

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Laboratorio 1: Introduzione al Matlab



Matlab = Matrix Laboratory

Utilizza come oggetti fondamentali da gestire le matrici

Matlab consente:

- L'accesso ad un ambiente di calcolo
- L'utilizzo di funzioni specializzate (librerie)
- Un ambiente di sviluppo integrato
 - Debug
 - Notebook
 - OOP



- Facilità d'uso:
 - Linguaggio interpretato – niente compilazione
 - Ha moltissime funzioni disponibili
 - E' possibile programmare funzioni ad hoc
 - Non ci si deve preoccupare di programmazione a basso livello
 - Niente dichiarazioni di tipi, allocazione di memoria, funzioni virtuali, polimorfismo etc...
 - Tuttavia, se si vuole, si possono usare classi e compilatori (ma noi non lo faremo in questo corso)
- Esportabilità
 - I programmi Matlab sono file di testo
 - Utilizzabili senza modifiche in tutti i sistemi operativi



Come si svolgono le esercitazioni



Aprite Matlab in una finestra, e tenete questa presentazione aperta in un'altra

Eseguite le operazioni indicate in questa presentazione.

Le operazioni da eseguire sono quelle negli screenshot di Matlab, oppure quelle evidenziate così:

Non esitate a fare delle prove cambiando i valori dei parametri usati.

The image shows a presentation slide on the left and a MATLAB Live Editor window on the right. The presentation slide, titled "Come si svolgono le esercitazioni", contains the following text:

Aprite Matlab in una finestra, e tenete questa presentazione aperta in un'altra
Eseguite le operazioni indicate in questa presentazione.
Le operazioni da eseguire sono quelle negli **screenshot** di Matlab, oppure quelle evidenziate in **blu**
Non esitate a fare delle prove cambiando i valori dei parametri usati.

The MATLAB Live Editor window displays the following content:

Parametri della sinusoide discreta
La sinusoide discreta in forma canonica è:
$$x(n) = A \cos(2\pi \nu n + \phi) = A \cos(\omega n + \phi)$$

Con
$$A \in \mathbb{R}_0^+; \nu \in [0, \frac{1}{2}]; \phi \in [-\pi, \pi]; \omega = 2\pi \nu$$

Utilizzare i comandi per scegliere i valori dei parametri e visualizzare il risultato.
Attenzione: in questo esempio ed in tutti i seguenti, il valore di ϕ è espresso come multiplo di π

Code block (with line 2 highlighted in blue):

```
1 %%
2 n=-30:30;
3 A = 0.73 ;
4 phi = pi * 0
5 nu = 0.207 ;
6
7 plot(n,A*cos(2*pi*nu*n+phi), 'k-o');
8 title(sprintf('cos( 2 \pi %4.3f n
9 axis([min(n) max(n) -1 1]); grid
```

Command Window:

```
fx K>>
```

The MATLAB interface also shows a plot of a discrete cosine wave and a Command History window with the following commands:

```
plot(abs(...
plot(abs(...
plot(real...
hold on; ...
hold on; ...
makeFigures
figure; p...
axis([0 1...
xlabel('t...
grid;
%-- 12/10...
```



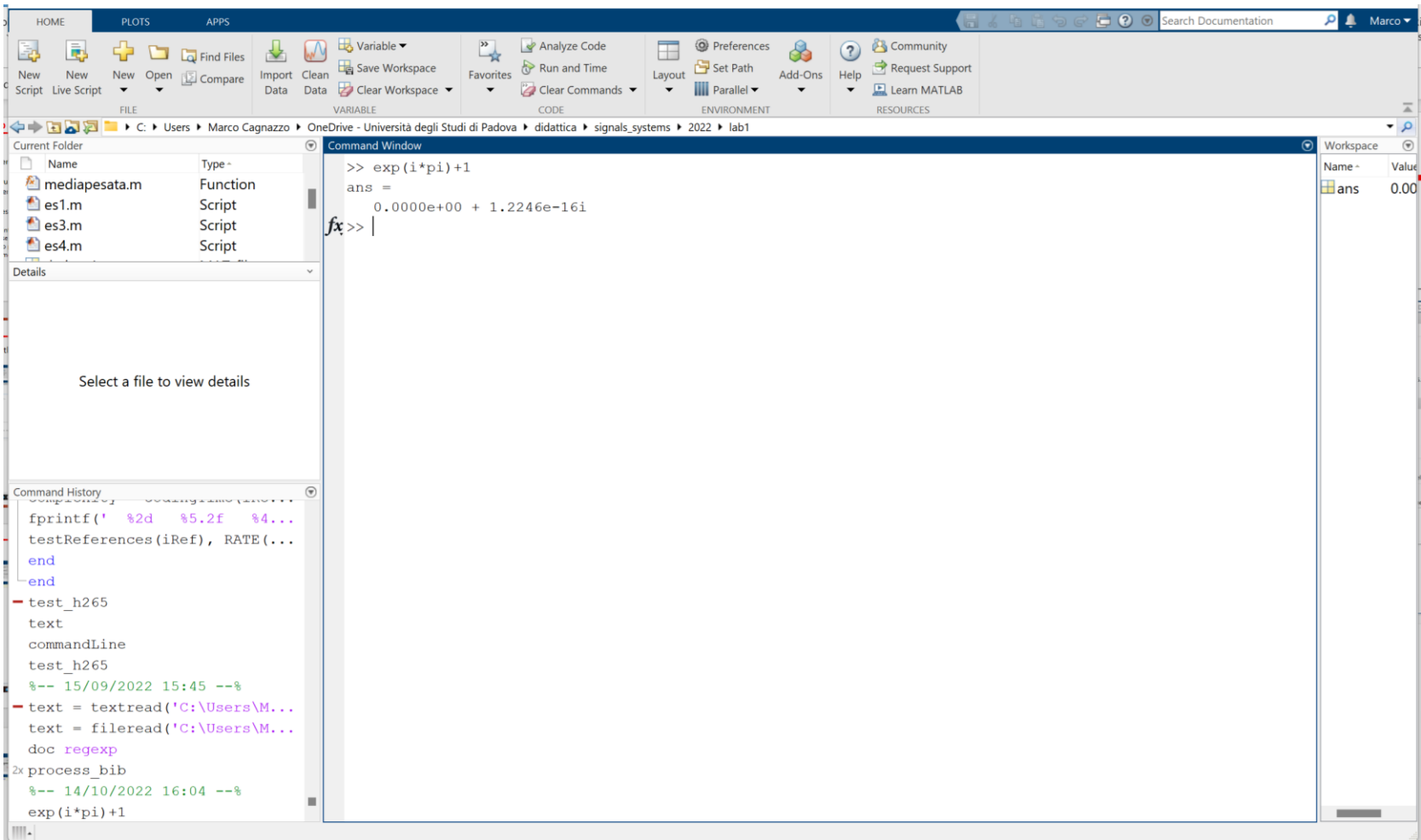
Come eseguire il codice Matlab

In Matlab ci sono vari modi per scrivere le istruzioni e farle eseguire:

- **Modalità interattiva:** ogni istruzione scritta viene valutata dall'interprete ed immediatamente elaborata nella Command Window (console)
- **Script:** in un file di testo con estensione “.m” si scrive una sequenza di istruzioni.
 - Lo script può essere eseguito interamente (F5)
 - Oppure suddiviso in "sezioni" eseguite individualmente (CTRL+ENTER)
 - Si può anche selezionare una o più istruzioni da uno script, una funzione o dallo storico, ed eseguirle premendo F9
- **Live Script:** in un file con estensione “.mlx”
 - Rispetto allo script, le istruzioni possono essere intervallate da un testo (per esempio, descrizione del codice e dei parametri, formule matematiche, ecc.)
 - Il live script può essere eseguito come uno script, oppure *interattivamente* utilizzando un'interfaccia grafica
 - **I live script sono l'analogo Matlab dei Notebook utilizzati in altri linguaggi/ambienti come in Python**



La finestra principale (Desktop) di Matlab è divisa in vari pannelli





Il Desktop di Matlab

Non tutti i pannelli sono sempre visibili: per attivarli, cliccare su Switch Window



The screenshot displays the MATLAB desktop environment. The top toolbar contains various icons for file operations, workspace management, and help. A red arrow points to the 'Switch Window' icon (a square with a circle) in the top right of the toolbar. The main workspace is divided into several panes:

- Current Folder:** Shows a list of files in the current directory, including 'mediapesata.m' (Function) and 'es1.m', 'es3.m', 'es4.m' (Scripts).
- Command Window:** Contains the command `>> exp(i*pi)+1` and the output `ans = 0.0000e+00 + 1.2246e-16i`.
- Workspace:** Shows the variable `ans` with a value of `0.00`.
- Command History:** Lists previous commands, including `fprintf`, `testReferences`, `test_h265`, `text`, `commandLine`, `test_h265`, `doc regexp`, `process_bib`, and `exp(i*pi)+1`.



Command Window

The screenshot displays the MATLAB environment. The Command Window is the central focus, showing the prompt `>>` followed by the command `exp(i*pi)+1`. The output is `ans = 0.0000e+00 + 1.2246e-16i`. A red arrow points from a text box to the Command Window. The Command History window at the bottom left shows a list of previous commands and their execution times.

Command History

```
fprintf(' %2d %5.2f %4...  
testReferences(iRef), RATE(...  
end  
end  
- test_h265  
text  
commandLine  
test_h265  
%-- 15/09/2022 15:45 --%  
- text = textread('C:\Users\M...  
text = fileread('C:\Users\M...  
doc regexp  
2x process_bib  
%-- 14/10/2022 16:04 --%  
exp(i*pi)+1
```

La *Command Window*: è il prompt interattivo di Matlab, dove possono essere eseguite operazioni e comandi, come il calcolo di $e^{i\pi} + 1$. È l'equivalente di una finestra di terminale o console. Si può usare anche per muoversi nel File System e per lanciare eseguibili esterni.



The screenshot shows the MATLAB environment. The top toolbar includes options like 'New Script', 'Open', 'Find Files', 'Save Workspace', and 'Run and Time'. The 'Current Folder' browser on the left shows a directory structure: `C:\Users\Marco Cagnazzo\OneDrive - Università degli Studi di Padova\didattica\signals_systems\2022\lab1`. It lists files: `mediaplata.m` (Function), `es1.m` (Script), `es3.m` (Script), and `es4.m` (Script). The 'Command Window' displays the command `>> exp(i*pi)+1` and its output: `ans = 0.0000e+00 + 1.2246e-16i`. The 'Command History' window at the bottom left shows a list of executed commands, including `fprintf`, `testReferences`, `test_h265`, `textread`, `fileread`, `doc regex`, `process_bib`, and `exp(i*pi)+1`.

Current Folder mostra il contenuto della cartella corrente, il cui path è in alto. Selezionando un file, i dettagli sono visibili qui



Navigazione tra cartelle

Usate il pannello Current Folder per navigare nel File System, per creare ed aprire file o cartelle

In alternativa, le stesse operazioni possono essere effettuate nella Command Window:

- Create una cartella di lavoro per il corso di segnali e sistemi, spostatevi in essa e create una sottocartella per la prima esercitazione.

Scaricare dal moodle tutto il materiale necessario per la prima esercitazione, estrarlo (unzip) se necessario e copiarlo nella sottocartella creata

Dovreste vedere questo: 

```
Command Window
>> mkdir segnali_sistemi
>> cd segnali_sistemi\
>> mkdir lab1
>> cd lab1
fx >> |
```



Current Folder	
Name	Type ^
esempio_strutture_pr..	Script
freq_num.m	Script
script_esempio.m	Script
sinusoidi_discrete.mlx	Live Script
Laboratorio_1.pdf	Documento Adob...



The screenshot shows the MATLAB interface with the Command Window open. The Command Window contains the following text:

```
>> exp(i*pi)+1
ans =
    0.0000e+00 + 1.2246e-16i
fx>> |
```

The Command History window at the bottom shows a list of commands. A red arrow points to the entry 'test_h265'.

```
fprintf(' %s\n', ...
testReference(iRef), RATE(...
end
end
- test_h265
text
commandLine
test_h265
%-- 15/09/2022 15:45 --%
- text = textread('C:\Users\M...
text = fileread('C:\Users\M...
doc regexp
2x process_bib
%-- 14/10/2022 16:04 --%
exp(i*pi)+1
```

La Command History mostra gli ultimi comandi del prompt. Ci sono molti modi per sfruttare la Command History: Nella Command Window, freccia in alto permette di percorrere la History; se si cominciano a battere dei caratteri, verranno richiamati i comandi della History che cominciano con quei caratteri. I comandi possono essere selezionati dalla Command History ed eseguiti premendo F9. Oppure possono essere copiati e poi incollati nel prompt oppure in un file di comandi (detto script).



The screenshot displays the MATLAB environment. The Command Window shows the following code and output:

```
>> exp(i*pi)+1
ans =
    0.0000e+00 + 1.2246e-16i
fx>> |
```

The Workspace panel on the right shows a table with the following data:

Name	Value
ans	0.0000e+00 + 1.2246e-16i

A red arrow points from the text box to the 'ans' variable in the Workspace panel.

È possibile visualizzare tutte le variabili memorizzate nello spazio di memoria corrente usando il pannello Workspace



Si consiglia di attivare il **pannello Editor** se non è già attivo: scrivere “edit” nella command window

The screenshot displays the MATLAB Editor interface. The top menu bar includes HOME, PLOTS, APPS, EDITOR, PUBLISH, and VIEW. The EDITOR tab is active, showing a toolbar with icons for New, Open, Save, Compare, Go To, Find, Refactor, Profiler, Run, Step, Stop, etc. The main workspace shows an 'Editor - untitled' window with a single line of code: '1 |'. The Command Window at the bottom shows the following commands and output: '>> exp(i*pi)+1', 'ans =', '0.0000e+00 + 1.2246e-16i', '>> edit', and 'fx>>'. The Command History window on the left shows a list of recent commands, including 'testReferences(iRef), RATE(...)', 'end', 'test_h265', 'text', 'commandLine', 'test_h265', 'text = textread('C:\Users\M...', 'text = fileread('C:\Users\M...', 'doc regexp', '2x process_bib', 'exp(i*pi)+1', and 'edit'.



Dal pannello editor si possono modificare e lanciare script e live script

The screenshot displays the MATLAB Editor interface. The top menu bar includes HOME, PLOTS, APPS, EDITOR, PUBLISH, and VIEW. The EDITOR tab is active, showing a toolbar with icons for file operations (New, Open, Save, Compare, Print), navigation (Go To, Find, Bookmark), code editing (Refactor, Analyze), and execution (Run, Step, Stop). The current folder is 'C:\Users\Marco Cagnazzo\OneDrive - Università degli Studi di Padova\didattica\signals_systems\2022\lab1'. The editor window shows a script named 'untitled' with the following code:

```
1 |
```

The Command Window shows the execution of the script:

```
>> exp(i*pi)+1  
ans =  
0.0000e+00 + 1.2246e-16i  
>> edit  
fx>>
```

The Command History window shows the following commands:

```
testReferences(iRef), RATE(...  
end  
end  
- test_h265  
text  
commandLine  
test_h265  
%-- 15/09/2022 15:45 --%  
- text = textread('C:\Users\M...  
text = fileread('C:\Users\M...  
doc regexp  
2x process_bib  
%-- 14/10/2022 16:04 --%  
exp(i*pi)+1  
edit
```

The Workspace window shows a variable 'ans' with a value of 0.00.



Apertura ed esecuzione di file

Doppio clic su un file per aprirlo

Per eseguirlo, click su Run, oppure F5, oppure scrivere il nome (senza estensione) nella CW



The screenshot shows the MATLAB R2022a interface. The top menu bar includes HOME, PLOTS, APPS, EDITOR, PUBLISH, and VIEW. The EDITOR tab is active, showing a toolbar with icons for New, Open, Save, Print, Go To, Find, Bookmark, Refactor, Analyze, Run, and Stop. The Run button is highlighted with a red arrow. The current folder is C:\Users\Marco Cagnazzo\OneDrive - Università degli Studi di Padova\didattica\corsi\signals_systems\23-24\labs\lab1. The editor window shows a script named freq_num.m with the following code:

```
1 % Grafici sinusoidi discrete: studio del valore della frequenza numerica
2
3 % Caso 1
4 A=1; phi = 0; nu = 0.03;
5 n = 0:40;
6 x = A*cos(2*pi*nu*n+phi);
7 % Creazione grafico
8 figure(1);
9 stem(n,x);
10 xlabel('n'); ylabel('x(n)');
11 title(sprintf('Freq: %4.3f',nu));
12
13 % Caso 2
14 A=1; phi = 0; nu = 0.97;
15 n = 0:40;
16 x = A*cos(2*pi*nu*n+phi);
17 % Creazione grafico
18 figure(2);
19 stem(n,x);
20 xlabel('n'); ylabel('x(n)');
21 title(sprintf('Freq: %4.3f',nu));
22
```

The Command Window shows the following commands:

```
fx >>
rmdir tmp
cd student
mkdir student
cd student
ls
clc
```

The workspace shows the following variables:

Name	Value
A	0.5000
n	1x61 double
...	0.0100

The status bar at the bottom shows Zoom: 100%, UTF-8, CRLF, script, Ln 7, Col 19.



**Eseguite queste istruzioni nel prompt (Command Window),
dovreste ottenere una schermata come questa →**

>> a=10 – crea la variabile a e le assegna il valore 10

>> b=[] – crea la variabile b e le assegna il vettore vuoto

>> size() – restituisce le dimensioni della variabile (in questo caso: 1x1 per lo scalare a e 0x0 per la variabile “vuota” b)

Notare che il tipo viene assegnato automaticamente al momento della creazione della variabile: per default sono double (guardare nel pannello Workspace)

```
Command Window

>> a=10

a =

    10

>> size(a)

ans =

     1     1

>> b=[]

b =

     []

>> size(b)

ans =

     0     0

fx >> |
```

Eseguite il codice mostrato



- + somma
- - sottrazione
- * moltiplicazione
- / divisione
- ^ potenza
- Notare che l'unità immaginaria si indica con **1i**

```
Command Window
>> 6*4
ans =
    24
>> 2^10
ans =
    1024
>> sqrt(-1)
ans =
    0.0000 + 1.0000i
>> 1i^2
ans =
    -1
fx >> |
```

Eseguite il codice mostrato



Il simbolo % introduce un commento: tutto quello che segue sulla stessa linea è ignorato

Il simbolo ; sopprime l'output e separa le righe delle matrici

Il simbolo . è usato per distinguere le operazioni elemento per elemento da quelle su matrici

Il simbolo , è usato per separare le colonne di vettori e matrici

Il simbolo ' è usato per il trasposto coniugato

Le parentesi **quadre** sono usate per **creare** vettori e matrici, le parentesi **tonde** per **accedere** agli elementi delle matrici e per chiamare le funzioni

Il primo elemento di un vettore **x** è accessibile con **x (1)**

```
>> x = [1, 2, 3]
x =
     1     2     3
>> y = [3, 2, 1]'
y =
     3
     2
     1
>> z=y+[1; 1; 1]
z =
     4
     3
     2
>> w=2*z-x';
>> w(1)
ans =
     7
```

Eseguite il codice mostrato



Creare un vettore

Nell'esempio si mostrano diversi modi di creare vettori riga e colonna

La virgola separa le colonne, il punto e virgola separa le righe

L'apostrofo opera il trasposto coniugato (sui reali è equivalente al trasposto)

```
Command Window
>> rowVector = [1 2 3]
rowVector =
     1     2     3
>> anotherRow=[1,2,3]
anotherRow =
     1     2     3
>> columnVector=[1;2;3]
columnVector =
     1
     2
     3
>> anotherColumn = rowVector'
anotherColumn =
     1
     2
     3
fx >> |
```

Eseguite il codice mostrato



Creare una matrice

Nell'esempio si mostrano diversi modi una matrice:

- elencandone i valori
- come trasposta di un'altra matrice
- con delle funzioni che creano matrici speciali

Command Window

```
>> matrix = [1,2,3; 4 5 6]
matrix =
     1     2     3
     4     5     6
>> matrix2 = matrix'
matrix2 =
     1     4
     2     5
     3     6
>> matrix3 = ones(3)
matrix3 =
     1     1     1
     1     1     1
     1     1     1
>> matrix4 = zeros(2)
matrix4 =
     0     0
     0     0
>> matrix5 = eye(4)
matrix5 =
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
fx >> |
```

Eseguite il codice mostrato



Operazioni su matrici

Nell'esempio si mostra il comando `whos` che mostra lo spazio di memoria corrente, ed una moltiplicazione matriciale

```
Command Window
>> a = [2,3 ; 4, 5];
>> b = [ 2, 5, 1; 4, 6, 7];
>> whos a b
  Name      Size      Bytes  Class  Attributes
  a         2x2         32  double
  b         2x3         48  double

>> a*b
ans =
    16    28    23
    28    50    39
fx >>
```

Eseguite il codice mostrato



Operazioni tra matrici

Nell'esempio si mostra la somma tra matrici e la differenza tra prodotto matriciale * e prodotto elemento per elemento .*
Si mostra anche la divisione elemento per elemento

Command Window

```
>> a = [2 3 5; 4 1 0; 1 4 -1];  
>> b = [1 0 0; 0 1 0; 0 0 1];  
>> c=a*b  
c =  
     2     3     5  
     4     1     0  
     1     4    -1  
>> d=a.*b  
d =  
     2     0     0  
     0     1     0  
     0     0    -1  
>> [3 4 5]./[6 1 .5]  
ans =  
     0.5000     4.0000    10.0000  
fx >>
```



Eseguite il codice mostrato



In Matlab si possono definire gli intervalli numerici con l'operatore due punti
Se non si specifica il passo, per default è uno

```
Command Window
>> start = 3; stop = 20;
>> a=start:stop
a =
  Columns 1 through 9
     3     4     5     6     7     8     9    10    11
  Columns 10 through 18
    12    13    14    15    16    17    18    19    20
>> step = 5;
>> b = start:step:stop
b =
     3     8    13    18
fx >> |
```

Eseguite il codice mostrato



Cose importanti da ricordare:

L'indicizzazione di vettori e matrici comincia da 1

L'accesso si esegue con le parentesi tonde

Quindi, il primo elemento di un vettore è:

$x(1)$

Mentre in C sarebbe $x[0]$

L'accesso a sottomatrici e sottovettori è molto flessibile.

Nell'esempio, la sottomatrice N è il blocco delle righe 1,2 e delle colonne 2,3

Usando il passo degli intervalli è possibile accedere, **sia in lettura sia in scrittura**, a righe o a colonne regolarmente spaziate

Nell'esempio, $1:2:end$ sono tutti gli indici dispari della matrice M

Eseguite il codice mostrato

Command Window

```
>> M = [ 1 2 3; 4 5 6; 7 8 9]
M =
     1     2     3
     4     5     6
     7     8     9
>> x=M(1,1)
x =
     1
>> N=M(1:2,2:3)
N =
     2     3
     5     6
>> % Sottomatrice: righe e colonne dispari
>> K = M(1:2:end, 1:2:end)
K =
     1     3
     7     9
>> %Interallacciamento di vettori
>> x1 = zeros(1,3)
x1 =
     0     0     0
>> x2 = ones(1,3)
x2 =
     1     1     1
>> y(1:2:5) = x1; y(2:2:6) = x2
y =
     0     1     0     1     0     1
```




Funzioni su vettori: la maggior parte delle funzioni di Matlab è definita su vettori:

Command Window

```
>> t = -5:5;  
>> x = sin(t)  
x =  
Columns 1 through 10  
    0.9589    0.7568   -0.1411   -0.9093   -0.8415         0    0.8415    0.9093    0.1411   -0.7568  
Column 11  
   -0.9589  
>> y = exp(t)  
y =  
Columns 1 through 10  
    0.0067    0.0183    0.0498    0.1353    0.3679    1.0000    2.7183    7.3891   20.0855   54.5982  
Column 11  
  148.4132
```

fx >>



Qualche funzione utile:

Command Window

```
>> x = rand % Valori casuali tra 0 e 1
x =
    0.9058
>> X = rand(2,3) % Matrice 2x3 di valori casuali
X =
    0.1270    0.6324    0.2785
    0.9134    0.0975    0.5469
>> mCol = max(X) % massimo colonna per colonna
mCol =
    0.9134    0.6324    0.5469
>> XV = X(:) % "vettorizzazione" di X
XV =
    0.1270
    0.9134
    0.6324
    0.0975
    0.2785
    0.5469
>> max(XV)
ans =
    0.9134
```

fx >> |

Command Window

```
>> X = rand(2,3) % Matrice 2x3 di valori casuali
X =
    0.9575    0.1576    0.9572
    0.9649    0.9706    0.4854
>> mean(X) % media per colonne
ans =
    0.9612    0.5641    0.7213
>> mean(X,2) % media per righe
ans =
    0.6908
    0.8070
>> mean(X(:)) % media globale
ans =
    0.7489
>> sum(X) % somma per colonne
ans =
    1.9224    1.1282    1.4425
>> X > 0.5 % operatore logico
ans =
    2x3 logical array
    1    0    1
    1    1    0
```

r



- Nella Command Window
 - » **who** – mostra tutte le variabili
 - » **whos** – mostra il nome delle variabili, il tipo e la dimensione
 - » **clear *var1 var2*...** - cancella le variabili riportate
 - » **clearvars** – cancella tutte le variabili
 - » **close all** – chiude tutte le figure
 - » **save** – salva le variabili in un file .mat
 - » **load** - carica le variabili da un file .mat



• Nella Command Window

>> **lookfor**
keyword

Ricerca la keyword in tutta la documentazione (può essere lento)

>> **help**

command_name

descrizione sintetica del comando o della libreria (toolbox)

>> **doc**

command_name

Versione navigabile e interattiva della documentazione

Command Window

```
>> lookfor Toeplitz
toeplitz           - Toeplitz matrix.
chow              - Chow matrix (singular Toeplitz lower Hessenberg matrix).
kms               - Kac-Murdock-Szego Toeplitz matrix.
parter            - Parter matrix (Toeplitz with singular values near pi).
prolate           - Prolate matrix (symmetric, ill-conditioned Toeplitz matrix).
toeppd            - Symmetric positive definite Toeplitz matrix.
toeppen           - Pentadiagonal Toeplitz matrix (sparse).
>> help toeplitz
toeplitz Toeplitz matrix.
toeplitz(C,R) is a non-symmetric Toeplitz matrix having C as its
first column and R as its first row.

toeplitz(R) is a symmetric Toeplitz matrix for real R.
For a complex vector R with a real first element, T = toeplitz(r)
returns the Hermitian Toeplitz matrix formed from R. When the
first element of R is not real, the resulting matrix is Hermitian
off the main diagonal, i.e.,  $T_{i,j} = \text{conj}(T_{j,i})$  for  $i \neq j$ .

Class support for inputs C,R:
    float: double, single
    integer: uint8, int8, uint16, int16, uint32, int32, uint64, int64

See also hankel.

Documentation for toeplitz
Other functions named toeplitz

>> doc toeplitz
fx>>
```



Risultato di

>> **doc toeplitz**

The screenshot shows the MATLAB Help Center interface. The browser address bar shows 'Help'. The page title is 'Help Center' with a search bar. The navigation tabs are 'Documentation', 'Examples', 'Functions', and 'Apps'. The left sidebar shows a 'CONTENTS' menu with links to 'Documentation Home', 'MATLAB', 'Mathematics', 'Elementary Math', and 'Constants and Test Matrices'. Under 'ON THIS PAGE', there are links for 'toeplitz', 'Syntax', 'Description', 'Examples', 'Input Arguments', 'More About', 'Extended Capabilities', 'Version History', and 'See Also'. The main content area is titled 'toeplitz' and describes it as a 'Toeplitz matrix'. It includes a 'Syntax' section with two forms: $T = \text{toeplitz}(c,r)$ and $T = \text{toeplitz}(r)$. The 'Description' section explains that $T = \text{toeplitz}(c,r)$ returns a nonsymmetric Toeplitz matrix with c as its first column and r as its first row, and that $T = \text{toeplitz}(r)$ returns a symmetric Toeplitz matrix where r defines the first row or column. The 'Examples' section includes a 'Create Symmetric Toeplitz Matrix' example with a code block:

```
r = [1 2 3];  
toeplitz(r)
```

 and the output:

```
ans = 3x3  
  
    1    2    3  
    2    1    2
```



>> **disp('text')** – mostra il testo tra apici (stringa)

Es: `disp('have a nice day')`

`disp(['I am ', num2str(18), ' year old'])`

N.B. **num2str** – converte il numero in una stringa



fprintf e sprintf

fprintf e sprintf sono molto più potenti di display

Esempi:

Command Window

```
>> number = 10; name = 'John';  
>> fprintf('%s has %d apples\nWriting Pi with 4 decimal digits gives %1.4f\n', name, number, pi)  
John has 10 apples  
Writing Pi with 4 decimal digits gives 3.1416
```

Il simbolo % introduce uno specificatore di formato

Il primo %s significa: stampa la prima variabile (`name`) come stringa

Il secondo %d: stampa la seconda (`number`) come numero intero

La terza %1.4f: stampa la terza variabile (`pi`) come numero decimale con 4 cifre dopo la virgola

Differenza tra **fprintf** e **sprintf**: il primo stampa a schermo (o su file) il secondo crea una stringa



Eseguite il codice mostrato



I grafici di Matlab sono mostrati in finestre dette figure

Il comando principale per creare un grafico è `plot`

`plot(t,x)` crea i punti di coordinate specificate dai parametri d'ingresso

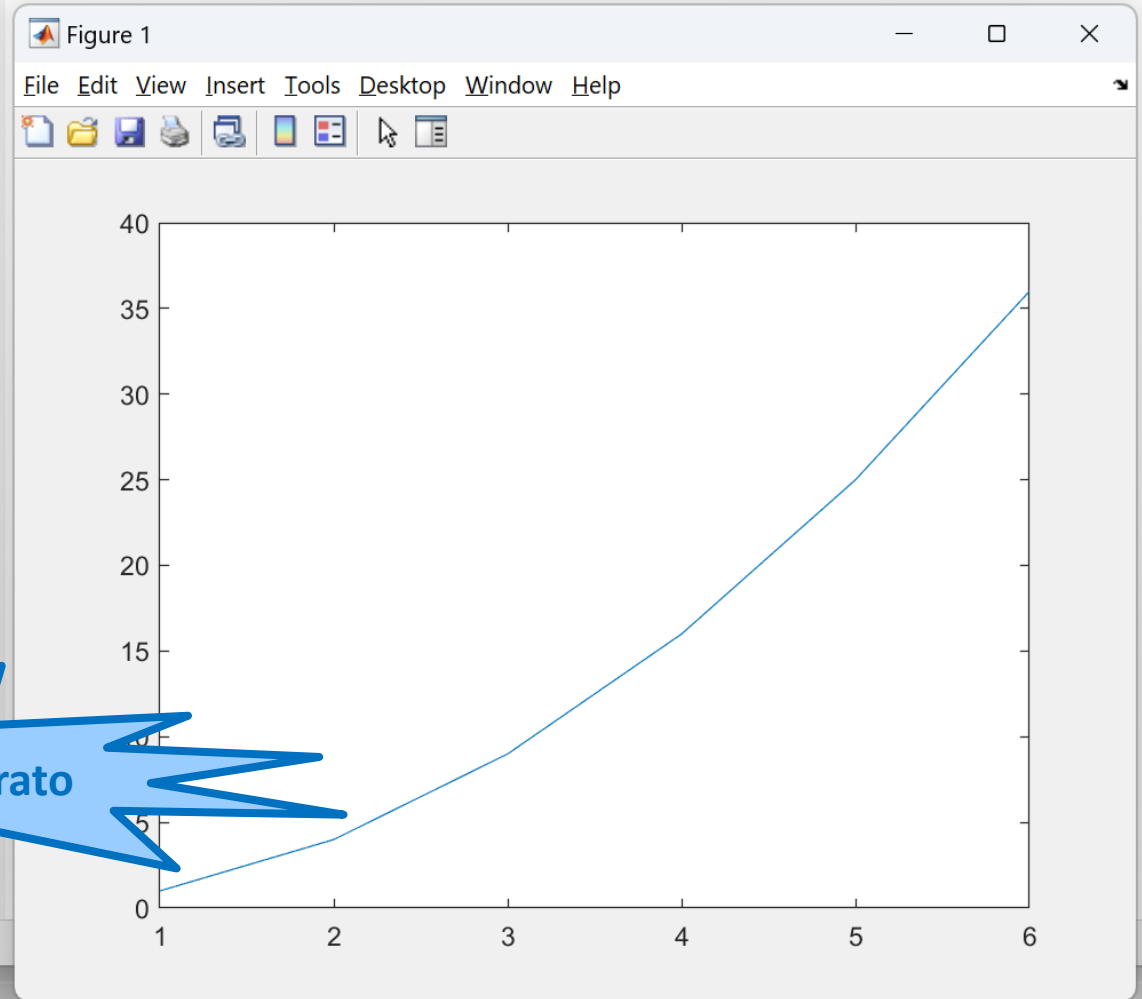
C'è grande flessibilità nella parametrizzazione del `plot`, vedere gli esempi seguenti

Domanda: perché si usa `.` e `^` e non semplicemente `^` ?

Eseguite il codice mostrato

Command Window

```
>> t = 1:6;  
>> x = t.^2; %Notare il punto!  
>> plot(t,x)  
fx>>
```





Dopo le ascisse t e le ordinate x , si possono specificare i parametri di formato in modo implicito o esplicito

La stringa `'ro--'` determina:

Colore rosso del plot

Marker dei punti (cerchio)

Linea tratteggiata

La coppia di parametri `'Linewidth', 2` specifica lo spessore della linea

Si nota che Matlab traccia il grafico unendo con segmenti di retta i punti specificati in t e x

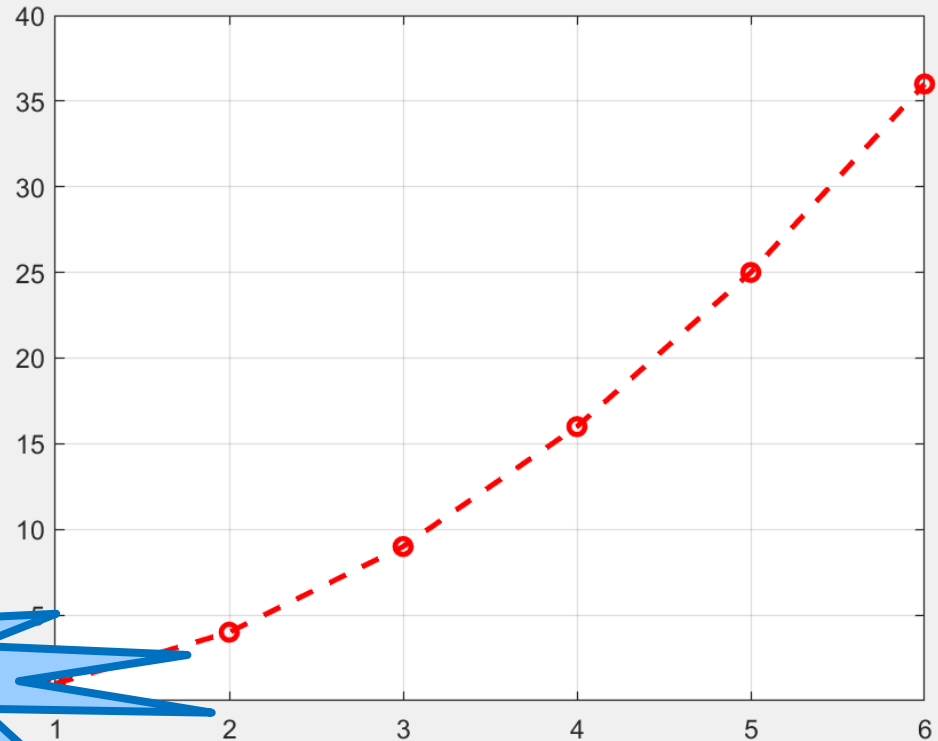
Per approfondire (in un secondo momento) riferirsi alla documentazione di plot con il comando `doc plot`

Command Window

```
>> t = 1:6;  
>> plot(t,x, 'ro--', 'Linewidth', 2);  
>> grid;  
fx >>
```

Figure 1

File Edit View Insert Tools Desktop Window Help



Eseguite il codice mostrato



Matlab lavora a tempo discreto

I grafici 'continui' sono in realtà sempre discreti con un'opportuna scelta del passo dei campioni

Per esempio, per disegnare una sinusoide smorzata con frequenza 1, scegliamo un passo di campionamento 1/100, che ci permette di disegnare 100 punti per periodo. Matlab disegnerà una linea retta tra punti consecutivi

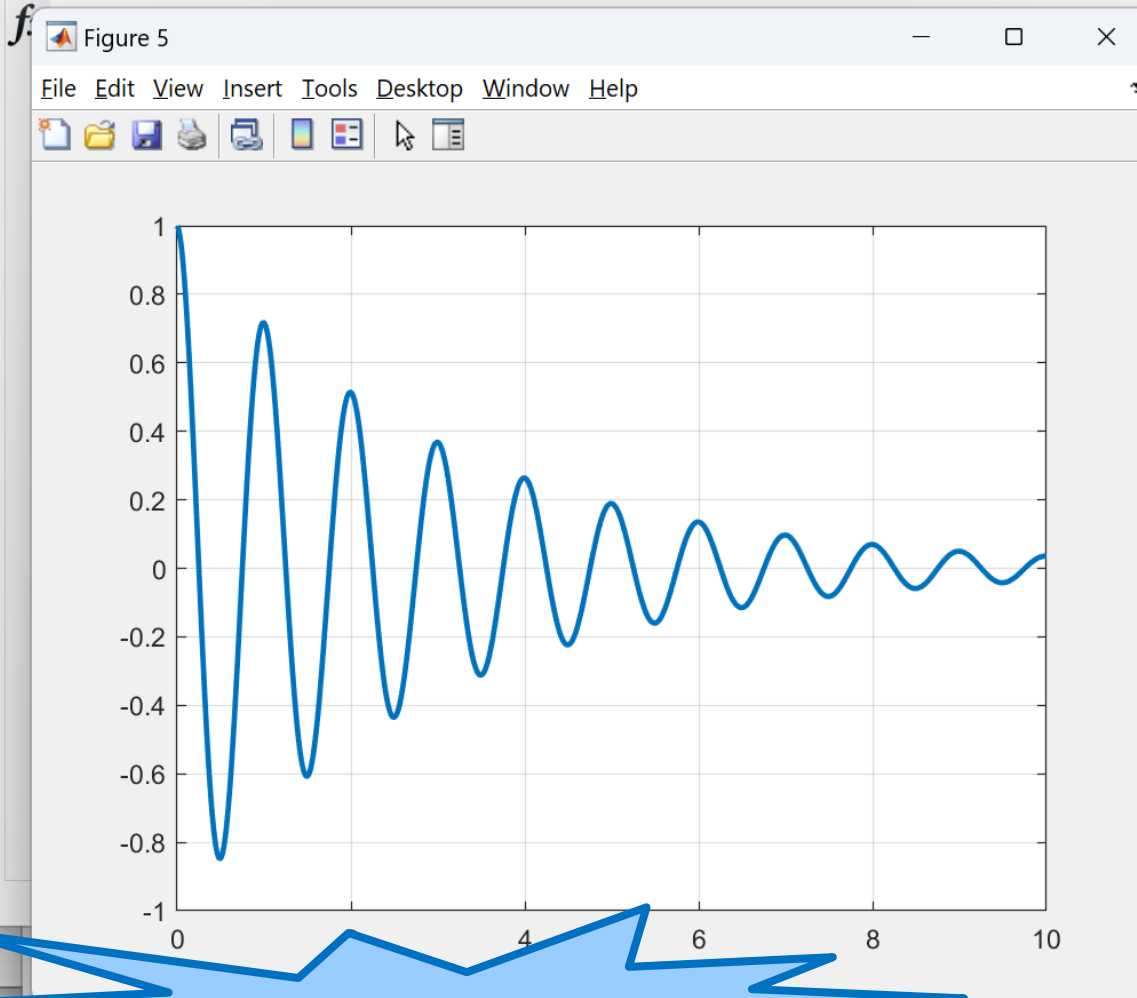
plot per default disegna nell'ultima figura attiva e, se non ce ne sono, ne crea una nuova

Per forzare la creazione di una nuova figura, usare il comando figure

Per forzare plot ad usare una figura specifica, per esempio la figura 2, usare figure(2)

Command Window

```
>> f = 1;  
>> t = 0:1e-2:10;  
>> T = 3;  
>> x = cos(2*pi*f*t) .* exp(-t/T);  
>> figure; plot(t,x,'LineWidth', 2); grid
```



Eseguite il codice mostrato



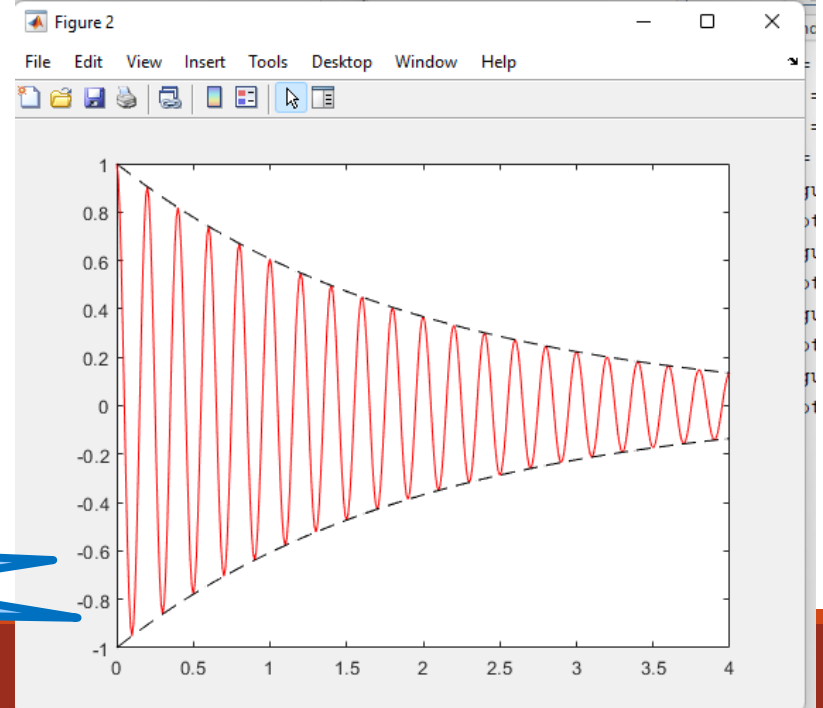
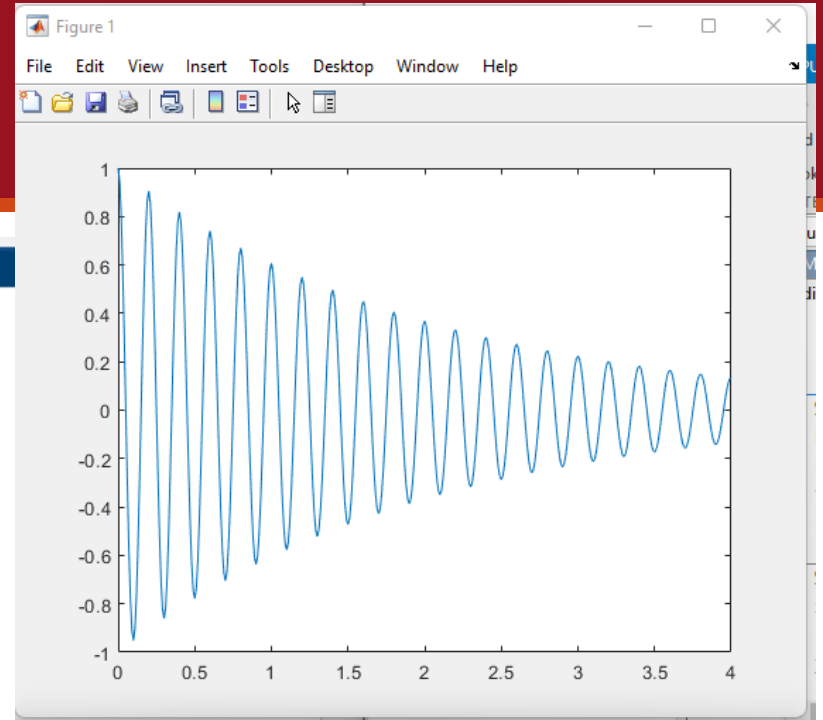
Command Window

```
>> t = 0:1e-2:4;  
>> v1 = (cos(2*pi*f*t));  
>> v2 = exp(-t/2);  
>> x = v1.*v2;  
>> figure(1);  
>> plot(t,x);  
>> figure(2);  
>> plot(t,x,'r',t,v2,'--k',t,-v2,'--k');  
fx >> |
```

Nella figura 2, si disegnano tre curve,
ognuna rappresentata da una terna:
ascissa, ordinata, formato

formato è una stringa, dove si indica
colore e/o formato della linea

Eseguite il codice mostrato





subplot (m, n, k) divide la figura in $m \times n$ sottofinestre e mette il plot seguente nella finestra numero k

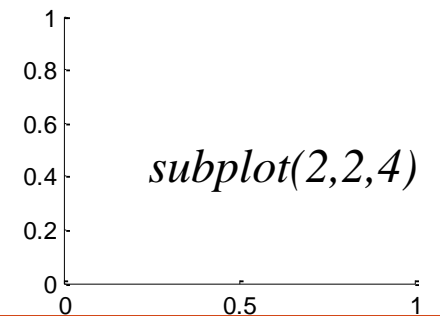
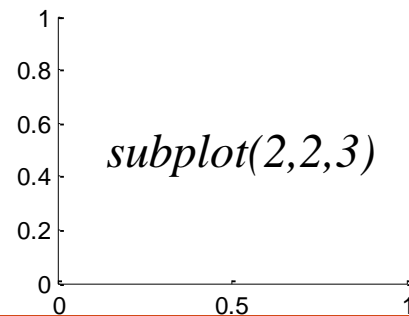
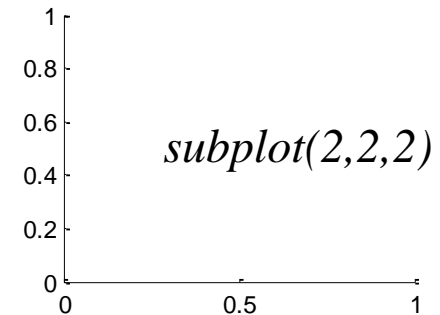
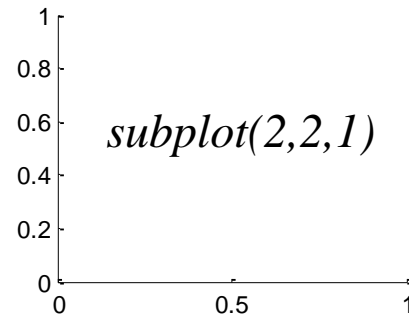
Esempio

subplot(2,2,1)

subplot(2,2,2)

subplot(2,2,3)

subplot(2,2,4)





In ogni figura si può mettere un plot diverso

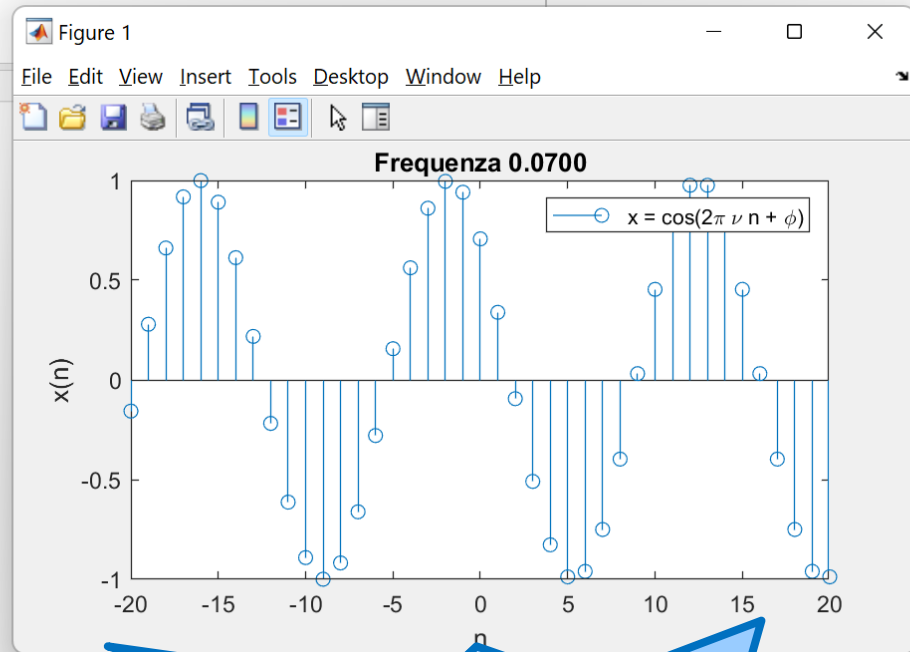
```
>> x = -pi:pi/100:pi;  
y = sin(x);  
subplot(2, 2, 1);  
plot(x, y, 'r')  
subplot(2, 2, 2);  
plot(x, y, 'b')  
subplot(2, 2, 3);  
plot(x, y, 'g')  
subplot(2, 2, 4);  
plot(x, y, 'k')
```



stem crea grafici adeguati ai segnali discreti

Command Window

```
>> n = -20:20;  
>> nu = 0.07;  
>> phi = pi/4;  
>> x = cos(2*pi*nu*n + phi);  
>> figure(1);  
>> stem(n,x);  
>> xlabel('n'); ylabel('x(n)');  
>> legend('x = cos(2\pi \nu n + \phi)');  
>> title(sprintf('Frequenza %3.4f',nu))  
fx>>
```



Cosa fanno i comandi xlabel, legend, title?

Qual è il periodo di questo segnale?

Ripetere i comandi, creando ogni volta una nuova figura, con $\nu = 1.07$, $\nu = 0.93$ e $\nu = -0.07$

Che cosa osservate?

Eseguite il codice mostrato



Le strutture di programmazione si usano all'interno di un file (funzione o script) e si usano un modo molto simile a tutti gli altri linguaggi di programmazione

A differenza di altri linguaggi, il carattere usato per la negazione è la **tilde**

```
Editor - C:\Users\Marco Cagnazzo\OneDrive - Università degli Studi di Padova\didattica\corsi\signals_systems\23-24\la
freq_num.m x freq_num.m x sinusoidi_discrete.mlx x script_esempio.m x sinusoidi_discrete.mlx x
1      %% Controllo dei parametri
2      t= 0:1e-2:10; f = -3; phi = 1; A=10;
3      x = A*cos(2*pi*f*t+phi);
4      if f<0
5          disp('La sinusoida NON è in forma canonica');
6      elseif f==0
7          disp('Segnale costante')
8      else
9          disp('Sinusoide in forma canonica');
10     end
11     if phi ~= 0
12         disp('Fase non nulla');
13     end
14     %% Stampa i quadrati
15
16     for number = 1:3
17         fprintf('Il quadrato di %d è %d\n',number, number*number);
18     end
19
20     %% ricerca fattori primi di un numero
21     N = 371;
22     number = 2;
23     while number<=N
24         if ~mod(N,number)
25             fprintf('%d diviso %d fa %d\n',N,number, N/number);
26             N= N/number;
27         else
28             number = number +1;
29         end
30     end
```



- Script e funzioni: .m
- Live script: .mlx
- Dati: .mat
- Figure: .fig



- Gli **script** sono file di testo che contengono una sequenza di comandi MATLAB
- Non accettano in ingresso degli argomenti, né restituiscono variabili in uscita. Lavorano sui dati del workspace (condivisione spazio di memoria)
- Sono comodi quando si devono ripetere lunghe sequenze di istruzioni, cambiando ogni volta qualche parametro
- Uno script, **una volta salvato**, può essere eseguito in vari modi:
 - Dalla command window, scrivere il nome del file (senza .m!)
 - Attenzione, il file deve essere nel Path oppure nella cartella corrente!
 - Dalla finestra dell'editor, premere F5
 - Dalla finestra dell'editor, cliccare su Run



Esercizio

Aprire nell'editor il file
script_esempio.m

Eseguire il file dalla
command window o con
F5

Cambiare i parametri della
sinusoide ed osservare il
risultato

Osservare che le variabili
create appaiono nel
Workspace corrente

```
Editor - C:\Users\Marco Cagnazzo\OneDrive - Università degli Studi di Padova\didattica\corsi\signals_systems\23-24\labs\lab1\script_esempio.m
freq_num.m x freq_num.m x sinusoidi_discrete.mlx x script_esempio.m x sinusoidi_discrete.mlx x +
1 % Un semplice script per la creazione di grafici
2 clear variables
3 close all
4 % Parametri dell'asse dei tempi
5 T0 = 0; T1 = 10; step = 1e-2;
6 t = T0:step:T1;
7
8 % Parametri della sinusoide
9 A = 2; phi = pi/4;
10 freq = 0.325;
11
12 x = A * cos(2*pi*freq*t +phi);
13 figure(1);
14 plot(t,x,'r--','LineWidth',2);
15 grid; title(sprintf('Sinusoide a frequenza %4.3f',freq));
16
```

Spesso negli script
si cancella il
workspace e si
chiudono le figure



Esempio di script con sezioni

Aprire il file `freq_num.m` (doppio click in Current Folder o scrivere `edit freq_num` nella command window)

Eseguire le sezioni del file come indicato; modificare i parametri ed eseguire di nuovo

```
Editor - C:\Users\Marco Cagnazzo\OneDrive - Università degli Studi di Padova\didattica\corsi\signals_systems\23-24\labs\lab1\student\freq
freq_num.m x sinusoidi_discrete.mlx x freq_num.m * x +
1 %% Grafici sinusoidi discrete: studio del valore della frequenza numerica
2
3 %% Caso 1
4 A=1; phi = 0; nu = 0.03;
5 n = 0:40;
6 x = A*cos(2*pi*nu*n+phi);
7 % Creazione grafico
8 close(1); % Chiude la figura 1 se esiste già
9 figure(1);
10 stem(n,x);
11 xlabel('n'); ylabel('x(n)');
12 title(sprintf('Freq: %4.3f',nu));
13
14 %% Caso 2
15 A=1; phi = 0; nu = 0.97;
16 n = 0:40;
17 x = A*cos(2*pi*nu*n+phi);
18 % Creazione grafico
19 close (2); % Come sopra
20 figure(2);
21 stem(n,x);
22 xlabel('n'); ylabel('x(n)');
23 title(sprintf('Freq: %4.3f',nu));
24
```

Le sezioni sono delimitate da righe che cominciano con %% (due simboli di percento seguiti da almeno uno spazio)
Nell'Editor, premendo CTRL+ENTER si esegue unicamente la Sezione corrente
Con CTRL+SHIFT+ENTER si esegue la sezione corrente e si passa alla seguente
Nei due casi precedenti il file NON viene salvato: utile per delle prove!
Infine, con F5 o Run, il file viene salvato ed eseguito interamente

Esercizio

Aprire nell'editor il file `script_esempio.m`

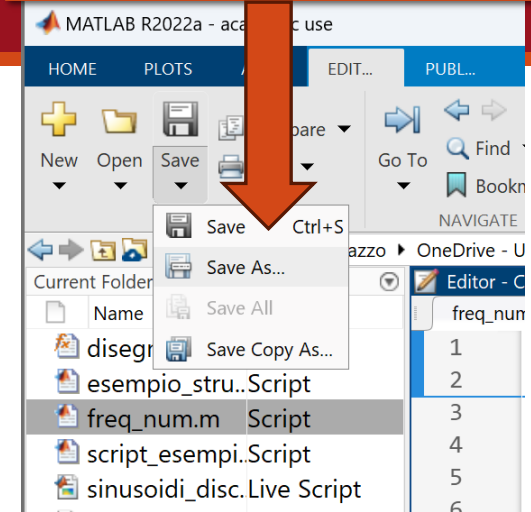
Eseguire **Save as** e salvarlo come `disegnaCos.m`

Trasformarlo in funzione cambiando la prima riga del file **ed eliminando le assegnazioni dei parametri d'ingresso, come mostrato**

Lanciare la funzione dalla command window, esempio:

```
>> x=disegnaCos(0:1e-2:5, 1, 0.2, 0);
```

Dove trovare Save as...



```
Editor - C:\Users\Marco Cagnazzo\OneDrive - Università degli Studi di Padova\didattica\corsi\signals_systems\23-24\labs\lab1\student
freq_num.m x sinusoidi_discrete.mlx * x esempio_strutture_progr.m x disegnaCos.m x script_esempio.m x +
1 function x = disegnaCos(t,A,freq,phi)
2 % disegnaCos calcola e disegna una senoide in forma canonica
3 %x = disegnaCos(t,A,freq,phi) calcola la senoide nei punti indicati dal
4 %parametro t. Gli altri parametri sono quelli della forma canonica:
5 %ampiezza, frequenza, fase iniziale
6
7 x = A * cos(2*pi*freq*t +phi);
8 figure(1);
9 plot(t,x,'r--','LineWidth',2);
10 grid; title(sprintf('Senoide a frequenza %4.3f',freq));
```



Esempio di funzione

Osservazioni importanti

Un file 'funzione' deve cominciare con la keyword **function**

Dopo **function** c'è la lista dei parametri di uscita tra parentesi quadre, opzionali nel caso di un solo parametro

Segue il nome della funzione: usare lo stesso nome del file

Infine la lista di parametri d'ingresso tra parentesi tonde e separati dalla virgola

```
Editor: C:\Users\Marco C... \OneDrive\Università degli Studi di Padova\didattica\segnais_systems\23-24\labs\lab1\student...
freq_num... x sinusoide discrete... esempio_st... disegnaCos.m x script_esempio.m x +
1 function x = disegnaCos(t,A,freq,phi)
2 % disegnaCos calcola e disegna una senoide in forma canonica
3 %x = disegnaCos(t,A,freq,phi) calcola la senoide nei punti indicati dal
4 %parametro t. Gli altri parametri sono
5 %ampiezza, frequenza, fase iniziale
6
7 x = A * cos(2*pi*freq*t +phi);
8 figure(1);
9 plot(t,x,'r--','LineWidth',2);
10 grid; title(sprintf('Sinusoide a frequenza %4.3f',freq));
```

Il primo blocco di commenti è il testo che appare se si digita **help disegnaCos** nella command window (provare!)

Lo spazio di memoria delle funzioni non è condiviso con la command window
All'uscita dalla funzione, tutte le variabili interne non saranno più accessibili



I **live script** sono la versione Matlab dei notebook

Si tratta di script con caratteristiche aggiuntive:

- Si può inframezzare del testo (CTRL+E) e delle formule (CTRL+SHIFT+E) al codice Matlab
- Si possono inserire slider, menù a tendina, ecc. per assegnare i valore dei parametri
- L'esecuzione è interattiva
- Si possono facilmente generare dei report



Esercizio

1. Aprire il live script `sinusoidi_discrete.mlx`

The screenshot shows a MATLAB Live Editor window titled "Live Editor - C:\Users\Marco Cagnazzo\OneDrive - Università degli Studi di Padova\didattica\corsi\signals_systems\23-24\labs\lab1\sinusoidi_discrete.mlx". The window has two tabs: "freq_num.m" and "sinusoidi_discrete.mlx".

1. Parametri della sinusoide discreta
La sinusoide discreta in forma canonica è:
$$x(n) = A \cos(2\pi \nu n + \phi) = A \cos(\omega n + \phi)$$

Con
$$A \in \mathbb{R}_0^+; \nu \in \left[0, \frac{1}{2}\right]; \phi \in [-\pi, \pi]; \omega = 2\pi\nu \in [0, \pi]$$

Utilizzare i comandi per scegliere i valori dei parametri e visualizzare il risultato.
Attenzione: in questo esempio ed in tutti i seguenti, Il valore di ϕ è espresso come multiplo di π

```
1 n=-30:30;  
2 A = 0.63 _____ ;  
3 phi = pi* 0 _____ ;  
4 nu = 0.207 _____ ;  
5  
6 plot(n,A*cos(2*pi*nu*n+phi), 'k-o')  
7 title(sprintf('cos( 2 \pi %4.3f n %+3.2f \pi)',nu,phi/pi));  
8 axis([min(n) max(n) -1 1]); grid
```

2. Verifica dell'intervallo rilevante dei valori di frequenza numerica
Se nella forma canonica prendiamo ν al di fuori di $\left[0, \frac{1}{2}\right]$, otteniamo una sinusoide in forma non canonica. Tuttavia, esisterà sempre una sinusoide in forma canonica equivalente. Nel seguito, usare lo slider per variare la frequenza numerica al di fuori dell'intervallo $\left[0, \frac{1}{2}\right]$

```
9 %  
10 n=-30:30;  
11 A = 1;  
12 phi = pi/4;
```

The right side of the window displays two plots. The top plot is titled "cos(2 π 0.207 n +0.00 π)" and shows a discrete cosine wave with a period of approximately 10 samples. The bottom plot is titled "cos(2 π ·(0.930) n +0.25 π)" and shows a discrete cosine wave with a period of approximately 10 samples, but with a phase shift of 0.25π.

2. Leggere le diverse sezioni ed eseguirne il codice cliccando prima sulla sezione e poi su Run Section

The screenshot shows the MATLAB Live Editor interface. The script is divided into two sections:

1. Parametri della sinusoide discreta

La sinusoide discreta in forma canonica è:

$$x(n) = A \cos(2\pi \nu n + \phi) = A \cos(\omega n + \phi)$$

Con

$$A \in \mathbb{R}_0^+; \nu \in \left[0, \frac{1}{2}\right]; \phi \in [-\pi, \pi]; \omega = 2\pi\nu \in [0, \pi]$$

Utilizzare i comandi per scegliere i valori dei parametri e visualizzare il risultato.

Attenzione: in questo esempio ed in tutti i seguenti, il valore di ϕ è espresso come multiplo di π

```
1 close all
2 n=-30:30;
3 A = 0.35 ;
4 phi = pi* -0.58 ;
5 nu = 0.065 ;
6
7 plot(n,A*cos(2*pi*nu*n+phi), 'k-o')
8 title(sprintf('cos( 2 \pi %4.3f n %3.2f \pi)',nu,phi/pi)
9 axis([min(n) max(n) -1 1]); grid
```

2. Verifica dell'intervallo rilevante dei valori di frequenza numerica

Se nella forma canonica prendiamo ν al di fuori di $\left[0, \frac{1}{2}\right]$, otteniamo

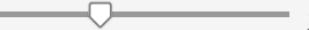
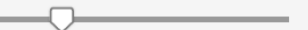
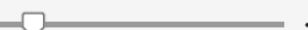
The execution results show two plots:

The first plot shows the discrete cosine wave $\cos(2\pi \cdot 0.065 n)$ for n from -30 to 30. The plot displays a periodic waveform with a period of approximately 15 samples.

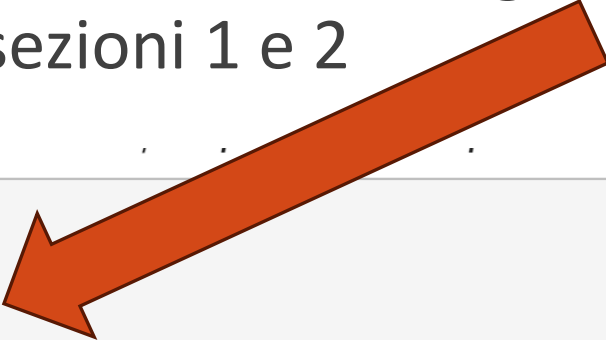
The second plot shows the discrete cosine wave $\cos(2\pi \cdot (1.040) n + 0.25\pi)$ for n from -30 to 30. The plot displays a periodic waveform with a period of approximately 10 samples, shifted by 0.25π .

An orange arrow points from the text "Run Section" to the "Run Section" button in the MATLAB Live Editor toolbar.

3. Modificare i parametri della sinusoida tramite gli slider e verificare quanto richiesto nelle sezioni 1 e 2

```
close all
n=-30:30;
A = 0.35  ;
phi = pi* -0.58  ;
nu = 0.065  ;

plot(n,A*cos(2*pi*nu*n+phi), 'k-o')
title(sprintf('cos( 2 \\pi %4.3f n %+3.2f \\pi)',nu,phi/pi));
axis([min(n) max(n) -1 1]); grid
```





Esempio di Live script

4. Eseguire l'esercizio della sezione 3 e verificare sperimentalmente l'identità tra i due segnali:

