Optimización inteligente

Tarea 4. Búsqueda de la sección dorada.

Ángel García Báez

2024-10-08

Breve introducción

Se planteo un ejercicio en clase donde se pedía encontrar el intervalo (espacio de búsqueda) donde sería factible encontrar el mínimo global de la función $f(x) = x^2 + \frac{54}{x}$ haciendo uso del método conocido como ***busqueda de la sección dorada***.

A continuación se muestra el pseudocódigo resumido de lo que busca hacer el algoritmo:

Método de busqueda de la sección dorada

Algoritmo

Paso 1: Elegir un límite inferior a y un limite superior b.

Elegir una tolerancia ϵ
Normalizar la variable x cuando:

$w = (x - a)/(b - a)$
$a_w = 0, b_w = 1, L_w = b_w - a_w$
$k = 1$

Paso 2: $w_1 = a_w + (0.618)L_w$

$w_2 = b_w - (0.618)L_w$
IF $f(w_1) < f(w_2); a_w = w_2$
ELSE $b_w = w_1$
$L_w = b_w - a_w$

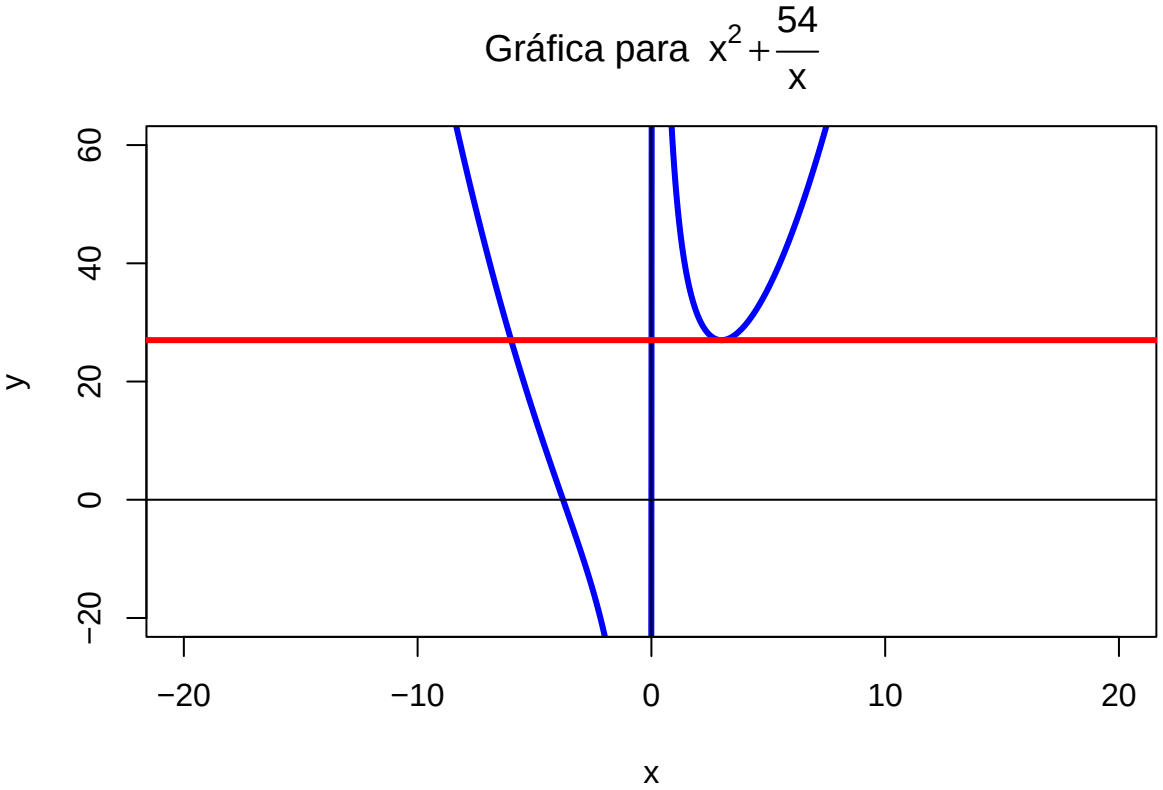
Paso 3: IF $|L_w| < \epsilon$ TERMINAR

ELSE $k = k + 1$; GO TO Paso 2

Una vez explicado lo que se debe de hacer y visto la estructura del método, se realizo la implementación de dicho algoritmo (Anexo 1) en lenguaje R para ponerlo a prueba contra la función dada en un inicio.

Experimentación con los resultados.

Primero, debido a que la función se puede graficar en un espacio de 2 dimensiones, se recurrió a esto para tener un referente visual del comportamiento de la misma:



La función presenta ciertos comportamientos interesantes, por ejemplo, a medida de que se acerca al 0 por la izquierda, dicha función tiende al infinito negativo (si buscara ahí el mínimo, nunca lo encontraría) y si se acerca al 0 por la derecha, el valor de la función tiende a ir hacia el infinito positivo (ahí tampoco es factible buscar).

Dado que la mayoría de los problemas que tienen que ver con cosas del mundo real implican que los valores sean positivos, se optara por buscar valores mayores o iguales a 0 de X tal que permitan llegar al mínimo que se ubica cuando $x = 3$ que devuelve un $f(x) = 27$.

Resolución de clase

Durante la clase, se dieron los parámetros de inicio para la búsqueda: $a = 0$, $b = 5$ y un $\epsilon = 0.01$, los cuales al ser evaluados en el algoritmo implementado, devuelven los siguientes resultados:

a	**b**	**Iter**
0.000000	5.000000	1
1.910000	5.000000	2
1.910000	3.819620	3
2.639475	3.819620	4
2.639475	3.368805	5
2.639475	3.090201	6
2.811652	3.090201	7
2.918058	3.090201	8
2.918058	3.024442	9
2.958696	3.024442	10
2.983811	3.024442	11

El algoritmo con los parámetros dados en clase logra encontrar un intervalo que contiene al valor mínimo de la función, dicho intervalo es relativamente pequeño y logra satisfacer el criterio de tolerancia en apenas 11 iteraciones, dando como resultado el intervalo:

$$[2.983811, 3.024442]$$

Este intervalo encontrado bajo los criterios dados en clase logra contener en su interior al valor de x que minimiza la función.

Solución de clase bajando el ϵ a 0.001

Con la finalidad de explorar el comportamiento del método al bajar más el criterio de tolerancia, se propone un $\epsilon = 0.001$ manteniendo $a = 0$ y $b = 5$.

A continuación se muestra el resultado de dicha evaluación:

a	b	Iter
0.000000	5.000000	1
1.910000	5.000000	2
1.910000	3.819620	3
2.639475	3.819620	4
2.639475	3.368805	5
2.639475	3.090201	6
2.811652	3.090201	7
2.918058	3.090201	8
2.918058	3.024442	9
2.958696	3.024442	10
2.983811	3.024442	11
2.983811	3.008921	12
2.993403	3.008921	13
2.993403	3.002993	14
2.997067	3.002993	15
2.997067	3.000729	16

El algoritmo logra encontrar en 16 iteraciones un intervalo más fino que el encontrado por la corrida anterior, dicho intervalo queda en:

$$[2.997067, 3.000729]$$

Conclusiones y comentarios finales

El método de la búsqueda dorada presenta una mayor versatilidad al no depender de generar los números de Fibonacci, lo cual se traduce en un menor costo computacional para una cantidad de iteraciones elevadas.

En los resultados obtenidos se pudo observar como dicha busqueda, al basarse en una cantidad ϵ de tolerancia, encuentra por si misma la cantidad de iteraciones necesarias hasta satisfacer el criterio establecido.

El algoritmo logra refinar aún más el intervalo para la primera corrida de $[2.983811, 3.024442]$ a $[2.997067, 3.000729]$ en apenas 5 iteraciones más.

Anexo 1: Código fuente del algotimo de busqueda de la sección dorada.

```
## #### Busqueda de la sección dorada ####
## # fx = Función de la cual se quiere saber el minimo
## # a =  limite inferior del intervalo
## # b 0= limite superior del intervalo
## # epsilon = tolerancia permitida
##
## Dorado = function(fx,a,b,epsilon){
##   #### Paso 1 ####
##   # Transformar los datos a un intervalo de 0 y 1
##   tw = function(x){(x-a)/(b-a)}
##   # Transformador de W a X
##   tx = function(w){w*(b-a) + a }
##
##   # Calcular el a y b remapeados en w
##   aw = tw(a); bw = tw(b)
##   # Definir Lw
##   Lw = bw - aw
##   # Definir k
##   k = 1
##
##   #### Paso 3 implicito por ser iterativo hasta que se cumpla la cond de paro
##   # Crear una estructura para guardar los resultados de las iteraciones
##   resultado = data.frame(a = a, b = b,Iter = k)
##   while(epsilon<Lw){
##     #### Paso 2 ####
##     # Definir el w1 y el w2 como pasos intermedios
##     w1 = aw + (0.618)*Lw
##     w2 = bw - (0.618)*Lw
##     # Evaluar los f(w1) y f(w2)
##     fw1 = fx(tx(w1))
##     fw2 = fx(tx(w2))
##     # Evaluar que región eliminar
##     if(fw1 < fw2){
##       # Asignar el nuevo aw
##       aw = w2
##     }else{# EN caso de que fw1 sea mayor a fw2
##       bw = w1
##     }
##     # Actualizo el valor de Lw y de K
##     Lw = bw-aw
##     k = k+1
##     # Agregar los valores
##     resultado[k,] = data.frame(a = tx(aw) , b = tx(bw),Iter = k)
##   }
##   # Regresar el intervalo transformado
##   return(resultado)
## }
```

Anexo 2: Código para hacer las evaluaciones del método.

```
## rm(list=ls())
##
## ##Cargar el script con el método de busqueda Fibonacci ####
##
## source("Tarea 4. Busqueda la sección dorada MAIN.R")
##
## #### Declaro la función que se desea minimizar ####
## fx = function(x){(x^2) +  (54/x)}
##
##
## #### Evaluación para el ejercicio de clase ####
## # a = 0
## # b = 5
## # epsilon = 0.01
## res1 = round(Dorado(fx,0,5,0.01),6)
## res1
##
## #### Evaluación propuesta de hacer más pequeño el epsilon####
## # a = 0
## # b = 5
## # epsilon = 0.001
## res2 = round(Dorado(fx,0,5,0.001),6)
## res2
##
## #### Evaluación propuesta de hacer más pequeño el epsilon####
## # a = 0
## # b = 5
## # epsilon = 0.001
## res3 = round(Dorado(fx,0,5,0.0001),6)
## res3
```