\frac{\partial f}{\partial x} = 4x^3 - 12x^2 + 4 = 0, \quad \frac{\partial f}{\partial y} = 2y = 0.$$

De la ecuación en  $y$  se obtiene directamente:

$$y = 0.$$

Para la ecuación en  $x$ , simplificamos dividiendo entre 4:

$$x^3 - 3x^2 + 1 = 0.$$

Esta ecuación cúbica puede tener una, dos o tres raíces reales.

## 2.2 Convexidad

La matriz Hessiana de la función

$$f(x, y) = x^4 - 4x^3 + 4x + y^2$$

es:

$$H(x, y) = \begin{bmatrix} 12x^2 - 24x & 0 \\ 0 & 2 \end{bmatrix}.$$

Para que la función sea convexa,  $H(x, y)$  debe ser definida positiva. Vemos que no lo es para todos los valores de  $x$ , por lo tanto la función no es convexa globalmente. Sin embargo, se puede analizar localmente para regiones donde  $12x^2 - 24x > 0$ , es decir, para  $x > 2$  o  $x < 0$ . Pues en estas regiones donde la Hessiana es definida positiva, la función tiene curvatura positiva en todas las direcciones, lo cual implica que alrededor del punto es similar a un tazón. En estas regiones la curvatura de la función no cambia de signo y las direcciones de descenso son estables. Por ello, los métodos de Gradiente, Newton y Quasi-Newton tienden a converger rápidamente hacia un mínimo local.

En cambio, cuando la Hessiana posee valores negativos (como ocurre en la región  $0 < x < 2$ ), la función presenta curvatura negativa en alguna dirección, generando zonas tipo silla de montar. Allí los métodos de optimización tienen comportamientos menos predecibles: el método de Gradiente puede oscilar o avanzar muy lentamente debido a cambios abruptos en la pendiente, mientras que el método de Newton puede producir pasos en direcciones incorrectas, dirigirse hacia un máximo local o incluso divergir por completo. Esto se debe a que la inversión del Hessiano introduce direcciones inestables cuando existen eigenvalores negativos.

Por este motivo, en regiones no convexas es más probable que los métodos queden atrapados en mínimos locales o se comporten de manera errática, y se vuelve necesario usar técnicas globales como Multi-start para explorar distintas zonas del dominio.

La estrategia Multi-start consiste en ejecutar un algoritmo de búsqueda de mínimo local desde muchos puntos iniciales distintos. Cada ejecución puede converger a un mínimo local diferente, y al comparar los valores finales se elige la mejor solución como candidato al mínimo global. Este enfoque es especialmente útil para funciones no convexas como la presente, ya que ayuda a evitar quedar atrapado en mínimos locales subóptimos. La efectividad del método depende tanto de la distribución de los puntos iniciales como de la capacidad del algoritmo local para converger rápidamente hacia un mínimo en la región donde se inicia el proceso.

## 2.3 Algoritmos de Optimización

Dado que la función

$$f(x, y) = x^4 - 4x^3 + 4x + y^2$$

no es convexa globalmente, es importante elegir algoritmos adecuados y considerar estrategias de múltiples puntos iniciales para buscar un mínimo global.

### 1. Método de Descenso por Gradiente (explicación detallada)

**Descripción corta.** El método de Gradiente es un método iterativo que actualiza la variable en la dirección de máximo descenso local:

$$x_{k+1} = x_k - \alpha \nabla f(x_k),$$

donde  $\alpha > 0$  es el *learning rate* (tasa de paso) y  $\nabla f(x_k)$  es el gradiente en  $x_k$ .

**Pseudocódigo:**

```
Elegir x0, tolerancia epsilon, paso alpha
Para k = 0,1,2,...:
    Calcular gradiente gk = grad(f)(xk)
    Si ||gk|| < epsilon: terminar
    Actualizar x_{k+1} = xk - alpha * gk
Fin
```

**Funcionamiento paso a paso.**

1. **Inicialización:** Se elige un punto inicial  $x_0$ , una tolerancia  $\varepsilon$  y un número máximo de iteraciones  $K_{\max}$ . También se selecciona un valor inicial del learning rate  $\alpha$ , pero antes de fijarlo se realizan pruebas con varios valores para determinar cuál es el más adecuado en cada región del dominio.
2. **Cálculo del gradiente:** En cada iteración  $k$  se calcula el gradiente  $g_k = \nabla f(x_k)$ . El gradiente señala la dirección de *máximo incremento* de la función. Por tanto, su opuesto  $-g_k$  indica la dirección de *máximo descenso local*, que es la dirección óptima para reducir el valor de la función.
3. **Actualización:** El punto se desplaza en la dirección de descenso mediante

$$x_{k+1} = x_k - \alpha g_k.$$

Este paso ajusta la posición según la magnitud del gradiente y el valor elegido de  $\alpha$ , controlando qué tan grande es el avance en la iteración.

4. **Criterio de parada:** Se detiene el algoritmo cuando se cumple alguna de las siguientes condiciones:

- **Cambio pequeño entre iteraciones:**

$$\|x_{k+1} - x_k\| < \varepsilon.$$

Esto indica que el algoritmo ha entrado en una zona donde la mejora por iteración es insignificante (estancamiento). Este criterio es útil incluso cuando el gradiente no disminuye estrictamente, pero las actualizaciones se vuelven muy pequeñas, señalando convergencia práctica.

- **Tope de iteraciones  $K_{\max}$ :** Garantiza seguridad computacional, evitando ciclos infinitos en casos de divergencia o estancamiento. Además, permite comparar el costo computacional entre configuraciones distintas del algoritmo.

5. **Iterar:** El proceso se repite hasta cumplir alguno de los criterios de parada anteriores.

### Influencia de cada parámetro:

- **Learning rate  $\alpha$ :** controla el tamaño del paso en cada iteración.
  - $\alpha$  **muy pequeño:** garantiza estabilidad y convergencia segura, pero requiere muchas iteraciones.
  - $\alpha$  **intermedio:** proporciona un equilibrio entre velocidad y estabilidad.
  - $\alpha$  **grande:** puede producir oscilaciones, saltarse el mínimo o incluso divergir, especialmente en regiones con alta curvatura.
- **Tolerancia  $\varepsilon$ :** define el nivel de precisión. Valores más pequeños exigen mayores iteraciones.
- **Máximo de iteraciones  $K_{\max}$ :** evita ciclos indefinidos y permite controlar el costo computacional.
- **Punto inicial  $x_0$ :** en funciones no convexas determina a qué mínimo local converge el algoritmo; por ello se evalúan puntos iniciales en distintas regiones.

### Protocolo experimental para seleccionar el mejor learning rate

Antes de ejecutar el algoritmo de descenso por gradiente de manera definitiva, se realizan pruebas en cada región del espacio utilizando múltiples puntos iniciales aleatorios. Cada uno de estos puntos se evalúa con distintos valores de learning rate, con el objetivo de identificar qué  $\alpha$  converge con mayor estabilidad, comparar velocidad (número de iteraciones), evitar valores que generen divergencias, determinar el valor más adecuado de  $\alpha$  para cada región.

Cada ejecución produce un registro que se almacena en el archivo `resultados_gradiente.csv`, el cual contiene la siguiente información:

Columna	Descripción	Utilidad
<code>x0</code>	Punto inicial $[x, y]$ desde donde comienza la búsqueda	Permite clasificar la región: $x > 2$ , $x < 0$ , $0 < x < 2$
<code>lr</code>	Learning rate usado en la ejecución	Comparar cuál LR funciona mejor en cada región
<code>x_final</code>	Punto obtenido al finalizar las iteraciones	Determinar el punto de convergencia
<code>f_final</code>	Valor de $f(x)$ en el punto final	Medir cuán bueno es el resultado
<code>num_iter</code>	Iteraciones necesarias para converger	Evaluar velocidad y estabilidad del algoritmo

Table 1: Contenido del archivo `resultados_gradiente.csv`

Este archivo se utiliza posteriormente para analizar de forma sistemática cuál learning rate es el más eficiente y estable en cada región del espacio.

**Objetivo.** Encontrar, por cada región del espacio ( $x > 2$ ,  $x < 0$ ,  $0 < x < 2$ ), qué valores de  $\alpha$  dan el mejor balance entre *estabilidad* y *rapidez* para el Descenso por Gradiente en la función estudiada.

## Procedimiento experimental recomendado (paso a paso).

1. **Definir regiones y puntos iniciales:** elegir varios puntos iniciales por región (ej.: 5 aleatorios por región) y además forzar algunos con  $y = 0$  si quieres observar el efecto en la componente  $y$ .
2. **Elegir candidatos de LR:** formar una lista de  $\alpha$  candidatos  $\{0.001, 0.005, 0.01, 0.05, 0.1\}$ .
  - **Empieza pequeño:** la lista incluye 0.001 y 0.005 para cubrir estabilidad fuerte.
  - **Incluye valores típicos:** 0.01 suele funcionar bien en funciones suaves y sirve como referencia.
  - **Explora valores más grandes:** 0.05 y 0.1 para detectar si es posible acelerar; si provocan oscilación se descartan.
  - **Motivación teórica:** la presencia de términos de cuarto grado ( $x^4$ ) implica que la curvatura puede crecer rápido con  $x$ , por eso no conviene empezar con LR demasiado grande.
3. **Para cada región y cada punto inicial:** ejecutar el descenso por gradiente para cada  $\alpha$  y guardar las métricas.

## Reglas prácticas para seleccionar el LR inicial a probar.

### Mejoras y alternativas (si quieres ir más allá)

- **Búsqueda en línea (line-search):** en vez de un LR fijo, usar una búsqueda de paso (Armijo/Wolfe) que garantiza reducción suficiente. Muy recomendado si buscas estabilidad.
- **LR adaptativos:** métodos como Adagrad/Adam/RMSProp (más comunes en ML) adaptan pasos por componente y pueden ayudar cuando las escalas en  $x$  y  $y$  difieren.
- **Escalado (preconditioning):** normalizar variables o usar una matriz de preconditionamiento para mitigar efectos de curvaturas muy distintas.

## 2. Método de Newton

**Descripción:** El método de Newton utiliza información de segunda derivada (Hessiano) para actualizar la variable:

$$x_{k+1} = x_k - H_k^{-1} \nabla f(x_k)$$

donde  $H_k = \nabla^2 f(x_k)$ .

### Recomendaciones para este problema:

- Converge rápidamente en regiones donde la Hessiana es positiva definida.
- Puede fallar o divergir en regiones donde la Hessiana tiene eigenvalores negativos ( $0 < x < 2$ ).
- Requiere calcular el Hessiano e invertirlo, por lo que es más costoso computacionalmente.

### Pseudocódigo:

```
Elegir x0, tolerancia epsilon
Para k = 0,1,2,...:
    Calcular gradiente gk = grad(f)(xk)
    Si ||gk|| < epsilon: terminar
    Calcular Hessiano Hk = Hess(f)(xk)
    Resolver Hk * pk = gk
    Actualizar x_{k+1} = xk - pk
Fin
```

### 3. Método Quasi-Newton (BFGS)

**Descripción:** Quasi-Newton aproxima el Hessiano inverso iterativamente sin calcularlo explícitamente:

$$x_{k+1} = x_k - W_k \nabla f(x_k)$$

donde  $W_k \approx H_k^{-1}$  se actualiza en cada iteración usando la fórmula BFGS.

#### Recomendaciones para este problema:

- Combina la velocidad de Newton con la simplicidad del Gradiente.
- Funciona bien localmente en regiones de curvatura positiva y es más estable que Newton si se parte de un punto alejado.
- También se recomienda usar Multi-start para aumentar la probabilidad de encontrar mínimos globales.

### Pseudocódigo:

```
Elegir x0, tolerancia epsilon
Inicializar W0 = identidad
Para k = 0,1,2,...:
    Calcular gradiente gk = grad(f)(xk)
    Si ||gk|| < epsilon: terminar
    Calcular pk = Wk * gk
    Actualizar x_{k+1} = xk - pk
    Actualizar Wk+1 usando fórmula BFGS
Fin
```

## 3 Comparación de Resultados y Graficación

### 3.1 Experimentos

Se realizaron experimentos utilizando los tres algoritmos: Descenso por Gradiente, Newton y Quasi-Newton (BFGS) sobre la función

$$f(x, y) = x^4 - 4x^3 + 4x + y^2$$

partiendo de múltiples puntos iniciales (**Multi-start**) para explorar diferentes regiones de la función y evaluar la convergencia local y global.

### 3.2 Criterios de Comparación

Los resultados se comparan considerando:

- Número de iteraciones hasta convergencia.
- Tiempo de cómputo (si se dispone).
- Valor final de la función  $f(x, y)$ .
- Sensibilidad al punto inicial.
- Trayectorias de iteración en el plano  $(x, y)$ .

### 3.3 Observaciones

- **Descenso por Gradiente:** converge más lentamente, especialmente si el punto inicial está cerca de regiones de curvatura plana o mínima local no global.
- **Método de Newton:** converge muy rápido cerca de mínimos locales con Hessiana positiva definida, pero puede fallar si el Hessiano es indefinido.
- **Quasi-Newton (BFGS):** balancea velocidad y estabilidad, evitando el cálculo directo del Hessiano, con convergencia superlineal en regiones locales.
- **Multi-start:** aumenta significativamente la probabilidad de encontrar el mínimo global, explorando varias regiones de la función no convexa.

### 3.4 Graficación de la Función y Trayectorias

Para visualizar la convergencia de los algoritmos se pueden utilizar gráficos de contorno de la función y superponer la trayectoria de iteraciones de cada algoritmo:

**Pseudocódigo para graficar trayectorias:**

```
Crear una malla de puntos (X,Y) para evaluar f(X,Y)
Calcular Z = f(X,Y)
Dibujar contornos de Z
Para cada algoritmo:
    Obtener historial de puntos iterativos [x_k, y_k]
    Trazar línea conectando los puntos del historial
Añadir leyenda y etiquetas
```

### 3.5 Resultados Esperados

- Las trayectorias muestran cómo cada algoritmo se aproxima a diferentes mínimos locales.
- Multi-start permite que al menos uno de los algoritmos alcance el mínimo global.
- Descenso por Gradiente puede requerir más iteraciones que Newton o BFGS.

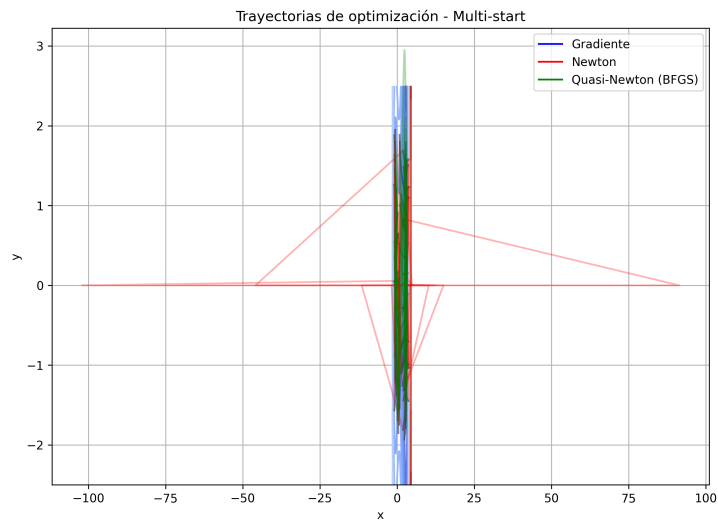


Figure 1: Contorno de  $f(x, y)$  con trayectorias de los algoritmos desde múltiples puntos iniciales.

### 3.6 Figura Ejemplo

## 4 Conclusiones

- El método de Newton converge más rápido localmente, pero requiere cálculo del Hessiano.
- El descenso por gradiente es más simple, pero puede necesitar más pasos.
- Quasi-Newton (BFGS) combina velocidad y menor cálculo de derivadas.