Optimización Continua y Aplicaciones (OCA)

(Obligatorio 2)

Algoritmos de Descenso y Búsqueda Lineal

Claudio Risso, Pablo Rodríguez Bocca

5 de setiembre de 2023

El objetivo de este obligatorio es ejercitar técnicas de optimización sin restricciones en $\mathbb{R}^n$, relevar sus principales características, modelar y resolver un problema de optimización.

1 Óptimos en $\mathbb{R}^1$ (Búsqueda Lineal)

Considere el problema de ajuste de aportes hidrológicos en la cuenca de la represa `hidroUY1`, como en el del Obligatorio 1 (i.e., datos en `aptsHUY1.txt`).

a) Implemente `[error,derr]=err(mu)` que calcula para un `mu` dado el valor del error cuadrático y el de su derivada (use la expresión analítica para el cálculo de la derivada). Estime el μ óptimo usando la *regla de Wolfe* y compare con el obtenido por bipartición.

Considere ahora la función $f(x) = \dfrac{e^{x-1} + e^{1-x}}{2} + \dfrac{x^2}{10}$.

b) Repita los cálculos de la parte a) en este nuevo caso.

c) Implemente una función `[tg,td]=wolfe(fun)` que retorne el intervalo heurístico que contiene al mínimo de una función $f : \mathbb{R} \to \mathbb{R}$ de descenso convexa. Se asume que la entrada `fun` implementa `[f,fp]=fun(t)`, esto es, retorna el valor de la función $f(t)$ y el de su derivada $f'(t)$.

2 Descenso por Gradiente y Conjugado

Considere la función $f(x, y) = 2x^2 + 5y^2 + 2xy - 12x - 8y + 10$ vista en teórico.

a) Implemente el algoritmo Steepest Descent con Búsqueda Lineal por bipartición ($tol = 1e^{-6}$) para encontrar el mínimo de f. Calcule analíticamente el óptimo y úselo como referencia para ver la evolución del error (estime orden y velocidad). Repita el cálculo con múltiples puntos de partida.

b) Repita la parte a) pero con descenso por Gradiente Conjugado usando la regla de Polak-Ribière. Compare con a).

c) Modifique los algoritmos de las partes a) y b) para que la Búsqueda Lineal sea con la Regla de Wolfe, haciendo uso de `[tg,td]=wolfe(fun)`. Repita los cálculos y compare los resultados.

Considere ahora la función de Rosenbrock $g(x, y) = (a - x)^2 + b(y - x^2)^2$, usando los parámetros $a = 1$ y $b = 100$.

d) Resuelva el problema usando Steepest Descent con la Regla de Wolfe para el punto de partida $(-1.5, 3)$. Analice la evolución del error. Plotee en $\mathbb{R}^2$ la secuencia de puntos (i.e. trayectoria seguida), junto con el conjunto de las curvas de nivel de la función. Explique el comportamiento.

e) Repita d) usando la dirección provista por el Gradiente Conjugado con la regla de Fletcher-Reeves y compare los resultados.

3 Ejemplo: La Curva Braquistócrona

Considere una rampa de base X y altura Y, desde cuya cima se lanza (i.e., se deja caer) una pequeña bolita, que podemos asumir no tiene inercia angular, ni sufrirá pérdidas por rozamiento durante su caída. Como referencia de la disposición de los objetos y coordenadas ver Fig-1.

El largo total de la rampa es $Z = \sqrt{X^2 + Y^2}$. Partiendo desde la cima con velocidad v_0 y por la conservación de la energía, al alcanzar la altura $y(t)$ se debe cumplir: $\frac{m \cdot v^2(y)}{2} + m \cdot g \cdot y = \frac{m \cdot v_0^2}{2} + m \cdot g \cdot Y$, de donde llegamos a:

$$v(y) = \sqrt{v_0^2 + 2g(Y - y)}. \tag{1}$$

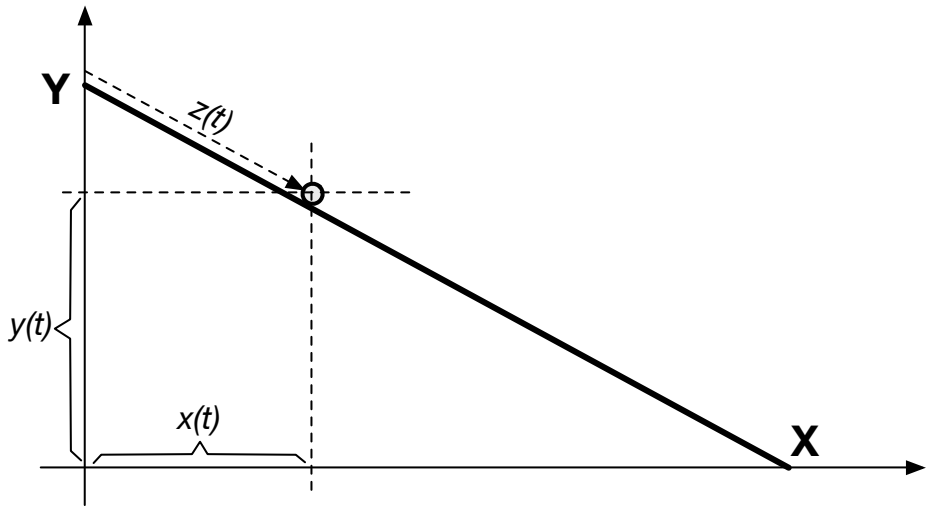

Figura 1: Rampa, bolita y coordenadas de referencia.

Por congruencia de triángulos $\frac{Y-y}{z} = \frac{Y}{Z} = \frac{1}{\sqrt{1+(X/Y)^2}}$, así que podemos substituir $(Y-y)$ en Eq-1 para llegar a:

$$v(z) = \sqrt{v_0^2 + \frac{2g}{\sqrt{1+(X/Y)^2}} \cdot z}. \tag{2}$$

Se busca calcular el tiempo T necesario para que la bolita recorra la rampa en toda su extensión. Recordando que $v(t) = \dot{z}(t) = \frac{dz(t)}{dt}$, separando variables e integrado llegamos a $T = \int_0^T dt = \int_0^Z \frac{dz}{v(z)}$ y resolviendo la última integral:

$$T = \frac{\sqrt{1+(X/Y)^2}}{g}\left(\sqrt{v_0^2 + 2gY} - v_0\right). \tag{3}$$

Si por ejemplo, $X = 2\text{m}$, $Y = 1\text{m}$ (i.e., $Z = \sqrt{5} \approx 2.2361\text{m}$) y liberamos la bolita desde el reposo (i.e., $v_0 = 0$), ésta alcanza el final de la rampa en $T \approx 1.0102\text{s}$. Si en cambio la rampa fuera quebrada, resultando de la unión perfecta (e.g., mediante un codo) entre la vertical y la horizontal, la bolita demora 0.4518s en la caída (surge de usar Eq-3 con $X = 0$, $Y = 1$), consiguiendo una velocidad final de $4.4272\frac{\text{m}}{\text{s}}$ (usar Eq-1), que le permite

recorrer los restantes 2m en otros 0.4518s, totalizando 0.9035s y logrando así llegar antes, incluso a costa de recorrer una distancia 34.2% más larga.

El problema de la curva braquistócrona (del griego: *brachistos* - el más corto y *chronos* - tiempo) consiste en encontrar la forma de la rampa que consigue que la bolita complete su recorrido en el menor tiempo posible. El problema fue propuesto (y resuelto analíticamente) por Johann Bernoulli en 1696 como un sofisticado desafío matemático que duró 18 meses, en el que intervinieron: Gottfried Leibniz, Guillaume de l'Hôpital e Isaac Newton, entre otros. En este obligatorio buscaremos una aproximación numérica.

Se pide:

a) Escriba la función `[T]=braqN1(y)` que retorne el tiempo total necesario para recorrer una rampa de $n =$`|y|`$+1$ tramos rectos, que tiene su primer punto en `(0,1)`, el último en `(2,0)` y puntos (x_i, y_i) intermedios. Los x_i's son equi-espaciados, mientras que los valores y_i son los contenidos en `y`, que en el caso general, no tienen por qué ser decrecientes.

   Sugerencia: Como valor de referencia para testear la función, copiar la forma de la rampa en Fig-1 en tramos colineales. Probar qué pasa si se entrega la bolita con la velocidad final de cierta rampa en otra igual en forma pero cuesta-arriba.

b) Escriba la función `[T,gradT]=braqN2(y)`, que agregue el vector `gradT` $\in \mathbb{R}^{(n-1)\times 1}$: una estimación numérica del gradiente $[\dots \frac{\partial T}{\partial y_i} \dots]$, tomando paso $dy_i = 1e^{-10}$ en todas las variables y_i.

c) Partiendo de una rampa de diez tramos colineales, corra los primeros cinco pasos de un algoritmo de descenso por gradiente con regla de Wolfe.

d) Opcional: Automatice el algoritmo anterior para los parámetros n (cantidad de tramos a usar) e $Iter$ (cantidad de iteraciones). Registre: valores finales de `T` y `|gradT|`, tiempo de ejecución (ver comandos `tic,toc`) y distancia $||, ||_2$ relativa a la solución de referencia en `solP3.txt`.