



Introduzione a Matlab

Informatica B AA 2017/2018

Giacomo Boracchi

giacomo.boracchi@polimi.it

6 Novembre 2017



Il Matlab

MATrix LABoratory



Cos'è Matlab (MATrix LABoratory):

- Ambiente di sviluppo e un linguaggio di programmazione per calcolo numerico
- È pensato (e ottimizzato) per operare su matrici (ma include generiche funzionalità matematiche)

Lo utilizzerete nei successivi corsi di calcolo numerico

MATLAB è uno strumento commerciale, su licenza NON gratuita,

- Student edition fornita dal Politecnico (maggiori dettagli a laboratorio)
- Dal 2015 Matlab utilizzabile anche fuori della rete polimi



Screenshot interfaccia MATLAB

The screenshot shows the MATLAB R2015b interface with several components and annotations:

- Toolbar:** Located at the top, containing icons for file operations (New, Open, Save, Find Files, Compare, Print), navigation (Go To, Find), editing (Insert, Comment, Indent), and execution (Run, Run and Advance, Run Section, Advance, Run and Time). An arrow points to the Run and Time icon with the text "Lancia i tool di MATLAB ed altro...".
- Current Folder:** A file browser on the left showing a directory with files like `calcolaSommaDivisori.m`, `cerca.m`, `inserisciRilievo.m`, `rilevamenti.m`, `scacchiera.m`, `sommaCompresi.m`, and `strLenRic.m`. An arrow points to this window with the text "Contenuto della directory corrente".
- Editor:** The central window showing MATLAB code. The code includes a loop that calls `inserisciRilievo()` and calculates the mean of heights. An arrow points to the code with the text "Codice nell'Editor".
- Command Window:** On the right, showing the execution of `a = 1321` and the resulting value `1321`. An arrow points to the output with the text "Finestra dei comandi".
- Workspace:** At the bottom left, showing a table with the variable `a` and its value `1321`. An arrow points to this window with the text "Mostra le variabili nel workspace".

Lancia i tool di MATLAB ed altro...

Contenuto della directory corrente

Codice nell'Editor

Mostra le variabili nel workspace

Finestra dei comandi



Esiste una soluzione alternativa: Octave

- identico nella concezione, molto simile per gli aspetti operativi
- disponibile gratuitamente, vedi www.gnu.org/software/octave/
- Occorre installare un'interfaccia grafica qtoctave
- Vedrete tutto a laboratorio



Screenshot dell'interfaccia OCTAVE

```
Octave
GNU Octave, version 3.0.0
Copyright (C) 2007 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE.  For details, type `warranty'.

Octave was configured for "i686-pc-msdosmsvc".

Additional information about Octave is available at http://www.octave.org.

Please contribute if you find this software useful.
For more information, visit http://www.octave.org/help-wanted.html

Report bugs to <bug@octave.org> (but first, please read
http://www.octave.org/bugs.html to learn how to write a helpful report).

For information about changes from previous versions, type `news'.

- Use `pkg list' to see a list of installed packages.
- SciTE editor installed. Use `edit' to start the editor.
- MSYS shell available (C:\Programmi\Octave\msys).
- Graphics backend: jhandles.

octave-3.0.0.exe:1>
```

Linea di comando dell'interprete



Screenshot dell'interfaccia grafica QtOctave

The screenshot shows the QtOctave application window titled "Virtual Appliance [Running]" and "QtOctave [Empty]". The interface includes a menu bar (File, View, Analysis, Data, Equations, Matrix, Plot, Statistics, Config, Help), a toolbar with various icons, and a main workspace area. On the left side, there are three panels: "Variables' List", "Commands' List", and "Navigator".

Mostra le variabili nel workspace: An arrow points to the "Variables' List" panel, which shows a table with columns "Name", "Size", and "Bytes". A single entry "Local..." is visible.

Finestra dei comandi: An arrow points to the main workspace area, which displays the Octave terminal output. The text includes: "Eaton and others.", "the source code for copying conditions.", "WARRANTY; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. For details, type 'warranty'.", "Octave was configured for 'i486-pc-linux-gnu'.", "Additional information about Octave is available at <http://www.octave.org>.", "Please contribute if you find this software useful.", "For more information, visit <http://www.octave.org/help-wanted.html>", "Report bugs to <bug@octave.org> (but first, please read <http://www.octave.org/bugs.html> to learn how to write a helpful report).", "For information about changes from previous versions, type 'news'.", and ">>>>".

Contenuto della directory corrente: An arrow points to the "Navigator" panel, which shows the current directory path "/home/infob" and a list of files and folders:

Name	Size	Type
projects		Folder
Desktop		Folder

The main workspace area has a prompt "ne>>" followed by a text input field containing "Insert your commands here...".



Caratteristiche del linguaggio di Matlab (1)

Linguaggio di alto livello

- Simile a linguaggi di programmazione C, Java, Pascal
- Possiede comandi sintetici per effettuare complesse elaborazioni numeriche

Linguaggio interpretato, i comandi e istruzioni

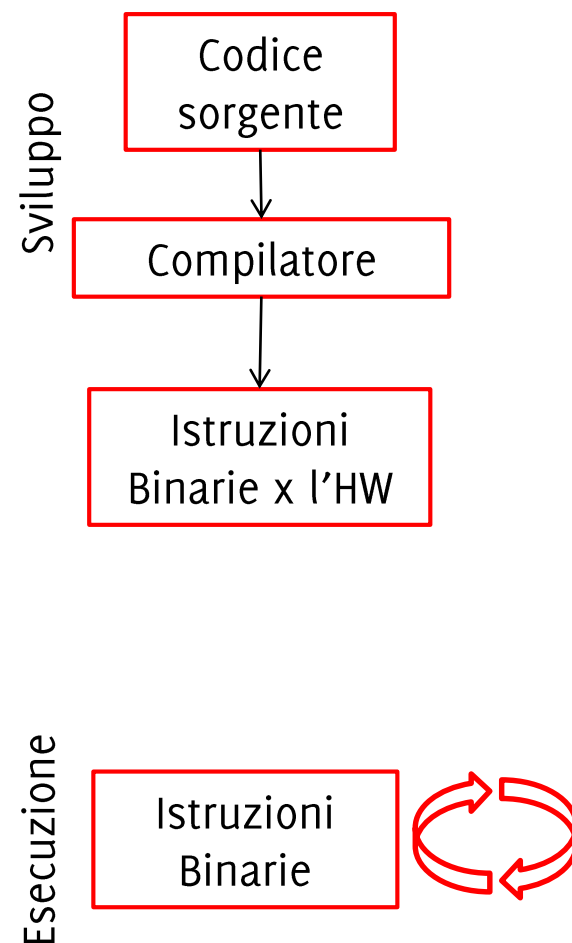
- NON sono tradotti in codice eseguibile dall'hardware
- Ma invia istruzioni ad un altro programma, **l'interprete**, che li analizza ed esegue azioni da essi descritte



Linguaggi Compilati vs Interpretati

Linguaggi **Compilati**:

- Il compilatore è un programma che **traduce** le istruzioni del codice sorgente in codice macchina (in binario)
- L'esecuzione del programma non richiede la presenza del codice sorgente, né del compilatore.
- I programmi sono efficienti
- Il programma è facilmente portabile su piattaforme analoghe

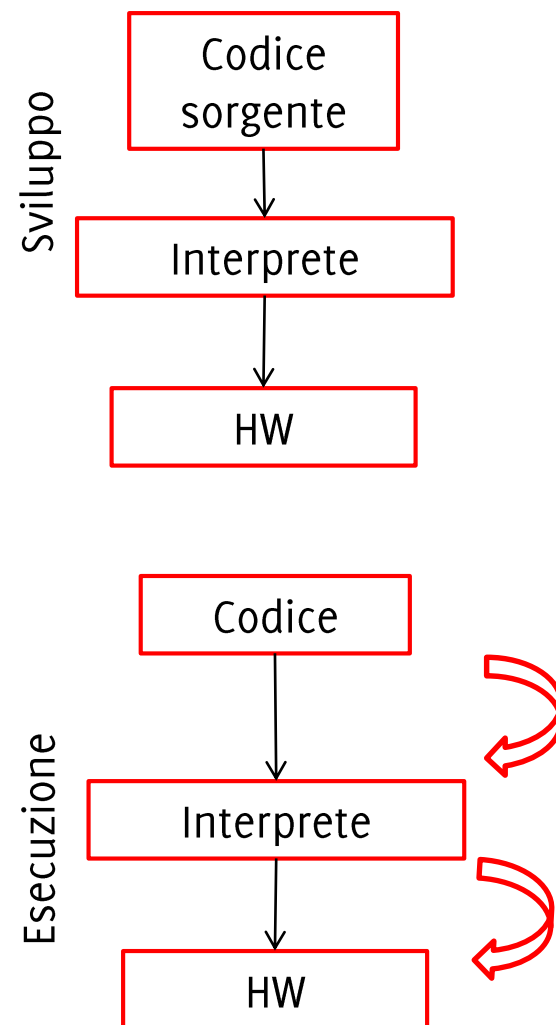




Linguaggi Compilati vs Interpretati

Linguaggi Interpretati:

- L'interprete è un programma che **esegue** istruzioni contenute nel codice sorgente
- L'esecuzione del programma richiede la presenza del codice (talvolta il sorgente) e dell'interprete.
- I programmi sono meno efficienti di quelli compilati
- Portabilità meno pratica
- Sviluppo più facile: è possibile eseguire le istruzioni mentre si scrive il codice sorgente





Caratteristiche del linguaggio di Matlab (2)

Linguaggio dinamico (non tipizzato)

- **NON occorre dichiarare le variabili**
 - risultano definite al primo assegnamento
 - vengono incluse in una struttura detta *tabella dei simboli*
- **il tipo delle variabili è *dinamico***
 - a una variabile si possono assegnare, successivamente, valori di tipo diverso (scalari, stringhe, vettori, matrici...)



Nella pratica Matlab...

...può far riferimento a 3 cose diverse:

- Il linguaggio Matlab che utilizziamo per codificare i programmi
- L'interprete Matlab che viene invocato per eseguire i nostri programmi
- L'ambiente di sviluppo integrato (IDE) che permette di scrivere ed eseguire i programmi e di interfacciarsi con la command window



Le Istruzioni



Le Istruzioni e la Command Window

Le **istruzioni** possono essere **inviare** direttamente **all'interprete** se scritte nella command window (dopo il simbolo `>>`)

- La command window è come una «super calcolatrice»
- La command window ha un'interfaccia testuale che inizia con `>>`



Screenshot interfaccia MATLAB

MATLAB R2015b - academic use

HOME PLOTS APPS EDITOR PUBLISH VIEW

New Open Save Find Files Compare Go To Print Find Breakpoints Run Run and Advance Run Section Advance Run and Time

C:\Users\Boracchi\Documents\MATLAB

Current Folder

- calcolaSommaDivisori.m
- cerca.m
- inserisciRilievo.m
- rilevamenti.m
- scacchiera.m
- sommaCompresi.m
- strLenRic.m

Details

Select a file to view details

Workspace

Name	Value
a	1321

```
1 n = input('inserire nr rilievi');
2
3 for ii = [1 : n]
4     rilievo(ii) = inserisciRilievo();
5 end
6
7 % atezza media rilievi con
8 % lat tra 20 e 60 e long tra 30 e 40
9
10 altezze = [];
11 for ii = [1 : n]
12     if (rilievo(ii).lat > 20 && rilievo(ii).
13         && rilievo(ii).lon > 30 && rilie
14         altezze = [altezze, rilievo(ii).alt]
15     end
16 end
17
18 altezzaMedia = mean(altezze)
19
20 %%
21 LAT = [rilievo.lat];
22 LON = [rilievo.lon];
23 ALT = [rilievo.alt];
24
25 indx = find(LAT > 20 & LAT < 60 & LON > 30 &
```

Command Window

```
-----
Your MATLAB license will expire
Please contact your system admin
MathWorks to renew this license.
-----

Academic License

>> a = 1321

a =

    1321

fx >>
```

Finestra dei comandi



Esempio: le operazioni aritmetiche

Nella command window è possibile eseguire operazioni aritmetiche

```
>> 5 + 7
```

```
ans =
```

```
12
```

```
>> 5 / 7
```

```
ans =
```

```
0.7143
```

ans è una variabile «di default» che contiene il risultato di un'istruzione che sia un assegnamento



Esempio: le operazioni aritmetiche

Nella command window è possibile eseguire operazioni aritmetiche

```
>> 5 + 7
```

```
ans =
```

```
12
```

```
>> 5 / 7
```

```
ans =
```

```
0.7143
```

```
>> 5 * 7
```

```
ans =
```

```
35
```

```
>> 'a' + 2
```

```
ans =
```

```
99
```

```
>> 5 ^ 7
```

```
ans =
```

```
78125
```

Elevamento a potenza

I caratteri alfanumerici si indicano con l'apice singolo: sono sempre legati agli interi mediante la tabella ASCII



Istruzioni e Codice Sorgente

Le istruzioni possono essere contenute in un **file sorgente**, in particolare:

- uno script
- una funzione

e quindi eseguite in maniera sequenziale.

L'esecuzione di uno codice sorgente può essere visto come l'inserimento delle varie istruzioni nella command window.



Screenshot interfaccia MATLAB

MATLAB R2015b - academic use

HOME PLOTS APPS EDITOR PUBLISH VIEW

Insert Comment Indent Breakpoints Run Run and Advance Run and Time

FILE NAVIGATE EDIT BREAKPOINTS RUN

C:\Users\Boracchi\Documents\MATLAB

Current Folder

- calcolaSommaDivisori.m
- cerca.m
- inserisciRilievo.m
- rilevamenti.m
- scacchiera.m
- sommaCompresi.m
- strLenRic.m

Details

Select a file to view details

Workspace

Name	Value
a	1321

Editor - C:\Users\Boracchi\Documents\MATLAB\rilevamenti.m

```
1 n = input('inserire nr rilievi');
2
3 for ii = [1 : n]
4     rilievo(ii) = inserisciRilievo();
5 end
6
7 % atezza media rilievi con
8 % lat tra 20 e 60 e long tra 30 e 40
9
10 altezze = [];
11 for ii = 1:n
12     if (rilievo(ii).lat > 20 & rilievo(ii).lat < 60 &
13         rilievo(ii).lon > 30 & rilievo(ii).lon < 40)
14         altezze = [altezze, rilievo(ii).alt];
15     end
16 end
17
18 altezzaMedia = mean(altezze)
19
20 %%
21 LAT = [rilievo.lat];
22 LON = [rilievo.lon];
23 ALT = [rilievo.alt];
24
25 indx = find(LAT > 20 & LAT < 60 & LON > 30 & LON < 40);
```

Command Window

```
-----
Your MATLAB license will expire
Please contact your system admin
MathWorks to renew this license.
-----

Academic License

>> a = 1321

a =

    1321

fx >>
```

Editor, permette di scrivere funzioni e script



Le istruzioni **possono terminare** con ';' ma non è obbligatorio
Di default, il risultato di ogni istruzione viene visualizzato nella
command window.

Il ';' **blocca la visualizzazione** del risultato dell'istruzione

- Maggiore velocità di esecuzione
- Visualizzazione più compatta

Regola di buona programmazione

- Inserire sempre il ';', a meno che non si voglia ispezionare il
valore di una variabile a scopo di debugging



Gli array (le variabili)

MATrix LABoratory...



Creazione ed Inizializzazione di una Variabile

Le variabili sono **create** mediante **inizializzazione**

- Cioè alla prima istruzione in cui compaiono, non occorre dichiarare le variabili come in C

Modi di inizializzazione

- **Assegnamento**
- Lettura dati da tastiera
- Lettura da file



Istruzione di Assegnamento

Come in C,

nomeVariabile = espressione

A differenza del C:

- non si deve (non è possibile) dichiarare la variabile **nomeVariabile** precedentemente.
- L'assegnamento sopra comporta una dichiarazione contestuale della variabile **nomeVariabile**.
- È possibile eseguire assegnamento tra array
- Non è richiesto il ; al termine dell'istruzione
- Il risultato di un'operazione che non comporta un assegnamento viene assegnato alla variabile **ans**



Assegnamento ed Inizializzazione

Quando assegno un valore ad una variabile che non è stata inizializzata (e.g., **a**), la **variabile viene creata**

```
>> a = 7
```

```
a =
```

```
7
```

Ovviamente non è possibile assegnare ad una variabile, il valore di una variabile che non esiste:

```
>> a = v
```

Undefined function or variable 'v'.

(messaggio di errore dell'interprete Matlab)



Caratteristiche del linguaggio di Matlab (3)

In Matlab tutto è un array, i.e.,

- **scalari:** array 1x1
- **vettori:** array con una sola riga o colonna
- **matrici:** array con due dimensioni
- **matrici multidimensionali:** array con più di 2 dimensioni

Il **tipo** delle variabili è definito dal valore che contengono (e viene definito al momento dell'assegnamento)



Il Workspace

Tutte le variabili vengono salvate nel workspace, che corrisponde allo spazio di memoria del programma

E' possibile visualizzare le variabili ed il workspace:

- Il comando **whos** (visualizza tutte le variabili)
- Il comando **whos nomeVariabile** (visualizza solo **nomeVariabile**)
- Il pannello del Workspace

Per pulire il workspace e rimuovere tutte le variabili presenti si usa il comando: **clear**

```
>> clear
```

```
>> whos
```

```
>>
```

Nessuna variabile è più presente nel workspace



Operazioni algebriche, assegnamento e confronto tra scalari

Input	Output	Commento
<code>1234/6</code>	<code>ans= 205.6667</code>	calcolo di un valore scalare
<code>a=1234/6</code>	<code>a = 205.6667</code>	assegnamento alla variabile a del risultato di 1234/6
<code>2/5</code>	<code>ans = 0.40000</code>	divisione
<code>5/0</code>	<code>ans = Inf</code>	divisione per zero
<code>5^2</code>	<code>ans = 25</code>	potenza
<code>1+1==2</code> <code>1+1~=2</code>	<code>ans = 1</code> <code>ans = 0</code>	1 = vero, 0 = falso, “==” uguale, “~=” diverso (codice ASCII di ‘~’: 126; alt126)



Operazioni algebriche, assegnamento e confronto tra scalari

Input	Output	Commento
1234/6	ans= 205.6667	calcolo di un valore scalare
a=1234/6	a = 205.6667	assegnamento alla variabile a del risultato di 1234/6
2/5	ans = 0.40000	divisione
5/0	ans = Inf	divisione per zero
5^2	ans = 25	potenza
1+1==2 1+1~2	ans = 1 ans = 0	1 = vero, 0 = falso, "==" uguale, "~=" diverso (codice ASCII di '~': 126; alt126)

La negazione in Matlab è rappresentata dal simbolo ~



Definizione di Vettori

I vettori sono definiti tra parentesi quadre:

- In un vettore riga gli elementi sono separati da virgole (o spazi)
- In un vettore colonna gli elementi sono separati da ; (o andando a capo)

Es:

`>> a = [1 2 3]`

a =

1 2 3

`>> a = [1, 2, 3]`

a =

1 2 3

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

a =

1

2

3



Operatori per Array: Trasposizione

L'operatore ' esegue la **trasposizione** (i.e. trasforma un vettore riga ad una colonna e viceversa)

```
>> a = [1 2 3]
a =
     1     2     3
```

```
>> a'
ans =
     1
     2
     3
```



Definizione Array Mediante Incremento Regolare

L'operatore : definisce vettori ad incremento regolare:

`[inizio : step : fine]`

Definisce un vettore che ha:

- primo elemento `inizio`
- secondo elemento `inizio + step`
- terzo elemento `inizio + 2*step`
- ...
- fino al più grande valore `inizio + k*step` che non supera `fine` (`fine` potrebbe non essere incluso)



Note sulla Creazione dei Vettori

Il valore di **step** può essere qualsiasi, anche negativo.

Se non precisato, **step** vale 1

Le parentesi [] possono essere omesse

Attenzione che i vettori definiti per incremento regolare possono essere vuoti (es `>> [10 : -1 : 10]`)

È ovviamente possibile modificare i valori di un array mediante assegnamento

- Di un singolo elemento (come in C)
- Di una parte dell'array
- Di tutto l'array



Assegnamento tra Array

In Matlab è possibile eseguire direttamente assegnamenti tra array

```
nomeArray1 = nomeArray2
```

Copia i valori contenuti in **nomeArray2** in **nomeArray1**

Non è necessario che **entrambi** gli array abbiano le stesse dimensioni

Es

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

```
>> a = a + 1
```

```
a =
```

```
2     3     4
```



Accedere agli Elementi di un Vettore

Notazione simile al C

nomeVettore (indice)

- Restituisce il valore contenuto in **nomeVettore** alla posizione **indice**.
- Come nel C, una volta specificato l'indice si accede all'elemento del vettore come ad una qualsiasi variabile (per assegnamenti ed altre operazioni)

Differenze importanti:

- Si usano le parentesi tonde () invece delle quadre []
- Il primo elemento di **nomeVettore** è alla posizione 1 (l'indice deve essere sempre positivo)



È possibile **modificare** un valore in un **vettore**

1. Accedendo all'elemento del vettore
2. Assegnando un nuovo valore nella posizione specifica

```
>> a = [1 : 3]
```

```
a =
```

```
1 2 3
```

```
>> a(3) = 6
```

```
a =
```

```
1 2 6
```

È possibile eseguire **l'assegnamento tra vettori**, anche quando i due vettori non hanno le stesse dimensioni: il vettore a cui viene assegnato il valore viene ridefinito

```
>> b = [1 : 4]
```

```
b =
```

```
1 2 3 4
```

```
>> a = b
```

```
a =
```

```
1 2 3 4
```



Accedere agli Elementi di una Array

Viene segnalato un **errore** quando si **accede** ad una **posizione che non corrisponde ad un elemento** dell'array (vale anche per matrici e array multidimensionali)

```
>> a = [1 : 3]    >> a(2)
```

```
ans =
```

```
2
```

```
a =
```

```
1    2    3
```

```
>> a(4)
```

```
Index exceeds matrix dimensions
```

```
>> a(1.3)
```

```
Subscript indices must either  
be real positive integers or  
logicals
```



Accedere agli Elementi di una Array

Viene segnalato un **errore** quando si **accede** ad una **posizione che non corrisponde ad un elemento** dell'array (vale anche per matrici e array multidimensionali)

```
>> a = [1 : 3]   >> ii = 2;
```

```
a =
```

```
    1    2    3
```

```
>> a(ii)
```

```
ans =
```

```
    2
```

È possibile utilizzare una variabile per definire l'indice, come in C

```
>> a(ii) = a(ii - 1) + a(ii + 1)
```

```
a =
```

```
    1    4    3
```



Operazioni Aritmetiche tra Vettori

Le **operazioni aritmetiche** sono quelle dell'algebra lineare

- La somma tra vettori $\mathbf{c} = \mathbf{a} + \mathbf{b}$ è definita elemento per elemento

$$c(i) = a(i) + b(i), \quad \forall i$$

è possibile solo quando \mathbf{a} e \mathbf{b} hanno la stessa dimensione (che poi coincide con quella di \mathbf{c})

- Prodotto tra vettori è il prodotto riga per colonna, restituisce uno scalare

$$\mathbf{c} = \mathbf{a} * \mathbf{b}, \text{ i.e. } c = \sum_i a(i)b(i)$$

\mathbf{a} deve essere un vettore riga e \mathbf{b} colonna e devono avere lo stesso numero di elementi, \mathbf{c} è un numero reale



Operazioni Puntuali

E' possibile eseguire operazioni **puntuali**, che si applicano cioè ad ogni elemento del vettore separatamente

$$\mathbf{c} = \mathbf{a} \text{ .* } \mathbf{b}, \text{ restituisce } c(i) = a(i) * b(i) \forall i$$

$$\mathbf{c} = \mathbf{a} \text{ ./ } \mathbf{b}, \text{ restituisce } c(i) = a(i)/b(i) \forall i$$

$$\mathbf{c} = \mathbf{a} \text{ .}^{\wedge} \mathbf{b}, \text{ restituisce } c(i) = a(i)^{b(i)} \forall i$$

Come in algebra lineare, le **operazioni tra vettori (array) e scalari** sono possibili, e corrispondono ad operazioni puntuali.

Se **k** è uno scalare e **b** è un vettore

$$\mathbf{c} = \mathbf{k} * \mathbf{b} = \mathbf{k} \text{ .* } \mathbf{b} \quad c(i) = k * b(i) \forall i$$



Attenzione: Elevamento a Potenza

```
>> v1 = [2      3      5      4]
```

```
>> v1^2
```

Error using ^

Inputs must be a scalar and a square matrix.

To compute elementwise POWER, use POWER (.^)
instead.

L'elevamento a potenza fa' riferimento al prodotto vettoriale
(equivale a $\mathbf{v1} * \mathbf{v1}$ che vale solo per matrici quadrate)

Per elevare a potenza ogni singolo elemento di $\mathbf{v1}$ si usa:

```
>> v1.^2
```

```
ans =
```

```
4      9      25      16
```

che equivale a fare $\mathbf{v1} .* \mathbf{v1}$



Operazioni Aritmetiche su Array

Operazione	Sintassi Matlab	Commenti
Array addition	$a + b$	Array e matrix addition sono identiche
Array subtraction	$a - b$	Array e matrix subtraction sono identiche
Array multiplication	$a .* b$	Ciascun elemento del risultato è pari al prodotto degli elementi corrispondenti nei due operandi
Matrix multiplication	$a * b$	Prodotto righe per colonne dell'algebra lineare
Array right division	$a ./ b$	risultato(i,j)=a(i,j)/b(i,j)
Array left division	$a .\ b$	risultato(i,j)=b(i,j)/a(i,j)
Matrix right division	a / b	$a * \text{inversa}(b)$
Matrix left division	$a \ b$	$\text{inversa}(a) * b$
Array exponentiation	$a .^ b$	risultato(i,j)=a(i,j)^b(i,j)



Concatenare i Vettori

L'operatore `,` e `;` permettono di concatenare vettori, purché le dimensioni siano compatibili (devono essere entrambi riga o colonna).

Esempio:

```
>> a = [1,2,3]
```

```
a =
```

```
1
```

```
2
```

```
3
```



Concatenare i Vettori

L'operatore , e ; permettono di concatenare vettori, purché le dimensioni siano compatibili (devono essere entrambi riga o colonna).

Esempio:

```
>> a = [1, 2, 3]
```

```
a =
```

```
    1    2    3
```

```
>> b = [a, a + 3, a + 6]
```

```
b =
```

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



Concatenare i Vettori

L'operatore `,` e `;` permettono di concatenare vettori, purché le dimensioni siano compatibili (devono essere entrambi riga o colonna).

Esempio:

```
>> a = [1, 2, 3]
```

```
a =
```

```
    1    2    3
```

```
>> b = [a, a + 3, a + 6]
```

```
b =
```

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

```
>> b = [a, a + 3]
```

```
b =
```



```
 1  2  3  1  2  3  3
```

Viene interpretato
come `b = [a, a, +3]`
ATTENZIONE agli spazi



Concatenare i Vettori

Esempi

- $\mathbf{a} = [0 \ 7+1];$  contenuto di a
- $\mathbf{b} = [\mathbf{a}(2) \ 5 \ \mathbf{a}];$  secondo elemento di a

Risultato

- $\mathbf{a} = [0 \ 8]$
- $\mathbf{b} = [8 \ 5 \ 0 \ 8]$



Stringhe

Come in C sono array di caratteri

I valori vengono assegnati mediante apici singoli '

Esempi:

```
>> msg = 'ciao mamma';
```

```
>> msg = [msg , ' torno per cena']
```

```
msg =
```

```
ciao mamma torno per cena
```

```
>> msg(1) = 'C'
```

```
msg =
```

```
Ciao mamma torno per cena
```



Le matrici vengono definite affiancando vettori di dimensioni compatibili

- Usiamo sempre gli operatori , (spazio) e ; (vai a capo)
- L'operazione di **trasposizione** inverte le righe e le colonne della matrice

Es :

```
>> a = [1 , 2 ; 3 ,4 ]
```

a =

1	2
3	4

a' =

1	3
2	4



Le Matrici: operatore CAT

La concatenazione dei vettori avviene mediante operatore **CAT** che richiede **dimensioni consistenti** dei vettori

```
>> a = [1 : 3]
```

```
a =
```

```
    1    2    3
```

```
>> A = [a; b]
```

```
Error using vertcat
```

```
CAT arguments dimensions are  
not consistent.
```

```
>> b = [4; 5; 6] >> A = [a, b]
```

```
b =
```

```
    4
```

```
    5
```

```
    6
```

```
Error using horzcat
```

```
CAT arguments dimensions are  
not consistent.
```

```
>> A = [a; b']
```

```
A =
```

```
    1    2    3
```

```
    4    5    6
```




Accedere agli Elementi di una Matrice

Per accedere agli elementi di una matrice occorre specificare un valore per ogni indice

nomeMatrice(indice1, indice2)

Seleziona il valore alla riga **indice1** colonna **indice2** nella variabile **nomeMatrice**

Es

```
>> A = [1 : 3; 4 : 6; 7: 9 ]
```

```
A =
```

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

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

```
ans =
```

```
    6
```

```
>> A(3,5)
```

```
Index exceeds  
matrix dimensions.
```



Operazioni Aritmetiche con Matrici

Operazioni per gli array

- Array operation: eseguita sugli elementi corrispondenti degli array coinvolti (devono avere lo stesso numero di righe e colonne); si indica aggiungendo un punto prima dell'operatore aritmetico

- $a = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ $b = \begin{bmatrix} 2 & 3 \\ 5 & 7 \end{bmatrix}$ $a .* b = \begin{bmatrix} 2 & 6 \\ 15 & 28 \end{bmatrix}$

- Matrix operation: segue le regole dell'algebra lineare (prodotto righe per colonne)

- $a = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ $b = \begin{bmatrix} 2 & 3 \\ 5 & 7 \end{bmatrix}$ $a * b = \begin{bmatrix} 12 & 17 \\ 26 & 37 \end{bmatrix}$

$$(a * b)_{ij} = \sum_k a_{ik} b_{kj}$$



Matrix left division

Serve per risolvere sistemi di equazioni lineari. Ad esempio:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3 \end{cases}$$

può essere espresso come $Ax = b$ con

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

di conseguenza, $x = A^{-1}B$ oppure $x = A \setminus B$

che corrisponde a calcolare la matrice A^{-1} che è l'inversa di A rispetto all'operatore prodotto matriciale



Definizione ed Estensione Automatica di Array

A differenza del C, un **assegnamento** in una posizione in cui il vettore non è definito (invece di avere segmentation fault) **estende l'array** inserendo 0

```
>> C = 1
```

```
C =
```

```
1
```

```
>> C(3) = 3
```

```
C =
```

```
1    0    3
```

```
>> C(2,3) = 5
```

```
C =
```

```
1    0    3
```

```
0    0    5
```

N.B. Assegnare un valore ad un elemento è diverso da accedere



Array Multidimensionali

È possibile specificare una terza (quarta, quinta...) dimensione lungo la quale indicizzare un array.

Ad esempio le immagini a colori sono definite con tre piani colore (RGB), quindi

- un'immagine a colori 10 Mpixels, aspect ratio (3:4) richiede una matrice 3D di $2736 \times 3648 \times 3$ elementi
- 10 sec di video full HD (1080 x 768) a 24fps richiede una matrice 4D di $1080 \times 768 \times 3 \times (10 \times 24)$ elementi



Esempi di Operazioni su Matrici

<pre>a=[1 2; 3, 4]</pre>	<pre>a = 1 2 3 4</pre>	a ora è una matrice 2x2, ";" separa le righe; virgola (opzionale) separa elementi
<pre>a</pre>	<pre>a = 1 2 3 4</pre>	restituisce il valore della variabile a
<pre>x=[-1.3 sqrt(3) (1+2)/5]</pre>	<pre>x = -1.30000 1.73205 0.60000</pre>	Elementi del vettore possono essere espressioni aritmetiche
<pre>x(5)= abs(x(1))</pre>	<pre>x = -1.30000 1.73205 0.60000 0.00000 1.30000</pre>	Notazione con () per accedere a elementi di un array; abs valore assoluto; NB: vettore x esteso per includere nuovo elemento; elementi non assegnati sono nulli



Esempi di Operazioni su Matrici

$b=a'$	$b =$ 1 3 2 4	matrice trasposta (scambiate righe e colonne)
$c=a+b$	$c =$ 2 5 5 8	somma di matrici, elemento per elemento (sottrazione con "-" simile)
$x=[-1 \ 0 \ 2];$ $y=x'$	$y =$ -1 0 2	il ";" blocca l'output, ma non impedisce la valutazione



Le Variabili ed i Tipi



Il Workspace

Tutte le variabili vengono salvate nel workspace, che corrisponde alla memoria

E' possibile visualizzare le variabili ed il workspace:

- Il comando **whos** (visualizza tutte le variabili)
- Il comando **whos nomeVariabile** (visualizza solo **nomeVariabile**)
- Il pannello del Workspace

Per pulire il workspace e rimuovere tutte le variabili presenti si usa il comando: **clear**

```
>> clear
```

```
>> whos
```

```
>>
```

Nessuna variabile è più presente nel workspace



Tipo Double

Di default, valori numerici danno luogo a variabili di tipo **double**:
un double contiene uno **scalare** espresso con doppia precisione
(64 bit)

È possibile vedere il tipo delle variabili mediante **whos**

whos nomeVariabile

```
>> a = 7;
```

```
>> whos a
```

Name	Size	Bytes	Class	Attributes
a	1x1	8	double	



Tipo Char

Una variabile di tipo char contiene uno **scalare** o un **array** di valori a 16 bit (8 bit in Octave), ciascuno dei quali rappresenta un carattere

- Es: `frase = 'questa e` una stringa' ;`



Nome della variabile



Array di 1x21 caratteri

- NB: stringhe racchiuse tra apici *singoli*
- `whos commento;`

Name	Size	Bytes	Class	Attributes
<code>frase</code>	<code>1x21</code>	<code>42</code>	<code>char</code>	



Tipo Complex

In Matlab è possibile rappresentare anche numeri **complessi**
parti reali e immaginarie possono essere positive e negative
nell'intervallo di valori $[10^{-308}, 10^{308}]$

```
>> a = [sqrt(-1) 7]
a =
    0 + 1.0000i    7.0000
```

```
>> whos
```

Name	Size	Bytes	Class
a	1x2	32	double

Attributes complex



Tipo Complex

I double possono essere utilizzati per esprimere numeri

- Reali, es $\text{var1} = -10.7$;
- Immaginari, es $\text{var2} = 4i$; $\text{var3} = 4j$;
- Complessi, es $\text{var3} = 10.3 + 10i$;

Es: $x = [-1.3 \quad 3.1+5.3j \quad 0]$

NB Meglio non usare mai i e j come nome di variabile



Gestione Dinamica delle Variabili

I tipi delle variabili possono cambiare:

- mediante conversione esplicita
- mediante assegnamento: **il tipo** di una variabile è **definito dal valore** contenuto

```
>> a = [1 3 5] .^ (0.2)
```

```
a =
```

```
    1.0000    1.2457    1.3797
```

```
>> whos a
```

Name	Size	Bytes	Class	Attributes
a	1x3	24	double	

```
>> a = 'cia';
```

```
>> whos a
```

Name	Size	Bytes	Class	Attributes
a	1x3	6	char	



Script



Script (m-file)

Uno script è un **file di testo** contenente una **sequenza di comandi MATLAB**

- non deve contenere caratteri di formattazione (solo testo puro)
- viene salvato con estensione `.m`

I comandi all'interno di uno script sono eseguiti **sequenzialmente**, come se fossero scritti nella finestra dei comandi

- Per eseguire il file (quindi la sequenza di istruzioni che contiene) si digita il suo nome (senza `.m`)
- I risultati appaiono nella finestra dei comandi (se non usiamo il `;`)

Provare ad eseguire uno script in modalità Debug!



Uno script può

- essere ri-eseguito
- essere facilmente modificato
- essere facilmente inviato

Uno script opera sulle variabili del workspace:

- posso introdurne di nuove durante l'esecuzione
- posso accedere alle variabili nel workspace
- posso modificare le variabili del workspace

ATTENZIONE: spesso è bene ripulire il workspace (con il comando `clear`) come prima cosa, in modo da evitare di eseguire script su variabili che compaiono già nel workspace



Commenti

Il simbolo di commento può essere messo in qualsiasi punto della linea.

MATLAB ignorerà tutto quello che viene scritto alla destra del simbolo `%`.

Per esempio:

```
>> % This is a comment.  
>> x = 2+3 % So is this.  
x =  
    5
```



Come Creare uno Script

Può essere creato utilizzando un qualsiasi editor di testo

- Ricordarsi di salvare il file come “solo testo” e di dare l'estensione .m
- Il file di **script** deve essere **presente nella directory corrente** o il **folder** contenente lo script deve **comparire nel path** di Matlab



Nomi degli Script

Il nome del file deve **iniziare con una lettera** e può contenere cifre e il carattere underscore, fino a 31 caratteri

Non dare lo stesso nome al file di script e a una variabile

Non chiamare uno script con lo stesso nome di un comando o funzione MATLAB.

Per verificare se esiste già qualcosa che ha un certo nome si può utilizzare la funzione `exist`.



Strutturare e Documentare uno Script

1. Sezione dei commenti:
 - Il nome del programma e le parole chiave, nella prima riga
 - La data di creazione e i nomi degli autori nella seconda riga
 - La definizione dei nomi delle variabili per ogni variabile di input e di output
 - Il nome di ogni funzione creata dall'utente che viene usata nel programma
 - Il comando help visualizza tutta la sezione dei commenti all'inizio dello script
2. Sezione di Input: inserimento dei dati in input e/o uso di funzioni di input
3. Sezione di calcolo
4. Sezione di output: uso di funzioni per visualizzare i risultati del programma



Dati su cui Opera Uno Script

Gli script non accettano argomenti d'entrata e d'uscita

Usano

- variabili già presenti nel workspace
- variabili acquisite da tastiera o file
- nuove variabili introdotte nello script

Le variabili interne allo script diventano variabili del workspace

- Permangono anche dopo l'esecuzione dello script



Sezione di Calcolo

Calcoli matematici

Assegnamenti

Strutture di controllo

- Condizioni
- Cicli

Comandi per la costruzione di grafici

Chiamate a funzioni



Esempio di alcuni comandi (analizzeremo quelli più importanti)

- Il prompt accetta i comandi del sistema operativo (DOS, UNIX...)
 - Esempio: in ambiente dos, dir mostra il contenuto della directory corrente
- help richiama la guida in linea
- diary può essere utilizzato per salvare la sessione di lavoro
- who, whos e workspace mostrano l'elenco delle variabili definite
- save permette di salvare in un file le variabili definite. Load le ricarica
- clear cancella tutte le variabili
- close chiude tutte le figure



Altre Operazioni sugli Array

SubArray e cancellazione elementi



Sottoarray (un vettore come indice di un vettore)

Si denota un sottoinsieme di un array usando vettori per valori degli indici

nomeVettore (vettoreIndici)

restituisce un vettore che comprende gli elementi di **nomeVettore** che compaiono nelle posizioni **vettoreIndici**

Estende quindi l'accesso ad un singolo elemento dell'array

nomeVettore (indice)



Sottovettori definiti da vettori di indici

Quindi, la notazione

$$\mathbf{a}(\mathbf{v})$$

corrisponde a

$$[\mathbf{a}(\mathbf{v}(1)), \mathbf{a}(\mathbf{v}(2)), \dots, \mathbf{a}(\mathbf{v}(\text{end}))]$$

Attenzione che i valori di \mathbf{v} devono essere interi positivi e minori delle dimensioni di \mathbf{a} , devono essere indici validi.



La Keyword `end`

All'interno di **vettoreIndici** si può usare la keyword **end** che assume il valore dell'ultimo indice disponibile su una specifica dimensione di **nomeVettore**.

In questo modo non occorre conoscere le dimensioni del vettore

Esempi

```
>> a = [1 : 10]
```

```
a =
```

```
    1    2    3    4    5    6    7    8    9   10
```

Toglie l'ultimo
elemento

```
>> b = a(1 : end - 1)
```

```
b =
```

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

Legge il vettore
dall'ultimo elemento
al primo

```
>> b = a(end : -1 : 1)
```

```
b =
```

```
   10    9    8    7    6    5    4    3    2    1
```



Esempi

```
> v=[6 8 4 2 4 5 1 3]
```

```
v =      6      8      4      2      4      5      1      3
```

```
>> v([1 4 7])
```

```
>> v(2:2:6)
```

```
>> v(3:end-2)
```

```
>> v(v)
```

```
>> v([1, 1, 1, 2, end])
```



Esempi

```
> v=[6 8 4 2 4 5 1 3]
```

```
v =      6      8      4      2      4      5      1      3
```

```
>> v([1 4 7])
```

```
ans =      6          2          1
```

primo, quarto settimo elemento

```
>> v(2:2:6)
```

```
>> v(3:end-2)
```

```
>> v(v)
```

```
>> v([1, 1, 1, 2, end])
```



Esempi

```
> v=[6 8 4 2 4 5 1 3]
```

```
v =      6      8      4      2      4      5      1      3
```

```
>> v([1 4 7])
```

```
ans =      6          2          1
```

primo, quarto settimo elemento

```
>> v(2:2:6)
```

```
ans =      8          2          5
```

2:2:6 è il vettore [2, 4, 6]: indica secondo, quarto, sesto elemento

```
>> v(3:end-2)
```

```
>> v(v)
```

```
>> v([1, 1, 1, 2, end])
```



Esempi

```
> v=[6 8 4 2 4 5 1 3]
```

```
v =      6      8      4      2      4      5      1      3
```

```
>> v([1 4 7])
```

```
ans =      6          2          1
```

primo, quarto settimo elemento

```
>> v(2:2:6)
```

```
ans =      8          2          5
```

2:2:6 è il vettore [2, 4, 6]: indica secondo, quarto, sesto elemento

```
>> v(3:end-2)
```

```
ans =      4          2          4          5
```

dal terzo al terz'ultimo elemento

```
>> v(v)
```

```
>> v([1, 1, 1, 2, end])
```




Esempi

```
> v=[6 8 4 2 4 5 1 3]
```

```
v =      6      8      4      2      4      5      1      3
```

```
>> v([1 4 7])
```

```
ans =      6          2          1
```

primo, quarto settimo elemento

```
>> v(2:2:6)
```

```
ans =      8          2          5
```

2:2:6 è il vettore [2, 4, 6]: indica secondo, quarto, sesto elemento

```
>> v(3:end-2)
```

```
ans =      4          2          4          5
```

dal terzo al terz'ultimo elemento

```
>> v(v)
```

```
ans =      5      3      2      8      2      4      6      4
```

i valori di v usati come indice

```
>> v([1, 1, 1, 2, end])
```



Esempi

```
> v=[6 8 4 2 4 5 1 3]
```

```
v =      6      8      4      2      4      5      1      3
```

```
>> v([1 4 7])
```

```
ans =      6          2          1
```

primo, quarto settimo elemento

```
>> v(2:2:6)
```

```
ans =      8          2          5
```

2:2:6 è il vettore [2, 4, 6]: indica secondo, quarto, sesto elemento

```
>> v(3:end-2)
```

```
ans =      4          2          4          5
```

dal terzo al terz'ultimo elemento

```
>> v(v)
```

```
ans =      5      3      2      8      2      4      6      4
```

i valori di v usati come indice

```
>> v([1, 1, 1, 2, end])
```

```
ans = 6      6      6      8      3
```

Indici ripetuti fanno replicare i valori nel sottovettore



Modificare un Sotto-Array

È possibile effettuare l'assegnamento tra sottovettori per modificare una parte del vettore

$$\mathbf{v1}(\mathbf{vettoreIndici}) = \mathbf{v2}$$

Viene però richiesto che $\mathbf{v2}$ abbia le stesse dimensioni di $\mathbf{v1}(\mathbf{vettoreIndici})$

```
>> a = [1 : 10]
```

```
a =
```

```
  1  2  3  4  5  6  7  8  9 10
```

```
>> a(1 : 3) = [0 0 0]
```

```
a =
```

```
  0  0  0  4  5  6  7  8  9 10
```



Modificare un Sotto-Array

È possibile effettuare l'assegnamento tra sottovettori per modificare una parte del vettore

$$\mathbf{v1}(\mathbf{vettoreIndici}) = \mathbf{v2}$$

Viene però richiesto che $\mathbf{v2}$ abbia le stesse dimensioni di $\mathbf{v1}(\mathbf{vettoreIndici})$

```
>> a =
```

```
    0  0  0  4  5  6  7  8  9 10
```

```
>> a(2 : 2 : end) = 2 * a(2 : 2 : end)
```



Modificare un Sotto-Array

È possibile effettuare l'assegnamento tra sottovettori per modificare una parte del vettore

$$\mathbf{v1}(\mathbf{vettoreIndici}) = \mathbf{v2}$$

Viene però richiesto che $\mathbf{v2}$ abbia le stesse dimensioni di $\mathbf{v1}(\mathbf{vettoreIndici})$

```
>> a =  
      0 0 0 4 5 6 7 8 9 10  
>> a(2 : 2 : end) = 2 * a(2 : 2 : end)  
a =  
      0 0 0 8 5 12 7 16 9 20
```



Modificare un Sotto-Array

È possibile effettuare l'assegnamento tra sottovettori per modificare una parte del vettore

$$\mathbf{v1}(\mathbf{vettoreIndici}) = \mathbf{v2}$$

Viene però richiesto che $\mathbf{v2}$ abbia le stesse dimensioni di $\mathbf{v1}(\mathbf{vettoreIndici})$

```
>> a =  
      0 0 0 4 5 6 7 8 9 10  
>> a(2 : 2 : end) = 2 * a(2 : 2 : end)  
a =  
      0 0 0 8 5 12 7 16 9 20  
>> a(1 : 2 : end) = a(end : -2 : 1)
```



Modificare un Sotto-Array

È possibile effettuare l'assegnamento tra sottovettori per modificare una parte del vettore

$$\mathbf{v1}(\mathbf{vettoreIndici}) = \mathbf{v2}$$

Viene però richiesto che $\mathbf{v2}$ abbia le stesse dimensioni di $\mathbf{v1}(\mathbf{vettoreIndici})$

```
>> a =  
      0 0 0 4 5 6 7 8 9 10  
>> a(2 : 2 : end) = 2 * a(2 : 2 : end)  
a =  
      0 0 0 8 5 12 7 16 9 20  
>> a(1 : 2 : end) = a(end : -2 : 1)
```



Modificare un Sotto-Array

È possibile effettuare l'assegnamento tra sottovettori per modificare una parte del vettore

$$\mathbf{v1}(\mathbf{vettoreIndici}) = \mathbf{v2}$$

Viene però richiesto che $\mathbf{v2}$ abbia le stesse dimensioni di $\mathbf{v1}(\mathbf{vettoreIndici})$

```
>> a =  
      0 0 0 4 5 6 7 8 9 10  
>> a(2 : 2 : end) = 2 * a(2 : 2 : end)  
a =  
      0 0 0 8 5 12 7 16 9 20  
>> a(1 : 2 : end) = a(end : -2 : 1)  
                        20 16 12 8 0
```




Modificare un Sotto-Array

È possibile effettuare l'assegnamento tra sottovettori per modificare una parte del vettore

$$\mathbf{v1}(\mathbf{vettoreIndici}) = \mathbf{v2}$$

Viene però richiesto che $\mathbf{v2}$ abbia le stesse dimensioni di $\mathbf{v1}(\mathbf{vettoreIndici})$

```
>> a =  
    0  0  0  4  5  6  7  8  9 10  
>> a(2 : 2 : end) = 2 * a(2 : 2 : end)  
a =  
    0  0  0  8  5 12  7 16  9 20  
>> a(1 : 2 : end) = a(end : -2 : 1)  
a =  
   20  0 16  8 12 12  8 16  0 20
```



Sottoarray: Applicazione a Matrici

Si denota un sottoinsieme di un array usando vettori per valori degli indici

nomeMatrice (vettore1 , vettore2)

restituisce una matrice che comprende gli elementi di **nomeMatrice** alle righe di indice in **vettore1** e alle colonne di indice in **vettore2**.



Sottoarray: Applicazione a Matrici

```
m = 9      8      7
      6      5      4
      3      2      1
      0     11     12
      0      0      0
```

```
>> m([1 4], [2 3])
```



Sottoarray: Applicazione a Matrici

m =	9	8	7
	6	5	4
	3	2	1
	0	11	12
	0	0	0

```
>> m([1 4], [2 3])  
ans = 8 7  
      11 12
```

tutti gli elementi sulle
righe 1 e 4 e sulle colonne 2 e 3



Sottoarray: Applicazione a Matrici

```
m = 9      8      7
     6      5      4
     3      2      1
     0     11     12
     0      0      0
```

```
>> m([1 4], [2 3])
ans = 8      7
      11     12
```

tutti gli elementi sulle
righe 1 e 4 e sulle colonne 2 e 3

```
>> m(1:2:5, 1:end)
```



Sottoarray: Applicazione a Matrici

```
m = 9      8      7
     6      5      4
     3      2      1
     0     11     12
     0      0      0
```

```
>> m([1 4], [2 3])
ans = 8      7
      11     12
```

tutti gli elementi sulle
righe 1 e 4 e sulle colonne 2 e 3

```
>> m(1:2:5, 1:end)
ans = 9      8      7
      3      2      1
      0      0      0
```

tutti gli elementi delle
righe 1, 3 e 5



Sottoarray: Applicazione a Matrici

```
m = 9      8      7
      6      5      4
      3      2      1
      0     11     12
      0      0      0
```

```
>> m(1:2:5, :)
```

```
ans = 9      8      7
      3      2      1
      0      0      0
```

notazione ':' abbreviata per 1:end,
cioè tutti i valori di quell'indice



Sottoarray: Applicazione a Matrici

```
m = 9      8      7
     6      5      4
     3      2      1
     0     11     12
     0      0      0
```

```
>> m(1:2:5, :)
```

```
ans = 9      8      7
       3      2      1
       0      0      0
```

notazione ':' abbreviata per 1:end,
cioè tutti i valori di quell'indice

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

```
m = 9      8      7
     -1     -2     -3
     3      2      1
     -4     -5     -6
     0      0      0
```

uso della notazione dei sottoarray per
individuare elementi oggetto di
assegnamento



Esempio

```
% inizializzare una matrice 5x5 con tutti valori a zero  
  
% modificare la colonna centrale in 1  
  
% modificare la riga centrale in 3  
  
% sommare 2 ai valori della colonna centrale  
  
% porre a 2 gli elementi nel primo quadrante  
  
% copiare nell'ultima riga la prima riga letta al  
contrario
```



Esempio

```
% inizializzare una matrice 5x5 con tutti valori a zero
A(5,5) = 0;
% modificare la colonna centrale in 1
A(:, 3) = 1;
% modificare la riga centrale in 3
A(3, :) = 3;
% sommare 2 ai valori della colonna centrale
A(:, 3) = A(:, 3) + 2; % NB termini a dx e sx
dell'uguale hanno la stessa dimensione
% porre a 2 gli elementi nel primo quadrante
A(1 : 2, 1 : 2) = 2;
% copiare nell'ultima riga la prima riga letta al
contrario
A(end, :) = A(1, end : -1 : 1)
```



Assegnamenti con Scalari

È possibile associare a qualsiasi sotto array un valore scalare

$$\mathbf{nomeVettore}(\mathbf{vettoreIndici}) = \mathbf{k}$$

Fa sì che a tutti gli elementi di **nomeVettore** alle posizioni **vettoreIndici** venga assegnato il valore **k**

In questo modo è possibile inizializzare nuovi vettori.

```
>> a = [1 : 10]
```

```
a =
```

```
    1    2    3    4    5    6    7    8    9   10
```

```
>> a(1 : 3) = 0
```

```
a =
```

```
    0    0    0    4    5    6    7    8    9   10
```



Assegnamenti con Scalari

Esempio

- $m(1:4, 1:3) = 3$

$$\longrightarrow \begin{bmatrix} 3 & 3 & 3 \\ 3 & 3 & 3 \\ 3 & 3 & 3 \\ 3 & 3 & 3 \end{bmatrix}$$

Il modo con cui uno scalare viene assegnato a un array dipende dalla forma dell'array che viene specificata a sinistra dell'assegnamento

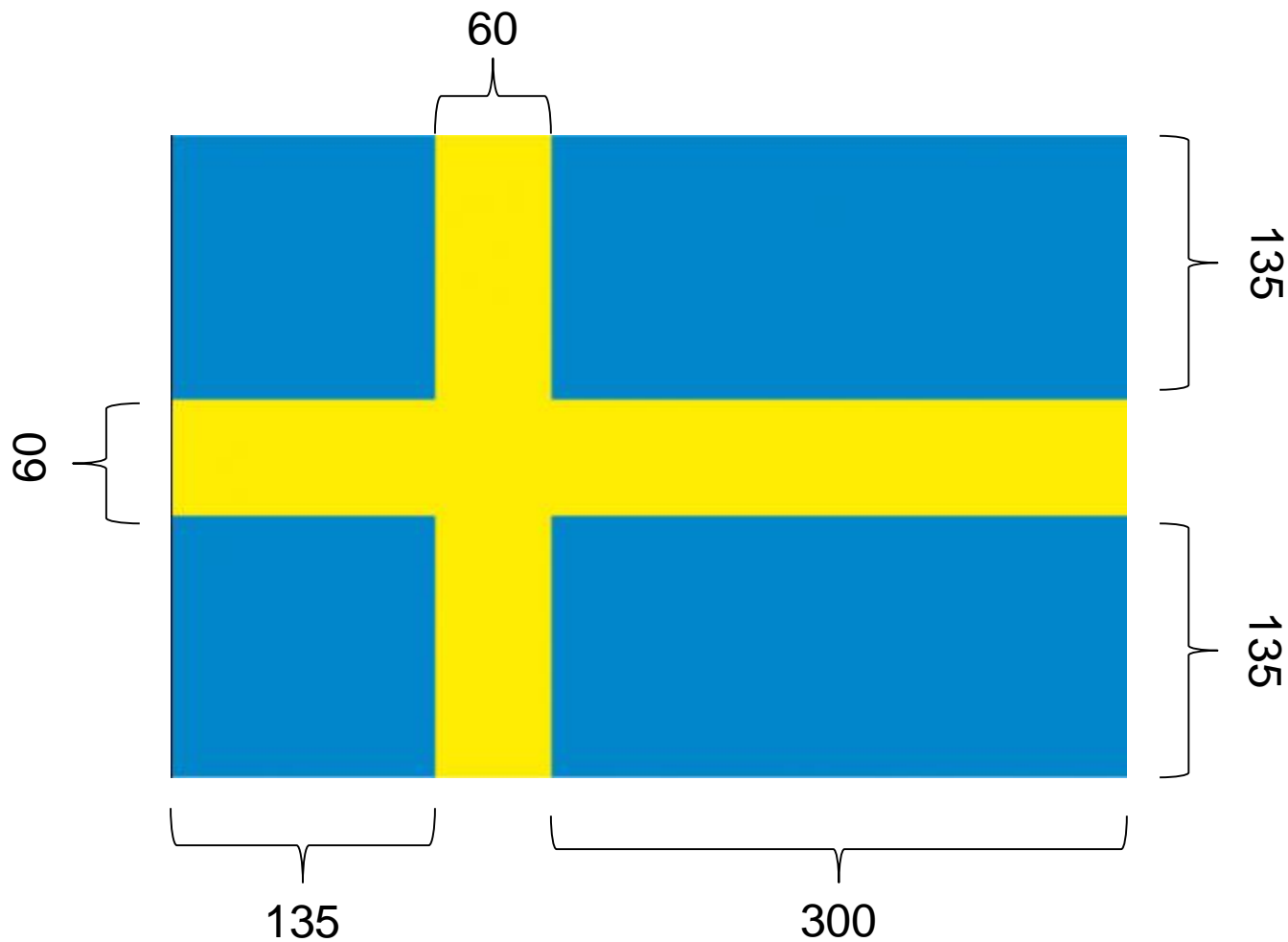
Esempio

- $m(1:2, 1:2) = 4$

$$\longrightarrow \begin{bmatrix} 4 & 4 & 3 \\ 4 & 4 & 3 \\ 3 & 3 & 3 \\ 3 & 3 & 3 \end{bmatrix}$$



Disegnare la bandiera svedese

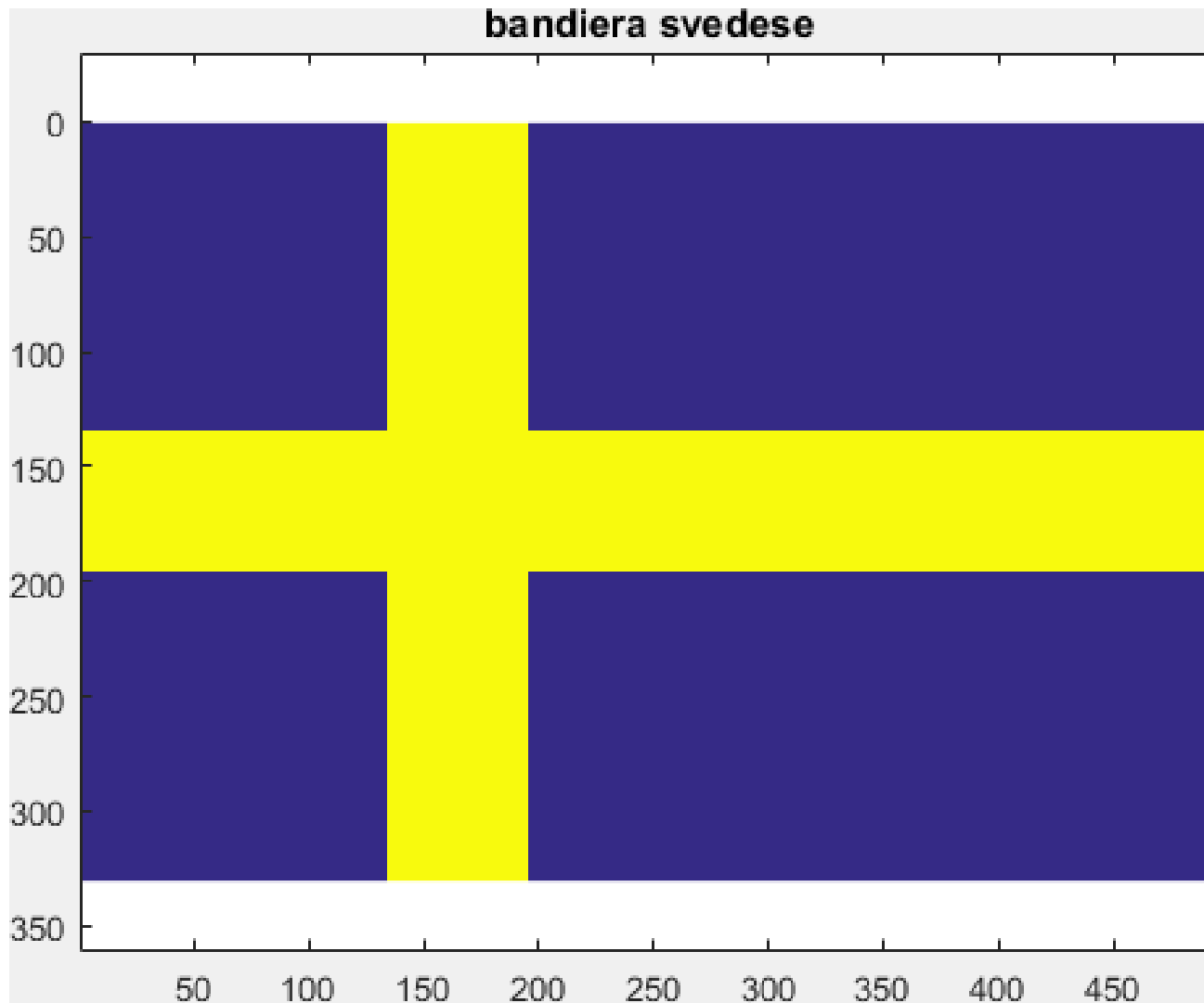




Bandiera Svedese (usare colormap)

```
close all
clear
A(330, 495) = 0;
A(:, 135: 195) = 1;
A(135: 195, :) = 1;

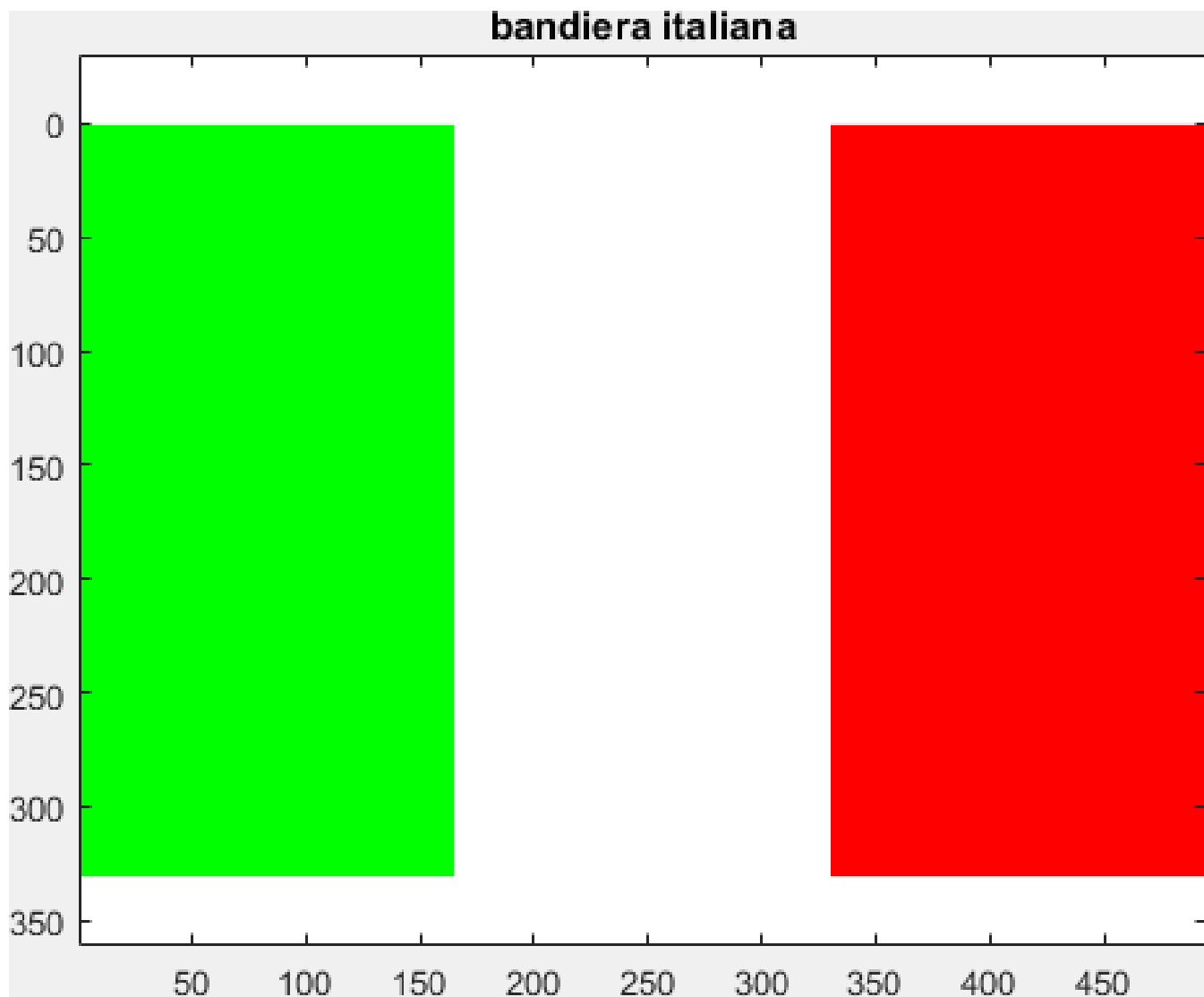
figure, imagesc(A), title('bandiera
svedese')
axis equal
%TODO: cambiare colormap per renderla più
realistica
```





Bandiera Italiana (immagine 3D)

```
clear A
A(330, 495, 3) = 0;
% verde
A(:, 1 : 495/3, 1) = 1;
% bianco
A(:, 495/3 : (2*495)/3, :) = 1;
% rosso
A(:, (2*495)/3 : end, 1) = 1;
figure, imagesc(A), title('bandiera
italiana')
axis equal
% TODO: sistemare i colori con la corretta
composizione RGB
```



GRAN CONTEST DELLE BANDIERINE

Regole:

- La bandiera deve essere realizzata interamente con uno script Matlab, senza interazione con l'utente
- E' necessario usare operazioni vettoriali, si possono usare cicli. Non assegnamenti pixel per pixel su tutta l'immagine.
- Potete presentare una sola bandiera a persona. Inviare una mail all'Ing. Trovò (cc Boracchi) che include sia
 - Un'immagine della bandiera realizzata da voi in Matlab (salvare l'immagine come png)
 - Inviare lo script (opportunamente indentato e commentato) che genera l'immagine
 - Nel soggetto "Bandierine Info B"
- Deadline: Domenica 19 Novembre



Come creare le immagini a colori:

- E' possibile riprodurre i colori in due modi:
 - creando una matrice 3D B che ha $N_righe \times N_colonne \times 3$ dove il terzo piano indica il colore ($B(:, :, 1)$ è il rosso, $B(:, :, 2)$ il verde, $B(:, :, 3)$ il blu) e visualizzando con `imshow`
 - Utilizzando la `colormap` e `imagesc`

Premi:

- Un punto alla bandierina più bella se lo studente è in grado di descrivere come ha raggiunto il risultato.



Esempi di bandiere realizzate dai vostri colleghi...

Bandiera della Catalogna





Esempi di bandiere realizzate dai vostri colleghi...



New Zealand Airforce



Esempio

Visualizzare a schermo la tabellina del 10



Esempio

Visualizzare a schermo le prime 10 tabelline (utilizzando solo operazioni matriciali)



Array Vuoto

Un array vuoto si definisce così:

```
nomeVettore = []
```

Può essere una forma di dichiarazione di una variabile

```
>> a = []
```

```
a =
```

```
 []
```

```
>> whos a
```

Name	Size	Bytes	Class	Attributes
a	0x0	0	double	



Cancellare Parti di un Vettore

Quando si assegna il valore [] ad un elemento di un vettore, il corrispondente elemento viene rimosso e il vettore ridimensionato: non si crea un 'buco'

```
>> a = [1 : 5]
```

```
a =
```

```
    1    2    3    4    5
```

```
>> whos a
```

Name	Size	Bytes	Class	Attributes
a	1x5	40	double	

```
>> a(3) = []
```

```
a =
```

```
    1    2    4    5
```

```
>> whos a
```

Name	Size	Bytes	Class	Attributes
a	1x4	32	double	



Cancellare Parti di una Matrice

L'array vuoto [] non è assegnabile a singoli elementi di matrici (non si possono "creare buchi")

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

```
m =
```

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

```
>> m(2 , : ) = 5
```

```
m =
```

```
    1    1    1
    5    5    5
    1    1    1
```

```
>> m(3,4) = []
```

??? Subscripted assignment dimension mismatch.



Cancellare Parti di una Matrice

È però assegnabile a intere righe o colonne di matrici, che vengono cancellate (ricompattando la matrice)

```
>> m(:, 2) = []
```

```
m =
```

```
    1    1  
    5    5  
    1    1
```

```
>> whos m
```

Name	Size	Bytes	Class	Attributes
m	3x2	48	double	



Memorizzazione degli Array

Gli array vengono salvati linearmente in memoria.

In particolare le matrici sono memorizzate

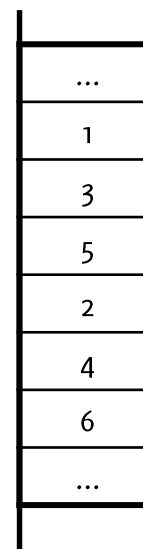
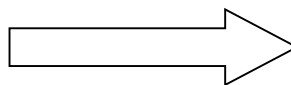
- per colonna: colonna 1, poi colonna 2, 3, etc.
- ogni colonna memorizzata per indici di riga crescenti

Array memorizzati in forma lineare nella RAM variando

- più velocemente i primi indici
- più lentamente quelli successivi

NB: opposto a quanto avviene in C

1	2
3	4
5	6





Array: forma *linearizzata*

Si può accedere a un array a più dimensioni come se ne avesse una sola

Usando un unico indice si segue l'ordine della memorizzazione

```
>> a = [1 2 3; 4 5 6; 7 8 9; 10 11 12]
```

```
a =
```

```
    1 |    2 |    3 |
    4 |    5 |    6 |
    7 |    8 |    9 |
   10 |   11 |   12 |
```

Vertical green arrows point downwards from each element in the first column (1, 4, 7, 10), the second column (2, 5, 8, 11), and the third column (3, 6, 9, 12). A small green arrow points downwards from the element 6.

```
>> a(3, 2)
```

```
ans =
```

```
    8
```

```
>> a(10)
```

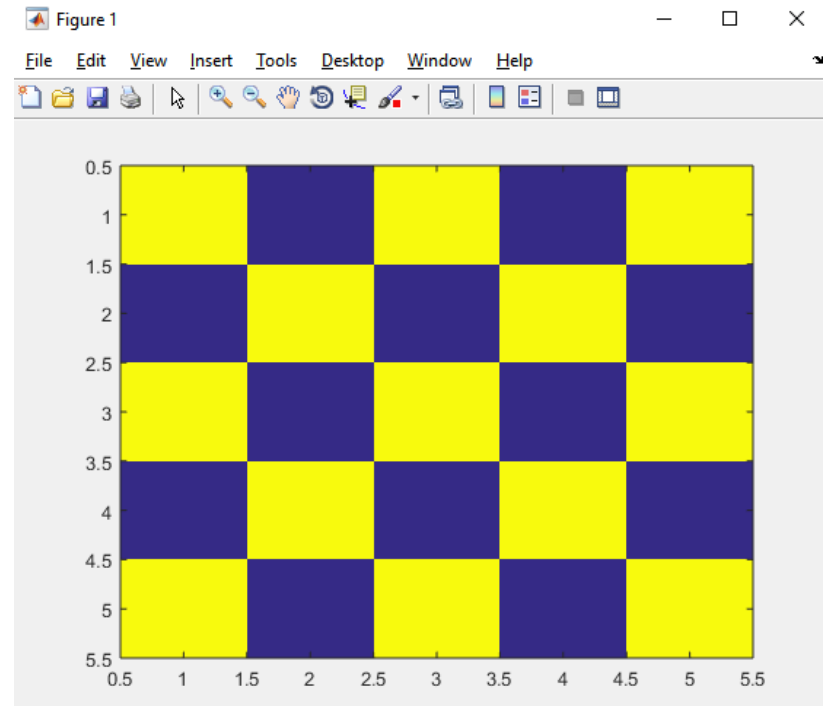
```
ans =
```

```
    6
```



Esempio

Scrivere uno script che stampa una scacchiera 5 x 5 (alternando elementi uguali a 0 ed uguali ad 1)





Variabili Predefinite



Variabili predefinite

Matlab definisce un insieme di variabili predefinite (es, **pi**)

Queste variabili spesso rappresentano importanti costanti della matematica (**pi** è pigreco, **i** e **j** sono $\sqrt{-1}$)

- Attenzione! Il valore di queste variabili può essere modificato, per esempio

- **circ1=2*pi*10;**

- **pi = 3;**

- **circ2=2*pi*10;**


- Il valore di circ2 non sarà più la circonferenza di un cerchio

E` fortemente sconsigliato modificare il valore di una variabile predefinita (\Rightarrow evitare di usare variabili **i** e **j** come contatori)



Variabili predefinite più comuni

Variabile	Scopo
pi	contiene 15 cifre significative di π
i, j, 1i, 1j	contiene il valore i ($\sqrt{-1}$)
inf (o Inf)	rappresentazione dell'infinito (ottenuto di solito come risultato di una divisione per 0)
NaN, nan	Not-A-Number è il risultato di una operazione matematica non definita, es 0/0
clock	contiene la data e l'orario corrente. E` un vettore di sei elementi (anno, mese, giorno, ora, minuti, secondi)
date	contiene la data corrente sotto forma di stringa
eps	epsilon: la più piccola differenza rappresentabile tra due numeri
ans	Variabile speciale usata per immagazzinare risultati non assegnati ad altre variabili



Input/output



Acquisizione Dati da Tastiera (input)

Funzione input

```
valore = input(stringaDaVisualizzare) ;
```

Matlab stampa a video la **stringaDaVisualizzare** e attende un input in formato Matlab

- Un numero (i.e., uno scalare)
- Un carattere (delimitato da apici singoli)
- Array, se racchiuso tra [e], oppure
- Stringa, se racchiusa tra ' e ', oppure
- Una qualsiasi espressione Matlab

Il dato inserito dall'utente viene memorizzato nella variabile **valore**

se **stringaDaVisualizzare** è una sequenza di caratteri, questa deve essere racchiusa tra apici singoli



Stampa dei Risultati (1)

I risultati di un'operazione sono mostrati immediatamente se non si inserisce il ;

Altre due funzioni: **disp**

- accetta come parametro un array. Se questo array è di tipo char, lo stampa
- Occorre spesso concatenare stringhe mediante l'operatore horzcat (i.e., ,)
- viene usato in congiunzione con la funzione **num2str** che trasforma un valore numerico in stringa
- Esempio:

```
str = ['il valore di pi e` ' num2str(pi)];  
disp(str);
```
- Stampa: "il valore di pi e` 3.1416"



Scrittura con `fprintf`

Altro modo: `fprintf`:

```
fprintf( '\nInserire a:' );
```

Sintassi:

```
fprintf ( stringaControllo );
```

- *stringaControllo* sequenze di caratteri (i.e., stringa) delimitata da doppi apici singoli `' '`.
- Possono essere
 - **caratteri normali** (lettere, cifre, punteggiatura)
 - caratteri speciali (es, vai a capo)
 - Placeholders (e.g. `'%d'` per il contenuto di variabili)

I caratteri nella *stringaControllo* vengono riportati a schermo.



stringaControllo:

Alcuni caratteri speciali per la stampa

- `'\n'` manda a capo
- `'\t'` spazia di un «tab»

Alcuni caratteri di conversione

- `%d` intero decimale
- `%f` numero reale
- `%c` carattere
- `%s` sequenza di caratteri (stringa)



disp vs. fprintf

disp è in grado di stampare anche valori complessi

```
>> x=2*(1-2*i)^3;
```

```
>> str=['disp: x = ' num2str(x)];
```

```
>> disp(str);
```

```
disp: x = -22+4i
```

fprintf ne stampa solo la parte reale

```
>> fprintf('fprintf: x = %8.4f\n', x);
```

```
fprintf : x = -22.0000
```



disp vs. fprintf (2)

disp stampa correttamente matrici e vettori

```
>> a = [1 1 1; 1 1 1]
```

```
>> disp(a)
```

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

fprintf stampa solo su una riga (ok vettori, problemi con matrici)

```
>> fprintf('%d', a)
```

```
111111>>>
```




disp vs. fprintf (3)

disp permette di stampare anche vettori concatenati con stringhe se le dimensioni sono compatibili

```
>> x = [1 2 3]
```

```
>> disp(['hai inserito ' num2str(x)])
```

```
hai inserito 1 2 3
```

fprintf opera diversamente

```
>> >> fprintf('hai inserito %d\n',x)
```

```
hai inserito 1
```

```
hai inserito 2
```

```
hai inserito 3
```



Esempio

Modificare lo script della scacchiera per richiedere all'utente le dimensioni e stampare anche un messaggio a schermo