1 Ecuaciones no lineales

En esta práctica se estudian los métodos de iteración funcional y se repasa de nuevo la definición de funciones simples.

1.1 Definición de funciones simples

Se puede utilizar la instrucción *inline* para crear funciones desde el teclado, sin necesidad de generarlas con el editor en un fichero.m. Su sintaxis es la siguiente:

Nombre_funcion = **inline(***'definicion entre comillas simples***')**

Ejemplo:

```
» f=inline('sqrt(exp(x))')
» p=f(4)          % Evaluación en el punto 4
» X=linspace(-1,4);Y=f(X);plot(X,Y) % Representación en el intervalo [-1,4]
```

1.2 Funciones y variables simbólicas

Las funciones definidas hasta ahora son numéricas, puesto que las variables de entrada son valores numéricos. También es posible definir funciones que dependan de variables simbólicas:

```
» syms z real
» F=sqrt(exp(z))
» p=subs(F,z,4)          % Evaluación en el punto 4
» ezplot(F,[-1,4])       % Representación en el intervalo [-1,4]
```

1.3 Métodos Analíticos

MATLAB dispone de la función **solve** que resuelve, cuando es posible, una ecuación simbólica de forma analítica. Su sintaxis es

solve(f, 'x') Resuelve la ecuación *f* en la variable *x*

Ejercicios recomendados:

1. Utilizar la función **solve** para obtener, si es posible, las raíces de las siguientes funciones, representándolas previamente
   a) $f(x)=x\cos(x)-e^x+1$ en $[-1,1]$.
   b) $f(x)=1/2+x^2/4-x\,\mathrm{sen}(x)-\cos(2x)/2$ en $[-1,1]$
   c) $f(x)=x\ln(x+1) - x/2 - x\ln(2) + \frac{1}{2}$ en $[0,2]$

1.4 Métodos de punto fijo o de iteración funcional

MATLAB no tiene funciones específicas de punto fijo, por lo que es necesario definírselas. Ello conlleva que no tiene funciones propias para calcular raíces de multiplicidad impar (salvo las polinómicas).

Repasando, la convergencia de los métodos de punto fijo exige que se cumplan las siguientes condiciones:

Teorema I

Si g(x) es continua en [a,b] y verifica que g(a)≥a y g(b)≤b, entonces la función g(x) tiene un punto fijo en [a,b]. Si además g'(x) está definida en (a,b) y existe una constante positiva k<1 tal que para cualquier punto x del intervalo |g'(x)| ≤k<1, entonces el punto fijo es único.

Si bien se pueden realizar los cálculos manualmente, resulta mejor aprovechar la potencia gráfica de MATLAB utilizando funciones simbólicas.

Despejemos de diferentes formas la primera función a) del ejercicio 1 recomendado

$$f(x)=x\cos(x)-e^x+1 \quad g_1(x)=\frac{e^x-1}{\cos(x)} \quad g_2(x)=\arccos\left(\frac{e^x-1}{x}\right) \quad g_3(x)=\ln\left(x\cos(x)+1\right)$$

Vamos a ver de forma gráfica si cumple las condiciones del teorema en [-1,1].

```
» clear all
» syms x real;
» g1=(exp(x)-1)/cos(x); % Expresion de la funcion
» dg1=diff(g1,x)  % Calculo de su derivada
» subplot(2,1,1);ezplot(g1,[-1,1]); title('Funcion');grid
» subplot(2,1,2);ezplot(dg1,[-1,1]);title('Derivada');grid
```

En la imagen superior vemos la función y $f([-1,1])\not\subset ([-1,1]$, esto es, $\forall x\in[-1,1]: f(x)\notin [-1,1]$.

En la imagen inferior está la derivada, y $f'([-1,1])\not\subset ([-1,1]$, esto es, $\exists x\in[-1,1]: |f'(x)|>1$

No cumple por tanto ninguna de las condiciones del teorema, y la sucesión no convergerá.

Ejercicios recomendados:

2. Comprobar si las funciones $g_2(x)$ y $g_3(x)$ verifican las condiciones del Teorema de punto fijo en algún subintervalo del [-1,1].
3. Verificar que la siguiente función es de punto fijo y está asociada a la c) del ejercicio 1.

$$g(x)=\frac{x-1}{2\ln\left(\frac{x+1}{2}\right)}$$

4. Descargar la función T22_pfijo.m y verificar que se ha programado según el algoritmo siguiente:

   Entrada: nombre del archivo que contiene la función g(x), valor inicial 'z', error admisible 'ex' de la raíz y número máximo de iteraciones 'niter'.

   Salida: vector x de sucesiones de aproximaciones a la solución.

Paso 1 Definir $x_{niterx1}$ y hacer $x_1=z$

Paso 2 Repetir para k desde 2 hasta niter los Pasos 5-6

Paso 3 Hacer $x_k=g(x_{k-1})$;

Paso 4 SI $|x_k-x_{k-1}|<ex$ FINALIZAR BUCLE

Paso 5 SI k>niter, MENSAJE 'Finde iteraciones sin convergencia'

Paso 6 SINO eliminar elementos $x_{k+1}, x_{k+2}, \ldots x_{niter}$

Comprobar su funcionamiento usando como función de iteración la del ejercicio anterior para encontrar la aproximación con un error menor que 0'001 tomando un valor inicial aleatorio en el intervalo [0,2] y realizando un máximo de 100 iteraciones.

5. Utilizar T22_pfijo para aplicar el método de Newton modificado (raíz doble) a la ecuación c) del ejercicio 1 ¿Cuál es el intervalo de convergencia del método?

1.5 Aplicación con MATLAB

Aunque, como ya se ha indicado, MATLAB no tiene programas de punto fijo o que permitan obtener raíces pares, sí tiene funciones que calculan mínimos y se pueden aprovechar para estos casos teniendo en cuenta que toda raíz par genera en su entorno un máximo o mínimo relativo. Para ello se utiliza la instrucción ***fminbnd*** que encuentra el mínimo de una función *fun* , definida de forma numérica, en un intervalo *[x1,x2]*. Como toda raíz doble define un mínimo o un máximo relativo en su entorno, se debe utilizar la función o su opuesta dependiendo del caso (se recuerda que el máximo de una función se da para el mismo valor de *x* que el mínimo de la función opuesta).

[x,fval,exitflag,output] =fminbnd(fun,x1,x2,options)

Las opciones del método se dan en la variable *options*, utilizada en la práctica anterior (ver *fzero*) y que se asignaban mediante la instrucción *optimset* (hacer *doc fminbnd* para ver las opciones válidas).

Así, las raíces del ejercicio 1 se podrían obtener mediante:

```
» F1=inline(vectorize(-f1));
» opt=optimset('Display','none','TolX',1e-5,'MaxIter',100)
» [x1,fval1,exitflag1,output1] =fminbnd(F1,-0.4,0.4,opt)
» F2=inline(vectorize(f2));
» [x2,fval2,exitflag2,output2] =fminbnd(F2,-1,1)
» F3=inline(vectorize(f3));
» [x3,fval3,exitflag3,output3] =fminbnd(F3,0,2)
```

2 Anexo: Instrucciones utilizadas o relacionadas

2.1 Resolución de ecuaciones no lineales

solve('ecuación', 'x')	Resuelve la ecuación en la variable x
roots(V)	Da las raíces del polinomio cuyos coeficientes son las componentes del vector V.
fzero('f',z)	Busca la raíz de *f(x)* próxima a *z*
optimset	Define las opciones del método iterativo que utiliza la instrucción ***fzero***
fminbnd	Obtienen el mínimo de una función unimodal

2.2 Funciones polinómicas

poly(v)	Da el polinomio cuyas raíces son las componentes del vector v
expand('expr')	Expande lo más posible una expresión algebraica, realizando totalmente los productos y potencias, hasta presentar el resultado como una suma de términos. Aplica reglas de ángulos múltiples para expresiones trigonométricasy aplica formalmente las propiedades de las funciones logarítmicas y exponenciales. También descompone fracciones algebraicas de numerador polinómico en sumas de fracciones
factor('expr')	Escribe una expresión algebraica expandida como producto de factores (inversa de la anterior).
simplify('expr')	Simplifica lo más posible una expresión algebraica
simple('expr')	Halla la forma más simplificada posible de la expresión algebraica
collect('expr', 'x')	Agrupa la expresión algebraica polinómica en potencias ordenadas de la variable x. si no se especifica la variable, toma por defecto la variable simbólica principal (definida por el comando symvar)
conv(a,b)	Da el vector con los coeficientes del polinomio producto de los polinomios cuyos coeficientes son los elementos de los vectores a y b
[q,r]=deconv(a,b)	Da el vector q con los coeficientes del polinomio cociente de los polinomios cuyos coeficientes son los elementos de los vectores a y b, y el vector r, que es polinomio resto de la división
sym2poly(polinom)	Escribe el vector de coeficientes del polinomio especificado (operación inversa a la anterior)
polyder(a)	Da el vector cuyos coeficientes son los del polinomio primera derivada del polinomio a
[r,p,k]=residue(a,b)	Da los vectores columna r, p y k tales que $b(s)/a(s)=r_1/(s-p_1)+r_2/(s-p_2)+...+r_n/(s-p_n)+k(s)$
[a,b]=residue(r,p,k)	Realiza la operación inversa de la anterior.

2.3 Funciones simbólicas

subs(f,x)	Evalúa la función simbólica f en el punto x
diff(f,n)	Calcula la derivada n-sima de la función simbólica f

2.4 Representacióngráfica

ezplot(f,[a,b])	Representa la función simbólica *f* en el intervalo [a,b]
subplot(m,n,p)	Cuadricula la ventana con m filas y n columnas, y pone el siguiente dibujo en la posición p-sima (se cuenta de izquierda a derecha y de arriba abajo)