Introduzione all'ambiente

MATLAB©

Aritmetica di macchina.

Costrutti di programmazione.

Algebra lineare.

Lezione 2 (19 febbraio 2003)

Lucia Gastaldi

Formato di **memorizzazione** dei numeri

- **Singola** (o semplice) precisione, 4 Bytes
- **Doppia** precisione, 8 Bytes

$\boxed{\text{Come vengono utilizzati questi Bytes?}}$

Si considera la forma esponenziale di un numero reale:

$$x = 123456.789 = (-1)^0 1.23456789 \cdot 10^5 = (-1)^s m \cdot \beta^e$$

$s = 0, 1$; m mantissa; β base (es: 2,10); e esponente

1	$8bits$	$23bits$	$4Bytes$
s	e	m	

1	$11bits$	$52bits$	$8Bytes$
s	e	m	

Insieme dei numeri **rappresentabili**

$$x = (-1)^s m \cdot \beta^e = (-1)^s (.a_1 a_2 \cdots a_t) \cdot \beta^e = (-1)^s \beta^e \sum_{i=1}^{t} a_i \beta^{-i}$$

con le seguenti restrizioni:

$L \le e \le U$ tipicamente $L < 0$ e $U > 0$
$\beta \ge 2$
$0 \le a_i \le \beta - 1$ per $i = 1, \cdots, t$

Codifiche particolari

valore	esponente	mantissa
± 0	$L-1$	0
`Inf`	$U+1$	0
`NaN`	$U+1$	$\neq 0$

Lo standard IEC (International Electronical Commission) prevede:

$$\beta = 2, \quad t = 53, \quad L = -1021, \quad U = 1024$$

$$.1000 \cdot 10^1 \quad .0100 \cdot 10^2 \quad .0010 \cdot 10^3 \quad .0001 \cdot 10^4$$

sono tutte rappresentazioni di 1.

Per avere **unicità della rappresentazione** si richiede:

$$a_1 \neq 0; \quad m \geq \beta^{t-1} \quad \Rightarrow \beta^{-1} \leq m < 1$$

La rappresentazione di x si dice **normalizzata**.

$$x_{min} = \beta^{L-1} \leq |x| \leq \beta^U(\beta^t - 1) = x_{max}$$

Si può rinunciare alla restrizione che $a_1 \neq 0$ nel caso $e = L$ e si ha la rappresentazione **floating-point denormalizzata**.

$$\beta^{-t} \leq m < 1 \quad \text{e} \quad x_{min} = \beta^{L-t}$$

Arrotondamento di un numero reale.

$$x = (-1)^s \beta^e \cdot \sum_{i=1}^{\infty} a_i \beta^{-i} \qquad \textbf{numero reale}$$

Se $L \leq e \leq U$ allora l'arrotondamento di x è definito da

$$fl(x) = \begin{cases} (-1)^s \beta^e \cdot \sum_{i=1}^{t} a_i \beta^{-i} & \text{se } a_{t+1} < \frac{\beta}{2} \\ (-1)^s \beta^e \cdot \left(\sum_{i=1}^{t} a_i \beta^{-i} + \beta^{-t}\right) & \text{se } a_{t+1} \geq \frac{\beta}{2} \end{cases}$$

$$\frac{|x - fl(x)|}{|x|} \leq \frac{\beta^{-t}\beta^e}{2|m|\beta^e} \leq \frac{1}{2}\beta^{1-t} = \mathtt{u}$$

`eps`	*epsilon macchina*	`eps`$+1 > 1$	`eps`$= \beta^{1-t}$
`u`	*precisione di macchina*		`u=eps/2`

Aritmetica IEC559

Lo standard IEC559 provvede a definire le operazioni sull'insieme dei numeri di macchina in modo che ogni operazione produce un risultato all'interno del sistema stesso.

Attenzione Le operazioni di macchina non godono della proprietà associativa.

Sia $\beta = 10$, $t = 8$ e $\quad a = 0.23371258 \cdot 10^{-4}$

$$b = 0.33678429 \cdot 10^2 \quad c = -0.33677811 \cdot 10^2$$

allora si ha:

$$\begin{aligned} (a+b)+c &= 0.33678452 \cdot 10^2 - 0.33677811 \cdot 10^2 \\ &= 0.64100000 \cdot 10^{-3} \\ a+(b+c) &= 0.23371258 \cdot 10^{-4} + 0.61800000 \cdot 10^{-3} \\ &= 0.61437126 \cdot 10^{-3} \end{aligned}$$

Risultati per alcune operazioni eccezionali

eccezione	esempi	risultato
operazione non valida	0/0, $0 \cdot \infty$	`NaN`
overflow		`Inf`
divisone per zero	1/0	`Inf`
underflow		numeri sottonormali

Cifra di guardia

Problema: Sia $\beta = 10$ e $t = 2$. Eseguire la sottrazione tra 1 e 0.99.

$$\begin{array}{lcl} 10^1 \cdot 0.1 & & 10^1 \cdot 0.10 \\ 10^0 \cdot 0.99 & \Rightarrow & \underline{10^1 \cdot 0.09} \\ & & 10^1 \cdot 0.01 \;\; \rightarrow \boxed{10^0 \cdot 0.10} \end{array}$$

Con la cifra di guardia si ottiene:

$$\begin{array}{lcl} 10^1 \cdot 0.1 & & 10^1 \cdot 0.10 \\ 10^0 \cdot 0.99 & \Rightarrow & \underline{10^1 \cdot 0.09\boxed{9}} \\ & & 10^1 \cdot 0.00\boxed{1} \;\; \rightarrow \boxed{10^0 \cdot 0.01} \end{array}$$

Problema 1: scrivere un **M-file** per calcolare le radici dell'equazione di secondo grado:

$$x^2 - 2bx + c = 0.$$

1) Siano $b = \frac{10^\alpha+1}{2}$ e $c = 10^\alpha$. Risolvere l'equazione con $\alpha = 2, 7, 12, 17$.

2) Trovare una motivazione per i risultati ottenuti.

Formula **stabile** per il calcolo delle radici:

Se $b \geq 0$ $\quad x1 = b + \sqrt{b^2 - c}$, $\quad x2 = c/x1$

Se $b < 0$ $\quad x1 = b - \sqrt{b^2 - c}$, $\quad x2 = c/x1$

if,else,elseif

`if` valuta una *espressione logica* ed esegue un gruppo di istruzioni a seconda del valore dell'espressione logica.

```
if espressione logica
    istruzioni
end
```

```
if espressione logica
    istruzioni
elseif espressione logica
    istruzioni
else
    istruzioni
end
```

Operatori di relazione

Operatore	Descrizione
<	Minore di
<=	Minore di o uguale a
>	Maggiore di
>=	Maggiore di o uguale a
==	Uguale a
~=	Diverso da

Esempio

```
>> A=[2 7 6; 9 0 7; 6 3 2];
>> B=[8 7 2; 8 1 7; 1 2 1];
>> A==B

ans =

     0     1     0
     0     0     1
     0     0     0
```

Gli elementi in cui la relazione è **vera** hanno valore 1.
Gli elementi in cui la relazione è **falsa** hanno valore 0.

Operatori logici

<table>
<tr><th>Operatore</th><th>Descrizione</th></tr>
<tr><td>&</td><td>e</td></tr>
<tr><td>|</td><td>o</td></tr>
<tr><td>~</td><td>non</td></tr>
</table>

- Un'espressione con l'operatore & è vera se sono veri entrambi gli operandi. In termini numerici, l'espressione è vera se entrambi gli operandi sono diversi da zero.

```
>> u=[1 2 0 2 1 0];
>> v=[0 1 1 0 2 0];
>> u&v

ans =

     0     1     0     0     1     0
```


- Un'espressione con l'operatore | è vera se almeno uno degli operandi è vero. In termini numerici, l'espressione è falsa se entrambi gli operandi sono uguali a zero.

```
>> u|v

ans =

     1     1     1     1     1     0
```

- Un'espressione in cui si usa l'operatore ~, si nega l'operando. In termini numerici, ogni elemento $\neq 0$ diventa 0 e ogni elemento 0 diventa 1.

```
>> ~u

ans =

     0     0     1     0     0     1
```

Precedenze fra gli operatori

1 Parentesi.

2 Trasposto, potenza.

3 Segno, negazione logica.

4 Moltiplicazione, divisione.

5 Somma e sottrazione.

6 Operatore :.

7 Operatori di relazione.

8 Operatore logico **e**.

9 Operatore logico **o**.

Soluzione del Problema 1

```
c=10.^alfa;
b=(c+1)/2;
if b>=0.
x1=b+sqrt(b*b-c)
x2=c/x1
else
x1=b-sqrt(b*b-c)
x2=c/x1
end
```


Problema 2: scrivere un **M-file** che calcoli il valore di un polinomio nei punti x assegnati. I coefficienti del polinomio sono contenuti in un vettore `a` di dimensione $n+1$ essendo n il grado del polinomio. Il polinomio ha quindi la seguente espressione:

$$p(x) = \sum_{i=0}^{n} a_i x^i$$

1) Sia x il vettore che contiene i punti dell'intervallo $[0.995, 1.005]$ equispaziati a distanza $1.e-5$. Fare il grafico del polinomio:

$$p(x) = x^6 - 6x^5 + 15x^4 - 20x^3 + 15x^2 - 6x + 1.$$

2) Osservato che $p(x) = (x-1)^6$, confrontare il grafico precedente con quello che si ottiene usando la formula compatta.

for

Il ciclo `for` esegue un gruppo di istruzioni un numero fissato di volte.

`for` *indice* = *inizio* : *incremento* : *fine*
  istruzioni
`end`

Incremento di default: 1.

Se *incremento*>0, allora il ciclo termina quando la variabile *indice* è maggiore di *fine*.

Se *incremento*<0, allora il ciclo termina quando la variabile *indice* è minore di *fine*.

Osservazione

Si ricorda che

$$\sum_{i=1}^{n} b_i = b_1 + b_2 + b_3 + \cdots + b_n$$

L'operazione `+` è un'operazione binaria, cioè opera tra due addendi. Quindi si somma prima $b_1 + b_2$ al risultato si aggiunge b_3 e così via. Indicata con S una variabile di accumulo, per realizzare la sommatoria si deve usare un ciclo `for` nel modo seguente:

```
S=0.          inizializzazione della variabile di accumulo
for i=1:n
   S=S+b(i);     b è un array che contiene gli addendi b_i
end
```


Soluzione del Problema 2

```
close
a=[1,-6,15,-20,15,-6,1];
x=[0.995:1.e-5:1.005];
p=a(1);
for i=2:length(a)
p=p+a(i).*x.^i;
end
plot(x,p)
p1=(x-1).^6;
hold on
plot(x,p1,'r')
```


while

Il ciclo `while` esegue un gruppo di istruzioni fintanto che l'espressione di controllo rimane vera.

`while` *espressione di controllo*
    istruzioni
`end`

L'espressione di controllo è una qualunque espressione logica.

Algebra lineare

Problema 4: Date due matrici A, $m \times n$, e b, $m \times p$, esiste una *unica* matrice X tale che

$$AX = b \qquad \text{oppure} \qquad XA = b?$$

Esempio L'equazione $7x = 21$ ha un'unica soluzione?

Per trovare la risposta si divide:

$$x = 21/7 = 3$$

OSS *Non* abbiamo calcolato prima il reciproco di 7 e poi lo abbiamo moltiplicato per 21!

Lo stesso principio vale quando si deve risolvere un sistema lineare.

Tre casi possibili:

- Sistemi quadrati, $m = n$.
- Sistemi sovradeterminati, $m > n$.
- Sistemi sottodeterminati, $m < n$.

La risoluzione del sistema lineare si ottiene usando i simboli di divisione: *slash* / e *backslash* \.

$X = A\backslash b$ Indica la soluzione del sistema $AX = b$.
$X = b/A$ Indica la soluzione del sistema $XA = b$.

L'operatore *backslash* usa algoritmi differenti per trattare diversi tipi di matrici:

- Permutazioni di matrici triangolari.
- Matrici simmetriche e definite positive.
- Matrici quadrate, non singolari.
- Sistemi rettangolari sovradeterminati.
- Sistemi rettangolari sottodeterminati.

Come procede MATLAB nella risoluzione di un sistema lineare?

Casi particolari

- $D \in \mathbb{R}^{n\times n}$ matrice diagonale con $d_{ii} \neq 0$ per $i = 1, \dots, n$:

$$Dx = b \Leftrightarrow x_i = \frac{b_i}{d_{ii}} \text{ per } i = 1, \dots, n.$$

La matrice può essere memorizzata in un vettore D.

```
>> D=[1 2 3 4 5];
>> b=[1 1 1 1 1];
>> x=b./D

x =

    1.0000    0.5000    0.3333    0.2500    0.2000
```


- $U \in \mathbb{R}^{n\times n}$ matrice triangolare superiore:

$$\begin{pmatrix} 1 & 2 & 0 \\ 0 & 2 & 1 \\ 0 & 0 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \\ 9 \end{pmatrix}$$

Si ricava dall'ultima equazione $x_3 = 3$, si sostituisce nell'equazione precedente:

$$2x_2 + x_3 = 3 \to 2x_2 = 3 - x_3 = 0 \to x_2 = 0;$$

si sostituiscono x_3 e x_2 nella prima equazione:

$$x_1 + 2x_2 + 0x_3 = 1 \to x_1 = 1.$$

Metodo di sostituzione all'indietro

Sia $U \in \mathbb{R}^{n\times n}$ matrice triangolare superiore; se $u_{ii} \neq 0$ per $i = 1,\dots,n$, la soluzione del sistema lineare $Ux = b$ è data da:

$$x_n = \frac{b_n}{u_{nn}}$$
$$x_i = \frac{1}{u_{ii}}\left(b_i - \sum_{j=i+1}^{n} u_{ij}x_j\right) \quad \text{per } i = n-1,\dots,1.$$

Metodo di sostituzione in avanti

Sia $L \in \mathbb{R}^{n\times n}$ matrice triangolare inferiore; se $\ell_{ii} \neq 0$ per $i = 1,\dots,n$, la soluzione del sistema lineare $Lx = b$ è data da:

$$x_1 = \frac{b_1}{\ell_{11}}$$
$$x_i = \frac{1}{\ell_{ii}}\left(b_i - \sum_{j=1}^{i-1} \ell_{ij}x_j\right) \quad \text{per } i = 2,\dots,n.$$

- **Caso generale**

Sia A una matrice quadrata di dimensione n non singolare, cioè tale che

$$\det(A) \neq 0.$$

Sia $b \in \mathbb{R}^n$. Allora il sistema lineare

$$Ax = b$$

ha un'unica soluzione $x \in \mathbb{R}^n$.

Idea: trasformare il sistema lineare assegnato in un sistema lineare equivalente che sia "più semplice".

Si eseguono un numero finito M di trasformazioni in modo tale che la matrice finale A_M sia una matrice triangolare superiore U.

Operazioni ammissibili per trasformare un sistema lineare in un sistema equivalente

- Scambio di riga: l'equazione i-esima del sistema si scambia con l'equazione j-esima del sistema.
- Moltiplicazione di una riga per uno scalare: l'equazione i-esima viene moltiplicata per un numero reale $\alpha \neq 0$.
- Somma fra righe: l'equazione i-esima viene sommata all'equazione j-esima.

Eliminazione di Gauss

Consideriamo il seguente sistema di 3 equazioni in 3 incognite:

$$\begin{pmatrix} 2 & 6 & 4 \\ 1 & 0 & 2 \\ 2 & 2 & 1 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 10 \\ -1 \\ 5 \end{pmatrix},$$

vogliamo trsformarlo in un sistema del tipo

$$\begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}.$$

Primo passo

Pongo $A^{(0)} = A$ e $b^{(0)} = b$ cioè

$$A^{(0)} = \begin{pmatrix} 2 & 6 & 4 \\ 1 & 0 & 2 \\ 2 & 2 & 1 \end{pmatrix} \quad b^{(0)} = \begin{pmatrix} 10 \\ -1 \\ 5 \end{pmatrix}.$$

Trasformo $A^{(0)}$ in $A^{(1)}$ e $b^{(0)}$ in $b^{(1)}$ in modo tale che $A^{(1)}$ abbia la seguente forma

$$A^{(1)} = \begin{pmatrix} 2 & 6 & 4 \\ 0 & \times & \times \\ 0 & \times & \times \end{pmatrix} \quad b^{(1)} = \begin{pmatrix} 10 \\ \times \\ \times \end{pmatrix}.$$

Primo passo (continua)

Faccio una combinazione lineare fra le prime due righe di $A^{(1)}$ in modo da "eliminare" l'incognita x_2 dalla seconda equazione. Precisamente moltiplico la prima equazione per $1/2 (= a_{21}/a_{11})$ e la sottraggo dalla seconda. Analogamente procedo con la terza riga e sottraggo la prima riga dalla terza (moltiplico la prima equazione per $1 (= a_{31}/a_{11})$) e ottengo

$$A^{(1)} = \begin{pmatrix} 2 & 6 & 4 \\ 0 & -3 & 0 \\ 0 & -4 & -3 \end{pmatrix} \quad b^{(1)} = \begin{pmatrix} 10 \\ -6 \\ -5 \end{pmatrix}.$$

Secondo e ultimo passo Trasformo $A^{(1)}$ in $A^{(2)}$ e $b^{(1)}$ in $b^{(2)}$ in modo tale che $A^{(2)}$ sia triangolare superiore.

Le prime due righe vengono riportate uguali.
Faccio una combinazione lineare tra la seconda e la terza riga in modo da eliminare anche l'incognita x_2 dalla terza equazione. Moltiplico la seconda riga per $4/3 = (a_{32}/a_{22})$ e la sottraggo dalla terza riga:

$$U = A^{(2)} = \begin{pmatrix} 2 & 6 & 4 \\ 0 & -3 & 0 \\ 0 & 0 & -3 \end{pmatrix} \quad y = b^{(2)} = \begin{pmatrix} 10 \\ -6 \\ 3 \end{pmatrix}.$$

Risolvendo il sistema $Ux = y$ con il metodo delle sostituzioni all'indietro si ottiene la soluzione $x = (1, 2, -1)'$.

Algoritmo di eliminazione di Gauss

for $k = 1, \ldots, n-1$
  for $i = k+1, \ldots, n$
    $m_{ik} = \dfrac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$
    for $j = k+1, \ldots, n$
      $a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)}$
    end
    $b_i^{(k+1)} = b_i^{(k)} - m_{ik} b_k^{(k)}$
  end
end

Costo dell'algoritmo di eliminazione di Gauss

Per calcolare il k-esimo passo, devo eseguire

$$\begin{array}{c} n-k \text{ divisioni} \\ (n-k)(n-k+1) \text{ moltiplicazioni} \\ (n-k)(n-k+1) \text{ somme/sottrazioni} \end{array}$$

sommando per $k = 1, \ldots, n-1$ si ha che il numero delle operazioni eseguite è dell'ordine di n^3 o meglio:

$$\texttt{flops} \cong \frac{2}{3} n^3.$$

Fattorizzazione LU

Costruiamo le seguenti matrici:

$$L = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ m_{21} & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ m_{n1} & \cdots & m_{nn-1} & 1 \end{pmatrix} \quad U = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & a_{nn}^{(n)} \end{pmatrix}$$

Teorema

$$LU = A.$$

Si dice che la matrice triangolare inferiore L e la matrice triangolare superiore U forniscono una **fattorizzazione LU** della matrice A.

Fattorizzazione (continua)

Supponiamo di avere una fattorizzazione LU della matrice A.
Per risolvere il sistema $Ax = b$ si procede come segue:

$$LUx = b.$$

Quindi si risolvono in sequenza due sistemi relativi a matrici triangolari:

$$\begin{array}{l} Ly = b \\ Ux = y. \end{array}$$

Attenzione: l'algoritmo fallisce se per qualche valore di k si ha $a_{kk}^{(k)} = 0$.

Esempio

$$A = \begin{pmatrix} 2 & 4 & 3 \\ 1 & 2 & 3 \\ 0 & 1 & 2 \end{pmatrix} \quad b = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}.$$

$$A^{(1)} = \begin{pmatrix} 2 & 4 & 3 \\ 0 & 0 & 3/2 \\ 0 & 1 & 2 \end{pmatrix} \quad b^{(1)} = \begin{pmatrix} 1 \\ 3/2 \\ 1 \end{pmatrix}.$$

Scambiando le righe del sistema si ottiene:

$$\tilde{A}^{(1)} = \begin{pmatrix} 2 & 4 & 3 \\ 0 & 1 & 2 \\ 0 & 0 & 3/2 \end{pmatrix} \quad \tilde{b}^{(1)} = \begin{pmatrix} 1 \\ 1 \\ 3/2 \end{pmatrix}.$$

Esercizio:

Consideriamo la matrice non singolare

$$A = \begin{pmatrix} 1 & 1 + 0.5 * 10^{-15} & 3 \\ 2 & 2 & 20 \\ 3 & 6 & 4 \end{pmatrix}$$

Calcolare la fattorizzazione LU di A mediante la function `lu_gauss` che si utilizza mediante il seguente comando

```
>> [l,u]=lu_gauss(A)
```

Calcolare la differenza $A - LU$.

Strategia di pivoting

Per evitare possibili divisioni per 0 e per rendere l'algoritmo di eliminazione (oppure l'algoritmo di fattorizzazione LU) stabili rispetto alla propagazione degli errori di arrotondamento si usa la **strategia di pivoting** che consiste nello scambio opportuno di righe.

Il risultato della fattorizzazione LU è:

$$PA = LU$$

essendo P una *matrice di permutazione* che tiene conto degli scambi di righe avvenuti.

Algoritmo di eliminazione di Gauss con pivoting

for $k = 1, \ldots, n-1$
  cerco più piccolo p tale che $|a_{pk}^{(k)}| = \max_{k \le i \le n} |a_{ik}^{(k)}|$
  scambio la riga k con la riga p
  for $i = k+1, \ldots, n$
    $m_{ik} = \dfrac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$
    for $j = k+1, \ldots, n$
      $a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)}$
    end
    $b_i^{(k+1)} = b_i^{(k)} - m_{ik} b_k^{(k)}$
  end
end

Le funzioni MATLAB per la fattorizzazione

Funzione	Significato
`lu`	Fattorizzazione $PA = LU$.
`chol`	Fattorizzazione di Cholesky $A = LL^T$.
`qr`	Fattorizzazione $A = QR$.
`qz`	Fattorizzazione QZ.
`eig`	Decomposizione spettrale (autovalori, autovettori).
`svd`	Decomposizione in valori singolari $A = U\Sigma V^T$.
`schur`	Decomposizione di Schur $A = UTU^H$.

Altre funzioni di MATLAB per l'algebra lineare

Funzione	Significato
`hess`	Forma di Hessenberg.
`orth`	Base ortonormale dello spazio immagine di A.
`null`	Base ortonormale del nucleo della matrice A.
`subspace`	Angolo tra vettori o sottospazi.
`planerot`	Matrice rotazione nel piano.
`det`	Determinante.
`rank`	Rango.
`inv`	Inversa.
`pinv`	Pseudoinversa di Moore-penrose.
`cond`	Numero di condizionamento (norma Euclidea).
`condest`	Stima numero di condizionamento (norma 1).

Altre fattorizzazioni

Fattorizzazione di Cholesky.

Data una matrice A simmetrica e definita positiva, la fattorizzazione di Cholesky esprime A come il prodotto di una matrice triangolare e della sua trasposta.

$$A = R'R$$

```
>> A=pascal(6)

A =

     1     1     1     1     1     1
     1     2     3     4     5     6
     1     3     6    10    15    21
     1     4    10    20    35    56
     1     5    15    35    70   126
     1     6    21    56   126   252
```



```
>> R=chol(A)

R =

     1     1     1     1     1     1
     0     1     2     3     4     5
     0     0     1     3     6    10
     0     0     0     1     4    10
     0     0     0     0     1     5
     0     0     0     0     0     1
```


Fattorizzazione QR.

La fattorizzazione QR esprime una matrice rettangolare A come il prodotto di una matrice ortogonale e di una matrice triangolare superiore. Si possono avere anche permutazioni delle colonne di A.

$$A = QR \qquad \text{oppure} \qquad AP = QR.$$

Una matrice Q si dice ortogonale se vale

$$Q'Q = I.$$

... per avere altre informazioni

```
>> help elmat ⏎
```

```
>> help matfun ⏎
```


Esercizio: Si consideri il sistema lineare $Ax = b$ con $A \in \mathbb{R}^{n \times n}$ *matrice di Hilbert* di elementi

$$a_{ij} = \frac{1}{i+j-1}, \quad i = 1, \ldots, n,$$

e $b \in \mathbb{R}^n$ tale che la soluzione del sistema sia $x = (1, \ldots, 1)^T$.

Calcolare con MATLAB la fattorizzazione LU con pivoting e risolvere il sistema al variare di n. Sia $\widehat{x}$ la soluzione calcolata.

Riportare l'errore relativo

$$E = \frac{\|x - \widehat{x}\|}{\|x\|}$$

in un grafico in scala semilogaritmica (usare il comando `semilogy`) per diversi valori di n.
Riportare sullo stesso grafico anche la norma della matrice $R = LU - PA$ e $\|b - A\widehat{x}\|/\|b\|$.
Per calcolare le norme usare il comando `norm`.

La soluzione **calcolata** $\widehat{x}$, in generale, non è uguale alla soluzione **esatta** x del sistema $Ax = b$.

Allora $\widehat{x}$ si può pensare come la soluzione **esatta** di un sistema **perturbato** del tipo

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

Per semplicità poniamo $\delta A = 0$.

$$\frac{\|\widehat{x} - x\|}{\|x\|} \leq \|A^{-1}\| \, \|A\| \frac{\|\delta b\|}{\|b\|}$$

$K(A) = \|A^{-1}\| \, \|A\|$ si dice **numero di condizionamento della matrice** A.

`cond` = reciproco del numero di condizionamento, *misura la distanza della matrice A dalle matrici singolari.*

Matrici sparse

Una matrice sparsa è una matrice che ha molti elementi nulli. Quindi basta memorizzare gli elementi diversi da zero.

```
>> help sparfun ⏎
```

Esempio

riga.mat contiene gli indici di riga;
col.mat contiene gli indici di colonna;
val.mat contiene gli elementi della matrice.

Usare il comando `load` per caricare i file:

```
>> load riga
>> load col
>> load val
```

Con il comando `sparse` si genera la matrice sparsa `a`

```
a=sparse(i,j,S)
```

Usare il comando `spy` per visualizzare la struttura della matrice

```
spy(a)
```


`a` è una matrice triangolare superiore.
Costruire la matrice simmetrica di cui `a` è la parte triangolare superiore:

```
>> b=a+a';
>>for i=1:size(a,1)
b(i,i)=b(i,i)/2;
end
```

Calcolare gli autovalori di `b` usando i comandi `eig` e `eigs`.