**Francesca Mazzia**

**Dipartimento di Matematica**

**Università di Bari**

# Algoritmi per la soluzione di sistemi lineari.

## Sistemi triangolari inferiori

Sia $L$ triangolare inferiore. La matrice è non singolare se gli elementi principali $l_{i,i} \neq 0$, per $i = 1, \cdots, n$.

Il sistema $Lx = b$ può scriversi:

$$\begin{aligned}
& l_{1,1}x_1 && = b_1 \\
& l_{2,1}x_1 + l_{2,2}x_2 && = b_2 \\
& l_{3,1}x_1 + l_{3,2}x_2 + l_{3,3}x_3 && = b_3 \\
& \ldots \\
& \ldots \\
& l_{n,1}x_1 + l_{n,2}x_2 - \ldots + l_{n,n}x_n && = b_n
\end{aligned}$$

da cui si ottiene:

$$x_1 = \frac{b_1}{l_{1,1}}, \qquad x_2 = \frac{b_2 - l_{2,1}x_1}{l_{2,2}}$$

$$x_3 = \frac{b_3 - l_{3,1}x_1 - l_{3,2}x_2}{l_{3,3}}$$

in generale

$$x_i = \frac{b_i - \sum_{j=1}^{i-1} l_{i,j} x_j}{l_{i,i}} \qquad i = 1, 2, \ldots, n$$

Essendo $l_{i,i} \neq 0$ queste formule determinano in modo univoco $x_i$.

La procedura descritta si chiama algoritmo di sostituzione in avanti, in Matlab è descritta dal seguente algoritmo:

```
function b = soll(L,b)
n = length(b);
b(1) = b(1)/L(1,1);
for i=2:n
  b(i) = (b(i) - L(i,1:i-1)*b(1:i-1))/L(i,i);
end
```

L'istruzione matlab `L(i,1:i-1)*b(1:i-1)` esegue il prodotto riga per colonne fra il vettore `L(i,1:i-1)` e il vettore `b(1:i-1)`.

`L(i,1:i-1)` elementi

`( L(i,1), L(i,2), ···, L(i,i-1))`

`b(1:i-1)` elementi

`( b(1), b(2), ···, b(i-1))`

Costo computazionale:

moltiplicazioni :

`L(i,1:i-1)*b(1:i-1)`

richiede i-1 moltiplicazioni

viene eseguita per `i=2,n` sommando

$$\sum_{2}^{n}(i-1) = \sum_{1}^{n-1} i = \frac{n(n-1)}{2}$$

Il numero di addizioni è dello stesso tipo.

## Sistemi triangolari superiori

$$\begin{aligned} u_{1,1}x_1 + u_{1,2}x_2 + \ldots + u_{1,n}x_n &= b_1 \\ u_{2,2}x_2 + \ldots + u_{2,n}x_n &= b_2 \\ \ldots & \\ \ldots & \\ u_{n,n}x_n &= b_n \end{aligned}$$

La matrice è non singolare se $u_{i,i} \neq 0$.

La soluzione di questo sistema è:

$$x_n = \frac{b_n}{u_{n,n}}, \qquad x_{n-1} = \frac{b_{n-1} - u_{n-1,n}x_n}{u_{n-1,n-1}}$$

$$x_{n-2} = \frac{b_{n-2} - u_{n-2,n-1}x_{n-1} - u_{n-2,n}x_n}{u_{n-2,n-2}}$$

In generale:

$$x_i = \frac{1}{u_{i,i}} \left( b_i - \sum_{j=i+1}^{n} u_{i,j} x_j \right) \qquad i = n, n-1, \ldots, 1$$

La procedura descritta si chiama algoritmo di sostituzione all'indietro, in Matlab è descritta dal seguente algoritmo:

```
function b = solu(U,b)
n = length(b);
b(n) = b(n)/U(n,n);
for i=n-1:-1:1
  b(i) = (b(i) - U(i,i+1:n)*b(i+1:n))/U(i,i);
end
```

## Matrici di permutazione

$P_{r,s}$ matrici di permutazione elementari si ottengono dalla matrice $I$ di dimensione $n \times n$ scambiando la $r$.ma riga con la $s$.ma.

$$P_{r,s} = \begin{array}{c} \\ \\ \\ \\ r \\ \\ \\ \\ s \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccccccccc} r & & & & & & s & & & & \end{array} \\ \begin{pmatrix} 1 & 0 & & & & & & & & & \\ 0 & \ddots & \ddots & & & & & & & & \\ & \ddots & 1 & & & & & & & & \\ & & & 0 & 0 & \dots & 0 & 1 & & & \\ & & & 0 & 1 & & & 0 & & & \\ & & & \vdots & & \ddots & & \vdots & & & \\ & & & 0 & & & 1 & 0 & & & \\ & & & 1 & 0 & \dots & 0 & 0 & & & \\ & & & & & & & & 1 & \ddots & \\ & & & & & & & & \ddots & \ddots & 0 \\ & & & & & & & & & 0 & 1 \end{pmatrix} \end{array}$$

$P_{r,s}A$ scambia le righe $r$ ed $s$

$AP_{r,s}$ scambia le colonne $r$ ed $s$

## Proprietà delle matrici di permutazione

Siano $P, P_1, P_2$ matrici di permutazione $n \times n$ e $A$ una matrice $n \times n$ allora:

1. PA è come A con le righe permutate.

   AP è come A con le colonne permutate

2. $P^{-1} = P^T$

3. $\det(P) = \pm 1$

4. $P_1 P_2$ è un matrice di permutazione.

Algoritmo di eliminazione di Gauss per risolvere $Ax = b$:

1. Fattorizzare $A$ in $A = PLU$ con

   P = matrice di permutazione

   L = matrice triangolare inferiore con elementi diagonali uguali a 1

   U = matrice triangolare superiore non singolare

2. Risolvere $Pz = b$, permutando le righe di $b$: $z = LUx = P^{-1}b$

3. risolvere$Ly = P^{-1}b$,$y = Ux$ con l'algoritmo di sostituzione in avanti

4. risolvere $Ux = L^{-1}(P^{-1}b)$ con l'algoritmo di sostituzione all'indietro : $x = U^{-1}(L^{-1}(P^{-1}b))$

## Algoritmo di eliminazione di Gauss

Consideriamo il sistema lineare:

$$\begin{aligned}
a_{1,1}x_1 + a_{1,2}x_2 + a_{1,3}x_3 + a_{1,4}x_4 &= b_1 \\
a_{2,1}x_1 + a_{2,2}x_2 + a_{2,3}x_3 + a_{2,4}x_4 &= b_2 \\
a_{3,1}x_1 + a_{3,2}x_2 + a_{3,3}x_3 + a_{3,4}x_4 &= b_2 \\
a_{4,1}x_1 + a_{4,2}x_2 + a_{4,3}x_3 + a_{4,4}x_4 &= b_2
\end{aligned}$$

Sia $a_{1,1} \neq 0$, poniamo $m_{i,1} = a_{i,1}/a_{1,1}$, $i = 2, 3, 4$

e sottraiamo $m_{i,1}$ moltiplicato per la prima equazione dalla equazione $i$esima.

$$\begin{aligned}
a_{1,1}x_1 + \quad & a_{1,2}x_2 + a_{1,3}x_3 + a_{1,4}x_4 = b_1 \\
& a^{(1)}_{2,2}x_2 + a^{(1)}_{2,3}x_3 + a^{(1)}_{2,4}x_4 = b^{(1)}_2 \\
& a^{(1)}_{3,2}x_2 + a^{(1)}_{3,3}x_3 + a^{(1)}_{3,4}x_4 = b^{(1)}_3 \\
& a^{(1)}_{4,2}x_2 + a^{(1)}_{4,3}x_3 + a^{(1)}_{4,4}x_4 = b^{(1)}_4
\end{aligned}$$

con

$$a^{(1)}_{i,j} = a_{i,j} - m_{i,1}a_{1,j}, \qquad b^{(1)}_i = b_i - m_{i,1}b_1$$

La variabile $x_1$ è stata eliminata dalle ultime tre equazioni. Sia $a^{(1)}_{2,2} \neq 0$, poniamo $m_{i,2} = a^{(1)}_{i,2}/a^{(1)}_{2,2}, \qquad i = 3,4$ e sottraiamo $m_{i,2}$ moltiplicato per la seconda equazione dalla equazione $i$esima.

$$\begin{array}{rrl} a_{1,1}x_1+ & a_{1,2}x_2+ & a_{1,3}x_3 + a_{1,4}x_4 = b_1 \\ & a^{(1)}_{2,2}x_2+ & a^{(1)}_{2,3}x_3 + a^{(1)}_{2,4}x_4 = b^{(1)}_2 \\ & & a^{(2)}_{3,3}x_3 + a^{(2)}_{3,4}x_4 = b^{(2)}_3 \\ & & a^{(2)}_{4,3}x_3 + a^{(2)}_{4,4}x_4 = b^{(2)}_4 \end{array}$$

con

$$a^{(2)}_{i,j} = a^{(1)}_{i,j} - m_{i,2}a^{(1)}_{2,j}, \qquad b^{(2)}_i = b^{(1)}_i - m_{i,2}b^{(1)}_2$$

La variabile $x_2$ è stata eliminata dalle ultime due equazioni.

Infine, sia $a_{3,3}^{(2)} \neq 0$, poniamo $m_{i,3} = a_{i,3}^{(2)}/a_{3,3}^{(2)}$, $i =$ 4 e sottraiamo $m_{i,3}$ moltiplicato per la terza equazione dalla equazione $i$esima.

La variabile $x_3$ viene eliminata dall' ultima equazione.

Otteniamo il sistema triangolare superiore:

$$\begin{array}{llll} a_{1,1}x_1+ & a_{1,2}x_2+ & a_{1,3}x_3+ & a_{1,4}x_4 = b_1 \\ & a_{2,2}^{(1)}x_2+ & a_{2,3}^{(1)}x_3+ & a_{2,4}^{(1)}x_4 = b_2^{(1)} \\ & & a_{3,3}^{(2)}x_3+ & a_{3,4}^{(2)}x_4 = b_3^{(2)} \\ & & & a_{4,4}^{(3)}x_4 = b_4^{(3)} \end{array}$$

con

$$a_{i,j}^{(3)} = a_{i,j}^{(2)} - m_{i,3}a_{3,j}^{(2)}, \qquad b_i^{(2)} = b_i^{(3)} - m_{i,2}b_2^{(3)}$$

Questo algoritmo può essere esteso ad un sistema di qualsiasi ordine.

Sia $A_1 = A$ e

$$M_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -m_{2,1} & 1 & 0 & 0 \\ -m_{3,1} & 0 & 1 & 0 \\ -m_{4,1} & 0 & 0 & 1 \end{pmatrix}$$

allora

$$A_2 = M_1 A_1 =$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ -m_{2,1} & 1 & 0 & 0 \\ -m_{3,1} & 0 & 1 & 0 \\ -m_{4,1} & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{pmatrix} =$$

$$= \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ 0 & a_{2,2}^{(1)} & a_{2,3}^{(1)} & a_{2,4}^{(1)} \\ 0 & a_{3,2}^{(1)} & a_{3,3}^{(1)} & a_{3,4}^{(1)} \\ 0 & a_{4,2}^{(1)} & a_{4,3}^{(1)} & a_{4,4}^{(1)} \end{pmatrix}$$

$$M_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -m_{3,2} & 1 & 0 \\ 0 & -m_{4,2} & 0 & 1 \end{pmatrix}$$

$$A_3 = M_2 A_2 =$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -m_{3,2} & 1 & 0 \\ 0 & -m_{4,2} & 0 & 1 \end{pmatrix} \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ 0 & a_{2,2}^{(1)} & a_{2,3}^{(1)} & a_{2,4}^{(1)} \\ 0 & a_{3,2}^{(1)} & a_{3,3}^{(1)} & a_{3,4}^{(1)} \\ 0 & a_{4,2}^{(1)} & a_{4,3}^{(1)} & a_{4,4}^{(1)} \end{pmatrix} =$$

$$= \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ 0 & a_{2,2}^{(1)} & a_{2,3}^{(1)} & a_{2,4}^{(1)} \\ 0 & 0 & a_{3,3}^{(2)} & a_{3,4}^{(2)} \\ 0 & 0 & a_{4,3}^{(2)} & a_{4,4}^{(2)} \end{pmatrix}$$

Infine

$$M_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -m_{4,3} & 1 \end{pmatrix}$$

$$U = M_3 A_3 =$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -m_{4,3} & 1 \end{pmatrix} \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ 0 & a_{2,2}^{(1)} & a_{2,3}^{(1)} & a_{2,4}^{(1)} \\ 0 & 0 & a_{3,3}^{(2)} & a_{3,4}^{(2)} \\ 0 & 0 & a_{4,3}^{(2)} & a_{4,4}^{(2)} \end{pmatrix}$$

$$= \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ 0 & a_{2,2}^{(1)} & a_{2,3}^{(1)} & a_{2,4}^{(1)} \\ 0 & 0 & a_{3,3}^{(2)} & a_{3,4}^{(2)} \\ 0 & 0 & 0 & a_{4,4}^{(3)} \end{pmatrix}$$

$U = M_3 M_2 M_1 A$ triangolare superiore.

Poniamo

$L = M_1^{-1} M_2^{-1} M_3^{-1}$

allora

$A = LU$

Possiamo scrivere la matrice

$M_1 = I - m_1 e_1^T$ con $m_1 = (0, m_{2,1}, m_{3,1}, m_{4,1})^T$

Si dimostra che

$M_1^{-1} = I + m_1 e_1^T$

infatti, essendo $(e_1^T m_1) = 0$

$M_1^{-1} M_1 = (I + m_1 e_1^T)(I - m_1 e_1^T) =$

$= I + m_1 e_1^T - m_1 e_1^T + m_1 (e_1^T m_1) e_1^T = I.$

$M_2 = I - m_2 e_2^T$ con $m_2 = (0, 0, m_{3,2}, m_{4,2})^T$

e $M_2^{-1} = I + m_2 e_2^T$

$M_3 = I - m_3 e_3^T$ con $m_3 = (0, 0, 0, m_{4,3})^T$

$M_3^{-1} = I + m_3 e_3^T$

Inoltre, essendo $e_i^T m_j = 0, \quad i < j$,

$L = (I + m_1 e_1^T)(I + m_2 e_2^T)(I + m_3 e_3^T) =$

$= I + m_1 e_1^T + m_2 e_2^T + m_3 e_3^T$

quindi

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ m_{2,1} & 1 & 0 & 0 \\ m_{3,1} & m_{3,2} & 1 & 0 \\ m_{4,1} & m_{4,2} & m_{4,3} & 1 \end{pmatrix}$$

```
function [L,U] = gauss(A)
[n,m] = size(A);


for k=1:n-1
    if abs(A(k,k)) <= realmin
      error(' Un elemento pivotale e'' piccolo')
    else
      L(k+1:n,k)= A(k+1:n,k)/A(k,k);
    end
    A(k+1:n,k+1:n)=A(k+1:n,k+1:n) - ....
    L(k+1:n,k)*A(k,k+1:n);
end
U = triu(A);
end
```

## Numero di operazioni

**1)** Calcolo dei coefficienti $m_{i,j}$. Questi sono $n-1$ per la prima colonna, $n-2$ per la seconda, fino ad 1 per la penultima. In totale abbiamo:

$$\sum_{k=1}^{n-1}(n-k) = \frac{n(n-1)}{2}$$

divisioni.

**2)** Calcolo degli $a_{i,j}^{(k)}$. Questi sono $(n-1)^2$ al primo passo ($k=2$), $(n-2)^2$ al secondo passo fino ad 1 all'ultimo passo ($k=n-1$). In totale il numero degli $a_{i,j}^{(k)}$ calcolati sono:

$$\sum_{k=1}^{n-1}(n-k)^2 = \frac{n(n-1)(2n-1)}{6} \sim \frac{n^3}{3}.$$

Riassumendo:

moltiplicazioni e divisioni:$\sim n^3/3$ sottrazioni: $\sim n^3/3$.

Problemi dell'algoritmo di fattorizzazione LU senza permutazioni:

Non può essere eseguita su matrici come:

$$P = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

$$A = \begin{pmatrix} 1 & 2 & 5 \\ -1 & -2 & -7 \\ 1 & 1 & 3 \end{pmatrix}$$

Presenta problemi di instabilità numerica:

$$\begin{cases} 10^{-17}x_1 + x_2 = 1 \\ x_1 + x_2 = 2 \end{cases}$$

Il problema è ben posto, ed ha come soluzione esatta $x \simeq (1,1)^T$. Applicando il metodo di Gauss ed operando in virgola mobile in base 2 in doppia precisione, si ottengono i fattori $L$ ed $U$ seguenti:

$$L = \begin{pmatrix} 1 & 0 \\ 10^{17} & 1 \end{pmatrix}, \qquad U = \begin{pmatrix} 10^{-17} & 1 \\ 0 & -10^{17} \end{pmatrix}$$

il prodotto $LU$ non è $A$ bensì la matrice seguente

$$\tilde{A} = \begin{pmatrix} 10^{-17} & 1 \\ 1 & 0 \end{pmatrix}$$

confrontando $\tilde{A}$ con $A$ si vede che l' elemento $a_{22}$ di $A$ è stato cambiato, pertanto la soluzione numerica che è $\tilde{x} = (0,1)^T$ è totalmente sbagliata.

Teorema. Se $A$ è non singolare, allora esistono due matrici di permutazione $P_1$ e $P_2$, una matrice triangolare inferiore $L$ con elementi diagonali uguali a uno e una matrice triangolare superiore $U$ tali che $P_1 A P_2 = LU$. Solo una fra $P_1$ e $P_2$ è necessaria.

$P_1 A$ scambia le righe di A,

$AP_2$ scambia le colonne,

$P_1 A P_2$ scambia righe e colonne di $A$.

Possiamo scegliere $P_2 = I$ e $P_1$ in modo tale che $a_{1,1}$ sia il più garnde elemento in valore assoluto nella colonna, il che implica che gli elementi $m_{j,1} = a_{j,1}/a_{1,1}$ sono limitati da 1 in valore assoluto. In generale al passo $i$ dell'eliminazione di Gauss, ordiniamo le righe da $i$ a $n$ in modo da portare l'elemento più grande in valore assoluto in posizione $i, i$. Questa tecnica si chiama eliminazione di Gauss con Pivot Parziale.

Possiamo scegliere $P_2$ e $P_1$ in modo tale che $a_{1,1}$ sia il più grande elemento in valore assoluto di tutta la matrice. In generale al passo $i$ dell'eliminazione di Gauss, ordiniamo le righe e le colonne da $i$ a $n$ in modo da portare l'elemento più grande in valore assoluto in posizione $i, i$. Questa tecnica si chiama eliminazione di Gauss con Pivot Totale.

Algoritmo di eliminazione di Gauss con pivot parziale.

Se eseguiamo le permutazioni partendo dalla matrice A calcoliamo:

$$M_3 P_3 M_2 P_2 M_1 P_1 A = U$$

Per ricavarci la matrice $P$ e la matrice $L$ tali che $PA = LU$ poniamo

$$\tilde{M}_2 = P_3 M_2 P_2^T = P_3(I - m_2 e_2^T) P_3^T = I - P_3 m_2 e_2^T$$

$$\tilde{M}_1 = P_3 P_2 M_2 P_2^T P_3^T = P_3 P_2 (I - m_1 e_1^T) P_2^T P_3^T$$

$$= I - P_3 P_2 m_1 e_1^T$$

e

$$P = P_3 P_2 P_1$$

quindi

$$L = \tilde{M}_1^{-1} \tilde{M}_2^{-1} M_3^{-1}$$

cioe' gli elementi dei vettori $m_i$ che costituiscono le colonne di L vanno permutati.

La funzione Matlab che esegue la fattorizzazione $LU$ con pivot è **lu**. Essa può essere richiamata con l'istruzione:

$\gg$ [L,U,P]=lu(A)

Supponiamo che $\det(A) = 0$ e $U$ abbia l'ultimo elemento diagonale nullo.

Quindi l'ultima riga di $U$ è un vettore nullo che si ottiene moltiplicando l'ultima riga di $L^{-1}$ per le righe di $A$, cioè:

$$l_{n,1}a_1^T + \ldots + l_{n,n-1}a_{n-1}^T + l_{n,n}a_n^T = 0$$

con $l_{n,i}$ non tutti nulli e ciò implica la lineare dipendenza delle righe di $A$.

L'annullarsi del determinante di una matrice quadrata è condizione necessaria e sufficiente affinchè i vettori riga siano linearmente dipendenti.

Lo stesso discorso può farsi con i vettori colonna.

Sia $Ax = b$ e $(A + \delta A)\widehat{x} = b + \delta b$, $\widehat{x} = x + \delta x$

Studiamo il comportamento di $\delta x$ in funzione della perturbazione sui dati di input.

Lemma. Sia $\|\cdot\|$ una norma matriciale. Allora $\|B\| < 1$ implica che $I - B$ è invertibile e $\|(I - B)^{-1}\| \leq 1/(1 - \|B\|)$.

Troviamo una maggiorazione relativa per $\|\delta x\|$
$\delta A x + (A + \delta A)\delta x = \delta b$
$(A + \delta A)\delta x = \delta b - \delta A x$
$(I + A^{-1}\delta A)\delta x = A^{-1}(\delta b - \delta A x)$
Se $\|A^{-1}\|\|\delta A\| < 1$ possiamo applicare il lemma e

$$\delta x = (I + A^{-1}\delta A)^{-1} A^{-1}(\delta b - \delta A x)$$

passando alle norme

$$\|\delta x\| \leq \|(I + A^{-1}\delta A)^{-1}\| \|A^{-1}\| (\|\delta b\| + \|\delta A\| \|x\|)$$

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

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

$\|b\| = \|Ax\| \leq \|A\|\|x\|$

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

La quantità $k(A) = \|A\|\|A^{-1}\|$ è il numero di condizione della matrice $A$

Residuo : $r = A\widehat{x} - b$

$r = 0$ se $\widehat{x} = x$

$A\delta x = A\widehat{x} - Ax = A\widehat{x} - b$

$\delta x = A^{-1}r$

$\|\delta x\| \leq \|A^{-1}\|\|r\|$

Si consideri la matrice $A$ e il vettore $b$ definiti da:

$$A = \begin{bmatrix} 1.2969 & 0.8648 \\ 0.2161 & 0.1441 \end{bmatrix}, b = \begin{bmatrix} 0.8642 \\ 0.1440 \end{bmatrix}.$$

La soluzione esatta del sistema $Ax = b$ è $x = [2, -2]^T$.

Sia $\widehat{x} = [0.9911, -0.4870]^T$, e sia $r = A\widehat{x} - b$, quindi $r = [-10^{-8}, 10^{-8}]^T$.

Questo valore di $r$ fa intendere che $\widehat{x}$ è una buona approssimazione della soluzione esatta anche se in realtà non lo è.

La matrice inversa di $A$ è data da

$$A^{-1} = -10^{-8} \begin{bmatrix} 0.8648 & -0.1441 \\ -1.2969 & 0.2161 \end{bmatrix}$$

per cui $\kappa_\infty(A) = 3\ 10^8$, quindi $A$ è mal condizionata.

Il residuo non è una buona indicazione di precisione della soluzione di un sistema quando la matrice è mal condizionata. Infatti, vale la disuguaglianza

$$\frac{\|\delta x\|}{\|x\|} \leq \kappa(A)\frac{\|r\|}{\|b\|}$$

che si ottiene da

$$\|\delta x\| \leq \|A^{-1}\|\|r\|$$

dividendo per $\|x\|$ e sfruttando la disuguaglianza $\|b\| \leq \|A\| \cdot \|x\|$.

Sistema lineare Ax=b :

$$A = \begin{bmatrix} 1 & 1 \\ 0.99 & 1 \end{bmatrix}, b = \begin{bmatrix} 2 \\ 1.99 \end{bmatrix}$$

soluzione $x = [1, 1]^T$. La matrice inversa di $A$ è

$$A^{-1} = \frac{1}{0.01} \begin{bmatrix} 1 & -1 \\ -0.99 & 1 \end{bmatrix}$$

$\kappa_\infty(A) = 400$. Perturbiamo $A$ sommando ad essa la seguente matrice

$$\delta A = 0.002 \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}.$$

Essendo $\|A^{-1}\|_\infty = 200$ e $\|\delta A\|_\infty = 0.004$ si ha $\|A^{-1}\|_\infty \|\delta A\|_\infty = 0.8$, quindi il sistema perturbato

$$(A + \delta A)\widehat{x} = b$$

ha matrice dei coefficienti non singolare. La soluzione del sistema perturbato è
$\widehat{x} = [0.2016..., 1.794...]^T$, per cui
$\|\delta x\|_\infty / \|x\|_\infty = 0.3992....$ La maggiorazione calcolata ci dice $\|\delta x\|_\infty / \|x\|_\infty \leq 4$.

Ci sono sistemi che non sono mal condizionati ma per i quali il numero di condizione della matrice è grande. Esempio:

$$A = \begin{bmatrix} u & 0 \\ 0 & 1/u \end{bmatrix}.$$

con $u$ sufficientemente piccolo, ha numero di condizione $\kappa_1(A) = (1/u)^2$. Se si moltiplica la seconda equazione per $u^2$, cioè se si scala la matrice, risulta

$$A' = \begin{bmatrix} u & 0 \\ 0 & u \end{bmatrix}$$

che ha numero di condizione pari ad 1.

Il condizionamento di una matrice può essere modificato dallo *scaling*.
Si va alla ricerca di due matrici diagonali $D_1$ e $D_2$ in modo da minimizzare $\kappa(D_1 A D_2)$.

Matrice di Hilbert

La matrice di Hilbert rappresenta un classico esempio di matrice *mal condizionata* reale, nel senso che ha origine da modelli reali.

Gli elementi della matrice di Hilbert sono dati dalla espressione seguente:

$$a_{ij}^{(n)} = \frac{1}{i+j-1}, \quad i,j = 1,...,n$$

La tabella riporta i valori del numero di condizione di A, in norma 2 e in norma $\infty$, per diversi valori di $n$.

| $n$ | $\kappa_2(A^{(n)})$ | $\kappa_\infty(A^{(n)})$ |
|---|---|---|
| 2 | 1.505 | 27 |
| 3 | $5.241\ 10^2$ | $7.480\ 10^2$ |
| 4 | $1.551\ 10^4$ | $2.837\ 10^4$ |
| 5 | $4.766\ 10^5$ | $9.436\ 10^5$ |
| 6 | $1.495\ 10^7$ | $2.907\ 10^7$ |
| 7 | $4.754\ 10^8$ | $9.852\ 10^8$ |
| 8 | $1.526\ 10^{10}$ | $3.387\ 10^{10}$ |
| 9 | $4.932\ 10^{11}$ | $1.099\ 10^{12}$ |
| 10 | $1.603\ 10^{13}$ | $3.535\ 10^{13}$ |

## Matrici diagonali dominanti

Sia $A$ una matrice di ordine $n$ diagonale dominante per colonne. Allora i fattori di $A$ ottenuti con il metodo di Gauss coincidono con quelli che si ottengono con il metodo di Gauss con pivot parziale.

**Esempio:** matrice diagonale dominante per righe

$$A = \begin{bmatrix} 3 & 1 & 1 \\ 4 & 8 & 2 \\ 2 & 2 & 5 \end{bmatrix}$$

I fattori di $A$ ottenuti con il metodo di Gauss sono:

$$L = \begin{bmatrix} 1 & & \\ \frac{4}{3} & 1 & \\ \frac{2}{3} & \frac{1}{5} & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 3 & 1 & 1 \\ & \frac{20}{3} & \frac{2}{3} \\ & & \frac{21}{5} \end{bmatrix},$$

mentre i fattori ottenuti con il metodo di Gauss con pivot parziale sono:

$$L = \begin{bmatrix} 1 & & \\ \frac{3}{4} & 1 & \\ \frac{1}{2} & \frac{2}{5} & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 4 & 8 & 2 \\ & -5 & -\frac{2}{3} \\ & & \frac{21}{5} \end{bmatrix}.$$

Nel primo caso $L$ non è diagonale predominante, nel secondo caso è $U$ a non essere diagonale predominante. Utilizzando il metodo di Gauss con pivot totale si ottengono i seguenti fattori:

$$L = \begin{bmatrix} 1 & & \\ \frac{1}{8} & 1 & \\ \frac{1}{4} & \frac{1}{6} & 1 \end{bmatrix}, \qquad U = \begin{bmatrix} 8 & 2 & 4 \\ & \frac{9}{2} & 1 \\ & & \frac{7}{3} \end{bmatrix}.$$

In questo caso i fattori sono entrambi diagonali predominanti.

## Matrici simmetriche definite positive

Si incontrano nelle applicazioni.

Sono le più semplici da trattare.

La loro fattorizzazione non richiede il procedimento di pivoting:
si può dimostrare che $a_{k,k}^{(k)} > 0$ .

Sia $D$ la matrice diagonale formata dagli elementi pivotali, cioè dagli elementi sulla diagonale principale della matrice $U$, si ha:

$$A = LU = LDD^{-1}U = LD\hat{U}$$

$\hat{U} = D^{-1}U$ è una matrice triangolare superiore con elementi diagonali uguali ad 1.

Poichè $A$ è simmetrica si ha:

$$A^T = \hat{U}^T D L^T = A = L D \hat{U},$$

con $\hat{U}^T$ triangolare inferiore e $L^T$ triangolare superiore.

L'uguaglianza è verificata se $\hat{U}^T = L$ quindi

$$A = L D L^T.$$

Poichè abbiamo detto che gli elementi sulla diagonale di $D$ sono positivi, si può definire la matrice $D^{1/2}$, con elementi le radici quadrate degli elementi di $D$. Si ha:

$$A = L D^{1/2} D^{1/2} L^T.$$

Ponendo $S = L D^{1/2}$ si ha:

$$A = S S^T.$$

Questa fattorizzazione è detta fattorizzazione di Cholesky.

È più semplice della fattorizzazione $LU$ perchè ha come incognite solo gli elementi della matrice triangolare inferiore $S$.

La fattorizzazione si può ottenere direttamente uguagliando gli elementi di $A$ e quelli di $SS^T$.

Consideriamo la prima riga di $A$ e la prima riga di $SS^T$

$$
\begin{aligned}
&a_{1,1} = s_{1,1}^2 \\
&a_{1,j} = s_{1,1} s_{j,1} \quad \text{per} \quad j = 2, \ldots, n.
\end{aligned}
$$

ed è quindi possibile ricavare:

$$
\begin{aligned}
&s_{1,1} = a_{1,1}^{1/2} \\
&s_{j,1} = a_{1,j} / s_{1,1} \quad \text{per} \quad j = 2, \ldots, n.
\end{aligned}
$$

In generale si avrà:

$$
\begin{aligned}
&a_{i,i} = \textstyle\sum_{k=1}^{i} s_{i,k}^2 && \text{per} \quad i = 1, \ldots, n \\
&a_{i,j} = \textstyle\sum_{k=1}^{i} s_{i,k} s_{j,k} && \text{per} \quad i = 1, \ldots, n \\
& && \qquad\quad j = i+1, \ldots, n.
\end{aligned}
$$

Data la simmetria di $A$ è sufficiente considerare solo la parte triangolare superiore. Dalle due precedenti relazioni si ricava:

$$
\begin{aligned}
s_{i,i} &= \left(a_{i,i} - \sum_{k=1}^{i-1} s_{i,k}^2\right)^{1/2} \\
&\quad \text{per} \quad i = 1, \ldots, n \\
s_{j,i} &= \left(a_{i,j} - \sum_{k=1}^{i-1} s_{i,k} s_{j,k}\right) / s_{i,i} \\
&\quad \text{per} \quad \begin{cases} i = 1, \ldots, n \\ j = i+1, \ldots, n \end{cases} .
\end{aligned}
$$

Le istruzioni Matlab per eseguire la fattorizzazione di Cholesky sono le seguenti:

```
for i=1:n
    s(i,i) = a(i,i);
    for k=1:i–1
        s(i,i)=s(i,i) – s(i,k)^2;
    end
    s(i,i)=s(i,i)^(.5);
    for j = i+1:n
        s(j,i)=a(i,j);
        for k=1:i–1
            s(j,i) = s(j,i) – s(i,k)*s(j,k);
        end
        s(j,i)=s(j,i)/s(i,i);
    end
end
```

La funzione Matlab che calcola la fattorizzazione di Cholesky di una matrice è chol. Se A è definita positiva allora dopo l'istruzione

$\gg$ R = chol(A);

la variabile R è la matrice triangolare superiore della fattorizzazione di Cholesky.