UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Métodos Directos

Damián Ginestar Peiró

Departamento de Matemática Aplicada

Universidad Politécnica de Valencia

Curso 2022-2023

Índice

Introducción

Sea el sistema lineal compatible y determinado

$$Ax = b$$

con $A = (a_1|\cdots|a_n)$

Fórmula de Cramer

$$x_i = \frac{\det\left(a_1|\cdots|a_{i-1}|b|a_{i+1}|\cdots|a_n\right)}{\det(A)}$$

El coste del cálculo de determinantes por el método del desarrollo por menores es del orden de $n!$. Así, para $n = 50$ y con un ordenador con 1 gigaflop de velocidad

$$\text{coste} \approx 4.8\,10^{49} \quad \text{años}$$

Sistemas triangulares

Sea el sistema lineal

$$Ax = b$$

donde A es triangular sup.

$$\left.\begin{array}{ccccccccccc}
a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1,n-1}x_{n-1} & + & a_{1n}x_n & = & b_1 \\
 & & a_{22}x_2 & + & \cdots & + & a_{2n-1}x_{n-1} & + & a_{2n}x_n & = & b_2 \\
 & & & & \ddots & & & & & & \\
 & & & & a_{n-2,n-2}x_{n-2} & + & a_{n-2,n-1}x_{n-1} & + & a_{n-2,n}x_n & = & b_{n-2} \\
 & & & & & & a_{n-1,n-1}x_{n-1} & + & a_{n-1,n}x_n & = & b_{n-1} \\
 & & & & & & & & a_{nn}x_n & = & b_n
\end{array}\right\}$$

$$\begin{array}{ll}
i = n, & x_n = b_n/a_{nn} \\
i = n-1, & x_{n-1} = (b_{n-1} - a_{n-1,n}x_n)/a_{n-1,n-1} \\
i = n-2, & x_{n-2} = (b_{n-2} - a_{n-2,n}x_n - a_{n-2,n-1}x_{n-1})/a_{n-2,n-2} \\
 & \quad = (b_{n-2} - \sum_{j=n}^{n-1} a_{i,j}x_j)/a_{n-2,n-2}
\end{array}$$

Sistemas triangulares

Algoritmo : Sustitución regresiva

Input: A y b
Output: solución x

for i=n:1
  $x_i = b_i / a_{ii}$
  for j=n:i+1
    $b_i = b_i - a_{ij} x_j$
  end for
end for

- Requiere $\sum_{i=1}^{n}(n-i+1) = (n+1)n - \frac{n(1+n)}{2} = \Theta\left(\frac{n^2}{2}\right)$ flops

Operaciones elementales

Dada una matriz $A \in \mathbb{R}^{n\times n}$, consideralmos la fila i-ésima

$$E_i = \begin{pmatrix} a_{i1} & a_{i2} & \cdots & a_{in} \end{pmatrix} .$$

Las operaciones elementales que se se pueden definir sobre la matriz A son:

Tipo I: Intercambio de filas E_i con E_j. Denotaremos esta operación por $E_i \leftrightarrow E_j$.

Tipo II: Multiplicación de la fila E_i por cualquier constante no nula λ. Denotaremos esta operación por $\lambda E_i \to E_i$.

Tipo III: Multiplicación de la fila E_j por cualquier constante no nula λ y sumársela a la fila E_i. Denotaremos esta operación por $E_i + \lambda E_j \to E_i$.

Operaciones elementales

Definición

Se llaman matrices elementales a las que se obtienen a partir de la matriz identidad mediante la aplicación de una operación elemental.

Tipo I: Matriz de permutación es una matriz cuadrada que en cada fila y columna sólo tiene un único elemento distinto de cero que vale 1. La matriz de permutaciones que intercambia la fila i por la fila j es

$$P_{ij} = \begin{array}{c} \\ i \\ \\ j \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccccc} & & & i & & j & \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & & & & \vdots & & \vdots \\ \vdots & & 0 & & \vdots & & \vdots \\ & & \vdots & & & & \\ 0 & \cdots & \vdots & 0 & 1 & \cdots & 0 \\ \vdots & & & & \vdots & & \vdots \\ \vdots & & 0 & & \vdots & & \vdots \\ 0 & \cdots & 1 & 0 & 0 & \cdots & 0 \\ \vdots & & & & & & \vdots \\ \vdots & & 0 & & 0 & & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{pmatrix} \end{array}.$$

Operaciones elementales

- Toda matriz de permutaciones es invertible y satisface que $P^{-1} = P^T$.

Tipo II: Multiplicación de la fila $i-$ésima de la matriz I por cualquier constante no nula λ. Se denota por $E_i(\lambda)$. Esta matriz elemental permite realizar transformaciones elementales del tipo II.

Tipo III: Se obtiene a partir de la matriz I sumando un múltiplo de una fila j a la fila i. Se denota por $E_{ij}(\lambda)$.

Factorización LU

Recordemos que el esquema del método de Gauss se basa en pasar de una matriz (la matriz ampliada del sistema) a una matriz equivalente que tiene forma escalonada.

Para cada $i = 1, \ldots, n$,

Paso 1. Determinar el pivote

Paso 2. Reducir a la forma escalonada utilizando. $\lambda_{ki} = a_{ki} a_{ii}^{-1}$, $k = i, \ldots, n$, y realizando la operación $(E_k - \lambda_{ki} E_i) \to E_k$, para $k = i, \ldots, n$.

Paso 3. Resolver las incógnitas

$$x_i = \frac{b_i - \sum_{j=i+1}^{n} a_{ij} x_j}{a_{ii}}$$

Factorización LU

El proceso de triangularización se puede interpretar de forma matricial.

$$A = \begin{pmatrix} \alpha & w^T \\ v & B \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ v/\alpha & I \end{pmatrix} \begin{pmatrix} \alpha & w^T \\ 0 & C \end{pmatrix}$$

y se sigue el mismo proceso para la matriz C.

Factorización LU

Se observa que obtener la forma escalonada de la matriz A de coeficientes del sistema consiste en premultiplicar la matriz A por las matrices elementales correspondientes a cada una de las operaciones elementales realizadas.
Si partimos de una matriz

$$A_{3\times 3} = [a_{ij}^{(1)}]_{i,j=1,2,3}$$

para eliminar los elementos de la primera columna debemos premultiplicar la matriz A por la matriz elemental triangular inferior

$$M^{(1)} = \begin{pmatrix} 1 & 0 & 0 \\ m_{21} & 1 & 0 \\ m_{31} & 0 & 1 \end{pmatrix}, \quad m_{i1} = \frac{-a_{i1}^{(1)}}{a_{11}^{(1)}}, \ i = 2, 3.$$

Factorización LU

Para eliminar los de la segunda columna premultiplicamos nuevamente por una matriz

$$M^{(2)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & m_{32} & 1 \end{pmatrix}, \quad m_{32} = \frac{-a_{32}^{(2)}}{a_{22}^{(2)}}.$$

Así,

$$U = M^{(2)} M^{(1)} A$$

con U una matriz escalonada. Tendremos

$$A = M^{(1)-1} M^{(2)^{-1}} U = LU$$

Factorización LU

Algoritmo de Gauss

Input: A y b

Output: L y U escritas sobre A.

Para $k = 1,$ hasta $n-1$
  Si $a_{kk} = 0$ Entonces fin
  En otro caso
    Para $i = k+1, \ldots, n$
      $a_{ik} = a_{ik}/a_{kk}$
      Para $j = k+1, \ldots, n$
        $a_{ij} = a_{ij} - a_{ik}a_{kj}$
      end para
    end para
end para

- En el bucle j, hay $(n-(k+1)+1)$ flops
- En el bucle i, $\sum_{i=k+1}^{n}(n-k) = (n-k)^2$ flops
- En la etapa k, $\sum_{k=1}^{n-1}(n-k)^2 = \sum_{k=1}^{n-1} k^2 = \Theta(\frac{1}{3}n^3)$ flops.

Algoritmo de Doolittle

Supongamos que la matriz A admite factorización LU $A = LU$, o sea,

$$a_{ij} = \sum_{k=1}^{r} l_{ik} u_{kj} \ , \quad 1 \leq i, j \leq n, \quad r = \min(i, j)$$

Si se usa la condición $l_{kk} = 1$, $k = 1, \ldots, n$ podemos calcular

$$u_{kj} = a_{kj} - \sum_{p=1}^{k-1} l_{kp} u_{pj} \ , \ j \geq k$$

$$l_{ik} u_{kk} = a_{ik} - \sum_{p=1}^{k-1} l_{ip} u_{pk}$$

Algoritmo de Doolittle

Algoritmo de Doolittle

for $k = 1 : n$

for $j = k : n$

$$u_{kj} = a_{kj} - \sum_{p=1}^{k-1} l_{kp} u_{pj}$$

end

for $i = k + 1 : n$

$$l_{ik} = \left(a_{ik} - \sum_{p=1}^{k-1} l_{ip} u_{pk} \right) / u_{kk}$$

end

$l_{kk} = 1$

end

El algoritmo de Crout es similar al algoritmo de Doolittle pero se exige que la matriz U tiene unos en la diagonal principal.

Implementación de la LU

```
function [A] = lu_kji (A)
%versión kij (row)
[n,n]=size(A);
for k=1:n-1
   A(k+1:n,k)=A(k+l:n,k)/ A(k,k);
   for j=k+1:n
    for i=k+1:n
      A(i,j)=A(i,j)-A(i,k)*A(k,j);
    end
  end
end
```

```
function [A] = lu_jki (A)
% version jki (column)
 [n,n]=size(A);
 for j=1:n
   for k=1:j-1
    for i=k+1:n
         A(i,j)=A(i,j)-A(i,k)*A(k,j);
    end
   end
 end
for i=j+1:n
   A(i,j)=A(i,j)/A(j,j)
end
end
```

Análisis del error

Recordemos el concepto de norma matricial inducida

$$\|A\|_p = \max_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p} \ .$$

se tienen los casos particulares

- $\|A\|_1 = \max\limits_{1 \leq j \leq n} \sum\limits_{i=1}^{m} |a_{ij}|$.
- $\|A\|_\infty = \max\limits_{1 \leq i \leq m} \sum\limits_{j=1}^{n} |a_{ij}|$.
- $\|A\|_2 = \sigma_{\max}(A)$, es el valor singular más grande de A.

Análisis del error

Dado que los cálculos que se realizan con un ordenador se hacen con precisión finita, veremos cómo se ve afectada la solución del sistema a pequeñas perturbaciones.

Dado el sistema $Ax = b$, consideramos el sistema $(A + \delta A)\,\hat{x} = b + \delta b$,
Si tomamos $\delta x = \hat{x} - x$ se cumple

$$\delta x = A^{-1}\left(-\delta A\hat{x} + \delta b\right)$$

Tomando normas

$$\|\delta x\| \leq \left\|A^{-1}\right\| \left(\|\delta A\|\,\|\hat{x}\| + \|\delta b\|\right)$$

Análisis del error

O sea,

$$\frac{\|\delta x\|}{\|\hat{x}\|} \leq \left\|A^{-1}\right\| \|A\| \left(\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|A\| \, \|\hat{x}\|}\right)$$

Definición

A la cantidad

$$\kappa(A) = \|A\| \left\|A^{-1}\right\|$$

se llama Número de condición de la matriz A con la norma $\|\cdot\|$.

Así

$$\frac{\|\delta x\|}{\|\hat{x}\|} \leq \kappa(A) \left(\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|A\| \, \|\hat{x}\|}\right)$$

Análisis del error

- El número de condición depende de la norma matricial que se utilice. Así

$$\kappa_2(A) = \|A\|_2 \|A^{-1}\|_2 = \frac{\sigma_1(A)}{\sigma_n(A)}$$

  donde $\sigma_1(A)$ es el valor singular más alto de A y $\sigma_n(A)$ es el valor singular más bajo de A.

- Para matrices simétricas y definidas positivas

$$\kappa_2(A) = \frac{\lambda_{\max}}{\lambda_{\min}}$$

  donde $\lambda_{\max}$ es el autovalor más alto de A y $\lambda_{\min}$ es el autovalor más bajo de A.

Análisis del error

- Diremos que una matriz está mal condicionada si su número de condición es alto.
- Si una matriz está mal condicionada, errores en los coeficientes de la matriz o en el término independiente pueden dar lugar a errores en la solución del sistema.
- Un ejemplo de matriz mal condicionada es la matriz de Hilbert

$$H_n(i,j) = h_{ij} = \frac{1}{i+j-1}\ , \quad i,j = 1,\dots n$$

El número de condición de estas matrices crece exponencialmente con n. La solución de

$$H_n x = b$$

suele fallar para $n > 12$ usando doble precisión.

Ejemplos de número de condición

- Dada la matriz

$$A = \begin{pmatrix} 8 & -5 \\ 4 & 10 \end{pmatrix}, \quad A^{-1} = \begin{pmatrix} 0.1 & 0.05 \\ -0.04 & 0.08 \end{pmatrix}$$

tiene un número de condición $\kappa_1(A) = \|A\|_1 \|A^{-1}\|_1 = 2.1$

- Dada la matriz

$$B = \begin{pmatrix} 0.66 & 3.34 \\ 1.99 & 10.01 \end{pmatrix}, \quad B^{-1} = \begin{pmatrix} 250.25 & 83.5 \\ 49.75 & -16.5 \end{pmatrix}$$

tiene un número de condición $\kappa_1(B) = \|B\|_1 \|B^{-1}\|_1 = 4005$

Ejemplos de número de condición

- Dada la matriz diagonal de orden 100 definida por

$$a_{11} = 1, \quad a_{ii} = 0.1, \quad 2 \leq i \leq 100$$

se cumple que $\|A\|_2 = 1$, $\|A^{-1}\|_2 = 10$ con lo que $\kappa_2(A) = 10$.
Si calculamos su determinante:

$$\det(A) = 1(0.1)^{99} = 10^{-99}$$

- No siempre que las matrices tienen su determinante pequeño tienen un alto número de condición.

Pivotación

El algoritmo de Gauss puede fallar si alguno de los pivotes es nulo, y va a ser necesario intercambiar las filas de la matriz para garantizar el funcionamiento del algoritmo. A esta estrategia se le denomina pivotación.

$$\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

Pivotación

- La necesidad de la pivotación se ha introducido por la presencia de pivotes nulos.
- Si se permite el intercambio de filas, el algoritmo de Gauss nos lleva a una descomposición para la matriz A de la forma

$$PA = LU \ ,$$

  donde P es una matriz de permutación.
- En aritmética exacta éste es el único caso en el que la pivotación es esencial.
- No obstante, en aritmética de coma flotante hay casos en los que si no se usa pivotación se produce una pérdida de dígitos significativos de la solución.

Ejemplo

Consideremos el sistema

$$\begin{aligned} 0.0003x_1 + 1.566x_2 &= 1.569 \\ 0.3454x_1 - 2.436x_2 &= 1.018 \end{aligned}$$

Si utilizamos el algoritmo de Gauss con una aritmética de 4 cifras significativas, llegamos a una solución

$$x_2 = 1.001, \quad x_1 = 3.333 \ ,$$

que está lejos de la solución exacta $x_1 = 10$, $x_2 = 1$.

Si intercambiamos las filas

$$\begin{aligned} 0.3454x_1 - 2.436x_2 &= 1.018 \\ 0.0003x_1 + 1.566x_2 &= 1.569 \end{aligned}$$

se llega a una solución mucho más cercana a la exacta.

Pivotación parcial

Input: A invertible y b

Para $k = 1,$ hasta $n - 1$

Determinar $p \in \{k, k+1, \ldots, n\}$ tal que $|a_{pk}| = \max_{k \leq i \leq n} |a_{ik}|$

Intercambiar fila k con fila p

Para $i = k+1, \ldots, n$

$l_{ik} = a_{ik}/a_{kk}$

Para $j = k+1, \ldots, n$

$a_{ij} = a_{ij} - l_{ik} a_{kj}$

end para

end para

end para

- flops $\Theta(\frac{1}{3}n^3)$
- número de comparaciones $\Theta(n^2)$

Función de Matlab

```
function [x]=Gauss(A,b)
% Solución de Ax=b mediante eliminación de Gauss
     n=size(A,1); x=zeros(n,1);
   for i=1:n-1
% Transformación matriz A en n-1 etapas
     [p,maxk]=max(abs(A(i:n,i)));
         maxk=maxk+i-1;
     if i~=maxk   % pivotacion parcial
      A([i maxk],:) = A([maxk i],:);
      b([i maxk])= b([maxk i]);
     end
   j=i+1:n;
   A(j,i) = A(j,i)/A(i,i);
   A(j,j) = A(j,j)-A(j,i)*A(i,j);
   b(j)= b(j)-b(i)*A(j,i);
 end
for i=n:-1:1
% Sustitución regresiva
  x(i)=(b(i)-A(i,i+1:n)*x(i+1:n))/A(i,i);
end
```

Refinamiento iterativo

Cuando se utiliza el método de Gauss, se consigue una solución del sistema $x^* = x^{(1)}$ afectada de cierto error.
Para mejorar la solución se introduce

$$r^{(1)} = b - Ax^{(1)}$$

y se tiene en cuenta que

$$A^{-1}r^{(1)} = A^{-1}b - x^{(1)} = x - x^{(1)} = \Delta x^{(1)}$$

y se realiza el siguiente esquema:

Para $k = 1, 2, \ldots$

$r^{(k)} = b - Ax^{(k)}$

$A\Delta x^{(k)} = r^{(k)}$

$x^{(k+1)} = x^{(k)} + \Delta x^{(k)}$

end **para**

Equilibrado

Para resolver un sistema de la forma

$$Ax = b$$

se elige una matriz diagonal D y se resuleve el sistema

$$DAx = Db$$

de forma que el número de condición de DA es más pequeño que el de A.
Una posibilidad es elegir d_{ii} como el inverso de la norma-2 de la fila i-ésima.
Otra posibildad es resolver

$$D_{fil}AD_{col}\bar{x} = D_{fil}b \ , \quad x = D_{col}\bar{x}$$

Cuando $D_{col} = D_{fil}^{-1}$ se llama balanceado de la matriz

Equilibrado

Dado el sistema

$$Ax = b$$

consideramos

$$D_1 A D_2 \tilde{x} = \tilde{b}\ , \quad x = D_2 \tilde{x}, \quad \tilde{b} = D_1 b$$

Interesa encontrar un escalado que minimice el número de condición

$$\kappa_p(A) = \|A\|_p \left\|A^{-1}\right\|_p$$

Equilibrado

Teorema

$A \in \mathbb{R}^{m\times n}$, si

$$D_1 = \operatorname{diag}\left(\|A(i,:)\|_p\right)^{-1}, \quad D_2 = \operatorname{diag}\left(\|A(:,j)\|_p\right)^{-1},$$

entonces

$$\kappa_p\left(AD_2\right) \leq n^{1-1/p}\min_{D\in\mathcal{D}_n}\kappa_p(AD) \text{ si } \operatorname{rango}(A) = n$$
$$\kappa_p\left(D_1A\right) \leq m^{1/p}\min_{D\in\mathcal{D}_n}\kappa_p(DA) \text{ si } \operatorname{rango}(A) = m$$

Equilibrado

Esto nos indica que equilibrar las filas o las columnas es una estrategia casi óptima.

Si consideramos el sistema

$$\begin{pmatrix} 1 & 10^4 \\ 1 & 10^{-4} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 10^4 \\ 1 \end{pmatrix}, \quad \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0.9999 \\ 0.9999 \end{pmatrix},$$

usando el método de Gauss con pivotación parcial y una precisión de tres dígitos se obtiene la solución $(0, 1.00)$.
Si se considera

$$\begin{pmatrix} 10^{-4} & 1 \\ 1 & 10^{-4} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix},$$

se obtiene la solución $(1, 1)$.

Matrices tridiagonales

Un caso especial de sistemas de ecuaciones que aparecen frecuentemente son aquellos cuya matriz de coeficientes es tridiagonal, o sea, un sistema de ecuaciones con la siguiente estructura

$$\begin{pmatrix} b_1 & c_1 & 0 & \cdots & & \\ a_1 & b_2 & c_2 & \cdots & & \\ & & & \cdots & & \\ & & & a_{n-2} & b_{n-1} & c_{n-1} \\ & & & 0 & a_{n-1} & b_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{pmatrix} .$$

Matrices tridiagonales

Dada una matriz tridiagonal

$$A = \begin{pmatrix} b_1 & c_1 & 0 & \cdots & & \\ a_1 & b_2 & c_2 & \cdots & & \\ & & & \cdots & & \\ & & & a_{n-2} & b_{n-1} & c_{n-1} \\ & & & 0 & a_{n-1} & b_n \end{pmatrix}$$

Su factorización LU se puede expresar

$$L = \begin{pmatrix} 1 & 0 & 0 & \cdots & & \\ \beta_2 & 1 & 0 & \cdots & & \\ & & & \cdots & & \\ & & & \beta_{n-1} & 1 & 0 \\ & & & 0 & \beta_n & 1 \end{pmatrix} \quad U = \begin{pmatrix} \alpha_1 & c_1 & 0 & \cdots & & \\ 0 & \alpha_2 & 0 & \cdots & & \\ & & & \cdots & & \\ & & & & \alpha_{n-1} & c_{n-1} \\ & & & 0 & \beta_n & \alpha_n \end{pmatrix}$$

$$\alpha_1 = b_1, \quad \beta_i = \frac{a_i}{\alpha_{i-1}}, \quad \alpha_i = b_i - \beta_i c_{i-1}, \ i = 2, \ldots, n$$

Matrices tridiagonales

Algoritmo de Thomas

- La matriz se tiene almacenada en los vectores a, b, c, y el término independiente en el vector d.
- Primero se copia en x el vector d

$$x = d;$$

- Luego se triangulariza la matriz y el término independiente mediante el siguiente bucle

```
Para j=1:n-1
  mu=a(j)/b(j);
  b(j+1)=b(j+1)-mu*c(j);
  x(j+1)=x(j+1)-mu*x(j);
end para
```

Matrices tridiagonales

- Posteriromente, se realiza la sustitución regresiva

```
x(n)=x(n)/b(n);
Para j=n-1:-1:1
  x(j)=(x(j)-c(j)*x(j+1))/b(j);
end para
```

- No utiliza pivotación parcial y es mucho más rápido que el algoritmo de Gauss.

Matrices banda

Sea $A \in \mathbb{R}^{n\times n}$ es una matriz banda con $r =$ ancho de banda superior y $s =$ ancho de banda inferior. La descomposició LU de A está formada por matrices L y U que también son banda.
Si A está almacenada en formato denso la siguiente función implementa la descomposición LU de A

```
function [L,U]=blu(A,r,s)
% calcula L matriz triang. inferior con ancho de banda r
% U triangular superior con ancho de banda s
% L*U =A
n=size(A,1);
for k=1:n-1
   for i=k+1:min(k+r,n)
      A(i,k)=A(i,k)/A(k,k);
      for j=k+1:min(k+s,n)
          A(i,j)=A(i,j)-A(i,k)*A(k,j);
       end
    end
end
L=eye(n)+tril(A,-1);
U=triu(A);
```

Matrices a bloques

Si tenemos

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} I & 0 \\ A_{21}A_{11}^{-1} & I \end{pmatrix} \begin{pmatrix} A_{11} & A_{12} \\ 0 & S_{22} \end{pmatrix}$$

donde $S_{22} = A_{22} - A_{21}A_{11}^{-1}A_{12}$ es el complemento de Schur de A_{22}

Matrices a bloques

Otra posibilidad

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{pmatrix}$$

1. Se calcula la factorización $A_{11} = L_{11}U_{11}$
2. Se resuelve $U_{11}^T L_{21}^T = A_{21}^T$, $L_{11}U_{12} = A_{12}$ para calcular L_{21} y U_{12}.
3. Se forma el complemento de Schur $S_{22} = A_{22} - L_{21}U_{12}$.
4. Se calcula la factorización $S_{22} = L_{22}U_{22}$

Matrices a bloques

- Dada una matriz

$$A = C + UBV$$

donde C es una matriz fácilmente invertible. Se puede calcular su inversa usando la formula de Sherman-Morrison

$$A^{-1} = (C + UBV)^{-1} = C^{-1} - C^{-1}U(I + BVC^{-1}U)^{-1}BVC^{-1}$$

donde se supone que $I + BVC^{-1}U$ es no singular.

- Si

$$A = B + uv^T$$

la fórmula de Sherman-Morrison queda

$$A^{-1} = B^{-1} - \frac{B^{-1}uv^TB^{-1}}{1 + v^TB^{-1}u}$$

Matrices simétricas y definidas positivas

Recordemos que una matrix A es simétrica si cumple que $A = A^T$.
Además diremos que una matriz es definida positiva si cumple

$$x^T Ax > 0 \ , \quad \forall x \neq 0 \ .$$

Si A es una matriz simétrica y definida postiva se puede encontrar una matriz triangular inferior, L, de forma que

$$LL^T = A \ .$$

Esta descomposición se denomina descomposición de Cholesky de la matriz A.

Matrices simétricas y definidas positivas

Veamos cómo se puede calcular la matriz L. Para ello, consideremos un caso 3×3.

$$\begin{pmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{pmatrix} \begin{pmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_{33} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} ,$$

calculando el producto, se tienen las relaciones

$$\begin{aligned} a_{11} &= l_{11}^2 , \\ a_{22} &= l_{21}^2 + l_{22}^2 , \\ a_{33} &= l_{31}^2 + l_{32}^2 + l_{33}^2 , \\ a_{12} &= l_{11} l_{21} , \\ a_{13} &= l_{11} l_{31} , \\ a_{23} &= l_{21} l_{31} + l_{22} l_{32} \end{aligned}$$

Matrices simétricas y definidas positivas

o sea,

$$
\begin{aligned}
l_{11} &= (a_{11})^{\frac{1}{2}} \ , \\
l_{21} &= \frac{a_{12}}{l_{11}} \ , \\
l_{22} &= \left(a_{22} - l_{21}^2\right)^{\frac{1}{2}} \ , \\
l_{31} &= \frac{a_{13}}{l_{11}} \ , \\
l_{32} &= \frac{a_{23} - l_{21} l_{31}}{l_{22}} \ , \\
l_{33} &= \left(a_{33} - l_{31}^2 - l_{32}^2\right)^{\frac{1}{2}} \ .
\end{aligned}
$$

Matrices simétricas y definidas positivas

Para una matriz $n \times n$, los elementos de L se pueden calcular mediante las expresiones

$$
\begin{aligned}
l_{ii} &= \left(a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2 \right)^{\frac{1}{2}} , \\
l_{ji} &= \frac{1}{l_{ii}} \left(a_{ij} - \sum_{k=1}^{i-1} l_{ik} l_{jk} \right) , \quad j = i+1, i+2, \ldots , n \ .
\end{aligned}
$$

Matrices simétricas y definidas positivas

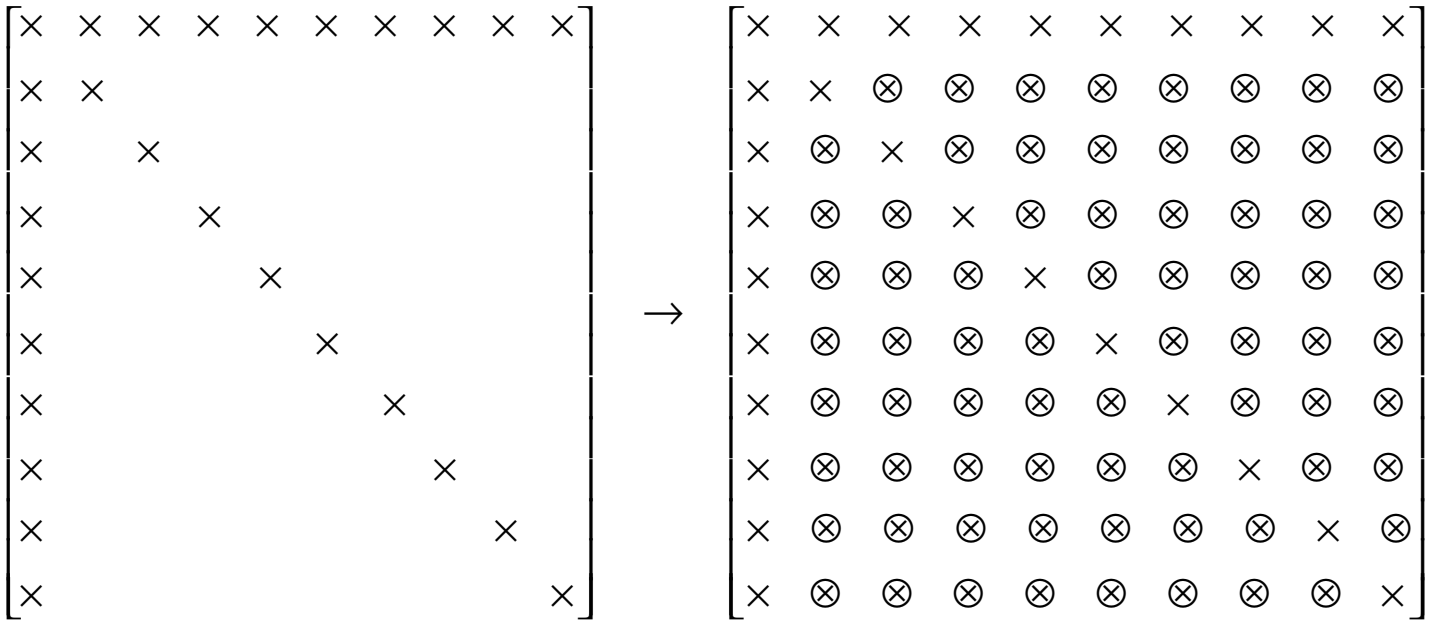
La siguiente función implementa la descomposición de Choleski.

```
function L=cholf(A)
% calcula la descomposicion de Choleski de A
% A=LLT

n=size(A,1);
L=zeros(n,n);
for j=1:n
  k=1:j-1
    L(j,j)=sqrt(A(j,j)-L(j,k)*L(j,k)');
    for i=j+1:n
       L(i,j)=(A(i,j)-L(i,k)*L(i,k)')/L(j,j);
    end
end
```

Matrices dispersas

Uno de los principales problemas que se tienen para aplicar el método de Gauss a una matriz dispersa, es el fenómeno que se conoce como relleno.



Matrices dispersas

Ejemplos:

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0.1 \\ 0 & 1 & 0 & 0 & 0.1 \\ 0 & 0 & 1 & 0 & 0.1 \\ 0 & 0 & 0 & 1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 1 \end{pmatrix} = LU$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0.1 & 0.1 & 0.1 & 0.1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 & 0.1 \\ 0 & 1 & 0 & 0 & 0.1 \\ 0 & 0 & 1 & 0 & 0.1 \\ 0 & 0 & 0 & 1 & 0.1 \\ 0 & 0 & 0 & 0 & 0.96 \end{pmatrix}$$

Matrices dispersas

Ejemplos:

$$A' = \begin{pmatrix} 1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 1 & 0 & 0 & 0 \\ 0.1 & 0 & 1 & 0 & 0 \\ 0.1 & 0 & 0 & 1 & 0 \\ 0.1 & 0 & 0 & 0 & 1 \end{pmatrix} = L'U'$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0.1 & 1 & 0 & 0 & 0 \\ 0.1 & -0.01 & 1 & 0 & 0 \\ 0.1 & -0.01 & -0.01 & 1 & 0 \\ 0.1 & -0.01 & -0.01 & -0.01 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0 & 0.99 & -0.01 & -0.01 & -0.01 \\ 0 & 0 & 0.99 & -0.01 & -0.01 \\ 0 & 0 & 0 & 0.99 & -0.01 \\ 0 & 0 & 0 & 0 & 0.99 \end{pmatrix}$$

Matrices dispersas

- **Matrices densas**

$$P_f A P_c = LU$$

  Las matrices de permutación se eligen para mantener la estabilidad numérica.

- **Matrices dispersas**

$$P_f A P_c = LU$$

  Las matrices de permutación se eligen para mantener la estabilidad numérica y preservar un patrón disperso para las matrices L y U.

Matrices dispersas

Pasos para le resolución de un sistema disperso:

1. Ordenamiento previo de las filas y columnas de A para minimizar el relleno de L y U
2. Determinación de la posición de los no ceros de L y U. Dimensionar la memoria necesaria para almacenar L y U.
3. Cálculo y almacenamiento de L y U en estructuras dispersas.

Matrices dispersas

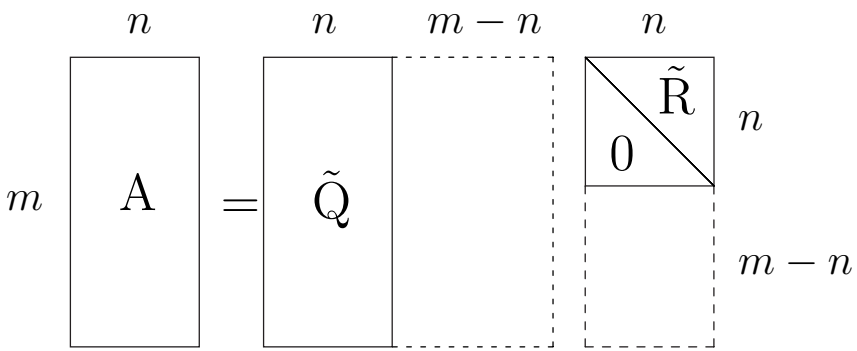
- Encontrar las permutaciones que minimizan el relleno es un problema combinatorio muy complicado (NP-completo, o sea, su coste crece exponencialmente con n).
- Los algoritmos de reordenamiento suelen ser de tipo heurístico y actúan localmente, como el algoritmo del minimum degree, o globalmente mediante técnicas de particionado de grafos como el Cuthill-Mckee.

Descomposición QR

Definición

Diremos que $A \in \mathbb{R}^{m\times n}$, con $n \geq m$ admite una factorización QR, si existe una matriz ortogonal $Q \in \mathbb{R}^{m\times n}$ y una matriz triangular superior $R \in \mathbb{R}^{m\times n}$ con filas nulas desde la fila n en adelante, de forma que

$$A = QR$$



Forma reducida

La construcción de la matriz Q se puede interpretar como un proceso de ortonormalización a partir de los vectores columna de A.

Algoritmo de Gram-Schmidt

Dados los vectores $x_1, x_2, \ldots, x_n$ se construyen los vectores ortogonales

$$q_1 = x_1$$

$$q_{k+1} = x_{k+1} - \sum_{i=1}^{k} \frac{(q_i, x_{k+1})}{(q_i, q_i)} q_i$$

A las columnas de A, $a_1, a_2, \ldots, a_n$ se aplica:

$$\tilde{q}_1 = \frac{a_1}{\|a_1\|_2}$$

$$q_{k+1} = a_{k+1} - \sum_{j=1}^{k} (\tilde{q}_j, a_{k+1})\, \tilde{q}_j$$

$$\tilde{q}_{k+1} = \frac{q_{k+1}}{\|q_{k+1}\|_2}$$

Algoritmo de Gram-Schmidt

- El método de Gram-Schmidt así formulado no es estable numéricamente ya que los vectores pierden la ortogonalidad debido a los errores de redondeo. Por ello, se suele impementar el Método de Gram-schmidt modificado.
- Este método después de calcular $(\tilde{q}_1, a_{k+1})\,\tilde{q}_1$ en el paso $k+1$, este vector se resta de a_{k+1},

$$a_{k+1}^{(1)} = a_{k+1} - (\tilde{q}_1, a_{k+1})\,\tilde{q}_1$$

Algoritmo de Gram-Schmidt

- Este nuevo vector se proyecta en la dirección de $\tilde{q}_2$ y la proyección obtenida se resta de $a_{k+1}^{(1)}$,

$$a_{k+1}^{(2)} = a_{k+1}^{(1)} - \left(\tilde{q}_2, a_{k+1}^{(1)}\right) \tilde{q}_2$$

y así hasta calcular $a_{k+1}^{(k)}$.

$$\begin{aligned} a_{k+1}^{(k)} &= a_{k+1} - (\tilde{q}_1, a_{k+1})\, \tilde{q}_1 - (\tilde{q}_2, a_{k+1} - (\tilde{q}_1, a_{k+1})\, \tilde{q}_1)\, \tilde{q}_2 + \cdots \\ &= a_k + 1 - \sum_{j=1}^{k} (\tilde{q}_j, a_{k+1})\, \tilde{q}_j \end{aligned}$$

Algoritmo de Gram-Schmidt

```
function [Q R]=cgs(A)
%    Calcula la factorizacion QR de A
%    usando el metodo de Gram-Schmidt algorithm clasico.

[n m] = size(A);
Q = zeros(n,m);
R = zeros(m,m);

for j=1:m

  qhat = A(:,j);
  R(:,j) = Q'*qhat
  qhat = qhat - Q*R(:,j);
  R(j,j) = norm(qhat);
  Q(:,j) = qhat/R(j,j);

end
```

Algoritmo de Gram-Schmidt

```
function [Q, R] = mgs(A)
%    Calcula la factorizacion QR de A
%    usando el metodo de Gram-Schmidt modificado.

[n m] = size(A);
Q = zeros(n,m); R = zeros(m,m); qhat = A(:,1);
 R(1,1) = norm(qhat);
 qhat = qhat/R(1,1);
 Q(:,1) = qhat;
 for j=2:m
   R(j-1,j:m) = qhat'*A(:,j:m);
   A(:,j:m) = A(:,j:m) - qhat*R(j-1,j:m);
   qhat = A(:,j);
   R(j,j) = norm(qhat);
   qhat = qhat/R(j,j);
   Q(:,j) = qhat;
end
```

Transformaciones de Householder

Las transformaciones de Householder son transformaciones ortogonales, que se pueden expresar como

$$H = I - 2vv^T$$

donde v es un vector unitario.
Se tiene

$$H^2 = (I - 2vv^T)(I - 2vv^T) = I - 4vv^T + 4vv^Tvv^T = I$$

así $H = H^{-1} = H^T$.
Dado un vector X se elige v de forma que

$$Hx = \begin{pmatrix} \alpha \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \alpha e_1$$

Transformaciones de Householder

Se toma

$$u = x - s\|x\|_2 e_1$$

donde $s = \pm 1 = \text{sign}\,(x_1)$

$$Hx = \left(I - 2\frac{uu^T}{u^T u}\right) x = s\|x\|_2 e_1$$

Se tiene

$$u^T x = \|x\|_2^2 - s x_1 \|x\|_2$$
$$u^T u = 2\left(\|x\|_2^2 - s x_1 \|x\|_2\right) = 2u^T x$$

Transformaciones de Householder

Así

$$Hx = x - 2u\frac{u^T x}{u^T u} = x - u = s\|x\|_2 e_1$$

Se puede escribir

$$u^T u = -2s\|x\|_2 u_1$$

donde

$$u_1 = x_1 - s\|x\|_2$$

Definiendo $w = u/u_1$

$$H = I - 2\frac{ww^T}{w^T w} = I + s\frac{u_1}{\|x\|_2}ww^T = I - \tau ww^T$$

donde

$$\tau = -s\frac{u_1}{\|x\|_2}$$

Transformaciones de Householder

Se pueden usar las transformaciones de Householder para llevar una matriz A a una matriz traingular superior.
Dada la matriz A, el primer caso consiste en

$$A_1 = H(a_1)A = H_1 A = \begin{pmatrix} * & * & * & \cdots & * \\ 0 & * & * & \cdots & * \\ 0 & * & * & \cdots & * \\ \vdots & \vdots & & & \\ 0 & * & * & \cdots & * \end{pmatrix}$$

Transformaciones de Householder

Se considera

$$H_2 = \left(\begin{array}{c|ccc} 1 & 0 & 0 & \cdots \; 0 \\ \hline 0 & & & \\ \vdots & & H\left(\tilde{a}_2\right) & \\ 0 & & & \end{array}\right)$$

y se tiene

$$A_2 = H_2 A_1 = \begin{pmatrix} * & * & * & * & \cdots & * \\ 0 & * & * & * & \cdots & * \\ 0 & 0 & * & * & \cdots & * \\ 0 & 0 & * & * & \cdots & * \\ \vdots & \vdots & & & & \vdots \\ 0 & 0 & * & * & \cdots & * \end{pmatrix}$$

y, por tanto

$$H_{m-1} H_{m-2} \cdots H_2 H_1 A = R$$

Rotaciones de Givens

Las rotaciones de Givens tienen la forma genérica siguiente:

$$G = \begin{pmatrix} \cos(\alpha) & -\operatorname{sen}(\alpha) \\ \operatorname{sen}(\alpha) & \cos(\alpha) \end{pmatrix}$$

de forma que

$$G \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \tilde{a} \\ 0 \end{pmatrix} \ , \quad GG^T = I \ .$$

Rotaciones de Givens

Se tiene pues que

$$\cos(\alpha)a - \text{sen}(\alpha)b = \tilde{a}$$
$$\text{sen}(\alpha)a + \cos(\alpha)b = 0$$

o sea,

$$\text{sen}(\alpha) = -\frac{b}{\sqrt{a^2+b^2}}, \cos(\alpha) = \frac{a}{\sqrt{a^2+b^2}}.$$

Las rotaciones de Gives se aplican a pares sucesivos de filas para conseguir hacer los ceros suficientes para obtener el factor R de la descomposición QR.

$$G_N G_{N-1} \cdots G_2 G_1 A = R.$$

Problema de mínimos cuadrados

Se parte de un sistema sobredeterminado

$$Ax = b\ , \quad A \in \mathbb{R}^{m\times n}, \quad m > n.$$

La solución de mínimos cuadrados

$$A^T Ax = A^T b$$

Se tiene que

$$\kappa(A^T A) = (\kappa(A))^2$$

Si A está mal condicionada, se usa la descomposición $A = QR$.

Problema de mínimos cuadrados

Se quiere minimizar

$$\|Ax - b\|^2$$

La norma no se ve alterada si se multiplica por una matriz ortogonal

$$\left\|Q^T (Ax - b)\right\|^2 = \left\|Rx - Q^T b\right\|^2$$

Separando las n primeras filas

$$\left\| \begin{pmatrix} R_1 \\ 0 \end{pmatrix} x - \begin{pmatrix} Q_1^T b \\ Q_2^T b \end{pmatrix} \right\|^2 =$$

$$\left\| \begin{pmatrix} R_1 x - Q_1^T b \\ Q_2^T b \end{pmatrix} \right\|^2 = \left\|R_1 x - Q_1^T b\right\|^2 + \left\|Q_2^T b\right\|^2$$

El mínimo es

$$\boxed{R_1 x = Q_1^T b}$$

Problema de mínimos cuadrados

Se parte ahora de un sistema indeterminado

$$Ax = b\ , \quad A \in \mathbb{R}^{m\times n}, \quad m < n.$$

Se calcula la descomposición QR

$$A^T = QR \quad \Rightarrow \quad A = R^T Q^T = \begin{pmatrix} R_1^T & 0 \end{pmatrix} \begin{pmatrix} Q_1^T \\ Q_2^T \end{pmatrix}$$

Así

$$Ax = R_1^T q_1^T x + 0 Q_2^T x = b$$

y, por tanto,

$$\boxed{x = Q_1 \left(R_1^T\right)^{-1} b}$$