

Laboratorio di Calcolo Numerico
Risoluzione di sistemi di equazioni lineari
Condizionamento
Fattorizzazione LU
Stabilità

Ángeles Martínez Calomardo
<http://www.math.unipd.it/~acalomar/DIDATTICA/>
angeles.martinez@unipd.it

Laurea in Informatica
A.A. 2018–2019

Matrici

- Le variabili per Matlab/Octave hanno una struttura di tipo **matriciale**.
 - ▶ Gli scalari sono considerati matrici 1×1 .
 - ▶ I vettori riga sono matrici $1 \times n$.
 - ▶ I vettori colonna sono matrici $n \times 1$.
- Per definire una **matrice** se ne possono innanzitutto assegnare direttamente gli elementi riga a riga. Ad esempio digitando

```
>> A = [1 2 3; 4 5 6; 7 8 9]
```

si produce

```
A =  
    1    2    3  
    4    5    6  
    7    8    9
```

- Notiamo che il punto e virgola separano righe diverse.

Matrici

- L'elemento in riga i e colonna j di A si accede con $A(i,j)$. Per la matrice A dell'esempio precedente

```
>> A(2,3)
ans = 6
```

Esercizio

Costruire una matrice 2×3 con i primi sei numeri interi come coefficienti. Azzerare gli elementi $A(1,1)$ e $A(2,2)$.

- Soluzione.

```
>> A = [1 2 3; 4 5 6]
A =
     1     2     3
     4     5     6

>> A(1,1) = 0;
>> A(2,2) = 0;
>> A

A =
     0     2     3
     4     0     6
```

Estrarre parti di una matrice

Si può fare tramite l'uso del carattere due punti :

Ad esempio, per individuare la seconda riga di A, basta scrivere

```
>> A(2,:) 
```

Per individuare la terza colonna di A si scrive invece

```
>> A(:,3) 
```

- Per estrarre una intera sottomatrice da una matrice assegnata, basta specificare un insieme di righe e colonne. Esempio:

```
A =
```

```
1    1    1
-2    3    1
4   -6    7
```

```
>> M=A(2:3,:) 
```

```
M =
```

```
-2    3    1
4   -6    7
```

Matrici

Matlab allarga una matrice quanto basta per sistemare un elemento dato.

```
>> A=[1 1 1; -2 3 1; 4 -6 7]
```

```
A =
```

```
    1    1    1
   -2    3    1
    4   -6    7
```

```
>> A(5,5)=2
```

```
A =
```

```
    1    1    1    0    0
   -2    3    1    0    0
    4   -6    7    0    0
    0    0    0    0    0
    0    0    0    0    2
```

```
>> A=A(1:3,1:3)
```

```
A =
```

```
    1    1    1
   -2    3    1
    4   -6    7
```

```
>> A(3,6)=9
```

```
A =
```

```
    1    1    1    0    0    0
   -2    3    1    0    0    0
    4   -6    7    0    0    9
```

Comandi predefiniti che operano su matrici

- Comandi che generano matrici:

`rand(m,n)` matrice $m \times n$ con coefficienti random.

`eye(n)` matrice identità di ordine n .

`ones(n)` matrice di ordine n con coefficienti tutti uguali ad 1.

`zeros(n)` matrice di ordine n con coefficienti tutti uguali a 0.

- Altri comandi importanti che operano con matrici:

`inv(A)` calcola l'inversa della matrice A ;

`[n,m] = size(A)` restituisce il numero di righe e di colonne di A ;

`det(A)` calcola il determinante di A .

`eig(A)` calcola gli autovalori di A .

Esercizio

Creare una matrice quadrata A di ordine 4 con tutti gli elementi uguali a 1 e calcolarne il determinante.

Che cosa succede se proviamo a calcolare l'inversa di A ?

Comandi predefiniti che operano su matrici

Soluzione dell'esercizio

```
>> A = ones(4)
```

```
A =
```

```
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
```

```
>> det(A)
```

```
ans = 0
```

```
>> inv(A)
```

```
warning: inverse: matrix singular to machine precision , rcond = 0
```

```
ans =
```

```
Inf Inf Inf Inf
Inf Inf Inf Inf
Inf Inf Inf Inf
Inf Inf Inf Inf
```

Operazioni tra matrici

- Essendo A, B, C matrici con coefficienti reali ed s uno scalare, si definiscono le operazioni:
 - $C = s * A$ prodotto di una matrice per uno scalare.
 - $C = A'$ trasposizione di una matrice.
 - $C = A + B$ somma di due matrici di dimensione $m \times n$.
 - $C = A - B$ sottrazione di due matrici $m \times n$.
 - $C = A * B$ prodotto di A (m righe e n colonne) per B (n righe e p colonne).
 - $C = A .* B$ $c_{ij} = a_{ij} \cdot b_{ij}$ (prodotto di due matrici componente a componente).
- Esempio:

```
>> A = [2  -1; 3  4; -2  7]
```

```
A =
```

```
    2    -1
    3     4
   -2     7
```

```
>> B = A'
```

```
B =
```

```
    2     3    -2
   -1     4     7
```


Operazioni tra matrici: esempi

Riportiamo un esempio di somma, prodotto, e prodotto componente a componente di due matrici.

```
>> A = [4 -1 0; -1 4 -1; 0 -1 4];  
>> B = [1 2 3; -4 -5 -6; 0 1 2];
```

Il carattere ; usato alla fine di qualunque istruzione sopprime l'output a video.

```
>> C = A+B  
C =  
     5     1     3  
    -5    -1    -7  
     0     0     6
```

```
>> C = A*B  
C =  
     8     13     18  
    -17    -23    -29  
     4      9     14
```

```
>> C = A.*B  
C =  
     4     -2     0  
     4    -20     6  
     0     -1     8
```

Estrazione di diagonali da una matrice

Comando `diag`

Il comando `diag` estrae diagonali da una matrice e le restituisce in un vettore.

Esempi: Data la matrice $A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 5 & 7 \\ 3 & 3 & 4 & 8 \\ 0 & -1 & 2 & 3 \end{pmatrix}$,

```
>> diag(A)
ans =
```

```
1
3
4
3
```

```
>> diag(A, -1)
ans =
```

```
2
3
2
```

```
>> diag(A, 2)
ans =
```

```
3
7
```

Estrazione della parte triangolare inferiore (`tril`) e superiore (`triu`)

```
>> tril(A)
ans =
     1     0     0     0
     2     3     0     0
     3     3     4     0
     0    -1     2     3
```

```
>> tril(A,-1)
ans =
     0     0     0     0
     2     0     0     0
     3     3     0     0
     0    -1     2     0
```

```
>> triu(A)
ans =
     1     2     3     4
     0     3     5     7
     0     0     4     8
     0     0     0     3
```

Costruzione di una matrice diagonale

Comando `diag`

- A partire da un vettore

```
>> v=[ -1 -2 5 9];
```

```
>> D=diag(v)  
D =
```

Diagonal Matrix

-1	0	0	0
0	-2	0	0
0	0	5	0
0	0	0	9

```
>> w=[-1 -2 -3];
```

Si può scegliere su quale diagonale posizionare il vettore w , anche se in questo caso la matrice risultante non è più una matrice diagonale:

```
>> D=diag(w, -1)  
D =
```

0	0	0	0
-1	0	0	0
0	-2	0	0
0	0	-3	0

Costruzione di una matrice diagonale

Comando `diag`

- A partire dagli elementi diagonali di una matrice A data

```
>> A =  
    1    2    3    4  
    2    3    5    7  
    3    3    4    8  
    0   -1    2    3
```

```
>> diag(diag(A))  
ans =
```

Diagonal Matrix

```
    1    0    0    0  
    0    3    0    0  
    0    0    4    0  
    0    0    0    3
```

Norme vettoriali

Dato un vettore $x \in \mathbb{R}^n$ le principali norme vettoriali sono:

1. $\|x\|_1 = \sum_{i=1}^n |x_i|$ (norma 1)
2. $\|x\|_2 = \sqrt{\sum_{i=1}^n |x_i|^2}$ (norma 2 o euclidea)
3. $\|x\|_\infty = \max_{1 \leq i \leq n} |x_i|$ (norma infinito)

In Matlab queste norme si calcolano con il comando `norm`:

<code>norm(x)</code>	norma euclidea del vettore x
<code>norm(x,1)</code>	norma 1 del vettore x
<code>norm(x,inf)</code>	norma infinito del vettore x

Norme matriciali

- Analogamente per calcolare in Matlab/Octave la norma di una matrice A si usa il comando `norm(A)`.
- Se nessun ulteriore parametro viene specificato tale comando restituisce la norma 2 della matrice ovvero:

$$\|A\|_2 = \sqrt{\rho(A^T A)}$$

dove $\rho(A)$ è il raggio spettrale della matrice A . Altre possibilità sono:

- ▶ norma 1

$$\|A\|_1 = \max_j \sum_{i=1}^n |a_{ij}|, \text{ in Matlab/Octave } \texttt{norm(A,1)};$$

- ▶ norma infinito

$$\|A\|_\infty = \max_i \sum_{j=1}^n |a_{ij}|, \text{ in Matlab/Octave } \texttt{norm(A,inf)};$$

- ▶ norma di Frobenius

$$\|A\|_F = \sqrt{\sum_{i=1}^n \sum_{j=1}^n |a_{ij}|^2}, \text{ in Matlab/Octave } \texttt{norm(A,'fro')};$$

Norme matriciali

Esempi

```
>>> A = [5 -4 2; 1 7 -6; 1 1 9]  
A =
```

```
    5    -4     2  
    1     7    -6  
    1     1     9
```

```
>> norm(A,1)  
ans = 17
```

```
>> norm(A,inf)  
ans = 14
```

```
>> norm(A, 'fro')  
ans = 14.6287388383278
```

```
>>> norm(A)  
ans = 12.0560586095913
```


Soluzione di un sistema lineare

Il sistema lineare $Ax = b$ si risolve in Matlab/Octave con il comando $A \backslash b$.

Se A è una matrice quadrata invertibile generale, l'operatore \backslash restituisce la soluzione $x = A^{-1}b$ calcolata con il metodo di eliminazione di Gauss con pivoting.

```
>> A=[10 -7 0; -3 2 6; 5 -1 5]
```

```
A =
```

```
    10    -7     0
    -3     2     6
     5    -1     5
```

```
>> b=[7; 4; 6]
```

```
b =
```

```
    7
    4
    6
```

```
>> x=A\b
```

```
x =
```

```
    0
   -1
    1
```

Condizionamento dei sistemi lineari

Consideriamo il sistema di equazioni lineari $Ax = b$, con $A \in \mathbb{R}^{n \times n}$ e $b \in \mathbb{R}^n$.

Sia $\delta x = e = \bar{x} - x$ l'errore sul risultato in seguito ad una perturbazione δb sul termine noto b (per semplicità, assumiamo $\delta A = 0 \in \mathbb{R}^{n \times n}$). Possiamo pensare dunque che il sistema che si risolve sia

$$A(x + \delta x) = b + \delta b \quad (1)$$

Da cui, poichè $Ax = b$ si ha

$$A\delta x = \delta b; \quad \delta x = A^{-1} \delta b \quad (2)$$

Rispetto ad una qualsiasi norma matriciale indotta da quella vettoriale, seguono le maggiorazioni

$$\|\delta x\| = \|A^{-1} \delta b\| \leq \|A^{-1}\| \|\delta b\| \quad (3)$$

$$\|b\| = \|Ax\| \leq \|A\| \|x\| \implies \frac{1}{\|x\|} \leq \frac{\|A\|}{\|b\|} \quad (4)$$

Condizionamento di un sistema lineare

Moltiplicando tra di loro la (3) e la (4), si ha infine

$$\frac{\|\delta x\|}{\|x\|} \leq \|A\| \|A^{-1}\| \frac{\|\delta b\|}{\|b\|}, \quad (5)$$

dove

$$\boxed{\kappa(A) = \|A\| \|A^{-1}\|} \quad (6)$$

si chiama *numero di condizionamento* della matrice A .

Esso è sempre ≥ 1 , in quanto si ha:

$$1 = \|I\| = \|AA^{-1}\| \leq \|A\| \|A^{-1}\| = \kappa(A)$$

Malcondizionamento

Considerando che

$$A(x + \delta x) = b + \delta b$$

si ha

$$A\bar{x} = b + \delta b \quad \implies \quad \delta b = A\bar{x} - b = -r$$

Vediamo dunque che la norma dell'errore relativo e quella del residuo relativo sono legate mediante la relazione:

$$\frac{\|x - \bar{x}\|}{\|x\|} \leq K(A) \frac{\|r\|}{\|b\|}.$$

- Per valori molto grandi di $\kappa(A)$, diciamo per $\kappa(A) > 10^3$, l'errore relativo sulla soluzione può essere molto grande anche se è piccolo l'errore relativo sui dati.
- Vale a dire che *a residuo piccolo può non corrispondere un errore piccolo*. In questi casi si parla di *malcondizionamento* del sistema o della matrice.
- Numeri di condizionamento diversi si hanno in corrispondenza a scelte diverse della norma matriciale.

Numero di condizionamento in Matlab/Octave

Il numero di condizionamento di una matrice definito come $\|A\|_2 \cdot \|A^{-1}\|_2$ si calcola in Matlab/Octave con il comando `cond`.

```
>> A = [4 -1 2; 1 3 1; 0 -3 5]
A =

     4     -1      2
     1      3      1
     0     -3      5
>> cond(A)
ans = 2.4249
```

Specificando come secondo parametro del comando `cond` uno tra i seguenti valori: 1, inf, 'fro', si ottiene il numero di condizionamento in norma 1, infinito e di Frobenius, rispettivamente.

```
>> cond(A,1)
ans = 3.7183
>> cond(A,inf)
ans = 3.1549
>> cond(A,'fro')
ans = 3.8379
```

Esempi di matrici malcondizionate

Matrici di Hilbert

- Una classe di matrici malcondizionate è fornita dalle matrici di Hilbert di ordine n i cui elementi sono $h_{ij} = \frac{1}{i+j-1}$.
- In Matlab/Octave tali matrici si creano tramite il comando `hilb(n)`.

```
>> H = hilb(3)
H =

    1.00000    0.50000    0.33333
    0.50000    0.33333    0.25000
    0.33333    0.25000    0.20000

>> cond(H)
ans = 524.06
>> H1 = hilb(10);
>> cond(H1)
ans = 1.6025e+13
```

- Più grande è il condizionamento, meno accurata potrebbe essere la soluzione del sistema lineare.

Esempi di matrici malcondizionate

Matrici di Hilbert

Il seguente script (*scripthilb.m*) crea le matrici di Hilbert di ordine n , per $n = 3, \dots, 15$ e ne calcola il numero di condizionamento.

```
for n=3:15
    H=hilb(n);
    k=cond(H);
    fprintf('\n \t [N ]   %3d           [COND K(H)]   %6.4e ',n, k)
end
fprintf('\n')
```

I numeri di condizionamento calcolati sono:

[N]	3	[COND]	5.2406 e+02
[N]	4	[COND]	1.5514 e+04
[N]	5	[COND]	4.7661 e+05
[N]	6	[COND]	1.4951 e+07
[N]	7	[COND]	4.7537 e+08
[N]	8	[COND]	1.5258 e+10
[N]	9	[COND]	4.9315 e+11
[N]	10	[COND]	1.6025 e+13
[N]	11	[COND]	5.2260 e+14
[N]	12	[COND]	1.6776 e+16
[N]	13	[COND]	1.7590 e+18
[N]	14	[COND]	3.0821 e+17
[N]	15	[COND]	4.4333 e+17

>>

Esempi di matrici malcondizionate

Esercizio

Si generi la matrice di Hilbert H di ordine 12 e si risolva il sistema lineare $Hx = b$, in cui b corrisponde alla soluzione vera $x = [1, \dots, 1]^T$.

Si visualizzino il numero di condizionamento di H e l'errore e il residuo relativo.

Risoluzione esercizio

Riportiamo di seguito lo script (*solhilb.m*) che risolve l'esercizio.

```
n = 12;
xvera = ones(n,1);
H = hilb(n);
kappa = cond(H);
fprintf('\n \t [COND K_2(H)]           %3.3e ', kappa);
b = H* xvera;
x = H\b;
err_rel = norm(x-xvera)/norm(xvera);
fprintf('\n \t [ERRORE RELATIVO]       %3.3e ', err_rel);
res_rel = norm(b-H*x)/norm(b);
fprintf('\n \t [RESIDUO RELATIVO]      %3.3e ', res_rel);
fprintf('\n ');
```

Commenti all'esercizio

L'esecuzione dello script fornisce i seguenti risultati

```
>> solhilb  
  
[COND K_2(H)]      1.680e+16  
[ERRORE RELATIVO]  1.641e-01  
[RESIDUO RELATIVO] 1.166e-16
```

Si osserva che:

- il residuo relativo è circa la precisione di macchina mentre l'errore relativo è $1.64 \cdot 10^{-1}$, maggiore di quindici ordini di grandezza rispetto al residuo.
- ciò non è sorprendente se si considera che il numero di condizionamento è molto grande.
- Matlab/Octave produce il messaggio di warning per indicare che la matrice è molto malcondizionata e quindi i risultati potrebbero non essere attendibili.

Esempi di matrici malcondizionate

Matrice di Vandermonde

Un altro esempio di matrice malcondizionata è la matrice di Vandermonde di ordine n , definita a partire da un vettore $x = x_1, \dots, x_n$ come $V_{ij} = x_i^{j-1}$.

In Matlab/Octave tale matrice si crea con il comando `vander(x)`.

Il seguente script *vandercond.m*, dato il vettore $x = 0 : 1/(n-1) : 1$, crea le matrici `vander(x)` per $n = 3, \dots, 10$ e ne calcola il numero di condizionamento. Inoltre, realizza un grafico semilogaritmico del condizionamento in funzione della dimensione della matrice.

```
vcond=zeros(10);  
for n=3:10  
    x=1:1/(n-1):2;  
    A=vander(x)  
    c=cond(A)  
    vcond(n)=c;  
    pause;  
end  
semilogy(3:10,vcond(3:10),'r-');
```

Esempi di matrici malcondizionate

Matrice di Vandermonde

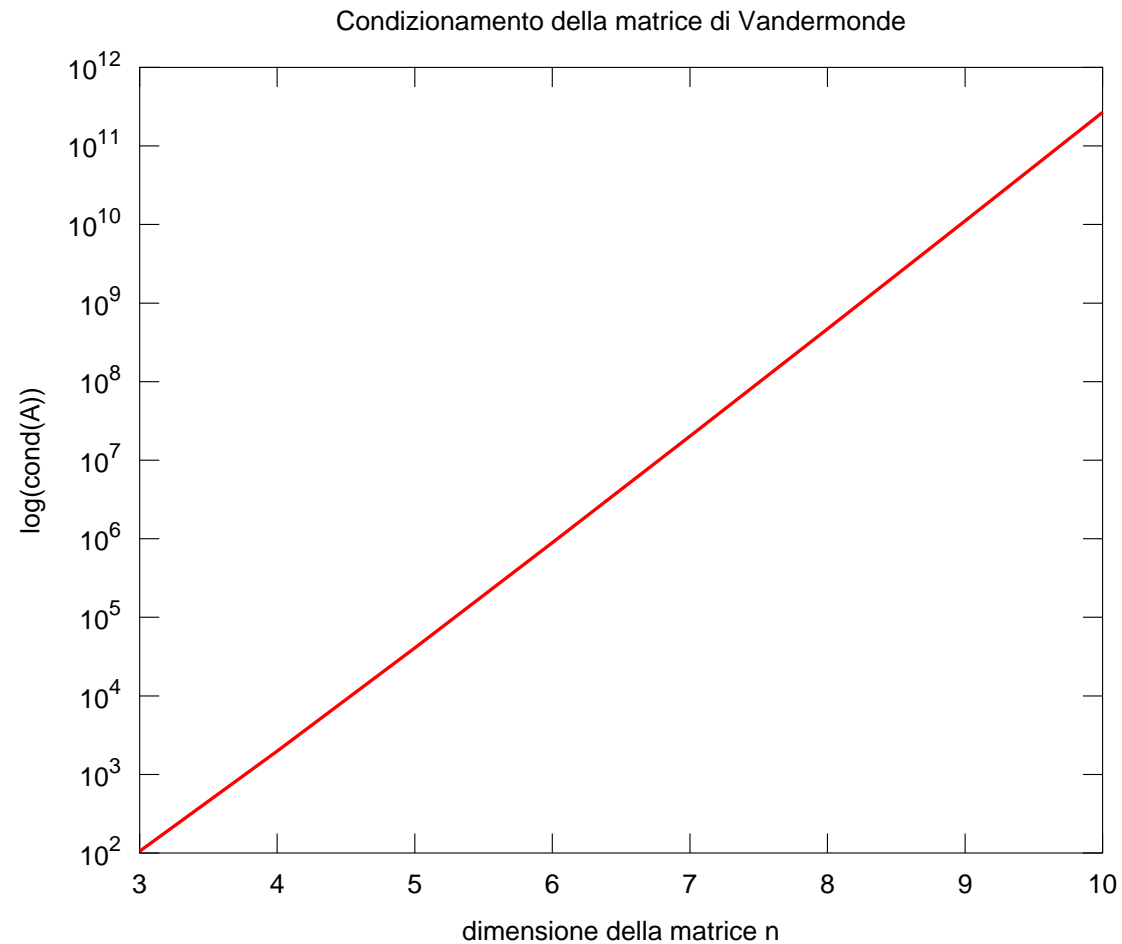


Figura: Grafico in scala semilogaritmica del condizionamento della matrice di Vandermonde al crescere della dimensione n .

Fattorizzazione LU

In Matlab il comando `lu` calcola la fattorizzazione LU di A , ottenuta mediante eliminazione di Gauss con pivoting parziale per righe.

La sintassi è:

$$[L, U, P] = \text{lu}(A)$$

con P matrice di permutazione tale che $PA = LU$.

Se il comando `lu` viene chiamato

$$[L, U] = \text{lu}(A)$$

allora U è la matrice triangolare superiore ottenuta dal MEG e la matrice L è in realtà $P^{-1}L$.

Dalla fattorizzazione LU alla soluzione del sistema $Ax = b$

Data la matrice A se la sua fattorizzazione LU è stata ottenuta mediante eliminazione di Gauss con pivoting si ha:

$$PA = LU \implies PAx = L \underbrace{Ux}_y = Pb \iff \begin{cases} Ly = Pb \\ Ux = y \end{cases}$$

Da un sistema siamo passati a dover risolvere due sistemi, ma più semplici perchè triangolari:

1. $Ly = Pb$ si risolve mediante *sostituzione in avanti*
2. $Ux = y$ si risolve mediante *sostituzione all'indietro*

Eliminazione di Gauss e Fattorizzazione LU

Sia $Ax = b$ con $A \in \mathbb{R}^{n \times n}$ regolare

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}.$$

Supponiamo che ad ogni passo k del processo di eliminazione, con $1 \leq k \leq n-1$, sia $a_{kk}^{(k-1)} \neq 0$ (elemento *pivot*). Poniamo $A^{(1)} = A$.

Al primo passo (per $k=1$): Si azzerano tutti gli elementi della 1^a colonna eccetto il primo.

Per azzerare a_{21} si trasforma la 2^a riga di A nel seguente modo:

$$R_2 \leftarrow R_2 - (\ell_{21} R_1), \quad (2^a \text{ riga} - \text{moltiplicatore} \times 1^a \text{ riga}).$$

dove $\ell_{21} = \frac{a_{21}}{a_{11}}$.

Analogamente si possono azzerare tutti gli altri elementi della 1^a colonna:

$$\boxed{\ell_{i1} = \frac{a_{i1}}{a_{11}}} \implies R_i \leftarrow R_i - (\ell_{i1} R_1), \quad 2 \leq i \leq n$$

Eliminazione di Gauss e Fattorizzazione LU

Quindi, dopo il primo passo, la matrice $A^{(1)}$ è stata trasformata, mediante *trasformazioni elementari* (di Gauss) nella matrice:

$$A^{(2)} = \begin{pmatrix} a_{11}^{(2)} & a_{12}^{(2)} & \cdots & a_{1n}^{(2)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \cdots & \cdots & \cdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} \end{pmatrix}$$

Per $k = 2$, secondo passo. Escludendo la prima riga (rimasta inalterata) e la prima colonna di $A^{(2)}$, riappliciamo il procedimento alla sua sottomatrice $n - 1 \times n - 1$.

Proseguendo così, dopo $n - 1$ passi si ottiene la matrice *triangolare superiore* $A^{(n)} = U$.

Gli elementi l_{ij} formeranno la matrice triangolare inferiore L , con $l_{ii} = 1, i = 1, \dots, n$, tale che $A = LU$.

Algoritmo di eliminazione di Gauss

L'algoritmo del metodo di eliminazione di Gauss si può schematizzare come segue

```
for  $k = 1, \dots, n - 1$  do  
  for  $i = k + 1, \dots, n$  do  
    
$$l_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$$
  
    for  $j = k, \dots, n$  do  
      
$$a_{ij}^{(k)} = a_{ij}^{(k)} - l_{ik} a_{kj}^{(k)}$$
  
    end for  
  end for  
end for
```

Esercizio

A partire dallo pseudocodice dell'Algoritmo precedente si scriva una **function** Matlab/Octave: **function**[L,U] = lugauss(A) che restituisce i due fattori triangolari L ed U.

Test della function `lugauss`

Esercizio

Si scriva uno script che definisca la matrice

$$A = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{pmatrix}$$

e il termine noto $b = (3 \ 4 \ 3)^T$, risolva il sistema $Ax = b$ (soluzione vera $x = (1 \ 1 \ 1)^T$), richiamando la function `lugauss` e risolvendo i due sistemi triangolari usando il comando `\` di Matlab/Octave .

Stabilità dell'algoritmo di eliminazione di Gauss

Pivoting

- La fattorizzazione LU non sempre è accurata a causa degli errori di arrotondamento: il fatto che al passo k l'elemento diagonale $a_{kk}^{(k-1)}$ (pivot) sia *piccolo*, pur non impedendo la conclusione del calcolo della fattorizzazione, **come succederebbe nel caso si incontrasse un pivot nullo**, può comunque comportare gravi perdite di accuratezza.
- L'uso della tecnica del **pivoting** migliora di molto l'accuratezza della fattorizzazione LU.
- **Pivoting parziale**: Scegliere come elemento *pivot* il massimo in modulo tra gli elementi nella sottocolonna k .

Questo garantisce che tutti i moltiplicatori siano in modulo ≤ 1 e impedisce di conseguenza la crescita eccessiva degli elementi nella matrice U nel caso generale.

Stabilità dell'algoritmo di eliminazione di Gauss

Esercizio proposto

Si risolva il sistema lineare $Ax = b$ dove, fissato $\varepsilon = 10^{-14}$,

$$A = \begin{pmatrix} 1 & 1 & -3 \\ 2 & 2 - \varepsilon & 4 \\ 1 & 9 & 4 \end{pmatrix}$$

e il termine noto $b = A\bar{x}$ si ottiene dopo aver imposto la soluzione vera pari a $\bar{x} = (1 \ 1 \ 1)^T$.

Una volta risolto il sistema e ricavata la soluzione approssimata x , si calcoli il residuo relativo $\frac{\|b - Ax\|}{\|b\|}$ e l'errore relativo $\frac{\|x - \bar{x}\|}{\|\bar{x}\|}$ commentando i risultati ottenuti.

Si ripetano gli stessi passi utilizzando poi la fattorizzazione LU con pivot fornita dalla function Matlab/Octave `[L,U,P] = lu(A)`.