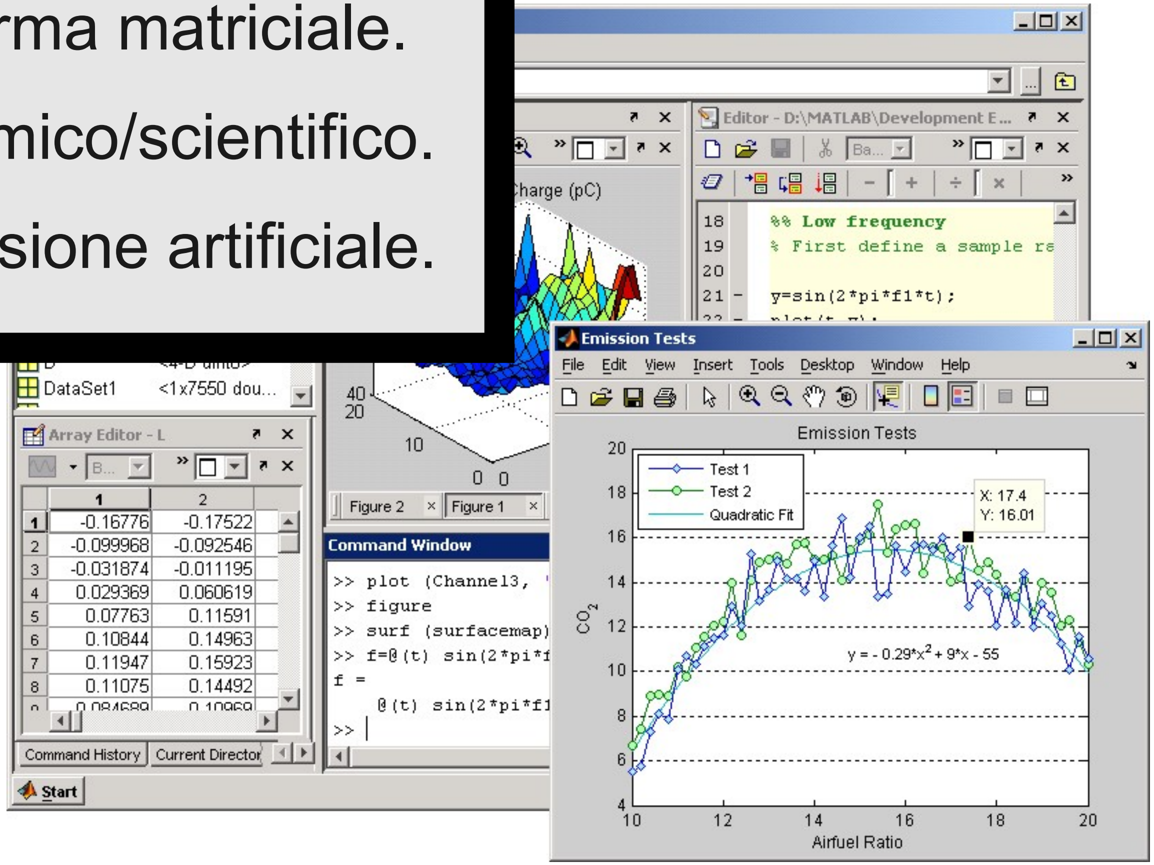
Corso di Visione Artificiale

Matlab per Visione

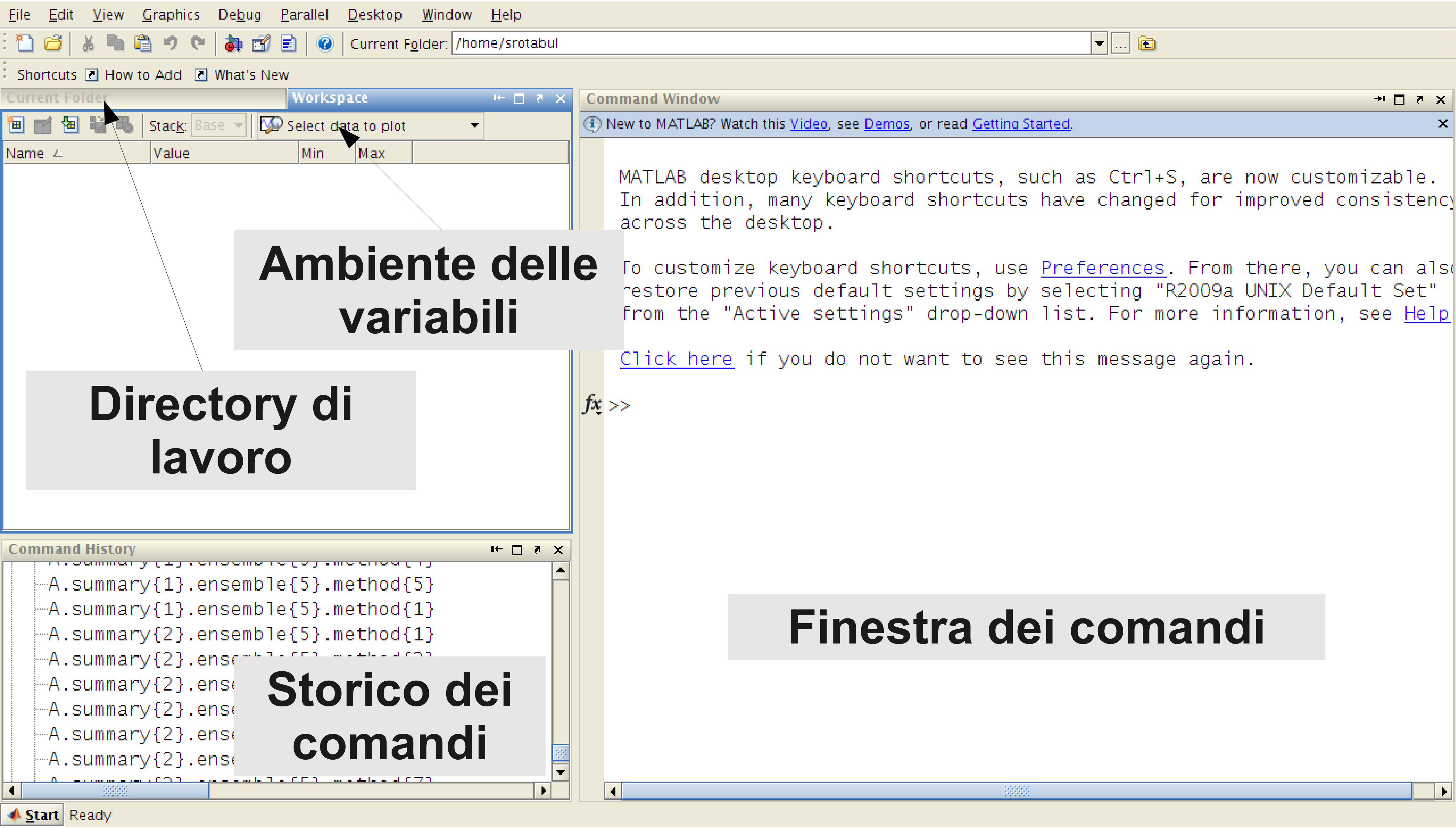
Samuel Rota Bulò

Cos'è Matlab?

- **MATLAB** - MATrix LABoratory
  - Ambiente di sviluppo ed esecuzione
  - Linguaggio di programmazione (principalmente vettoriale)
- Manipolazione dei dati in forma matriciale.
- Diffuso in ambiente accademico/scientifico.
- Largamente utilizzato per visione artificiale.



Il desktop di Matlab



L'HELP di Matlab

- Matlab mette a disposizione un'ottima documentazione.
- Dal menu Help|Product Help
- Se vogliamo ottenere informazioni riguardo ad un certo comando possiamo invocare

```
>> help comando
```

per ottenere la documentazione testuale del comando nella finestra dei comandi

```
>> doc comando
```

per aprire una finestra con la documentazione più dettagliata

Matrici

- Una matrice è una “griglia” bidimensionale di numeri, caratterizzata da un numero M di righe e N di colonne.

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M,1} & a_{M,2} & \cdots & a_{M,N} \end{bmatrix}$$

- In Matlab creiamo una matrice racchiudendo una sequenza di numeri tra parentesi quadre [ ]. Gli elementi di una riga sono separati da virgola o spazio. Le righe sono separate da ; o un invio.

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{bmatrix}$$

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

Scalari e Vettori

- Un **numero scalare** è una matrice 1x1. In Matlab scrivendo un numero automaticamente creiamo una matrice 1x1.

```
>> 1
ans =
      1
```

- Un **vettore riga** è una matrice 1xN.

$$\begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}$$

```
>> [1 2 3 4]
ans =
      1     2     3     4
```

- Un vettore colonna è una matrice Mx1.

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

```
>> [1; 2; 3]
ans =
      1
      2
      3
```

Variabili

- Le matrici che abbiamo creato finora sono anonime. Per assegnare un nome ad una matrice scriviamo il nome della variabile seguito da = e la matrice.

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{bmatrix}$$

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

- Ora la variabile A è nell'ambiente delle variabili e può essere riutilizzata. Ogni volta che scriviamo la variabile A è come se avessimo scritto la matrice associata.

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

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

- Se assegniamo ad una variabile un nuovo valore quello precedente verrà perso.

Variabili

- Ogni assegnamento di variabile produce implicitamente un output. Per sopprimerlo basta terminare l'assegnamento con un ;

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

- Anche quando scriviamo un matrice senza assegnarla ad una variabile, in realtà, matlab effettua un assegnamento implicito di variabile. Infatti crea una variabile chiamata **ans**

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

```
>> [1 2 3; 4 5 6];
>> ans
     1     2     3
     5     6     7
```

- Possiamo eliminare variabili dall'ambiente con la funzione **clear**
  - clear elimina tutte le variabili nell'ambiente
  - clear A B C … elimina tutte le variabili elencate.

Tipi di dato primitivi

Tipo	Descrizione
double	Floating-points in precisione doppia (8 bytes)
uint8	Intero a 8-bit senza segno (1 byte)
uint16	Intero a 16-bit senza segno (2 bytes)
uint32	Intero a 32-bit senza segno (4 bytes)
int8	Intero a 8-bit con segno (1 byte)
int16	Intero a 16-bit con segno (2 bytes)
int32	Intero a 32-bit con segno (4 bytes)
single	Floating-points in precisione singola (4 bytes)
char	Caratteri (2 bytes). Es: 'a', 'b', 'c', ...
logical	Valori booleani 0/1 (1 byte)

- Possiamo avere matrici di diverso tipo.
- Il nome di ciascun tipo può essere utilizzato come funzione di conversione di tipo:

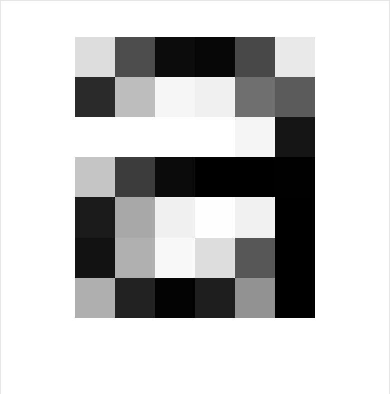
```
>> A=[1.1 2.5 3.6 0];
>> uint8(A)
ans =
    1    3    4    0
>> logical(A)
ans =
     1     1     1     0
```

- Le stringhe in Matlab sono sequenze di caratteri tra apici e corrispondono a vettori riga di tipo char.

```
>> 'prova'
ans=
prova
>> ['p' 'r' 'o' 'v' 'a']
ans=
prova
```

Immagini digitali

- Un'immagine (di intensità) in Matlab è rappresentata da una matrice, dove il primo pixel ha coordinate (1,1).
- Se la matrice è di tipo **uint8**/**uint16**/**uint32**, allora l'immagine si intende a 8/16/32 bit.
- Se la matrice è di tipo **double**/**single** allora l'immagine è considerata a valori reali nell'intervallo [0,1]. Quindi intensità >1 sono bianche, <0 sono nere.
- Se la matrice è **logica**, l'immagine si intende in bianco e nero.



255	255	255	255	255	255	255	255	255	255
255	255	221	77	12	7	72	233	255	255
255	255	42	189	246	240	111	91	255	255
255	255	255	255	255	255	245	21	255	255
255	255	197	60	11	0	0	1	255	255
255	255	27	168	240	254	241	0	255	255
255	255	18	176	248	221	87	0	255	255
255	255	175	34	3	30	146	0	255	255
255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255

Immagini digitali

- Per caricare un'immagine da file utilizziamo la funzione **imread**.

```
>> I=imread('filename');
```

- Per visualizzare un'immagine utilizziamo la funzione **imshow**.

```
>> imshow(I)
```

- Per salvare un'immagine su file utilizziamo la funzione **imwrite**.

```
>> imwrite(I,'filename')
```

- Per ottenere le dimensioni di una qualunque matrice (e quindi anche immagini) utilizziamo la funzione **size**:

```
>> A=[1 2 3; 4 5 6];
>> size(A)
ans =
     2     3
```

```
>> [M,N]=size(A)
M=
     2
N=
     3
```

Conversioni di tipo per immagini

Tipo	Descrizione
im2uint8	Converte un'immagine qualsiasi in un'immagine a 8-bit
im2uint16	Converte un'immagine qualsiasi in un'immagine a 16-bit
mat2gray	Converte una matrice double in un'immagine a valori reali, ovvero una matrice double on valori in [0,1]
im2double	Converte un'immagine qualsiasi in un'immagine a valori reali.
im2bw	Converte un'immagine qualsiasi in un'immagine in bianco e nero.

Funzioni per creazione di matrici

Comando	Descrizione
`a:b` `a:i:b`	[a a+1 a+2 … a+n] dove n=floor(b-a) [a a+i a+2i … a+ni] dove n=floor((b-a)/i)
`linspace(a,b,n)`	a:(b-a)/(n-1):b
`zeros(M,N)`	Crea una matrice MxN di zeri.
`ones(M,N)`	Crea una matrice MxN di 1
`eye(M,N)`	Crea una matrice MxN di zeri con 1 sulla diagonale. Se M=N abbiamo la matrice identica
`rand(M,N)`	Crea una matrice MxN di valori random con distribuzione uniforme tra 0 e 1
`randn(M,N)`	Crea una matrice MxN di valori random con distribuzione normale di media 0 e varianza 1.
`true(M,N)`	Crea una matrice logica MxN di 1
`false(M,N)`	Crea una matrice logica MxN di 0

Indicizzazione di vettori

- Sia v un vettore e i l'indice di un elemento di v. Con v(i) selezioniamo l'i-esimo elemento di v.

```
>> v=[2 4 6 8];
>> v(2)
ans=
     4
```

- Ora, i non deve necessariamente essere uno scalare, ma può essere un vettore di indici. In questo caso, v(i) rappresenta il sotto-vettore di v composto dai soli elementi di v indicizzati da i.

```
>> v([1 3 4])
ans =
     2     6     8
```

- Tra gli indici utilizzabili per indicizzare un elemento di un vettore c'è **end** che rappresenta l'indice dell'ultimo elemento.

```
>> v(2:end)
ans =
     4     6     8
```

```
>> v([1 end-1])
ans =
     2     6
```

```
>> v(1:2:end)
ans =
     2     6
```

Indicizzazione di vettori

- Sia v un vettore. Con v(:) otteniamo v come vettore colonna.

```
>> v=[2 4];
>> v(:)
ans=
     2
     4
```

```
>> v=[2; 4];
>> v(:)
ans=
     2
     4
```

- L'indicizzazione di un vettore non è solo utile in lettura ma anche in scrittura.

```
>> v=[1 2 3 4];
>> v(1)=0
v =
     0     2     3     4
>> v(2:end)=[2 1 0]
v =
     1     2     1     0
```

```
>> v=[1 2 3 4];
>> v([1 end])=1
v =
     1     2     3     1
```

```
>> v=[1 2];
>> v(4)=5
v =
     1     2     0     5
```

- Se scriviamo in una locazione che è fuori dal vettore, questo viene ridimensionato automaticamente inizializzando a 0 i nuovi elementi.

Indicizzazione di matrici

- Sia M una matrice e i,j le coordinate di un elemento di M. Con M(i,j) selezioniamo l'(i,j)-esimo elemento di M.

```
>> M=[1 2 3; 4 5 6];
>> M(1,2)
ans=
     2
```

- Come abbiamo visto per i vettori, se gli indici i,j sono vettori, M(i,j) seleziona una sotto-matrice di M

```
>> M=[1 2 3; 4 5 6];
>> M(1:end,2:3)
ans=
     2     3
     5     6
>> M(:,end)
ans=
     3
     6
```

```
>> M=[1 2 3; 4 5 6];
>> M(1,:)
ans=
     1     2     3
>> M(:,:)
ans=
     1     2     3
     4     5     6
```

- Possiamo usare : invece di 1:end

Indicizzazione di matrici

- Come abbiamo visto per i vettori, l'indicizzazione delle matrici è utile anche in scrittura.

```
>> M=[1 2 3; 4 5 6];
>> M(1,2)=0
M=
     1     0     3
     4     5     6
```

```
>> M=[1 2 3; 4 5 6];
>> M(:,[1 3])=[0 1; 1 0]
M=
     0     2     1
     1     5     0
```

- Un altro sistema di indicizzazione utilizza matrici logiche. Se abbiamo una matrice M e una matrice logica D della stessa dimensione di M con M(D) indicizziamo gli elementi corrispondenti a 1 in D.

```
>> M=[1 2 3; 4 5 6];
>> D=logical([1 0 1; 0 1 0]);
>> M(D)
ans=
     1
     5
     3
```

```
>> M=[1 2 3; 4 5 6];
>> D=logical([1 0 1; 0 1 0]);
>> M(D)=0
M=
     0     2     0
     4     0     6
```

- M(D) genera un vettore colonna selezionando gli elementi colonna per colonna.

Indicizzazione di matrici

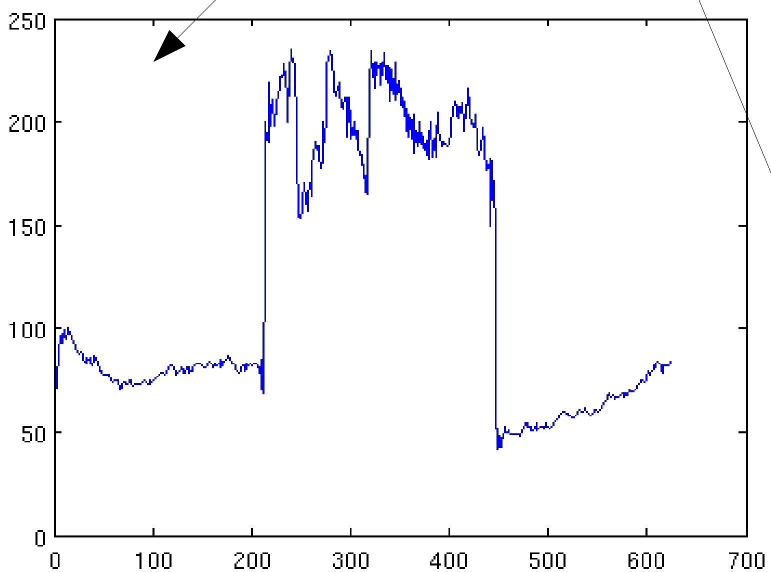
- Possiamo infine linearizzare una matrice M in un vettore colonna scrivendo M(:)

```
>> M=[1 2 3; 4 5 6];
>> M(:)
ans=
     1
     4
     2
     5
     3
     6
```

Di nuovo gli elementi sono selezionati colonna per colonna.

Immagini e indicizzazione

```
>> I=imread('flower.jpg');
>> imshow(I);
>> imshow(I(end:-1:1,:));
>> [M,N]=size(I);
>> ir=50:M-50;
>> ic=50:N-50;
>> imshow(I(ir,ic));
>> ir=1:2:M;
>> ic=1:2:N;
>> imshow(I(ir,ic));
>> plot(I(M/2,:))
```



Addizione

- Siano A=(a_{ij}) ,B=(b_{ij}) matrici MxN
- Sia c uno scalare.

```
>> A+B
```

$$(A+B)_{ij} = a_{ij} + b_{ij}$$

```
>> c+A
```

$$(c+A)_{ij} = c + a_{ij}$$

```
>> A+c
```

$$(A+c)_{ij} = a_{ij} + c$$

Sottrazione

- Siano A=(a_{ij}) ,B=(b_{ij}) matrici MxN
- Sia c uno scalare.

```
>> A-B
```

$$(A-B)_{ij} = a_{ij} - b_{ij}$$

```
>> c-A
```

$$(c-A)_{ij} = c - a_{ij}$$

```
>> A-c
```

$$(A-c)_{ij} = a_{ij} - c$$

Moltiplicazione puntuale

- Siano A=(a_{ij}) ,B=(b_{ij}) matrici MxN
- Sia c uno scalare.

```
>> A.*B
```

$$(A.*B)_{ij} = a_{ij} \cdot b_{ij}$$

```
>> c.*A
```

$$(c.*A)_{ij} = c \cdot a_{ij}$$

```
>> A.*c
```

$$(A.*c)_{ij} = a_{ij} \cdot c$$

Moltiplicazione

- Sia A=(a_{ij}) matrice MxN
- Sia B=(b_{ij}) matrice NxL
- Sia c uno scalare.

```
>> A*B
```

$$(A * B)_{ij} = \sum_{k=1}^{N} a_{ik} \cdot b_{kj}$$

```
>> c*A
```

$$(c * A)_{ij} = c \cdot a_{ij}$$

```
>> A*c
```

$$(A * c)_{ij} = a_{ij} \cdot c$$

Trasposizione

- Sia A=(a_{ij}) matrice MxN

```
>> A.'
```

$$(A.')_{ij} = a_{ji}$$

```
>> A'
```

$$(A')_{ij} = \overline{a}_{ji}$$

- coniugato complesso
- nota che

$$\overline{a}_{ji} = a_{ji} \text{ se } a_{ji} \in \mathbb{R}$$

Inversione

- Sia A=(a$_{ij}$) matrice NxN

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

$$inv(A) = A^{-1}$$

$$A^{-1}A = I$$

$$AA^{-1} = I$$

Divisione (a destra) puntuale

- Siano A=(a_{ij}), B=(b_{ij}) matrici MxN
- Sia c uno scalare.

```
>> A./B
```

$$(A./B)_{ij} = \frac{a_{ij}}{b_{ij}}$$

```
>> c./A
```

$$(A./c)_{ij} = \frac{c}{a_{ij}}$$

```
>> A./c
```

$$(A./c)_{ij} = \frac{a_{ij}}{c}$$

Elevamento a potenza puntuale

- Siano A=(a$_{ij}$) e B=(b$_{ij}$) matrice MxN
- Sia c uno scalare.

```
>> A.^c
```

$$(A.\hat{}c)_{ij} = a_{ij}^{c}$$

```
>> A.^B
```

$$(A.\hat{}B)_{ij} = a_{ij}^{b_{ij}}$$

Operazioni con matrici

Comando	Descrizione
`-A`	Nega gli elementi della matrice A
`A'`	Trasposta di A
`A*B a*B B*a`	Moltiplicazione tra matrici, matrice/scalare
`A.*B a.*B B.*a`	Moltiplicazione puntuale di matrici, matrice/scalare
`A/B a/B B/a`	Divisione a destra tra matrici, matrice/scalare
`A./B a./B B./a`	Divisione a destra puntuale tra matrici, matrice/scalare
`A\B a\B B\a`	Divisione a sinistra tra matrici, matrice/scalare
`A.\B a.\B B.\a`	Divisione a sinistra puntuale tra matrici, matrice/scalare
`A+B a+B B+a`	Addizione tra matrici, matrice/scalare
`A-B a-B B-a`	Sottrazione di matrici, matrice/scalare
`a^B B^a`	Elevamento a potenza matrice/scalare
`a.^B B.^a`	Elevamento puntuale di potenza matrice/scalare

NOT

- Sia A=(a$_{ij}$) matrice logica MxN

```
>> ~A
```

$$(\sim A)_{ij} = \neg a_{ij}$$

AND

Siano A=(a_{ij}) e B=(b_{ij}) matrici logiche MxN

```
>> A&B
```

$$(A\&B)_{ij} = a_{ij} \wedge b_{ij}$$

OR

Siano A=(a_{ij}) e B=(b_{ij}) matrici logiche MxN

```
>> A|B
```

$$(A|B)_{ij} = a_{ij} \vee b_{ij}$$

Operatori relazionali

- Siano A=(a_{ij}), B=(b_{ij}) matrici MxN
- Sia c uno scalare.
- Gli operatori relazionali danno origine a matrici logiche.

```
>> A==B
>> A~=B
>> A>B
>> A>=B
>> A<B
>> A<=B
```

```
>> c==A
>> c~=A
>> c>A
>> c>=A
>> c<A
>> c<=A
```

```
>> A==c
>> A~=c
>> A>c
>> A>=c
>> A<c
>> A<=c
```

if,elseif,else

```
if espressione1
    ...
elseif espressione2
    ...
elseif ...
    ...
else
    ...
end
```

for e while

```
for indice=vettore
   ...
end

while espressione
   ...
end
```

funzioni

```
function [r1,r2,..]=nome(a1,a2,...)
   ...
end
```