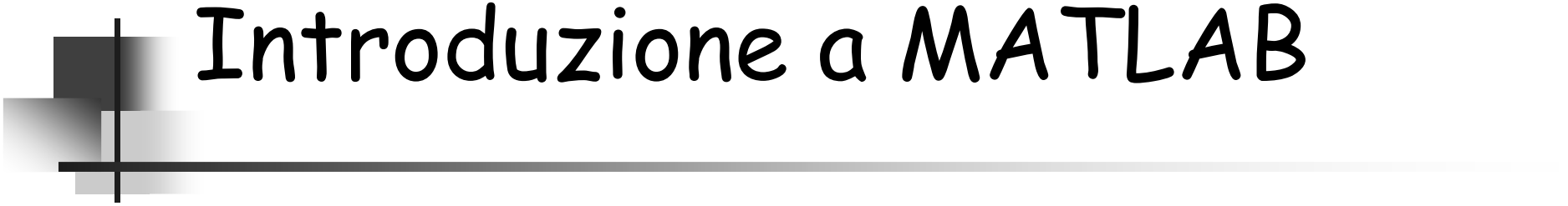
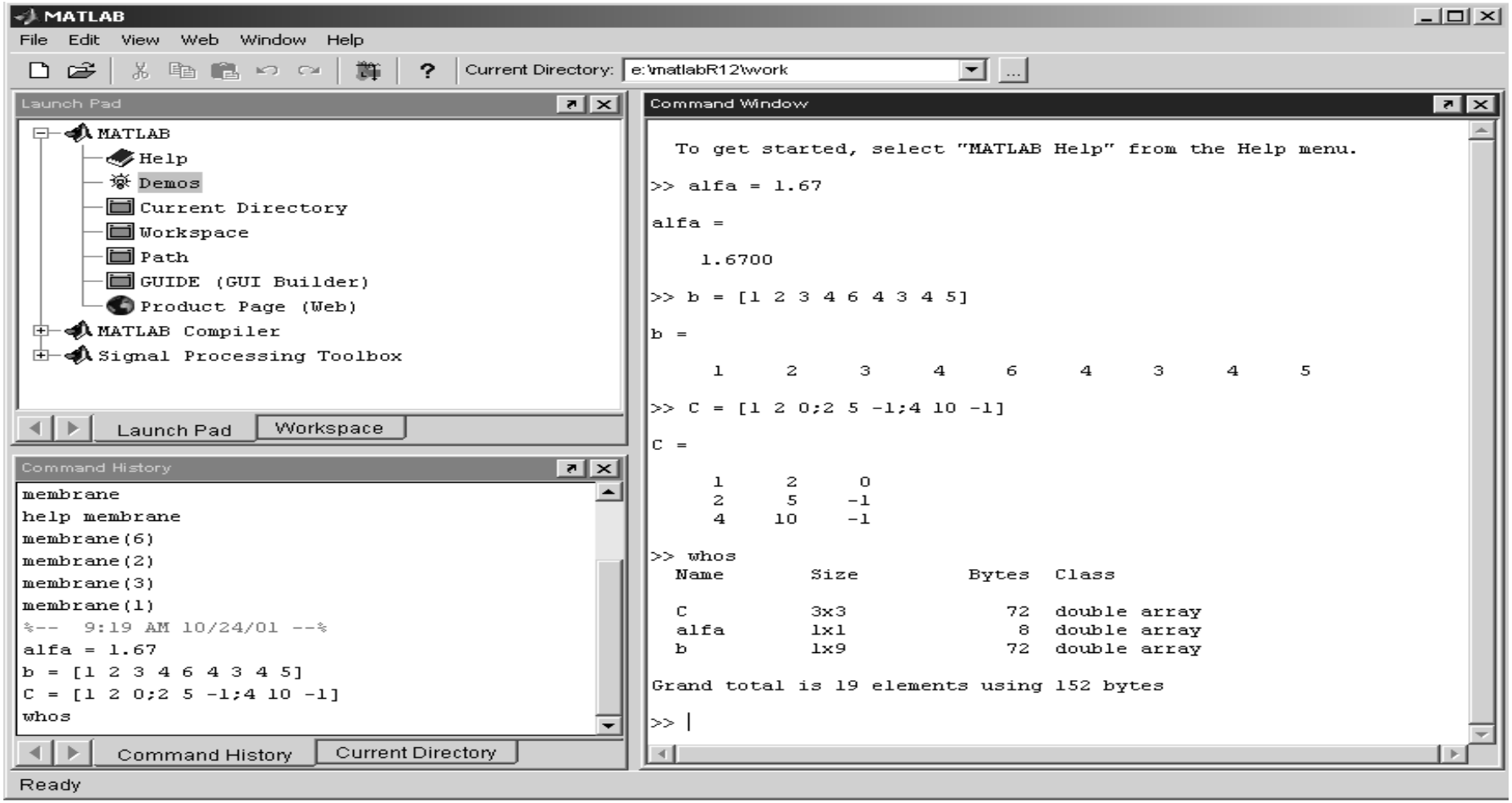


Introduzione a MATLAB

MATLAB=MATrix LABoratory

- È un sistema in cui ogni dato è rappresentato sotto forma di una matrice di numeri.
- Permette di:
  - Effettuare operazioni tra matrici.
  - Rappresentare in forma grafica il contenuto delle matrici.

Interfaccia utente



Linea di comando

La linea di comando di MATLAB è indicata da un prompt come in DOS: >> .

Accetta dichiarazioni di variabili, espressioni e chiamate a tutte le funzioni disponibili nel programma.

Tutte le funzioni di MATLAB non sono altro che files di testo, simili a quelli che l'utente può generare con un text editor, e vengono eseguite semplicemente digitandone il nome sulla linea di comando.

MATLAB permette inoltre di richiamare le ultime righe di comandi inseriti usando le frecce in alto e in basso.

Help di Matlab

MATLAB presenta un help in linea con informazioni sulla sintassi di tutte le funzioni disponibili.

Per accedere a queste informazioni, basta digitare:

help nome_funzione

È anche possibile avere un help di tutte le funzioni di una certa categoria; ad esempio per sapere quali sono le funzioni specifiche per l'analisi ed il controllo di sistemi dinamici, basta digitare:

help control

Per sapere quali sono le varie categorie di funzioni disponibili (i cosiddetti *toolbox*), basta digitare:

help

I files di Matlab

I files interpretati dal programma sono file di testo ASCII con estensione .m ; sono generati con un text editor e sono eseguiti in MATLAB semplicemente digitandone il nome sulla linea di comando (senza estensione!).

È possibile inserire dei commenti al loro interno precedendo ogni linea di commento col percento %

Attenzione! Può essere molto utile andare nelle directories dove si trova il programma ed analizzare come le varie funzioni sono state implementate.

Ciò è possibile poichè ogni funzione ed ogni comando MATLAB richiama un file .m

Punteggiatura

Le istruzioni (siano esse contenute in un file .m lanciato da MATLAB, oppure digitate direttamente dalla linea di comando) vanno normalmente terminate con un punto e virgola, altrimenti è visualizzato il risultato dell'applicazione dell'istruzione.

```
Es: var1=6;
Es: var2=linspace(-10,10,10000);
```

Nomi delle variabili

Le variabili seguono le regole dei linguaggi di programmazione come il Pascal o il C. MATLAB è

case-sensitive

e accetta nomi di variabili lunghi fino ad un massimo di 19 caratteri alfanumerici, con il primo obbligatoriamente alfabetico. Sono nomi leciti

x , x1, X1, ossido, silice

NON sono leciti:

1X, nome_troppo_lungo_per_essere_valido

Visualizzazione risultato

Per visualizzare il contenuto di una variabile è sufficiente digitarne il nome senza punto e virgola sulla linea di comando.

Tutti i calcoli effettuati in MATLAB sono eseguiti in doppia precisione.

Il risultato dell'ultima operazione è memorizzato nella variabile ans.

Esempio 1

- Matrice 1x1, è composta da 1 riga e da 1 colonna:

```
>> alfa = 1.67
alfa =
        1.6700
```

Esempio 2

- Matrice 1x9, è composta da 1 riga e da 9 colonne:

```
>> b = [1 2 3 4 6 4 3 4 5]
b =
     1 2 3 4 6 4 3 4 5
```

Esempio 3

- Matrice 3x3, è composta da 3 righe e da 3 colonne:

```
>> C = [1 2 0; 2 5 –1;4 10 -1]
C =
     1    2    0
     2    5   –1
     4   10   -1
```

Comando whos

- Digitando **whos** si ottiene il riepilogo dei dati introdotti dall'utente.

```
>> whos
  Name      Size            Bytes  Class
  C           3x3                72  double array
  alfa        1x1                 8  double array
  b           1x9                72  double array
```

Comando whos

- Digitando **whos** si ottiene il riepilogo dei dati introdotti dall'utente.

```
>> whos
  Name      Size            Bytes  Class
  C           3x3                72  double array
  alfa        1x1                 8  double array
  b           1x9                72  double array
```

Ogni numero è memorizzato in floating point in doppia precisione ed occupa 8 bytes

Operazioni aritmetiche

- Potenza: ^
- Prodotto: *
- Divisone: /
- Somma: +
- Sottrazione: -

Esempio

```
>> c = [ 1 2 3]
>> d = c+2
d =
     3 4 5
>> e = d*1.5
e =
    4.5000  6.0000  7.5000
```

Operazioni su matrici

- Somma di matrici: +
- Sottrazione tra matrici: -
- Le due matrici devono avere le stesse dimensioni:
  - Due matrici 2x2 possono essere sommate
  - Una matrice 3x3 non può essere sommata ad una matrice 2x2.

Esempio 1

```
>> a=[1 2;3 4]
a =
     1     2
     3     4
>> b = [3 4;5 6]
b =
     3     4
     5     6
```

```
>> c = a+b
c =
     4     6
     8    10
```

Esempio 2

```
>> a=[1 2;3 4]
a =
     1     2
     3     4
>> b = [3 4 5;5 6 7;0 0 0]
b =
     3     4     5
     5     6     7
     0     0     0
```

```
>> c = a+b
??? Error using ==> +
Matrix dimensions
  must agree.
```

Prodotto di due matrici

- Si effettua utilizzando *
- È necessario che le dimensioni delle due matrici siano concordi:
  - È possibile moltiplicare una matrice 2x1 per una matrice 1x2, ottenendo una matrice 2x2.
  - Non è possibile moltiplicare una matrice 1x2 per una matrice 1x2.

Esempio 1

```
>> a=[1;2]
a =
     1
     2
>> b =[3 4]
b =
     3     4
```

```
>> c = a*b
c =
     3     4
     6     8
```

Esempio 2

```
>> c=a*a
??? Error using ==> *
Inner matrix dimensions must agree.
```

Operazioni elemento per elemento

- MATLAB permette di eseguire operazioni tra gli elementi di due matrici che abbiamo identica dimensione:
  - Prodotto elemento per elemento: `.*`
  - Divisione elemento per elemento: `./`
  - Potenza elemento per elemento: `.^`

Esempio

```
>> a=[4 16 256]
a =
     4    16   256
>> b=[0.5 0.5 0.25]
b =
    0.5000    0.5000    0.2500
>> c = a.^b
c =
     2     4     4
```

Funzioni predefinite

- Radice quadrata: **`sqrt(x)`**
- Arrotondamento all'intero più vicino: **`round(x)`**
- Parte intera di un numero: **`fix(x)`**
- Segno di un numero: **`sign(x)`**
- Funzioni trigonometriche: **`sin(x), cos(x),…`**
- Funzioni iperboliche: **`sinh(x), cosh(x),…`**
- Funzioni trigonometriche inverse: **`asin(x),…`**
- Esponenziale: **`exp(x)`**
- Logaritmo naturale: **`log(x)`**
- Logarimo in base 10: **`log10(x).`**

Valori predefiniti

- PI greco: **`pi`**
- Unità immaginaria: **`i`**, **`j`**
- Infinito: **`Inf`**
- Base del logaritmo naturale: **`eps`**

Grafica

- MATLAB permette di rappresentare graficamente i dati contenuti in una matrice.
- Esempio:

```
>> plot(e)
>> hold on
>> plot(d)
>> grid on
```

Grafica

- MATLAB permette di rappresentare graficamente i dati contenuti in una matrice.
- Esempio:

```
>> plot(e)
>> hold on
>> plot(d)
>> grid on
```

Disegna su di un piano cartesiano i segmenti che uniscono tutti i punti contenuti nella matrice e.

Grafica

- MATLAB permette di rappresentare graficamente i dati contenuti in una matrice.
- Esempio:

```
>> plot(e)
>> hold on
>> plot(d)
>> grid on
```

Fa si che al prossimo disegno il contenuto del piano cartesiano non venga cancellato.

Grafica

- MATLAB permette di rappresentare graficamente i dati contenuti in una matrice.
- Esempio:

```
>> plot(e)
>> hold on
>> plot(d)
>> grid on
```

Disegna su di un piano cartesiano i segmenti che uniscono tutti i punti contenuti nella matrice d.

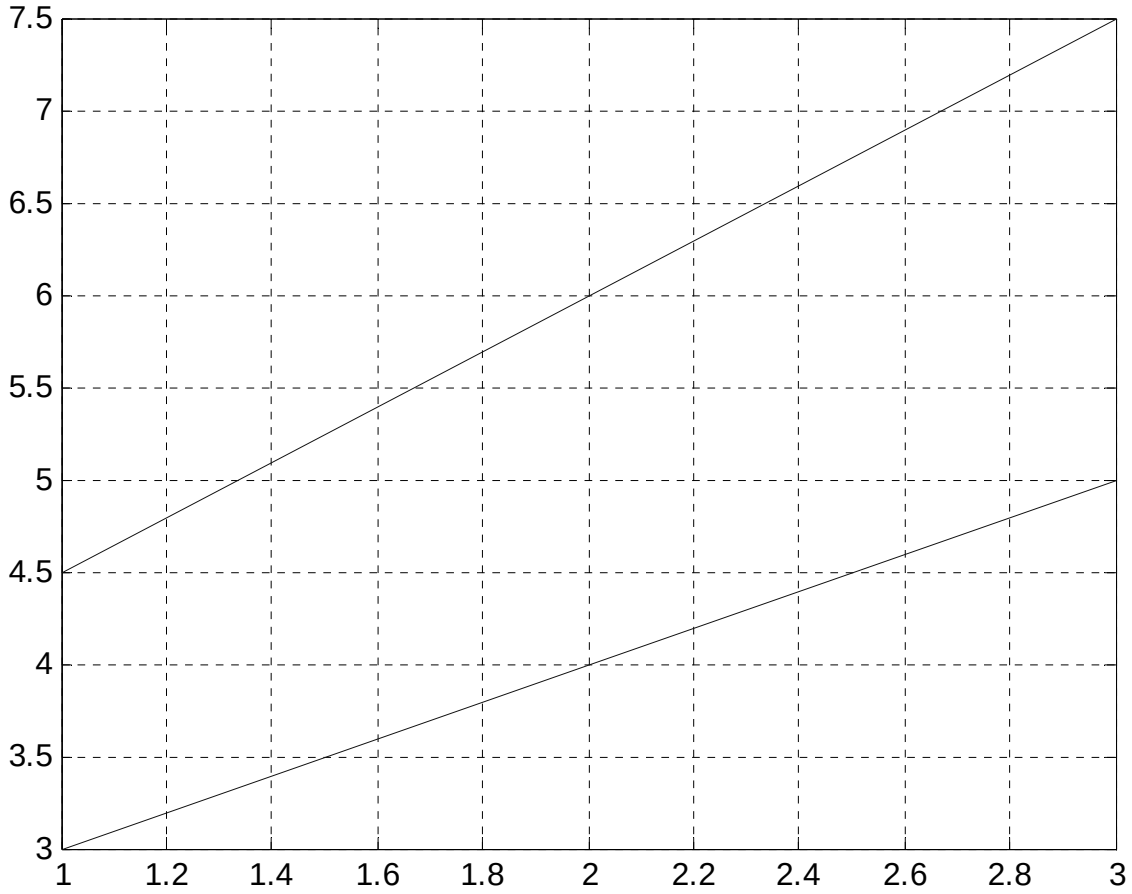
Grafica

- MATLAB permette di rappresentare graficamente i dati contenuti in una matrice.
- Esempio:

```
>> plot(e)
>> hold on
>> plot(d)
>> grid on
```

Visuallizza sul piano cartesiano una griglia per i valori sull'asse x e sull'asse y.

Risultato



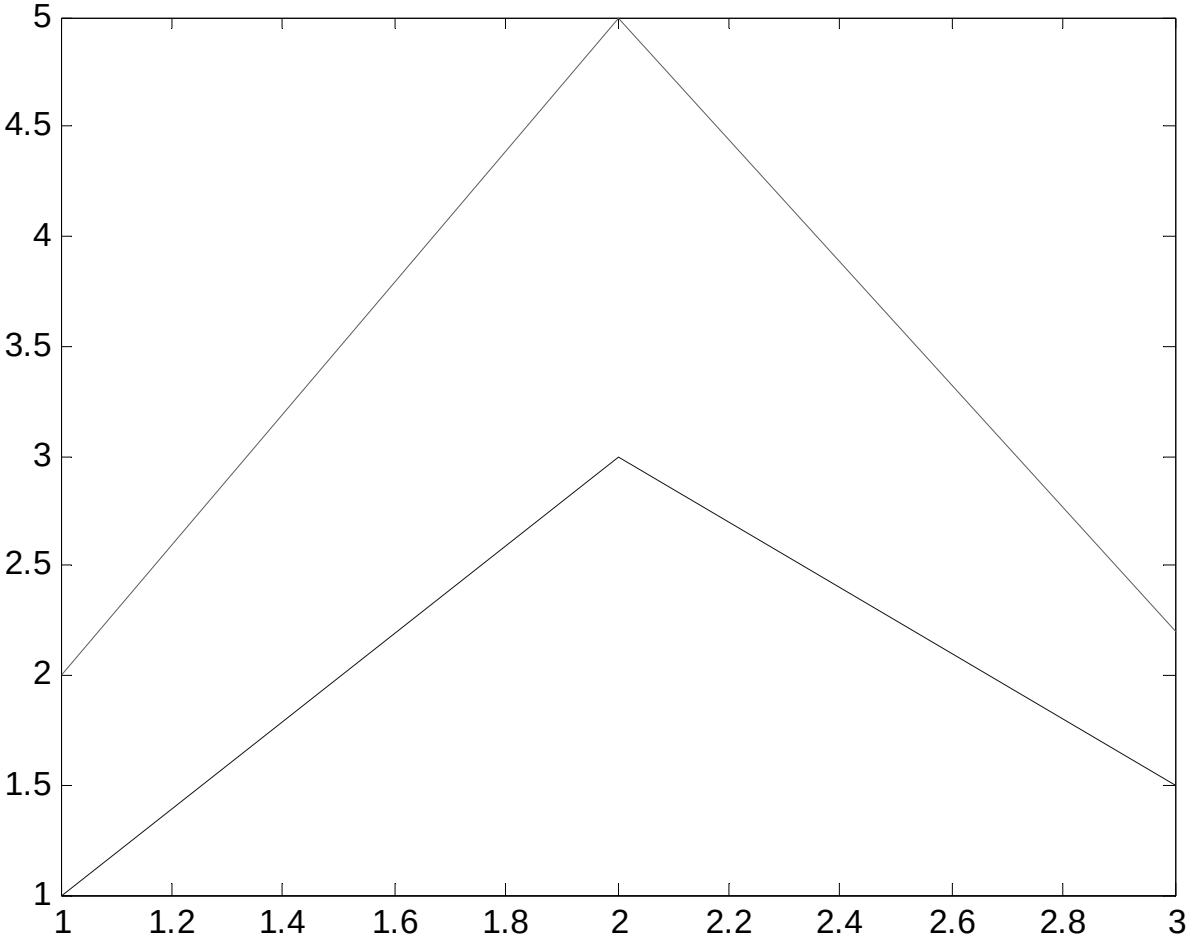
Tipi di grafico

- MATLAB offre diverse possibilità:
  - Grafici a linea
  - Grafici a barre
  - Grafici con simboli definiti dall'utente
- Esempio:

```
f=[1 2; 3 5; 1.5 2.2]
```

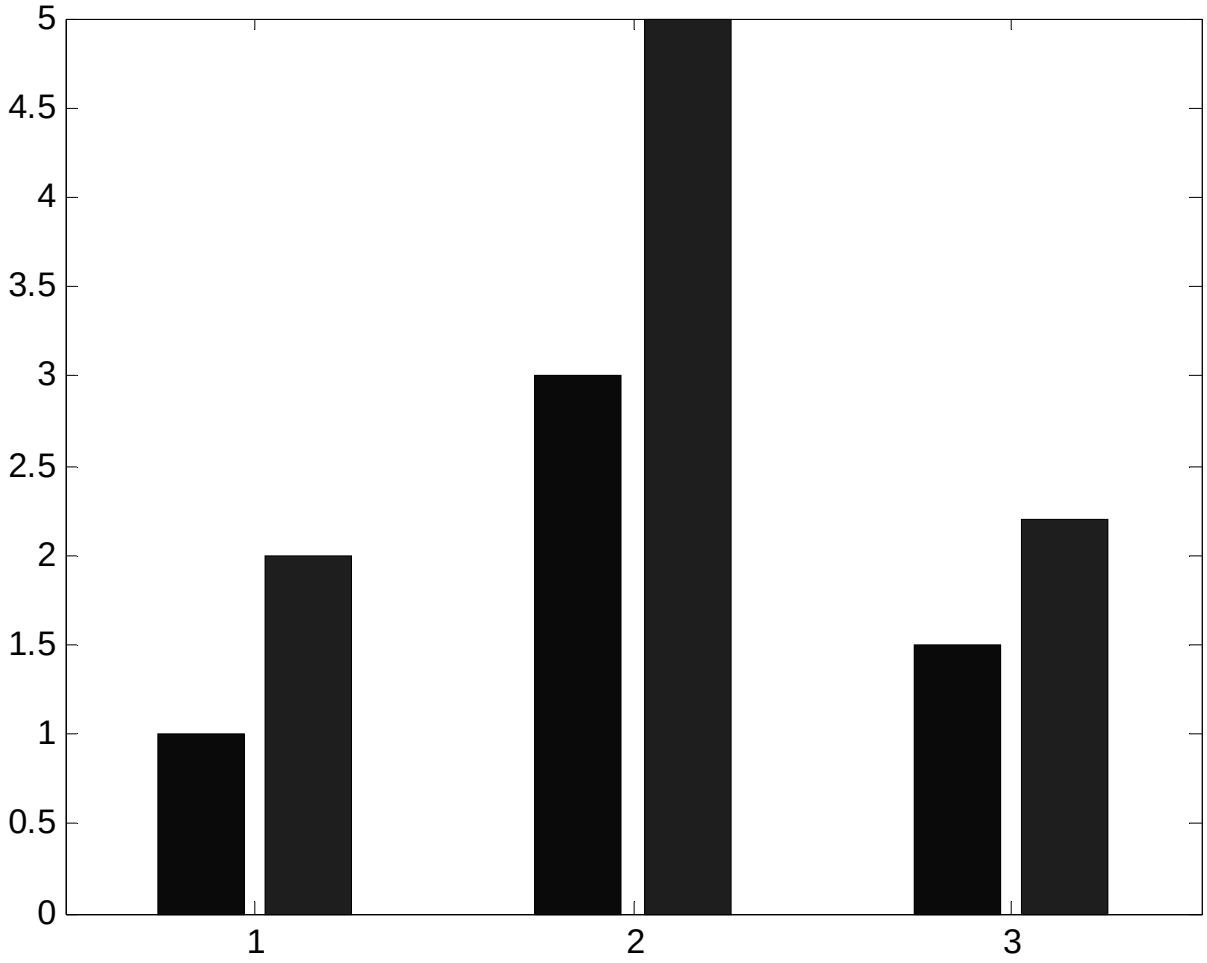
Grafici a linea

```
plot(f)
```



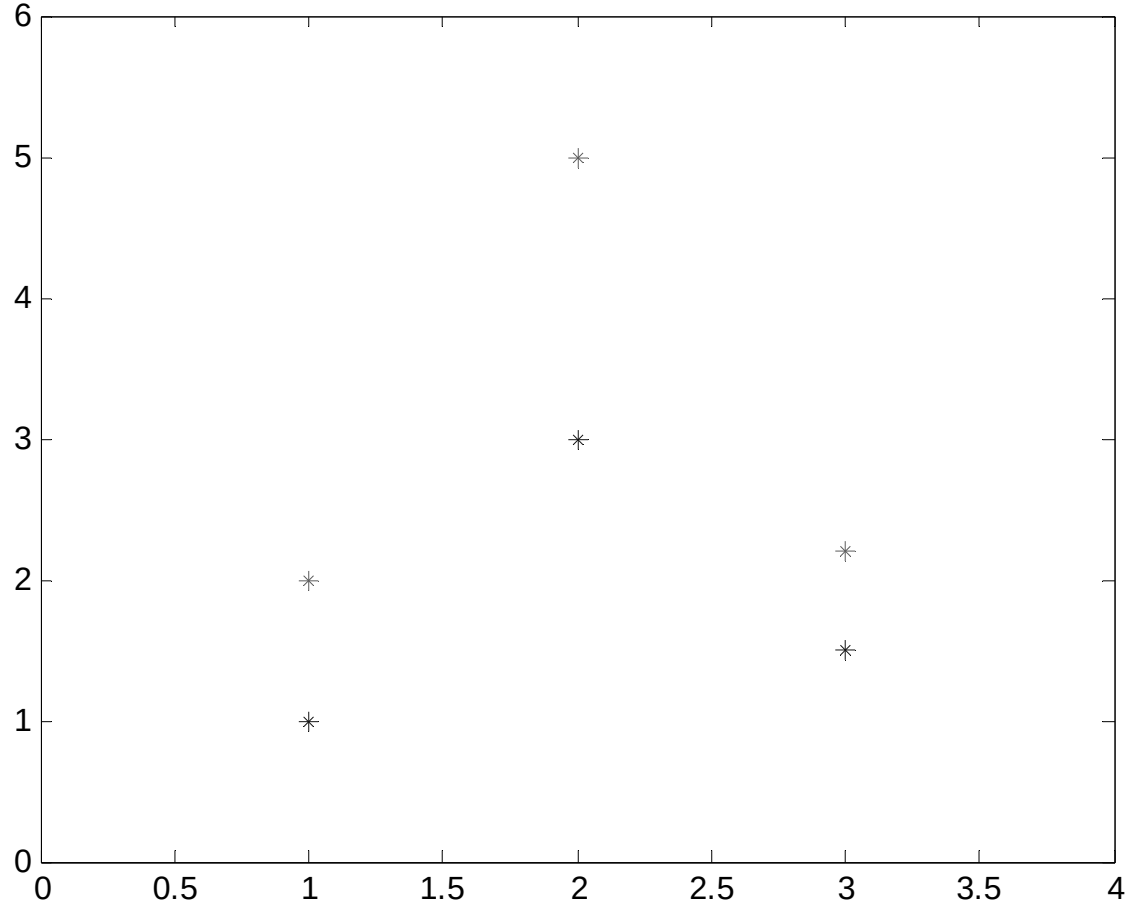
Grafici a barre

`bar(f)`



Grafici con simboli definiti dall'utente

```
plot(f,'*')
```



Disegno di funzioni 2D

- MATLAB permette di disegnare funzioni 2D su di un intervallo definito dall'utente.
- È necessario:
  - Definire la funzione da disegnare.
  - Definire l'interevallo su cui effettuare il disegno.

Esempio

- Si rappresenti la seguente funzione sull'intervallo I=[-1,2]:

$$f(x) = (2x - \sqrt{2})^2 \sin(2x)$$

Definizione della funzione

- Si utilizza la seguente sintassi:

```
Funzione = 'espressione'
```

- Nel nostro caso si ha:

```
>> f='(2*x-sqrt(2))^2*sin(2*x)'
```

Definizione dell'intervallo

- L'intervallo viene definito al momento del disegno della funzione utilizzando la seguente sintassi:

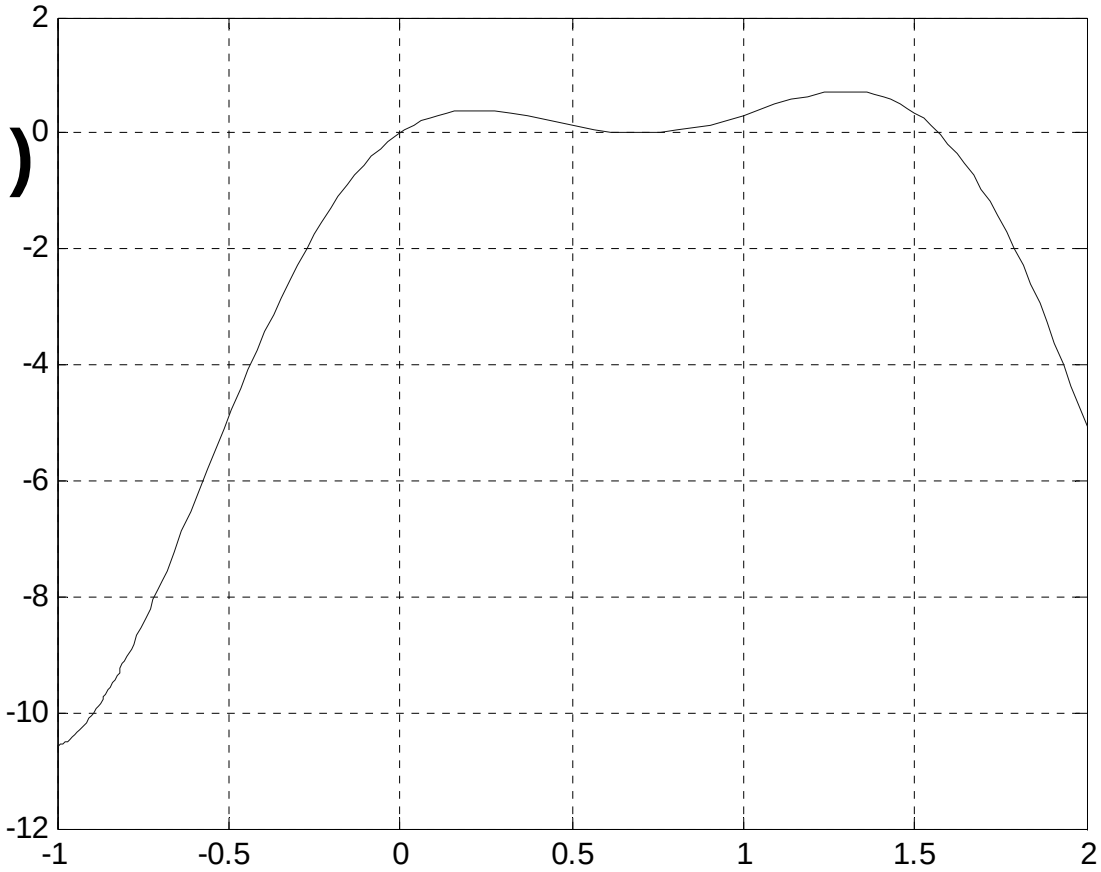
```
fplot(funzione, intervallo)
```

- Nel nostro caso si ha:

```
>> fplot(f, [-1,2])
```

Risultato

```
fplot(f, [-1,2])
```



Disegno di funzioni 3D

- MATLAB permette di disegnare funzioni 3D su di un dominio definito dall'utente.
- È necessario:
  - Definire il dominio su cui effettuare il disegno.
  - Definire la funzione da disegnare.

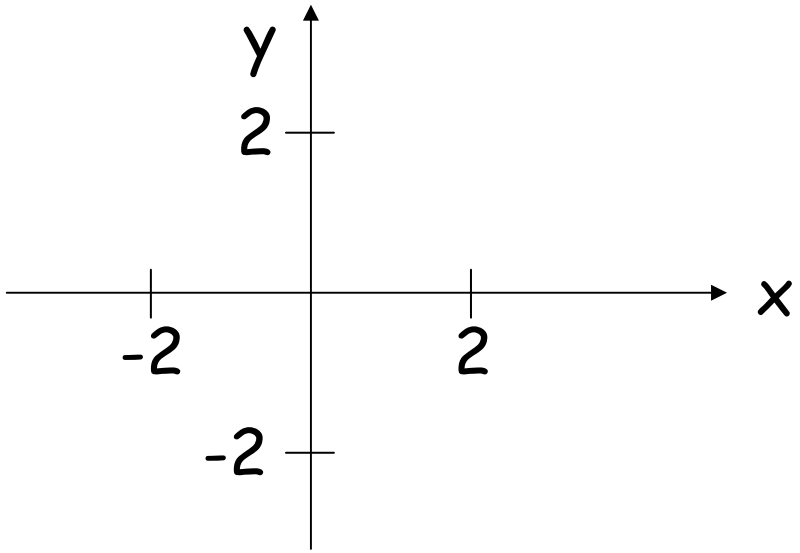
Esempio

- Si rappresenti la funzione f(x,y) sul dominio D={-2 ≤ x ≤ 2; -2 ≤ y ≤ 2} dove:

$$f(x,y) = xe^{-(x^2-y^2)}$$

Dominio

- È l'insieme di valori (x,y) per cui si vuole disegnare la funzione.
- Nel nostro esempio si ha:



Definizione del dominio

- Si utilizza la seguente sintassi:

$$[x,y]= \text{meshgrid}(X_{min}:X_{step}:X_{max},Y_{min}:Y_{step}:Y_{max});$$

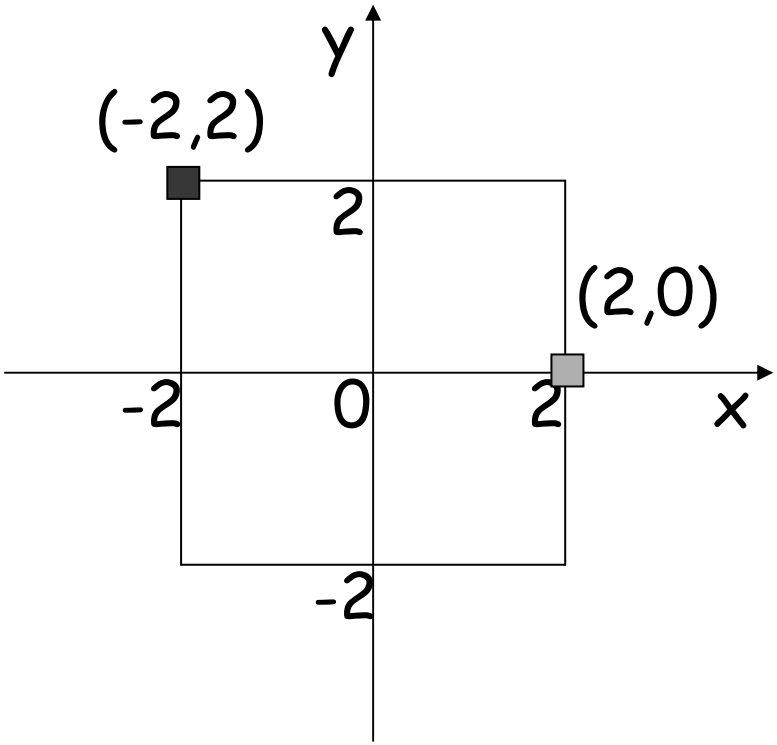
- X_{min} è il più piccolo valore di x per cui si vuole disegnare la funzione.
- X_{max} è il più grande valore di x per cui si vuole disegnare la funzione.
- X_{step} è la distanza tra due punti adiacenti sull'asse x.

Definizione del dominio

- Un punto P_{ij} appartenente al dominio è definito dalla coppia (x_{ij}, y_{ij}) dove:
  - x_{ij} è l'emento memorizzato in x alla riga i, colonna j.
  - y_{ij} è l'emento memorizzato in y alla riga i, colonna j.

Esempio

```
>>[x,y] = meshgrid(-2:2:2,-2:2:2);
x =
     -2      0      2
     -2      0      2
     -2      0      2
y =
     -2     -2     -2
      0      0      0
      2      2      2
```



Definizione della funzione

- Si utilizza la stessa sintassi usata per le funzioni 2D.
- Nel nostro esempio si ha:

```
>> f='x.*exp(-x.^2-y.^2)'
```

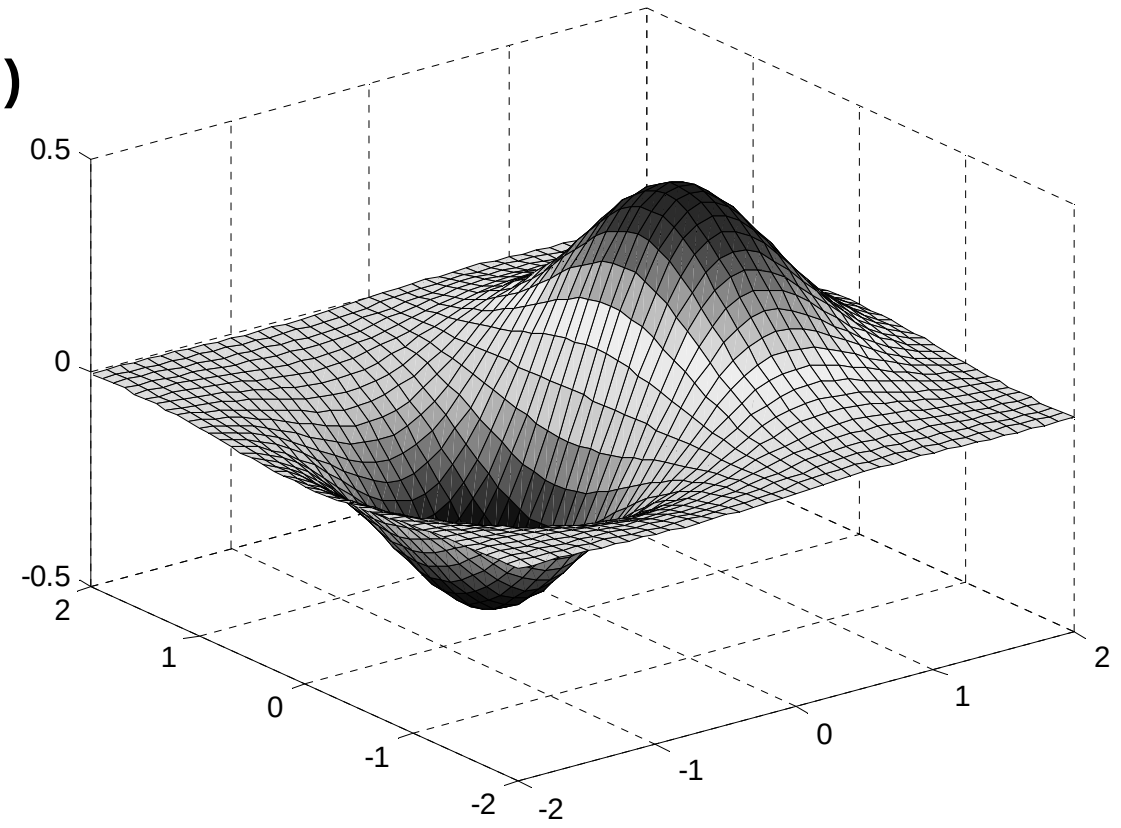
- **x** e **y** fanno riferimento alle matrici utilizzate per definire il dominio.
- Le operazioni algebriche tra **x** e **y** devono essere eseguire elemento per elemento.
- È necessario usare: **.*** **./** **.^**

Disegno della funzione

- Si effettua in due passi:
  - **`z=eval(f)`**: calcola il valore della funzione per tutte le coppie di punti del dominio.
  - **`surf(x,y,z)`**: disegna la funzione.

Risultato

```
>> [x,y]=meshgrid(-2:.1:2,-2:.1:2);
>> z=eval(f);
>> surf(x,y,z)
```



IF-THEN-ELSE

```
if condizione1,
        operazioni1;
elseif condizione2,
        operazioni2;
else
        operazioni3;
end;
```

Condizioni

- Maggiore: >
- Minore: <
- Maggiore-uguale: >=
- Minore-uguale: <=
- Uguale: ==
- Diverso: ~=
- AND logico: &
- OR logico: |
- NOT logico: ~

Cicli

Matlab

```
for k=1:s:N,
   Operazioni;
end;

while condizione,
   Operazioni;
end;
```

C/C++

```
for (k=1;k<=N;k+=step){
   blocco istruzioni
}

while (condizione){
   blocco istruzioni
}
```

Stringhe input e output

Il testo in MATLAB viene inserito sempre tra apici:

Es.: string='Ciao';

Per visualizzare stringhe o messaggi si adopera la funzione disp.

Es.: disp('Premere un tasto');

Messaggi di errore e Input

La funzione error mostra un messaggio di errore ed interrompe l'esecuzione di un file .m

Es.: error('A deve essere simmetrica');

La funzione input mostra un messaggio e permette l'inserimento di dati.

Es.: num_di_iter=input('Inserire il numero di iterazioni: ');

Esempio

```
%soluzioni di una equazione di secondo grado
disp('soluzioni di ax^2+bx+c')
a=input('dammi il coefficiente a ');
b=input('dammi il coefficiente b ');
c=input('dammi il coefficiente c ');
delta=b^2-4*a*c;
if delta==0
disp('soluzioni coincidenti')
        x=-b/(2*a)
elseif delta<0,
disp('Non ci sono soluzioni reali')
else
disp('Soluzioni')
x1=(-b+sqrt(delta))/(2*a)
x2=(-b-sqrt(delta))/(2*a)
end;
```

Leggere i dati da file

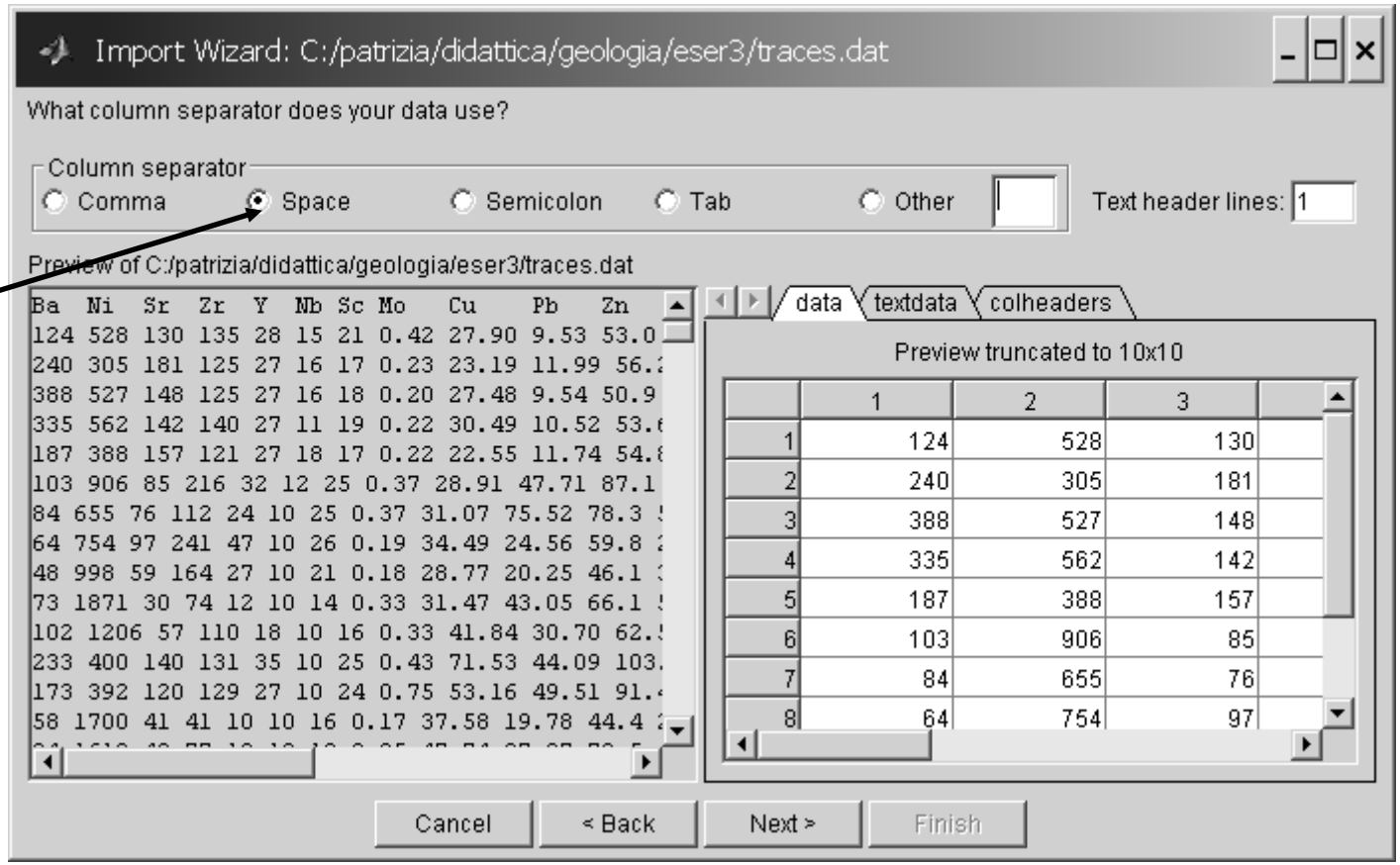
Per importare i dati da file esiste uno strumento molto comodo, simile a quello che abbiamo usato in Excel.

Dal menu File si seleziona Import Data e poi si sceglie la directory e il nome del file da importare.

Si apre quindi una finestra di dialogo che permette di vedere se Matlab interpreta corretamente i dati.

Importare dati da un file

Si deve selezionare il separatore

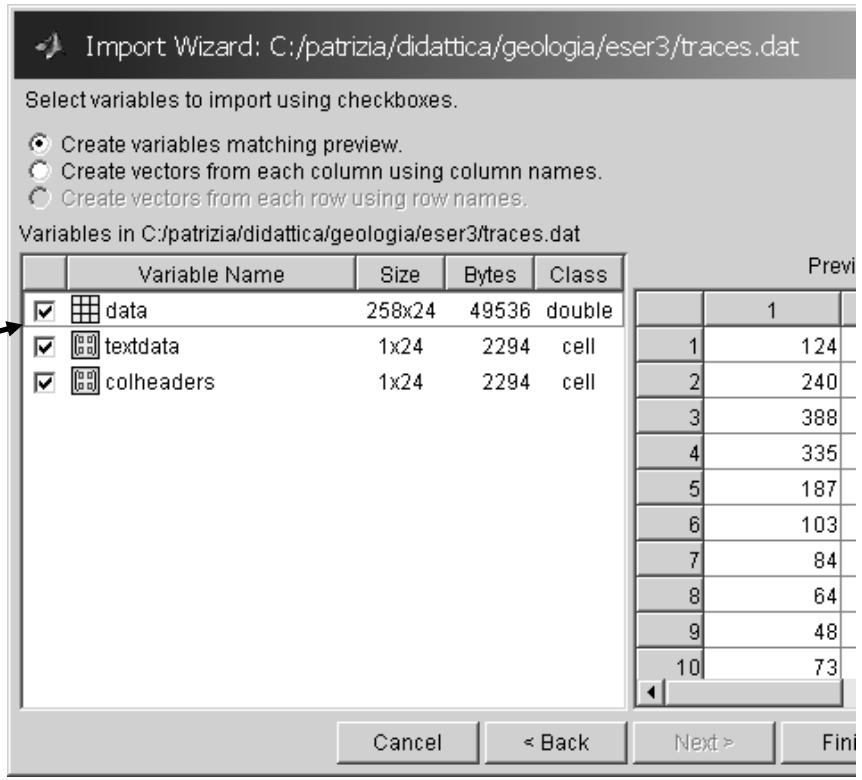


Si vede l'anteprima troncata 10x10

Mettere i dati in matrici

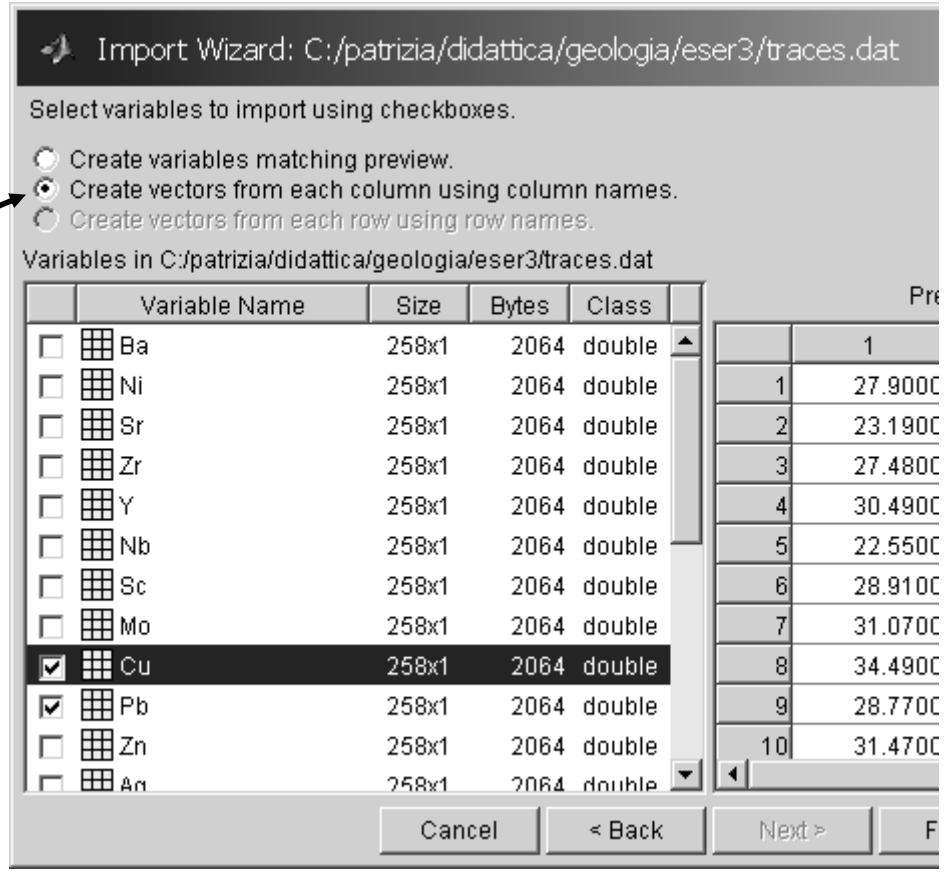
E' possibile scegliere come mettere i dati nelle matrici per poi eseguire le successive elaborazioni:

Tutti i dati in una unica matrice



Mettere i dati in matrici

Oppure mettere ogni colonna in vettore con il nome uguale al testo iniziale



Quando si sono fatte tutte le scelte si clicca su Finish

Salvare le variabili

Il procedimento utilizzato per importare i dati, anche se molto comodo, non può, essere richiamato da un file di programma (.m).

Per ovviare questo inconveniente si possono salvare tutte le variabili presenti in un certo momento (oppure solo quelle che ci interessano) su un file (con estensione .mat) che poi può, in qualunque momento, essere richiamato sia da comando di linea sia da file di programma.

Per salvare TUTTE le variabili si utilizza il comando File | Save Workspace as...

Caricare le variabili

Se si vuole salvare solo una variabile

```
>> save 'rame.mat' Cu    (ovvero save 'nomefile' variabile )
>> clear                 (pulisco il workspace)
>> load 'rame.mat'
>> whos                  (verifico il workspace)
  Name      Size          Bytes  Class

  Cu      258x1            2064  double array
```

Data fitting

Supponiamo di avere due serie di misure di altrettante grandezze e vogliamo trovare la legge che le lega.

Se pensiamo che la legge che lega le due grandezze sia lineare, attraverso il coefficiente di correlazione siamo in grado di valutare quanto bene una retta lega le due grandezze, ma non possiamo ricavare i parametri della retta.

Metodo dei minimi quadrati

Consideriamo N coppie di misure $(x_1 , y_1), (x_2 , y_2) , \dots ,$ (x_N, y_N) di due grandezze x ed y fra le quali sappiamo, o supponiamo, che esista una relazione lineare.

Supponiamo altresì per semplicità, che l'errore di misura su una delle variabili (per esempio la x, per fissare le idee) sia trascurabile rispetto a quello dell'altra.

La relazione lineare tra le due grandezze sia

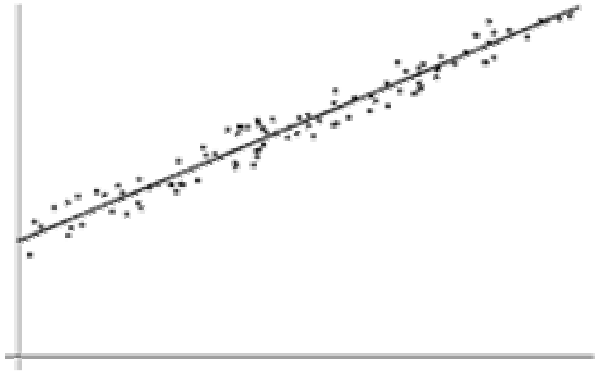
$$y=ax+b$$

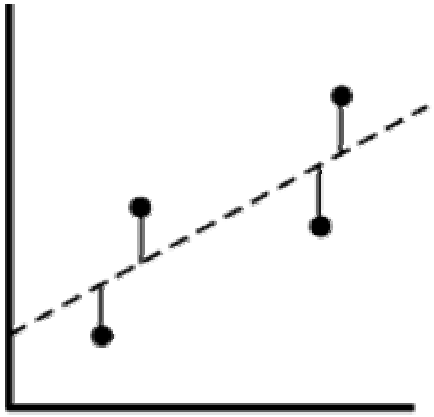
Vogliamo determinare a e b che corrispondono ai dati sperimentali.

Metodo dei minimi quadrati

Per trovare questi parametri dobbiamo cercare un "criterio" che ci permetta di definire la "miglior retta" che interpola i dati.

Si definisce "miglior retta" nel senso dei minimi quadrati quella che minimizza la somma dei quadrati degli scarti (distanze) dei dati dalla retta.





Metodo dei minimi quadrati

Definiamo le seguenti quantità:

$$S_{xy} = \sum_{i=1}^{N} \frac{x_i y_i}{\sigma_i^2} \qquad S_{xx} = \sum_{i=1}^{N} \frac{x_i^2}{\sigma_i^2}$$

$$S_x = \sum_{i=1}^{N} \frac{x_i}{\sigma_i^2} \qquad S_y = \sum_{i=1}^{N} \frac{y_i}{\sigma_i^2} \qquad S_0 = \sum_{i=1}^{N} \frac{1}{\sigma_i^2}$$

dove N è il numero dei dati e σ_i è l'errore sul dato y_i.

Metodo dei minimi quadrati

Si trova che la retta nel senso dei minimi quadrati è quella con i seguenti valori dei parametri:

$$a = \frac{S_{xy} S_0 - S_x S_y}{S_{xx} S_0 - S_x^2}$$

$$b = \frac{S_y S_{xx} - S_x S_{xy}}{S_{xx} S_0 - S_x^2}$$

Metodo dei minimi quadrati

Nel caso che l'errore sia per uguale per tutte le misure le formule si semplificano e diventano:

$$a = \frac{N\sum_{i=1}^{N} x_i y_i - \sum_{i=1}^{N} x_i \sum_{i=1}^{N} y_i}{N\left(\sum_{i=1}^{N} x_i^2\right) - \left(\sum_{i=1}^{N} x_i\right)^2}$$

$$b = \frac{\sum_{i=1}^{N} x_i^2 \sum_{i=1}^{N} y_i - \sum_{i=1}^{N} x_i \sum_{i=1}^{N} x_i y_i}{N\left(\sum_{i=1}^{N} x_i^2\right) - \left(\sum_{i=1}^{N} x_i\right)^2}$$

Un programma Matlab per i minimi quadrati

```
N=8;
x=[0.1,0.15,0.2,0.25,0.3,0.35,0.4,0.45];
y=[0.393,0.608,0.81,1.008,1.199,1.4,1.6,1.84];
corrcoef(x,y)                    % controllo che stiano su una retta
sx=sum(x);                       % somma degli elementi di x
sy=sum(y);                       % somma degli elementi di y
sommax2=sum(x.*x);               % somma di x^2
sxy=x*transp(y);                 % somma di x*y
den=N*sommax2-sx^2;
a=(N*sxy-sx*sy)/den
b=(sommax2*sy-sx*sxy)/den
```

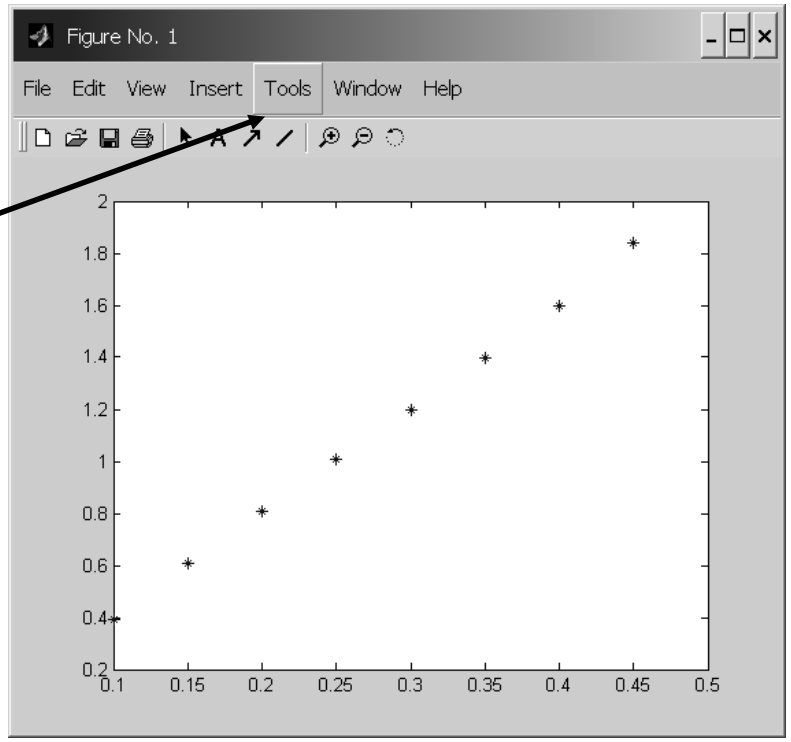
Data Fitting Tool

Lo stesso risultato si può ottenere utilizzando il tool Basic Fitting nella finestra di plot.

plot(x,y,'*');

Tools

Basic Fitting



Basic Fitting Tool