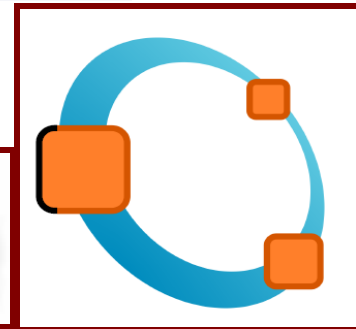


# MATLAB / Octave



Octave



Prima Parte

# Introduzione

- MATLAB/Octave è un linguaggio ad alto livello, pensato per il “numerical computation”
  - MATLAB/Octave è anche un ambiente di lavoro dove in una finestra è possibile dare comandi per risolvere numericamente un’ampia gamma di problemi (in-line command)

# MATLAB e Octave

- Il linguaggio di MATLAB/Octave è **interpretato**, gli **errori sono facili da trovare** ma non è **“lento” in esecuzione**.
  - Il codice può anche essere compilato e si può interfacciare con codici scritti in altri linguaggi: Fortran, C, C++.
- MATLAB è un prodotto commerciale che richiede una licenza



# Matlab Documentation

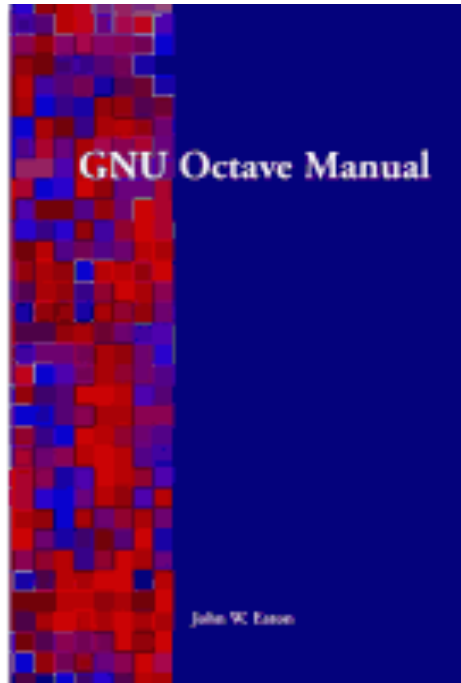
The screenshot displays the MATLAB Documentation website for the R2016b version. The browser's address bar shows the URL <http://it.mathworks.com/help/index.html>. The page has a blue header with the MathWorks logo and navigation links: [Prodotti](#), [Soluzioni](#), [Università](#), [Assistenza](#), [Community](#), and [Eventi](#). Below the header is a search bar labeled 'Search R2016b Documentation' and a 'Documentation' dropdown menu. On the left, a 'CONTENTS' sidebar lists various toolboxes under 'Product Documentation', including MATLAB, Simulink, Aerospace Blockset, and many others. The main content area features a 'MATLAB' section with a description of the language, a 'R2016b' section with sub-sections for the 'MATLAB Family', 'Simulink Family', and 'Polyspace Family', and a 'Hardware Support' section. The 'MATLAB Family' sub-section lists toolboxes like Parallel Computing, Math, Statistics, and Optimization, and Control Systems. The 'Simulink Family' sub-section lists toolboxes like Simulink, Event-Based Modeling, Physical Modeling, Control Systems, and Signal Processing and Communications. The 'Polyspace Family' sub-section lists toolboxes like DO Qualification Kit, IEC Certification Kit, Polyspace Bug Finder, Polyspace Code Prover, and Polyspace Products for Ada.

<http://it.mathworks.com/help/index.html>

# MATLAB e Octave

- Octave è un software libero. Si può ridistribuire e/o modificare sotto le condizioni della GNU General Public License come pubblicato dalla Free Software Foundation
- Le versioni più recenti di Octave vengono distribuite con una interfaccia grafica molto simile a quella di Matlab

# Octave Reference Manual

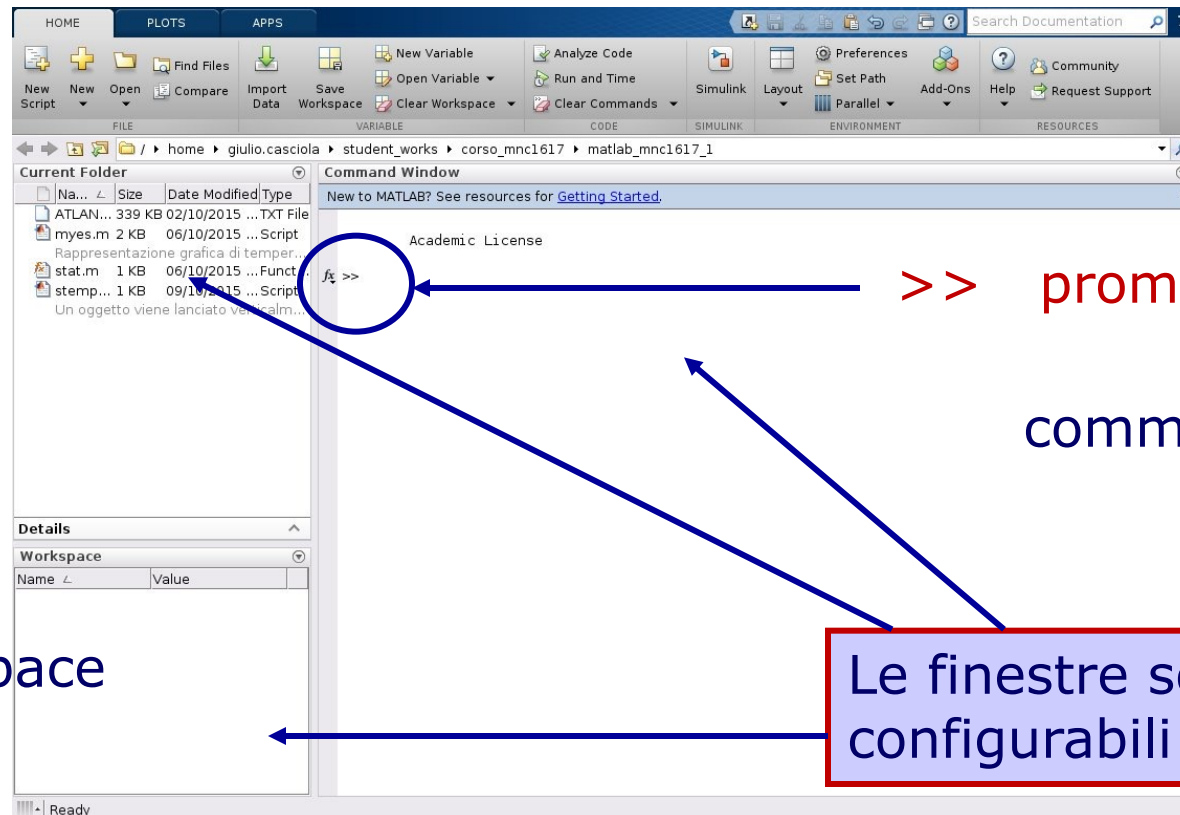
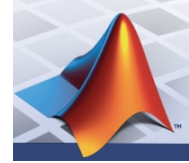


**GNU Octave Manual**  
by John W. Eaton

[Online Manual](#) - in HTML format

# Starting Matlab

Per partire con Matlab cliccare sull'icona relativa o lanciarlo dal menu delle applicazioni; si aprirà l'ambiente di lavoro:




>> prompt

command window

workspace

Le finestre sono configurabili a piacere

# Starting Matlab

- Nella command window appare il prompt `>>` che indica che Matlab è pronto a ricevere comandi e fare calcoli
- Per arrestare una computazione avviata si può usare `ctrl-c` che permetterà di tornare al prompt
- Per uscire dall'ambiente, digitare `quit` al prompt o chiudere la finestra con 



# Ambiente Matlab

Come detto Matlab è prima di tutto un ambiente in cui si possono eseguire delle elaborazioni numeriche.

Matlab esegue operazioni aritmetiche (+, -, \*, /), potenze (^), logaritmi ed esponenziali (log, exp), funzioni trigonometriche (sin, cos, ecc.) e può usufruire di numerose funzioni predefinite richiamabili come istruzioni/comandi di S.O. da shell. Nell'ambiente Matlab si possono memorizzare informazioni alfanumeriche sotto forma di variabili, array, strutture.

# Esempi

- ❑ Un semplice calcolo:

```
>> 1/7
```

```
ans=0.14286
```

- ❑ Per calcolare la radice quadrata di 2:

```
>> sqrt(2)
```

```
ans=1.4142
```

- ❑ Per visionare la lista dei file nella directory corrente:

```
>> ls
```

- ❑ Un qualunque comando di S.O.:

```
>> pwd
```

- ❑ Una espressione complessa; memorizza con il nome y il risultato:

```
>> y=3*exp(2*sqrt(x+1))
```

# Command Window

La Command Window permette di ...

- usare Matlab proprio come una calcolatrice
- fare *Copy* e *Paste* di operazioni per una facile ripetizione di comandi
- Usare 'up-arrow' e 'down-arrow' key



per riprendere i comandi  
eseguiti in precedenza

# Matlab Help

- L'Help di Matlab è uno strumento estremamente potente per imparare.
- L' Help non solo contiene informazioni teoriche, ma mostra anche esempi per l'implementazione (provare)

## Command Window

New to MATLAB? See resources for [Getting Started](#).

Academic License

```
>> help sqrt
```

**sqrt** Square root.

**sqrt(X)** is the square root of the elements of X. Complex results are produced if X is not positive.

See also [sqrtm](#), [realsqrt](#), [hypot](#).

[Reference page for sqrt](#)

[Other functions named sqrt](#)

 >>

# Espressioni

- Le **espressioni** sono la componente fondamentale delle istruzioni in Matlab.
  - Una espressione valuta un valore, che si può stampare, testare, memorizzare in una variabile, passare ad una funzione, o assegnare un nuovo valore ad una variabile con un operatore di assegnazione.
  - Una espressione da sola può essere una istruzione. Più istruzioni contengono una o più espressioni che specificano dati con cui si deve operare.
  - Le espressioni includono variabili, riferimenti ad array, costanti, e chiamate a funzioni, così come combinazioni di queste con vari operatori.

# Variabili

- Il nome di una **variabile** in Matlab deve essere una sequenza di lettere, cifre e underscore (\_), ma non può iniziare con una cifra.
- Non c'è limite al numero di caratteri nel nome di una variabile.
- Matlab è case-sensitive nei nomi di variabili. I simboli **a** e **A** sono variabili differenti.

# Variabili predefinite

- Un certo numero di variabili hanno un significato predefinito. Per esempio, `pwd` vale come la directory corrente, e `pi` significa il rapporto fra la circonferenza di un cerchio e il suo diametro (pigreca).
- Matlab possiede una lunga lista di variabili predefinite. Alcuni di questi simboli predefiniti sono costanti e non possono essere cambiati.

# Stato delle variabili

- **clear [options]**

cancella i nomi corrispondenti ai dati  
pattern della tabella dei simboli

>>clear a

>>clear all

- **who [options]**

- **whos [options]**

producono la lista dei simboli correnti  
definiti, corrispondenti ai dati.



# Tipi di dati

- I tipi di dati predefiniti sono
  - **scalari** “reali” e “complessi”
  - **range numerici**
  - **vettori e matrici** “reali” e “complessi”
  - ...

# Oggetti Dati Numerici

- Tutti i dati di tipo numerico sono memorizzati per default come numeri in "double precision"
- Poiché i sistemi usano Ansi/IEEE std.754 1985 floating point format, possono essere memorizzati i valori nel range da  $2.225e-308$  a  $1.797e+308$ , e la relativa precisione (U) è circa  $1.110e-16$ .
- I valori esatti sono dati rispettivamente dalle variabili `realmin`, `realmax` e `eps/2`

# Output

- Poiché Matlab solitamente stampa un valore non appena è stato valutato, la più semplice di tutte le funzioni di I/O è una espressione

```
>> a=15.2
```

```
a = 15.200
```

- Questo funziona anche dando come espressione il nome di una variabile (o la variabile di default **ans**) che verrà stampata insieme al suo valore.
- Terminando un comando/espressione con un “**punto e virgola**” si chiederà a Matlab di non stampare la conferma del comando

# Definire un vettore

- Per definire un vettore riga e memorizzarlo con il nome **b**, si può dare il comando:

```
>> b=[3,5,8]
```

- Matlab confermerà di aver eseguito quanto richiesto stampando il vettore riga

```
3      5      8
```

- Per definire un vettore colonna e memorizzarlo con il nome **c**, si può dare il comando:

```
>> c=[3;5;8]
```

# Definire un range

>>1:5

- definisce l'insieme di valori 1,2,3,4,5.

>>1:2:5

- definisce l'insieme di valori 1,3,5.
- Una espressione **range** è definita dal valore del primo elemento nel range, un valore opzionale per l'incremento fra gli elementi, e un valore massimo per gli elementi
- La **base**, l'**incremento** e il **limite** sono separati da ":" e possono essere una qualunque espressione o chiamata a funzione

# Definire un range

```
>>1:3:5
```

- definisce l'insieme di valori 1,4.

```
>>5:-3:1
```

- definisce l'insieme di valori 5,2.

```
>>k=2:2:20
```

- definisce il range dei numeri pari fra 2 e 20 e li memorizza nel vettore riga k

```
>>k=1:10
```

- definisce il range dei numeri interi da 1 a 10 e li memorizza nel vettore riga k

# Ancora su range

- Si noti che il valore superiore (o quello inferiore, se l'incremento è negativo) definito in range non viene sempre incluso nell'insieme dei valori
- Range definito da valori floating point può produrre risultati sorprendenti, perché viene usata l'aritmetica floating point
- Se risulta importante includere gli estremi di un range ed un ben preciso numero di elementi, si usi la funzione **linspace()**

# Definire una matrice

- Per definire una matrice e memorizzarla con il nome **A**, si può dare il comando:

```
>> A=[1,1,2;3,5,8;13,21,34]
```

ma anche

```
>> A=[1,1,2;  
      3,5,8;  
      13,21,34]
```

- Matlab confermerà di aver eseguito quanto richiesto stampando la matrice in colonne allineate

```
1      1      2  
3      5      8  
13     21     34
```



# Output di matrici

- Il comando

```
>> B=rand(3,2) ;
```

definisce una matrice di 3 righe e 2 colonne dove ogni elemento sarà un numero casuale fra zero ed uno; si noti che avendo usato il ";" non verrà visualizzato quanto memorizzato in **B**

- per visualizzare il valore di una qualunque variabile basta digitarne il nome, cioè

```
>> B(2,1)
```

# Elementi in matrici

- Gli oggetti matrice possono essere di qualsiasi dimensione, e possono essere dinamicamente ridimensionati (reshaped e resized)
- E' facile estrarre:
  - **Righe**,  $A(i, :)$  seleziona l' i-esima riga della matrice,
  - **Colonne**,  $A(:, j)$  seleziona la j-esima colonna della matrice, o
  - **sotto-matrici**,  $A([i1:i2], [j1:j2])$  seleziona le righe dalla i1 alla i2 e le colonne dalla j1 alla j2.

# Funzioni di dimensione oggetti

## `length(A)`

ritorna il numero elementi del vettore `A`; se `A` è una matrice ritorna il numero di righe o di colonne, a seconda di quale sia il più grande fra i due

## `columns(A)`

ritorna il numero di colonne di `A`

## `rows(A)`

ritorna il numero di righe `A`.

## `[nr,nc]=size(A)`

il numero di righe viene assegnato ad `nr` e il numero di colonne viene assegnato ad `nc`.

# Aritmetica Matriciale

- Gli operatori aritmetici in Matlab/Octave implementano l'aritmetica matriciale. Per moltiplicare la matrice **A** per lo scalare **2**, digitare:  
**>> 2\*A**
- Per moltiplicare le due matrici **A** e **B**, digitare:  
**>> A\*B**
- Per definire la trasposta di una matrice o un vettore trasposto si usa l'operatore **'**:  
**>> A'**
- Per fare il prodotto matrice vettore colonna, digitare:  
**>> A\*b'**  
**>> A\*c**

# Aritmetica Classica

- Per operare fra scalari Matlab/Octave ha i seguenti operatori: `“.+”` `“.-”` `“.*”` `“./”` `“.^”`
- Per moltiplicare ogni elemento di A per 2  
`>> 2.*A`
- Per moltiplicare le due matrici A e B, elemento per elemento (A e B devono avere le stesse dimensioni):  
`>> A.*B`
- Per fare elevare alla seconda ogni singolo elemento della matrice A:  
`>> A.^2`
- Per moltiplicare due vettori a e b elemento per elemento:  
`>> c=a.*b` %cosa fa invece `a*b` ?

# Risolvere sistemi lineari

- Per risolvere il sistema lineare  $\mathbf{Ax}=\mathbf{b}$ , si usi l'operatore di "left division"  $\backslash$  :

`>> A\b`

- Questo è concettualmente equivalente ad invertire la matrice  $\mathbf{A}$  ma evita il calcolo della matrice inversa
- Se la matrice dei coefficienti è singolare, Matlab stamperà un messaggio di "warning"

# Funzioni

- Una funzione è uno script per un particolare calcolo ed ha un nome. Per esempio, la funzione `sqrt` calcola la radice quadrata di un numero.
- Un certo numero di funzioni sono predefinite, che significa che sono disponibili per ogni programma. La funzione `sqrt` è una funzione predefinita.
- In aggiunta, l'utente può definire le proprie funzioni (vedremo più avanti).

# Chiamata di funzione

- Una espressione di chiamata a funzione consiste nel nome di una funzione e nella lista degli argomenti fra parentesi.
  - Gli argomenti sono espressioni che forniscono i dati su cui opera la funzione.
  - Quando c'è più di un argomento, questi sono separati da virgole.
  - Se non ci sono argomenti, si possono omettere le parentesi.



# Argomenti di funzioni

- La maggior parte delle funzioni si aspettano un ben preciso numero di argomenti.
  - `sqrt(x^2+y^2)`      % un argomento
  - `ones(n,m)`              % due argomenti
  - `rand()`                  % nessun argomento
  - `rand("seed",1)`      % due argomenti
- Alcune funzioni come `rand` prevedono un numero variabile di argomenti e si comportano diversamente a seconda del loro numero.

# Valori di ritorno di funzioni

- La maggior parte delle funzioni ritornano un valore

`y=sqrt(x)`

- In Matlab le funzioni possono ritornare più valori.

`[U,S,V]=svd(A)`

- Calcola la decomposizione in valori singolari della matrice **A** e assegna le tre matrici risultanti ad **U**, **S** e **V**.

# Funzioni elementari comuni

<code>abs(x)</code>	valore assoluto
<code>sign(x)</code>	funzione segno
<code>sqrt(x)</code>	radice quadrata
<code>exp(x)</code>	esponenziale ( $e^x$ )
<code>log(x)</code>	logaritmo naturale
<code>log10(x)</code>	logaritmo in base 10
<code>rem(x,y)</code>	resto della divisione $x/y$
<code>round(x)</code>	approssimazione all'intero più vicino
<code>ceil(x)</code>	approssimazione all'intero più vicino verso $\infty$
<code>floor(x)</code>	approssimazione all'intero più vicino verso $-\infty$
<code>fix(x)</code>	approssimazione all'intero più vicino verso 0
<code>sin(x)</code>	seno
<code>cos(x)</code>	coseno
<code>tan(x)</code>	tangente
<code>asin(x)</code>	arcoseno
<code>acos(x)</code>	arcocoseno
<code>atan(x)</code>	arcotangente

# Altre Funzioni utili

## se $v$ è un vettore:

<b>max</b> ( $v$ )	massimo degli elementi di $v$
$[y,j] = \mathbf{max}(v)$	$y$ è il massimo degli elementi di $v$ ; $j$ è la posizione di $y$ nel vettore $v$ .
<b>min</b> ( $v$ )	minimo degli elementi di $v$
$[y,j] = \mathbf{min}(v)$	$y$ è il minimo degli elementi di $v$ ; $j$ è la posizione di $y$ nel vettore $v$ .
<b>mean</b> ( $v$ )	media degli elementi di $v$

## se $X$ è una matrice:

<b>max</b> ( $X$ )	vettore contenente i valori massimi di ogni colonna di $X$
$[y,j] = \mathbf{max}(X)$	$y$ è il vettore contenente i valori massimi di ogni colonna di $X$ ; $j$ è il vettore contenente i numeri delle righe occupate dagli elementi di $y$ nella matrice $X$ .
<b>min</b> ( $X$ )	vettore contenente i valori minimi delle colonne di $X$
$[y,j] = \mathbf{min}(X)$	$y$ è il vettore contenente i valori minimi di ogni colonna di $X$ ; $j$ è il vettore contenente i numeri delle righe occupate dagli elementi di $y$ nella matrice $X$ .
<b>mean</b> ( $X$ )	vettore contenente i valori medi delle colonne di $X$

# m-file (file .m) ed Editor

## 1) Creare un m-file

File → New File

## 2) Aprire un m-file esistente

File → Open File selezionare un m-file

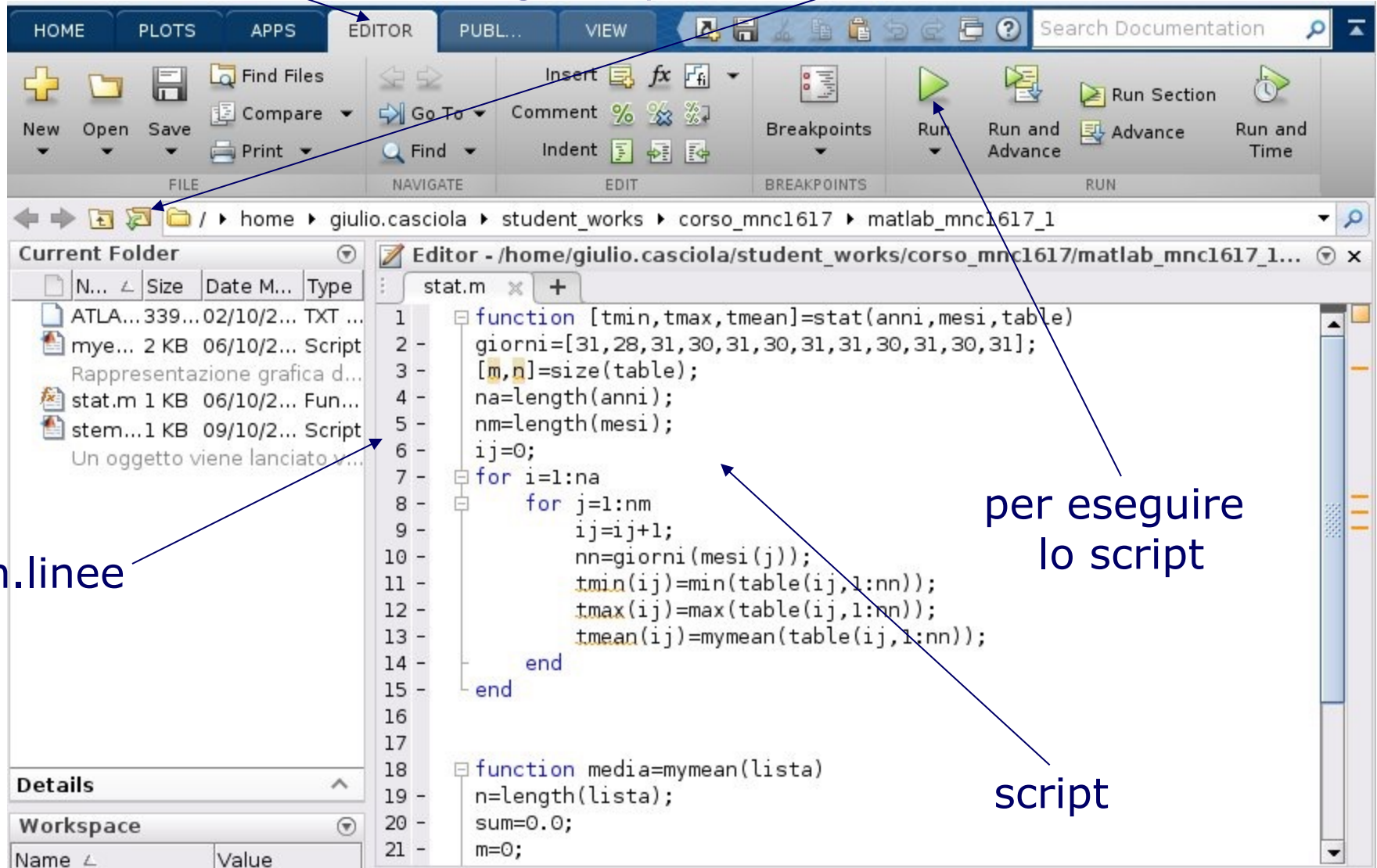
A seguito di questi si apre l'Editor di Matlab, un editor full-screen

**Attenzione:** usare la cartella "matlab\_mnc2324\_1" per conservare questi esempi; si renda corrente questa cartella in Matlab/Octave.

# Editor

bottoni

sfogliare per selezionare la cartella corrente



# Esempio

Problemi che possiamo risolvere usando  
built-in-function di MATLAB/Octave

- Si realizzi uno script contenente istruzioni/comandi Matlab/Octave e lo si salvi in un m-file;
- Esecuzione dello script richiamandolo o eseguendolo.

# Esercizio

Fare uno script per calcolare il determinante di una matrice.

- Definire una matrice  $A$  4x4 di numeri reali
- Calcolare il determinante di  $A$  (built-in function `det` )
- Salvare lo script come M-file di nome `sdet.m`
- Chiamare/eseguire lo script: `sdet`



# Creare uno script

Usare l'Editor per scrivere il seguente script:

```
%script di esempio  
A=[1,2,3,4;0,1,2,3;0,0,1,2;0,0,0,1];  
d=det(A);  
%stampa del risultato  
d
```

- Salvare il programma usando 'salva con nome' e dando 'sdet' come nome
- Salvare il file nella cartella che abbiamo reso corrente in MATLAB/Octave
- Se la cartella non è stata riconosciuta da MATLAB/Octave renderla corrente con cd o il File Browser

# Eseguire uno script

Dalla Command Window dare il comando:

```
>>sdet
```

- MATLAB/Octave cerca nelle cartella corrente se esiste un M-file con questo nome
- Se lo trova lo carica in memoria e lo esegue, cioè esegue istruzione per istruzione il contenuto dello script; nella fattispecie procede così:

```
d =
```

```
    1
```

```
>>
```

# Attenzione



Se abbiamo scritto male qualche comando Matlab segnalerà degli errori sulla command window; nostro compito è capire il messaggio, correggere lo script, salvarlo e rieseguirlo, fino a che Matlab esegua senza errori.

Una volta prodotto un risultato dovremo valutare se è corretto in base a quello che ci aspettavamo; lo script può eseguire senza segnalare errori, ma produrre risultati errati.

# Provate da soli

Seguendo il foglio dell'  
Esercitazione 1: Ambiente Matlab  
realizzare degli script per la  
risoluzione degli esercizi A

# Input ed Output

- Ci sono due classi distinte di funzioni di input ed output.
  - Il primo insieme è relativo alle funzioni disponibili in MATLAB/Octave.
  - Il secondo insieme è relativo alle funzioni della **standard I/O library** usata nel linguaggio C che offre maggior flessibilità e controllo.

# Output

- Per stampare il valore di una variabile (senza che venga stampato il suo nome), si usi la funzione **disp**.

```
>> disp(pi)
```

```
3.1416
```

Si noti che l'output di **disp** finisce sempre con un newline.

- Esempi:

```
>> disp(X)    visualizza il contenuto dell'array, senza  
               visualizzare il nome dell'array
```

```
>> disp('The value of X is:') visualizza la stringa  
    The value of X is:
```

```
>>
```

# Formato di Output

- Il comando **format** permette un certo controllo su come Matlab stampa i valori con la funzione **disp**
- Opzioni del comando **format**
  - Controlla il formato dell'output prodotto da **disp** e dà un **echoing** normale. Opzioni valide:

• short	5/6 cifre	3.1416
• long	15 cifre	3.141592653589793
• short e	5/6 cifre	3.1416e+00
• long e	15 cifre	3.14159265358973e+00
• short E	5/6 cifre	3.1416e+00
• long E	15 cifre	3.14159265358973e+00

# Input

Per leggere un valore ed assegnarlo ad una variabile si usa la funzione `input`, utile per un input interattivo.

`R = input('How many apples')` dà all'utente il prompt visualizzando il testo e attende l'input da tastiera.

L'input può essere ogni espressione MATLAB/Octave, che viene valutata, usando le variabili nel workspace corrente, e il risultato è memorizzato in R. Se l'utente preme il tasto return senza digitare nulla, `input` restituisce una matrice vuota.

`R = input('What is your name','s')` dà all'utente il prompt visualizzando il testo e attende in input da tastiera una stringa. L'input digitato non viene valutato; i caratteri sono semplicemente memorizzati come una stringa MATLAB/Octave.



# I/O standard library

`fscanf` ed `fprintf` del C

```
fprintf(format, args);
```

```
fprintf(fid, format, args);
```

## Esempi:

```
fprintf('%10.7f', val);
```

```
fprintf('%10.7f %10.7f', val1, val2);
```

```
fprintf('%d %20.16e %e \n', n, val1, val2);
```

**Legenda:** `%d` formato intero  
`%f` forma fixed point  
`%e` forma scientifica

# Esercizio

Fare uno script per calcolare il fattoriale di un numero

- Input: n
- Utilizzare la built-in function "factorial"
- Output : fc
- Salvare lo script con nome sfattoriale.m
- eseguire lo script: sfattoriale.m

# Creare uno script

Usare l'Editor per scrivere il seguente script:

```
%script di esempio  
n=input('digita un intero positivo: ');  
fc=factorial(n);  
disp('Il fattoriale è ');  
disp(fc);
```

- Salvare il programma usando `salva` con nome e dando 'sfattoriale' come nome
- Salvare il file in una cartella riconosciuta da MATLAB

# Eeguire uno script

Dalla Command Window dare il comando:

```
>>sfattoriale
```

- MATLAB/Octave cerca nelle directory di sua pertinenza se esiste un m-file con questo nome
- Se lo trova lo carica in memoria e lo esegue, cioè esegue istruzione per istruzione il contenuto dello script; nella fattispecie procede così:

```
digita un intero positivo: _
```

- Il cursore lampeggia in attesa che l'utente digiti un numero come input

```
digita un intero positivo: 7
```

```
Il fattoriale è
```

```
5040
```

```
>>
```

# Modifichiamo lo script

Usare l'Editor per modificare lo script:

```
%script di esempio  
n=input('digita un intero positivo: ');  
fc=factorial(n);  
fprintf('Il fattoriale è %10.7f \n',fc);
```

- Salvare nuovamente lo script utilizzando 'save' e rieseguire.

# Provate da soli

Seguendo il foglio dell'  
Esercitazione 1: Ambiente Matlab  
realizzare degli script per la  
risoluzione degli esercizi B.

## Attenzione

Si suggerisce di iniziare ogni script con i seguenti comandi Matlab:

**clear all** : pulisce il work space

**clc** : pulisce la command window