TP2 DE OPTIMIZACIÓN

Ciro Carvallo

May 30, 2018

Resumen: El problema a resolver fue hallar la curva que minimiza el tiempo T de caída de una partícula por efecto de la gravedad. Para esto consieramos el funcional $T(\phi) = \int_0^1 \sqrt{\frac{1+\phi'^2}{2g(1-\phi}}dx$ y al discretizar ϕ como una función lineal a trozos, el problema se traduce en encontrar un punto de $\mathbb{R}^n$ que sea el mínimo de $\sqrt{\frac{2}{g}}\sum_{i=0}^{N}\sqrt{1+\left(\frac{x_{i+1}-x_i}{\phi_i-\phi_{i+1}}\right)^2}(\sqrt{1-\phi_{i+1}}-\sqrt{1-\phi_i})$. El trabajo consiste en resolver el problema con los métodos de Newton y método del gradiente con búsqueda lineal, por la razón de oro, utilizando la condición de Armijo y la condición de Wolfe.

Discretización del funcional

El funcional que se busca minimizar es $\mathrm{T}(\phi)=\int_0^1 \sqrt{\frac{1+\phi'^2}{2g(1-\phi(x))}}dx$, discretizando la función ϕ obtenemos $\phi(x) = \sum_{i=0}^{N}\left(\phi_i + \frac{(\phi_{i+1}-\phi(x-x_i)}{x_{i+1}-x_i}\right)\mathbb{1}_{[x_i,x_{i+1}]}(x)$ donde $\phi_i = \phi(x_i)$.

$$\phi'(x) = \sum_{i=0}^{N}\left(\frac{\phi_{i+1}-\phi_i}{x_{i+1}-x_i}\right)\mathbb{1}_{[x_i,x_{i+1}]}(x) \tag{1}$$

Por lo tanto:

$$\sqrt{1+\phi'(x)^2} = \sqrt{\sum_{i=0}^{N} 1+\left(\frac{\phi_{i+1}-\phi_i}{x_{i+1}-x_i}\right)^2}\mathbb{1}_{[x_i,x_{i+1}]}(x) \tag{2}$$

$$\sqrt{2g(1-\phi(x))} = \sqrt{2g}\sqrt{\sum_{i=0}^{N} 1-\phi_i-\left(\frac{\phi_{i+1}-\phi_i}{x_{i+1}-x_i}\right)(x-x_i)}\mathbb{1}_{[x_i,x_{i+1}]}(x) \tag{3}$$

Juntando (2) y (3) podemos escribir la discretización del funcional como:

$$T = \int_0^1 \frac{1}{\sqrt{2g}} \sum_{i=0}^{N} \sqrt{\frac{1 + \left(\frac{\phi_{i+1}-\phi_i}{x_{i+1}-x_i}\right)^2}{1 - \phi_i - \left(\frac{\phi_{i+1}-\phi_i}{x_{i+1}-x_i}\right)(x - x_i)}} \mathbb{1}_{[x_i, x_{i+1}]}(x) dx \tag{4}$$

$$T = \frac{1}{\sqrt{2g}} \sum_{i=0}^{N} \int_0^1 \sqrt{\frac{1 + \left(\frac{\phi_{i+1}-\phi_i}{x_{i+1}-x_i}\right)^2}{1 - \phi_i - \left(\frac{\phi_{i+1}-\phi_i}{x_{i+1}-x_i}\right)(x - x_i)}} \mathbb{1}_{[x_i, x_{i+1}]}(x) dx \tag{5}$$

$$T = \frac{1}{\sqrt{2g}} \sum_{i=0}^{N} \int_{x_i}^{x_{i+1}} \sqrt{\frac{1 + \left(\frac{\phi_{i+1}-\phi_i}{x_{i+1}-x_i}\right)^2}{1 - \phi_i - \left(\frac{\phi_{i+1}-\phi_i}{x_{i+1}-x_i}\right)(x - x_i)}} dx \tag{6}$$

$$T = \frac{1}{\sqrt{2g}} \sum_{i=0}^{N} \sqrt{1 + \left(\frac{\phi_{i+1} - \phi_i}{x_{i+1} - x_i}\right)^2} \int_{x_i}^{x_{i+1}} \frac{1}{\sqrt{1 - \phi_i - \left(\frac{\phi_{i+1}-\phi_i}{x_{i+1}-x_i}\right)(x - x_i)}} dx \tag{7}$$

Haciendo un cambio de variables, $y = 1 - \phi_i - \left(\frac{\phi_{i+1}-\phi_i}{x_{i+1}-x_i}\right)(x - x_i)$, por lo tanto, $dy = -\left(\frac{\phi_{i+1}-\phi_i}{x_{i+1}-x_i}\right) dx$. Luego T es:

$$\frac{1}{\sqrt{2g}} \sum_{i=0}^{N} \sqrt{1 + \left(\frac{\phi_{i+1} - \phi_i}{x_{i+1} - x_i}\right)^2} \left(\frac{x_{i+1} - x_i}{\phi_i - \phi_{i+1}}\right) \int_{1-\phi_i}^{1-\phi_{i+1}} \frac{1}{\sqrt{y}} dy \tag{8}$$

$$\frac{1}{\sqrt{2g}} \sum_{i=0}^{N} \sqrt{\frac{(x_{i+1} - x_i)^2 + (\phi_{i+1} - \phi_i)^2}{(x_{i+1} - x_i)^2}} \left(\frac{x_{i+1} - x_i}{\phi_i - \phi_{i+1}}\right)^2 \int_{1-\phi_i}^{1-\phi_{i+1}} \frac{1}{\sqrt{y}} dy \tag{9}$$

$$\frac{1}{\sqrt{2g}} \sum_{i=0}^{N} \sqrt{1 + \left(\frac{x_{i+1} - x_i}{\phi_i - \phi_{i+1}}\right)^2} \int_{1-\phi_i}^{1-\phi_{i+1}} \frac{1}{\sqrt{y}} dy \tag{10}$$

Como

$$\int_{1-\phi_i}^{1-\phi_{i+1}} \frac{1}{\sqrt{y}} dy = 2(\sqrt{1 - \phi_{i+1}} - \sqrt{1 - \phi_i}) \tag{11}$$

Finalmente se obtiene:

$$T = \sqrt{\frac{2}{g}} \sum_{i=0}^{N} \sqrt{1 + \left(\frac{x_{i+1} - x_i}{\phi_i - \phi_{i+1}}\right)^2} (\sqrt{1 - \phi_{i+1}} - \sqrt{1 - \phi_i}) \tag{12}$$

Gradiente analítico del funcional

La i-ésima derivada parcial $\frac{\partial T}{\partial x_i}$ para $1 < i < N$ es:

$$\sqrt{\frac{2}{g}}\sqrt{1+2\left(\frac{x_i - x_{i-1}}{\phi_i - \phi_{i-1}}\right)}\left(\frac{\sqrt{1-\phi_i}-\sqrt{1-\phi_{i-1}}}{\phi_i - \phi_{i-1}}\right) + \sqrt{\frac{2}{g}}\sqrt{1+2\left(\frac{x_{i+1} - x_i}{\phi_{i+1} - \phi_i}\right)}\left(\frac{\sqrt{1-\phi_i}-\sqrt{1-\phi_{i+1}}}{\phi_{i+1} - \phi_i}\right)$$

Método del gradiente con condición de Armijo

En cada paso la iteración del algoritmo de descenso esta dada por $x_{k+1} = x_k - \alpha_k d_k$ donde d_k es el gradiente numérico en el punto x_k. En primer lugar, utilicé la condición de Armijo para determinar el parámetro α_k en cada iteración. El programa **armijo** toma como input una función g tal $g(\alpha) = f(x - \alpha d)$ donde f es la función que minimizamos y d la dirección. El algoritmo finaliza cuando se cumple la condición e Armijo.

En la primer tabla y el gráfico 1a muestra el valor de $T(x_k)$ en función de la iteración k, tomando la cantidad de puntos $N = 30$ y punto inicial $x_0 = linspace(0, 1, 32)[1 : 31]$. Es decir, x_0 equiespaciado entre 0 y 1.

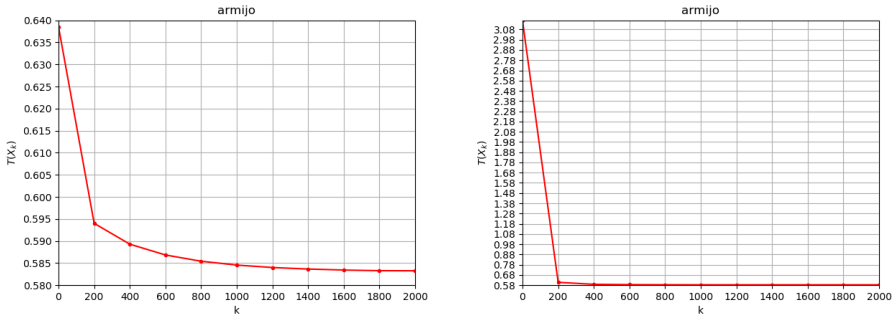
T(X_k) como función de k											
k	0	200	400	600	800	1000	1200	1400	1600	1800	2000
T_k	0.6385	0.5940	0.5892	0.5868	0.5854	0.5845	0.5840	0.5836	0.5834	0.5832	0.5832

Si bien en la tabla hay iteraciones para las cuales el valor de la función T que figura es idéntico, en realidad esta disminuye en cada paso como lo garantiza Armijo, aunque este descenso e del orden de 10^{-5}. Para este método a partir de la iteración 1400 el valor de la función T llega a 0.583.

En la segunda tabla y el gráfico 1b se muestra el valor de $T(x_k)$ en función de la iteración k, tomando la cantidad de puntos $N = 30$ y punto inicial $x_0 = linspace(0, 1, 32)[1 : 31][:: -1]$ que es el reverso del x_0 anterior.

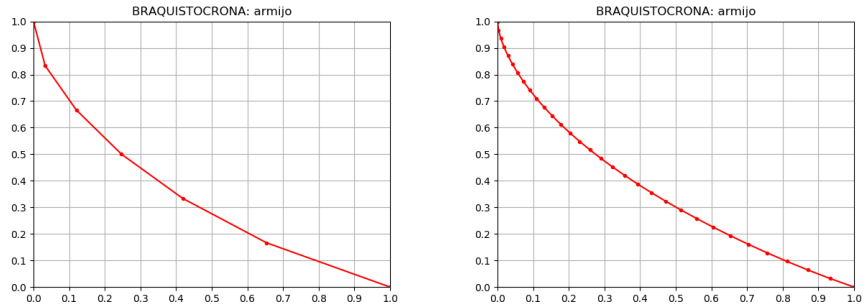
T(X_k) como función de k											
k	0	200	400	600	800	1000	1200	1400	1600	1800	2000
T_k	3.1681	0.6079	0.5884	0.5882	0.5851	0.5842	0.5838	0.5836	0.5835	0.5834	0.5834

Al igual que en la primer tabla, se observa que el valor de la funcion T decrece más rápidamente en las primeras 200 iterciones y al término de las 2000 la diferencia en el valor de T es del orden de 10^{-4}



(a) $x_0 = linspace(0, 1, 32)[1 : 31]$ (b) $x_0 = linspace(0, 1, 32)[1 : 31][:: -1]$

Figure 1: Descenso de T



(a) 5 puntos (b) 30 puntos

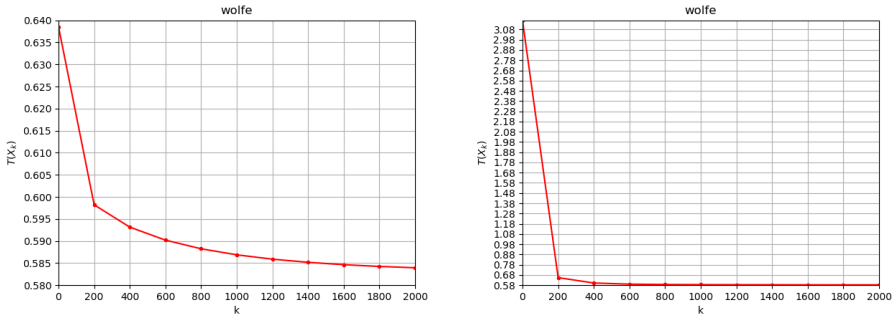
Figure 2: curvas con armijo

Los gráficos 2a y 2b muestran las curvas halladas con 5 y 30 puntos con 200 y 2000 iteraciones respectivamente.

Método del gradiente con condición de Wolfe

En segundo lugar, utilizo la condición de Wolfe para determinar α_k en cada iteración del método de descenso. La función **wolfe** toma, al igual que **armijo**, una función $g(\alpha) = f(x - \alpha d)$ y finaliza una vez que se cumpla la condición. En el gráfico 3a y la siguiente tabla se muestra el valor de $T(x_k)$ en función de la iteración k, tomando la cantidad de puntos N=30 y $x_0 = linspace(0, 1, 32)[1 : 31]$.

T(X_k) como función de k											
k	0	200	400	600	800	1000	1200	1400	1600	1800	2000
T_k	0.6385	0.5982	0.5931	0.5902	0.5882	0.5869	0.5859	0.5852	0.5846	0.5842	0.5839



(a) $x_0 = linspace(0, 1, 32)[1 : 31]$ (b) $x_0 = linspace(0, 1, 32)[1 : 31][:: -1]$

Figure 3: Descenso de T

En este caso, recién en las últimas 200 iteraciones se llega al valor $T = 0.583$ a diferencia del caso en anterior que necesita cerca de 1200 iteraciones para llegar a ese valor. Nuevamente, el descenso más pronunciado se da en las primeras 200 iteraciones.

La segunda tabla y el gráfico 3b muestran los valores correspondientes para $x_0 = linspace(0, 1, 32)[1 : 31][:: -1]$

T(X_k) como función de k											
k	0	200	400	600	800	1000	1200	1400	1600	1800	2000
T_k	3.1681	0.6538	0.6010	0.5901	0.5865	0.5851	0.5844	0.5840	0.5834	0.5832	0.5831

Para este punto inicial también se llega al valor $T = 0.583$ luego de 1600 iteraciones, luego la diferencia en el valor final con el primer punto inicial es del orden e 10^{-4}. En las últimas 400 iteraciones prácticamente se repite el mismo valor.

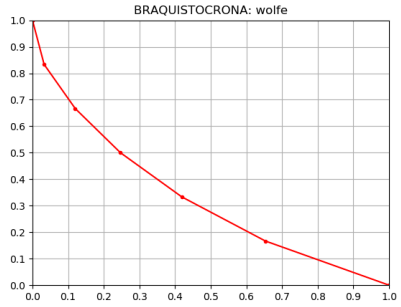
Los gráficos 4a y 4b muestran las curvas halladas con 5 y 30 puntos con 200 y 2000 iteraciones respectivamente.

Método del gradiente con razón aurea

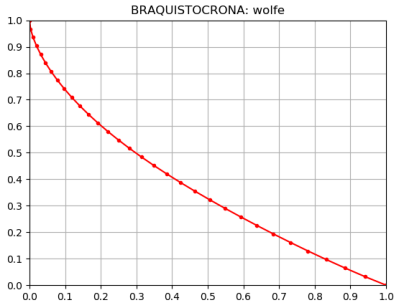
Por último,determinamos el valor de α_k utilizando el métoo de la razón aurea. El programa **aurea** toma la misma función $g(\alpha) = f(x_k - \alpha d_k)$ que los anteriores,donde f nuevamente es la función a minimizar. También toma un parámetro b y una tolerancia. El intervalo en el cual determina al parámetro es (0,b) y finaliza una vez que la longitud del intervalo es menor que la tolerancia, en ete caso fijé $b = 2$ y la $tolerancia = 0.005$.

Al igual que para los métodos anteriores realicé tablas y gráficos 5a y 5b donde se observa el valor de $T(x_k)$ en función de la iteración k, tomando la cantidad de puntos $N = 30$.

Esta tabla corresponde a $x_0 = linspace(0, 1, 32)[1 : 31]$:

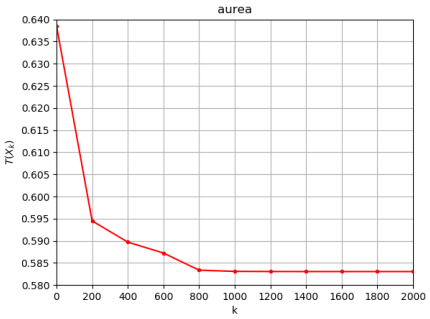


(a) 5 puntos

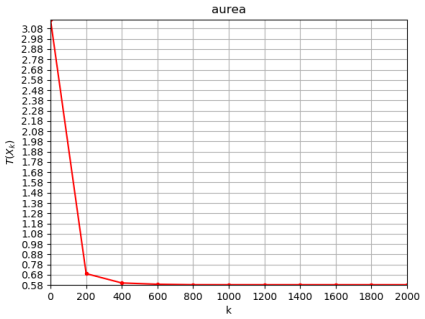


(b) 30 puntos

Figure 4: curvas con wolfe



(a) $x_0 = linspace(0, 1, 32)[1:31]$



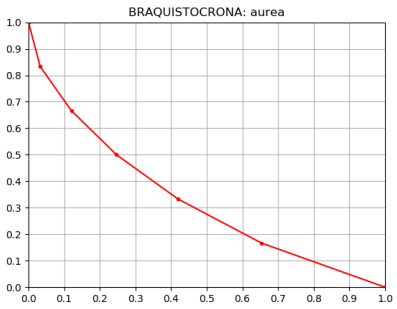
(b) $x_0 = linspace(0, 1, 32)[1:31][::-1]$

Figure 5: Descenso de T

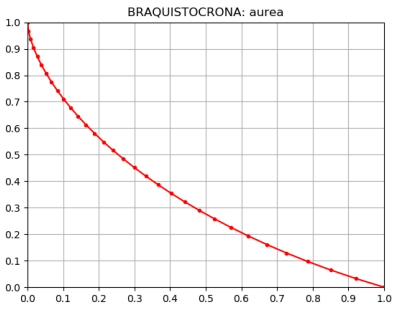
T(X_k) como función de k											
k	0	200	400	600	800	1000	1200	1400	1600	1800	2000
T_k	0.6385	0.5944	0.5897	0.5872	0.5834	0.5831	0.5831	0.5830	0.5830	0.5830	0.5830

Con este método, se necesitaron 800 iteraciones para llegar al valor T=0.583, son menos que las que necesitó **armijo** y **wolfe**. La segunda corresponde a $x_0 = linspace(0, 1, 32)[1:31][::-1]$.

T(X_k) como función de k											
k	0	200	400	600	800	1000	1200	1400	1600	1800	2000
T_k	3.1681	0.6915	0.6003	0.5879	0.5834	0.5832	0.5832	0.5831	0.5831	0.5831	0.5831



(a) 5 puntos



(b) 30 puntos

Figure 6: curvas Braquistocrona

En la tabla se observa que el valor de $T(x_{2000})$ obtenido con **aurea** difiere del obtenido con **armijo** y **wolfe** en menos de 10^{-3}. Los gráficos 6a y 6b se muestran las curvas halladas con 5 y 30 puntos, hechas con 200 y 2000 iteraciones respectivamente.

Método de Newton

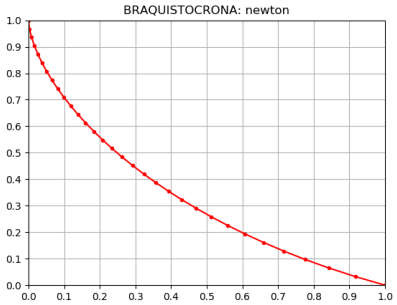
El método de Newton se basa en la iteración $x_{k+1} = x_k - Hd_k$, donde d_k es el gradiente de la función que se busca minimizar y H la inversa del Hessiano. El programa **newton** toma como input el punto x_0 inicial, la función a minimizar y la cantidad de iteraciones que va a realizar. El hessiano y el gradiente son opcionales, en caso de no agregarlos el programa utiliza el gradiente calculado numéricamente y el hessiano también numérico.

Nuevamente analizo el caso en el que la cantidad de puntos es 30, con punto inicial aquel que es equiespaciado entre 0 y 1. Sin embargo, a diferencia de los métodos de descenso, solamente son necesarias 10 iteraciones. En la siguiente tabla pueden observarse los valores de T en cada iteración.

T(X_k) como función de k											
k	0	1	2	3	4	5	6	7	8	9	10
T_k	0.6385	0.6209	0.5919	0.5840	0.5831	0.5831	0.5831	0.5831	0.5831	0.5831	0.5831

En el gráfico 7a puede observarse la curva obtenida. Claramente el método de Newton, partiendo de un punto equiespaciado entre 0 y 1, converge más rápido que los diversos métodos de descenso. Solamente en 10 iteraciones logró llegar al mismo valor de T que les tomó cerca de 1000 iteraciones a los anteriores. Sin embargo, este resultado está fuertemente relacionado con que x_0 es similar al x_{min}.

En cambio, cuando $x_0 = linspace(0, 1, 32)[1 : 31][:: -1]$ entonces durante la iteración e **newton** el hessiano deja de ser inversible. Por otro lado, el método



(a) 5 puntos

Figure 7: curvas con newton

de descenso con **armijo, aurea** y **wolfe** convergen nuevamente a un punto tal que el valor de T es 0.583.

En conclusión, si bien el método de Newton alcanza un valor cercano al mínimo en menos iteraciones que los métodos de descenso, al alejarse de este los métodos de descenso continúan convergiendo mientras que Newton no lo garantiza.