Laboratorio di Calcolo Numerico

A.A. 2019-2020

Ingegneria Meccanica

Introduzione

a

MatLab

Docente: Domenico Vitulano

Email: domenico.vitulano@sbai.uniroma1.it

Ufficio: Via A. Scarpa 16,

Pal. RM2, I piano, Stanza n. 13

Tel. 06 4976 6633

Ricevimento: consultare la pagina web dedicata al corso

Il materiale didattico e disponibile sui siti:

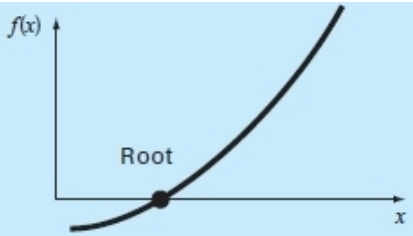
https://www.sbai.uniroma1.it/users/vitulano-domenico

https://elearning.uniroma1.it/

Programma

Roots of equations

Solve $f(x) = 0$ for x.



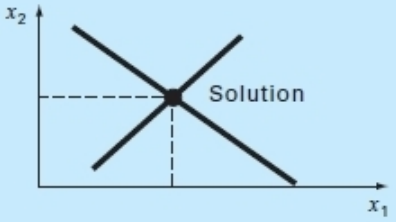
Linear algebraic equations

Given the a's and the c's, solve

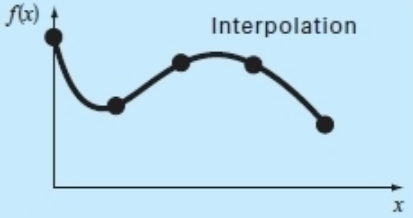
$a_{11}x_1 + a_{12}x_2 = c_1$

$a_{21}x_1 + a_{22}x_2 = c_2$

for the x's.



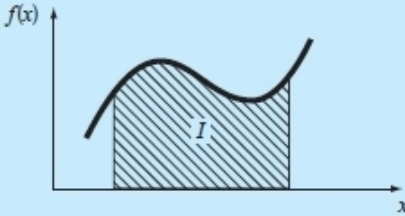
Curve fitting



Integration

$I = \int_a^b f(x)\,dx$

Find the area under the curve.



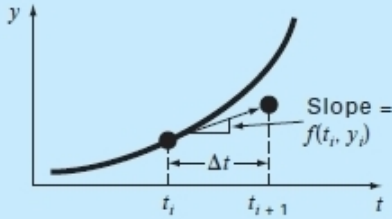
Ordinary differential equations

Given

$\frac{dy}{dt} \simeq \frac{\Delta y}{\Delta t} = f(t, y)$

solve for y as a function of t.

$y_{i+1} = y_i + f(t_i, y_i)\,\Delta t$

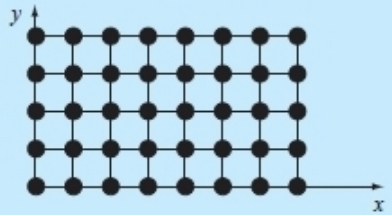


Partial differential equations

Given

$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y)$

solve for u as a function of x and y



Introduzione
a
MatLab

Matlab

Matlab (MATrix LABoratory) è un sistema software integrato per il calcolo tecnico e scientifico

- Linguaggio di programmazione ad alto livello interpretato, con particolari facilitazioni nelle elaborazioni di matrici

- Contiene i costrutti tipici dei linguaggi di programmazione
- Possiede un ampio insieme di tipi di dato predefiniti
- Supporta la programmazione orientata agli oggetti
- Usato sia per creare rapidamente piccoli programmi test (programmazione in the small) che per applicazioni più complesse (programmazione in the large)

Matlab

Matlab (MATrix LABoratory) è un sistema software integrato per il calcolo tecnico e scientifico

- Grafici in 2 e 3 dimensioni
- Funzioni per la visualizzazione di vettori e matrici
- Funzioni per l'impostazione dell'aspetto della visualizzazione (annotazioni, colori, linee,…)
- Funzioni per elaborare immagini e creare animazioni
- Funzioni per creare interfacce

Matlab

Matlab (MATrix LABoratory) è un sistema software integrato per il calcolo tecnico e scientifico

- Programmi interni per la risoluzione dei problemi dell'Analisi Numerica
  - contiene funzioni elementari, algoritmi di calcolo, algebra lineare, …
- Pacchetti per svariati tipi di applicazioni (Toolbox) --- l'elaborazione numerica dei segnali e delle immagini, la simulazione di sistemi dinamici, il calcolo simbolico, wavelet, ecc.
- Interazione con altri linguaggi di programmazione (per es. C e Fortran)

Matlab

- Creato da Cleve Moler (Univ. Del New Mexico) alla fine degli anni '70 per fornire agli studenti un facile accesso al software per l'elaborazione di matrici sviluppato in LINPACK e EISPACK

- Proprietà della MathWorks dal 1984, è diventato uno standard nella ricerca, nella didattica e anche nell'industria

- Ambiti applicativi:

  matematica e calcolo numerico; sviluppo di modelli, simulazioni e prototipi; analisi dati; visualizzazione scientifica; applicazioni con interfaccia utente grafica

Octave

- Ambiente integrato per il calcolo scientifico e la visualizzazione grafica

- Distribuito gratuitamente dalla GNU www.octave.org

  http://sourceforge.net/projects/octave/files/Octave%20Windows%20binaries/Octave%203.2.4%20for%20Windows%20MinGW32%20Installer/Octave-3.2.4_i686-pc-mingw32_gcc-4.4.0_setup.exe/download

- E' compatibile con Matlab: la maggior parte dei programmi Matlab possono essere eseguiti in ambiente Octave senza necessità di modifiche (e viceversa)
- Ha un' interfaccia grafica diversa da Matlab

Installazione Matlab

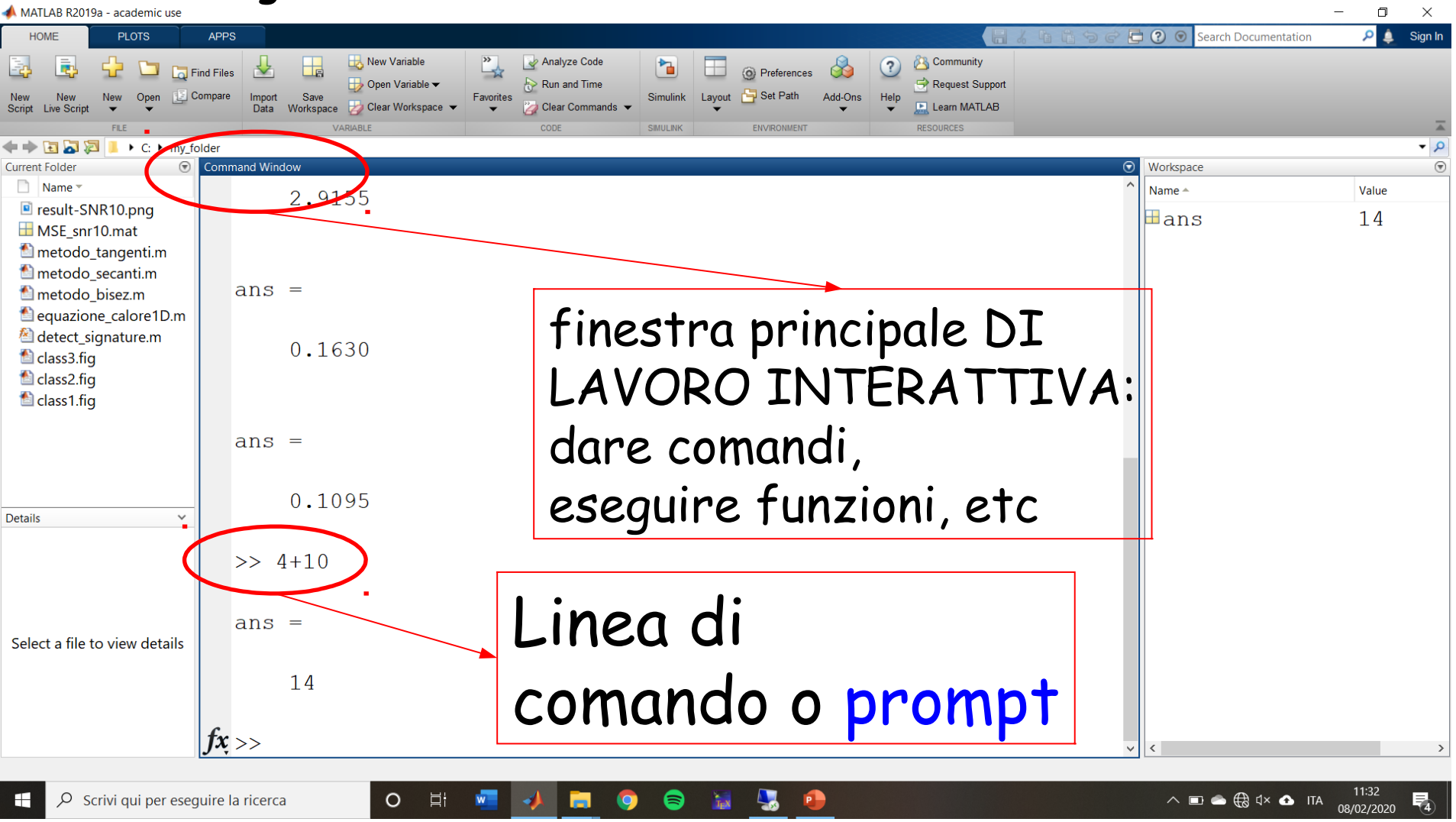
Matlab fa parte del pacchetto di software gratuiti a disposizione degli studenti Sapienza.

1. Visitare la pagina
   https://www.uniroma1.it/it/pagina/software-gratuito
2. cliccare su Vaial PortaleMATLAB per scaricare l'installer di Matlab
3. cliccare sull'installer e seguire la procedura guidata per l'installazione. Verrà richiesta la creazione di un account MathWorks per il quale occorre utilizzare la propria e-mail istituzionale

NOTA : Se si utilizza una rete diversa da quella Sapienza, è necessario contattare licenze.campus@uniroma1.it e richiedere l'Activation key

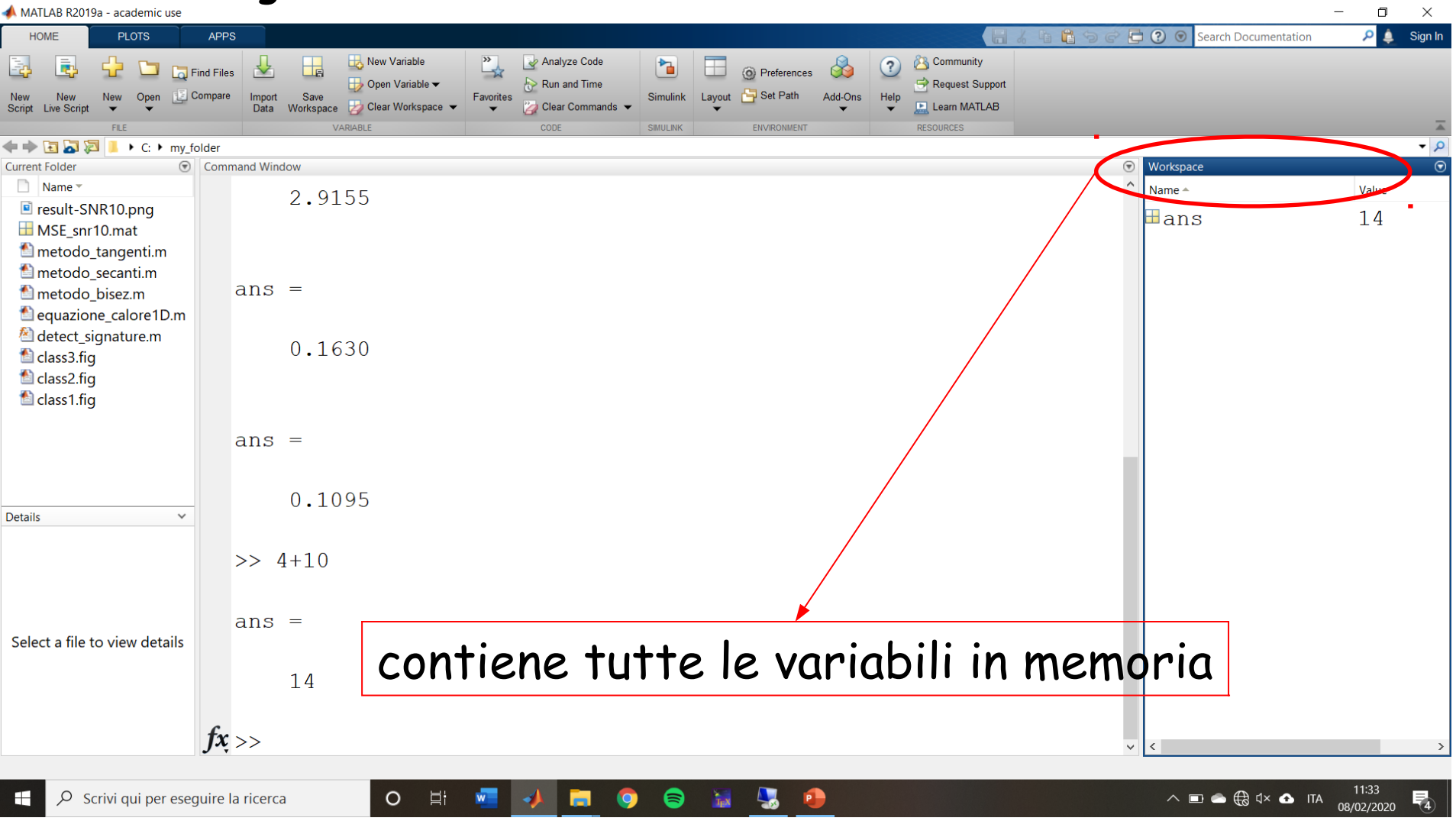
Matlab

Interfaccia grafica: Finestre



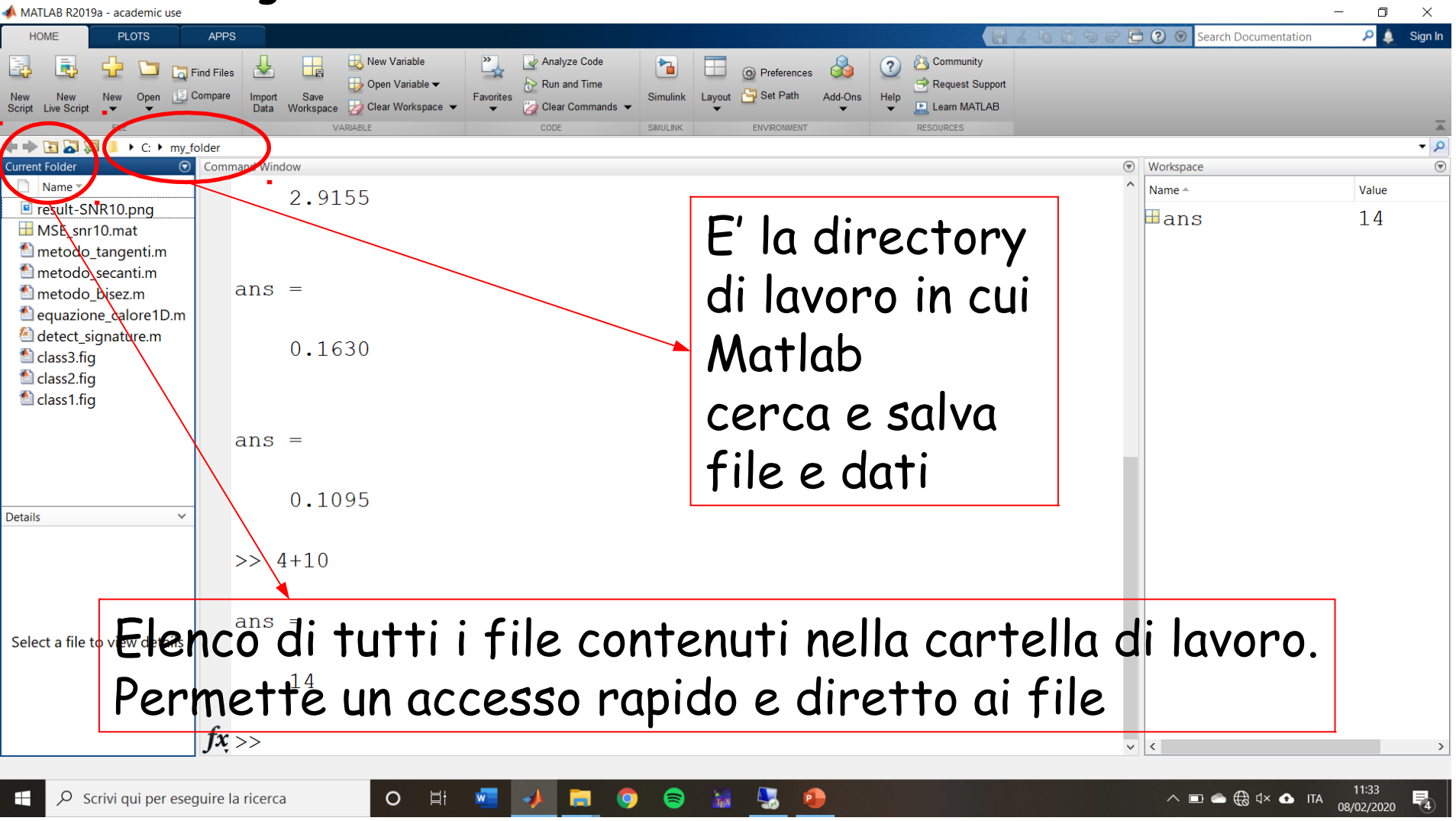
Matlab

Interfaccia grafica: Finestre



Matlab

Interfaccia grafica: Finestre

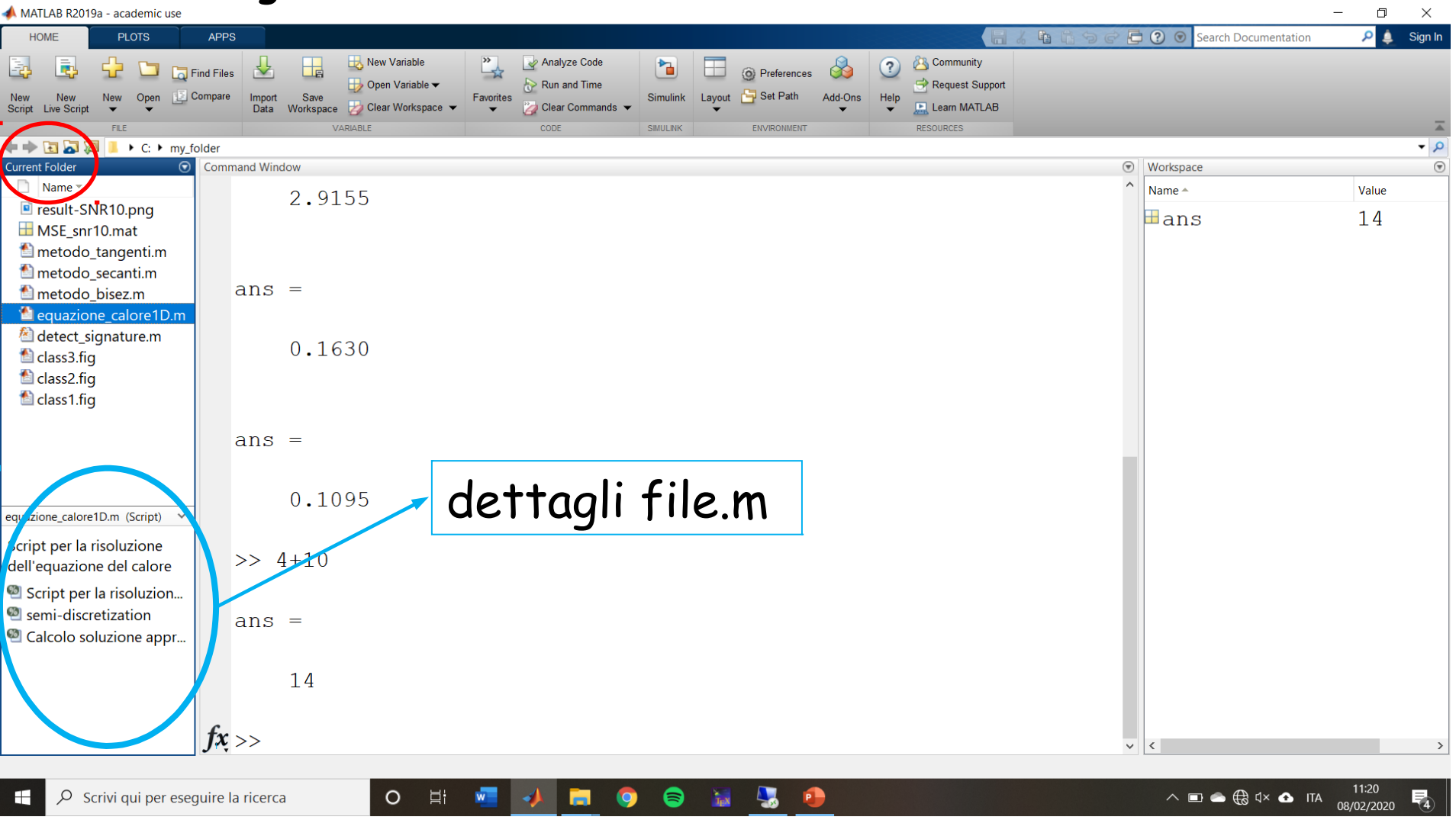


E' la directory di lavoro in cui Matlab cerca e salva file e dati

Elenco di tutti i file contenuti nella cartella di lavoro. Permette un accesso rapido e diretto ai file

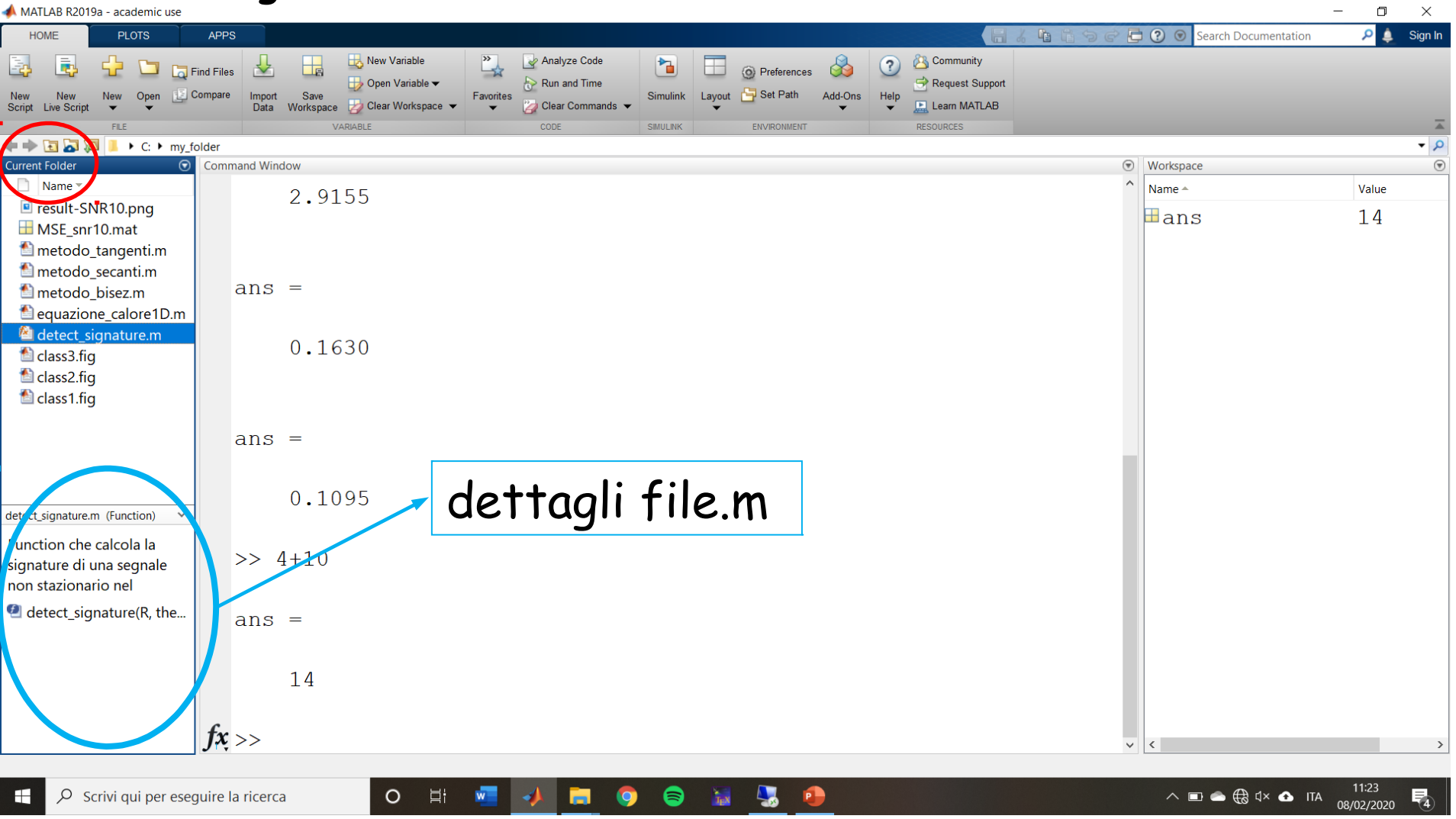
Matlab

Interfaccia grafica: Finestre



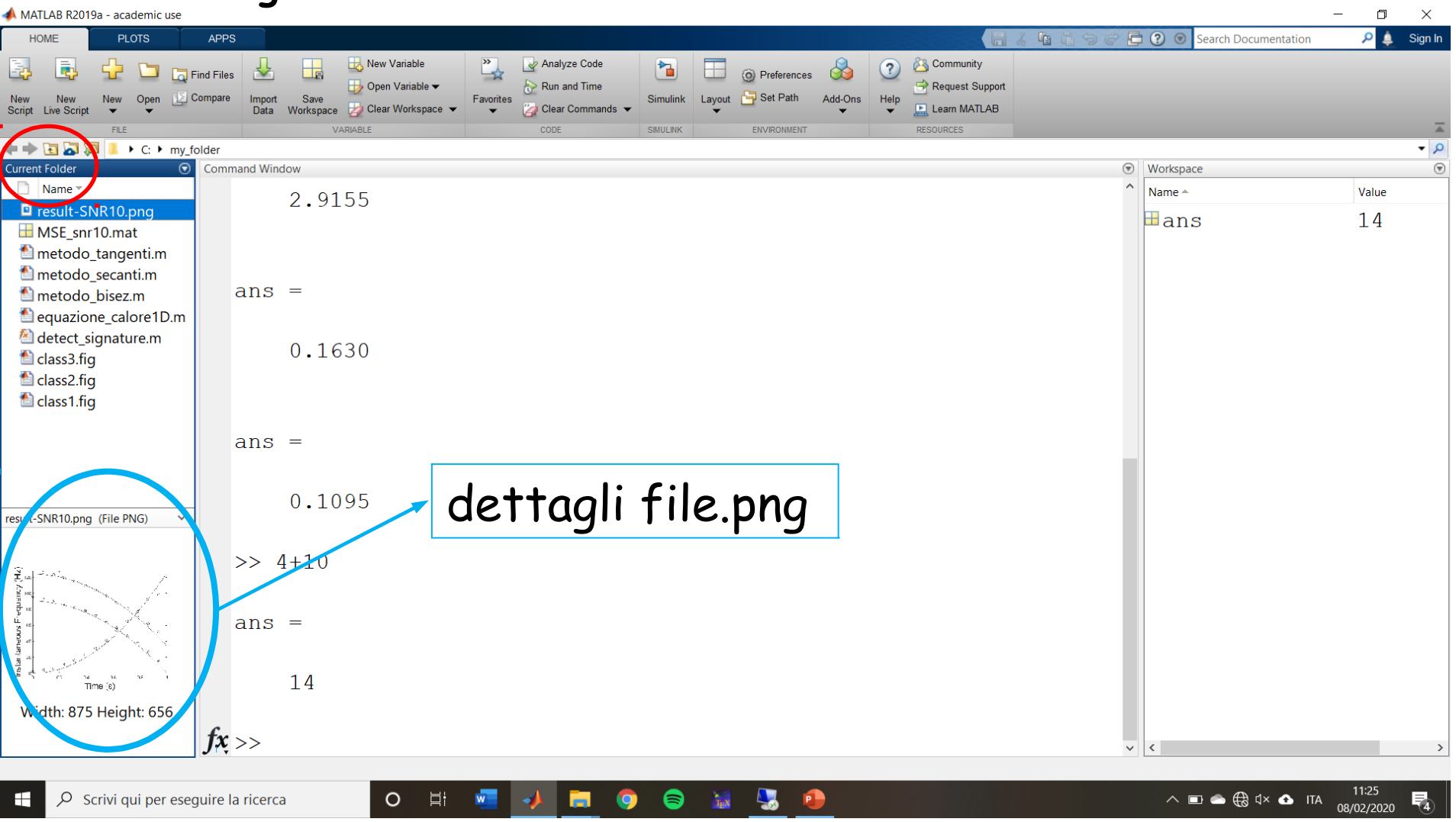
Matlab

Interfaccia grafica: Finestre



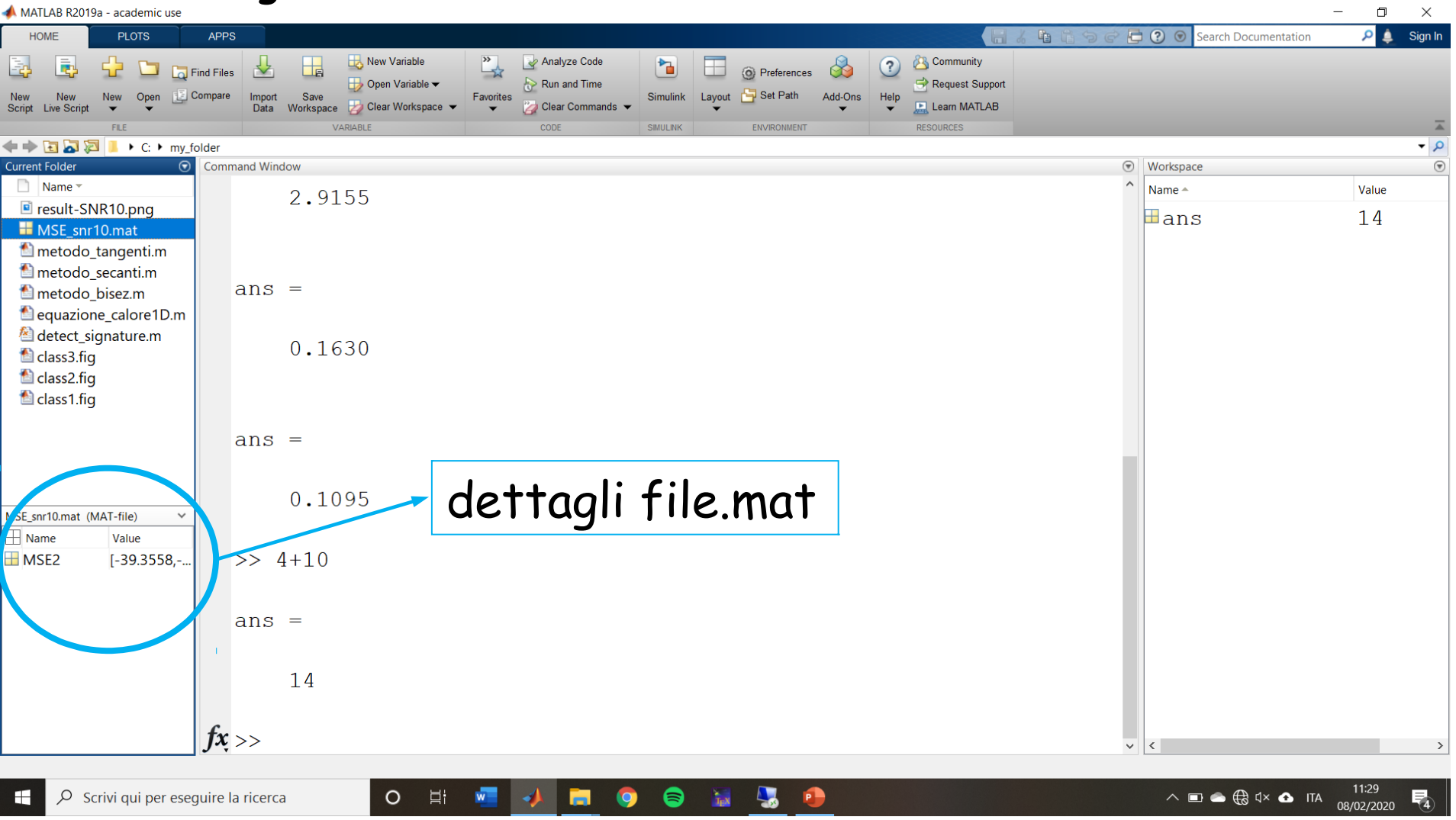
Matlab

Interfaccia grafica: Finestre



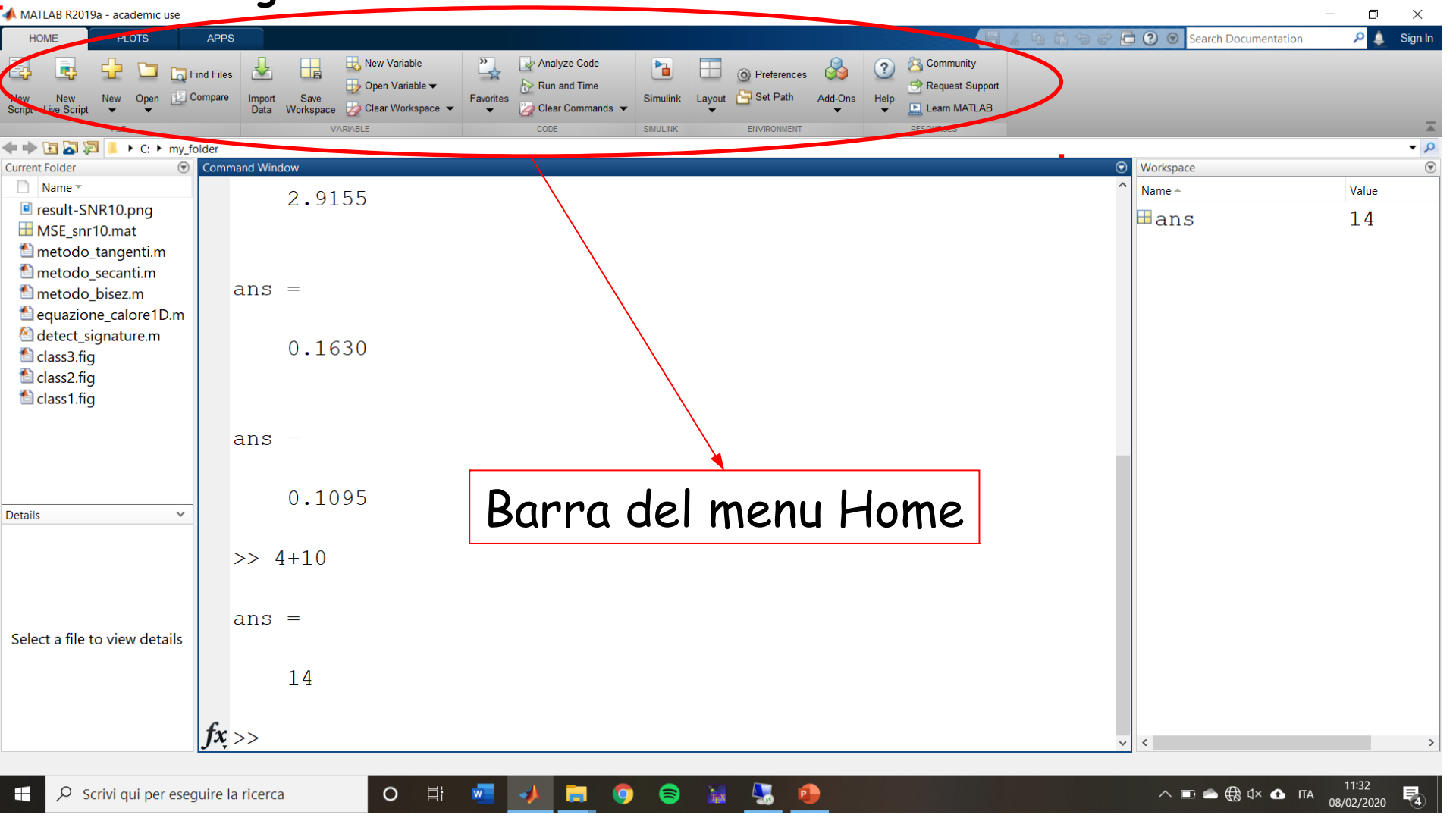
Matlab

Interfaccia grafica: Finestre



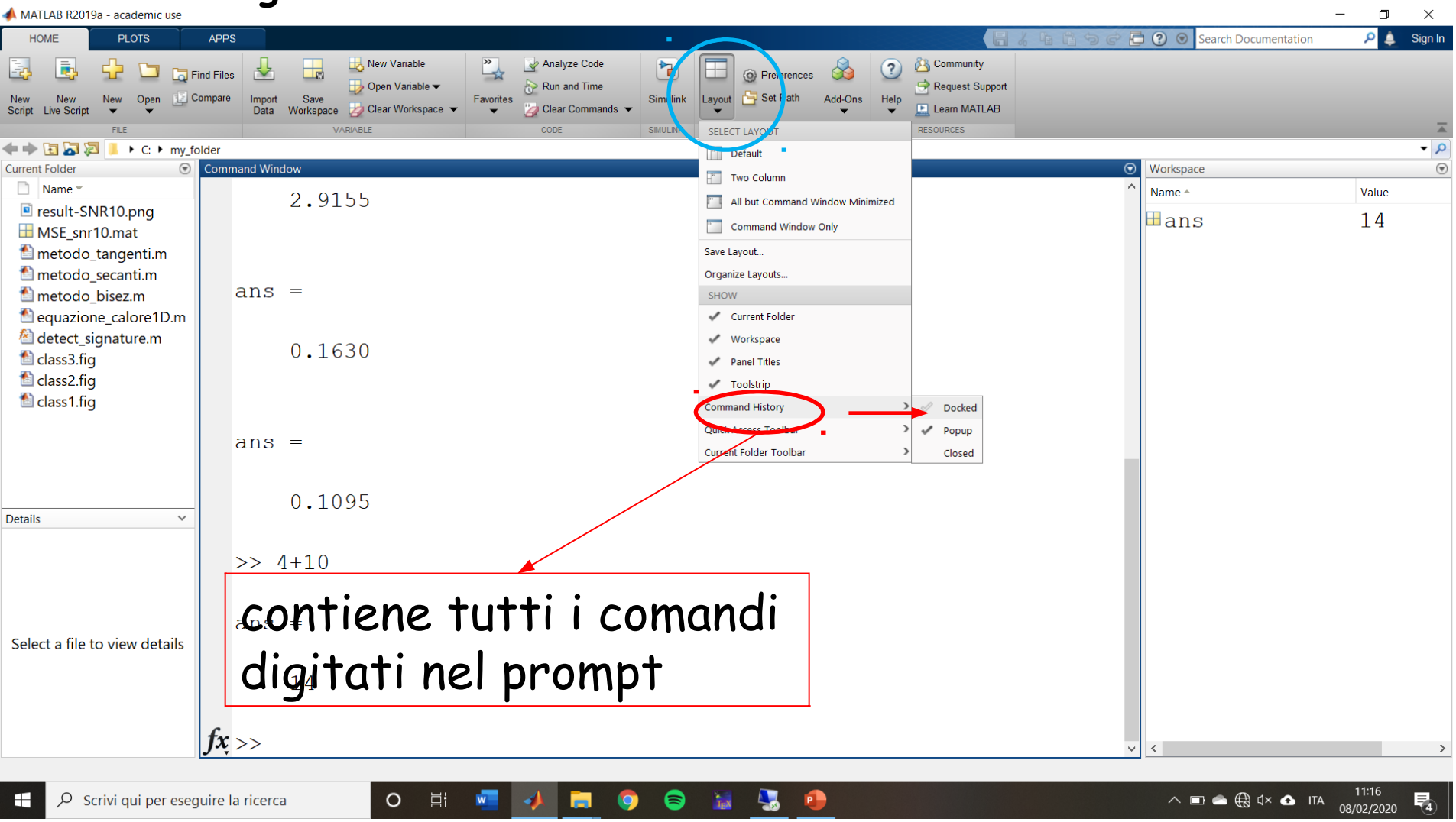
Matlab

Interfaccia grafica: Finestre



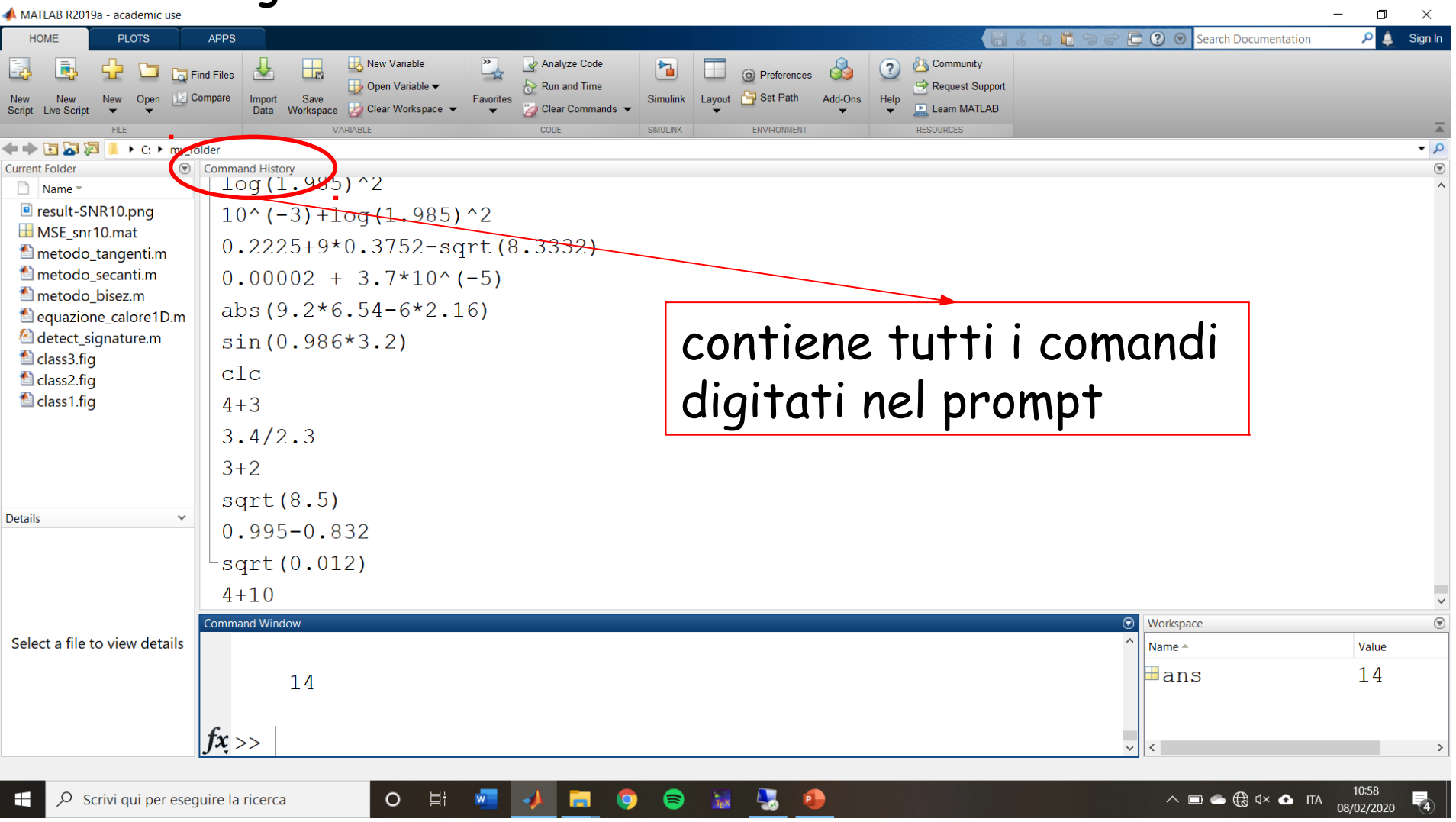
Matlab

Interfaccia grafica: Finestre



Matlab

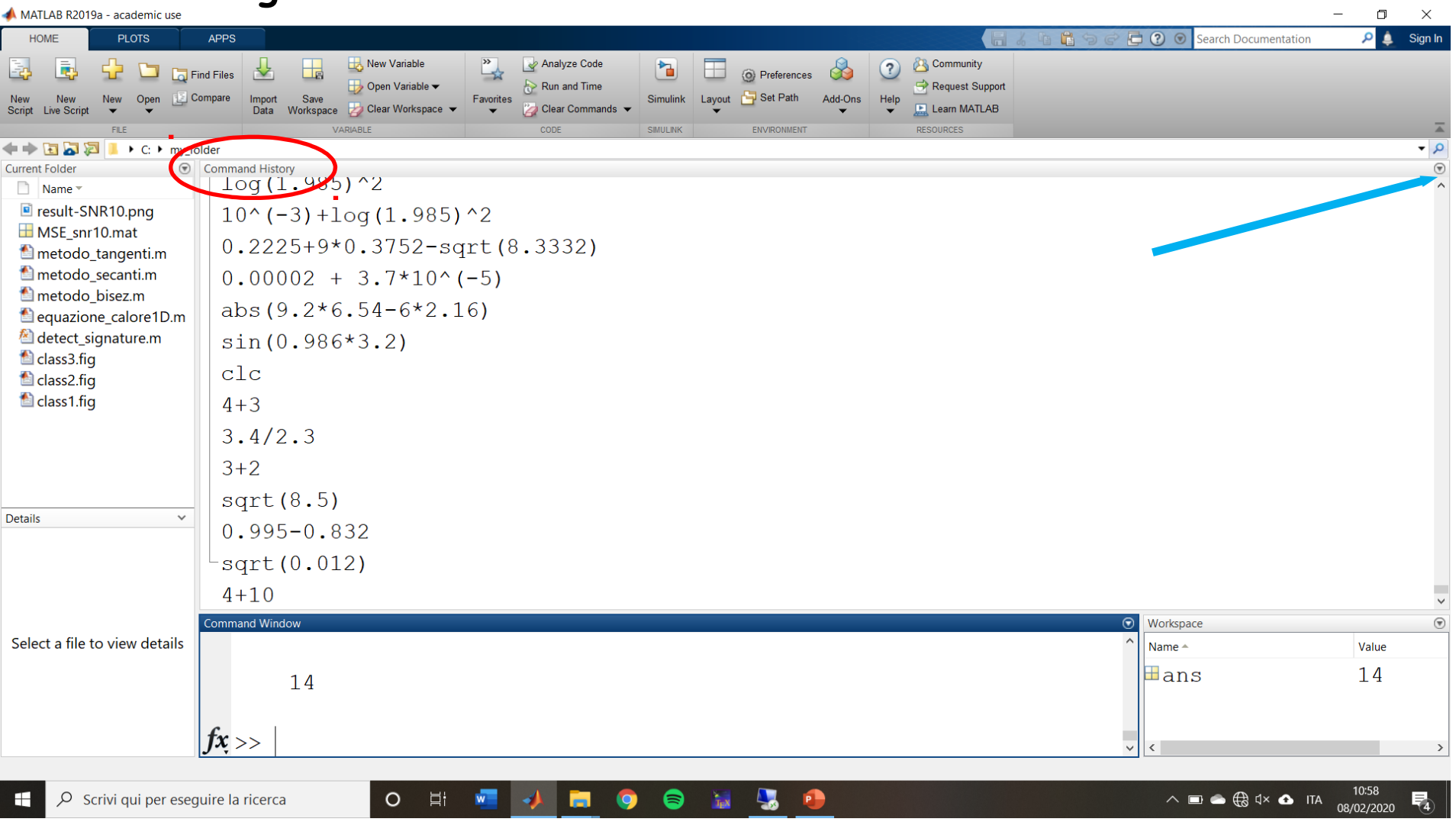
Interfaccia grafica: Finestre



contiene tutti i comandi digitati nel prompt

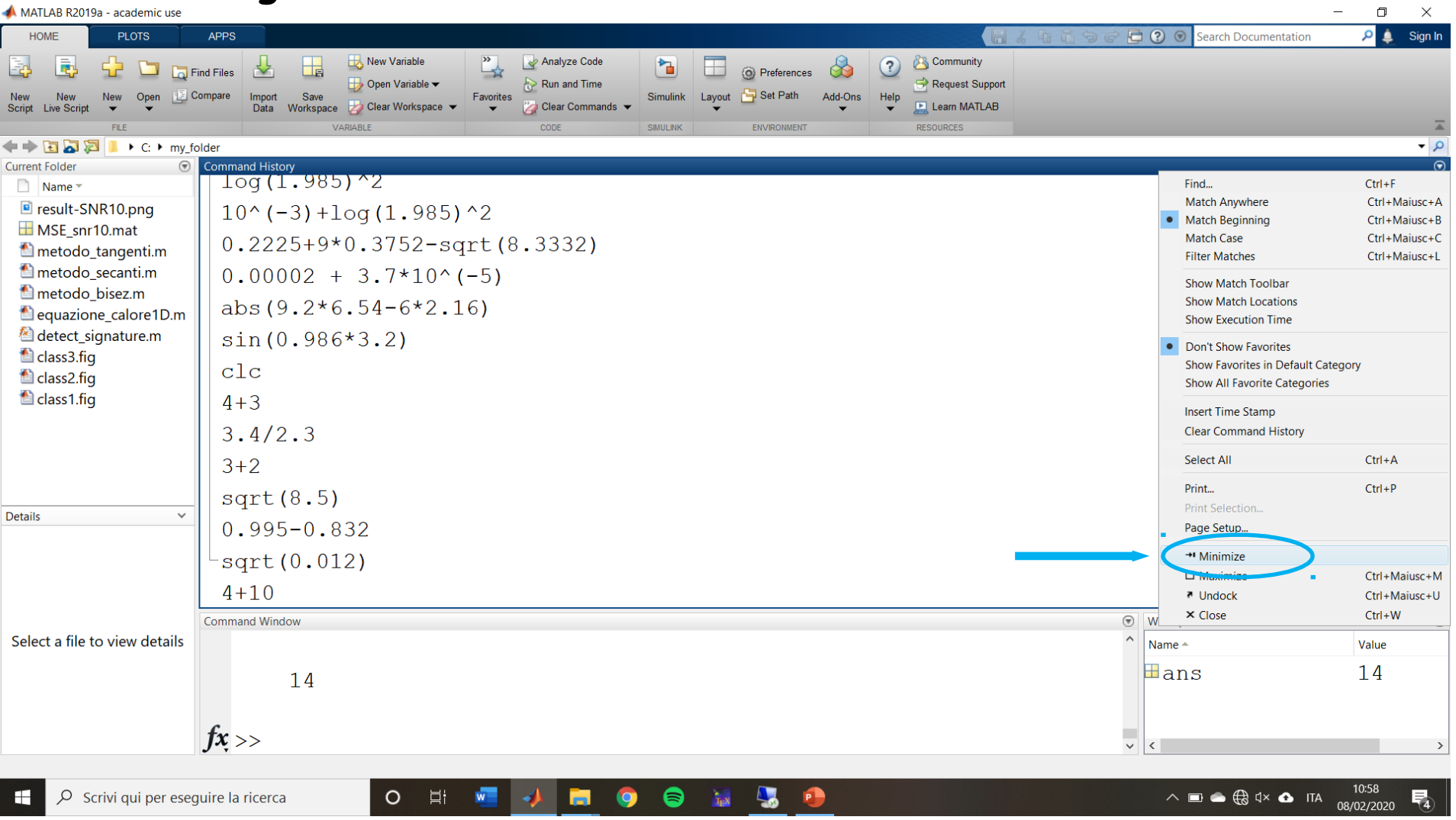
Matlab

Interfaccia grafica: Finestre



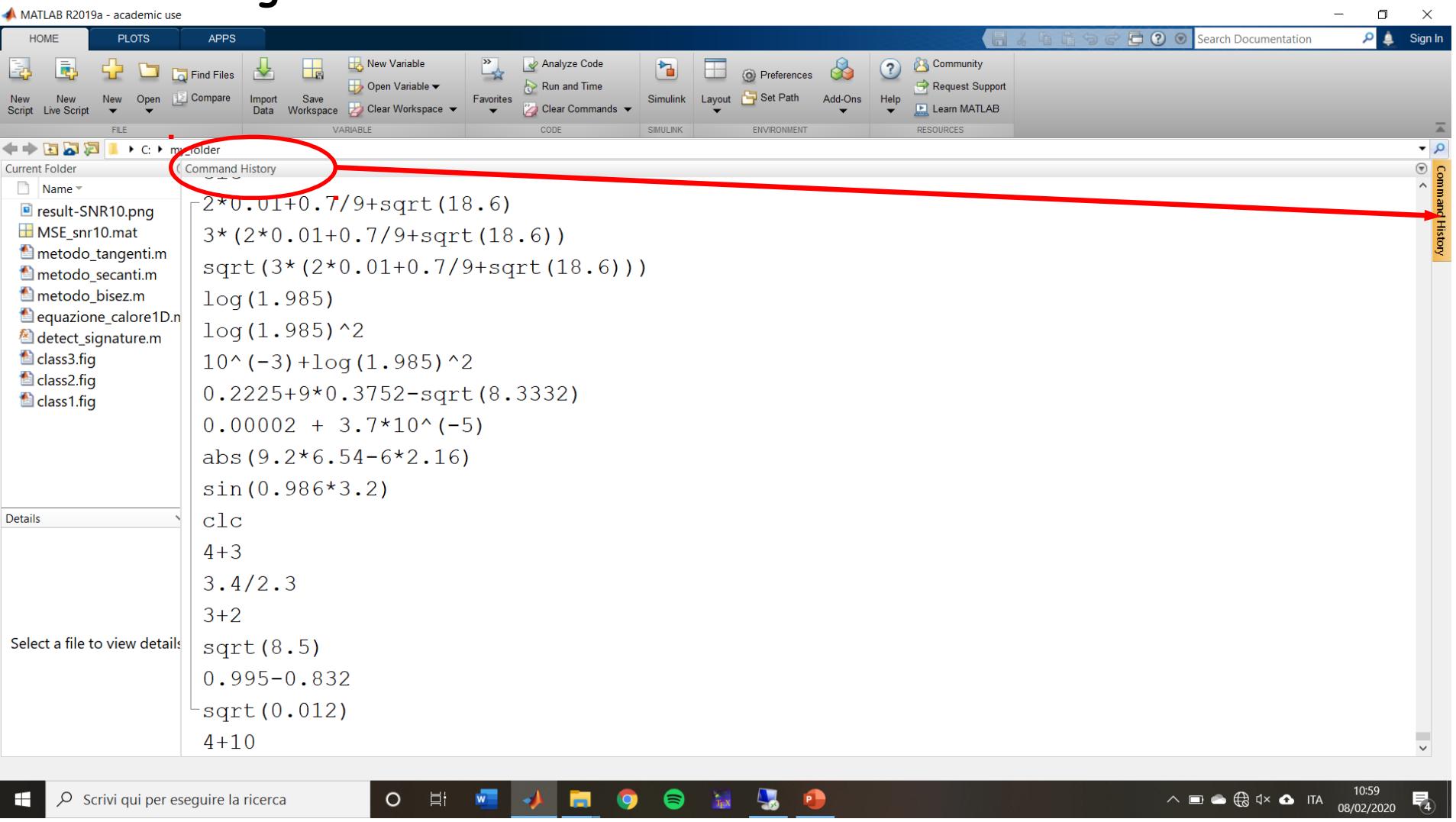
Matlab

Interfaccia grafica: Finestre



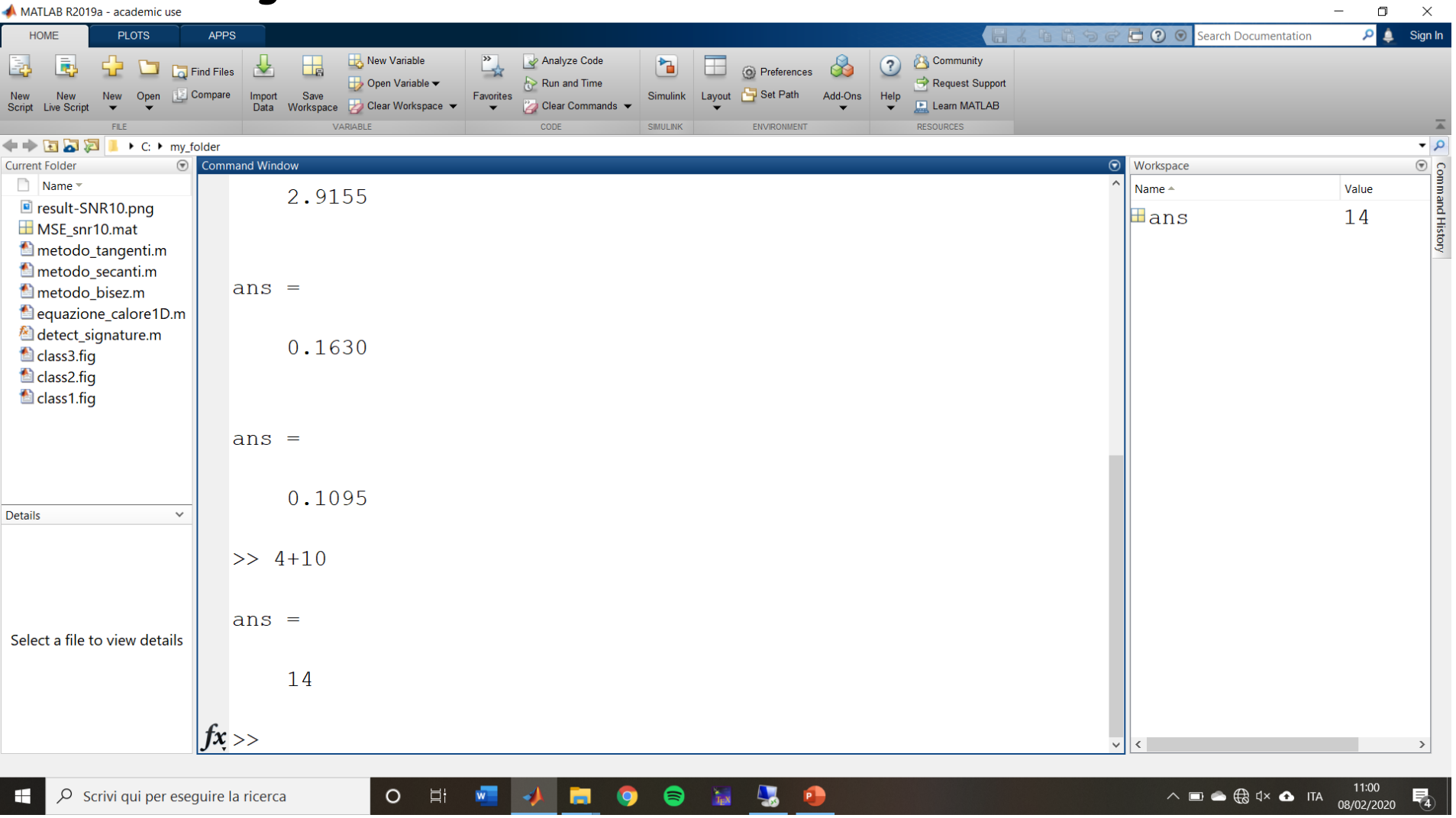
Matlab

Interfaccia grafica: Finestre



Matlab

Interfaccia grafica: Finestre



Matlab

Interfaccia grafica: Command window

Matlab lavora in modo interattivo, cioè l'utente digita una istruzione ed ha immediatamente la risposta. Il prompt su cui si digita l'istruzione è la coppia di caratteri >>

```
>>   comando
```
(Per eseguire, digitare Enter)

Esempio

```
>> 4+10   (Enter)

ans =

     14

>>
```

Matlab

Interfaccia grafica: Command window

- Per uscire dalla sessione di lavoro interattiva usare il comando:

```
>> quit
```

- Per cancellare il contenuto della finestra usare il comando:

```
>> clc
```

- Per ripetere le ultime operazioni effettuate usare i tasti: ↑ e ↓

- Più comandi sulla stessa riga devono essere separati da una virgola

```
>> 3+2, 5*10-4, 55-22
```

- Un'istruzione molto lunga si può scrivere su più righe consecutive usando ...

```
>> 3+2+4+(5*3)-5 ...
-5+10*3 (Enter)
```

Matlab come calcolatrice

Operatori

- operazioni elementari

  somma + differenza -

  prodotto * divisione /

Esempi

- >> 3+5

  ans =

  8

  >> 36/3*2

  ans =

  24

  >> 36/(3*2)

  ans =

  6

Matlab come calcolatrice

Operatori

- operatori relazionali

maggiore `>` maggiore o uguale `>=`

minore `<` minore o uguale `<=`

uguale `==` diverso `~=`

Esempi

```
>> 5<3 (Enter)
ans =
  logical
   0
>> 2*3 == 6 (Enter)
ans =
  logical
   1
```

Esempi

```
>> -4~=4 (Enter)
ans =
  logical
   1
>> 6<=6 (Enter)
ans =
  logical
   1
```

Matlab come calcolatrice

Operatori

- operatori logici and & or | not ~

Esempi

```
>> (3<4) & (-3>=0)   (Enter)
ans =
  logical
   0

>> (3<4) | (-3>=0)   (Enter)
ans =
  logical
   1
```

Esempi

```
>> (3<4) & (-3>=0)     (Enter)
ans =
  logical
   0

>> (3<4) & ~(-3>=0)   (Enter)
ans =
  logical
   1
```

- elevamento a potenza ^

Esempio

```
>> 2^3    (Enter)

ans =

    8
```

Matlab come calcolatrice

Costanti predefinite

Costante	Costanti Matlab
Infinito	inf
π	pi
Unità immaginaria	i
Numero massimo rappresentabile (2^{1023})	realmax
Numero minimo rappresentabile (2^{-1022})	realmin
Precisione di macchina $(2.220446049250313 \cdot 10^{-16})$	eps
Forma indeterminata	nan

Esempi

```
>> 1/0   (Enter)
ans =
   Inf
>> 2.3/Inf  (Enter)
ans =
     0
>> 2*pi   (Enter)
ans =
    6.2832
>> i^2   (Enter)
ans =
    -1
>> 0/0    (Enter)
ans =
   NaN
```

Matlab come calcolatrice

Costanti predefinite

Costante	Costanti Matlab
Infinito	inf
π	pi
Unità immaginaria	i
Numero massimo rappresentabile (2^{1023})	realmax
Numero minimo rappresentabile (2^{-1022})	realmin
Precisione di macchina $(2.220446049250313 \cdot 10^{-16})$	eps
Forma indeterminata	nan

Rappresentazione dei numeri in Matlab:

forma a virgola mobile (floating point)

su parole di 64 bit (doppia precisione)

Matlab come calcolatrice

Funzioni predefinite

Funzione	Funzioni Matlab
Seno	sin(x)
Coseno	cos(x)
Tangente	tan(x)
Arcsin	asin(x)
Arccos	acos(x)
Arctan	atan(x)
Logaritmo naturale	log(x)
Esponenziale	exp(x)
Valore assoluto	abs(x)
Radice quadrata	sqrt(x)
segno	sign(x)

Esempi

```
>> sin(pi/2)
ans =
     1
>> 180/pi*atan(1)
ans =
    45
>> abs(-1.5)
ans =
     1.5
>> sqrt(-2)
ans =
   0.0000 + 1.4142i
>> log(exp(-3))
ans =
    -3
```

Matlab

Help: per informazioni sulle funzioni di Matlab (vedere anche l' help da menù)

>> **help** nome_funzione

informazioni su una specifica funzione

Esempio: come si usa la funzione log?

```
>> help log

 LOG    Natural logarithm.
    LOG(X) is the natural logarithm of the elements of X.
    Complex results are produced if X is not positive.

    See also LOG2, LOG10, EXP, LOGM.
```

Matlab

Esempio: esiste una funzione che calcola la radice quadrata di un numero?

```
>> lookfor square
cir        - Cox-Ingersoll-Ross(CIR)mean-reverting square root diffusion
             class file
magic      - Magic square.
hypot      - Robust computation of the square root of the sum of squares
realsqrt   - Real square root.
sqrt       - Square root.
lscov      - Least squares with known covariance.
lsqnonneg  - Linear least squares with nonnegativity constraints.
sqrtm      - Matrix square root.
cgs        - Conjugate Gradients Squared Method.
.
.
>> 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.
    Overloaded methods:
       codistributed/sqrt
    Reference page in Help browser
       doc sqrt
```

Matlab

Digitando solo il comando help si ha l' elenco degli argomenti (pacchetti disponibili)

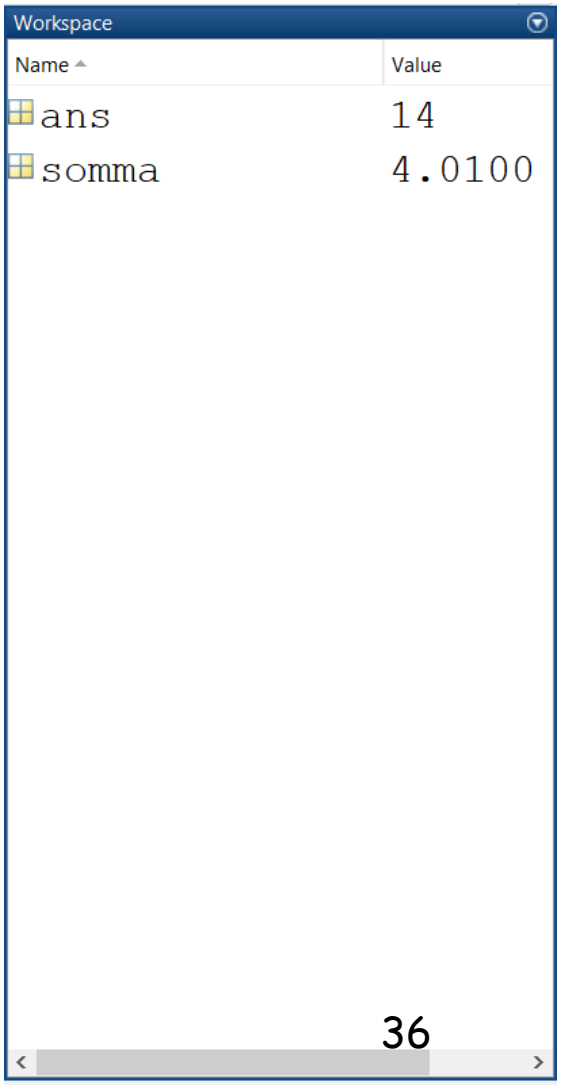
```
>> help

HELP topics:

matlab\general           - General purpose commands.
matlab\ops               - Operators and special characters.
matlab\lang              - Programming language constructs.
matlab\elmat             - Elementary matrices and matrix manipulation.
matlab\randfun           - Random matrices and random streams.
matlab\elfun             - Elementary math functions.
matlab\specfun           - Specialized math functions.
matlab\matfun            - Matrix functions - numerical linear algebra.
matlab\datafun           - Data analysis and Fourier transforms.
matlab\polyfun           - Interpolation and polynomials.
matlab\funfun            - Function functions and ODE solvers.
matlab\sparfun           - Sparse matrices.
```

```
>> help  nome_argomento
```

Produce l'elenco e la descrizione delle funzioni relative all'argomento selezionato

Matlab

Il risultato di una espressione può essere assegnato ad una variabile la quale viene salvata nel workspace e rimane disponibile per successivi utilizzi.



Esempio:

>> somma = 0.22 + 3.8 - 0.01 (Enter)

somma =

4.0100

Se il risultato di una espressione non viene assegnato ad una variabile definita dall'utente, allora viene automaticamente assegnato alla variabile ans (answer)

Esempio:

```
>> 4+10   (Enter)

ans =

     14
```

Matlab

In Matlab non è necessario dichiarare le variabili. Esse vengono automaticamante definite in seguito ad una assegnazione

La assegnazione è data dal comando =

Nota:
1. il nome di una variabile è composto da caratteri alfanumerici
2. il primo deve essere alfabetico
3. c'è differenza tra lettere maiuscole e minuscole

Esempio

```
>> d = 2;
```

attribuisce alla variabile d il valore 2 (verificare nel workspace)

```
>>  c = 4;
```

attribuisce alla variabile c il valore 4

```
>>  b = c * d;
```

attribuisce alla variabile b il prodotto delle variabili c e d

Matlab

Per visualizzare il contenuto di una variabile, basta digitare il suo nome

Esempio: per visualizzare il contenuto di b

```
>> b

b =

     8
```

Oppure usare il comando disp

```
>> disp(b)
     8
```

Se si omette il punto e virgola ; alla fine delle istruzioni di comando, viene visualizzato l'output di ogni istruzione

```
>> b=8*2

b =

    16
```

Matlab

I caratteri char si indicano tra 2 apici

Esempio 1 : attribuire alla variabile A il carattere f

```
>> A = 'f';
>> disp(A)
f
```

Esempio 2 : attribuire alla variabile cognome la stringa Rossi

```
>> cognome = 'Rossi';
>> disp(cognome)
Rossi
```

Matlab

Per cancellare tutte le variabili contenute nel Workspace si usa il comando clear

Esempio:

```
>> b=5;, c=b*2-1;, d= c-b;
>> who
Your variables are:

b  c  d

>> clear
>> who
>>
```

Matlab

Per cancellare solo alcune variabili contenute nel Workspace, il comando clear deve essere seguito dall'elenco dei nomi delle variabili separati da uno spazio

clear b c (cancella solo le variabili b e c)

Esempio:

```
>> b=5;, c=b*2-1;, d= c-b;
>> who
Your variables are:

b  c  d

>> clear b c
>> who
Your variables are:
d
```

Matlab

Esempio:

```
>> b=5;, c=b*2-1;, d= c-b;
>> who
Your variables are:

b   c   d

>> save datilezione
>> clear
>> who
>>
>> load datilezione
>> who
Your variables are:

b   c   d
```

Matlab

E' possibile salvare una o più variabili e riusarle in sessioni successive senza dover rieseguire i comandi con cui sono state create

save nomefile

Salva tutte le variabili contenute nel Workspace nel file nomefile.mat
Il nomefile è scelto dall'utente

load nomefile

Carica tutte le variabili salvate nel file nomefile.mat nel Workspace

Matlab

Per salvare nel file nomefile.mat solo alcune variabili, è necessario elencare tali variabili, separate da uno spazio, dopo il nomefile

save nomefile var1 var2 var3

Salva le variabili var1, var2 e var3 nel file nomefile.mat

Esempio:

```
>> b=5;, c=b*2-1;, d= c-b;
>> save datilezione b c
>> clear b c
>> who
Your variables are:
d
>> load datilezione
>> who
Your variables are:
b  c  d
```

Rappresentazione dei numeri

$$x = \pm(1 + a_{-1}2^{-1} + a_{-2}2^{-2} + \cdots + a_{-t}2^{-t})\, 2^{e}$$

	s	*n*	*t*	Numero totale di bits
Doppia precisione	1	11	52	**64**

$L \leq e \leq U$

	Massimo ($2^{U+1}\,(1-2^{-t-1})$)	*Minimo* (2^{L})
Doppia precisione (U=1023, t=52, L=-1022)	$1.79 \cdot 10^{308}$	$2.2 \cdot 10^{-308}$

Errore di arrotondamento: $\frac{|x - fl(x)|}{|x|} \leq \frac{1}{2}\varepsilon$

$\varepsilon = \beta^{1-t} = 2^{-52}$ `>> eps` `2.220446049250313e-016`

Nota. 53 cifre significative in base 2 corrispondono a **15 cifre significative** in base 10.

Matlab

Matlab ha classi di dati predefinite

- double: numeri in doppia precisione compresi tra -10^{308} e 10^{308} (8 bytes per elemento)
- uint8: interi a 8 bits per elemento senza segno compresi tra 0 e 255 (usato per le immagini)
- uint16: interi a 16 bits per elemento senza segno compresi tra 0 e 65535
- uint32: interi a 32 bits per elemento senza segno compresi tra 0 e 4294967295
- int8: interi a 8 bits per elemento con segno compresi tra –128 e 127
- int16: interi a 16 bits per elemento con segno compresi tra -32768 e 32767
- int32: interi a 16 bits per elemento con segno compresi tra
  -2147483648 e 2147483647
- single: numeri in singola precisione compresi tra - 10^{38} e 10^{38} (4 bytes per elemento)
- char: caratteri (2 bytes per elemento)
- logical: 0 o 1 (1 byte per elemento)

Matlab

Indipendentemente dal sistema di rappresentazione dei numeri in Matlab, l'utente può scegliere il formato di visualizzazione usando il comando format

format nomeformato

Visualizza i numeri secondo il formato nomeformato

Digitare help format per conoscere tutti i formati di visualizzazione disponibili

Attenzione: il comando format non cambia la precisione con cui vengono eseguiti i calcoli !!!

Matlab

Esempi:

```
>> format short      % 4 cifre  dopo la virgola (opzione di default)
>> sqrt(2)
ans =
    1.4142

>> format short e          % forma esponenziale (potenze di 10)
>> sqrt(2)
ans =
  1.4142e+000

>> format long       %      14 cifre dopo la  virgola
>> sqrt(2)
ans =
   1.41421356237310

>> format long e           % forma esponenziale
>> sqrt(2)
ans =
    1.41421356237309e+000
```

Matlab

I nomi delle classi sono anche funzioni che permettono la conversione da una classe ad un'altra

Esempio: se `x` è una variabile double, il comando `int8(x)` converte `x` in una variabile intera a 8 bits

```
>> x=sqrt(2)
x =
    1.414213562373095e+000
>> int8(x)
ans =
     1
>> int8(x*10)
ans =
    14
```

Matlab

Esempio:

```
>> x =
       349021

>> int8(x)
ans =
   127

>> int16(x)
ans =
   32767

>> int32(x)
ans =
       349021
```

Vettori

Un array è un insieme di valori ordinati, secondo uno o più indici, a cui ci si riferisce con un singolo nome di variabile

Un array ad un indice è detto vettore

Un array a due indici è detto matrice

In Matlab si possono definire facilmente vettori e matrici

Le variabili in Matlab hanno una struttura vettoriale, per esempio gli scalari sono matrici di dimensione 1x1

Vettori

Un vettore si definisce elencando le sue componenti separate da uno spazio e racchiudendole tra parentesi quadre [ ]

Vettore riga

```
>>  x = [10 20 30 40]

x =

     10     20     30     40
```

è equivalente a

```
>>  x = [10,20,30,40]

x =

     10     20     30     40
```

In questo caso le componenti sono separate da una virgola

Vettori

Vettore colonna

```
>> x=[10; 20; 30; 40]
x =
     10
     20
     30
     40
```

In questo caso le componenti sono separate da un punto e virgola

Anche per visualizzare il contenuto di variabili che sono vettori si può usare il comando disp

```
>> disp(x)
     10
     20
     30
     40
```

Vettori

Per convertire un vettore riga in uno colonna (e viceversa) si usa il comando ' (apice) che produce il trasposto della variabile a cui è applicato

```
>>  v=x'

v =

     10      20      30      40
```

Per estrarre un elemento di un vettore:

nome_vettore(posizione elemento)

Esempio: estrarre il secondo elemento di v

```
>> v(2)

ans =

     20
```

Nota: Gli indici di un vettore sono sempre numeri interi e strettamente positivi

Vettori - la notazione :

Per estrarre contemporaneamente più di un elemento di un vettore si usa il comando : (colon)

nome_vettore(inizio:fine)

Esempio: estrarre dal primo al terzo elemento di v

```
>> v(1:3)

ans =

     10      20      30
```

Esempio: estrarre dal terzo al quarto elemento di v

```
>> v(3:4)

ans =

     30      40
```

Vettori - la notazione :

Per estrarre contemporaneamente più di un elemento di un vettore non consecutivi ed equispaziati

nome_vettore(inizio:passo:fine)

Esempio: estrarre gli elementi di v di indice pari (passo = 2)

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

ans =

     20     40
```

Esempio: estrarre tutti gli elementi di v di indice pari ma da destra verso sinistra (passo = -2)

```
>> v(end:-2:1)

ans =

     40     20
```

Vettori - la notazione :

Il comando : può essere usato anche per generare vettori

Nome_vettore = (minimo:incremento:massimo)

Nome_vettore= vettore di elementi equispaziati (di una quantità=incremento) nell'intervallo [minimo,massimo]

Esempio: Generare un vettore costituito da elementi compresi tra 1.5 e 2.5 con incremento 0.1

```
>> x=[1.5:0.1:2.5]

x =

    1.5    1.6    1.7    1.8    1.9    2.0    2.1    2.2    2.3    2.4    2.5
```

Esempio: Generare un vettore costituito da elementi compresi tra 100 e 80 con incremento -5

```
>> x=[100:-5:80]

x =

    100     95     90     85     80
```

Se non specificato, l'incremento è da intendersi pari a 1

Vettori

I vettori non vengono dimensionati. La loro dimensione può essere modificata in corso di lavoro

Esempio: Sia x = [3 1 4 5] e si assegni il valore 10 all'ottavo elemento di x

```
>> x =  [3 1 4 5]

x =

     3     1     4     5

>> x(8) = 10

x =

     3     1     4     5     0     0     0    10
```

alle posizioni non definite viene assegnato il valore zero

Matlab

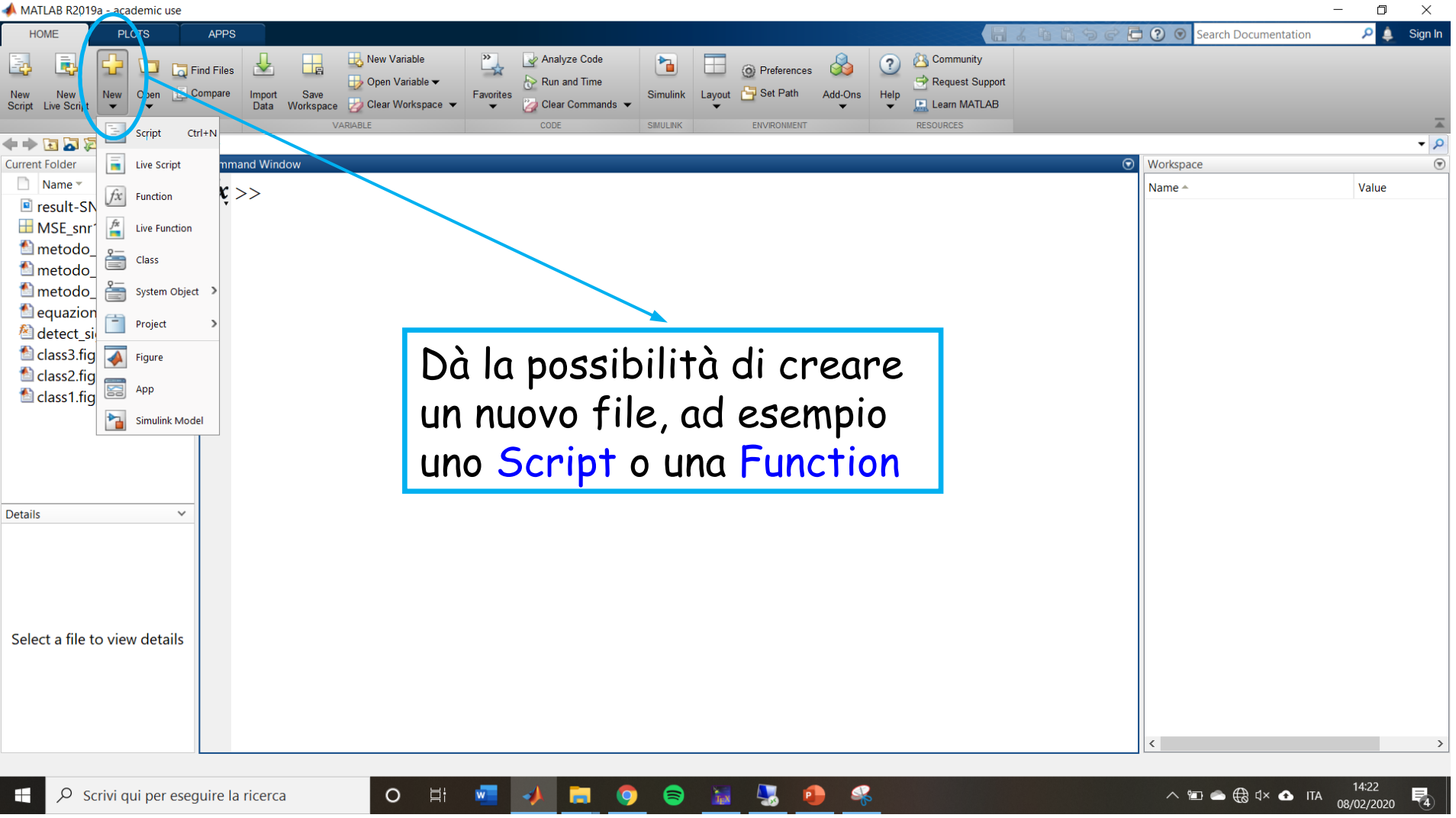
I files contenenti codice Matlab sono chiamati **M-files** e hanno estensione **.m**. Esistono 2 tipi di M-files:

- **Script** :
  1. non accetta argomenti di entrata o di uscita e opera su dati del Workspace;
  2. quando si richiama uno script, Matlab esegue i comandi presenti nel file.
- **Function** :
  1. può accettare argomenti in entrata (**input**) e argomenti in uscita (**output**);
  2. quando una function è opportunamente richiamata, Matlab esegue i comandi presenti nel corpo della function;
  3. una variabile definita nel corpo della function è locale alla function.

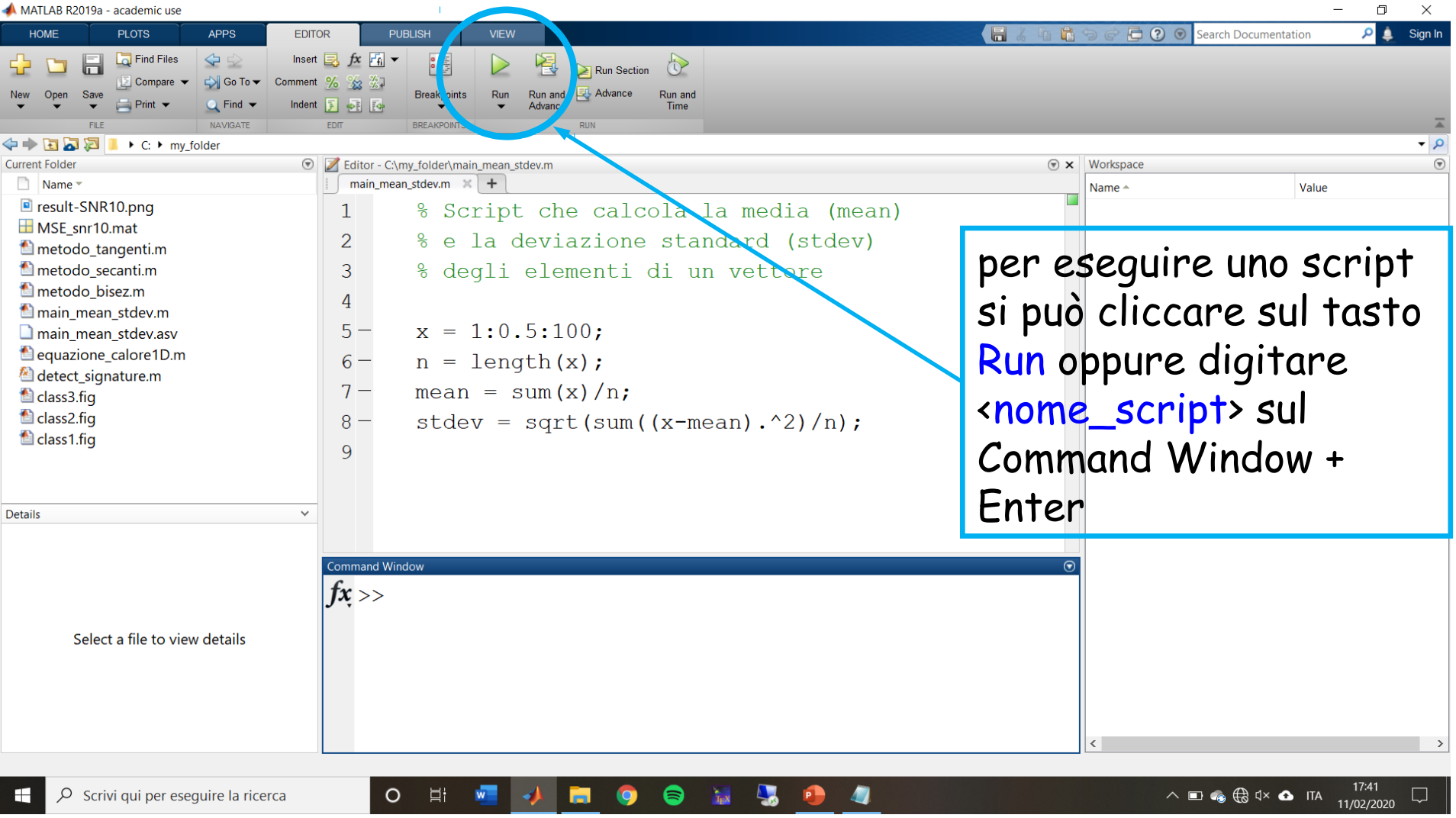
E' possibile organizzare il proprio codice in uno script o in una function che viene opportunamente richiamata.

Attenzione: A parità di contenuto (la sequenza di operazioni da eseguire è la stessa), script e function producono lo **stesso risultato** !!!

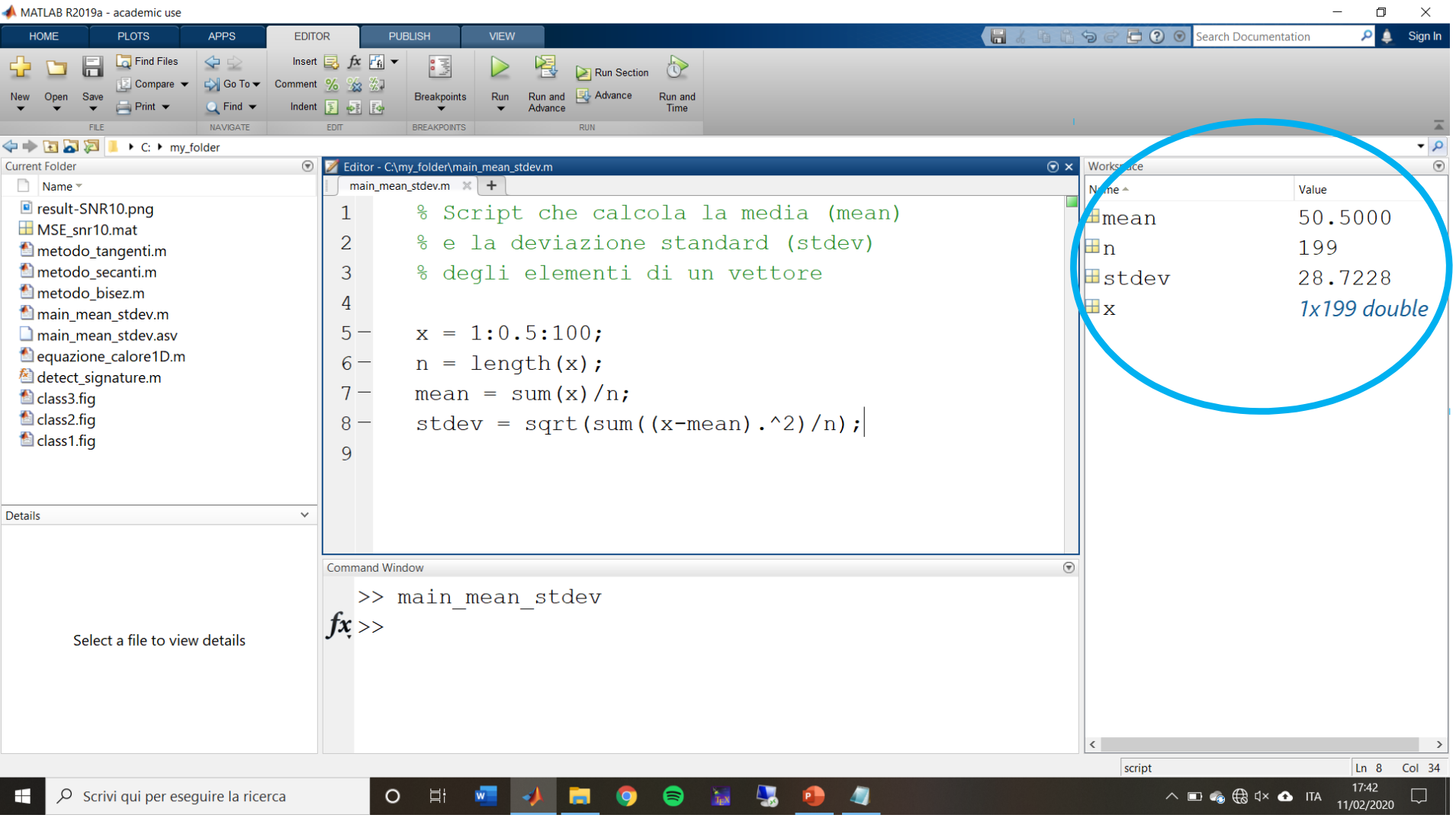
Creazione M-files



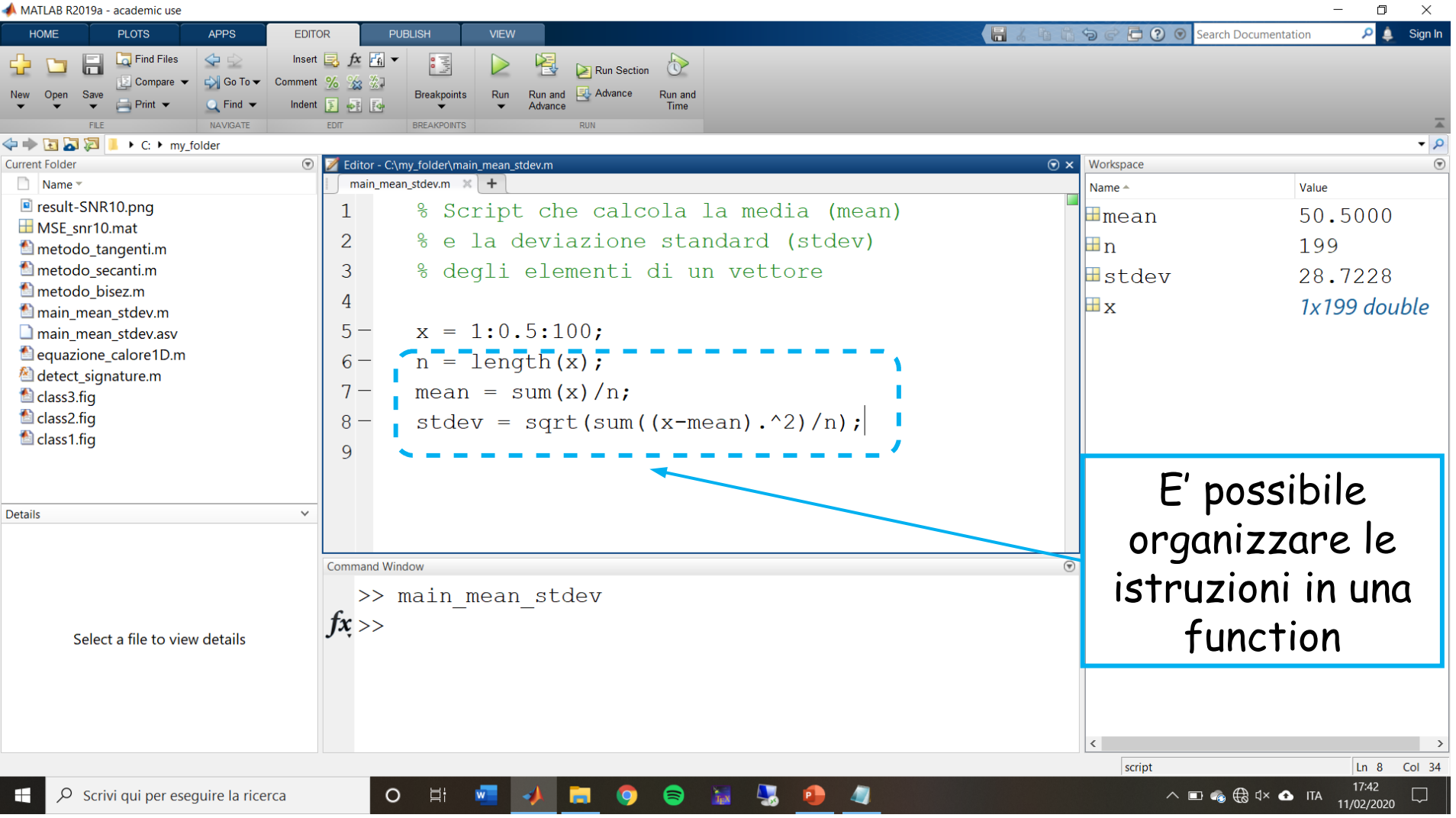
Esempio


MATLAB R2019a - academic use
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
Sign In
New
Open
Save
Find Files
Compare
Print
Go To
Find
Insert
Comment
Indent
Breakpoints
Run
Run and Advance
Run Section
Advance
Run and Time
FILE
NAVIGATE
EDIT
BREAKPOINTS
RUN
C: ▸ my_folder
Current Folder
Name
result-SNR10.png
MSE_snr10.mat
metodo_tangenti.m
metodo_secanti.m
metodo_bisez.m
main_mean_stdev.m
main_mean_stdev.asv
equazione_calore1D.m
detect_signature.m
class3.fig
class2.fig
class1.fig
Details
Select a file to view details
Editor - C:\my_folder\main_mean_stdev.m
main_mean_stdev.m
1 % Script che calcola la media (mean)
2 % e la deviazione standard (stdev)
3 % degli elementi di un vettore
4
5 x = 1:0.5:100;
6 n = length(x);
7 mean = sum(x)/n;
8 stdev = sqrt(sum((x-mean).^2)/n);
9
Command Window
>>
Workspace
Name
Value
per eseguire uno script si può cliccare sul tasto Run oppure digitare <nome_script> sul Command Window + Enter
Scrivi qui per eseguire la ricerca
ITA
17:41
11/02/2020

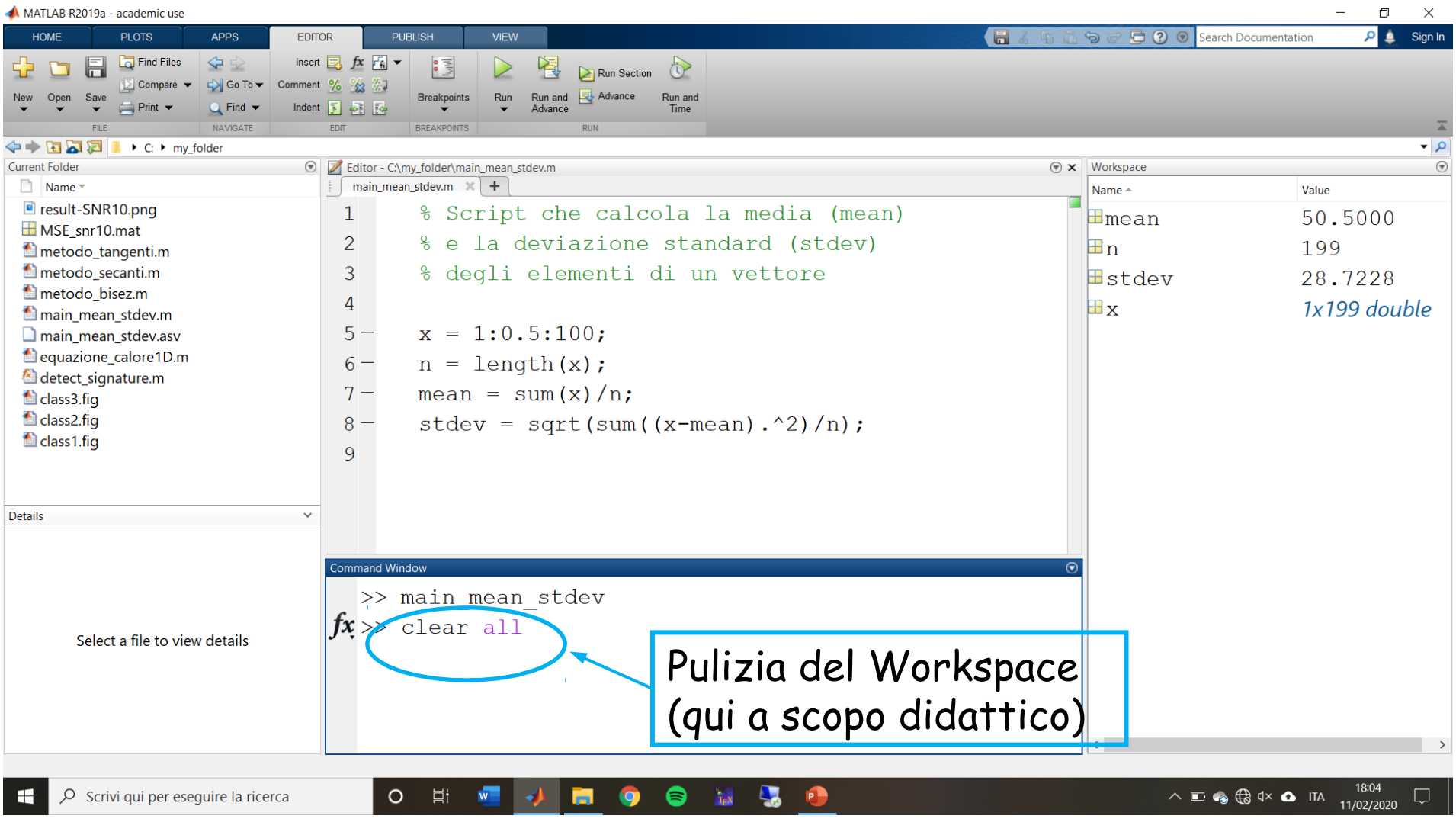
Esempio


MATLAB R2019a - academic use
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
Sign In
New
Open
Save
Find Files
Compare
Print
Go To
Find
Insert
Comment
Indent
Breakpoints
Run
Run and Advance
Run Section
Advance
Run and Time
FILE
NAVIGATE
EDIT
BREAKPOINTS
RUN
C: ▸ my_folder
Current Folder
Name
result-SNR10.png
MSE_snr10.mat
metodo_tangenti.m
metodo_secanti.m
metodo_bisez.m
main_mean_stdev.m
main_mean_stdev.asv
equazione_calore1D.m
detect_signature.m
class3.fig
class2.fig
class1.fig
Details
Select a file to view details
Editor - C:\my_folder\main_mean_stdev.m
main_mean_stdev.m
1 % Script che calcola la media (mean)
2 % e la deviazione standard (stdev)
3 % degli elementi di un vettore
4
5 x = 1:0.5:100;
6 n = length(x);
7 mean = sum(x)/n;
8 stdev = sqrt(sum((x-mean).^2)/n);
9
Command Window
>> main_mean_stdev
>>
Workspace
Name
Value
mean 50.5000
n 199
stdev 28.7228
x 1x199 double
script
Ln 8 Col 34
Scrivi qui per eseguire la ricerca
ITA
17:42
11/02/2020

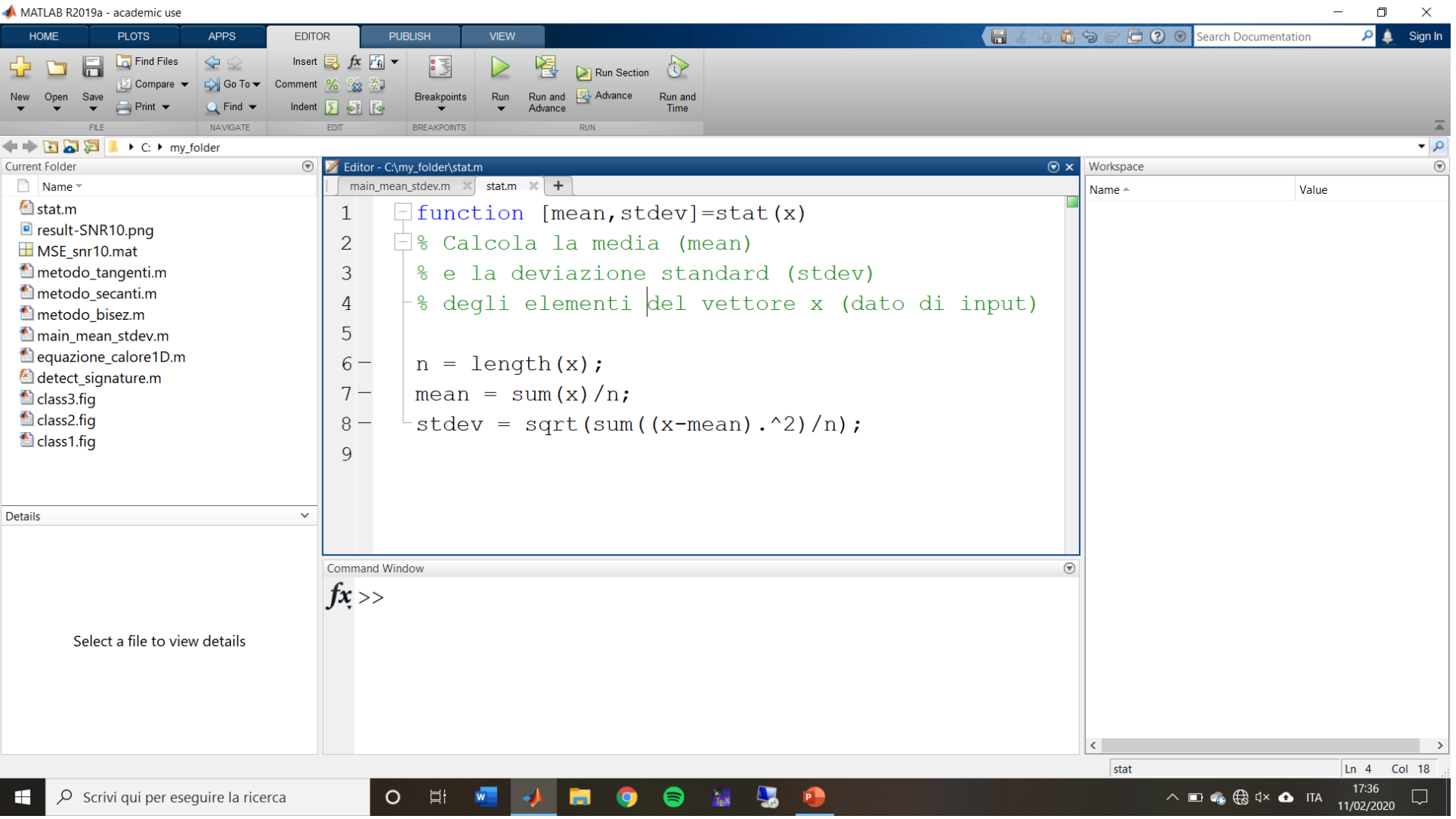
Esempio


MATLAB R2019a - academic use
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
Sign In
New
Open
Save
Find Files
Compare
Print
Go To
Find
Insert
Comment
Indent
Breakpoints
Run
Run and Advance
Run Section
Advance
Run and Time
FILE
NAVIGATE
EDIT
BREAKPOINTS
RUN
C: ▸ my_folder
Current Folder
Name
result-SNR10.png
MSE_snr10.mat
metodo_tangenti.m
metodo_secanti.m
metodo_bisez.m
main_mean_stdev.m
main_mean_stdev.asv
equazione_calore1D.m
detect_signature.m
class3.fig
class2.fig
class1.fig
Details
Select a file to view details
Editor - C:\my_folder\main_mean_stdev.m
main_mean_stdev.m
1 % Script che calcola la media (mean)
2 % e la deviazione standard (stdev)
3 % degli elementi di un vettore
4
5 x = 1:0.5:100;
6 n = length(x);
7 mean = sum(x)/n;
8 stdev = sqrt(sum((x-mean).^2)/n);
9
Command Window
>> main_mean_stdev
>>
Workspace
Name
Value
mean 50.5000
n 199
stdev 28.7228
x 1x199 double
E' possibile organizzare le istruzioni in una function
script
Ln 8 Col 34
Scrivi qui per eseguire la ricerca
ITA
17:42
11/02/2020

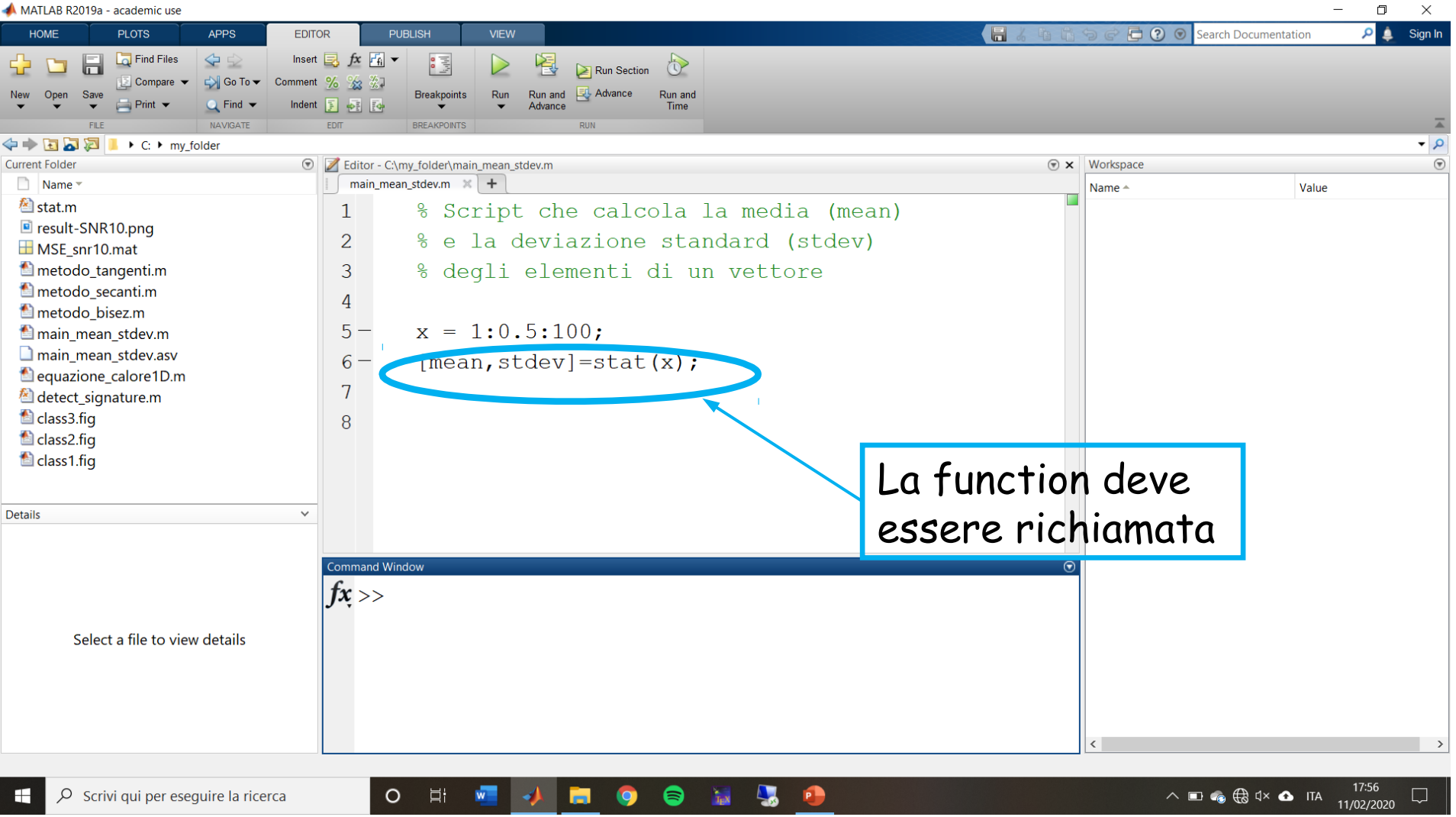
Esempio


MATLAB R2019a - academic use
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
Sign In
New
Open
Save
Find Files
Compare
Print
Go To
Find
Insert
Comment
Indent
Breakpoints
Run
Run and Advance
Run Section
Advance
Run and Time
FILE
NAVIGATE
EDIT
BREAKPOINTS
RUN
C: ▸ my_folder
Current Folder
Name
result-SNR10.png
MSE_snr10.mat
metodo_tangenti.m
metodo_secanti.m
metodo_bisez.m
main_mean_stdev.m
main_mean_stdev.asv
equazione_calore1D.m
detect_signature.m
class3.fig
class2.fig
class1.fig
Details
Select a file to view details
Editor - C:\my_folder\main_mean_stdev.m
main_mean_stdev.m
1 % Script che calcola la media (mean)
2 % e la deviazione standard (stdev)
3 % degli elementi di un vettore
4
5 x = 1:0.5:100;
6 n = length(x);
7 mean = sum(x)/n;
8 stdev = sqrt(sum((x-mean).^2)/n);
9
Command Window
>> main_mean_stdev
>> clear all
Workspace
Name
Value
mean 50.5000
n 199
stdev 28.7228
x 1x199 double
Pulizia del Workspace
(qui a scopo didattico)
Scrivi qui per eseguire la ricerca
ITA
18:04
11/02/2020

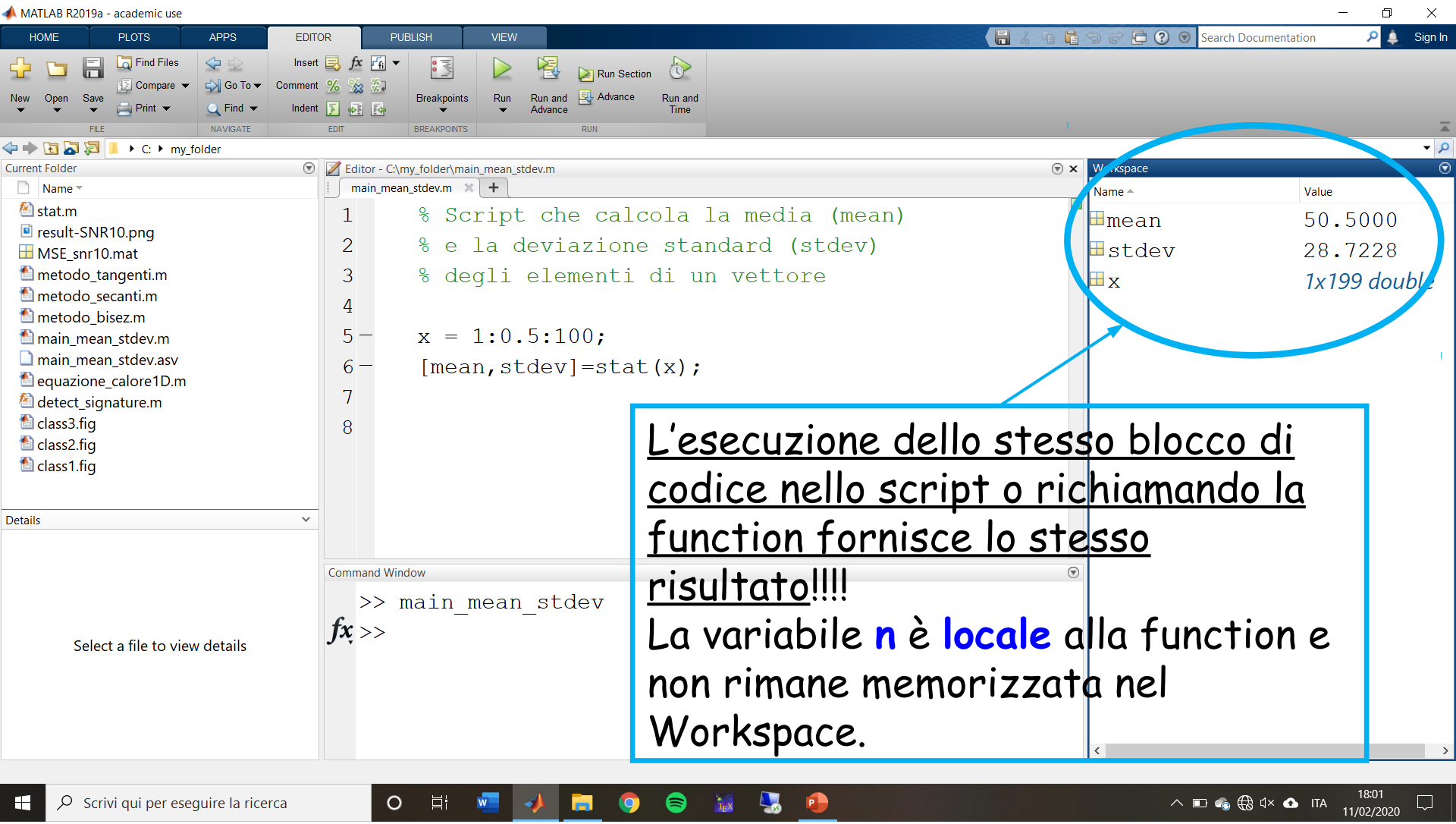
Esempio


MATLAB R2019a - academic use
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
Sign In
New
Open
Save
Find Files
Compare
Print
Go To
Find
Insert
Comment
Indent
Breakpoints
Run
Run and Advance
Run Section
Advance
Run and Time
FILE
NAVIGATE
EDIT
BREAKPOINTS
RUN
C: ▸ my_folder
Current Folder
Name
stat.m
result-SNR10.png
MSE_snr10.mat
metodo_tangenti.m
metodo_secanti.m
metodo_bisez.m
main_mean_stdev.m
equazione_calore1D.m
detect_signature.m
class3.fig
class2.fig
class1.fig
Details
Select a file to view details
Editor - C:\my_folder\stat.m
main_mean_stdev.m
stat.m
1 function [mean,stdev]=stat(x)
2 % Calcola la media (mean)
3 % e la deviazione standard (stdev)
4 % degli elementi del vettore x (dato di input)
5
6 n = length(x);
7 mean = sum(x)/n;
8 stdev = sqrt(sum((x-mean).^2)/n);
9
Command Window
>>
Workspace
Name
Value
stat
Ln 4 Col 18
Scrivi qui per eseguire la ricerca
ITA
17:36
11/02/2020

Esempio


MATLAB R2019a - academic use
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
Sign In
New
Open
Save
Find Files
Compare
Print
Go To
Find
Insert
Comment
Indent
Breakpoints
Run
Run and Advance
Run Section
Advance
Run and Time
FILE
NAVIGATE
EDIT
BREAKPOINTS
RUN
C: ▸ my_folder
Current Folder
Name
stat.m
result-SNR10.png
MSE_snr10.mat
metodo_tangenti.m
metodo_secanti.m
metodo_bisez.m
main_mean_stdev.m
main_mean_stdev.asv
equazione_calore1D.m
detect_signature.m
class3.fig
class2.fig
class1.fig
Details
Select a file to view details
Editor - C:\my_folder\main_mean_stdev.m
main_mean_stdev.m
1 % Script che calcola la media (mean)
2 % e la deviazione standard (stdev)
3 % degli elementi di un vettore
4
5 x = 1:0.5:100;
6 [mean,stdev]=stat(x);
7
8
Command Window
>>
Workspace
Name
Value
La function deve essere richiamata
Scrivi qui per eseguire la ricerca
ITA
17:56
11/02/2020

Esempio


MATLAB R2019a - academic use
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
Sign In
New
Open
Save
Find Files
Compare
Print
Go To
Find
Insert
Comment
Indent
Breakpoints
Run
Run and Advance
Run Section
Advance
Run and Time
FILE
NAVIGATE
EDIT
BREAKPOINTS
RUN
C: ▸ my_folder
Current Folder
Name
stat.m
result-SNR10.png
MSE_snr10.mat
metodo_tangenti.m
metodo_secanti.m
metodo_bisez.m
main_mean_stdev.m
main_mean_stdev.asv
equazione_calore1D.m
detect_signature.m
class3.fig
class2.fig
class1.fig
Details
Select a file to view details
Editor - C:\my_folder\main_mean_stdev.m
main_mean_stdev.m
1 % Script che calcola la media (mean)
2 % e la deviazione standard (stdev)
3 % degli elementi di un vettore
4
5 x = 1:0.5:100;
6 [mean,stdev]=stat(x);
7
8
Command Window
>> main_mean_stdev
>>
Workspace
Name
Value
mean
50.5000
stdev
28.7228
x
1x199 double
L'esecuzione dello stesso blocco di codice nello script o richiamando la function fornisce lo stesso risultato!!!!
La variabile n è locale alla function e non rimane memorizzata nel Workspace.
Scrivi qui per eseguire la ricerca
ITA
18:01
11/02/2020

Function files

- La prima riga del file definisce la sintassi della funzione
-

function [output]=myfunc(input)

output = elenco delle variabili richieste in output separate da virgole

Se non si richiede alcuna variabile in output si scrive solo []

subito dopo il comando function

input = elenco delle variabili di input separate da virgole

myfunc = nome della funzione

Nota: Il nome con cui viene richiamata la funzione dal Command Window è quello del file in cui è memorizzata

Si consiglia **chiamare il file .m** in cui si memorizza la funzione **con il nome della funzione** myfunc

Nota: Nelle righe successive è consigliabile scrivere l'help della funzione.

Dopo lo help si scrive la sequenza di istruzioni

Function files

-Le variabili di input e di output possono essere variabili semplici, vettori, matrici o altre strutture dati

-Il nome della funzione deve sempre iniziare con una lettera dello alfabeto, può contenere numeri e underscore ma non può contenere spazi

-La funzione termina con l'ultima istruzione (nelle versioni più recenti di Matlab si consiglia l'uso del comando end) oppure al primo eventuale comando **return**

-Dopo la prima riga di comando, si possono scrivere dei commenti alla funzione

Per esempio, si può descrivere cosa fa la funzione, le variabili di input richieste e quelle di output restituite. Tali commenti vanno preceduti ad ogni riga dal simbolo **%** e vengono visualizzati come help della funzione

Esempio

```
-------------------------------------------------------------------------
    --
function [mean,stdev]=stat(x)
% Calcola la media (mean) e la deviazione standard (stdev) degli
    elementi
% del vettore x (dato di input)

n = length(x);
mean = sum(x)/n;
stdev = sqrt(sum((x-mean).^2)/n);
-------------------------------------------------------------------------
    --
```

Esempio

Nome della funzione (coincide al nome dato al file .m)

```
--------------------------------------------------------------------------
    --
function [mean,stdev]=stat(x)
% Calcola la media (mean) e la deviazione standard (stdev) degli
    elementi
% del vettore x (dato di input)

n = length(x);
mean = sum(x)/n;
stdev = sqrt(sum((x-mean).^2)/n);
--------------------------------------------------------------------------
    --
```

Esempio


```
--
function [mean,stdev]=stat(x)
% Calcola la media (mean) e la deviazione standard (stdev) degli
    elementi
% del vettore x (dato di input)

n = length(x);
mean = sum(x)/n;
stdev = sqrt(sum((x-mean).^2)/n);
--
```

Esempio

```
-------------------------------------------------------------------
    --
function [mean,stdev]=stat(x)
% Calcola la media (mean) e la deviazione standard (stdev) degli
    elementi
% del vettore x (dato di input)

n = length(x);
mean = sum(x)/n;
stdev = sqrt(sum((x-mean).^2)/n);
-------------------------------------------------------------------
    --
```

Help della funzione

Sequenza di istruzioni

Esempio

```
-------------------------------------------------------------------------
    --
function [mean,stdev]=stat(x)
% Calcola la media (mean) e la deviazione standard (stdev) degli
    elementi
% del vettore x (dato di input)

n = length(x);
mean = sum(x)/n;
stdev = sqrt(sum((x-mean).^2)/n);
-------------------------------------------------------------------------
    --
```

Sequenza di istruzioni

Definizione delle variabili da restituire in output

Esempio

--

Scritta e salvata nel file stat.m, si può chiamare dal Command Window nel modo seguente

```
>>x=rand(50,1);
>>[mean,stdev]=stat(x);
```

Prima di chiamare la funzione è necessario definire la variabile da dare in input

La funzione stat è chiamata sulla variabile x e assegna gli output alle variabili mean e stdev che sono create solo quando è invocata la funzione stat

Le istruzioni precedenti sono equivalenti a

```
>>y=rand(50,1);
>>[my,sdy]=stat(y);
```

E' importante l'ordine e il tipo delle variabili di input e output e non il nome (che può essere diverso da quello usato nel file .m)

Function files

-Una funzione può essere chiamata dal Command window oppure all'interno di uno script o di un'altra funzione

Oss: Per usare una funzione all'interno di uno script, la funzione deve essere nella stessa directory dello script o deve essere una funzione predefinita (contenuta nel path)

- Le variabili definite all'interno della funzione sono locali e si perdono al termine dell'esecuzione della funzione, tranne le variabili richieste in output

- Per esempio la variabile n della funzione stat non sarà contenuta nella lista delle variabili dello workspace perché non è uno degli output

```
function [mean,stdev]=stat(x)
%  Calcola la media (mean) e la deviazione standard (stdev)
degli
 % elementi del vettore x (dato di input)
n= length(x);
mean = sum(x)/n;
stdev = sqrt(sum((x-mean).^2)/n);
```

Strutture dati

E' possibile definire vettori a più dimensioni

N=2 A(i,j) (matrici)

N=3 A(i,j,k)

In generale A(i,j,……,k)

K indici

Esempio: Un' immagine a colori RGB si memorizza nella variabile I tale che

I(:,:,1) matrice componente del rosso
I(:,:,2) matrice componente del verde
I(:,:,3) matrice componente del blu

La dimensione di I è mxnx3, dove mxn è la dimensione della immagine
Per conoscere la dimensione di I si usa la funzione size

Strutture dati

Esempio:

```
>> M = [-2,1,0 ; 0,3,2 ; 1,0,0];
>> disp(M)
    -2     1     0
     0     3     2
     1     0     0

>> M1 = [1 0.2 -1 ; 2 5 -3.4]
M1 =
    1.0000    0.2000   -1.0000
    2.0000    5.0000   -3.4000

>> M2 = [1 0.2 -1 , 2 5 -3.4]
M2 =
    1.0000  0.2000  -1.0000  2.0000  5.0000  -3.4000
```

Strutture dati

Esempio:

```
>> A = ones(6);
>> disp(A)
     1     1     1     1     1     1
     1     1     1     1     1     1
     1     1     1     1     1     1
     1     1     1     1     1     1
     1     1     1     1     1     1
     1     1     1     1     1     1
>> A2 = rand(2,3);
>> disp(A2)
    0.7943    0.5285    0.6020
    0.3112    0.1656    0.2630

>> A3 = 2 + (5-2)*rand(4);
>> disp(A3)
    3.4199    3.6492    4.2612    2.1619
    3.0550    4.7516    3.1413    3.5924
    4.4925    2.8575    3.7035    4.3375
    3.7558    4.2716    2.2276    4.8020
```

Strutture dati

Esempio:

```
>> B=randn(4,2);
>> disp(B)
   -0.2248   -1.1201
   -0.5890    2.5260
   -0.2938    1.6555
   -0.8479    0.3075

>> disp(B+1)
    0.7752   -0.1201
    0.4110    3.5260
    0.7062    2.6555
    0.1521    1.3075

>> disp(B.^2)
    0.0505    1.2547
    0.3470    6.3807
    0.0863    2.7407
    0.7190    0.0946
```

Strutture dati

Esempio:

```
>> A = ones(6);
>> B = randn(6);
>> C = 10*ones(6);
>> I(:,:,1) = A; I(:,:,2) = B; I(:,:,3) = C;
>> size(I)
ans =
     6     6     3
>> I(:,:,3)
ans =
    10    10    10    10    10    10
    10    10    10    10    10    10
    10    10    10    10    10    10
    10    10    10    10    10    10
    10    10    10    10    10    10
    10    10    10    10    10    10
>> I(3,2,1)    % estrae l'elemento di posizione (3,2) nella prima
               %  componente di I
ans =
     1
```

Alcune funzioni utili

fliplr(v): inverte l'ordine degli indici di un vettore riga se v è una matrice, inverte l'ordine degli indici di colonna

flipud(v): inverte l'ordine degli indici di un vettore colonna se v è una matrice, inverte l'ordine degli indici di riga

Esempio:

```
>> v = [3 1 -4 0];
>> a = fliplr(v)
a =
     0    -4     1     3
>> b = flipud(v')
b =
     3
     1
    -4
     0
```

Alcune funzioni utili

Esempio:

```
>> A = [3 1 -4; 10 0 3; -1 2 0]
A =
     3     1    -4
    10     0     3
    -1     2     0
>> a = fliplr(A)
a =
    -4     1     3
     3     0    10
     0     2    -1
>> b = flipud(A)
b =
    -1     2     0
    10     0     3
     3     1    -4
```

Alcune funzioni utili

diff(v) calcola la differenza tra componenti successive di un vettore [v(2)-v(1), v(3)-v(2), …., v(end)-v(end-1)]. La dimensione del vettore risultante è length(v)-1

Se v è una matrice, la matrice risultante contiene le differenze successive tra le righe di v. La dimensione della matrice di output è (m-1)xn se la dimensione di v è mxn.

diff(v,k) calcola le differenze di ordine k tra le componenti di v.

Esempio:

```
>> v = [9 1 7 4 3 0 9 -1];
>>diff(v)
ans =
    -8     6    -3    -1    -3     9   -10
>>diff(v,2)
ans =
    14    -9     2    -2    12   -19
>>diff(v,3)
ans =
   -23    11    -4    14   -31
```

Alcune funzioni utili

Esempio:

```
>> A = [9 1 7; 4 3 0 ;9 -1 5]
A =
     9     1     7
     4     3     0
     9    -1     5

>> diff(A)
ans =
    -5     2    -7
     5    -4     5

>> diff(A,2)
ans =
    10    -6    12
```

Alcune funzioni utili

indici = find(condizione su v)
restituisce un vettore costituito dagli indici corrispondenti alle componenti di v che verificano la condizione.

Esempio: ind = find(v == 0) Restituisce le posizioni delle componenti nulle di v

Se si omette la condizione,cioè ind = find(v), ind contiene le posizioni delle componenti positive di v

Esempio:

```
>> V = [6  1  0.3  -5  10  -4  8  3];
>> ind = find(V <= 0);
>> ind
ind =
     4     6
>> V(ind)
ans =
    -5    -4
```

Strutture dati

Cell permette di collezionare in un'unica variabile oggetti di vario tipo (vettori, matrici, variabili numeriche o logiche, caratteri o stringhe)
Ma anche variabili dello stesso tipo ma di dimensione diversa.
Variabili di tipo cell si definiscono tra parentesi graffe. Le componenti si elencano una dopo l'altra e separate da virgole
C = {componente1, componente2, …. , componenteN};
C{i} estrae la i-esima componente di C
C(i) indica la i-esima componente di C

Esempio:

```
>> C = {'ciao', [4 3 1 2],3,[1 7 2; 0 5 8; 1 0 9]};
```

% definisce una variabile cell composta da 4 elementi: una stringa, un vettore, un numero e una matrice.

```
>> whos
Name        Size                Bytes  Class
  C         1x4                   488  cell array
```

```
>>C{2}  % estrae il secondo elemento della variabile di tipo cell C
ans =
     4     3     1     2
```

Strutture dati

Nota: una variabile di tipo Cell contiene copie di variabili e non puntatori a variabili. Quindi se C ={A,B}, con A, B due generiche variabili, il contenuto di C non cambia se in seguito A e B sono modificate

Esempio:

```
>> A = [2 5 3 6];
>> B = [1 5; 8 9;3 6];
>> C = {A,B};
>> B = 1;
>> C{2}
ans =
     1     5
     8     9
     3     6

Nota: C(2) = {1}   (per modificarla)
```

Strutture dati

C può essere inizializzata con il comando C = cell(m,n) : C è composta da mxn matrici vuote

Celldisp(C): visualizza il contenuto di una cell

Cellplot(C): disegna il contenuto di una cell

Le celle possono essere annidate, cioè un elemento di una cell può essere esso stesso una cell.

Esempio: C = cell(1,4); A = [1 2 3; 4 5 6]; v = [1 4 6];
C{1} = {A,v};
C{2} = 4;
C{4} = 'stringa';

Per estrarre il contenuto della matrice A si digita il comando
C{1}{1}
gli indici sono ordinati da sinistra verso destra, dalla cell più esterna a quella più interna

Strutture dati

Una variabile di tipo Struct è simile ad una di tipo cell in quanto permette di raggruppare variabili di vario tipo. Una variabile di tipo struct si costruisce definendo dei campi

S.field1 = variabile1
S.field2 = variabile2
…
S.fieldN = variabileN

Gli elementi di una variabile struct si estraggono specificandone il campo.

Esempio:

```
>> S.stringa = 'ciao';
>> S.vettore = [4 3 1 2];
>> S.numero = 3;
>> S.matrice = [1 7 2; 0 5 8; 1 0 9];
>> S.vettore    % estrae l'elemento nel campo vettore
ans =
     4     3     1     2
```

Strutture dati

Esempio:

```
>> S
S =
     stringa: 'ciao'
     vettore: [4 3 1 2]
      numero: 3
     matrice: [3x3 double]
```

- Il comando whos fornisce le informazioni associate ad una variabile con una struttura di tipo struct i cui campi sono
  Name dimension bytes class

-il comando dir da informazioni su una variabile o directory. I campi sono
  Name date bytes isdir

Strutture dati

isstruct(s): restituisce 1 se s è di tipo struct, o altrimenti

fieldnames(s): estrae i campi associati ad una variabile di tipo struct. L'output è una cell.

Alcune funzioni

Varargin = variabile di tipo cell contenente le variabili di input di una funzione. Si usa quando gli input di una funzione possono variare (esistono parametri di default)

Oss: l'uso di questa funzione richiede un ordine preciso delle variabili di input che devono essere assegnate all'inizio della funzione combinata con il comando nargin.

Nargin = numero degli elementi in varargin

OSS: per l'output si usa varargout combinata con nargout.

Esempio: function [R] = confronta(x,varargin)

```
if nargin = 0
   T = 256;
else
   T = varargin{1};
end

ind = find(x>T);
if isempty(ind),  R = 0, else R=1; end
```

Alcune funzioni

hist(x,N) : istogramma di x
calcola le occorrenze dei simboli in x rispetto ad un numero prefissato N di bin (determinano per esempio degli intervalli)

Esempio: x = [120 34 67 90 1 67 90 2 4 245 2 67];

hist(x,256) % disegna usando bar

h = hist(x,256) % conserva in h le occorrenze

Che differenza c'è con la funzione histc?

Operazioni su matrici

reshape(M,m2,n2) riorganizza il contenuto della matrice M di dimensioni m1xn1 in una matrice di dimensione m2xn2 (segue l'ordinamento lessico-grafico degli elementi della matrice)

Esempio:

```
>A = [1 2 3;0 4 6;3 2 1;9 7 2];
>B = reshape(A,2,6);
>disp(A)
     1     2     3
     0     4     6
     3     2     1
     9     7     2
>disp(B)
     1     3     2     2     3     1
     0     9     4     7     6     2
```

Operazioni su matrici

repmat(M,m2,n2) replica la matrice M m2xn2 volte. M diventa un blocco di una matrice di dimensioni superiori

Esempio:

```
>A = [1 2 3;0 4 6;3 2 1;9 7 2];
>C = repmat(A,2,3);
>disp(C)
```

1	2	3	1	2	3	1	2	3
0	4	6	0	4	6	0	4	6
3	2	1	3	2	1	3	2	1
9	7	2	9	7	2	9	7	2
1	2	3	1	2	3	1	2	3
0	4	6	0	4	6	0	4	6
3	2	1	3	2	1	3	2	1
9	7	2	9	7	2	9	7	2

Save, Load

save nome_file.mat variabili

memorizza l'insieme di variabili nel file nome_file.mat

Le variabili memorizzate possono essere richiamate in seguito con il comando

load nome_file.mat

OSS: le variabili sono memorizzate nel workspace con lo stesso nome con il quale sono state memorizzate!

Programmazione MATLAB

Alcune strutture di programmazione elementari

- Operatori relazionali: `<`, `<=`, `>`, `>=`, `==`, `=`
- Operatori logici: `&` (*and*), `/` (*or*), `~` (*not*)
- Cicli controllati da un contatore: `for - end`
- Cicli condizionati: `while - end`
- Strutture condizionali: `if - elseif - else - end`
- Uscita incondizionata: `break`

Nota: tutte le strutture possono essere scritte su più righe oppure su un'unica riga separate da virgole.

Ciclo for

```
for indice = m:p:n
   blocco di istruzioni
end
```

Ripete il blocco di istruzioni un numero fissato di volte

indice contatore

m valore iniziale del contatore

p incremento del contatore (positivo o negativo)

n valore finale del contatore

indice,m, p, n possono essere variabili intere o reali

Nota: indice = m:n equivale a indice=m:1:n

Esempi

File esempio.m

```
cont = 1;
for  k=0:0.1:1
    cont = cont + 1;
end
```

```
>> esempio
>> disp(cont)
      12
```

Esercizio: sia x = 1 + (i-2)*0.5 , i = 1,….,14 e y = e^x.
Calcolare il prodotto scalare tra i due vettori.

Istruzioni condizionali

```
if condizione
   blocco istruzioni 1
else
   blocco istruzioni 2
end
```

Esegue il blocco di istruzioni 1 se la condizione è vera, altrimenti esegue il blocco di istruzioni 2

Esempio

```
>>if x>=0, disp(x), else, disp('numero negativo'), end
```

Nota: else si può omettere.

Istruzioni condizionali

```
if condizione1
   blocco istruzioni1
elseif condizione2
   blocco istruzioni2
else
   blocco istruzioni3
End
```

Esegue il blocco istruzioni1 se la condizione 1, altrimenti esegue il blocco 2 se la condizione 2 è vera, altrimenti esegue il blocco istruzioni 3

Es: stampare i numeri dispari tra 1 e 10

Ciclo while

while condizione

    blocco di istruzioni

end

Esegue il blocco di istruzioni se è la condizione è vera

Esempio: visualizzare i primi 4 numeri naturali

```
>>x=1;
>>while x~=5, disp(x), x=x+1; end
     1
     2
     3
     4
```

Ciclo while

Esempio:

```
a =10;
b = 5;
while a
        a = a-1;
        while b
                b = b-1;
        end
end
```

Esegue il ciclo fino a quando a e b sono entrambe 0

Nota: In un contesto logico una variabile numerica restituisce 1 (vero) se la variabile è diversa da zero, altrimenti 0 (falso).

Istruzione break

break: arresta l'esecuzione dei cicli for o while

(non può essere utilizzata all'esterno di un ciclo)

```
X = rand(10,1);
Y=[];
for i = 1:10
    if X(i) >= 0.7
       break,
    else Y(i)=1/(0.7-X(i));
    end
end
```

Switch

Switch-case-Otherwise:

```
switch  nome_variabile
case valore1
1° blocco di istruzioni
case valore2
2° blocco di istruzioni
…….
otherwise
ultimo blocco di istruzioni
                    end
```

Se il valore di nome_variabile è valore1 viene eseguito il 1° blocco di istruzioni. Se è valore2, il 2° blocco, e così via, altrimenti, se non è alcuno dei valori elencati, esegue l'ultimo blocco di istruzioni.

Switch

Esempio:

```
a = rem(b,3);        % resto della divisione per 3
switch  a
  case 0
     c = b/3;
  case 1
     c = (b-a)/3;
  otherwise
     c = (b-a)/3;
end
```

Cenni sul Calcolo Simbolico

Matlab esegue non solo operazioni numeriche ma anche manipolazioni e soluzioni di espressioni matematiche (in forma analitica) simboliche .

Usando il Symbolic Math Toolbox è possibile risolvere/calcolare:

equazioni algebriche e trascendenti, integrali, derivate, limiti, serie, ODE etc.

Cenni sul Calcolo Simbolico

L'uso del Symbolic Math Toolbox permette di usare funzioni simboliche che hanno lo stesso nome delle funzioni numeriche:

```
>> help sym/nomefunzione
```

Cenni sul Calcolo Simbolico

```
>> help diff

diff Difference and approximate derivative.
   diff(X), for a vector X, is [X(2)-X(1)  X(3)-X(2) ... X(n)-X(n-1)].
   diff(X), for a matrix X, is the matrix of row differences,
     [X(2:n,:) - X(1:n-1,:)].
   diff(X), for an N-D array X, is the difference along the first
     non-singleton dimension of X.
   diff(X,N) is the N-th order difference along the first non-singleton
     dimension (denote it by DIM). If N >= size(X,DIM), diff takes
     successive differences along the next non-singleton dimension.
   diff(X,N,DIM) is the Nth difference function along dimension DIM.
     If N >= size(X,DIM), diff returns an empty array.

   Examples:
. . .
```

Cenni sul Calcolo Simbolico

```
>> help sym/diff

diff   Differentiate.
   diff(S) differentiates a symbolic expression S with
respect to its
   free variable as determined by SYMVAR.
   diff(S,'v') or diff(S,sym('v')) differentiates S with
respect to v.
   diff(S,n), for a positive integer n, differentiates S n
times.
   diff(S,'v',n) and diff(S,n,'v') are also acceptable.
   diff(S,'v1','v2',...) or diff(S,sym('v1'),sym('v2'),...)
differentiates
   S with respect to v1, v2, ...

   Examples:
. . .
```

Cenni sul Calcolo Simbolico

Il Symbolic Math Toolbox permette di definire un **oggetto simbolico** (struttura dati che memorizza una rappresentazione stringa del simbolo).

Per creare oggetti simbolici in Matlab si utilizza la funzione sym.

Per esempio:

```
>> x = sym('x')

x =

x
```

Nota: in alternativa si usi syms

Cenni sul Calcolo Simbolico

Esempio:

```
>> syms x y real
>> f = log(x^2 + y^2)

f =
log(x^2+y^2)

>> g = sqrt(y/x)

g =
(y/x)^(1/2)

>> h=f+g

f =
 log(x^2 + y^2) + (y/x)^(1/2)
```

Cenni sul Calcolo Simbolico

collect(E)	raccoglie i coefficienti con la stessa potenza di x
expand(E)	applica regole algebriche per espandere l'espressione E
factor(E)	esprime E come prodotto di polinomi con coefficienti razionali
poly2sym(p)	converte i coefficienti del vettore p in un polinomio simbolico
sym2poly(E)	converte l'espressione E nel vettore di coefficienti
pretty(E)	visualizza l'espressione E in forma matematica
simple(E)	ricerca la forma dell'espressione E più corta in termini di numero di caratteri,utilizzando differenti semplificazioni algebriche
simplify(E)	semplifica l'espressione E
subs(E,old,new)	sostituisce *new* al posto di *old* nell'espressione E

Cenni sul Calcolo Simbolico

Esempio:

```
>>  syms x y

>>  expand((x+1)^3)

        x^3+3*x^2+3*x+1

>> syms x
>> f = 3*x + abs(x-1)

f =

3*x + abs(x - 1)

>> pretty(f)
3 x + |x - 1|
```

Cenni sul Calcolo Simbolico

Esempio: ***funzione solve***

Risolve equazioni algebriche trascendenti:

```
>> solve('x+8=0')
ans =
-8

>> eq='exp(2*x)=54';

>> solve(eq)
ans =

log(54)/2
```

Cenni sul Calcolo Simbolico

diff(E)	Restituisce la derivata dell'espressione E rispetto alla variabile indipendente di default (x)
int(E)	Restituisce l'integrale dell'espressione E
limit(E)	Restituisce il valore del limite di E per x che tende a 0 (default)
symsum(E)	Restituisce la somma dell'espressione E rispetto alla sua variabile k da 0 a k-1
taylor(f,x,a,'Order',n)	Restituisce il polinomio di Maclaurin di f di ordine n-1, valutato nel punto x=a

Cenni sul Calcolo Simbolico

Esempi:

```
>> syms x
>> diff(sin(x)+exp(x))

ans =

cos(x) + exp(x)

>> diff(sin(x)*exp(x))

ans =

exp(x)*cos(x) + exp(x)*sin(x)
```

Cenni sul Calcolo Simbolico

Esempi:

```
>> syms x y
>> diff(sin(x)*exp(x*y),y)

ans =

x*exp(x*y)*sin(x)

>> diff(sin(x)*exp(x),2)

ans =

2*exp(x)*cos(x)
```

Cenni sul Calcolo Simbolico

Esempi:

```
>> syms x
>> limit(sin(x)/x)
ans =
 1
```

```
>> sym x;
>> g = exp(x);
>> taylor( g , x , 0 ,'Order',7 )
ans =

x^6/720 + x^5/120 + x^4/24 + x^3/6 + x^2/2 + x + 1
```

Esercizi

1) Scrivere una funzione matlab che riceva in input i parametri ***n***, ***a***, ***b***, generi un vettore ***x*** di lunghezza ***n*** costituito da numeri compresi tra *a* e *b* e restituisca in output i vettori ***y*** e ***j***. ***y*** contiene gli elementi positivi ***x*** mentre ***j*** contiene gli indici delle componenti x(i)>0.

Esercizi

Usando lo help di Matlab, stabilire l'utilità delle seguenti funzioni:

error
nargchk
ndims
numel
isreal, isnumeric, islogical
isa
bin2dec, dec2bin
num2str
fopen, fclose
fread, fwrite

Esercizi

2) Scrivere uno script Matlab che:

a)assegni due vettori interi **w** e **v** di dimensione ***n***.

b)se le componenti w_i e v_i $(i=1\ldots,n)$ sono entrambe pari, calcoli la loro media e la memorizzi nelle componenti di un nuovo vettore **P** di dimensione ***m x n***;

c)stampi su schermo il vettore **P**.

3) Scrivere una funzione Matlab che riceva in input un vettore reale **V** e restituisca in output il massimo elemento v_{max} di **V** e l'indice corrispondente.

4) Scrivere una funzione Matlab che riceva in input un vettore reale **S**, ordini le componenti di **S** in ordine crescente calcolando anche il numero di scambi **s** necessari e restituisca in output il vettore riordinato e il numero di scambi effettuati.

Esercizi

5) Scrivere uno script Matlab che:

- legga in input una matrice reale ***A*** di dimensione ***h x l*** e una matrice reale **B** di dimensione ***l x m*** ;
- calcoli la matrice ***C=AB*** di dimensione ***h x m*** con il prodotto righe-colonne sia con i cicli for che con le istruzioni di Matlab;
- stampi ***C*** per colonne.

6) Scrivere uno script Matlab che, assegnata una matrice reale quadrata **T** di ordine **n**:

a)calcoli l'elemento massimo t_{max} della prima colonna di **T**;

b)se t_{max} è diverso dall'elemento t_{11}, lo porti, con uno scambio di righe, al posto di t_{11};

c)stampi la nuova matrice.

Esercizi

7) Le radici dell'equazione di secondo grado $ax^2+bx+c=0$ sono date da

$x_1= -(b+(\Delta)^{1/2}) / (2a),\ \ x_2= -(b- +(\Delta)^{1/2})/ (2a)$

purché $\Delta = b^2-4ac > 0$.

Scrivere uno script Matlab che:

- assegni i valori a,b,c ;
- calcoli Δ;
- se $\Delta < 0$ stampi il messaggio *'La soluzione non esiste'*;
  se $\Delta = 0$ stampi il messaggio *'Le radici sono coincidenti"*
  e il valore della radice; se $\Delta > 0$ calcoli e stampi x_1 , x_2.