Optimización inteligente

Tarea 8. Método de bisección y método de la secante

Ángel García Báez

2024-11-10

Contents

1 Breve introducción **2**
1.1 Método de la bisección . . . 2
1.2 Método de la secante . . . 2

2 Experimentación con los resultados. **3**

3 Experimentación para el método de la bisección **4**
3.1 Resolución de clase . . . 4
3.2 Solución de clase cambiando el valor de inicio en $a = 0, b = 5$ . . . 5
3.3 Solución de clase cambiando el valor de inicio en $a = 2, b = 50$ . . . 6
3.4 Solución de clase cambiando el valor de inicio en $a = -5, b = 10$ . . . 7

4 Experimentación para el método de la secante **8**
4.1 Resolución de clase . . . 8
4.2 Solución de clase cambiando el valor de inicio en $L = 1, R = 5$ . . . 10
4.3 Solución de clase cambiando el valor de inicio en $L = 2, R = 50$ . . . 11
4.4 Solución de clase cambiando el valor de inicio en $L = -5, R = 10$ . . . 12

5 Conclusiones y comentarios finales **15**

6 Anexo 1: Código fuente del algotimo de Bisección **16**

7 Anexo 2: Código fuente del algoritmo de la secante **17**

8 Anexo 3: Código para hacer las evaluaciones de ambos métodos **19**

1 Breve introducción

Se planteo un ejercicio en clase donde se pedía encontrar el valor de x que lograra encontrar el mínimo global de la función $f(x) = x^2 + \frac{54}{x}$ haciendo uso de los métodos conocidos como ***método de bisección*** y ***método de la secante***.

A continuación se muestran los pseudocódigos resumidos de lo que busca hacer cada uno de los métodos.

1.1 Método de la bisección

Algoritmo

Paso 1: Elegir dos puntos a y b, tales que $f'(a) < 0$ y $f'(b) > 0$

Elegir una tolerancia ϵ

Paso 2: Calcular $z = \frac{a+b}{2}$. Evaluar $f'(z)$

Paso 3: IF $|f'(z)| \leq \epsilon$ THEN TERMINAR

Paso 4: IF $f'(z) < 0$ THEN $a = z$. GOTO Paso 2

IF $f'(z) > 0$ THEN $b = z$. GOTO Paso 2

1.2 Método de la secante

Algoritmo

Paso 1: Elegir 2 puntos L y R, tales que $f'(L) * f'(R) < 0$

Proporcionar una tolerancia ϵ

Paso 2: Calcular $z = R - \frac{(f'(R))*(R-L)}{(f'(R)-f'(L))}$. Evaluar $f'(z)$

Paso 3: IF $|f'(z)| \leq \epsilon$ THEN TERMINAR

Paso 4: IF $f'(z) < 0$ THEN $L = z$. GOTO Paso 2

IF $f'(z) > 0$ THEN $R = z$. GOTO Paso 2

Nota:

Para estimar las derivadas cuando por alguna razón no se conoce su forma analítica, se pueden hacer de la siguiente forma:

$$f'(x_k) = \frac{f(x_k + \Delta x) - f(x_k - \Delta x)}{2\Delta x}$$

$$f''(x_k) = \frac{f(x_k + \Delta x) - 2f(x_k) + f(x_k - \Delta x)}{(\Delta x)^2}$$

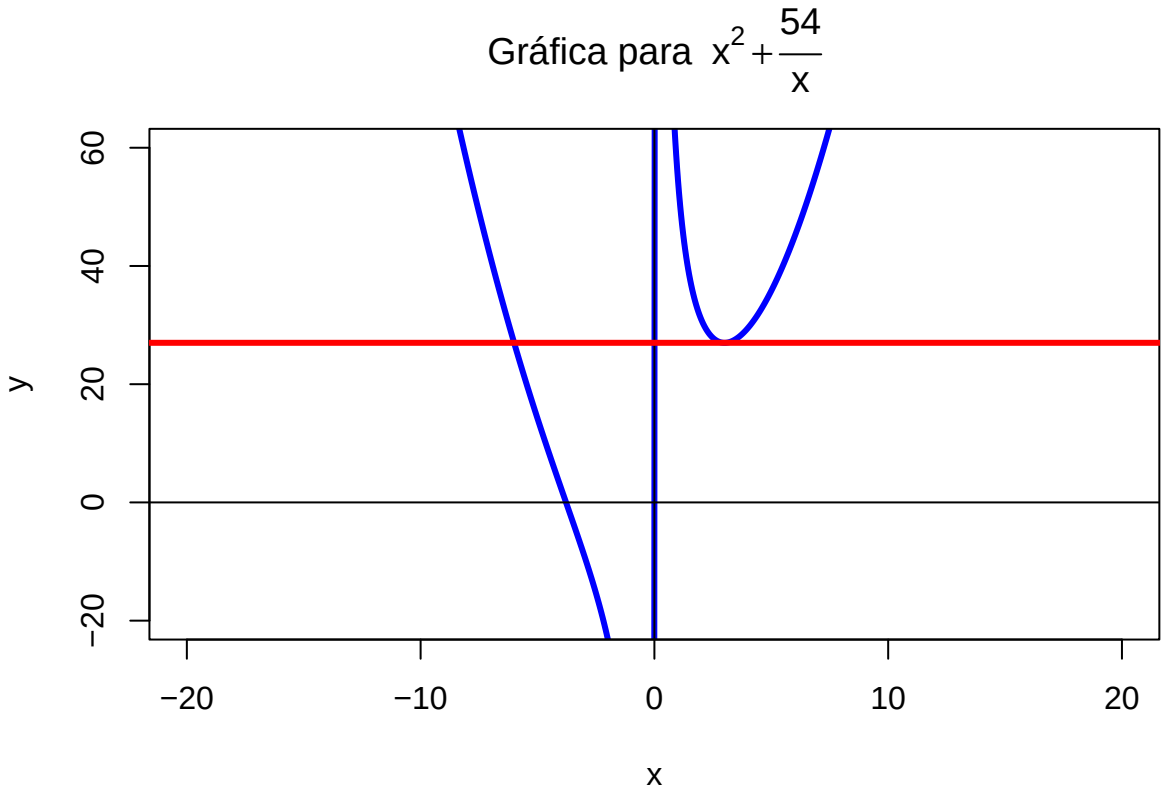
donde Δx es un valor pequeño.

Nótese que se requiere evaluar la función en 3 puntos: x_k, $x_k + \Delta x$ y $x_k - \Delta x$.

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.

2 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 $f(x) = 27$.

3 Experimentación para el método de la bisección

3.1 Resolución de clase

Durante la clase, se dieron los parámetros de inicio para la búsqueda: $a = 2$, $b = 5$ y $tol = 0.001$, adicionalmente dado que se empleo la versión que calcula la derivada con un método numérico, se agrega el parámetro $\Delta = 0.001$ para el cambio en esa derivada. Los valores proporcionados al algoritmo implementado, devuelven los siguientes resultados:

iter	**a**	**b**	**z**	**f'(z)**	**eval**
1	2.000000	5.000000	3.500000	2.5918363748	2
2	2.000000	3.500000	2.750000	-1.6404968120	4
3	2.750000	3.500000	3.125000	0.7203994338	6
4	2.750000	3.125000	2.937500	-0.3830360353	8
5	2.937500	3.125000	3.031250	0.1855730133	10
6	2.937500	3.031250	2.984375	-0.0942423750	12
7	2.984375	3.031250	3.007812	0.0467526924	14
8	2.984375	3.007812	2.996094	-0.0234687408	16
9	2.996094	3.007812	3.001953	0.0117104623	18
10	2.996094	3.001953	2.999023	-0.0058619507	20
11	2.999023	3.001953	3.000488	0.0029285445	22
12	2.999023	3.000488	2.999756	-0.0014656299	24
13	2.999756	3.000488	3.000122	0.0007317255	26

El algoritmo logra llevar a un intervalo lo suficientemente pequeño para satisfacer el criterio de tolerancia establecido en apenas 13 iteraciones del método y con 26 evaluaciones de la función objetivo (2 evaluaciones para realizar el calculo de la derivada por método numérico). El intervalo al que llega donde garantiza la existencia el optimo, es en el intervalo $[2.999756, 3.000488]$ y la mejor solución a la que llega el algoritmo tal que satisface el criterio de tolerancia es: $x = 3.000122$.

3.2 Solución de clase cambiando el valor de inicio en $a = 0, b = 5$

Para explorar como se comporta el algoritmo si se le inicia 0, que es un valor donde presenta una discontinuidad, se propone este cambio, con la finalidad de observar su comportamiento y que tanto repercute en el número de iteraciones o sí es que siquiera llega a converger el algoritmo.

iter	a	b	z	f'(z)	eval
1	0.000000	5.000000	2.500000	-3.6400013824	2
2	2.500000	5.000000	3.750000	3.6599997269	4
3	2.500000	3.750000	3.125000	0.7203994338	6
4	2.500000	3.125000	2.812500	-1.2016675297	8
5	2.812500	3.125000	2.968750	-0.1894813046	10
6	2.968750	3.125000	3.046875	0.2769446397	12
7	2.968750	3.046875	3.007812	0.0467526924	14
8	2.968750	3.007812	2.988281	-0.0705892729	16
9	2.988281	3.007812	2.998047	-0.0117270544	18
10	2.998047	3.007812	3.002930	0.0175603171	20
11	2.998047	3.002930	3.000488	0.0029285445	22
12	2.998047	3.000488	2.999268	-0.0043962718	24
13	2.999268	3.000488	2.999878	-0.0007331185	26

Se observa que pese a que el valor de 0 se sabe que genera problemas, el método se mantiene consistente logrando sortear este problema y alcanzando así la convergencia en apenas 13 iteraciones, presentando un intervalo donde se encuentra el optimo de $[2.999268, 3.000488]$ y logrando encontrar un valor de x tal que satisface el criterio de tolerancia, por lo que la mejor solución a la que llega con estos parámetros es $x = 2.999878$.

3.3 Solución de clase cambiando el valor de inicio en $a = 2, b = 50$

Para explorar como se comporta el algoritmo si se le da un intervalo de búsqueda muy amplio, se propone hacer el cambio de $b = 50$ para contrastar su desempeño contra las corridas anteriores en donde el intervalo era más estrecho.

iter	a	b	z	f'(z)	eval
1	2.000000	50.000000	26.000000	51.9201183431	2
2	2.000000	26.000000	14.000000	27.7244897945	4
3	2.000000	14.000000	8.000000	15.1562499868	6
4	2.000000	8.000000	5.000000	7.8399999136	8
5	2.000000	5.000000	3.500000	2.5918363748	10
6	2.000000	3.500000	2.750000	-1.6404968120	12
7	2.750000	3.500000	3.125000	0.7203994338	14
8	2.750000	3.125000	2.937500	-0.3830360353	16
9	2.937500	3.125000	3.031250	0.1855730133	18
10	2.937500	3.031250	2.984375	-0.0942423750	20
11	2.984375	3.031250	3.007812	0.0467526924	22
12	2.984375	3.007812	2.996094	-0.0234687408	24
13	2.996094	3.007812	3.001953	0.0117104623	26
14	2.996094	3.001953	2.999023	-0.0058619507	28
15	2.999023	3.001953	3.000488	0.0029285445	30
16	2.999023	3.000488	2.999756	-0.0014656299	32
17	2.999756	3.000488	3.000122	0.0007317255	34

Al aumentar considerablemente el intervalo de búsqueda para que oscile entre $[2, 50]$, el método prueba que esto no supone un mayor problema para el algoritmo, puesto que apenas en 17 iteraciones logra llegar al mismo lugar que la primer evaluación con el intervalo inicial $[2, 5]$, tardando apenas 4 iteraciones más para llegar a la misma solución que la primer corrida. Un intervalo de $[2.999756, 3.000488]$ con un valor de $x = 3.000122$ que satisface el criterio de tolerancia y minimiza la función.

3.4 Solución de clase cambiando el valor de inicio en $a = -5, b = 10$

Como ultimo experimento, se propone poner a prueba al método en un escenario donde esta de por medio la discontinuidad, para ello se inicializa el valor inferior del intervalo en $a = -5$ y se amplia el valor superior del intervalo a $b = 5$p ara probar si es que el algoritmo logra converger dado que comienza desde los negativos.

iter	a	b	z	f'(z)	eval
1	-5.000000	10.000000	2.500000	-3.6400013824	2
2	2.500000	10.000000	6.250000	11.1175999646	4
3	2.500000	6.250000	4.375000	5.9287753628	6
4	2.500000	4.375000	3.437500	2.3050822579	8
5	2.500000	3.437500	2.968750	-0.1894813046	10
6	2.968750	3.437500	3.203125	1.1430966018	12
7	2.968750	3.203125	3.085938	0.5013985193	14
8	2.968750	3.085938	3.027344	0.1625844638	16
9	2.968750	3.027344	2.998047	-0.0117270544	18
10	2.998047	3.027344	3.012695	0.0758506868	20
11	2.998047	3.012695	3.005371	0.0321683407	22
12	2.998047	3.005371	3.001709	0.0102474043	24
13	2.998047	3.001709	2.999878	-0.0007331185	26

Sorprendentemente, el algoritmo logra cruzar la discontinuidad en 0 y converger a una solución factible. Apenas con 13 iteraciones es que el algoritmo pasa del intervalo $[-10, 5]$ a el intervalo $[2.998047, 3.001709]$ en donde encuentra un valor que satisface el criterio de tolerancia y minimiza la función, el cual es $x = 2.999878$.

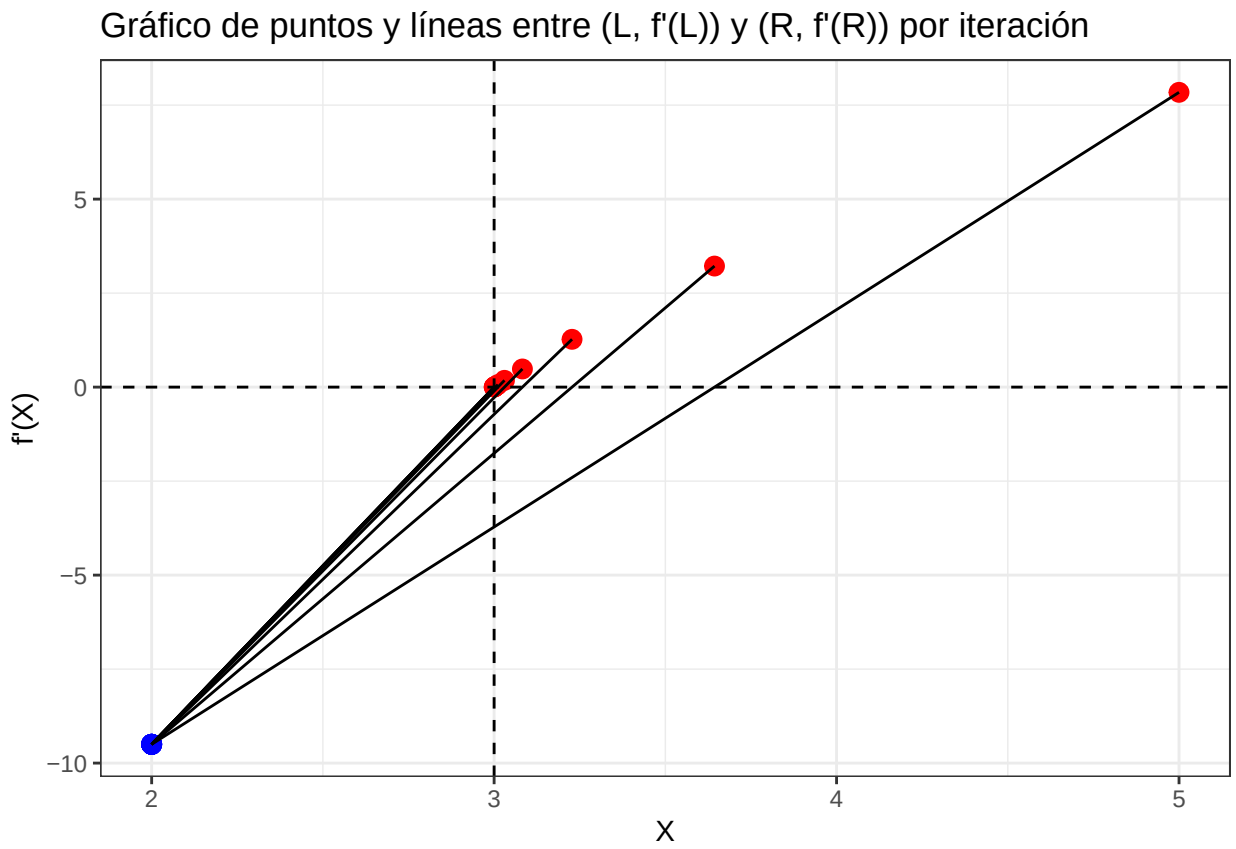
4 Experimentación para el método de la secante

4.1 Resolución de clase

Durante la clase, se dieron los parámetros de inicio para la búsqueda: $L = 2$, $R = 5$ y $tol = 0.001$, adicionalmente dado que se empleo la versión que calcula la derivada con un método numérico, se agrega el parámetro $\Delta = 0.001$ para el cambio en esa derivada. Los valores proporcionados al algoritmo implementado, devuelven los siguientes resultados:

iter	**L**	**R**	**z**	**f'(z)**	**f'(L)**	**f'(R)**	**eval**
1	2	5.000000	3.643599	3.2196499245	-9.500003	7.839999914	6
2	2	3.643599	3.227565	1.2713803337	-9.500003	3.219649925	12
3	2	3.227565	3.082671	0.4828430787	-9.500003	1.271380334	18
4	2	3.082671	3.030305	0.1800190681	-9.500003	0.482843079	24
5	2	3.030305	3.011145	0.0666209262	-9.500003	0.180019068	30
6	2	3.011145	3.004103	0.0245854843	-9.500003	0.066620926	36
7	2	3.004103	3.001511	0.0090633911	-9.500003	0.024585484	42
8	2	3.001511	3.000557	0.0033399034	-9.500003	0.009063391	48
9	2	3.000557	3.000205	0.0012305941	-9.500003	0.003339903	54
10	2	3.000205	3.000076	0.0004533909	-9.500003	0.001230594	60

De forma complementaria y únicamente para esta corrida, se realizo el gráfico del como se comportan los puntos para ir aproximando la secante hasta que se encuentra con el optimo, o un valor muy cercano al optimo.



Resulta curioso el comportamiento del algoritmo, puesto que no mueve el valor de $L = 2$ para nada en todo su recorrido, el que si va moviendo lentamente para aproximar el optimo es el valor de R hasta que en apenas 10 iteraciones logra encontrar un valor de $x = 3.000076$ que satisface el criterio de tolerancia como un valor que logra minimizar la función.

4.2 Solución de clase cambiando el valor de inicio en $L = 1, R = 5$

Para explorar como se comporta el algoritmo si se le inicia en 1, que es un valor muy cercano a la discontinuidad que se produce en 0, se propone este cambio, con la finalidad de observar su comportamiento y que tanto repercute en el número de iteraciones o sí es que siquiera llega a converger el algoritmo.

iter	L	R	z	f'(z)	f'(L)	f'(R)	eval
1	1	5.000000	4.475936	6.25645547	-52.00005	7.83999991	6
2	1	4.475936	4.102638	4.99703297	-52.00005	6.25645547	12
3	1	4.102638	3.830625	3.98119178	-52.00005	4.99703297	18
4	1	3.830625	3.629320	3.15902497	-52.00005	3.98119178	24
5	1	3.629320	3.478736	2.49525406	-52.00005	3.15902497	30
6	1	3.478736	3.365239	1.96219439	-52.00005	2.49525406	36
7	1	3.365239	3.279233	1.53678406	-52.00005	1.96219439	42
8	1	3.279233	3.213807	1.19939137	-52.00005	1.53678406	48
9	1	3.213807	3.163897	0.93331770	-52.00005	1.19939137	54
10	1	3.163897	3.125743	0.72451367	-52.00005	0.93331770	60
11	1	3.125743	3.096532	0.56132371	-52.00005	0.72451367	66
12	1	3.096532	3.074142	0.43421102	-52.00005	0.56132371	72
13	1	3.074142	3.056966	0.33546745	-52.00005	0.43421102	78
14	1	3.056966	3.043781	0.25892675	-52.00005	0.33546745	84
15	1	3.043781	3.033655	0.19969744	-52.00005	0.25892675	90
16	1	3.033655	3.025875	0.15392535	-52.00005	0.19969744	96
17	1	3.025875	3.019896	0.11858983	-52.00005	0.15392535	102
18	1	3.019896	3.015300	0.09133336	-52.00005	0.11858983	108
19	1	3.015300	3.011766	0.07032200	-52.00005	0.09133336	114
20	1	3.011766	3.009049	0.05413276	-52.00005	0.07032200	120
21	1	3.009049	3.006960	0.04166366	-52.00005	0.05413276	126
22	1	3.006960	3.005353	0.03206265	-52.00005	0.04166366	132
23	1	3.005353	3.004118	0.02467168	-52.00005	0.03206265	138
24	1	3.004118	3.003167	0.01898302	-52.00005	0.02467168	144
25	1	3.003167	3.002436	0.01460517	-52.00005	0.01898302	150
26	1	3.002436	3.001874	0.01123643	-52.00005	0.01460517	156
27	1	3.001874	3.001442	0.00864441	-52.00005	0.01123643	162
28	1	3.001442	3.001109	0.00665013	-52.00005	0.00864441	168
29	1	3.001109	3.000853	0.00511584	-52.00005	0.00665013	174
30	1	3.000853	3.000656	0.00393547	-52.00005	0.00511584	180
31	1	3.000656	3.000505	0.00302741	-52.00005	0.00393547	186

iter	L	R	z	f'(z)	f'(L)	f'(R)	eval
32	1	3.000505	3.000388	0.00232885	-52.00005	0.00302741	192
33	1	3.000388	3.000299	0.00179146	-52.00005	0.00232885	198
34	1	3.000299	3.000230	0.00137807	-52.00005	0.00179146	204
35	1	3.000230	3.000177	0.00106007	-52.00005	0.00137807	210
36	1	3.000177	3.000136	0.00081545	-52.00005	0.00106007	216

Persiste el comportamiento donde el valor de L no se modifica para nada mientras que el cambio en el limite del intervalo se produce sobre el valor de R. Resulta llamativo como el acercar el valor de L cerca de la discontinuidad produce que aumenten considerablemente el numero de iteraciones, puesto que ahora se requieren hasta 36 iteraciones para llegar a un valor de $x = 3.000177$ que satisface el criterio de tolerancia, cuando en la primer corrida eran necesarias apenas 10 iteraciones para hacer converger al algoritmo.

4.3 Solución de clase cambiando el valor de inicio en $L = 2, R = 50$

Para explorar como se comporta el algoritmo si se le da un intervalo de búsqueda muy amplio, se propone hacer el cambio de $R = 50$ para contrastar su desempeño contra las corridas anteriores en donde el intervalo era más estrecho.

iter	L	R	z	f'(z)	f'(L)	f'(R)	eval
1	2	50.000000	6.165207	10.90972575	-9.500003	99.97840000	6
2	2	6.165207	3.938756	4.39673797	-9.500003	10.90972575	12
3	2	3.938756	3.325360	1.76738543	-9.500003	4.39673797	18
4	2	3.325360	3.117466	0.67857239	-9.500003	1.76738543	24
5	2	3.117466	3.042968	0.25418536	-9.500003	0.67857239	30
6	2	3.042968	3.015789	0.09424060	-9.500003	0.25418536	36
7	2	3.015789	3.005812	0.03480208	-9.500003	0.09424060	42
8	2	3.005812	3.002140	0.01283300	-9.500003	0.03480208	48
9	2	3.002140	3.000789	0.00472947	-9.500003	0.01283300	54
10	2	3.000789	3.000291	0.00174264	-9.500003	0.00472947	60
11	2	3.000291	3.000107	0.00064205	-9.500003	0.00174264	66

Al aumentar considerablemente el intervalo de búsqueda para que oscile entre $[2, 50]$, el método prueba que esto no supone un mayor problema para el, puesto que apenas y requiere de 11 iteraciones para converger, una más que la primer corrida del algoritmo. Logra encontrar una solución factible que satisface el criterio de tolerancia en $x = 3.000107$

4.4 Solución de clase cambiando el valor de inicio en $L = -5, R = 10$

Como ultimo experimento, se propone poner a prueba al método en un escenario donde esta de por medio la discontinuidad, para ello se inicializa el valor inferior del intervalo en $L = -5$ y se amplia el valor superior del intervalo a $R = 5$p ara probar si es que el algoritmo logra converger dado que comienza desde los negativos.

iter	L	R	z	f'(z)	f'(L)	f'(R)	eval
1	-5.000000	10.000000	0.768501	-89.896562	-12.16000	19.460000	6
2	0.768501	10.000000	8.357255	15.941355	-89.89656	19.460000	12
3	0.768501	8.357255	7.214233	13.390906	-89.89656	15.941355	18
4	0.768501	7.214233	6.378564	11.429892	-89.89656	13.390906	24
5	0.768501	6.378564	5.745734	9.855771	-89.89656	11.429892	30
6	0.768501	5.745734	5.253971	8.551719	-89.89656	9.855771	36
7	0.768501	5.253971	4.864340	7.446522	-89.89656	8.551719	42
8	0.768501	4.864340	4.551018	6.494822	-89.89656	7.446522	48
9	0.768501	4.551018	4.296153	5.666576	-89.89656	6.494822	54
10	0.768501	4.296153	4.086975	4.941069	-89.89656	5.666576	60
11	0.768501	4.086975	3.914082	4.303367	-89.89656	4.941069	66
12	0.768501	3.914082	3.770381	3.742164	-89.89656	4.303367	72
13	0.768501	3.770381	3.650414	3.248455	-89.89656	3.742164	78
14	0.768501	3.650414	3.549907	2.814725	-89.89656	3.248455	84
15	0.768501	3.549907	3.465463	2.434461	-89.89656	2.814725	90
16	0.768501	3.465463	3.394353	2.101871	-89.89656	2.434461	96
17	0.768501	3.394353	3.334361	1.811716	-89.89656	2.101871	102
18	0.768501	3.334361	3.283672	1.559228	-89.89656	1.811716	108
19	0.768501	3.283672	3.240791	1.340058	-89.89656	1.559228	114
20	0.768501	3.240791	3.204478	1.150248	-89.89656	1.340058	120
21	0.768501	3.204478	3.173703	0.986217	-89.89656	1.150248	126
22	0.768501	3.173703	3.147603	0.844737	-89.89656	0.986217	132
23	0.768501	3.147603	3.125455	0.722921	-89.89656	0.844737	138
24	0.768501	3.125455	3.106653	0.618198	-89.89656	0.722921	144
25	0.768501	3.106653	3.090684	0.528292	-89.89656	0.618198	150
26	0.768501	3.090684	3.077117	0.451200	-89.89656	0.528292	156
27	0.768501	3.077117	3.065587	0.385164	-89.89656	0.451200	162
28	0.768501	3.065587	3.055787	0.328650	-89.89656	0.385164	168
29	0.768501	3.055787	3.047456	0.280323	-89.89656	0.328650	174
30	0.768501	3.047456	3.040371	0.239026	-89.89656	0.280323	180
31	0.768501	3.040371	3.034347	0.203756	-89.89656	0.239026	186

iter	L	R	z	f'(z)	f'(L)	f'(R)	eval
32	0.768501	3.034347	3.029223	0.173650	-89.89656	0.203756	192
33	0.768501	3.029223	3.024864	0.147962	-89.89656	0.173650	198
34	0.768501	3.024864	3.021157	0.126052	-89.89656	0.147962	204
35	0.768501	3.021157	3.018002	0.107370	-89.89656	0.126052	210
36	0.768501	3.018002	3.015319	0.091446	-89.89656	0.107370	216
37	0.768501	3.015319	3.013036	0.077875	-89.89656	0.091446	222
38	0.768501	3.013036	3.011093	0.066312	-89.89656	0.077875	228
39	0.768501	3.011093	3.009440	0.056461	-89.89656	0.066312	234
40	0.768501	3.009440	3.008033	0.048070	-89.89656	0.056461	240
41	0.768501	3.008033	3.006836	0.040924	-89.89656	0.048070	246
42	0.768501	3.006836	3.005818	0.034839	-89.89656	0.040924	252
43	0.768501	3.005818	3.004951	0.029657	-89.89656	0.034839	258
44	0.768501	3.004951	3.004214	0.025245	-89.89656	0.029657	264
45	0.768501	3.004214	3.003586	0.021489	-89.89656	0.025245	270
46	0.768501	3.003586	3.003052	0.018291	-89.89656	0.021489	276
47	0.768501	3.003052	3.002597	0.015569	-89.89656	0.018291	282
48	0.768501	3.002597	3.002210	0.013252	-89.89656	0.015569	288
49	0.768501	3.002210	3.001881	0.011279	-89.89656	0.013252	294
50	0.768501	3.001881	3.001601	0.009600	-89.89656	0.011279	300
51	0.768501	3.001601	3.001362	0.008171	-89.89656	0.009600	306
52	0.768501	3.001362	3.001160	0.006954	-89.89656	0.008171	312
53	0.768501	3.001160	3.000987	0.005919	-89.89656	0.006954	318
54	0.768501	3.000987	3.000840	0.005037	-89.89656	0.005919	324
55	0.768501	3.000840	3.000715	0.004287	-89.89656	0.005037	330
56	0.768501	3.000715	3.000608	0.003649	-89.89656	0.004287	336
57	0.768501	3.000608	3.000518	0.003105	-89.89656	0.003649	342
58	0.768501	3.000518	3.000441	0.002643	-89.89656	0.003105	348
59	0.768501	3.000441	3.000375	0.002249	-89.89656	0.002643	354
60	0.768501	3.000375	3.000319	0.001914	-89.89656	0.002249	360
61	0.768501	3.000319	3.000272	0.001629	-89.89656	0.001914	366
62	0.768501	3.000272	3.000231	0.001387	-89.89656	0.001629	372
63	0.768501	3.000231	3.000197	0.001180	-89.89656	0.001387	378
64	0.768501	3.000197	3.000168	0.001004	-89.89656	0.001180	384
65	0.768501	3.000168	3.000143	0.000855	-89.89656	0.001004	390

Los resultados muestran que el algoritmo es capaz de brincar la discontinuidad y posicionarse en un valor por encima del 0 hasta converger, lo llamativo del caso, es que el valor en el que se queda apalancado $L = 0.768501$ al estar tan cerca de la discontinuidad produce que el algoritmo vaya convergiendo muy lentamente, haciendo que se tome hasta 65 iteraciones para lograr una solución factible en $x = 3.000143$.

5 Conclusiones y comentarios finales

Tras implementar ambos métodos y de ponerlos a prueba bajo condiciones muy similares, ocurre que el algoritmo de la bisección es consistente con los resultados a los que llega sin que le afecte demasiado la presencia de la discontinuidad, puesto que opera promediando el valor de los extremos del intervalo. Dicho algoritmo tiene la particularidad de requerir más iteraciones que el método de la secante en el escenario inicial.

El método de la secante requiere menos iteraciones en el escenario inicial que el método de la bisección, pero dicho método presenta complicaciones graves cuando alguno de sus valores se encuentra cerca del punto de discontinuidad, pues esto hace que el algoritmo converja más lento y se disparen el número de iteraciones, ademas que al evaluar la derivada de los extremos del intervalo, no es posible posicionar un intervalo donde alguno de los extremos sea un valor que produzca una discontinuidad debido a que esto produciría que el algoritmo nunca converja.

6 Anexo 1: Código fuente del algotimo de Bisección

```
## #### Método de Bisección ####
## # fx = función que se desea minímizar
## # a = Punto que cumple que f'(x) <0
## # b = Punto que cumple que f'(x) >0
## # delta = Error definido para la derivada
## # tol = Tolerancia para la convergencia
## biseccion = function(fx,a,b,delta,tol){
##   # Paso 1. Se definieron los parametros como entrada.
##   # Definir el calculo numérico de la derivada.
##   fp = function(fx,x,delta){
##     # Crear los valores necesarios que se repiten
##     a1 = fx(x + delta)
##     a2 = fx(x - delta)
##     ## Evaluar la primera y segunda derivada
##     fp1 = (a1-a2)/(2*delta)
##     ## Regresar los valores
##     return(fp1)
##   }
##   # Definir una z provisional para el bucle
##   # y los objetos donde se guardaran los resultados
##   fz = 10 # Valor provicional de la derivada de fx en Z
##   iter = 0 # Iniciar el contador de iteraciones
##   eval = 0 # Evaluaciones de la función por la derivada análitica
##   # Data frame con los resultados
##   resultados = data.frame(iter = 0,a = 0, b = 0,
##                           z = 0, fz = 0, eval = 0)
##   names(resultados)[5] <- "f'(z)"
##   # Paso 3. IF |f'(z)| < epsilon THEN terminar
##   # Esto se va a convertir en la condición de paro
##   while(abs(fz)>tol){
##     # Paso 2. Calcular Z = (a+b)/2
##     z = (a+b)/2
##     # Calcular f'(z)
##     fz = fp(fx,z,delta)
##     # Actualizar el contador de evaluaciones
##     eval = eval + 2
##     #se actualiza el contador de iteraciones
##     iter = iter +1
##     # Aqui actualizo la tabla de datos
##     resultados[iter,] = c(iter,a,b,z,fz,eval)
##     # Paso 4. IF f'(z) < 0 THEN a = Z e ir al paso 2
##     if(fz < 0){
##       # Actualizar el valor de a
##       a = z
##     }else{
##       # IF f'(z) > 0 THEN b= Z e ir al paso 2
##       b = z
##     }
##   }
##   # Reportar los resultados
##   return(resultados)
## }
```

7 Anexo 2: Código fuente del algoritmo de la secante

```
## #### Método de la secante ####
## # fx = función que se desea minímizar
## # L = Punto que cumple que f'(x) <0
## # R = Punto que cumple que f'(x) >0
## # delta = Error definido para la derivada
## # tol = Tolerancia para la convergencia
## secante = function(fx,L,R,delta,tol){
##   # Paso 1. Se definieron los parametros como entrada.
##   # Definir el calculo numérico de la derivada.
##   fp = function(fx,x,delta){
##     # Crear los valores necesarios que se repiten
##     a1 = fx(x + delta)
##     a2 = fx(x - delta)
##     ## Evaluar la primera y segunda derivada
##     fp1 = (a1-a2)/(2*delta)
##     ## Regresar los valores
##     return(fp1)
##   }
##   # Definir una z provisional para el bucle
##   # y los objetos donde se guardaran los resultados
##   fz = 10 # Valor provisional de la derivada de fx en Z
##   iter = 0 # Iniciar el contador de iteraciones
##   eval = 0 # Evaluaciones de la función por la derivada analítica
##   # Data frame con los resultados
##   resultados = data.frame(iter = 0,L = 0, R = 0,
##                           z = 0, fz = 0,fl = 0,fr = 0,eval = 0)
##   names(resultados)[5] <- "f'(z)"
##   names(resultados)[6:7] <- c("f'(L)","f'(R)")
##   # Paso 3. IF |f'(z)| < epsilon THEN terminar
##   # Esto se va a convertir en la condición de paro
##   while(abs(fz)>tol){
##     # Paso 2. Calcular Z = R - (f'(R))*(R-L) /(f'(R) - f'(L))
##     # Calcular FR
##     FR = fp(fx,R,delta)
##     FL = fp(fx,L,delta)
##     z = R - (FR * (R-L))/(FR-FL)
##     # Calcular f'(z)
##     fz = fp(fx,z,delta)
##     # Actualizar el contador de evaluaciones
##     eval = eval + 6
##     #se actualiza el contador de iteraciones
##     iter = iter +1
##     # Aqui actualizo la tabla de datos
##     resultados[iter,] = c(iter,L,R,z,fz,FL,FR,eval)
##     # Paso 4. IF f'(z) < 0 THEN a = Z e ir al paso 2
##     if(fz < 0){
##       # Actualizar el valor de a
##       L = z
##     }else{
##       # IF f'(z) > 0 THEN b= Z e ir al paso 2
##       R = z
##     }
```

```
##   }
##   # Reportar los resultados
##   return(resultados)
## }
```

8 Anexo 3: Código para hacer las evaluaciones de ambos métodos

```
##
## #### Método de bisección ####
##
## # Librería para hacer las tablas bonitas
## library(flextable)
## # Llamo el código
## source("Tarea 8. Método de Bisección y método de la secante MAIN.R")
##
## # Declaro la función que se desea minimizar
## fx = function(x){(x^2) +  (54/x)}
## #### Evaluación para el ejercicio de clase ####
## # a = 2
## # b = 5
## # Tol= 0.001
## # Delta = 0.001
## res1 = biseccion(fx,2,5,0.001,0.001)
## # Reportar los resultados bonitos
## autofit(align(theme_box(flextable(res1)), align = "center", part = "all"))
##
## #### Evaluación para a = 0 y b = 5####
## # a = 0
## # b = 5
## # tol= 0.001
## # Delta = 0.001
## res2 = biseccion(fx,0,5,0.001,0.001)
## # Reportar los resultados bonitos
## autofit(align(theme_box(flextable(res2)), align = "center", part = "all"))
##
## #### Evaluación para a = 2 y b = 50####
## # a = 2
## # b = 5
## # tol= 0.001
## # Delta = 0.001
## res3 = biseccion(fx,2,50,0.001,0.001)
## # Reportar los resultados bonitos
## autofit(align(theme_box(flextable(res3)), align = "center", part = "all"))
##
##
## #### Evaluación para a = -5 y b = 10####
## # a = -5
## # b = 10
## # tol= 0.001
## # Delta = 0.001
## res4 = biseccion(fx,-5,10,0.001,0.001)
## # Reportar los resultados bonitos
## autofit(align(theme_box(flextable(res4)), align = "center", part = "all"))
##
##
## #### Método de la secante####
##
## # Llamo el código
## source("Tarea 8. Método de Bisección y método de la secante MAIN.R")
```

```
## # Declaro la función que se desea minimizar
## fx = function(x){(x^2) +  (54/x)}
## #### Evaluación para el ejercicio de clase ####
## # L = 2
## # R = 5
## # tol= 0.001
## # Delta = 0.001
## res1 = secante(fx,2,5,0.001,0.001)
## # Reportar los resultados bonitos
## autofit(align(theme_box(flextable(res1)), align = "center", part = "all"))
##
## # Crear el gráfico
## library(ggplot2)
## ggplot(res1) +
##   # Puntos para (L, f(L))
##   geom_point(aes(x = L, y = `f'(L)`), color = "blue", size = 3) +
##   # Puntos para (R, f(R))
##   geom_point(aes(x = R, y = `f'(R)`), color = "red", size = 3) +
##   # Líneas que conectan cada par (L, f(L)) y (R, f(R)) en cada iteración
##   geom_segment(aes(x = L, y = `f'(L)`, xend = R, yend = `f'(R)`), color = "black") +
##   geom_vline(xintercept = 3, linetype = "dashed", color = "black") +
##   geom_hline(yintercept = 0, linetype = "dashed", color = "black")+
##   labs(title = "Gráfico de puntos y líneas entre (L, f'(L)) y (R, f'(R)) por iteración",
##        x = "X",
##        y = "f'(X)") +
##   theme_bw()
##
## #### Evaluación para L= 1 y R = 5####
## # L = 1
## # R = 5
## # tol= 0.001
## # Delta = 0.001
## res2 = secante(fx,1,5,0.001,0.001)
## # Reportar los resultados bonitos
## autofit(align(theme_box(flextable(round(res2,8))),align="center",part="all"))
##
## #### Evaluación para a = 2 y b = 50####
## # a = 2
## # b = 50
## # tol= 0.001
## # Delta = 0.001
## res3 =secante(fx,2,50,0.001,0.001)
## # Reportar los resultados bonitos
## autofit(align(theme_box(flextable(round(res3,8))),align="center",part="all"))
##
## #### Evaluación para a = -5 y b = 10####
## # a = -5
## # b = 10
## # tol= 0.001
## # Delta = 0.001
## res4 = secante(fx,-5,10,0.001,0.001)
## # Reportar los resultados bonitos
## autofit(align(theme_box(flextable(round(res4,8))),align="center",part="all"))
```