

Laboratorio di MatLab

(seconda parte)

Carla Bertocchi
Vanna Lisa Coli
Alessandro Benfenati

Dipartimento di Matematica e Informatica - Università di Ferrara
carla.bertocchi@unimore.it

1 Richiami

- Indici
- Accesso ai dati di matrici e vettori
- Vettorizzazione
- Stampa a video

2 Tipi di dato

- Strutture
- Cell Array

3 Gestione I/O nelle funzioni

4 Grafica 2D

- Comandi base
- Istruzioni utili
- Figure nel piano

L'importanza di avere un help

**Consultare l'help o il doc per
TUTTE le informazioni
sulle funzioni native**

Si ricorda:

- | | |
|--------------------------------|---|
| <code>help NOMEFUNZIONE</code> | visualizza un rapido help sulla shell con alcuni esempi di utilizzo |
| <code>doc NOMEFUNZIONE</code> | apre il manuale sotto forma di browser ed è completo di spiegazione delle funzioni e di vari esempi di utilizzo |

È disponibile per tutte le funzioni presenti in MatLab (`sin`, `tan`, ...).

Richiami: indici

L'istruzione $a:h:b$ genera un vettore il cui primo elemento è a , l'ultimo è b e i cui elementi sono distanziati di passo h .

Esempio 1

Generare un vettore i cui estremi siano -1 e 0 e i cui elementi siano distanziati ad un passo 0.1.

```
>> x = -1:0.1:0
x =
  Columns 1 through 7
   -1.0000   -0.9000   -0.8000   -0.7000   -0.6000
  -0.5000   -0.4000
  Columns 8 through 11
   -0.3000   -0.2000   -0.1000           0
```

Esempio 2

Generare un vettore che contenga i numeri pari compresi fra 28 e 47.

```
>> x = 28:2:47
x =
    28    30    32    34    36    38    40    42    44    46
```

Richiami: indici

Esempio 3

Generare un vettore che contenga i multipli di 10 compresi fra 20 e 100.

```
>> x = 20:10:100
x =
    20    30    40    50    60    70    80    90   100
```

Il comando `linspace(a,b,n)` genera un vettore con estremi a e b e di n elementi.

Esempio 4

Generare un vettore di 13 elementi compresi fra -5 e 1 .

```
>> x = linspace(-5, 1, 13)
x =
Columns 1 through 7
-5.0000 -4.5000 -4.0000 -3.5000 -3.0000
-2.5000 -2.0000
Columns 8 through 13
-1.5000 -1.0000 -0.5000 0 0.5000 1.0000
```

Richiami: indici

Per accedere ad un elemento specifico del vettore v si utilizza la notazione con le parentesi tonde: $v(5)$ accede al 5° elemento del vettore v .

È possibile accedere contemporaneamente a vari elementi del vettore

Esempio 5

Dato il vettore $v = \text{linspace}(2,12,7)$, salvare in w gli elementi di posto pari di v .

```
>> v = linspace(2,12,7)
v =
    2.0000    3.6667    5.3333    7.0000    8.6667
   10.3333   12.0000
>> w = v(2:2:end)
w =
    3.6667    7.0000   10.3333
```

Il comando `end` riconosce automaticamente l'indice finale del vettore.

Richiami: indici

Esempio 5

Dato il vettore `v=linspace(0,10,17)`, memorizzare gli elementi di posto 1,2,5,6,10,12,13,14 e 17 con un'unica stringa di comando.

```
>> v=linspace(0,10,17);      >> w = v([1:2,5:6,10:14,17]);
>> v'                        >> disp(w)
ans =                        0
    0
  0.6250
  1.2500
  1.8750
  2.5000
  3.1250
  3.7500
  4.3750
  5.0000
  5.6250
  6.2500
  6.8750
  7.5000
  8.1250
  8.7500
  9.3750
 10.0000
    0.6250
    2.5000
    3.1250
    5.6250
    6.2500
    6.8750
    7.5000
    8.1250
   10.0000
```

Richiami: indici

Esempio 6

Data la matrice $A = \text{rand}(5)$, memorizzare in B la sottomatrice di A composta dalle prime 3 righe e dalle prime 3 colonne. Memorizzare in C le prime due righe di A .

```
>> A = rand(5)
```

```
A =
```

0.4170	0.0923	0.4192	0.6705	0.8007
0.7203	0.1863	0.6852	0.4173	0.9683
0.0001	0.3456	0.2045	0.5587	0.3134
0.3023	0.3968	0.8781	0.1404	0.6923
0.1468	0.5388	0.0274	0.1981	0.8764

```
>> B = A(1:3,1:3)
```

```
B =
```

0.4170	0.0923	0.4192
0.7203	0.1863	0.6852
0.0001	0.3456	0.2045

```
>> C = A(1:2,:)
```

```
C =
```

0.4170	0.0923	0.4192	0.6705	0.8007
0.7203	0.1863	0.6852	0.4173	0.9683

Richiami: vettorizzazione

Data una matrice A di ordine $p \times q$, il comando $A(:)$ *vettorizza* la matrice, cioè crea un vettore in cui sono memorizzati gli elementi di A ordinati per colonna.

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

```
A =
```

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

```
>> x = A(:)
```

```
x =
```

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

Il comando `reshape(x,[m,n])` riordina i dati contenuti nell'array x su m righe ed n colonne.

```
>> reshape(x,[2,3])
```

```
ans =
```

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

```
>> reshape(x,[2,4])
```

Error using `reshape`

To RESHAPE the number of elements must not change.

MatLab memorizza i dati contenuti negli array per colonne, a differenza del C che memorizza gli array per riga.

Uso di fprintf e della formattazione

Il comando `fprintf` consente di stampare diversi formati di dati:

<code>%g</code>	automaticamente riconosce il tipo di dato (sconsigliato)
<code>%d</code>	numero intero
<code>%f</code>	floating point
<code>%e</code>	formato esponenziale
<code>%c</code>	singolo carattere
<code>%s</code>	stringa

Caratteri speciali

<code>%%</code>	simbolo percento
<code>\n</code>	va a capo
<code>\t</code>	tabulazione orizzontale

È possibile scegliere quante cifre stampare dopo la virgola: il comando

```
fprintf('%0.5f \n', pi)
```

stampa le prime 5 cifre decimali di π e va a capo. Utilizzando invece

```
fprintf('%4d \n', 32)
```

si riservano 4 spazi per la stampa di un numero intero.

Strutture

I dati di tipo struct sono le classiche strutture comuni ai linguaggi di programmazione. Esse sono array multidimensionali con all'interno vari campi, che possono essere di vario tipo.

Creazione e riempimento di una struttura

Per esempio, se si vuole creare la struttura S con i campi nome, cognome e anno, si procede nel seguente modo

```
% Si crea una struttura con i campi Nome, Cognome ed Anno  
% inizializzati ai valori tra parentesi graffe  
>> S = struct('Nome',{'Cleve'},...  
              'Cognome',{'Moler'},...  
              'Anno',{39});
```

Per accedere ad un campo della struttura si usa la notazione
nome_struttura.nome_campo:

```
>> S.Nome  
ans =  
Cleve
```

Strutture

Per modificare il valore di un campo si utilizza la sintassi
`nome_struttura.nome_campo = nuovo_valore.`

Modifica valore di campo

Per modificare i campi della struttura precedentemente create si usano i seguenti comandi:

```
>> S.Nome = 'Jack';  
>> S.Cognome = 'Little';  
>> S.Anno = 19;
```

È possibile aggiungere dinamicamente campi a una struttura.

Aggiunta di campi

```
>> S.Titolo = 'Presidente';
```

Si può creare una struttura in maniera dinamica ex novo.

Strutture

Si possono creare vettori di strutture, sfruttando l'allocazione dinamica della memoria di MatLab.

Vettori di strutture

```
S = struct('Nome',{},...% crea una struttura vuota
           'Cognome',{},...
           'Anno',{});

S(1).Nome    = 'Frank';
S(1).Cognome  = 'Herbert';
S(1).Anno    = 1920;

S(2).Nome    = 'Isaac';
S(2).Cognome  = 'Asimov';
S(2).Anno    = 1920;

S(2) % visualizza i campi dell'elemento 2
```

Cell array

I cell array sono particolari tipi di dato che possono contenere dati di qualsiasi tipo. La creazione di questo tipo di dato segue una notazione *vettoriale*.

Creazione di cell array ed accesso ai suoi elementi

Nel seguente codice viene creato un cell array di 4 elementi, organizzati come una tabella a due righe e due colonne.

```
>> A = cell(2,2);    % Viene inizializzato il cell array
>> A{1,1} = 'ciao';  % elemento di posto 1,1
>> A{1,2} = 4;       % elemento di posto 1,2
>> A{2,1} = [1,2];   % elemento di posto 2,1
>> A{2,2} = [4;3];   % elemento di posto 2,2
>> A                 % Struttura simile a quella delle matrici
A =
    'ciao'          [
    [1x2 double]    [2x1 double]
>> A{1,2} % l'accesso avviene tramite le parentesi graffe
ans =
    4
>> A{2,2}
ans =
    4
    3
```

Gestione I/O

Per poter gestire in maniera ottimale gli input e gli output delle funzioni si possono usare i seguenti comandi: `varargin`, `nargin`, `varargout`, `nargout`.

- `varargin` gestisce le variabili in input, di cui a priori non si sa il numero;
- `nargin` gestisce il numero di dati in input;
- `varargout` gestisce le variabili in output, di cui a priori non si sa il numero;
- `nargout` gestisce il numero di dati in output.

Uso di `nargin` e `nargout`

Se nella funzione `[x1,x2] = roots_2deg(a,b,c)` (vista nelle prime slide) si utilizzassero i comandi

```
fprintf('Numero di parametri in ingresso : %d\n',nargin);  
fprintf('Numero di parametri in uscita   : %d\n',nargout);
```

si otterrebbe

```
Numero di parametri in ingresso : 3  
Numero di parametri in uscita   : 2
```

`varargin` e `varargout` sono variabili di tipo `cell`

Gestione I/O: esempio

```
function [a,varargout] = prova_args(x,y,varargin)
%
% Function MatLab creata per la comprensione
% dell'utilizzo di varargin, varargout, nargin e nargout

if nargin < 2
    error('Attenzione! Assegnare almeno 2 variabili in input !!!')
end

% Se in input viene dato anche il terzo parametro (opzionale)
% allora viene utilizzato, altrimenti viene settato il
% valore di default che in questo caso vale 1
if nargin == 3
    z = varargin{1};
else
    z = 1;
end

c = x+y;
b = x-y;
a = c^z;
```

...continua nella prossima slide...

Gestione I/O: esempio

```

switch nargout
case 0
    fprintf('Funzione chiamata senza variabili in output\n');
case 1
    % niente da fare (primo output assegnato di default)
case 2
    varargout{1} = c;
case 3
    varargout{1} = c;
    varargout{2} = b;
otherwise
    % nel caso in cui il numero di parametri
    % richiesti non rientri nella casistica considerata
    error('Attenzione! Numero massimo di output : 3 !!!')
end % end dello switch
end % end della function

```

Il primo if-then-else controlla il numero di parametri in entrata, che in questo esempio deve essere almeno 2. Il secondo controlla se è presente il parametro di input opzionale.

Il costrutto switch gestisce l'output in base a quanti dati in uscita sono stati richiesti nella chiamata alla funzione.

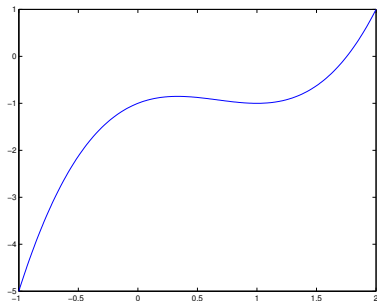
Grafica 2D

Si supponga di dover disegnare il grafico del polinomio

$$p(x) = x^3 - 2x^2 + x - 1$$

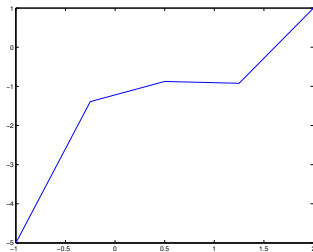
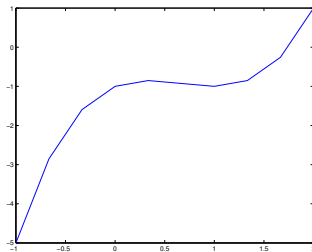
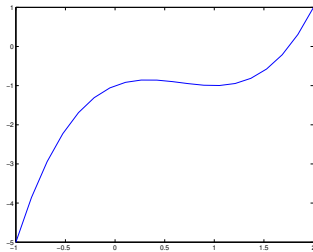
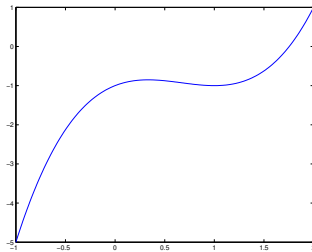
nell'intervallo $[-1, 2]$. In MatLab è presente l'istruzione `plot` che consente di creare grafici bidimensionali. Ricordando la sintassi vettoriale di MatLab, si ha:

```
>> x = linspace(-1,2,50);  
>> y = x.^3-2*x.^2+x-1;  
>> plot(x,y)
```



MatLab collega con una linea i punti $[x(i), y(i)]$ e $[x(i+1), y(i+1)]$: più punti di discretizzazione vengono utilizzati, migliore qualità visiva avrà il grafico.

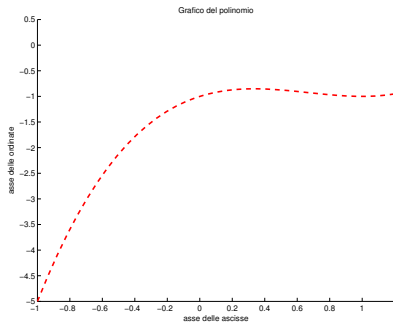
Grafica 2D

 $n = 5$  $n = 10$  $n = 20$  $n = 50$

Grafica 2D

È possibile personalizzare in vari modi il grafico:

```
x = linspace(-1,2,50);  
y = x.^3-2*x.^2+x-1;  
plot(x,y,'r--','Linewidth',2)  
box off  
axis([-1 1.2 -5 0.5])  
xlabel('asse delle ascisse')  
ylabel('asse delle ordinate')  
title('Grafico del polinomio')
```



Nel dettaglio:

- ▶ `axis([x1 x2 y1 y2])` consente di limitare la visualizzazione tra `x1` e `x2` per l'asse delle ascisse e fra `y1` e `y2` per l'asse delle ordinate;
- ▶ l'opzione `r--` consente di usare il colore rosso e di usare una linea tratteggiata;
- ▶ i comandi `xlabel` e `ylabel` consentono di mettere etichette all'asse delle `x` e delle `y`, rispettivamente;
- ▶ `title` consente di inserire una stringa come titolo del grafico.

Grafica 2D

Si supponga di dover plottare le seguenti funzioni

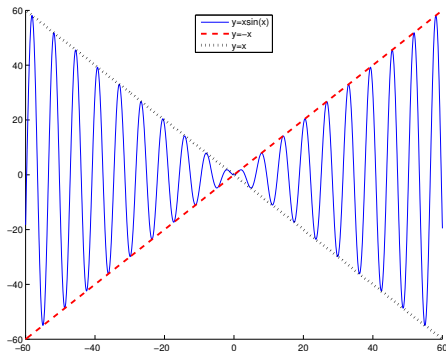
$$f(x) = x \sin(x)$$

$$g(x) = x$$

$$h(x) = -x$$

nell'intervallo $[-20\pi, 20\pi]$.

```
x = -20*pi:0.1:20*pi;
f = x.*sin(x);
h = x;
g = -x;
plot(x,f)
hold on
plot(x,h,'r--','Linewidth',2)
plot(x,g,'k:', 'Linewidth',3)
axis([-60 60 -60 60])
box off
legend('y=xsin(x)',...
      'y=-x',...
      'y=x',...
      'Location','North');
```



Grafica 2D

I tre puntini ... consentono di scrivere un'istruzione MatLab su più righe, ma il programma legge l'intera istruzione come se fosse su di una riga unica.

Vediamo nel dettaglio i vari comandi.

- ▷ **Linewidth** consente di specificare lo spessore della linea. Di default è 1.
- ▷ dopo aver dichiarato la variabile indipendente (x) e quella dipendente (f), si possono specificare i colori, lo stile e i marker dei punti.
 - le lettere **r,b,k,c,y,m** identificano i colori della linea: red, blue, black, cyan, yellow, magenta;
 - le scritture **-, .-, :, --** identificano lo stile della linea: continuo, punto-linea, punteggiata, tratteggiata.
 - le scritture **o,+,s,...** identificano lo stile dei markers: tondo, +, quadrato, etc...

È possibile inserire tutte queste opzioni in un'unica chiamata: per esempio **'hm,-'** disegna una linea di color magenta, con markers esagonali e una linea punteggiata e tratteggiata.

- ▷ **axis([x1 x2 y1 y2])** consente di limitare la visualizzazione tra x1 e x2 per l'asse delle ascisse e fra y1 e y2 per l'asse delle ordinate;
- ▷ **legend** consente di disegnare la legenda del grafico, mettendo fra apici le descrizione delle linee del grafico *nell'ordine in cui son state plottate*.
- ▷ **box off** consente di eliminare la "scatola" che appare attorno al grafico

Grafica 2D

Supponiamo di avere i seguenti dati :

x	y
147	1600
150	1300
220	1800
282	1900
312	2400
374	2600
412	2300
423	2600
457	2700
583	2800
602	2900
623	3100

Si vogliono plottare nello stesso grafico i punti (x_i, y_i) e la retta di regressione

$$y = 3.2x + 1092.88$$

introducendo nel grafico un titolo (*Dati relativi alle vendite*), un'etichetta per l'asse delle x (m^2), un'etichetta per l'asse delle y (*Volume di vendita in euro*) e una legenda (composta da due voci: dati, modello). Inoltre, dare stili diversi alla retta e ai punti.

Grafica 2D

```
% Dati
```

```
x = [147,150,220,282,312,374,412,423,457,583,602,623];
```

```
y = [1600,1300,1800,1900,2400,2600 2300,2600,2700,2800,2900,3100];
```

```
% Retta di regressione
```

```
r = 3.2*x + 1092.88;
```

```
% plot dei punti
```

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

```
% sovrascrivo i grafici
```

```
hold on
```

```
% plot della retta
```

```
plot(x,r,'r')
```

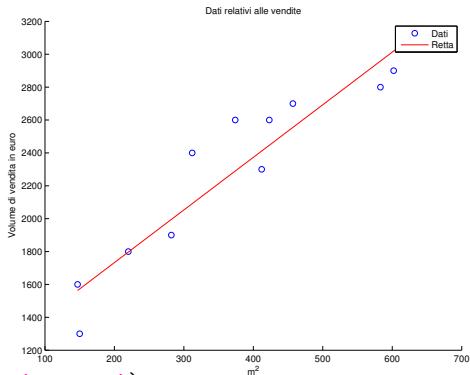
```
box off
```

```
xlabel('m^2')
```

```
ylabel('Volume di vendita in euro')
```

```
title('Dati relativi alle vendite')
```

```
legend('Dati','Retta')
```



Grafica 2D

```
% Dati
```

```
x = [147,150,220,282,312,374,412,423,457,583,602,623];
```

```
y = [1600,1300,1800,1900,2400,2600 2300,2600,2700,2800,2900,3100];
```

```
% Retta di regressione
```

```
r = 3.2*x + 1092.88;
```

```
% plot dei punti
```

```
plot(x,y,'o',...
```

```
'MarkerSize',10,...
```

```
'MarkerEdgeColor','k',...
```

```
'Linewidth',2)
```

```
hold on
```

```
% sovrascrivo i grafici
```

```
plot(x,r,'r-.',...
```

```
'Linewidth',2)
```

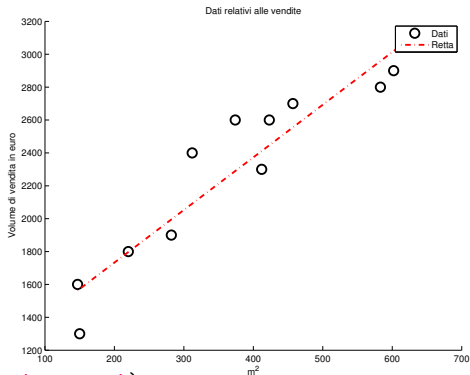
```
box off
```

```
xlabel('m^2')
```

```
ylabel('Volume di vendita in euro')
```

```
title('Dati relativi alle vendite')
```

```
legend('Dati','Retta')
```



Grafica 2D

```
% Dati
```

```
x = [147,150,220,282,312,374,412,423,457,583,602,623];
```

```
y = [1600,1300,1800,1900,2400,2600 2300,2600,2700,2800,2900,3100];
```

```
% Retta di regressione
```

```
r = 3.2*x + 1092.88;
```

```
plot(x,y,'d',...
```

```
'MarkerSize',10,...
```

```
'MarkerEdgeColor','b',...
```

```
'MarkerFaceColor','g',...
```

```
'Linewidth',2)
```

```
hold on
```

```
% sovrascrivo i grafici
```

```
plot(x,r,'m--',...
```

```
'Linewidth',2)
```

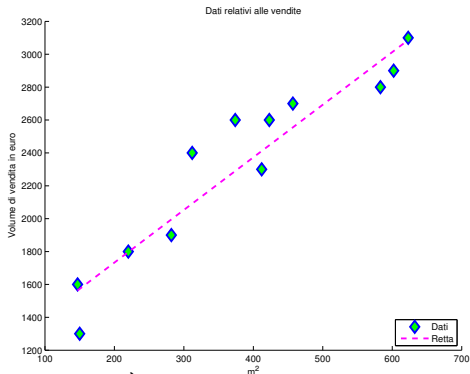
```
box off
```

```
xlabel('m^2')
```

```
ylabel('Volume di vendita in euro')
```

```
title('Dati relativi alle vendite')
```

```
legend('Dati','Retta','Location','SouthEast')
```



Grafica 2D

Una breve tabella riassuntiva di alcune opzioni per il comando plot.

Colore	Significato	Simbolo	Significato	Linea	Significato
w	bianco	.	punto	-	linea
y	giallo	o	circoletto	:	linea punteggiata
r	rosso	x	per	-.	punto e linea
g	verde	+	più	-	tratteggio
b	blu	*	stella		
k	nero	p	fiore		
m	magenta	s	quadrato		
c	ciano	d	rombo		
		h	esagono		
		v	triang. rovesciato		
		^	triangolo dritto		
		<	triangolo sinistro		
		>	triangolo destro		

Grafica 2D: subplot

Il comando `subplot(m,n,i)` consente di creare una griglia con `m` righe e `n` colonne. Alla `i`-esima posizione verrà posizionato un grafico con le caratteristiche elencate.

Un polinomio e le sue derivate

Si supponga di dover plottare nell'intervallo $[-5, 5]$ il seguente polinomio

$$p(x) = -x^5 - 0.2x^4 + 0.6x^3 - 2x^2 + x + 0.5$$

e le sue derivate. Le seguenti istruzioni in MatLab consentono di valutarlo nell'intervallo desiderato:

```
x = -5:0.01:5;  
p = [-1 -0.2 0.6 -2 1 0.5];  
y = polyval(p,x);
```

I seguenti comandi invece consentono di calcolarne le derivate in maniera compatta:

```
% derivata prima      % derivata terza      % derivata quinta  
dp=polyder(p);        d3p=polyder(d2p);    d5p=polyder(d4p);  
dy=polyval(dp,x);     d3y=polyval(d3p,x);  d5y=polyval(d5p,x);  
% derivata seconda    % derivata quarta    % derivata sesta  
d2p=polyder(dp);      d4p=polyder(d3p);    d6p=polyder(d5p);  
d2y=polyval(d2p,x);   d4y=polyval(d4p,x);  d6y=polyval(d6p,x);
```

Grafica 2D: subplot

<continua> Un polinomio e le sue derivate

```
subplot(2,3,1)
plot(x,y);
xlabel('x')
ylabel('y')
box off
title('p(x)')
```

```
subplot(2,3,2);
plot(x,dy);
xlabel('x')
ylabel('y')
box off
title('p^{(1)}(x)')
```

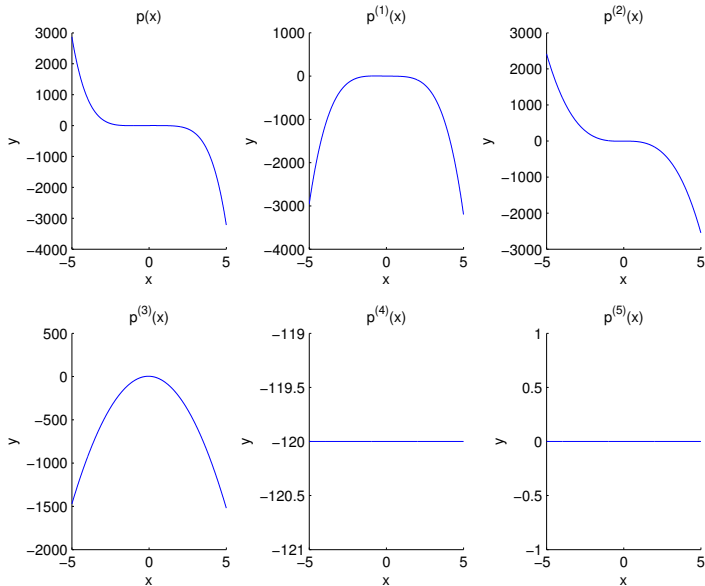
```
subplot(2,3,3);
plot(x,d2y);
xlabel('x')
ylabel('y')
box off
title('p^{(2)}(x)')
```

```
subplot(2,3,4);
plot(x,d3y);
xlabel('x')
ylabel('y')
box off
title('p^{(3)}(x)')
```

```
subplot(2,3,5);
plot(x,d5y);
xlabel('x')
ylabel('y')
box off
title('p^{(4)}(x)')
```

```
subplot(2,3,6);
plot(x,d6y);
xlabel('x')
ylabel('y')
box off
title('p^{(5)}(x)')
```

Grafica 2D: subplot



Grafica 2D: figure

Il comando `figure` consente di creare una nuova finestra grafica.

```
x = -2:0.01:1;
y = x.^2.*sin(x);
plot(x,y) % plotta il grafico di y in una finestra
figure;   % crea una finestra vuota
y2 = sin(x);
plot(x,y2) % plotta il grafico di y2 nella finestra appena creata
```

Se è necessario agire su una finestra già creata si usa il comando `figure(N)`.

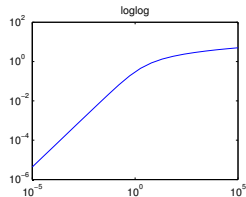
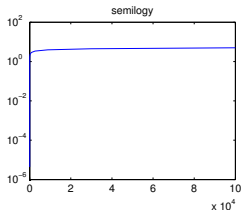
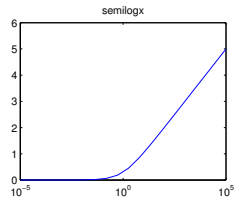
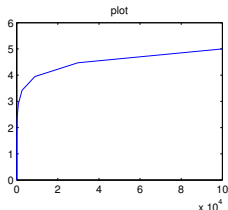
```
x = -2:0.01:1;
y = x.^2.*sin(x);
figure(1)
plot(x,y)
figure(2);
y2 = sin(x);
plot(x,y2)
figure(1)
title('Primo test')
figure(2)
xlabel('x')
```

Il comando `figure(N)` consente di creare una nuova figura "etichettata" `N`, e, una volta creata, si può lavorare su tale finestra richiamandola tramite lo stesso comando.

Grafica 2D: semilog

I comandi `semilogx`, `semilogy`, `loglog` consentono di plottare i grafici utilizzando una scala logarithmica sull'asse delle x , delle y e su entrambe, rispettivamente.

```
x = logspace(-5,5,20);  
y = log10(x+1);  
subplot(2,2,1)  
plot(x,y)  
title('plot')  
  
subplot(2,2,2)  
semilogx(x,y)  
title('semilogx')  
  
subplot(2,2,3)  
semilogy(x,y)  
title('semilogy')  
  
subplot(2,2,4)  
loglog(x,y)  
title('loglog')
```



Grafica 2D: altri comandi utili

text

`text(x,y,str)` inserisce la stringa specificata nel punto di coordinate (x,y) ; una variante di questa istruzione è `gtext(str)`, che inserisce la stringa specificata nel punto in cui si clicca con il mouse. In `legend`, `xlabel`, `ylabel`, `text`, `title` la stringa può contenere simboli specificati mediante comandi TeX.

`fill(x,y,colore)` crea un poligono di vertici aventi coordinate $x(i), y(i)$ e lo riempie con il colore specificato; l'istruzione chiude il poligono

print -dps NOMEFILE.ps

Permette di salvare l'immagine che si trova nella finestra grafica corrente in un file PostScript di nome assegnato NOMEFILE.

print -depsc2 NOMEFILE2.eps

Permette di salvare l'immagine che sta nella finestra grafica corrente in un file di nome NOMEFILE2, il cui contenuto è un'immagine a colori in formato Encapsulated Level 2 PostScript.

Grafica 2D: altri comandi utili

axis

<code>v = axis</code>	nel vettore <code>v</code> è riportata la scala usata
<code>axis auto</code>	ritorna alla modalità di scalatura automatica
<code>axis equal</code>	fissa la stessa unità di misura sui due assi
<code>axis square</code>	rende il sistema di riferimento quadrato, usando unità di misura diverse sui due assi
<code>axis on; axis off</code>	abilita e disabilita la visualizzazione degli assi, con le etichette per gli assi
<code>axis ij; axis xy</code>	pone l'origine degli assi in alto a destra, con valori crescenti dell'asse <code>y</code> verso il basso; <code>axis xy</code> è la modalità normale

grid on \\ grid off \\ grid

Abilita o disabilita la visualizzazione di una griglia nel piano cartesiano; `grid` da solo permette di vedere se la specifica è abilitata o meno.

Grafica 2D: circonferenza

L'equazione cartesiana della circonferenza di raggio r è

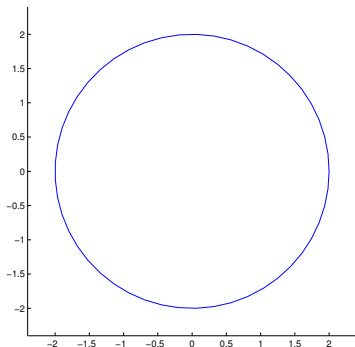
$$x^2 + y^2 = r^2$$

La sua forma parametrica è quindi

$$\begin{cases} x = r \cos(\theta) \\ y = r \sin(\theta) \end{cases}$$

con $\theta \in [0, 2\pi)$.

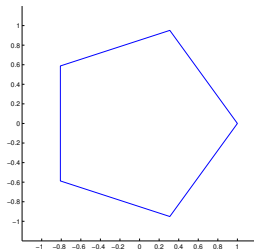
```
t = linspace(0,2*pi,100);  
r = 2;  
x = r*cos(t);  
y = r*sin(t);  
plot(x,y)  
axis square  
axis(1.2*[-r r -r r])  
box off
```



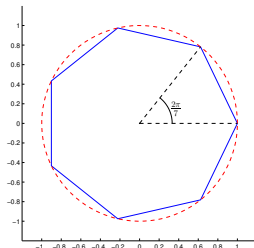
Grafica 2D: poligono regolare

Per disegnare un poligono di n lati, si prendono $n + 1$ punti equidistanti sulla circonferenza unitaria.

```
t = linspace(0,2*pi,6);
r = 1;
x = r*cos(t);
y = r*sin(t);
plot(x,y)
axis square
axis(1.2*[-r r -r r])
box off
```



```
t = linspace(0,2*pi,8);
r = 1;
x = r*cos(t);
y = r*sin(t);
plot(x,y)
axis square
axis(1.2*[-r r -r r])
box off
```

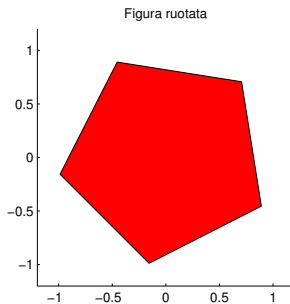
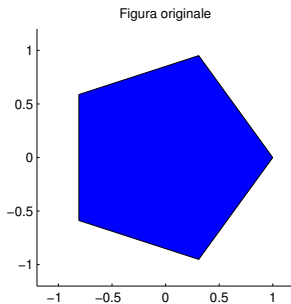


Grafica 2D: matrici di rotazione

Dal corso di Matematica Discreta. Una matrice del tipo

$$A = \begin{pmatrix} \cos(\varphi) & -\sin(\varphi) \\ \sin(\varphi) & \cos(\varphi) \end{pmatrix}$$

è una matrice di rotazione: considerando un vettore x , il prodotto $y = Ax$ corrisponde alla rotazione di x attorno all'origine di un angolo φ in senso antiorario.



Grafica 2D: matrici di rotazione

```
% numero di lati
n = 5;

% Angolo di rotazione
phi = pi/4;
% Matrice di rotazione
A = [cos(phi) -sin(phi)
sin(phi) cos(phi)];

t = linspace(0,2*pi,n+1);
```

```
% Matrice contenente le
% coordinate dei punti:
% nella prima riga si salvano
% le ascisse dei punti,
% sulla seconda le ordinate.
% L'i-esima colonna
% contiene le coordinate
% dell'i-esimo punto
P = zeros(2,n+1);
P(1,:) = cos(t);
P(2,:) = sin(t);
```

```
% Plot poligono originale
subplot(1,2,1)
fill(P(1,:),P(2,:), 'b');
hold on;
plot(P(1,1),P(2,1), 'pk');
axis square
axis([-1.2 1.2 -1.2 1.2])
box off
title('Figura originale')
```

```
% Rotazione
P_rot = A*P;

% Plot poligono ruotato
subplot(1,2,2)
fill(P_rot(1,:),P_rot(2,:), 'r');
hold on;
plot(P_rot(1,1),P_rot(2,1), 'pk');
axis square
axis([-1.2 1.2 -1.2 1.2])
box off
title('Figura ruotata')
```