

INTRODUZIONE A MATLAB

1

- ➔ Linguaggio per risolvere problemi di calcolo numerico *MATrix LABoratory*;
- ➔ Marchio registrato da *MathWorks Inc.* (U.S.A.)
- ➔ Può essere ampliato da pacchetti specifici (toolbox)
 - ➔ *Wavelet Toolbox, Signal processing Toolbox*
- ➔ È un interprete in grado di eseguire:
 - ➔ Istruzioni native (built-in)
 - ➔ Istruzioni contenute in files

2

La linea di comando di MATLAB è indicata dal prompt

>>

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 dei segnali, basta digitare:

help signal

Per sapere quali sono le varie categorie di funzioni disponibili (i **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 **%**

PUNTEGGIATURA E VARIABILI

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

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

ISTRUZIONI ELEMENTARI

Si supponga di aver definito in memoria una matrice A di dimensione 2×3 e una variabile ans.

who: elenco delle variabili definite in memoria

```
>> who
your variables are:
A      ans
```

whos: Informazioni sulle variabili definite in memoria

```
>> whos
Name Size Bytes Class
A      2x3    48   double array
ans    1x1     8   double array
Grand total is 7 elements using 56 bytes
```

save: salva tutte le variabili in memoria nel file file.mat

```
>> save file A
```

clear: cancella tutte le variabili in memoria o una in particolare se specificata

```
>> clear A
>> clear
```

load: richiama in memoria le variabili salvate nel file specificato

```
>> load file
```

OPERATORI SCALARI

Gli operatori disponibili sono:

- +, -, *, /, ^,
- sin, cos, tan,
- asin, acos, atan,
- exp, log (naturale), log10 (in base 10),
- abs, sqrt, sign

ELEMENTI DI BASE DI MATLAB

L'inserimento di un vettore o di una matrice in generale viene effettuato tra parentesi quadre, separando gli elementi delle righe con spazi o virgole, e le diverse righe con punti e virgola (oppure andando a capo ad ogni nuova riga).

```
>> x = [1, 2, 3]; % vettore riga
>> y = [1; 4; 7]; % vettore colonna
>> A = [1 2 3; 4 5 6; 7 8 9]; % matrice
>> A = [1 2 3
        4 5 6
        7 8 9 ];
```

Per far riferimento agli elementi di una matrice A:

- l'elemento a_{mn} è indirizzato come $A(m, n)$;

```
>> A(2,3)
6
```
- la riga m-esima è indirizzata come $A(m, :)$, dove tutte le colonne sono indicate con due punti;

```
>> A(2, :)
4 5 6
```
- la colonna n-esima è indirizzata come $A(:, n)$, dove tutte le righe sono indicate con due punti;

```
>> A(:,3)
3
6
9
```

11

OPERAZIONI SULLE MATRICI

Date due matrici A e B di dimensione opportune, si possono definire le seguenti operazioni:

```
>> S=A+B; % somma di due matrici
>> P=A*B; % prodotto righe per colonne
           % di due matrici
>> At=A'; % trasposta di una matrice
>> Ai=inv(A); % inversa di una matrice
```

12

Altre funzioni operanti su *matrici* (e, quindi, su *vettori*, riga o colonna) sono:

max, min,
sort,
sum, prod,

Esistono poi particolari operatori (`.*`, `./`, `.^`) che permettono di effettuare operazioni su vettori *elemento per elemento*, senza ricorrere a cicli. Ad esempio, se A e B sono due matrici, per sommare elemento per elemento le due matrici basta fare:

```
>> C=A.+B;
```

13

A. Murli - Introduzione a Matlab

- Altre funzioni che operano invece essenzialmente su matrici sono:

det ← Determinante della matrice

```
>> det(A)
0
```

size ← Dimensioni della matrice

```
>> size(A)
3 3
```

rank ← Rango della matrice

```
>> rank(A)
2
```

14

A. Murli - Introduzione a Matlab

Esistono poi varie funzioni predefinite per la creazione di matrici:

`eye(n)` : matrice identità n righe n colonne
`zeros(m,n)`: matrice di 0 con m righe e n colonne
`ones(m,n)` : matrice di 1 con m righe e n colonne
`rand(m,n)` : matrice casuale di valori tra 0 e 1
`diag(X)` : se x è un vettore con n elementi, produce una matrice quadrata diagonale di dimensione n per n con gli elementi di x sulla diagonale. Se invece x è una matrice quadrata di dimensione n per n , produce un vettore di n elementi pari a quelli sulla diagonale di x .


15

A. Murli - Introduzione a Matlab

RISOLUZIONE DI SISTEMI LINEARI

Calcolare il valore di x , con $Ax=B$  $x=A^{-1}B$

```
>> x=A\B;  
>> x=inv(A)*B;
```

Calcolare il valore di x , con $xC=D$  $x=DC^{-1}$

```
>> x=D/C;  
>> x=D*inv(C);
```

➤ **slash** / determina la divisione con la matrice posta a destra

➤ **backslash** \ determina la divisione con la matrice posta a sinistra

16

A. Murli - Introduzione a Matlab

GRAFICI IN MATLAB

17

La grafica è una delle caratteristiche più sviluppate di MATLAB

- Permette di tracciare più grafici sulla stessa finestra o su più finestre dette "**figure**"
- Per default MATLAB traccia grafici sulla finestra 1
- Volendo aprire più finestre grafiche occorre digitare il comando **figure(n)** dove **n** definisce il numero della finestra
- Da questo punto in poi MATLAB tratterà grafici sulla finestra **n**-esima fino a quando non si cambierà finestra con un nuovo comando **figure**
- La chiusura della finestra **n**-esima avviene con il comando **close(n)**

18

Per definire intervalli si utilizza l'operatore colon (:)

Ad esempio, il vettore le cui componenti sono i valori compresi tra 0 e 2 con passo 0.1 è definito come:

```
» a=[0:0.1:2]
a =
Columns 1 through 7
0 0.1000 0.2000 0.3000 0.4000 0.5000
0.6000
Columns 8 through 14
0.7000 0.8000 0.9000 1.0000 1.1000
1.2000 1.3000
Columns 15 through 21
1.4000 1.5000 1.6000 1.7000 1.8000
1.9000 2.0000
```

19

A. Murli - I grafici in Matlab

LA FUNZIONE *plot*

Se x è un vettore contenente le ascisse dei punti di un fissato insieme di coppie del piano e y è il vettore delle corrispondenti ordinate, $\text{plot}(x,y)$ disegna la spezzata congiungente tali punti

Esempio:

diagrammare il valore della funzione $\text{sen}(x)$ da zero a 2π

```
» x=[0:pi/100:2*pi];
» y = sin(x);
» plot(x,y)
```

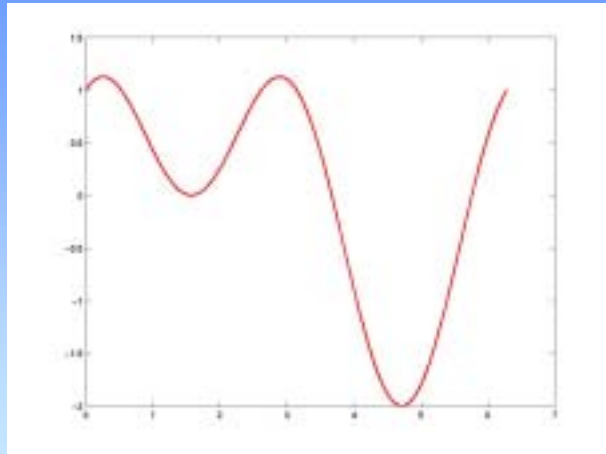


20

A. Murli - I grafici in Matlab

Esempio:

```
» x=[0:0.01:2*pi];  
» y=sin(x)+cos(2*x);  
» plot(x,y)
```



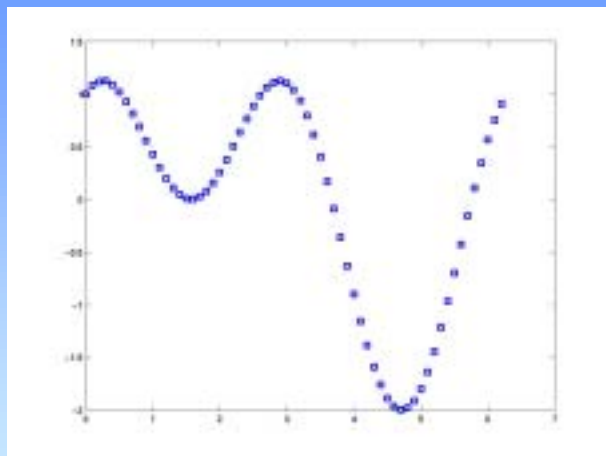
A.

21

La funzione *plot* è in grado di tracciare le curve impiegando svariati simboli:

Esempio:

```
» x=[0:0.1:2*pi];  
» y=sin(x)+cos(2*x);  
» plot(x,y,'s')
```



A.

22

La sintassi di *plot* nel caso in cui si vogliono utilizzare simboli, colori o tipi diversi di linee è la seguente:

`plot(x,y,'stile')`

Linea continua	-	Punto	.	Colore rosso	r
Linea tratteggiata	--	Più	+	Colore verde	g
Linea punteggiata	:	Cerchio	o	Colore blu	b
Linea tratto punto	-.	Stella	*	Colore bianco	w
		Croce	x	Colore invisibile	i
		Quadrato	s		

Esempio:

```
>> plot(x,y,'b+:')
```

traccia una linea blu e punteggiata, con il simbolo + in corrispondenza di ogni valore

23

A. Murli - I grafici in Matlab

LA FUNZIONE *axis*

Per *creare degli assi* cartesiani si usa la funzione *axis*

```
>>axis([x_min,x_max,y_min,y_max])
```

```
>>axis('string')
```

- Nella prima forma si impongono i limiti inferiore e superiore degli assi cartesiani
- Nella seconda forma, 'string' indica l'aspetto degli assi;

ad esempio:

String=square per avere i due assi uguale

String=normal per sfruttare tutto lo schermo

String=auto restituisce l'asse in scala default, in maniera automatica

24

A. Murli - I grafici in Matlab

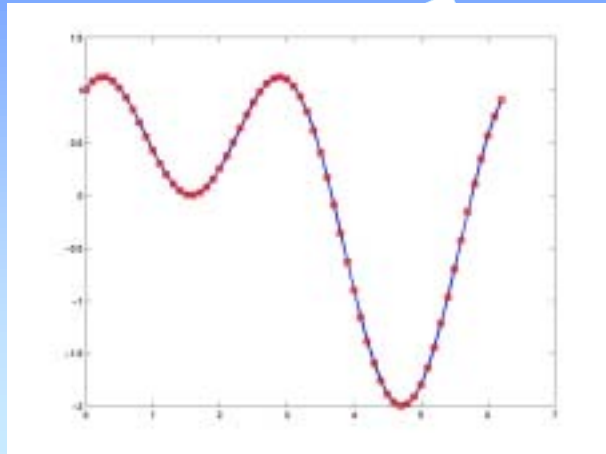
LA FUNZIONE *hold*

La funzione *hold* conserva il riferimento di assi cartesiani e il corrispondente grafico già esistente; il grafico successivo verrà sovrapposto al precedente.

Si usa nella forma
hold on
hold off

Esempio:

```
» x=[0:0.1:2*pi];  
» y=sin(x)+cos(2*x);  
» plot(x,y);  
» hold on  
» plot(x,y,'s')
```



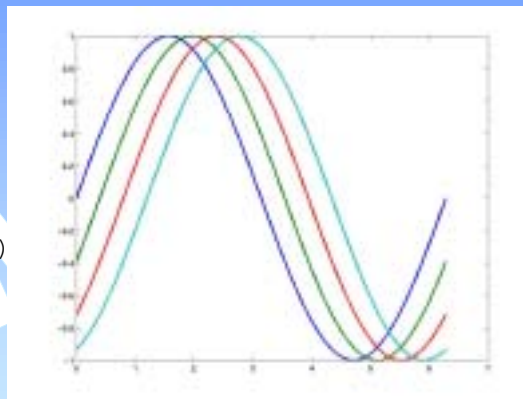
25

A. Murli - I grafici in Matlab

E' possibile riportare sulla stessa figura più funzioni rappresentate da diverse coppie (x,y)

Esempio:

```
» x = [0:pi/100:2*pi];  
» y = sin(x);  
» y2 = sin(x - .40);  
» y3 = sin(x - .8);  
» y4 = sin(x - 1.2);  
» plot(x,y,x,y2,x,y3,x,y4)
```



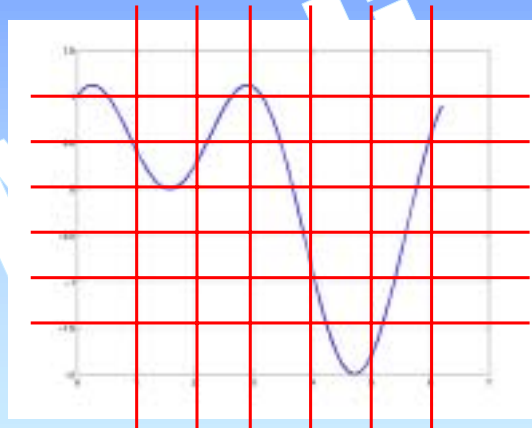
26

A. Murli - I grafici in Matlab

- La funzione *zoom* permette l'ingrandimento di regioni del grafico.
- L'attivazione della funzione *grid* traccia un reticolato sul grafico.

Viene usata nella forma
grid on
grid off

» grid on

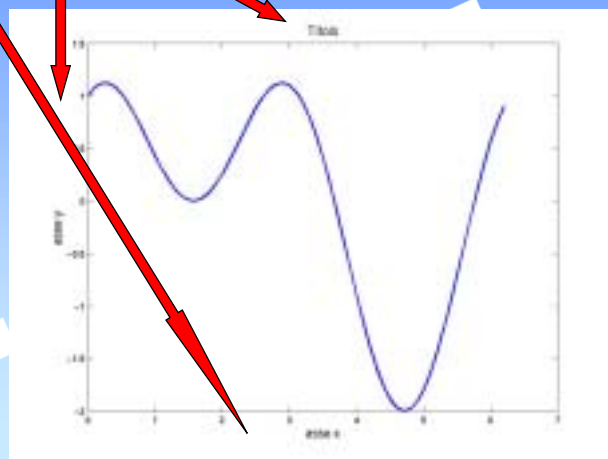


27

Il comando *clf* pulisce la finestra corrente, mentre *figure* ne apre una nuova.

Le istruzioni *xlabel*, *ylabel* e *title* etichettano gli assi e la figura:

```
» title('Titolo')  
» xlabel('asse x')  
» ylabel('asse y')
```



28

LA FUNZIONE *text*

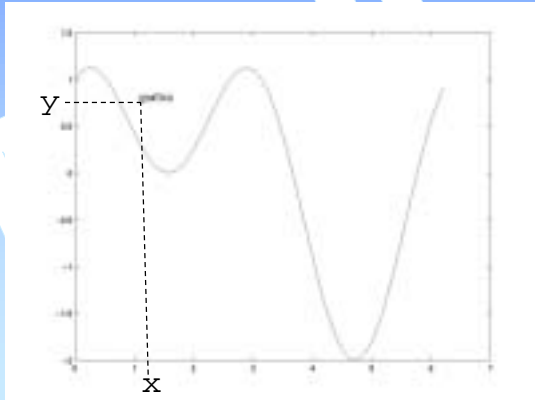
La funzione *text* permette di scrivere una didascalia sul grafico.

```
text(x,y,'testo')
```

dove x,y sono le coordinate da cui inizierà il testo, sapendo che $(0,0)$ è l'angolo in basso a sinistra e $(1,1)$ quello in alto a destra

Esempio:

```
» text(x,y,'grafico');
```



29

A. Murli - I grafici in Matlab

GRAFICI TRIDIMENSIONALI

Grafici tridimensionali sono tipicamente tracciati per mezzo delle funzioni *plot3*, *mesh* e *surf*.

La *plot3* consente di tracciare una curva nello spazio a partire dalle sue equazioni parametriche.

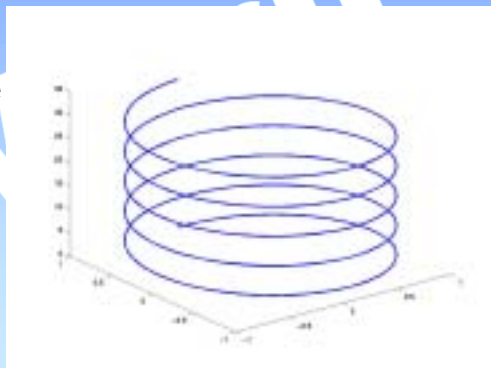
Esempio:

Dalle equazioni parametriche dell'elicoide

$$\begin{cases} x = \sin t \\ y = \cos t \\ z = t \end{cases}$$

```
» t=[0:pi/50:10*pi];
```

```
» plot3(sin(t),cos(t),t);
```



30

A. Murli - I grafici in Matlab

TRACCIAMENTO DI PIU' GRAFICI NELLA STESSA FINESTRA

Il comando `subplot(m,n,p)` divide la finestra corrente in m righe, n colonne e seleziona la finestra p-esima,

ad esempio

```
subplot(2,2,3)
```

spezza la finestra in quattro sottofinestre e seleziona quella in basso a sinistra (riga 2, colonna 1)

il comando `subplot(m,n,p)` deve essere, dunque, usato prima del `plot`, per fornire informazioni sulla sottofinestra in cui visualizzare il grafico desiderato

31

A. Murli - I grafici in Matlab

ALCUNI ESERCIZI SU MATLAB

OPERAZIONI MATRICIALI

Scrivere un file .m contenente le istruzioni relative alla risoluzione dei seguenti quesiti:

1. Risolvere il seguente sistema lineare, ponendo $x=A^{-1}b$, con A matrice dei coefficienti e b vettore dei termini noti:

$$2x_1 - 4x_2 + 7x_3 + 4x_4 = 5$$

$$9x_1 + 3x_2 + 2x_3 - 7x_4 = -1$$

$$5x_1 + 2x_2 - 3x_3 + x_4 = -3$$

$$6x_1 - 5x_2 + 4x_3 - 3x_4 = 2$$

2. calcolare il prodotto scalare $s=u*v^T$, della seguente coppia di punti:

$$u = (5, 3, -2, -4, -1) \quad v = (2, -1, 0, -7, 2)$$

33

A. Murli - Introduzione a Matlab

3. Data la matrice $A = \begin{pmatrix} 5 & 3 & -6 \\ 7 & 2 & 0 \\ -4 & 8 & 1 \end{pmatrix}$ calcolare:

```
A1=A*A;
```

```
A2=A'*A;
```

```
A3=A.*A;
```

```
d1=diag(A);
```

```
d2=diag(A,1);
```

```
e=exp(A);
```

```
sq=sqrt(A);
```

```
e1=exp(log(A+7));
```

```
m=max(A);
```

```
sn=sign(A);
```

34

A. Murli - Introduzione a Matlab

per concludere:

- ✓ con l'istruzione `save` salvare in un file `ristest.mat` i risultati degli esercizi svolti:
`x;s;A1;A2;A3;d1;d2;e;sq;el;m;sn ;`
- ✓ con l'istruzione `clear` liberare la memoria dai risultati delle operazioni eseguite;
- ✓ con il `load` caricare il file `ristest`;
- ✓ commentare i risultati ottenuti nell'esercizio 3 richiamando le relative variabili.

35

GRAFICI BIDIMENSIONALI

Graficare in due finestre diverse (`figure(1)` e `figure(2)`) la funzione:

$$f(t) = e^{-10t} + \sin(0.05t)$$

con due scale di tempi diverse:

```
>>t1=[0:0.01:1]      >>t2=[0:200]
>>t1=[0:0.01:1];
>>y1=exp(-10*t1)+sin(0.05*t1);
>>t2=[0:200];
>>y2=exp(-10*t2)+sin(0.05*t2);
>>figure(1)
>>plot(t1,y1)
>>figure(2)
>>plot(t2,y2)
```

36

TRACCIAMENTO DI PIU' GRAFICI NELLA STESSA FINESTRA

- ✓ *cancellare le due finestre dell'esercizio precedente;*
- ✓ *utilizzando il comando `subplot(m,n,p)` spezzare la finestra corrente in quattro sottofinestre e graficare in esse quattro funzioni a scelta, in opportuni intervalli di variabilità.*