UNIVERSIDAD DEL BÍO-BÍO
FACULTAD DE CIENCIAS
DEPARTAMENTO DE MATEMÁTICA

Métodos Numéricos 220138
Laboratorio 3

Método de Bisección

Además de definir funciones mediante programas de tipo *function* se pueden definir mediante el comando `inline` objetos de tipo función que pueden ser evaluados, pero además son cómodos para ser pasados como argumentos de otras funciones:

```
>> clear
>> f=inline('x.^2.*sin(x.^3)', 'x');
>> f([1 2.5 3.8])
>> int = quad(f,0,1)
```

'x' indica la variable que se usa, en este caso es x. En el ejemplo anterior el último comando entrega la integral de la función f en el intervalo $[0, 1]$.

1. Haga un programa que calcule la raíz de una ecuación $f(x) = 0$ mediante el *método de bisección*. Los datos del programa deben ser la función f, los extremos del intervalo $[a, b]$ donde se busca la raíz, y la tolerancia `tol` con la que se desea calcular ésta. El programa debe tener una salida de error en el caso en que la función f no cambie de signo en el intervalo inicial.

2. Aplique el método para hallar soluciones, con una tolerancia menor que 10^{-4}, para los siguientes problemas en los intervalos que se solicita

   - $2 + \cos(e^x - 2) - e^x = 0$ en $[0.5, 1.5]$
   - $2x\cos(2x) - (x + 1)^2 = 0$ para $-3 \leq x \leq -2$ y para $-1 \leq x \leq 0$

3. Calcule con el programa anterior todas las raíces de las siguientes ecuaciones con error menor que 10^{-4}:

$$x^2 = 2, \qquad x^3 - 3x + 1 = 0 \qquad \text{y} \qquad \cos x = x.$$

Indicación: Para determinar la localización de las raíces es buena idea utilizar gráficos, por ejemplo, para la segunda función anterior tenemos.

```
>> f2=inline('x.^3-3*x+1','x'); %introduccion de la funcion
>> x=(-3:.1:3); %intervalo donde se desea graficar
>> plot(x,feval(f2,x),x,zeros(length(x),1)) %hay dos graficos, feval indica que
>>                                          %se evalue f2 en x.
```

Esquemas de punto fijo

Recordamos que toda ecuación $f(x) = 0$ es equivalente a un problema de punto fijo $g(x) = x$ para $g(x) = x + d(x)f(x)$, donde $d(x) \neq 0$ para todo x. La función $d(x)$ queda a "libre elección", y juega un rol importante en las aceleraciones del método.

Este algoritmo puede ser muy rápido, y de sencilla programación. Por ejemplo, para

$$g(x) = x - \frac{e^{-x}(12 + e^x x \sin(x^2))}{2x^2 \cos(x^2) + (1 + x)\sin(x^2)}$$

primero verificamos de manera gráfica que tiene un punto fijo, y simultaneamente escogemos la aproximación inicial

```
>> gi=inline('x-(exp(-x)*(12+exp(x)*x*sin(x^2))/
          (2*x^2*cos(x^2)+(1+x)*sin(x^2)))'); % es g
>> fplot(gi,[1.8 2.5]);
>> grid on;
>> axis([1.8 2.2 1.8 2.2]) %intervalo en x para cada variable
```

Del gráfico notamos que la intersección (solución buscada) está cerca de $x = 2$. Ahora obtenemos una aproximación con una tolerancia $tol = 0.5 \times 10^{-10}$

```
>> g=inline('[x-(exp(-x)*(12+exp(x)*x*sin(x^2))/
          (2*x^2*cos(x^2)+(1+x)*sin(x^2))),x]'); %es para graficar g y la identidad
   x0=2;      %esto se puede escribir como un programa rutero o como una funcion
   tol=0.5*10^{-16};
   for i=1:100
    xi=g(x0);
    iteraciones=[x0,xi]
    if (abs(x0-xi) <= tol),
       break;
    end
    x0=xi;
   end
   sol=x0  %muestra la solucion
   iter=i  %muestra el numero de iteraciones
```

A continuación mostraremos como opera el método, para ello contruiremos animaciones que nos permitan visualizar en forma geométrica diferentes situaciones. Por simplicidad el ejemplo que utilizamos es $g(x) = 0.8x + 0.3$. Los siguientes comandos muestran de manera gráfica que hay un punto fijo

```
>> clear all
>> x=-4:8;
>> plot(x,0.8*x+0.3,'r',x,x,'b'),
>> grid on;
>> title('la identidad esta en azul')
```

Para realizar la animación se necesita la función `anima.m` la cual se descarga de la página del curso. Como ejemplo tomamos $x_0 = 7$ y $x_0 = -4$.

```
>> g=inline('0.8*x+0.3');
>> x0=7;
>> anima(g,x0); %convergencia por descenso
>>
>> x0=-4;
>> anima(g,x0); %convergencia por ascenso
```

Las animaciones anteriores nos mostraron la formación de la escalera, que es clásica cuando una sucesión converge monótonamente. Utilizamos $g(x) = -0.8x + 3$ para mostrar la convergencia tipo "telaraña".

```
>> g=inline('-0.8*x+3');
>> x0=-4;
>> anima(g,x0);
```

Ahora vemos como opera el método en forma geométrica cuando él es divergente.

```
>> g=inline('-1.2*x-0.2');
>> x0=2;
>> anima(g,x0); % huida hacia arriba
>>
>> x0=0.5;
>> anima(g,x0); % huida hacia abajo
```

Tarea: (a) Utilizar la función `anima.m` para observar la convergencia o divergencia de $g(x) = 0.1x^2 + 1$ con aproximación inicial $x0 = -4$. (b) Adaptar la funcion anima.m para ver la convergencia o divergencia de $g(x) = x^2 - 0.5$ con $x0 = 0.5$ y $g(x) = x^2 - 1$ con $x0 = 0.5$ (la idea es tener una buena visualización). (c) Realice un código en matlab que, dado g, x_0, el número de iteraciones máximo, y una tolerancia prescrita, le permita realizar iteraciones de punto fijo.

Método de Newton

El archivo `newton.m` (bájelo de la página del curso) contiene el siguiente programa para el cálculo de la raíz de una ecuación $f(x) = 0$ mediante el *método de Newton-Raphson*:

```
function [raiz,iter]=newton(f,Df,x0,tol,maxit)

k=0;
raiz=x0;
corr=tol+1;
while (k<maxit) & (abs(corr)>tol)
    k=k+1;
    xk=raiz;
    fxk=feval(f,xk);
    Dfxk=feval(Df,xk);
    if (Dfxk==0)
        error('La derivada de la funcion se anula.')
    end
    corr=fxk/Dfxk;
    raiz=xk-corr;
    iter=k;
end

if (abs(corr)>tol)
    error('Se excedio el numero maximo de iteraciones.')
end
```

1. Calcule con el programa anterior todas las raíces de las siguientes ecuaciones con error menor que 10^{-4}:

$$x^2 = 2, \qquad x^3 - 3x + 1 = 0 \qquad \text{y} \qquad \cos x = x.$$

   Estos ejemplos ya los resolvió usando el método de bisección, compare la cantidad de iteraciones que requieren en ambos métodos. Encuentre las raíces con error menor que 10^{-12}, ¿aumentan mucho las iteraciones del algoritmo de Newton?.

2. Con el programa anterior resuelva los ejercicios del listado 2

El comando MATLAB `fzero` permite calcular la raíz de una ecuación $f(x) = 0$ cercana a un punto dado x_0. Este comando combina de manera automática un método inicial de convergencia garantizada con otro final de convergencia veloz.

1. Utilice el `help` de MATLAB para conocer la sintaxis del comando `fzero`.

2. Calcule, mediante este comando, las raíces de las ecuaciones del Ej. 1(a), primero con la tolerancia prefijada por el comando y luego con error menor que 10^{-12}.