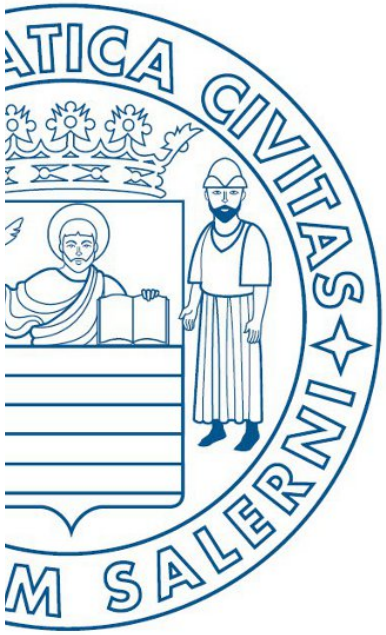


UNIVERSITÀ DEGLI STUDI DI SALERNO

Università di Salerno
Dipartimento di
Ingegneria Industriale

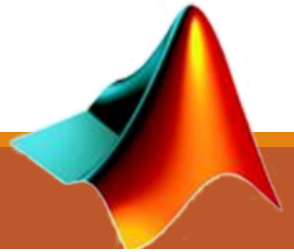
Fondamenti di Informatica

Grafici e Modelli Matematici

Prof. Christian Esposito

Corso di Laurea in Ingegneria Meccanica e Gestionale (Classe I)

A.A. 2017/18



MATLAB

OUTLINE

- Grafici in MATLAB
  - Diagrammi x,y
  - Istogrammi
- Modelli Matematici
  - Ricerca della funzione
  - Interpolazione ed Estrapolazione

Grafici in MATLAB (1)

MATLAB mette a disposizioni varie funzioni al fine di rappresentare graficamente in 2D o anche 3D un insieme di dati su un piano cartesiano. La principale funzioni per la realizzazione di diagrammi bi-dimensionali o xy è plot():

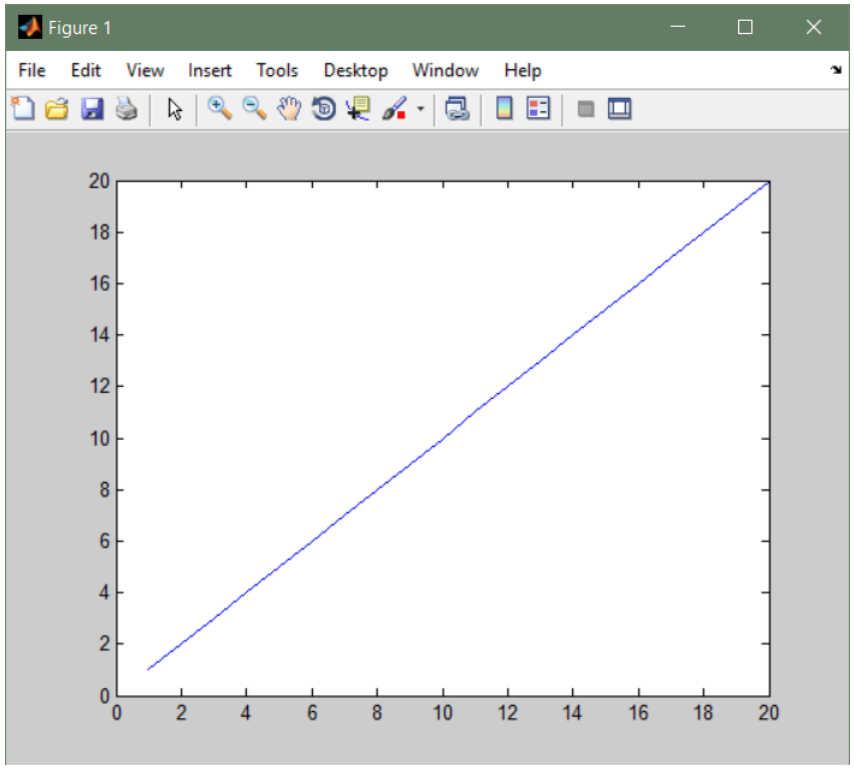
- **plot(*x*, *y*)**
  - MATLAB genera un grafico basandosi sull'array *x* per l'asse X e sull'array *y* per l'asse Y
    - ***NOTA***: Gli array devono essere della stessa lunghezza

- **plot(*x*)**
  - MATLAB genera un grafico lineare basandosi sull'array *x* sia per l'asse X e sia per l'asse Y

Grafici in MATLAB (2)

- ***Esempio 1***

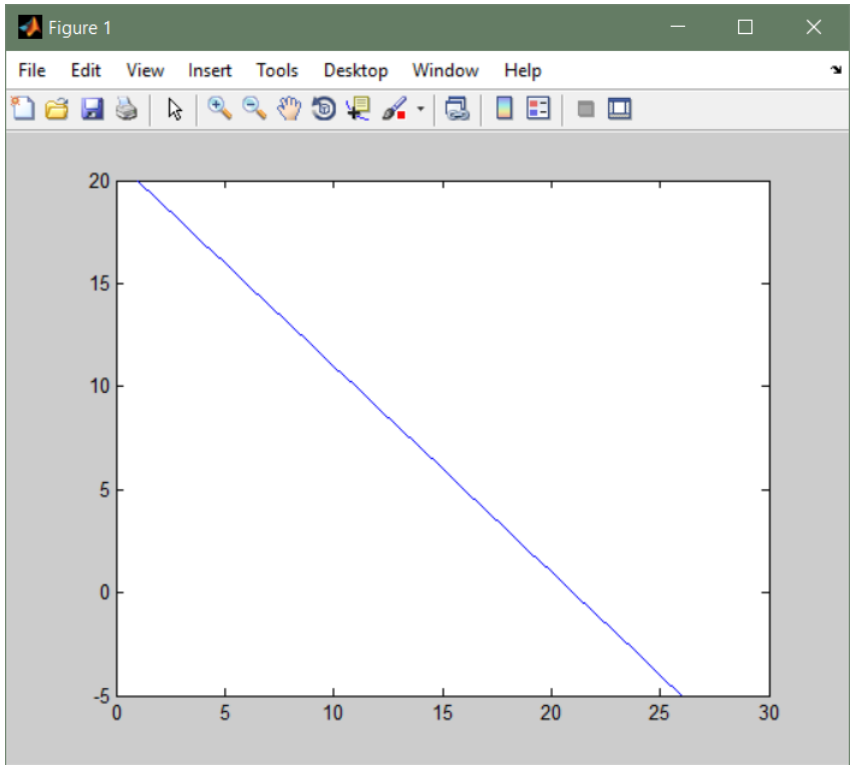
```
x = [1:20]
plot(x)
```



Grafici in MATLAB (3)

- ***Esempio 2***

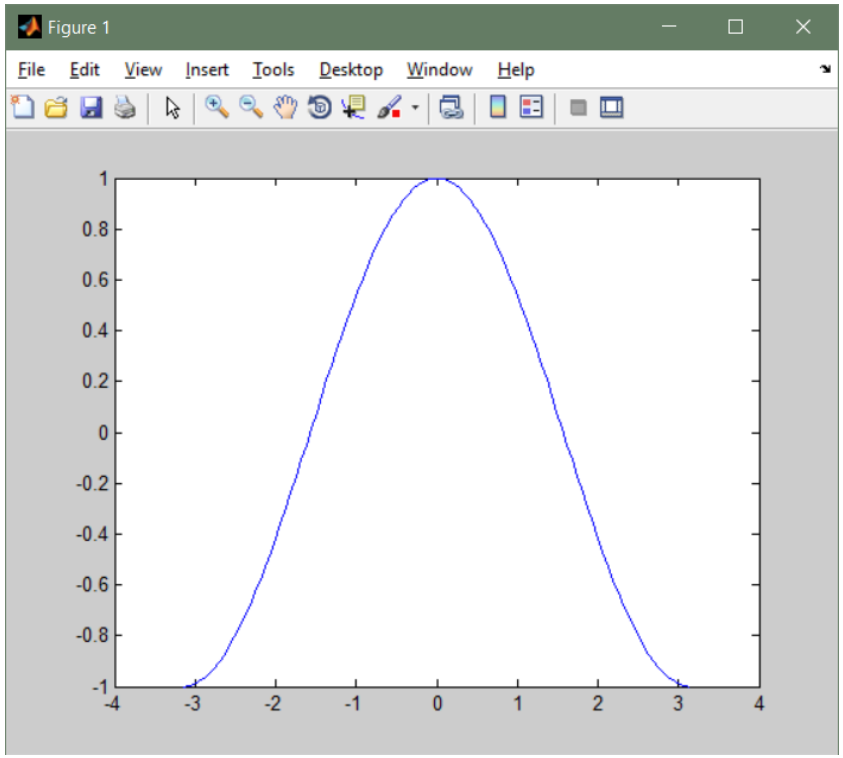
```
x = [20:-1:-5]
plot(x)
```



Grafici in MATLAB (4)

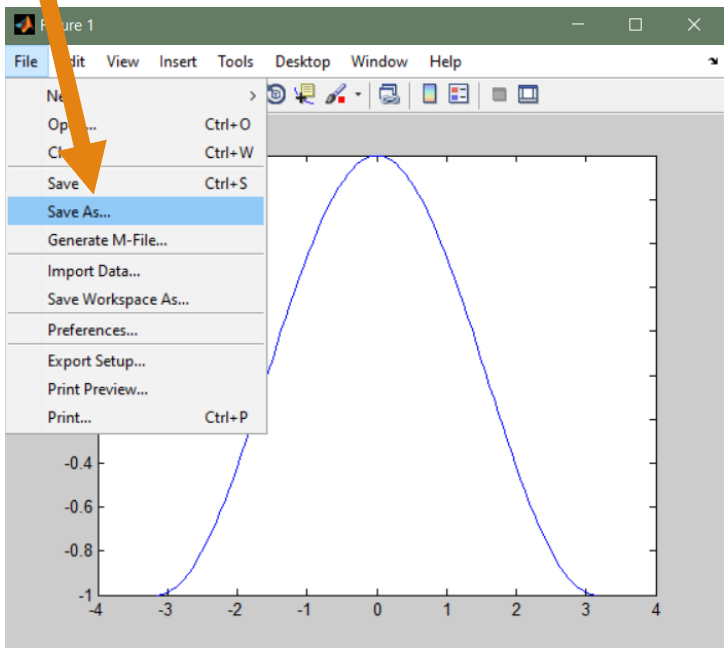
- ***Esempio 3***

```
x = linspace(-pi, pi);
y = cos(x);
plot(x,y)
```



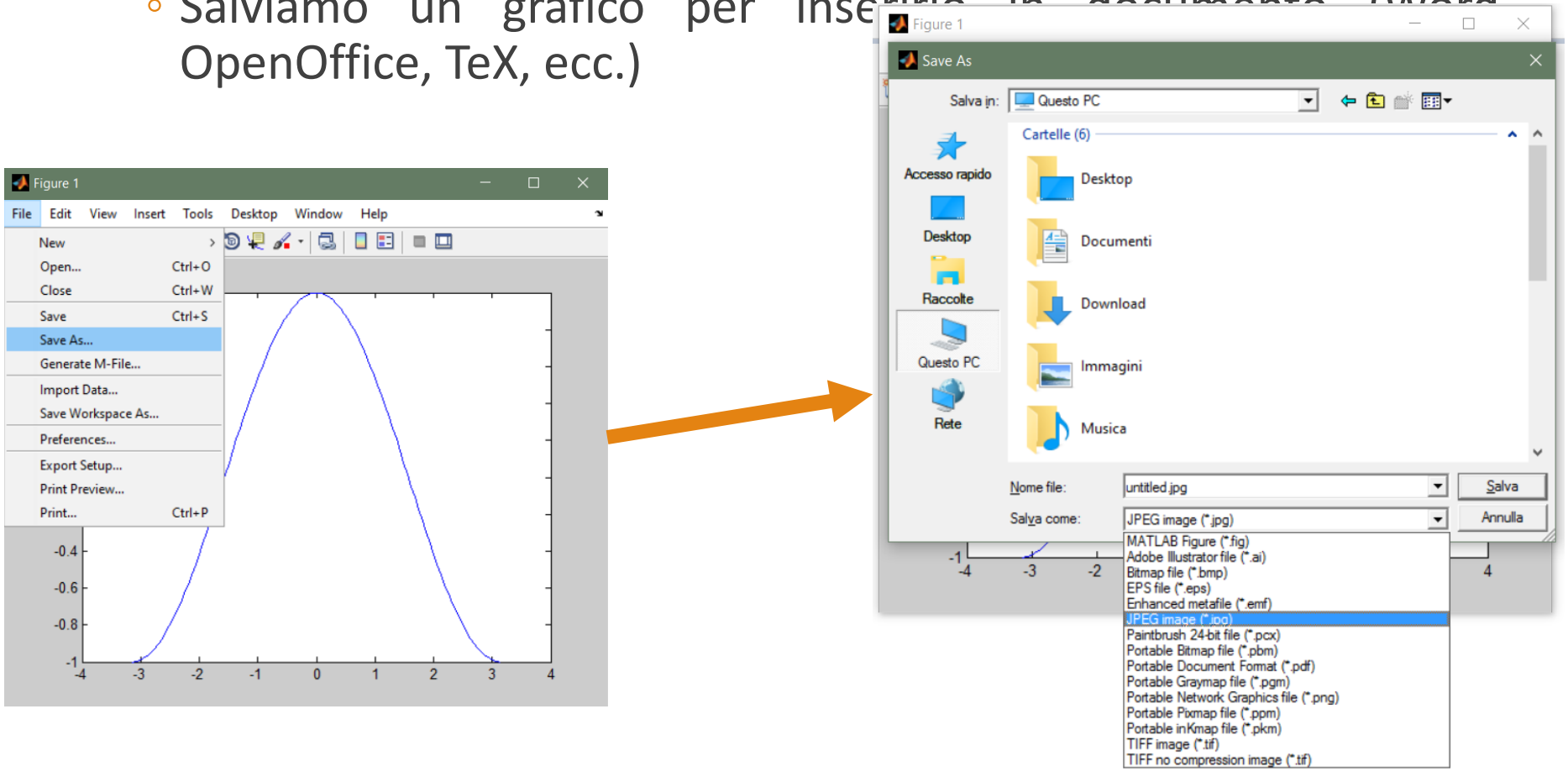
Grafici in MATLAB (5)

- Salviamo un grafico per inserirlo in documento (Word, OpenOffice, TeX, ecc.)



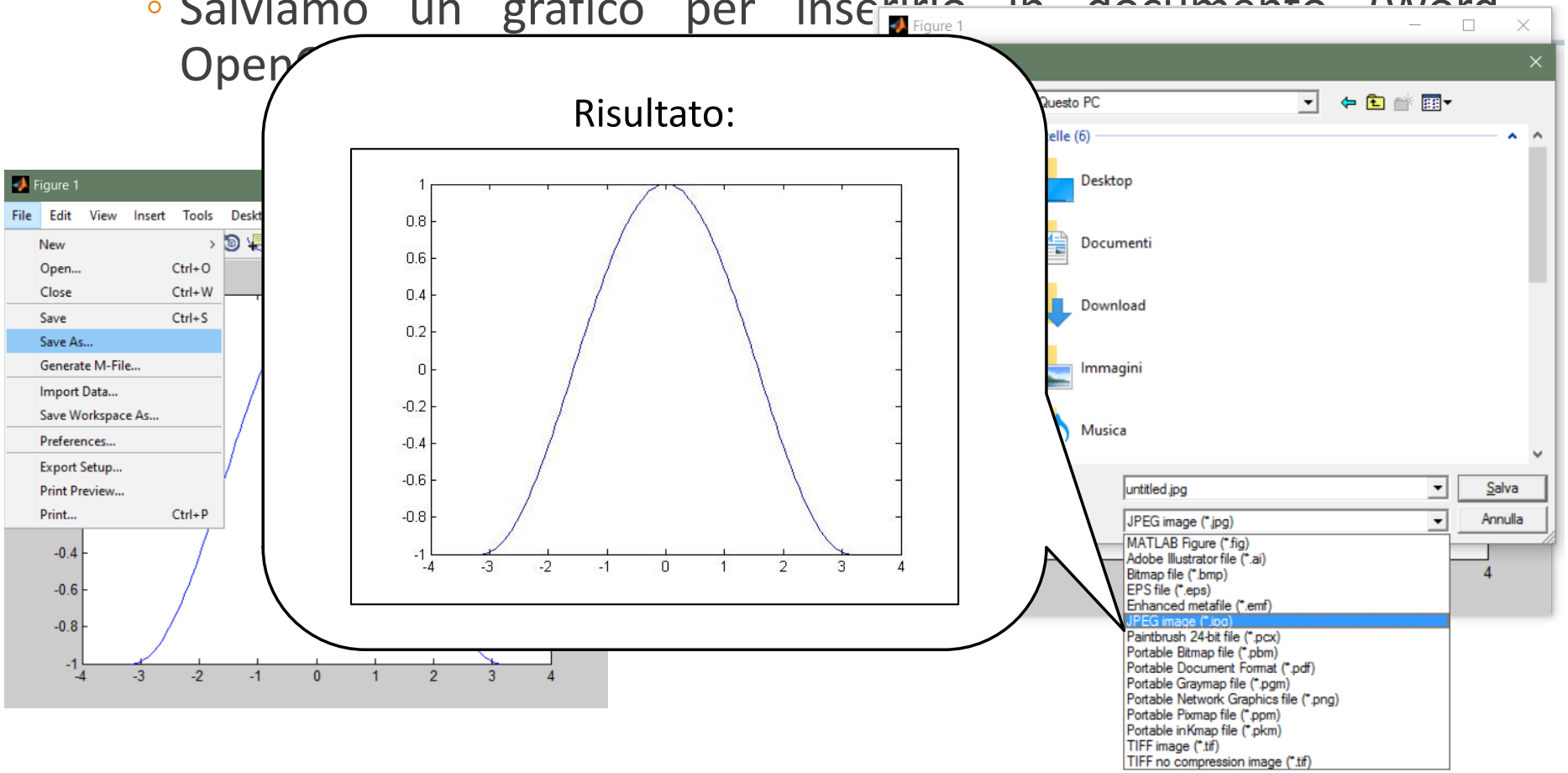
Grafici in MATLAB (5)

- Salviamo un grafico per inserirlo in documento (Word, OpenOffice, TeX, ecc.)



Grafici in MATLAB (5)

- Salviamo un grafico per inserirlo in documento (Word, OpenOffice, ...)

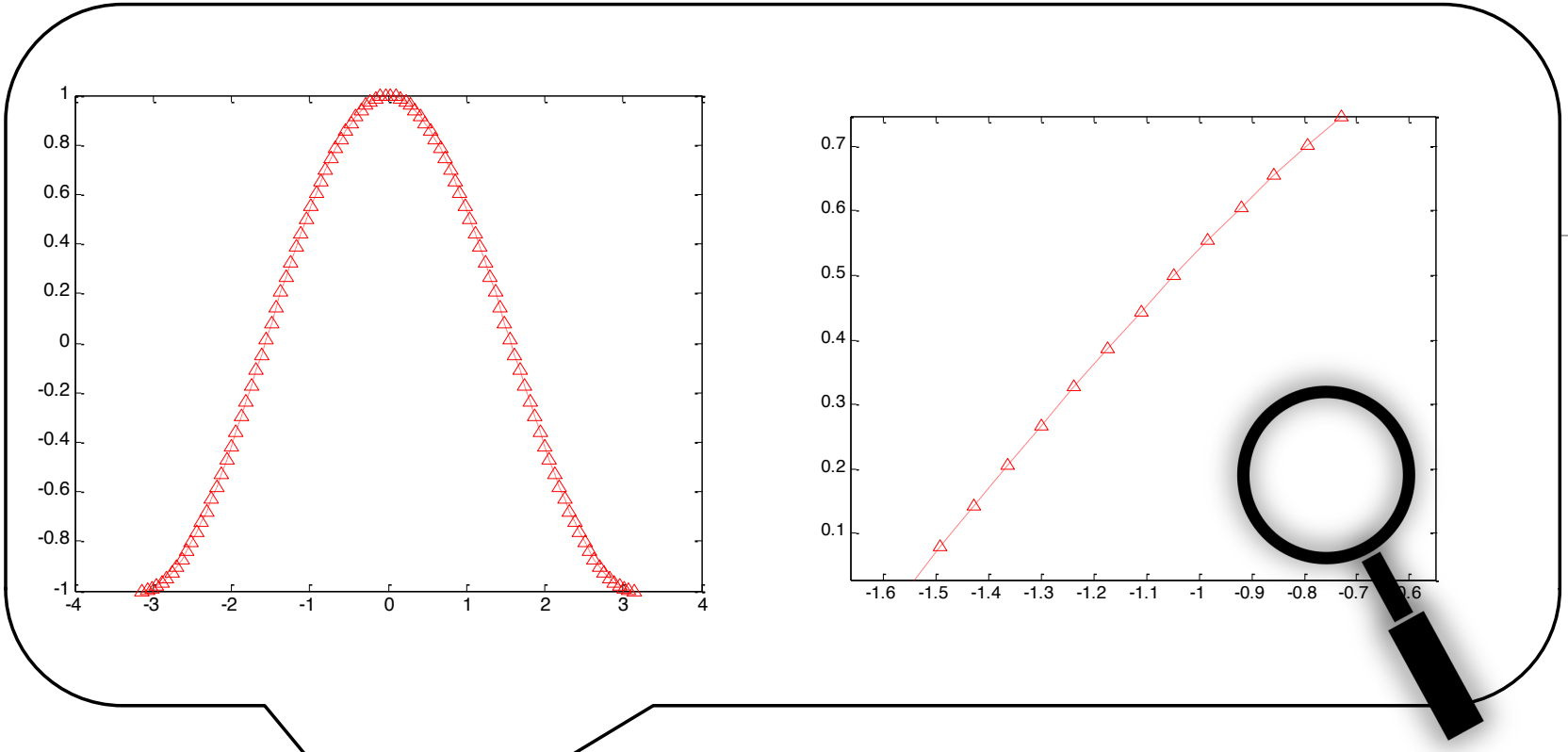


Grafici in MATLAB (6)

- Personalizzazione dei grafici
  - E' possibile personalizzare:
    - ***Colore*** **della linea** e degli **indicatori**
    - ***Stile*** degli **indicatori** (cerchio, puntino, ecc.)
    - ***Stile*** della ***linea*** (tratteggiata, ecc.)

- La personalizzazione avviene tramite una stringa particolare

- ***Esempio***
  - plot(x, y, ***'r^:'***)
    - ***r*** → indica il colore di linea e indicatori (***red*** in questo esempio)
    - ^ → indica lo stile degli indicatori (**triangolo** in questo esempio)
    - **:** → indica lo stile della linea (***punteggiata*** in questo esempio)



- ***Esempio***
  - plot(x, y, ***'r^:'***)
    - ***r*** → indica il colore di linea e indicatori (***red*** in questo esempio)
    - ^ → indica lo stile degli indicatori (**triangolo** in questo esempio)
    - **:** → indica lo stile della linea (***punteggiata*** in questo esempio)

Grafici in MATLAB (7)

- plot(x, y, '***123***')

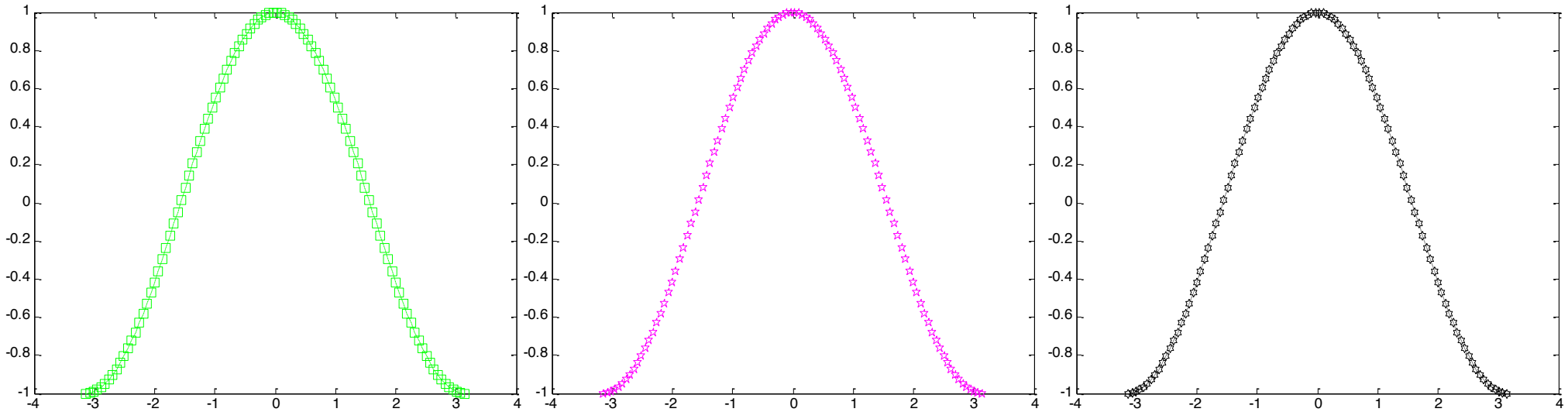
1	Colori
b	blue
g	green (verde)
r	red (rosso)
c	cyan (ciano)
m	magenta
y	yellow (giallo)
k	black (nero)
w	white (bianco)
	default

2	Indicatori
.	punto
o	cerchio
x	croce a x
+	più
s	square (quadrato)
^	triangolo
*****	stella
h	hexagon (esagono)
	nessun indicatore

3	Stile Linea
-	continua
:	punteggiata
-.	punto e tratteggio
--	tratteggiata
	nessuna linea

Grafici in MATLAB (8)

- ***Esempi (x e y dall'Esempio 2)***
  1. `plot(x,y,'gs--')`
  2. `plot(x,y,'mp')`
  3. `plot(x,y,'kh:')`

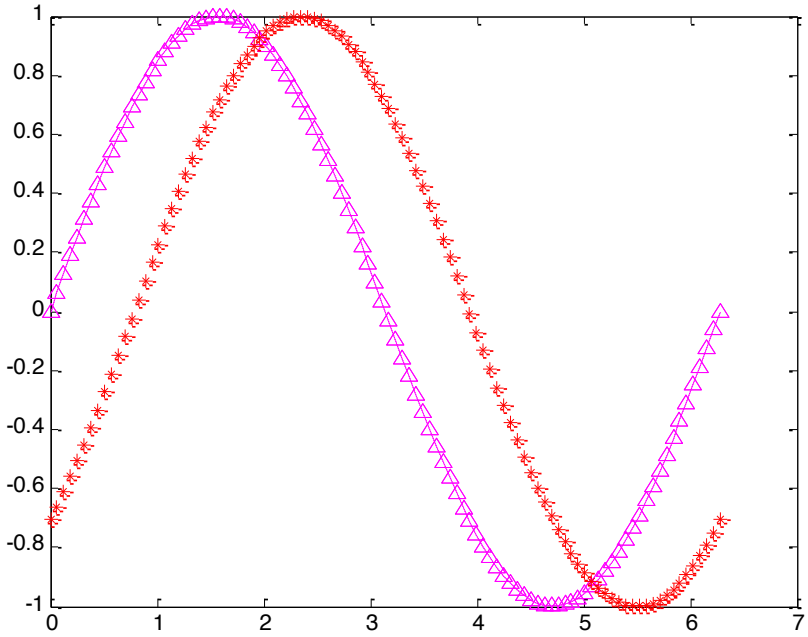


1 2 3

Grafici in MATLAB (9)

- Con il comando **`hold on`** è possibile creare un **grafico che rappresenti più informazioni**, mediante **più linee e/o indicatori**

```
x = linspace(0,2*pi,100);
y1 = sin(x);
y2 = sin(x-pi/4);
hold on;
plot(x,y1,'m^--');
plot(x,y2,'r*');
```



Grafici in MATLAB (10)

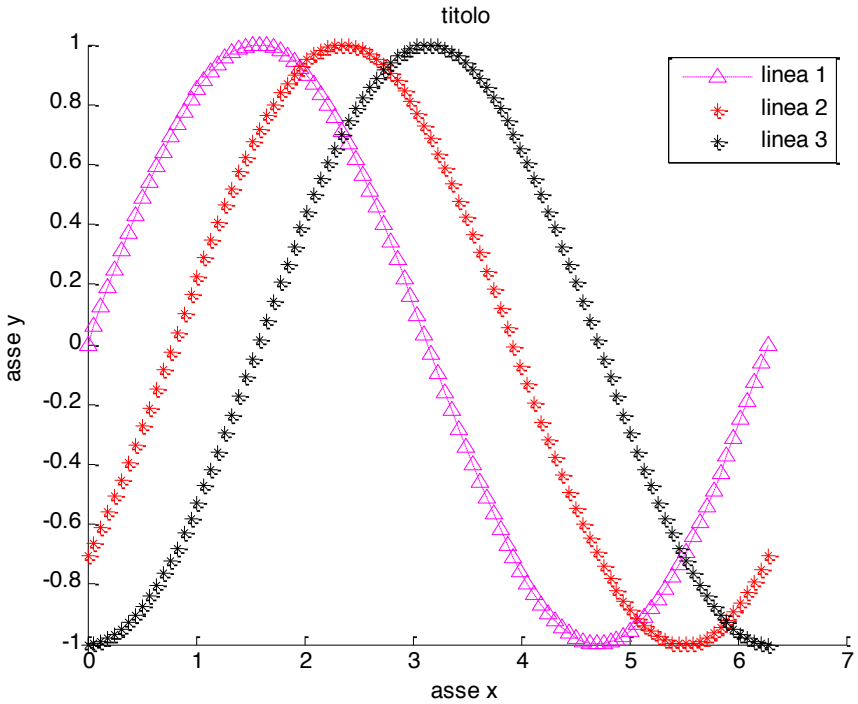
- MATLAB permette di aggiungere varie etichette al grafico

- **title('titolo')**
  - Aggiunge l'etichetta relativa al titolo del grafico

- **xlabel('asse x')**
  - Aggiunge l'etichetta relativa all'asse X

- **ylabel('asse y')**
  - Aggiunge l'etichetta relativa all'asse Y

- **legend('linea 1', 'linea 2', 'linea 3')**
  - Aggiunge la leggenda al grafico



Grafici in MATLAB (11)

- **plot(x,y)**
  - Grafico *lineare* in x e y
- **semilogx(x,y)**
  - Grafico in *scala logaritmica* sull'asse x e *lineare* sull'asse y
- **semilogy(x,y)**
  - Grafico *lineare* sull'asse x ed in *scala logaritmica* sull'asse y
- **loglog(x,y)**
  - Grafico in *scala logaritmica* sull'asse x e y

Grafici in MATLAB (12)

- E' anche possibile creare dei grafici in sotto-figure
  - **subplot(m, n, p)**
    - Crea una finestra contenente una griglia di sotto-figure, inizialmente vuota, composta da **m** righe ed **n** colonne
    - **p** indica la posizione *corrente* della sotto-figura all'interno della griglia (la numerazione di **p** è indicata nell'esempio seguente – griglia: **m**=2, **n**=3)

1	2	***3***
4	5	6

p=***3***

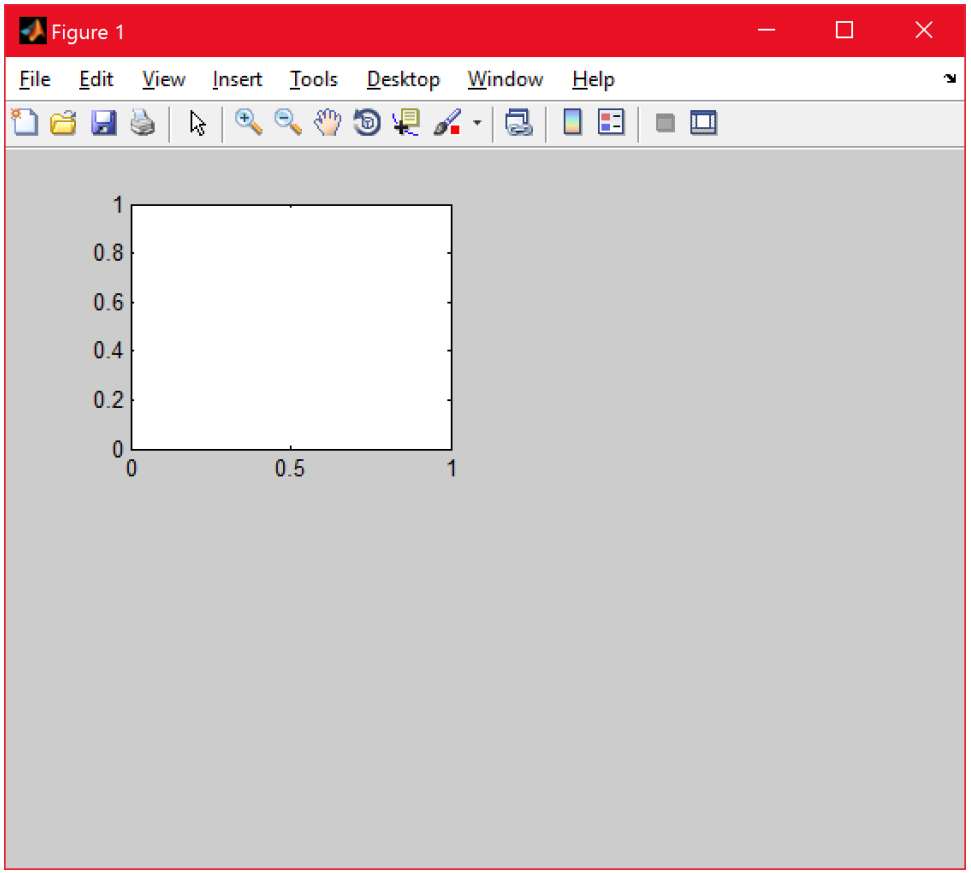
  - Indicata la griglia e la posizione con ***subplot***, sarà possibile utilizzare la funzione **plot** (o altre funzioni per i grafici) per creare il grafico nella sotto-figura specificata
    - È comunque possibile personalizzare il titolo, le etichette degli assi, il tipo, ecc.

Grafici in MATLAB (13)

```
x = linspace(0,2*pi,100);
y1 = sin(x);
y2 = sin(x-pi/4);
y3 = sin(x-pi/2);
y4 = sin(x-2*pi/3);
```

Grafici in MATLAB (13)

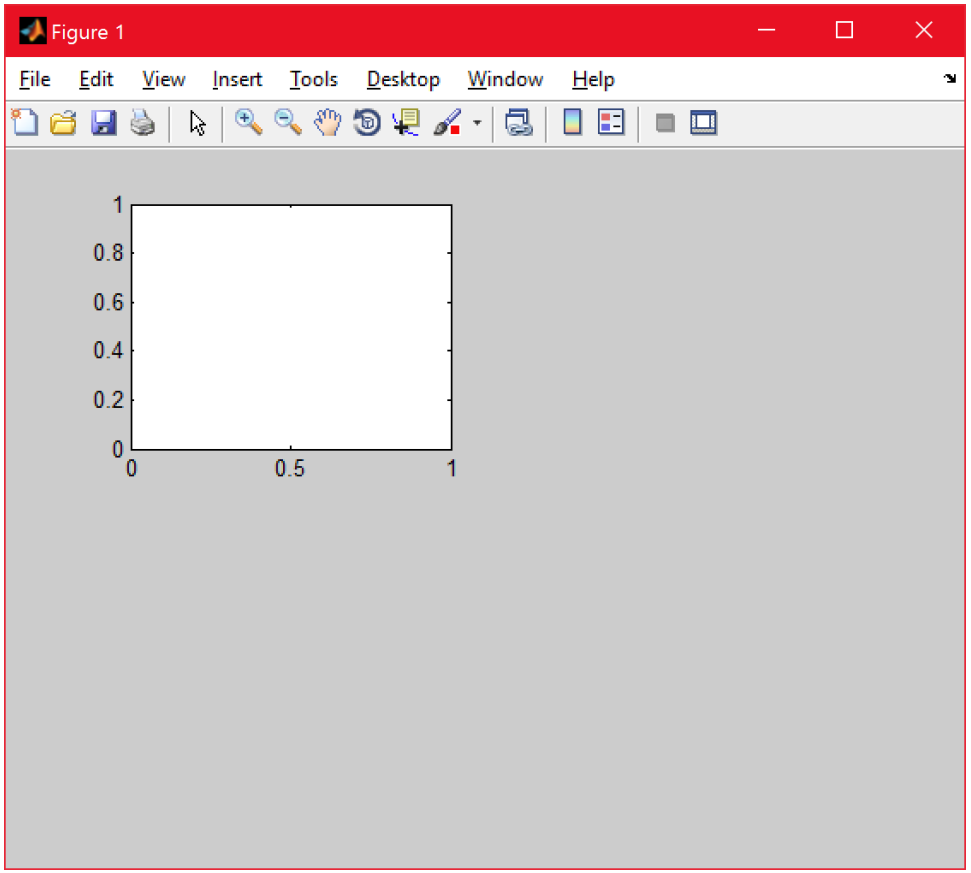
```
x = linspace(0,2*pi,100);
y1 = sin(x);
y2 = sin(x-pi/4);
y3 = sin(x-pi/2);
y4 = sin(x-2*pi/3);
hold on;
subplot(2,2,1);
```



Grafici in MATLAB (13)

```
x = linspace(0,2*pi,100);
y1 = sin(x);
y2 = sin(x-pi/4);
y3 = sin(x-pi/2);
y4 = sin(x-2*pi/3);
hold on;
subplot(2,2,1);
```

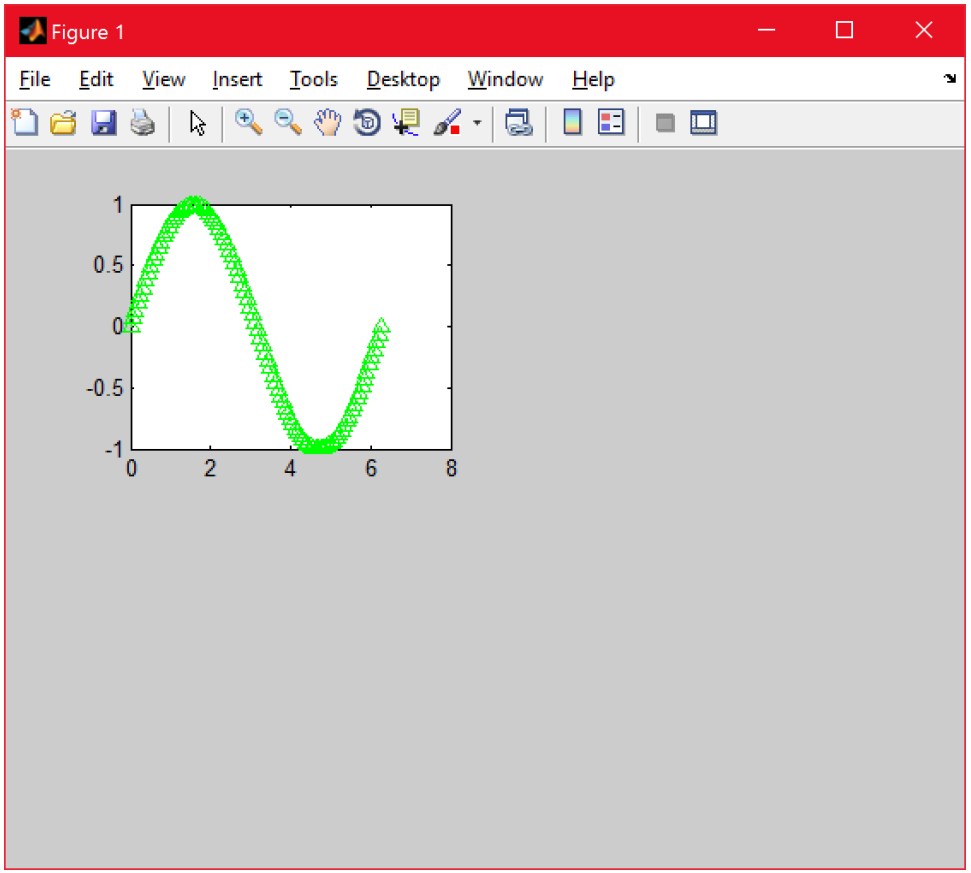
Crea una griglia di 2 righe e di 2 colonne ed imposta la posizione corrente a 1



Grafici in MATLAB (13)

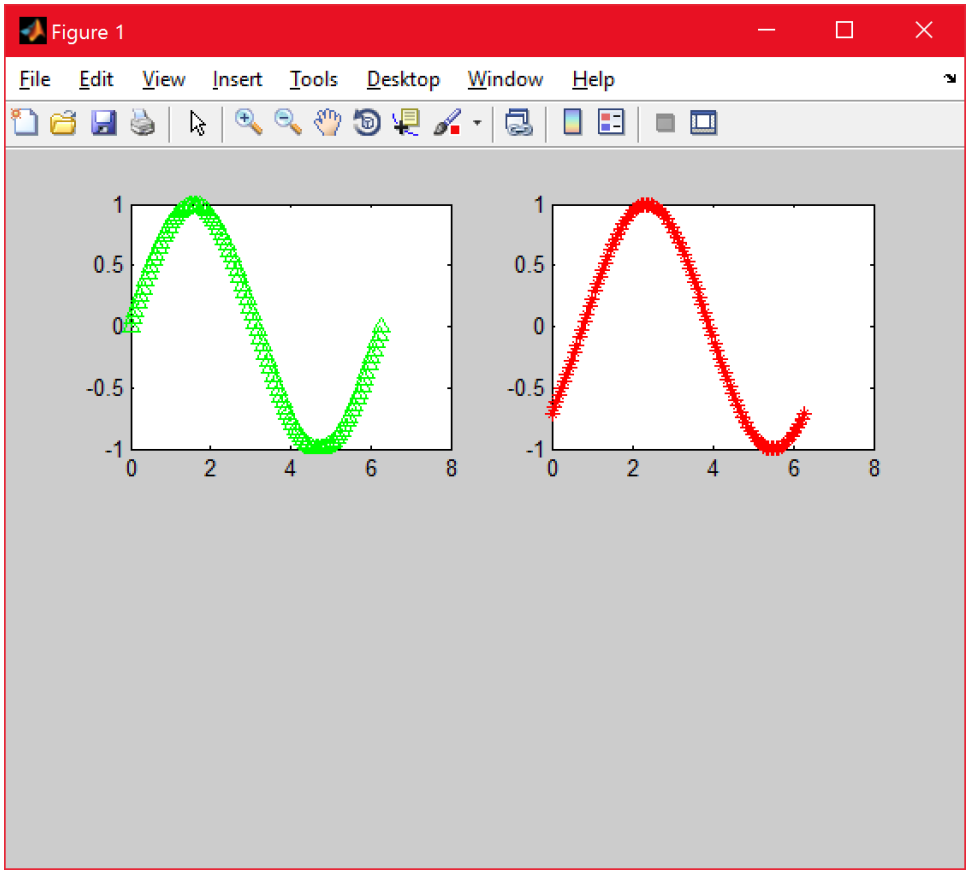
```
x = linspace(0,2*pi,100);
y1 = sin(x);
y2 = sin(x-pi/4);
y3 = sin(x-pi/2);
y4 = sin(x-2*pi/3);
hold on;
subplot(2,2,1);
plot(x,y1,'g^:');
```

Crea il primo grafico nella posizione 1



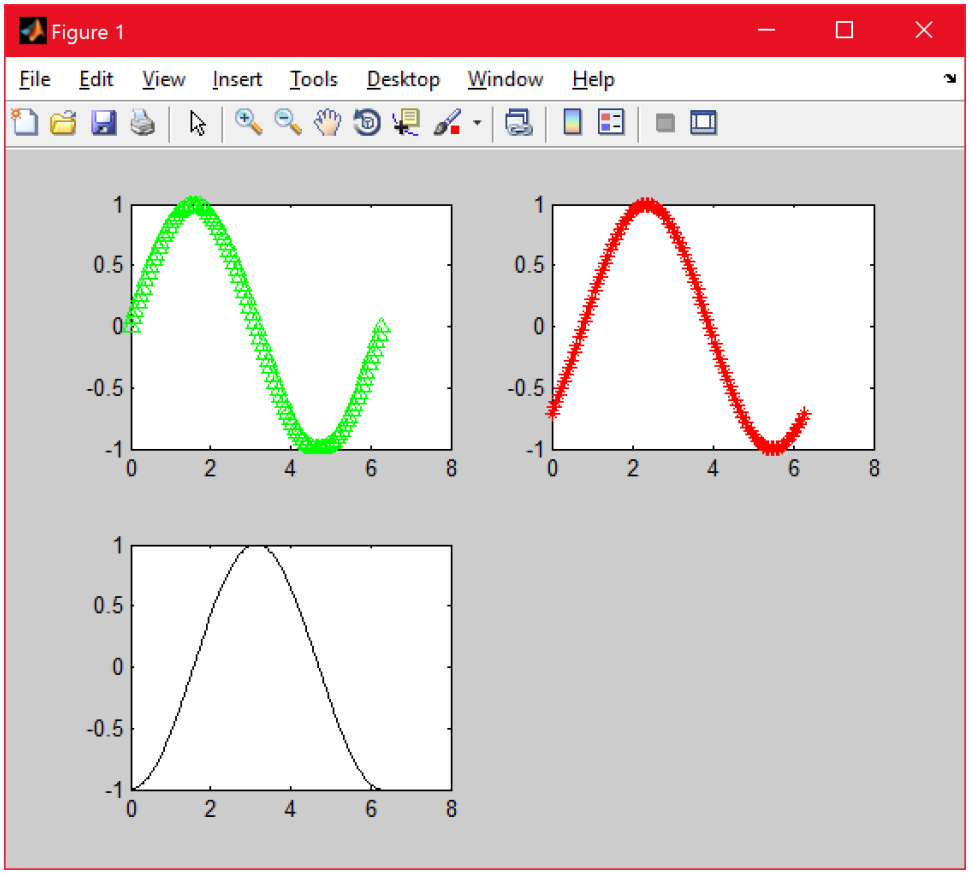
Grafici in MATLAB (13)

```
x = linspace(0,2*pi,100);
y1 = sin(x);
y2 = sin(x-pi/4);
y3 = sin(x-pi/2);
y4 = sin(x-2*pi/3);
hold on;
subplot(2,2,1);
plot(x,y1,'g^:');
subplot(2,2,2);
plot(x,y2,'r*--');
```



Grafici in MATLAB (13)

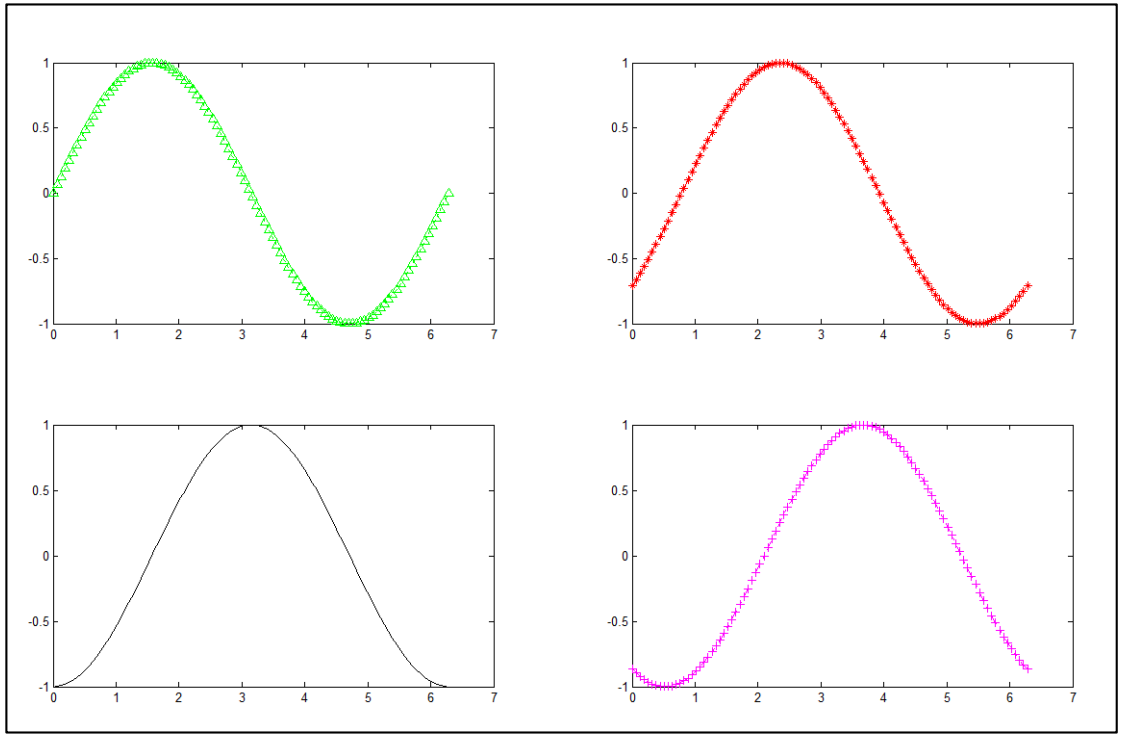
```
x = linspace(0,2*pi,100);
y1 = sin(x);
y2 = sin(x-pi/4);
y3 = sin(x-pi/2);
y4 = sin(x-2*pi/3);
hold on;
subplot(2,2,1);
plot(x,y1,'g^:');
subplot(2,2,2);
plot(x,y2,'r*--');
subplot(2,2,3);
plot(x,y3,'k');
```



Grafici in MATLAB (13)

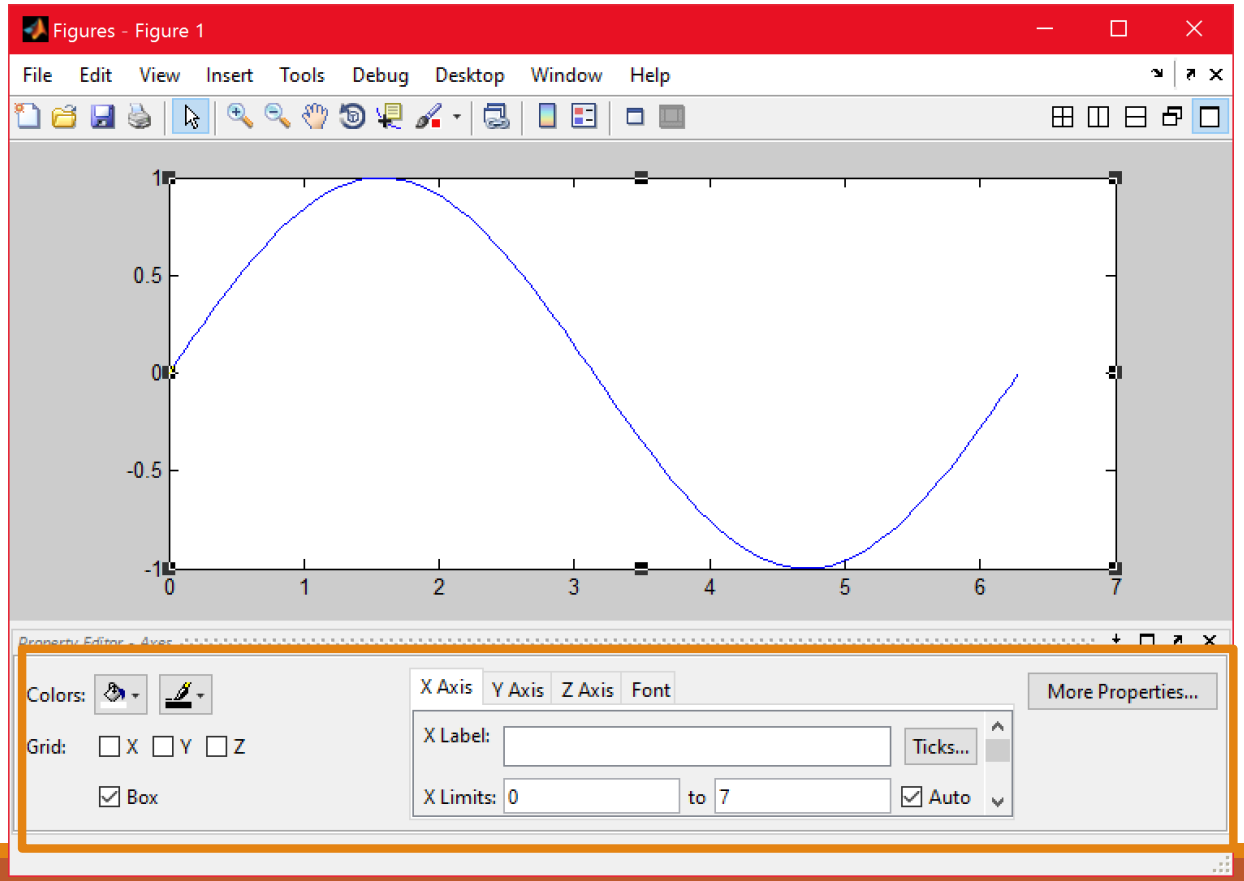
```
x = linspace(0,2*pi,100);
y1 = sin(x);
y2 = sin(x-pi/4);
y3 = sin(x-pi/2);
y4 = sin(x-2*pi/3);
hold on;
subplot(2,2,1);
plot(x,y1,'g^:');
subplot(2,2,2);
plot(x,y2,'r*--');
subplot(2,2,3);
plot(x,y3,'k');
subplot(2,2,4);
plot(x,y4,'m+-.');
hold off;
```

Output Finale:



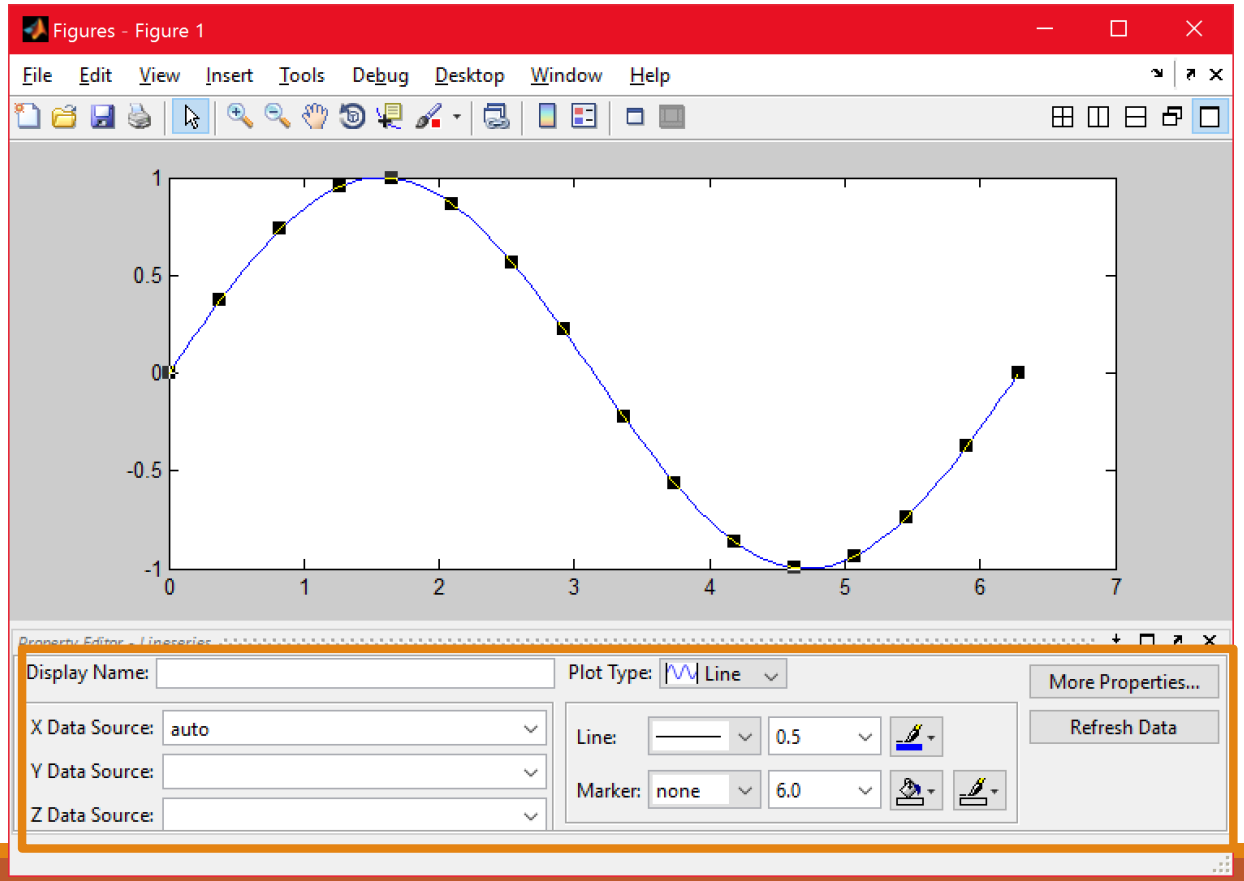
Grafici in MATLAB (14)

- MATLAB mette a disposizione un editor che consente di modificare le varie proprietà del grafico
  - Titolo grafico
  - Etichette assi
  - Definire griglie
  - Font
  - Tipo di grafico
  - …



Grafici in MATLAB (14)

- MATLAB mette a disposizione un editor che consente di modificare le varie proprietà del grafico
  - Titolo grafico,
  - Etichette assi
  - Definire griglie
  - Font
  - Tipo di grafico
  - …



Grafici in MATLAB (15)

- MATLAB permette di generare diversi tipi di grafici, come:
  - Istogrammi, grafici a torta, grafici a barre
  - Superfici
  - Ecc.

- Per ulteriori informazioni è possibile reperire la guida di MATLAB con i comandi
  - `help graph2d`
  - `help graph3d`

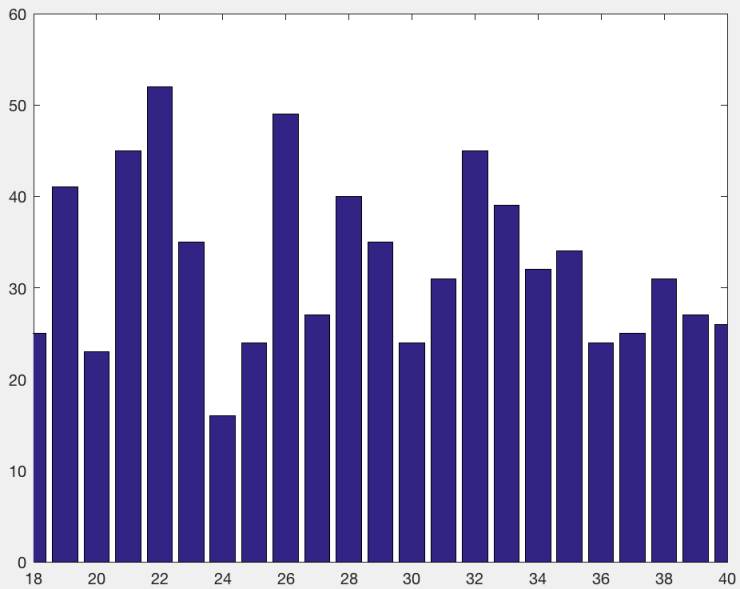
Istogrammi (1)

- MATLAB permette a disposizione una funzione per la costruzione di un grafico ad istogramma, ovvero una rappresentazione grafica di un insieme di dati x,y costituita da più rettangoli adiacenti:
  - ogni rettangolo ha per base un certo intervallo della variabile e un'altezza calcolata come densità di frequenza, ovvero pari al rapporto fra la frequenza (assoluta) associata alla classe e l'ampiezza della classe.

- Consideriamo l'esempio di un'intervista, e dell'analisi dell'età degli intervistati. Su un totale di 750 persone intervistate si ha la seguente ripartizione:
  - eta = [18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]
  - frequenza = [25, 41, 23, 45, 52, 35, 16, 24, 49, 27, 40, 35, 24, 31, 45, 39, 32, 34, 24, 25, 31, 27, 26]

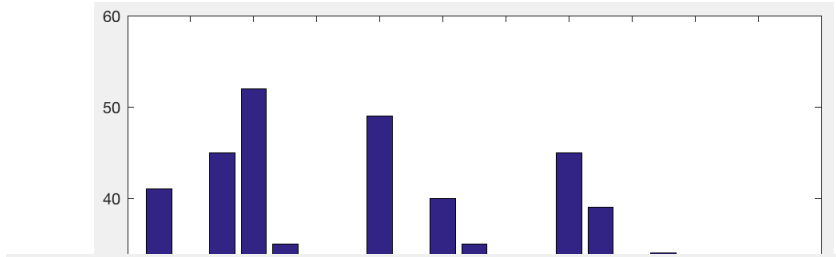
Istogrammi (2)

```
>> bar(eta, frequenza);
>> axis([18 40 0 60]);
```



Istogrammi (2)

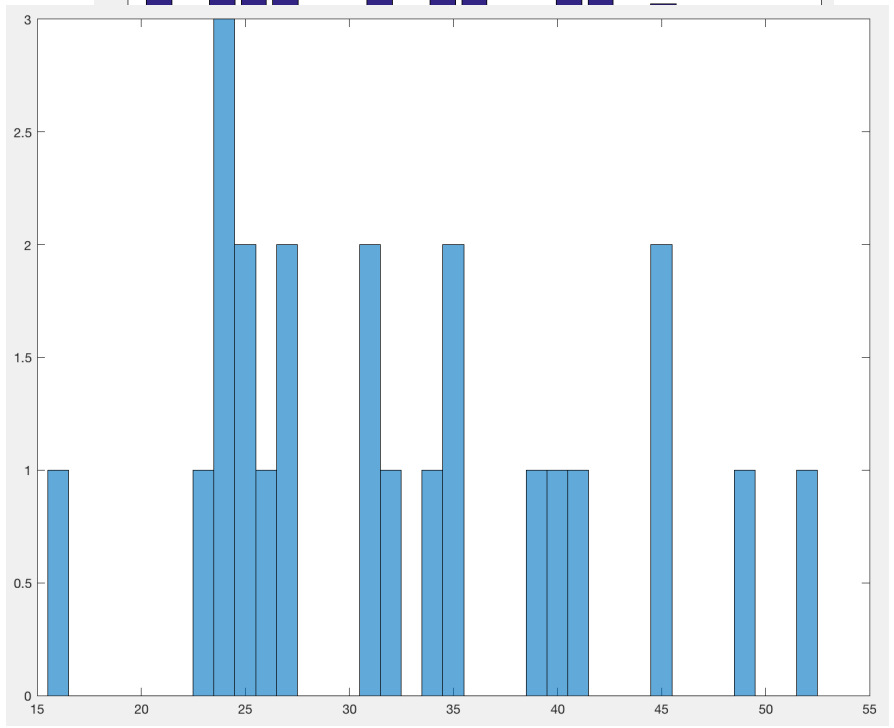
```
>> bar(eta, frequenza);
>> axis([18 40 0 60]);
```



- Analizziamo la frequenza dei campioni, usando l'istogramma.

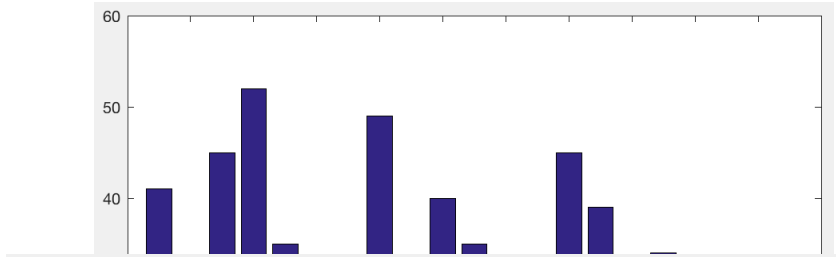
```
>> histogram (frequenza);
```

- Come determinare il campione più frequente?



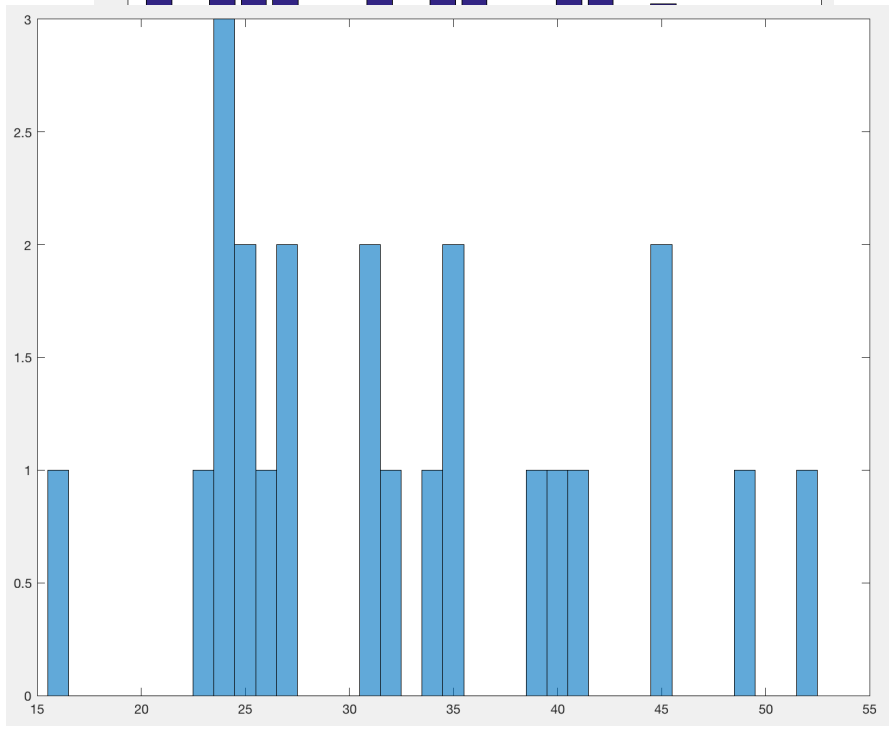
Istogrammi (2)

```
>> bar(eta, frequenza);
>> axis([18 40 0 60]);
```



- Analizziamo la frequenza dei campioni, usando l'istogramma.

```
>> histogram (frequenza);
```



- Come determinare il campione più frequente?

```
>> [M,F]=mode(frequenza)
M =     24
F =      3
```

Istogrammi (3)

- Come determinare le frequenze per ogni campione?

```
>> unqx = unique(frequenza)
unqx =
    Columns 1 through 9
16    23    24    25    26    27    31    32    34
    Columns 10 through 16
35    39    40    41    45    49    52
>> valueCount = histc(frequenza, unqx)
valueCount =
    Columns 1 through 9
1     1     3     2     1     2     2     1     1
    Columns 10 through 16
2     1     1     1     2     1     1
```

Istogrammi (3)

- Come determinare le frequenze per ogni campione?

```
>> unqx = unique(frequenza)
unqx =
    Columns 1 through 9
16    23    24    25    26    27    31    32    34
    Columns 10 through 16
35    39    40    41    45    49    52
>> valueCount
valueCount =
    Columns 1 through 9
1     1     3     2     1     2     2     1     1
    Columns 10 through 16
2     1     1     1     2     1     1
```

Il ritorno rappresenta i singoli campioni dell'input, senza ripetizioni, ovvero i valori unici.

Istogrammi (3)

- Come determinare le frequenze per ogni campione?

```
>> unqx = unique(frequenza)
unqx =
    Columns 1 through 9
16    23    24    25    26    27    31    32    34
    Columns 10
35    39    40
```

Dato ogni campione unico, otteniamo in uscita la sua frequenza dell'array dei campioni di partenza.

```
>> valueCount = histc(frequenza, unqx)
valueCount =
    Columns 1 through 9
1     1     3     2     1     2     2     1     1
    Columns 10 through 16
2     1     1     1     2     1     1
```

Istogrammi (4)

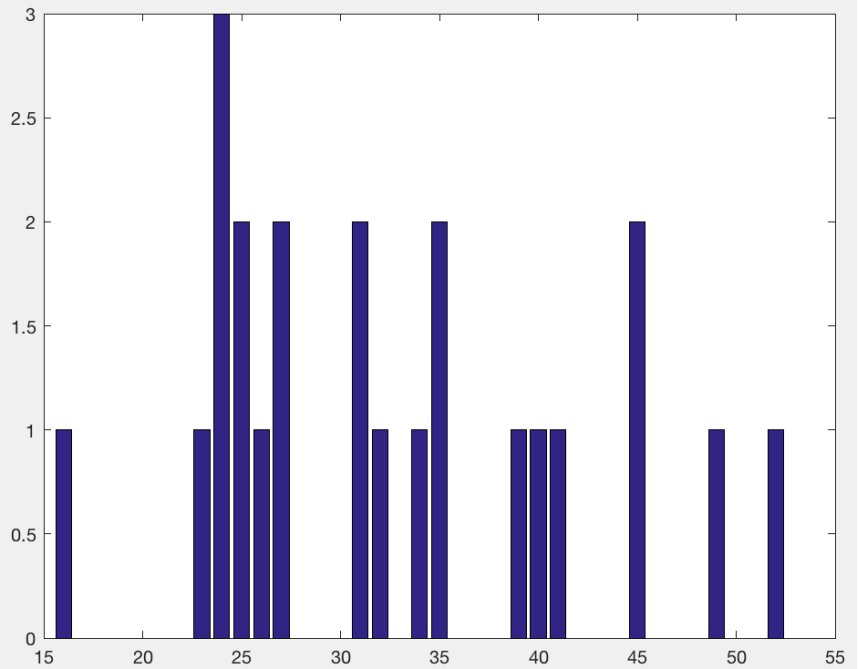
- Se plottiamo con bar unqx e valueCount, cosa otteniamo?

```
>> bar(unqx, valueCount)
```

Istogrammi (4)

- Se plottiamo con bar unqx e valueCount, cosa otteniamo?

```
>> bar(unqx, valueCount)
```



Ricerca di Funzione (1)

- La ricerca di funzione consiste nel trovare una funzione che descrive un insieme di dati. Le funzioni più considerate sono:
  - Funzione lineare

    $$y(x) = mx + b$$

  - Funzione potenza

    $$y(x) = bx^m$$

  - Funzione esponenziale

    $$y(x) = be^{mx}$$

  - Funzione Logaritmica

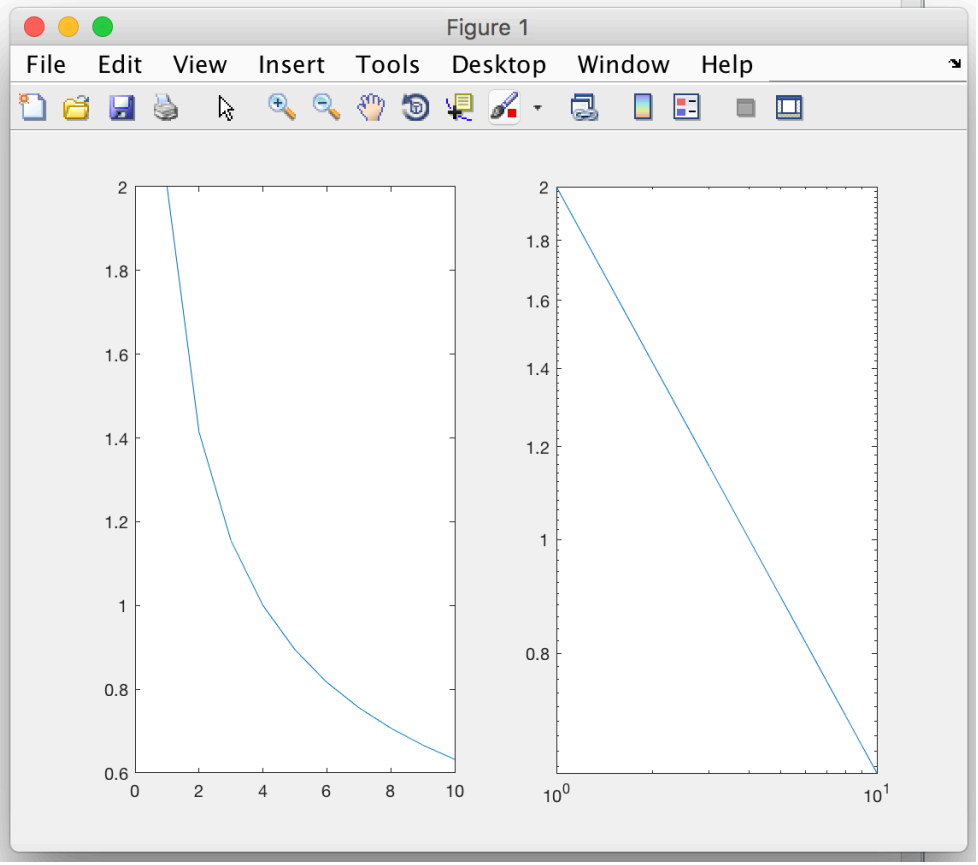
    $$y(x) = m\,log(x) + b$$

Ricerca di Funzione (2)

- Ogni funzione è rappresentata da una retta se viene scelto un particolare tipo di assi cartesiani:
  - La funzione lineare è rappresentata da una retta se gli assi del diagramma sono in scala lineare.
  - La funzione potenza è una retta se entrambi gli assi del diagramma sono logaritmici.
  - La funzione esponenziale sono rette in un diagramma semilogaritmico su asse y.
  - La funzione logaritmica è una retta in un diagramma semilogaritmico su asse x.

Ricerca di Funzione (3)

```
>> x = 0:10;
>> y = 2.*x.^(-0.5);
>> hold on;
>> subplot(1,2,1);
>> plot(x,y)
>> subplot(1,2,2);
>> loglog(x,y)
>> hold off;
>>
```



Ricerca di Funzione (4)

- Nella rappresentazione grafica, si ricerca una linea retta perché è una forma facile da riconoscere.
- Il procedimento per ricercare una funzione è il seguente:
  1. Esaminare i dati in prossimità dell'origine degli assi:
     a) la funzione lineare e quella logaritmica passano per l'origine solo se b = 0;
     b) la funzione potenza passa per l'origine solo se m > 0;
     c) la funzione esponenziale non passa mai per l'origine, tranne quando b = 0, che è un caso anomalo.

Ricerca di Funzione (5)

2. Rappresenta i dati in un diagramma con assi lineari. Se si ha una retta, i dati sono rappresentabili da una funzione lineare. Altrimenti, se ci sono dati per x = 0:
   a) Se y(0) = 0, si deve provare con la funzione potenza o logaritmica;
   b) Se y(0) ≠ 0, si deve provare con la funzione esponenziale.
3. Se si pensa che una funzione potenza possa rappresentare i dati, bisogna creare un diagramma logaritmico e verificare se i dati formano una retta.
4. Se la funzione esponenziale è una candidata cercate una retta sul diagramma semilogaritmico in y, per la funzione logaritmica quello semilogaritmico in x.

Ricerca di Funzione (6)

5. I diagrammi logaritmici e semi-logaritmici devono essere utilizzati solo per identificare il tipo di funzione corretta per la rappresentazione dei dati, non per calcolare i coefficienti b e m.

   MATLAB mette a disposizione una funzione per l'identificazione dei valori da assegnare ai coefficienti: polyfit.

La funzione polyfit(x, y, n) trova i coefficienti di un polinomio di grado n che meglio approssima i dati di coordinate (x,y), secondo il metodo dei quadrati minimi.

Ricerca di Funzione (7)

Il metodo dei minimi quadrati è una tecnica di regressione che permette di trovare una funzione, rappresentata da una curva ottima, che si avvicini il più possibile ad un insieme di dati.

La funzione trovata deve essere quella che minimizza la somma dei quadrati delle distanze tra i dati osservati e quelli della curva che rappresenta la funzione stessa.

Vogliamo determinare una funzione lineare che meglio approssima i nostri dati sperimentali e poter decidere sulla bontà di questa approssimazione.

Ricerca di Funzione (8)

Sia $f(x) = mx + q$, la coppia di dati (x_i, y_i) appartiene al grafico di f(x) se e solo se vale la relazione $y_i = mx_i + q$; quindi l'errore $\delta_i = mx_i + q - y_i$ misura la distanza che c'è tra il dato sperimentale (x_i , y_i) ed il dato teorico $(x_i, f(x_i))$. Abbiamo n errori $\delta_1, \delta_2, ..., \delta_n$.

Prendiamo come misura di quanto f(x) approssima i dati la media aritmetica degli errori elevati al quadrato:

$$f(m,q) = \sum_{i=1}^{n} \frac{(mx_i + q - y_i)^2}{n}$$

Vogliamo determinare m e q tale da rendere minima f(m,q).

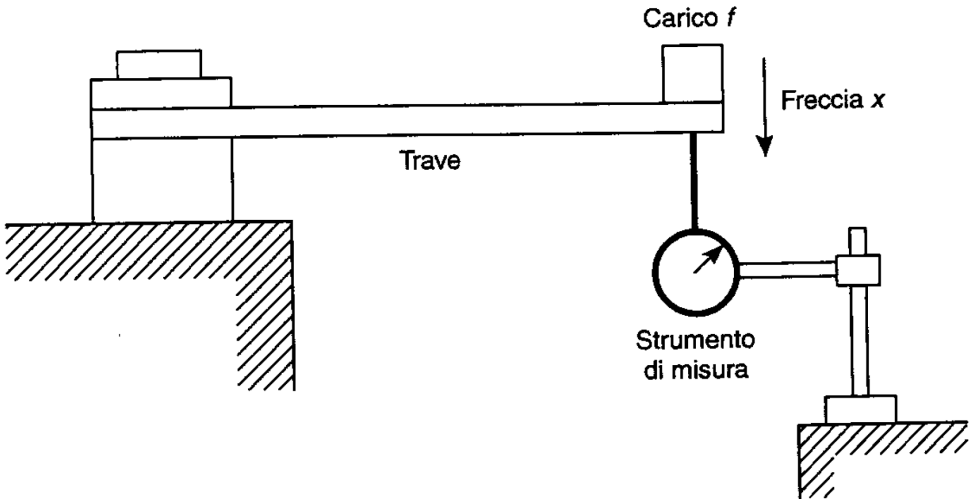
Ricerca di Funzione (9)

Siccome siamo interessati a funzioni che rappresentano delle rette con assi lineari, logaritmici o semilogaritmici, siamo interessati a polinomi di primo grado come $w = p_1 z + p_2$.

- Nel caso di funzione lineare p_1 e p_2 sono m e b, e sono in ritorno dell'invocazione p = polyfit(x,y,1).
- Nel caso di funzione potenza, p = polyfit(log10(x), log10(y),1), e p_1 è m mentre p_2 è $\log_{10} b$, così b è 10^{p_2}.
- Nel caso di funzione esponenziale, p = polyfit(x, log10(y),1), e p_1 sarà m mentre p_2 è $\log_{10} b$, così che b è ottenibile da 10^{p_2}.
- Nel caso di funzione logaritmica, bisogna invocare p = polyfit(log10(x), y,1), e p_1 sarà m e p_2 b.

Ricerca di Funzione (10)

La tecnica di ricerca delle funzioni è utile in tutte le branche dell'ingegneria, con interessanti applicazioni.



La freccia elastica di una trave a sbalzo è lo spostamento che la sua estremità libera subisce in seguito all'applicazione di un carico concentrato all'estremità libera.

Dato un determinato carico f, si ha una serie di valori per la freccia x prodotta da una trave.

Ricerca di Funzione (11)

Carico f (libbre)	Freccia x (pollici)
0	0
100	0,09
200	0,18
300	0,28
400	0,37
500	0,46
600	0,55
700	0,65
800	0,74

Cerchiamo la funzione che rappresenta i seguenti dati.

```
>> carico = 0:100:800;
>> freccia = [0, 0.09, 0.18, 0.28, 0.37, 0.46, 0.55, 0.65, 0.74]
```

Ricerca di Funzione (12)

```
>> subplot(2,1,1);
>> plot(carico, freccia, 'o');
>> xlabel('Carico applicato (libbre)'),
ylabel('Freccia (pollici)');
```

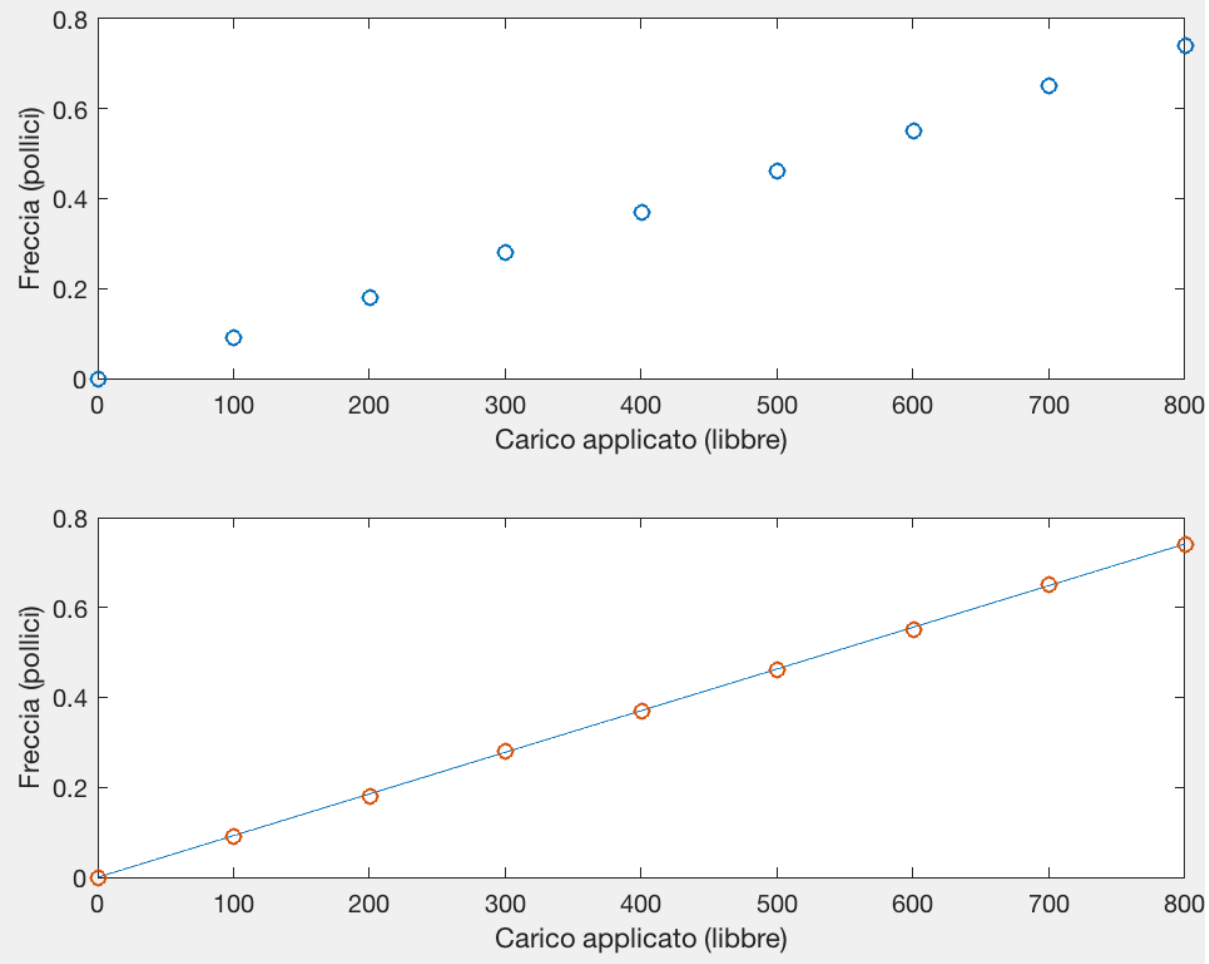
Vedendo i dati sul grafico con assi lineari, si nota che sono disposti lungo una retta, che può essere descritta dalla relazione f = kx, dato che x(0) = 0. k è chiamata costante elastica della trave, che può essere trovato da polyfit().

Ricerca di Funzione (13)

```
>> p = polyfit(carico,freccia,1);
>> k = 1/p(1);
>> f = [0:2:800];
>> x=f./k;
>> subplot(2,1,2);
>> plot(f,x, carico, freccia, 'o');
>> xlabel('Carico applicato (libbre)'),
ylabel('Freccia (pollici)');
```

Questo script ha determinato la costante elastica della trave pari a 1079 libbre/pollice.

Ricerca di Funzione (14)



Interpolazione ed Estrapolazione (1)

Dopo aver scoperto una relazione funzionale, è possibile utilizzarla per

- l'interpolazione - prevedere i casi che ricadono all'interno dell'intervallo dei dati originali;

  Es. Stimare il carico necessario per ottenere una freccia di 150 libbre.

- l'estrapolazione – prevedere casi che si verificano all'esterno dell'intervallo dei dati originali.

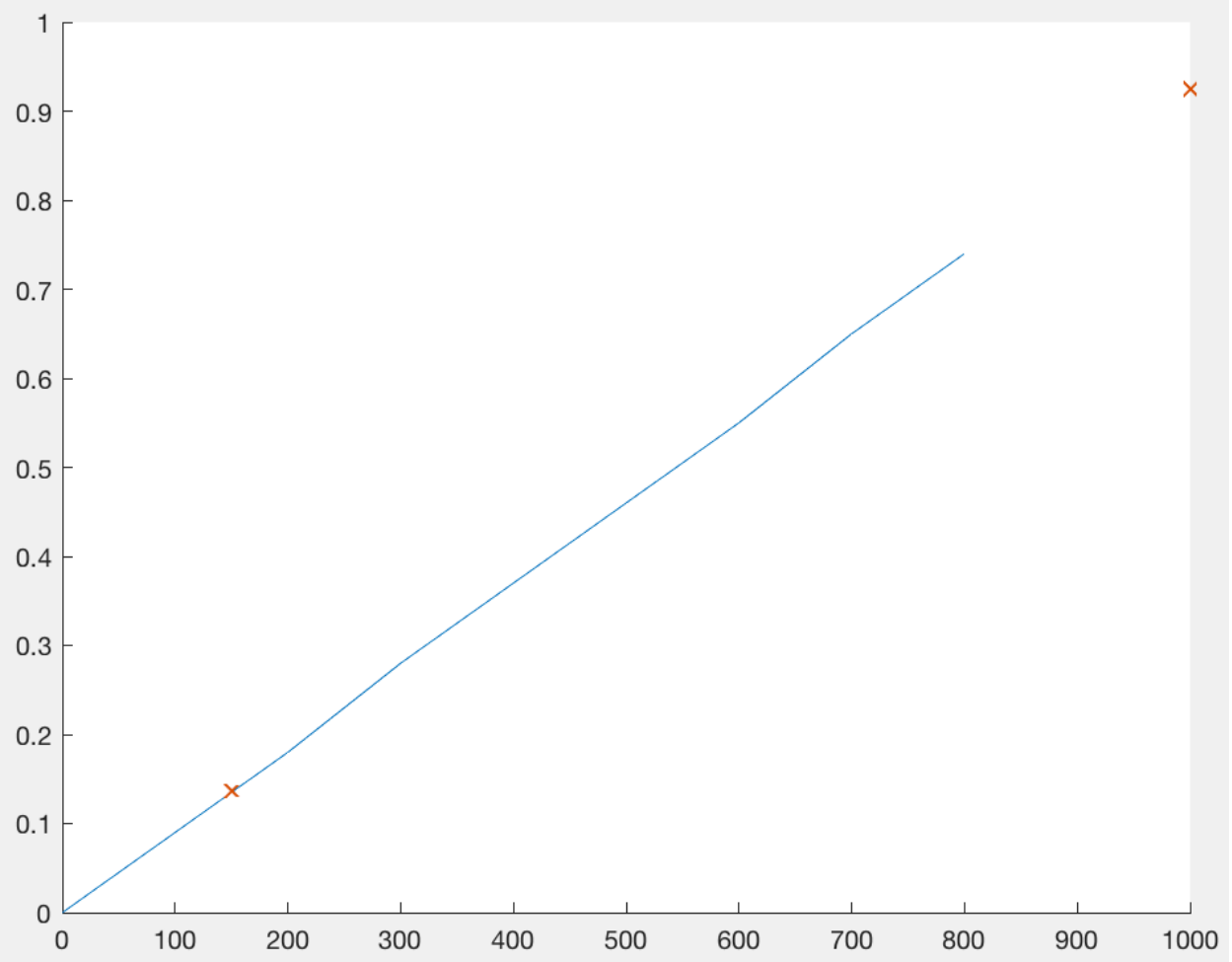
  Es. Stimare il carico necessario per ottenere una freccia di 1000 libbre.

Interpolazione ed Estrapolazione (2)

MATLAB offre una funzione per la valutazione di un polinomio in un determinato punto x.

```
>> carico = 0:100:800;
freccia = [0, 0.09, 0.18, 0.28, 0.37, 0.46, 0.55, 0.65, 0.74];
>> p = polyfit(carico,freccia,1);
>> x = [150, 1000];
>> y = polyval(p, x)
y =    0.1372    0.9249
```

Interpolazione ed Estrapolazione (3)



Interpolazione ed Estrapolazione (4)

MATLAB offre delle funzioni per la realizzazione dell'interpolazione senza effettuare la ricerca di una funzione caratterizzante. Utilizzare le linee rette per collegare i punti da interpolare è il metodo più facile, ciò è realizzato dalla funzione interp1.

```
>> carico = 0:100:800;
freccia = [0, 0.09, 0.18, 0.28, 0.37, 0.46, 0.55,
0.65, 0.74];
>> x_int = [80, 150, 240];
>> y = interp1(carico, freccia, x_int)
y =    0.0720    0.1350    0.2200
```

Interpolazione ed Estrapolazione (4)

MATLAB offre delle funzioni per la realizzazione dell'interpolazione senza effettuare la ricerca di una linee rette per odo più facile, ciò è

I valori della variabile indipendente devono essere in ordine ascendente e i valori di interpolazione x_int devono essere compresi nell'intervallo di valori della variabile indipendente.

```
>> carico = 0:100:800;
freccia = [0, 0.09, 0.18, 0.28, 0.37, 0.46, 0.55,
0.65, 0.74];
>> x_int = [80, 150, 240];
>> y = interp1(carico, freccia, x_int)
y =    0.0720    0.1350    0.2200
```

Interpolazione ed Estrapolazione (5)

Una procedura alternativa consiste nell'impiegare un polinomio di grado inferiore fra punti adiacenti. Questa soluzione è l'interpolazione spline, che restituisce delle curve regolari che si adattano molto bene ai punti noti.

Solitamente si impiegano polinomi di terzo grado, così da realizzare un'interpolazione spline cubica. Ciascuno di questi polinomi ha la seguente forma:

$$y_i(x) = a_i(x-x_i)^3+b_i(x-x_i)^2+c_i(x-x_i)+d_i$$

per $x_i \leq x \leq x_{i+1}$ e $i = 1, 2, \ldots, n$. I coefficienti di ogni singolo polinomio sono determinati affinché, il polinomio passi per i punti nodi nelle sue estremità, le pendenze e le curvature dei polinomi devono essere uguali nei punti comuni.

Interpolazione ed Estrapolazione (6)

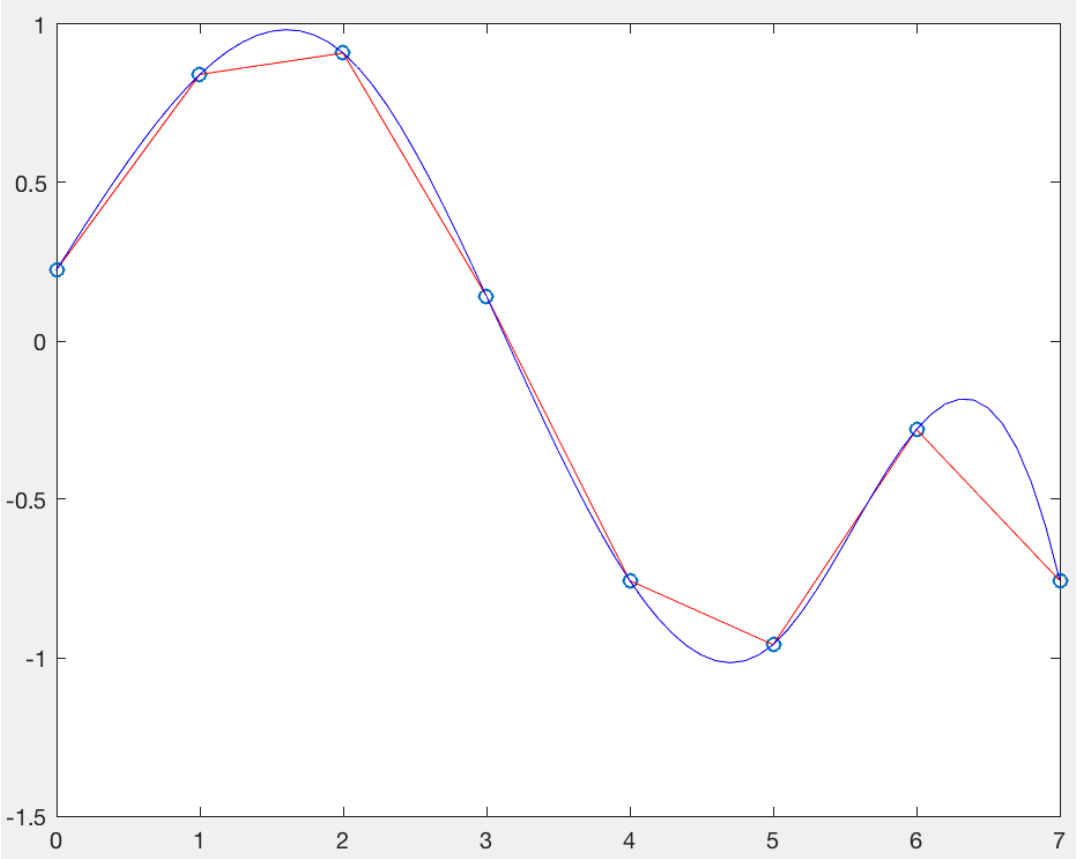
```
>> value = [0 0.2243; 1 0.8415; 2 0.9093; 3
0.1411; 4 -0.7568; 5 -0.9589; 6 -0.2794; 7
-0.7568];
>> x_int = 0:.1:7;

>> y1 = interp1(value(:,1)', value(:,2)', x_int);

>> y2 = spline(value(:,1)', value(:,2)', x_int);

>> plot(value(:,1)', value(:,2)', 'o',
x_int, y1,'r', x_int, y2, 'b')
```

Interpolazione ed Estrapolazione (6)



Interpolazione ed Estrapolazione (6)

```
>> value = [0 0.2243; 1 0.8415; 2 0.9093; 3
0.1411; 4 -0.7568; 5 -0.9589; 6 -0.2794; 7 -
0.7568];
```

spline è la funzione offerta da MATLAB per l'interpolazione spline cubica. Restituisce i valori della variabile dipendente dati i valori noti e la valorizzazione della variabile indipendente.

```
>> y2 = spline(value(:,1)', value(:,2)', x_int);
```

```
>> plot(value(:,1)', value(:,2)', 'o',
x_int, y1,'r', x_int, y2, 'b')
```

Interpolazione ed Estrapolazione (6)

```
>> value = [0 0.2243; 1 0.8415; 2 0.9093; 3
0.1411; 4 -0.7568; 5 -0.9589; 6 -0.2794; 7 -
0.7568];
```

spline è la funzione offerta da MATLAB per l'interpolazione spline cubica. Restituisce i valori della variabile dipendente dati i valori noti e la valorizzazione della variabile indipendente.

```
>> y2 = spline(value(:,1)', value(:,2)', x_int);
```

spline non consente di ottenere i coefficienti dei polinomi, allo scopo MATLAB offre la funzione unmkpp().

Interpolazione ed Estrapolazione (7)

```
>>    [breaks,    coeffs,    m,    n,    d]    =
unmkpp(spline(value(:,1)', value(:,2)'))

breaks =      0      1      2      3      4      5
6     7

coeffs =   -0.0904   -0.0035    0.7111    0.2243
-0.0904   -0.2747    0.4329    0.8415    0.1655
-0.5459   -0.3877    0.9093    0.1348   -0.0495
-0.9832    0.1411    0.1207    0.3550   -0.6778
-0.7568   -0.4318    0.7171    0.3943   -0.9589
-0.4318   -0.5784    0.5329   -0.2794

m =     7

n =     4

d =     1
```

Interpolazione ed Estrapolazione (7)

```
>>    [breaks,    coeffs,    m,    n,    d]    =
unmkpp(spline(value(:,1)', value(:,2)'))

breaks =      0      1      2      3      4      5
6     7

coeffs =   -0.0904   -0.0035    0.7111    0.2243
-0.0904   -0.2747    0.4329    0.8415    0.1655
-0.5459   -0.3877    0.9093    0.1348   -0.0495
-0.9832    0.1411    0.1207    0.3550   -0.6778
-0.7568   -0.4318    0.7171    0.3943   -0.9589
-0.4318   -0.5784    0.5329   -0.2794

m =      7

n =      4

d =      1
```

breaks rappresenta i valori della variabile indipendente; coeffs sono i coefficienti, una riga per polinomio (per un totale di length(breaks) - 1), in una matrice M x N, mentre D è la dimensione di ogni coefficiente.

I Residui (1)

Data una funzione di interpolazione dei dati noti, chiamata f, si definiscono residui gli scarti tra i valori noti assunti dalle variabili dipendenti e il ritorno della funzione quando assume come input i valori della variabile indipendente:

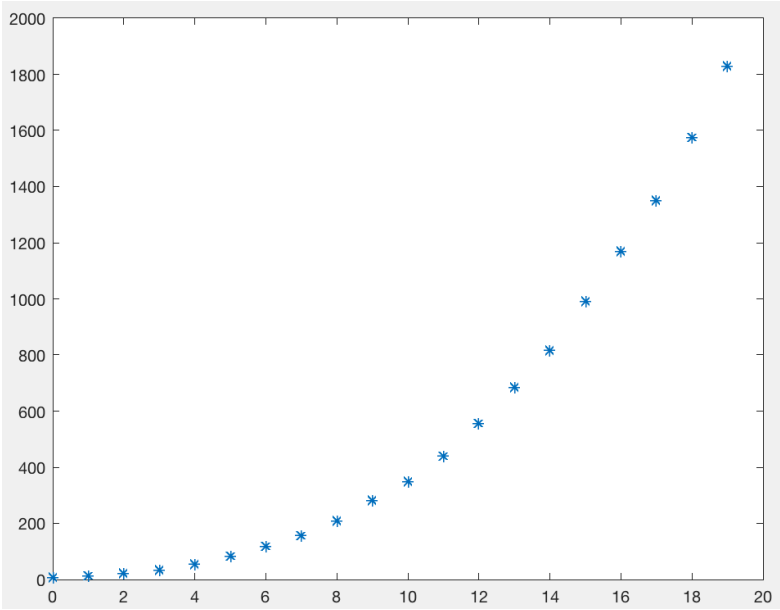
$$\varrho_i = f(x_i)-y_i$$

per i = 1, 2, … , n. I residui possono essere impiegati per determinare la bontà di interpolazione della funzione f rispetto ai dati noti (x_i,y_i).

Se il diagramma dei residui mostra una regolarità, allora esiste una funzione che approssima meglio i dati noti

I Residui (2)

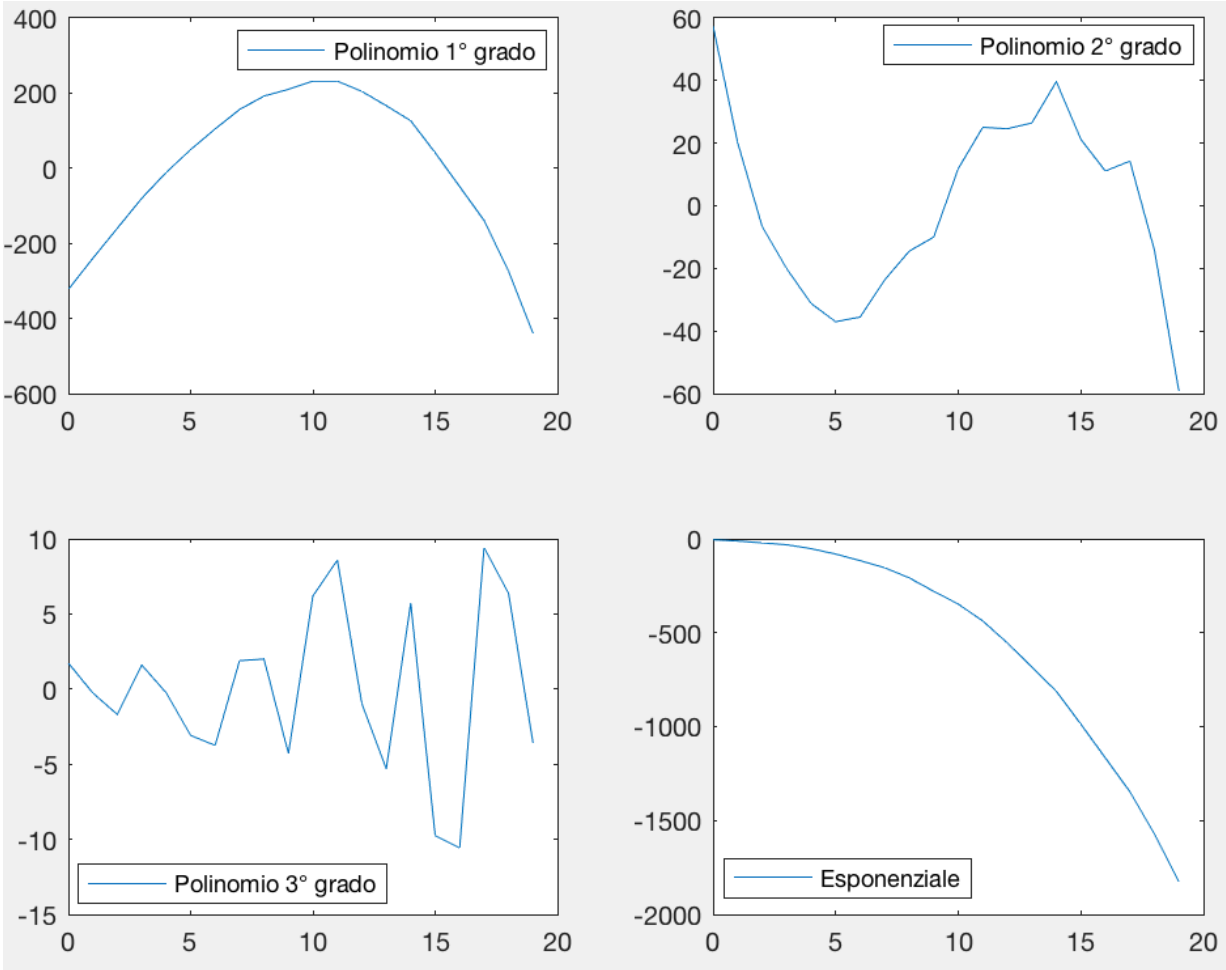
```
>> t = 0:19;
>> y = [6,13,23,33,54,83,118,156,210,282,...
350,440,557,685,815,990,1170,1350,1575,1830];
>> plot(t,y,'*')
>> p1 = polyfit(t,y,1);
>> p2 = polyfit(t,y,2);
>> p3 = polyfit(t,y,3);
>> p4 = polyfit(t,log10(y),1);
>> res1 = polyval(p1,t)-y;
>> res2 = polyval(p2,t)-y;
>> res3 = polyval(p3,t)-y;
>> res4 = polyval(p4,t)-y
```



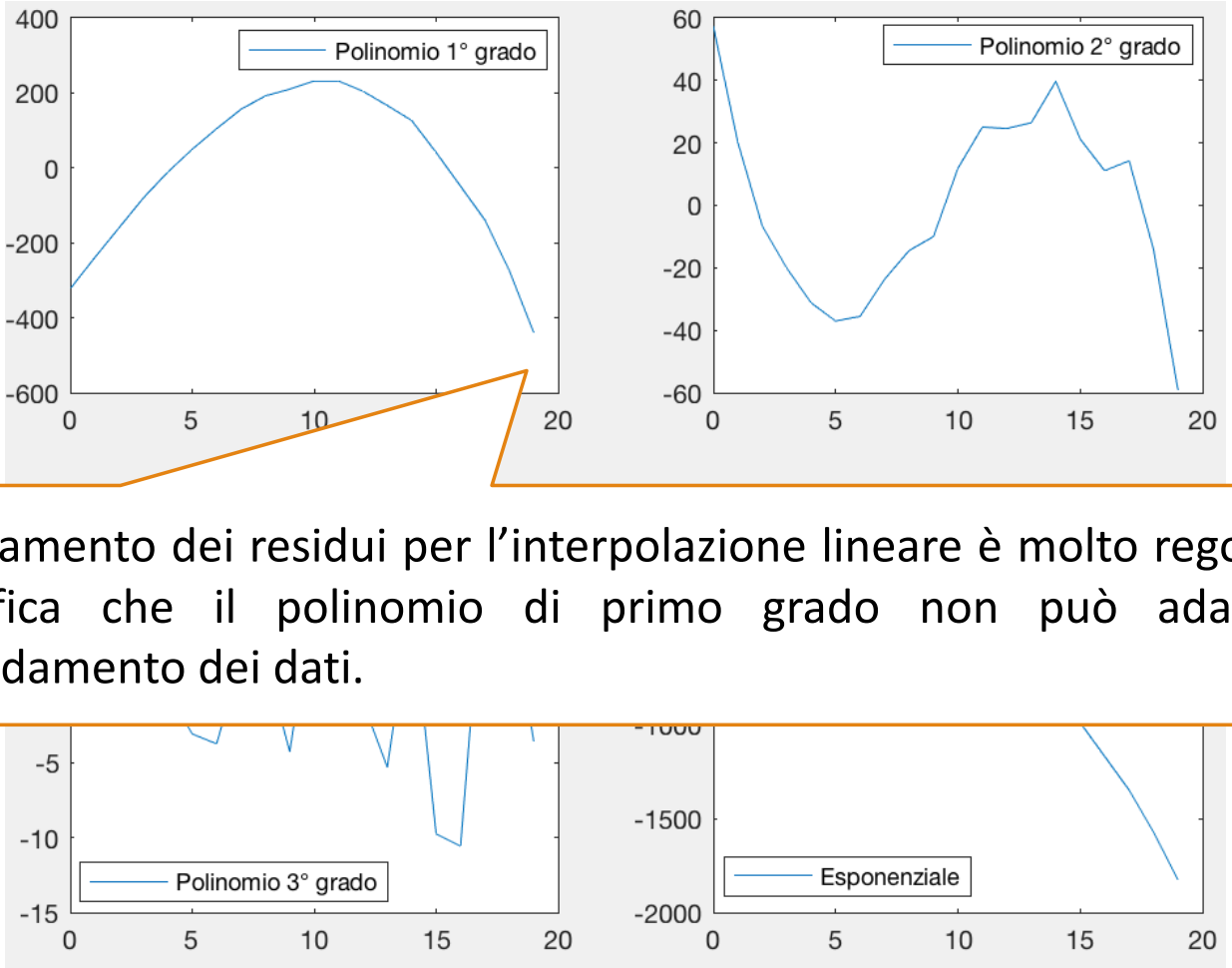
I Residui (3)

```
>> hold on
>> subplot(2,2,1)
>> plot(t,res1)
>> legend('Polinomio 1° grado')
>> subplot(2,2,2)
>> plot(t,res2)
>> legend('Polinomio 2° grado')
>> subplot(2,2,3)
>> plot(t,res3)
>> legend('Polinomio 3° grado')
>> subplot(2,2,4)
>> plot(t,res4)
>> legend('Esponenziale')
```

I Residui (4)

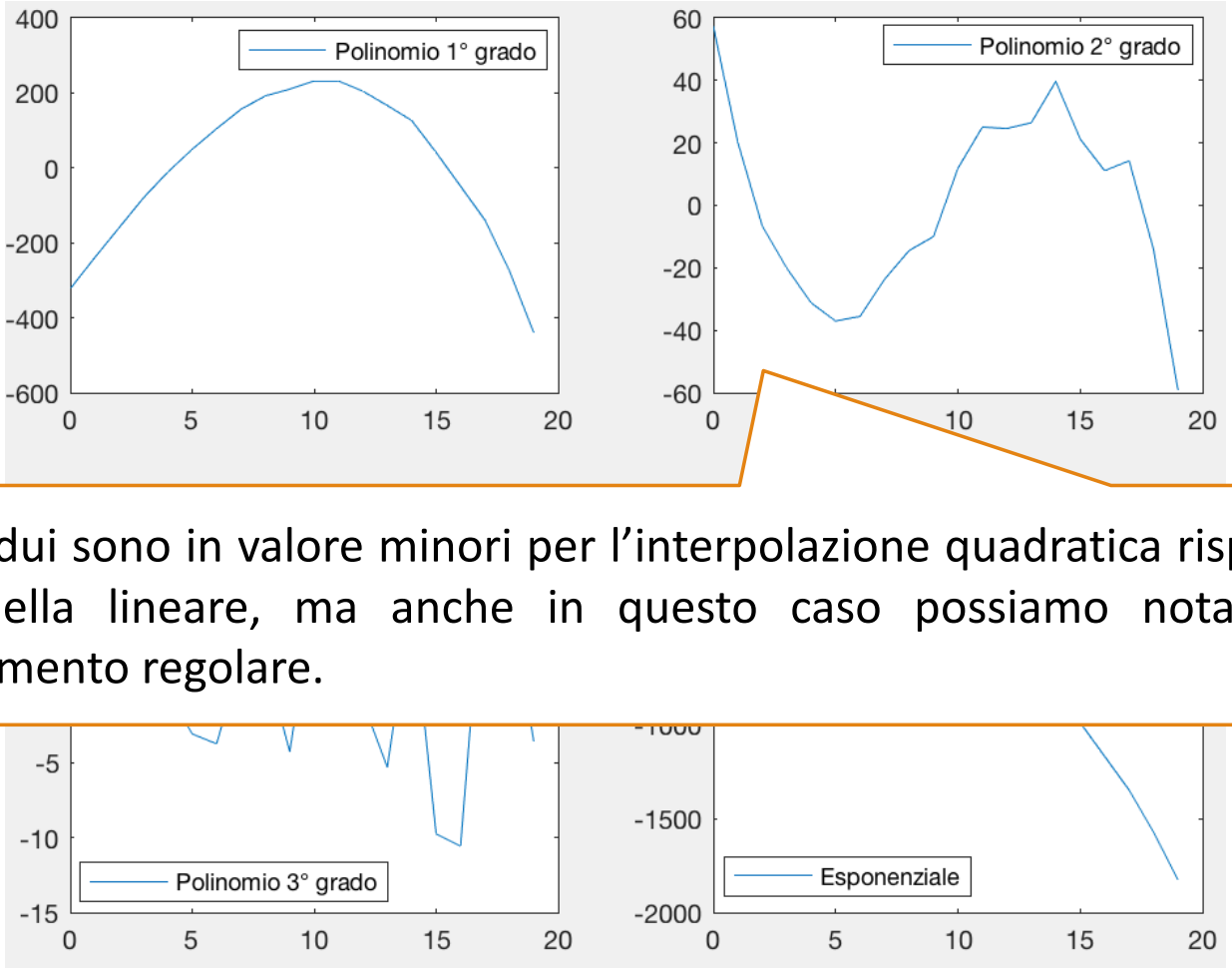

Polinomio 1° grado
Polinomio 2° grado
Polinomio 3° grado
Esponenziale

I Residui (4)



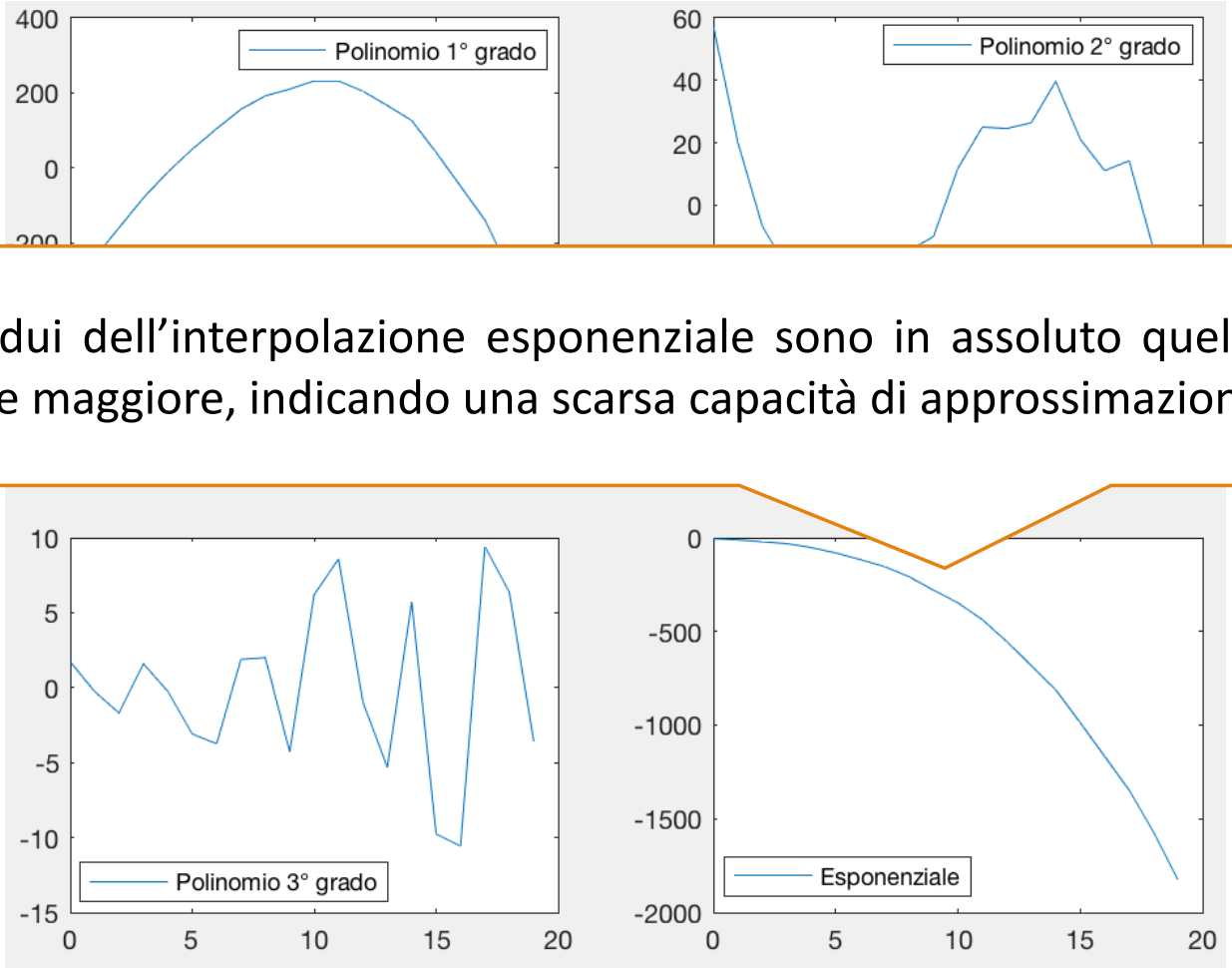
L'andamento dei residui per l'interpolazione lineare è molto regolare. Significa che il polinomio di primo grado non può adattarsi all'andamento dei dati.

I Residui (4)



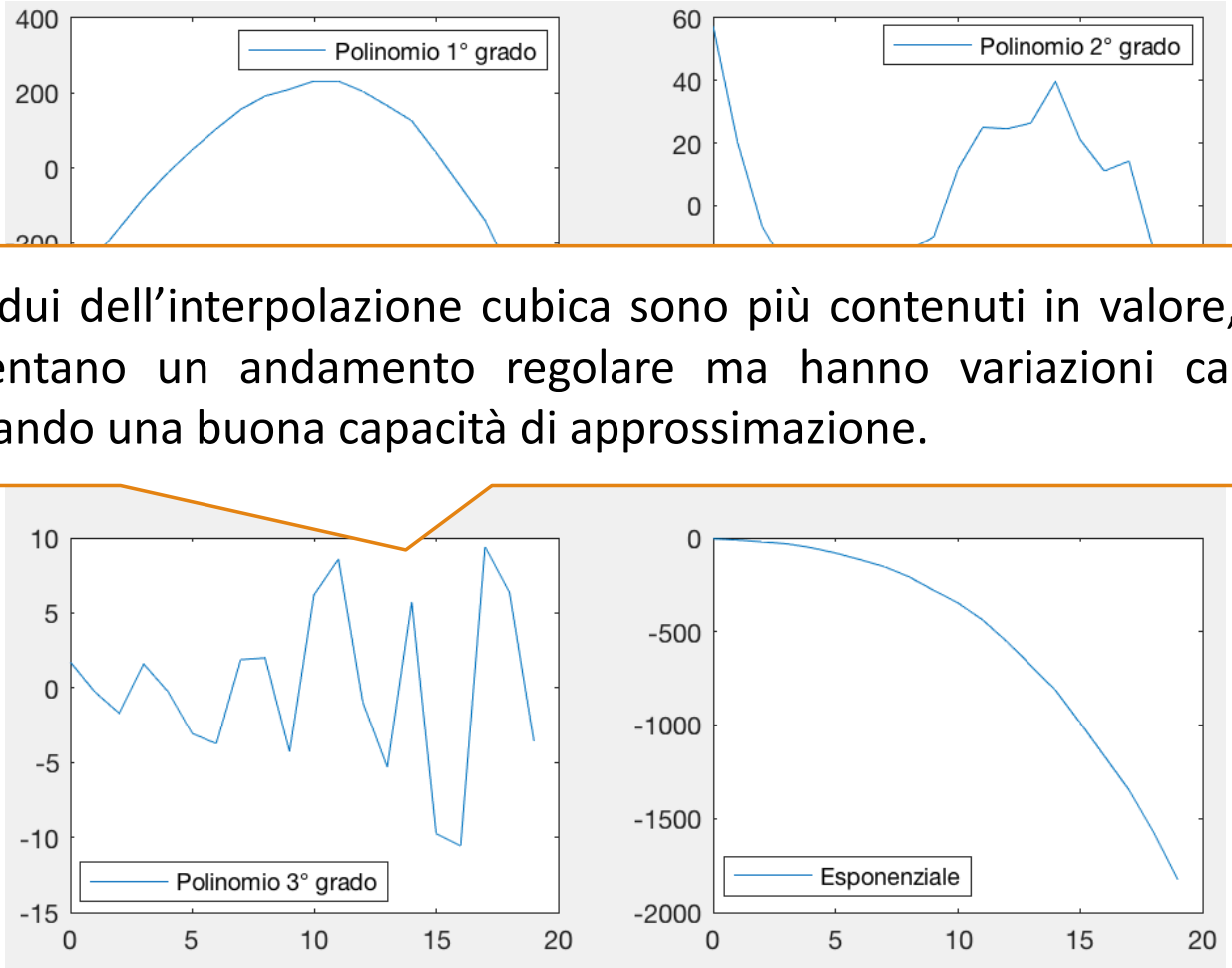
I residui sono in valore minori per l'interpolazione quadratica rispetto a quella lineare, ma anche in questo caso possiamo notare u andamento regolare.

I Residui (4)



I residui dell'interpolazione esponenziale sono in assoluto quelli dal valore maggiore, indicando una scarsa capacità di approssimazione.

I Residui (4)



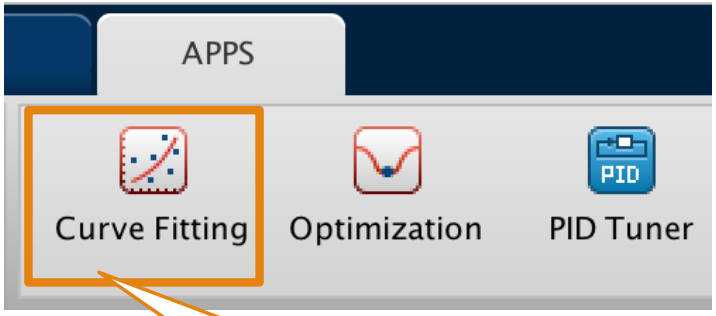
I residui dell'interpolazione cubica sono più contenuti in valore, non presentano un andamento regolare ma hanno variazioni casuali, indicando una buona capacità di approssimazione.

L’interfaccia Basic Fitting (1)

MATLAB dispone dell’interfaccia Basic Fitting che permette di adattare rapidamente delle curve a un insieme di dati, e svolgere le seguenti operazioni:

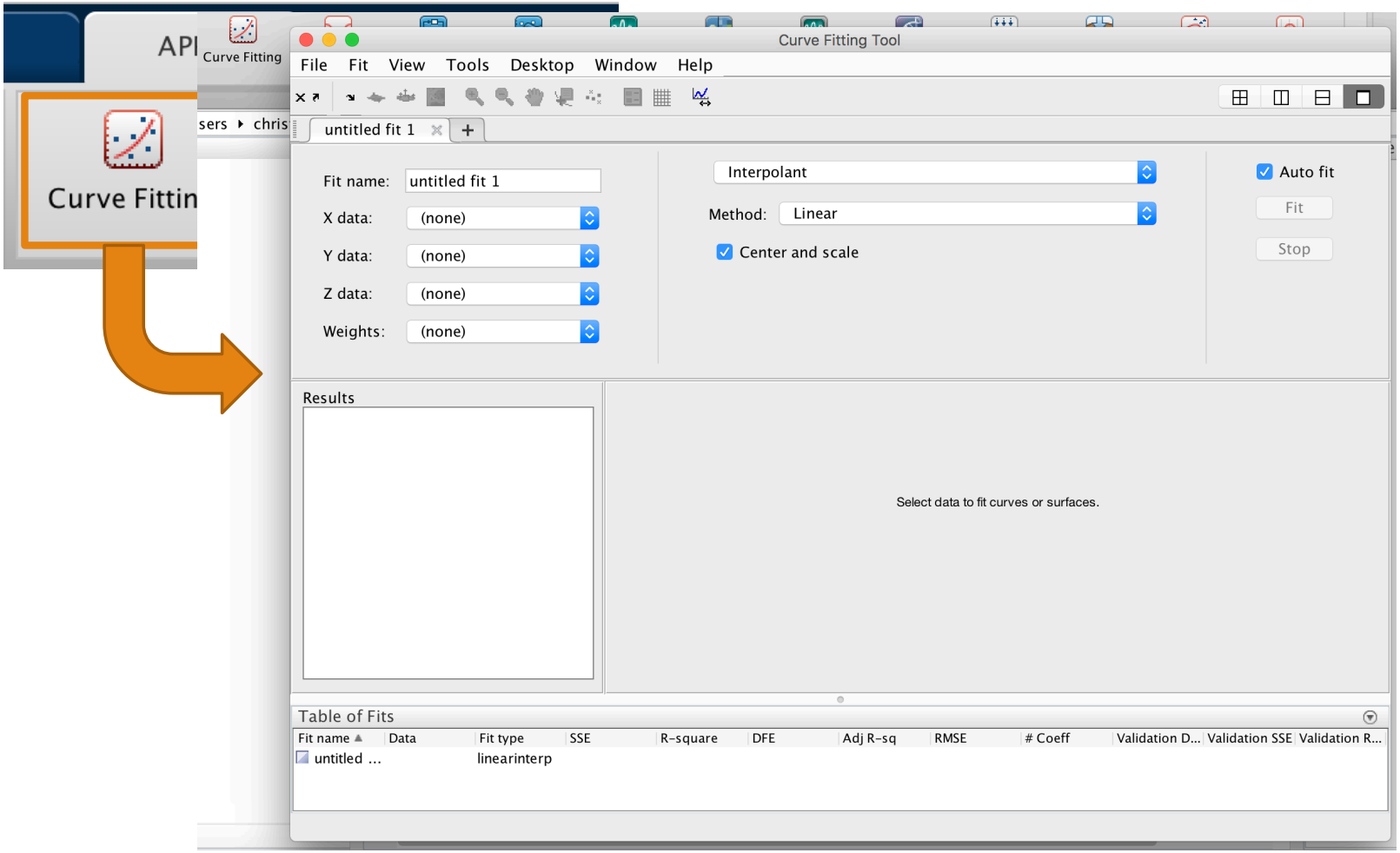
- Approssimare un insieme di dati con funzioni polinomiali o spline;
- Rappresentare in un diagramma più curve che approssimano lo stesso insieme di dati;
- Rappresentare in un diagramma i residui, ovvero l’errore tra il valore approssimato e quello reale;
- Esaminare i risultati numerici dell’approssimazione;
- Salvare il risultato dei calcoli e dei diagrammi ottenuti.

L’interfaccia Basic Fitting (2)

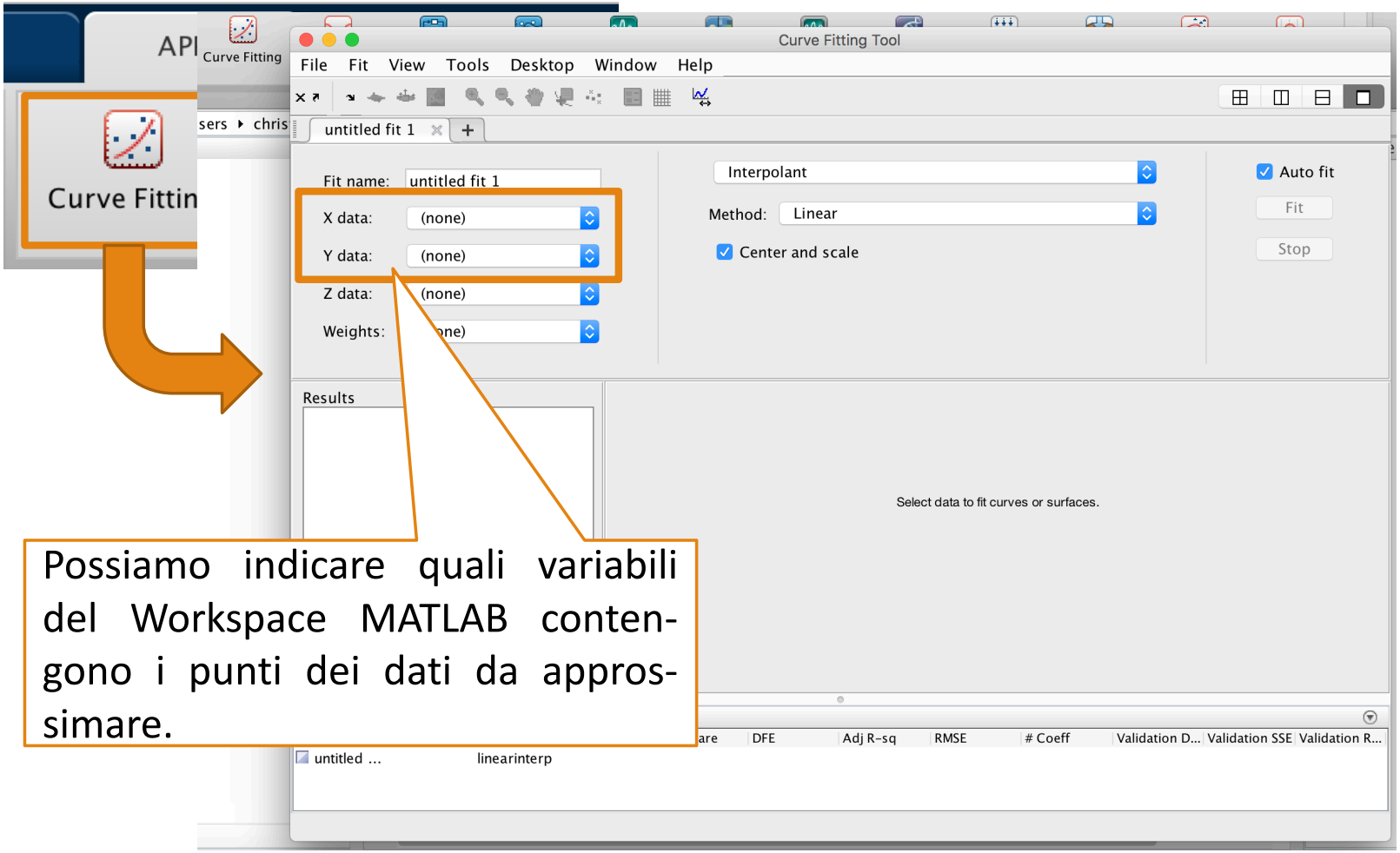


Pulsante per poter invocare l’interfaccia Basic Fitting.

L’interfaccia Basic Fitting (2)

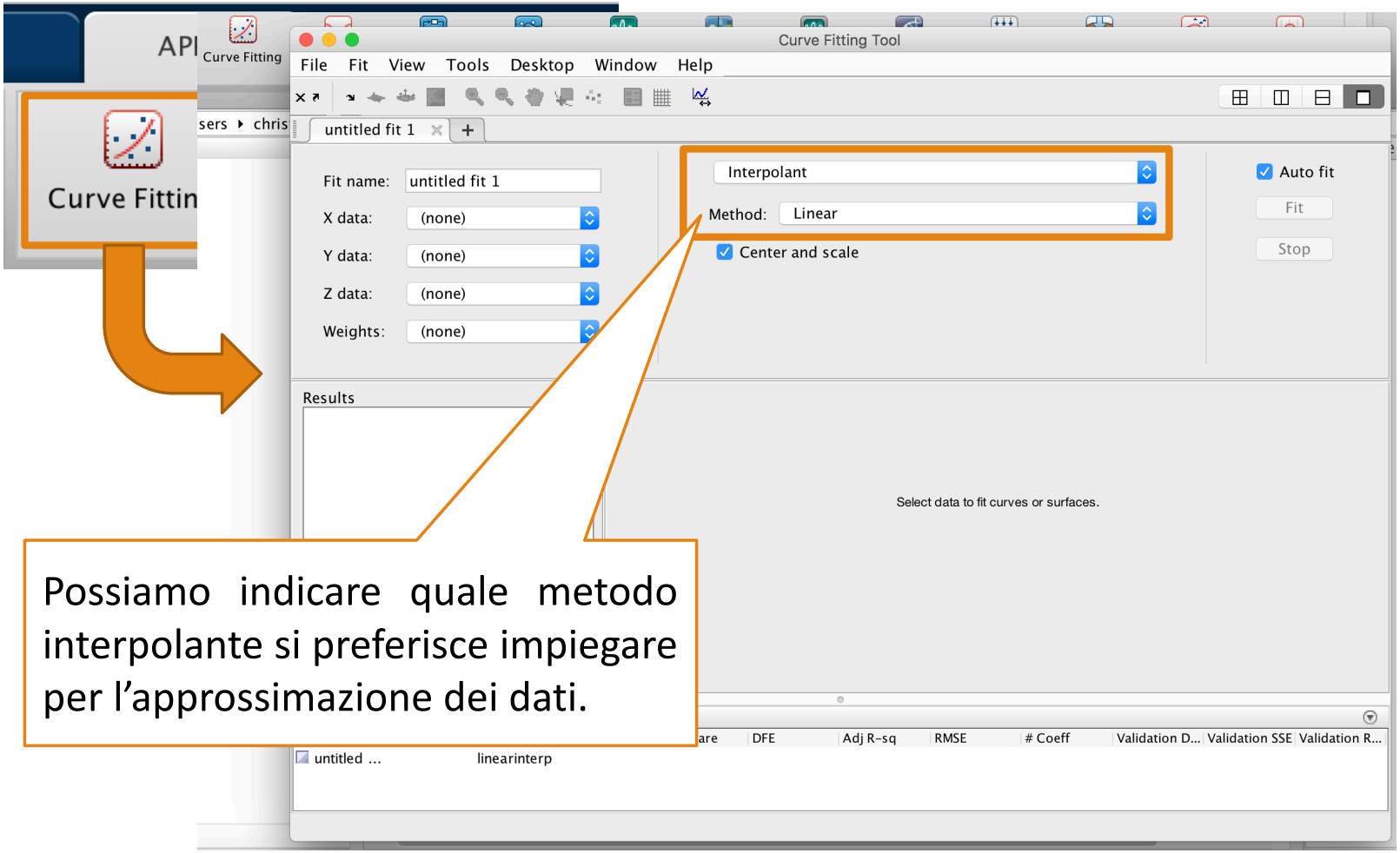


L’interfaccia Basic Fitting (2)



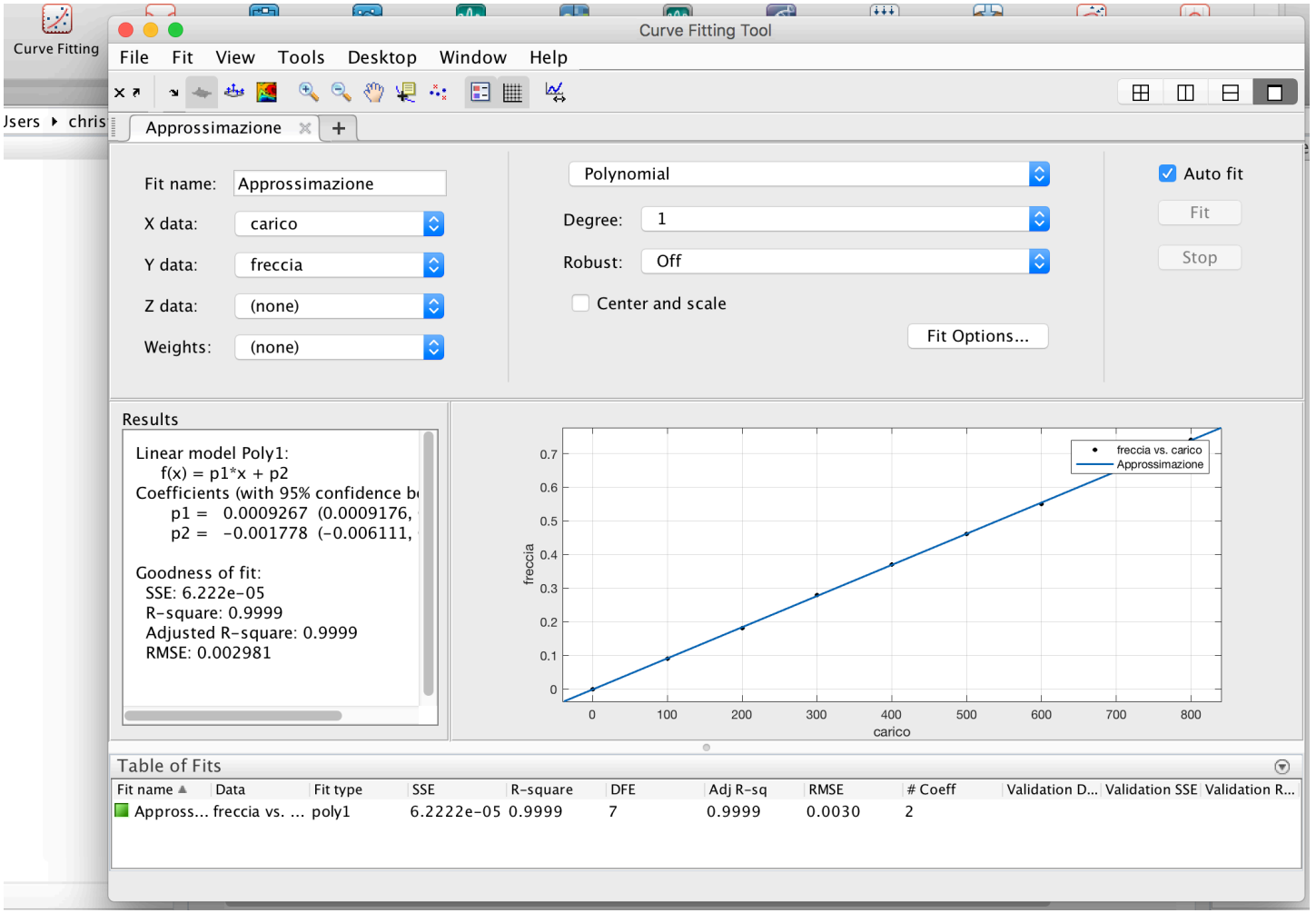
Possiamo indicare quali variabili del Workspace MATLAB contengono i punti dei dati da approssimare.

L’interfaccia Basic Fitting (2)

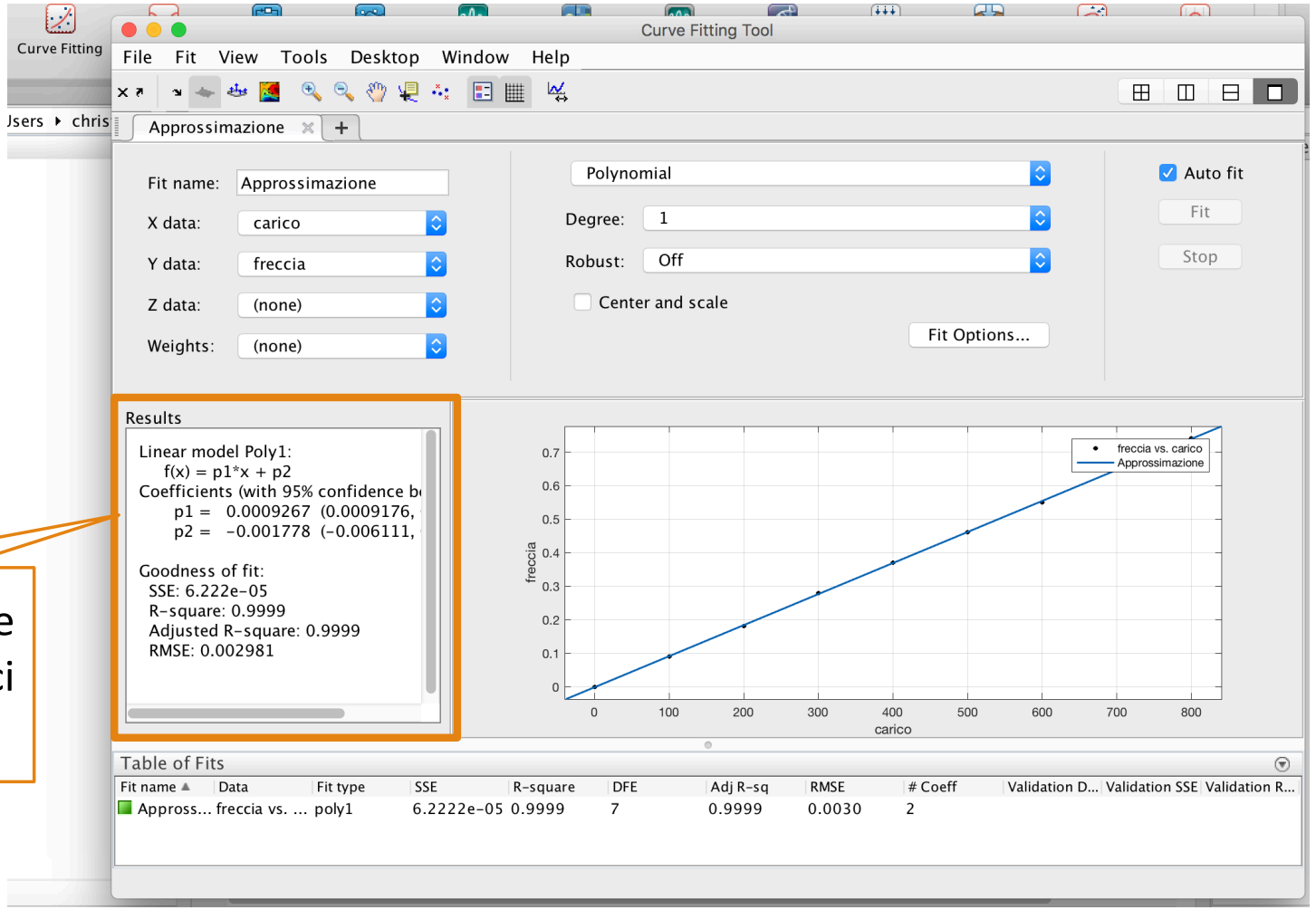


Possiamo indicare quale metodo interpolante si preferisce impiegare per l’approssimazione dei dati.

L’interfaccia Basic Fitting (3)

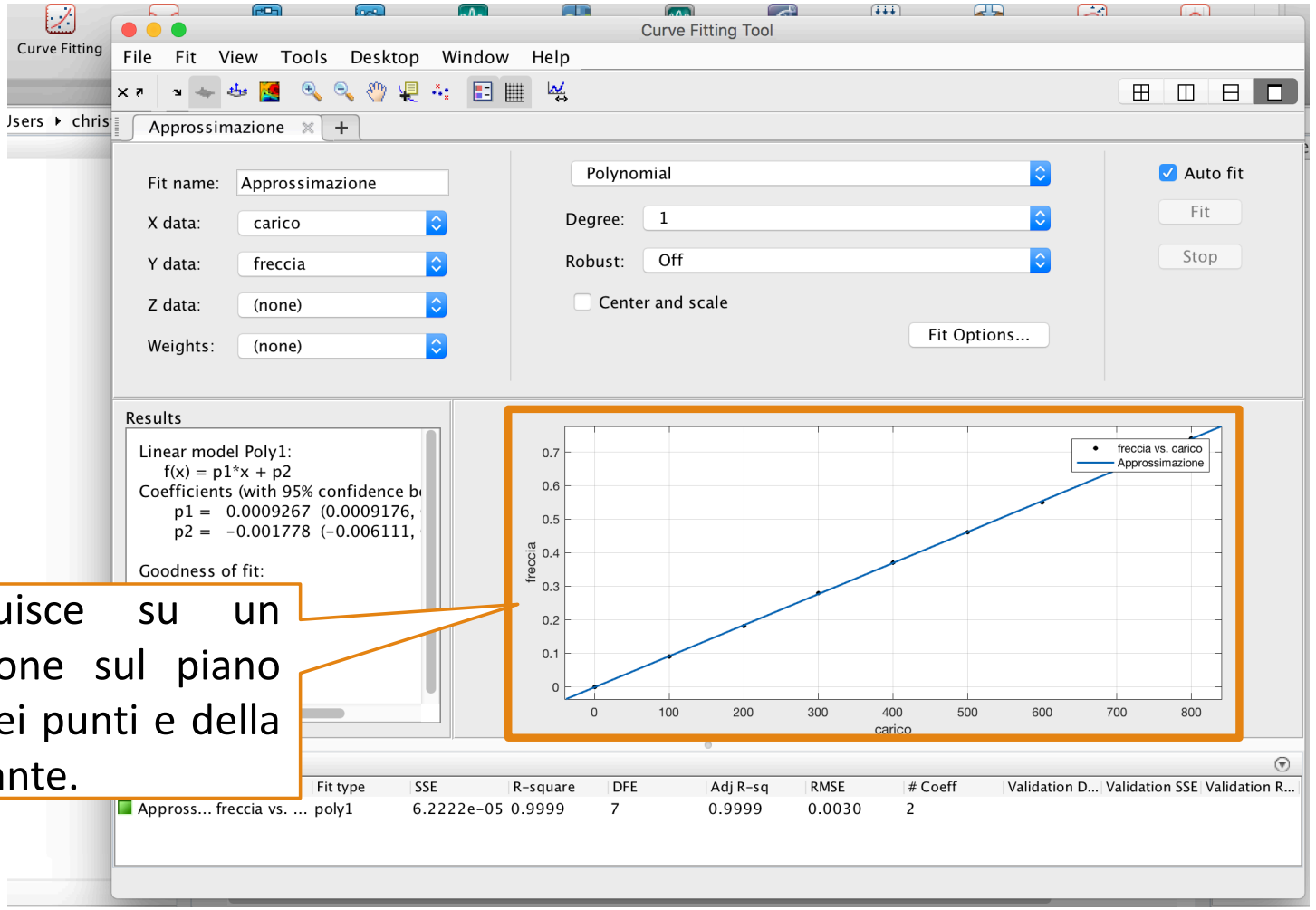


L’interfaccia Basic Fitting (3)



L’interfaccia restituisce i risultati numerici dell’approssimazione.

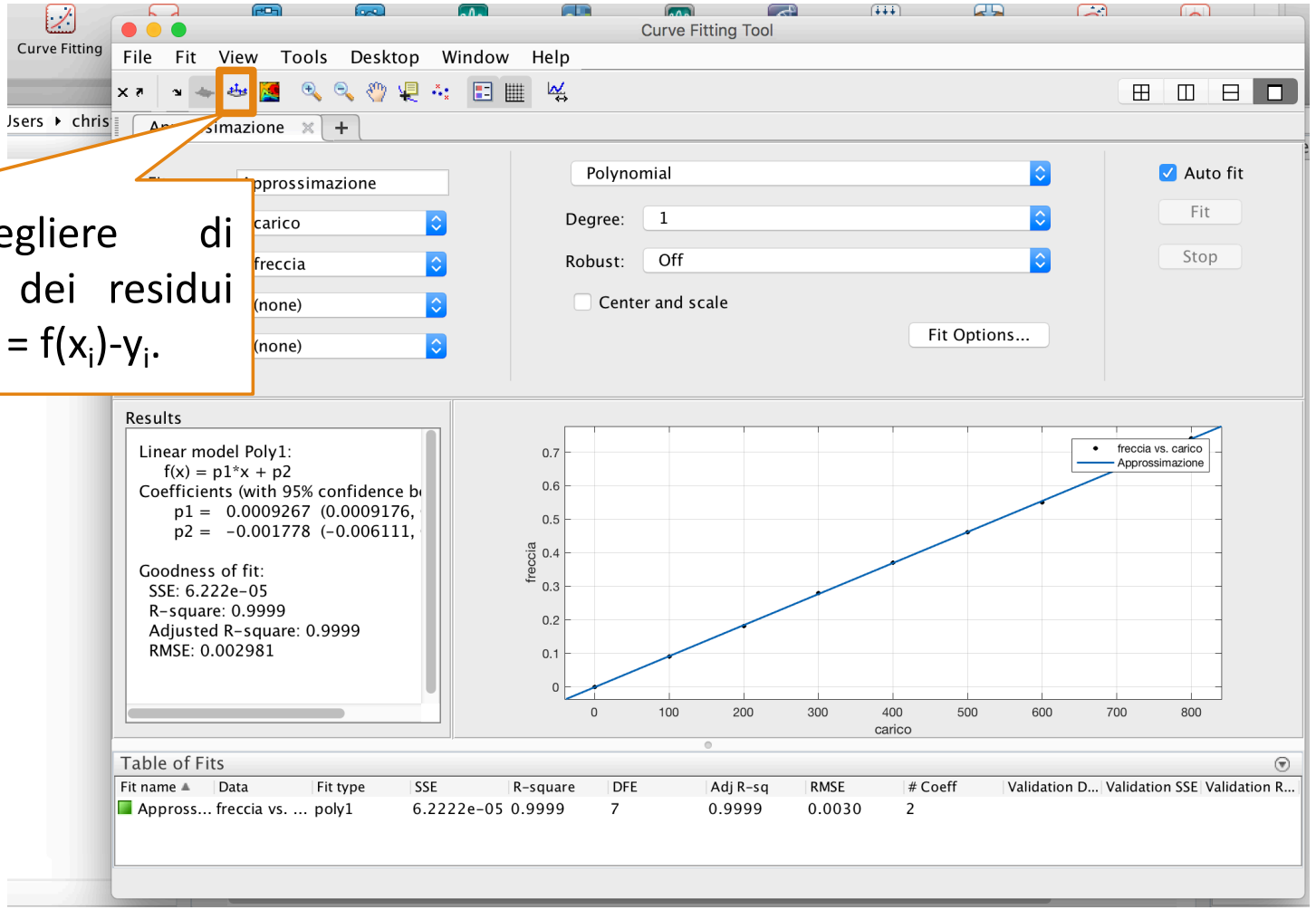
L’interfaccia Basic Fitting (3)



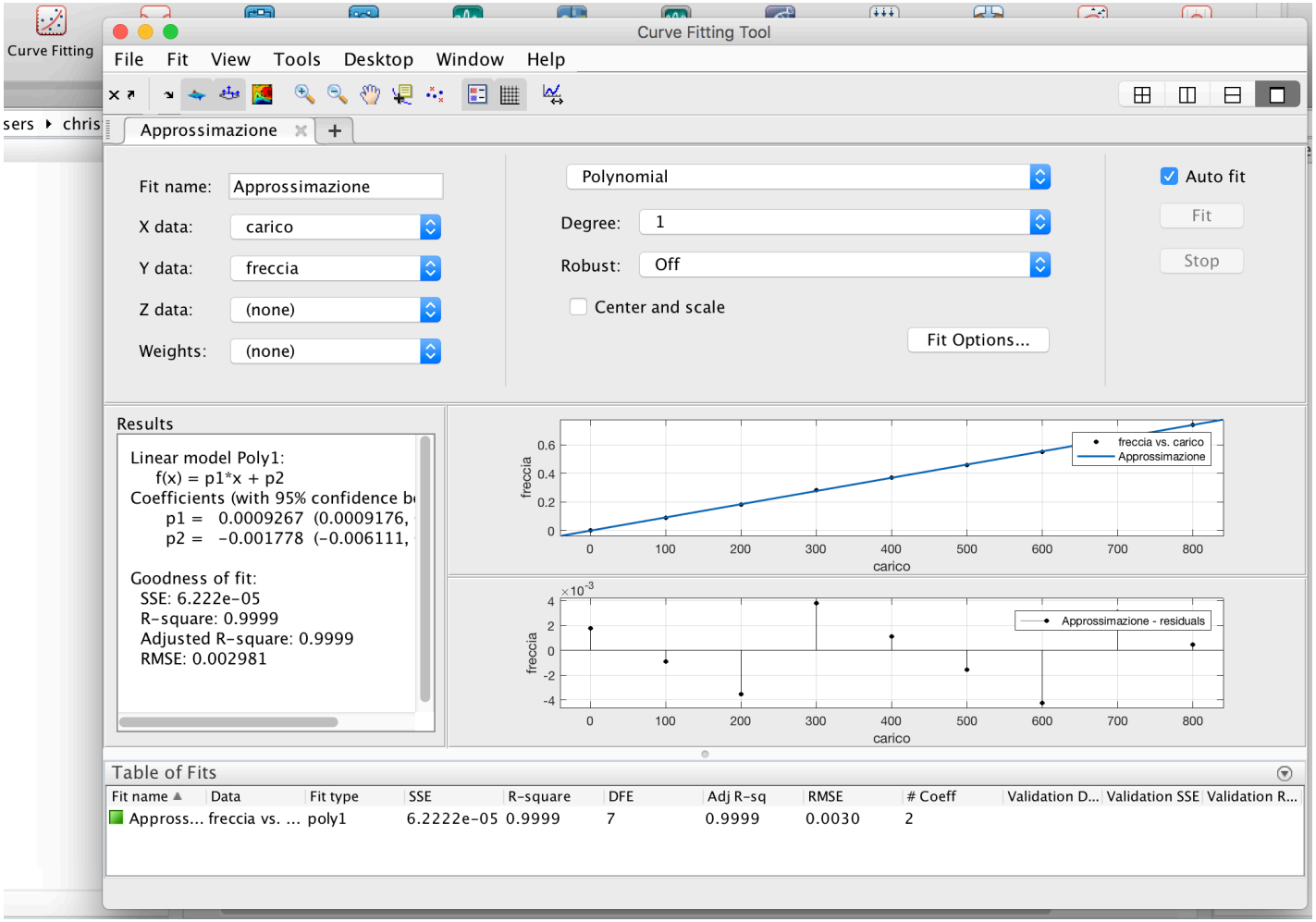
L’interfaccia restituisce su un grafico la disposizione sul piano cartesiano lineare dei punti e della funzione approssimante.

L’interfaccia Basic Fitting (3)

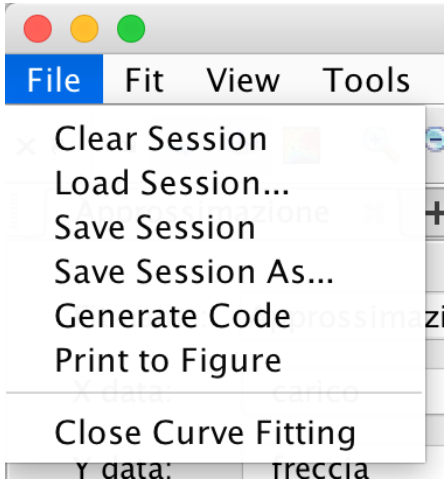
L’utente può scegliere di aggiungere il grafico dei residui dell’approssimazione $r_i = f(x_i)-y_i$.

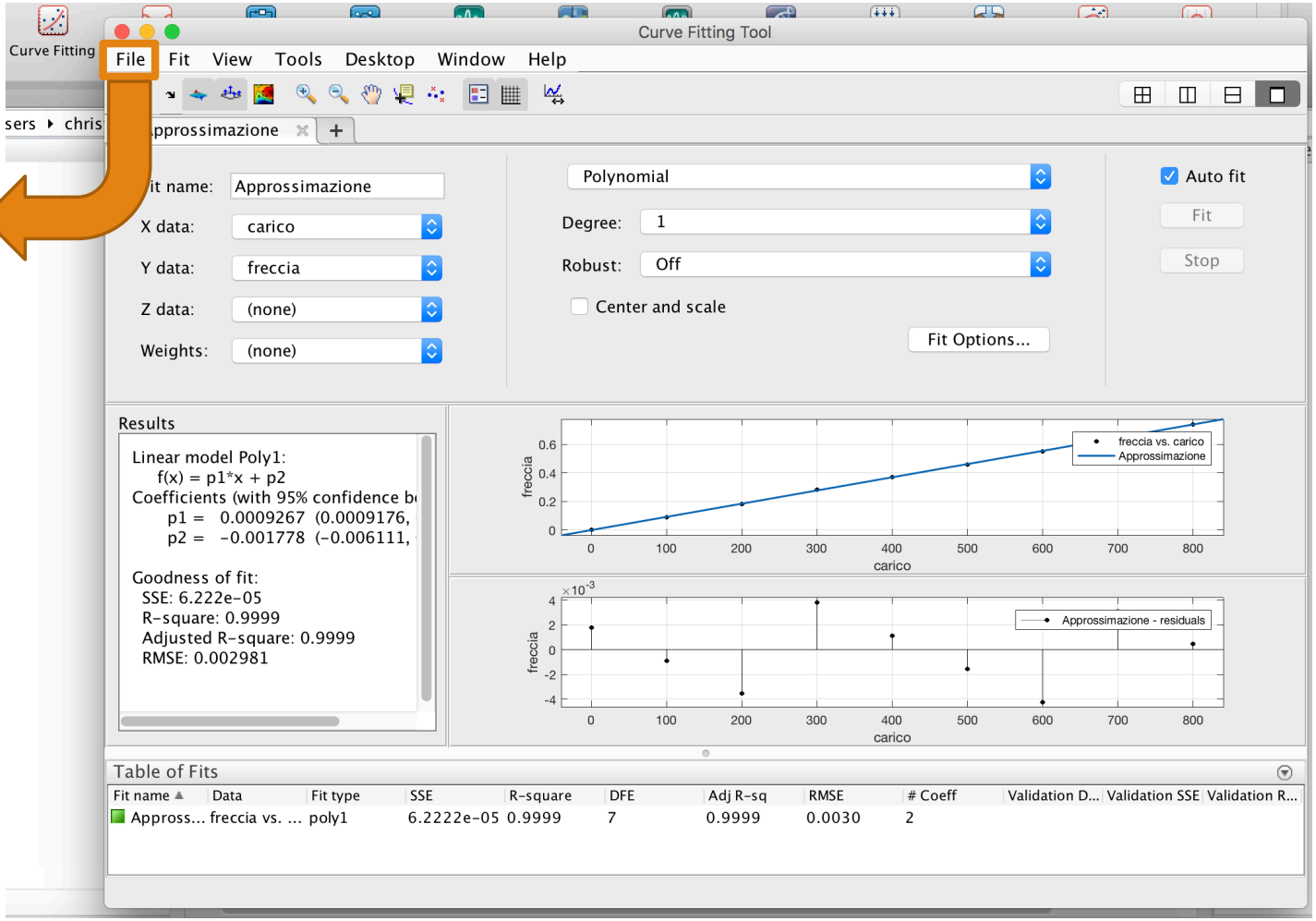


L’interfaccia Basic Fitting (3)

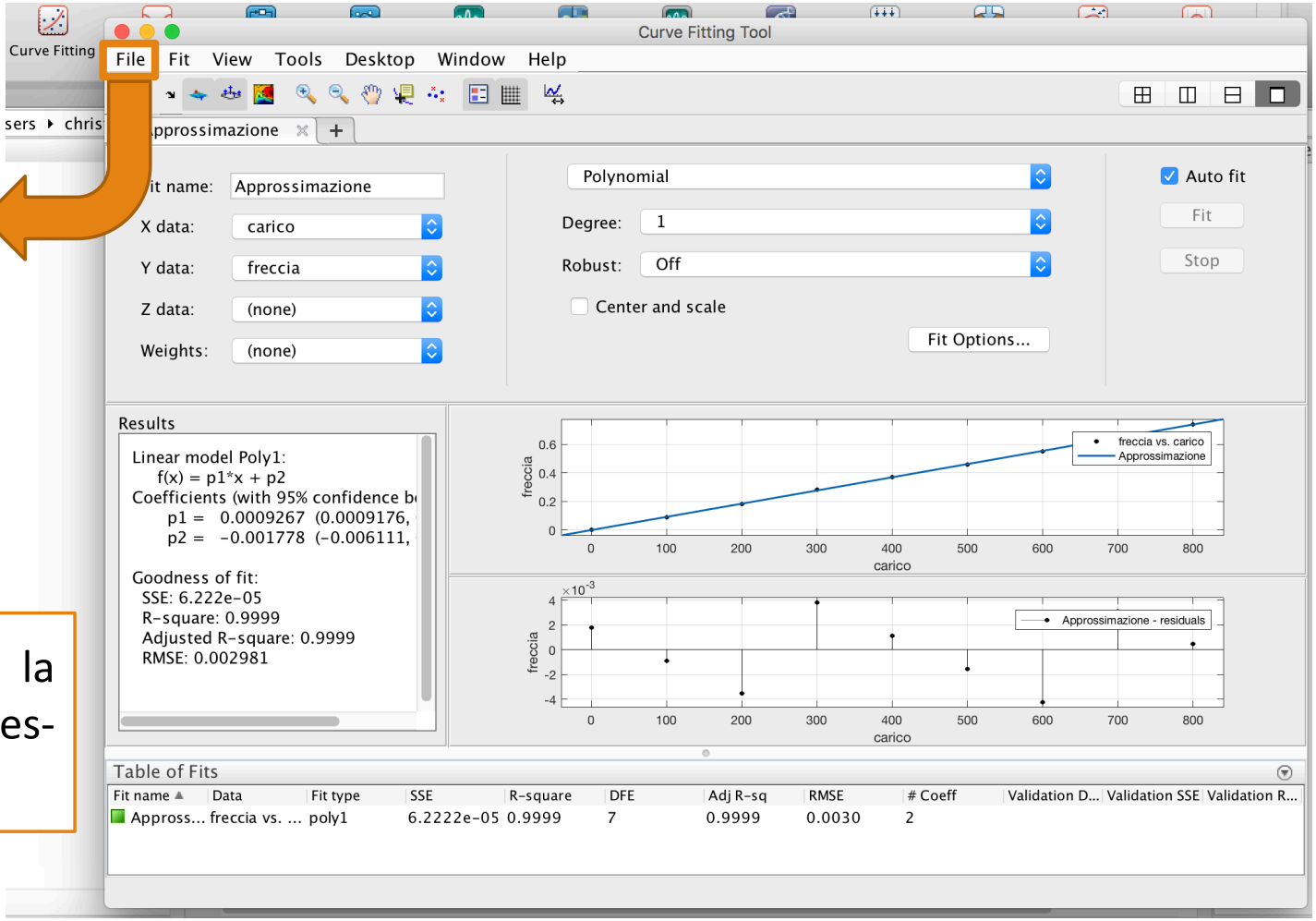


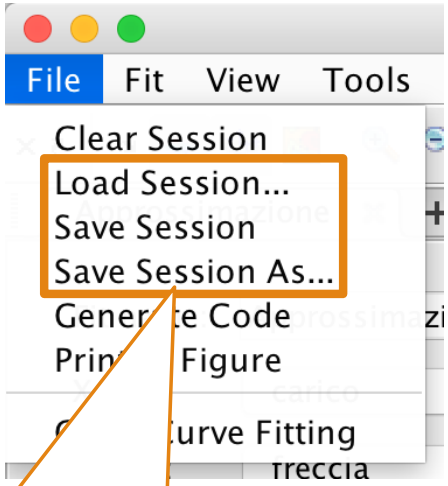
L’interfaccia Basic Fitting (3)





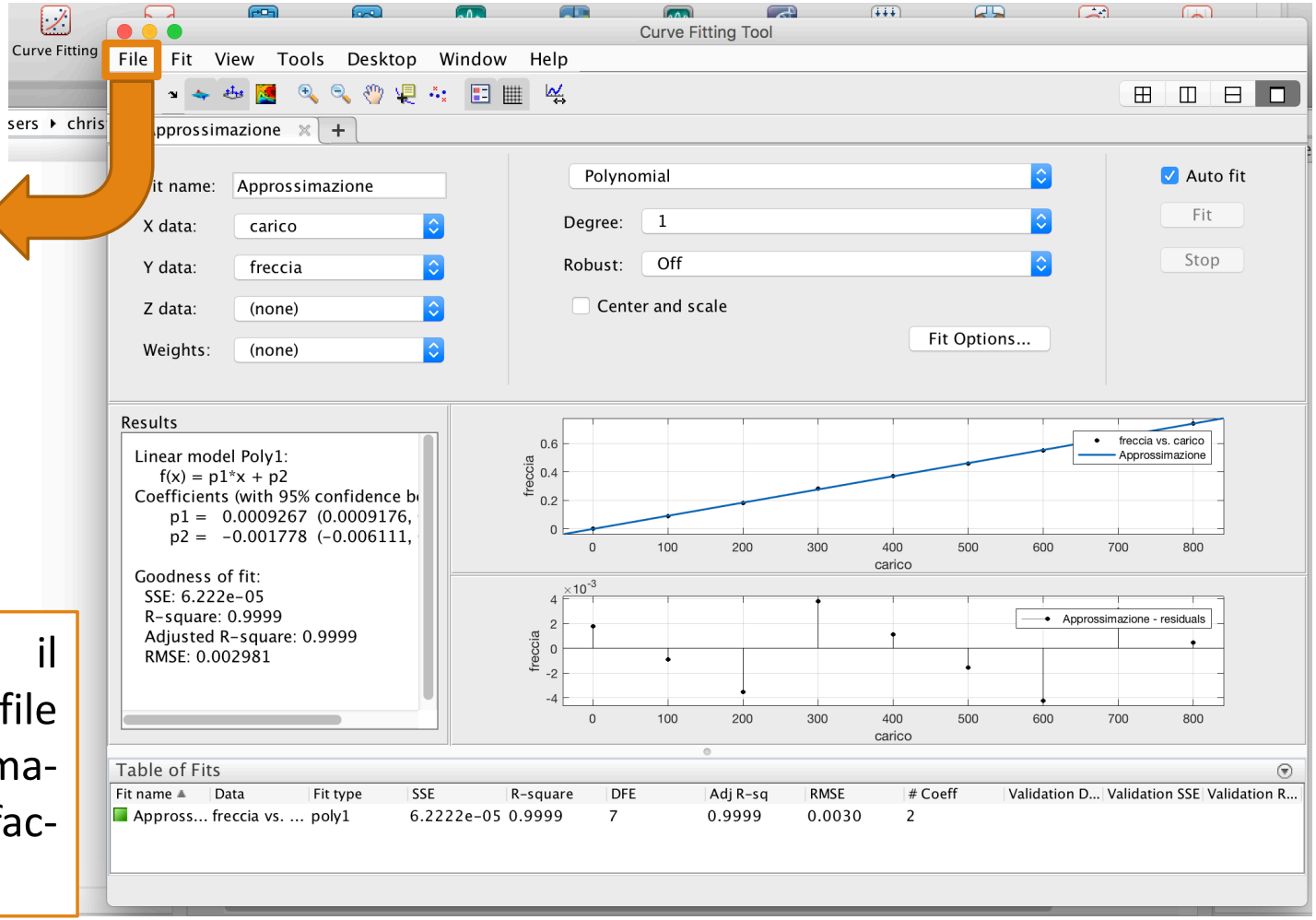
L’interfaccia Basic Fitting (3)





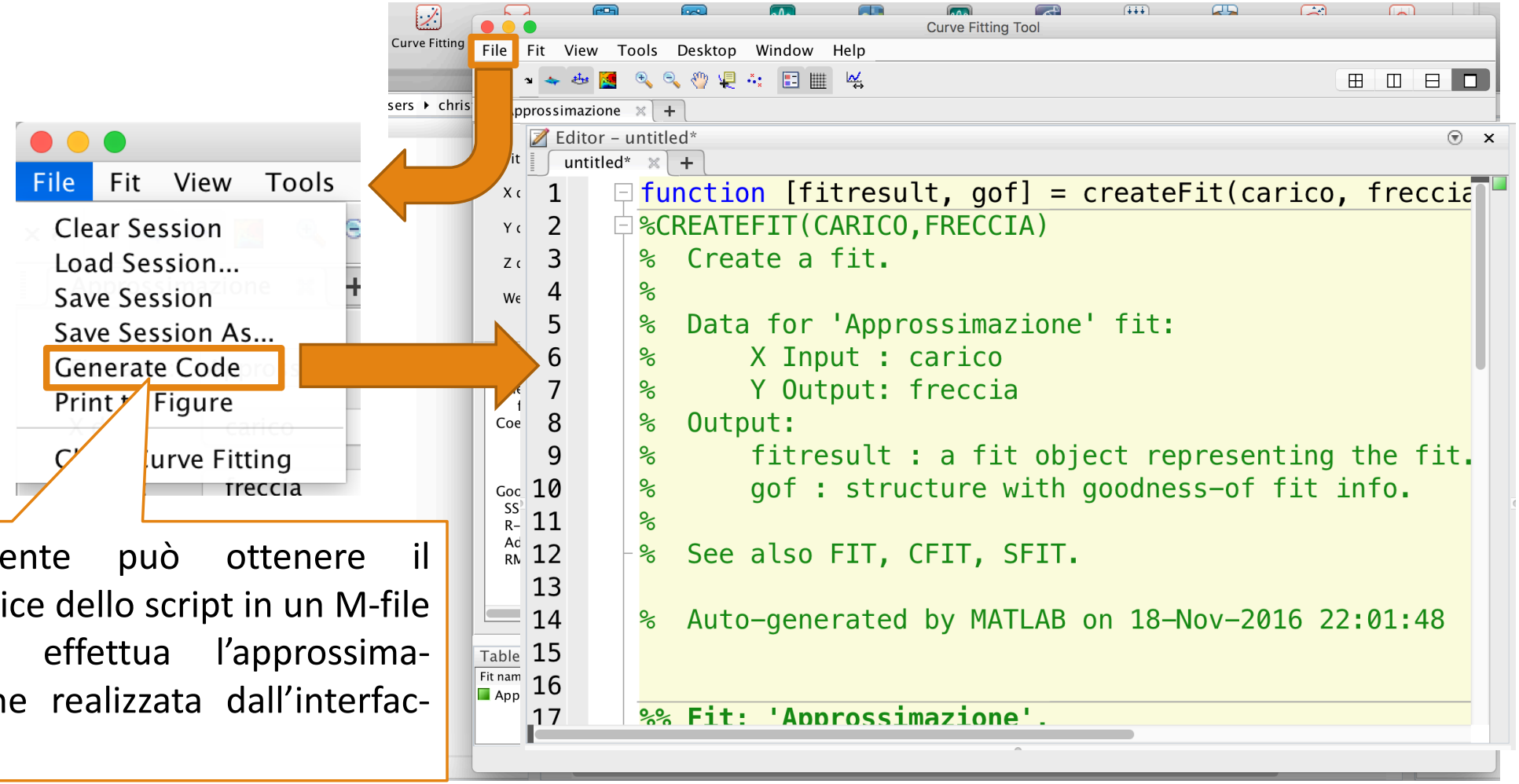
L’utente può salvare la sessione e caricarla succes-sivamente.

L’interfaccia Basic Fitting (3)



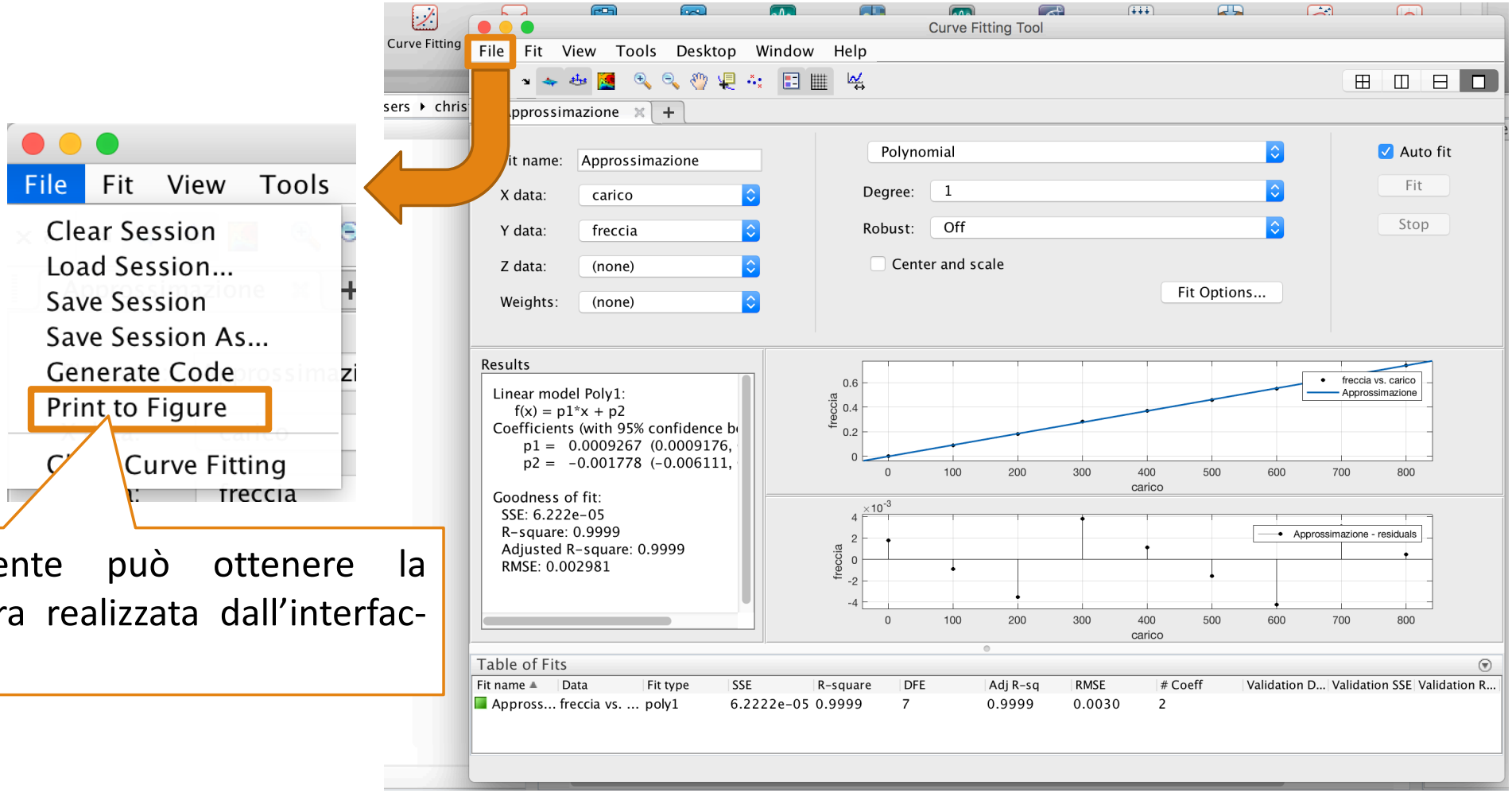
L’utente può ottenere il codice dello script in un M-file che effettua l’approssimazione realizzata dall’interfaccia.

L'interfaccia Basic Fitting (3)



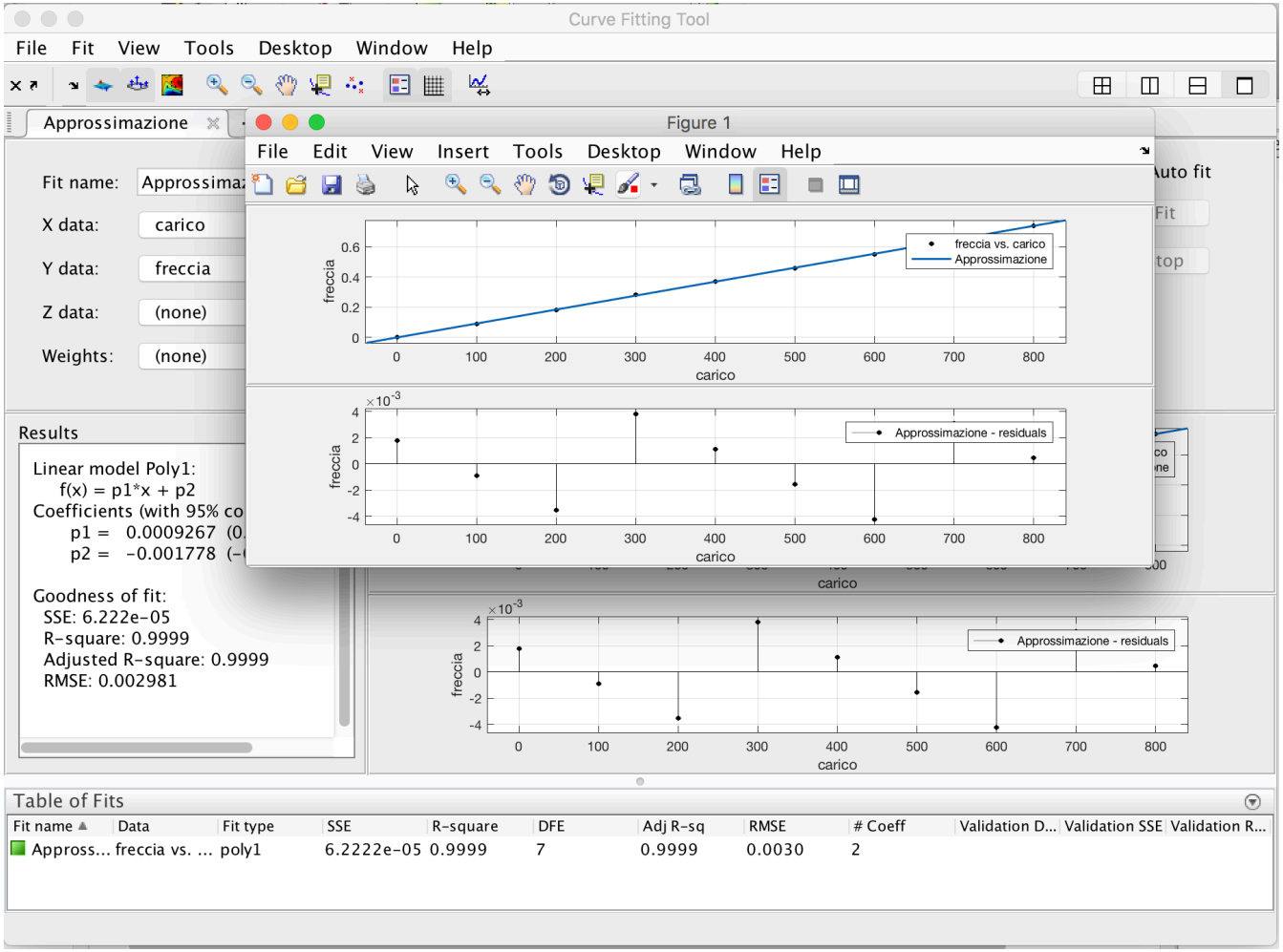
L'utente può ottenere il codice dello script in un M-file che effettua l'approssimazione realizzata dall'interfaccia.

L’interfaccia Basic Fitting (3)



L’utente può ottenere la figura realizzata dall’interfaccia.

L’interfaccia Basic Fitting (4)



Riferimenti

- Capitolo 5
  - Paragrafi 1 [**Diagrammi xy**], e 2 [**Altri comandi e tipi di diagrammi**].
- Capitolo 6
  - Paragrafi 1 [**Ricerca di funzione**], 2 [**Regressione**] e 3 [**L'interfaccia Basic Fitting**].
- Capitolo 7
  - Paragrafi 1 [**Statistica e Istogrammi**], e 4 [**Interpolazione**].