Tarea 4: Steepest Descent

Optimización I

Maestría en Computación

Centro de Investigación en Matemáticas

Esteban Reyes Saldaña
esteban.reyes@citmat.mx

19 de abril de 2021

Resumen

En esta tarea se implementó el método de descenso gradiente usando búsqueda en línea con backtracking y el método de bisección. Se presenta a continuación una descripción general del método, así como el pseudocódigo de los mpetodos implementados. En los resultados se incluyen pruebas para diferentes funciones tanto como para los puntos iniciales sugeridos por el Profesor como puntos iniciales aleatorios. Finalmente se incluyen algunas conclusiones observadas a partir de la experimentación.

---

1 Introducción

El objetivo general es encontrar los mínimos locales de una función $f : \mathbb{R}^n \to \mathbb{R}$ tal que $f(x) \in \mathcal{C}^1$ de manera iterativa. El algoritmo general se ve como

1. Eligа un punto inicial x_0 y haga $k = 0$.
2. Mientras no converja
   (a) Encuentre x_{k+1} tal que $f(x_{k+1}) < f(x_k)$
   (b) $k \to k + 1$
3. Regrese $x^* = x_k$.

Definición 1.1. Sea $f : \mathbb{R}^n \to \mathbb{R}$ una función continuamente diferenciable. El gradiente de f en el punto p está dado por

$$\nabla^T f(p) = \left[\frac{\partial f(p)}{\partial x_1}, \frac{\partial f(p)}{\partial x_2}, \ldots, \frac{\partial f(p)}{\partial x_n}\right]$$

En clase se vieron algunos resultados sobre el gradiente.

Teorema 1.1. Sea $f : \mathbb{R}^n \to \mathbb{R}$ una función continuamente diferenciable. Entonces para toda $x \in dom(f)$, $\nabla f(x)$ es perpendicular al conjunto de nivel

$$S = \{x \in \mathbb{R}^n | f(x) = c, c \text{ constante.}\}$$

Teorema 1.2. Sea $f : \mathbb{R}^n \to \mathbb{R}$ una función continuamente diferenciable. Entonces la dirección donde $f(x)$ crece más rápido es $\nabla f(x)$.

Corolario 1.1. Bajo las condiciones del Teorema (1.2), $f(x)$ decrece más rápido en la dirección $-\nabla f(x)$.

Sin embargo, existen algoritmos no monótonos en los que $f(x)$ no decrece en cada paso si no después de un número m de iteraciones, esto es

$$f(x_{k+1} < f(x_{k-j}))$$

para algún $j \in \{0, 1, \ldots, M\}$ con $M = m - 1$ si $k \geq m - 1$ y $M = k$ en otro caso. Así que ahora surgen tres preguntas

1. ¿Cómo elegir x_0?
2. ¿Qué criterio de paro usar?
3. ¿Cómo actualizar x_{k+1}?

Definición 1.2. Una **dirección de descenso** $d \in \mathbb{R}^n$ para $f \in \mathcal{C}^1$ es un vector tal que

$$f(x + td) < f(x)$$

para $t \in (0, T)$. Es decir, permite que el punto x más cerca al mínimo local x^* de la función objetivo $f : \mathbb{R}^n \to \mathbb{R}$.

Teorema 1.3. Si $g(x)^T d < 0$ entonces d es una dirección de descenso.

Observación. La dirección

$$d_k = -g(x_k)$$

es la elección más obvia de una dirección de búsqueda.

2 Método

2.1 Steepest Descent

Este es un método de búsqueda en línea que usa la dirección $d_k = -g(x_k)$ para moverse en cada iteración. El esquema general es

1. Dé un vector inicial x_0.
2. Haga $k = 0$ y $g_0 = \nabla f(x_0)$
3. Mientras $||g_k|| \neq 0$,
   (a) $\alpha_k = \arg \min_{\alpha > 0} f(x_k - \alpha g_k)$
   (b) $x_{k+1} = x_k - \alpha_k g_k$
   (c) $g_{k+1} = \nabla f(x_{k+1})$
   (d) $k \to k + 1$

Se debe seleccionar un tamaño de paso α_k tal que reduce suficientemente a $f(x)$ y al mismo tiempo se requiere que sea eficiente. Para ello se plantea encontrar el tamaño de paso como encontrar una aproximación del problema de optimización previo. La búsqueda en línea se hace en dos pasos iterativos

1. Una fase donde se acota el intervalo que contiene el tamaño de paso deseado,
2. una bisección o interpolación que calcula un "buen´ ´ tamaño de paso con dicho intervalo.

2.2 Condición de Armijo

Un tamaño de decrecimiento suficiente se puede medir por la siguiente ecuación

$$f(x_k + \alpha d_k) \leq f(x_k) + c_1 \alpha \nabla f_k^T d_k,$$

para alguna constante $c_1 \in (0, 1)$.
Una condición suficiente de descenso no es suficiente por sí mismoa para asegurar que el algoritmo progresa de manera adecuada ya que se satisiface para valores de α suficientemente pequeños.
Para obtener pasos más largos, se da una segunda condición, llamada condición de curvatura.

2.3 Condiciones de Wolfe

$$\begin{aligned} f(x_k + \alpha d_k) &\leq f(x_k) + c_1 \alpha \nabla f_k^T d_k, \quad (1) \\ \nabla f(x_k + \alpha d_k)^T d_k &\geq c_2 \nabla f_k^T d_k \quad (2) \end{aligned}$$

con $0 < c_1 < c_2 < 1$.

2.4 Pseudocódigo

2.4.1 Búsqueda con BackTracking

Input: $\hat{\alpha}$, $\rho \in (0,1)$, $c_1 \in (0,1)$
Output: Tamaño de paso α_k

Haga $alpha = \hat{\alpha}$
$inum = 0$
while $f(x_k + \alpha d_k) > f(x_k) + c_1 \alpha \nabla f_k^T d_k$ **do**
    $alpha \to \rho\alpha$
end while
Regresa $\alpha_k = \alpha$

2.4.2 Búsqueda en Línea con Bisección

Input: α_0, $0 < c_1 < c_2 < 1$
Output: Tamaño de paso α_k

Haga $alpha = 0$, $\beta = \infty$, $\alpha^i = \alpha_0$
$inum = 0$
while $f(x_k + \alpha_k d_k) > f(x_k) + c_1 \alpha_k \nabla f_k^T d_k$ **o** $\nabla f(x_k + \alpha_k d_k)^T d_k < c_2 \nabla f(x_k)^T d_k$ **do**
    if $f(x_k + \alpha_k d_k) > f(x_k) + c_1 \alpha_k \nabla f_k^T d_k$ **then**
        $\beta = \alpha_k$
        $\alpha_k = \dfrac{\alpha + \beta}{2}$
    else
        $\alpha = \alpha_k$
        if $\beta = \infty$ **then**
            $\alpha_k = 2\alpha$
        else
            $\alpha_k = \dfrac{\alpha + \beta}{2}$
        end if
    end if
end while
Regresa α_k

2.5 Función de Rosembrock

$$f(x) = \sum_{i=1}^{N-1} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$$

donde $x = [x_1, \ldots, x_N]^T \in \mathbb{R}^N$.

Gradiente. Primero notemos que para $2 \leq j \leq N-1$ se puede reescribir $f(x)$ como

$$\begin{aligned} f(x) &= \sum_{i=1}^{j-2} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2] \\ &\quad + [100(x_j - x_{j-1}^2)^2 + (1 - x_{j-1})^2] \\ &\quad [100(x_{j+1} - x_j^2)^2 + (1 - x_j)^2] \\ &\quad + \sum_{i=j+1}^{N-1} [100(x_{i-1} - x_i^2)^2 - (1 - x_i)^2]. \end{aligned}$$

Notemos que el primer y útlimo término son constantes con respecto a x_j, entonces se anularán al calcular la correspondiente derivada parcial.
Para D_1 se tiene que

$$\begin{aligned} \frac{\partial}{\partial x_1} f(x) &= 100(2)(x_2 - x_1^2)(-2x_1) - 2(1 - x_1) \\ &= -400(x_2 - x_1^2)x_1 - 2(1 - x_1). \end{aligned}$$

Para $D_j \in \{2, \ldots, N-1\}$,

$$\begin{aligned} \frac{\partial}{\partial x_j} f(x) &= [200(x_j - x_{j-1}^2)] + [200(x_{j+1} - x_j^2)(-2x_j)] \\ &\quad -[2(1 - x_j)] \\ &= 200(x_j - x_{j-1}^2) - 400(x_{j+1} - x_j^2)x_j \\ &\quad -2(1 - x_j) \end{aligned}$$

Para D_N,

$$\frac{\partial}{\partial x_N} f(x) = 200(x_N - x_{N-1}^2).$$

De lo anterior tenemos que

$$\nabla f(x) = \begin{bmatrix} -400(x_2 - x_1^2)x_1 - 2(1 - x_1) \\ \vdots \\ 200(x_j - x_{j-1}^2) - 400(x_{j+1} - x_j^2)x_j - 2(1 - x_j) \\ \vdots \\ 200(x_N - x_{N-1}^2). \end{bmatrix}$$

Hessiano. Para calcular el Hessiano, se debe considerar como casos particulares, aquellos que se apliquen a $\dfrac{\partial f(x)(x)}{\partial x_1}$ y $\dfrac{\partial f(x)(x)}{\partial x_N}$ (como se observó en el gradiente).
Para $D_{x_k x_1}$,

$$\frac{\partial^2 f(x)}{\partial x_k \partial x_1} = \begin{cases} 1200x_1^2 - 400x_2 + 2 & si \quad k = 1 \\ -400x_1 & si \quad k = 2 \\ 0 & e.o.c \end{cases}$$

Para $D_{x_k x_j}$ con $j \in \{2, \ldots, N-1\}$,

$$\frac{\partial^2 f(x)}{\partial x_k \partial x_j} = \begin{cases} 1200x_j^2 - 400x_2 + 2 & si \quad k = j \\ -400x_j & si \quad k = j+1 \\ -400x_{j-1} & si \quad k = j-1 \\ 0 & e.o.c \end{cases}$$

Para $D_{x_k x_N}$,

$$\frac{\partial^2 f(x)}{\partial x_k \partial x_1} = \begin{cases} -400x_{N-1} & si \quad k = N-1 \\ 200 & si \quad k = N \\ 0 & e.o.c \end{cases}$$

Así que el Hessiano tiene la forma

$$H(x) = \begin{pmatrix} 1200x_1^2 - 400x_2 + 2 & -400x_1 & \cdots & 0 \\ -400x_1 & 1200x_2^2 - 400x_3 + 202 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & -400x_{N-1} & 200 \end{pmatrix}.$$

2.6 Función de Wood

Para $n = 4$ esta función está dada por

$$\begin{aligned} f(x) &= 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 \\ &\quad +(x_3 - 1)^2 + 90(x_3^2 - x_4)^2 \\ &\quad +10{,}1[(x_2 - 1)^2 + (x_4 - 1)^2] \\ &\quad +19{,}8(x_2 - 1)(x_4 - 1). \end{aligned}$$

Gradiente. Derivando respecto a cada entrada obtenemos

$$\begin{aligned} \frac{\partial f(x)}{\partial x_1} &= 400(x_1^2 - x_2)x_1 + 2(x_1 - 1) \\ \frac{\partial f(x)}{\partial x_2} &= -200(x_1^2 - x_2) + 20{,}2(x_2 - 1) \\ &\quad +19{,}8(x_4 - 1) \\ \frac{\partial f(x)}{\partial x_3} &= 2(x_3 - 1) + 360(x_3^2 - x_4)x_3 \\ \frac{\partial f(x)}{\partial x_4} &= -180(x_3^2 - x_4) + 20{,}2(x_4 - 1) \\ &\quad +19{,}8(x_2 - 1). \end{aligned}$$

Hessiano

$$H_f(x) = \begin{bmatrix} 1200x_1^2 - 400x_1 + 2 & -400x_1 & 0 & 0 \\ -400x_1 & 220{,}2 & 0 & 19{,}8 \\ 0 & 0 & 1080x_3^2 - 360x_4 + 2 & -360x_3 \\ 0 & 19{,}8 & -360x_3 & 2020{,}2. \end{bmatrix}$$

2.7 Función f_3

Sea $\eta \sim \mathcal{N}(0, \sigma)$, $\lambda, \sigma > 0$. Se defie $f_3(x)$ por

$$f(x) = \sum_{i=1}^{n}(x_i - y_i)^2 + \lambda \sum_{i=1}^{n-1}(x_{i+1} - x_i)^2$$

con $y_i = t_i^2 + \eta$, $t_i = \dfrac{2}{n-1}(i-1) - 1$, para $i = 1, 2, \ldots, n$.

Gradiente. Para D_{x_1} tenemos

$$\frac{\partial f(x)}{\partial x_1} = 2(x_1 - y_1) - 2\lambda(x_2 - x_1)$$

Para D_j con $j \in \{2, 3, \ldots, n-1\}$,

$$\begin{aligned} \frac{\partial f(x)}{\partial x_j} &= \frac{\partial}{\partial x_j}\sum_{i=1}^{n}(x_i - y_i)^2 + \lambda \frac{\partial}{\partial x_j}\sum_{i=1}^{n-1}(x_{i+1} - x_i)^2 \\ &= \frac{\partial}{\partial x_j}(x_j - y_j)^2 + \lambda \frac{\partial}{\partial x_j}[(x_{j+1} - x_j)^2 \\ &\quad +(x_j - x_{j-1})^2] \\ &= 2(x_j - y_j) - 2\lambda(x_{j+1} - x_j) + 2\lambda(x_j - x_{j-1}). \end{aligned}$$

Para D_{x_n} tenemos

$$\frac{\partial f(x)}{\partial x_1} = 2(x_n - y_n) - 2\lambda(x_n - x_{n-1})$$

Por lo tanto

$$\nabla f(x) = \begin{bmatrix} 2(x_1 - y_1) - 2\lambda(x_2 - x_1) \\ \vdots \\ 2(x_j - y_j) - 2\lambda(x_{j+1} - x_j) + 2\lambda(x_j - x_{j-1}) \\ \vdots \\ 2(x_n - y_n) - 2\lambda(x_n - x_{n-1}) \end{bmatrix}$$

3 Resultados

3.1 Función de Rosembrock

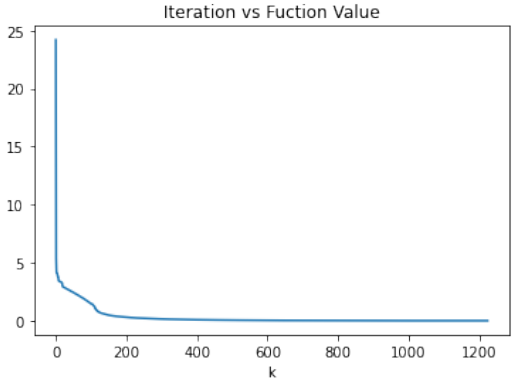
Para $n = 2$ con $x_0 = [1{,}2, 1]$ y parámetros

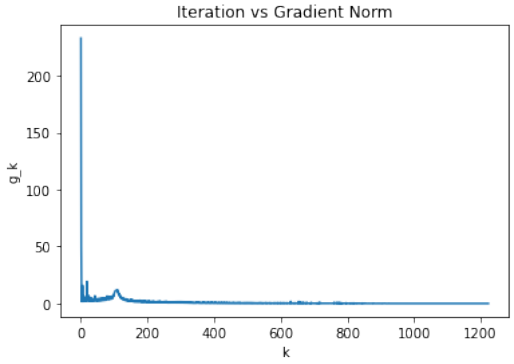
Parámetro	Valor
α	0.1
ρ	0.1
c_1	10^{-4}
c_2	0,9

se obtuvo

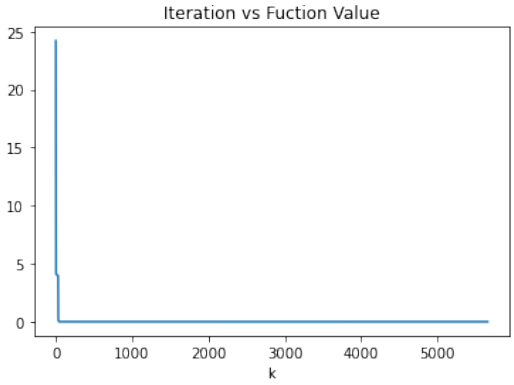
Algoritmo	Iteraciones
BackTracking	1222
Bisección	5654

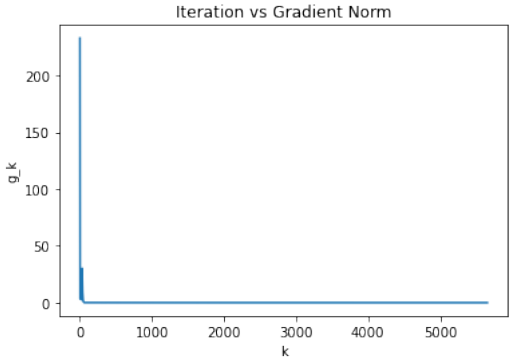
Gráficas para Backtracking





Gráficas para Bisección





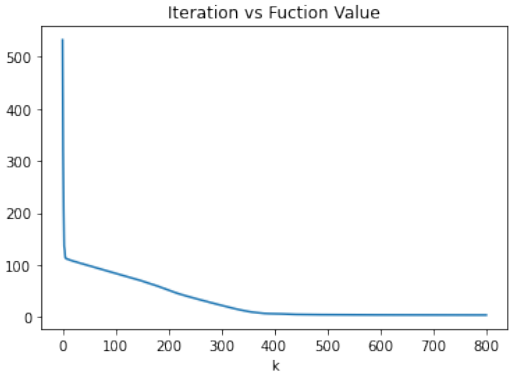
Para $n = 100$ con $x_0 = [-1{,}2, 1, \ldots, -1{,}2, 1]$ y parámetros

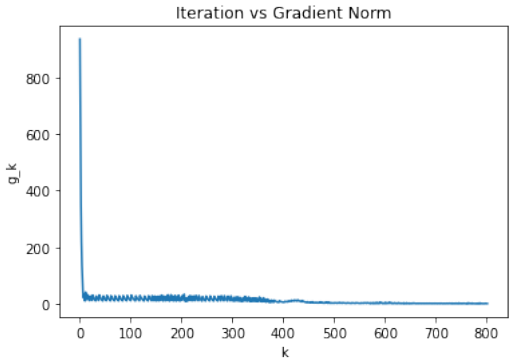
Parámetro	Valor
α	0.1
ρ	0.5
c_1	10^{-4}
c_2	0,9

se obtuvo

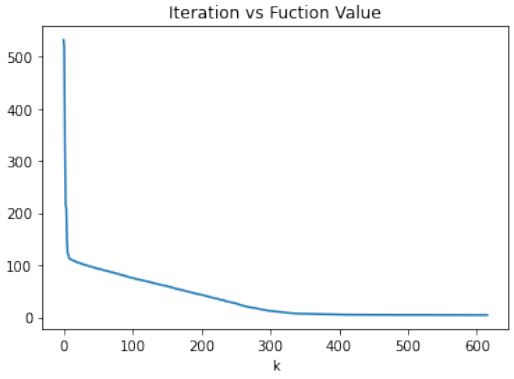
Algoritmo	Iteraciones
BackTracking	800
Bisección	617

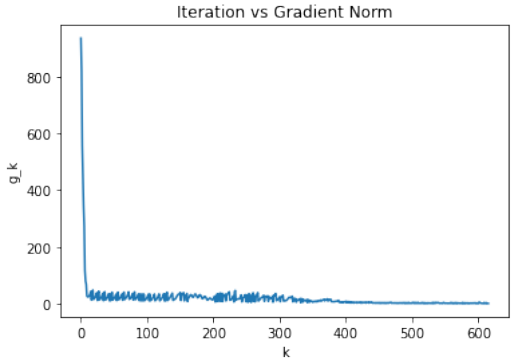
Gráficas para Backtracking





Gráficas para Bisección





Observación. Sin embargo no fue posible que el mínimo correspondiera al punto $[1, \ldots, 1]$. Para ello se introduce un punto inicial aleatorio.

3.1.1 Puntos iniciales aleatorios

Se generaron puntos iniciales usando la distribución normal de media cero y desviación 1. Se repitió el experimento 50 veces
Para $n = 2$ se obtuvo

Algoritmo	Tiempo Promedio	Promedio Iter
BackTracking	0.1238	906.14
Bisección	0.8856	1915.48

Para $n = 100$ se repitió un total de 5 veces. Se observó que al generar un punto inicial aleatorio sí se llegó al máximo global $[1, 1, \ldots, 1]$

Algoritmo	Tiempo Promedio	Promedio Iter
BackTracking	25.69	5154.0
Bisección	54.28	2344.2

3.2 Función de Wood

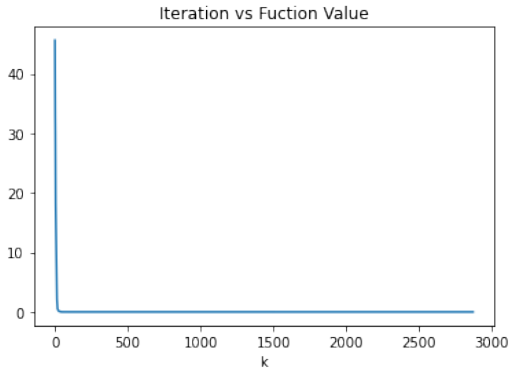
Para $x_0 = [-3{,}0, -1 - 0, -3{,}0, -1{,}0]$ y parámetros

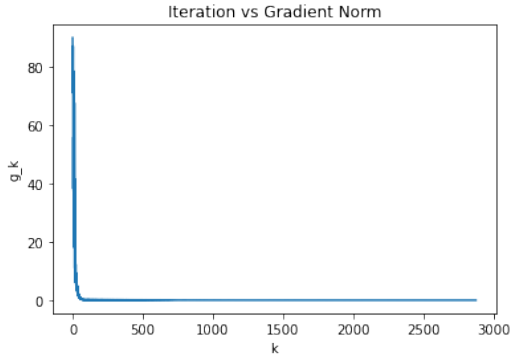
Parámetro	Valor
α	0.9
ρ	0.5
c_1	10^{-4}
c_2	0,9

se obtuvo

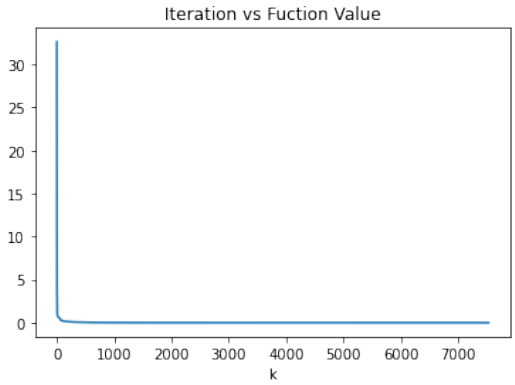
Algoritmo	Iteraciones
BackTracking	2871
Bisección	7527

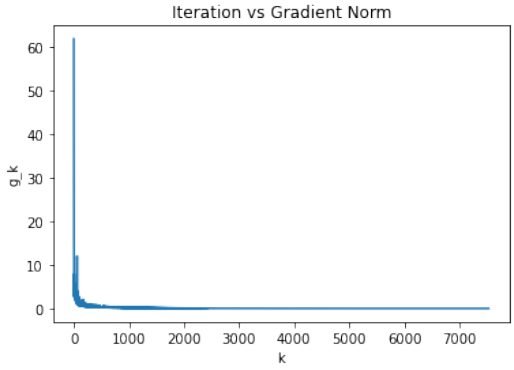
Gráficas para Backtracking





Gráficas para Bisección





3.2.1 Puntos iniciales aleatorios

Se generaron puntos iniciales usando la distribución normal de media cero y desviación uno. Se repitió el experimento 50 veces y se obtuvo

Algoritmo	Tiempo Promedio	Promedio Iter
BackTracking	0.061	311.66
Bisección	0.184	232.94

3.3 Función f_3

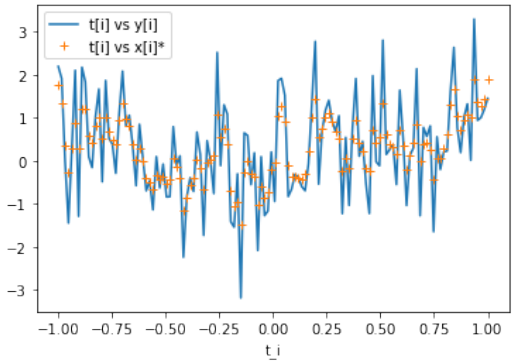
Se generó un vector inicial x_0 y *eta* de tamaño $n = 128$ de manera aleatoria siguientdo una distribución $\mathcal{N}(0, \sigma)$. Por motivos de practicidad se fijó $\sigma = 1$. Para $\lambda = 1$ y los parámetros

Parámetro	Valor
α	0.2
ρ	0.5
c_1	10^{-4}
c_2	0,9

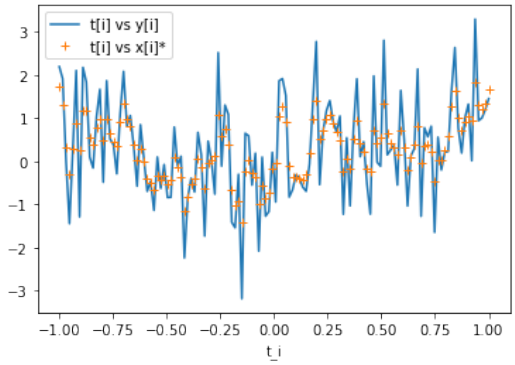
se obtuvo

Algoritmo	Iteraciones
BackTracking	2441
Bisección	8

Gráficas para Backtracking



Gráficas para Bisección



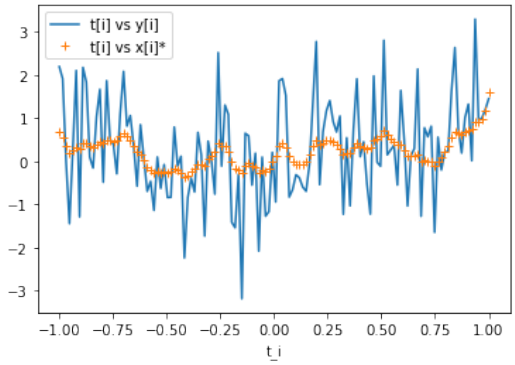
Para $\lambda = 10$ y los parámetros

Parámetro	Valor
α	0.2
ρ	0.5
c_1	10^{-4}
c_2	0,9

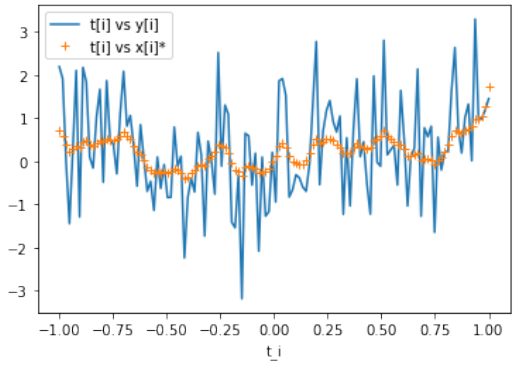
se obtuvo

Algoritmo	Iteraciones
BackTracking	19
Bisección	17

Gráficas para Backtracking



Gráficas para Bisección



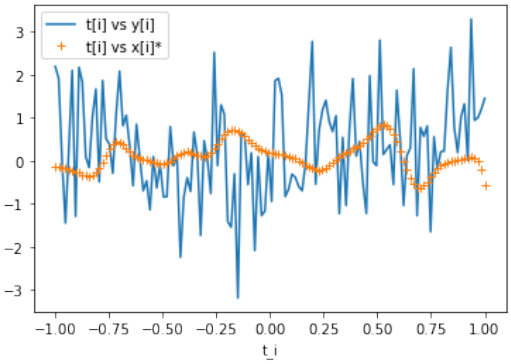
Para $\lambda = 1000$ y los parámetros

Parámetro	Valor
α	0.2
ρ	0.5
c_1	10^{-4}
c_2	0,9

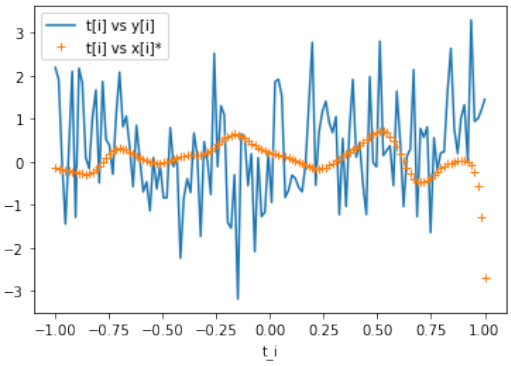
se obtuvo

Algoritmo	Iteraciones
BackTracking	13
Bisección	18

Gráficas para Backtracking



Gráficas para Bisección



3.3.1 Puntos iniciales aleatorios

Se generaron puntos iniciales usando la distribución normal de media cero y desviación uno. Se repitió el experimento 20 veces y se usó $\lambda = 10$. Los resultados fueron

Algoritmo	Tiempo Promedio	Promedio Iter
BackTracking	0.1075	128
Bisección	0.4285	128

4 Conclusiones

4.1 Rosembrock

Para esta función se observa un mejor desempeño en el algoritmo de backtracking que en el de bisección (con respecto al número de iteraciones). De hecho, en las gráficas mostradas el decrecimiento del valor $f(x_k)$ y $g(x_k)$ es más suave para bactracking que para bisección.
Además, para $n = 100$ y con el punto inicial sugerido, no fue posible llegar al mínimo $[1, 1, \ldots, 1]^T$. Pero cuando se utilizó un vector inicial aleatorio, el método sí fue capaz de llegar al mínimo de unos.
Para la experimentación aleatoria se observa, de nuevo, un mejor desempeño en el método de backtracking que en el de bisección.

4.2 Wood

En esta función, un cambio sutil en los parámetros, hacía llegar el punto inicial sugerido a otros mínimos locales. Se observó, de nuevo, que backtracking convergió en un menor número de iteraciones y en las gráficas se observa un decrecimiento tanto de $f(x_k)$ como de $||g(x_k)||$ rápido pero para el algoritmo de bisección se ve menor suavidad.

4.3 Función f_3

En esta función, el valor η está causando ruido en la "parábola´´ descrita por $f(x)$. Observemos que para $\lambda = 1$ el error de aproximación es menor que para $\lambda = 10, 1000$. En general, cuando se fijó x_0, la convergencia se logró en menos de 20 iteraciones.