Università di Salerno
Dipartimento di
Ingegneria Industriale

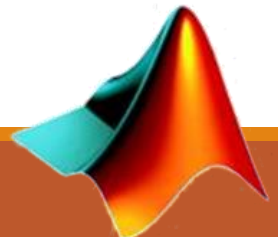
UNIVERSITÀ DEGLI STUDI DI SALERNO

Fondamenti di Informatica

MATLAB: Matrici e Operazioni su Matrici

Prof. Raffaele Pizzolante

A.A. 2016/17



MATLAB

MATLAB: Matrici e Operazioni su Matrici OUTLINE

- Matrici in MATLAB
  - Creazione di una matrice
  - Indici Matrici
  - Semplici Operazioni su Matrici
  - Operazioni Aritmetiche su Matrici

Matrici (Array 2D)

- Perché abbiamo bisogno delle matrici?

Impiegato	Stipendio annuale lordo	Giorni di vacanza	Giorni di malattia
Maria	12000	18	4
Antonio	21000	8	10
Francesco	30000	10	12
Chiara	60000	2	1

- Esistono numerosi scenari reali in cui le matrici vengono concretamente utilizzate
  - Elaborazione delle immagini
  - Gestione di dati tabellari
  - E molto altro ancora…

Matrici (Array 2D)

- Perché abbiamo bisogno delle matrici?

Impiegato	Stipendio annuale lordo	Giorni di vacanza	Giorni di malattia
Maria	12000	18	4
Antonio	21000	8	10
Francesco	30000	10	12
Chiara	60000	2	1

- Gli array (1D) che abbiamo visto finora possono essere visti come casi particolari di matrici (2D), con dimensione $1 \times n$ oppure $n \times 1$

Creare una Matrice – 1/4

- Possiamo **creare una matrice** digitando direttamente i suoi elementi in due modi diversi

- **Modo 1**

```
A = [2,4,10
     16,3,7]
```

Oppure

```
A = [2 4 10
     16 3 7]
```

- **Modo 2**

```
A = [2,4,10;16,3,7]
```

Oppure

```
A = [2 4 10;16 3 7]
```

RISULTATO:

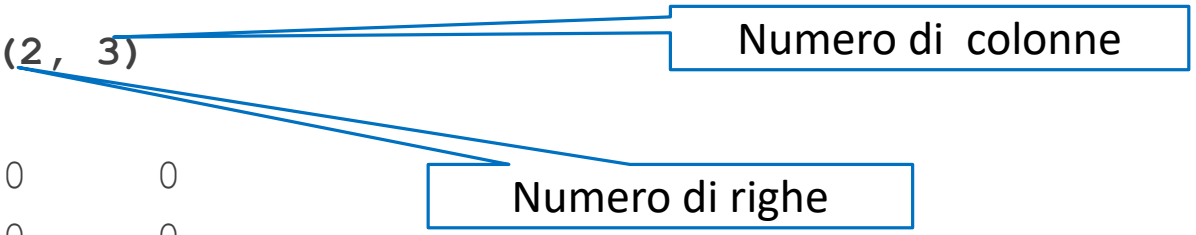
```
A =
     2     4    10
    16     3     7
```

- **OSSERVAZIONE IMPORTANTE**: Spazi e virgole separano gli elementi per colonne, mentre il punto e virgola separa gli elementi per riga

Creare una Matrice – 2/4

- **`zeros(nr, nc)`**
  - Crea una matrice **`nr`** (num. righe) x **`nc`** (num. colonne) composta da tutti **zero** (**0**)

```
>> x = zeros(2, 3)
   x =
       0     0     0
       0     0     0
```



- **`ones(nr, nc)`**
  - Crea una matrice **`nr`** x **`nc`** composta da tutti **uno** (**1**)

```
>> x = ones(2, 3)
   x =
       1     1     1
       1     1     1
```

Creare una Matrice – 3/4

- **eye(nr, nc)**
  - Crea una matrice identità **nr** x **nc**

```
>> x = eye(3, 3)
   x =
         1     0     0
         0     1     0
         0     0     1
```

- **rand(nr, nc)**
  - Crea una matrice **nr** x **nc** di **numeri pseudocasuali,** compresi **tra 0 e 1**

```
>> x = rand(2, 3)
   x =
      0.8147    0.1270    0.6324
      0.9058    0.9134    0.0975
```

Creare una Matrice – 4/4

- **`cat(n,A,B,C,..)`**
  - **Crea una nuova matrice** *concatenando* le matrici **`A, B, C,…`** lungo la dimensione **`n`**
    - Se **`n=1`**, la concatenazione è fatta per righe
    - Se **`n=2`**, la concatenazione è fatta per colonne

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

```
>> cat(1,A,B)
ans =
     1     2
     3     4
     5     6
     7     8
```

Equivalente a

```
>> [A; B]
```

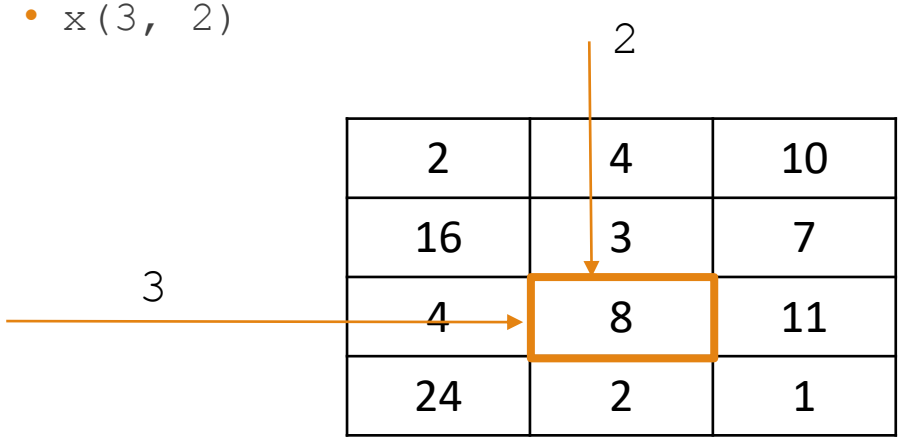
```
>> cat(2,A,B)
ans =
     1     2     5     6
     3     4     7     8
```

Equivalente a

```
>> [A, B]
```

Indicizzare una Matrice – 1/7

- **L'indicizzazione di una matrice è molto simile a quella di un array**
  - `x(r, c)` → `r` indica la riga, `c` indica la colonna
  - ***Esempio***
    - `x = [2,4,10; 16,3,7; 4,8,11; 24,2,1];`
    - `x(3, 2)`



Indicizzare una Matrice – 2/7

- **Indicizzare righe e colonne di una matrice**
  - `x(r, :)` → si riferisce all'intera riga `r`
  - `x(:, c)` → si riferisce all'intera colonna `c`

- ***Esempio (intera riga)***
  - `x(3, :)`

2	4	10
16	3	7
4	8	11
24	2	1

- ***Esempio (intera colonna)***
  - `x(:, 1)`

2	4	10
16	3	7
4	8	11
24	2	1

Indicizzare una Matrice – 2/7

- ***Esempio (intera matrice)***
  - `x(:, :)`

2	4	10
16	3	7
4	8	11
24	2	1

Indicizzare una Matrice – 3/7

- **Indicizzare sotto-array e sotto-matrici**
  - ***Esempio 1***
    - `x(3:4,2:3)`

2	4	10
16	3	7
4	8	11
24	2	1

  - ***Esempio 2***
    - `x(2:3, 2)`

2	4	10
16	3	7
4	8	11
24	2	1

Indicizzare una Matrice – 3/7

- **Indicizzare sotto-array e sotto-matrici**
  - ***Esempio 3***
    - `x(1:3, 3)`

2	4	10
16	3	7
4	8	11
24	2	1

  - ***Esempio 4***
    - `x(:, 1:2)`

2	4	10
16	3	7
4	8	11
24	2	1

Indicizzare una Matrice – 4/7

- Quindi, tramite l'operatore ***due punti*** **:** di MATLAB, è possibile selezionare all'interno di una matrice
  - Righe
  - Colonne
  - Sotto-array e sotto-matrici

Indicizzare una Matrice – 4/7

- È possibile indicizzare una matrice usando una singola coordinata

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

Esempi

```
>> A(2)
ans =
     4

>> A(3)
ans =
     7
```

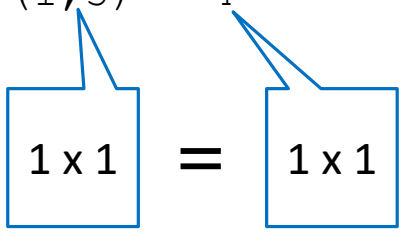
```
>> A(5)
ans =
     5

>> A(6)
ans =
     8
```

Indicizzare una Matrice – 5/7

- La dimensioni delle matrici a sinistra e a destra dell'assegnazione devono essere identiche

Esempio 1

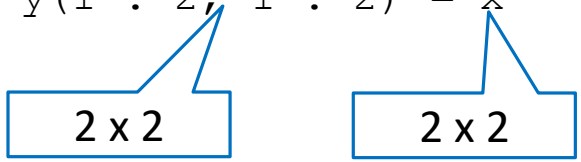
```
y = zeros(3,3);
y(1,3) = 4
```




```
y =
     0     0     4
     0     0     0
     0     0     0
```

Esempio 2

```
y = zeros(3,3);
x = ones(2,2);
y(1 : 2, 1 : 2) = x
```

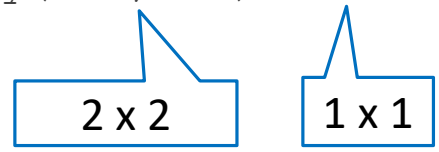



```
y =
     1     1     0
     1     1     0
     0     0     0
```

Indicizzare una Matrice – 6/7

- La dimensioni delle matrici a sinistra e a destra dell'assegnazione devono essere identiche, **a meno che** la matrice di destra non abbia dimensione *1* x *1*

```
y = zeros(3,3);

y(1:2,1:2) = 4
```

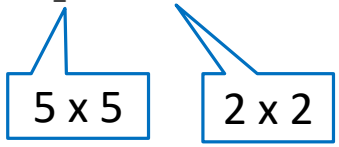



```
y =
     4     4     0
     4     4     0
     0     0     0
```

Indicizzare una Matrice – 6/7

- La dimensioni delle matrici a sinistra e a destra dell'assegnazione devono essere identiche, **a meno che** la matrice di destra non abbia dimensione *1* x *1*, **oppure** si sta sovrascrivendo una matrice

```
>> y = zeros(5,5);

>> x = ones(2,2);

>> y = x
```




```
y =

    1     1

    1     1
```

Indicizzare una Matrice – 7/7

- Le righe e le colonne di una matrice possono essere cancellate ponendole uguali all'array vuoto
- ***Esempio***

`A`

2	4	10
16	3	7
4	8	11
24	2	1

- `A(2,:)=[]`

2	4	10
~~16~~	~~3~~	~~7~~
4	8	11
24	2	1

2	4	10
4	8	11
24	2	1

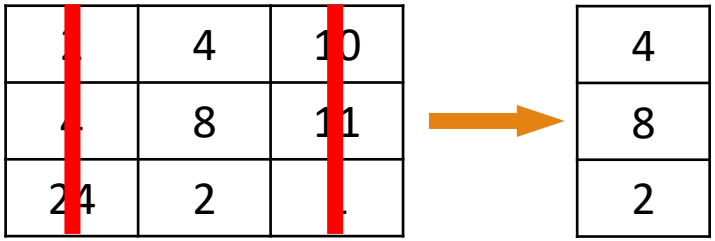
Indicizzare una Matrice – 7/7

- Le righe e le colonne di una matrice possono essere cancellate ponendole uguali all'array vuoto
- ***Esempio***

`A`

2	4	10
4	8	11
24	2	1

- `A(:,[1 3])=[]`



Semplici Operazioni su Matrici – 1/7

- **Trasposta di una matrice**

```
>> x =[ 2 4 6; 3 6 9]

x =

     2   4   6
     3   6   9


>> y = x '

y =

     2     3
     4     6
     6     9
```

Il simbolo ' è detto operatore di trasposizione

Semplici Operazioni su Matrici – 2/7
Massimo e Minimo

- **max(A)**
  - Trova il numero massimo in ogni colonna della matrice **A**

```
A =
    40    33    42
    48     2    47
>> m = max(A)
m =
    48    33    47
```

La funzione **min** funziona analogamente alla funzione **max**, ma restituisce il valore minimo

- È possibile ottenere il massimo elemento dell'intera matrice
  - Utilizzando la funzione **max** vista precedentemente per gli array, che restituisce l'elemento massimo in un dato array

```
>> max(max(A))
ans =
    48
```

Semplici Operazioni su Matrici – 4/7
Somma e Ordinamento

- **`sum(A)`**
  - Restituisce la somma degli elementi di **`A`**
    - Se **`A` è un array**, effettua la **somma gli elementi**
    - Se **`A` è una matrice**, restituisce (in un array) la **somma degli elementi di ciascuna colonna**
- **`sum(A, dim)`**
  - Somma gli elementi di **`A`** lungo la dimensione indicata da **`dim`**
    - Se **`dim=1`**, somma gli elementi per colonna
    - Se **`dim=2`**, somma gli elementi per riga
- **`sort(A)`**
  - Ordina ogni colonna della matrice **`A`** in maniera crescente

Semplici Operazioni su Matrici – 4/7

Somma e Ordinamento – Esempio *sum*

```
x =

    40    33    42
    48     2    47

>> sum(x, 1)

ans =
    88    35    89

>> sum(x, 2)

ans =

   115
    97
```

Semplici Operazioni su Matrici – 3/7
Media

- **`mean(A)`**
  - Trova la media di ogni colonna della matrice **`A`**

```
A =
    40    33    42
    48     2    47
>> mean(A)
ans =
   44.0000   17.5000   44.5000
```

  - Trova la media di ogni riga della matrice **`A`**

```
>> mean(A,2)
ans =
   38.3333
   32.3333
```

Semplici Operazioni su Matrici – 5/7
Dimensione e Numero di Elementi

- **`size(A)`**
  - Restituisce un array riga contenente le **dimensioni** (num. righe e num. colonne) della **matrice `A`**

```
A =
    40    33    42
    48     2    47
>> s = size(A)
s =
     2     3
```

Semplici Operazioni su Matrici – 5/7
Dimensione e Numero di Elementi

- **`size(A)`**
  - Restituisce un array riga contenente le **dimensioni** (num. righe e num. colonne) della **matrice `A`**

```
A =
    40    33    42
    48     2    47
>> s = size(A)
s =
     2     3
```

Numero di righe

Numero di colonne

Semplici Operazioni su Matrici – 5/7
Dimensione e Numero di Elementi

- **`length(A)`**
  - Restituisce **il numero di elementi** di **`A`**, se **`A` è un array**
  - Restituisce **il numero massimo tra righe e colonne**, se **`A` è una matrice**

- **`numel(A)`**
  - Restituisce il **numero di elementi** della matrice **`A`**

Semplici Operazioni su Matrici – 6/7
Ricerca

- **`find(A)`**
  - Restituisce un **array contenente gli indici degli elementi non nulli** della matrice **`A`**

```
>> A = zeros(4,4);
>> A(3,2)=5;
>> A(1,3)=2;
>> A(2,2)=7;
>> A
A =
     0     0     2     0
     0     7     0     0
     0     5     0     0
     0     0     0     0

>> find(A)
ans =
      6     7     9
```

Semplici Operazioni su Matrici – 6/7
Ricerca

- La funzione **`find`** può essere utilizzata mediante indicizzazione a due coordinate

```
>> x = [1 2 3; 7 8 9; 4 5 6]
x =
   1   2   3
   7   8   9     ← valore 7 (riga 2, colonna 1); valore 8 (riga 2, colonna 2); valore 9 (riga 2, colonna 3)
   4   5   6     ← valore 6 (riga 3, colonna 3);
>> [riga, colonna] = find(x > 5)
riga =
     2
     2
     2
     3
colonna =
     1
     2
     3
     3
```

Semplici Operazioni su Matrici – 6/7
Ricerca

- La funzione **find** può anche essere utilizzata mediante indicizzazione a singola coordinata

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

>> indici = find(x > 5)
indici =
     2
     5
     8
     9
```

Semplici Operazioni su Matrici – 7/7
Test di Non Nullità

- **`any(A)`**
  - Verifica **se ogni elemento** di **`A`** è **non nullo**
    - Restituisce 1 se almeno un elemento è non nullo, 0 altrimenti

```
A =
     0     0     2     0
     0     7     0     0
     0     5     0     0
     0     0     0     0

>> any(A)
ans =
     0     1     1     0
```

Semplici Operazioni su Matrici – 7/7
Test di Non Nullità

- **any(A, dim)**
  - Se **dim=1**, il comportamento della funzione è equivalente a quello di **any(A)**
  - Se **dim=2**, si verifica quanto segue

```
A =
     0     0     2     0
     0     7     0     0
     0     5     0     0
     0     0     0     0
>> any(A,2)
ans =
     1
     1
     1
     0
```

Operazioni Aritmetiche su Matrici – 1/4

- Le operazioni aritmetiche viste per gli array sono naturalmente estese alle matrici
  - ***Somma***

```
>> x =[ 2 4 6; 3 6 9];
>> x = x + x
x =
    4    8   12
    6   12   18
```

  - ***Sottrazione***

```
>> y = x - 1
y =
    3    7   11
    5   11   17
```

Operazioni Aritmetiche su Matrici – 1/4

- ***Problema:*** sottrarre 1 da tutti i valori in `y` che sono più grandi di 4, e memorizzare il risultato in `y` stessa

- ***Possibile soluzione***

```
>> indici = find(y > 4)

indici =

     2

     3

     4

     5

     6

>> y = y(2:6)-1

y =

     4     6    10    10    16
```

- **Soluzione più compatta**

```
>> y = y(find(y > 4))-1
y =
     4
     6
    10
    10
    16
```

Operazioni Aritmetiche su Matrici – 2/4

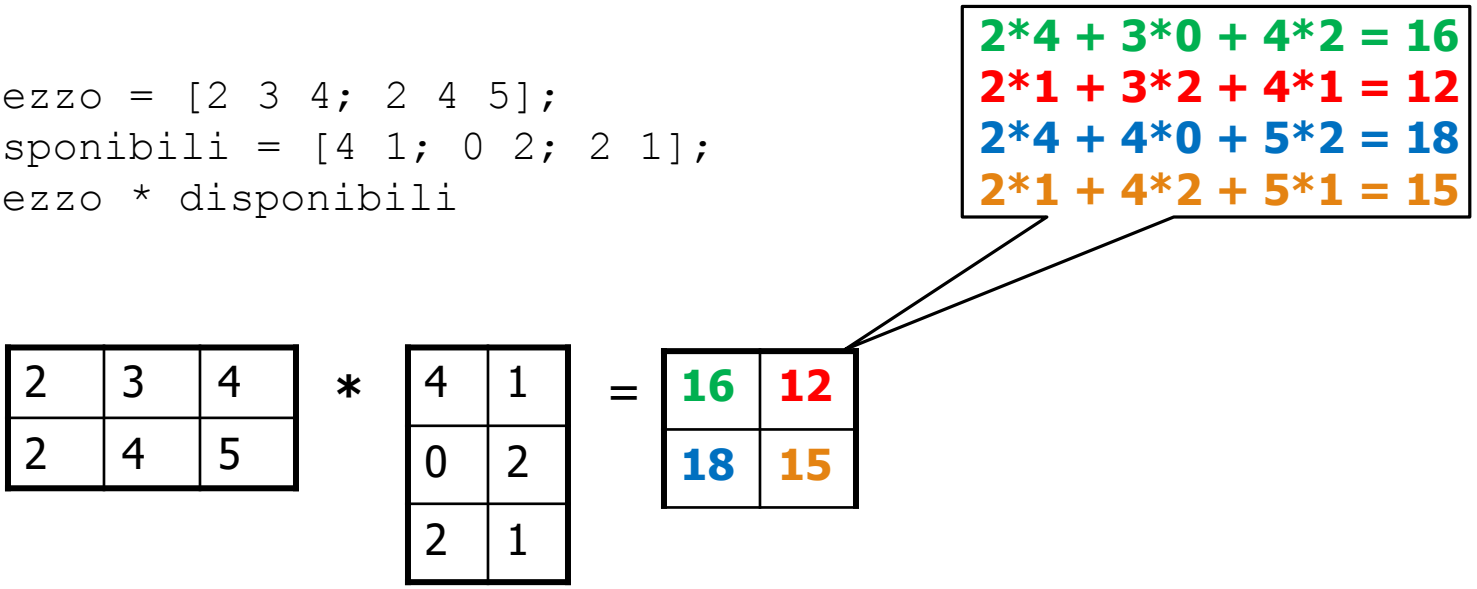
- ***Moltiplicazione scalare***

```
>> prezzo = [10 20 30; 2 3 4];
>> nuovo_prezzo = prezzo * 2
nuovo_prezzo =
     20    40    60     4     6     8
```

Operazioni Aritmetiche su Matrici – 3/4

- ***Moltiplicazione matriciale***

```
prezzo = [2 3 4; 2 4 5];
disponibili = [4 1; 0 2; 2 1];
prezzo * disponibili
```

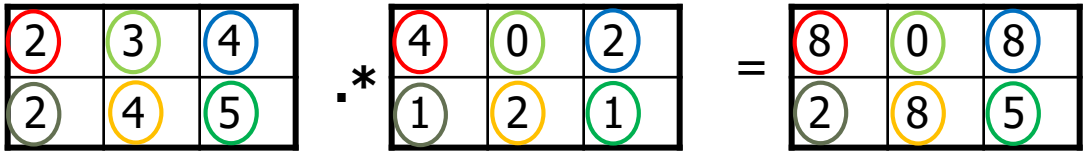



```
ans =

    16    12
    18    15
```

Operazioni Aritmetiche su Matrici – 4/4

- ***Moltiplicazione elemento per elemento***

```
prezzo = [2 3 4; 2 4 5];
disponibili = [4 1; 0 2; 2 1];
prezzo .* disponibili
```




```
ans =

     8     0     8
     2     8     5
```

Riferimenti

- Capitolo 2
  - Paragrafi 1, 2, 3, 4 e 8
  - **No** Esempio 2.1, **No** Esempio 2.2, **No** Esempio 2.4, **No** Esempio 2.5