Métodos Híbridos para la Solución de Ecuaciones No Lineales. Implementaciones en VBA Excel

Walter Mora F.
wmora2@yahoo.com.mx
Escuela de Matemática
Instituto Tecnológico de Costa Rica

Introducción

Cuando se enseña un curso introductorio de análisis numérico como una asignatura, usualmente los métodos iterativos para resolver ecuaciones no lineales (tales como el método de Newton y el método de la secante), se presentan de manera aislada mientras que en la práctica computacional, estos métodos nunca se usan de forma aislada (conocidas sus debilidades), sino acompañados de otros métodos que les agregan robustez, como el método de bisección. Los paquetes más conocidos implementan, para resolver ecuaciones no lineales generales, un híbrido conocido como el método de Dekker-Brent que usa secante, interpolación cuadrática inversa y bisección. En este trabajo se construyen híbridos más elementales pero robustos: Newton-Bisección y Secante-Bisección y sus respectiva implementación (elemental) en VBA para Excel.
Palabras claves: Métodos Numéricos, ecuaciones no lineales, método de Newton, Bisección, método de la secante, interpolación inversa.

1.1 Método de Newton.

El método de Newton es muy apetecido por su rapidez de convergencia, si la aproximación inicial (o alguna iteración posterior) llega a estar suficientemente cercana un cero de f, entonces se espera que se duplique, aproximadamente, la cantidad de decimales exactos en cada iteración.

Informalmente, si $f \in C^2[a,b]$ y tiene un cero x^* en $[a,b]$, entonces si x_0 es una aproximación "suficientemente cercana" a x^*, la iteración

$$x_{n+1} = x_n - \underbrace{\frac{f(x_n)}{f'(x_n)}}_{\text{corrección}}, \quad n = 0,1,2,\ldots \quad \text{converge a} \quad x^*$$

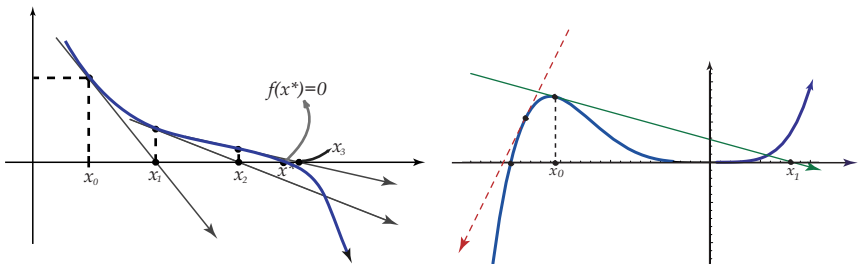


Figura 1.1 Método de Newton. Escogencias de x_0.

¿Qué significa "suficientemente cercano" a x^*?

Esto se puede aclarar en el sentido del teorema

Teorema 1.1 *Sea x^* un cero simple de f y sea $I_\varepsilon = \{x \in \mathbb{R} : |x - x^*| \leq \varepsilon\}$. Asumamos que $f \in C^2(I_\varepsilon)$ y que ε es lo suficientemente pequeño de tal manera que en este intervalo $|f'(x)| \geq m > 0$ y existe $M > 0$ tal que $|f''(x)| \leq M$. Si en el método de Newton, algún x_j satisface*

$$x_j \in I_\varepsilon, \quad y \quad |x_j - x^*| < 2\frac{m}{M} \tag{1.1}$$

entonces $\lim_{n\longrightarrow\infty} x_n = x^*$

De acuerdo al teorema, para asegurar la convergencia es suficiente escoger x_0 y un intervalo I_ε de tal manera que

$$\frac{M|x_0 - x^*|}{2m} < 1.$$

La dificultad de aplicar este criterio es, por supuesto que desconocemos x^* (además de m, M y ε) Esto hace que la práctica usual sea proceder por ensayo y error, buscando una buena aproximación x_0.

Estimación del error

Recordemos que el método de Newton lo podemos poner como un problema de punto fijo: $x_{n+1} = g(x_n)$.

De acuerdo a la teoría de punto fijo, en un intervalo alrededor de x^* suficientemente pequeño que contiene a x_n y a ξ,

$$|x_n - x^*| = \left| \frac{1}{1 - g'(\xi)} \, (x_{n+1} - x_n) \right|$$

Esto nos dice que si $g'(x) \approx 0$ en un intervalo alrededor de x^*, la diferencia entre dos iteraciones consecutivas nos sirve como un buen estimador del error.

Sin embargo, aunque es de esperar que $|x_{n+1} - x_n|$ decrece hasta que $|x_n - x^*| < \delta$, los errores de redondeo dominan (fenómeno de "cancelación") y es de esperar que $|x_{n+1} - x_n|$ varíe irregularmente. Así, un buen criterio de parada es

$$\text{Parar cuando} \quad |x_{n+1} - x_n| \geq |x_n - x_{n-1}| \quad \text{y} \quad |x_n - x_{n-1}| < \delta \tag{1.2}$$

Limitaciones

Computacionalmente podríamos tener que lidiar con problemas de “overflow”, división por cero, convergencia a otra raíz (no la raíz deseada) o no convergencia, además de que este método requiere la derivada de la función que es algo que no está al alcance para algunas funciones, por ejemplo, definidas por integrales, series infinitas o como solución de ecuaciones diferenciales.

EJEMPLO 1.1 Consideremos la función $f(x) = 0.2\,\mathrm{sen}(16x) - x + 1.75$

Esta función tiene un cero en $[1,2]$. Si iniciamos con $x_0 = 1.5,$ el método de Newton presenta un comportamiento errático. Es claro que el paso por una zona de extremos relativos es muy peligrosa para el método de Newton.

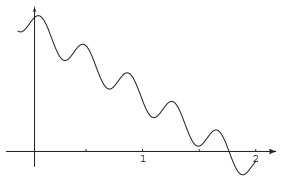


Figura 1.2 Gráfica de $f(x)$

n	Newton x_i	Error Estimado
92	1548.750089	661.603651
93	3043.036515	1494.286425
94	-16029.74436	20488.92517
95	-11684.02425	4345.720103
96	-20779.58784	9095.563587

Tabla 1.1 Método de Newton aplicado a $0.2\,\mathrm{sen}(16x) - x + 1.75 = 0$ con $x_0 = 1.5$

1.2 Método de Bisección.

Si f es una función continua en $[a,b]$ tal que $f(a)f(b) < 0,$ entonces el método de bisección converge a un cero de f en $[a,b]$: Bisección es un método robusto. Este método solo usa la función f para verificar cambios de signo. Si $b-a \leq 1,$ el método gana una

cifra decimal exacta aproximadamente cada tres iteraciones y aproxima un cero de f con un error estimado $\leq 0.5\times 10^{-k}$ en un número de iteración igual o superior a

$$\left\lceil \ln\left(\frac{b-a}{0.5\times 10^{-k}}\right)/\ln 2 \right\rceil \quad \text{iteraciones}$$

El orden de convergencia del método en algunos casos particulares es lineal y en general no se puede establecer un orden de convergencia (el límite que define el orden de convergencia no existe en general).

En el método de bisección construye tres sucesiones a_n, b_n y m_n,

- Inicio: $a_1 = a$ y $b_1 = b$ y $m_1 = (a_1+b_1)/2 = a_1 + \frac{1}{2}(b-a)$.

- El nuevo intervalo $[a_k, b_k]$ se calcula con la fórmula

$$[a_k,b_k] = \begin{cases} [m_k,b_k] & \text{si} \quad f(a_k)f(m_k) > 0 \\ [a_k,m_k] & \text{si} \quad f(a_k)f(m_k) < 0 \end{cases}$$

- $m_k = a_k + \frac{1}{2}(b_k - a_k)$

- Estimación del error de la aproximación

$$m_k \approx x^* \quad \text{con error menor o igual a} \quad \frac{b_k-a_k}{2} = \frac{b-a}{2^k}$$

1.3 Híbrido Newton-Bisección.

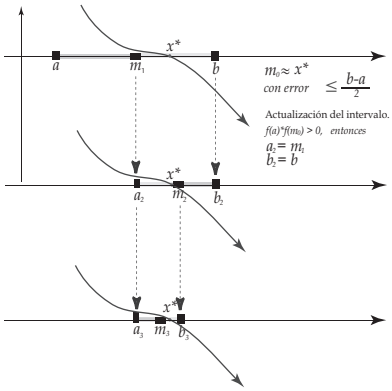


Figura 1.3 Método de Bisección.

Si el método de Newton falla (en algún sentido) en una iteración, podemos usar bisección para dar un pequeño salto y regresar al método de Newton lo más pronto posible.

Supongamos que $f(a)f(b) < 0$. Sea $x_0 = a$ o $x_0 = b$. En cada iteración una nueva aproximación x' es calculada y a y b son actualizados como sigue

1. si $x' = x_0 - \dfrac{f(x_0)}{f'(x_0)}$ cae en $[a,b]$ lo aceptamos, sino usamos bisección, es decir $x' = \dfrac{a+b}{2}$.
2. Actualizar: $a' = x',\ b' = b$ o $a' = a,\ b' = x',$ de tal manera que $f(a')f(b') \leq 0.$

Para garantizar que $x' = x_0 - \dfrac{f(x_0)}{f'(x_0)} \in [a,b],$ no debemos usar directamente este cálculo para evitar la división por $f'(x_0)$ (que podría causar problemas de "overflow" o división por cero). Mejor usamos un par de desigualdades equivalentes.

Observemos que $x_{n+1} \in]a_n, b_n[$ si y sólo si

$$a_n < x_n - \frac{f(x_n)}{f'(x_n)} < b_n,$$

entonces

• Si $f'(x_n) > 0$, $(a_n - x_n) f'(x_n) < -f(x_n)$ y $(b_n - x_n) f'(x_n) > -f(x_n)$.

• Si $f'(x) < 0$, $(a_n - x_n) f'(x_n) > -f(x_n)$ y $(b_n - x_n) f'(x_n) < -f(x_n)$

En este algoritmo, se pasa a bisección si $x' = x_0 - f(x_0)/f'(x_0)$ sale del intervalo, pero esto no indica necesariamente que la iteración de Newton tenga algún tipo de problema en ese paso. Lo que si es importante es que se podría escoger un intervalo $[a,b]$ hasta con extremos relativos (que podrían ser mortales para el método de Newton) y el tránsito sería seguro de todas formas. Un algoritmo similar aparece en [1] (pág. 366). En la implementación, se pasa a bisección si x' sale del intervalo o si la reducción del intervalo no es suficientemente rápida.

EJEMPLO 1.2 Consideremos la función $f(x) = 0.2\,\mathrm{sen}(16x) - x + 1.75$.

Esta función tiene un cero en $[1,2]$. Así que $a = 1$ y $b = 2$. Iniciamos con $x_0 = 1$. La tabla (1.2) muestra el método de Newton y Híbrido Newton-bisección aplicado a esta ecuación. El método de Newton diverge mientras que el híbrido aproxima la solución adecuadamente.

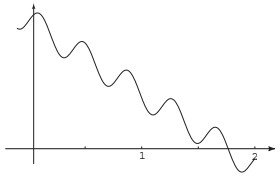


Figura 1.4 Gráfica de $f(x)$

n	Newton x_i	Error Estimado	Híbrido x_i	Error Estimado	Método Usado
1	1.170357381	0.170357381	1.17035738114819	0.170357381	Newton
2	0.915271273	0.255086108	1.58517869057409	0.414821309	Bisección
3	1.310513008	0.395241735	1.79258934528705	0.207410655	Bisección
4	1.539337071	0.228824064	1.76166924922784	0.030920096	Newton
5	1.476007274	0.063329797	1.76306225245136	0.001393003	Newton
6	1.565861964	0.08985469	1.76306130340890	9.49042×10^{-7}	Newton
	...		...		
50	-2873.507697				
51	-1720.319542				

Tabla 1.2 Método de Newton e híbrido Newton-bisección aplicado a $0.2\,\mathrm{sen}(16x) - x + 1.75 = 0$ en $[1,2]$.

1.3.1 Algoritmo e Implementación en VBA Excel.

Algoritmo 1.1: Híbrido Newton-bisección.

Entrada: Una función continua f, las aproximaciones iniciales a y b, δ y maxItr
Resultado: Una aproximación x_k del cero.

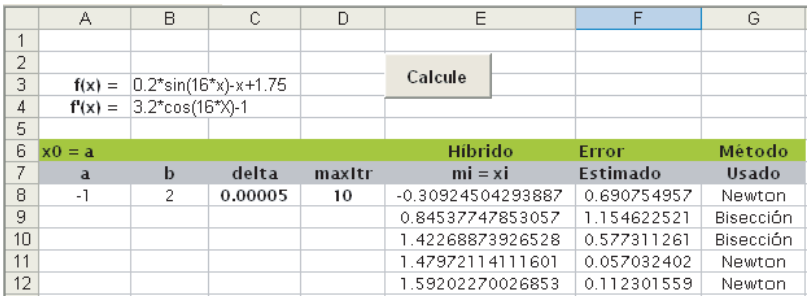
```
1  k = 0, x0 = a ;
2  repeat
3      test1 = (f'(x0) > 0 ∧ (a − x0) f'(x0) < −1 · f(x0) ∧ (b − x0) · f'(x0) > −1 · f(x0));
4      test2 = (f'(x0) < 0 ∧ (a − x0) f'(x0) > −1 · f(x0) ∧ (b − x0) · f'(x0) < −1 · f(x0));
5      if test1 Or test2 Or f(x0) = 0 then
6          x1 = x0 − f(x0)/f'(x0);
7          dx = |x1 − x0|;
8          x0 = x1;
9          if Sgn(f(a)) <> Sgn(f(x1)) then
10             b = x1
11         else
12             a = x1
13     else
14         x1 = a + 0.5 ∗ (b − a);
15         dx = (b − a)/2;
16         x0 = x1;
17         if Sgn(f(a)) <> Sgn(f(x1)) then
18             b = x1
19         else
20             a = x1
21     k = k + 1;
22 until dx < delta Or k > maxItr ;
23 return x1
```

Hoja Excel para el Híbrido Newton-bisección.

Para hacer una hoja Excel, usamos como referencia la figura (1.5).

Esta implementación usa el evaluador de funciones clsMathParser (para VBA Excel). Este evaluador se descarga en `http://digilander.libero.it/foxes/` Desde este sitio Web se baja un comprimido con los archivos “clsMathParser.cls”y “mMathSpecFun.bas”. Am-

bos se insertan abriendo el editor de VBA, seleccionando “Hoja 1” y con el botón derecho del mouse se importa cada uno de ellos, por separado.



	A	B	C	D	E	F	G
1							
2					Calcule		
3	f(x) =	0.2*sin(16*x)-x+1.75					
4	f'(x) =	3.2*cos(16*X)-1					
5							
6	x0 = a				Híbrido	Error	Método
7	a	b	delta	maxItr	mi = xi	Estimado	Usado
8	-1	2	0.00005	10	-0.30924504293887	0.690754957	Newton
9					0.84537747853057	1.154622521	Bisección
10					1.42268873926528	0.577311261	Bisección
11					1.47972114111601	0.057032402	Newton
12					1.59202270026853	0.112301559	Newton

Figura 1.5 Hoja Excel para el híbrido Newton-bisección.

Hay que observar que bisección usa un número de iteraciones conocido para alcanzar la tolerancia que se pide. El método de Newton puede ser muy lento en el caso de raíces múltiples. Es deseable establecer un número máximo de iteraciones y refinar el algoritmo de tal manera que después de este máximo se pase directamente a bisección.

El código VBA para Excel es:

```
Option Explicit
Private Sub CommandButton1_Click()
Dim fx, fpx, a, b, delta, maxItr, fi, co
    fx = Cells(3, 2)
    fpx = Cells(4, 2)
    a = Cells(8, 1)
    b = Cells(8, 2)
    delta = Cells(8, 3)
    maxItr = Cells(8, 4)
    Call hibridoNB(fx, fpx, a, b, delta, maxItr, 8, 5)
End Sub

Sub hibridoNB(fx, fpx, a0, b0, delta, maxItr, fi, co)
Dim f As New clsMathParser
Dim fp As New clsMathParser
```

```
Dim test1 As Boolean, test2 As Boolean
Dim k, x0, x1, dx, okfx, okfpx, fx0, fpx0, fx1, fa, a, b

okfx = f.StoreExpression(fx)
okfpx = fp.StoreExpression(fpx)

If Not okfx Then
    MsgBox ("Error en f: " + f.ErrorDescription)
    Exit Sub
End If

If Not okfpx Then
    MsgBox ("Error en fp: " + fp.ErrorDescription)
    Exit Sub
End If
'No modificar a y b
a = a0
b = b0
If Sgn(f.Eval1(a)) = Sgn(f.Eval1(b)) Then
        MsgBox ("Error: f(a)f(b)>=0")
        Exit Sub
End If

k = 0
x0 = a
Do
    fx0 = f.Eval1(x0)
    fpx0 = fp.Eval1(x0)
test1 = (fpx0 > 0 And (a - x0) * fpx0 < -1 * fx0 And (b - x0) * fpx0 > -1 * fx0)
test2 = (fpx0 < 0 And (a - x0) * fpx0 > -1 * fx0 And (b - x0) * fpx0 < -1 * fx0)
If test1 Or test2 Or fx0 = 0 Then
     x1 = x0 - fx0 / fpx0
     dx = Abs(x1 - x0)
     x0 = x1
     If Sgn(f.Eval1(a)) <> Sgn(f.Eval1(x1)) Then
           b = x1
     Else: a = x1
     End If
     Cells(fi + k, co) = x1
     Cells(fi + k, co + 1) = dx
     Cells(fi + k, co + 2) = "Newton"
 Else
    x1 = a + 0.5 * (b - a)
    dx = (b - a) / 2
    x0 = x1
    If Sgn(f.Eval1(a)) <> Sgn(f.Eval1(x1)) Then
          b = x1
    Else: a = x1
    End If
```

```
    Cells(fi + k, co) = x1
    Cells(fi + k, co + 1) = dx
    Cells(fi + k, co + 2) = "Bisecci\'on"
 End If
k = k + 1
Loop Until dx < delta Or k > maxItr
End Sub
```

1.4 Método de la Secante

Aunque el método de la secante es anterior al método de Newton, a veces se hace una derivación de este método basado en la iteración de Newton cambiando la derivada $f'(x_k)$ por una aproximación, lo cual puede ser muy bueno pues para algunas funciones, como las definidas por integrales o una serie, en los casos en los que la derivada no es fácil de obtener. Aquí vamos a proceder igual que antes, con una idea geométrica. El método de la secante tiene orden de convergencia de al menos $p = 1.61803$ pero en un sentido que haremos más preciso al final de esta sección, este método es más rápido que el método de Newton.

Iniciando con *dos aproximaciones iniciales* x_0 y x_1, en el paso $k+1$, x_{k+1} se calcula, usando x_k y x_{k-1}, como la intersección con el eje X de la recta (secante) que pasa por los puntos $(x_{k-1}, f(x_{k-1}))$ y $(x_k, f(x_k))$

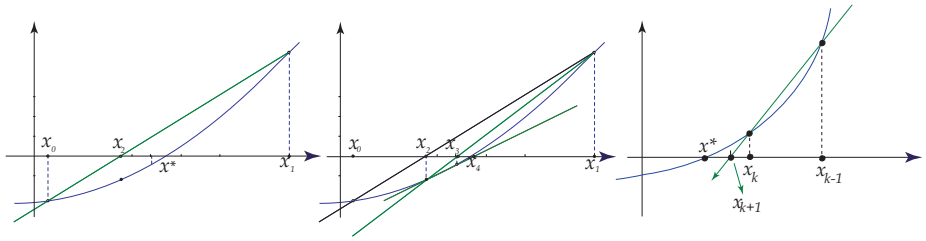


Figura 1.6 Método de la secante

Entonces, si $f(x_k) - f(x_{k-1}) \neq 0$,

$$x_{k+1} = \frac{x_{k-1} f(x_k) - x_k f(x_{k-1})}{f(x_k) - f(x_{k-1})}$$

Sin embargo, para cuidarnos del fenómeno de cancelación (cuando $x_k \approx x_{k-1}$ y $f(x_k) f(x_{k-1}) > 0$), rescribimos la fórmula como

$$x_{k+1} = x_k - f(x_k) \cdot \frac{(x_k - x_{k-1})}{f(x_k) - f(x_{k-1})}, \quad k \geq 1,$$

Aunque esta última versión no es totalmente segura, es al menos mejor que la anterior.

Como criterio de parada podemos usar (1.2).

EJEMPLO 1.3 Consideremos $P(x) = x^5 - 100x^4 + 3995x^3 - 79700x^2 + 794004x - 3160075$. En la figura (1.7) se muestra la gráfica de P con las primeras dos secantes usando $x_0 = 22.2$ y $x_1 = 17$.

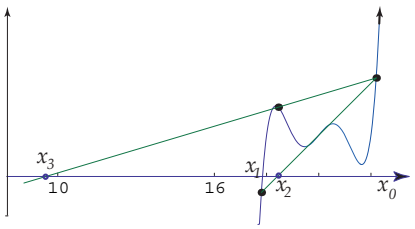


Figura 1.7

P tiene un cero, $x^* = 17.846365121...$, en el intervalo $[17, 22, 2]$. Vamos a aproximar este cero usando $x_0 = 17$ y $x_1 = 22.2$ y también con $x_0 = 22.2$ y $x_1 = 17$.

	$x_0 = 22.2$	$x_1 = 17$		$x_0 = 22.2$	$x_1 = 17$	
k	x_{k+1}	$\lvert x_k - x_{k+1}\rvert$	$f(x_i)$	x_{k+1}	$\lvert x_k - x_{k+1}\rvert$	$f(x_i)$
1	21.70509296	4.705	1.44	21.70509296	0.494	1.446
2	21.64664772	0.058	1.36	21.63787473	0.067	1.369
3	20.61844015	1.028	6.38	20.44319288	1.194	6.354
...						
23	17.84697705	0.013	0.02	21.2200121	0.429	3.503
24	17.84635118	0.000	-0.00	21.92553159	0.705	3.475
25	17.84636513	1.39×10^{-5}	5.79283×10^{-7}	111.120	89.194	627
26	17.84636512	1.37×10^{-8}	-1.86265×10^{-9}	21.925	89.194	3.475
...				...	...	
77				20.57831656	5110.34	6.410
78				20.57831656	1.06×10^{-14}	6.410

Tabla 1.3

La elección $x_0 = 22.2$ y $x_1 = 17$ muestra ser adecuada. Nos lleva a la aproximación correcta $x_{26} = 17.84636512$. En la otra elección hay un fenómeno de cuidado: La iteración 78 podría inducirnos a error pues nos presenta a $x_{78} = 20.57831656$ como un cero aproximado con error estimado 1.06×10^{-14} pero $f(x_{78}) = 6.410!$.

Notemos que hay varios intervalos (con extremos x_k, x_{k+1}) que no contienen un cero.

Convergencia

El método de la secante converge si elegimos aproximaciones iniciales adecuadas. La elección teórica puede ser en el sentido del siguiente teorema,

Teorema 1.2 *Sea x^* un cero simple de f. Sea $I_\varepsilon = \{x \in \mathbb{R} : |x - x^*| \leq \varepsilon\}$. Supongamos que $f \in C^2[I_\varepsilon]$. Para un ε suficientemente pequeño se define*

$$M_\varepsilon = \max_{s,t \in I_\varepsilon} \left| \frac{f''(s)}{2f'(t)} \right| \tag{1.3}$$

y asumamos que ε es lo suficientemente pequeño de tal manera que se cumpla $\varepsilon M_\varepsilon < 1$. Entonces, el método de la secante converge a una única *raíz $x^* \in I_\varepsilon$ para cualesquiera dos aproximaciones iniciales $x_0 \neq x_1$ en I_ε.*

Observe que el teorema exige que ε sea *suficientemente pequeño* de tal manera que la función de dos variables $g(s,t) = \left| \dfrac{f''(s)}{2f'(t)} \right|$ permanezca acotada en I_ε y que $\varepsilon M_\varepsilon < 1$.

1.5 Un Método Híbrido: Secante-Bisección

Uno de los problemas del método de la secante es que aunque un cero x^* de f esté entre las aproximaciones iniciales x_0 y x_1, no se garantiza que el método converja (si converge) a x^*, además de posibles "caídas".

EJEMPLO 1.4 Consideremos $f(x) = x^{20} - 1$. Si aplicamos el método de la secante con $x_0 = 0.5$ y $x_1 = 1.5$ tenemos un comportamiento no deseable, tal y como se muestra en la tabla (1.4)

n	x_n	Error estimado
2	0.5003007284	0.9996992716
3	0.5006013663	0.0003006379
4	25769.46097	25768.96037
5	0.5006013663041813	25768.96037
6	0.5006013663041813	0

Tabla 1.4 Problemas con el método de la secante

Para evitar comportamientos no deseables y asegurar que siempre la nueva iteración esté en un intervalo que contenga al cero que se quiere aproximar, combinamos el método de la secante con el método de bisección.

Para describir el procedimiento usamos $int\{b,c\}$ para denotar el más pequeño intervalo cerrado que contiene a b y c, es decir $int\{b,c\}$ es $[c,b]$ o $[b,c]$.

El algoritmo es como sigue: iniciamos con un intervalo $[a,b]$ en el que $f(a)$ y $f(b)$ tengan signos opuestos. Inicialmente el intervalo de bisección es $[c,b]$ pues iniciamos con $c = a$. En el proceso se actualizan a, b y c de la siguiente manera:

i. Si $f(a)$ y $f(b)$ tienen signos opuestos, se aplica una iteración del método de la secante (la cual actualiza a y b)

ii. Si $f(a)$ y $f(b)$ tienen signos iguales, se aplicará una iteración del método de la secante solo si se conoce que x_{n+1} estará en $int\{c,b\}$. Sino, se aplica bisección tomando como b el punto medio del intervalo $int\{c,b\}$.

iii. Finalmente se actualiza el intervalo $int\{c,b\}$ actualizando c por comparación del signo de $f(c)$ y $f(b)$.

En la figura (1.8) se muestra un caso particular con las primeras dos iteraciones.

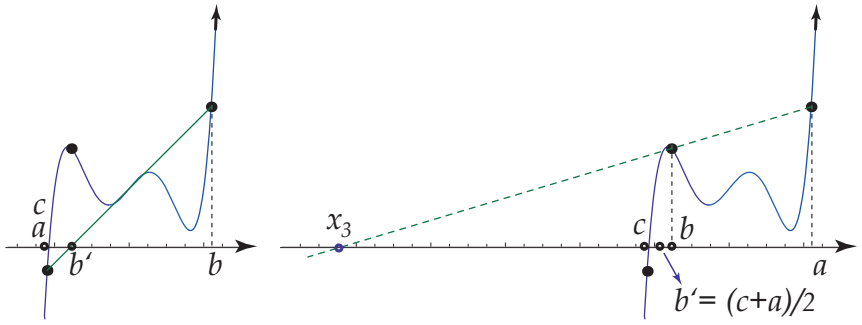


Figura 1.8

La figura (1.9) ilustra una posible actualización de c.

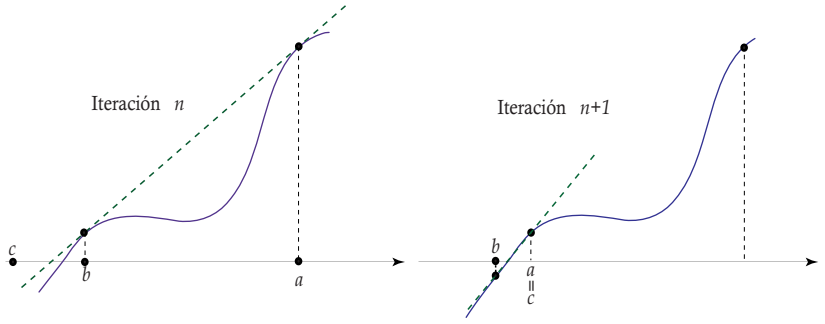


Figura 1.9

El algoritmo requiere saber a priori si $x_{n+1} \in int\{c,b\}$ pero sin aplicar la iteración (para evitar una posible "caída" evitamos la división por $f(x_n) - f(x_{n-1})$). Digamos que $int\{c,b\} = [p,q]$, entonces

$$x_{n+1} = x_n - f(x_n) \cdot \frac{(x_n - x_{n-1})}{f(x_n) - f(x_{n-1})} \in [p,q]$$

si y sólo si

$$p \leq x_n - f(x_n) \cdot \frac{(x_n - x_{n-1})}{f(x_n) - f(x_{n-1})} \leq q$$

si ponemos $\Delta f = f(x_n) - f(x_{n-1})$ entonces $x_{n+1} \in [p,q]$ si y sólo si se cumple alguna de las dos condiciones siguientes

i.) si $\Delta f > 0$ entonces $(p - x_n)\Delta f < -f(x_n)(x_n - x_{n-1}) < (q - x_n)\Delta f$

ii.) si $\Delta f < 0$ entonces $(p - x_n)\Delta f > -f(x_n)(x_n - x_{n-1}) > (q - x_n)\Delta f$

1.5.1 Algoritmo e Implementación en VBA Excel.

En este algoritmo,`TestSecante` hace la prueba $x_{n+1} \in int\{c,b\}$, es decir, determina si la siguiente iteración de secante estará en le intervalo $int\{c,b\}$.

SecanteItr(a,b) aplica una iteración del método de la secante con $x_0 = a$ y $x_1 = b$ y se procede a la actualización de a y b, es decir $a = x_1$ y $b = x_2$. Si los argumentos se reciben por referencia entonces son modificados en la subrutina. En código VBA Excel sería

```
Sub secanteItr(x0, x1)
Dim x2
 If f(x1) <> f(x0) Then
    x2 = x1 - f(x1) * ((x1 - x0) / (f(x1) - f(x0)))
    x0 = x1
    x1 = x2
 End If
End Sub
```

Algoritmo 1.2: Híbrido secante-bisección.

Entrada: una función continua f y $a,\ b$ tal que $f(a)f(b) < 0$ y δ
Resultado: una aproximación del cero.

$c = a$;
$n = 0$;
repeat
  if $f(a)f(b) < 0$ **then**
    Aplicar *SecanteItr*(a,b)
  else
    if `TestSecante` $=$*true* **then**
      Aplicar *SecanteItr*(a,b)
    else
      Aplicar bisección $b = b - 0.5(c-b)$;
  Intervalo para bisección;
  if $f(a)f(b) < 0$ **then**
    $c = a$
  $n = n+1$
until $(|c-b| < \delta(|b|+1))$ Or $(n >$ *maxItr* Or $f(b) < \delta)$;
return b;

Implementación en VBA Excel

Para hacer una hoja Excel, usamos como referencia la figura (1.10).

	A	B	C	D	E	F	G	H	I
1									
2			Híbrido Secante Bisección						
3									
4	f(x) =	x ^ 5 - 100x ^ 4 + 3995 x ^ 3 - 79700x ^ 2 + 794004x - 3160080							
5									
6							Método	Error	
7	a	b	delta	MaxItr	b	c	usado	Estimado	f(b)
8	21.34	22.45	5E-11	20	22.450000	21.340000	Secante1	0.097000404	-2.961297207
9					21.437000	22.450000	Secante1	0.903932827	-3.460053664
10					21.546067	22.450000	Bisección	0.451966414	-0.047000848

Figura 1.10 Hoja Excel para el híbrido secante-biseccion.

La subrutina principal es `HSecanteBiseccion`. Esta subrutina usa el evaluador de funciones clsmathparserhttp://digilander.libero.it/foxes/clsMathParser para leer y evaluar la función f.

También implementamos dos funciones,

`secanteItr(a,b,f)` y `x2estaEntre(c,b,a,f)`.

Ambas son llamadas desde `HSecanteBiseccion` y usan la función f. `secanteItr(a,b,f)` hace una iteración del método de la secante y actualiza a y b. La función `x2estaEntre(c,b,a,f)` devuelve 'true' o 'false' dependiendo de si la siguiente iteración (en la que $x_0 = a$ y $x_1 = b$) estará entre c y b.

```
Option Explicit
Private Sub CommandButton1_Click()
Dim fx, a, b, delta, maxItr
    fx = Cells(4, 2)
    a = Cells(8, 1)
    b = Cells(8, 2)
    delta = Cells(8, 3)
    maxItr = Cells(8, 4)
    Call HSecanteBiseccion(fx, a, b, delta, maxItr, 8, 5)
```

```
End Sub

Sub HSecanteBiseccion(fx, aa, bb, delta, maxItr, fi, co)
Dim f As New clsMathParser
Dim okf As Boolean
Dim test1 As Boolean
Dim test2 As Boolean
Dim a, b, n, c, fa, fb

okf = f.StoreExpression(fx)
If Not okf Then
    MsgBox ("error en f: " + f.ErrorDescription)
    Exit Sub
End If

a = aa
b = bb
c = a
fa = f.Eval1(a)
fb = f.Eval1(b)
n = 0
Do
    If Sgn(fa) <> Sgn(fb) Then
       Cells(fi + n, co) = b
       Cells(fi + n, co + 1) = c
       Call secanteItr(a, b, f) 'Actualiza a y b
       Cells(fi + n, co + 2) = "Secante1"
       Cells(fi + n, co + 3) = Abs(b - c)
    Else
       If x2estaEntre(b, c, a, f) Then
          Cells(fi + n, co) = b
          Cells(fi + n, co + 1) = c
          Call secanteItr(a, b, f) 'Modifica a y b
          Cells(fi + n, co + 2) = "Secante2"
          Cells(fi + n, co + 3) = Abs(b - c)
        Else
        'biseccin
          Cells(fi + n, co) = b
          Cells(fi + n, co + 1) = c
          b = b + 0.5 * (c - b)
          Cells(fi + n, co + 2) = "Biseccin"
          Cells(fi + n, co + 3) = Abs(b - c)
```

```
       End If
    End If
    fa = f.Eval1(a)
    fb = f.Eval1(b)
    Cells(fi + n, co + 4) = fb
    If Sgn(fa) = Sgn(fb) Then
       'nada
    Else: c = a
    End If
n = n + 1
Loop Until Abs(c - b) < delta Or n >= maxItr Or Abs(fb) < delta
End Sub

Sub secanteItr(x0, x1, f) 'Modifica x0 y x1
Dim x2, fx1, fx0
 fx1 = f.Eval1(x1)
 fx0 = f.Eval1(x0)
 x2 = x1 - fx1 * ((x1 - x0) / (fx1 - fx0))
 x0 = x1
 x1 = x2
End Sub

Function x2estaEntre(b, c, a, f)
Dim test1 As Boolean
Dim test2 As Boolean
Dim q, x0, x1, p, pf, Df, x2, fx1, fx0

q = Application.Max(b, c)
p = Application.Min(b, c)
x0 = a
x1 = b
fx1 = f.Eval1(x1)
fx0 = f.Eval1(x0)
Df = fx1 - fx0
pf = -1 * fx1 * (x1 - x0)
test1 = (Df > 0 And (p - x1) * Df < pf And pf < (q - x1) * Df)
test2 = (Df < 0 And (p - x1) * Df > pf And pf > (q - x1) * Df)

x2estaEntre = test1 Or test2
End Function
```

EJEMPLO 1.5 Considere la función $f(x) = x^5 - 100x^4 + 3995x^3 - 79700x^2 + 794004x - 3160080$. Sus ceros son $x^* = 18, 19, 20, 21, 22$. En la tabla (1.5) se muestra los valores de b y c al aplicar el método híbrido secante-bisección con $x_0 = 21.34$ y $x_1 = 22.45$ (para aproximar $x^* = 22$)

n	b	c	Método
1	21.4370004	21.34	Secante1
2	21.54606717	22.45	Secante1
3	21.99803359	22.45	Bisección
4	22.00708175	22.45	Secante2
5	21.99997119	21.99803359	Secante1
6	21.99999958	22.00708175	Secante1
7	22	22.00708175	Secante2

Tabla 1.5

En la línea 3 se observa que $b = 21.54606717$ y $c = 22.45$. En la siguiente iteración se usa bisección. Esto sucede porque si se aplica una iteración de la secante con los valores "actuales" $a = 21.4370004035587, b = 21.5460671728629$ (para los cuales no hay cambio de signo) entonces el valor futuro habría sido $x_{n+1} = 20.7894316058807$ y este valor *no* está en el intervalo actual $[c, b] = [22.45, 21.54606717]$, Entonces se usó bisección. Si no se hubiera hecho esto, entonces el método de la secante (clásico) hubiera convergido al cero $x^* = 21$ que no es el que está en el intervalo inicial.

EJEMPLO 1.6 En la tabla (1.6) se muestra los valores de b y c al aplicar el método híbrido secante-bisección, con $x_0 = 0.5$ y $x_1 = 2$, para aproximar el cero $x^* = 1$ de $f(x) = x^{20} - 1$. Recordemos es desempeño desastroso del método de la secante, que se mostró en la tabla (1.4), para este mismo problema.

n	b	c	Método	$\lvert b-c\rvert$
1	2.000000	0.500000	Secante1	1.43051×10^{-6}
2	0.500001	2.000000	Secante1	1.499997139
3	0.500003	2.000000	Bisección	0.749998569
4	1.250001	0.500001	Secante1	0.008646705
5	0.508648	1.250001	Secante1	0.73280628
6	0.517195	1.250001	Bisección	0.36640314
7	0.883598	1.250001	Bisección	0.18320157
8	1.066800	0.508648	Secante1	0.153141481
9	0.661790	1.066800	Secante1	0.293907403
1	0 0.772892	1.066800	Bisección	0.146953701
11	0.919846	1.066800	Bisección	0.073476851
12	0.993323	1.066800	Secante2	0.025927918
13	1.040871942	0.99332301	Secante1	0.004405128
14	0.997728138	1.040871942	Secante1	0.041636217
15	0.999235725	1.040871942	Secante2	0.040855271
16	1.000016672	0.999235725	Secante1	0.000764154
17	0.999999879	1.000016672	Secante1	1.66716×10^{-5}
18	1	1.000016672	Secante2	1.66716×10^{-5}

Tabla 1.6

Bibliografía

[1] Press, W.; Teukolsky, S.; Vterling, W.; Flannery, B. *Numerical Recipes in C. The Art of Scientific Computing.* 2da Ed. Cambridge Universite Press. 1992.

[2] Mathews, J; Fink, K.*Métodos Numéricos con MATLAB.* Prentice Hall, 3a. ed., 2000.

[3] Dahlquist, G. Björk, A. *Numerical Mathematics in Scientific Computation.* (`www.mai.liu.se/~akbjo`). Consultada en Mayo, 2006.

[4] Gautschi, W. *Numerical Analysis. An Introduction.* Birkhäuser, 1997.

[5] Stoer, J.; Burlish, R. *Introduction to Numerical Analysis.* 3rd ed.Springer, 2002.

[6] Ralston, A.; Rabinowitz, P. *A First Course in Numerical Analysis.* 2nd ed. Dover, 1978.

[7] Demidovich, B.; Maron, I. *Cálculo Numérico Fundamental*. 2da ed. Paraninfo (Madrid), 1985.

[8] Moler, K. *Numerical Computing with MATLAB.* (`http://www.mathworks.com/moler/`). Consultada en Enero, 2006.