



UNIVERSIDAD NACIONAL DE COLOMBIA

FACULTAD DE CIENCIAS - DEPARTAMENTO DE MATEMÁTICAS

OPTIMIZACIÓN NUMÉRICA - 2024 II

Murcia Porras, Juan Diego	jmurciap@unal.edu.co
Nosa Guzman, Carlos Enrique	cnosa@unal.edu.co
Acosta Ruiz, Jhon Deivid	jhacostar@unal.edu.co

Parcial 3

1. Introducción

En la optimización no lineal es necesario encontrar valores óptimos para funciones que pueden presentar múltiples mínimos locales y características complejas. Dentro de los métodos más utilizados para resolver estos problemas se encuentra el método de Newton, el cual, debido a su rapidez de convergencia cuadrática, resulta particularmente eficiente en la búsqueda de soluciones óptimas. Sin embargo, la aplicabilidad directa de este método puede verse limitada cuando la matriz Hessiana no es definida positiva o cuando el punto de inicio se encuentra *lejos* del óptimo. Para abordar estas dificultades, se han desarrollado estrategias de globalización del método de Newton, como la búsqueda lineal y los métodos de región de confianza. La búsqueda lineal ajusta el tamaño del paso en cada iteración para garantizar una reducción adecuada en la función objetivo, mientras que los métodos de región de confianza restringen la actualización del punto de iteración a una vecindad en la que el modelo local sea una buena aproximación de la función.

El presente trabajo tiene como objetivo implementar el método de Newton globalizado para resolver problemas de optimización no lineal, comparando diferentes estrategias de búsqueda lineal y métodos de región de confianza. Se considera, en particular, el problema de optimización de Rosenbrock en \mathbb{R}^2 , una función ampliamente utilizada como prueba en la literatura debido a su valle estrecho y curvatura pronunciada, lo que representa un desafío para los métodos de optimización (consultar [4]). A través de esta comparación, se busca evaluar el desempeño de cada enfoque y analizar sus ventajas y limitaciones en la búsqueda de la solución óptima.

En primer lugar, en el presente documento se plantea el problema y se resuelve de manera analítica y además se estudian ciertas características de la función objetivo del problema. Después de esto se continúa con la explicación formal de los dos métodos de globalización del método de Newton y también se proporciona un pseudocódigo que indica la implementación de cada

algoritmo. Posteriormente se procede hacer comparaciones de los métodos implementados en términos de convergencia y costo. Finalmente se realizan conclusiones de los resultados obtenidos y adicionalmente se proporciona el código usado en este trabajo.

1.1. Planteamiento del problema

Se considera el problema de optimización

$$\min_{x \in \mathbb{R}^2} f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2. \quad (1)$$

La función $f(x_1, x_2)$ es la conocida función de Rosenbrock, la cual es ampliamente utilizada como un caso de prueba para algoritmos de optimización.

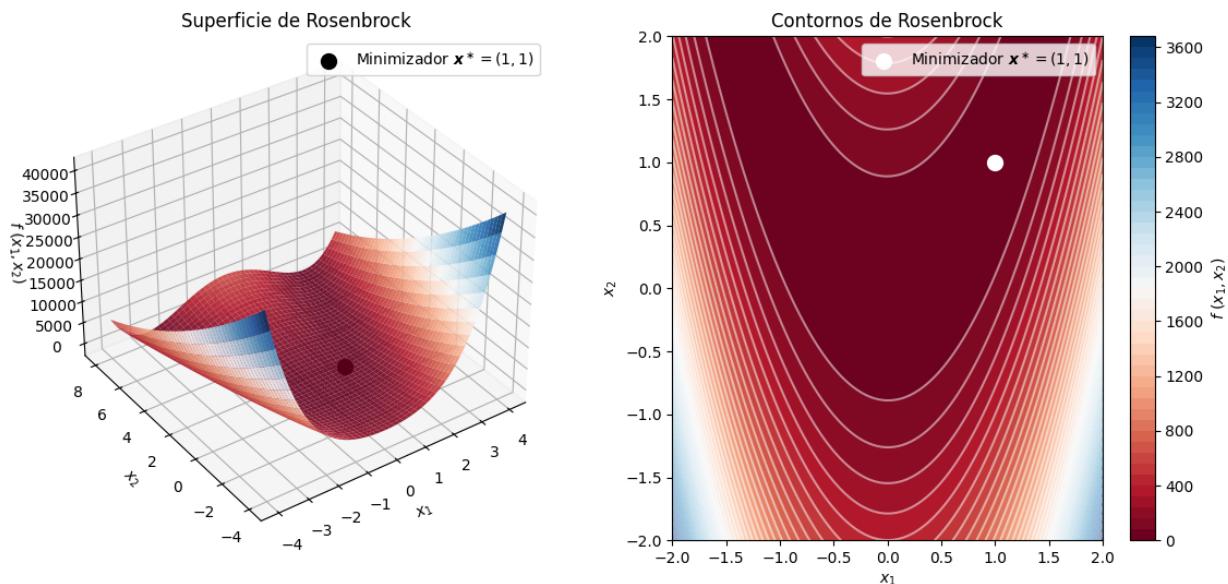


Figura 1: A la izquierda se muestra la representación tridimensional de la función de Rosenbrock sobre el conjunto $[-4, 4] \times [-4, 8]$, indicando el minimizador global. A la derecha se grafican los contornos de la función de Rosenbrock sobre $[-2, 2] \times [-2, 2]$, resaltando el minimizador \mathbf{x}^* . Realización propia usando Python.

Se observa que la función es una suma de dos términos cuadrados, lo que implica que siempre es no negativa. Su valor mínimo es cero y ocurre cuando ambos términos se anulan simultáneamente.

- Para que el primer término sea cero, se debe cumplir que

$$x_2 - x_1^2 = 0 \quad \Rightarrow \quad x_2 = x_1^2.$$

- Para que el segundo término sea cero, se debe cumplir que

$$1 - x_1 = 0 \quad \Rightarrow \quad x_1 = 1.$$

Sustituyendo $x_1 = 1$ en la ecuación $x_2 = x_1^2$, se obtiene $x_2 = 1^2 = 1$. Por lo tanto, el único punto que anula la función es $(x_1^*, x_2^*) = (1, 1)$, lo que implica que este es el único minimizador. Como la función es una suma de términos cuadrados, su mínimo global es $p^* := 0$ y ocurre en el punto $\mathbf{x}^* := (1, 1)$.

En la Figura 1 se muestran dos visualizaciones de la función de Rosenbrock, una como superficie en el espacio tridimensional y otra mediante contornos. La gráfica de la izquierda es una representación tridimensional de la función, que ilustra un valle estrecho y curvo con un mínimo global ubicado en $\mathbf{x}^* = (1, 1)$. La gráfica de la derecha presenta los contornos de nivel de la función, mostrando el mismo mínimo global como un punto blanco dentro de un conjunto de contornos concéntricos. Ambas visualizaciones destacan la complejidad de la función de Rosenbrock, particularmente su forma de valle largo y estrecho que dificulta la convergencia hacia el mínimo global para los algoritmos de optimización.

Haciendo uso del enfoque diferencial del problema de optimización, se realiza una descripción del gradiente y de la hessiana de la función objetivo. En efecto, el gradiente de la función $f(x_1, x_2)$ está dado por:

$$\nabla f(x_1, x_2) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix} = \begin{bmatrix} -400x_1(x_2 - x_1^2) - 2(1 - x_1) \\ 200(x_2 - x_1^2) \end{bmatrix}.$$

Para encontrar los puntos críticos, se resuelve $\nabla f(x_1, x_2) = 0$, lo que da lugar al sistema:

$$\begin{cases} -400x_1(x_2 - x_1^2) - 2(1 - x_1) = 0, \\ 200(x_2 - x_1^2) = 0. \end{cases}$$

De la segunda ecuación, se obtiene nuevamente que $x_2 = x_1^2$, y sustituyendo en la primera ecuación:

$$-400x_1(x_1^2 - x_1^2) - 2(1 - x_1) = 0 \quad \Rightarrow \quad x_1 = 1.$$

Por lo tanto, $\mathbf{x}^* = (1, 1)$ es el único punto crítico de la función.

La Figura 2 visualiza el campo vectorial del gradiente de la función de Rosenbrock, mostrando la dirección de máximo ascenso en cada punto. A la izquierda, se observa el campo en un rango amplio, donde las flechas (en rojo y azul) indican la dirección y magnitud del gradiente, convergiendo hacia el mínimo global. A la derecha, se amplía la vista cerca del mínimo $\mathbf{x}^* = (1, 1)$,

mostrando con mayor detalle la convergencia del gradiente en esa región. En conjunto, las gráficas ilustran cómo el gradiente guía la optimización hacia el mínimo de la función, pero también la variabilidad de su magnitud, lo que puede influir en la eficiencia de los algoritmos.

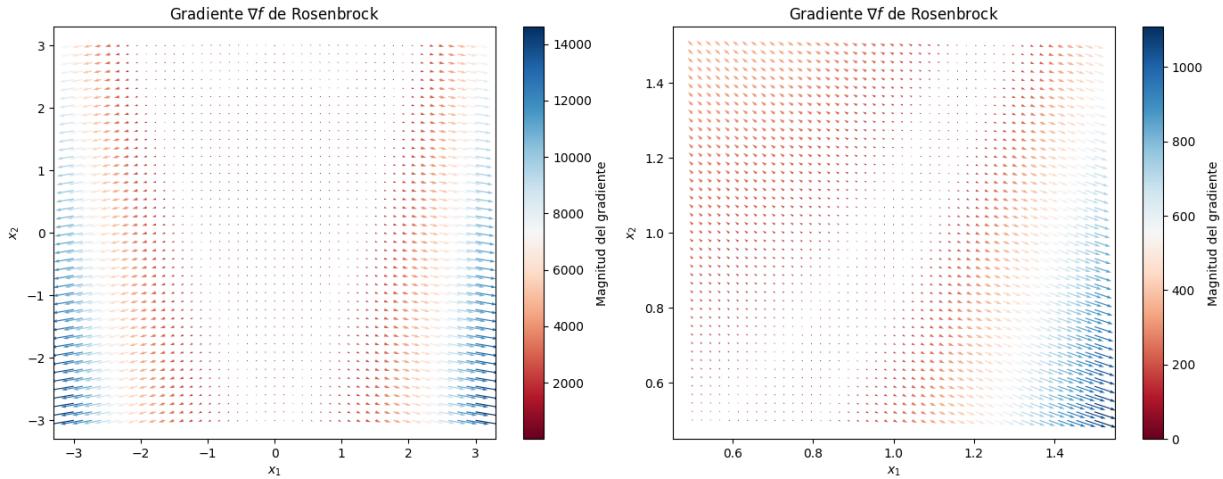


Figura 2: A la izquierda se muestra el campo vectorial dado por el gradiente ∇f en el rectángulo $[-3, 3] \times [-3, 3]$. A la derecha se grafica el mismo campo vectorial cerca del minimizador global. Realización propia.

La Hessiana de la función $f(x_1, x_2)$ está dada por:

$$H(x_1, x_2) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{bmatrix} = \begin{bmatrix} 1200x_1^2 - 400x_2 + 2 & -400x_1 \\ -400x_1 & 200 \end{bmatrix}.$$

Para determinar la naturaleza del punto crítico $\mathbf{x}^* = (1, 1)$, se evalúa la Hessiana en este punto:

$$H(\mathbf{x}^*) = H(1, 1) = \begin{bmatrix} 1200(1)^2 - 400(1) + 2 & -400(1) \\ -400(1) & 200 \end{bmatrix} = \begin{bmatrix} 802 & -400 \\ -400 & 200 \end{bmatrix}.$$

Los valores propios de esta matriz determinan la convexidad local de la función. Como ambos valores son positivos puesto que la matriz tiene sus menores principales positivos ($H(\mathbf{x}^*)_{1,1} = 802 > 0$ y $\det(H(\mathbf{x}^*)) = 400 > 0$), la Hessiana es definida positiva en $\mathbf{x}^* = (1, 1)$, lo que confirma que es un mínimo local (y global debido a la estructura de la función).

De manera general, la matriz $\nabla^2 f(x_1, x_2)$ es definida positiva si y solamente si

$$\frac{\partial^2}{\partial x_1^2} f(x_1, x_2) = 1200x_1^2 - 400x_2 + 2 > 0,$$

y,

$$\det(\nabla^2 f(x_1, x_2)) = 80000x_1^2 - 80000x_2 + 400 > 0,$$

estas condiciones se satisfacen en el conjunto

$$S = \left\{ (x_1, x_2) \in \mathbb{R}^2 \mid x_2 < x_1^2 + \frac{1}{200} \right\},$$

confirmando el último gráfico de la figura 3 en el cual se muestra numéricamente en qué región del espacio la Hessiana de la función de Rosenbrock es definida positiva.

La Figura 3 presenta tres gráficas relacionadas con la Hessiana de la función de Rosenbrock. A la izquierda, se muestra un mapa de calor del primer valor propio de la Hessiana, variando de valores bajos en azul a altos en rojo, con un rango numérico asociado. En el centro, se presenta un mapa de calor similar para el segundo valor propio, mostrando una distribución diferente de valores. A la derecha, se indica la región donde la Hessiana es definida positiva, diferenciando entre áreas donde lo es (gris claro) y donde no (gris oscuro), lo que es crucial para determinar la convexidad local de la función y la convergencia de métodos de optimización basados en el Hessiano.

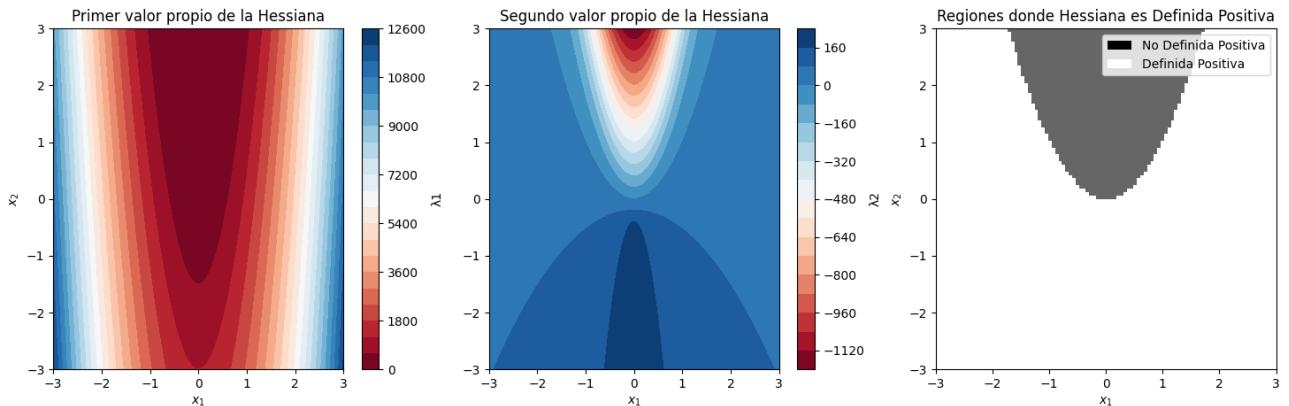


Figura 3: A la izquierda y en el centro se muestra el mapa de calor de los valores propios de la Hessiana de la función de Rosenbrock. A la derecha se indica la región donde la Hessiana es definida positiva de manera numérica. Realización propia.

2. Fundamentación

2.1. Búsqueda lineal

Los métodos de búsqueda lineal se basan en identificar, en cada iteración, una dirección de búsqueda p_k y determinar cuánto avanzar en esa dirección (tamaño de paso t_k) con el propósito de garantizar un decrecimiento eficiente de la función objetivo. El éxito de estos métodos depende de la efectividad simultánea en las elecciones tanto de la dirección p_k como del tamaño de paso t_k . La sección se fundamentará en el capítulo 3 de [5] y en el capítulo 11 de [6].

La iteración estará dada por

$$x_{k+1} = x_k + t_k p_k. \quad (2)$$

Se requiere que p_k sea dirección de descenso en x_k , esto es, $p_k^\top \nabla f(x_k) < 0$, pues esto garantiza que $f(x_k + tp_k) < f(x_k)$ para algún t positivo, y así, asumimos que el tamaño de paso satisface $t_k > 0$. La dirección de búsqueda generalmente tiene la forma $p_k = -B_k^{-1} \nabla f_k$ donde B_k es simétrica definida positiva. Note que si fijamos $t_k = 1$ y se elige $B_k = I_n$ nos encontramos en el Método del Descenso Más Pendiente, si elegimos $B_k = \nabla^2 f(x_k)$ estaremos en el Método de Newton, mientras que, si en cada paso calculamos B_k por una aproximación de bajo rango de la Hessiana estaremos en los Métodos Quasi-Newton.

Para garantizar que las direcciones p_k sean de descenso, las matrices B_k deben tomarse definidas positivas. Si B_k no es definida positiva, debe pasar a través de una transformación que la convierta en una nueva matriz similar que sí lo sea. A este procedimiento lo denominamos *regularización de B_k* . En este trabajo, aplicaremos, siguiendo [2], el método de regularización aditiva, que consiste en modificar B_k por $B_k + \tau I \succ 0$, donde $\tau \in \mathbb{R}^+$. El algoritmo se presenta a continuación.

RegularizarMatriz(B, τ_{warm})

Entrada: $B, \tau_{warm}, \beta = 10^{-3}$

Salida: $\tau^*, B^* \leftarrow B + \tau^* I$.

If $B \succ 0$: $(\tau^*, B^*) \leftarrow (0, B)$

Else:

$B \leftarrow B + \tau_{warm} I$

If $\min b_{ii} > 0$: $\tau = 0$

Else: $\tau \leftarrow -\min b_{ii} + \beta$

Mientras no se tenga $B + \tau I \succ 0$:

$\tau \leftarrow \max(2\tau, \beta)$

Fin mientras

$\tau^* \leftarrow \tau_{warm} + \tau$

$B^* = B + \tau^* I$

En el anterior algoritmo, es importante mencionar que τ_{warm} se toma como un valor inicial de τ basado en anteriores iteraciones del algoritmo. Por otro lado, verificar si una matriz es definida positiva suele hacerse intentando realizar su factorización de Cholesky.

Ahora, bien, después de regularizadas, todas estas elecciones del B_k se pueden realizar calculando un t_k en cada iteración sin necesariamente ser 1. Surge ahora la pregunta ¿Cómo elegimos t_k ? La técnica recibe el nombre de **Búsqueda Lineal** precisamente porque la exploración de un nuevo punto x_{k+1} se realiza a lo largo de la recta definida por $y(\alpha) = x_k + \alpha p_k$. Intuitivamente, querríamos elegir t_k como la solución del subproblema de optimización:

$$\min_{t > 0} f(x_k + t p_k). \quad (3)$$

Búsqueda Lineal

Entrada: $x_0, \epsilon > 0, f(x_k), \nabla f(x_k)$

Salida: x^*

$k \leftarrow 0$

Mientras $\|\nabla f(x_k)\| < \epsilon$ **hacer**

 Calcule B_k , por Newton o Quasi-Newton.

If $k = 1$: $\tau_{warm} = 0$ **Else**: $\tau_{warm} = \tau_{k-1}/2$ $(\tau_k, B_k) \leftarrow \text{RegularizarMatriz}(B_k, \tau_{warm})$

$p_k \leftarrow -B_k^{-1} \nabla f(x_k)$

 Determine $t_k \leftarrow \min_t f(x_k + t p_k)$

$x_{k+1} \leftarrow x_k + t_k p_k$

$k \leftarrow k + 1$

Fin mientras

$x^* \leftarrow x_k$

De esta manera al resolver (3) de manera exacta, obtendríamos el máximo beneficio de la dirección p_k , pero ello puede resultar muy costoso y, en la mayoría de los casos innecesario, por lo que en la práctica se acepta una solución aproximada. Sin embargo, esta elección no puede ser arbitraria; por ejemplo, si los pasos t_k son demasiado cortos, el algoritmo avanzaría muy lentamente hacia el óptimo. Por esta razón necesitamos imponer condiciones que nos ofrezcan el t_k más óptimo al menor costo.

Las **condiciones de Armijo y Wolfe** abordan este problema al establecer criterios adecuados para la elección de t_k . En la siguiente subsección, exploraremos cómo estas condiciones garantizan una búsqueda en línea efectiva y contribuyen a la convergencia del algoritmo.

2.1.1. Backtracking

El método de Backtracking es una estrategia sencilla y práctica para la elección de la longitud de paso t_k en la búsqueda lineal. Recordemos que el objetivo principal es garantizar que en cada iteración se obtenga una reducción significativa en el valor de la función.

Lo que se considera una reducción *significativa* se mide en términos de la serie de Taylor. Una aproximación lineal de $f(x_k + tp_k)$ se obtiene a partir de

$$f(x_k + tp_k) \approx f(x_k) + t \nabla f(x_k)^\top p_k.$$

Y dado que se requiere que la longitud de paso t_k produzca una reducción en f que sea al menos una fracción del descenso estimado por esta aproximación lineal, se impone la siguiente condición:

$$f(x_k + t_k p_k) \leq f(x_k) + \gamma t_k \nabla f(x_k)^\top p_k, \quad \text{con } \gamma \in (0, 1). \quad (4)$$

Esta desigualdad es conocida como la *condición de Armijo*. Así, sea p_k una dirección de búsqueda que satisface la condición de descenso suficiente, definimos t_k como el primer elemento de la secuencia:

$$1, \frac{1}{2}, \frac{1}{4}, \dots, 2^{-i}, \dots$$

que cumple (4). Esta estrategia de reducción progresiva del tamaño de paso es lo que da el nombre al método de **Backtracking**.

Backtracking

Entrada: $x_k, p_k, f(x_k), \nabla f(x_k)^\top, \gamma, \beta = \frac{1}{2}, t_{\max}$

Salida: t

$t \leftarrow t_{\max}$

Mientras $f(x_k + tp_k) > f(x_k) + \gamma t \nabla f(x_k)^\top p_k$ **hacer**

$t \leftarrow \beta t$

Fin mientras

2.1.2. Método de Wolfe

El método de Wolfe consiste en imponer dos condiciones a la longitud de paso t_k . La primera condición de Wolfe es exactamente la condición de Armijo, y, como ya se mencionó, garantiza que haya una reducción significativa (en el sentido que implica convergencia) de la función objetivo, y está dada por

$$f(x_k + t_k p_k) \leq f(x_k) + c_1 t_k \nabla f(x_k)^\top p_k, \quad \text{con } c_1 \in (0, 1). \quad (5)$$

Esta primera condición se suele denominar la *condición de suficiente decrecimiento*, e, intuitivamente, garantiza que el paso no sea demasiado largo. La segunda condición de Wolfe busca garantizar que haya una disminución significativa del gradiente al avanzar una longitud t_k , en términos de que haya una reducción relativa de la derivada direccional menor a una constante $c_2 \in (0, 1)$. Es decir,

$$\frac{\nabla f(x_k + t_k p_k)^T p_k}{\nabla f(x_k)^T p_k} \leq c_2, \text{ con } c_2 \in (0, 1). \quad (6)$$

O, dicho de otra forma, al ser p_k una dirección de descenso,

$$\nabla f(x_k + t_k p_k)^T p_k \geq c_2 \nabla f(x_k)^T p_k, \text{ con } c_2 \in (0, 1). \quad (7)$$

Esta segunda condición se suele denominar la *condición de curvatura*, e, intuitivamente, garantiza que el paso no sea demasiado corto. En realidad, la condición (6) no garantiza una disminución significativa del gradiente, pues puede que la derivada $\nabla f(x_k + t_k p_k)^T p_k$ sea positiva. Para que esto se garantice, se debe exigir esta desigualdad con un valor absoluto en el término a la izquierda de la desigualdad, o, dicho de otra forma,

$$|\nabla f(x_k + t_k p_k)^T p_k| \leq -c_2 \nabla f(x_k)^T p_k, \text{ con } c_2 \in (0, 1). \quad (8)$$

Cuando buscamos que se cumplan las condiciones (5) y (7), decimos que estamos realizando una búsqueda con las *condiciones débiles de Wolfe*. Cuando buscamos garantizar (5) y (8), decimos que utilizamos las *condiciones fuertes de Wolfe*. Se puede demostrar que si $0 < c_1 < c_2 < 1$, entonces siempre existe un punto que satisface las condiciones fuertes de Wolfe, y, por ende, también satisface las condiciones débiles de Wolfe. En la práctica, se suele tomar $c_1 \approx 10^{-4}$ y $c_2 \approx 0.9$. El siguiente código realiza la búsqueda de una longitud de paso en la que se satisfagan las condiciones débiles de Wolfe. Este algoritmo replica el método de bisección, y reduce el intervalo de búsqueda a $[t_{\min}, t_{\max}]$, donde t_{\min} es un paso “demasiado corto” (es decir, satisface (5), pero no (7)), y t_{\max} es un paso “demasiado largo” (es decir, no cumple (5)).

Condiciones débiles de Wolfe

Entrada: $x_k, p_k, f(x_k), \nabla f(x_k)^\top, c_1 = 10^{-4}, c_2 = 0,9, \lambda = 2, t_0$

Salida: t^* , que cumple las condiciones débiles de Wolfe.

$t_{min}, t_{max} = 0, +\infty$

Para $j = 0, 1, \dots$:

If t_j **no cumple** (5):

$t_{max} \leftarrow t$

$t_{j+1} \leftarrow (t_{min} + t_{max})/2$

Elif t_j **no cumple** (7),

$t_{min} \leftarrow t_k$

If $t_{max} < \infty$:

$t_{j+1} \leftarrow (t_{min} + t_{max})/2$

Else:

$t_{j+1} \leftarrow \lambda t_j$

Else:

$t^* \leftarrow t_j$

Romper iterador

Fin iterador

Ahora presentaremos un código que busca encontrar un valor que satisfaga las condiciones fuertes de Wolfe. Este código se basa en lo presentado dentro de [2]. Por facilidad en la notación, denotaremos por

$$\phi(t) := f(x_k + tp_k).$$

Para este caso, requerimos de una función auxiliar que llamaremos $\text{zoom}(t_{low}, t_{high})$, donde

- (a) El intervalo entre t_{low} y t_{high} contiene longitudes de paso t que satisfacen las condiciones fuertes de Wolfe.
- (b) t_{low} es, entre todas las longitudes de paso generadas hasta el momento, aquella que satisface (5) con el menor valor de ϕ encontrado.
- (c) t_{high} es escogido de manera que $\phi'(t_{low})(t_{high} - t_{low}) < 0$.

Note que no necesariamente se cumple $t_{low} < t_{high}$. La idea de este algoritmo es realizar bisección sobre este intervalo, e ir actualizando los extremos de tal manera que se sigan cumpliendo las condiciones (a),(b) y (c). Si los valores iniciales de t_{low} y t_{high} satisfacen estas condiciones, entonces zoom permite encontrar un valor t^* que satisface las condiciones fuertes de Wolfe.

zoom: Algoritmo auxiliar

Entrada: $t_{low}, t_{high}, x_k, p_k, f(x_k), \nabla f(x_k)^\top, c_1 = 10^{-4}, c_2 = 0,9$

Salida: t^* , que cumple las condiciones fuertes de Wolfe.

Para $j = 1, 2, \dots$

$$t_j = (t_{low} + t_{high})/2$$

If t_j no cumple (5) o $\phi(t_j) > \phi(t_{low})$:

$$t_{high} \leftarrow t_j$$

Elif t_j no cumple (8):

$$\text{If } \phi'(t_j)(t_{high} - t_{low}) \geq 0:$$

$$t_{high} \leftarrow t_{low}$$

$$t_{low} \leftarrow t_j$$

Else:

$$t^* \leftarrow t_j$$

Romper iterador

Fin iterador

Con esta función auxiliar, basta con encontrar valores de t_{low} y t_{high} que inicializan `zoom`. El siguiente algoritmo encuentra tales valores.

Condiciones fuertes de Wolfe

Entrada: $x_k, p_k, f(x_k), \nabla f(x_k)^\top, c_1 = 10^{-4}, c_2 = 0,9, \lambda = 2, t_1$

Salida: t^* , que cumple las condiciones fuertes de Wolfe.

$$t_0 = 0$$

Para $i = 1, 2, \dots$

If t_i no cumple (5) o $(\phi(t_i) > \phi(t_{i-1}) \text{ y } i \neq 1)$:

$$t^* \leftarrow \text{zoom}(t_{i-1}, t_i)$$

Romper iterador

Elif t_i cumple (8):

$$t^* \leftarrow t_i$$

Romper iterador

Elif $\phi'(t_i) \geq 0$:

$$t^* \leftarrow \text{zoom}(t_i, t_{i-1})$$

Romper iterador

Else: $t_{i+1} \leftarrow \lambda t_i$

Fin iterador

2.2. Región de confianza

Los métodos de región de confianza y los métodos de búsqueda en línea comparten la idea de construir un modelo cuadrático de la función objetivo para generar los pasos de optimización, pero lo utilizan de manera diferente. Mientras que los métodos de búsqueda en línea determinan

primero una dirección de búsqueda y luego buscan una longitud de paso adecuada, los métodos de región de confianza establecen una región alrededor del punto actual en la que el modelo es considerado una representación confiable de la función objetivo. En cada iteración, se busca minimizar este modelo dentro de la región, determinando simultáneamente la dirección y la longitud del paso. Si el paso no resulta adecuado, la región de confianza se reduce y se intenta nuevamente. El tamaño de esta región es un factor crucial: si es demasiado pequeña, el algoritmo avanza lentamente y desaprovecha oportunidades de progreso; si es demasiado grande, el modelo puede volverse inexacto, requiriendo ajustes adicionales. En la práctica, el tamaño de la región se adapta dinámicamente en función del desempeño del algoritmo en iteraciones previas: si el modelo predice bien el comportamiento de la función, la región se expande para permitir pasos más largos; si un paso falla, se reduce para mejorar la precisión de la aproximación. Esta sección está inspirada en el capítulo 4 de [3].

Supondremos que la función modelo m_k utilizada en cada iteración x_k es de tipo cuadrática. Además, m_k se construye a partir del desarrollo en serie de Taylor de f alrededor de x_k , expresado como:

$$f(x_k + p) = f_k + g_k^T p + \frac{1}{2} p^T \nabla^2 f(x_k + tp)p, \quad (9)$$

donde $f_k = f(x_k)$, $g_k = \nabla f(x_k)$ y t es un escalar dentro del intervalo $(0, 1)$. Al utilizar una aproximación B_k de la Hessiana en el término de segundo orden, la función modelo m_k se define como:

$$m_k(p) = f_k + g_k^T p + \frac{1}{2} p^T B_k p, \quad (10)$$

donde B_k es una matriz simétrica. La diferencia entre $m_k(p)$ y $f(x_k + p)$ es del orden de $O(\|p\|^2)$, lo cual es pequeño cuando $\|p\|$ es pequeño.

Si B_k coincide con la Hessiana real $\nabla^2 f(x_k)$, el error de aproximación en la función modelo m_k es del orden de $O(\|p\|^3)$, lo que hace que este modelo sea especialmente preciso cuando $\|p\|$ es pequeño. Esta elección, $B_k = \nabla^2 f(x_k)$, conduce al método de Newton de región de confianza. Para determinar cada paso, se busca resolver el siguiente subproblema:

$$\min_{p \in \mathbb{R}^n} m_k(p) = f_k + g_k^T p + \frac{1}{2} p^T B_k p \quad \text{sujeto a } \|p\| \leq \Delta_k, \quad (11)$$

donde $\Delta_k > 0$ representa el radio de la región de confianza. En la mayoría de los casos, se define $\|\cdot\|$ como la norma euclídea, por lo que la solución p_k^* de (4.3) es el minimizador de m_k dentro de la bola de radio Δ_k . Así, el método de región de confianza requiere resolver una secuencia de subproblemas (11), donde la función objetivo y la restricción (expresada como $p^T p \leq \Delta_k^2$) son cuadráticas.

Cuando B_k es definida positiva y se cumple que $\|B_k^{-1}g_k\| \leq \Delta_k$, la solución de (11) es sencilla: corresponde al mínimo sin restricciones $p_k^B = -B_k^{-1}g_k$ de la función cuadrática $m_k(p)$. En este caso, p_k^B se denomina el paso completo. En otros casos, la solución de (11) no es tan evidente, pero generalmente puede determinarse sin un costo computacional excesivo. En cualquier situación, como se explica a continuación, solo se necesita una solución aproximada para garantizar la convergencia y un buen comportamiento en la práctica.

Definición 2.1 Dado un paso p_k se define la **tasa de aceptación** ρ_k como

$$\rho_k = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)}. \quad (12)$$

Región de Confianza

Entrada: $\Delta_{\max} > 0$, $\Delta_0 \in (0, \Delta_{\max})$, $\eta \in (0, \frac{1}{4})$

Para $k = 0, 1, 2, \dots$ **hacer**

Obtener p_k resolviendo (aproximadamente) el subproblema (11)

Evaluar la tasa de aceptación ρ_k

Si $\rho_k < \frac{1}{4}$ **entonces**

$$\Delta_{k+1} = \frac{1}{4}\Delta_k$$

Si no, si $\rho_k > \frac{3}{4}$ y $\|p_k\| = \Delta_k$ **entonces**

$$\Delta_{k+1} = \min(2\Delta_k, \Delta_{\max})$$

Si no

$$\Delta_{k+1} = \Delta_k$$

Fin si

Si $\rho_k > \eta$ **entonces**

$$x_{k+1} = x_k + p_k$$

Si no

$$x_{k+1} = x_k$$

Fin si

Fin mientras

2.2.1. Región de confianza con modelo de Cauchy

El punto de Cauchy es una aproximación computacionalmente eficiente dentro de los métodos de región de confianza, utilizada para garantizar la convergencia global sin necesidad de resolver exactamente el subproblema cuadrático asociado. Aunque idealmente se busca el óptimo dentro de la región de confianza, en la práctica es suficiente encontrar un paso p_k que proporcione una reducción adecuada en el modelo. El punto de Cauchy, denotado como p_k^C , se obtiene al minimizar el modelo cuadrático a lo largo de la dirección del gradiente negativo, respetando la restricción impuesta por la región de confianza. Este enfoque permite obtener pasos de descenso confiables con menor costo computacional, asegurando que el algoritmo siga avanzando incluso cuando

una solución exacta del subproblema sería demasiado costosa de calcular. A continuación se presenta el pseudocódigo del cálculo del punto de Cauchy el cual tiene como objetivo solucionar de manera aproximada el subproblema (11), después de esto se discutirán los elementos que componen y sustentan este algoritmo.

Cálculo del punto de Cauchy

Entrada: $x_k, \nabla f(x_k), B_k, \Delta_k$

Paso 1: Calcular la dirección de descenso más pronunciado:

$$p_k^S = \arg \min_{p \in \mathbb{R}^n} (f_k + g_k^\top p) \quad \text{s.a.} \quad \|p\| \leq \Delta_k$$

Paso 2: Determinar el escalar τ_k que minimiza $m_k(\tau p_k^S)$ sujeto a la restricción de región de confianza:

$$\tau_k = \arg \min_{\tau \geq 0} m_k(\tau p_k^S) \quad \text{s.a.} \quad \|\tau p_k^S\| \leq \Delta_k$$

Paso 3: Calcular el punto de Cauchy: $p_k^C = \tau_k p_k^S$

Salida: p_k^C

En el primer paso del punto de Cauchy se tiene como función objetivo una función lineal en un dominio restringido por lo tanto la solución se encuentra en la frontera de tal dominio restringido, además, al ser la función objetivo lineal se tiene que la dirección de mayor descenso es opuesta al vector g_k , esto implica que

$$p_k^S = -\frac{\Delta_k}{\|g_k\|} g_k$$

de esta manera $\|p_k^S\| = \Delta_k$ y la dirección de p_k^S es opuesta a g_k . En el segundo paso se resuelve otro problema de minimización para hallar el tamaño de paso adecuado dentro de la región de confianza, este problema tiene como objetivo hallar $\tau \geq 0$ que minimice $m_k(\tau p_k^S)$ sujeto a la restricción de confianza $\|\tau p_k^S\| \leq \Delta_k$; la función objetivo solo depende de τ y se puede representar como

$$\begin{aligned} \mathcal{T}_k(\tau) := m_k(\tau p_k^S) &= f_k + \tau g_k^\top p_k^S + \frac{\tau^2}{2} p_k^{S\top} B_k p_k^S \\ &= f_k - \tau \Delta_k \|g_k\| + \tau^2 \frac{\Delta_k^2}{2\|g_k\|^2} g_k^\top B_k g_k. \end{aligned}$$

Por otra parte, la restricción $\|\tau p_k^S\| \leq \Delta_k$ es equivalente a tener $\tau \leq 1$ dado que $\|p_k^S\| = \Delta_k$. La función \mathcal{T}_k es una función cuadrática en τ y por ende es sencillo calcular el punto crítico

derivando e igualando a cero:

$$\frac{d}{d\tau} \mathcal{T}_k(\bar{\tau}) = -\Delta_k \|g_k\| + \bar{\tau} \frac{\Delta_k^2}{\|g_k\|^2} g_k^\top B_k g_k = 0 \quad \Rightarrow \quad \bar{\tau} = \frac{\|g_k\|^3}{\Delta_k g_k^\top B_k g_k}$$

No obstante el punto crítico no necesariamente es la solución al problema del paso 2 en el algoritmo de cálculo del punto de Cauchy, por ende es necesario analizar en diferentes casos:

- Si $g_k^\top B_k g_k \leq 0$ entonces la función \mathcal{T}_k es decreciente para $\tau \geq 0$ por tanto se toma $\tau^* = 1$ como solución al problema de optimización restringida correspondiente al paso 2.
- Si $g_k^\top B_k g_k > 0$ entonces la función \mathcal{T}_k es convexa, por lo tanto tiene un mínimo dado por $\bar{\tau} > 0$ sin embargo se debe cumplir que $\tau^* \leq 1$ por lo tanto se selecciona

$$\tau^* = \min \left(1, \frac{\|g_k\|^3}{\Delta_k g_k^\top B_k g_k} \right).$$

En resumen, para solucionar el problema del paso dos de cálculo del punto de Cauchy se toma el siguiente valor de τ^* :

$$\tau^* = \begin{cases} 1 & \text{si } g_k^\top B_k g_k \leq 0, \\ \min \left(1, \frac{\|g_k\|^3}{\Delta_k g_k^\top B_k g_k} \right) & \text{si } g_k^\top B_k g_k > 0. \end{cases}$$

Por tanto, el algoritmo del cálculo del punto de Cauchy se puede hacer de manera analítica y se presenta en el pseudocódigo a continuación.

Cálculo del punto de Cauchy

Entrada: $x_k, \nabla f(x_k), B_k, \Delta_k$

Paso 1: Calcular la dirección de descenso más pronunciado:

$$p_k^S = -\frac{\Delta_k}{\|g_k\|} g_k$$

Paso 2: Determinar el escalar τ_k que minimiza $m_k(\tau p_k^S)$ sujeto a la restricción de región de confianza:

$$\tau_k = \begin{cases} 1 & \text{si } g_k^\top B_k g_k \leq 0, \\ \min \left(1, \frac{\|g_k\|^3}{\Delta_k g_k^\top B_k g_k} \right) & \text{si } g_k^\top B_k g_k > 0. \end{cases}$$

Paso 3: Calcular el punto de Cauchy: $p_k^C = \tau_k p_k^S$

Salida: p_k^C

2.2.2. Región de confianza método Dogleg

El método de región de confianza con modelo cuadrático se basa en la idea de aproximar la función objetivo f en una vecindad del punto actual x_k mediante un modelo de segundo orden que incorpore tanto la información del gradiente como la de la curvatura. La premisa es que, en una región suficientemente pequeña alrededor de x_k , la función f puede ser bien aproximada por una función polinómica de segundo orden, lo cual permite determinar de forma simultánea la dirección y la magnitud del paso de actualización.

Cuando se utiliza un modelo cuadrático

$$m_k(p) = f(x_k) + g_k^T p + \frac{1}{2} p^T B_k p,$$

con B_k definida positiva, el minimizador no restringido es

$$p_k^N = -B_k^{-1} g_k,$$

denominado *paso completo de Newton*. Si $\|p_k^N\| \leq \Delta_k$ (donde Δ_k es el radio de la región de confianza), entonces la solución del subproblema es $p^*(\Delta_k) = p_k^N$. En cambio, si $\|p_k^N\| > \Delta_k$, la restricción impuesta por la región hace que la solución exacta del subproblema no pueda obtenerse directamente.

Para estos casos, el método Dogleg approxima la trayectoria de la solución restringida mediante una curva compuesta por dos segmentos de línea:

1. El primer segmento se extiende desde el origen hasta el *paso de descenso más pronunciado*, denotado por

$$p_k^C = -\frac{g_k^T g_k}{g_k^T B_k g_k} g_k.$$

Este paso corresponde a minimizar el modelo linealizado (ignorando el término cuadrático), siendo una buena aproximación cuando Δ_k es pequeño.

2. El segundo segmento une p_k^C con el paso completo de Newton p_k^N .

La trayectoria Dogleg se define formalmente mediante la función parametrizada $\tilde{p}(\tau)$ para $\tau \in [0, 2]$:

$$\tilde{p}(\tau) = \begin{cases} \tau p_k^C, & 0 \leq \tau \leq 1, \\ p_k^C + (\tau - 1)(p_k^N - p_k^C), & 1 \leq \tau \leq 2. \end{cases}$$

Esta construcción reemplaza la trayectoria curva real de la solución $p^*(\Delta_k)$ por un camino compuesto por dos segmentos lineales, lo que facilita su cálculo. La idea es que, si $\|p_k^N\| \leq \Delta_k$,

se acepta p_k^N como solución. En caso contrario, la solución restringida se obtiene de la siguiente manera:

- Si el paso de descenso más pronunciado p_k^C ya se encuentra fuera de la región ($\|p_k^C\| \geq \Delta_k$), se toma el punto en la frontera en la dirección de p_k^C , es decir, se normaliza p_k^C a la frontera:

$$p = \Delta_k \frac{p_k^C}{\|p_k^C\|}.$$

- De lo contrario, se utiliza un punto intermedio a lo largo de la trayectoria Dogleg, que se puede expresar como

$$p = p_k^C + \phi(p_k^N - p_k^C),$$

donde el parámetro $\phi \in [0, 1]$ se determina de forma que $\|p\| = \Delta_k$; es decir, ϕ se elige resolviendo la ecuación escalar

$$\|p_k^C + \phi(p_k^N - p_k^C)\|^2 = \Delta_k^2.$$

El método Dogleg se fundamenta en dos propiedades importantes:

- La norma $\|\tilde{p}(\tau)\|$ es una función creciente de τ .
- El valor del modelo $m_k(\tilde{p}(\tau))$ es una función decreciente de τ .

Estas propiedades garantizan que, al avanzar a lo largo de la trayectoria Dogleg, se incrementa la magnitud del paso y se reduce el valor del modelo. Así, cuando el paso completo de Newton no es factible, la intersección única de la trayectoria Dogleg con la frontera $\|p\| = \Delta_k$ proporciona una solución aproximada efectiva para el subproblema.

En la práctica, el método Dogleg es especialmente útil porque:

- Combina de forma natural el beneficio del paso de descenso más pronunciado, que garantiza una reducción significativa cuando Δ_k es pequeño, con el potencial del paso completo de Newton, que favorece una rápida convergencia local.
- Es computacionalmente eficiente, ya que evita la resolución exacta del subproblema restringido mediante métodos iterativos costosos.
- Se adapta dinámicamente al tamaño de la región de confianza: si el modelo predice bien el comportamiento de f , se permite un mayor Δ_k ; en caso contrario, se reduce para mejorar la precisión del modelo.

De esta forma, el método Dogleg se presenta como una estrategia poderosa para resolver el subproblema de la región de confianza con modelo cuadrático, ofreciendo un equilibrio entre robustez y velocidad de convergencia.

Método Dogleg

Entrada: $x_k, f(x_k), g_k = \nabla f(x_k), B_k$ (definida positiva), $\Delta_k > 0$

Salida: p_k (paso aproximado para el subproblema de la región de confianza)

Calcular el paso completo de Newton:

$$p_k^N \leftarrow -B_k^{-1}g_k.$$

Calcular el paso de descenso más pronunciado:

$$p_k^C \leftarrow -\frac{g_k^T g_k}{g_k^T B_k g_k} g_k.$$

Si $\|p_k^N\| \leq \Delta_k$ **entonces**

$p \leftarrow p_k^N$. **(Acepta el paso completo)**

Si no

Si $\|p_k^C\| \geq \Delta_k$ **entonces**

$p \leftarrow \Delta_k \frac{p_k^C}{\|p_k^C\|}$. **(Normaliza el paso de descenso a la frontera)**

Si no

Resolver para $\phi \in [0, 1]$ tal que

$$\|p_k^C + \phi(p_k^N - p_k^C)\| = \Delta_k.$$

Establecer

$$p \leftarrow p_k^C + \phi(p_k^N - p_k^C).$$

Fin si

Fin si

3. Resultados

En esta sección se presentan los resultados obtenidos al aplicar diferentes estrategias de optimización en la resolución algorítmica del problema planteado. Se han implementado y analizado métodos de búsqueda lineal y de región de confianza, comparando su desempeño en términos de eficiencia y precisión.

Inicialmente, se describen las funciones utilizadas y los procedimientos implementados. Posteriormente, se detallan los métodos de búsqueda lineal, incluyendo el enfoque de backtracking y el método de Wolfe, resaltando sus principios fundamentales y diferencias. Luego, se abordan los métodos basados en regiones de confianza, explorando las estrategias con modelos cuadráticos y de Cauchy.

Los resultados obtenidos se presentan mediante tablas y gráficos comparativos, lo que permite evaluar el desempeño de cada método en distintos escenarios. Además, se discuten las ventajas y desventajas observadas en la práctica, proporcionando un análisis crítico sobre su aplicabilidad.

3.1. Búsqueda lineal

3.1.1. Backtracking

A continuación se presentan los resultados correspondientes a una serie de experimentos numéricos en los que se ha aplicado el método de búsqueda en línea, utilizando la estrategia de Backtracking para la selección del tamaño del paso. La ejecución del algoritmo depende de diversos parámetros, entre los cuales se destacan: el vector de partida x_0 , el tamaño inicial del paso t_{\max} , el factor de reducción β (que determina la disminución del paso en cada iteración), el coeficiente γ empleado en la condición de Armijo para asegurar una reducción adecuada de la función objetivo, la tolerancia establecida para determinar la convergencia, y el número máximo de iteraciones permitidas.

Para ilustrar el comportamiento del método, se realizaron pruebas iniciales con los siguientes valores:

$$x_0 = (-1.2, 1.0), \quad t_{\max} = 1.0, \quad \beta = 0.5, \quad \gamma = 10^{-4}, \quad \text{max_iter} = 100.$$

Convergencia del método de Newton con Backtracking en la función de Rosenbrock

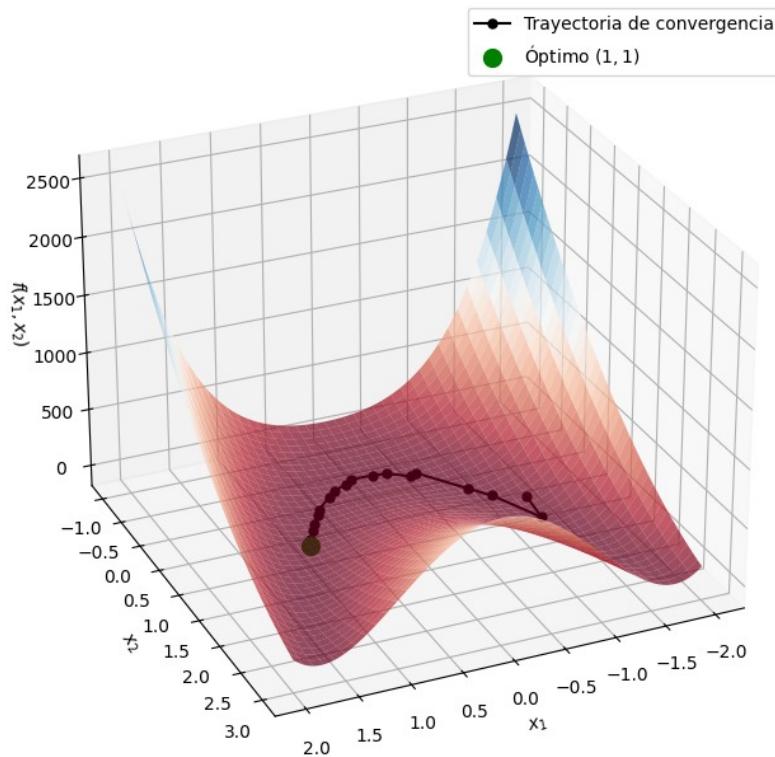


Figura 4: Visualización de la trayectoria de convergencia del método Backtracking.

En un primer gráfico 3D se visualiza la trayectoria de convergencia del algoritmo. En dicha representación, la trayectoria sobre la superficie de la función de Rosenbrock se observa siguiendo un recorrido que se asemeja a una parábola, lo que resulta coherente con la estructura del estrecho valle característico de esta función. La visualización tridimensional permite apreciar cómo el algoritmo se aproxima al mínimo, adaptándose a la curvatura del paisaje de la función.

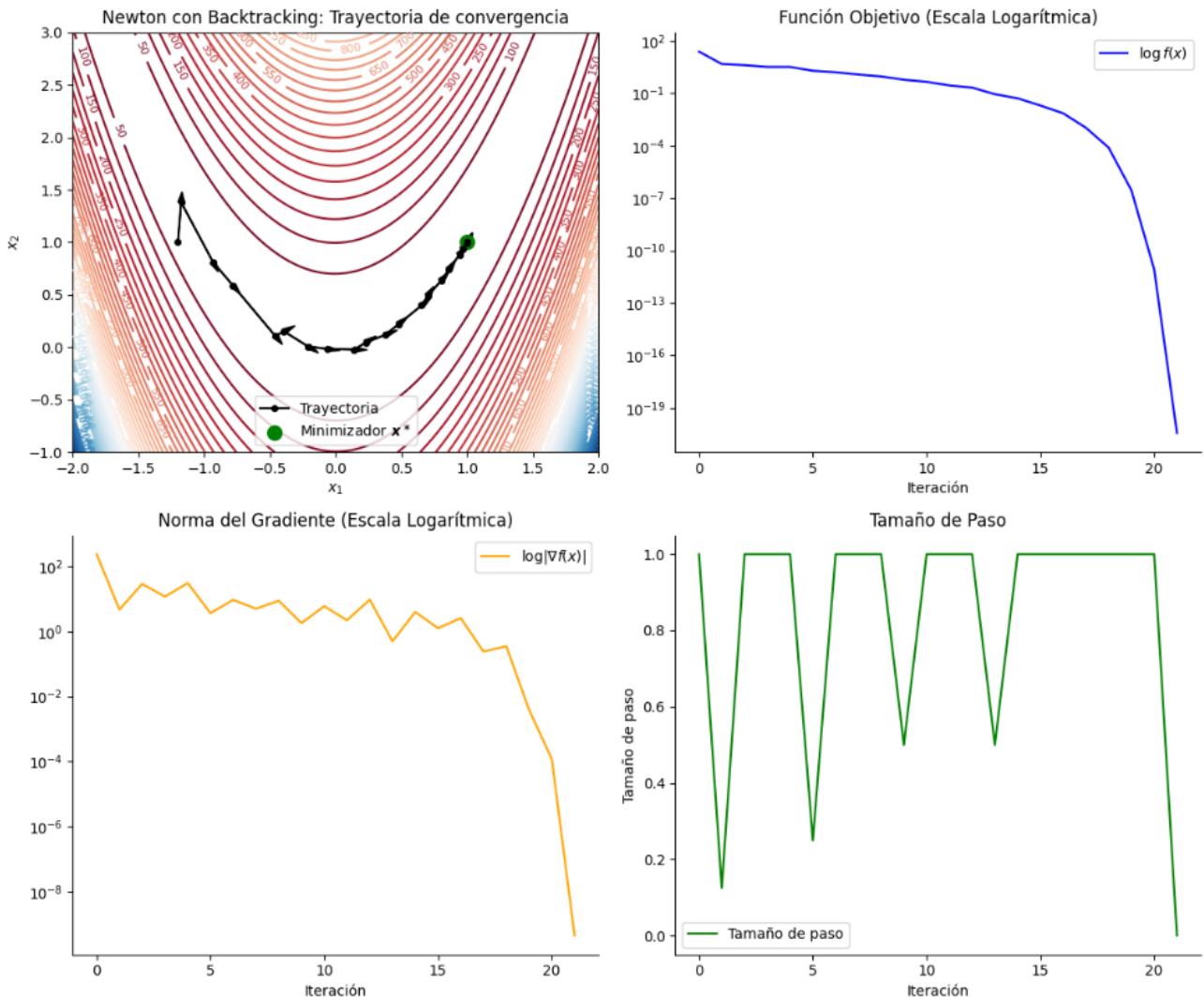


Figura 5: Análisis de la convergencia del método Backtracking. Los gráficos muestran (de izquierda a derecha y de arriba a abajo): la trayectoria de las iteraciones en el espacio (el punto verde indica el minimizador encontrado), la reducción de la función objetivo (en escala logarítmica), la disminución de la norma del gradiente (también en escala logarítmica) y el radio de la región de confianza. Realización propia.

Se presentan, además, gráficos en 2D organizados en una cuadrícula de cuatro subgráficos que muestran distintos aspectos del proceso de convergencia. En el primer subgráfico se muestra la trayectoria de las iteraciones en el espacio x_1-x_2 ; en este gráfico se observa el recorrido seguido por el algoritmo, donde el punto marcado en verde indica el minimizador obtenido. La trayectoria se representa mediante puntos conectados y se incorporan flechas que indican la dirección de cada paso, lo que permite visualizar claramente la evolución del algoritmo en el espacio de búsqueda. En otro gráfico se representa la evolución de la función objetivo $f(x)$ en escala logarítmica, evidenciándose que, a partir de la iteración 18, el valor de $f(x)$ desciende de forma muy rápida. De manera similar, otro subgráfico muestra la norma del gradiente en escala logarítmica, revelando

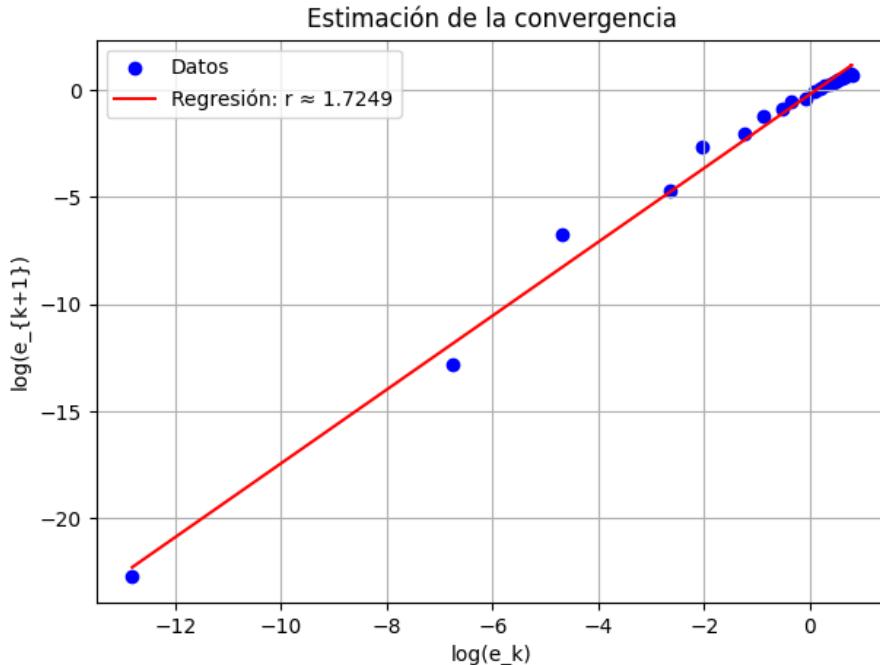


Figura 6: Estimación por regresión lineal de la tasa de convergencia del método de Newton con búsqueda lineal por medio de backtracking. Escala logarítmica. C estimado: 0.820365, r estimado: 1.724856, Coeficiente r^2 : 0.995444.

una disminución abrupta después de la iteración 18, lo que indica que el algoritmo entra en una fase de convergencia acelerada conforme se aproxima al óptimo. También se presenta la evolución del tamaño del paso utilizado en cada iteración, donde se observa que, en la mayoría de las ocasiones, el método selecciona el valor máximo $t_{\max} = 1,0$; en algunos casos, sin embargo, se reduce el tamaño del paso para cumplir con la condición de disminución suficiente.

Debido a que la ejecución del método puede variar en términos de tiempo de ejecución entre diferentes ejecuciones, se repite el experimento 250 veces para obtener una distribución estadística de los tiempos. El análisis del histograma de tiempos muestra que la moda se sitúa en aproximadamente 0.0014 segundos, el tiempo promedio es de 0.00112287 segundos y la varianza es de $2,00776 \times 10^{-7}$ segundos, lo que evidencia un desempeño computacional consistente.

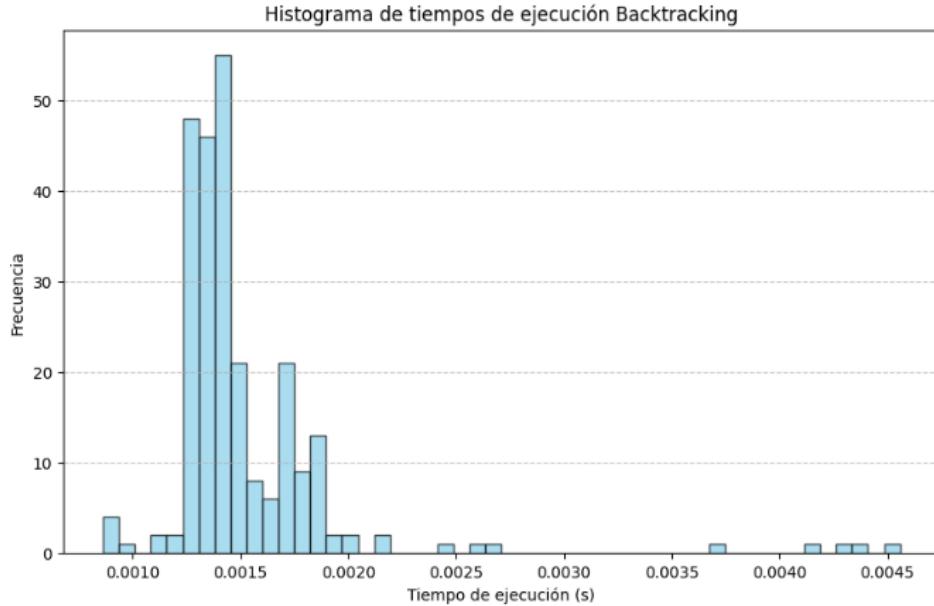


Figura 7: Histograma de los tiempos de ejecución (en segundos) para el modelo Backtracking. Se observa una distribución unimodal con pico de alrededor 0.0014 segundos. Realización propia.

Se realiza un análisis de convergencia mediante regresión lineal sobre los errores de convergencia, relacionando $\log(e_k)$ con $\log(e_{k+1})$. El análisis arroja una pendiente de aproximadamente 1.72, lo que indica que la convergencia es superlineal, es decir, la reducción del error se acelera de forma notable a medida que el algoritmo se acerca al óptimo.

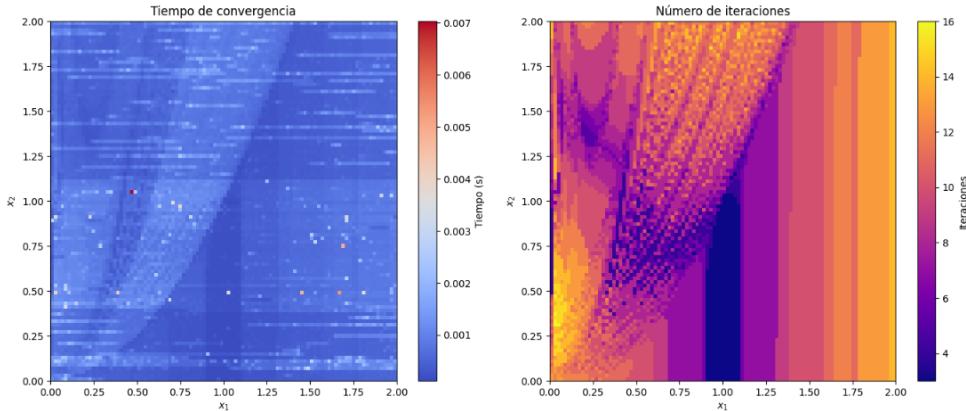


Figura 8: Del lado izquierdo se presenta el mapa de calor asociado al tiempo de convergencia, a la derecha el mapa de calor del numero de iteraciones necesarias para la convergencia. Realización propia.

Asimismo, se generan mapas de calor que permiten visualizar la dependencia del rendimiento del método respecto al punto inicial. En uno de los gráficos se muestra el tiempo de convergencia en función del punto inicial sobre un dominio predefinido, mientras que en otro se representa

el número de iteraciones necesarias para alcanzar la convergencia. Estos mapas revelan que, en la mayoría de los casos, los puntos iniciales permiten una convergencia rápida, aunque en determinadas regiones, particularmente en la parte inferior izquierda, el proceso se ralentiza significativamente. Asimismo, el mapa correspondiente al número de iteraciones evidencia que algunos puntos iniciales requieren un número considerablemente mayor de iteraciones para alcanzar una solución óptima. Estos patrones indican que la dificultad de la optimización está estrechamente relacionada con la compleja geometría de la función de Rosenbrock, caracterizada por un valle estrecho que complica la búsqueda del mínimo.

Durante la ejecución del método se realizan, en promedio, aproximadamente 50 evaluaciones de la función $f(x)$, 22 evaluaciones del gradiente y 22 evaluaciones de la Hessiana. Estos valores indican que el costo computacional de cada iteración es moderado y que el método es capaz de obtener resultados precisos en un número razonable de iteraciones. Los gráficos y análisis presentados confirman la efectividad del método Backtracking aplicado a la función de Rosenbrock, tanto en términos de convergencia como de eficiencia computacional, lo que respalda la viabilidad de esta estrategia en contextos de optimización numérica.

3.1.2. Condiciones débiles de Wolfe

En esta sección, se presentan los resultados para la aplicación del método de Newton con búsqueda lineal por medio de condiciones débiles de Wolfe. En este caso, hay dependencia del punto inicial x_0 , el tamaño inicial de paso α , la constante de la condición de suficiente decrecimiento c_1 , la constante de la condición de curvatura c_2 , la razón de crecimiento de paso λ y el máximo número de iteraciones permitidas `max_iter`. A menos que se indique lo contrario, se asumirán los siguientes valores en los parámetros.

$$x_0 = (-1.2, 1.0), \quad \alpha = 1, \quad c_1 = 10^{-4}, \quad c_2 = 0.9, \quad \lambda = 2 \quad \text{max_iter} = 100.$$

En la Figura 9 se muestran algunas imágenes correspondientes a la convergencia del método con estos parámetros. Se puede observar que se obtiene un comportamiento muy similar al realizado por el método de Backtracking. Se realizó una estimación por regresión lineal de la convergencia del error, así como se presentó en el método de backtracking; se obtuvo que, para este método, $e_{k+1} = Ce_k^r$, donde $C \approx 0.800$ y $r \approx 1.656$ con un coeficiente r cuadrado de 0,987. Esto indica convergencia superlineal del método.

Los valores de los parámetros obtenidos afectan el comportamiento del método. Para el valor de c_1 , tanto el número de iteraciones como el tiempo aumentan a medida que este valor aumenta. Los resultados se pueden observar en la Figura 10. Asimismo, se evalúa el comportamiento del método cuando c_2 varía. Notamos que, a pesar de alcanzar un menor número de iteraciones con

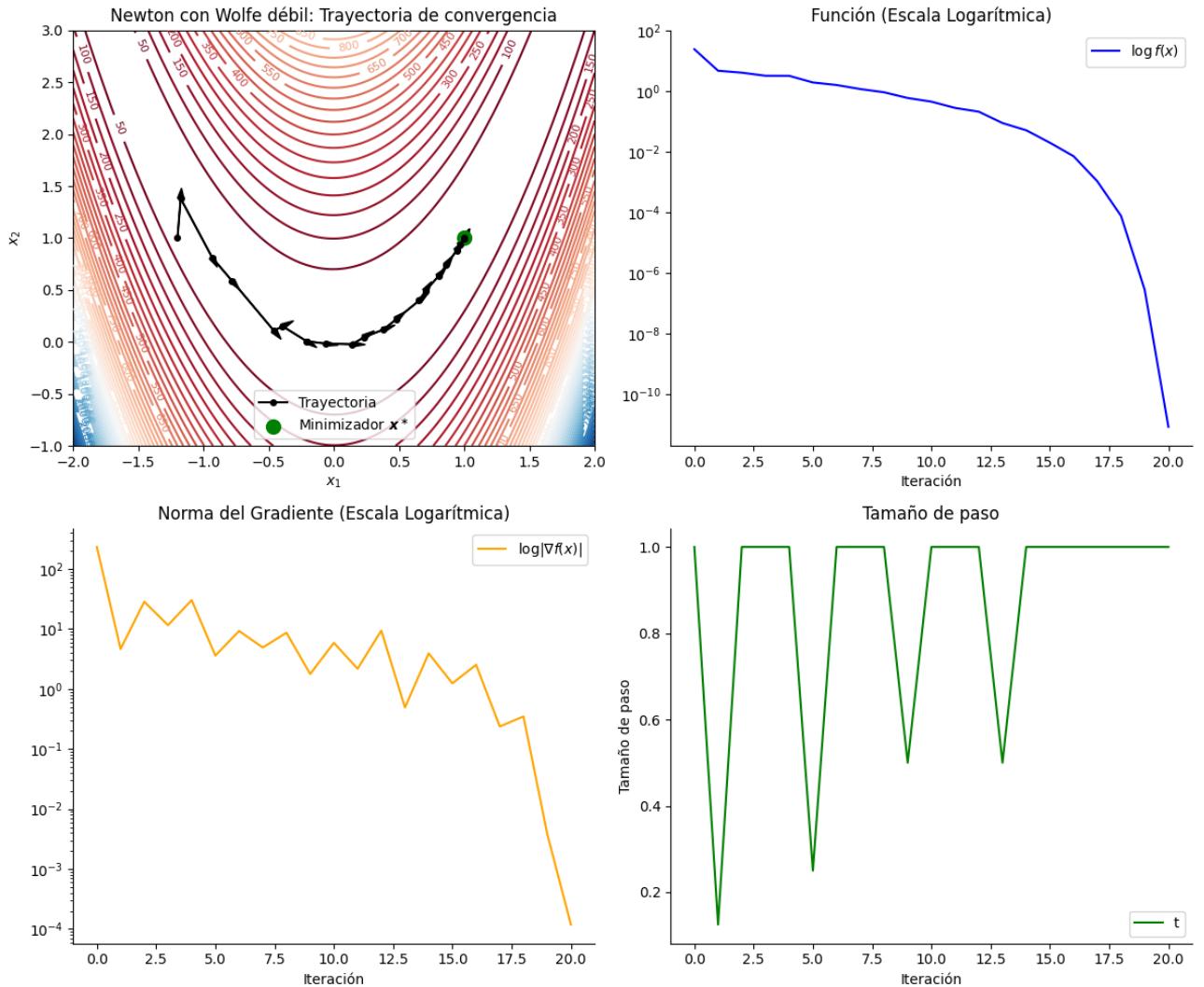


Figura 9: Análisis de convergencia en el método de Newton con búsqueda lineal con condiciones débiles de Wolfe (de izquierda a derecha y de arriba a abajo): trayectoria del método, reducción de la evaluación de la función, reducción de la norma del gradiente, tamaño de paso en cada iteración.

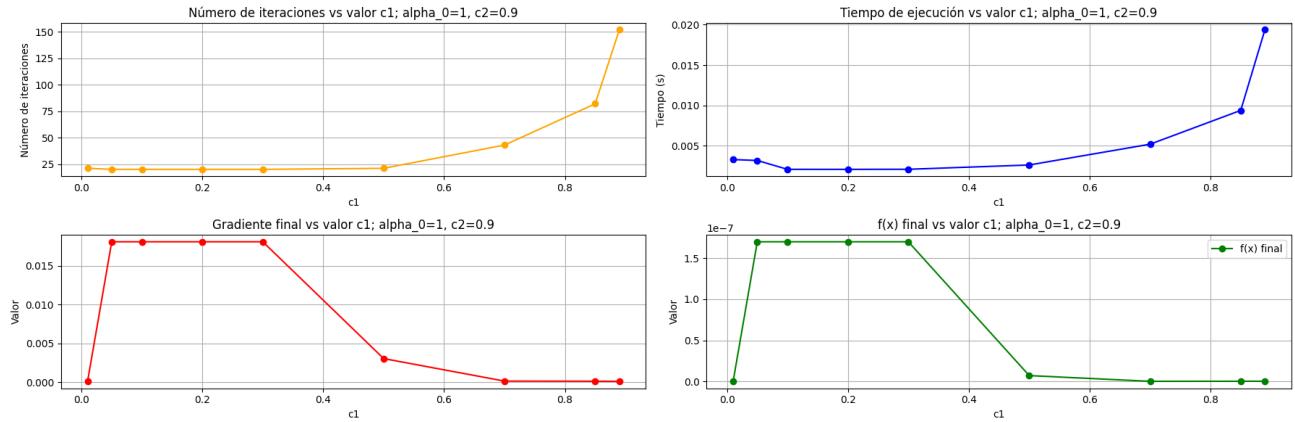


Figura 10: Gráficas del comportamiento del método de Newton con búsqueda lineal con condiciones débiles de Wolfe dependiendo del valor de $c_1 \in (0, 0, 9)$. Se fija $c_2 = 0,9$ y $\alpha = 1$.

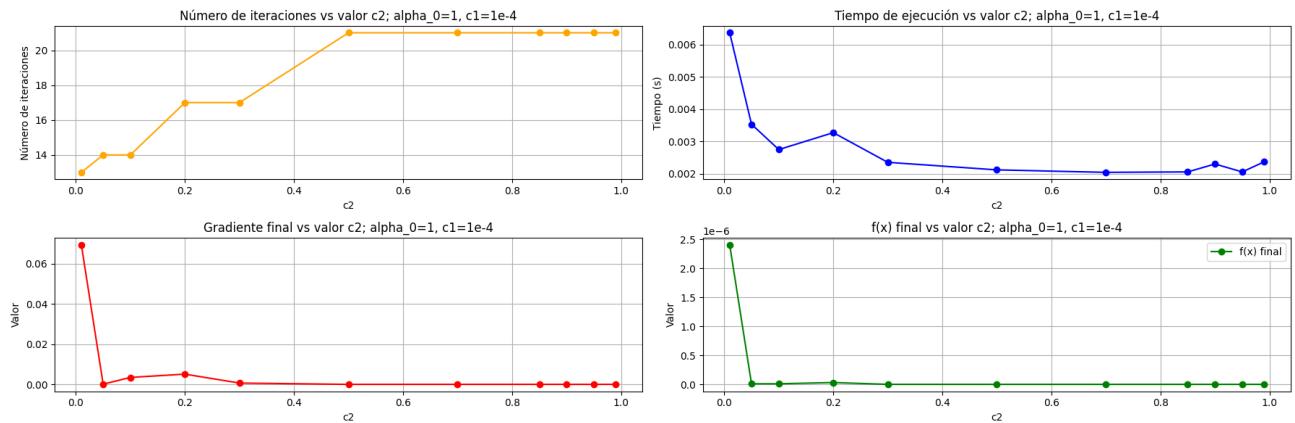


Figura 11: Gráficas del comportamiento del método de Newton con búsqueda lineal con condiciones débiles de Wolfe dependiendo del valor de $c_2 \in (0,01,0,9)$. Se fija $c_1 = 10^{-4}$ y $\alpha = 1$.

un c_2 menor (es decir, siendo más exigente con la condición de curvatura), el tiempo para el cálculo de cada uno de estos pasos aumenta. Por ello, los mejores comportamientos se evidencian cuando se toma $c_2 = 0,9$. Parece que la eficiencia del método se empeora si se toma un c_2 muy pequeño. Los resultados se pueden ver en la Figura . Por último, también se puede intentar analizar cómo varía el comportamiento del método si se toma un valor inicial de paso distinto a $\alpha = 1$. Los resultados se pueden visualizar en la Figura 12. Aunque los resultados para el tiempo son inestables, se puede notar que el mejor tiempo se obtuvo con $\alpha = 1$, ratificando, como suele mencionarse en la literatura, que el método de Newton da una buena estimación del paso inicial ([2]).

En la Figura 13, se muestra el histograma correspondiente a la ejecución del algoritmo con los parámetros iniciales un total de 250 veces. Se obtuvo un tiempo promedio $\mu \approx 0,00135s$, con una varianza de $\sigma^2 = 2,46 \times 10^{-7}$. Por último, en la Figura 14, se visualizan los tiempos que

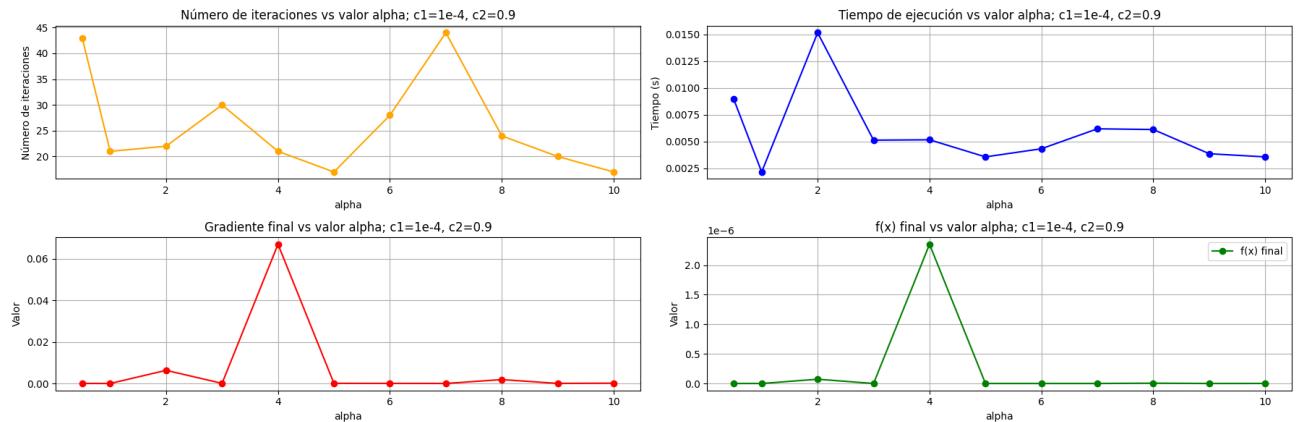


Figura 12: Gráficas del comportamiento del método de Newton con búsqueda lineal con condiciones débiles de Wolfe dependiendo del valor de $\alpha \in (0, 10)$. Se fija $c_1 = 10^{-4}$ y $c_2 = 0,9$.

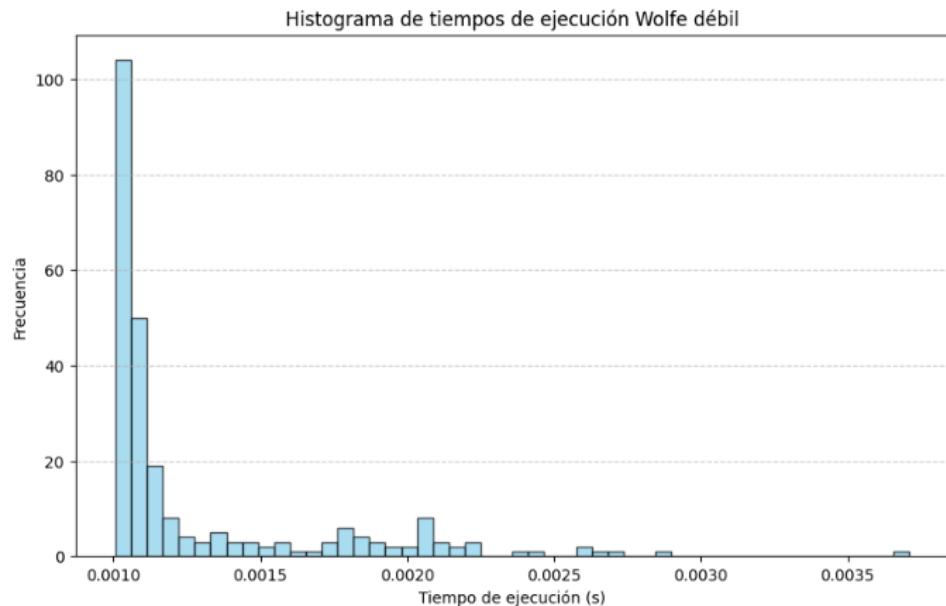


Figura 13: Histograma de los tiempos de ejecución (en segundos) para el método de Newton con búsqueda lineal por condiciones débiles de Wolfe. Se observa una distribución unimodal con pico de alrededor 0.0011 segundos. Realización propia.

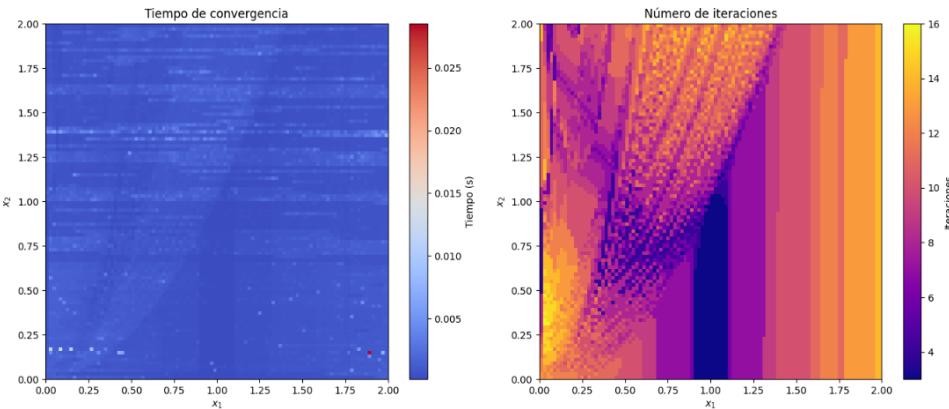


Figura 14: Comportamiento del método de Newton con búsquedas lineales por medio de condiciones débiles de Wolfe dependiendo del punto inicial. A la izquierda el mapa de calor asociado al tiempo de convergencia, a la derecha el mapa de calor del número de iteraciones necesarias para la convergencia. Realización propia.

demoró el método en converger, así como la cantidad de iteraciones requeridas.

3.1.3. Condiciones fuertes de Wolfe

En esta sección, se presentan los resultados para la aplicación del método de Newton con búsquedas lineales por medio de condiciones fuertes de Wolfe. Así como en las condiciones débiles, hay dependencia del punto inicial x_0 , el tamaño inicial de paso α , la constante de la condición de suficiente decrecimiento c_1 , la constante de la condición de curvatura c_2 , y el máximo número de iteraciones permitidas max_iter . A menos que se indique lo contrario, se asumirán los siguientes valores en los parámetros.

$$x_0 = (-1.2, 1.0), \quad \alpha = 1, \quad c_1 = 10^{-4}, \quad c_2 = 0.9, \quad \text{max_iter} = 100.$$

En la Figura 15, se muestran algunas imágenes correspondientes a la convergencia del método con estos parámetros. Se observa un comportamiento distinto con respecto a la elección de los anteriores métodos. Para este método, se estimó que $e_{k+1} = Ce_k^r$, donde $C \approx 0.759$ y $r \approx 1.606$ con un coeficiente cuadrado de 0,971504. Esto evidencia una convergencia superlineal del método.

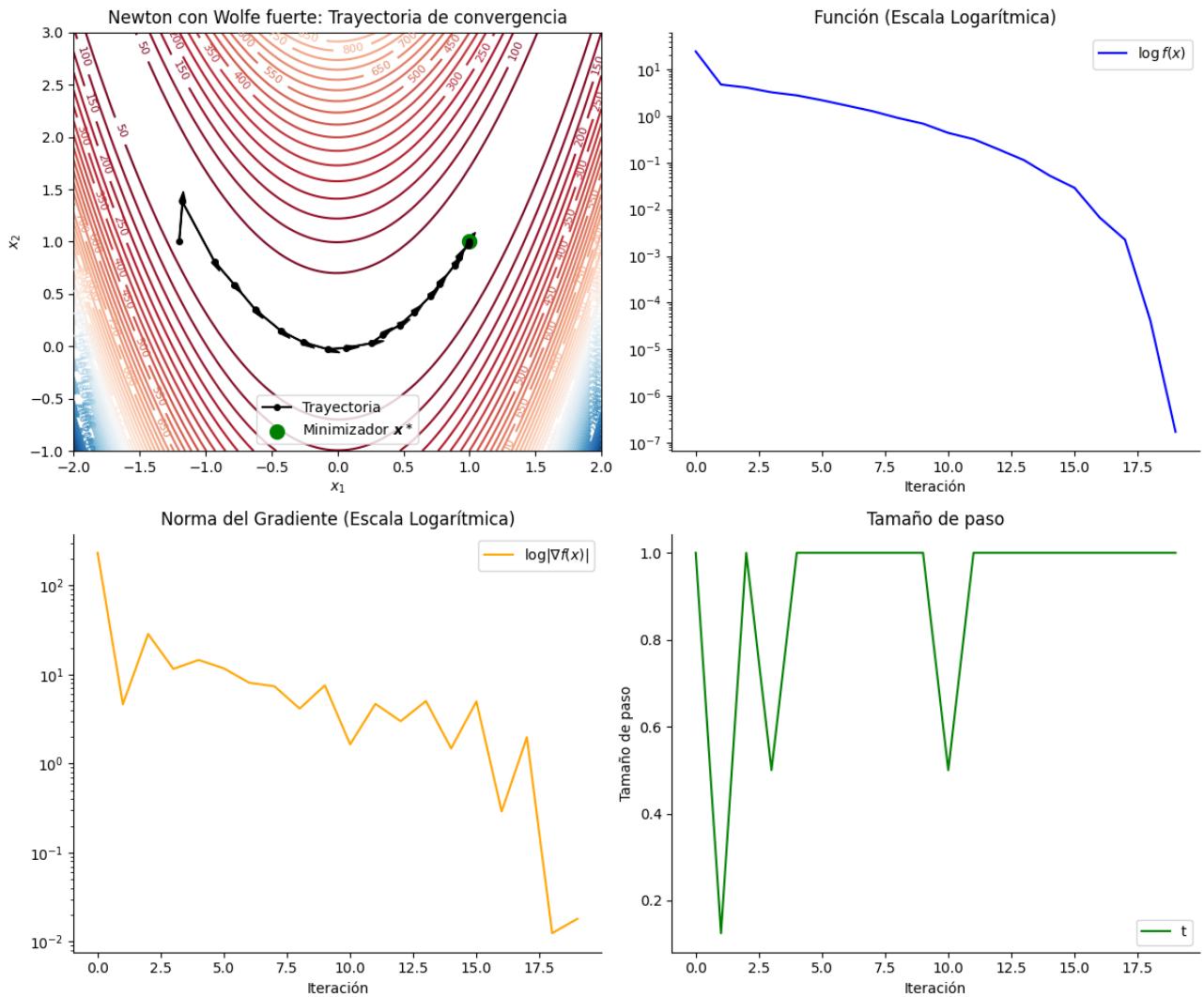


Figura 15: Análisis de convergencia en el método de Newton con búsqueda lineal con condiciones fuertes de Wolfe (de izquierda a derecha y de arriba a abajo): trayectoria del método, reducción de la evaluación de la función, reducción de la norma del gradiente, tamaño de paso en cada iteración.

Los comportamientos del algoritmo para los distintos valores de c_1 , c_2 y α siguen una tendencia muy similar al método con las condiciones débiles de Wolfe. Un mayor valor de c_1 resulta en una mayor exactitud en la estimación final del método a costa de mayor tiempo y cantidad de iteraciones, mientras que un mayor valor de c_2 , aunque reduce el número de iteraciones, resulta en un mayor tiempo de ejecución. Asimismo, el valor de paso inicial $\alpha = 1$ logra uno de los mejores registros en todos los aspectos. En el cuaderno adjunto, se visualizan las gráficas correspondientes a las variaciones de los parámetros.

Se realizó igualmente el histograma para 250 ejecuciones del método con los parámetros iniciales. En la Figura 17, se ve la distribución de los resultados. Se obtuvo una distribución bimodal

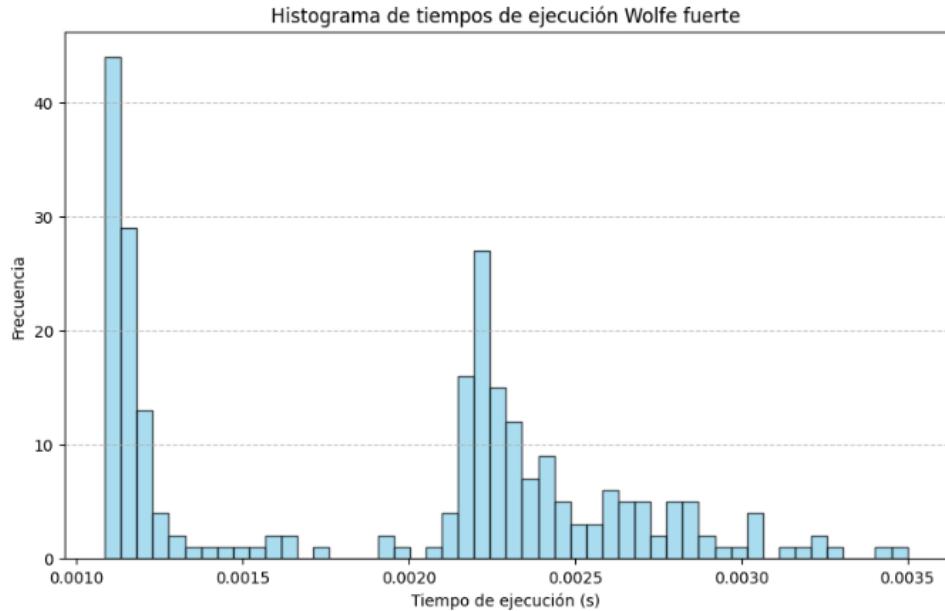


Figura 16: Histograma de los tiempos de ejecución (en segundos) para el método de Newton con búsqueda lineal por condiciones fuertes de Wolfe. Se observa una distribución bimodal con media en 0.0016 segundos. Realización propia.

de los tiempos con media $\mu = 0,00161$ y varianza $\sigma^2 = 5,59 \times 10^{-7}$. Por último, la Figura 17 permite visualizar los mapas de calor para el tiempo y número de iteraciones necesarias variando el punto inicial.

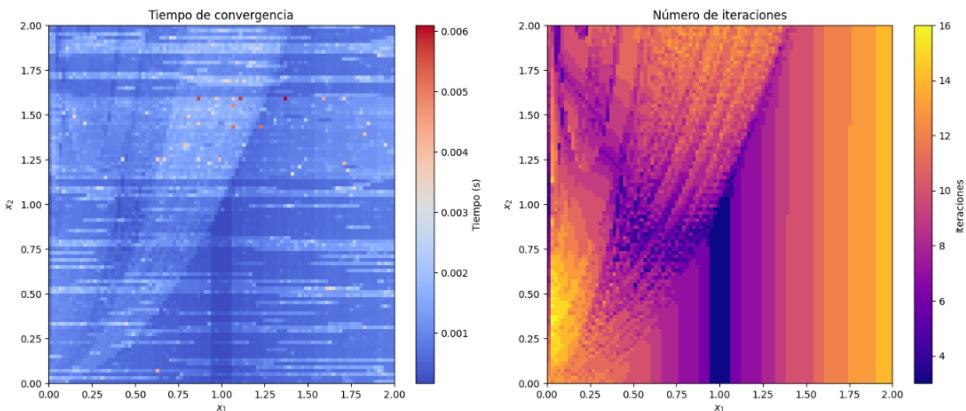


Figura 17: Del lado izquierdo se presenta el mapa de calor asociado al tiempo de convergencia, a la derecha el mapa de calor del numero de iteraciones necesarias para la convergencia. Realización propia.

3.2. Región de confianza

3.2.1. Región de confianza con modelo de Cauchy

Los resultados presentados en esta sección provienen de una serie de experimentos numéricos utilizando el método de región de confianza con el modelo de Cauchy. La ejecución del algoritmo depende de varios parámetros: el punto inicial x_0 , el valor máximo del paso Δ_{\max} , el valor inicial del paso Δ_0 , el parámetro η , la tolerancia permitida ϵ y el número máximo de iteraciones `max_iter`.

Inicialmente, el algoritmo se ejecuta con los siguientes valores:

$$x_0 = (-1.2, 1.0), \quad \Delta_{\max} = 2.0, \quad \Delta_0 = 1.0, \quad \eta = 0.02, \quad \text{max_iter} = 10000.$$

Para evaluar la convergencia del método, se analizan la trayectoria de las iteraciones en el espacio, la reducción de la función objetivo en escala logarítmica, la disminución de la norma del gradiente en escala logarítmica y el tiempo de ejecución acumulado. La Figura 18 ilustra estos resultados, mostrando la evolución del algoritmo hasta alcanzar el minimizador, representado en color verde.

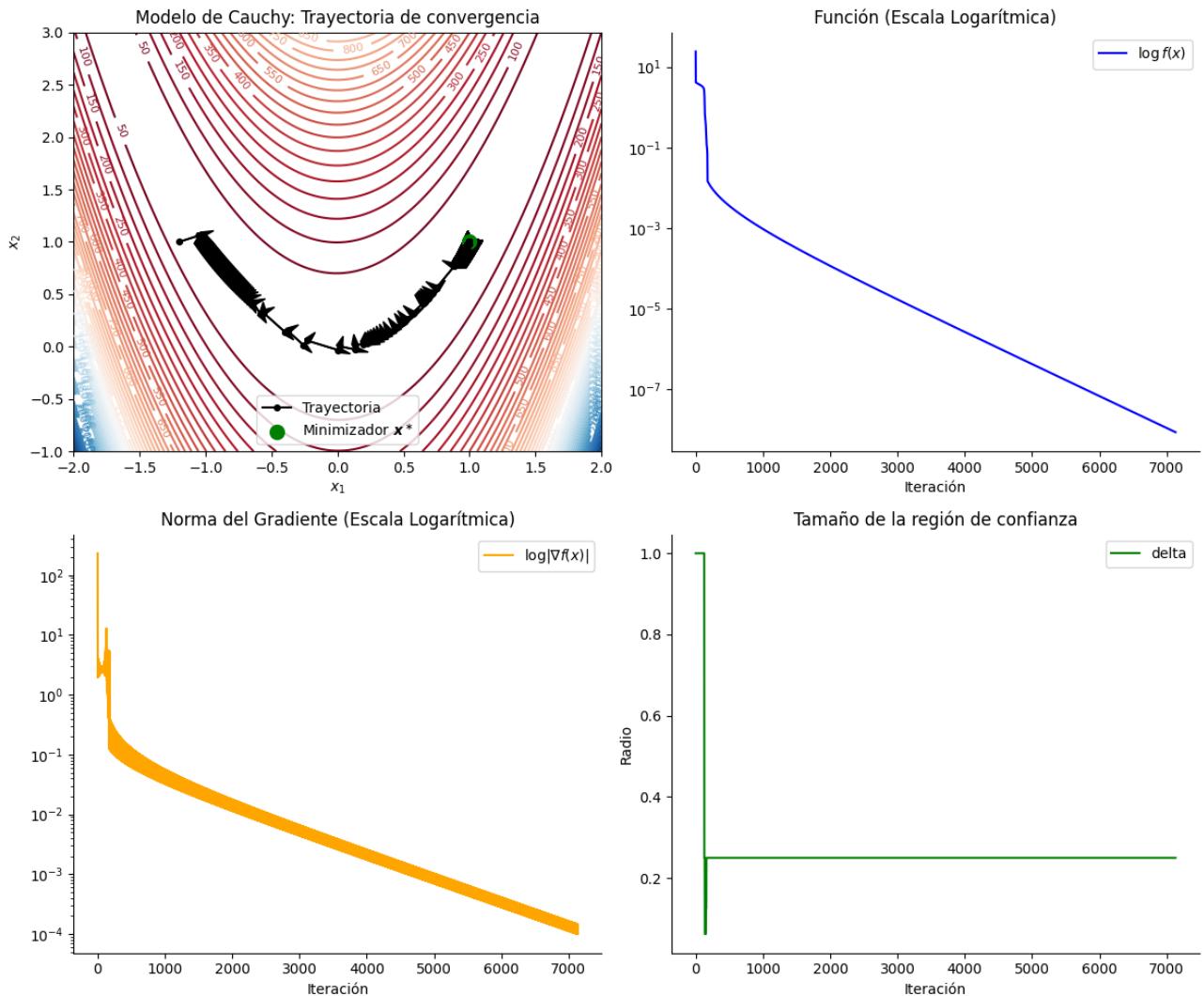


Figura 18: Análisis de la convergencia del modelo de Cauchy. Los gráficos muestran (de izquierda a derecha y de arriba a abajo): la trayectoria de las iteraciones en el espacio (el punto verde indica el minimizador encontrado), la reducción de la función objetivo (en escala logarítmica), la disminución de la norma del gradiente (también en escala logarítmica) y el radio de la región de confianza. Realización propia.

En la Figura 18 se observa que el método sigue una trayectoria cercana a la parábola $x_2 - x_1^2 = 0$ hasta alcanzar el minimizador. Esto se debe a que la dirección del cálculo del punto de Cauchy está determinada por el descenso más pronunciado. Se evidencia que el método ajusta progresivamente su trayectoria para acercarse al minimizador, manteniéndose a su vez próximo a la parábola mencionada. En términos de convergencia, tanto la función objetivo como la norma del gradiente presentan un comportamiento decreciente y aproximadamente lineal en escala logarítmica. Para cuantificar la tasa de convergencia, se define el error en la iteración k como $e_k := \|x_k - x^*\|$. Si se supone que la sucesión de errores sigue la relación $e_{k+1} \approx C e_k^r$, $0 < C < 1$, donde C regula la magnitud del error en cada iteración, y r controla la tasa de reducción, entonces

a partir de los resultados obtenidos, el ajuste por mínimos cuadrados proporciona los valores $C = 0.996779$ y $r = 0.999623$, lo que confirma que el método sigue una convergencia lineal, dado que $r \approx 1$. La última gráfica muestra el radio de la región de confianza que se estabiliza en las primeras iteraciones manteniendo un valor constante de 0.25.

El tiempo de ejecución del algoritmo finaliza en aproximadamente 0.65 segundos, tras 7132 iteraciones, evaluando la función 14264 veces, y calculando 7133 veces el gradiente y la hessiana. Como resultado, se obtiene una norma del gradiente de $\|\nabla f(x)\| = 9.98182 \times 10^{-5}$, un valor de la función objetivo de $f(x) = 8.45098 \times 10^{-9}$, y una solución final $x = (0.9999082, 0.99981592)$.

Para evaluar la variabilidad del tiempo de ejecución, el algoritmo se ejecuta 250 veces con los mismos valores iniciales. La Figura 27 muestra el histograma de estos tiempos, donde se observa una distribución bimodal con picos alrededor de 0.3 y 0.6 segundos, con una media de $\mu_t = 0.385084$ y una varianza de $\sigma_t^2 = 0.014227$.

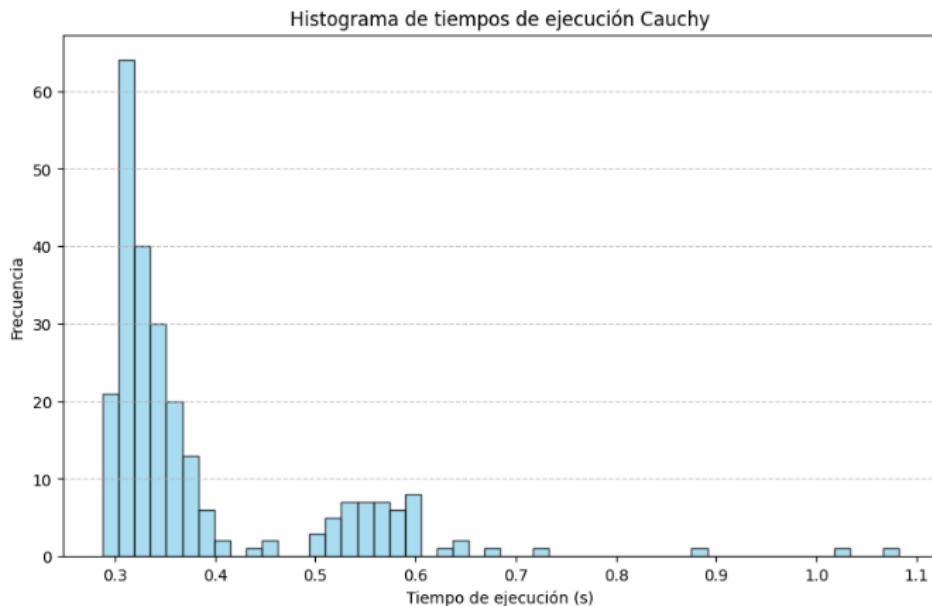


Figura 19: Histograma de los tiempos de ejecución (en segundos) para el modelo de Cauchy. Se observa una distribución bimodal con picos alrededor de 0.3 y 0.6 segundos. Realización propia.

Para evaluar la sensibilidad del algoritmo con respecto a los parámetros Δ_{\max} y η , se realizaron múltiples ejecuciones variando estos valores mientras se mantenían fijos los demás parámetros. A continuación, se presentan los resultados obtenidos.

Cuadro 1: Efecto de la variación de Δ_{\max} en el rendimiento del algoritmo.

Δ_{\max}	Tiempo (s)	Iteraciones	Último $\ \nabla f(x)\ $	Último $f(x)$
0.5	0.3463	7132	9.9818×10^{-5}	8.4509×10^{-9}
1.0	0.3492	7132	9.9818×10^{-5}	8.4509×10^{-9}
2.0	0.3339	7132	9.9818×10^{-5}	8.4509×10^{-9}
5.0	0.3492	7132	9.9818×10^{-5}	8.4509×10^{-9}
10.0	0.3484	7132	9.9818×10^{-5}	8.4509×10^{-9}

En el Cuadro 1 se observa que el número de iteraciones y los valores finales de la norma del gradiente y de la función objetivo permanecen constantes al modificar Δ_{\max} . Esto indica que la elección de este parámetro no influye en la convergencia del método en este caso particular. Sin embargo, se detectan pequeñas fluctuaciones en el tiempo de ejecución, lo que sugiere que este parámetro podría afectar la eficiencia computacional en otras instancias del problema.

Cuadro 2: Efecto de la variación de η en el rendimiento del algoritmo.

η	Tiempo (s)	Iteraciones	Último $\ \nabla f(x)\ $	Último $f(x)$
0.001	0.3587	7132	9.9818×10^{-5}	8.4509×10^{-9}
0.01	0.3333	7132	9.9818×10^{-5}	8.4509×10^{-9}
0.02	0.3505	7132	9.9818×10^{-5}	8.4509×10^{-9}
0.05	0.3331	6788	9.9854×10^{-5}	9.8726×10^{-9}
0.1	0.3290	6788	9.9854×10^{-5}	9.8726×10^{-9}
0.25	0.1433	3073	9.9852×10^{-5}	1.1044×10^{-8}

El Cuadro 2 muestra el impacto de la variación del parámetro η . Se observa que, para valores pequeños ($\eta \leq 0.02$), el número de iteraciones y los valores finales de la función objetivo y la norma del gradiente son prácticamente los mismos. Sin embargo, al aumentar η a 0.05 o más, el número de iteraciones comienza a reducirse significativamente, indicando que el método converge más rápido. Para $\eta = 0.25$, el número de iteraciones se reduce a menos de la mitad en comparación con los casos anteriores, con una mejora sustancial en el tiempo de ejecución. No obstante, el valor final de la función objetivo es ligeramente mayor, lo que sugiere un posible compromiso entre velocidad y precisión. En conclusión, el parámetro Δ_{\max} no afecta de manera significativa la convergencia del algoritmo en este problema, mientras que el valor de η influye

directamente en la cantidad de iteraciones necesarias para alcanzar la solución, impactando el tiempo de ejecución. Un valor de η demasiado grande podría comprometer la estabilidad del método, mientras que un valor pequeño ralentiza la convergencia sin aportar mejoras en precisión.

En la Figura 20, la gráfica de la izquierda muestra la convergencia en el plano de los distintos puntos iniciales. Los puntos verdes representan los casos en los que el algoritmo logra converger, mientras que los puntos rojos indican aquellos en los que no lo consigue. La gráfica central muestra la distribución de los tiempos de ejecución, mientras que la gráfica de la derecha ilustra la distribución del número de iteraciones necesarias para la convergencia.

Para realizar este experimento numérico, se generaron aleatoriamente 300 puntos en el espacio siguiendo una distribución normal $\mathcal{N}(\mathbf{x}^*, \sigma^2 = 2)$ y luego se aplicó el método, almacenando información sobre su convergencia, el tiempo de ejecución y el número de iteraciones. Los parámetros del método fueron:

$$\Delta_{\max} = 2.0, \quad \Delta_0 = 1.0, \quad \eta = 0.02, \quad \text{max_iter} = 10000.$$

Los resultados indican que el método presenta una tasa de convergencia del 86.66 % para todos los puntos de partida considerados (posteriormente, veremos que esto se debe al número máximo de iteraciones fijado). Además, el tiempo de ejecución promedio es de $\mu_t = 0.312468$ con una varianza de $\sigma_t^2 = 0.027142$, mientras que el número medio de iteraciones es de $\mu_{\text{iteraciones}} \approx 5979$.

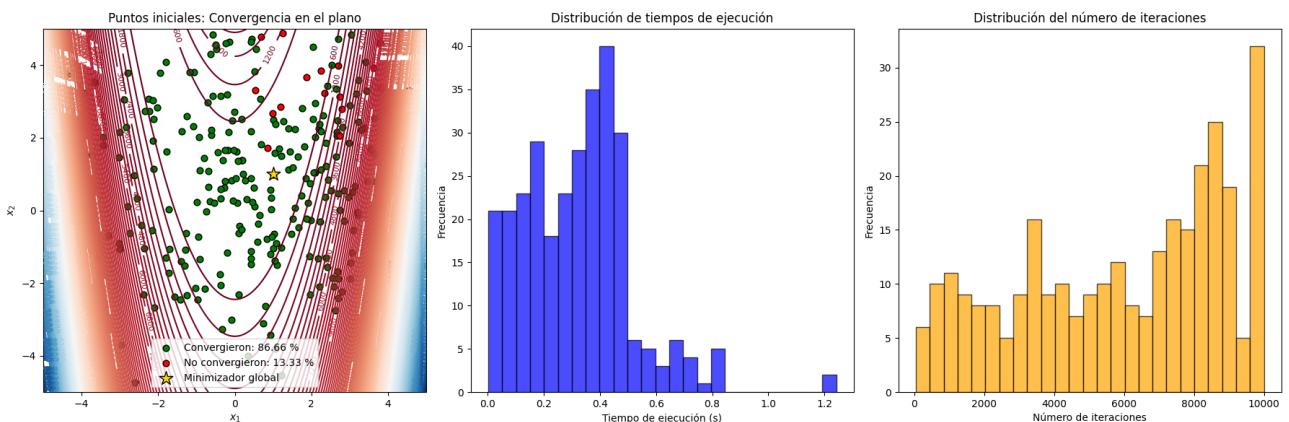


Figura 20: Análisis del desempeño del modelo de Cauchy. Se muestran: (izquierda) la convergencia en el plano para diversos puntos iniciales (verde: convergen, rojo: no convergen), (centro) la distribución de los tiempos de ejecución y (derecha) la distribución del número de iteraciones. Realización propia usando Python.

Se puede evidenciar que en el histograma de la distribución del número de iteraciones hay una alta frecuencia de valores cercanas a 10000 iteraciones, a partir de esto se plantea la hipótesis de que la no convergencia de algunos puntos iniciales se debe a que el algoritmo no tiene la

suficiente cantidad de iteraciones máximas o que el tamaño de paso es muy pequeño para lograr una convergencia. Es por esto que se realiza nuevamente un experimento numérico en donde se generaron aleatoriamente 1000 puntos en el espacio siguiendo una distribución normal $\mathcal{N}(\mathbf{x}^*, \sigma^2 = 4)$ y luego se aplicó el método, almacenando información sobre su convergencia, el tiempo de ejecución y el número de iteraciones. Los parámetros del método fueron:

$$\Delta_{\max} = 10.0, \quad \Delta_0 = 1.0, \quad \eta = 0.02, \quad \text{max_iter} = 100000.$$

Los resultados indican que el método presenta una tasa de convergencia del 100 % para todos los puntos de partida considerados. Además, el tiempo de ejecución promedio es de $\mu_t = 0.258008$ con una varianza de $\sigma_t^2 = 0.027224$, mientras que el número medio de iteraciones es de $\mu_{\text{iteraciones}} \approx 7017$. Nótese que en la Figura 21 se exhibe el comportamiento del algoritmo permitiendo un tamaño de paso más grande y un mayor número de iteraciones, en el histograma de número de iteraciones no se muestran valores cercanos al número máximo indicando que en este caso la cantidad máxima de iteraciones es suficiente.

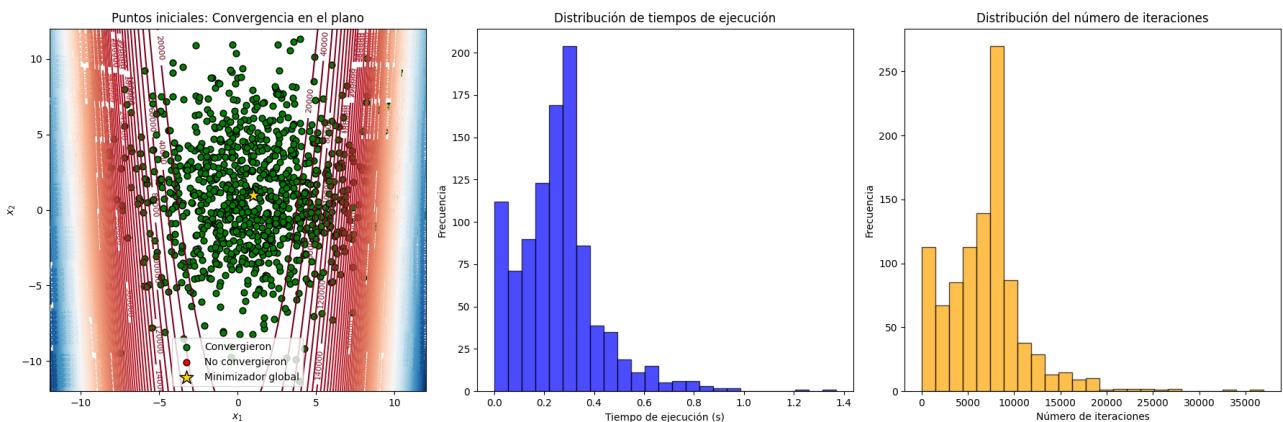


Figura 21: Análisis del desempeño del modelo de Cauchy. Se muestran: (izquierda) la convergencia en el plano para diversos puntos iniciales (verde: convergen, rojo: no convergen), (centro) la distribución de los tiempos de ejecución y (derecha) la distribución del número de iteraciones. Realización propia usando Python.

En la Figura 23, los mapas de calor muestran el rendimiento del método de optimización basado en la región de confianza aplicado a la función de Rosenbrock. Los gráficos fueron generados evaluando el método de optimización en una malla de puntos dentro del dominio $[0, 2] \times [0, 2]$, donde cada punto representa una condición inicial diferente para el algoritmo. Para cada punto, se ejecutó el método de optimización basado en la región de confianza con modelo Cauchy, registrando tanto el tiempo de ejecución como el número de iteraciones necesarias hasta la convergencia. En el primer gráfico, que representa el tiempo de convergencia, se observa que la mayoría de los puntos iniciales permiten una rápida convergencia, pero existen regiones donde el

proceso es más lento, especialmente en la parte superior derecha. De manera similar, el segundo gráfico, que indica el número de iteraciones necesarias para la convergencia, revela que ciertos puntos iniciales requieren hasta 10000 iteraciones para alcanzar una solución óptima. Estos patrones sugieren que la dificultad de la optimización está correlacionada con la geometría de la función de Rosenbrock, caracterizada por un valle estrecho que complica la búsqueda de mínimos. Para mejorar el desempeño del método, se recomienda una selección adecuada del punto inicial, preferiblemente cerca del valle de la función, y una estrategia adaptativa en la actualización del radio de la región de confianza para evitar estancamientos en zonas de alta dificultad.

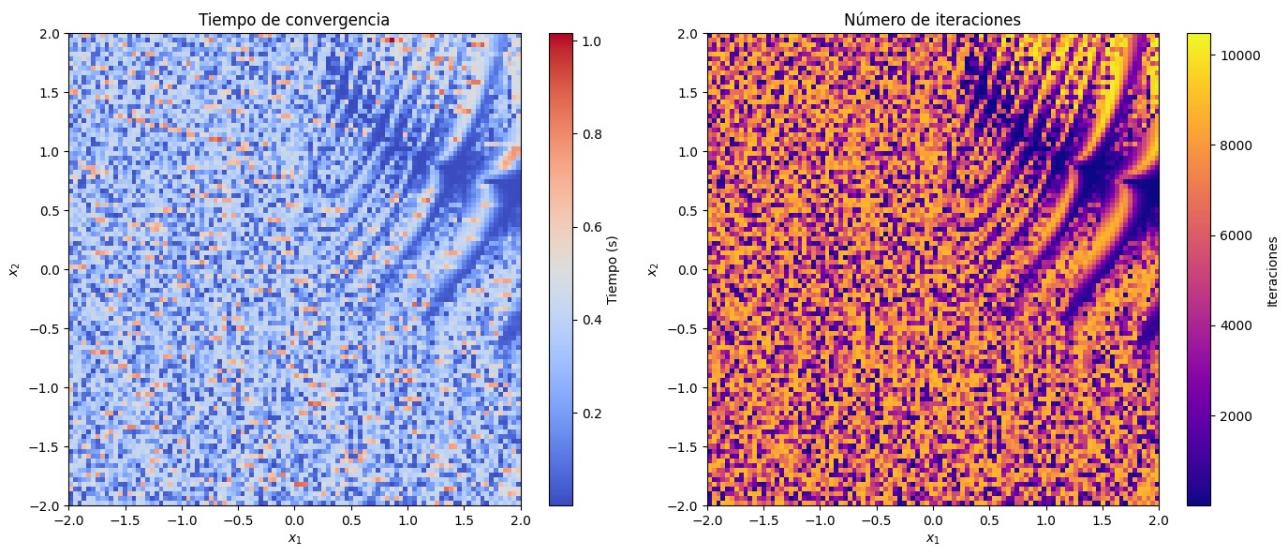


Figura 22: En la izquierda se muestra el mapa de calor asociado al tiempo de convergencia y en la derecha se tiene el mapa de calor del numero de iteraciones necesarias para convergencia. Realización propia usando Python.

Evaluando con los mismos parámetros mencionados anteriormente y aumentando la grilla a 200×200 apreciamos un 'trozo de parábola' en la esquina superior derecha donde si se escoge allí el punto inicial su número de iteraciones disminuye drásticamente con respecto a su vecindad.

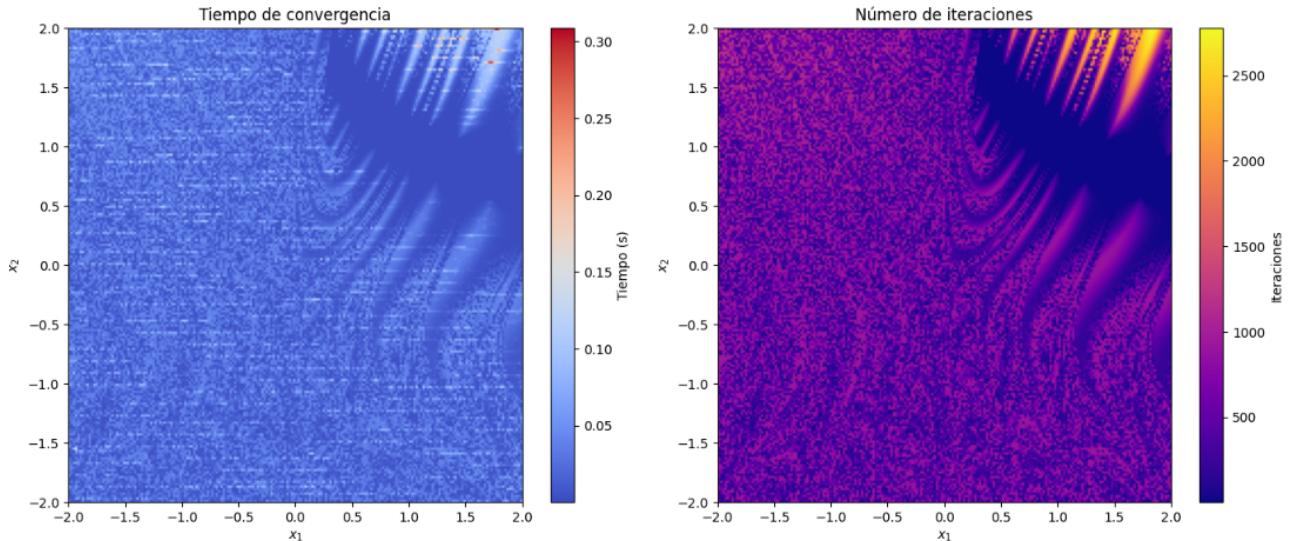


Figura 23: En la izquierda se muestra el mapa de calor asociado al tiempo de convergencia y en la derecha se tiene el mapa de calor del numero de iteraciones necesarias para convergencia. Realización propia usando Python.

Estos resultados sugieren que el método tiene un alto grado de estabilidad en un amplio rango de condiciones iniciales, con tiempos de ejecución relativamente homogéneos y una cantidad de iteraciones controlada en la mayoría de los casos. Sin embargo, la presencia de puntos que no convergen resalta la importancia de elegir cuidadosamente las condiciones iniciales para garantizar el éxito del algoritmo.

3.2.2. Región de confianza con método Dogleg

Se estudia el método Dogleg con parámetros de entrada $x_0 = (-1,2,1,0)$, tolerancia $tol = 10^{-6}$, radio inicial de la región de confianza $\Delta_0 = 0,1$, parámetro de aceptación $\eta = 0,1$ y máximo de 100 iteraciones. La visualización de la trayectoria de convergencia en 2D muestra que el algoritmo converge en aproximadamente 10 pasos, combinando pasos muy largos—que permiten un avance significativo—with pasos cortos, que actúan como correcciones finas. La gráfica de contornos de la función de Rosenbrock evidencia cómo la trayectoria sigue un recorrido definido, en el que se aprecian movimientos notables hacia el valle característico de la función. Asimismo, la evolución de la función objetivo y la norma del gradiente, ambas representadas en escala logarítmica, indican una disminución muy rápida conforme el método se aproxima al óptimo; sin embargo, aunque se alcanza rápidamente la vecindad del mínimo, cumplir la estricta tolerancia resulta más difícil en esa etapa final.

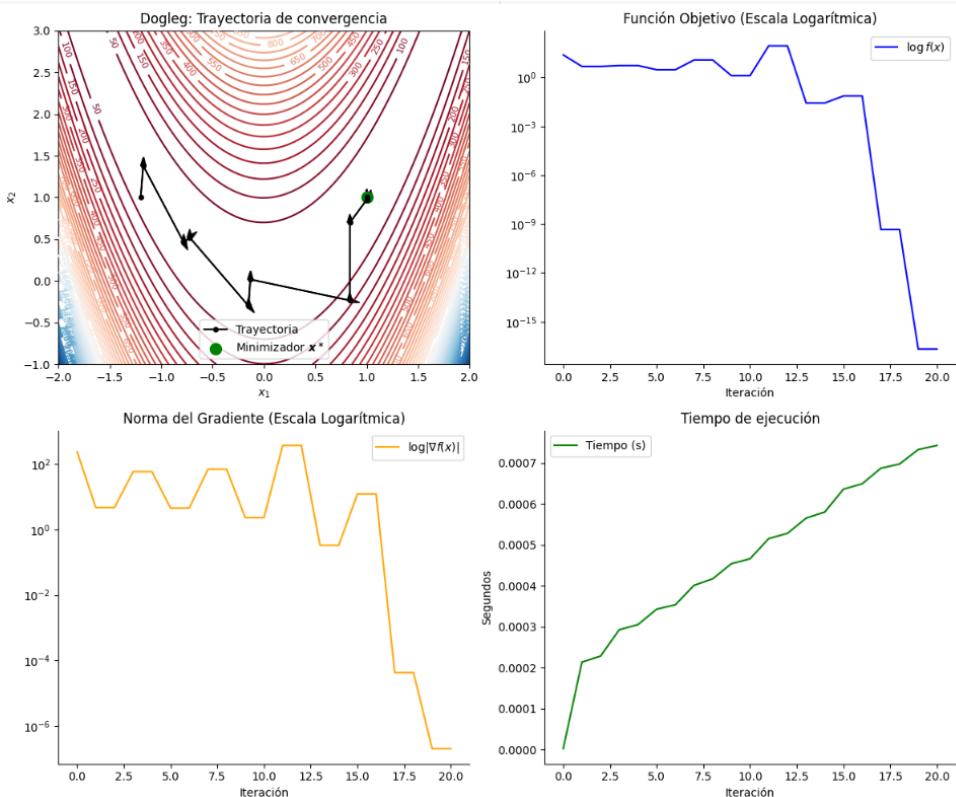


Figura 24: Análisis de la convergencia del modelo de Dogleg. Los gráficos muestran (de izquierda a derecha y de arriba a abajo): la trayectoria de las iteraciones en el espacio (el punto verde indica el minimizador encontrado), la reducción de la función objetivo (en escala logarítmica), la disminución de la norma del gradiente (también en escala logarítmica) y el radio de la región de confianza. Realización propia.

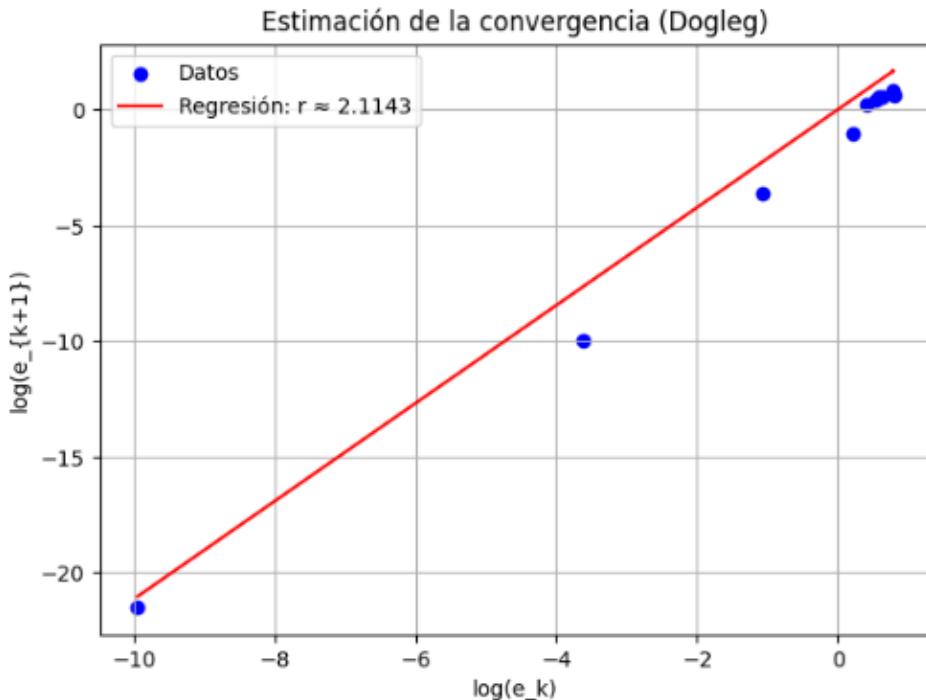


Figura 25: Análisis de la convergencia del modelo de Dogleg. Los gráficos muestran (de izquierda a derecha y de arriba a abajo): la trayectoria de las iteraciones en el espacio (el punto verde indica el minimizador encontrado), la reducción de la función objetivo (en escala logarítmica), la disminución de la norma del gradiente (también en escala logarítmica) y el radio de la región de confianza. Realización propia.

El análisis de convergencia se complementa con un estudio basado en los errores de convergencia, calculados como la norma de la diferencia entre el vector iterado y el óptimo conocido $(1, 1)$. Al ajustar una regresión lineal sobre la escala logarítmica de estos errores, se obtiene un valor $r = 2,1143$. Este resultado indica que el error en la iteración $k + 1$ se reduce aproximadamente como el error en la iteración k elevado a la potencia de 2.1143, lo que evidencia una convergencia de orden cuadrático o, incluso, ligeramente supercuadrática. En otras palabras, conforme el método se acerca al mínimo, la disminución del error se acelera de manera drástica, lo cual es un comportamiento deseable en métodos basados en aproximaciones de segundo orden.

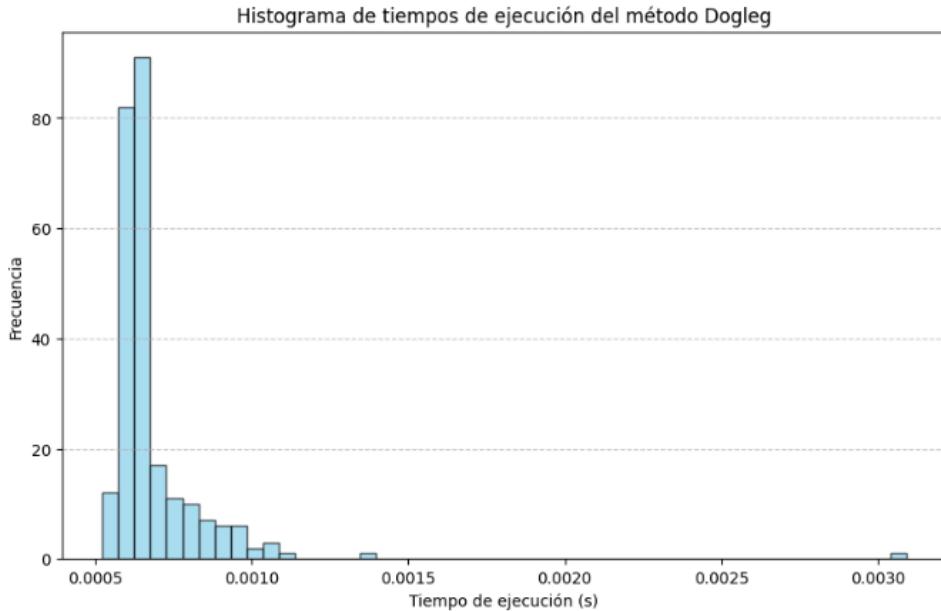


Figura 26: Histograma de los tiempos de ejecución (en segundos) para el modelo Dogleg. Se observa una distribución unimodal con pico de alrededor 0.0006 segundos. Realización propia.

El rendimiento temporal del método Dogleg se evalúa mediante la ejecución de 250 corridas, lo que permite obtener una robusta distribución estadística de los tiempos de ejecución. El análisis del histograma de tiempos muestra que la moda se sitúa en 0.0006 segundos, lo que subraya la rapidez del método; el tiempo medio de ejecución se estima en 0.008 segundos, y la varianza es de 0.0099. Estos resultados confirman la alta eficiencia computacional del método, haciendo notar que el algoritmo es extremadamente rápido en la práctica.

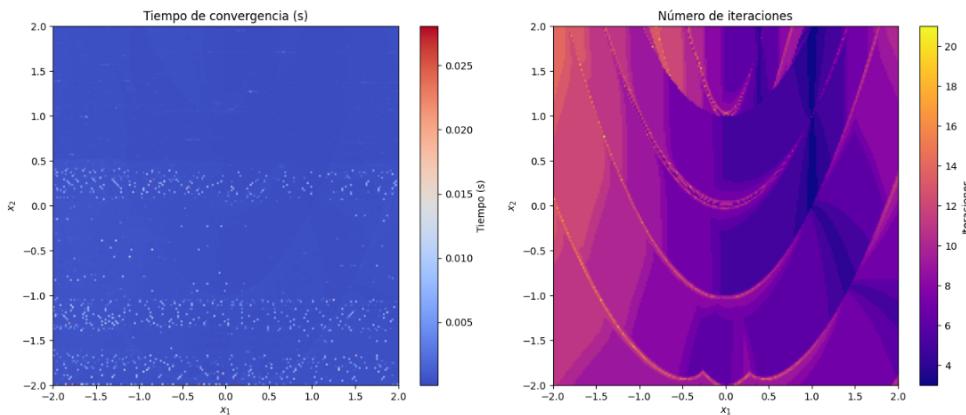


Figura 27: En la izquierda se muestra el mapa de calor asociado al tiempo de convergencia y en la derecha se tiene el mapa de calor del numero de iteraciones necesarias para convergencia. Realización propia usando Python.

Además, se construyen mapas de calor para visualizar el comportamiento del método en función

del punto inicial. En el mapa de calor del número de iteraciones, se observan sucesivas curvas parabólicas, las cuales parecen disponer como si se tratase de un túnel: en las líneas de cada parábola se requieren más iteraciones para alcanzar el óptimo, mientras que entre estas se evidencia una reducción en el número de iteraciones necesarias. Este patrón sugiere que, a lo largo del dominio, la dificultad para optimizar la función de Rosenbrock varía de manera significativa, lo que puede atribuirse a la compleja geometría del valle estrecho y curvado de la función.

En resumen, la trayectoria de convergencia del método Dogleg en 2D se caracteriza por pasos largos que permiten un avance sustancial y pasos cortos para correcciones finas, logrando una reducción muy rápida de la función objetivo y de la norma del gradiente conforme se aproxima al óptimo.

3.3. Comparación entre los métodos

A continuación, se muestra una comparación de algunas métricas de efectividad de los algoritmos propuestos, con los parámetros dados anteriormente. Recuerde que todos estos métodos fueron iterados desde $x_0 = (-1.2, 1)$.

Método	C	r	μ_t	σ_t^2
Backtracking	0.820	1.725	0.0011	2.077×10^{-7}
Wolfe débil	0.800	1.656	0.0014	2.465×10^{-7}
Wolfe fuerte	0.760	1.606	0.0016	5.59×10^{-7}
Cauchy	0.997	1.000	0.365	0.0100
Dogleg	0.353372	2.1143	0.008	9.55×10^{-7}

Cuadro 3: Comparación de velocidad de convergencia para la función de Rosenbrock con $x_0 = (-1.2, 1)$. Estimaciones de los parámetros de velocidad de convergencia, tiempos promedio de ejecución, y varianza en los tiempos de ejecución.

A partir de aquí, se puede concluir que los métodos con mejores parámetros de convergencia estimados son Backtracking y Dogleg. El método que menor tiempo demoró en promedio es Dogleg, seguido de los métodos de búsqueda lineal, liderados en desempeño por el método de backtracking. La siguiente tabla resume el último valor alcanzado por cada uno de los métodos.

Método	$ \nabla f $	f
Backtracking	4.473328×10^{-10}	3.743976×10^{-21}
Wolfe débil	4.473328×10^{-10}	3.743976×10^{-21}
Wolfe fuerte	1.022826×10^{-6}	2.754213×10^{-13}
Cauchy	9.981822×10^{-5}	8.450988×10^{-9}
Dogleg	1.268565×10^{-09}	8.588586×10^{-22}

Cuadro 4: Comparación de métodos según precisión de valor de llegada para la función de Rosenbrock, con $x_0 = (-1.2, 1)$. De izquierda a derecha, norma del gradiente y evaluación de la función en el punto final de la iteración.

Es importante tener en cuenta la precisión de máquina al interpretar estos resultados. De esta forma, las diferencias entre los valores del orden de 10^{-16} se pueden asumir despreciables; sin embargo, cabe resaltar que el método que mejor precisión obtuvo en la evaluación de la función cercano al óptimo fue el método Dogleg. Otro parámetro que puede ser de interés es la cantidad de evaluaciones que se realizan en cada método. Este parámetro se muestra a continuación. Aquí

Método	eval(f)	eval(∇f)	eval(H)	Iteraciones
Backtracking	50	22	22	21
Wolfe débil	50	43	22	21
Wolfe fuerte	67	42	21	20
Cauchy	21397	7133	7133	7133
Dogleg	21	21	10	10

Cuadro 5: Evaluaciones de f , ∇f y H para cada método

podemos apreciar de mejor manera la similitud en la construcción de los métodos de búsqueda lineal, pues, al diferenciarse sobre todo en la manera de optimizar el tamaño de paso t_k , se realizan casi las mismas evaluaciones en la Hessiana. Vemos también que Wolfe requiere casi el doble de evaluaciones del gradiente con respecto a Backtracking para obtener una aproximación casi indistinguible.

Por último, se muestra una comparación del promedio de tiempos e iteraciones para la aplicación de los métodos con distintos puntos iniciales de la región $[-2, 2] \times [-2, 2]$ (aquellos que se usaron para la construcción de los mapas de calor).

Método	t_{prom}	$\text{iter}_{\text{prom}}$
Backtracking	7.38×10^{-4}	9.45
Wolfe débil	9.39×10^{-4}	9.46
Wolfe fuerte	1.06×10^{-3}	9.67
Cauchy	2.94×10^{-1}	5.35×10^3
Dogleg	4.62×10^{-4}	7.84

Cuadro 6: Tiempo y número de iteraciones promedio por método para distintos puntos iniciales en la región $[-2, 2] \times [-2, 2]$ (tomados de manera uniforme)

4. Conclusiones

El análisis comparativo realizado en este trabajo, mediante la aplicación de diferentes estrategias de optimización a la función de Rosenbrock iniciada en $x_0 = (-1.2, 1.0)$, permite concluir que la estructura compleja del valle estrecho y curvado de esta función impone importantes desafíos para los algoritmos de optimización no lineal. Se han evaluado métodos de búsqueda lineal, tales como Backtracking, Wolfe débil y Wolfe fuerte, así como métodos basados en regiones de confianza, representados en este estudio por el modelo de Cauchy y el método Dogleg. Cada uno de estos métodos presenta ventajas y desventajas en cuanto a la velocidad de convergencia, la precisión final y el costo computacional.

Los métodos de búsqueda lineal, como Backtracking y las variantes basadas en las condiciones de Wolfe, muestran una convergencia rápida en términos de reducción del valor de la función y de la norma del gradiente, aunque se evidencia que el descenso se vuelve brusco en fases intermedias (por ejemplo, a partir de la iteración 18 en el caso de Backtracking). En contraste, el modelo de Cauchy, aunque garantiza una aproximación confiable en etapas iniciales, requiere un mayor número de evaluaciones y, en consecuencia, un tiempo de ejecución mucho mayor. Se conjectura que esto se debe a la similitud que tiene el método de Cauchy con el método de descenso del gradiente.

El método Dogleg, por su parte, destaca por su capacidad para combinar la ventaja de un paso de descenso pronunciado con la precisión del paso completo de Newton. En la visualización en 2D se observa que Dogleg converge en apenas 10 pasos, alternando entre pasos largos que impulsan el avance significativo y pasos cortos que actúan como correcciones finas. Esta característica se refleja en una rápida disminución tanto del valor de la función objetivo como de la norma del gradiente conforme el método se aproxima al óptimo. Asimismo, el estudio de convergencia basado en errores iterativos revela un valor de $r = 2,1143$, lo cual indica una convergencia de orden cuadrático o incluso ligeramente supercuadrático, es decir, el error se reduce de manera drástica en cada iteración cuando el algoritmo se acerca al mínimo. Adicionalmente, el análisis de las evaluaciones realizadas muestra que Dogleg requiere significativamente menos evaluaciones

del gradiente y de la Hessiana, lo que reduce el costo computacional total del algoritmo.

Los mapas de calor, que representan el número de iteraciones necesarias para la convergencia en función del punto inicial, revelan patrones interesantes: en determinadas regiones del espacio, especialmente donde la función presenta una mayor complejidad geométrica, el número de iteraciones se agrupa en patrones parabólicos que sugieren “túneles” de alta dificultad. Esta variabilidad confirma que la elección del punto inicial y la estrategia para la actualización del radio de la región de confianza son aspectos fundamentales para el rendimiento global del algoritmo.

En síntesis, los resultados obtenidos permiten concluir que, si bien cada método presenta sus propias fortalezas, los métodos con mejores parámetros de convergencia estimados son Backtracking y Dogleg. En particular, el método Dogleg se destaca por su alta eficiencia y rapidez, mostrando una convergencia muy rápida (alrededor de 10 iteraciones) y una tasa de convergencia cuadrática o ligeramente superior. Estas características, junto con un bajo costo computacional en términos de evaluaciones, hacen del método Dogleg la opción más prometedora para la resolución de problemas de optimización no lineal, especialmente en casos complejos como la función de Rosenbrock. Para sacar conclusiones más definitivas respecto al uso de estos métodos se requeriría evaluar su desempeño con otro tipo de funciones.

5. Referencias

1. Boyd, S., & Vandenberghe, L. (2004). Convex optimization. Cambridge University Press.
2. Escobar, H. M. M. (2001). Optimización no lineal y dinámica. Universidad Nacional de Colombia, Fac. de Ciencias, Depto de Matemáticas y Estadística.
3. Nocedal, J., & Wright, S. J. (Eds.). (1999). Sequential quadratic programming. En Numerical Optimization (pp. 526-573). Springer-Verlag. https://doi.org/10.1007/0-387-22742-3_18
4. Rosenbrock, H. H. (1960). An automatic method for finding the greatest or least value of a function. The Computer Journal, 3(3), 175-184. <https://doi.org/10.1093/comjnl/3.3.175>
5. Nocedal, J., & Wright, S. J. (Eds.). (1999). Sequential quadratic programming. En Numerical Optimization (pp. 526-573). Springer-Verlag. https://doi.org/10.1007/0-387-22742-3_3
6. G.Nash, A. Sofer, I.Griva. (Eds.). (2008). Linear and Nonlinear Optimization.(pp. 357-400). https://doi.org/10.1137/1.978089871773_0