Tema 4: Ecuaciones no lineales

Ejercicio 1. Se considera la función $f(x) = x^2 - e^{-x} - 1$.

1. Comprobar gráficamente y verificar analíticamente que dicha función tiene una raíz en el intervalo [1, 2].

2. Aplicar el método de Newton para calcular dicha solución tomando como valor inicial el punto medio del intervalo e iterando 7 veces. El código empleado debe proporcionar (comando fprintf): el número de iteración k (%d), la solución aproximada x_k con 16 decimales (%.16f), una estimación del error e_k (%0.2e) y el nº de cifras decimales correctas cif_dec(%d) para cada iteración.

Nota: Para estimar el error e_k en la iteración k-ésima se puede utilizar $e_k \approx |x_{k+1} - x_k|$, siendo x_k e l valor obtenido en la iteración k-ésima.

3. A la vista de los resultados obtenidos en el apartado anterior ¿cuál es el orden de convergencia del método en éste caso? Justificar la respuesta.

Ejercicio 2 El polinomio $p(x) = x^3 - x^2 - x - 1$ tiene una única raíz real dada por s = 1.839286755214161. Se van a aplicar los siguientes métodos iterativos para calcular la solución empezando con x0=1.5:

- Método 1: Método de Newton, llamar x_n a las iteraciones
- Método 2: $y_{n+1} = \sqrt[3]{1 + y_n + y_n^2}$

1. Implementar la aplicación de los dos métodos para que iteren 10 veces y se guarden los errores de cada iteración en un vector (error1 y error2) y en cada iteración se muestren los resultados (fprintf) con el siguiente formato:

- Número de iteración (%2d).
- Estimación de s con 16 decimales (%.16f)
- Error relativo en notación científica con 2 decimales (%.2e)

La salida será del tipo:

Iter: 1 x = 1.8211496982469146 Error rel: 9.86e-003
Iter: 2 x = ... Error rel ...

- En cada caso calcular el vector de cifras decimales correctas (cif1 y cif2) ¿Cuántas cifras decimales correctas proporciona la iteración final de cada método? ¿Cuántas iteraciones son necesarias para alcanzar la precisión de la máquina con el método de Newton?
- Representar gráficamente (semilogy) los vectores de errores relativos obtenidos en cada iteración.

2. Orden de convergencia.
- ¿Qué tipo de convergencia proporciona el método 2? En una convergencia lineal se verifica $e_{n+1} / e_n \approx K$. Determinar el valor de K para este método y usarlo para estimar cuántas iteraciones son necesarias para ganar una cifra decimal. ¿En qué iteración se alcanzarían 10 cifras decimales correctas?
- Como se aprecia en los resultados y como sabemos que el método de Newton tiene convergencia cuadrática $|e_{n+1}| \approx K \cdot |e_n|^2$, determinar el valor de la constante K a partir de las estimaciones del error obtenidas.

Ejercicio 3. Dada la ecuación $f(x) = x^2 - e^{-x} = 0.$

a) Realizar una gráfica de la función f(x) en el intervalo [-2, 2]. A partir de la gráfica, determinar un intervalo de longitud 1 donde se encuentre la raíz. Demostrar analíticamente que en dicho intervalo existe al menos una raíz.

b) Método 1: Implementar y ejecutar el método $x_{n+1} = \sqrt{e^{-x_n}}$ para encontrar la raíz de f(x) partiendo de $x_0 = 1$ El método debe iterar hasta que el error $e_n \approx |x_n - x_{n-1}| < 1e-10$. El código deberá en cada iteración imprimir el número de iteración, la aproximación de la raíz obtenida y el error, utilizando el formato fprintf(Iter %d Sol %.15f Error %.2e\n',…..).

c) Método 2: Implementar y ejecutar el siguiente método

$$z_0 = 0.5, x_0 = 1, \begin{cases} z_{n+1} = z_n - \dfrac{f(z_n)}{f'(x_n)} \\ x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)} \end{cases}$$

que calculará la solución aproximada x_n de la raíz, iterando hasta que el error $e_n \approx |x_n - z_n| < 1e-10$. El código deberá en cada iteración imprimir: el número n de iteración, la solución x_n obtenida y una estimación del error utilizando el formato 'Iter %d Sol %.15f Error %e\n'

Ejercicio 4. (Diapositivas clase) Se va a aplicar el método de Newton para calcular la raices indicadas de las siguientes funciones:

a) f(x) = sin(x)-1, cuya raiz es s = $\pi/2$ (raíz doble).

b) f(x) = $x^2 - 2$, cuya raíz es s = $\sqrt{2}$ (raíz simple).

1. Representar gráficamente las funciones en un intervalo que contenga a las raíces y ver que en el primer caso se trata de una raíz doble y en el segundo de una raíz simple.

2. Como vimos en el Tema 1, el error absoluto está directamente relacionado con el número de decimales correctos (d). Si empleamos estrategia redondeo al más próximo (en vez de truncamiento), se demuestra que $E_{abs} \approx \frac{1}{2}10^{-d}$ y por lo tanto, $d \approx -\log_{10}(2E_{abs})$. Aplicar el método de Newton durante 10 iteraciones en los casos anteriores (en los que conocemos el valor de la raiz) y calcular el nº de decimales exactos. Mostrar los resultados del valor aproximado, del error y del nº de decimales exactos en cada iteración (comando fprintf). Comentar los resultados y estudiar la velocidad de convergencia del método en cada caso.

Ejercicio 5. Se considera la función f(x)=x-exp(-x)-1.

1. Verificar y visualizar que dicha función tiene una raíz en el intervalo [1 , 2] .
2. Aplicar el método de Newton para calcular dicha solución iterando 8 veces y tomando como valor inicial el punto medio del intervalo. El código empleado debe proporcionar: el número de iteración k (%d), la solución x_k obtenida en cada iteración con 16 decimales (%.16f), y una estimación del error e_k cometido en cada iteración (formato: %5.2e). Incluir código y resultados pedidos.

Nota: Para estimar el error e_k en iteración k-ésima se puede utilizar $e_k \approx |x_{k+1} - x_k|$.

3. A la vista de los resultados obtenidos en el apartado anterior: ¿Cuántas iteraciones son necesarias para obtener un error aproximado de 0?¿Cuál es el orden de convergencia del método en éste caso? Justificar la respuesta.

Ejercicio 6. Se quiere resolver la ecuación x=cos(x).

1. Resolver la ecuación x=cos(x) es equivalente a calcular una raíz (o cero) de la función **f(x)=x-cos(x).**
- Crear la función fun.m de Matlab que implemente el cálculo de la función f(x)= x-cos(x) en x: recibe como argumento de entrada x y devuelve el valor de la función en x (llamamos f), y opcionalmente el de su derivada en x (le llamamos fp):

```
function [f,fp]=fun(x)
f = x - cos(x);  %  Devuelve el valor de la función  f(x) = x-cos(x)
if nargout==1,   end  %  Si el nº de argumentos de salida es 1, se interrumpe el programa.
% En otro caso, devuelve el valor de su derivada en x.
fp = 1+sin(x);   % Cálculo de la derivada de f en x
end
```

Observación: 1. La segunda opción la utilizaremos en el método de Newton que requiere evaluar la función y su derivada en un punto..
2. El puntero a la función @fun (en este caso) o bien la cadena de caracteres del nombre de la función 'fun', (en este caso) se pasará como argumento a las funciones que implementen los diferentes métodos de resolución de ecuaciones no lineales.

- Utilizando la función fun.m, ver gráficamente que f(x) tiene una raíz en el intervalo [0, 1] (pintar también el eje x) y comprobar analíticamente.

2. Se va a aplicar el Método de Newton para calcular la raiz de f(x).
- Crear la función newton.m que implemente el cálculo de N iteraciones (3er argumento de entrada) del método de Newton aplicado a la función fun (1er argumento de entrada) empezando en el punto x0 (2º argumento de entrada):

```
function s=newton(fun,x0,N)    (*)
% fun   puntero a la función: debe devolver el valor de f y f' en un punto o vector.
% x0  punto inicial
% N número iteraciones
for k=1:N,
  [f fp]=fun(x0); % Llamada a función fun para obtener valores de f(x0) y f'(x0)  (*1)
  if f==0, break; end  % Ya he obtenido la raiz y termino
  x1 = x0-(f/fp);      % Iteracion de Newton.
  %  aquí introduciremos estimación del error
  fprintf('%2d -> %18.15f\n',k,x1);  % Volcar datos iteración. Añadir que muestre estimación
error formato .%0.2e
  x0=x1;   % Actualizar x0 con el ultimo valor x1 para volver a iterar.
end
s = x1;  % valor aproximado de la solución dado por último término de la sucesión.
end
```

(*) Si la llamada a la función es usando la cadena de caracteres 'fun' por ejemplo s=newton('fun',0.5,10), requeriría utilizar el comando feval en la función newton en sentencia (*1). En caso de usar el puntero @fun por ejemplo s=newton(@fun,0.5,10) no requiere el uso de feval en la sentencia (*1).

- Modificar la función anterior para que en cada iteración estime también el error a través de e_k~abs(x_k - x_{k+1}) y muestre los resultados de cada iteración: k,x_k,e_k (e_k formato 0.2e). Aplicar la función newton anterior iterando 10 veces con x0=0.5 ¿Cuál es el error del resultado final? ¿Cuántas iteraciones habrían sido suficientes?
- Modificar la función anterior, creando una función **function s=newton2(fun,x0,N,tol)**, para que itere hasta alcanzar la solución con una precisión determinada (error<=tol, siendo tol el 4º argumento de la función) y en todo caso el número de iteraciones no sea mayor que N. Aplicar dicha función con N= 100, x0=0.5 y tol=1e-10 ¿Cuántas iteraciones realiza el método?

Ejercicio 7. Aplicar el método de la bisección para calcular la raíz de la ecuación del Ejercicio 3 y comparar los resultados con los obtenidos mediante el método de Newton.
1. Crear la función biseccion.m que implementa el cálculo de N iteraciones (4er argumento de entrada) del método de la bisección aplicado a la función fun (1er argumento de entrada) empezando en el intervalo [a,b] (2º y 3er argumentos de entrada):

```
function s=biseccion(fun,a,b,N)     % Método de la bisección.
% Argumentos Entrada: fun puntero a la función;  [a,b] intervalo donde está la solución
%  N: número máximo de iteraciones
% Argumento Salida:   s es la aproximación de la raiz
fa=fun(a);fb=fun(b); % Evalúo la función f en a y en b.
if fa*fb > 0
   fprintf('Método no aplicable en intervalo [a,b]\n')
 end
for i=1:N
   c=(a+b)/2
   fc=fun(c);    %evalúo la función en punto medio intervalo
   if fa*fc < 0
      b=c;     %fb=fc; (no es necesario)
   else
      a=c;
      fa=fc;
   end
end
```

```
s=a % pueder ser s=b o s=(a+b)/2,
fprintf('La raiz aproximada es %12.8f\n',s)
end
```

- Aplicar en el intervalo [0,1] para que itere 10 veces ¿Cuál es el error de la última solución?

2. Modificar el código anterior para que el método itere hasta calcular la solución con una precisión (tol) dada (error<=tol) y en todo caso el número de iteraciones no sea mayor que N. Como argumentos de salida se espera que proporcione la solución aproximada s y el nº n de iteraciones realizadas

```
function [s,n]=biseccion2(fun,a,b,tol,N)
% Entrada: como en función anterior, se añade tol precisión deseada.
%          N: nº max de iteraciones (opcional, por defecto 20)
% Salida : s, estimación de la raíz y n, número iteraciones realizadas.
s=NaN;n=NaN;
if nargin==4, N=20; end
fa=fun(a); fb=fun(b); % Evaluación de fx en extremos intervalo
if (fa*fb>0), fprintf('ERROR: Método no aplicable en intervalo [a,b]\n'); end
n=1;
while ( ((b-a)/2 > tol) & (n<=N) ) % Condiciones salida
s = (a+b)/2;
fs=fun(s);   % Evalúa la función en la estimación de la raiz
if (fs*fa <0), b=s; fb=fs; else a=s; fa=fs; end
n=n+1;   end
s = (a+b)/2;     % Mejor hipótesis dado el intervalo final [a,b]
end
```

Ejercicio 8. Se considera la ecuación $f(x) = e^x - 3x^2 = 0.$

1. Dibujar la gráfica de la función en el intervalo [-1, 4] ¿Cuántas raíces tiene la función en ese intervalo?
2. Se considera el siguiente método iterativo para el cálculo de raíces de la función f(x)

$$x_{n+1} = x_n - \frac{e^{x_n} - 3x_n^2}{e^{x_n} - 6x_n}$$

tomando como valores iniciales $x_0 = -1/2, x_0 = 1, x_0 = 4$. Suponemos que el error del método se estima mediante $e_n = |x_{n+1} - x_n|$. Codificar un script que implemente el método iterativo y encuentre las raíces de la función, arrancando en cada uno de los valores iniciales dados, deteniéndose cuando $e_n < 10^{-13}$. El script debe proporcionar los datos necesarios para completar 3 tablas como la siguiente (una tabla por cada valor de arranque). Aquellos campos de la tabla que no estén definidos se dejarán en blanco.

n	x_n	$e_n = \lvert x_{n+1} - x_n \rvert$	$e_n / (e_{n-1})^2$
1			
2			
3			

3. Sabiendo que el método es de convergencia cuadrática (o de orden 2), estimar un valor aproximado de la constante K que verifica la relación

$$K \approx e_n / (e_{n-1})^2$$

Calcular el valor de K para cada valor inicial y completar la siguiente tabla

x_0	K
-1/2	
1	
4	

¿A cuál de las 3 raíces el método converge más rápido?, ¿y a cuál más lento?