Resolución numérica de Sistemas de Ecuaciones Lineales

Práctica con Matlab

1. Resolución de sistemas lineales. Métodos directos

Dado el sistema lineal $Ax = b$ Matlab utiliza distintos métodos para su resolución dependiendo de la estructura de la matriz A .

- Si $A \in \mathcal{M}_n$ aplicará la factorización LU .
- Si $A \in \mathcal{M}_n$ es definida positiva aplicará la factorización de Cholesky.
- Si $A \notin \mathcal{M}_n$ aplicará la factorización QR.
- Si el sistema es incompatible o sobredeterminado (tiene más ecuaciones que incógnitas) busca la solución que minimiza $Ax - b = 0$ en el sentido de los mínimos cuadrados, realizando previamente una factorización QR de A.

Para resolver el sistema $Ax = b$ se utiliza el operador \ de la forma: `x=A \ b`. No es práctico el resolverlo de la forma $x = A^{-1}b$ o con notación de Matlab `x=inv(A)*b`. Se puede observar que el uso del primer operador mejora la velocidad de cálculo de MATLAB en cerca de un 50 %.
Nota: Aunque Matlab resuelve cualquier tipo se sistema de ecuaciones lineales, no indica si el sistema es compatible , incompatible, determinado o indeterminado.

1.1. Condicionamiento de un sistema

Comandos de Matlab

`norm(A,1)`	$\|\| \|\|_1$ = máx de las sumas de los valores absolutos de las columnas de A.
`norm(A,inf)`	$\|\| \|\|_\infty$ = máx de las sumas de los valores absolutos de las filas de A.
`norm(A)`	$\|\| \|\|_2$ = norma euclídea
`cond(A)`)	número de condición de A.
Nota:	Si `cond(A)`$\simeq$1 se dice bien condicionada, y si cond(A) $\uparrow\uparrow$ 1 está mal condicionada. En la práctica es más conveniente trabajar con $0 \leq$ `rcond(A)` ≤ 1 que coincide con la inversa de `cond(A)`. Cuanto más cercano está este valor a 1 mejor condicionada está A.

[**R . S. Wilson**]

Consideremos el sistema $\left\{\begin{array}{rl} 10x_1+7x_2+8x_3+7x_4= & 32\\ 7x_1+5x_2+6x_3+5x_4= & 23\\ 8x_1+6x_2+10x_3+9x_4= & 33\\ 7x_1+5x_2+9x_3+10x_4= & 31\end{array}\right.$

1. Compara la solución exacta del sistema x con la solución $\widetilde{x}$ obtenida al tomar como vector de términos independientes $\widetilde{b}=(32'1,22'9,33'1,30'9)^t$. Obtén los errores relativos de las soluciones y de los datos.
2. Repite el apartado anterior si tomamos $\widetilde{A}=\begin{pmatrix} 10 & 7 & 8{,}1 & 7{,}2\\ 7{,}08 & 5{,}04 & 6 & 5\\ 8 & 5{,}98 & 9{,}89 & 9\\ 6{,}99 & 4{,}99 & 9 & 9{,}98\end{pmatrix}$ como matriz de coeficientes.
3. Calcula el numero de condición de A aplicando su definición y con el comando de Matlab .

1.2. Matrices elementales

Comandos de Matlab relativos a vectores y matrices

$\overrightarrow{v}=(v_1,\ldots,v_n)$	$[v_1,v_2,\cdots,v_n]$
$x=x_0:\triangle x:x_n$	$\triangle x$ indica el incremento a salto entre dos valores consecutivos
linspace(x_0,x_n,n)	n número de valores, equidistribuidos, entre x_0 y x_n incluidos
$A\in\mathcal{M}_{m\times n}$	$A=\left[\underbrace{a_{11}\ a_{12}\ldots a_{1n}}_{\text{primera fila}};\underbrace{a_{21}\ a_{22}\ldots a_{2n}}_{\text{segunda fila}};\cdots;\underbrace{a_{m1}\ a_{m2}\ldots a_{mn}}_{\text{última fila}}\right]$
Elementos de A	$a_{ij}\equiv A(i,j)$, $\underset{\text{fila i}}{A(i,:)}$, $\underset{\text{columa i}}{A(:,i)}$, $\underset{\text{submatriz}}{A([i\ j],[k\ l])}$
`inv(A)`	inversa de la matriz A
`[m n]=size(A)`	m = nº filas de A y n=nº columnas de A
`A'`	traspuesta de A
`zeros(n,m))`	matriz nula de orden nxm
`ones(A)`	matriz de unos de orden nxm
`eye(n)`	matiz identidad de orden n
`eig(A)`	valores propios de A
`diag(A)`	diagonal principal de A
`diag(v)`	Matriz diagonal cuya diagonal es v
Nota:	Anteponiendo un · a las operaciones * , potencia y / éstas se realizan elemento a elemento.

Práctica 2	**Matrices elementales**

Construye tres funciones, pij(n,i,j), pijt(n,i,j,t) y qit(n,i,t) que calculen cada una de las matrices elementales que hemos definido P_{ij}, $P_{ij}(t)$ y $Q_i(t)$ respectivamente.

Sobre la matriz $A = \begin{pmatrix} 2 & -1 & 10 & 0 \\ 0 & 3 & -1 & 8 \\ 10 & -1 & 2 & 0 \\ -1 & 11 & -1 & 3 \end{pmatrix}$ realiza las siguientes operaciones elementales:

a) Intercambiar las filas 1ª y 2ª. *b)* Multiplica la fila 3ª por 5. *c)* Suma a la 1ª fila la 3ª multiplicada por 2.

```
function p=pij(n,i,j)
%matriz elemental de orden n que permite permutar las filas i y j.
p=eye(n);          % partimos de la identidad de orden n.
p(i,i)=0;p(j,j)=0;p(i,j)=1;p(j,i)=1;% modificamos los elementos necesarios.

function p=pijt(n,i,j,t)
% matriz elemental de orden n que suma a la fila i
% la fila j multiplicada por el escalar t
p=eye(n);  % partimos  de la identidad de orden n.
p(i,j)=t;  % modificamos los elementos necesarios.

function p=qit(n,i,t)
%matriz elemental de orden n que multiplica a la fila i por el escalar t.
p=eye(n);  % partimos  de la identidad de orden n.
p(i,i)=t;  % modificamos los elementos necesarios.
```

a) `pij(4,1,2)*A`   *b)* `qit(4,3,5)*A`   *c)* `pijt(4,1,3,2)*A`

1.3. Eliminación Gaussiana

Práctica 3	**Eliminación Gaussiana directa**

Resuelve directamente aplicando el método de Gauss $\left\{\begin{aligned} x_2 + 2x_3 + x_4 &= 12 \\ x_1 + 2x_2 + x_3 + 3x_4 &= 20 \\ x_1 + x_2 - x_3 + x_4 &= 4 \\ x_2 + 8x_3 + 12x_4 &= 74 \end{aligned}\right.$

Ficheros .m	
ReducTrian	Transforma un sistema en otro equivalente de matriz A triangular superior
`SolTri`	Resuelve un sistema cuya matriz A es triangular superior
`gaussSin`	Resuelve un sistema por eliminación Gaussiana sin pivotaje parcial
`gussCon`	Resuelve un sistema por eliminación Gaussiana con pivotaje parcial

El método de Gauss *sin pivotaje parcial* no será válido si en algún paso i el elemento a_{ii} es cero. Por eso es necesario introducir el procedimiento de *pivotaje parcial* .

Para ello en el caso de que en el i-ésimo paso el pivote sea nulo, se intercambia la fila i por una fila j tal que $a_{ji} \neq 0$ para algún $j > i$. Si no fuera posible, el sistema no es compatible determinado.
En la práctica se toma el pivote de mayor valor absoluto posible. Ello dota al sistema de una mayor estabilidad frente a los errores de redondeo.

Práctica 4 **Eliminación gaussiana con y sin pivotaje parcial**

Aplica los programas definidos anteriormente a los siguientes sistemas y explica qué ocurre en cada caso.

$$(a) \left\{ \begin{array}{rl} 2x_1 + x_2 = & -1 \\ x_1 - 2x_2 + x_3 = & -2 \\ 3x_1 - x_2 = & 1 \end{array} \right. \qquad (b) \left\{ \begin{array}{rl} x_2 + 2x_3 + x_4 = & 12 \\ x_1 + 2x_2 + x_3 + 3x_4 = & 20 \\ x_1 + x_2 - x_3 + x_4 = & 4 \\ x_2 + 8x_3 + 12x_4 = & 74 \end{array} \right.$$

1.4. Factorización LU y Factorización de Cholesky

Comandos de Matlab

[L,U]=lu(A)	Devuelve una matriz triangular superior **U** , una matriz de permutación triangular inferior **L**, con unos en la diagonal principal, tal que A = L * U.
[L,U,P]=lu(A)	Devuelve una matriz triangular superior **U** , una matriz triangular inferior **L**, con unos en la diagonal principal y una matriz de permutación **P** tal que P*A = L*U.

Nota: Con estos comandos la solución del sistema $Ax = b$ utilizando la descomposición **LU** se calcula como: `[L U]=lu(A); x=U\(L\b)`

Práctica 5 **Factorización LU**

Resuelve estos sistemas aplicando la factorización LU a la matriz de coeficientes .

$$(a) \left\{ \begin{array}{rl} x_1 + 2x_2 + x_3 + 3x_4 = & 0 \\ x_1 + x_2 + x_3 + 4x_4 = & -2 \\ 2x_1 + x_2 + 4x_3 + 10x_4 = & -7 \\ -x_1 - 3x_2 + 7x_3 + 5x_4 = & -9 \end{array} \right. \qquad (b) \left\{ \begin{array}{rl} x_1 + 2x_2 - 2x_3 + x_4 = & 2 \\ 4x_1 + 5x_2 - 7x_3 + 6x_4 = & 8 \\ 5x_1 + 25x_2 - 15x_3 - 3x_4 = & 12 \\ 6x_1 - 12x_2 - 6x_3 + 22x_4 = & 10 \end{array} \right.$$

$$(c) \left\{ \begin{array}{rl} -3x_2 + x_3 + 2x_4 + 2x_5 = & 2 \\ 2x_5 = & 2 \\ x_1 + 2x_2 - 2x_3 + x_4 + x_5 = & 3 \\ 4x_1 + 2x_2 - 8x_3 + 8x_4 + 9x_5 = & 15 \\ 5x_1 + x_2 - 5x_3 + 13x_4 + 11x_5 = & 25 \end{array} \right. \qquad (d) \left\{ \begin{array}{rl} x_1 + x_2 + x_3 + x_4 + x_5 = & 1 \\ x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 = & 2 \\ -2x_1 + x_2 - 3x_3 + x_4 = & -1 \\ x_1 - x_2 + x_3 - x_4 + x_5 = & -1 \\ x_3 + x_4 - x_5 = & 1 \end{array} \right.$$

Comandos de Matlab	
R = chol(A)	Devuelve una matriz R triangular superior, tal que $R^t * R = A$, cuando A es definida positiva (si A no lo es, produce un mensaje de error). Para aplicar cholesky A debe ser simétrica.
[R, p] = chol(A)	No da nunca mensaje de error. Si A es definida positiva $p = 0$. Si no lo es $p > 0$. R es matriz triangular superior de orden $q = p - 1$. Además R cumple que $R^t * R = A(1 : q, 1 : q)$

Práctica 6 **Factorización de Cholesky**

Resuelve estos sistemas aplicando la factorización de Cholesky a la matriz de coeficientes .

$$(a)\left\{\begin{array}{rr} 2x_1 - x_2 = & 4 \\ -x_1 + 2x_2 - x_3 = & -8 \\ -x_2 + 2x_3 = & 8 \end{array}\right. \qquad (b)\left\{\begin{array}{rr} 4x_1 + x_2 + x_3 + x_4 = & 3 \\ x_1 + 3x_2 - x_3 + x_4 = & -4 \\ x_1 - x_2 + 2x_3 = & 4 \\ x_1 + x_2 + 2x_4 = & -2 \end{array}\right.$$

2. Métodos iterativos

En los métodos iterativos se parte de una aproximación inicial, x^0 , de la solución del sistema y se genera, a partir de esta aproximación, una sucesión, $(x^k)_{k\in\mathbb{N}}$, de vectores que si converge lo hace a la solución del sistema.
Para generar esta sucesión, se repite el mismo esquema de operaciones hasta que:

- se obtiene una aproximación a la solución con una precisión especificada de antemano,
- o se sobrepasa un número máximo de iteraciones.

2.1. Método de Jacobi

Se basa en la descomposición $A = D - (D - A)$ donde $D = \begin{pmatrix} a_{11} & 0 & \dots & 0 \\ 0 & a_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_{nn} \end{pmatrix}$

Por tanto el método de Jacobi se define $\left\{\begin{array}{l} x^0 \in \mathbb{R}^n \\ x^{k+1} = D^{-1}(D - A)x^k + D^{-1}b, k \in \mathbb{N}\bigcup\{0\} \end{array}\right.$

Definición: Se llama la matriz del método de Jacobi a $J = D^{-1}(D - A) = I - D^{-1}A$

Práctica 7 **Matriz de Jacobi**

Calcula la matriz de Jacobi de $A = \begin{pmatrix} 2 & -2 & 0 \\ 2 & 3 & -1 \\ \alpha & 0 & 2 \end{pmatrix}$. Indica algún valor de α para el cuál el método de Jacobi sea convergente.

Propiedad: Si $A \in \mathcal{M}_n$ es una matriz diagonal estrictamente dominante, el método iterativo de Jacobi converge a la única solución del sistema $Ax = b$ para cualquier x^0.

Propiedad:Para cualquier $x^0 \in \mathbb{R}^n$, la sucesión $(x^k)_k$ definida por la fórmula de iteración $x^{k+1} = Gx^k + c\,,\ k = 1, 2, \ldots,\ c \neq 0$, converge a la única solución x del sistema $x = Gx + c$ si y sólo si $\rho(G) < 1$.

Nota: Es recomendable reordenar las ecuaciones del sistema de modo que la matriz de coeficientes del sistema reordenado sea lo más cercana posible a una matriz diagonal estrictamente dominante por filas (colocando primero las ecuaciones que tienen un coeficiente dominante y luego las restantes).
Vimos que existen varios criterios de parada:

1. Basado en el error absoluto en la aproximación: $\| x^{k+1} - x^k \| < \epsilon$
2. Basado en el error relativo en la aproximación: $\| x^{k+1} - x^k \| < \epsilon \| x^{k+1} \|$
3. Basado en el residuo: $\| r^k \| < \epsilon \| b \|$ donde r^k es el vector residuo en la iteración k: $r^k = b - Ax^k$

Práctica 8 **Método de Jacobi**

Resuelve los sistemas siguientes aplicando el algoritmo de Jacobi, estudiando previamente su convergencia y aplica el primer criterio de parada para $\epsilon = 10^{-4}$.

$$(a) \left\{ \begin{array}{rr} 2x_1 - x_2 + x_3 = & 2 \\ x_1 + x_2 + 3x_3 = & 5 \\ 3x_1 + 3x_2 + 5x_3 = & 11 \end{array} \right. \qquad (b) \left\{ \begin{array}{rr} 2x_1 - x_2 + 10x_3 = & 11 \\ 3x_2 - x_3 + 8x_4 = & 10 \\ 10x_1 - x_2 + 2x_3 = & 11 \\ -x_1 + 11x_2 - x_3 + 3x_4 = & 12 \end{array} \right.$$

Aplica el algoritmo de Jacobi $x^{k+1} = Jx^k + D^{-1}b$ recursivamente y completa la tabla:

k	x_1^k	x_2^k	x_3^k	$\parallel x^k - x^{k+1} \parallel$
0	0	0	0	
1	1	$3,66$	$1,66$	$4,15$
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15	1,0008	1,0003	0,9993	$6{,}7\,10^{-4}$

2.2. Método iterativo de Gauss-Seidel

Descomponemos $A = L - (L - A)$ donde $L = \begin{pmatrix} a_{11} & 0 & \dots & 0 \\ a_{21} & a_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}$

el método de Gauss-Seidel se escribe

$$\begin{cases} x^0 \in \mathbb{R}^n \\ x^{k+1} = L^{-1}(L-A)x^k + L^{-1}b = (I - L^{-1}A)x^k + L^{-1}b, k \in \mathbb{N}\bigcup\{0\} \end{cases}$$

Definición: Se llama matriz de iteración de Gauss - Seidel a $\mathcal{S} = L^{-1}(L-A) = I - L^{-1}A$

Práctica 9 **Método de Gauss-Seidel**

2.-Resuelve los sistemas siguientes aplicando el algoritmo de Gauss-Seideldiando previamente su convergencia y aplica el primer criterio de parada para $\epsilon = 10^{-4}$.

$$(a) \begin{cases} 2x_1 - x_2 + x_3 = & 2 \\ x_1 + x_2 + 3x_3 = & 5 \\ 3x_1 + 3x_2 + 5x_3 = & 11 \end{cases} \qquad (b) \begin{cases} 2x_1 - x_2 + 10x_3 = & 11 \\ 3x_2 - x_3 + 8x_4 = & 10 \\ 10x_1 - x_2 + 2x_3 = & 11 \\ -x_1 + 11x_2 - x_3 + 3x_4 = & 12 \end{cases}$$

Aplica el algoritmo de Gauss-Seidel $x^{k+1} = Sx^k + L^{-1}b$ recursivamente y completa la tabla:

k	x_1^k	x_2^k	x_3^k	x_4^k	$\parallel x^k - x^{k+1} \parallel$
0	0	0	0	0	
1	1,1	1,1	0,99	0,92	2,11
2					
3					
4					
5	1	1	1	1	$3{,}45\,10^{-4}$