Gradiente descendente con backtracking, bisección o interpolación y Line Search

Resumen

Una de las desventajas que tenía el método de gradiente descendente con tamaño de paso fijo, es que cambiando un poco este parámetro los resultados podían empeorar mucho. Por ello, es que en este trabajo se utilizaron tres métodos para encontrar el tamaño de paso en el algoritmo: backtracking, bisección o interpolación, y Line Search.

Introducción

Los algoritmos de optimización sin restricciones, comienzan con un vector inicial x_0, y buscan de forma iterativa el mínimo de una función. Para decidir la dirección, utilizan información de la función como el Hessiano o el gradiente. Un parámetro importante para estos algoritmos es el tamaño de paso α que debe de brindar el suficiente decrecimiento de la función f, de la misma forma que debe de hacerlo de forma eficiente.

Condiciones de Wolfe

Son dos condiciones: de suficiente decrecimiento y de curvatura . Son un conjunto de desigualdades para búsquedas lineales inexactas, que buscan encontrar el mínimo de una función.

$$f(x_k + \alpha_k d_k) \leq f(x_k) + c_1 \alpha_k \nabla f_k^T d_k, \tag{1}$$

$$\nabla f(x_k + \alpha_k d_k)^T d_k \geq c_2 \nabla f_k^T d_k, \tag{2}$$

con $0 < c_1 < c_2 < 1$.

Condiciones de Goldstein

También buscan que α sea suficiente decrecimiento pero no sea tan corto, y se representan con desigualdades. Estas se utilizan mucho con los métodos tipo Newton, pero no para los métodos cuasi-Newton.

$$f(x_k) + (1-c)\alpha \nabla f_k^T d_k \leq f(x_k + \alpha d_k) \leq f(x_k) + c\alpha \nabla f_k^T d_k. \tag{3}$$

En este trabajo se utilizan las condiciones de Wolfe para los métodos estudiados, la razón es que se busca no dejar el tamaño de paso fijo y encontrar uno con un algoritmo iterativo.

Métodos

Backtracking

Lo que busca este algoritmo es ir en dirección al gradiente, primero hace una búsqueda en dirección a este y luego se utiliza el método de gradiente descedente con el parámetro obtenido.

Algorithm 1 Suficiente decrecimiento y backtracking

$\hat{\alpha} > 0, \rho \in (0,1), c1 \in (0,1)$
$\alpha = \hat{\alpha}$
while $f(x_k + \alpha d_k) \leq f(x_k) + c1\alpha \nabla_k^T d_k$ **do**
    $\alpha = \rho\alpha$
return α

Gradiente descendente con backtracking

Algorithm 2 Gradiente descendiente con backtracking

Inicializar $x_0, \tau > 0, \rho \in (0,1)$ y $0 < c_1 < 1$
d = - gradiente f(x)
while norma de gradiente mayor a τ **do**
    Encontrar α con backtracking
    Actualizar x = x + αd

Bisección o interpolación con condiciones de Wolfe

El objetivo de este algoritmo es encontrar un α que satisfaga la condición de suficiente decrecimiento pero que no sea tan pequeño. El procedimiento es encontrar un α más grande en cada iteración. Este algoritmo se denota como eficiente, ya que busca calcular el gradiente de la función el mínimo de veces.

Algorithm 3 Bisección optimización

$0 < c_1 < c_2 < 1$
$\alpha = 0, \beta = \infty, \alpha_i = \alpha_0, i = 0$
for i = 0 hasta máximo de iteraciones **do**
    if $f(x + \alpha_i d) > f(x) + c_1 \alpha_i \nabla f(x) d$ **then**
        $\beta = \alpha_i, \alpha_{i+1} = \frac{1}{2}(\alpha + \beta)$
    if $\nabla f(x + \alpha_i d) d < c_2 \nabla f d$ **then**
        $\alpha = \alpha_i$,
        if $\beta \gg 0$ **then**
            $\alpha_{i+1} = 2\alpha$
            $\alpha_{i+1} = \frac{1}{2}(\alpha + \beta)$
    En otro caso, retorna.
    Regresar α

Line Search

Este algoritmo busca complir con las condiciones fuertes de Wolfe, para cualesquiera parámetros c_1 y c_2 que satisfagan $0 < c_1 < c_2 < 1$. Tiene dos etapas, la primera consiste en empezar con un valor de α y aumenta hasta encontrar otro tamaño de α aceptable; en el último caso invoca una función llamada **zoom** que disminuye el tamaño de α en el intervalo hasta que encuentra otro aceptable.

Función zoom

Algorithm 4 Zoom

$\phi(\alpha_j) = f(x + \alpha d)$
$\phi(0) = f(x)$
if $\phi(\alpha_j) > \phi(0) + c_1 \alpha_j \phi'(0) \geq \alpha(\alpha_{low})$ **then**
    $\alpha_{high} = \alpha_j$
else
    g = $\phi'(\alpha_j)$
    if $|\phi'(\alpha_j)| \leq -c_2 \phi'(0)$ **then**
        $\alpha_* = \alpha_j$
    if $\phi'(\alpha_j)(\alpha_{high} - \alpha_{low}) \geq 0$ **then**
        $\alpha_{high} = \alpha_{low}$
    $\alpha_{low} = \alpha_j$

Algorithm 5 Line Search

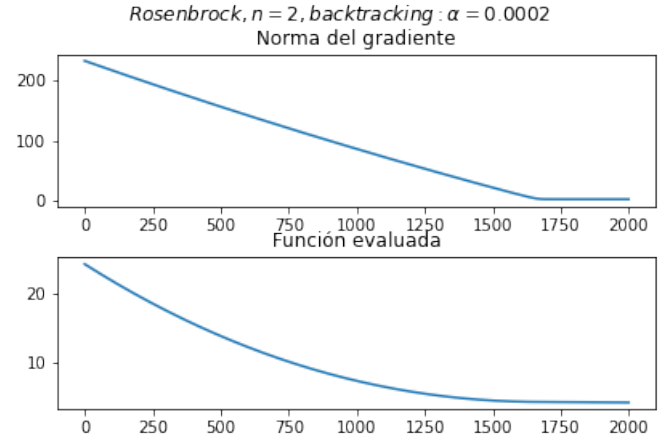
$\alpha_0 = 0, \alpha_{max} > 0, \alpha_1 \in (0, \alpha_{max}), i = 1$
g = $f(x + \alpha_0 d)$
if $g > f(x) + c_1 \alpha_1 f'(x) || \phi(\alpha_i) > \geq \phi(\alpha_{i-1})$ **then**
    $\alpha^* = zoom(\alpha_{i-1}, \alpha_i)$
Evaluar $f'(x + \alpha^* d)$
if $|f'(x + \alpha d)| \leq -c_2 f'(x)$ **then**
    Return: $\alpha^* = \alpha_i$
if $f(x + \alpha d) \geq 0$ **then**
    Return: $\alpha^* = zoom(\alpha_i, \alpha_{i-1})$
$\alpha_{i+1} \in (\alpha, \alpha_{max})$

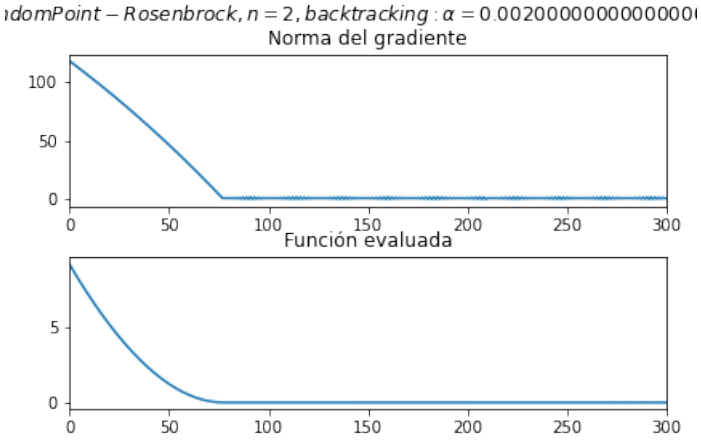
Resultados

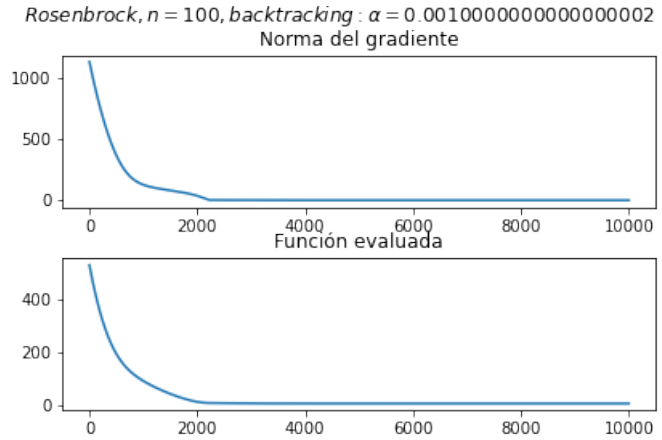
A continuación se muestran los resultados de los tres algoritmos, en cada uno se especifica la función y el α que encontró el método.

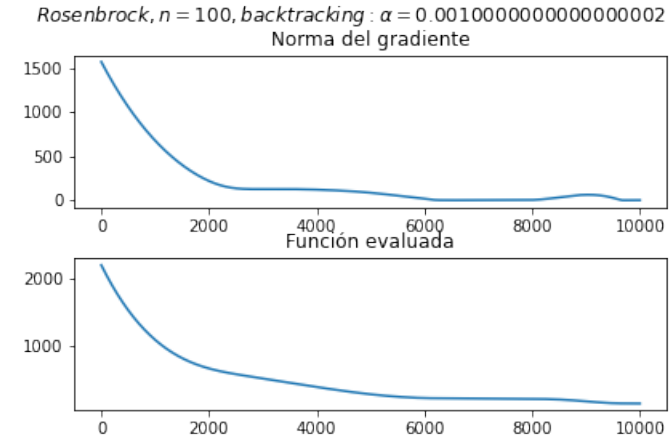
Backtracking

El mínimo que se encontró con este método y con una inicialización de $[-1{,}2, 1]$, fue: $[-1{,}0377, 1{,}0831]$, y con la inicialización aleatoria: $[0{,}9554, 0{,}9151]$, y sabemos que el óptimo global es $[1, 1]$.

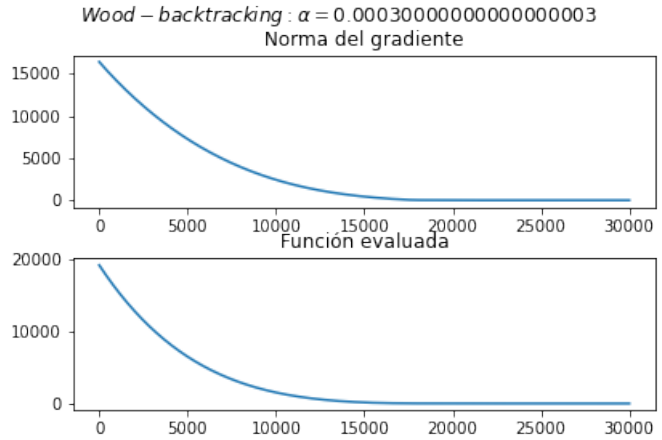


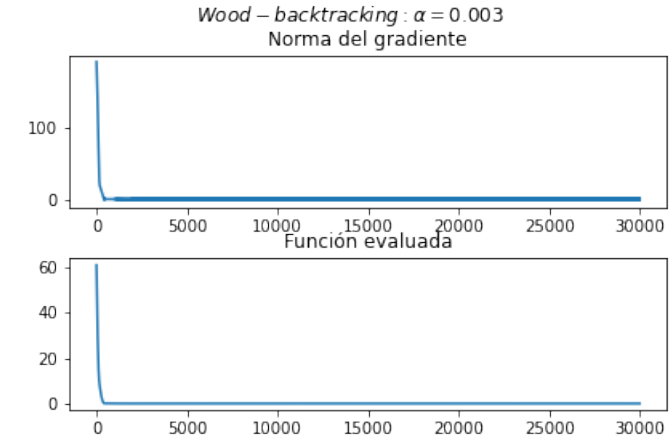




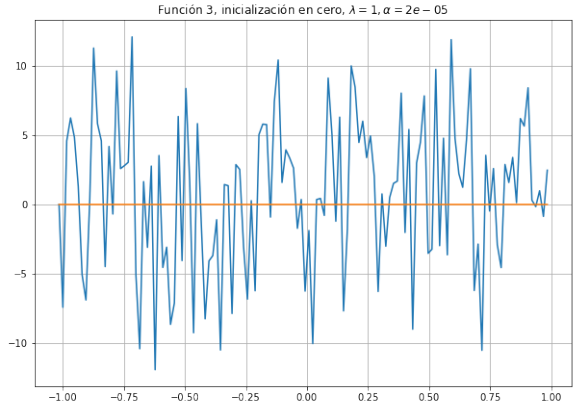


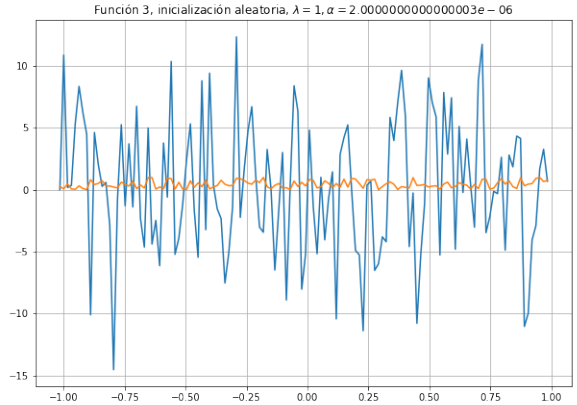
Para la función de Wood, el mínimo se encuentra en $[1, 1, 1, 1]$ y se encontró con la inicialización de $[-3, -1, -3, -1]$, fue de $[-1{,}1221, 1{,}2667, 0{,}8333, 0{,}6948]$. Sin embargo, con la inicialización aleatoria, se encontró un mejor resultado: $[0{,}9898, 0{,}9739, 1{,}01208, 1{,}0241]$.

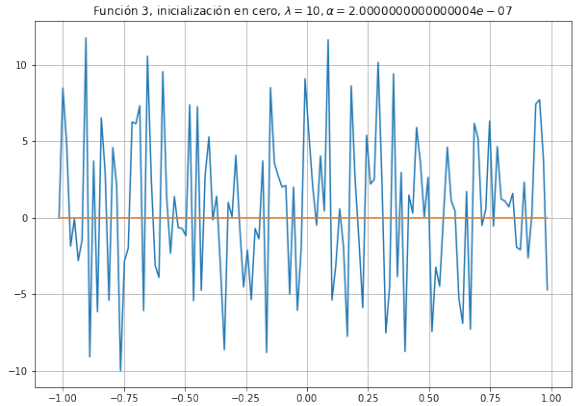


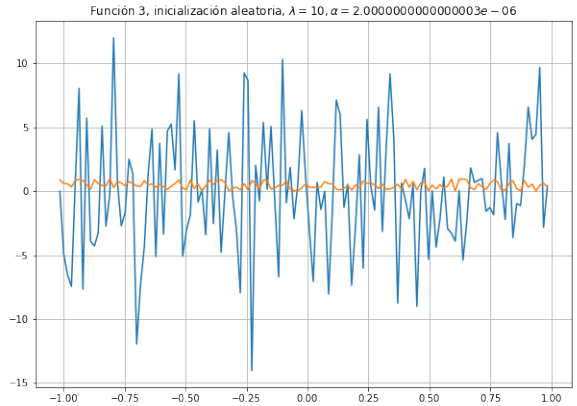


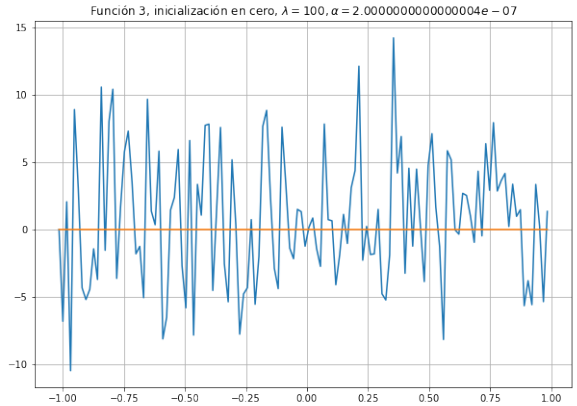
Para la última función, pese al cambio de parámetros el óptimo se bajaba al vector de ceros.

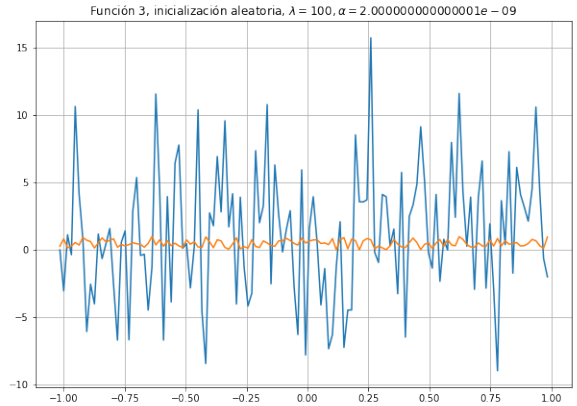






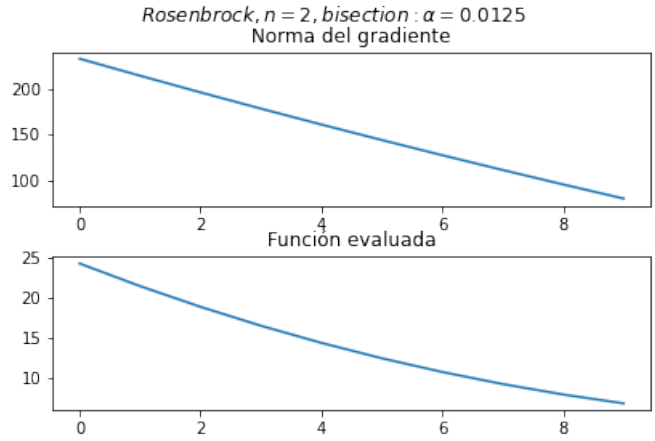


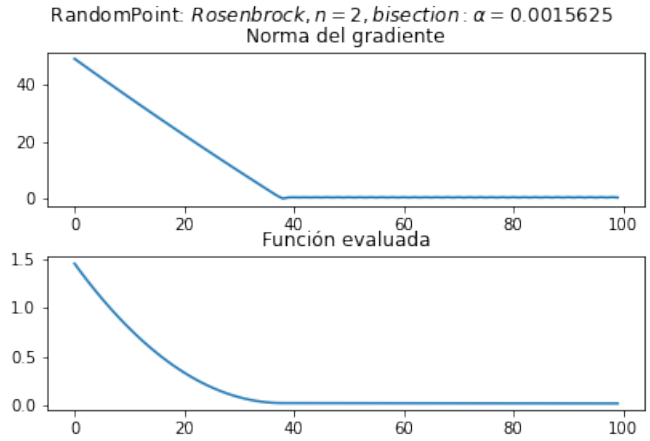




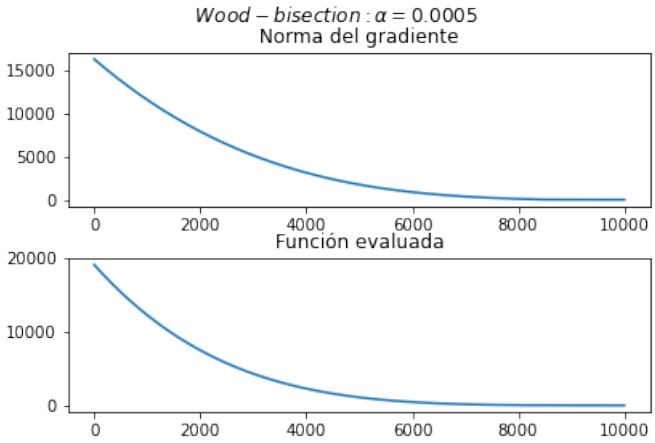
Bisección o interpolación

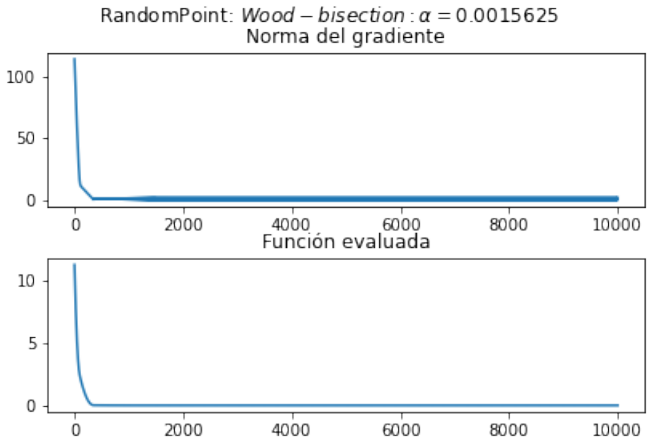
Este método mejoró con la inicialización de $[-1{,}2, 1]$, se obtuvo: $[-1{,}0849, 1{,}0485]$. Y con la inicialización aleatoria: $[0{,}8549, 0{,}7318]$ los resultados empeoraron.

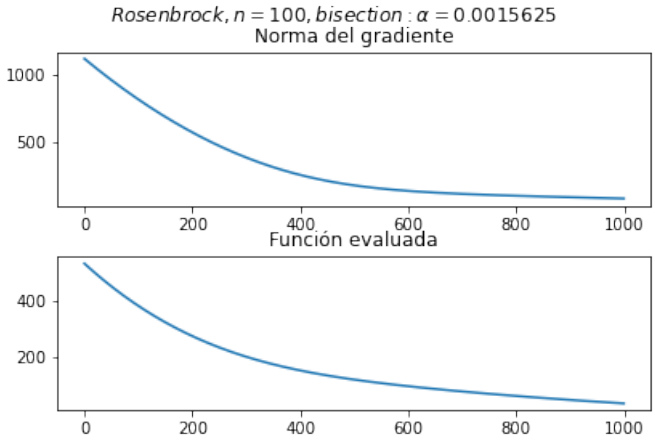


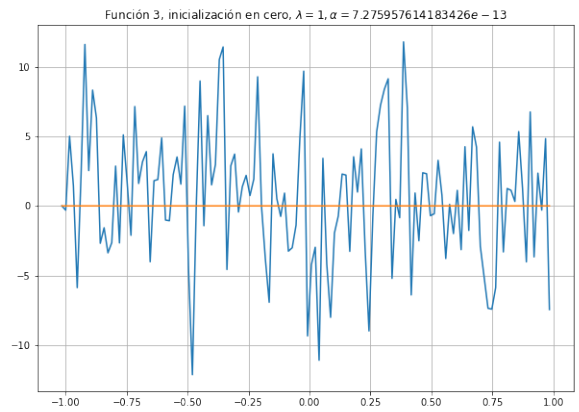


En el caso de la función de Wood, para la inicialización $[-3, -1, -3, -1]$, se obtuvo: $[-0{,}8149, 0{,}6903, -0{,}7903, 0{,}6556]$ y mejoró mucho con la inicialización aleatoria: $[0{,}9987, 0{,}9974, 1{,}0011, 1{,}002]$.

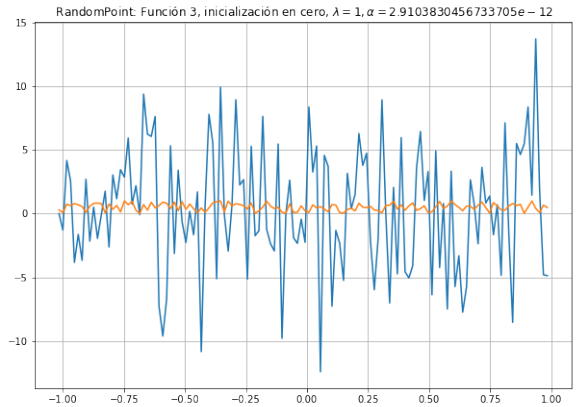


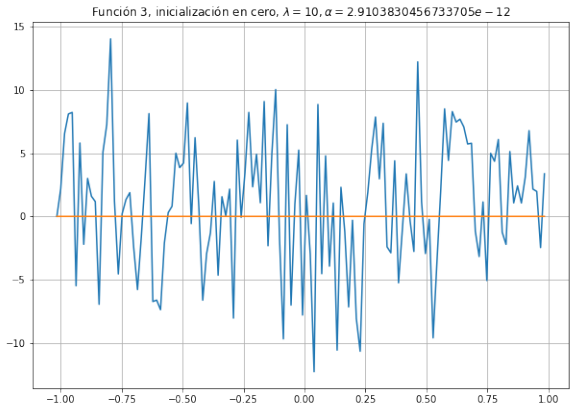


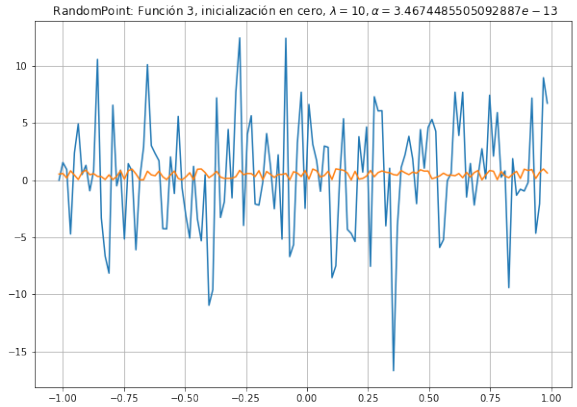


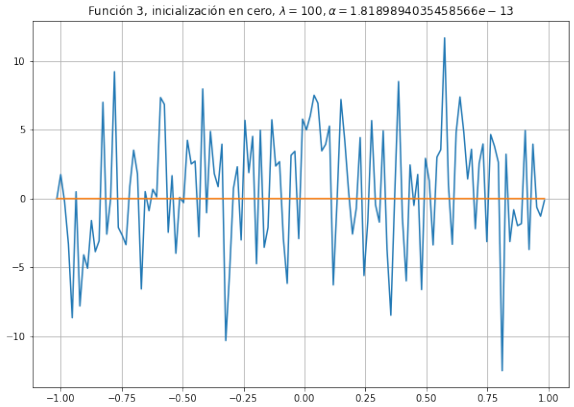


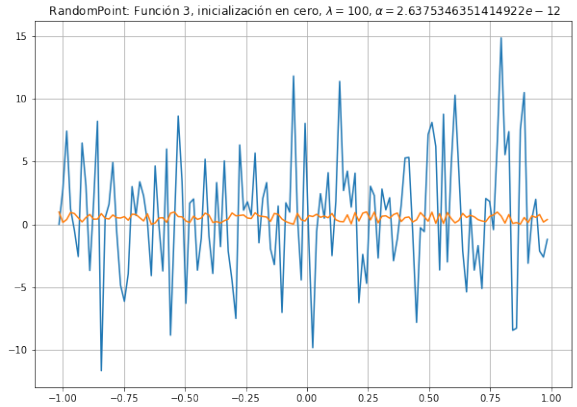
En el caso de la última función, también el punto óptimo era el eje x.





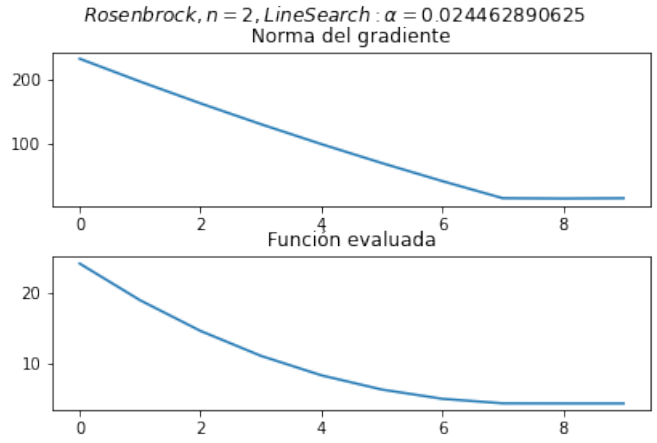


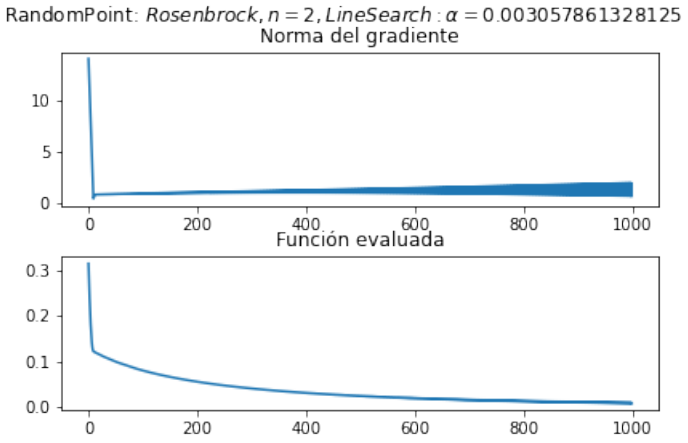


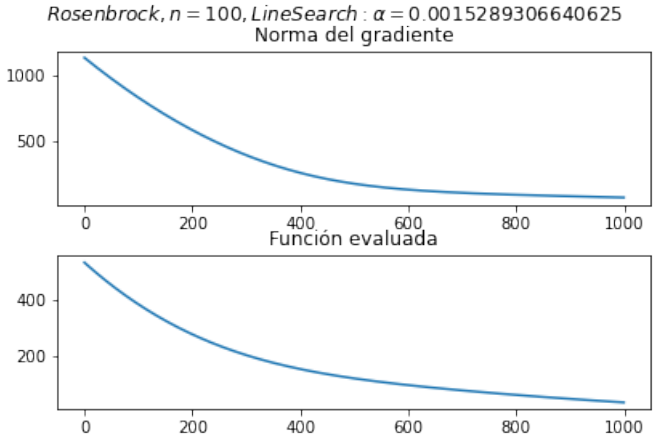


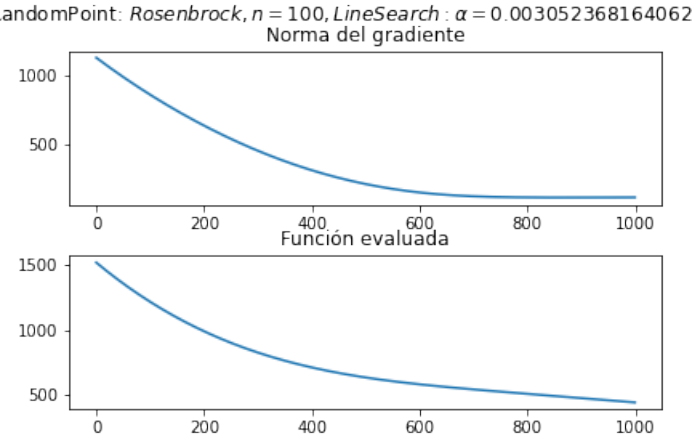
Line Search

Para la primera función de Rosenbrock, se obtuvo con la inicialización complicada: $[-1{,}0160, 1{,}0676]$ y con la inicialización aleatoria: $[0{,}9171, 0{,}8456]$.

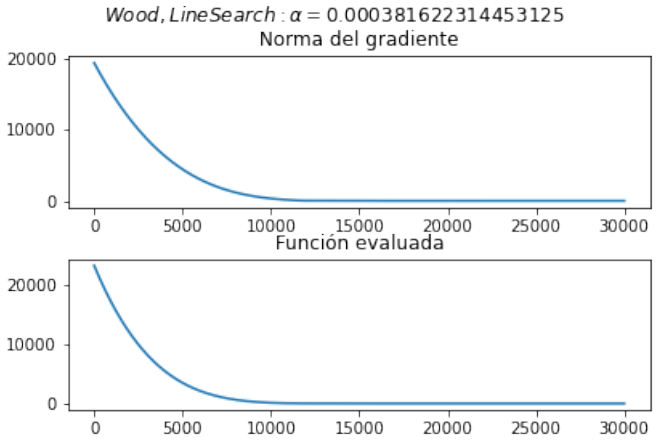


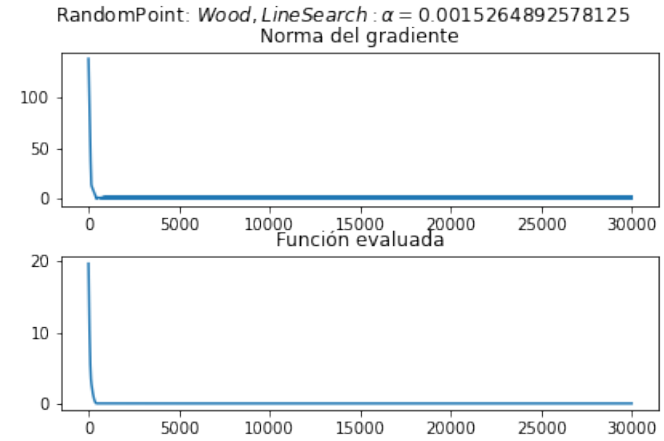




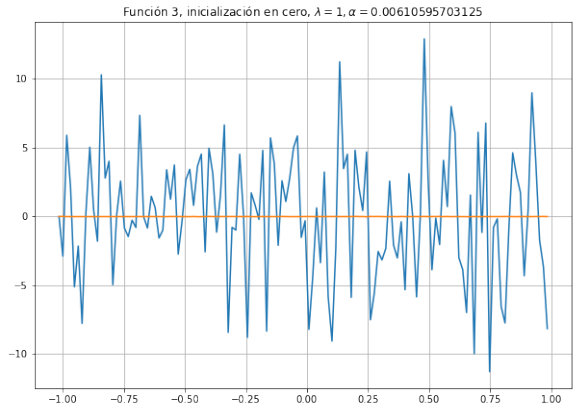


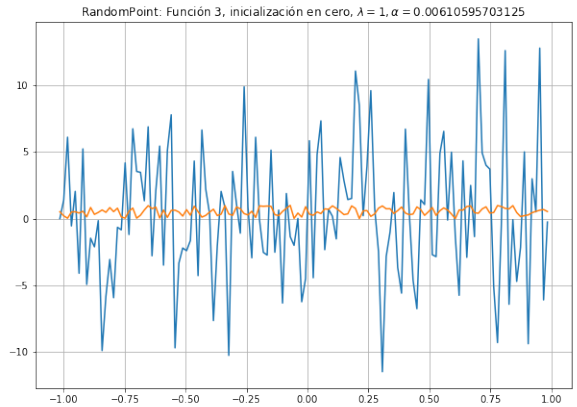
Para la función de Wood se obtuvo: $[-0{,}7495, 0{,}5717, 1{,}1706, 1{,}3719]$ y con la aleatoria: $[0{,}9987, 0{,}9974, 1{,}0011, 1{,}0024]$.

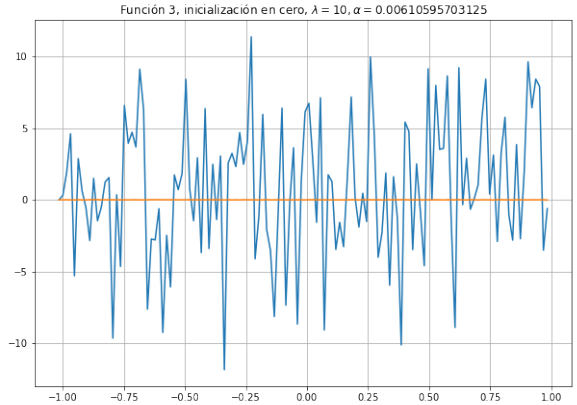


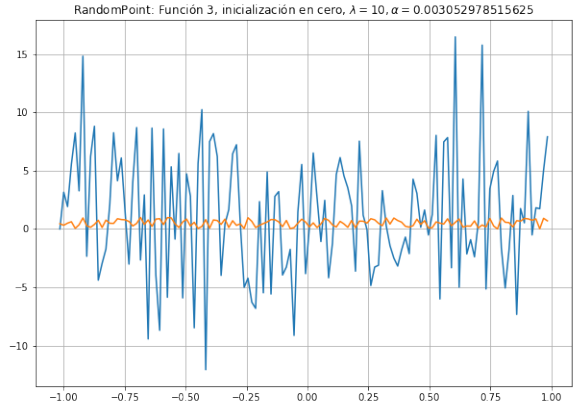


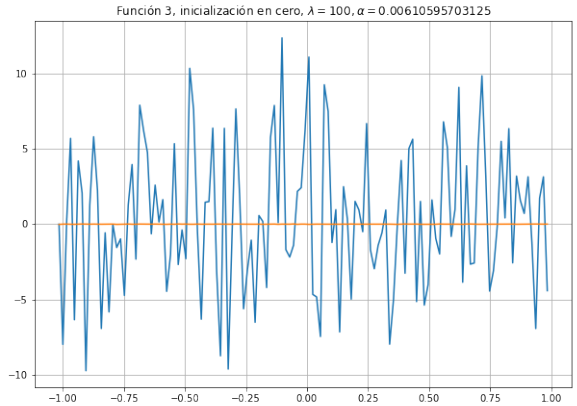
También con este método, el óptimo se quedaba en ceros.

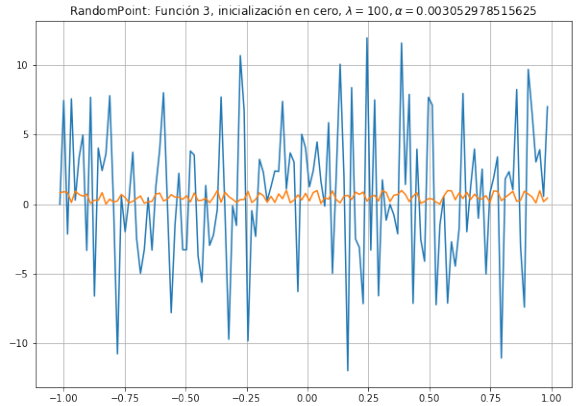












Conclusiones

Se observó que los óptimos cuyas inicializaciones empezaban de forma aleatoria tenían mejores resultados en la mayoría de los casos. Sin embargo, no se obtuvo un mejor método en general. Una desventaja respecto al gradiente descedente con el paso fijo es que es más fácil sólo cambiar el parámetro de α con el algoritmo anterior, que tener más parámetros por cambiar, aunque haya unos más usados que otros como $c_1 = 10e - 4$, es más complicado con más parámetros. No obstante, los resultados mejoraron respecto al tamaño de paso fijo utilizado anteriormente.

Referencias