



SIS

Scuola Interdipartimentale
delle Scienze, dell'Ingegneria
e della Salute



Laurea Magistrale in STN

Applicazioni di Calcolo Scientifico e Laboratorio di ACS (12 cfu)

prof. Mariarosaria Rizzardi

Centro Direzionale di Napoli – Isola C4

studio: n. 423 – Lato Nord, 4° piano

Tel.: 081 547 6545

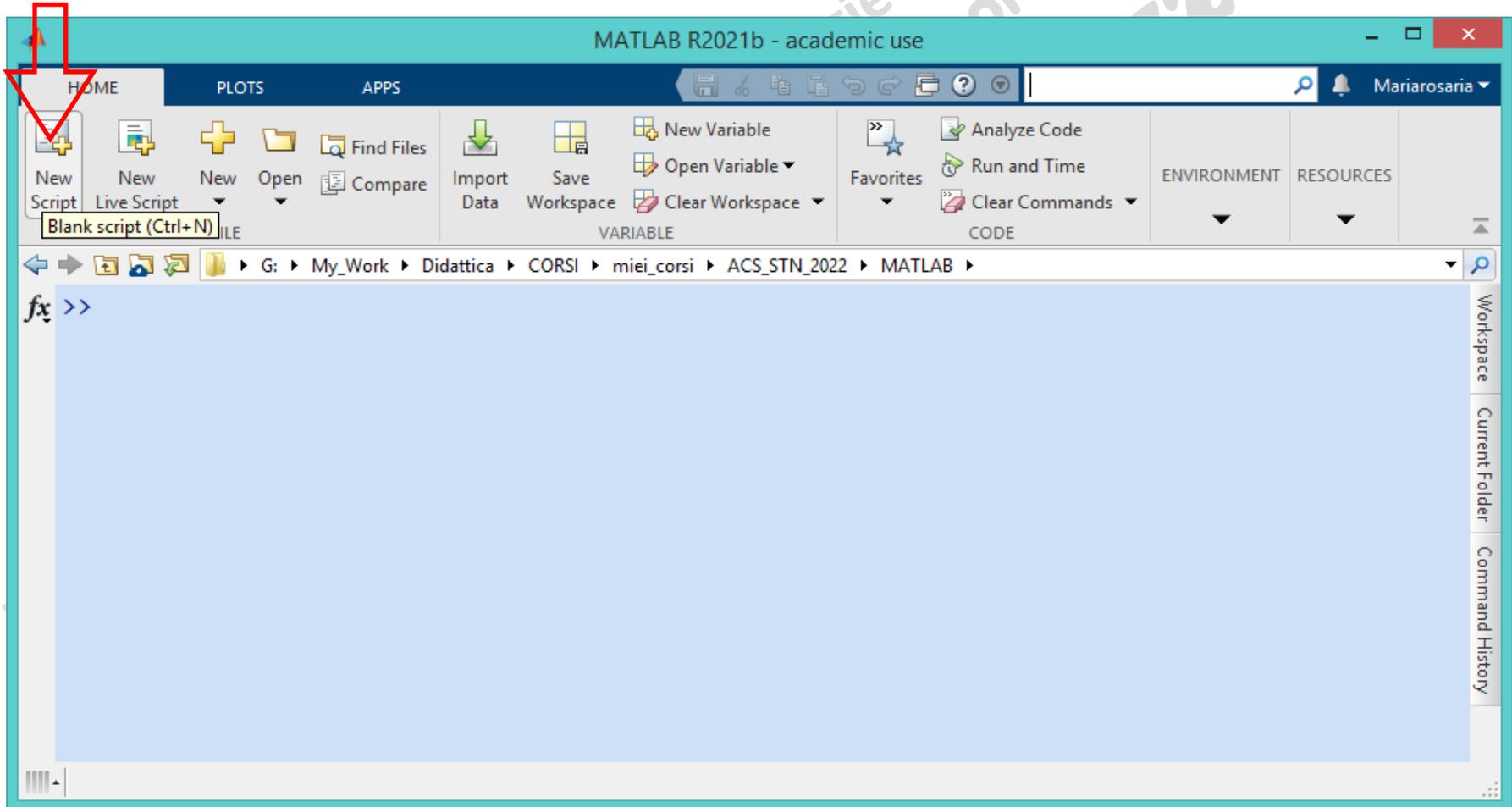
email: mariarosaria.rizzardi@uniparthenope.it

Argomenti trattati

- **L'Editor di MATLAB (m file).**
- **Il Live Editor di MATLAB (mlx file).**
- **Uso avanzato del Live Editor di MATLAB.**

Si voglia scrivere un programma MATLAB (m file) per visualizzare il grafico di una funzione $y=f(x)$ in un intervallo.

Nella Command Window
selezionare:





grafico_funzione.m nome del file. Se c'è un * il file non è salvato

```

1 % grafico_funzione.m
2 clear; clc
3
4 %% input data ← il %% individua una sezione (section)
5 N=input('Enter the number of points: N = ');
6 a=input('Enter the left endpoint of the interval: a = ');
7 b=input('Enter the right endpoint of the interval: b = ');
8
9 %%
10 fstr=input('Enter the function to be evaluated: f(t) = ', 's');
11 pf=['@(t)' fstr]; aggiunge @(t) in testa alla stringa (anonymous function)
12 pf=str2func(pf); % construct function handle from character vector
13 % converte la stringa in anonymous function
14 %% display
15 x=linspace(a,b,N)';
16 y=pf(x);
17
18 figure(1); clf
19 plot(x,y)
20 set(gca, 'FontSize', 14)
21 xlabel('x'); ylabel('y')
22 title(['Plotted function: f(t) = ' fstr], 'FontWeight', 'normal')
23

```

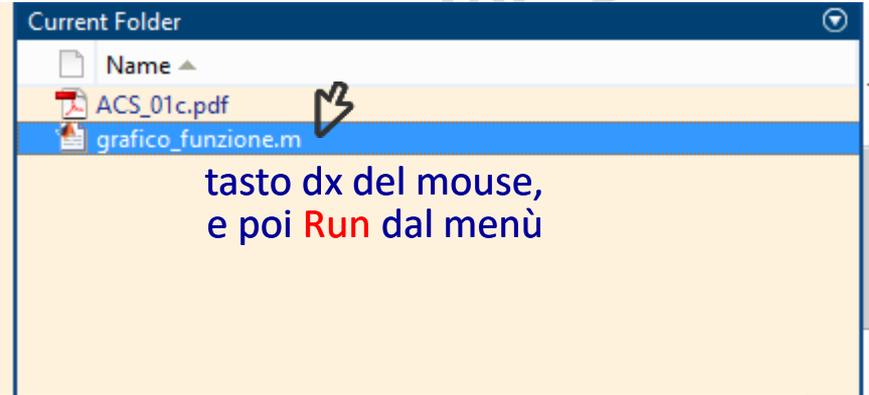
input di stringa

per lanciare l'esecuzione dello script nella Command Window:

1) Nell'Editor



2) In Current Folder



3) Nella Command Window

```
>> grafico_funzione
```

Esecuzione nella Command Window

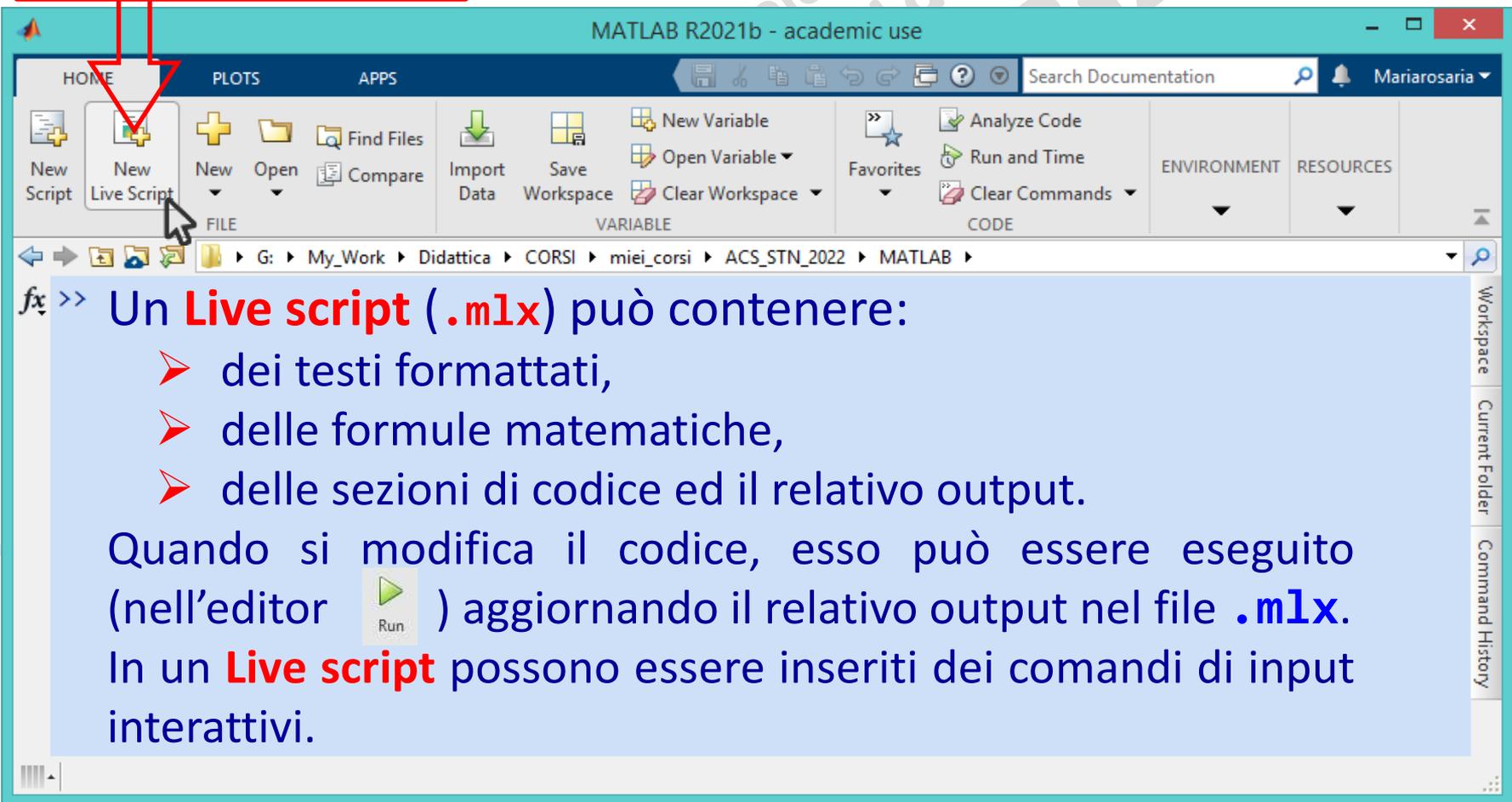
The image shows the MATLAB R2022b interface. The Command Window contains the following text:

```
Enter the number of points: N = 101  
Enter the left endpoint of the interval: a = -2*pi  
Enter the right endpoint of the interval: b = +2*pi  
Enter the function to be evaluated: f(t) = cos(t) + 0.5*sin(10*t)  
fx >> |
```

A red arrow points from the word 'stringa' (in a red box) to the function definition line. Below the Command Window is a plot titled 'Plotted function: f(t) = cos(t) + 0.5*sin(10*t)'. The plot shows a complex periodic wave with a period of approximately 2 units on the x-axis. The x-axis ranges from -6 to 6, and the y-axis ranges from -1.5 to 1.5.

Si voglia scrivere un programma MATLAB per visualizzare il grafico di una funzione $y=f(x)$ in un intervallo, mediante il **Live Editor** con l'aggiunta di **comandi interattivi**.

Nella Command Window selezionare:



The screenshot shows the MATLAB R2021b interface. The ribbon at the top has tabs for HOME, PLOTS, and APPS. Under the HOME tab, the 'New Live Script' button is highlighted with a red arrow. Below the ribbon, the Command Window contains the following text:

```
fx >> Un Live script (.mlx) può contenere:
```

- dei testi formattati,
- delle formule matematiche,
- delle sezioni di codice ed il relativo output.

Quando si modifica il codice, esso può essere eseguito (nell'editor ) aggiornando il relativo output nel file **.mlx**. In un **Live script** possono essere inseriti dei comandi di input interattivi.

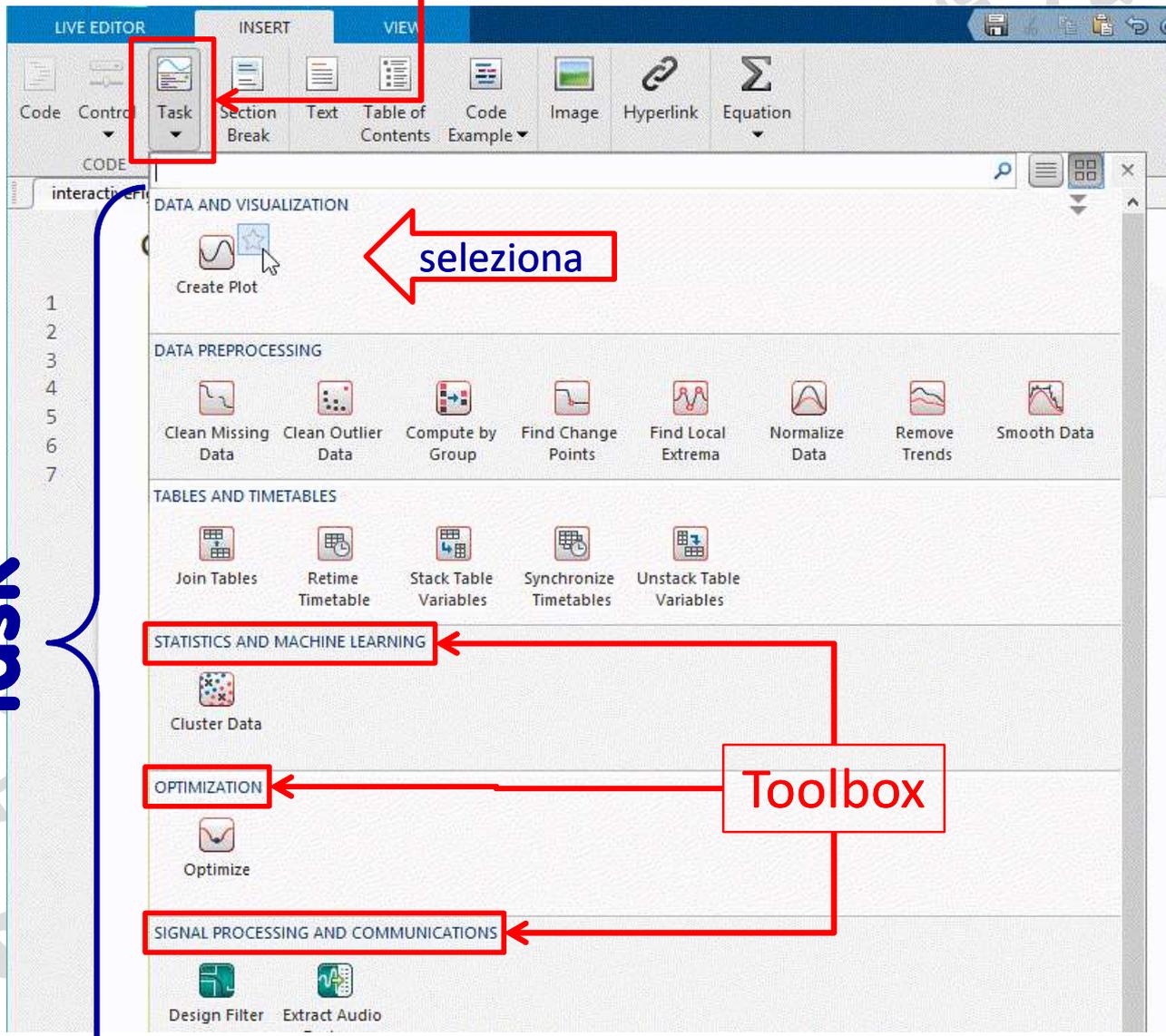
Nel Live Editor

The screenshot displays the MATLAB Live Editor interface. At the top, there are tabs for LIVE EDITOR, INSERT, FIGURE, and VIEW. Below these are various toolbars including FILE, NAVIGATE, TEXT, CODE, and SECTION. A red arrow points from the word "testo" to the text area containing the title "Grafico di una funzione f(x) per x in [a,b]". Another red arrow points from the word "codice" to the code editor area containing the following MATLAB code:

```
1 N = 201;  
2 a = -2*pi;  
3 b = +2*pi;  
4 pf=@(t) cos(t) + 0.5*sin(10*t);  
5 x = linspace(a,b,N)';  
6 y = pf(x);  
7 plot(x,y)
```

A third red arrow points from the word "output" to the plot area, which shows a blue line graph of a complex periodic function. The x-axis ranges from -8 to 8, and the y-axis ranges from -1.5 to 1.5. The plot shows a high-frequency oscillation superimposed on a lower-frequency wave. At the bottom of the interface, there is a status bar with information such as Zoom: 110%, UTF-8, LF, script, Ln 7, and Col 10.

Con il **Live Editor**, invece di scrivere il codice per visualizzare il grafico, si può inserire un **Task**



Task

seleziona

Toolbox

... altri Task

Laurea Magistrale in Applicazioni di Progettazione

```
4 pf=@(t) cos(t) + 0.5*sin(10*t);
5 x = linspace(a,b,N)';
6 y = pf(x);
7 %plot(x,y
```

Task: Create Plot

Create Plot

h3 = plot of x and y

Select visualization

Search for a visualization Filter by Category All

Creates a 2-D line plot of the data in Y versus the corresponding values in X

Select data

X x

Y y

Select optional visualization parameters

select

plot Add

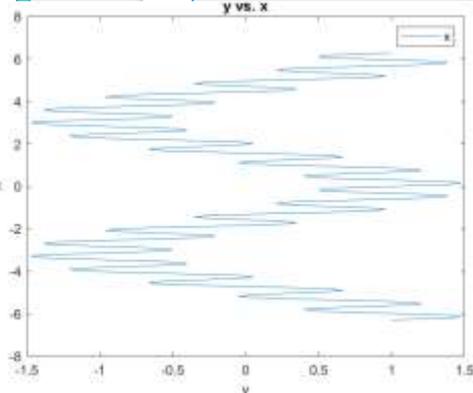
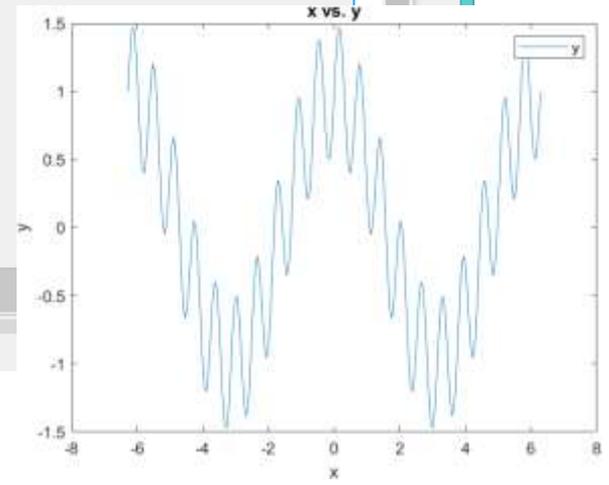
← compilare i campi

→ output

↑ visualizza/nasconde codice

← output

← compilare

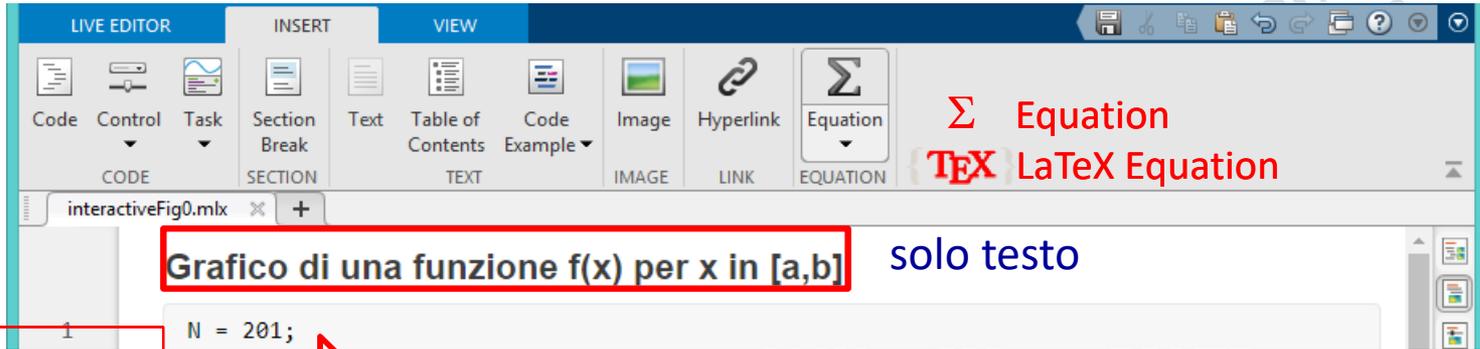


Select data

X y

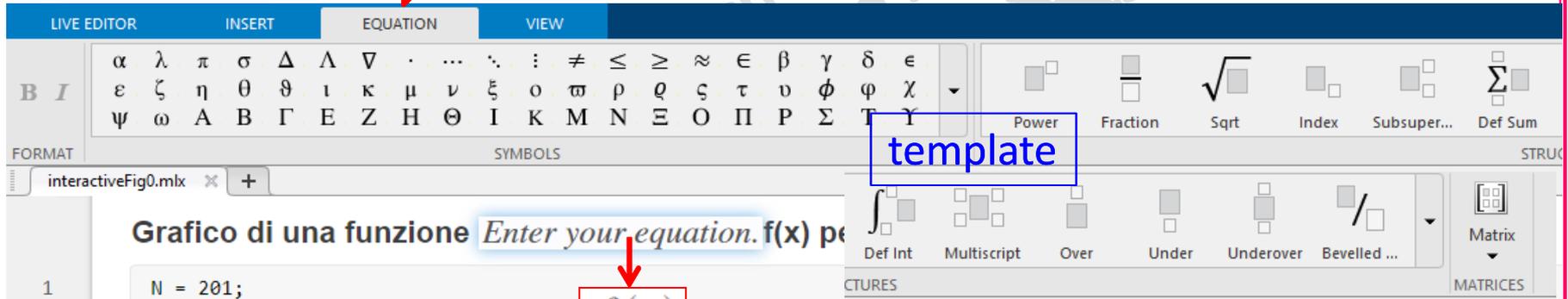
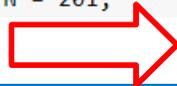
Y x

Si possono inserire **formule matematiche** nel testo



1

Equation

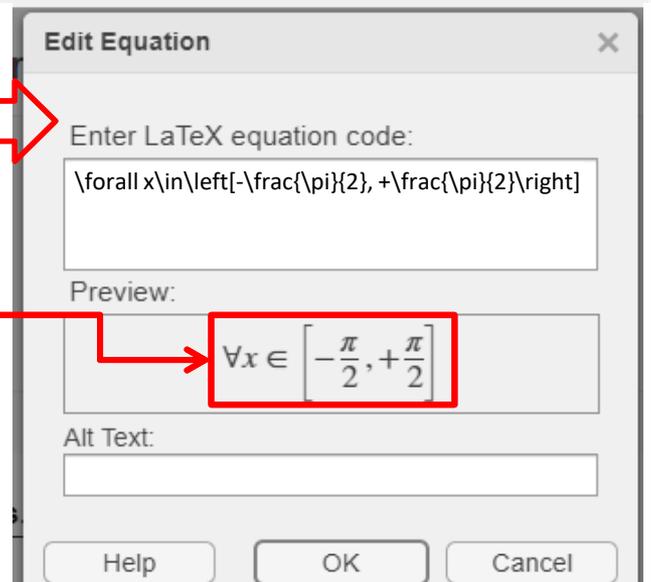


template

$f(x)$

2

LaTeX



risultato

$\forall x \in \left[-\frac{\pi}{2}, +\frac{\pi}{2}\right]$

LaTeX* (pronunciato **latek**) è un sistema software per il publishing di documenti. È un *linguaggio di markup* **WYSIWYM** (What You See Is What You Mean) a differenza di **Word** che è **WYSIWYG** (What You See Is What You Get).

* **LaTeX**: abbreviazione di Lamport's TeX, dal nome di Leslie Lamport che aggiunse una libreria di macro a **TeX** (dal greco $\tau\epsilon\chi$ =tecnologia e arte), il sistema di composizione tipografica creato da Donald Knuth nel 1978.

download: [LaTeX_GreekLetters_SpecialCharacters_MATLAB.pdf](#)

LATEX

Chi vuole usare **LaTeX** per scrivere documenti può scaricarlo il manuale da:

[http://users.softlab.ntua.gr/~sivann/books/LaTeX-User's Guide and Reference Manual-lamport94.pdf](http://users.softlab.ntua.gr/~sivann/books/LaTeX-User's-Guide-and-Reference-Manual-lamport94.pdf)

ed installare, oltre alla **libreria LaTeX** (**MiKTeX** - <https://miktex.org/download>), un **LaTeX editor** che ne semplifichi l'uso. Per un elenco vedere:

<https://beebom.com/best-latex-editors/>

Tra questi, per gli **utenti Windows**, molto utile è **TeXnicCenter**:

<https://www.texniccenter.org/download/>

Per gli **utenti Linux**, come LaTeX editor c'è **Kile**:

<https://kile.sourceforge.io/download.php>

Si voglia scrivere un programma MATLAB per visualizzare il grafico di una funzione $y=f(x)$ in un intervallo, mediante il **Live Editor** con l'aggiunta di **comandi interattivi (Control)**.

The screenshot shows the MATLAB Live Editor interface. The 'LIVE EDITOR' tab is active, and the 'CONTROL' button in the toolbar is highlighted with a red box. Below the toolbar, the code editor displays the following code:

```
1 N = |
2 a = -2*pi;
3 b = +2*pi;
4 pf=@(t) cos(t) + 0.5*sin(10*t);
```

A red arrow points to the cursor position after 'N =', with the text: **insert Control: numeric slider in questa posizione**. To the right, a numeric slider for 'N' is shown, with a value of 101. Below the slider, a configuration dialog box is open, showing the following settings:

- LABEL**: Enter text to display when code is hidden. Label: value of N.
- VALUES**: Enter value or select workspace variable. Min: 1, Max: 201, Step: 1.
- DEFAULTS**: Enter or select from workspace. Default value: 101.
- EXECUTION**: Run On: Value changed, Run: Current section.

At the bottom of the code editor, the text **compilare i campi** is written in blue.

LIVE EDITOR **INSERT** **VIEW**

Code Control Task Section Break Text Table of Contents Code Example Image Hyperlink Equation

CODE SECTION TEXT IMAGE LINK EQUATION

interactiveFig0.mlx * +

Grafico di una funzione $f(x) \forall x \in [-2\pi, +2\pi]$

1 N = 133 numeric slider

2 N = 133 visualizza il valore di N

3 a = -2*pi;

4 b = +2*pi;

5 pf=@(t) cos(t) + 0.5*sin(10*t);

6 x = linspace(a,b,N)';

7 y = pf(x);

8 %plot(x,y)

aggiungendo qui ';' non viene visualizzato il valore di N

insert Control: drop down

la funzione **eval()** valuta come codice MATLAB il suo argomento stringa; **str2func()** converte la stringa argomento in *function handle*

insert Control:
edit field in questa posizione

1 N = 64

2 a = a -pi

▼ LABEL

Enter text to display when code is hidden

Label Enter a:

▼ TYPE

Data type MATLAB code

▼ DEFAULTS

Enter or select from workspace

Default value -pi

▼ EXECUTION

Run Current section

compilare
i campi

▼ LABEL

Enter text to display when code is hidden

Label pf:

▼ ITEMS

Enter labels or values to add to drop down

Item labels @(t)cos(t)+0.5*sin(10*t)
 @(t)cos(t/2)

Item values "@(t)cos(t)+0.5*sin(10*t)"
 "@(t)cos(t/2)"

Select a variable to add its content to drop down

Variable select

▼ DEFAULTS

Default item @(t)cos(t)+0.5*si...

▼ EXECUTION

Run Current section

1

2

3

4

5

6

7

6

7

8

y

0

-0.5

Argomenti trattati

- **Il linguaggio di programmazione di MATLAB:**
 - ❖ **Costrutti di controllo (c. di selezione e c. ripetitivi).**
 - ❖ **Script e function.**
 - ❖ **Funzioni ricorsive.**
 - ❖ **Alcune funzioni predefinite.**
 - ❖ **Array multidimensionali.**

Linguaggio di programmazione di MATLAB

I costrutti di controllo

In un programma, le istruzioni normalmente vengono eseguite **in sequenza**, cioè una dopo l'altra nell'ordine in cui compaiono. Per modificare l'ordine di esecuzione delle istruzioni, tutti i linguaggi di programmazione mettono a disposizione le **istruzioni di controllo** del flusso delle operazioni (o **costrutti di controllo**).

**costrutti
di controllo**

**costrutti
condizionali**

- if-then
- if-then-else
- if-then-elseif
- switch-case
- try-catch

**costrutti
ripetitivi**

- for-do
- while-do

Sintassi: **if-then**

Esempio

```
...  
if espressione-logica  
    istruzioni-1  
end  
...
```

Se **espressione-logica** è **vera** allora vengono eseguite prima le **istruzioni-1** e poi quelle che seguono **end**; se **espressione-logica** è **falsa** si saltano le **istruzioni-1**.

```
A=input('valore intero ');  
if rem(A,2) == 0  
    disp('A è pari')  
    b=A/2;  
end  
valore intero 6  
A è pari
```

Sintassi:

if-then-else

Esempio

```
if espressione-logica
    istruzioni-1
else
    istruzioni-2
end
```

Se **espressione-logica** è **vera** allora si eseguono le **istruzioni-1**, altrimenti (se è **falsa**) si eseguono le **istruzioni-2**.

```
A=input('valore intero ');
if rem(A,2) == 0
    disp('A è pari')
    b=A/2;
else
    disp('A è dispari')
end
valore intero 5
A è dispari
```

Sintassi:

`if-then-elseif`

Esempio

```
if   espress-logica1
      istruzioni-1
elseif espress-logica2
      istruzioni-2
else
      istruzioni-3
end
```

È un **if** a più vie. Conviene, se possibile, usare al suo posto uno **switch-case**.

```
% a*x^2+b*x+c=0
a=1; b=2; c=-1;
d=b*b-4*a*c; discriminante dell'equazione
if   d == 0
      disp('r. coincidenti')
elseif d > 0
      disp('r. distinte')
else
      disp('r. complesse')
end
r. distinte
```

Sintassi:

switch-case

Esempio

```
switch espressione
    (numerica o stringa)
case valore-1
    istruzioni-1
case valore-2
    istruzioni-2
...
otherwise
    istruzioni-n
end
```

```
a=1; b=2; c=-1;
d=b*b-4*a*c;
switch sign(d)
    case 0
        disp('r. coincidenti')
    case 1
        disp('r. distinte')
    otherwise
        disp('r. complesse')
end
r. distinte
```

Sintassi:

try-catch

Esempio

```
try
  :
  istruzioni-1
  :
catch
  :
  istruzioni-2
  :
end
```

istruzioni eseguite in caso
di errore

```
clear; x=3;
try
  y=x
catch
  disp('variabile non definita')
  disp(lasterr)
end
y =
    3
try
  y=A
catch
  disp('variabile non definita')
  disp(lasterr)
end
variabile non definita
Unrecognized function or
variable 'A'.
```

la variabile **x** è definita:
nessun errore

la variabile **A** non è
definita: errore

Sintassi: for-do-loop

Esempio

```
for indice=iniz : passo : fine  
    istruzioni-1  
end
```

Le **istruzioni-1** sono ripetute per ciascun valore assunto dalla variabile *indice*, che parte dal valore *iniz* e si incrementa ogni volta di *passo* finchè non supera *fine*. Se *iniz* > *fine* le istruzioni non sono eseguite.

Se *passo* < 0, le istruzioni sono eseguite solo se *iniz* ≥ *fine*, e ripetute fintantoché sia *indice* ≥ *fine*.

Se *passo*=1, si può scrivere

```
for indice=iniz : fine
```

```
m=5; n=5; format rat  
for i = 1:m  
    for j = 1:n  
        A(i,j)=1/(i+j-1);  
    end  
    disp([i A(i,:)])  
    for j = n:-2:1 passo < 0  
        A(i,j)=1/A(i,j);  
    end  
end  
fprintf('\n\n'); disp(A)
```

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

Sintassi: `while-do`

Esempio

```
while espressione-logica  
    istruzioni-1  
end
```

Le **istruzioni-1** sono ripetute fintantochè **espressione-logica** è vera; altrimenti si esce dal ciclo.

Se inizialmente **espressione-logica** è vera, in **istruzioni-1** il suo valore deve essere variato, altrimenti si provoca un cosiddetto **loop** infinito.

```
n=1;  
while    prod(1: n) < 1e100  
        n=n+1;  
end  
disp(n)  
    70
```

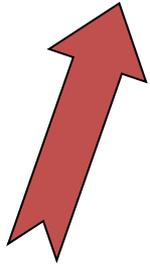
Tipi di m-file

<i>Script file</i>	<i>Function file</i>
Non hanno parametri di input e di output (<u>non</u> hanno <i>riga di intestazione</i>)	Possono avere parametri di input e di output (<u>hanno</u> <i>riga di intestazione</i>)
Operano sulle variabili nel workspace (globali) (<i>quelle già attive o ne creano di nuove</i>).	Operano per default su variabili interne (<i>locali al function workspace</i>) oppure sui parametri. Altrimenti usare global
Sono interpretati ad ogni esecuzione.	Per default sono tradotti all'atto della prima chiamata.
Nome del file qualsiasi.	Nome della function uguale al nome del file.

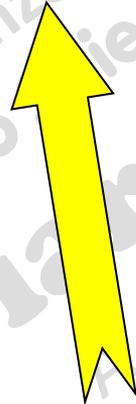
Sintassi:

function-file

function [out₁, out₂, ...] = nome_fun(in₁, in₂, ...)



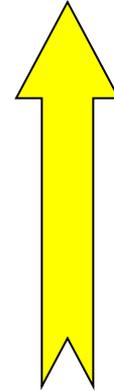
parola chiave



parametri di
uscita



nome function



parametri di
ingresso

Esempio di function file

Function file per il **fattoriale** (iterativo) di un intero

file **fattoriale.m**

```
function nFatt = fattoriale(n)
    nFatt=1;
    if n <= 1
        return
    end
    for k=2:n
        nFatt=nFatt*k;
    end
end
```

Nella Command Window

```
>> nFatt = fattoriale(0)
nFatt =
     1
>> nFatt = fattoriale(2)
nFatt =
     2
>> nFatt = fattoriale(3)
nFatt =
     6
>> nFatt = fattoriale(4)
nFatt =
    24
>> nFatt = fattoriale(5)
nFatt =
   120
>> nFatt = fattoriale(10)
nFatt =
 362880
>> factorial(10)
ans =
 362880
```

MATLAB ha moltissime
function predefinite

Il linguaggio di programmazione di **MATLAB** consente nei *function-files* la **ricorsione** (cioè la possibilità per una funzione di chiamare sé stessa).

Con la ricorsione si risolve un problema mediante la soluzione dello stesso tipo di problema, ma più “piccolo”, fino ad arrivare al **caso base** di cui è nota la soluzione.

Function file per il **fattoriale ricorsivo**

caso base per interrompere le chiamate ricorsive

```
file fat_ricors.m  
  
function nFatt = fat_ricors(n)  
    if n <= 2  
        nFatt=n;  
    else  
        nFatt=n*fat_ricors(n-1);  
    end
```

chiamata ricorsiva su un problema di dimensione minore

Functions Predefinite

MATLAB mette a disposizione moltissime funzioni ed i suoi toolbox ne aggiungono altre, specifiche per particolari applicazioni. In tal modo gli algoritmi possono essere tradotti nel linguaggio **MATLAB** in modo più sintetico.

Esempio: fattoriale di n

Function file per il **fattoriale**
iterativo mediante ciclo **while**

```
function nfat = fatt(n)
    k=n; nfat=n;
    while k>2
        k=k-1;
        nfat=nfat*k;
    end
end
```

Funzioni MATLAB

nfat=prod(2:n)

nfat=factorial(n)

nfat=gamma(n+1)

Gamma function $\Gamma(z) = \int_0^{\infty} e^{-t} t^{z-1} dt$ è tale che $\Gamma(n) = (n-1)!$

Esempio: somma/prodotto delle componenti di un array

```
function S = somma(a)
S=0;
for k=1:numel(a)
    S=S+a(k);
end
```

S=sum(a)

prodotto

Somma cumulativa

S=cumsum(a)

```
x=ones(1,5)
x =
    1    1    1    1    1
disp(sum(x))
5
disp(cumsum(x))
    1    2    3    4    5
disp(cumsum(x,'reverse'))
    5    4    3    2    1
disp(cumsum(x)/sum(x))
    0.2    0.4    0.6    0.8    1
```

S=prod(a)

Prodotto cumulativo

S=cumprod(a)

```
x=1:5
x =
    1    2    3    4    5
disp(prod(x))
120
disp(cumprod(x))
    1    2    6   24   120
disp(cumprod(x,'reverse'))
120   24    6    2    1
```

Esempi: ancora su **somma/prodotto**

```
A=1:6; A=reshape(A,2,3)
```

```
A =
```

```
1 3 5  
2 4 6
```

```
disp(sum(A))
```

somma lungo le righe (1^a dimensione) - **default**

```
3 7 11
```

```
disp(sum(A,2))
```

somma lungo le colonne (2^a dimensione)

```
9  
12
```

somma lungo le righe (1^a dimensione) - **default**

```
disp(cumsum(A))
```

```
↓ 1 3 5  
3 7 11
```

somma lungo le colonne (2^a dimensione)

```
disp(cumsum(A,2))
```

```
→ 1 4 9  
2 6 12
```

```
disp(prod(A))
```

```
2 12 30
```

```
disp(prod(A,2))
```

```
15
```

```
48
```

```
disp(cumprod(A))
```

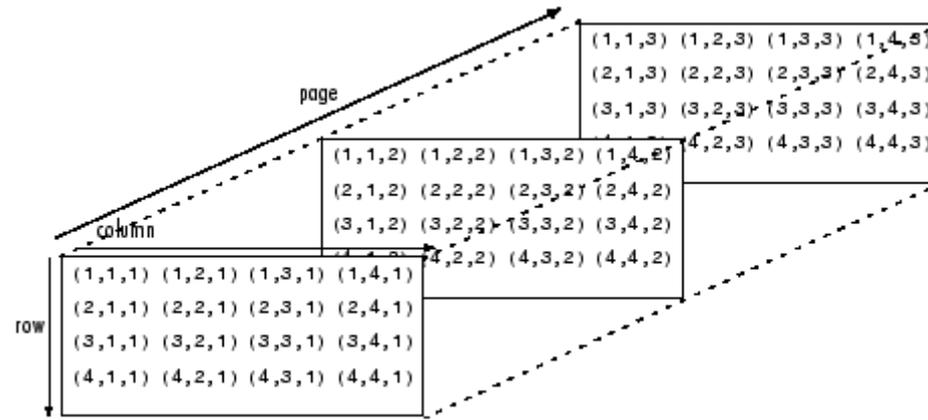
```
1 3 5  
2 12 30
```

```
disp(cumprod(A,2))
```

```
1 3 15  
2 8 48
```

MATLAB consente l'uso di array multidimensionali

Esempio



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

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

```
A(:, :, 2)=[10 11 12; 13 14 15; 16 17 18]
```

```
A =
     1     2     3
     4     5     6
     7     8     9
     10    11    12
     13    14    15
     16    17    18
1ª faccia
```

```
A(:, :, 1) =
     1     2     3
     4     5     6
     7     8     9
```

```
A(:, :, 2) =
     10    11    12
     13    14    15
     16    17    18
2ª faccia
```

```
10    11    12
13    14    15
16    17    18
```

```
B=A(:); disp(B(1:13))
```

```

1
4 1ª colonna, 1ª faccia
7
2
5 2ª colonna, 1ª faccia
8
3
6 3ª colonna, 1ª faccia
9
10 1ª colonna, 2ª faccia
13
16
11 2ª colonna, 2ª faccia
```