

(#391) . BÚSQUEDA DE SOLUCIONES (V): MÉTODO DE LA POSICIÓN FALSA

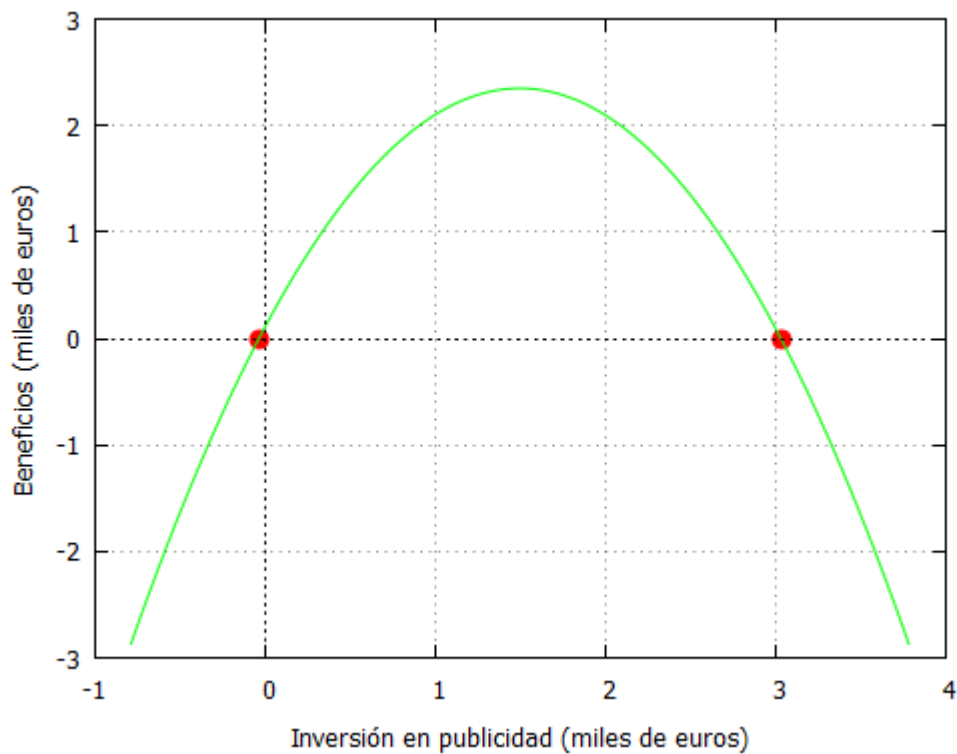
[MONOTEMA] Una variante del [método de la secante](#) es el método de la posición falsa (*Regula Falsi*), que básicamente ejecuta el procedimiento de la secante pero cambiando levemente el algoritmo para que entre dos puntos de dos iteraciones sucesivas siempre se encuentra la solución, de forma análoga a como ocurre con el método de la bisección. Lo haremos, como siempre, desde el punto de vista práctico, y siguiendo a [Burden, Faires & Burden \(2017\)](#).

Función de partida

Emplearemos la misma función que en los ejemplos anteriores, donde se relaciona la inversión en publicidad con los beneficios.

$$f(x) = -x^2 + 3x + 0.1$$

```
f_ceronegativo:ev(funcion,x=-0.03297097167558977), numer;  
f_ceropositivo:ev(funcion,x=3.03297097167559), numer;  
plot2d([[discrete, [[-0.0329709716755897,f_ceronegativo],  
[3.03297097167559, f_ceropositivo]]],  
funcion],[x,-1,4],[y,-3,3],  
[xlabel, "Inversión en publicidad (miles de euros)"],  
[ylabel, "Beneficios (miles de euros)"],  
[style, points, lines], [color, red, green], [legend,  
false]);
```



Mét

odo de la posición falsa

(1) Objetivo: Aproximarse numéricamente a la solución p de una función $f(x)$, tal que $f(p) = 0$

(2) Condiciones: La función $f(x)$ debe ser continua en el intervalo $[a, b]$ considerado.

(3) Descripción rápida: Partimos del método de la secante:

$$p_n = p_{n-1} - \frac{f(p_{n-1})(p_{n-1} - p_{n-2})}{f(p_{n-1}) - f(p_{n-2})}$$

Para calcular p_2 empleamos la fórmula anterior (tras proveer las semillas p_0 y p_1). Sin embargo, para calcular p_3 , usaremos los valores $(p_2, p_1, f(p_2), f(p_1))$ sólo si $f(p_2)$ y $f(p_1)$ tienen signo opuesto. Si ambos tienen el mismo signo, entonces en lugar del par $(p_1, f(p_1))$, escogemos $(p_0, f(p_0))$. Y se sigue ese mismo criterio para el resto de iteraciones.

(4) Convergencia: Un criterio suficiente (pero no necesario) para que lo haga es que:

$$\frac{f(p_0)f''(p_0)}{(f'(p_0))^2} < 1$$

Si se cumple la fórmula anterior cuando está en un intervalo alrededor de la raíz entonces la solución converge en cualquier punto de ese intervalo. Para emplear este criterio, hemos de añadir el requerimiento de que la función sea dos veces derivable.

Normalmente, el método de la posición falsa converge de forma más lenta que el de Newton y el de la secante.

(5) Estimación del error: Hay diversas opciones, aunque una de las más recomendables es el error relativo:

$$\frac{|p_N - p_{N-1}|}{|p_N|} < \epsilon, p_N \neq 0$$

Implementación en Maxima

Vamos a implementar el algoritmo partiendo de los dos valores con los que iniciábamos el método de la bisección: [2.5,3.5], que actúan como las primeras semillas en el método de la posición falsa, y vamos a adaptar en Maxima la propuesta de [Ramírez \(2008\)](#):

```
falsapos(arg):=block([a:arg[1],b:arg[2],f:arg[3],xm],
  xm:(a*ev(f,x=b)-b*ev(f,x=a))/(ev(f,x=b)-ev(f,x=a)),
  if (ev(f,x=xm)*ev(f,x=a)<0) then [a,xm,f] else [xm,b,f] );
  ini:[2.5,3.5,-(x^2)+(3*x)+0.1]; for i:1 thru 10 do
    (print(ini[1]),ini:falsapos(ini));
```

Y nos genera estos resultados:

```
2.5
2.95
3.021739130434783
3.031481481481481
3.03277399056109
3.03294493097818
3.032967529289182
```

3.032970516620634

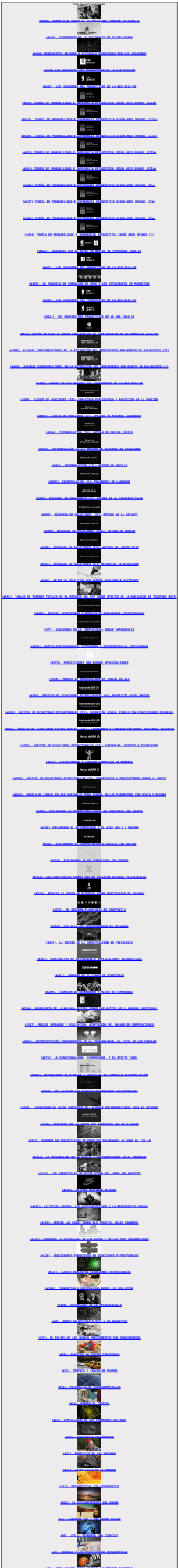
3.032970911521146

3.032970963723678

Para trabajar un poco estos procedimientos, los estudiantes pueden intentar programar una rutina con Maxima donde se codifiquen los procedimientos de [Newton](#), [secante](#) y posición falsa, y comparar las aproximaciones.

Conclusión

En este post hemos explicado brevemente en qué consiste el método de la posición falsa. [Burden, Faires & Burden \(2017\)](#) no recomiendan por lo general este método, pero no deja de ser interesante para trabajar las rutinas de programación y para entender los diferentes procedimientos numéricos de búsquedas de raíces.



(#390) . BÚSQUEDA DE SOLUCIONES (IV): MÉTODO DE LA SECANTE

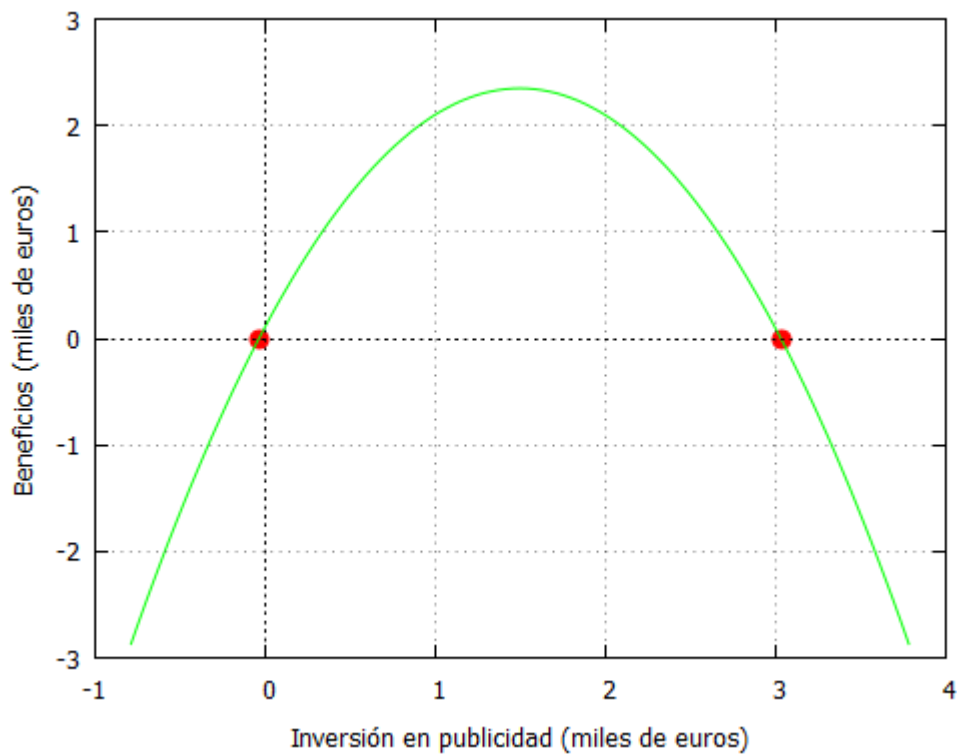
[MONOTEMA] Una modificación del [método de Newton](#) permite una aproximación a este sin computar la derivada. Se trata del método de la secante, que explicaremos a continuación. Lo haremos, como siempre, desde el punto de vista práctico, y siguiendo a [Burden, Faires & Burden \(2017\)](#).

Función de partida

Emplearemos la misma función que en los ejemplos anteriores, donde se relaciona la inversión en publicidad con los beneficios.

$$f(x) = -x^2 + 3x + 0.1$$

```
f_ceronegativo:ev(funcion,x=-0.03297097167558977), numer;  
f_ceropositivo:ev(funcion,x=3.03297097167559), numer;  
plot2d([[discrete, [[-0.0329709716755897,f_ceronegativo],  
[3.03297097167559, f_ceropositivo]]],  
funcion],[x,-1,4],[y,-3,3],  
[xlabel, "Inversión en publicidad (miles de euros)"],  
[ylabel, "Beneficios (miles de euros)"],  
[style, points, lines], [color, red, green], [legend,  
false]);
```



Mét

odo de la secante

(1) Objetivo: Aproximarse numéricamente a la solución p de una función $f(x)$, tal que $f(p) = 0$

(2) Condiciones: La función $f(x)$ debe ser continua en el intervalo $[a, b]$ considerado.

(3) Descripción rápida: Partimos del método de Newton, que recordemos permite construir la sucesión $\{p_n\}_{n=0}^{\infty}$ donde:

$$p_n = p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})}$$

Pero esa derivada puede ser compleja de calcular, por lo que podemos aproximarla de la siguiente forma:

$$f'(p_{n-1}) \approx \frac{f(p_{n-1}) - f(p_{n-2})}{p_{n-1} - p_{n-2}}$$

De este modo, obtenemos:

$$p_n = p_{n-1} - \frac{f(p_{n-1})(p_{n-1} - p_{n-2})}{f(p_{n-1}) - f(p_{n-2})}$$

(4) Convergencia: Un criterio suficiente (pero no necesario) para que lo haga es que:

$$\frac{f(p_0)f''(p_0)}{(f'(p_0))^2} < 1$$

Si se cumple la fórmula anterior cuando está en un intervalo alrededor de la raíz entonces la solución converge en cualquier punto de ese intervalo. Para emplear este criterio, hemos de añadir el requerimiento de que la función sea dos veces derivable.

Normalmente, el método de la secante converge de forma más lenta que el de Newton, pero más rápido que otros, como el de la bisección.

(5) Estimación del error: Hay diversas opciones, aunque una de las más recomendables es el error relativo:

$$\frac{|p_N - p_{N-1}|}{|p_N|} < \epsilon, p_N \neq 0$$

Implementación en Maxima

Vamos a implementar el algoritmo partiendo de los dos valores con los que iniciábamos el método de la bisección: [2.5,3.5], que actúan como las primeras semillas en el método de la secante:

```
f(i):=-x[i]^2+3*x[i]+0.1;
x[0]:2.5;
x[1]:3.5;
for i:2 thru 8 do
(
x[i]:x[i-1]-((x[i-1]-x[i-2])*f(i-1)/(f(i-1)-f(i-2))),
print(i-2,x[i-2]));
datos: makelist([[i-2], x[i-2]], i, 2, 8);
matriz_resultados:apply(matrix,datos);
```


Le hemos dicho al programa que nos retorne en el output siguiente:

0	Iteración	2.5
1	Iteración	3.5
2	Iteración	2.95
3	Iteración	3.021739130434783
4	Iteración	3.033284564740307
5	Iteración	3.032969818745901
6	Iteración	3.032970971557676

Como se puede observar, el método converge rápido, y también podemos establecer criterios de parada con programaciones similares a las realizadas con los otros métodos.

Otra forma de hacerlo

El método de la secante, como su propio nombre indica, traza rectas secantes a la función y que pasan por el eje X. La idea es ir aproximando secantes entre puntos de la función hasta converger en la solución. Para ello, es útil considerar la ecuación de la recta que une los puntos $(a, f(a))$ y $(b, f(b))$:

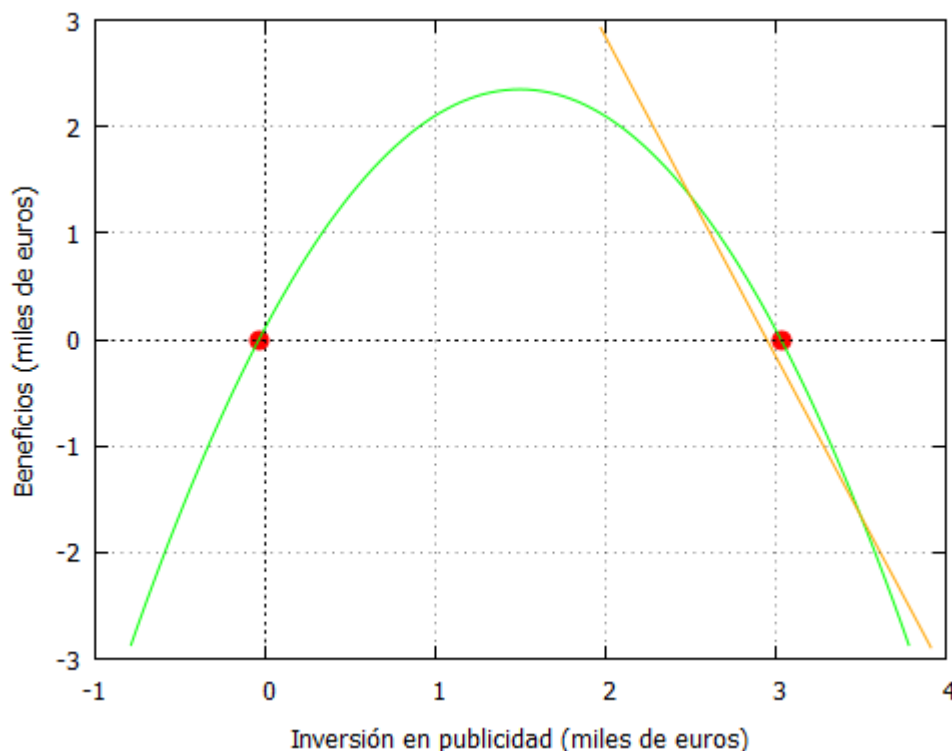
$$y = \frac{f(b) - f(a)}{b - a}(x - a) + f(a)$$

Podemos verlo gráficamente si estipulamos que a y b sean 2.5 y 3.5, respectivamente, que son los valores empleados como intervalos en el método de la bisección:

```

funcion:-(x^2)+(3*x)+0.1;
a:2.5;
fa:(ev(funcion,x=a));
b:3.5;
fb:(ev(funcion,x=b));
secante:((fb-fa)*(x-a)/(b-a))+fa, ratsimp;
f_ceronegativo:ev(funcion,x=-0.03297097167558977), numer;
f_ceropositivo:ev(funcion,x=3.03297097167559), numer;
plot2d([[discrete, [[-0.0329709716755897,f_ceronegativo],
[3.03297097167559, f_ceropositivo]]], funcion,
secante],[x,-1,4],[y,-3,3],
[xlabel, "Inversión en publicidad (miles de euros)"],
[ylabel, "Beneficios (miles de euros)"],
[style, points, lines, lines], [color, red, green, orange],
[legend, false]);

```



Est a primera recta secante, corta a la función en $f(a)$ y $f(b)$, y ya nos quedamos bastante cerca de la solución. Simplemente hay que seguir trazando secantes iterando entre puntos consecutivos, es decir, el siguiente paso sería realizar el mismo cálculo pero tomando $f(b)$ en lugar de $f(a)$, y $f(m)$ en lugar de $f(b)$, siendo $f(m)$ el valor de la función para el

punto m (el punto de corte de la primera secante con el eje X). En Maxima, una forma de computarlo se puede encontrar en [Ramírez \(2008\)](#). Nosotros vamos a adaptar ese código de la siguiente manera:

```
secante(arg):=block([a:arg[1],b:arg[2],f:arg[3],xsec],
  xsec:(a*ev(f,x=b)-b*ev(f,x=a))/(ev(f,x=b)-ev(f,x=a)),
  [b,xsec,f] );
  ini:[2.5,3.5,-(x^2)+(3*x)+0.1];
for i:1 thru 10 do (print(float(ini[1])),ini:secante(ini));
```

El resultado es:

```
2.5
3.5
2.95
3.021739130434783
3.033284564740307
3.0329698187459
3.032970971557676
3.032970971675589
```

Conclusión

En este post hemos explicado brevemente en qué consiste el método de la secante, una de las variantes del [método de Newton](#), que provee una convergencia más lenta, pero que tiene la ventaja de no tener que calcular derivadas en cada iteración.

