**Máster en Materiales y Sistemas Sensores para Tecnologías Medioambientales (Erasmus Mundus)**

# PRÁCTICAS DE CÁLCULO NUMÉRICO

Damián Ginestar Peiró

**ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DEL DISEÑO**

**UNIVERSIDAD POLITÉCNICA DE VALENCIA**

# Práctica 1

# Resolución de sistemas de ecuaciones lineales

Como se ha visto en la teoría, hay dos tipos de métodos para obtener una solución numérica de un sistema de ecuaciones, los métodos directos y los métodos iterativos. Los métodos directos son, generalmente, variantes del método de Gauss y para utilizarlos es necesario conocer explícitamente los elementos de la matriz. Los métodos iterativos obtienen una aproximación numérica tras un número finito de iteraciones. Para utilizar estos métodos se suficiente conocer un método para realizar el producto de la matriz de coeficientes por un vector.

## 1.1. Métodos directos

Los métodos directos para la resolución de ecuaciones están implementados en el núcleo de Matlab, de forma que resulten lo más eficientes posibles para clases generales de matrices.

La manera más usual de resolver un sistema de ecuaciones en Matlab es haciendo uso del operador . Así, por ejemplo para resolver el sistema

$$\begin{aligned} x_1 + x_2 + 3x_4 &= 0 \\ 2x_1 + x_2 - x_3 + x_4 &= 1 \\ 3x_1 - x_2 - x_3 + 2x_4 &= -3 \\ -x_1 + 2x_2 + 3x_3 - x_4 &= 4 \end{aligned}$$

se introducen las instrucciones

```
A = [ 1, 1, 0, 3 ;
      2, 2, -1, 1;
      3, -1, -1, 2;
      -1, 2, 3,-1];
b=[0;1;-3;4];
x=A\b
```

obteniendo como resultado $x = (-1, 2, 0, 1)^T$.

El operador \ tiene implementados distintos algoritmos de forma que

- Si $A$ es una matriz triangular usa una sustitución regresiva o progresiva para resolver el sistema.
- Si $A$ es simétrica trata de calcular una descomposición de Cholesky de $A$.
- Si $A$ no es simétrica o la descomposición de Choleski falla, se usa un algoritmo de Gauss con pivotación parcial.

De una forma similar, podemos usar el operador \ para resolver simultáneamente varios sistemas de ecuaciones con la misma matriz de coeficientes. Así, por ejemplo, podemos escribir

```
A=pascal(3);
B=magic(3);
x=A\B
```

obtenemos,

```
x=
     19   -3   -1
    -17    4   13
      6    0   -6
```

que es el resultado $x = A^{-1}B$, que se obtiene sin necesidad de calcular $A^{-1}$.

## 1.2. Descomposición LU y factorización de Cholesky

El Matlab permite obtener la descomposición LU de una matriz $A$ mediante la función `lu( )`. Así podemos escribir

```
A=magic(3);
[L,U]=lu(A)
```

obteniendo como resultados

```
L=
    1.0000         0         0
    0.3750    0.5441    1.0000
    0.5000    1.0000         0

U=
    8.0000    1.0000    6.0000
         0    8.5000   -1.0000
         0         0    5.2941
```

Por otra parte, ya hemos visto que una matriz simétrica y definida positiva admite una factorización de Cholesky, o sea, una factorización

$$A = LL^T = R^T R \ .$$

Matlab dispone de una función denominada `chol( )` que realiza esta descomposición si es posible. Así, si escribimos

```
A=pascal(6);
R=chol(A)
```

obtenemos el resultado

```
R=
    1    1    1    1    1    1
    0    1    2    3    4    5
    0    0    1    3    6   10
    0    0    0    1    4   10
    0    0    0    0    1    5
    0    0    0    0    0    1
```

## 1.3. Métodos iterativos

Una posible implementación para Matlab del método de Jacobi viene dada en la siguiente función

```
function [x]=jacob(A,b)
% esta rutina implementa el metodo Jacobi
% basico.
%

tol=1.e-4;
itmax=1000;

[n,n]=size(matriz);
xo=zeros(n,1);
it=0;
error=1000.0;

while it<=itmax & error >tol
 it=it+1;
 D=diag(A);
 D1=inv(D);
 x=D1*(D-A)*xo+D1*b;
 verr=x'-xo;
 error=norm(verr)/norm(x);
 xo=x';
end
   x=xo;
   disp('el numero de iteraciones es')
   disp(it)
   disp('el error es')
   disp(error)
```

Por otro lado, el método de Gauss-Seidel se puede implementar como en la siguiente función

```
function [x]=gaussseidel(matriz,vector)
% esta rutina implementa el metodo de Gauss-Seidel
% basico.
%
tol=1.e-4;
itmax=1000;
[n,n]=size(matriz);
```

```
xo=zeros(n,1);
it=0;
error=1000.0;

% calculo de las matrices
D=diag(diag(matriz))
E=-(tril(matriz)-D)
F=-(triu(matriz)-D)

while it<=itmax & error >tol
   it=it+1
   x=(D-E)\(F*x0+vector);
   error=norm(x-x0)/ norm(x)
   x0=x;
end
   disp('el numero de iteraciones es')
   disp(it)
   disp('el error es')
   disp(error)
```

Por último una función que implementa el método SOR para un $\omega$ dado es la siguiente

```
 function [x]=sor1(matriz,vector,w);
[n,n]=size(matriz);
tol=1.e-4;
itmax=300;
x0=zeros(n,1);
it=0;
error=1000.0;
% calculo de las matrices
D=diag(diag(matriz))
E=-(tril(matriz)-D)
F=-(triu(matriz)-D)

while it<=itmax & error >tol
   it=it+1
   x=(D-w*E)\(w*F*x0+(1-w)*D*x0+w*vector);
   error=norm(x-x0)/ norm(x)
```

```
   x0=x;
end

   disp('el numero de iteraciones es')
   disp(it)

   disp('el error es')
   disp(error)
```

El Matlab dispone de funciones implementadas para la resolución de sistemas de ecuaciones lineales mediante métodos iterativos como `bicg( )`, `bicgstab` , `cgs( )`, `gmres( )` , `lsqr( )` , `pcg( )` , etc., pero se basan en métodos más complejos que los que hemos estudiado en la teoría y sobrepasan las pretensiones del curso.

## 1.4. Ejercicios

1. Una viga horizontal flexible empotrada en un extremo $A$ y libre en el extremo $B$ se considera que tiene cuatro grados de libertad traslacional $u_1, \ldots, u_4$, donde $u_i$ se localiza a $i/5$ de la distancia de $A$ a $B$. Si se aplica una carga unitaria en $u_3$, el vector $u = (u_1, u_2, u_3, u_4)^T$, satisface el sistema

   $$Ku = r \ ,$$

   donde la matriz de rigidez $K$ y el vector de cargas $r$, vienen dadas por

   $$\begin{pmatrix} 5 & -4 & 1 & 0 \\ -4 & 6 & -4 & 1 \\ 1 & -4 & 6 & -4 \\ 0 & 1 & -4 & 5 \end{pmatrix} \ , \quad r = \begin{pmatrix} 0 \\ 0 \\ EI \\ 0 \end{pmatrix} \ .$$

   la constante $EI$ depende del material de la viga y de su geometría. Calcular $u$ cuando $EI = 1$.

2. La distribución de temperatura en el estado estacionario de una placa plana se puede aproximar en 9 puntos internos aplicando la ecuación de Laplace discretizada en cada punto. Si se mantienen los lados de una placa cuadrada a temperaturas

constantes de 0° C y 100° C, las temperaturas en los 9 puntos se pueden obtener como la solución del siguiente sistema

$$\begin{aligned} -4T_1 + T_2 + T_4 &= -100 \\ T_1 - 4T_2 + T_3 + T_5 &= -100 \\ T_2 - 4T_3 + T_6 &= -200 \\ T_1 - 4T_4 + T_5 + T_7 &= 0 \\ T_2 + T_4 - 4T_5 + T_6 + T_8 &= 0 \\ T_3 + T_5 - 4T_6 + T_9 &= -100 \\ T_4 - 4T_7 + T_8 &= 0 \\ T_5 + T_7 - 4T_8 + T_9 &= 0 \\ T_6 + T_8 - 4T_9 &= -100 \end{aligned}$$

Obtener la temperatura en los distintos puntos de la placa.

3. Dado el sistema

$$\begin{aligned} x_1 + x_2 &= 4 \\ 2x_1 + 2x_3 &= 4 \\ 3x_2 + 3x_3 &= 4 \end{aligned}$$

a) Resuelve el sistema mediante el método de Gauss sin pivotación. b) Resuelve el sistema mediante el método de Gauss con pivotación parcial.

4. Escribid una función de Matlab que implemente el algoritmo de Thomas para una matriz tridiagonal simétrica. Utilízala para obtener la solución del sistema

$$\begin{pmatrix} 4 & -1 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & -1 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} .$$

5. Recuerda que una matriz simétrica y definida positiva admite factorización de Cholesky. Modifica la función `trisuper( )` para construir una nueva función que permita resolver un sistema de ecuaciones asociado a una matriz triangular inferior. Usar esta nueva función, la función `trisuper( )` y la función `chol( )`, para resolver el

siguiente sistema

$$\begin{pmatrix} 4 & -1 & 1 \\ -1 & 4{,}25 & 2{,}75 \\ 1 & 2{,}75 & 3{,}5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} -3 \\ 15{,}75 \\ 17 \end{pmatrix}$$

6. Construye el sistema de ecuaciones que han de cumplir los coeficientes de un polinomio de tercer grado

$$y = c_3x^3 + c_2x^2 + c_1x + c_0 \ ,$$

para que pase por los puntos $(12, 8)$, $(3, 27)$, $(6, 64)$, $(5, 125)$. Utiliza la función `cond( )` para obtener el número de condición de la matriz del sistema. Obtén la solución del sistema y dibuja los puntos y el polinomio obtenido.

7. Construye una función de Matlab que implemente el método SSOR. Dado el sistema

$$\begin{aligned} 4x_1 + 3x_2 &= 24 \\ 3x_1 + 4x_2 - x_3 &= 30 \\ -x_2 + 4x_3 &= -24 \end{aligned}$$

compara tres iteraciones del método SOR y 3 iteraciones del método SSOR tomando $x_0 = (1, 1, 1)^T$ y $\omega = 1{,}25$. Compara estos resultados refiriéndolos a la solución obtenida con el operador .