Computación de Altas Prestaciones

Sesión 12

1

CAP-MUIinf

Contenido

1. Matrices definidas positivas.
2. Matrices simétricas.
3. La descomposición de Cholesky.
4. Matrices dispersas.
5. Estructuras de datos de matrices dispersas.
6. La descomposición LU dispersa.
7. La descomposición de Cholesky dispersa.
8. Resolución de sistemas de ecuaciones lineales en Matlab.
9. Algoritmos de reordenación de matrices dispersas en Matlab.
10. Librerías numéricas.
11. Bibliografía.

2

CAP-MUIinf

1. Matrices definidas positivas

- Una matriz $A \in \Re^{nxn}$ es definida positiva si, para todo $x \in \Re^n$, se cumple que $x^TAx>0$.
- Aplicaciones en problemas de:
  - Mínimos cuadrados: Las matrices $A \cdot A^T$ son siempre simétricas y definidas positivas.
  - Optimización no lineal.
  - Cálculo estructural.
  - Simulación de redes de distribución de agua.
  - Etc.

DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

3

CAP-MUIinf

2. Matrices simétricas

- Si A es una matriz simétrica, el sistema de ecuaciones $Ax=b$ podemos resolverlo empleando la descomposición LU.
- Sin embargo, es posible aprovechar la simetría para reducir el número de operaciones.
- Algunos de los métodos que aprovechan dicha simetría calculan una descomposición:
  - $A=LDL^T$ donde L es triangular inferior unidad y D es diagonal.
  - $A=LTL^T$, donde L es triangular inferior unidad y T es tridiagonal.

DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

4

3. La descomposición de Cholesky

- Dada una matriz A simétrica y definida positiva, podemos calcular su descomposición de Cholesky $A=G\cdot G^T$:

$$\begin{pmatrix} a_{1,1} & a_{2,1} & \cdots & a_{n,1} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{pmatrix} = \begin{pmatrix} g_{1,1} & & & \\ g_{2,1} & g_{2,2} & & \\ \vdots & \vdots & \ddots & \\ g_{n,1} & g_{n,2} & \cdots & g_{n,n} \end{pmatrix} \begin{pmatrix} g_{1,1} & g_{2,1} & \cdots & g_{n,1} \\ & g_{2,2} & \cdots & g_{n,2} \\ & & \ddots & \vdots \\ & & \cdots & g_{n,n} \end{pmatrix}$$

- El coste de la descomposición de Cholesky es $o(n^3/3)$, frente a la descomposición LU que es $o(2n^3/3)$.
- Para obtener los elementos de G, igualaremos los elementos de A como el producto de la fila de G por la columna de G^T.
- Utilizaremos sólo los elementos de A de la parte triangular inferior ($a_{i,j}$ con $i \geq j$).

5

CAP-MUIinf

3. La descomposición de Cholesky

- Ejercicio: Obtener la descomposición de Cholesky de la siguiente matriz A:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 8 & 4 \\ 3 & 4 & 11 \end{pmatrix}$$

- Aplicaremos las fórmulas columna a columna (empezando por la primera) y desde el elemento de la diagonal hacia abajo.

6

CAP-MUIinf

3. La descomposición de Cholesky

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 8 & 4 \\ 3 & 4 & 11 \end{pmatrix} = \begin{pmatrix} g_{11} & 0 & 0 \\ g_{21} & g_{22} & 0 \\ g_{31} & g_{32} & g_{33} \end{pmatrix} \begin{pmatrix} g_{11} & g_{21} & g_{31} \\ 0 & g_{22} & g_{32} \\ 0 & 0 & g_{33} \end{pmatrix}$$

Columna 1 $\longrightarrow \begin{cases} 1 = g_{11} \cdot g_{11} \Rightarrow 1 = g_{11}^2 \Rightarrow g_{11} = \sqrt{1} = 1 \\ 2 = g_{21} \cdot g_{11} \Rightarrow 2 = g_{21} \cdot 1 \Rightarrow g_{21} = 2 \\ 3 = g_{31} \cdot g_{11} \Rightarrow 3 = g_{31} \cdot 1 \Rightarrow g_{31} = 3 \end{cases}$

Columna 2 $\longrightarrow \begin{cases} 8 = g_{21} \cdot g_{21} + g_{22} \cdot g_{22} \Rightarrow 8 = 2 \cdot 2 + g_{22}^2 \Rightarrow g_{22} = \sqrt{8-4} = 2 \\ 4 = g_{31} \cdot g_{21} + g_{32} \cdot g_{22} \Rightarrow 4 = 3 \cdot 2 + g_{32} \cdot 2 \Rightarrow g_{32} = \dfrac{4-6}{2} = -1 \end{cases}$

Columna 3 $\longrightarrow \begin{cases} 11 = g_{31} \cdot g_{31} + g_{32} \cdot g_{32} + g_{33} \cdot g_{33} \Rightarrow 11 = 3 \cdot 3 + (-1) \cdot (-1) + g_{33}^2 \Rightarrow \\ \Rightarrow g_{33} = \sqrt{11-9-1} = 1 \end{cases}$

7

CAP-MUIinf

3. La descomposición de Cholesky

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 8 & 4 \\ 3 & 4 & 11 \end{pmatrix} = \begin{pmatrix} g_{11} & 0 & 0 \\ g_{21} & g_{22} & 0 \\ g_{31} & g_{32} & g_{33} \end{pmatrix} \begin{pmatrix} g_{11} & g_{21} & g_{31} \\ 0 & g_{22} & g_{32} \\ 0 & 0 & g_{33} \end{pmatrix}$$

$$\Downarrow$$

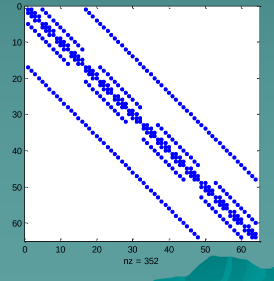
$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 8 & 4 \\ 3 & 4 & 11 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 2 & 0 \\ 3 & -1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 \\ 0 & 2 & -1 \\ 0 & 0 & 1 \end{pmatrix}$$

8

CAP-MUIinf

4. Matrices dispersas

- Son matrices con un alto porcentaje de elementos iguales a 0. Aparecen en cálculo de estructuras, dinámica de fluidos, transferencia de calor, telecomunicaciones, etc.
- Matlab dispone de un soporte excelente para trabajar con matrices dispersas.



DSIC DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

9

CAP-MUIinf

4. Matrices dispersas

- Operaciones con matrices dispersas: producto matriz por vector, matriz por matriz, factorización LU, etc.
- Producto matriz por vector: no altera la estructura (no aparecen nuevos ceros) y se optimiza fácilmente.
- Matriz por matriz: La estructura de la matriz resultado puede ser dispersa o no. Ejemplos:
  - El producto de matrices banda es banda.
  - El producto de triangulares es triangular.
- Factorización LU: Normalmente, la estructura de L y U es diferente a la de la matriz original. Eventualmente, L y U pueden llegar a ser densas.

DSIC DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

10

CAP-MUIinf

4. Matrices dispersas

- En muchos casos, los problemas dispersos son de gran dimensión.
- Hay muchos problemas que son tratables en tiempo razonable sólo si el problema no es denso.
- Si pretendemos resolver un sistema de ecuaciones de gran dimensión y calculamos la descomposición LU (o la descomposición de Cholesky) de la matriz de coeficientes y sus factores L y U (o G y G^T) se convierten en densos, posiblemente no se podrá resolver por falta de memoria ➔ Se debe evitar.

11

CAP-MUIinf

5. Estructuras de datos de matrices dispersas

- Existen muchos esquemas de almacenamiento (a veces diseñados *ad hoc* para un tipo concreto de matriz).
- Consideremos esta matriz de ejemplo:

$$A = \begin{pmatrix} 1 & 0 & 0 & -1 & 0 \\ 2 & 0 & -2 & 0 & 3 \\ 0 & -3 & 0 & 0 & 0 \\ 0 & 4 & 0 & -4 & 0 \\ 5 & 0 & -5 & 0 & 6 \end{pmatrix}$$

- ¿Cómo podríamos almacenarla sin incluir los ceros?

12

CAP-MUIinf

5. Estructuras de datos de matrices dispersas

- Almacenamiento coordenado: basado en los "tripletes" *fila*, *columna*, *valor.*
- Ejemplo para almacenamiento coordenado por columnas:

$$A = \begin{pmatrix} 1 & 0 & 0 & -1 & 0 \\ 2 & 0 & -2 & 0 & 3 \\ 0 & -3 & 0 & 0 & 0 \\ 0 & 4 & 0 & -4 & 0 \\ 5 & 0 & -5 & 0 & 6 \end{pmatrix}$$

filas = [1, 2, 5, 3, 4, 2, 5, 1, 4, 2, 5]

columnas = [1, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5]

valores = [1, 2, 5, -3, 4, -2, -5, -1, -4, 3, 6]

- Se puede usar como método de entrada en Matlab.
- Esquema sencillo, pero difícil de manipular internamente.
- Presenta un cierto grado de redundancia en el vector columnas.

DSIC DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

13

CAP-MUIinf

5. Estructuras de datos de matrices dispersas

- Almacenamiento "Compressed Sparse Column" (CSC o columna comprimida):

$$A = \begin{pmatrix} 1 & 0 & 0 & -1 & 0 \\ 2 & 0 & -2 & 0 & 3 \\ 0 & -3 & 0 & 0 & 0 \\ 0 & 4 & 0 & -4 & 0 \\ 5 & 0 & -5 & 0 & 6 \end{pmatrix}$$

filas = [1, 2, 5, 3, 4, 2, 5, 1, 4, 2, 5]

columnas = [1, 4, 6, 8, 10, 12]

valores = [1, 2, 5, -3, 4, -2, -5, -1, -4, 3, 6]

- Los índices de fila o los valores de la columna j están entre las posiciones columnas(j) y columnas(j+1)-1.
- Existe un almacenamiento "Compressed Sparse Row" (CSR o fila comprimida) totalmente análogo.

DSIC DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

14

CAP-MUIinf

5. Estructuras de datos de matrices dispersas

- El almacenamiento de "Columna Comprimida" es el usado para tratar las matrices dispersas en Matlab. En el almacenamiento, no es estrictamente necesario que los índices de fila estén ordenados (en Matlab sí deben estar ordenados).
- Ventajas e inconvenientes del almacenamiento:
  - Existen diferentes funciones para manipular matrices dispersas desde archivos .mex.
  - Los elementos de la matriz están colocados de forma continua en memoria ➔ se recorren de forma eficiente si se recorren por columnas.
  - El acceso a una fila no es eficiente (hay que hacer búsquedas en el vector "filas"). Si hay que acceder de forma repetida a filas de una matriz dispersa en Matlab, es mejor obtener la traspuesta de la matriz y acceder a sus columnas.

DSIC DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

15

CAP-MUIinf

5. Estructuras de datos de matrices dispersas

- Los algoritmos que sólo usan (no modifican) la matriz son (en general) sencillos. Ejemplo: producto matriz por vector.
- Los que modifican la matriz son bastante más complicados.
- Al estar basado en vectores, la inserción de nuevos elementos (o la eliminación) es costosa. Si insertamos o eliminamos un elemento de la matriz, todos los demás deben moverse una posición.
- Lógicamente, se pueden almacenar matrices mediante estructuras de memoria dinámica y punteros (listas enlazadas). La inserción en esos esquemas es mucho más eficiente, pero los recorridos en memoria de los elementos de la matriz son más lentos (a través de punteros).

DSIC DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

16

5. Estructuras de datos de matrices dispersas

- Ejercicio. La siguiente función implementa el producto de una matriz por un vector. Modifica dicha función teniendo en cuenta que la matriz será dispersa y estará almacenada en formato CSC.

```
function y=prod_mat_vector_denso(A,x)
[m, n] = size(A);
y = zeros(m, 1);
for i=1:m
    for j=1:n
        y(i) = y(i) + A(i,j) * x(j);
    end
end
```

Versión filas

```
function y=prod_mat_vector_denso(A,x)
[m, n] = size(A);
y = zeros(m, 1);
for j=1:n
    for i=1:m
        y(i) = y(i) + A(i,j) * x(j);
    end
end
```

Versión columnas

17

CAP-MUIinf

5. Estructuras de datos de matrices dispersas

```
function y=prod_mat_vector_denso(A,x)
[m, n] = size(A);
y = zeros(m, 1);
for j=1:n
    for i=1:m
        y(i) = y(i) + A(i,j) * x(j);
    end
end
```

filas = [1, 2, 5, 3, 4, 2, 5, 1, 4, 2, 5]

valores = [1, 2, 5, -3, 4, -2, -5, -1, -4, 3, 6]

columnas = [1, 4, 6, 8, 10, 12]

$$A = \begin{pmatrix} 1 & 0 & 0 & -1 & 0 \\ 2 & 0 & -2 & 0 & 3 \\ 0 & -3 & 0 & 0 & 0 \\ 0 & 4 & 0 & -4 & 0 \\ 5 & 0 & -5 & 0 & 6 \end{pmatrix}$$

```
function y=prod_mat_vector_disperso(filas, columnas, valores, m, x)
n = length(x);
y = zeros(m, 1);
for j = 1:n
    for i = columnas(j) : columnas(j+1) -1
        y(filas(i)) = y(filas(i)) + valores(i) * x(j);
    end
end
```


18

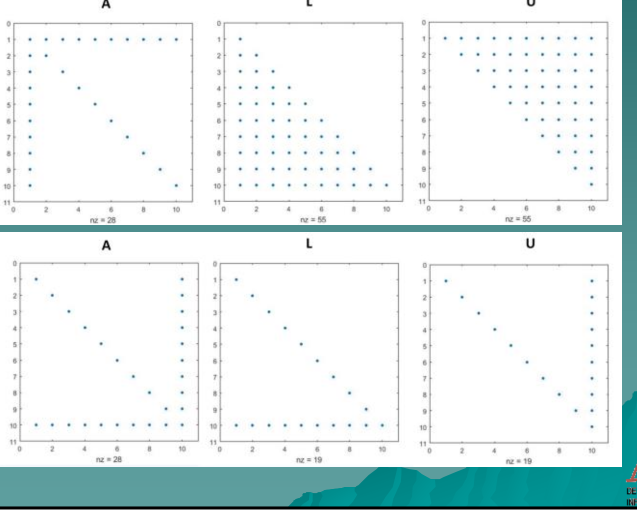
6. La descomposición LU dispersa

- Llenado: al factorizar una matriz pueden aparecer coeficientes no nulos donde inicialmente había ceros.
- Para reducir el fenómeno de llenado, se pueden utilizar técnicas de reordenación de las ecuaciones.
- No obstante, también es preciso garantizar la estabilidad mediante pivotación.
- Principal problema de la LU dispersa: Encontrar una reordenación de las ecuaciones que garantice la estabilidad (pivotación parcial) y minimice el llenado.

19

6. La descomposición LU dispersa

- Ejemplo, obteniendo la descomposición LU sin pivotación:



20

CAP-MUIinf

6. La descomposición LU dispersa

- Fases que debe tener un código que resuelva el sistema Ax=b mediante la LU para matrices dispersas no simétricas:
  1) Análisis + factorización (idealmente): Determinar una secuencia de pivotamientos que:
     a) Minimice el llenado.
     b) Garantice la estabilidad numérica.

     Es preciso usar los números concretos (para garantizar la estabilidad, se trata de coger pivotes tan grandes como sea posible). Al mismo tiempo, se genera el reordenamiento y la factorización.
  2) Resolución: Resolución de los sistemas triangulares.
- Típicamente, la fase de factorización suele ser varias veces más costosa que las otras fases.
- Si se resuelven varios sistemas de ecuaciones lineales con una misma matriz A, la fase de reordenamiento y factorización se amortiza.

DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

21

CAP-MUIinf

7. La descomposición de Cholesky dispersa

- Para una matriz simétrica y definida positiva, no es necesario pivotar para garantizar la estabilidad: el algoritmo es suficientemente estable.
- Sin embargo, el reordenamiento para minimizar el llenado debe conservar la simetría. Para ello, el mismo intercambio que se hace por filas, también se realiza por columnas:

$$PAP^T$$

- Así se garantiza que la matriz resultante también sea simétrica y definida positiva.

DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

22

CAP-MUIinf

7. La descomposición de Cholesky dispersa

- Fases que debe tener un código que resuelva el sistema Ax=b mediante la descomposición de Cholesky de matrices dispersas:
  1) Análisis: Determinar una secuencia de pivotaciones que minimice el relleno. Para esta fase, es suficiente con conocer la estructura de no-ceros de la matriz, no es necesario disponer de los valores concretos.
  2) Factorización: Calcular la factorización de Cholesky, utilizando el reordenamiento hallado en la fase 1.
  3) Resolución: Resolución de los sistemas triangulares.

23

CAP-MUIinf

8. Resolución de sistemas de ecuaciones lineales con Matlab

- Pretendemos resolver el sistema Ax=b, siendo A densa o dispersa.

- Mediante la LU:

$$Ax = b \Rightarrow LUx = b \Rightarrow \begin{cases} Ly = b \\ Ux = y \end{cases}$$

```
>> [L,U]=lu(A);
>> y=L\b;  % Ly=b
>> x=U\y;  % Ux=y
```

- Mediante Cholesky:

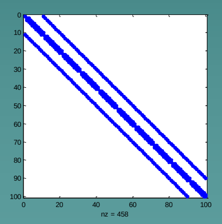
$$Ax = b \Rightarrow LL^T x = b \Rightarrow \begin{cases} Ly = b \\ L^T x = y \end{cases}$$

```
>> L=chol(A, 'lower');
>> y=L\b;  % Ly=b
>> x=L'\y;  % L^Tx=y
```

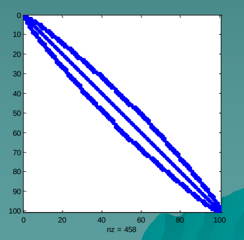
24

9. Algoritmos de reordenación de matrices dispersas en Matlab

- Existen muchos algoritmos de reordenación para reducir el llenado.
- Reordenación en Matlab:
  1) Cuthill-McKee Inverso Simétrico (symrcm). Válido sobre matrices simétricas y no simétricas.
     Ejemplo: Efecto sobre la matriz membrana:




```
A=crea_matriz(10,0.5);
← spy(A)
p=symrcm(A);
R=A(p,p);
spy(R) →
```



25

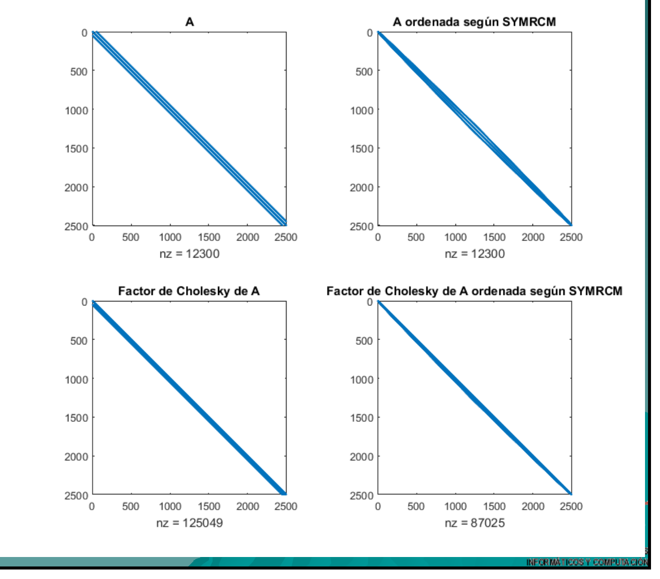
9. Algoritmos de reordenación de matrices dispersas en Matlab

```
A=crea_matriz(50,0.5);
spy(A)

p=symrcm(A);
R=A(p,p);
spy(R)

L=chol(A,'lower');
spy(L)

Lr=chol(R,'lower');
spy(Lr);
```



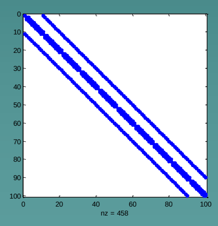
26

9. Algoritmos de reordenación de matrices dispersas en Matlab

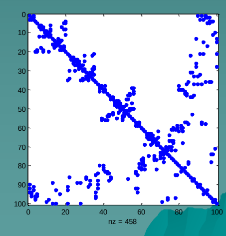
2) Mínimo Grado Aproximado (amd). Válido sobre matrices simétricas y no simétricas.

Existe el método llamado Mínimo Grado Aproximado Simétrico (symamd), válido para matrices simétricas y definidas positivas.

Ejemplo: Efecto sobre la matriz membrana:




```
A=crea_matriz(10,0.5);
🡐 spy(A)
p=amd(A);
R=A(p,p);
spy(R)   🡒
```



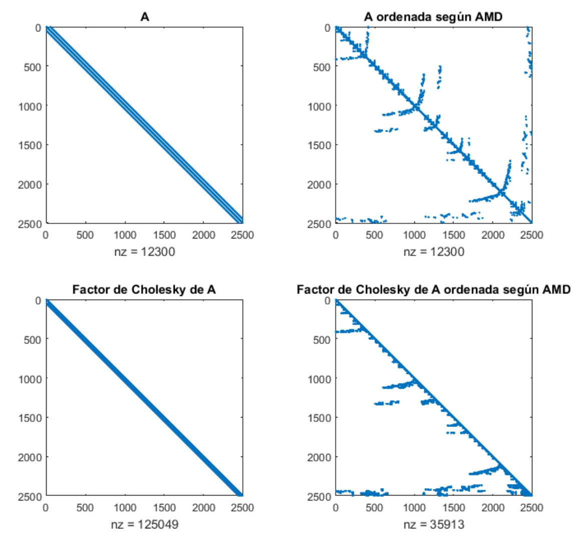
9. Algoritmos de reordenación de matrices dispersas en Matlab

```
A=crea_matriz(50,0.5);
spy(A)

p=amd(A);
R=A(p,p);
spy(R)

L=chol(A,'lower');
spy(L)

Lr=chol(R,'lower');
spy(Lr);
```



CAP-MUIinf

9. Algoritmos de reordenación de matrices dispersas en Matlab

- Pretendemos resolver el sistema Ax=b, siendo A dispersa, empleando un algoritmo de ordenación:
  - Mediante la descomposición LU:

$$Ax = b \Rightarrow PAP^T Px = Pb \Rightarrow LUPx = Pb \Rightarrow \begin{cases} Ly = Pb \\ UPx = y \Rightarrow Uz = y \\ Px = z \Rightarrow x = P^T z \end{cases}$$

  - Mediante la descomposición de Cholesky:

$$Ax = b \Rightarrow PAP^T Px = Pb \Rightarrow LL^T Px = Pb \Rightarrow \begin{cases} Ly = Pb \\ L^T Px = y \Rightarrow L^T z = y \\ Px = z \Rightarrow x = P^T z \end{cases}$$

DSIC DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

29

CAP-MUIinf

9. Algoritmos de reordenación de matrices dispersas en Matlab

- Si resolvemos Ax=b, donde A es dispersa, y usamos la ordenación de Cuthill-McKee Inverso Simétrico:

```
>>p=symrcm(A);
>>R=A(p,p);
>>[L,U]=lu(R);
>>br=b(p);
>>y=L\br;
>>xr=U\y;
>>x(p)=xr;
```

Mediante la descomposición LU

```
>>p=symrcm(A);
>>R=A(p,p);
>>L=chol(R, 'lower');
>>br=b(p);
>>y=L\br;
>>xr=L'\y;
>>x(p)=xr;
```

Mediante la descomposición de Cholesky

DSIC DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

30

CAP-MUIinf

10. Librerías numéricas

- Librerías numéricas disponibles:
  - Reordenación de matrices: Metis, Scotch, PORD.
  - Resolución de sistemas de ecuaciones lineales: UFMPACK, SuperLU, MUMPS, PaStiX, Pardiso, etc.
  - SuiteSparse: Incluye un amplio conjunto de librerías numéricas invocables desde Matlab.

31

CAP-MUIinf

11. Bibliografía

- Direct Methods for Sparse Matrices. Iain S. Duff, Albert M. Erisman, John Ker Reid. Oxford Science Publications.
- Direct Methods for Sparse Linear Systems. Timothy A. Davis. SIAM.

32