

# Laboratorio di Elaborazione di Immagini

## Esercitazione 1:

### INTRODUZIONE A MATLAB

2017-2018



# Lezioni

**Lunedì dalle 15.30 alle 18.30**

- Niente pausa e finiamo alle 18:00
- 8 lezioni da 3 ore ciascuna

# MATLAB

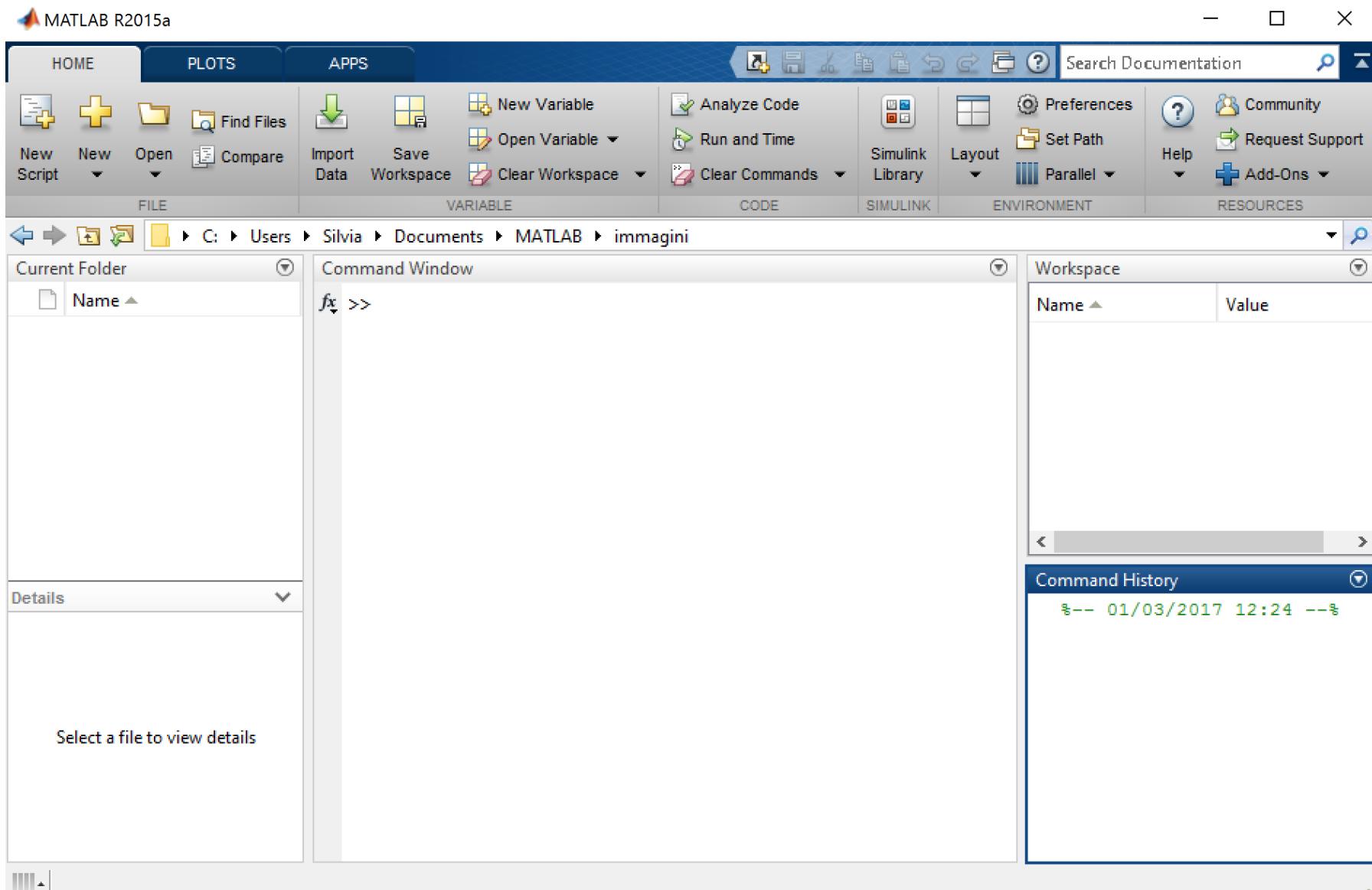
## Cos'è MATLAB?

- MATLAB **non** è un linguaggio di programmazione
- MATLAB è un ambiente per il calcolo numerico, al cui interno contiene un linguaggio di programmazione proprietario
- MATLAB è disponibile sia per sistemi Unix (Mac Os e Linux) sia per Windows

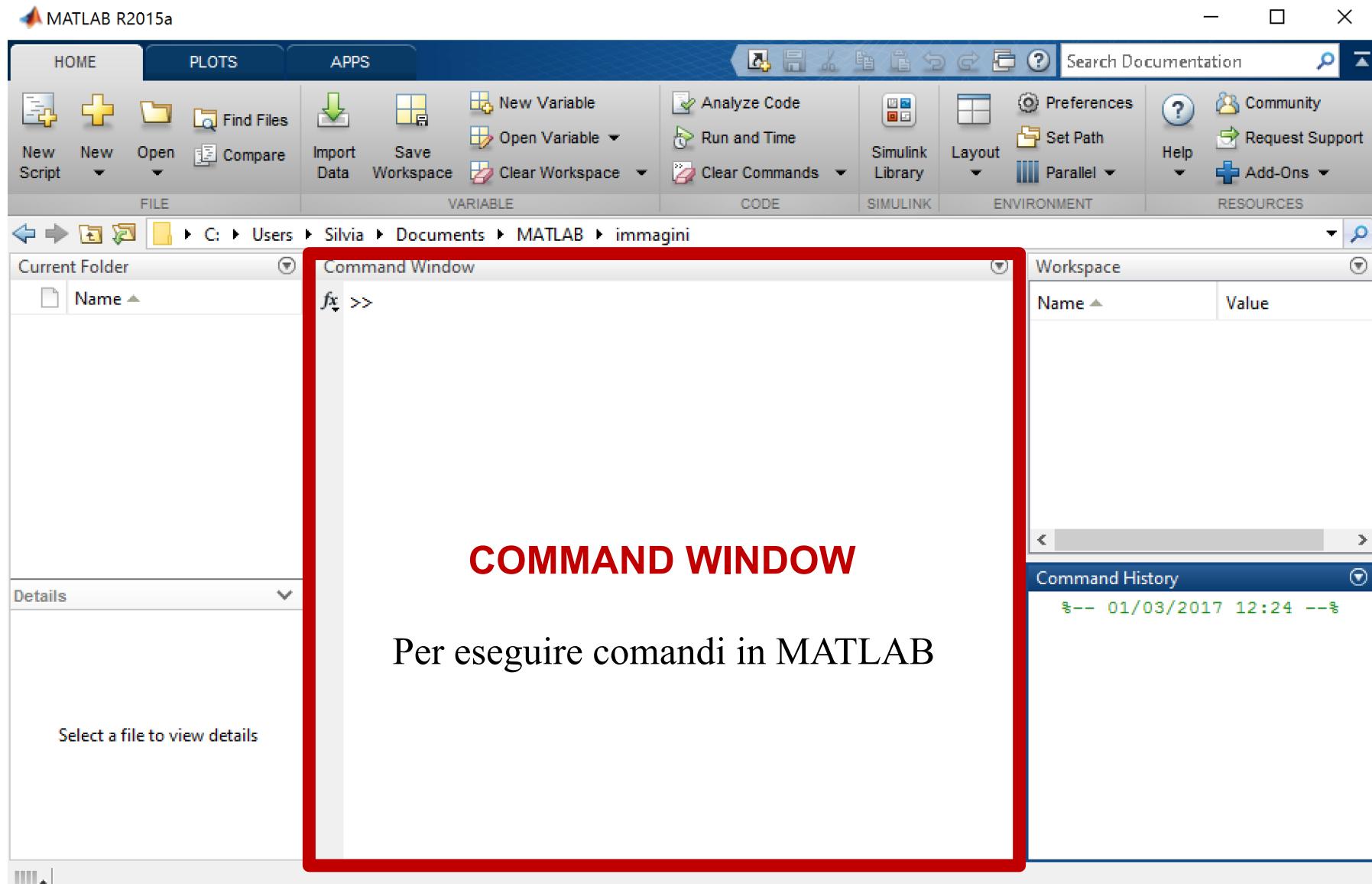
# MATLAB

- **MATLAB** è la contrazione delle parole ***MAT*rix *LA*boratory**
- Dal nome, MATLAB è stato progettato per il calcolo matriciale, con efficienza computazionale per lo svolgimento di calcoli

# Interfaccia MATLAB



# Interfaccia MATLAB

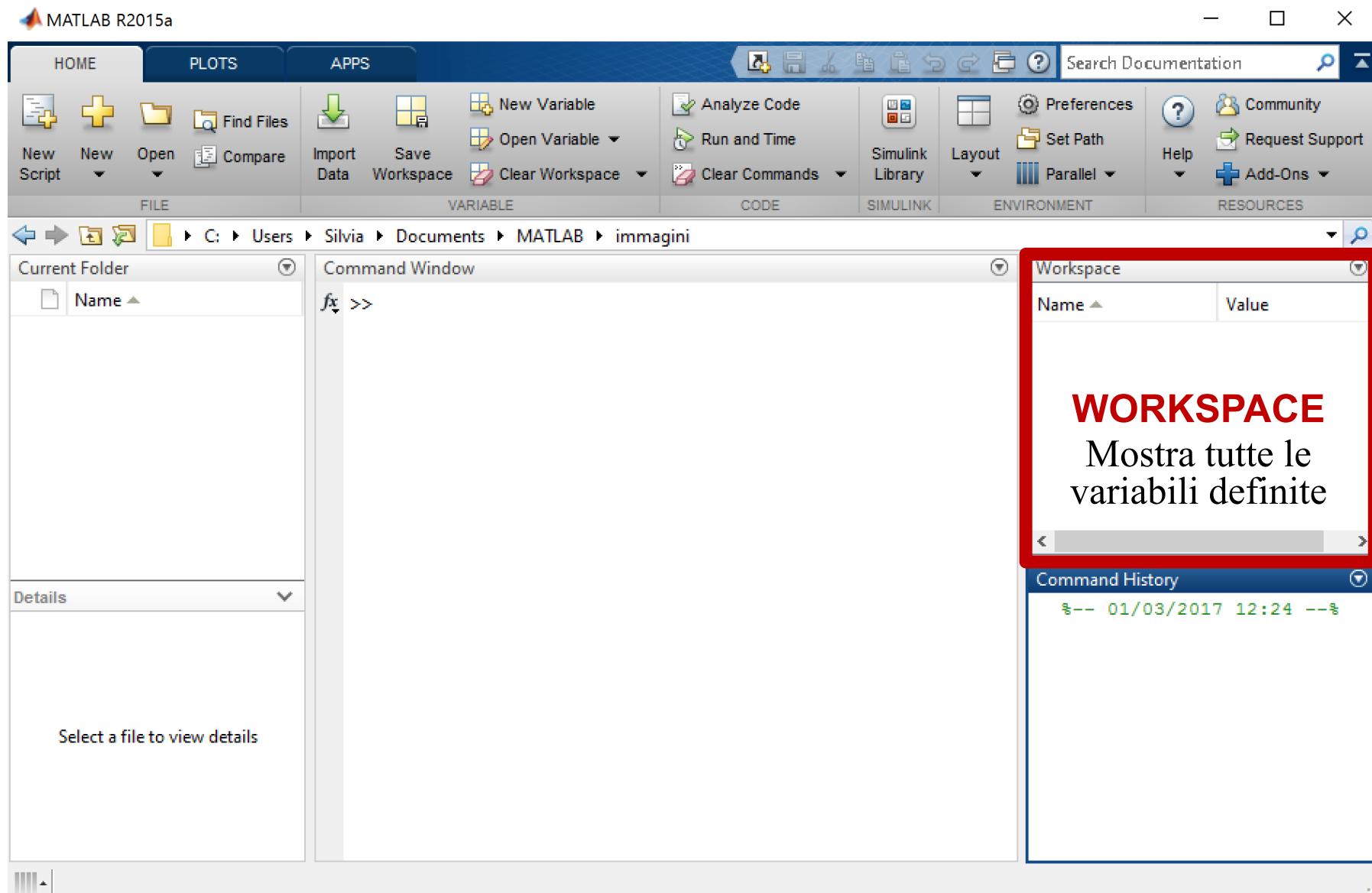


## COMMAND WINDOW

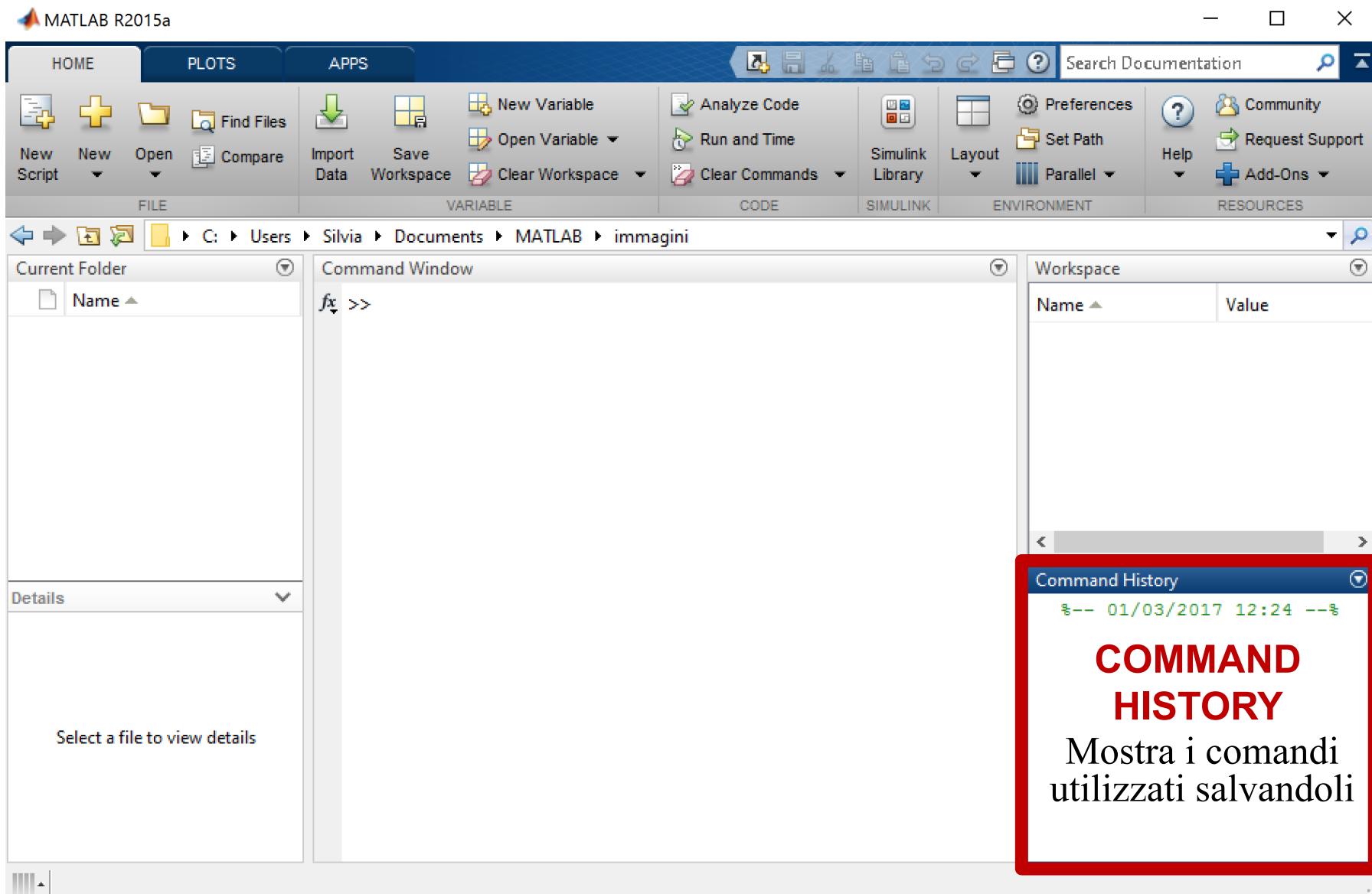
Per eseguire comandi in MATLAB



# Interfaccia MATLAB

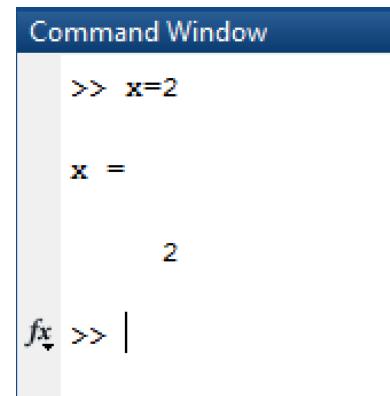


# Interfaccia MATLAB



# Variabili in MATLAB

- La dichiarazione della variabile non necessita la descrizione del ‘tipo’:



The screenshot shows the MATLAB Command Window with the title bar "Command Window". Inside, the command `>> x=2` is entered. The variable `x` is assigned the value `2`. The window has a dark blue header and a light gray body. The cursor is positioned at the end of the command line.

<nome variabile>=<valore>

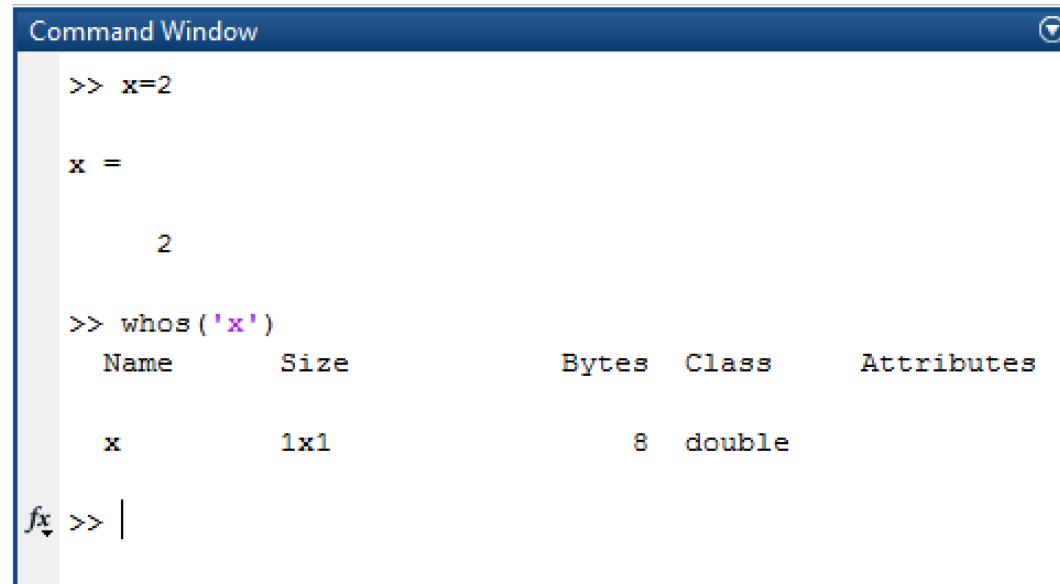
# Variabili in MATLAB

- Regole per nominare le variabili:
  - Deve essere univoco nei primi 63 caratteri
  - Deve iniziare con una lettera
  - Non deve contenere spazi bianchi o simboli di punteggiatura
  - Può contenere qualsiasi combinazione di lettere, cifre e underscore
  - Sono CASE-SENSITIVE
  - Consiglio: non usare parole chiave di Matlab
- Esistono alcune variabili predefinite:
  - pi

# Variabili in MATLAB

- MATLAB è tipizzato implicitamente

**WARNING:** a volte è più un problema che un vantaggio



The screenshot shows the MATLAB Command Window with the following content:

```
Command Window
>> x=2
x =
2
>> whos('x')
  Name      Size            Bytes  Class       Attributes
  x            1x1                 8  double
```

# Help in MATLAB

- Conoscete un comando, ma non vi ricordate il suo scopo:

`help <function>`

```
Command Window

>> help whos
whos - List variables in workspace, with sizes and types

This MATLAB function displays in alphabetical order all variables in the
currently active workspace, with information about their sizes and types.

whos
whos(variables)
whos(location)
whos(variables,location)
S = whos(__)

Reference page for whos

See also clear, exist, what, who

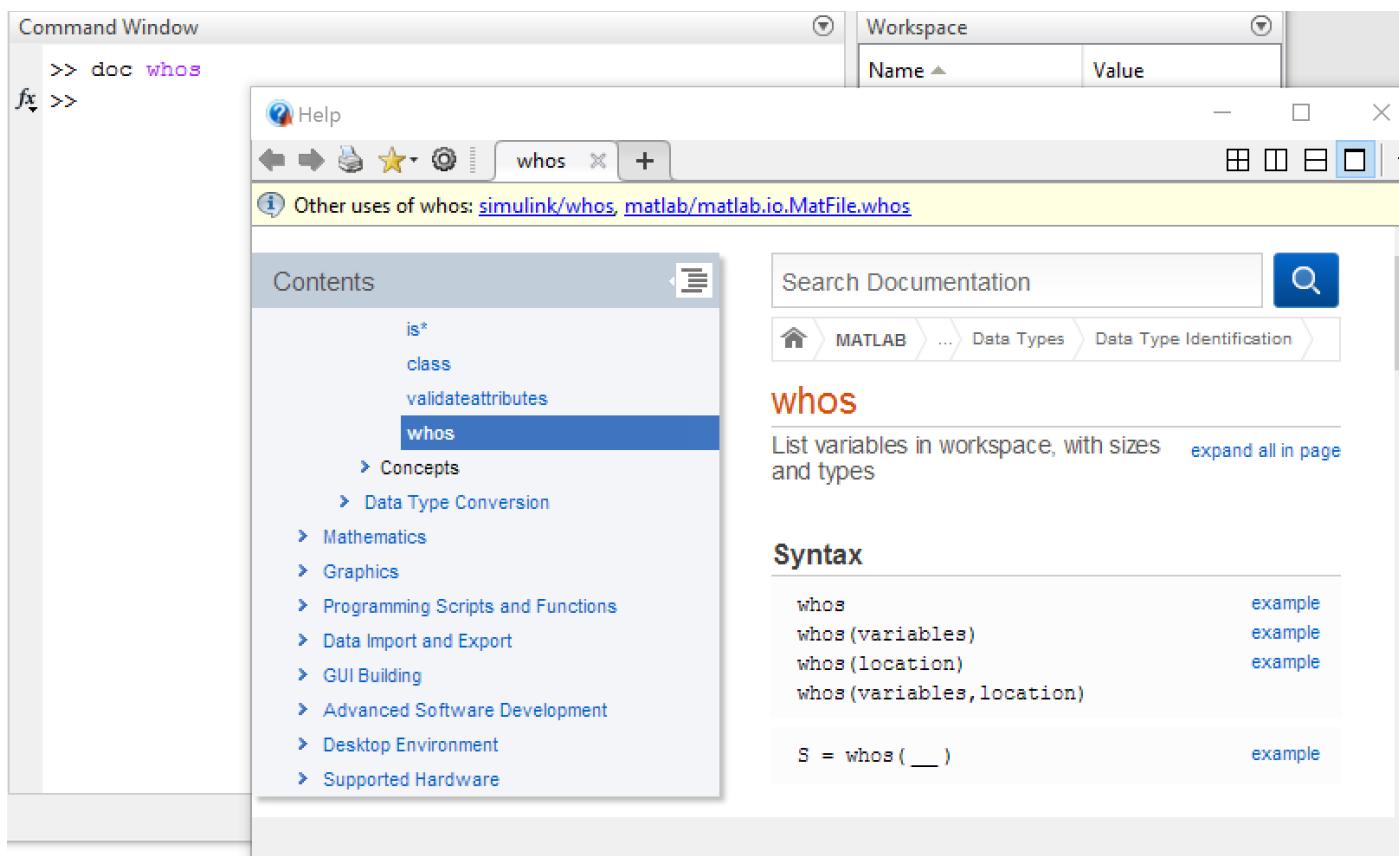
Other uses of whos
simulink/whos

fx >> |
```

# Help in MATLAB

- Conoscete un comando, ma non vi ricordate il suo scopo:

doc <function>

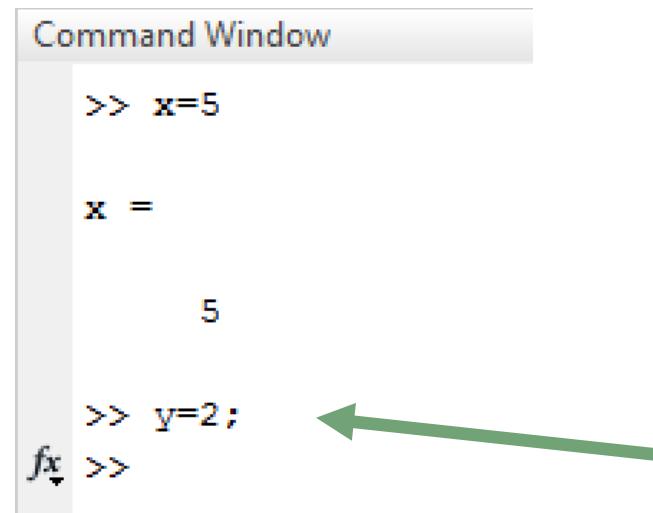


# Workspace in MATLAB

- Who, whos → mostra i dettagli delle variabili nel workspace
- Save → salva il workspace in un file \*.mat
- Load → carica le variabili da un file \*.mat
- Clear → svuota il workspace cancellando tutte le variabili

# MATLAB: soppressione output

- Il punto e virgola alla fine del comando sopprime l'output



The screenshot shows the MATLAB Command Window with the following content:

```
Command Window
>> x=5
x =
5
>> y=2;
fx >>
```

A green arrow points from the text "Il punto e virgola alla fine del comando sopprime l'output" to the semicolon at the end of the command `y=2;`.

# MATLAB: comandi base

```
Command Window
>> x=17;
>> y=4;
>>
>> x+y

ans =
21

>> x-y

ans =
13

>> x*y

ans =
68

>> x/y

ans =
4.2500

fx >>
```

# MATLAB: operatori logici

- ==
- <
- >
- ~ (not)
- ~= (not equal)

# MATLAB: comandi base

- Costrutto 'IF ELSE'

```
Command Window
>> if x == 5
y = 1
else
y = 0
end

y =
0

fx >>
```

# MATLAB: comandi base

- Costrutto 'IF        ELSEIF        ELSE'

Command Window

```
>> if x == 5
y = 1
elseif x == 17
y = 2
else
y = 0
end

y =
2

fx >>
```

# MATLAB: comandi base

- Costrutto ‘FOR’

Command Window

```
>> x = 0;
>> for i = 1:3
x = x + 1
end

x =
1

x =
2

x =
3

fx >>
```

# MATLAB: comandi base

- Costrutto ‘SWITCH’

```
Command Window
>> x = 3;
>> switch x
case 1
y = 'primo'
case 2
y = 'secondo'
case 3
y = 'terzo'
otherwise
y = 'non classificato'
end

y =
terzo

fx >>
```

# MATLAB: comandi base

- Costrutto ‘WHILE’

Command Window

```
>> x=6;
>> n=0;
>> while (x~=2)
    x=x-1;
    n=n+1;
end
>> iterazione=n

iterazione =
4
```

fx

# MATLAB: comandi base

- Comando 'BREAK'

Command Window

```
>> n=5;
for i=0:20
    if(mod(n, 7)==0)
        iterazione=i
        break;
    else
        n=n+3;
    end
end

iterazione =
```

3

fx

n =  
8  
n =  
11  
n =|  
14

# MATLAB: array

- MATLAB prevede diversi modi per dichiarare un array

Command Window

```
>> x = 1:5

x =
1 2 3 4 5

fx >>
```

Command Window

```
>> x = [1, 2, 3, 4, 5]

x =
1 2 3 4 5

fx >>
```

Command Window

```
>> x = [1;2;3;4;5]

x =
1
2
3
4
5

fx >>
```

# MATLAB: array

- In MATLAB gli array partono da 1

```
Command Window
>> x(1)

ans =
1

>>
>> x(5)

ans =
5

>> x(1) = 0

x =
0
2
3
4
5

fx >>
```

Per accedere si usano  
le parentesi tonde

# MATLAB: array

- L'operatore ':' permette di accedere a più valori

Command Window

```
>> x(2:4)

ans =
    2
    3
    4
```

fx >>

Un intervallo compreso  
tra i due valori

Command Window

```
>> x(:)

ans =
    0
    2
    3
    4
    5
```

fx >>

L'interezza dell'array  
senza dover sapere da  
quante celle è formato



# MATLAB: array

- L'operatore ':' permette anche di definire degli intervalli

```
Command Window
>> x = 1:6

x =
    1     2     3     4     5     6

>> x = 1:1:6

x =
    1     2     3     4     5     6

>> x = 1:2:6

x =
    1     3     5

fx >>
```

# MATLAB: array

- L'operatore ':' permette anche di definire degli intervalli

```
Command Window
>> x = 1:6
x =
    1     2     3     4     5     6
>> x = 1:1:6
x =
    1     2     3     4     5     6
>> x = 1:2:6
x =
    1     3     5
fx >>
```

Un intervallo compreso  
tra i due valori 1 e 6 con  
un distanza tra un  
valore e l'altro di 2

# MATLAB: array

- Attenzione alle dimensioni degli array

```
Command Window
>> x = [1,2,3]
x =
    1     2     3
>> y = [1;2;3]
y =
    1
    2
    3
>> x+y
Error using +
Matrix dimensions must agree.
fx >>
```

# MATLAB: array

- L'apice identifica l'operazione TRASPOSTA

```
Command Window
>> y = [1;2;3]

y =
1
2
3

>> y'
ans =
1     2     3
          ↑
          ←

>> x+y'
ans =
2     4     6

fx >>
```

# MATLAB: array

- L'operatore \* di default indica il prodotto vettoriale!

Command Window

```
>> x * y
ans =
22

>> (x(1)*y(1))+(x(2)*y(2))+(x(3)*y(3))
ans =
22

>> y * x
ans =
1     3     5
2     6    10
3     9    15
```

fx >>

1x3 \* 3x1 = 1x1

3x1 \* 1x3 = 3x3

# MATLAB: array

- Usate `.*` per il prodotto tra elementi

```
Command Window
>> x .* y'

ans =
    1     6    15

>> [x(1)*y(1), x(2)*y(2), x(3)*y(3)]

ans =
    1     6    15

fx >>
```

# MATLAB: matrici

Command Window

```
>> a = [1:2:6;2:2:6]

a =

    1     3     5
    2     4     6

>> b = ones(2,3)

b =

    1     1     1
    1     1     1

>> c = zeros(2,5)

c =

    0     0     0     0     0
    0     0     0     0     0

>> R = rand(2,3)

R =

    0.8147    0.1270    0.6324
    0.9058    0.9134    0.0975

fx >>
```

# MATLAB: matrici

- le matrici funzionano esattamente come gli array
- Di fatto gli array sono matrici 1XN (o NX1)

```
>> X = rand(2,2)  
  
X =  
  
    0.8147    0.1270  
    0.9058    0.9134  
  
>> X(1,2)  
  
ans =  
  
    0.1270  
  
>> X(2,1)  
  
ans =  
  
    0.9058  
  
% >> |
```

```
>> X = [1,2; 3,4]  
  
X =  
  
    1    2  
    3    4  
  
>> X'  
  
ans =  
  
    1    3  
    2    4  
  
>> X + X  
  
ans =  
  
    2    4  
    6    8
```

# MATLAB: matrici

- L'operatore  $*$  indica il prodotto tra matrici

```
>> X*X  
  
ans =  
  
    7    10  
   15    22  
  
>> X.*X  
  
ans =  
  
    1    4  
    9   16  
  
fx >> |
```

# MATLAB: matrici

- Comando ‘length’ ritorna la lunghezza di un vettore o il numero di righe di una matrice
- Comando ‘size’ ritorna il numero di righe e colonne

```
>> x = [1,2,3,4];
>> length(x)

ans =
    4

>> size(x)

ans =
    1     4
```

```
>> x = rand(3,2)

x =
    0.9575    0.9706
    0.9649    0.9572
    0.1576    0.4854

>> length(x)

ans =
    3

>> size(x)

ans =
    3     2
```

# MATLAB: comandi utili

```
>> % somma degli elementi di una matrice
>> X = [1,2; 3,4]

X =

    1     2
    3     4

>> somma = 0

somma =

    0

>> [x,y] = size(X);
>> for i=1:x
for j=1:y
somma = somma + X(i,j);
end
end
>> somma

somma =

    10

>> |
```

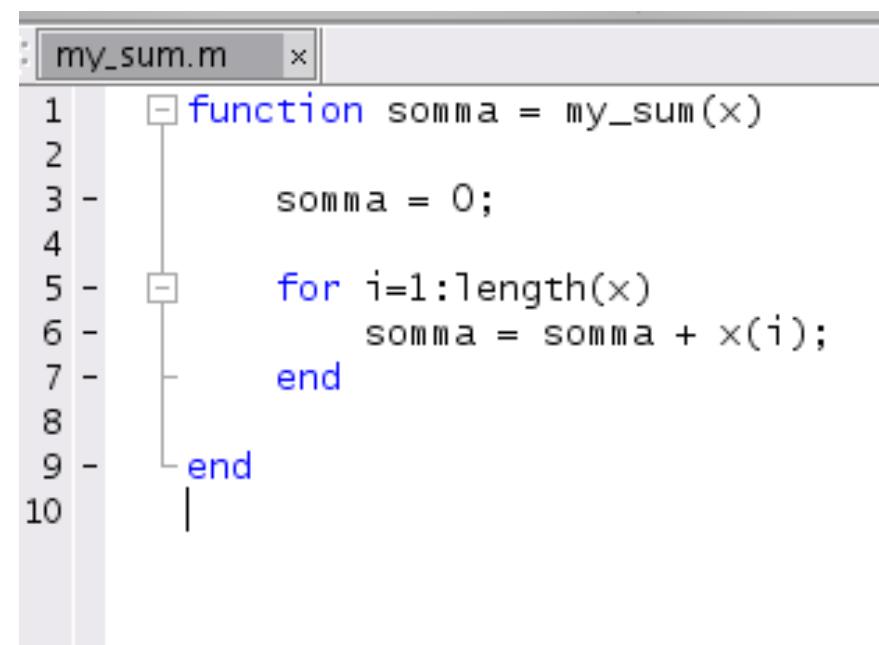
# MATLAB: comandi utili

- MATLAB ha già implementato praticamente tutte le funzioni matematiche

```
>> sum(X(:))  
  
ans =  
  
10  
  
>> mean(X(:))  
  
ans =  
  
2.5000  
  
>> median(X(:))  
  
ans =  
  
2.5000  
  
fx >> |
```

# MATLAB: funzioni

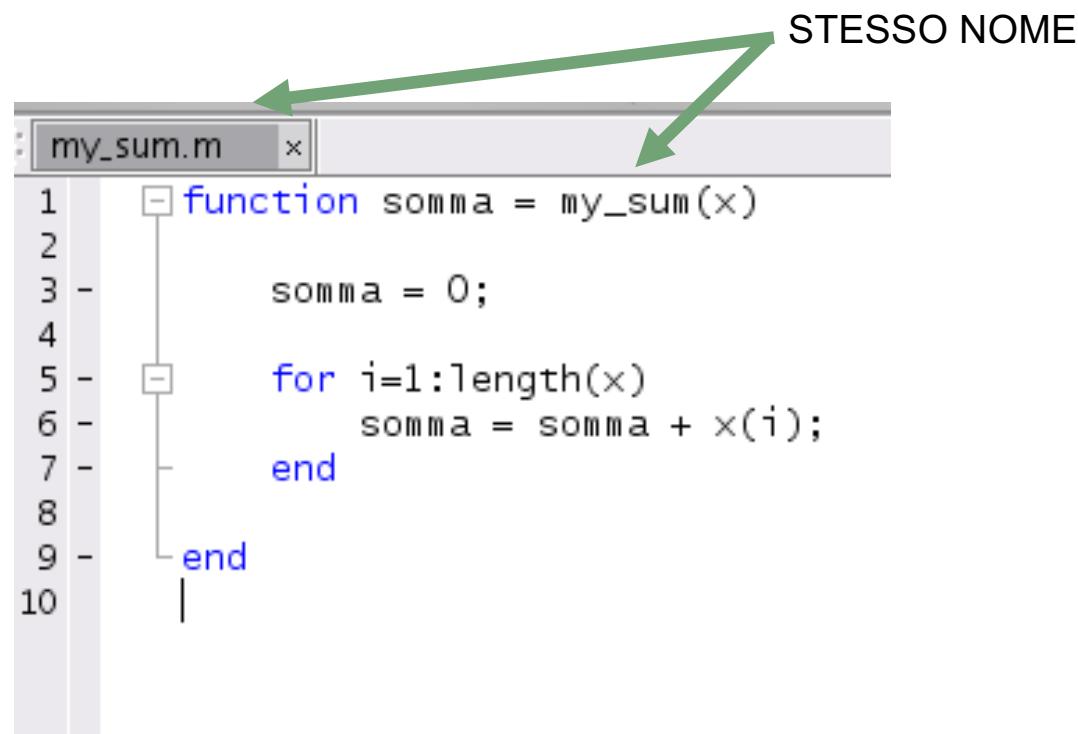
- In MATLAB possiamo definire le nostre funzioni in file con l'estensione .m



```
my_sum.m
1 function somma = my_sum(x)
2
3 -     somma = 0;
4
5 -     for i=1:length(x)
6 -         somma = somma + x(i);
7 -     end
8
9 - end
```

# MATLAB: funzioni

- In MATLAB possiamo definire le nostre funzioni in file con l'estensione .m



STESO NOME

```
1 function somma = my_sum(x)
2
3 -     somma = 0;
4
5 -     for i=1:length(x)
6 -         somma = somma + x(i);
7 -     end
8
9 - end
```

# MATLAB: funzioni

- In MATLAB possiamo definire le nostre funzioni in file con l'estensione .m

```
my_sum.m
1 function somma = my_sum(x)
2
3 -     somma = 0;
4
5 -     for i=1:length(x)
6 -         somma = somma + x(i);
7 -     end
8
9 - end
```

RETURN  
IMPLICITO

# MATLAB: funzioni

- MATLAB è super-ottimizzato per le operazioni matriciali, non usiamo i cicli se non strettamente necessario!

```
>> x = ones(10000,1);
>> tic; x_somma = sum(x); toc
Elapsed time is 0.000041 seconds.
>> x_somma

x_somma =
10000
```

fx >> |

```
>> x = ones(10000,1);
>> tic; x_somma = my_sum(x); toc
Elapsed time is 0.002982 seconds.
>> x_somma

x_somma =
10000
```

fx >> |

# MATLAB: funzioni

- MATLAB è super-ottimizzato per le operazioni matriciali, non usiamo i cicli se non strettamente necessario!

```
>> x = ones(10000,1);
>> tic; x_somma = sum(x); toc
Elapsed time is 0.000041 seconds.
>> x_somma

x_somma =
10000
```

fx >> |

```
>> x = ones(10000,1);
>> tic; x_somma = my_sum(x); toc
Elapsed time is 0.002982 seconds.
>> x_somma

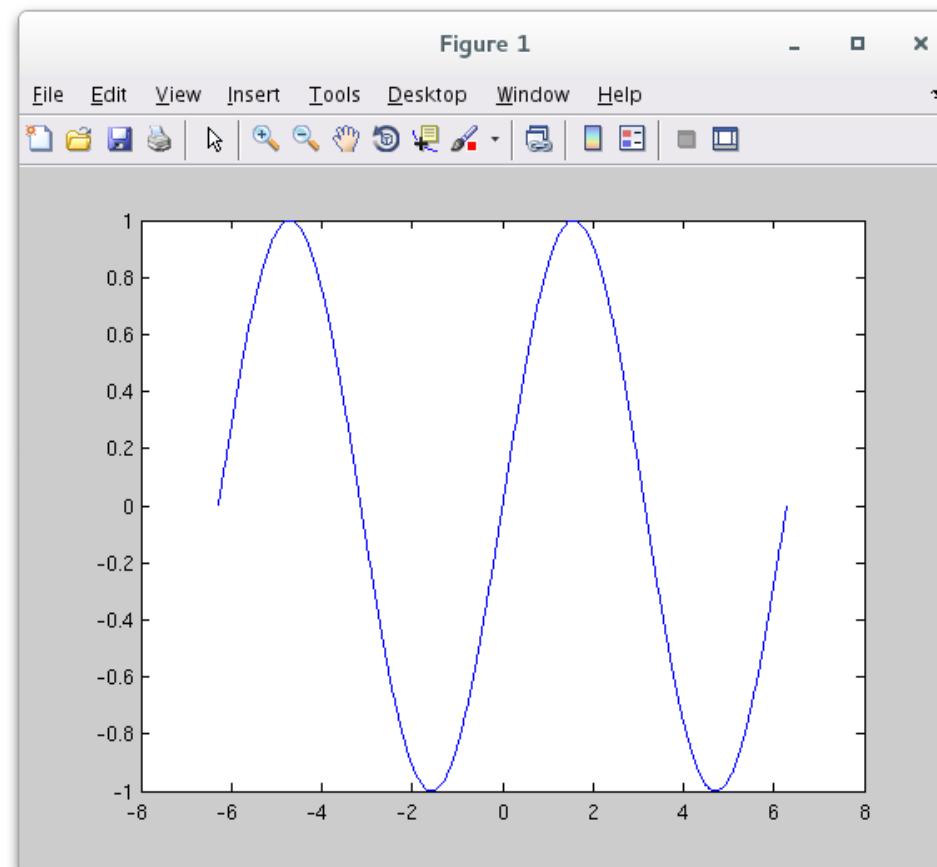
x_somma =
10000
```

fx >> |

- La funzione sum di MATLAB è 72 volte più veloce di my\_sum

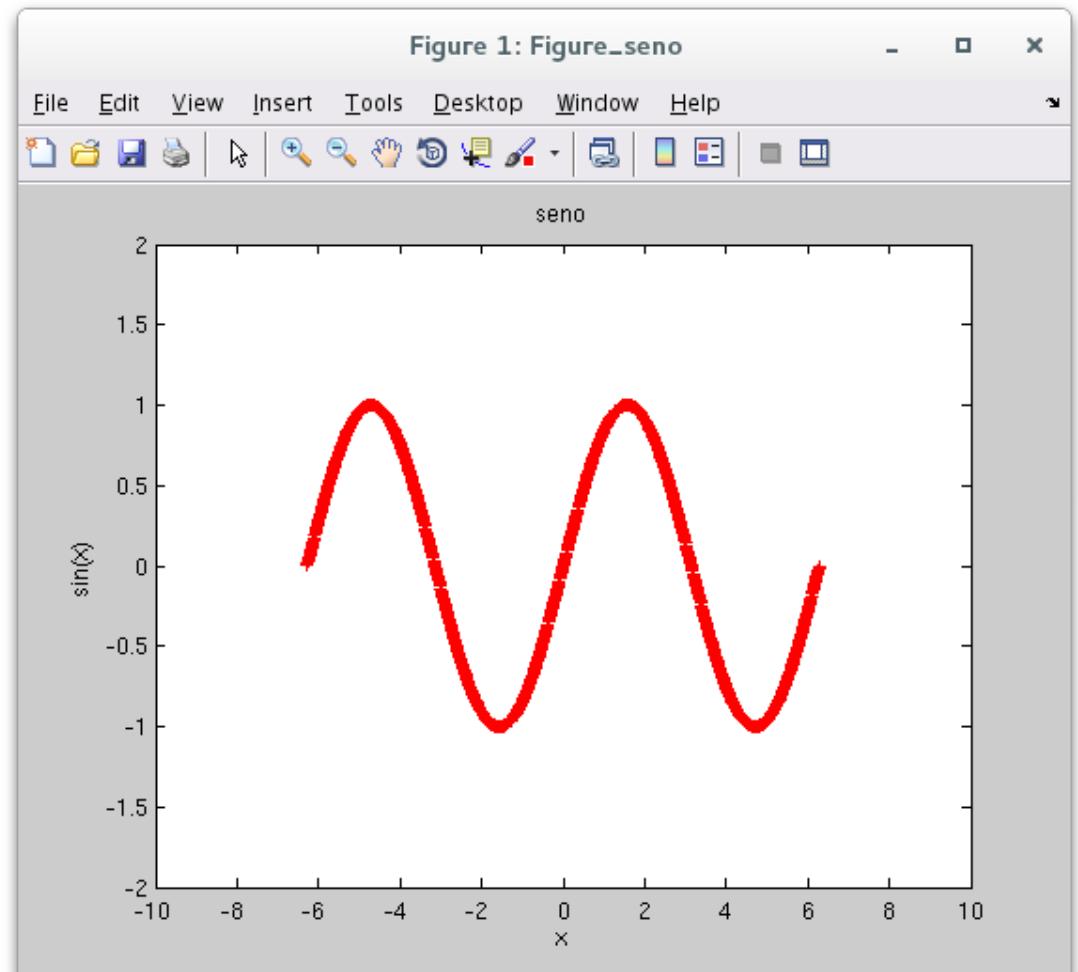
# MATLAB: plot

```
>> x = linspace(-2*pi,2*pi,1000);  
  
>> y = sin(x);  
  
>> plot(x,y)
```



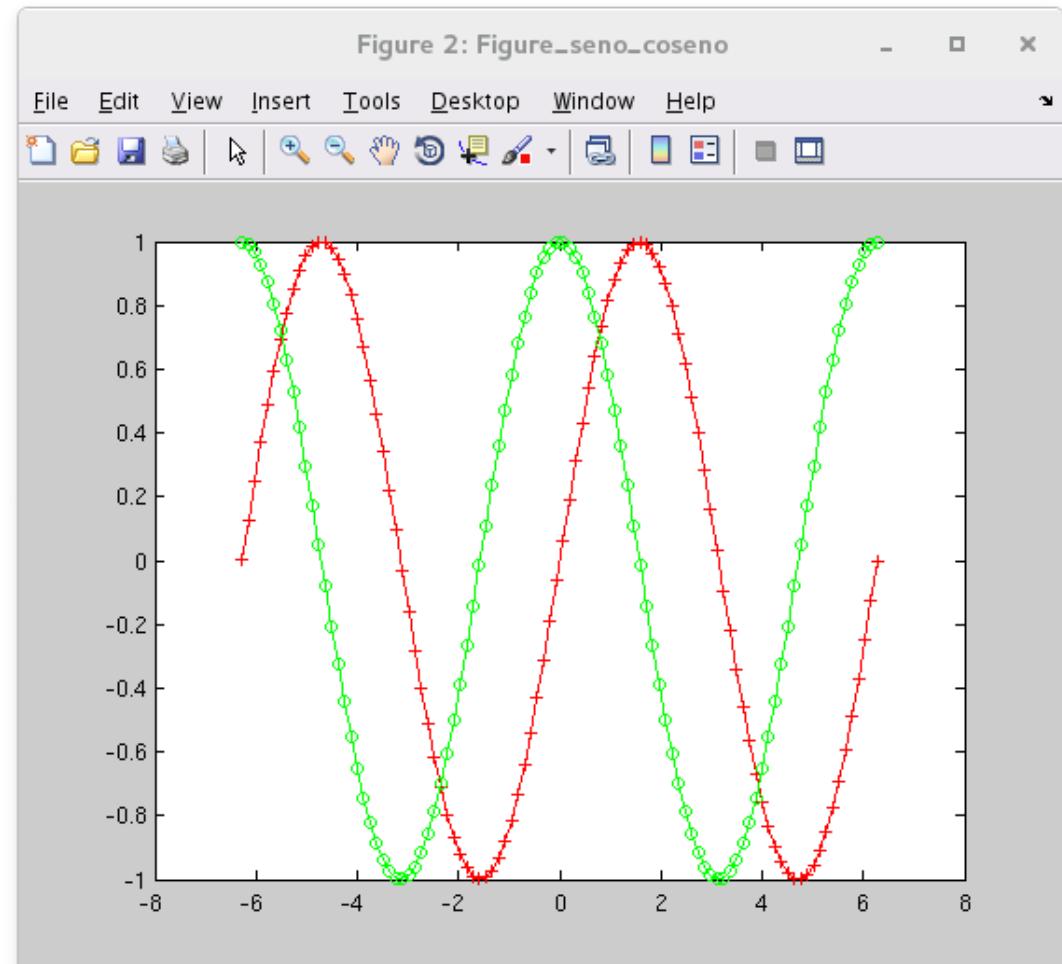
# MATLAB: plot

```
>> x = linspace(-2*pi,2*pi,1000);  
  
>> y = sin(x);  
  
>> figure('Name', 'Figure_seno');  
  
>> plot(x, y, 'r+-');  
  
>> xlabel('x');  
  
>> ylabel('sin(x)');  
  
>> title('seno');  
  
>> axis([-10 10 -2 2]);
```



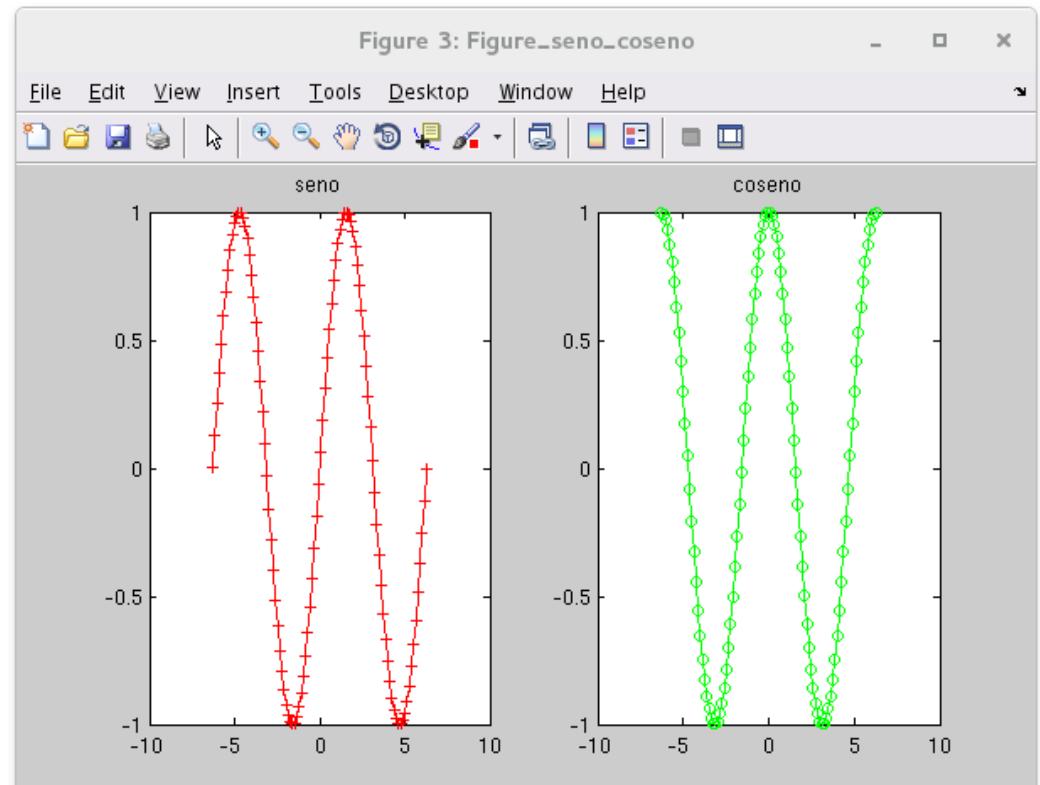
# MATLAB: multiple plot: 'hold on'

```
>> x = linspace(-2*pi,2*pi,1000);  
  
>> y = sin(x);  
  
>> z = cos(x);  
  
>> figure('Name', 'Figure_seno_coseno');  
  
>> plot(x, y, 'r+-');  
  
>> hold on;  
  
>> plot(x,z,'go-')  
  
>> hold off;
```



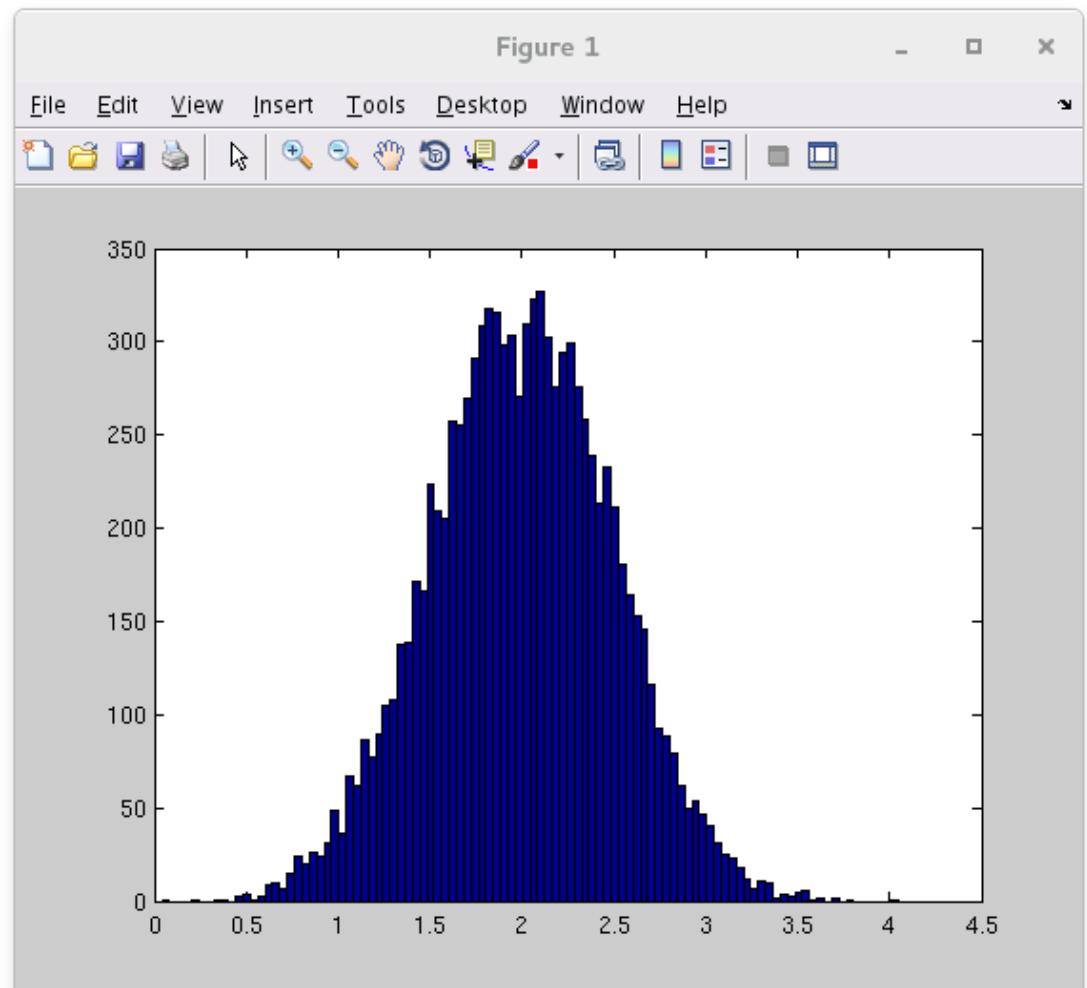
# MATLAB: multiple plots: ‘subplot’

```
>> x = linspace(-2*pi,2*pi,1000);  
  
>> y = sin(x);  
  
>> z = cos(x);  
  
>> figure('Name', 'Figure_seno_coseno');  
  
>> subplot(1, 2, 1)  
  
>> plot(x, y, 'r+-')  
  
>> title('seno')  
  
>> subplot(1, 2, 2)  
  
>> plot(x,z,'go-')  
  
>> title('coseno')
```



# MATLAB: histograms

```
>> x = randn(1,10000);  
  
>> hist(x,100)  
  
>> x = randn(1,10000) * 0.5 + 2;  
  
>> hist(x,100)
```



# MATLAB: histograms

```
>> figure('Name','histograms');

>> subplot(121)

>> x = randn(1,10000);

>> hist(x,100)

>> subplot(122)

>> x = randn(1,10000) * 0.5 + 2;

>> hist(x,100)
```

