Práctica III: Método de Bisección para encontrar las raíces de una función.

Sea f una función continua en $[a, b]$ que satisface $f(a)f(b) < 0$. Entonces f tiene, necesariamente, al menos un cero en (a, b). Supongamos por simplicidad que este cero es único, y llamémosle α.

Nota: en la práctica, para localizar un intervalo (a, b) en cuyos extremos la función cambie de signo se puede recurrir a hacer un gráfico de la función con la ayuda del comando fplot. Por ejemplo, para representar la función $f(x) = 2x^2 - 10x + 3$ haremos:

```
fplot('2x^2-10*x+3',[-6,6])
grid
```

El comando grid se utiliza para superponer una malla al gráfico. Obsérvese que en este comando el código de la función ha de introducirse entre comillas simples. La figura 1 muestra la representación gráfica obtenida.

Resulta muy práctico definir la función cuya raíz o raíces queremos calcular, mediante el comando inline. Ello nos va a permitir posteriormente pasar una función como argumento de otra, lo que va a ser muy útil en la práctica, como veremos más adelante:

```
ff=inline('2*x^2-10*x+3')
fplot(ff,[-6,6])
grid
```

La estrategia del método de bisección consiste en cada paso en dividir en dos partes iguales el intervalo dado y seleccionar el subintervalo donde f experimenta un cambio de signo. Concretamente:

1. Sean $I^{(0)} = (a, b)$ e $I^{(k)}$ el subintervalo seleccionado en la etapa k.
2. Elegimos como $I^{(k+1)}$ el subintervalo de $I^{(k)}$ en cuyos extremos f experimenta un cambio de signo. Siguiendo este procedimiento, se garantiza que cada $I^{(k)}$ seleccionado de esta forma contendrá a α.

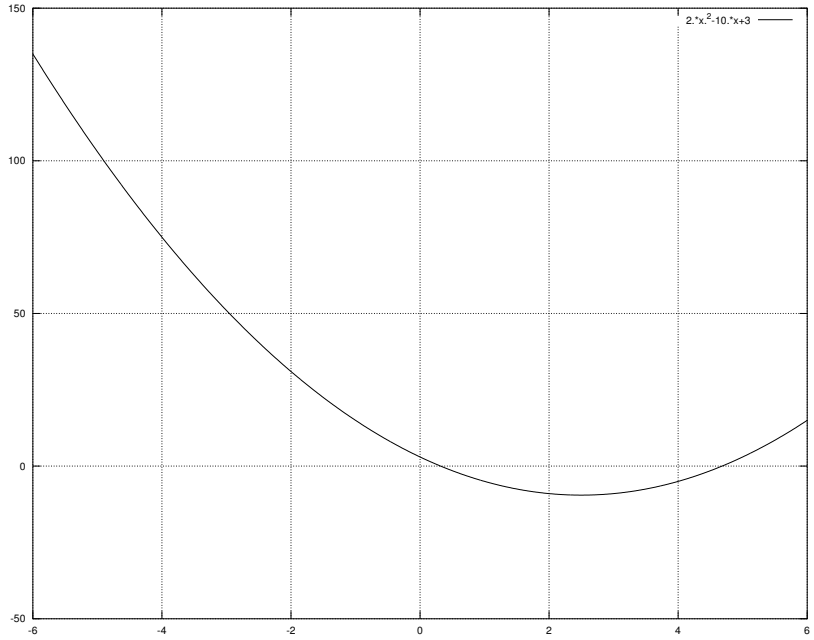

Figura 1: Representación gráfica de la función $f(x) = 2x^2 - 10x + 3$, obtenida con el comando fplot.

3. La sucesión $\left\{x^{(k)}\right\}$, de los puntos medios de estos subintervalos $I^{(k)}$, tenderá a α puesto que la longitud de los subintervalos tiende a cero cuando k tiende a infinito.

Desarrollo del algoritmo de bisección:

1. *Inicialización:* sea (a, b) un intervalo tal que $f(a)f(b) < 0$. Hacer $k = 0$, $a^{(0)} = a$, $b^{(0)} = b$, $I^{(0)} = (a^{(0)}, b^{(0)})$, $x^{(0)} = (a^{(0)} + b^{(0)})/2$. Fijar la tolerancia o error máximo ε.

2. En cada etapa $k \geq 1$ seleccionar el subintervalo $I^{(k)} = (a^{(k)}, b^{(k)})$ del intervalo $I^{(k-1)} = (a^{(k-1)}, b^{(k-1)})$ como sigue:

   a) Dado $x^{(k-1)} = (a^{(k-1)} + b^{(k-1)})/2$, si $f(x^{(k-1)}) = 0$ o $\left(b^{(k-1)} - a^{(k-1)}\right) < \varepsilon$ entonces $\alpha = x^{(k-1)}$ y el método termina. Mostrar α.

   b) En caso contrario:

      1) si $f(a^{(k-1)})f(x^{(k-1)}) < 0$ hacer $a^{(k)} = a^{(k-1)}$, $b^{(k)} = x^{(k-1)}$;
      2) si $f(x^{(k-1)})f(b^{(k-1)}) < 0$ hacer $a^{(k)} = x^{(k-1)}$, $b^{(k)} = b^{(k-1)}$.

   Definir $x^{(k)} = (a^{(k)} + b^{(k)})/2$ e incrementar $k = k + 1$

   Repetir el paso 2.

Nótese que cada subintervalo $I^{(k)}$ contiene al cero α. Además, la sucesión $\{x(k)\}$ converge necesariamente a α puesto que en cada paso la longitud $|I^{(k)}| = b^{(k)} - a^{(k)}$ de $I^{(k)}$ se divide por dos. Puesto que $|I^{(k)}| = (1/2)^k |I^{(0)}|$, el error en la etapa k satisface

$$|e^{(k)}| = |x^{(k)} - \alpha| < \frac{1}{2}|I(k)| = \left(\frac{1}{2}\right)^{k+1} |I^{(0)}| = \left(\frac{1}{2}\right)^{k+1} (b-a)$$

Para garantizar que $|e^{(k)}| < \varepsilon$, para una tolerancia dada ε, basta con llevar a cabo k_{min} iteraciones, siendo k_{min} el menor entero que satisface la desigualdad:

$$k > \frac{\ln\left(\frac{b-a}{\epsilon}\right)}{\ln(2)} - 1$$

Ejercicio: Implementar el algoritmo de bisección en MATLAB/OCTAVE en una función llamada biseccion. Esta función debe recibir como argumentos la función cuyo cero se quiere calcular, el intervalo donde va a buscar dicho cero y la tolerancia o error de aproximación ε. De esta forma, la función biseccion debe poder invocarse mediante:

biseccion(fun,a,b,tol)

donde la función fun se definirá previamente mediante un comando inline. Por ejemplo, para hallar el cero de la función $f(x) = 2x^2 - 10x + 3$ en el intervalo $[-2, 2]$, con error inferior a una milésima, el algoritmo de bisección deberá invocarse mediante:

ff=inline('2*x^2-10*x+3')

biseccion(ff, -2, 2, 0.001)

Solución:

Una manera sencilla de implementar el algoritmo de bisección es la siguiente: :(Pinchar aquí para descargar el archivo .m)

```
function raiz=biseccion(fun,a,b,tol)
% Resolución de la ecuación fun(x)=0 mediante el algoritmo de ↩
   bisección.
% Argumentos:
% fun: es la función cuya raiz se desea calcular (deberá definirse
%      previamente mediante un comando 'inline'.
% a: Extremo inferior del intervalo inicial
% b: Extremo superior del intervalo inicial
% tol: tolerancia; si |b-a|<tol el algoritmo se detiene
    if a>b  % Se comprueba que a<b
```

```
        aa=a; % Si b>a se permutan sus valores.
        a=b;
        b=aa;
    end%if
    xm=(a+b)/2;
    if fun(a)*fun(b)>0 % Comprueba que el signo difiere
                       % en los extremos del intervalo
        disp ('ERROR:Debe introducir otro intervalo')
    elseif abs(b-a)<tol
        raiz=xm;
    else
        while  ((fun(xm)~=0)&& abs(b-a)>tol)
            if fun(xm)*fun(b)<0
                a=xm;
            else
                b=xm;
            end% if
            xm=(a+b)/2;
        end% while
        raiz=xm;
    end% if
    raiz;
end% function
```

Aplicamos ahora este algoritmo a la función $f(x) = 2x^2 - 10x + 3$:

```
>>> ff=inline('2*x^2-10*x+3')
ff =
f(x) = 2*x^2-10*x+3
>>> biseccion(ff, -2, 2, 0.001)
ans = 0.32080
```

Para obtener el resultado con mayor precisión (más dígitos decimales), introducimos los comandos:

```
>>> format long
>>> biseccion(ff, -2, 2, 0.001)
ans = 0.320800781250000
```

Si observamos la figura 1 vemos que hay otra raiz entre 4 y 6. Para hallar su valor ejecutamos:

```
>>> biseccion(ff, 4, 6, 0.001)
ans = 4.67919921875000
```

Podemos comprobar que estos valores son efectivamente raíces de la función que hemos definido. Al aplicar la función ff a cada uno de ellos debemos obtener el valor 0:

```
>>> ff(0.320800781250000)
-0.00218152999877930
>>> ff(4.67919921875000)
-0.00218152999877930
```

Como vemos, no se obtiene 0. Ello se debe a que la tolerancia es de una milésima; si reducimos la tolerancia a una cienmillonésima, obtenemos un resultado más próximo a la verdadera raíz:

```
>>> biseccion(ff, -2, 2, 1e-8)
ans = 0.320550527423620
>>> ff(0.320550527423620)
ans = 7.02692171117292e-09
```

Y para la segunda raiz:

```
>>> biseccion(ff, 4, 6, 1e-8)
ans = 4.67944947257638
>>> ff(4.67944947257638)
ans = 7.02691949072687e-09
```