Tema 2: Métodos numéricos con MATLAB

Métodos numéricos para resolver sistemas de ecuaciones, integración de funciones y búsqueda de raíces.

2.1 Búsqueda de raíces con la sentencia 'fzero'

- Matlab tiene una sentencia 'fzero' que busca la raíz de una función f(x) no lineal.
- Su sintaxis más sencilla es:

fzero('f(x)', x0) donde f(x) es la función explicita de x y x0 es la primera estimación de la raiz.

ejemplo: >>fzero('cos(x)-x',0)

ans 0.7391

Busca un intervalo en el que la función cambie de signo y de este encuentra el valor de x que hace la función cero.

No permite encontrar múltiple raíces.

•Se debe comprobar si ha encontrado una raíz verdadera

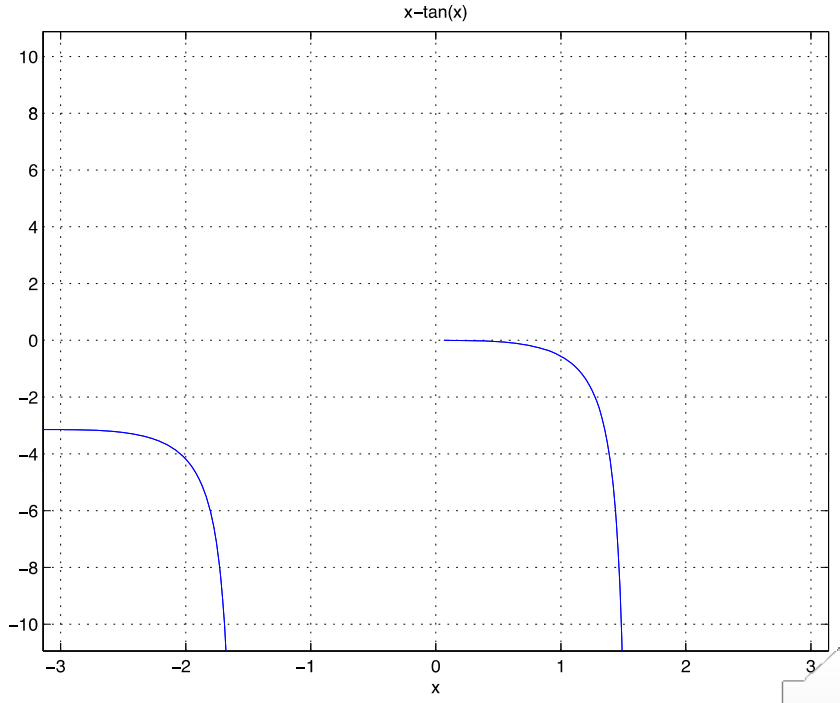
• EJEMPLO:

>>raiz=fzero('x-tan(x)',1)

>>raiz-tan(raiz)

Ans=1.2093e15

>>ezplot('x-tan(x)',[-pi pi]), grid



2.1 Búsqueda de raíces con la sentencia 'fzero'

•Si el argumento es un vector de dos elementos, se toma como el intervalo de búsqueda:

```
>>fzero('x-tan(x)',[-1 1])
```

- La tolerancia (determinar la exactitud de la raiz) y la salida de los resultados se controlan con un tercer argumento, la estructura optimset:

```
>>raiz=fzero('x-tan(x)',1,optimset('Display','iter','TolX',1e-7))
```

Muestra los cálculos realizados en cada iteración

Determina la tolerancia de la solución

- Una forma abreviada para determinar la tolerancia:

```
>>raiz=fzero('x-tan(x)',[-1 1],1e-7)
```

- 'fzero' también opera con funciones:

```
>>fzero('cos',0.5)
```

SOLO SE PONE EL NOMBRE DE LA FUNCION

2.1 Búsqueda de raíces con la sentencia 'fzero'

- Cuando la función tiene varios argumentos:

Además de la variable x, el resto de argumentos se introducen después del optimset, separados por comas.

>>fzero('jacobi',0.5,optimset('TolX',1e-12),α,β,p)

•Recordatorio: polinomio de Jacobi de grado p para el vector $[x_1 \cdots x_n]$ dados los coeficientes α y β

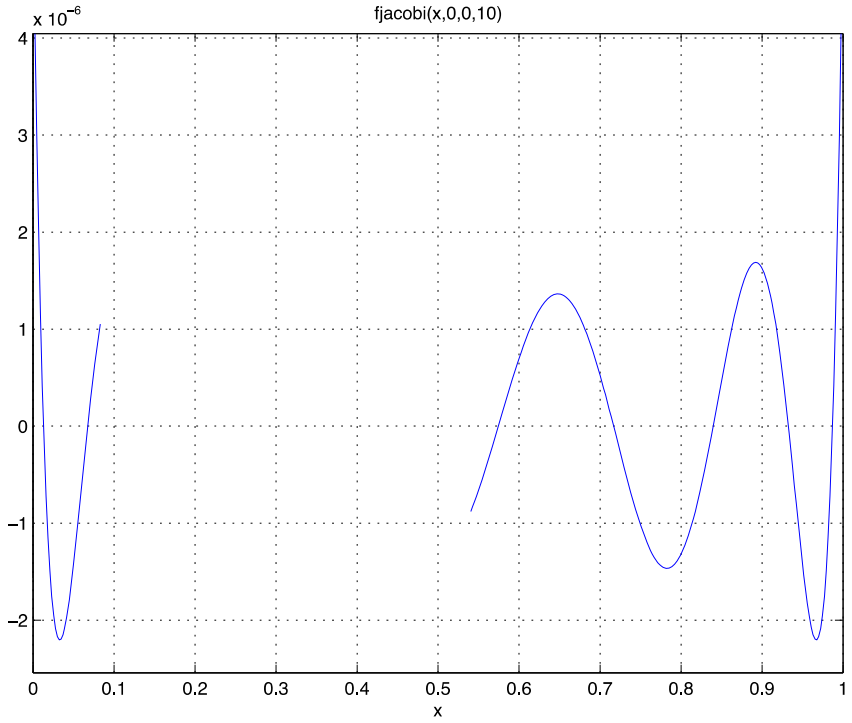
$$J_p(x,\alpha,\beta,p)$$

La variable en la que se busca la raíz debe ser el primer argumento de la función.

•‘fzero’ no encuentra múltiples raíces. Vamos a construir un programa que sí lo hace, usando ‘jacobi’ como ejemplo

>>ezplot(‘fjacobi(x,α,β,10)’,[0 1]),grid

El polinomio de Jacobi de grado p tiene p raíces reales comprendidas en el intervalo [0 1]. Representamos la función Jacobi en este intervalo.



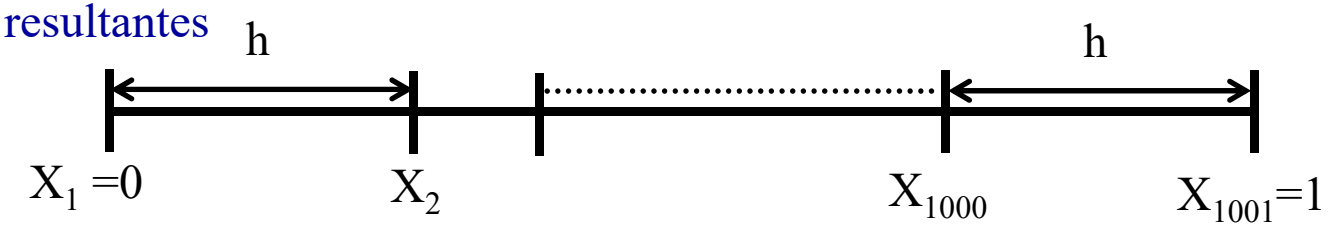
2.1 Búsqueda de raíces con la sentencia 'fzero'

raicesjacobi.m Para hacerlo general consideramos como argumentos de entrada alfa, beta y p.

1º) Primera línea de la función:

function raices=raicesjacobi(x,alfa,beta,p)

2º) División del intervalo en 1000 incrementos y evaluar la función en los puntos resultantes



intervalo=[0 1]; vector que va de 0 a 1.
N=1000;
h=(intervalo(2)-intervalo(1))/N;
x=linspace(intervalo(1),intervalo(2),N+1); linspace (0,1,N+1)
y=jacobi(x,alfa,beta,p);

INFORMATICA APLICADA A LA INGENIERÍA QUIMICA: 2ª parte MATLAB

1.Función llama a función

```
function raices=raicesejemplo;
N=1000;
intervalo=[-10 10];
h=(intervalo(2)-intervalo(1))/N;
x=linspace(intervalo(1),intervalo(2),N+1);
y=ejemplo(x);

contador=0;
raices=ones(1,1);
for k=1:N
    if y(k)*y(k+1)<0
        contador=contador+1;
        raices(contador)=fzero('ejemplo',[x(k)
x(k+1)],optimset('TolX',1e-12));
    elseif y(k)==0
        contador=contador+1;
        raices(contador)=x(k);
    end
end

if y(N+1)==0
    contador=contador+1;
    raices(contador)=x(N+1);
end
raices
plot(x,y),grid
```

```
function y=ejemplo(x)
y=(x-2).*(x-4);
```

2. Programa-función-función.

Sea la ecuación

$$y = Ax^3 + Bx^2 + Cx + D$$

Donde A, B, C y D son números enteros.

•Crear una función en Matlab que contenga la ecuación anterior.

•Crear una función en Matlab que llame a la función anterior y calcule las raíces de la ecuación en el intervalo [0 5].

•Crear un programa principal que ejecute la función anterior para los siguientes valores:

A=1 B=-6 C=11 D=-6

```
function y=ExSept1314a(x,A,B,C,D)

y=A.*x.^3 + B.*x.^2 + C.*x + D;
```

```
function calculoraices=ExSept1314b(A,B,C,D)

N=1000;
intervalo=[0 5];
h=(intervalo(2)-intervalo(1))/N;
x=linspace(intervalo(1),intervalo(2),N+1);
y=ExSept1314a(x,A,B,C,D);

contador=0;
ceros=ones(1,1);
for k=1:N
    if y(k)*y(k+1)<0
        contador=contador+1;
        ceros(contador)=fzero('ExSept1314a',[x(k) x(k+1)],optimset('TolX',1e-
12),A,B,C,D);
    elseif y(k)==0
        contador=contador+1;
        ceros(contador)=x(k);
    end
end

if y(N+1)==0
    contador=contador+1;
    ceros(contador)=x(N+1);
end
ceros
plot(x,y),grid
```

```
A=1;
B=-6;
C=11;
D=-6;

ExSept1314b(A,B,C,D)
```

Nombre: Exsept1314b

```
function calculoraices=ExSept1314b(A,B,C,D)

N=1000;
intervalo=[0 5];
h=(intervalo(2)-intervalo(1))/N;
x=linspace(intervalo(1),intervalo(2),N+1);
y=ExSept1314a(x,A,B,C,D);

contador=0;
ceros=ones(1,1);
for k=1:N
    if y(k)*y(k+1)<0
        contador=contador+1;
        ceros(contador)=fzero('ExSept1314a',
[x(k) x(k+1)],optimset('TolX',1e-12),A,B,C,D);
    elseif y(k)==0
        contador=contador+1;
        ceros(contador)=x(k);
    end
end

if y(N+1)==0
    contador=contador+1;
    ceros(contador)=x(N+1);
end
ceros
plot(x,y),grid
```

Nombre: Exsept1314a

```
function y=ExSept1314a(x,A,B,C,D)

y=A.*x.^3 + B.*x.^2 + C.*x + D;
```

Nombre: Exsept1314c

```
A=1;
B=-6;
C=11;
D=-6;

ExSept1314b(A,B,C,D)
```

ESQUEMA FUNCIONAMIENTO PROGRAMA -FUNCIONES

PROGRAMA

Aporta datos y llama a función 1. Después de todos los datos aparece el nombre del archivo función 1 al que llama:
y=buscoraices(x)

Nombre de la función 1: buscoraices.m

function y=buscoraices(x)
Hace su cometido sobre la funcion 2, y para ello llama a función2
y=ecuacion(x)
y aplica sobre función2 su busqueda

Nombre de la función 2: ecuacion.m

function y=ecuacion(x)
Aporta la función matemática
Ejemplo: y=x+2

OTRAS BUSQUEDAS DE RAICES:

Función "**roots**": Se aplica para el cálculo de raíces de un polinomio.

Ejemplo: x^2+x-6 Creo un vector poli con los coeficientes del polinomio y le doy un nombre.

```
poli=[1 1 -6]

poli =

     1     1    -6
```

Aplico roots al vector poli y al resultado lo llamo r. r va a ser un vector columna con las raíces del polinomio.

```
>> r=roots(poli)

r =

    -3
     2
```