Optimización inteligente

Tarea 5. Búsqueda exhaustiva.

Ángel García Báez

2024-10-11

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 ***búsqueda exhaustiva***.

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

Método de busqueda de búsqueda exhaustiva

Algoritmo

Paso 1: $x_1 = a; \Delta x = (b - a)/n$

(n es el n´umero de puntos intermedios)

$x_2 = x_1 + \Delta x; x_3 = x_2 + \Delta x$

Paso 2: IF $f(x_1) \geq f(x_2) \leq f(x_3)$ el mínimo se encuentra en (x_1, x_3). **TERMINAR**

ELSE $x_1 = x_2; x_2 = x_3; x_3 = x_2 + \Delta x$
GOTO Paso 3

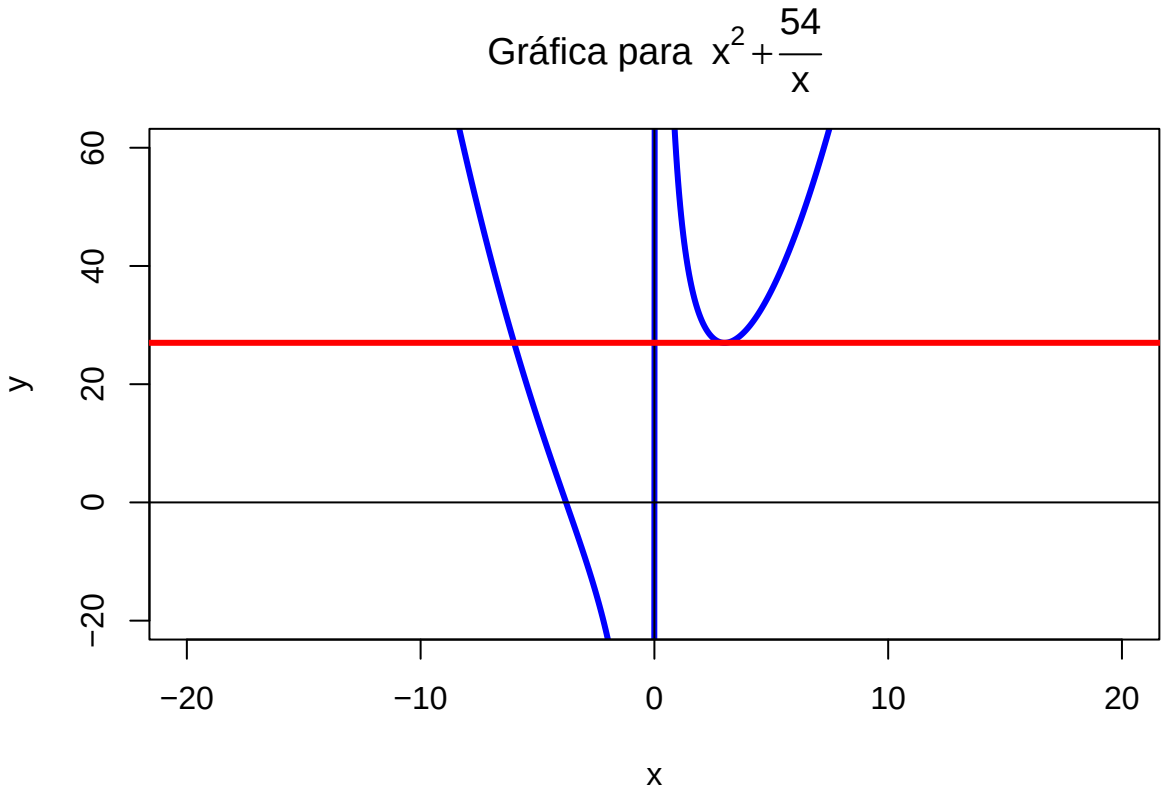
Paso 3: ¿Es $x_3 \leq b$? Si lo es, ir al Paso 2

ELSE no existe un mínimo en (a, b) o un punto extremo (a ó b) es el mínimo.

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 $n = 10$, los cuales al ser evaluados en el algoritmo implementado, devuelven los siguientes resultados:

a	**b**	**n**
0.0	1.0	1
0.5	1.5	2
1.0	2.0	3
1.5	2.5	4
2.0	3.0	5
2.5	3.5	6

El algoritmo con los parámetros dados en clase logra encontrar un intervalo que contiene al valor mínimo de la función en apenas 6 iteraciones, dicho intervalo es el siguiente:

$$[2.5, 3.5]$$

Dicho intervalo logra contener al mínimo, siendo una relativa buena solución para acortar el espacio de búsqueda.

Solución de clase aumentando $n = 50$

Con la finalidad de explorar el comportamiento del método al aumentar el parámetro n que regula el tamaño del paso, se propone un $n = 10$ manteniendo $a = 0$ y $b = 5$. A continuación se muestra el resultado:

a	b	n
0.0	0.2	1
0.1	0.3	2
0.2	0.4	3
0.3	0.5	4
0.4	0.6	5
0.5	0.7	6
0.6	0.8	7
0.7	0.9	8
0.8	1.0	9
0.9	1.1	10
1.0	1.2	11
1.1	1.3	12
1.2	1.4	13
1.3	1.5	14
1.4	1.6	15
1.5	1.7	16
1.6	1.8	17
1.7	1.9	18
1.8	2.0	19
1.9	2.1	20
2.0	2.2	21
2.1	2.3	22
2.2	2.4	23
2.3	2.5	24
2.4	2.6	25
2.5	2.7	26
2.6	2.8	27
2.7	2.9	28
2.8	3.0	29
2.9	3.1	30

El algoritmo logra encontrar en 30 iteraciones un intervalo más fino que el encontrado por la corrida anterior, dicho intervalo queda en: $[2.9, 3.1]$**.**

Conclusiones y comentarios finales

El método de búsqueda exhaustiva (o búsqueda de los 3 puntos) resulta ser algo lento, debido a que por la forma en que está construido, camina dando pequeños pasito de longitud $(b-a)/n$, por lo que requiere de muchas iteraciones para lograr encontrar un intervalo pequeño donde se garantice la existencia de un mínimo.

Se pudo observar, mediante ambas corridas que con un tamaño $n = 10$ logra llegar al intervalo $[2.5, 3.5]$ en apenas 6 iteraciones pero al aumentar el tamaño $n = 50$ logra encontrar el intervalo $[2.9, 3.1]$ en 30 iteraciones.

Se concluye que el método logra su cometido, pero requiere de un tamaño de n relativamente grande y eso se traduce en varias iteraciones para lograr encontrar un intervalo que contenga al optimo y además, sea un espacio de búsqueda relativamente pequeño.

Anexo 1: Código fuente del algotimo de búsqueda exhaustiva.

```
## #### Búsqueda Exhaustiva ####
## # fx = Función de la cual se quiere saber el mínimo
## # a =  limite inferior del intervalo
## # b = limite superior del intervalo
## # n = tolerancia permitida
##
## BE = function(fx,a,b,n){
##   #### Paso 1 ####
##   # Definir el punto x1 y el punto deltaX
##   x1 = a
##   Dx = (b-a)/n;Dx
##   # Definir el punto x2 y el punto x3
##   x2 = x1 + Dx
##   x3 = x2 + Dx
##   #### Paso 3 como condición de paro ####
##   # Creo un objeto para guardaar el resultado de las iteraciones #
##   contador = 0
##   resultados = data.frame(a = 0,b = 0, n = 0)
##   while(x3<=b){
##     #### Paso 2 ####
##     # Evaluar las funciones #
##     fx1 = fx(x1)
##     fx2 = fx(x2)
##     fx3 = fx(x3)
##     # Agregar los valores de x1 y x3
##     contador = contador + 1
##     resultados[contador,] = c(x1,x3,contador)
##     # Verificar si se cumple la condición
##     if(fx1 >= fx2 & fx2<=fx3){
##       # EL minimo se encuentra en (x1,x3)
##       return(resultados)
##     } else{
##       # Actualizar los valores de los 3 puntos para que avancen
##       x1 = x2
##       x2 = x3
##       x3 = x2 + Dx
##     }
##   }
##   return(paste0("No existe mínimo en (",a,",",b,") o un punto extremo es el mínimo."))
## }
```

Anexo 2: Código para hacer las evaluaciones del método.

```
## ##Cargar el script con el método de búsqueda exhaustiva ####
##
## source("Tarea 5. Busqueda exhaustiva MAIN.R")
##
## # Función a evaluar #
## fx = function(x){(x^2) +  (54/x)}
##
##
## #### Evaluación para el ejercicio de clase ####
## # a = 0
## # b = 5
## # n = 10
## res1 = round(BE(fx,0,5,10),6)
## res1
##
##
## #### Evaluación propuesta de hacer más grande el n##
## # a = 0
## # b = 5
## # n = 50
## res2 = round(BE(fx,0,5,50),6)
## res2
```