

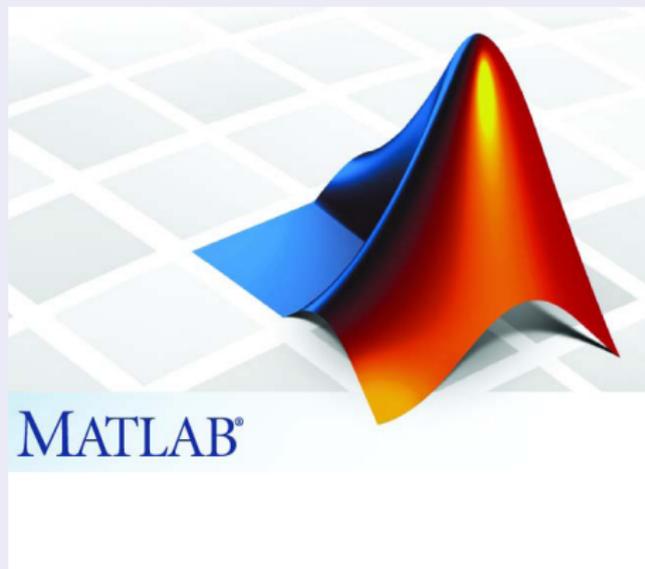
**UNIVERSITÀ DEGLI STUDI DI FIRENZE**  
Facoltà di Scienze Matematiche, Fisiche e Naturali  
Corso di Calcolo Numerico - a.a. 2007-2008

---

## Introduzione a MATLAB

---

*Carlotta Giannelli*



- 1 Introduzione
- 2 Matrici
- 3 Area di Lavoro
- 4 Visualizzazione di variabili
- 5 Espressioni
- 6 Funzioni *in – line*
- 7 Istruzioni per il controllo del flusso di esecuzione
- 8 *m – files*
- 9 Grafica

# Outline

- 1 Introduzione
- 2 Matrici
- 3 Area di Lavoro
- 4 Visualizzazione di variabili
- 5 Espressioni
- 6 Funzioni *in – line*
- 7 Istruzioni per il controllo del flusso di esecuzione
- 8 *m – files*
- 9 Grafica

# MATLAB: MATrix LABoratory

- Ambiente di sviluppo interattivo *matrix-oriented*
- Interfaccia alle librerie di algebra lineare
- Elevate potenzialità grafiche in 2D e 3D
- *Toolboxes*
  - ▶ Control System
  - ▶ Filter Design
  - ▶ Financial
  - ▶ Fuzzy Logic
  - ▶ Image Processing
  - ▶ Neural Network
  - ▶ Optimization
  - ▶ Partial Differential Equations
  - ▶ Signal Processing
  - ▶ Spline
  - ▶ Statistics
  - ▶ Symbolic Math
  - ▶ Wavelet
  - ▶ e altri ancora ...
- *Simulink*



# MATLAB Help

- Help in linea

- ▶ `help`
- ▶ `help arg`
- ▶ `help help`
- ▶ `lookfor arg`

- Help Browser

- ▶ `helpwin`
- ▶ `helpwin arg`
- ▶ `doc`
- ▶ `doc arg`

- Demos

- `www.mathworks.com`

# MATLAB Help

## ● Help in linea

- ▶ help
- ▶ help arg
- ▶ help help
- ▶ lookfor arg

## ● Help Browser

- ▶ helpwin
- ▶ helpwin arg
- ▶ doc
- ▶ doc arg

## ● Demos

## ● www.mathworks.com

```

Command Window
>> help
HELP topics

matlab\general      - General purpose commands.
matlab\ops          - Operators and special characters.
matlab\lang         - Programming language constructs.
matlab\elemat       - Elementary matrices and matrix manipulation.
matlab\elemfun      - 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.
matlab\scrIBE       - Annotation and Plot Editing.
matlab\graph2d      - Two dimensional graphs.
matlab\graph3d      - Three dimensional graphs.
matlab\specgraph    - Specialized graphs.
matlab\graphics     - Handle Graphics.
matlab\uitools      - Graphical user interface tools.
matlab\strfun       - Character strings.
matlab\imagesci     - Image and scientific data input/output.
matlab\iofun        - File input and output.
matlab\audiovideo   - Audio and Video support.
matlab\timefun      - Time and dates.
matlab\datatypes    - Data types and structures.
matlab\verctrl      - Version control.
matlab\codetools    - Commands for creating and debugging code.
matlab\helptools    - Help commands.
matlab\winfun       - Windows Operating System Interface Files (COM/DDE)
matlab\demos        - Examples and demonstrations.
toolbox\local       - Preferences.
matlab\matlabx1     - MATLAB Builder for Excel
simulink\simulink   - Simulink
simulink\blocks     - Simulink block library
  
```

# MATLAB Help

- Help in linea

- ▶ help
- ▶ help arg
- ▶ help help
- ▶ lookfor arg

- Help Browser

- ▶ helpwin
- ▶ helpwin arg
- ▶ doc
- ▶ doc arg

- Demos

- [www.mathworks.com](http://www.mathworks.com)

```
Command Window

To get started, select MATLAB Help or Demos from the Help menu.

>> help sin
SIN      Sine.
        SIN(X) is the sine of the elements of X.

        See also asin, sind.

        Overloaded functions or methods (ones with the same name in other directories)
        help\_svm/sin.m

        Reference page in Help browser
        doc\_sin

>>
```

# MATLAB Help

- Help in linea

- ▶ help
- ▶ help arg
- ▶ help help
- ▶ lookfor arg

- Help Browser

- ▶ helpwin
- ▶ helpwin arg
- ▶ doc
- ▶ doc arg

- Demos

- [www.mathworks.com](http://www.mathworks.com)

```

Command Window
>> help help
HELP Display help text in Command Window.
HELP, by itself, lists all primary help topics. Each primary topic
corresponds to a directory name on the MATLABPATH.

HELP / lists a description of all operators and special characters.

HELP FUN displays a description of and syntax for the function FUN.
When FUN is in multiple directories on the MATLAB path, HELP displays
information about the first FUN found on the path and lists
PATHNAME/FUN for other (overloaded) FUNs.

HELP PATHNAME/FUN displays help for the function FUN in the PATHNAME
directory. Use this syntax to get help for overloaded functions.

HELP DIR displays a brief description of each function in the MATLAB
directory DIR. DIR can be a relative partial pathname (see HELP
PARTIALPATH). When there is also a function called DIR, help for both
the directory and the function are provided.

HELP CLASSNAME.METHODNAME displays help for the method METHODNAME of
the fully qualified class CLASSNAME. To determine CLASSNAME for
METHODNAME, use CLASS(OBJ), where METHODNAME is of the same class as
the object OBJ.

HELP CLASSNAME displays help for the fully qualified class CLASSNAME.

T = HELP('TOPIC') returns the help text for TOPIC as a string, with
each line separated by /n. TOPIC is any allowable argument for HELP.

REMARKS:
1. Use MORE ON before running HELP to pause HELP output after a
screenful of text displays.
2. In the help syntax, function names are capitalized to make them

```

# MATLAB Help

- Help in linea

- ▶ help
- ▶ help arg
- ▶ help help
- ▶ lookfor arg

- Help Browser

- ▶ helpwin
- ▶ helpwin arg
- ▶ doc
- ▶ doc arg

- Demos

- [www.mathworks.com](http://www.mathworks.com)

```
Command Window
>>
>> help lookfor
LOOKFOR Search all M-files for keyword.
LOOKFOR XYZ looks for the string XYZ in the first comment line
(the H1 line) of the HELP text in all M-files found on MATLABPATH.
For all files in which a match occurs, LOOKFOR displays the H1 line.

For example, "lookfor inverse" finds at least a dozen matches,
including the H1 lines containing "inverse hyperbolic cosine"
"two-dimensional inverse FFT", and "pseudoinverse".
Contrast this with "which inverse" or "what inverse", which run
more quickly, but which probably fail to find anything because
MATLAB does not ordinarily have a function "inverse".

LOOKFOR XYZ -all searches the entire first comment block of
each M-file.

In summary, WHAT lists the functions in a given directory,
WHICH finds the directory containing a given function or file, and
LOOKFOR finds all functions in all directories that might have
something to do with a given key word.

See also dir, help, who, what, which.

Reference page in Help browser
doc lookfor

>>
>>
>>
>>
>>
```

# MATLAB Help

## • Help in linea

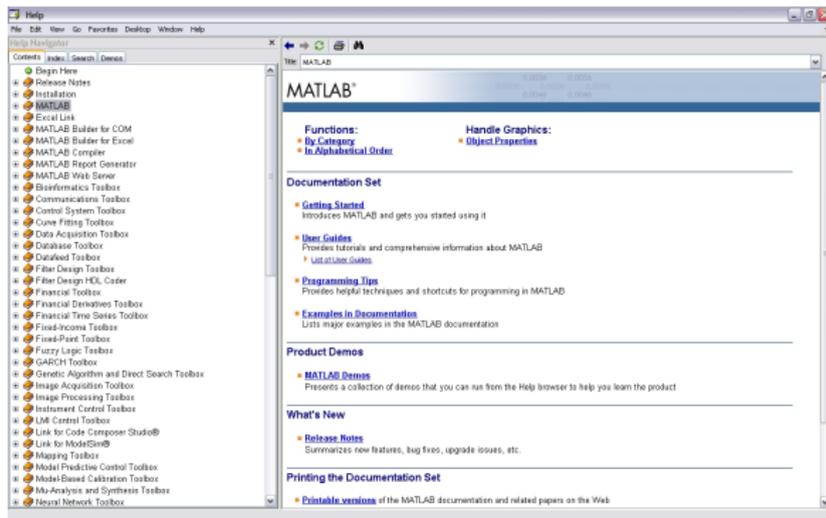
- ▶ help
- ▶ help arg
- ▶ help help
- ▶ lookfor arg

## • Help Browser

- ▶ helpwin
- ▶ helpwin arg
- ▶ doc
- ▶ doc arg

## • Demos

## • www.mathworks.com





# MATLAB Help

- Help in linea

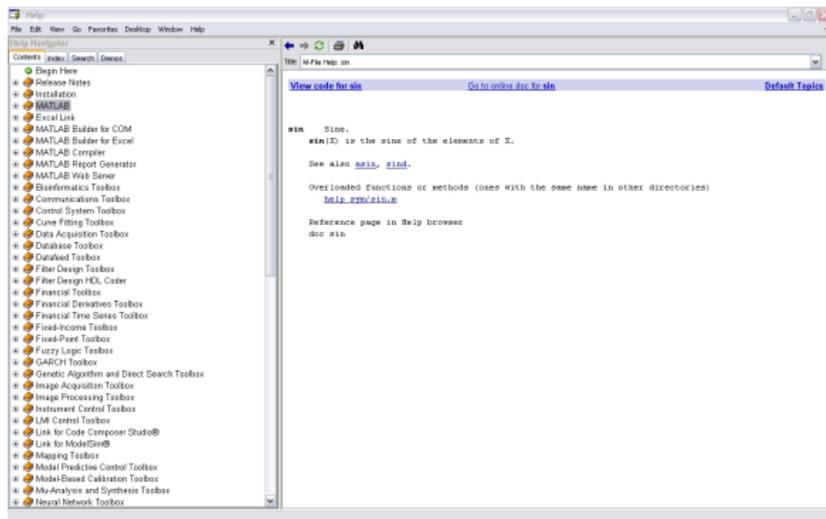
- ▶ help
- ▶ help arg
- ▶ help help
- ▶ lookfor arg

- Help Browser

- ▶ helpwin
- ▶ helpwin arg
- ▶ doc
- ▶ doc arg

- Demos

- [www.mathworks.com](http://www.mathworks.com)



# MATLAB Help

## • Help in linea

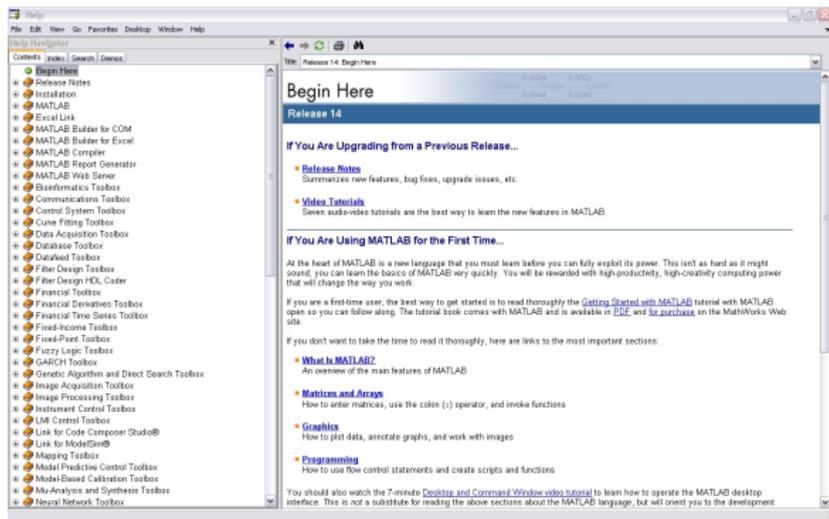
- ▶ help
- ▶ help arg
- ▶ help help
- ▶ lookfor arg

## • Help Browser

- ▶ helpwin
- ▶ helpwin arg
- ▶ doc
- ▶ doc arg

## • Demos

• [www.mathworks.com](http://www.mathworks.com)



# MATLAB Help

- Help in linea

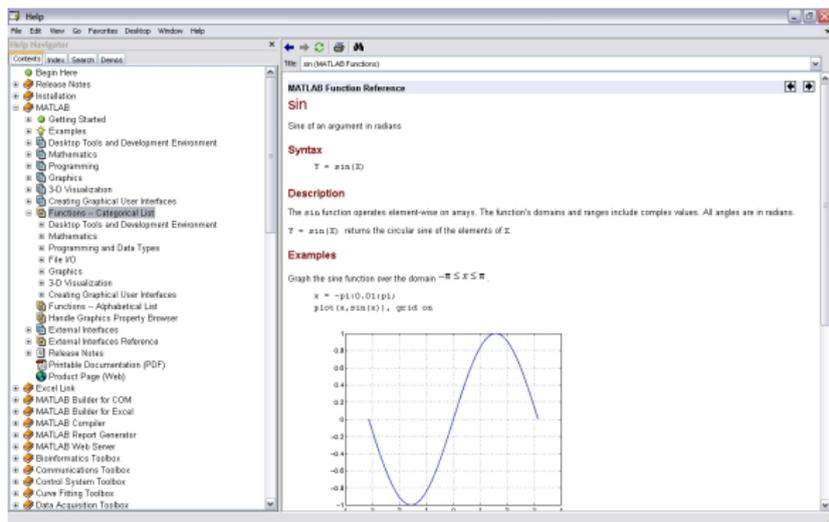
- ▶ help
- ▶ help arg
- ▶ help help
- ▶ lookfor arg

- Help Browser

- ▶ helpwin
- ▶ helpwin arg
- ▶ doc
- ▶ doc arg

- Demos

- [www.mathworks.com](http://www.mathworks.com)



The screenshot shows the MATLAB Help Browser interface. The left pane displays a tree view of the Help Browser with 'Functions - Alphabetical List' selected. The right pane displays the 'sin' function reference page, which includes the following content:

**MATLAB Function Reference**

**sin**

Sine of an argument in radians

**Syntax**

`T = sin(X)`

**Description**

The `sin` function operates element-wise on arrays. The function's domains and ranges include complex values. All angles are in radians. `T = sin(X)` returns the circular sine of the elements of `X`.

**Examples**

Graph the sine function over the domain  $-\pi \leq X \leq \pi$ .

```
x = -pi:0.01:pi;
plot(x,sin(x)), grid on
```

The plot shows a sine wave oscillating between -1 and 1 over the interval  $[-\pi, \pi]$ .

# MATLAB Help

- Help in linea

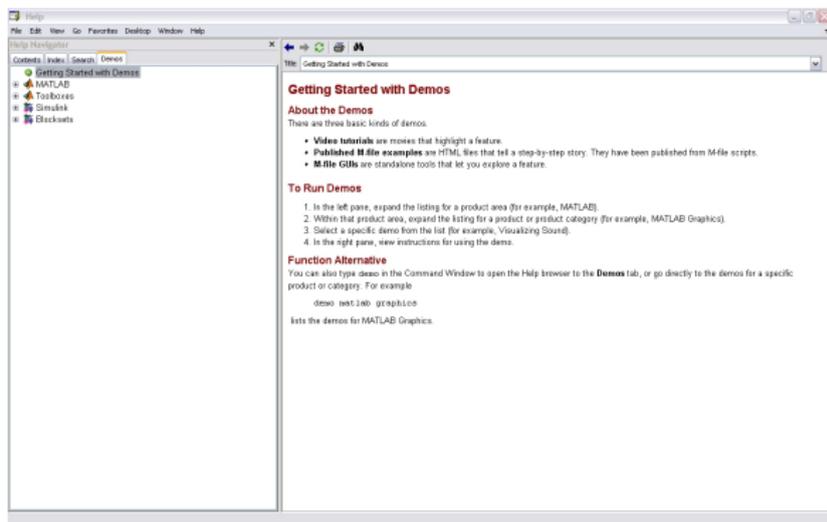
- ▶ help
- ▶ help arg
- ▶ help help
- ▶ lookfor arg

- Help Browser

- ▶ helpwin
- ▶ helpwin arg
- ▶ doc
- ▶ doc arg

- Demos

- [www.mathworks.com](http://www.mathworks.com)



# MATLAB Help

- Help in linea

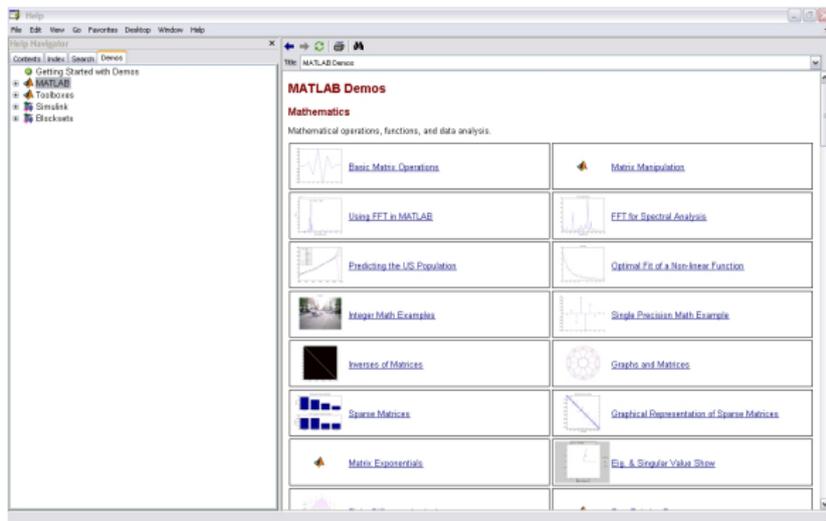
- ▶ `help`
- ▶ `help arg`
- ▶ `help help`
- ▶ `lookfor arg`

- Help Browser

- ▶ `helpwin`
- ▶ `helpwin arg`
- ▶ `doc`
- ▶ `doc arg`

- Demos

- [www.mathworks.com](http://www.mathworks.com)



# MATLAB Help

- Help in linea
  - ▶ help
  - ▶ help arg
  - ▶ help help
  - ▶ lookfor arg
- Help Browser
  - ▶ helpwin
  - ▶ helpwin arg
  - ▶ doc
  - ▶ doc arg
- Demos
- [www.mathworks.com](http://www.mathworks.com)

The screenshot shows the MathWorks website interface. At the top, there is a navigation bar with links for 'Products & Services', 'Industries', 'Academia', 'Support', 'User Community', and 'Company'. Below this, there are several promotional banners and sections:

- Featured Products:** Lists products like SimHydraulics, SimEvents, SystemTest, PolySpace, Simulink, and Distributed Computing Toolbox.
- New MATLAB and Simulink Student Version:** Promotes a new student version with a 'Learn More' link.
- R2007b Now Available:** Announces the release of MATLAB R2007b, scheduled for September 1, 2007.
- Featured Resources:** Includes links to articles on modeling standards, MATLAB's capabilities, and a video tutorial on dynamic data exchange.
- Events:** Promotes the 'SIMULINK® | 2007 World Tour' with a 'Free one-day event in 28 cities worldwide'.

The footer contains copyright information for 1994-2007 The MathWorks, Inc. and links to 'Site help', 'Patents', 'Trademarks', 'Privacy Policy', 'Preventing Piracy', and 'RSS'.

# Outline

- 1 Introduzione
- 2 Matrici**
- 3 Area di Lavoro
- 4 Visualizzazione di variabili
- 5 Espressioni
- 6 Funzioni *in – line*
- 7 Istruzioni per il controllo del flusso di esecuzione
- 8 *m – files*
- 9 Grafica

## Definizione di matrici

Convenzioni di base:

- separare gli elementi di una riga con spazi vuoti o virgole;
- usare un punto e virgola per indicare la fine di ogni riga;
- racchiudere l'intera lista degli elementi tra parentesi quadre.

Alcuni modi possibili per definire una matrice:

- definizione elemento per elemento;
- definizione a blocchi;
- generazione tramite funzioni predefinite (matrici elementari);
- caricamento da files di dati esterni.

## Definizione elemento per elemento

- Per definire la matrice

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix},$$

si può utilizzare il comando:

```
» A = [1 2 3; 4 5 6; 7 8 9]
```

oppure il comando

```
» A = [1 2 3
       4 5 6
       7 8 9]
```

Entrambi producono il risultato:

```
A =
```

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

- Il dimensionamento di una matrice è automatico:

```
» A = [1 0; 0 1]
```

```
A =
```

```
1     0
0     1
```

- Gli elementi precedentemente non definiti sono inizializzati a 0:

```
» A(2,4) = 1
```

```
A =
```

```
1     0     0     0
0     1     0     1
```

- Gli indici delle matrici devono essere interi positivi.*

## Definizione elemento per elemento

- Per definire la matrice

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix},$$

si può utilizzare il comando:

```
» A = [1 2 3; 4 5 6; 7 8 9]
```

oppure il comando

```
» A = [1 2 3
       4 5 6
       7 8 9]
```

Entrambi producono il risultato:

A =

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

- Il dimensionamento di una matrice è automatico:

```
» A = [1 0; 0 1]
```

A =

```
1     0
0     1
```

- Gli elementi precedentemente non definiti sono inizializzati a 0:

```
» A(2,4) = 1
```

A =

```
1     0     0     0
0     1     0     1
```

- Gli indici delle matrici devono essere interi positivi.*

## Definizione elemento per elemento

- Per definire la matrice

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix},$$

si può utilizzare il comando:

```
» A = [1 2 3; 4 5 6; 7 8 9]
```

oppure il comando

```
» A = [1 2 3
        4 5 6
        7 8 9]
```

Entrambi producono il risultato:

A =

1	2	3
4	5	6
7	8	9

- Il dimensionamento di una matrice è automatico:

```
» A = [1 0; 0 1]
```

A =

1	0
0	1

- Gli elementi precedentemente non definiti sono inizializzati a 0:

```
» A(2,4) = 1
```

A =

1	0	0	0
0	1	0	1

- Gli indici delle matrici devono essere interi positivi.*

## Definizione elemento per elemento

- Per definire la matrice

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix},$$

si può utilizzare il comando:

```
» A = [1 2 3; 4 5 6; 7 8 9]
```

oppure il comando

```
» A = [1 2 3
        4 5 6
        7 8 9]
```

Entrambi producono il risultato:

A =

1	2	3
4	5	6
7	8	9

- Il dimensionamento di una matrice è automatico:

```
» A = [1 0; 0 1]
```

A =

1	0
0	1

- Gli elementi precedentemente non definiti sono inizializzati a 0:

```
» A(2,4) = 1
```

A =

1	0	0	0
0	1	0	1

- Gli indici delle matrici devono essere interi positivi.*

## Definizione elemento per elemento

- Per definire la matrice

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix},$$

si può utilizzare il comando:

```
» A = [1 2 3; 4 5 6; 7 8 9]
```

oppure il comando

```
» A = [1 2 3
      4 5 6
      7 8 9]
```

Entrambi producono il risultato:

A =

1	2	3
4	5	6
7	8	9

- Il dimensionamento di una matrice è automatico:

```
» A = [1 0; 0 1]
```

A =

1	0
0	1

- Gli elementi precedentemente non definiti sono inizializzati a 0:

```
» A(2,4) = 1
```

A =

1	0	0	0
0	1	0	1

- Gli indici delle matrici devono essere interi positivi.*

## Definizione elemento per elemento

- Per definire la matrice

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix},$$

si può utilizzare il comando:

```
» A = [1 2 3; 4 5 6; 7 8 9]
```

oppure il comando

```
» A = [1 2 3
       4 5 6
       7 8 9]
```

Entrambi producono il risultato:

A =

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

- Il dimensionamento di una matrice è automatico:

```
» A = [1 0; 0 1]
```

A =

```
1     0
0     1
```

- Gli elementi precedentemente non definiti sono inizializzati a 0:

```
» A(2,4) = 1
```

A =

```
1     0     0     0
0     1     0     1
```

- Gli indici delle matrici devono essere interi positivi.*

## Definizione elemento per elemento

- Per definire la matrice

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix},$$

si può utilizzare il comando:

```
» A = [1 2 3; 4 5 6; 7 8 9]
```

oppure il comando

```
» A = [1 2 3
        4 5 6
        7 8 9]
```

Entrambi producono il risultato:

A =

1	2	3
4	5	6
7	8	9

- Il dimensionamento di una matrice è automatico:

```
» A = [1 0; 0 1]
```

A =

1	0
0	1

- Gli elementi precedentemente non definiti sono inizializzati a 0:

```
» A(2,4) = 1
```

A =

1	0	0	0
0	1	0	1

- Gli indici delle matrici devono essere interi positivi.*

## Definizione a blocchi

- Concatenazione di un vettore riga

```
» A = [1 2 3; 4 5 6; 7 8 9]
```

```
A =
```

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

```
» b = [10 11 12]
```

```
b =
```

```
    10    11    12
```

```
» C = [A; b]
```

```
C =
```

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

- Concatenazione di un vettore colonna

```
» A = [1 2 3; 4 5 6; 7 8 9]
```

```
A =
```

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

```
» b = [10; 11; 12]
```

```
b =
```

```
    10
    11
    12
```

```
» C = [A b]
```

```
C =
```

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

## Definizione a blocchi

- Concatenazione di un vettore riga

```
» A = [1 2 3; 4 5 6; 7 8 9]
```

```
A =
```

```

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

```
» b = [10 11 12]
```

```
b =
```

```

    10    11    12
```

```
» C = [A; b]
```

```
C =
```

```

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

- Concatenazione di un vettore colonna

```
» A = [1 2 3; 4 5 6; 7 8 9]
```

```
A =
```

```

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

```
» b = [10; 11; 12]
```

```
b =
```

```

    10
    11
    12
```

```
» C = [A b]
```

```
C =
```

```

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

## Definizione a blocchi

- Concatenazione di un vettore riga

```
» A = [1 2 3; 4 5 6; 7 8 9]
```

```
A =
```

```

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

```
» b = [10 11 12]
```

```
b =
```

```

    10    11    12
```

```
» C = [A; b]
```

```
C =
```

```

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

- Concatenazione di un vettore colonna

```
» A = [1 2 3; 4 5 6; 7 8 9]
```

```
A =
```

```

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

```
» b = [10; 11; 12]
```

```
b =
```

```

    10
    11
    12
```

```
» C = [A b]
```

```
C =
```

```

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

## Definizione a blocchi

- Concatenazione di un vettore riga

```
» A = [1 2 3; 4 5 6; 7 8 9]
```

```
A =
```

```

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

```
» b = [10 11 12]
```

```
b =
```

```

    10    11    12
```

```
» C = [A; b]
```

```
C =
```

```

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

- Concatenazione di un vettore colonna

```
» A = [1 2 3; 4 5 6; 7 8 9]
```

```
A =
```

```

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

```
» b = [10; 11; 12]
```

```
b =
```

```

    10
    11
    12
```

```
» C = [A b]
```

```
C =
```

```

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

## Definizione a blocchi

- Concatenazione di un vettore riga

```
» A = [1 2 3; 4 5 6; 7 8 9]
```

```
A =
```

```

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

```
» b = [10 11 12]
```

```
b =
```

```

    10    11    12
```

```
» C = [A; b]
```

```
C =
```

```

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

- Concatenazione di un vettore colonna

```
» A = [1 2 3; 4 5 6; 7 8 9]
```

```
A =
```

```

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

```
» b = [10; 11; 12]
```

```
b =
```

```

    10
    11
    12
```

```
» C = [A b]
```

```
C =
```

```

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

## Definizione a blocchi

- Concatenazione di un vettore riga

```
» A = [1 2 3; 4 5 6; 7 8 9]
```

```
A =
```

```

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

```
» b = [10 11 12]
```

```
b =
```

```

    10    11    12
```

```
» C = [A; b]
```

```
C =
```

```

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

- Concatenazione di un vettore colonna

```
» A = [1 2 3; 4 5 6; 7 8 9]
```

```
A =
```

```

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

```
» b = [10; 11; 12]
```

```
b =
```

```

    10
    11
    12
```

```
» C = [A b]
```

```
C =
```

```

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

## Definizione a blocchi

- Concatenazione di un vettore riga

```
» A = [1 2 3; 4 5 6; 7 8 9]
```

```
A =
```

```

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

```
» b = [10 11 12]
```

```
b =
```

```

    10    11    12
```

```
» C = [A; b]
```

```
C =
```

```

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

- Concatenazione di un vettore colonna

```
» A = [1 2 3; 4 5 6; 7 8 9]
```

```
A =
```

```

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

```
» b = [10; 11; 12]
```

```
b =
```

```

    10
    11
    12
```

```
» C = [A b]
```

```
C =
```

```

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

## Definizione a blocchi

- Concatenazione di un vettore riga

```
» A = [1 2 3; 4 5 6; 7 8 9]
```

```
A =
```

```

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

```
» b = [10 11 12]
```

```
b =
```

```

    10    11    12
```

```
» C = [A; b]
```

```
C =
```

```

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

- Concatenazione di un vettore colonna

```
» A = [1 2 3; 4 5 6; 7 8 9]
```

```
A =
```

```

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

```
» b = [10; 11; 12]
```

```
b =
```

```

    10
    11
    12
```

```
» C = [A b]
```

```
C =
```

```

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

## Matrici elementari: zeros, ones, rand, eye

- `zeros(n,m)`: matrice  $n \times m$  con elementi tutti uguali a 0

```
» Z = zeros(2,4)
```

```
Z =
```

```
    0    0    0    0
    0    0    0    0
```

- `ones(n,m)`: matrice  $n \times m$  con elementi tutti uguali a 1

```
» O = ones(3,5)
```

```
O =
```

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

- `rand(n,m)`: matrice  $n \times m$  con elementi casuali tra 0 e 1

```
» R = rand(2,3)
```

```
R =
```

```
    0.9501    0.2311    0.6068
    0.4860    0.8913    0.7621
```

- `eye(n)`: matrice identità di ordine  $n$

```
» E = eye(3)
```

```
E =
```

```
    1    0    0
    0    1    0
    0    0    1
```

## Matrici elementari: zeros, ones, rand, eye

- **zeros(n,m)**: matrice  $n \times m$  con elementi tutti uguali a 0

```
» Z = zeros(2,4)
```

```
Z =
```

```
    0    0    0    0
    0    0    0    0
```

- **ones(n,m)**: matrice  $n \times m$  con elementi tutti uguali a 1

```
» O = ones(3,5)
```

```
O =
```

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

- **rand(n,m)**: matrice  $n \times m$  con elementi casuali tra 0 e 1

```
» R = rand(2,3)
```

```
R =
```

```
    0.9501    0.2311    0.6068
    0.4860    0.8913    0.7621
```

- **eye(n)**: matrice identità di ordine  $n$

```
» E = eye(3)
```

```
E =
```

```
    1    0    0
    0    1    0
    0    0    1
```

## Matrici elementari: zeros, ones, rand, eye

- **zeros(n,m)**: matrice  $n \times m$  con elementi tutti uguali a 0

```
» Z = zeros(2,4)
```

```
Z =
```

```
    0    0    0    0
    0    0    0    0
```

- **ones(n,m)**: matrice  $n \times m$  con elementi tutti uguali a 1

```
» O = ones(3,5)
```

```
O =
```

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

- **rand(n,m)**: matrice  $n \times m$  con elementi casuali tra 0 e 1

```
» R = rand(2,3)
```

```
R =
```

```
    0.9501    0.2311    0.6068
    0.4860    0.8913    0.7621
```

- **eye(n)**: matrice identità di ordine  $n$

```
» E = eye(3)
```

```
E =
```

```
    1    0    0
    0    1    0
    0    0    1
```

## Matrici elementari: zeros, ones, rand, eye

- `zeros(n,m)`: matrice  $n \times m$  con elementi tutti uguali a 0

```
» Z = zeros(2,4)
```

```
Z =
```

```
    0    0    0    0
    0    0    0    0
```

- `ones(n,m)`: matrice  $n \times m$  con elementi tutti uguali a 1

```
» O = ones(3,5)
```

```
O =
```

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

- `rand(n,m)`: matrice  $n \times m$  con elementi casuali tra 0 e 1

```
» R = rand(2,3)
```

```
R =
```

```
    0.9501    0.2311    0.6068
    0.4860    0.8913    0.7621
```

- `eye(n)`: matrice identità di ordine  $n$

```
» E = eye(3)
```

```
E =
```

```
    1    0    0
    0    1    0
    0    0    1
```

## Matrici elementari: zeros, ones, rand, eye

- `zeros(n,m)`: matrice  $n \times m$  con elementi tutti uguali a 0

```
» Z = zeros(2,4)
```

```
Z =
```

```
    0    0    0    0
    0    0    0    0
```

- `ones(n,m)`: matrice  $n \times m$  con elementi tutti uguali a 1

```
» O = ones(3,5)
```

```
O =
```

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

- `rand(n,m)`: matrice  $n \times m$  con elementi casuali tra 0 e 1

```
» R = rand(2,3)
```

```
R =
```

```
    0.9501    0.2311    0.6068
    0.4860    0.8913    0.7621
```

- `eye(n)`: matrice identità di ordine  $n$

```
» E = eye(3)
```

```
E =
```

```
    1    0    0
    0    1    0
    0    0    1
```

## Matrici elementari: zeros, ones, rand, eye

- `zeros(n,m)`: matrice  $n \times m$  con elementi tutti uguali a 0

```
» Z = zeros(2,4)
```

```
Z =
```

```
    0    0    0    0
    0    0    0    0
```

- `ones(n,m)`: matrice  $n \times m$  con elementi tutti uguali a 1

```
» O = ones(3,5)
```

```
O =
```

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

- `rand(n,m)`: matrice  $n \times m$  con elementi casuali tra 0 e 1

```
» R = rand(2,3)
```

```
R =
```

```
    0.9501    0.2311    0.6068
    0.4860    0.8913    0.7621
```

- `eye(n)`: matrice identità di ordine  $n$

```
» E = eye(3)
```

```
E =
```

```
    1    0    0
    0    1    0
    0    0    1
```

## Matrici elementari: zeros, ones, rand, eye

- `zeros(n,m)`: matrice  $n \times m$  con elementi tutti uguali a 0

```
» Z = zeros(2,4)
```

```
Z =
```

```
    0    0    0    0
    0    0    0    0
```

- `ones(n,m)`: matrice  $n \times m$  con elementi tutti uguali a 1

```
» O = ones(3,5)
```

```
O =
```

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

- `rand(n,m)`: matrice  $n \times m$  con elementi casuali tra 0 e 1

```
» R = rand(2,3)
```

```
R =
```

```
    0.9501    0.2311    0.6068
    0.4860    0.8913    0.7621
```

- `eye(n)`: matrice identità di ordine  $n$

```
» E = eye(3)
```

```
E =
```

```
    1    0    0
    0    1    0
    0    0    1
```

## Matrici elementari: zeros, ones, rand, eye

- `zeros(n,m)`: matrice  $n \times m$  con elementi tutti uguali a 0

```
» Z = zeros(2,4)
```

```
Z =
```

```
    0    0    0    0
    0    0    0    0
```

- `ones(n,m)`: matrice  $n \times m$  con elementi tutti uguali a 1

```
» O = ones(3,5)
```

```
O =
```

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

- `rand(n,m)`: matrice  $n \times m$  con elementi casuali tra 0 e 1

```
» R = rand(2,3)
```

```
R =
```

```
    0.9501    0.2311    0.6068
    0.4860    0.8913    0.7621
```

- `eye(n)`: matrice identità di ordine  $n$

```
» E = eye(3)
```

```
E =
```

```
    1    0    0
    0    1    0
    0    0    1
```

## Matrici elementari: zeros, ones, rand, eye

- `zeros(n,m)`: matrice  $n \times m$  con elementi tutti uguali a 0

» `Z = zeros(2,4)`

Z =

```

0     0     0     0
0     0     0     0

```

- `ones(n,m)`: matrice  $n \times m$  con elementi tutti uguali a 1

» `O = ones(3,5)`

O =

```

1     1     1     1     1
1     1     1     1     1
1     1     1     1     1

```

- `rand(n,m)`: matrice  $n \times m$  con elementi casuali tra 0 e 1

» `R = rand(2,3)`

R =

```

0.9501    0.2311    0.6068
0.4860    0.8913    0.7621

```

- `eye(n)`: matrice identità di ordine  $n$

» `E = eye(3)`

E =

```

1     0     0
0     1     0
0     0     1

```

# Caricamento da files di dati esterni

## ● load

- ▶ creare il file `A.dat` contenente le tre linee di dati:

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

- ▶ eseguire il comando:  
» `load A.dat`
- ▶ è stata creata la matrice di esempio

A:

» A

A =

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

## ● *m-files*

- ▶ creare il file `exe.m` contenente le tre linee di codice:

```
A = [1 2 3
      4 5 6
      7 8 9]
```

- ▶ eseguire il comando:  
» `exe`
- ▶ è stata creata la matrice di esempio

A:

» A

A =

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

# Operatore :

- Definizione di vettori riga con elementi equispaziati:

```
» v = 1:5
```

```
v =
```

```
    1    2    3    4    5
```

```
» v = 10:-2:2
```

```
v =
```

```
   10    8    6    4    2
```

```
» v = 0:1/3:1
```

```
v =
```

```
    0    0.3333    0.6667    1.0000
```

- Riferimento a sottomatrici:

```
A(vriga,vcolonna)
```

```
» r = A(1,:)
```

```
r =
```

```
    1    2    3
```

```
» c = A(:,2)
```

```
c =
```

```
    2
```

```
    5
```

```
    8
```

```
» S = A(1:2,end:-1:2)
```

```
S =
```

```
    3    2
```

```
    6    5
```

# Outline

- 1 Introduzione
- 2 Matrici
- 3 Area di Lavoro**
- 4 Visualizzazione di variabili
- 5 Espressioni
- 6 Funzioni *in – line*
- 7 Istruzioni per il controllo del flusso di esecuzione
- 8 *m – files*
- 9 Grafica

## Gestione dell'area di lavoro: `who` e `whos`

Le variabili utilizzate sono memorizzate nell'**area di lavoro** (*Workspace*).

- Informazioni sulle variabili presenti:

» `who`

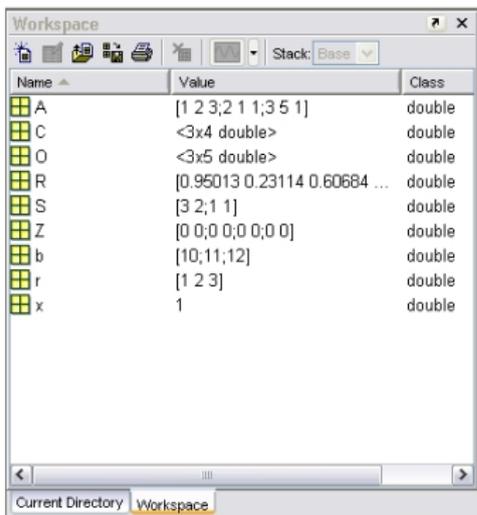
Your variables are:

A C O R S Z b r x

» `whos`

Name	Size	Bytes	Class
A	3x3	72	double array
C	3x4	96	double array
O	3x5	120	double array
R	1x6	48	double array
S	2x2	32	double array
Z	4x2	64	double array
b	3x1	24	double array
r	1x3	24	double array
x	1x1	8	double array

Grand total is 61 elements using 488 bytes



## Gestione dell'area di lavoro: `clear`, `save`, `load`

- Eliminazione delle variabili presenti
  - ▶ eliminazione di tutte le variabili
    - » `clear`
  - ▶ eliminazione della variabili A e b
    - » `clear A b`
- Salvataggio di dati
  - ▶ salvataggio di tutte le variabili
    - » `save`
    - Saving to: `matlab.mat`
  - ▶ salvataggio della variabili A e b nel file di nome `file1.mat`
    - » `save file1 A b`
- Richiamo di dati
  - ▶ caricamento nell'area di lavoro di variabili salvate nel file `matlab.mat`
    - » `load`
    - Loading from: `matlab.mat`
  - ▶ caricamento nell'area di lavoro di variabili salvate in un file `".mat"`
    - » `load file1`

# Outline

- 1 Introduzione
- 2 Matrici
- 3 Area di Lavoro
- 4 Visualizzazione di variabili**
- 5 Espressioni
- 6 Funzioni *in – line*
- 7 Istruzioni per il controllo del flusso di esecuzione
- 8 *m – files*
- 9 Grafica

## Visualizzazione di variabili

Due modi possibili per visualizzare il valore di una variabile:

- 1 digitando il nome della variabile e premendo invio  
⇒ viene visualizzato sia il nome della variabile che il suo valore;
- 2 utilizzando il comando `disp`  
⇒ viene visualizzato soltanto il valore della variabile in argomento.

Il comando:

```
» format [opzioni]
```

specifica il formato di visualizzazione del valore di una variabile.

- N.B. Il comando `format` definisce unicamente il formato di visualizzazione e **non** il tipo di una variabile.

## Visualizzazione di variabili

Due modi possibili per visualizzare il valore di una variabile:

- 1 digitando il nome della variabile e premendo invio  
⇒ viene visualizzato sia il nome della variabile che il suo valore;
- 2 utilizzando il comando `disp`  
⇒ viene visualizzato soltanto il valore della variabile in argomento.

Il comando:

```
» format [opzioni]
```

specifica il formato di visualizzazione del valore di una variabile.

- N.B. Il comando `format` definisce unicamente il formato di visualizzazione e **non** il tipo di una variabile.

## Visualizzazione di variabili

Due modi possibili per visualizzare il valore di una variabile:

- 1 digitando il nome della variabile e premendo invio  
⇒ viene visualizzato sia il nome della variabile che il suo valore;
- 2 utilizzando il comando `disp`  
⇒ viene visualizzato soltanto il valore della variabile in argomento.

Il comando:

```
» format [opzioni]
```

specifica il formato di visualizzazione del valore di una variabile.

- N.B. Il comando `format` definisce unicamente il formato di visualizzazione e **non** il tipo di una variabile.

## Alcune opzioni del comando `format`

- Formato *fixed-point* con 5 cifre

```
» format short, x
```

```
x =
```

```
0.3333
```

- Formato *fixed-point* con 15 cifre

```
» format long, x
```

```
x =
```

```
0.333333333333333
```

- Segno

```
» format +, x
```

```
x =
```

```
+
```

- Formato *floating-point* con 5 cifre

```
» format short e, x
```

```
x =
```

```
3.3333e-001
```

- Formato *floating-point* con 15 cifre

```
» format long e, x
```

```
x =
```

```
3.333333333333333e-001
```

- Formato razionale

```
» format rat, x
```

```
x =
```

```
1/3
```

# Outline

- 1 Introduzione
- 2 Matrici
- 3 Area di Lavoro
- 4 Visualizzazione di variabili
- 5 Espressioni**
- 6 Funzioni *in – line*
- 7 Istruzioni per il controllo del flusso di esecuzione
- 8 *m – files*
- 9 Grafica

## Espressioni

MATLAB è un interprete le cui istruzioni sono del tipo

» [variabile =] espressione

dove le espressioni sono costituite mediante identificatori di variabile, costanti, operatori e funzioni.

- L'espressione al membro destro dell'istruzione viene valutata e il risultato viene assegnato alla variabile specificata.
- Su una stessa riga di comando è possibile scrivere più istruzioni separandole mediante una ',' o un ';'.
- Se si omette il membro sinistro dell'istruzione allora il risultato dell'espressione viene assegnato alla variabile di *default* `ans` che viene poi mantenuta nell'area di lavoro. Tale variabile può quindi essere successivamente utilizzata.

## Espressioni

MATLAB è un interprete le cui istruzioni sono del tipo

» [variabile =] espressione

dove le espressioni sono costituite mediante identificatori di variabile, costanti, operatori e funzioni.

- L'espressione al membro destro dell'istruzione viene valutata e il risultato viene assegnato alla variabile specificata.
- Su una stessa riga di comando è possibile scrivere più istruzioni separandole mediante una ',' o un ';'.
- Se si omette il membro sinistro dell'istruzione allora il risultato dell'espressione viene assegnato alla variabile di *default* `ans` che viene poi mantenuta nell'area di lavoro. Tale variabile può quindi essere successivamente utilizzata.

## Operatori aritmetici

### Operatore

### Descrizione

- $+$ ,  $-$   $\Rightarrow$  addizione e sottrazione di matrici;
  - $*$   $\Rightarrow$  prodotto matrice-matrice;
  - $\backslash$   $\Rightarrow$  divisione a sinistra  
( $A \backslash B \equiv A^{-1} * B$ , se  $A$  è non singolare);
  - $/$   $\Rightarrow$  divisione a destra  
( $A / B \equiv A * B^{-1}$ , se  $B$  è non singolare).
- 
- Le dimensioni degli operandi a cui vengono applicati gli operatori aritmetici binari devono essere compatibili con le usuali regole algebriche.
  - Le regole di precedenza degli operatori sono quelle usuali dell'algebra. Per modificarle basta utilizzare le '(' ')

## Operazioni elemento per elemento

- $\gg C = A .* B \quad \Rightarrow \quad C = (c_{ij}), \text{ con } c_{ij} = a_{ij} * b_{ij};$
- $\gg C = A ./ B \quad \Rightarrow \quad C = (c_{ij}), \text{ con } c_{ij} = b_{ij} / a_{ij}.$
- $\gg C = A ./ B \quad \Rightarrow \quad C = (c_{ij}), \text{ con } c_{ij} = a_{ij} / b_{ij};$
- $\gg C = A .^ B \quad \Rightarrow \quad C = (c_{ij}), \text{ con } c_{ij} = a_{ij} ^ b_{ij}.$

```
 $\gg A$ 
```

```
A =
```

```

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

```
 $\gg B$ 
```

```
B = 2*ones(3)
```

```

2     2     2
2     2     2
2     2     2
```

```
 $\gg A .* B$ 
```

```
ans =
```

```

2     4     6
8    10    12
14    16    18
```

```
 $\gg A ./ B$ 
```

```
ans =
```

```

0.5000    1.0000    1.5000
2.0000    2.5000    3.0000
3.5000    4.0000    4.5000
```

## Operazioni elemento per elemento

- »  $C = A .* B \Rightarrow C = (c_{ij})$ , con  $c_{ij} = a_{ij} * b_{ij}$ ;
- »  $C = A ./ B \Rightarrow C = (c_{ij})$ , con  $c_{ij} = b_{ij} / a_{ij}$ .
- »  $C = A ./ B \Rightarrow C = (c_{ij})$ , con  $c_{ij} = a_{ij} / b_{ij}$ ;
- »  $C = A .^ B \Rightarrow C = (c_{ij})$ , con  $c_{ij} = a_{ij} ^ b_{ij}$ .

```
» A
```

```
A =
```

```

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

```
» B
```

```
B = 2*ones(3)
```

```

     2     2     2
     2     2     2
     2     2     2
```

```
» A.*B
```

```
ans =
```

```

     2     4     6
     8    10    12
    14    16    18
```

```
» A./B
```

```
ans =
```

```

    0.5000    1.0000    1.5000
    2.0000    2.5000    3.0000
    3.5000    4.0000    4.5000
```

## Operatori relazionali e logici

Operatore	Descrizione
• <	⇒ minore;
• <=	⇒ minore o uguale;
• >	⇒ maggiore;
• >=	⇒ maggiore o uguale;
• ==	⇒ uguaglianza;
• ~ =	⇒ non uguaglianza;
• &	⇒ <i>and</i> logico;
•	⇒ <i>or</i> logico;
• ~	⇒ <i>not</i> logico.

Si applicano a matrici e restituiscono una matrice di risultati pari a 1, se la relazione è verificata, 0 altrimenti.

## Operazioni con uno scalare

Se uno degli operandi è uno scalare, allora

- qualunque operatore è da interpretarsi come preceduto da un punto,
- lo scalare viene interpretato come una matrice con le stesse dimensioni dell'altro operando e valori tutti uguali al valore dello scalare.

```
» A
```

```
A =
```

```

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

```
» A+1
```

```
ans =
```

```

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

```
» A*2
```

```
ans =
```

```

2     4     6
8    10    12
14    16    18
```

```
» A/3
```

```
ans =
```

```

0.3333    0.6667    1.0000
1.3333    1.6667    2.0000
2.3333    2.6667    3.0000
```

# Outline

- 1 Introduzione
- 2 Matrici
- 3 Area di Lavoro
- 4 Visualizzazione di variabili
- 5 Espressioni
- 6 Funzioni *in – line***
- 7 Istruzioni per il controllo del flusso di esecuzione
- 8 *m – files*
- 9 Grafica

## Funzioni *in – line*

MATLAB mette a disposizione numerose funzioni *in-line* per la elaborazione di scalari, vettori e matrici.

### Tipi di funzione

- variabili e costanti predefinite
  - ▶ restituiscono il valore di alcune variabili e costanti matematiche o relative al sistema *floating-point*;
- funzioni scalari (vedi `help elfun`)
  - ▶ una funzione scalare applicata ad una matrice opera elemento per elemento;
- funzioni vettoriali (vedi `help datafun`)
  - ▶ una funzione vettoriale applicata ad una matrice opera colonna per colonna, restituendo un vettore riga;
- funzioni matriciali (vedi `help elmat`).

## Alcune variabili e costanti predefinite

- `ans` : risposta (*answer*) più recente
- `eps` :  $\epsilon$  di macchina
- `realmax`: numero *floating point* positivo più grande utilizzabile
- `realmin`: numero *floating point* positivo più piccolo utilizzabile

- `pi` : *floating* di  $\pi$

```
» pi
```

```
ans =
```

```
3.1416
```

- `i`, `j`: unità immaginaria ( $i = j = \sqrt{-1}$ )

```
» i
```

```
ans =
```

```
0 + 1.0000i
```

- `inf`: infinito ( $\infty$ )

```
» 1/0
```

```
Warning: Divide by zero.
```

```
ans =
```

```
Inf
```

```
» -1/0
```

```
Warning: Divide by zero.
```

```
ans =
```

```
-Inf
```

- `NaN`: not-a-number

```
» 0/0
```

```
Warning: Divide by zero.
```

```
ans =
```

```
NaN
```

# Funzioni scalari

## Funzioni matematiche di base:

- trigonometriche:

- ▶ `sin(x)`: seno di  $x$ ,
- ▶ `cos(x)`: coseno di  $x$ ,
- ▶ `tan(x)`: tangente di  $x$ ,
- ▶ `asin(x)`: arcoseno di  $x$ ,
- ▶ `acos(x)`: arcocoseno di  $x$ ,
- ▶ `atan(x)`: arcotangente di  $x$ ,
- ▶ ...

- esponenziali:

- ▶ `sqrt(x)`: radice quadrata di  $x$ ,
- ▶ `exp(x)`: esponenziale di  $x$ ,
- ▶ `log(x)`: logaritmo naturale di  $x$ ,
- ▶ `log10(x)`: logaritmo in base 10 di  $x$ ,
- ▶ `log2(x)`: logaritmo in base 2 di  $x$ ,
- ▶ ...

- complesse:

- ▶ `real(x)`: parte reale di  $x$ ,
- ▶ `imag(x)`: parte immaginaria di  $x$ ,
- ▶ `conj(x)`: coniugato di  $x$ ,
- ▶ ...

- arrotondamento:

- ▶ `fix(x)`: troncamento di  $x$  all'intero più piccolo più vicino,
- ▶ `round(x)`: arrotondamento di  $x$  all'intero più vicino,
- ▶ `floor(x)`: parte intera inferiore di  $x$  (`⌊x⌋`),
- ▶ `ceil(x)`: parte intera superiore di  $x$  (`⌈x⌉`),
- ▶ ...

# Funzioni vettoriali

Alcune operazioni di base:

- `length(v)`: restituisce la lunghezza del vettore  $v$ ;
- `transpose(v)`: restituisce il trasposto del vettore  $v$  (analogo a  $v'$ );
- `ctranspose(v)`: restituisce il trasposto coniugato del vettore  $v$  (analogo a  $v'$ ).

Funzioni per l'analisi dei dati contenuti in un vettore:

- `max(v)`: restituisce la componente più grande del vettore  $v$ ;
- `min(v)`: restituisce la componente più piccola del vettore  $v$ ;
- `mean(v)`: restituisce il valore medio degli elementi del vettore  $v$ ;
- `sort(v)`: restituisce gli elementi del vettore  $v$  in ordine crescente;
- `sum(v)`: restituisce la somma degli elementi del vettore  $v$ ;
- `prod(v)`: restituisce il prodotto degli elementi del vettore  $v$ ;
- ...

# Funzioni vettoriali

Alcune operazioni di base:

- `length(v)`: restituisce la lunghezza del vettore  $v$ ;
- `transpose(v)`: restituisce il trasposto del vettore  $v$  (analogo a  $v'$ );
- `ctranspose(v)`: restituisce il trasposto coniugato del vettore  $v$  (analogo a  $v'$ ).

Funzioni per l'analisi dei dati contenuti in un vettore:

- `max(v)`: restituisce la componente più grande del vettore  $v$ ;
- `min(v)`: restituisce la componente più piccola del vettore  $v$ ;
- `mean(v)`: restituisce il valore medio degli elementi del vettore  $v$ ;
- `sort(v)`: restituisce gli elementi del vettore  $v$  in ordine crescente;
- `sum(v)`: restituisce la somma degli elementi del vettore  $v$ ;
- `prod(v)`: restituisce il prodotto degli elementi del vettore  $v$ ;
- ...

# Funzioni matriciali

Alcune informazioni di base sulle matrici:

- `size(A)`: restituisce la dimensione della matrice `A`;
- `numel(A)`: restituisce il numero di elementi della matrice `A`.

Funzioni per la manipolazione di matrici:

- `reshape(A)`: modifica le dimensioni della matrice `A`;
- `diag(A)`: estrae la diagonale di una matrice oppure crea una matrice diagonale;
- `tril(A)`: estrae la parte triangolare inferiore della matrice `A`;
- `triu(A)`: estrae la parte triangolare superiore della matrice `A`;
- `flipplr(A)`: inverte l'ordine delle colonne della matrice `A`;
- `flipud(A)`: inverte l'ordine delle righe della matrice `A`;
- `find(A)`: trova gli indici degli elementi non nulli della matrice `A`;
- `end`: ultimo indice;
- ...

# Funzioni matriciali

Alcune informazioni di base sulle matrici:

- `size(A)`: restituisce la dimensione della matrice `A`;
- `numel(A)`: restituisce il numero di elementi della matrice `A`.

Funzioni per la manipolazione di matrici:

- `reshape(A)`: modifica le dimensioni della matrice `A`;
- `diag(A)`: estrae la diagonale di una matrice oppure crea una matrice diagonale;
- `tril(A)`: estrae la parte triangolare inferiore della matrice `A`;
- `triu(A)`: estrae la parte triangolare superiore della matrice `A`;
- `fliplr(A)`: inverte l'ordine delle colonne della matrice `A`;
- `flipud(A)`: inverte l'ordine delle righe della matrice `A`;
- `find(A)`: trova gli indici degli elementi non nulli della matrice `A`;
- `end`: ultimo indice;
- ...

# Funzioni matriciali

## Analisi di matrici:

- `norm(A)` o `norm(A,2)`: calcola la norma 2 della matrice A, ovvero  $\|A\|_2$ ;
- `norm(A,1)`: calcola la norma 1 della matrice A, ovvero  $\|A\|_1$ ;
- `norm(A,inf)`: calcola la norma  $\infty$  della matrice A, ovvero  $\|A\|_\infty$ ;
- `rank(A)`: calcola il rango della matrice A;
- `det(A)`: calcola il determinante della matrice A;
- `trace(A)`: calcola la somma degli elementi diagonali di A;
- `null(A)`: calcola lo spazio nullo della matrice A;
- ...

## Equazioni lineari:

- `inv(A)`: calcola la matrice inversa di A;
- `/` e `\`: risoluzione di equazioni lineari;
- `cond(A)`: calcola il numero di condizionamento di A, ovvero  $k(A) = \|A\|_2 \|A^{-1}\|_2$ ;
- ...

# Funzioni matriciali

## Analisi di matrici:

- `norm(A)` o `norm(A,2)`: calcola la norma 2 della matrice A, ovvero  $\|A\|_2$ ;
- `norm(A,1)`: calcola la norma 1 della matrice A, ovvero  $\|A\|_1$ ;
- `norm(A,inf)`: calcola la norma  $\infty$  della matrice A, ovvero  $\|A\|_\infty$ ;
- `rank(A)`: calcola il rango della matrice A;
- `det(A)`: calcola il determinante della matrice A;
- `trace(A)`: calcola la somma degli elementi diagonali di A;
- `null(A)`: calcola lo spazio nullo della matrice A;
- ...

## Equazioni lineari:

- `inv(A)`: calcola la matrice inversa di A;
- `/` e `\`: risoluzione di equazioni lineari;
- `cond(A)`: calcola il numero di condizionamento di A, ovvero  $k(A) = \|A\|_2 \|A^{-1}\|_2$ ;
- ...

# Outline

- 1 Introduzione
- 2 Matrici
- 3 Area di Lavoro
- 4 Visualizzazione di variabili
- 5 Espressioni
- 6 Funzioni *in – line*
- 7 Istruzioni per il controllo del flusso di esecuzione**
- 8 *m – files*
- 9 Grafica

# Istruzioni per il controllo del flusso di esecuzione

Vedi `help lang`

## Principali istruzioni

- istruzione `if`;
- istruzione `for`;
- istruzione `while`;
- istruzione `switch`.

## Istruzione `if`

Sintassi:

```
if (espressione booleana)  
    istruzioni  
elseif (espressione booleana)  
    istruzioni  
else  
    istruzioni  
end
```

Dove:

- ***espressione booleana*** è una qualsiasi espressione costruita con operatori logici e/o relazionali (se il risultato è una matrice, la condizione è ritenuta vera se tutti gli elementi di tale matrice sono non nulli);
- i blocchi ***elseif*** ed ***else*** non sono obbligatori;
- il blocco ***elseif*** può essere ripetuto più di una volta.

# Istruzione `for`

Sintassi:

```
for ind = v  
    istruzioni  
end
```

***v*** può essere:

- un vettore riga

- ▶ il ciclo ***for*** viene ripetuto tante volte quanto vale la lunghezza del vettore ***v***, assegnando a ***ind*** i valori del vettore in sequenza;
- ▶ esempio

```
for i = 1:10  
    ...  
end
```

- una matrice

- ▶ il ciclo ***for*** viene ripetuto tante volte quante sono le colonne della matrice ***v***, assegnando a ***ind*** i valori di tali colonne in sequenza.

## Istruzione `while`

Sintassi:

```
while (espressione booleana)  
    istruzioni  
end
```

- ***espressione booleana*** è una qualsiasi espressione costruita con operatori logici e/o relazionali (se il risultato è una matrice, la condizione è ritenuta vera se tutti gli elementi di tale matrice sono non nulli);
- il ciclo ***while*** viene ripetuto fino a quando l'***espressione booleana*** risulta vera ( $\neq 0$ ).

## Istruzione `break`

- termina l'esecuzione di cicli ***for*** e ***while***;
- in cicli annidati, permette di uscire dal ciclo più interno.

## Istruzione `while`

Sintassi:

```
while (espressione booleana)  
    istruzioni  
end
```

- ***espressione booleana*** è una qualsiasi espressione costruita con operatori logici e/o relazionali (se il risultato è una matrice, la condizione è ritenuta vera se tutti gli elementi di tale matrice sono non nulli);
- il ciclo ***while*** viene ripetuto fino a quando l'***espressione booleana*** risulta vera ( $\neq 0$ ).

## Istruzione `break`

- termina l'esecuzione di cicli ***for*** e ***while***;
- in cicli annidati, permette di uscire dal ciclo più interno.

## Istruzione switch

Sintassi:

```
switch switch_expr
    case case_expr1
        istruzioni
    case {case_expr1, case_expr2, case_expr3,...}
        istruzioni
    ...
    [otherwise
     istruzioni]
end
```

- vengono eseguite le istruzioni associate al primo case per il quale **switch\_expr == case\_expr**;
- se l'espressione associata al **case** è composta da diversi elementi, viene eseguito il blocco di istruzioni corrispondente se **switch\_expr** è uguale ad uno qualsiasi delle espressioni nella lista;
- se nessuna delle espressioni associate ai **case** uguaglia la **switch\_expr** viene eseguito il blocco di istruzioni associato a **otherwise** (se esso esiste).

# Outline

- 1 Introduzione
- 2 Matrici
- 3 Area di Lavoro
- 4 Visualizzazione di variabili
- 5 Espressioni
- 6 Funzioni *in – line*
- 7 Istruzioni per il controllo del flusso di esecuzione
- 8 *m – files***
- 9 Grafica

## *m - file: script e function*

### *m - file*

File di estensione `.m` in cui sono memorizzate una serie di istruzioni MATLAB.

Due tipologie possibili:

- **script**

- ▶ sequenza di istruzioni MATLAB;
- ▶ le variabili definite sono globali;
- ▶ per eseguirlo basta digitare il nome del file;

- **function**

- ▶ utilizzano parametri di *input* e di *output*;
- ▶ le variabili definite sono locali.

# Function

Intestazione:

```
function [output1,output2,...] = nomefunction(input1,input2...)
```

dove

- `input1,input2,...` è la lista dei parametri formali di *input*;
- `output1,output2,...` è la lista dei parametri formali di *output*.

Esecuzione:

```
> [outputA1,outputA2,...] = nomefunction(inputA1,inputA2...)
```

dove

- `inputA1,inputA2,...` è la lista dei parametri attuali di *input*;
- `outputA1,outputA2,...` è la lista dei parametri attuali di *output*.

Nel caso di un solo *output* si possono omettere le [ ].

# Function

Intestazione:

```
function [output1,output2,...] = nomefunction(input1,input2...)
```

dove

- `input1,input2,...` è la lista dei parametri formali di *input*;
- `output1,output2,...` è la lista dei parametri formali di *output*.

Esecuzione:

```
» [outputA1,outputA2,...] = nomefunction(inputA1,inputA2...)
```

dove

- `inputA1,inputA2,...` è la lista dei parametri attuali di *input*;
- `outputA1,outputA2,...` è la lista dei parametri attuali di *output*.

Nel caso di un solo *output* si possono omettere le [ ].

# Function

- Per chiamare una *function* basta digitare il nome del file (senza suffisso) in cui è memorizzata (tale nome può anche non coincidere con il nome della *function*).
- L'esecuzione di una *function* termina in corrispondenza del comando `return` o in corrispondenza della fine del file.
- In un *m – file* possono essere memorizzate diverse *function*:
  - ▶ l'unica accessibile dall'esterno è la prima;
  - ▶ le successive sono da considerare come sottoprocedure della prima.
- Il carattere `%` permette di inserire una riga di commenti all'interno di un *m – file*:
  - ▶ le righe di commento consecutive che precedono (o seguono) immediatamente l'intestazione di una *function* o l'inizio di uno *script*, possono essere visualizzate digitando il comando:  

```
> help nomefile
```

# Function

- Per chiamare una *function* basta digitare il nome del file (senza suffisso) in cui è memorizzata (tale nome può anche non coincidere con il nome della *function*).
- L'esecuzione di una *function* termina in corrispondenza del comando `return` o in corrispondenza della fine del file.
- In un *m – file* possono essere memorizzate diverse *function*:
  - ▶ l'unica accessibile dall'esterno è la prima;
  - ▶ le successive sono da considerare come sottoprocedure della prima.
- Il carattere `%` permette di inserire una riga di commenti all'interno di un *m – file*;
  - ▶ le righe di commento consecutive che precedono (o seguono) immediatamente l'intestazione di una *function* o l'inizio di uno *script*, possono essere visualizzate digitando il comando:
    - » `help nomefile`

## Function: esempio

```
function y = linspace(d1, d2, n)
%Linspace Linearly spaced vector.
% Linspace(X1, X2) generates a row vector of 100 linearly
% equally spaced points between X1 and X2.
%
% Linspace(X1, X2, N) generates N points between X1 and X2.
% For N < 2, Linspace returns X2.
%
% See also LOGSPACE, :.
% Copyright 1984-2002 The MathWorks, Inc.
% $Revision: 5.12 $ $Date: 2002/02/05 13:47:28 $

if nargin == 2
    n = 100;
end

y = [d1+(0:n-2)*(d2-d1)/(floor(n)-1) d2];
```

## Comando feval

```
[outputA1,outputA2,...] = feval(string,inputA1,inputA2,...)
```

- esegue la *function* il cui nome è contenuto in *string*;
- modificando il valore di *string*, la stessa istruzione può lanciare l'esecuzione di *function* diverse.

Esempio:

```
» feval('linspace',0,1,7)
```

```
ans =
```

```
0    0.1667    0.3333    0.5000    0.6667    0.8333    1.0000
```

```
» linspace(0,1,7)
```

```
ans =
```

```
0    0.1667    0.3333    0.5000    0.6667    0.8333    1.0000
```

```
» [0:1/6:1]
```

```
ans =
```

```
0    0.1667    0.3333    0.5000    0.6667    0.8333    1.0000
```

# Outline

- 1 Introduzione
- 2 Matrici
- 3 Area di Lavoro
- 4 Visualizzazione di variabili
- 5 Espressioni
- 6 Funzioni *in – line*
- 7 Istruzioni per il controllo del flusso di esecuzione
- 8 *m – files*
- 9 Grafica**

# Grafica

## Vedi

- `help graph2d,`
- `help graph3d,`
- `help specgraph,`
- `help graphics.`

- Grafica 2D
- Grafica 3D
- Animazioni

## Grafica 2D: comando `plot`

- `plot(x,y)`: disegna la spezzata che congiunge i punti  $(x(i),y(i))$  dei due vettori  $x$  e  $y$  di uguale lunghezza;
- `plot(x,y,s)`: analogo a `plot(x,y)` dove però con la stringa  $s$  è possibile specificare diversi simboli e colori da assegnare al grafico;
- `plot(x1,y1,s1,x2,y2,s2,x3,t3,s3,...)`: combina i grafici definiti dalle triple  $(x,y,s)$ .

	Colore	Marker		Linea	
b	blue	.	point	-	solid
g	green	o	circle	:	dotted
r	red	x	x-mark	-.	dashdot
c	cyan	+	plus	-	dashed
m	magenta	*	star	(none)	no line
y	yellow	s	square		
k	black	d	diamond		
		...	...		

Table: Specifica della stringa  $s$ .

## Grafica 2D: comando `plot`

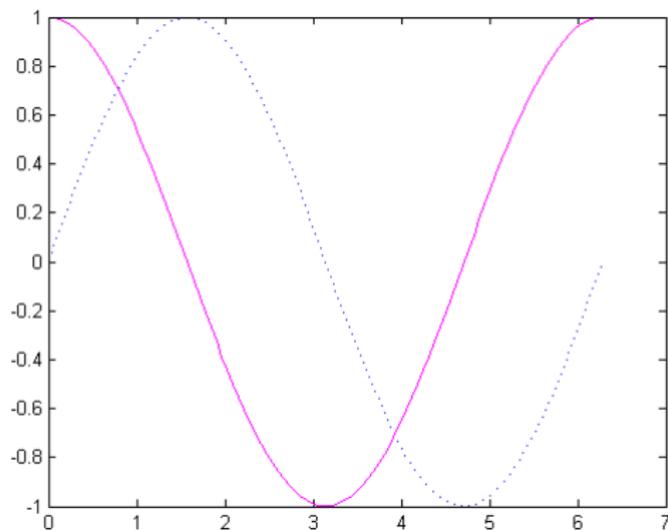
- `plot(x,y)`: disegna la spezzata che congiunge i punti  $(x(i),y(i))$  dei due vettori  $x$  e  $y$  di uguale lunghezza;
- `plot(x,y,s)`: analogo a `plot(x,y)` dove però con la stringa  $s$  è possibile specificare diversi simboli e colori da assegnare al grafico;
- `plot(x1,y1,s1,x2,y2,s2,x3,t3,s3,...)`: combina i grafici definiti dalle triple  $(x,y,s)$ .

	Colore	Marker		Linea	
b	blue	.	point	-	solid
g	green	o	circle	:	dotted
r	red	x	x-mark	-.	dashdot
c	cyan	+	plus	-	dashed
m	magenta	*	star	(none)	no line
y	yellow	s	square		
k	black	d	diamond		
		...	...		

Table: Specifica della stringa  $s$ .

## Grafica 2D: esempio

```
» x = linspace(0,2*pi); y1 = sin(x); y2 = cos(x);  
» plot(x,y1,'b:',x,y2,'m-');
```

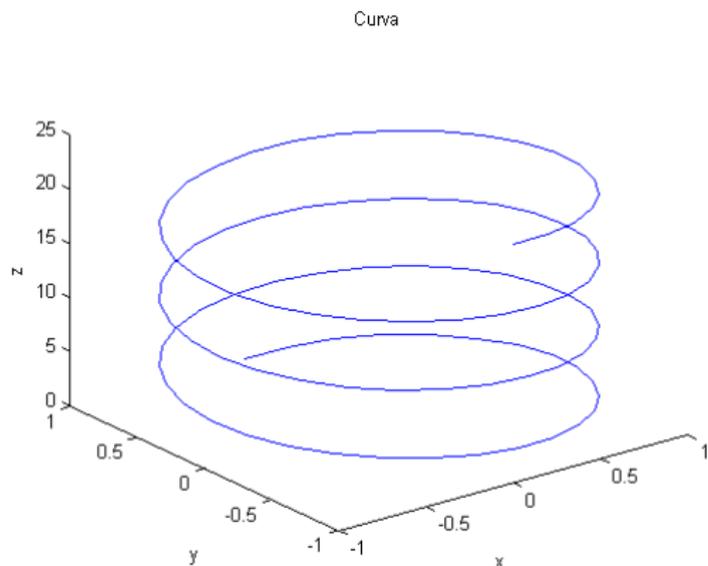


## Grafica 2D: altri comandi

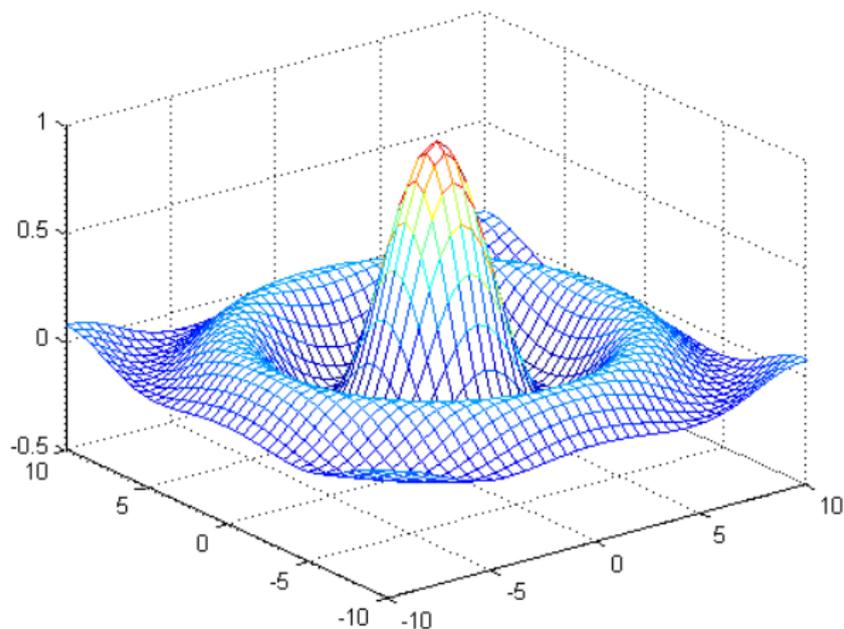
- `semilogx`: per creare grafici con scala logaritmica sull'asse  $x$ ;
- `semilogy`: per creare grafici con scala logaritmica sull'asse  $y$ ;
- `loglog`: per creare grafici con scala logaritmica su entrambi gli assi;
- `xlabel`, `ylabel`, `title`: consentono di assegnare rispettivamente un'etichetta all'asse  $x$ , all'asse  $y$  e un titolo al grafico;
- `axis`: permette di definire il *range* sui due assi;
- `hold on`, `hold off`: per mantenere attivo o disattivare un grafico;
- `legend`: per inserire la legenda delle curve;
- `print`: per salvare un grafico precedentemente creato;
- `text`, `gtext`: per inserire testo in una figura;
- `figure`: per creare una nuova finestra grafica;
- `shg`: per portare in evidenza la finestra grafica.

## Grafica 3D: esempio

- » `x = linspace(0,7*pi); y1 = sin(x); y2 = cos(x);`
- » `plot3(y1,y2,x);`
- » `xlabel('x'); ylabel('y'); zlabel('z'); title('Curva');`



## Grafica 3D: esempio di mesh



## Grafica 3D: esempio di surf

