Calcolo Numerico

A.A. 2014-2015

Lab n. 8

Metodi diretti per la soluzione di sistemi lineari

26 Novembre 2014

Matrici

- Una matrice si può definire come un **insieme di vettori riga** separati da un punto e virgola oppure di **vettori colonna** separati da uno spazio o una virgola

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

$$A = \begin{pmatrix} 2 & 5 & 7 \\ 1 & 7 & 9 \\ 8 & 1 & -4 \end{pmatrix} \in \mathbb{R}^{3\times 3}$$

Matrici

- Per conoscere la dimensione di una matrice A si usa il comando **`size(A)`** che restituisce un **vettore** di 2 componenti di cui la prima indica il numero di righe mentre la seconda il numero di colonne

Esempio: Se A = [3 5 8; -1 2 4]

```
>> size(A)

ans =
      2     3
```

num.righe num.colonne

```
>> [r c] = size(A)
```

Per memorizzare le dimensioni della matrice

Individuare elementi

`A(i,j)`	restituisce l'elemento i,j della matrice A
`A(i,:)`	restituisce il vettore riga corrispondente alla riga i della matrice A
`A(:,j)`	restituisce il vettore colonna corrispondente alla colonna j della matrice A
`A(i,m:p:n)`	restituisce un vettore riga contente gli elementi nelle colonne da m a n con passo p della i-esima riga della matrice A

Individuare elementi

- Per estrarre un **elemento** da una matrice il comando è

```
nome_matrice(indice_riga, indice_colonna)
```

Esempio: estrarre l'elemento di A di indice 2,3

```
>> A(2,3)
ans =
     0
```

$$A = \begin{pmatrix} 3 & 0 & 3 \\ 1 & 2 & 0 \end{pmatrix} \in \mathbb{R}^{2\times 3}$$

Esempio: estrarre l'elemento di A di indice 1,3

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

Individuare elementi

- Per estrarre una **riga** da una matrice il comando è

```
nome_matrice(indice_riga, :)
```

Esempio: Estrarre la prima riga di A

```
>> A(1,:)
ans =
     3     0     3
```

$$A = \begin{pmatrix} 3 & 0 & 3 \\ 1 & 2 & 0 \end{pmatrix} \in \mathbb{R}^{2\times 3}$$

Individuare elementi

- Per estrarre una **colonna** da una matrice il comando è

```
nome_matrice(:, indice_colonna)
```

Esempio: estrarre la terza colonna di A

```
>> A(:,3)
ans =
     3
     0
```

$$A = \begin{pmatrix} 3 & 0 & 3 \\ 1 & 2 & 0 \end{pmatrix} \in \mathbb{R}^{2\times 3}$$

Individuare elementi

Esempio: estrarre il vettore riga contenente gli elementi nelle colonne 1,3 della terza riga della matrice A

$$A = \begin{pmatrix} -5 & 2 & 1 & -7 & 9 \\ 2 & 4 & 20 & -2 & 1 \\ 4 & 10 & 19 & 1 & 4 \\ 3 & 5 & 4 & 0 & -1 \\ 1 & 1 & -4 & 3 & 5 \\ 0 & -1 & 1 & 6 & 4 \\ 7 & 6 & 0 & 20 & 3 \end{pmatrix}$$

```
>> A(3,1:2:3)

ans =

     4     19
```

Esempio: estrarre il vettore colonna contenente gli elementi nelle righe 2,4,6 della quarta colonna di A

```
>> A(2:2:end,4)

ans =

    -2
     0
     6
```

Estrarre sottomatrici

`A(m:k, i:j)` indica la sottomatrice di A con le righe da m a k, e le colonne da i a j.

`A(:, i:j)` sottomatrice delle colonne da i a j.

`A(m:k, :)` sottomatrice delle righe da m a k.

Esempio: Sia **`A=[1 2 0; 3 4 1; 2 6 3]`**, estrarre la sottomatrice con indici di riga da 2 a 3 e di colonna da 1 a 3

```
>> B = A(2:3,1:3)
   B = 3  4  1
       2  6  3
% è equivalente a B=A(2:3, :) ovvero   B=A(2:3, 1:end)
```

$$A = \begin{pmatrix} 1 & 2 & 0 \\ 3 & 4 & 1 \\ 2 & 6 & 3 \end{pmatrix}$$

Aggiungere elementi

A = [A u] aggiunge alla matrice A il vettore **u** come ultima colonna

N.B. u deve essere un vettore colonna; se fosse un vettore riga avremmo

A=[A u']

Esempio: Sia **A=[1 2 0; 3 4 1; 2 6 3]**. Aumentarne la dimensione aggiungendo il vettore **u = [-1; 4;0]** come ultima colonna

```
>> A = [A u]
A =
     1     2     0    -1
     3     4     1     4
     2     6     3     0
```

$$A = \begin{pmatrix} 1 & 2 & 0 \\ 3 & 4 & 1 \\ 2 & 6 & 3 \end{pmatrix}$$

Aggiungere elementi

`A = [A; v]` aggiunge alla matrice A il vettore `v` come ultima riga

N.B. v deve essere un vettore riga; se fosse un vettore colonna avremmo

`A = [A; v']`

Esempio: Sia `A=[1 2 0; 3 4 1; 2 6 3]`. Aumentarne la dimensione aggiungendo il vettore `u = [-1; 4;0]` come ultima riga

```
>> A = [A; u']
A =
     1     2     0
     3     4     1
     2     6     3
    -1     4     0
```

$$A = \begin{pmatrix} 1 & 2 & 0 \\ 3 & 4 & 1 \\ 2 & 6 & 3 \end{pmatrix}$$

Eliminare righe o colonne

- `A(i,:) =[]` elimina la **i**-ma riga di A
- `A(: , m:k)=[]` elimina tutte le colonne di indici da **m** a **k**

Esempio: Sia `A = [1 2 0; 3 4 1; 2 6 3]`. Eliminare la seconda riga di A. Eliminare la prima e la seconda colonna di A.

```
>> A(2,:)=[]
   A =
        1     2     0
        2     6     3
>> A(:,[1 2])=[]
A =
     0
     1
     3
```

$$A = \begin{pmatrix} 1 & 2 & 0 \\ 3 & 4 & 1 \\ 2 & 6 & 3 \end{pmatrix}$$

Annullare righe o colonne

- **`A(i,:) =0`** sostituisce alla **i**-ma riga di A un vettore di elementi nulli
- **`A(: , m:k)=0`** sostituisce tutte le colonne di indici da **m** a **k** con vettori di elementi nulli

Esempio: Sia **`A = [1 2 0; 3 4 1; 2 6 3].`** Annullare la prima e la seconda colonna.

```
>> A(:,[1 2])=0
A =
     0     0     0
     0     0     1
     0     0     3
```

$$A = \begin{pmatrix} 1 & 2 & 0 \\ 3 & 4 & 1 \\ 2 & 6 & 3 \end{pmatrix}$$

Il comando colon :

- **`A(:)`** organizza gli elementi della matrice in un **vettore colonna** (posiziona le colonne di A una sotto l'altra)
- Usando la stessa strategia **`A(k)`** indica il k-esimo elemento di A, ovvero il **k**-esimo elemento del vettore colonna in cui è memorizzata A

Esempio: Se **`A = [1 0 3; 4 2 1]`**

```
>> A(:)
ans =
     1
     4
     0
     2
     3
     1
>> A(5)
ans =
     3
```

Operazioni somma e differenza

- Gli operatori **+ - * /** si applicano direttamente a vettori e matrici quando le dimensioni sono conformi
- **Somma di vettori o matrici**

```
>> D = A + B
```

  - A e B devono avere la stessa dimensione
  - D ha la stessa dimensione di A e B

- **Differenza di vettori o matrici**

```
>> D = A - B
```

  - A e B devono avere la stessa dimensione
  - D ha la stessa dimensione di A e B

- **Trasposta di una matrice**

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

Operazioni somma e differenza

Esempio: Siano `v = [0 -4 5 7]` e `w = [1;9;2;0]`.

```
>> v+w     % non è corretto in quanto v è un vettore riga
   mentre w è un vettore colona

>> y=v+w'   % è corretta

   y =

      1     5     7     7
```

Esempio: Siano `A = [0 -4; 5 7]` e `B = [1 9; 2 0]`.

```
>> C = A - B

C =

    -1   -13
     3     7
```

Operazione prodotto

- **Prodotto righe-colonne di matrici C=A*B**
  - **$\#C_A = \#R_B$**
    - il numero di colonne di A deve essere uguale al numero di righe di B
  - C ha dimensione = [**$\#R_A$, $\#C_B$**]

```
>> C = A * B
```

- **Prodotto elemento per elemento D=A.*B**
  - **size(A) = size(B)**
    - A e B devono avere la stessa dimensione
  - D ha la stessa dimensione di A e B
  - D(i,j) = A(i,j)*B(i,j) per tutti gli indici i e j

```
>> D = A .* B
```

Operazione prodotto

Esempio: Siano `v = [0 –4 5 7]` e `w = [1;9;2;0]`.

```
>> v*w % prodotto scalare di due vettori
ans =
   -26
>> w*v
ans =
     0    -4     5     7
     0   -36    45    63
     0    -8    10    14
     0     0     0     0
```

`>> v*w'` e `w*v'` non sono corrette!

Operazione prodotto

Esempio: Siano `v = [0 -4 5 7]` e `w = [1;9;2;0]`.

```
>> v.*w'
ans =
     0   -36    10     0
```

`>> v.*w` non è corretta!

Esempio: Siano `A = [0 -4; 5 7]` e `B = [1 9; 2 0]`.

```
>> C = A * B
C =
    -8     0
    19    45
>> D = A.*B                    % Nota:  D ≠ C
D =
     0   -36
    10     0
```

Altre operazioni

- **Prodotto per uno scalare: `k*v`** moltiplica ogni elemento del vettore (o matrice) **`v`** per lo scalare **`k`**

Esempio: **`Sia v = [4 2 -1]`**

```
>> 3*v

ans =

    12     6    -3
```

- **Somma o differenza con uno scalare**: **`v + a`** somma ad ogni elemento di **`v`** (vettore o matrice) il numero **`a`**.

```
>> v-2

ans =

     2     0    -3
```

Altre operazioni

- **Potenza**: **`A^n`** (A matrice quadrata, n intero >0) moltiplica **A n** volte per se stessa

```
>> A = [4 1; 6 9];
>> A^3
   166   139
   834   861
```

Operazioni elemento per elemento

- **`C = A .*B`** Prodotto elemento per elemento C(i,j) = A(i,j)*B(i,j)
  - A e B devono avere le stesse dimensioni

- **`C = A./B`** Divisione elemento per elemento C(i,j)=A(i,j)/B(i,j)
  - A e B devono avere le stesse dimensioni

- **`C = A .^B`** Ciascun elemento di **A** viene elevato a potenza, con esponente l'elemento corrispondente di **B** C(i,j) = A(i,j)^B(i,j)

Operatori relazionali

- Si applicano a matrici o vettori aventi la **stessa dimensione** e sono **operazioni elemento per elemento**
  - `C = A < B` minore
  - `C = A <= B` minore o uguale
  - `C = A > B` maggiore
  - `C = A >= B` maggiore o uguale
  - `C = A == B` uguale
  - `C = A ~= B` diverso

- La matrice di output **C** è una matrice di tipo logical e ha la stessa dimensione di A e B
  - `C(i,j) = 1` se la relazione è verificata,
  - `C(i,j) = 0` se la relazione non è verificata

Operatori relazionali

Esempio:

```
>> A=[1 2 3; 4 5 6; 7 8 9]
A =
     1     2     3
     4     5     6
     7     8     9
>> B = [0 2 4; 3 5 6; 3 4 9]
B =
     0     2     4
     3     5     6
     3     4     9
>> A==B
ans =
     0     1     0
     0     1     1
     0     0     1
```

Verificare anche gli altri operatori

Operatori logici

- Si applicano a matrici o vettori aventi la **stessa dimensione** e sono **operazioni elemento per elemento**
  - `C = A & B` and
  - `C = A | B` or
  - `C = A ~ B` not
  - `C = xor(A, B)` or esclusivo

- `all(v)` restituisce 1 se tutti gli elementi del vettore v sono diversi da zero, 0 altrimenti. Se v è una matrice?

- `any(v)` restituisce 1 se almeno un elemento del vettore v è diverso da zero, 0 altrimenti. Se v è una matrice?

Operatori logici

Esempio: Siano **x=[0 5 3 7]** ed **y=[0 2 8 7]**

```
>> x>y
ans =
     0     1     0     0
>> x>4
ans =
     0     1     0     1
>> m=(x>y)&(x>4)
m =
     0     1     0     0
>> p= x|y
p =
     0     1     1     1
```

Funzioni

- **`sum(A)`** se A è un vettore calcola la somma degli elementi.
  Se A è una matrice? (usare lo help)
- **`prod(A)`** se A è un vettore calcola il prodotto degli elementi.
  Se A è una matrice?
- **`D=det(A)`** calcola il determinante di A (**A deve essere una matrice quadrata**)
- **`R=rank(A)`** calcola il rango di A
- **`N=norm(A,n)`** calcola la norma n di A.
  - Se **n = 1**, calcola la **norma 1**
  - **n = 2**, calcola la **norma 2**
  - **n = inf**, calcola la **norma infinito**

Funzioni

- **`B=inv(A)`** calcola la matrice inversa di A (**A è una matrice quadrata**)

- **`A=zeros(m,n)`** crea una matrice di zeri dimensione m x n
- **`A=ones(m,n)`** crea una matrice di uno dimensione m x n:
- **`D=eye(m,n)`** la matrice D ha tutti 1 in posizione diagonale
- **`D=eye(n)`** è la matrice identità di ordine n

Funzioni

- `D=diag(v)` costruisce una matrice diagonale con il vettore v sulla diagonale principale
- `v=diag(D)` estrae la diagonale principale della matrice D
- `w=diag(D,k)` se k > 0 estrae la k-esima diagonale superiore, se k < 0 estrae la k-esima diagonale inferiore
- `T = triu(A)` estrae la parte traingolare superiore di A. T è una matrice triangolare superiore
- `T = triu(A,k)` estrae gli elementi che sono al di sopra della k-esima diagonale di A.
- `T = tril(A)` `?` (usare lo help)
- `R=rand(m,n)` matrice di dimensione mxn i cui elementi sono numeri casuali uniformemente distribuiti tra 0 e 1
- `R= randn(m,n)` matrice mxn i cui elementi sono numeri casuali con distribuzione normale di media 0 e varianza 1

Esercizio 1

- Data la matrice A calcolare il determinante, il rango e le norme 1 e infinito.

$$A = \begin{pmatrix} 0 & 1 & -2 \\ -1 & 5 & 3 \\ 2 & -4 & 0 \end{pmatrix}$$

  - scambiando la prima e la terza riga di A, il rango della matrice cambia?
  - sostituendo alla seconda riga di A, una combinazione lineare della prima e della seconda riga, il rango cambia?
  - sostituendo alla seconda riga di A, una combinazione lineare della prima e della terza riga, il rango cambia?

Esercizio 1 - soluzione

```
>> A = [0 1 -2; -1 5 3; 2 -4 0];
>>   det(A)   %0*5*0+2*3*1+(-2)*(-4)*(-1)-2*5*(-2)-0*3*(-4)-(-
   1)*1*0
ans =
    18
>> rank(A)     %(A è non singolare!!!)
ans =
     3
>> norm(A,1) %(max(0+1+2,1+5+4,2+3+0)=max(3,10,5) = 10)
ans =
    10
>> norm(A,inf) %(max(0+1+2,1+5+3,2+4+0)= max(3,9,6) = 9)
ans =
     9
```

Esercizio 1 - soluzione

Scambiando la prima e la terza riga di A, il rango della matrice **non** cambia

```
A =
     0     1    -2
    -1     5     3
     2    -4     0
>> B = A;
>> B(1,:)=A(3,:);,  B(3,:)=A(1,:);
>> disp(B)
     2    -4     0
    -1     5     3
     0     1    -2
>> rank(A)
ans =
     3
>> rank(B)
ans =
     3
```

Esercizio 1 - soluzione

Sostituendo alla seconda riga di A, una combinazione lineare della prima e della seconda riga, il rango **non** cambia

```
>> B = A;
>> B(2,:)=B(1,:)-2*B(2,:);

>> B
B =
     0     1    -2
     2    -9    -8
     2    -4     0

>> rank(B)
ans =
     3
```

Esercizio 1 - soluzione

Sostituendo alla seconda riga di A, una combinazione lineare della prima e della terza riga, **il rango cambia** (le righe della matrice non sono più linearmente indipendenti!)

```
>> B=A;
>> B(2,:)=B(1,:)-2*B(3,:);
>> B
B =
     0     1    -2
    -4     9    -2
     2    -4     0
>> rank(B)
ans =
     2
>> det(B)
ans =
     0
```

Esercizio 1 - soluzione

```
>> norm(A,1)
ans =
    10
```

$$A = \begin{pmatrix} 0 & 1 & -2 \\ -1 & 5 & 3 \\ 2 & -4 & 0 \end{pmatrix}$$

È equivalente a

```
>> max(sum(abs(A)))
ans =
    10
```

```
>> norm(A,inf)
ans =
     9
```

È equivalente a

```
>> max(sum(abs(A')))
ans =
     9
```

Esercizio 2

- Costruire la matrice $A \in \mathbb{R}^{3\times 7}$
  - la I riga è **`a1 = 14,12,…,2`**
  - la II riga è **`a2 = 1,1,…,1`**
  - la III riga è **`a3 = 0,0,…,0`**
- Modificare l'elemento **`A(1,3)`** ponendolo uguale a **`3`**
- Estrarre 2 sottomatrici:
  - una costituita dalle ultime 3 colonne
  - una costituita dalla I e III riga e dalle colonne II e IV

Esercizio 3

- Scrivere la funzione **calcola_norma.m** che prenda come parametro di **input**
  - una matrice A
  - una stringa tipo_norma (che può assumere i valori 'uno' e 'inf')

```
function norma = calcola_norma(A, tipo_norma)
```

e restituisca in **output** la norma tipo_norma di A calcolata usando la defi

- Norma uno: $\|A\|_1 := \max_{1 \le j \le n} \sum_{i=1}^{n} |a_{ij}|$ *(per colonne)*
- Norma infinito: $\|A\|_\infty := \max_{1 \le i \le n} \sum_{j=1}^{n} |a_{ij}|$ *(per righe)*

Esercizio 4

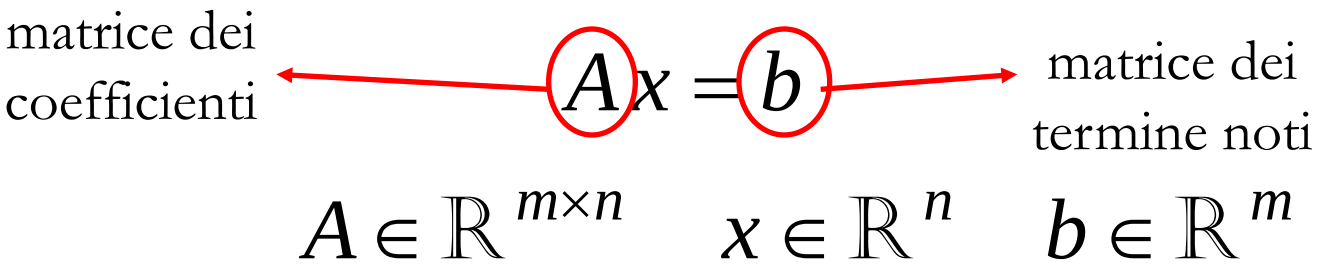
- Costruire le matrici A e B:
  - […] sta per parte intera; usare il comando floor

$$A = \begin{pmatrix} 1 & 1 & 1 & [\sqrt{1}] \\ 2 & 4 & -1 & [\sqrt{2}] \\ 3 & 9 & 1 & [\sqrt{3}] \\ \vdots & \vdots & \vdots & \vdots \\ 8 & 64 & -1 & [\sqrt{8}] \end{pmatrix}$$

$$B = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{pmatrix}$$

Sistemi lineari: metodi diretti

Sistemi Lineari



Teorema di Rouchè-Capelli

- rango(A) = rango(A|b) <=> il sistema è risolubile

- rango(A) = rango(A|b) = n => <u>unica soluzione</u>
- rango(A) = rango(A|b) = p < n => <u>∞^{n-p} soluzioni</u>

Risoluzione di sistemi Lineari

- **Sostituzione**
- **Cramer**
  - SE ***A*** è **quadrata** E **invertibile**
  - inutilizzabile per il costo computazionale (n-1)n!(n+1)
- **Algoritmo di Gauss**
  - **operazioni elementari sulle righe**: si trasforma il sistema di partenza in un sistema equivalente, dove la matrice dei coefficienti è triangolare superiore.
  - **pivotizzazione** (parziale o totale): da un punto di vista teorico non ha importanza la scelta del **pivot**, tale scelta è importante quando si implementa l'algoritmo al calcolatore
    - in analisi numerica si deve sempre tener presente che si lavora con una certa approssimazione e che si ha il problema della propagazione degli errori

Algoritmo di Gauss

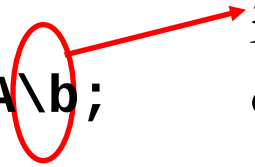
- Il metodo di eliminazione di Gauss trasforma, in $n-1$ passi, il sistema lineare $Ax=b$ con matrice dei coefficienti ***A* piena**, nel sistema equivalente $Ux=d$ con matrice dei coefficienti ***U* triangolare superiore**
- Il metodo utilizza le seguenti operazioni lecite
  - scambio di 2 equazioni
  - somma di un'equazione con un'altra moltiplicata per una costante

- Il metodo di eliminazione di Gauss può essere interpretato come la fattorizzazione della matrice di partenza A nel prodotto di due matrici triangolari.

Algoritmo di Gauss - Matlab

$$A \in \mathbb{R}^{nxn} \quad x \in \mathbb{R}^{n} \quad b \in \mathbb{R}^{n}$$

$$\det A \neq 0 \Rightarrow \exists! \, x \in \mathbb{R}^{n} \; t.c. \; Ax = b$$



il simbolo nn è quello della divisione!!

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

```
>> x = inv(A)*b;
```

- \ (backslash) **x=A\b** è la soluzione dell'equazione **Ax=b**.
  - la soluzione è calcolata mediante l'algoritmo Gaussiano con **pivot parziale:** ad ogni passo si sceglie il pivot con valore assoluto più grande
  - tempo richiesto minore del calcolo dell'inversa

Algoritmo di Gauss - Matlab

- Se **b** è un **vettore**, **x=A\b** sarà la soluzione del sistema lineare calcolata mediante il **metodo di eliminazione di Gauss**
- Se **B** è una **matrice** avente come colonne $b_1, b_2, \ldots b_n$, **X=A\B** sarà a sua volta una matrice, avente come colonne $x_1, x_2, \ldots x_n$ che sono le soluzioni dei sistemi lineari $Ax_1=b_1$, $Ax_2=b_2$, … $Ax_n=b_n$.

- Operatori di divisione **/ (slash)**
  - **Slash:** **X=B/A** è la matrice tale che **XA=B**.

Esempio: soluzione del sistema Ax=b

Risolvere il seguente sistema lineare

$$\begin{cases} & x_2 & -2x_3 = -3 \\ -x_1 & 5x_2 & 3x_3 = -3 \\ 2x_1 & -4x_2 & = 6 \end{cases}$$

```
>> A=[0  1  -2; -1  5  3; 2 -4 0];
>> b = [-3 -3 6]'; % b deve essere un vettore colonna
>> det(A) % ci assicuriamo che il determinante sia non nullo
>> x = A\b;
>> disp(x)
     1
    -1
     1
```

Esempio: calcolo dell'inversa

Calcolare l'inversa della matrice A

$$A = \begin{pmatrix} 0 & 1 & -2 \\ -1 & 5 & 3 \\ 2 & -4 & 0 \end{pmatrix}$$

```
>> IA = inv(A)
IA =
    0.6667    0.4444    0.7222
    0.3333    0.2222    0.1111
   -0.3333    0.1111    0.0556
```

Esempio: calcolo dell'inversa

```
>> I = eye(size(A)) % matrice identità
I =
     1     0     0
     0     1     0
     0     0     1
>> IA2 = A\E
IA2 =
    0.6667    0.4444    0.7222
    0.3333    0.2222    0.1111
   -0.3333    0.1111    0.0556
```

Esempio: calcolo dell'inversa

```
>> IA == IA2
ans =
     1     1     1
     1     1     1
     1     1     1
```

L'operazione **`IA2 = A\E`** equivale a risolvere i 3 sistemi

```
A * IA(1,:) = (1 0 0)'
A * IA(2,:) = (0 1 0)'
A * IA(3,:) = (0 0 1)'
```

Quale metodo è più veloce da un punto di vista computazionale?

Esercizio 5

- Scrivere una **function** Matlab **`MEG_nopivot.m`** che implementi il **metodo di eliminazione di Gauss (MEG) senza pivoting** per la risoluzione di sistemi lineari.
- Risolvere il seguente sistema lineare richiamando la function **`MEG_nopivot.m`**

$$\begin{cases} x_1 + 2x_2 + 3x_3 = 1 \\ 4x_1 + 5x_2 + 6x_3 = 0 \\ 7x_1 + 8x_2 = 2 \end{cases}$$

Metodo di eliminazione di Gauss (MEG)

Si eseguono n-1 passi

al passo k si definiscono gli elementi

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - m_{ik} a_{kj}^{(k-1)}, \quad i = k+1, \ldots, n \quad j = k, k+1, \ldots, n$$

$$b_{i}^{(k)} = b_{i}^{(k-1)} - m_{ik} b_{k}^{(k-1)}, \quad i = k+1, \ldots, n$$

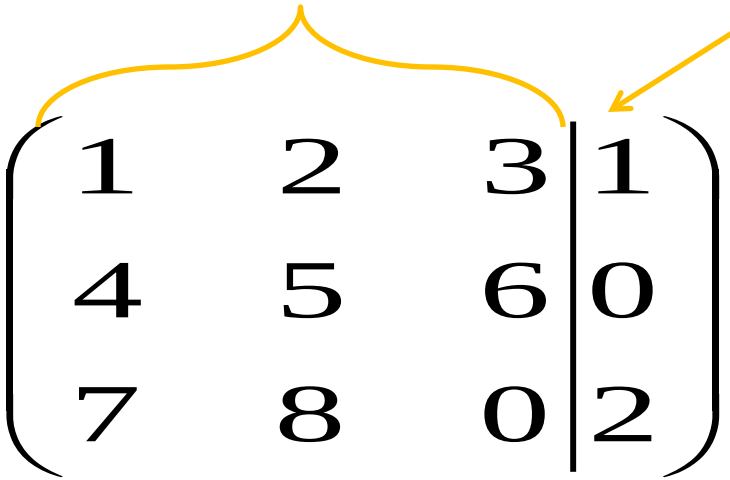
con $m_{ik} = \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}}$

MEG

- Per risolvere il sistema lineare

$$\begin{cases} x_1 + 2x_2 + 3x_3 = 1 \\ 4x_1 + 5x_2 + 6x_3 = 0 \\ 7x_1 + 8x_2 = 2 \end{cases}$$

si considerano la matrice A e il termine noto b associati



MEG

- **I passo:**
  - si moltiplica la prima riga per 4 e si sottrae alla seconda
  - si moltiplica la prima riga per 7 e si sottrae alla terza

$$\left(\begin{array}{ccc|c} 1 & 2 & 3 & 1 \\ 0 & -3 & -6 & -4 \\ 0 & -6 & -21 & -5 \end{array}\right)$$

Si annullano così il secondo e il terzo elemento della prima colonna di A

MEG

- **II passo:**
  - si moltiplica la seconda riga per 6/3 e si sottrae alla terza

$$\left(\begin{array}{ccc|c} 1 & 2 & 3 & 1 \\ 0 & -3 & -6 & -4 \\ 0 & 0 & -9 & 3 \end{array}\right)$$

Si annulla così il terzo elemento della seconda colonna di *A*

MEG

- Al termine del secondo passo si ottiene un sistema equivalente a quello dato in cui però la matrice dei coefficienti è **triangolare superiore**

$$\begin{cases} x_1 + 2x_2 + 3x_3 = 1 \\ -3x_2 - 6x_3 = -4 \\ -9x_3 = 3 \end{cases}$$

- A questo punto è possibile risolvere il sistema con il metodo di sostituzione all'indietro

$$x_3 = -3/9 = -1/3$$
$$x_2 = (-4 + 6x_3)/(-3) = 2$$
$$x_1 = 1 - 3x_3 - 2x_2 = 1 + 1 - 4 = -2$$

Dal command window…

Definiamo la matrice del sistema *A* e il vettore *b* dei termini noti

```
>> A = [1 2 3; 4 5 6; 7 8 0];
>> b = [1 0 2]';

>> x = MEG_nopivot(A,b);
la soluzione del sistema e' x=
-2.00000
+2.00000
-0.33333
```

Fattorizzazione LU

- Un altro modo di realizzare l'algoritmo di Gauss per $Ax=b$ con A quadrata non singolare è quello di passare attraverso la fattorizzazione LU

$$Ax = b \quad A \in \mathbb{R}^{nxn} \quad x, b \in \mathbb{R}^{n}$$

$$\det A \neq 0 \Rightarrow \exists\, L, U \quad t.c. \quad A = LU$$

dove:

- U è la **matrice triangolare superiore** ottenuta da A mediante l'algoritmo di Gauss con pivotizzazione parziale
- L è una **matrice quadrata invertibile** e a meno di permutazioni delle righe è una matrice triangolare inferiore con tutti 1 sulla diagonale

Risoluzione di un sistema con LU

$$A = LU \quad \Rightarrow \quad Ax = b \quad \Leftrightarrow \quad LUx = b$$

$$Ax = b \Leftrightarrow \begin{cases} Ly = b \\ Ux = y \end{cases}$$

sistema triangolare

Il sistema $Ax = b$ è equivalente a risolvere i due sistemi $Ly = b$ e $Ux=y$ con L e U matrici rispettivamente triangolare inferiore e superiore e tali che $LU=A$

In Matlab

```
>> [L U] = lu(A);
>> y = L\b;
>> x = U\y;
```

Il tempo complessivo richiesto dai tre comandi equivale a quello richiesto dall'algoritmo Gaussiano

Fattorizzazione LU

- Infatti data $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{pmatrix}$

Moltiplicare per 4 la prima riga e sottrarla alla seconda e moltiplicare la prima riga per 7 e sottrarla alla terza equivale a moltiplicare a destra la matrice A per la matrice

$$L_1 = \begin{pmatrix} 1 & 0 & 0 \\ -4 & 1 & 0 \\ -7 & 0 & 1 \end{pmatrix}$$

cioè

$$\begin{pmatrix} 1 & 0 & 0 \\ -4 & 1 & 0 \\ -7 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \\ 0 & -6 & -21 \end{pmatrix}$$

Fattorizzazione LU

Moltiplicare la seconda riga della matrice ottenuta al passo precedente per 2 e sottrarla alla terza equivale a moltiplicare a destra la matrice $L_1 A$ per la matrice

$$L_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -2 & 1 \end{pmatrix}$$

cioè

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -2 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \\ 0 & -6 & -21 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \\ 0 & 0 & -9 \end{pmatrix}$$

Fattorizzazione LU

Ma allora la matrice triangolare superiore U ottenuta precedentemente con il metodo di eliminazione di Gauss è tale che

$$L_2 L_1 A = U$$

e quindi

$$A = (L_2 L_1)^{-1} U = L_1^{-1} L_2^{-1} U = LU$$

Si verifica facilmente che $L_1^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 4 & 1 & 0 \\ 7 & 0 & 1 \end{pmatrix}$ e $L_2^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 2 & 1 \end{pmatrix}$

Quando conviene usare LU?

$$Ax = b_i \quad i = 1,2,\ldots,k$$

Quando si hanno più sistemi con la **stessa matrice dei coefficienti** A conviene decomporre una sola volta A in LU e risolvere i sistemi lineari con i comandi precedenti => risparmio di tempo

```
>> [L U] = lu(A);
>> y = L\b(:,i);  % i=1,…,k
>> x = U\y;
```

Esercizi

- Dopo essersi accertati che il sistema ammette un'unica soluzione trovare la soluzione con Gauss e mediante la decomposizione LU e confrontare i tempi di calcolo

$$A = \begin{pmatrix} 3 & 1 & -1 \\ 1 & 1 & -3 \\ 1 & 1 & 0 \end{pmatrix} \quad b = \begin{pmatrix} 0 \\ -5 \\ 1 \end{pmatrix}$$

- Provare a scrivere uno script per risolvere un sistema triangolare con il metodo di Cramer

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

MEG e Fattorizzazione LU

- Questi algoritmi sono **molto onerosi** computazionalmente e per questo è preferibile usarli per sistemi di **dimensioni ridotte** oppure in caso di matrici particolari, per esempio **tridiagonali**.

- In questo caso il numero di elementi diversi da zero della matrice è molto basso rispetto alla dimensione della matrice e, quindi, sia la fattorizzazione che le sostituzioni in avanti e indietro richiedono un numero di operazioni molto basso.

- Inoltre anche lo spazio occupato in memoria è molto basso. Infatti, basta memorizzare le 3 diagonali al posto dell'intera matrice.

Algoritmo di Thomas

- Spesso in applicazioni concrete ci si trova a dover risolvere sistemi lineari la cui matrice è tridiagonale, cioè del tipo:

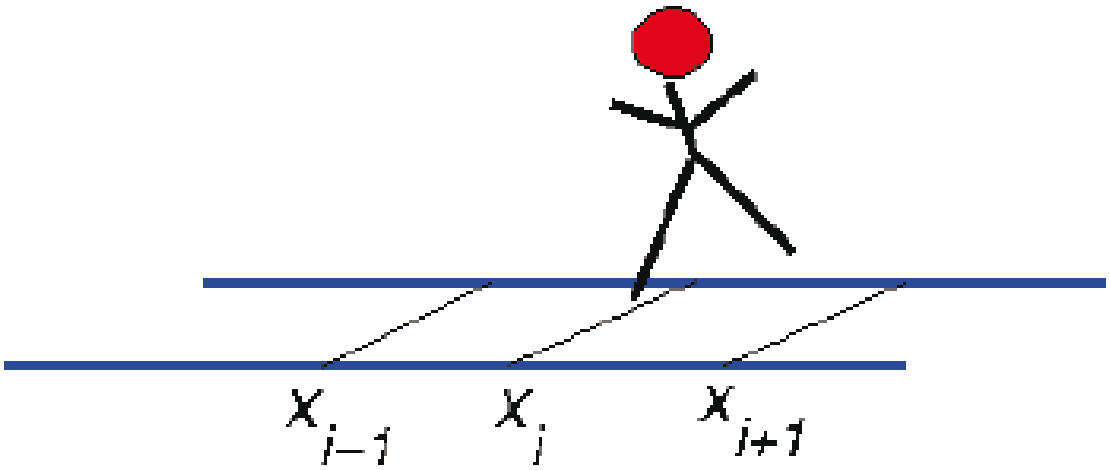
$$A = \begin{pmatrix} d_1 & s_1 & & & \\ a_2 & d_2 & s_2 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{N-1} & d_{N-1} & s_{N-1} \\ & & & a_N & d_N \end{pmatrix}$$

- Per risolvere sistemi tridiagonali un algoritmo molto efficiente è quello di Thomas

Esempio

Un ubriaco compie una **passeggiata casuale**, facendo un passo a sinistra o a destra a caso lungo una strada rettilinea. Quando raggiunge una estremità della strada, si ferma.

Calcolare la **probabilità** che l'ubriaco raggiunga l'estremità sinistra della strada partendo dalla posizione i.



La **probabilità** p_i, $i = 0, 1, ..., N$, di raggiungere l'estremo sinistro partendo dalla posizione i, soddisfa la relazione

$$\begin{cases} p_0 = 1 \quad p_N = 0 \\ \\ p_i = \frac{1}{2} p_{i-1} + \frac{1}{2} p_{i+1} \qquad i = 1, \ldots, N-1 \end{cases}$$

Si tratta di un **sistema lineare tridiagonale** nelle incognite

$$p_i,\ i = 1, \ldots, N-1$$

$$\begin{cases} p_1 - \frac{1}{2} p_2 = \frac{1}{2} \\ -\frac{1}{2} p_1 + p_2 - \frac{1}{2} p_3 = 0 \\ \cdots \quad \cdots \quad \cdots \quad \cdots \quad \cdots \quad \cdots \\ -\frac{1}{2} p_{N-3} + p_{N-2} - \frac{1}{2} p_{N-1} = 0 \\ -\frac{1}{2} p_{N-2} + p_{N-1} = 0 \end{cases}$$

Algoritmo di Thomas

- Sfrutta la struttura tridiagonale della matrice per calcolare in modo rapito la fattorizzazione LU della matrice A. Le matrici L,U che si ottengono risultano bidiagonali
  - i coefficienti incogniti si determinano imponendo l'uguaglianza $LU = A$

- Utilizza tali informazioni sulla struttura di L,U per risolvere efficientemente i due sistemi bidiagonali

$$Ly=b \; e \; Ux=y$$

Algoritmo di Thomas

$$A = \begin{bmatrix} d_1 & s_1 & 0 & \cdots & 0 \\ a_2 & d_2 & s_2 & \cdots & 0 \\ \cdots & \ddots & \ddots & \ddots & \cdots \\ 0 & 0 & \cdots & d_{n-1} & s_{n-1} \\ 0 & 0 & \cdots & a_n & d_n \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ \alpha_2 & 1 & 0 & \cdots & 0 \\ 0 & \alpha_3 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \alpha_n & 1 \end{bmatrix} \begin{bmatrix} u_1 & v_1 & 0 & \cdots & 0 \\ 0 & u_2 & v_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{n-1} & v_{n-1} \\ 0 & 0 & \cdots & 0 & u_n \end{bmatrix} = LU$$

Algoritmo di Thomas

$$
\begin{cases}
u_1 = d_1 & \\
v_i = s_i & i = 1, 2, \ldots, n-1 \\
\alpha_i = a_i / u_{i-1} & i = 2, 3, \ldots, n \\
u_i = d_i - \alpha_i \, v_{i-1} & i = 2, 3, \ldots, n
\end{cases}
$$

Soluzione del sistema lineare $AX = B$

$$
y_1 = b_1 \qquad y_i = b_i - \alpha_i \, y_{i-1} \qquad i = 2, 3, \ldots, n
$$

$$
x_n = y_n / u_n \qquad x_i = (y_i - v_i \, x_{i+1}) / u_i \qquad i = n-1, \ldots, 1
$$

Algoritmo di Thomas

- Scrivere una funzione Matlab che risolva un sistema lineare con **lo algoritmo di Thomas** e utilizzarla per risolvere il problema della **passeggiata casuale** per diversi valori di N
- Parametri di **input**:
  - A -> matrice dei coefficienti
  - b -> vettore dei termini noti
- parametri di **output**
  - x -> vettore soluzione
- prototipo della **function**

```
function x = algo_thomas(A,b)
```

Risultati Passeggiata aleatoria

N	||X-xtrue||	Tempo di calcolo	Occupazione di memoria
11	1.11e-16	0.000061 s	624 bytes
21	1.11e-16	0.000089 s	1376 bytes
51	4.33e-15	0.000179 s	3536 bytes
101	9.10e-15	0.000454 s	7136 bytes
501	4.61e-14	0.002566 s	35936 bytes
5001	3.20e-12	0.093172 s	359936 bytes
10001	6.95e-12	0.364083 s	719936 bytes

- Si osserva una notevole riduzione del tempo di calcolo, una discreta riduzione della memoria occupata e una maggiore precisione nella soluzione prodotta.