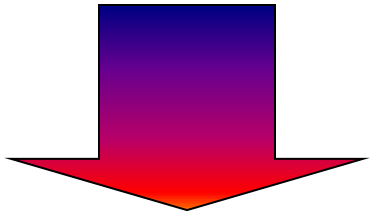
LA fattorizzazione LU

PROBLEMA

Risolvere m sistemi del tipo:

$$A\underline{x}_1 = \underline{b}_1 \; ; \; A\underline{x}_2 = \underline{b}_2 \; ; \; \dots \; ; \; A\underline{x}_m = \underline{b}_m$$

(con la stessa matrice dei coefficienti)

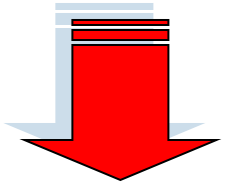


Primo approccio

Applicare Gauss + back-substitution ad ogni sistema

Qual è la complessità di tempo ?

Risoluzione di m sistemi lineari mediante l'algoritmo di Gauss



COMPLESSITA' DI TEMPO

$$m(T_{Gauss}(n)+T_{Back}(n)) \approx O(m(n^3/3+n^2/2)) = O(m(n^3/3))$$

MA .....

L'algoritmo di Gauss opera contemporaneamente

sulla matrice e sul vettore dei termini noti

poiché in questo caso la matrice dei coefficienti

e' la stessa

sui suoi elementi vengono ripetute

le stesse operazioni!

E' possibile evitare di modificare più volte

gli elementi di A ?

OVVERO

E' possibile separare le operazioni effettuate dall'algoritmo di Gauss su

A e su b ?

Algoritmo di Gauss

…

"ciclo sui passi"

```
for k=1 to n-1
```

"calcolo moltiplicatori"

```
for i=k+1 to n
a_{i,k} := a_{i,k}/a_{k,k}
endfor
```

"trasformazione elementi del vettore attivo e della matrice attiva

```
for i=k+1 to n
    b_i := b_i - a_{i,k} b_k
    for j=k+1 to n
a_{i,j} := a_{i,j} - a_{i,k} a_{k,j}
    endfor
endfor
endfor
```

Operazioni su b?

Operazioni su A ?

Che cosa si ottiene separando le operazioni su A e su b?

ESEMPIO

Applichiamo l'algoritmo di Gauss solo alla matrice A del sistema

$$A\underline{x} = b$$

$$A = \begin{pmatrix} 2 & 4 & -2 \\ 1 & -1 & 5 \\ 4 & 1 & -2 \end{pmatrix} \qquad \underline{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \qquad \underline{b} = \begin{pmatrix} 6 \\ 0 \\ -2 \end{pmatrix}$$

Applichiamo l'alg. di Gauss

- **passo 1**:

  $m_{2,1}=1/2$, $m_{3,1}=2$;

$$A^{(1)} = \begin{pmatrix} 2 & 4 & -2 \\ 0 & -3 & 6 \\ 0 & -7 & 2 \end{pmatrix}$$

- **passo 2**:

  $m_{3,2}=7/3$

$$U = A^{(2)} = \begin{pmatrix} 2 & 4 & -2 \\ 0 & -3 & 6 \\ 0 & 0 & -12 \end{pmatrix}$$

Costruiamo la matrice:

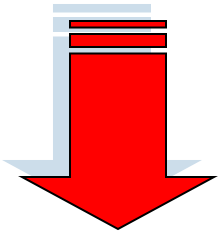
$$L = \begin{pmatrix} 1 & 0 & 0 \\ 1/2 & 1 & 0 \\ 2 & 7/3 & 1 \end{pmatrix}$$

Moltiplicatori del 1° passo

Moltiplicatore del 2° passo

Calcoliamo il prodotto riga per colonna L · U:

$$L \cdot U = \begin{pmatrix} 1 & 0 & 0 \\ 1/2 & 1 & 0 \\ 2 & 7/3 & 1 \end{pmatrix} \cdot \begin{pmatrix} 2 & 4 & -2 \\ 0 & -3 & 6 \\ 0 & 0 & -12 \end{pmatrix} = \; = \begin{pmatrix} 2 & 4 & -2 \\ 1 & -1 & 5 \\ 4 & 1 & -2 \end{pmatrix} = A$$



L · U = A

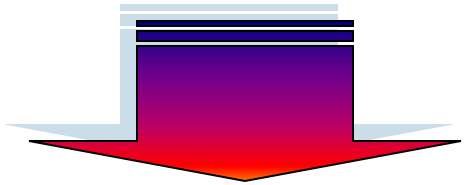
cioè

L'algoritmo di Gauss, applicato ad A, può essere visto come

Metodo di fattorizzazione

di A nel prodotto di due matrici:

$$A = L \cdot U$$



- **L triangolare inferiore**
- **U triangolare superiore**

In generale ….

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

$$U = A^{(n-1)} = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & \dots & a_{1,n} \\ 0 & a^{(1)}_{2,2} & a^{(1)}_{2,3} & \dots & a^{(1)}_{2,n} \\ 0 & 0 & a^{(2)}_{3,3} & \dots & a^{(2)}_{3,n} \\ \dots & \dots & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & a^{(n-1)}_{n,n} \end{pmatrix}$$

k-ma colonna =

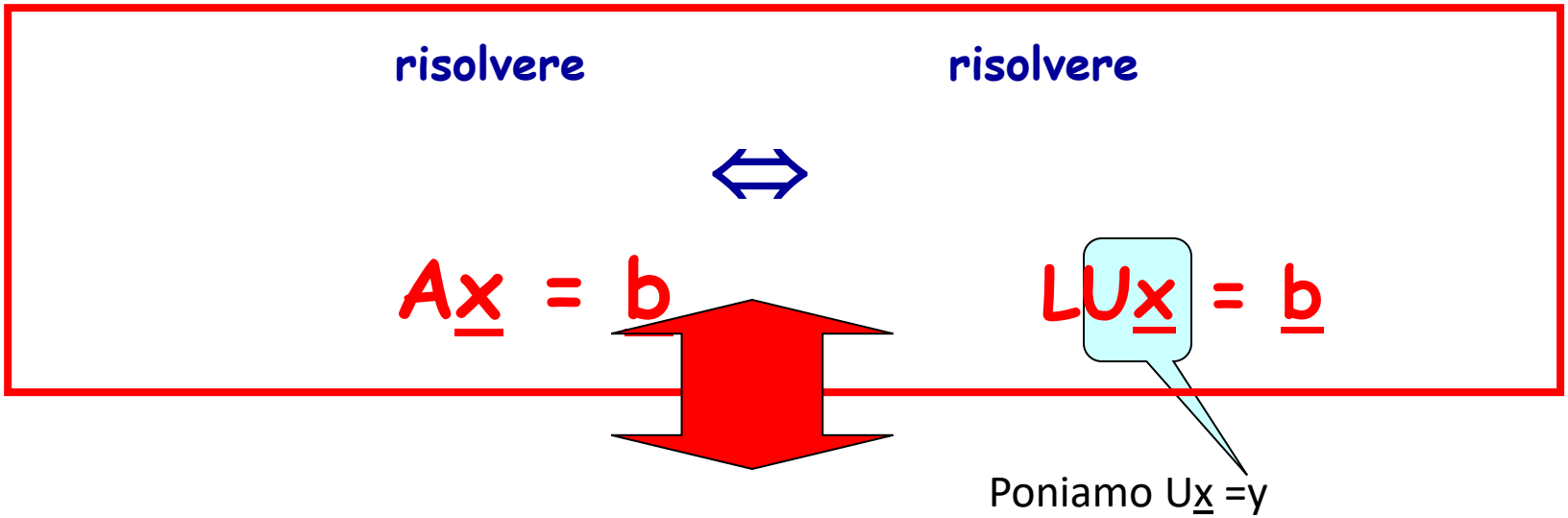
$m_{k,k} = 1$

$m_{i,k} = a_{i,k}/a_{k,k} \quad i=k+1,..,n$

moltiplicatori

Matrice ottenuta all'ultimo passo dell'algoritmo di Gauss

Calcolati i fattori L e U di A….



- bisogna risolvere
  (forward-substitution)

- calcolato y, bisogna risolvere Ux =y
  (back-substitution)

Secondo approccio

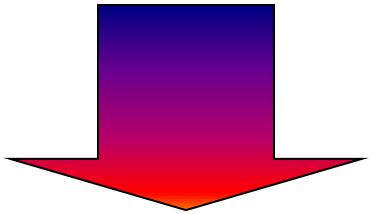
- **Calcolo fattorizzazione LU di A con l'algoritmo di Gauss (una sola volta)**
- **risoluzione di 2m sistemi triangolari :**

$$L\underline{y}_1 = \underline{b}_1 , \quad U\underline{x}_1 = \underline{y}_1;$$
$$L\underline{y}_2 = \underline{b}_2 , \quad U\underline{x}_2 = \underline{y}_2;$$
$$\dots \qquad \dots$$
$$L\underline{y}_m = \underline{b}_m , \quad U\underline{x}_m = \underline{y}_m;$$

Problema

Risolvere m sistemi del tipo:

$$A\underline{x}_1 = \underline{b}_1 \; ; \; A\underline{x}_2 = \underline{b}_2 \; ; \; \ldots \; ; \; A\underline{x}_m = \underline{b}_m$$



Secondo approccio

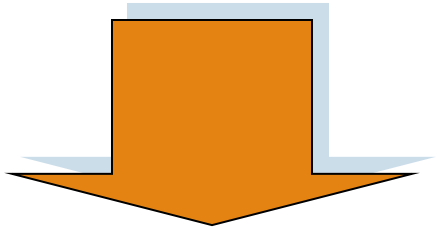
Calcolare la fattorizzazione LU di *A*

E risolvere i sistemi triangolari

Quanto si risparmia ?

COMPLESSITA' DI TEMPO

$$T_{LU}(n)+m(T_{forw}(n)+T_{back}(n)) = O(n^3/3)+2mO(n^2/2)=O(n^3/3+mn^2)$$



Il numero di operazioni si riduce di 1 ordine di grandezza!

Approccio piu' efficiente

PROBLEMA:

E' sempre possibile
fattorizzare una matrice *A*
nel prodotto
della matrice triangolare inferiore *L*
e della matrice triangolare superiore *U* ?

Teorema

Sia A una matrice di ordine n, e siano $A^{(k)}$ i suoi minori di ordine k, cioè:

$$A^{(k)} = (a_{ij}), \quad i,j=1,k$$

Se tali matrici sono non singolari allora esiste un'unica matrice triangolare inferiore L con elementi diagonali uguali ad 1 ed un'unica matrice triangolare superiore U tali che:

$$A = LU$$

inoltre

$$\det(A^{(k)}) = u_{11}\, u_{22} \ldots . u_{kk} \qquad k=1,n$$

Che cosa comporta il pivoting nella fattorizzazione LU ?

ESEMPIO

$$\underline{A}\underline{x} = \underline{b}$$

con:

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

Applichiamo l'algoritmo di Gauss con pivoting parziale per calcolare i fattori **L** e **U** di **A**

- **passo 1**:

  - **pivot in I colonna**:

$$\max_{1\le i\le 3} |a_{i,1}| = |a_{3,1}| = 4 \Rightarrow \text{pivot} = a_{3,1} = 4$$

$$A = \begin{pmatrix} 2 & 4 & -2 \\ 1 & -1 & 5 \\ \boxed{4} & 1 & -2 \end{pmatrix}$$

scambio 1° e 3° riga

$$\begin{pmatrix} 4 & 1 & -2 \\ 1 & -1 & 5 \\ 2 & 4 & -2 \end{pmatrix}$$

  - **trasformazione**:

$$\begin{pmatrix} 4 & 1 & -2 \\ 0 & -\frac{5}{4} & +\frac{11}{2} \\ 0 & \frac{7}{2} & -1 \end{pmatrix}$$

- **passo 2:**

  - **pivot in II colonna:**

$$\max_{2\le i\le 3} |a_{i,2}^{(1)}| = |a_{3,2}^{(1)}| = \frac{7}{2} \Rightarrow \text{pivot} = a_{3,2}^{(1)} = \frac{7}{2}$$

$$\begin{pmatrix} 4 & 1 & -2 \\ 0 & -\frac{5}{4} & +\frac{11}{2} \\ 0 & \boxed{\frac{7}{2}} & -1 \end{pmatrix}$$

scambio 2° e 3° riga

$$\begin{pmatrix} 4 & 1 & -2 \\ 0 & \boxed{\frac{7}{2}} & -1 \\ 0 & -\frac{5}{4} & \frac{11}{2} \end{pmatrix}$$

  - **trasformazione:**

$$\begin{pmatrix} 4 & 1 & -2 \\ 0 & \frac{7}{2} & -1 \\ 0 & 0 & \frac{36}{7} \end{pmatrix}$$

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

$$\tilde{L} = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ \frac{1}{4} & \frac{-5}{14} & 1 \end{pmatrix}$$

Matrice dei moltiplicatori

$$\tilde{U} = \begin{pmatrix} 4 & 1 & -2 \\ 0 & \frac{7}{2} & -1 \\ 0 & 0 & \frac{36}{7} \end{pmatrix}$$

Matrice finale

$$A' = \tilde{L}\tilde{U} = \begin{pmatrix} 4 & 1 & -2 \\ 2 & 4 & -2 \\ 1 & -1 & 5 \end{pmatrix}$$

A'=$\tilde{L}\tilde{U}$ coincide con A a meno di uno scambio di righe

……“a meno di uno scambio di righe “ ?

Qual è lo scambio di righe che Trasforma A in A’ ?

applichiamo ad *A* gli scambi effettuati col pivoting

- **passo 1**: scambio della 1° e 3° riga:

$$A = \begin{pmatrix} 2 & 4 & -2 \\ 1 & -1 & 5 \\ 4 & 1 & -2 \end{pmatrix} \qquad \begin{pmatrix} 4 & 1 & -2 \\ 1 & -1 & 5 \\ 2 & 4 & -2 \end{pmatrix}$$

- **passo 2**: scambio della 2° e 3° riga:

$$\begin{pmatrix} 4 & 1 & -2 \\ 1 & -1 & 5 \\ 2 & 4 & -2 \end{pmatrix} \qquad \begin{pmatrix} 4 & 1 & -2 \\ 2 & 4 & -2 \\ 1 & -1 & 5 \end{pmatrix} = A'$$

Che rapporto c'è tra A e A' ?

E' possibile ottenere A' moltiplicando A per un opportuna matrice P:

$$PA=A'$$

?

Posto $PA=A'$

Determiniamo P in modo tale che P debba scambiare le righe di A, ma non debba modificare gli elementi di A

Poiché $IA=A$

cioè

I non modifica A

IDEA: cerchiamo di costruire P da I

Consideriamo la matrice identica:

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

Durante l'esecuzione dell'algoritmo di eliminazione di Gauss con pivoting parziale, effettuiamo su I gli stessi scambi eseguiti su A

- **passo 1**: scambio 1° e 3° riga:

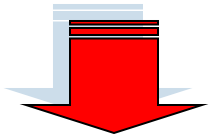
$$I = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

- **passo 2**: scambio 2° e 3° riga:

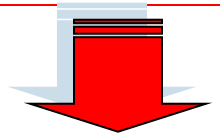
$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} = P$$

Calcolando il prodotto **PA** ….

$$PA = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 2 & 4 & -2 \\ 1 & -1 & 5 \\ 4 & 1 & -2 \end{pmatrix} = \begin{pmatrix} 4 & 1 & -2 \\ 2 & -4 & -2 \\ 1 & -1 & 5 \end{pmatrix} = A'$$



L'effetto del prodotto **PA** e' quello di modificare l'ordine delle righe di **A**



P : matrice di permutazione

In conclusione ..

L'algoritmo di eliminazione di Gauss con pivoting parziale costruisce le matrici $\tilde{L}$ e $\tilde{U}$ e la matrice di permutazione P tali che

$$A' = PA = \tilde{L}\tilde{U}$$

dove la matrice P conserva le informazioni sugli scambi effettuati

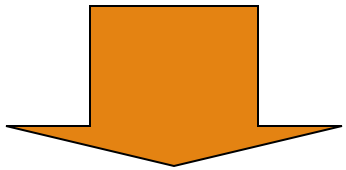
Come si risolve il sistema

$Ax=b$

una volta fattorizzata la matrice A
con pivoting parziale
(e quindi una volta ottenuta anche la P) ?

$$Ax = b \quad \Rightarrow \quad PAx = Pb = b'$$

$$PA = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 2 & 4 & -2 \\ 1 & -1 & 5 \\ 4 & 1 & -2 \end{pmatrix} = \begin{pmatrix} 4 & 1 & -2 \\ 2 & -4 & -2 \\ 1 & -1 & 5 \end{pmatrix}$$



P b b'

$$\begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 6 \\ 0 \\ -2 \end{pmatrix} = \begin{pmatrix} -2 \\ 6 \\ 0 \end{pmatrix}$$

PA x Pb

$$\begin{pmatrix} 4 & 1 & -2 \\ 2 & -4 & -2 \\ 1 & -1 & 5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} -2 \\ 6 \\ 0 \end{pmatrix}$$

$$Ax = b \quad \Rightarrow \quad PAx = Pb = b'$$

In effetti l'algoritmo di Gauss con pivoting parziale risolve il sistema PAx = Pb

1. Fattorizzazione $\tilde{L}\tilde{U}$ di PA
2. calcolo di b'=Pb

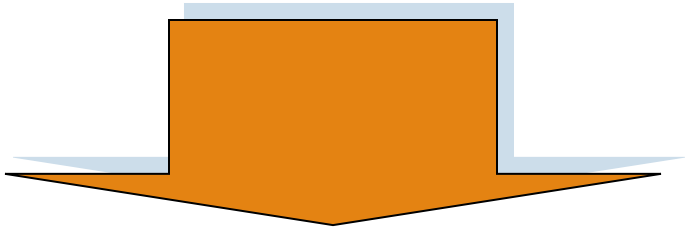
$\tilde{L}\tilde{U}$ x=b'

3. risoluzione di $\tilde{L}$ y = b' (forw sub)

risoluzione di $\tilde{U}$ x = y

Ma, è proprio necessario costruire P?

P memorizza gli scambi tra le righe di A



E' sufficiente costruire un vettore che conservi le informazioni sugli scambi effettuati

Come costruire tale vettore ?

- **I passo dell'algoritmo di Gauss**

$$ipiv = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \rightarrow \begin{pmatrix} 2 & 4 & -2 \\ 1 & -1 & 5 \\ 4 & 1 & -2 \end{pmatrix}$$

I riga
II riga
III riga

ipiv contiene gli
indici di riga
in ordine naturale
ipiv(i)=i

- I passo dell'algoritmo di Gauss

- scambio 1° e 3° riga di *A*

- dopo lo scambio

$$\begin{pmatrix} 2 & 4 & -2 \\ 1 & -1 & 5 \\ 4 & 1 & -2 \end{pmatrix} \qquad ipiv = \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix}$$

Si scambia *SOLO* il valore di ipiv(1)=1 con quello di ipiv(3)=3

ipiv(1)←ipiv(3)=3

ipiv(3)←ipiv(1)=1

- II passo dell'algoritmo di Gauss:

- scambio 2° e 3 ° riga di A

- dopo lo scambio

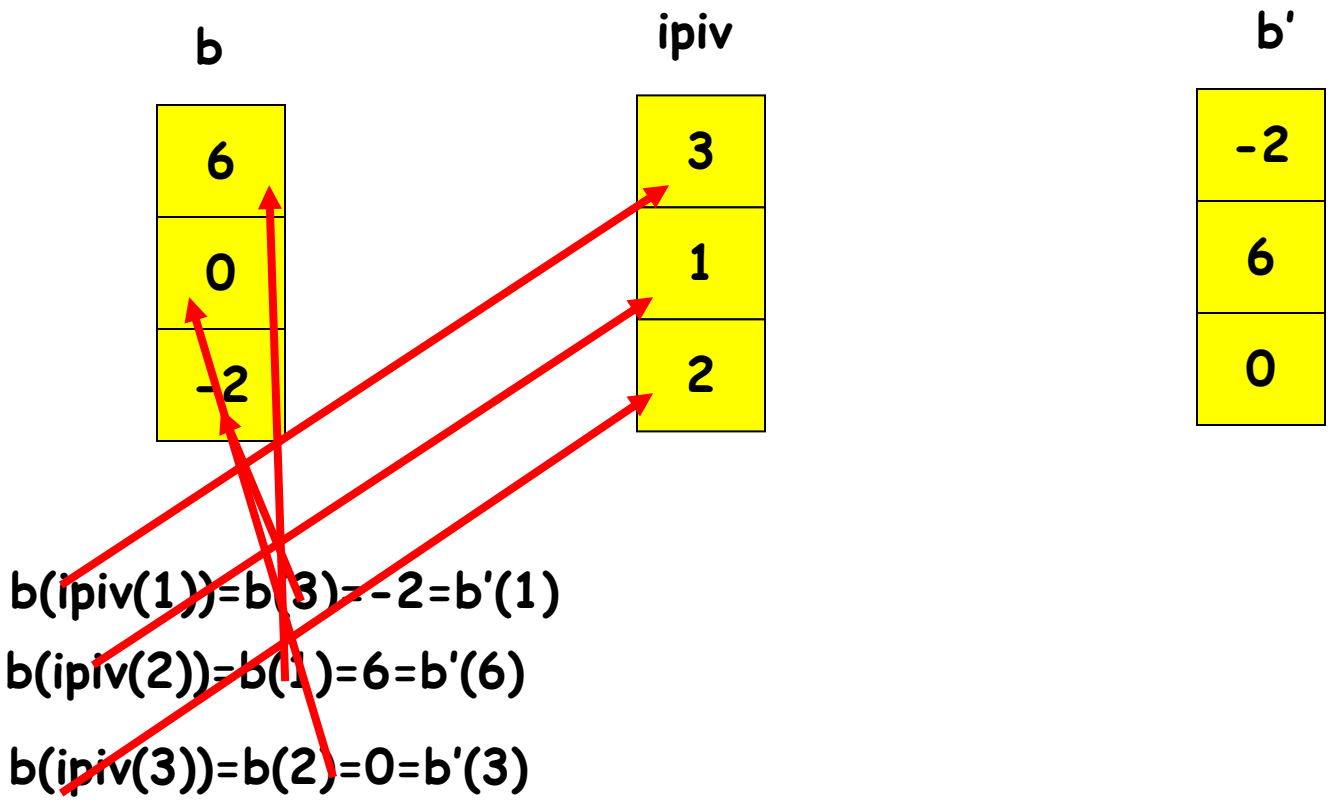
$$\begin{pmatrix} 2 & 4 & -2 \\ 1 & -1 & 5 \\ 4 & 1 & -2 \end{pmatrix} \qquad ipiv = \begin{pmatrix} 3 \\ 1 \\ 2 \end{pmatrix}$$

Si scambia SOLO il valore di ipiv(2)=2 con quello di ipiv(3)=1

ipiv(2)←ipiv(3)=1

ipiv(3)←ipiv(2)=2

Come calcolare b’ utilizzando b e ipiv?



$$b(ipiv(i)) = b'(i) \qquad i=1,2,3$$

Come calcolare A' utilizzando A e ipiv?

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

ipiv

3
1
2

$$\text{A'} = \begin{pmatrix} 4 & 1 & -2 \\ 2 & 4 & -2 \\ 1 & -1 & 5 \end{pmatrix}$$

a(ipiv(1),j)=a(3,j)

I riga di PA → **III riga di A**

a(ipiv(2),j)=a(1,j)

II riga di PA → **I riga di A**

a(ipiv(3),j)=a(2,j)

III riga di PA → **II riga di A**

In conclusione

Nell'algoritmo di fattorizzazione LU con pivoting parziale l'utilizzo di ipiv consente di

- **Non costruire la Matrice P**
- **Non effettuare fisicamente gli scambi di righe di A (scambio virtuale).**

In conclusione

Nella forward-substitution per la risoluzione di y = Pb

l'utilizzo di ipiv consente di:

$\tilde{L}$

- **conoscere gli scambi da effettuare su b**
- **non effettuare fisicamente gli scambi di(scambio virtuale).**

Algoritmo di Fattorizzazione LU

```
…
"ciclo sui passi"
for k=1 to n-1
    "cicli per ottenere la matrice L ed U"
    for i=k+1 to n
     "calcolo elementi di L "
      a_{i,k} := a_{i,k}/a_{k,k}
     "calcolo elementi di U "
       for j=k+1 to n
     a_{i,j} := a_{i,j}-a_{i,k}a_{k,j}
       endfor
    endfor
endfor
…
```

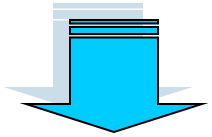
$T_{LU}(n) = O(n^3/3)$

Complessità di spazio

- **elementi di L (tranne elementi diagonali =1) memorizzati sui corrispondenti elementi di A;**
- **elementi di U memorizzati sui corrispondenti elementi di A;**

$$A \Leftarrow \begin{pmatrix} u_{1,1} & u_{1,2} & u_{1,3} & \dots & u_{1,n} \\ l_{2,1} & u_{2,2} & u_{2,3} & \dots & u_{2,n} \\ l_{3,1} & l_{3,2} & u_{3,3} & \dots & u_{3,n} \\ \dots & \dots & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ l_{n,1} & l_{n,2} & \dots & l_{n,n-1} & u_{n,n} \end{pmatrix}$$

Algoritmo 'in place'



$S_{LU}(n) = O(n^2)$

Esempio:

$$A = \begin{pmatrix} -4 & -1 & 0 & 0 \\ -1 & 4 & -1 & 0 \\ 0 & -1 & 4 & -1 \\ 0 & 0 & -1 & 4 \end{pmatrix}$$

Passo 1: Pivot=4=a11

NON è necessario effettuare scambi di righe!

$$A^{(1)} = \begin{pmatrix} -4 & -1 & 0 & 0 \\ 0 & 15/4 & -1 & 0 \\ 0 & -1 & 4 & -1 \\ 0 & 0 & -1 & 4 \end{pmatrix}$$

Passo 2: pivot = 15/4 = a $22^{(1)}$

NON è necessario effettuare scambi di righe:

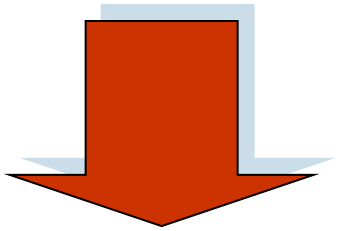
$$A^{(1)} = \begin{pmatrix} -4 & -1 & 0 & 0 \\ 0 & 15/4 & -1 & 0 \\ 0 & -1 & 4 & -1 \\ 0 & 0 & -1 & 4 \end{pmatrix} \Rightarrow A^{(2)} = \begin{pmatrix} -4 & -1 & 0 & 0 \\ 0 & 15/4 & -1 & 0 \\ 0 & 0 & 56/15 & -1 \\ 0 & 0 & -1 & 4 \end{pmatrix}$$

Passo 3: pivot = 56/15 =a $33^{(2)}$

NON è necessario effettuare scambi di righe:

$$A^{(3)} = \begin{pmatrix} -4 & -1 & 0 & 0 \\ 0 & 15/4 & -1 & 0 \\ 0 & 0 & 56/15 & -1 \\ 0 & 0 & 0 & 209/56 \end{pmatrix}$$

Ad ogni passo k, l'elemento massimo (in modulo) della colonna k, si trova già sulla diagonale



Il pivoting parziale,

in questo caso è INUTILE!

Per quale classe di matrici simmetriche si riesce a garantire che, ad ogni passo, il pivot si trovi già sulla diagonale ?

Nell'esempio …

$$A = \begin{pmatrix} -4 & -1 & 0 & 0 \\ -1 & 4 & -1 & 0 \\ 0 & -1 & 4 & -1 \\ 0 & 0 & -1 & 4 \end{pmatrix}$$

4 =|-4|= |a11| >|-1| +0+0=1

A è a diagonale dominante

$$|a_{i,i}| \geq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{i,j}|$$

$$A^{(1)} = \begin{pmatrix} -4 & -1 & 0 & 0 \\ 0 & 15/4 & -1 & 0 \\ 0 & -1 & 4 & -1 \\ 0 & 0 & -1 & 4 \end{pmatrix}$$

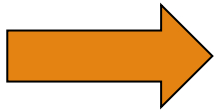
15/4 = a22 (1) >|-1| +0+0=1

$A^{(1)}$ è a diagonale dominante

$$|a^{(1)}{}_{i,i}| \geq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a^{(1)}{}_{i,j}|$$

Nell'esempio …

$$A^{(2)} = \begin{pmatrix} -4 & -1 & 0 & 0 \\ 0 & 15/4 & -1 & 0 \\ 0 & 0 & 56/15 & -1 \\ 0 & 0 & -1 & 4 \end{pmatrix}$$



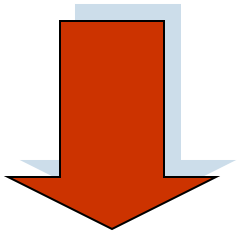
56/15 = a33 (2) >|-1|=1

$A^{(2)}$ è a diagonale dominante $\left|a^{(2)}_{i,i}\right| \geq \sum_{\substack{j=1 \\ j\neq i}}^{n} \left|a^{(2)}_{i,j}\right|$

Quindi …

A è a diagonale dominante

$$|a_{i,i}| \geq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{i,j}|$$



Al generico passo k
Le sottomatrici attive sono
A diagonale dominante

Inoltre …

A simmetrica e a diagonale dominante

Al passo k,
Il massimo della riga
coincide con il massimo della colonna
e si trova sulla diagonale

$$\max_{k\le j\le n}\left|a_{i,j}^{(k-1)}\right| = \max_{k\le i\le n}\left|a_{i,j}^{(k-1)}\right| = \left|a_{k,k}^{(k-1)}\right|$$

simmetria

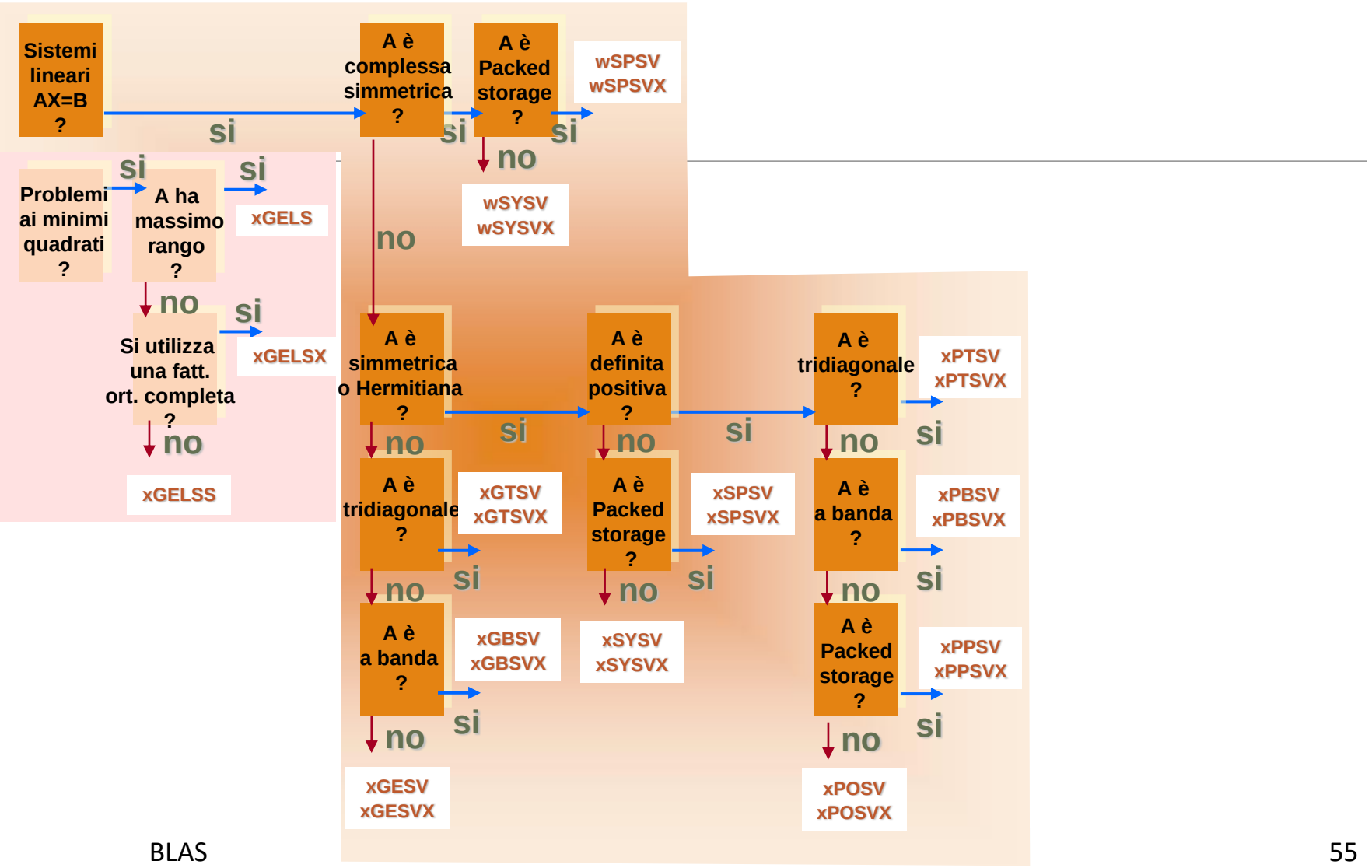
dominanza diagonale

Il pivoting parziale è INUTILE !

Esistono molti tipi di fattorizzazioni di una assegnata matrice.
In generale la scelta dell' algoritmo ottimale dipende da vari fattori, Come

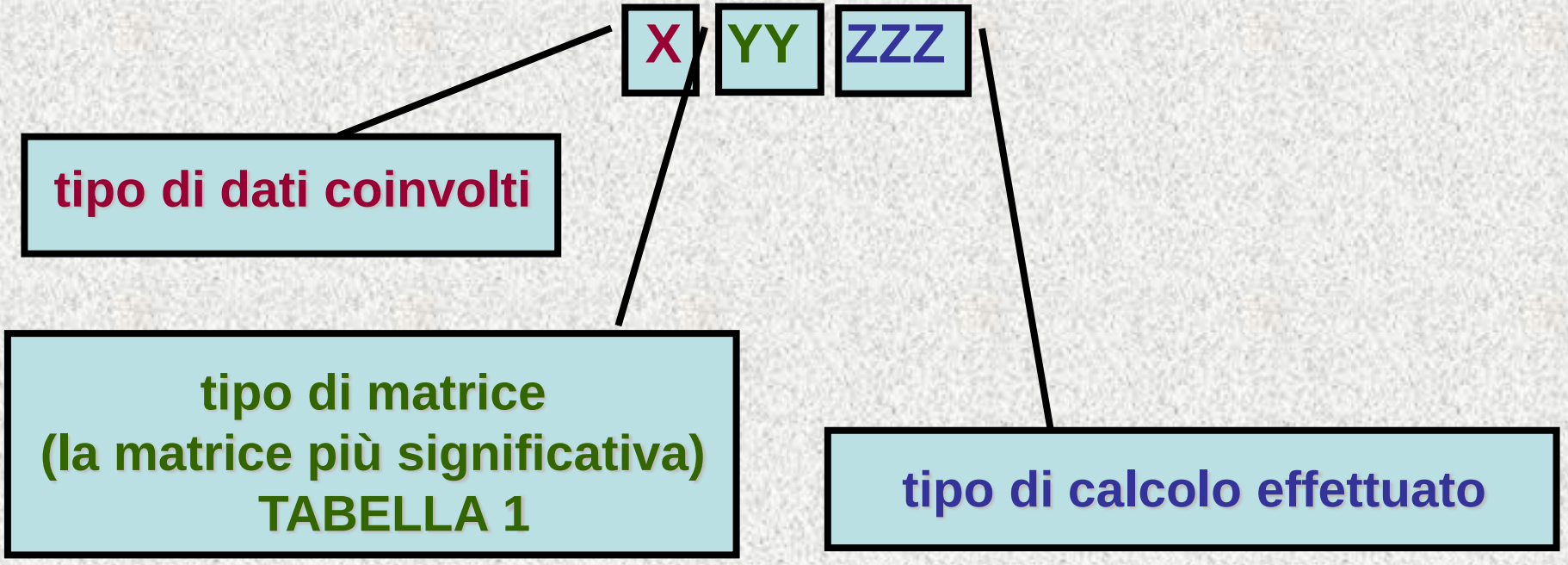
- la struttura della matrice (che può influenzare la memorizzazione della matrice)
- le proprietà della matrice (che può influenzare l'algoritmo)
- la conoscenza del tipo di fattorizzazione derivante
- la conoscenza a priori delle caratteristiche della matrice o invece la necessità di verificarle

albero delle routine driver di LAPACK



Schema di denominazione delle **Routines** di **LAPACK**

Tutte le driver e computational routines hanno nome



tipo di dati coinvolti (X)

S real
D double precision
C complex
Z complex*16 o double complex

le routines lavorano sia con
dati reali che con dati complessi

infatti

LAPACK fornisce lo stesso “range di funzionalità”
sia per dati reali che per dati complessi

tipo di matrice (YY)
tabella 1

BD	bidiagonal
GB	general band
GE	general (i.e., unsymmetric, in some cases rectangular)
GG	general matrices, generalized problem (i.e., a pair of general matrices) (*not implemented in Release 1.0*)
GT	general tridiagonal
HB	(complex) Hermitian band
HE	(complex) Hermitian
HG	upper Hessenberg matrix, generalized problem (i.e a Hessenberg and a triangular matrix) (*not implemented in Release 1.0*)
HP	(complex) Hermitian, packed storage
HS	upper Hessenberg
OP	(real) orthogonal, packed storage
OR	(real) orthogonal

tipo di matrice (YY)
tabella 1 (segue)

PB	symmetric or Hermitian positive definite band
PO	symmetric or Hermitian positive definite
PP	symmetric or Hermitian positive definite, packed storage
PT	symmetric or Hermitian positive definite tridiagonal
SB	(real) symmetric band
SP	symmetric, packed storage
ST	(real) symmetric tridiagonal
SY	symmetric
TB	triangular band
TG	triangular matrices, generalized problem (i.e., a pair of triangular matrices) (*not implemented in Release 1.0*)
TP	triangular, packed storage
TR	triangular (or in some cases quasi-triangular)
TZ	trapezoidal
UN	(complex) unitary
UP	(complex) unitary, packed storage

tipo di calcolo effettuato (ZZZ)

(dato il tipo di dati **x** ed il tipo di matrice **yy)**

DRIVER ROUTINES

xyySVX	risolve un sistema lineare
xyySV	risolve un sistema lineare
xyyLS	risolve minimi quadrati utilizzando fattorizzazioni QR o LQ
xyyLSS	risolve minimi quadrati utilizzando SVD
xyyEVX	determina autovalori per matrici simmetriche

...

tipo di calcolo effettuato (ZZZ)

dato il tipo di dati **x** ed il tipo di matrice **yy**

COMPUTATIONAL ROUTINES

xyyTRF esegue la fattorizzazione LU

xyyTRS utilizza la fattorizzazione per la Forward (Back)-Substitution

xyyCON stima il reciproco dell'indice di condizionamento (Metodo di Hager)

xyyRFS stima l'errore nella soluzione calcolata (ritornata da **xyyTRF**)

xyyTRI utilizza la fattorizzazione (ritornata da **xyyTRF**) per calcolare A^{-1}

xyyEQU calcola il fattore di scala per equilibrare A
(lo scaling non viene effettuato dalle **xyyEQU** ma ad esempio dalle auxiliary routines **xLAQyy**.)

...

Documentazione del software

interfaccia utente-software

- ✓ **Istruzioni d'uso**
- ✓ **Informazioni sull'organizzazione interna**
- ✓ **Trasferimento di esperienze**

Interna

Linee di commenti esplicativi di sequenze

Esterna

Manuale d'uso,help on line

Documentazione esterna

Scopo

breve descrizione dei problemi risolubili, algoritmo usato, raccomandazioni sull'uso

Specifiche

intestazione della procedura

Lista parametri input/output

tipo,dimensione e descrizione

Diagnostica

descrizione delle situazioni di errore previste

Complessità e accuratezza

eventuali informazioni su complessità di tempo e spazio accuratezza del risultato

Esempio d'uso

esempio di un semplice programma chiamante con elenco dei dati di input e dei risultati

Esempio : prodotto scalare di due vettori in Matlab

file prodsca.m

```
%scopo:calcolo del prodotto scalare di due vettori
%parametri di input: x,y vettori di uguale lunghezza
%parametri di output:s scalare,prodotto scalare di x ed y
%il programma si arresta se x ed y hanno dimensione diversa
%autore Carlo Rossi
function s=prodsca(x,y)
m=length(x);
n=length(y);
%controllo dimensioni
if ne(m,n)
   error('le dimensioni non coincidono')
else
   %calcolo  risultato
   s=0;
   for i=1:m
      s=s+x(i)*y(i);
   end
end
```

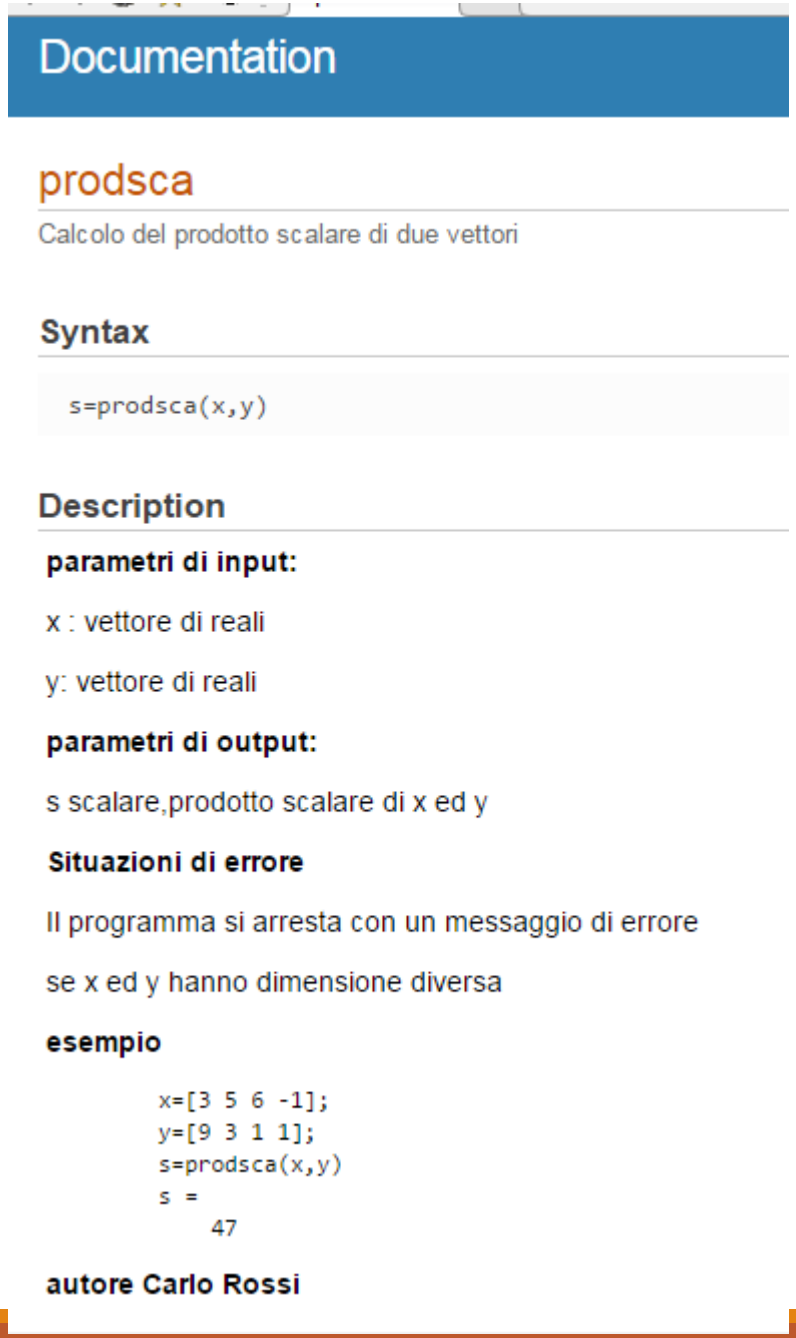
osservazione

non è un programma ottimale in Matlab serve solo come esempio!

Per generare la documentazione esterna posso generare una live function

Esempio da prodsca

>>doc prodsca



Documentation

prodsca

Calcolo del prodotto scalare di due vettori

Syntax

```
s=prodsca(x,y)
```

Description

parametri di input:

x : vettore di reali

y: vettore di reali

parametri di output:

s scalare,prodotto scalare di x ed y

Situazioni di errore

Il programma si arresta con un messaggio di errore

se x ed y hanno dimensione diversa

esempio

```
x=[3 5 6 -1];
y=[9 3 1 1];
s=prodsca(x,y)
s =
    47
```

autore Carlo Rossi

pdf

Calcolo del prodotto scalare di due vettori

parametri di input:

x : vettore di reali

y: vettore di reali

parametri di output:

s scalare,prodotto scalare di x ed y

Situazioni di errore

Il programma si arresta con un messaggio di errore

se x ed y hanno dimensione diversa

esempio

```
x=[3 5 6 -1];
y=[9 3 1 1];
s=prodsca(x,y)
s =
    47
```

autore Carlo Rossi

```
function s=prodsca(x,y)
m=length(x);
n=length(y);
%controllo dimensioni
if ne(m,n)
    error('le dimensioni non coincidono')
else
    %calcolo  risultato
    s=0;
    for i=1:m
        s=s+x(i)*y(i);
    end
end
```

Per effettuare i test di accuratezza e robustezza scrivo un live script e lo salvo in pdf

Test di accuratezza

```
format long
x=rand(1,200);
y=3.5*rand(200,1);
s=prodsca(x,y);
```

confronto con la routine del matlab,

il confronto si fa sull'errore relativo

```
q=x*y;
err=abs(q-s)/abs(q)
```

```
err =
     1.662499624741376e-16
```

test di robustezza

```
x=[5 -2 1.6 ];
y=[2 2.6 3.2 4.5];
s=prodsca(x,y)
```

```
Error using prodsca (line 19)
le dimensioni non coincidono
```

```
x=[5 -2 1.6 7];
y=[2 2.6 3.2 ];
s=prodsca(x,y)
```

```
Error using prodsca (line 19)
le dimensioni non coincidono
```

In conlusione

Scrivere un software non è scrivere un "programma che funziona",

ma

scrivere un programma

- **Robusto**
- **Efficiente**
- **Accurato**

Testato in modo da garantire questi attributi e corredato da una documentazione tale da garantire

- **una facile comprensione e modifica del software**
- **un semplice utilizzo da parte dell'utente**