An efficient gradient-free line search
Optimización I

Luis Ramón Guajardo — Lic. en Computación Matemática
Jafet Castañeda — 3 de junio de 2023

Resumen

En este proyecto presentamos un método de búsqueda lineal propuesto por los matemáticos *Arnold Neumaier* y *Morteza Kimiaei* de la Facultad de Matemáticas en la Universidad de Viena, Austria. El algoritmo (CLS) es una mejora del algoritmo de búsqueda lineal a partir de las condiciones de GOLDSTEIN, en donde cada tamaño de paso es controlado para evitar pasos convergentes a cero o pasos que interrumpen la condición de descenso. Implementamos el algoritmo (PYTHON) en diferentes escenarios y lo comparamos con otros algoritmos de descenso y particularmente aquellos que se basan en el método de búsqueda lineal o Máximo Descenso.

1. Introducción y definiciones previas

En este documento se presenta un método de búsqueda en línea para resolver un problema de optimización sin restricciones, i.e.

$$\min_{x\in\mathbb{R}^n} f(x)$$

Este método propone una nueva serie de condiciones de descenso para la elección del tamaño de paso, en las que se requiere únicamente información de **la función y del gradiente en el iterado actual** para determinar un tamaño de paso adecuado para la búsqueda en línea, a diferencia de otros métodos comunes, en donde para cada iteración se requieren evaluaciones adicionales del gradiente en otros iterados o en ocasiones hasta del Hessiano de la función.

Primeramente, es importante mencionar que en el problema de optimización que abordemos, requeriremos que la función f cumpla con las siguientes condiciones:

- f es acotada inferiormente, es decir,
$$\inf_{\alpha\geq 0} f(x(\alpha)) > -\infty$$
- f es diferenciable y continua en $\mathbb{R}^n$ y su gradiente es Lipschitz continuo con constante de Lipschitz γ, es decir, se tiene que
$$\|g(x) - g(y)\| \leq \gamma\|x - y\|, \quad \gamma > 0,\ x, y \in \mathbb{R}^n$$
donde $g(x) = \nabla f(x)^T$

Cabe resaltar que la norma $\|\cdot\|$ considerada en las condiciones anteriores es tal que satisface la desigualdad de Cauchy-Schwarz.

Otra definición que se utilizará es la de un minimizador local fuerte, el cual definimos como un $\hat{x} \in \mathbb{R}^n$ tal que f es continua y diferenciable en una vecindad centrada en $\hat{x}$ y además se cumple que $g(\hat{x}) = 0$ y el Hessiano evaluado en ese punto, es decir, $H(\hat{x}) = \nabla^2 f(\hat{x})$ es una matriz positiva definida.

En cuanto a notación, se manejará una similar a la vista en las presentaciones de clase y en las tareas. Para $k = 0, 1, 2, \ldots$ y una secuencia de puntos $x_0, x_1, x_2, \ldots$, consideramos

$$f_k = f(x_k)$$

$$g_k = g(x_k)$$

y

$$s_k = x_{k+1} - x_k$$
$$y_k = g_{k+1} - g_k$$

donde s_k corresponden a lo que llamaremos **pasos** entre iteraciones.

2. Descripción general del algoritmo

A continuación daremos una breve sinopsis del algoritmo **CLS (Curved Line Search)** presentado por *Arnold Neumaier* y *Morteza Kimiaei*. A diferencia de otros algoritmos de búsqueda lineal, CLS se caracteriza por implementar una nueva condición de descenso suficiente para el tamaño de paso. En la siguiente sección se profundiza dicha condición, pero en pocas palabras consiste en una **variación de la condición de Goldstein**. Recordemos que la condición de Goldstein nos garantiza que el tamaño de paso α alcance un descenso suficiente y a la vez que no sea muy corto. La condición está dada por la siguiente pareja de desigualdades

$$f(\boldsymbol{x}_k) + c_2\alpha\nabla f_k^T \boldsymbol{d}_k \leq f(\boldsymbol{x}_k + \alpha\boldsymbol{d}_k) \leq f(\boldsymbol{x}_k) + c_1\alpha\nabla f_k^T \boldsymbol{d}_k \tag{2.1}$$

con $0 < c_1 < c_2 < 1$ (Nótese que es una versión ligeramente diferente a la vista en el curso). Veremos que la nueva condición de descenso es más fácil de satisfacer que (2.1), pero a la vez cumple con el objetivo de lograr un descenso en la función y evitar tamaños de paso muy pequeños o muy grandes.

Una gran diferencia de CLS es que es un algoritmo eficiente que no hace uso de evaluaciones adicionales del gradiente, por lo que, su complejidad a comparación de otros métodos es menor.

Además, dicho algoritmo presenta un comportamiento regular en regiones fuertemente no convexas, aunque en ciertos casos acepta una mayor cantidad de tamaños de pasos a comparación de Goldstein o Wolfe. En cuanto a funciones cuadráticas convexas, se garantiza convergencia (para hallar el tamaño de paso) en a lo más **2** iteraciones.

El algoritmo del articulo *An efficient gradient-free line search* fue implementando en Matlab. En nuestro proyecto realizaremos la implementación en Python.

3. Desarrollo

El algoritmo presenta una **condición de suficiente descenso** (**SDC** por sus siglas en inglés) dada por

$$\mu(\alpha)|\mu(\alpha) - 1| \geq \beta \tag{3.1}$$

para un $\beta > 0$ fijo, donde

$$\mu(\alpha) := \frac{f(x(\alpha)) - f(x)}{\alpha\nabla f(x)^T d}$$

es el cociente de Goldstein, con d una dirección de descenso y $x(\alpha) = x + \alpha d$ (este es el caso más común, y con el que hemos trabajado en el curso, de búsqueda en línea recta).

Notemos que la condición (3.1) requiere que $\mu(\alpha)$ no sea cercana a 1, lo cual implicaría que el tamaño de paso α sea muy pequeño, y además que $\mu(\alpha)$ sea lo suficientemente positivo garantizando que $f(x(\alpha)) < f(x)$, por lo que se evitarían también tamaños de paso muy grandes.

En este sentido, notemos que la condición de Goldstein es equivalente a lo siguiente

$$f(x) + c_2\alpha\nabla f(x)^T d \leq f(x(\alpha)) \leq f(x) + c_1\alpha\nabla f(x)^T d$$

$$\Leftrightarrow c_2\alpha\nabla f(x)^T d \leq f(x(\alpha) - f(x) \leq c_1\alpha\nabla f(x)^T d$$
$$\Leftrightarrow c_1 \leq \frac{f(x(\alpha)) - f(x)}{\alpha\nabla f(x)^T d} \leq c_2 \qquad \text{Dado que } \nabla f(x)^T d < 0$$

Es decir, la condición de Goldstein equivale a

$$c_1 \leq \mu(\alpha) \leq c_2,$$

por lo tanto, dada la equivalencia con la condición de Goldstein, la condición (3.1) se cumple al tomar

$$\beta = c_1(1 - c_2) > 0$$

Por otra parte, tomando

$$c_1 = \frac{2\beta}{1 + \sqrt{1 - 4\beta}}$$
$$c_2 = \frac{1 + \sqrt{1 - 4\beta}}{2},$$

la condición (3.1) implica que la condición de Goldstein se cumple, o bien, que se cumple otra condición llamada **condición de rápido descenso** dada como

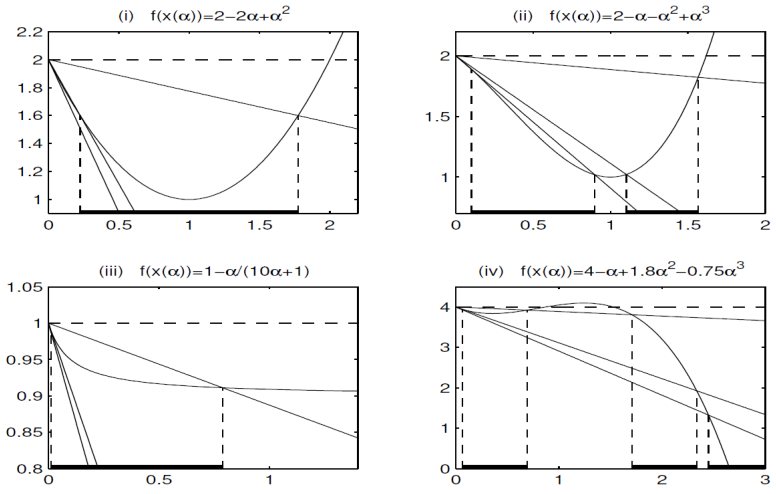
$$\mu(\alpha) \geq c_3 \tag{3.2}$$

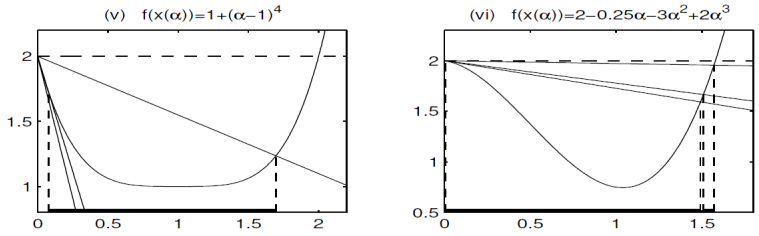
con

$$c_3 = \frac{1 + \sqrt{1 + 4\beta}}{2}$$

Todo lo anterior nos dice que el satisfacer la condición SDC (3.1) nos garantiza que los tamaños de paso elegidos proveerán un decrecimiento adecuado de la función f, al cumplirse alguna de las condiciones de descenso previamente referidas. Más aún, se cumplirá que el rango de tamaños de paso considerados con la condición (3.1) será igual o más grande que el rango considerado por la condición de Goldstein, ya que se incluyen aquellos α que cumplen con (3.2).

A continuación, se muestran ejemplos de algunas gráficas de funciones $f(x(\alpha))$ considerando el parámetro $\beta = 0.1$ y con las líneas con pendientes $c_1\nabla f(x)^T d$, $c_2\nabla f(x)^T d$, $c_3\nabla f(x)^T d$ trazadas. Estos ejemplos fueron extraídos del articulo *An efficient gradient-free line search.*





Al interpretar geométricamente la condición de Goldstein, tenemos que el cono definido por las lineas con pendientes $c_1\nabla f(x)^T d$ y $c_2\nabla f(x)^T d$ determina una sección de la gráfica donde podemos encontrar tamaños de paso aceptables. De la misma manera, la desigualdad (3.2) define otra linea con pendiente $c_3\nabla f(x)^T d$ que determina la frontera de otra sección de la gráfica para tamaños de paso aceptables. Si a todo ello le agregamos la condición de descenso $f(x(\alpha)) < f(x)$, obtenemos los intervalos de aceptación mostrados en las gráficas anteriores.

En el articulo se considera a $\mu = \frac{1}{2}$ y se establece que tiene un significado especial para satisfacer **SDC**. Primero, consideremos funciones continuas de clase $\mathcal{C}^2$, por lo cual se cumple que en una vecindad que contiene a un mínimo local, dichas funciones son acotadas inferiormente y por el Teorema de Taylor son casi cuadráticas. Por lo tanto, para un camino de búsqueda lineal y una función cuadrática estrictamente convexa se cumple que la siguiente definición

$$\begin{aligned} f + a\alpha + b\alpha^2 &\stackrel{\text{def}}{=} f(x) + \alpha g(x)^T d + \frac{\alpha^2}{2} p^T G(x) d \\ &= f(x + \alpha d) \end{aligned}$$

donde $a < 0 < b$. Ahora, si tomamos $\hat{\alpha} = -\frac{a}{2b}$, se cumple que

$$f + a\alpha + b\alpha^2 = f - \frac{a^2}{4b} + b(\alpha - \hat{\alpha})^2$$

Entonces, por definición de $\mu(\alpha)$ y por las igualdades anteriores se cumple que

$$\begin{aligned} \mu(\alpha) &= 1 + \frac{b\alpha}{a} \\ &= 1 - \frac{\alpha}{2\hat{\alpha}} \qquad\qquad \text{(Ya que } a = -2\hat{\alpha}b\text{)} \end{aligned}$$

Notemos que $\mu(\alpha) < 1$ para toda $\alpha > 0$. En consecuencia, a lo largo del rayo de búsqueda podemos calcular $\mu(\hat{\alpha}) = \frac{1}{2}$ y el mínimo $\hat{\alpha} = \alpha/2(1 - \mu(\alpha))$, para cualquier $\alpha > 0$. Consideremos entonces el siguiente teorema

Teorema 3.1. *Sea $\beta \in]0, \frac{1}{4}[$ y $g^T d < 0$. La ecuación $\mu(\hat{\alpha}) = \frac{1}{2}$ tiene una solución $\hat{\alpha} > 0$, y cualquier α suficientemente cercana a $\hat{\alpha}$ satisface la condición (3.1).*

Dado que la función de optimización es acotada inferiormente, por el Teorema 3.1 hemos encontrado una manera de calcular un tamaño de paso que cumple la condición de suficiente descenso. De esta manera, intentaremos encontrar con el algoritmo, un tamaño de paso α tal que $\mu(\alpha) \approx \frac{1}{2}$, lo cual se puede lograr con un proceso de bisección y actualización de intervalos de la forma $[\underline{\alpha}, \overline{\alpha}]$ que contengan un tamaño de paso óptimo $\hat{\alpha}$ tal que $\mu(\hat{\alpha}) \approx \frac{1}{2}$. Este procedimiento es descrito con el siguiente pesudocódigo del algoritmo CLS.

3.1. Pseudocódigo CLS

Algorithm 1 Propósito: Encontrar tamaño de paso α con $\mu(\alpha)|\mu(\alpha)-1| \geq \beta$

Require: $x(\alpha), f_0 = f(x(0)), (\nu = -g(x(0))^T d) > 0, \alpha_{init}, \alpha_{\text{máx}}, Q > 1, \beta \in (0, \frac{1}{4})$
Ensure: α
  first $= 1$
  $\underline{\alpha} = 0$
  $\overline{\alpha} = \infty$
  $\alpha = \alpha_{init}$
  while 1 **do**
    Calcular el cociente Goldstein $\mu(\alpha) = (f_0 - f(x(\alpha))/(\alpha v)$
    if $\mu(\alpha)|\mu(\alpha)-1| \geq \beta$ **then**
      break
    end if
    if $\mu(\alpha) > \frac{1}{2}$ **then**
      $\underline{\alpha} = \alpha$
    else if $\alpha = \alpha_{\text{máx}}$ **then**
      break
    else
      $\overline{\alpha} = \alpha$
    end if
    if first **then**
      first $= 0$
      if $\mu(\alpha) < 1$ **then**
        $\alpha = \frac{1}{2}\alpha/(1-\mu(\alpha))$
      else
        $\alpha = \alpha Q$
      end if
    else
      if $\overline{\alpha} = \infty$ **then**
        $\alpha = \alpha Q$
      else if $\alpha = 0$ **then**
        $\alpha = \frac{1}{2}\alpha/(1-\mu(\alpha))$
      else
        $\alpha = \sqrt{\underline{\alpha}\overline{\alpha}}$
      end if
    end if
    $\alpha = \text{mín}(\alpha, \alpha_{\text{máx}})$
  end while
  return α

En el pseudocódigo anterior existen parámetros fijos ajustables, estos siendo, α_{init} (tamaño de paso inicial) , $\alpha_{\text{máx}}$(máximo tamaño de paso),$Q > 1$ y $\beta \in (0, \frac{1}{4})$. Con esto establecido, analicemos el funcionamiento del pseudocódigo.

Primero, notemos que al prinicipio de cada iteración del ciclo while calculamos el cociente de Goldstein para verificar si se cumple (3.1), es decir, la condicion de suficiente descenso. Ahora, dado que queremos encontrar un tamaño de paso tal que $\mu(\alpha) \approx \frac{1}{2}$ vamos a recortar el intervalo $[\underline{\alpha}, \overline{\alpha}]$

Ahora, la variable booleana "first" funciona como bandera para manejar el caso cuadratico de manera optima. Recordemos por lo visto en la sección anterior que si $\mu(\alpha) < 1$ entonces tomemos $\alpha = \frac{1}{2}\alpha/(1-\mu(\alpha))$. Notemos que lo anterior ocurre en la primera iteración, por lo que, si en la siguiente iteración, $\mu(\alpha)$ no cumple **SDC**, sabemos que la función no es cuadrática. Consecuentemente, realizaremos un algoritmo de bisección simple:

Mientras obtengamos un intervalo $[\underline{\alpha}, \overline{\alpha}]$ tal que $\underline{\alpha} > 0$ y $\overline{\alpha} < \infty$, realizaremos el paso de interpolación con $\alpha = \frac{1}{2}\alpha/(1-\mu(\alpha))$ o el paso de extrapolación. Cuando encontramos el intervalo tal que $\underline{\alpha} > 0$ y $\overline{\alpha} < \infty$, realizamos el paso denominado como **Media Geométrica**, es decir, tomar $\alpha = \sqrt{\underline{\alpha}\overline{\alpha}}$. Posterior a esto, en la siguiente interación se actualiza $\underline{\alpha}$ o $\overline{\alpha}$. Terminamos la búsqueda lineal una vez que se satisfaga **SDC**.

3.2. Análisis de complejidad

Sean $0 < \kappa < \lambda < \infty$ parametros fijos ajustables. Buscaremos restringir α_{init} y α_{max} tal que se cumpla la siguiente desigualdad

$$\frac{\kappa\nu}{\|p\|^2} \leq \alpha_{init} \leq \alpha_{max} \leq \frac{\lambda\nu}{\|p\|^2}, \tag{3.3}$$

esto con el fin de que se cumpla el siguiente resultado de complejidad de búsqueda en línea.

Teorema 3.2. *Supongamos $0 < \kappa < \lambda < \infty$. Si la desigualdad (3.3) se cumple, entonces el número de evaluaciones de funciones en el algoritmo CLS está acotado por una constante en términos de $Q, \beta, \kappa, \lambda$ y la constante de Lipschitz γ referida en la Sección 1. Más precisamente:*

(1) Si $\mu(\alpha_{init}) > c_2$, entonces CLS termina en a lo más $\overline{L}_E + \overline{M}_E$ evaluaciones de funciones, donde

$$\overline{L}_E = \left\lceil \frac{\log \frac{Q\lambda}{\kappa}}{\log Q} \right\rceil, \quad \overline{M}_E = \left\lceil \log_2 \frac{\gamma\lambda \log Q}{2(c_2 - c_1)} \right\rceil$$

(2) Si $\mu(\alpha_{init}) < c_1$, entonces CLS termina en a lo más $\overline{L}_Q + \overline{M}_Q$ evaluaciones de funciones, donde

$$\overline{L}_Q = \left\lceil \frac{\log(\gamma\lambda)}{\log(2 - 2c_1)} \right\rceil, \quad \overline{M}_Q = \left\lceil \log_2 \frac{\gamma\lambda \log(\gamma\lambda)}{2(c_2 - c_1)} \right\rceil$$

(3) En otro caso, CLS termina en una sola evaluación de función.

Como consecuencia del Teorema anterior, se obtiene otro resultado de complejidad, esta vez para los métodos de descenso de gradiente o descenso en general que utilizan tamaños de paso calculados a partir del algoritmo CLS.

Teorema 3.3. *Dadas $0 < \kappa < \lambda < \infty$, supongamos que las direcciones de búsqueda del método, d_k, cumplen que*

$$\frac{g_k^T d_k}{\|g_k\|\|d_k\|} \leq -\delta < 0, \quad \text{para } k = 1, 2, \ldots$$

para algún $\delta > 0$ y que los tamaños de paso iniciales se escogen tales que la desigualdad (3.3) se cumple. Entonces:

(1) El número de iteraciones requeridas para alcanzar un punto x tal que $||g(x)|| \leq \varepsilon$ es $O(\varepsilon^{-2})$.

(2) Si el conjunto de subcurvas de nivel $\{x \in \mathbb{R}^n | f(x) \leq f(x_0)\}$ es acotado, entonces, comenzando con el punto inicial x_0, alguna subsecuencia de los puntos generados por el método converge a un punto crítico.

(3) Si f tiene un minimizador local fuerte $\hat{x}$ y además es su único punto crítico, entonces el número de iteraciones requeridas para alcanzar un punto x tal que $||g(x)|| \leq \varepsilon$ es $O(\log \varepsilon^{-1})$.

4. Resultados

4.1. Implementación en Python

Se realizó una implementación del algoritmo CLS en Python. Dado que que es un tipo búsqueda lineal, fue implementado para encontrar el tamaño de paso en el algoritmo de MÁXIMO DESCENSO. Con esto en mente, realizamos pruebas con dicho método para encontrar el mínimo de funciones vistas en clase y poder visualizar el camino de minimos resultantes del algoritmo. Para ello, consideremos las funciones *Rosembrock* y *Branin* (para 2 dimensiones), y veamos la solución encontrada por MD con CLS

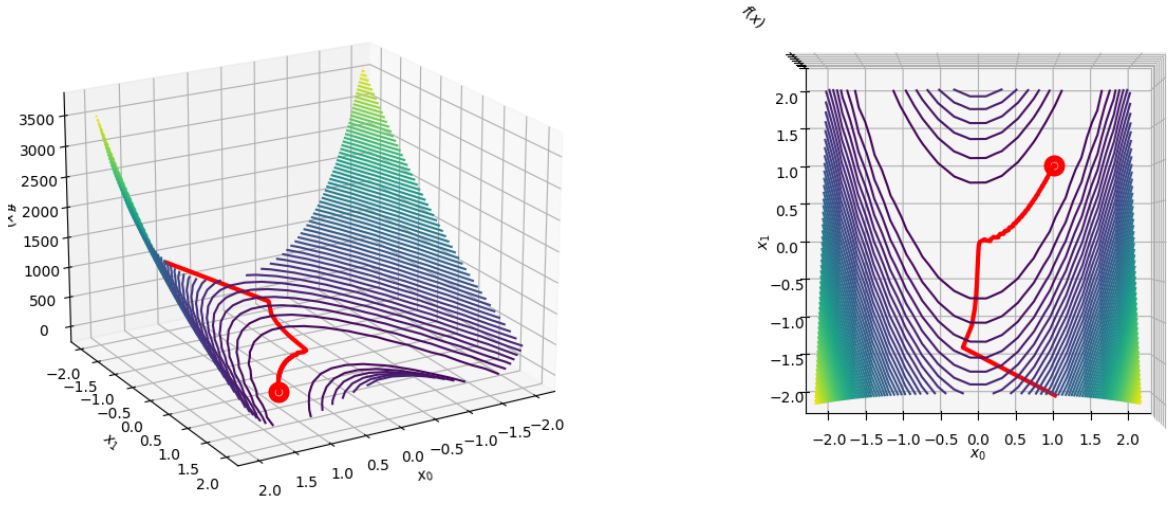


Figura 4.1: ROSEMBROCK → Arg mín = $[1.0, 1.0]$

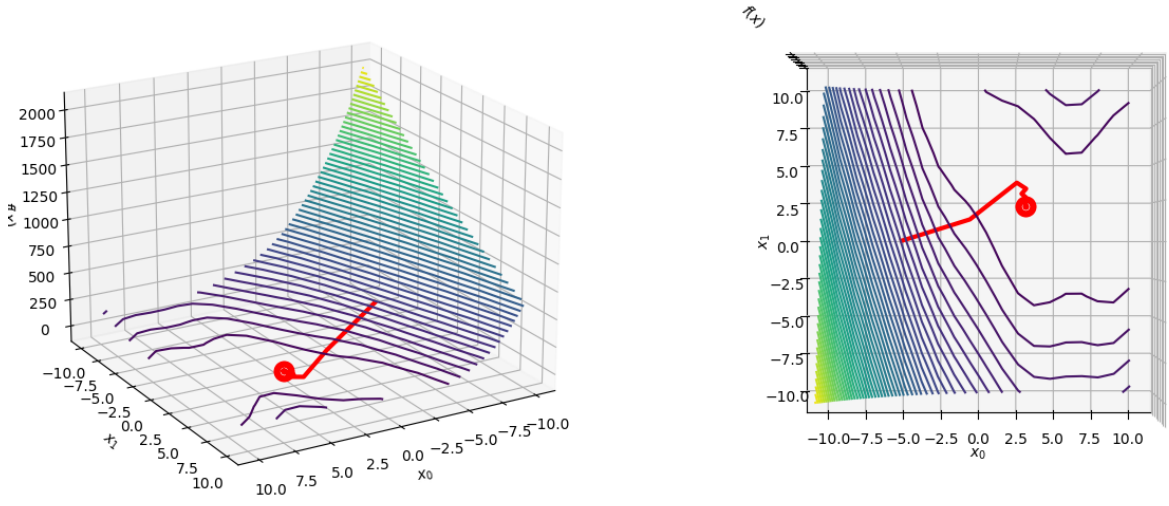


Figura 4.2: BRANIN → Arg mín = $[\pi, 2.275]$

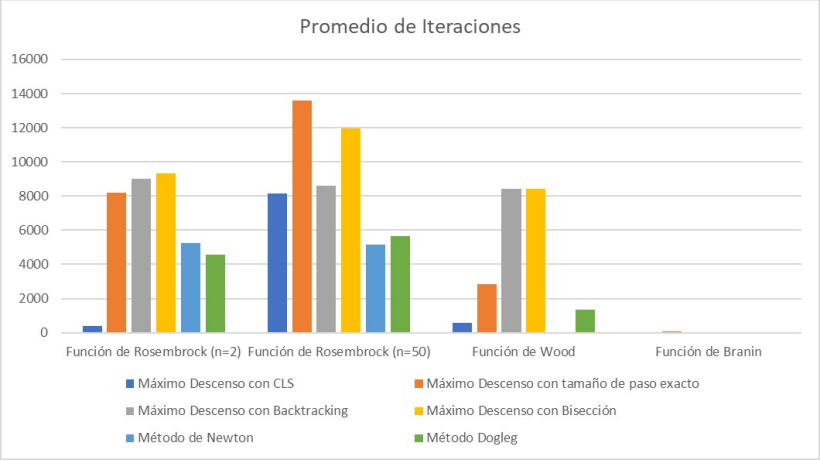
En las figuras anteriores, el punto rojo representa el mínimo de la función en cuestión y las líneas rojas representan el camino encontrado por el algoritmo para llegar a dicho mínimo. Cabe mencionar que en Rosembrock obtuvimos 277 iteraciones para encontrar el mínimo y en Branin lo obtuvimos en 41 iteraciones. Con esto establecido, realizamos comparaciones de CLS con diferentes métodos y poder analizar su eficiencia.

4.2. Comparación con métodos diferentes

El Maximo Descenso implementado con CLS fue comparado con otros algoritmos de Máximo Descenso, que utilizan diferentes estrategias para la obtención del tamaño de paso, las cuales son *Backtracking*, *Bisección* y la fórmula para el tamaño de paso exacto. Además, también se comparó con el método de Newton y el de región de confianza basado en Dogleg. Para obtener resultados viables, cada método se ejecutó 30 veces en 4 diferentes funciones, con puntos iniciales generados aleatoriamente, para posteriormente obtener el promedio de las iteraciones y el tiempo (en segundos) requeridos, y comparar con base en estos datos. Se obtuvieron los siguientes resultados

4.2.1. Iteraciones

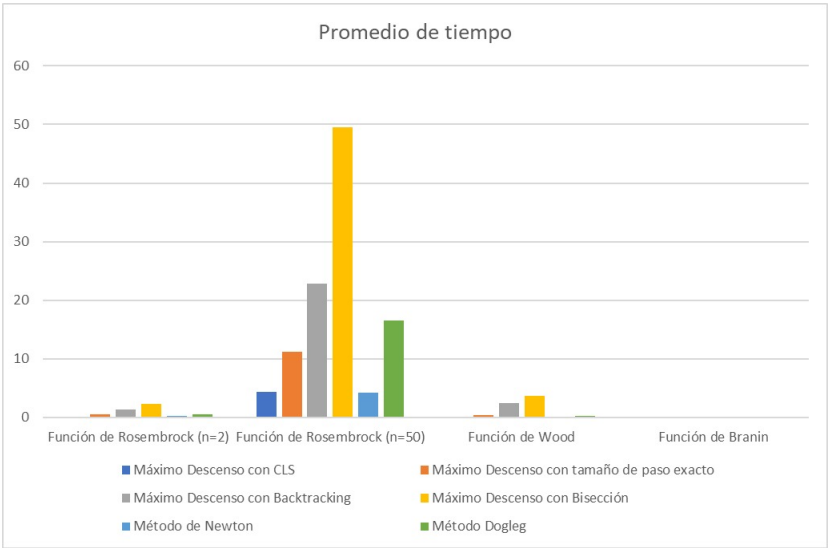
Promedio de iteraciones requeridas				
Método	Función de Rosembrock (n=2)	Función de Rosembrock (n=50)	Función de Wood	Función de Branin
Máximo Descenso con CLS	399.6666667	8159.9	556.1333333	30.7
Máximo Descenso con tamaño de paso exacto	8171.966667	13603.6	2824.733333	53.86666667
Máximo Descenso con Backtracking	9001.3	8593.333333	8435.1	31.5
Máximo Descenso con Bisección	9310.666667	11980.56667	8417.9	35.2
Método de Newton	5251.066667	5167.566667	14.76666667	4.133333333
Método Dogleg	4550.733333	5668.066667	1351.1	4



Por la tabla anterior y por la Figura 4.3 es posible observar que el número de iteraciones del Máximo Descenso con CLS es menor a cualquier otro Máximo Descenso. En ciertos casos llega a tener un numero menor de iteraciones que Newton y usualmente es mejor que Dogleg (el único método de región de confianza). Por lo tanto, en cuanto a el numero iteraciones, la opción más eficiente consiste usar Máximo Descenso con CLS si no es posible usar Newton.

4.2.2. Tiempos

Promedio de tiempo de ejecucion				
Método	Función de Rosembrock (n=2)	Función de Rosembrock (n=50)	Función de Wood	Función de Branin
Máximo Descenso con CLS	0.025511312	4.368589155	0.06417346	0.003797007
Máximo Descenso con tamaño de paso exacto	0.539671445	11.15083549	0.310680008	0.005615822
Máximo Descenso con Backtracking	1.336838595	22.7623025	2.438137468	0.005993756
Máximo Descenso con Bisección	2.31603953	49.45640543	3.612238193	0.009415452
Método de Newton	0.188700668	4.236665527	0.001636577	0.000300908
Método Dogleg	0.464211114	16.52643953	0.240597844	0.002269626



Podemos ver que en cuanto a tiempo, Máximo Descenso con CLS también obtiene muy buenos resultados, nuevamente siendo el método más eficiente en los algoritmos de Máximo Descenso y teniendo un desempeño mejor o comparable al método de Newton para todas las funciones, así como un desempeño por lo general mejor que el método de DogLeg.

5. Conclusiones

Por todo lo mostrado anteriormente, el algoritmo CLS representa una excelente alternativa a los métodos de descenso de gradiente convencionales, en términos de desempeño y practicidad, dada la gran ventaja que propone al evitar evaluaciones adicionales de gradiente y el uso de la matriz hessiana, la cual puede ser difícil de calcular en muchos casos. El algoritmo no es muy complejo de implementar y como vimos, tampoco representa un costo computacional elevado, lo cual también lo posiciona de forma competente ante otros métodos más robustos como el método de Newton o aquellos basados en región de confianza.

6. Bibliografía

- An efficient gradient-free line search - Arnold Neumaier, Morteza Kimiaiei, 2022
- Numerical Optimization - Stephen J. Wright, Jorge Nocedal, 1999
- Line Search Methods - University of Washington, s.f. Disponible en https://sites.math.washington.edu/~burke/crs/408/notes/nlp/line.pdf