Grado en Ingeniería Química – Matemáticas I – 1ºB
Práctica 2 – MatLab – Resolución numérica de Ecuaciones

0.- Preliminares

Recordad crear una carpeta en `c:\` , por ejemplo practica2 en la que guardar todo el trabajo.

Recordar decirle a MatLab que ponga esa carpeta como Carpeta de Trabajo:

```
>>cd c:\practica2
```

Recordar guardar el trabajo realizado durante la sesión mediante:

```
>> diary 'p2.txt'
```

y no olvidar al final hacer:.

```
>> diary off
```

Además, hay que guardar aparte las gráficas (formato BMP) y los M-ficheros que se creen.

En caso de duda sobre algún comando,por ejemplo el comando `finverse`) puedes teclear

```
>> help finverse
```

y te mostrará la ayuda sobre ese comando.

Para poder ver más decimales en los resultados necesitarás hacer:

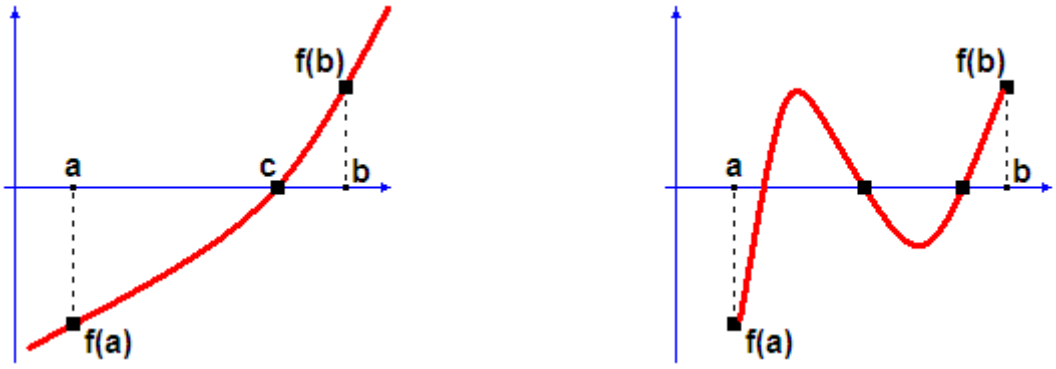
```
>>  format long
```

Ya vimos que cuando defino una función f(x) en la línea de comandos, para calcular f(3) en Matlab hago **`subs(f,x,3)`**

Cuando defino una función f(x) en un m-fichero y quiero calcular f(3) desde otro m-fichero haré **`feval(f,3)`**.

1.- Teorema de Bolzano:

Sea **f(x) continua en [a,b],** con **f(a)·f(b) < 0** (signos distintos)
Entonces EXISTE AL MENOS un c∈[a,b] t.q. f(c)=0



Es decir, f(x) tiene al menos un cero en ese intervalo, aunque puede tener varios
Nota: Si f'(x)≤0 en [a,b], ó f'(x)≥0 en [a,b] entonces ese cero es único.

2.-Metodo numéricos para la resolución de ecuaciones.

Se trata de resolver la ecuación f(x)=0, lo cual puede ser complicado analíticamente o imposible.
Por ejemplo, con: $f(x)=3x^5-4x^4+2x^3-x^2+5x-7$ ó $f(x)=e^x-2\cdot x$

Paso 1: Estableceremos intervalos [a,b] que contengan una única raíz de f(x)

Paso 2: Utilizaremos algún método numérico iterativo de forma que $\mathbf{x_k}$ será un aproximación a la solución exacta tras **k** iteraciones.

3.- Método de bisección.

Sea f : [a; b] → R de clase C^0 t. q. f(a)·f(b) < 0, por Bolzano sabemos que ∃c∈(a,b) t.q. f(c)=0

- Llamemos $[a_0,b_0]$ a nuestro intervalo [a,b]
- El error al tomar el punto medio como solución es error=$\frac{b-a}{2}$
- Si el error es mayor que el admitido haremos una iteración
  - Llamemos **m** al punto medio de $\mathbf{a_0}$ y $\mathbf{b_0}$ = **m** = $\frac{a_0+b_0}{2}$
  - Si f(m)=0, casualmente lo hemos encontrado: SOL = m. PARAMOS
  - Si f(m)≠0 elegimos entre { $[a_0,m]$ si es que $f(a_0)\cdot f(m)<0$, haciendo $[a_1,b_1]=[a_0,m]$; $[m,b_0]$ si es que $f(m)\cdot f(b_0)<0$, haciendo $[a_1,b_1]=[m,b_0]$
  - Calculo de nuevo el error con el nuevo intervalo
  - Si error > $error_{admitido}$
- SOL=punto medio el ultimo intervalo considerado

El error tras n iteraciones viene dado por **error**=$\frac{b-a}{2^{n+1}}$

Podemos saber el nº de iteraciones a priori, despejando el primer n∈N que cumpla:

error<$error_{admitido}$, es decir, $\frac{b-a}{2^{n+1}}$<$error_{admitido}$

Ejemplo: Hallar los ceros de $f(x)=x^2-2$

1º.- Crearemos un m-fichero en el que escribiremos la función con la que trabajaremos.

Así, para luego trabajar con otras funciones, sólo hay que modificar este m-fichero:

```
>> edit funcion
```

```
function y=funcion(x)
  y=x^2-2;  % ESCRIBIR AQUI LA FUNCION
```

2º.- Creamos otro m-fichero que contiene la función con el método de bisección:

```
>> edit biseccion
```

```
function [z,n]=biseccion(funcion,a,b,erroradmitido)
  n=0;                          % n lleva la cuenta del numero de iteraciones
  error=(b-a)/2;                % calcula el error al tomar el punto medio
  while (error>erroradmitido)   % mientras el error>error_admitido haremos una iteracion
    n=n+1;                        % incrementamos el numero de iteraciones realizadas
    m=(a+b)/2;                    % m es el punto medio
    ya=feval(funcion,a);          % ya=f(a)
    ym=feval(funcion,m);          % ym=f(m)
    if ym==0                      % si por casualidad hemos acertado salimos
      break;                        % salimos de la iteracion
    else                          % si ni hemos acertado
      if ya*ym<0                    % actualizamos el intervalo [a,b]
        b=m;
      else
        a=m;
      end
      error=(b-a)/2                 % calculamos de nuevo el error
    end
  end
  z=(a+b)/2;
end
```

3º.- Para usar el método bisección escribiremos:

```
>> [z,n]=biseccion('funcion', 0, 2, 0.000001)
```

Si nos fijamos le enviamos 4 argumentos:

- **'funcion'**: el nombre del m-fichero con la función de la que calcularemos los ceros
- **a**: el extremo izquierdo del intervalo

- **b**: el extremo derecho del intervalo
- **erroradmitido**: el máximo error aceptable

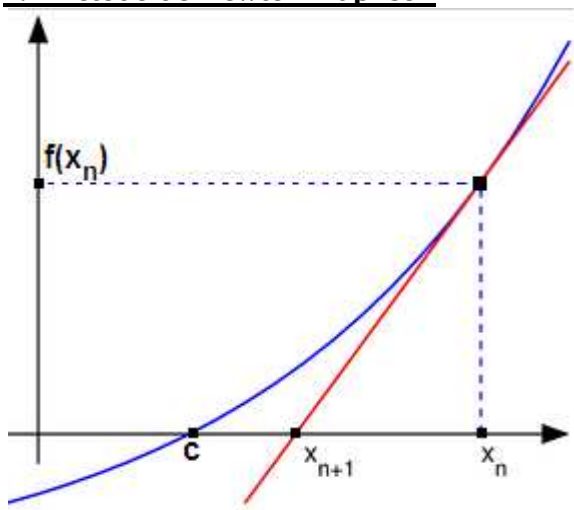
Nos devolverá 2 resultados:

- **z**: El valor aproximado del cero
- **n**: El nº de iteraciones realizadas

NOTA.1: Se supone que le enviamos un intervalo [a,b] que cumple con Bolzano. Por simplificar, nuestro programa no lo comprueba (aunque podría hacerlo).

NOTA.2: El método de bisección funciona siempre, pero converge lentamente (necesito muchas iteraciones).

4.- Método de Newton-Raphson



- Sea f∈$C^{2)}$[a,b], con $f(a)\cdot f(b) < 0$
  f'' *signo cte en [a,b]*
- Partimos de una aproximación $\mathbf{x_0}$ cercana al valor exacto **c**, que cumpla $\mathbf{f(x_0)\cdot f''(x_0) > 0}$
- Calcularemos la recta tangente en el pto $(x_n, f(x_n))$
- El punto de corte de la tangente con el eje de abcisas, $\mathbf{x_{n+1}}$, será una aproximación a **c** mejor que x_n.

$f'(x_n) = m = \frac{f(x_n)-0}{x_n - x_{n+1}}$

Despejando: $\boxed{x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}}$

Cota del error: Buscamos $\mathbf{m} \le |f'(x)|$, $\mathbf{M} \ge |f''(x)|$ $\forall x \in [a,b]$,
(lo mejor es hacer la gráfica de $|f'(x)|$ y de $|f''(x)|$ y elegir m y M)

Hay dos formas de medir el error tras la iteración k

a) $error_k < \frac{|f(x_k)|}{m}$, *(esta cota del error es más restrictiva que la siguiente)*

b) $error_k < \frac{M}{2\cdot m}\cdot (x_k - x_{k-1})^2$ (esta condición suele simplificarse y paramos de iterar cuando $|x_k - x_{k-1}| <$ ε prefijado)

Ejemplo: Hallar los ceros de $f(x)=x^2-2$. Probaré en el intervalo [0,2]

```
>> f=x^2-2
      f = x^2 - 2
>> subs(f,x,0)*subs(f,x,2)
      ans = -4                    % negativo, seguro que hay un cero
>> fp=diff(f,x)                   % fp es f'(x)
      fp = 2*x
>> fpp=diff(f,x,2)                % fpp es f''(x)
      fpp = 2
```

Veo que f″(x) tiene el mismo signo en todo el intervalo, si no lo veo a ojo, hago la gráfica de f″(x)

Ahora busco un candidato para x_0. Pruebo con $x_0=1$

```
>> x0=1
>> subs(f,x,x0)*subs(fpp,x,x0)
      ans = -2
```

No me sirve, pruebo con $x_0=1'5$

```
>> x0=1.5
>> subs(f,x,x0)*subs(fpp,x,x0)
       ans = 0.5000
```

Busco un valor para m: Dibujo |f'(x)| en el intervalo [0,2]

```
>> fplot('abs(2*x)',[0,2])
```

Como veo que EL MÍNIMO DE |f'(x)| ES CERO reconsidero el intervalo, porque m debe ser ≠0
Probaré en el intervalo [0'5,2]

```
>> subs(f,x,0.5)*subs(f,x,2)
       ans = -3.5000                    % negativo, seguro que hay un cero
```

Mantengo el x_0=1'5 que había comprobado antes que me servía como x_0
Busco un valor para m: Dibujo |f'(x)| en el intervalo [0'5,2]

```
>> fplot('abs(2*x)',[0.5,2])
```

Viendo la gráfica, tomo m=0'9

```
>> m=0.9
```

A iterar por ejemplo hasta que error < 0'000001 =10^{-6}

```
>> x1=x0-subs(f,x,x0)/subs(fp,x,x0), error=abs(subs(f,x,x1))/m,eps=abs(x1-x0)
       x1 = 1.416666666666667
       error = 0.007716049382716
       eps = -0.083333333333333
>> x2=x1-subs(f,x,x1)/subs(fp,x,x1), error=abs(subs(f,x,x2))/m, eps=abs(x2-x1)
       x2 = 1.414215686274510
       error = 6.674783203190296e-006
       eps = -0.002450980392157
>> x3=x2-subs(f,x,x2)/subs(fp,x,x2), error=abs(subs(f,x,x3))/m, eps=abs(x3-x2)
       x3 = 1.414213562374690
       error = 5.011793449385651e-012
       eps = -2.123899820016817e-006
```

CONCLUSIONES:

- Nos ha costado un poco más elegir y comprobar el intervalo
- Nos ha costado elegir el x_0 inicial cercano a la solución exacta c.
- Pero converge muy rápido (en la 2ª iteración, el error era de $6\cdot10^{-6}$, en la 3ª ya era de $5\cdot10^{-12}$)
- Podríamos hacer una función que iterase automáticamente cuando sepáis programar más.

5.- Ejercicios

5.1.- Resolver: $\mathbf{x=e^{-x}}$, con 4 decimales exactos

a) Halla previamente un intervalo donde esté garantizado que haya una única solución (no vale con una gráfica)
b) Antes de utilizar el método de bisección calcula cuantas iteraciones son necesarias
c) Calcúlalo con el método de bisección y compara el nº de iteraciones con el apartado b
d) Ahora con el método de Newton-Raphson.
e) Indica las diferencias y problemas encontrados al aplicar un método y otro.

5.2.- Halla el punto de corte de $\mathbf{y=x^2}$ con $\mathbf{y=e^x}$ Debes decirme: *el punto de corte es el* (….., …..)

5.3.- a) Demostrar que la ecuación $x = \tan(x)$ posee infinitas raíces reales.
b) Utilizar el método de bisección para aproximar las dos 1as raíces positivas (6 decimales)

5.4.- Utiliza el método de bisección para hallar $\sqrt[3]{17}$ (*observa que en los ejemplos he hallado* $\sqrt{2}$)

5.5.- Encontrar la solución positiva de $2\cdot\sin(x) = x$ con los dos métodos.

5.6.- La función $f(x) = \frac{4\cdot x-7}{x-2}$ se hace cero en x = 7/4.

Utilizar el método de Newton para resolver f(x) = 0 con las aproximaciones iniciales:
a) x_0 = 1'6; b) x_0 = 1'5; c) x_0 = 3 Explicar qué ocurre en cada caso.

5.7.- Calcular 4 iteraciones del método de Newton si $f(x) = x^{1/3}$ tomando x_0 = 0;1. ¿Qué ocurre?

5.8.- a) Halla un intervalo donde encuentres un cero de $f(x) = x^3+x-3$
b) Demuestra que es ese cero es único. Hállalo con los 2 métodos