

MATLAB[©]

- MATrix LABoratory
- Software esplicitamente realizzato per manipolazione di matrici
- Interattivo → Interprete di comandi
- Elemento base: matrice (che non richiede dimensionamento)
- Sviluppato agli inizi degli anni '80, come sottoprodotto dei progetti LINPACK ed EISPACK, da The MathWorks Inc.
- Disponibilita' del Laboratorio: Matlab 4.2 + Simulink su workstation SUN + Solaris.
- Esistono altre versioni di Matlab:
 - PC-Matlab
 - AT-Matlab (minimo 2 MB di RAM)
 - Matlab-386 (per PC-386 - memoria estesa)
- Le principali differenze tra le versioni sono nel numero massimo di elementi per vettore, nella sintassi di alcuni comandi e nel linguaggio con cui è stato sviluppato il SW. Fino alla versione 3.5 per Windows e per tutte quelle sotto DOS, il codice è stato sviluppato in C, mentre dalla versione 4.0 in poi si è fatto uso di C++. E' anche cambiata la filosofia del programma: mentre nelle prime versioni MATLAB appariva all'utente come un programma unico che eventualmente poteva chiamare procedure esterne, ora le funzionalità di calcolo, grafiche ecc. del programma possono essere richiamate da programmi esterni.
- All'utente "medio" MATLAB appare comunque come un ambiente integrato di calcolo, che include capacità grafiche, aritmetica IEEE e molte librerie speciali.

LABORATORIO

- Stazioni SUN - SparcClassic - alcune con drive per dischetti da 3.5 " oppure terminali X.
- Si deve prima avere una **login** e una **password**. La procedura per ottenerle è differenziata a seconda che si lavori a PD o a VI.
 - PD: seguire le istruzioni contenute all'interno delle dispense del corso di Elettronica Industriale.
 - VI: fare richiesta all'addetto nei giorni stabiliti
- La stazione SUN e' sempre accesa (led verde sul monitor) anche se il monitor risulta spento (screen saver...☺). Premere **ENTER** e tutto si riattiva.
- Viene richiesta la **login** e la **password**: introdurre i codici assegnati
- A questo punto partirà automaticamente l'ambiente a finestre. Se ciò non avvenisse, digitare il comando **openwin**.
- Da una finestra comandi, digitare il comando **matlab** per avviare il programma. Una volta avviato, matlab stampa il carattere di prompt:
 - »
- Per scrivere i programmi, usare gli editor di sistema (vi, emacs, textedit etc.).
- Si possono salvare dati e programmi su dischetto (solo sulle macchine con drive) col comando: **mcopy nome_file a:**
- Il dischetto viene espulso col comando **eject**.
- Se si devono trasferire dati per poi elaborarli con un PC-compatibile, bisogna prima convertire da formato UNIX a DOS col comando:
 - unix2dos nome_file_unix nome_file_dos**
- Se si deve importare un file DOS in UNIX, usare il comando **dos2unix** (per maggiori dettagli usare il manuale in linea)

MATRICI

- Come introdurre:
 - a) assegnazione di lista di elementi
 - b) generate da funzioni
 - c) create con programma (m-file)
 - d) caricate da file esterni (di dati)

a) Scriviamo:

```
» A=[1 2 3; 4 5 6; 7 8 9]↵
```

MATLAB risponde con

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

Se si mette un punto e virgola (;) dopo l'assegnazione, non c'è "echo"

Si può fare l'assegnazione anche su righe diverse:

```
A = [1 2 3 ↵      ← qui non ricompare il prompt  
     4 5 6 ↵      ← qui non ricompare il prompt  
     7 8 9];↵  
»
```

b) Vedremo più avanti

c) Si scrive il programma MATLAB che fa l'assegnazione della variabile. I file dei programmi devono avere sempre estensione **.m**.

Esempio: editare (con vi, emacs, textedit...) il file **gena.m** che assegna un certo valore (matriciale) alla variabile A. L'operazione va fatta al di fuori del programma MATLAB

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

Salvare il file **gena.m** ed avviare MATLAB. Per lanciare il programma gena.m, basta digitare il nome del programma (senza estensione) dalla linea comandi.

```
» gena↵
```

Si puo' verificare l'avvenuta assegnazione digitando il nome della variabile dalla linea comandi di MATLAB

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

N.B. E' cosa "astuta" non chiamare le variabili con nomi di programmi MATLAB :-)

I file di tipo .m contengono in generale sequenze di comandi elementari o chiamate a funzioni (si vedranno in seguito). MATLAB e' un interprete di questi comandi.

d) Per salvare una lista di variabili, si usa il comando
save nome_file lista_variabili

che salva le variabili scelte nel file nome_file.mat. Il formato di salvataggio puo' essere scelto (dare il comando **help save** dall'interno di MATLAB per dettagli)

Per recuperare dati salvati in precedenza (variabili col loro nome) con MATLAB o per importare file di dati creati da altri programmi usare
load nome_file

(usare **help load** per dettagli sui formati ammessi)

ELEMENTI DELLE MATRICI

- Possono essere una qualsiasi espressione MATLAB

Es: all'assegnazione

```
x = [-1.3 sqrt(3) (1+2+3)*4/5]
```

MATLAB risponde con

```
x =  
-1.3000  1.7321  4.8000  
»
```

N.B.1 Esistono vari formati per i dati visualizzati:

format short	4 decimali in virgola fissa
format short e	4 decimali + esponente base 10
format long	14 decimali
format long e	14 decimali + esponente
format hex	esadecimale
format ++	presenta solo il segno dei numeri

N.B.2 Le funzioni MATLAB vanno tutte scritte in minuscolo, mentre alle variabili si possono associare nomi di 19 caratteri al massimo. Inoltre, di default:

A ≠ a !!!! (comunque si puo' cambiare)

- Ogni elemento della variabile **x** precedentemente dichiarata (e contemporaneamente istanziata) puo' essere letto/scritto indipendentemente dagli altri. L'accesso avviene tramite selezione con indice. **ATTENZIONE**: gli indici partono da 1!

```
x(1)↵
```

```
ans=  
-1.3000
```

- ans e' la variabile di "risposta" (answer) ad una richiesta di valore di una certa variabile o di una funzione

- E' utile anche come variabile intermedia:

Es:

```

» x(1)↵
      ans = -1.3000
» ans * ans↵
      ans = 1.6900
»

```

- **ans** e' in generale la risposta di MATLAB quando manca l'assegnazione ad una variabile.
- In MATLAB non e' necessario provvedere al dimensionamento delle matrici (allocazione dinamica).

Es.:

```

» x = [-1  2  3];↵
» x(5) = abs(x(1))↵      dara' come risultato

```

```

x=
-1.0000  2.0000  3.0000  0.0000  1.0000

```

con l'elemento nella quarta posizione aggiunto automaticamente e posto a zero.

N.B. Un "trucco" per rendere piu' veloci le simulazioni: se si deve creare un vettore molto lungo aggiungendo un elemento ad ogni iterazione, conviene predisporre all'inizio di tutte le iterazioni un vettore della lunghezza massima richiesta con gli elementi tutti a zero (istruzione **zeros**, vedere help in linea).

- Matrici "grandi" si possono ottenere per accostamento di piu' matrici "piccole". Ad esempio. partendo dalla matrice A precedente:

```

» A = [A; [10  11  12]]      (nelle assegnazioni di matrici
                             si usano le parentesi quadre)

```

```

A=
 1  2  3
 4  5  6
 7  8  9
10 11 12

```

- Per le matrici, l'accesso avviene tramite 2 indici, il primo per la riga, il secondo per la colonna:

» $A(2,3)$

ans = 6

- Matrici "piccole" si possono estrarre da matrici grandi. Si ricorda che le matrici sono al massimo 2-D.

» $A = A(1:3, :)$ questa operazione ri-istanza A con gli elementi di tutte le colonne, dalla prima alla terza riga

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

N.B. I due punti (:) sono un operatore:

Generazione di sequenze

» $t = (1:10)$
 $t = 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10$

Selezione di un gruppo di elementi

» $a = t(1:3)$
 $a = 1 \ 2 \ 3$

Selezione di tutti gli elementi:

» $b = a(:)$
 $b = 1 \ 2 \ 3$

Generazione di sequenze con passo diverso da 1 (anche <0)

» $t = (1:2:9)$ (i numeri tra 1 e 9, passo 2)

t = 1 3 5 7 9

ESPRESSIONI

- 1) var = expr "assegnazione" (se sta su piu' di una riga, usare tre punti "...")
- 2) expr calcolo diretto ed assegnazione ad "ans"

VARIABILI

- Nomi riservati: tutti quelli delle funzioni
- Lista VARIABILI: si ha digitando il comando "who". Dice anche lo spazio rimanente in memoria (nelle versioni PC)
- Variabili speciali:

eps = 2^{-52} rappresenta l'accuratezza

∞ infinito

NaN risultato di 0/0 (Not A Number)

pi π

- Esempio:

» s = 1/0

s =

∞

Warning: divide by zero

(MATLAB dà indicazioni su possibili errori commessi durante l'elaborazione, come ad esempio la divisione per zero...)

- Il comando "whos" dà la lista variabili e la loro dimensione (in termini numero righe e colonne della matrice)

NUMERI E ARITMETICA

- Range: da 10^{-308} a 10^{+308}
- Accuratezza: 16 cifre decimali
- Operatori: +, *, -, / (divisione destra), \ (divisione sinistra), ^ (elevamento a potenza)

Esempio: $1/4 = 0.25$
 $1\backslash 4 = 4$

- Uso "normale" delle parentesi per cambiare la priorità tra gli operatori
- MATLAB consente l'uso di numeri complessi: si usa il simbolo "i" o "j" per indicare la parte immaginaria. Nelle versioni "vecchie" di MATLAB per PC, bisogna fare una dichiarazione "esplicita" dell'operatore complesso:

$i = \text{sqrt}(-1);$

- La dichiarazione di variabili complesse avviene moltiplicando la parte complessa per "i" o "j" e sommando la parte reale:

$\gg z = 3+4*i;$ (niente spazi tra i termini)

N.B. E' equivalente usare $z=3+4i;$

- Tutte le funzioni MATLAB accettano variabili di tipo complesso:

$\gg w = r*\text{exp}(i*\text{theta})$

HELP

- Il comando "help" dà la lista degli argomenti (funzioni, operatori, toolbox) disponibili. Funziona in maniera leggermente diversa nelle varie versioni di MATLAB.
- Il comando "help nome_comando" dà una breve spiegazione a video sul funzionamento del comando richiesto.

Esempio:

help eig spiega la sintassi e il modo di operare della
funzione che calcola gli autovalori di una
matrice

help + spiega come usare l'addizione

COMANDI UNIX E LETTURA DIRECTORY

- » ! comando_unix esegue comando_unix (vale anche per DOS)
- » dir elenco file directory
- » dir *.m elenco m-file
- » dir *.ext elenco file con estensione "ext"
- » type nome_file visualizza il contenuto del file
- » delete file cancella il file
- » chdir dir_name cambia directory di lavoro (non usare !cd...
non funziona come ci si aspetta)

Per un sommario dei comandi UNIX vedere la tabella allegata

OPERAZIONI SU MATRICI (FUNZIONI)

- In parte built-in (non visibili ma usabili)

esempio: `sin`, `cos`, `exp`, `'` (operatore di trasposizione di matrici)

- Le altre definite in m-file (file di testo che contiene elenco di comandi elementari MATLAB, funzioni built-in o chiamate ad altri m-file)

esempio: `fft.m`, `log10.m` etc...

- L'operatore di trasposizione è l'apice (apostrofo) `'`. (da non confondersi con l'accento ```).

Esempio:

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

```
» B = A'
```

```
B=
```

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

```
» x = [1  2  3]'
```

```
x=
```

```
 1
 2
 3
```

N.B. Un vettore riga viene "naturalmente" trasformato in una colonna (e viceversa)

- Se z è una matrice complessa (cioè gli elementi sono numeri complessi), z' è la matrice complessa coniugata trasposta
- Per ottenere solo la trasposta, bisogna usare:

`conj(z')`

ADDIZIONE E SOTTRAZIONE

- Tra elementi con le stesse dimensioni:

Esempio:

$$\gg A = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix};$$

$$\gg B = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix};$$

$$\gg A+B$$

ans =

$$\begin{bmatrix} 3 & 3 \\ 3 & 3 \end{bmatrix}$$

$$\gg A-B$$

ans =

$$\begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}$$

- Si possono fare addizioni e sottrazioni tra una matrice (o vettore) ed uno scalare

Esempio:

$$\gg x = [1 \quad 2 \quad 3];$$

$$\gg y = x + 1$$

$$y = \begin{matrix} 2 \\ 3 \\ 4 \end{matrix}$$

Viene sommato 1 a ciascun elemento...

PRODOTTO

- Prodotto interno: •
 Matrice • Matrice
 Matrice • Vettore
 Matrice • Scalare

Le dimensioni devono essere corrette
 Moltiplica ciascun elemento

$$\text{es: } \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \bullet \lambda = \begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{bmatrix}$$

DIVISIONE

- Sia A invertibile (quadrata, non singolare):
 $A \setminus B$ equivale a: $A^{-1}B \equiv \text{inv}(A) * B$
 B / A equivale a: $BA^{-1} \equiv B * \text{inv}(A)$
- Si usa nella soluzione di sistemi di equazioni lineari:

$$\begin{array}{ll} x = B/A & \text{soluzione di } x * A = B \\ x = A \setminus B & \text{soluzione di } A * x = B \end{array}$$

POTENZE

- Sia A matrice quadrata, p uno scalare (non necessariamente intero)

$$A^p \quad \text{p-esima potenza di A}$$

$$A * A * A * \dots * A \quad \text{se p intero}$$

$$V * D.^p / V \quad \text{se p non intero}$$

dove V è la matrice di autovettori, D l'insieme degli autovalori e "." è un operatore su array che si descriverà più avanti.

FUNZIONI TRASCENDENTI

- Esempi:

`exp(A)`

`sqrt(A)`

eseguono rispettivamente l'esponenziale e la radice quadrata dei singoli elementi di A

- `expm(A)`
`logm(A)`
`sqrtn(A)`

operano invece sulle matrici nella loro globalità:

$$\text{expm}(A) \equiv I + A + A^2/2 + \dots$$

$$\text{sqrtn}(A) \equiv A^{(1/n)}$$

OPERAZIONI SUGLI ARRAY

- Finora gli operatori aritmetici visti agivano sulle matrici nella loro globalità
- Le operazioni sugli array si riferiscono ad operazioni aritmetiche eseguite elemento per elemento.
- Gli operatori su array si distinguono da quelli per matrici per il simbolo "." prima dell'operatore (niente spazio tra punto e operatore)
- Addizione (. $+$) e sottrazione (. $-$) sono le stesse...

- Moltiplicazione (.*) e divisione (./ oppure ./\) si differenziano:

» x = [1 2 3]; y = [4 5 6];

» z = x.*y

z = 4 10 8

» z = x./y

z = 4 2.5 2

OPERATORI RELAZIONALI E LOGICI

- Operatori relazionali: <, <=, >, >=, ==, ~=
- Operano una comparazione elemento per elemento. Il risultato è

1 se il test è TRUE

0 se il test è FALSE

- Operatori logici: & (and), | (or), ~ (not)
- Consentono di combinare più operatori relazionali
- Le variabili su cui operano vanno considerate

TRUE se diverse da 0

FALSE se uguali a 0

- I risultati sono

1 se TRUE

0 se FALSE

ALTRE ISTRUZIONI

- » clear nome_variabibile (cancella dalla memoria la variabile richiesta)

- `»x []` (crea una matrice vuota)
- `»A(:, [2 4]) = []` (cancella le colonne 2 e 4)

FUNZIONI AVANZATE

- Finora si sono viste funzioni semplici. Ve ne sono altre, molto più complesse, realizzate tramite m-file (elenco di comandi elementari) e raccolte in forma di librerie, chiamate "toolbox". Nella versione SUN, sono attualmente disponibili i seguenti toolbox:

control	per controllo
robust	per controllo robusto
signal	per elaborazione numerica di segnali
ident	per identificazione di sistemi
optim	per ottimizzazione (interpolazioni etc..)
nnet	reti neurali

- Vi sono centinaia di funzioni disponibili. Ne vedremo solo alcune, in particolare quelle legate alla simulazione di sistemi di controllo digitali. In particolare:

Utilità di calcolo su matrici
 Analisi dati
 Polinomi
 Signal processing
 Controllo di flusso
 Grafica
 Control Toolbox

UTILITA' DI CALCOLO SU MATRICI

- Decomposizione:

$$\text{Upper-lower} \quad \begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix} = \begin{bmatrix} \# & 0 & 0 \\ \# & \# & 0 \\ \# & \# & \# \end{bmatrix} \begin{bmatrix} \% & \% & \% \\ 0 & \% & \% \\ 0 & 0 & \% \end{bmatrix}$$

$$[U,L]=lu(A)$$

Fattorizzazione di Cholesky $A=L'L$

$$L = chol(A)$$

- Inversione di matrice

$$A = inv(B)$$

- Determinante di matrice

$$g = det(A)$$

- Rango di una matrice

$$r = rank(A) \quad (\text{trovato per via numerica: per matrici mal condizionate puo' dare risultati sbagliati})$$

- Autovalori ed autovettori di una matrice

$$g = eig(A) \quad \text{da` i soli autovalori (in vettore)}$$

$$[V,g] = eig(A)$$

N.B. Si noti che, quando la funzione ritorna piu` di un argomento, questi vanno messi tra parentesi quadre.

ANALISI DEI DATI

- Massimo elemento di un array:

$$a = max(v) \quad \text{da` il valore del massimo}$$

$[a,b] = \max(v)$ da` il valore del massimo e il suo indice

- Minimo elemento di un array

$a = \min(v)$

$[a,b] = \min(v)$ (analogia con max...)

- Media aritmetica tra elementi

$b = \text{mean}(V(1:10))$

- Matrice di correlazione

$R = \text{xcorr}(x,y)$

- Differenza prima: dato x vettore di n elementi ($x = [x(1) \ x(2) \ \dots \ x(n)]$)
ritorna $[x(2)-x(1) \ x(3)-x(2) \ x(n)-x(n-1)]$

$y = \text{diff}(x)$

- Dimensione di una matrice o di un vettore:

$d = \text{size}(A)$ restituisce un vettore le cui componenti sono le
dimensioni di A

$[a,b]=\text{size}(A)$ mette in a il num. righe e in b il num. colonne

SIGNAL PROCESSING E POLINOMI

- Determinazione del polinomio caratteristico di una matrice:

» $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix};$

» $p = \text{poly}(A)$ p e` il polinomio caratteristico di A

$$p = 1 \quad -6 \quad -72 \quad -27$$

Il vettore **riga** p contiene i coefficienti dei termini del polinomio di grado decrescente:

$$p(\lambda) = \lambda^3 - 6\lambda^2 - 72\lambda - 27$$

- Dato il polinomio tramite i suoi coefficienti, se ne possono calcolare le radici:

$$\gg r = \text{roots}(p)$$

$$r = \begin{array}{l} 12.1229 \\ -5.7345 \\ -0.3884 \end{array}$$

N.B. Il risultato viene dato in forma di vettore colonna. Si poteva ottenere anche attraverso la funzione eig(A).

- Date le radici del polinomio, e` possibile costruire il polinomio:

$$\gg pr = \text{poly}(r)$$

$$pr = 1 \quad -6 \quad -72 \quad -27$$

- Filtro numerico:

$$y = \text{filter}(b,a,x)$$

realizza la seguente trasformazione sui dati:

$$y(n) = b(1)x(n) + b(2)x(n-1) + \dots + b(nb)x(n-nb+1) - a(2)y(n-1) - \dots - a(na)y(n-na+1)$$

N.B. a(1) va sempre posto a 1!

che corrisponde alla seguente funzione di trasferimento discreta:

$$H(z) = \frac{b(1) + b(2)z^{-1} + \dots + b(nb)z^{-(nb-1)}}{1 + a(2)z^{-1} + \dots + a(na)z^{-(na-1)}}$$

- Fast Fourier Transform:

`y = fft(x)`
 a
 padding)

l'array deve avere un numero di elementi pari
 una potenza di 2 (altrimenti c'è zero-

- ...e tantissime altre (vedere manualistica e ultime pagine dispense)

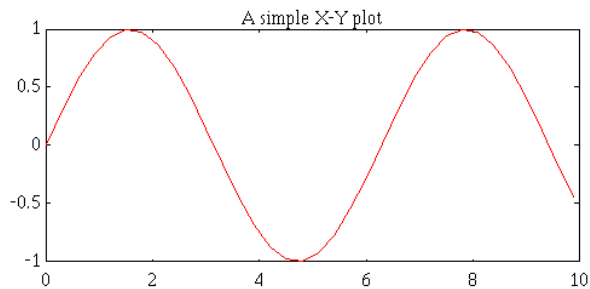
GRAFICA

- plot: grafici a scale lineari

`plot(x,y)`

grafico di y in funzione
 di x (stesse dimensioni)

Esempio: `t = 0:0.1:10`
`plot(t,sin(t))`



`plot([x' y'])`

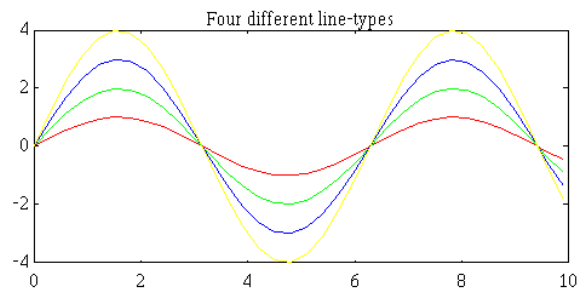
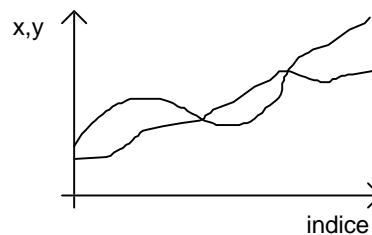
grafico di x e y in
 funzione dell'indice
 (manca il II argomento)

Esempio:

`a=sin(t);b=2*a;`

`c=3*a; d=4*a;`

`plot(t,[a' b' c' d'])`



Si possono usare diversi simboli per le tracce (utile quando si stampa in bn)

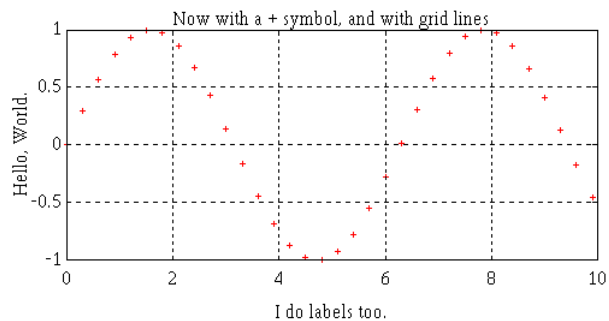
Si possono anche aggiungere griglie.

Esempio

```
plot(t,sin(t),'+')
```

```
grid
```

vedere help plot per dettagli



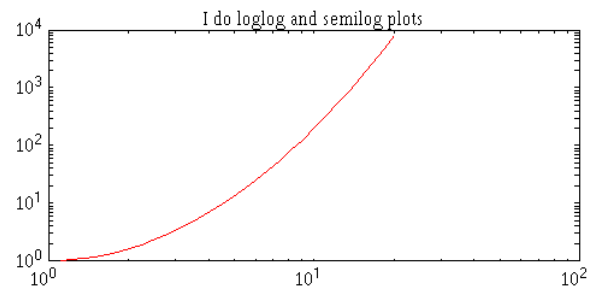
Si possono fare grafici con scale doppiamente logaritmiche

```
loglog(x,y)
```

o con un solo asse logaritmico

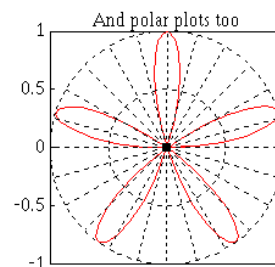
```
semilogx(x,y)
```

```
semilogy(x,y)
```



Si possono tracciare diagrammi polari

```
polar(r,theta)
```

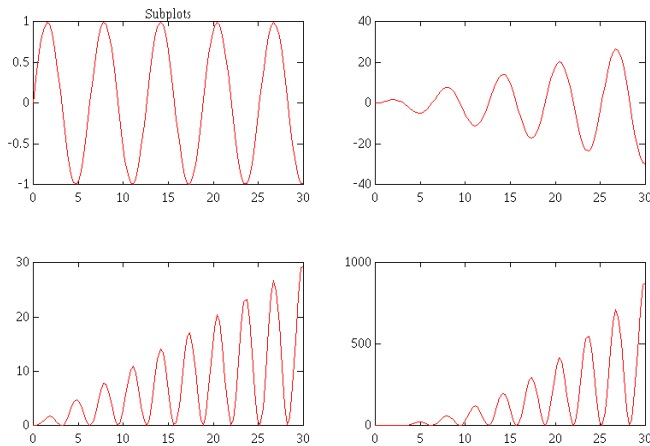


Si possono riportare più grafici sulla stessa videata, col comando subplot:

```
subplot(nmp), plot(t)
```

crea nxm grafici e abilita la scrittura del comando plot sul p-esimo di essi

Per ripristinare la normalità usare il comando subplot(111)



- `title('testo')`: pone la riga di testo nella parte superiore del grafico
- `xlabel('nome_asse_x')`, `ylabel('nome_asse_y')`: danno i nomi agli assi
- `axis` (vedere help): serve per selezionare il campo di visualizzazione (per zoomare)

STAMPA FIGURE

- Si possono stampare sia grafici MATLAB che finestre SIMULINK (questo programma verrà descritto più avanti)
- Comando `print` con diversi parametri, tra cui quelli che consentono di stampare in un formato particolare, tramite il programma `ghostscript`.
- La stampa deve sempre avvenire su file (per gli utenti SUN) in quanto non è abilitata la stampa grafica per gli studenti.
- Per la stampa di una finestra grafica MATLAB il comando è il seguente:
 - » `print -ddevice -fname_figura nome_file`
- Per la stampa di una finestra SIMULINK il comando è:
 - » `print -ddevice -snome_finestra nome_file`
- Le opzioni `-ddevice` sono le seguenti:

- dps Postscript
- deps Encapsulated Postscript
- dhpgl Formato Plotter HP 7475A
- dgif8 Formato gif a 8 bit
- dpcx256 Paintbrush a 256 colori
- dbj10e BubbleJet Canon
- dlaserjet HP Laserjet

- Allo stato attuale non funzionano invece -depson e -deps9high
- Si può stampare poi la figura da PC importando il file in un qualsiasi programma in grado di riconoscere i formati disponibili. Oppure si va in copisteria col dischetto contenente il file PS. **VEDERLO PRIMA CON IMAGETOOL!!!!**
- Ad esempio i formati PCX, PS, EPS e HPGL sono accettati da WORD per Windows

CONTROLLO DI FLUSSO

- Cicli "for"
 - » for i = 1:n
 - for j = 1:n
 - A(i,j) = i+j,
 - end ← metterli a fine ciclo!!!
 - end
 - » for i = 1:2:n ← incrementi di 2 ad ogni iterazione
- Cicli "while"
 - » while(alpha < beta)
 - alpha = alpha +1;
 - end
- Operazioni condizionate... "if"
 - » if (espressione)
 - istruzioni
 - elseif (espressione)

```
    istruzioni
else
    istruzioni
end
```

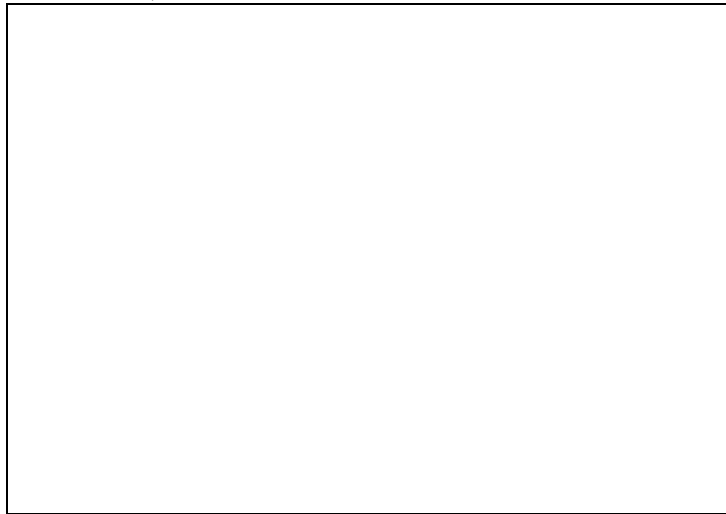
- Si può uscire in modo immediato da un ciclo con l'istruzione "break"
» while 1
 n = input('scrivi un numero');
 if n <= 0, break, end
 m = sqrt(n)
end

Questo programma stampa indefinitamente la radice del numero introdotto finchè non si scrive un numero minore o uguale a zero.

ISTRUZIONI AL MATLAB

- Modo interattivo: dal prompt si digitano in sequenza i comandi che si vogliono far eseguire a MATLAB. Il problema è che si perde tutto quando si esce dal programma. Esiste una funzione di registrazione (diary) che, però, registra anche gli eventuali errori.
- Modo batch: si crea un m-file che contiene la sequenza dei comandi.
- Suggerimento: usare una via di mezzo, ovvero provare in modo interattivo piccole sequenze di comandi e quando si è sicuri che funzionano, riportarle su un m-file.
- Un modo per rendere più "modulare" il sw per MATLAB è quello di creare un insieme di funzioni. Ancora, alcune elaborazioni che vanno fatte una volta per tutte, dovrebbero salvare i dati su un file .mat. I programmi successivi dovrebbero poi recuperare i dati già elaborati, con notevole risparmio di tempo.

- E' possibile rendere "interattivi" i programmi MATLAB attraverso una serie di funzioni che gestiscono l'input da tastiera (vedere help input)
- Per le versioni 4.x per SUN e PC esiste una modalità "debug" che consente di eseguire tratti separati di programma, mettere breakpoint, vedere le chiamate a funzione (vedere help debug)
- C'è anche una potentissima grafica 3-D, che a noi non serve ma che si può andare a vedere (comandi mesh, surface e altri.. vedere help)



CONTROL TOOLBOX

- E' un insieme di m-file che realizzano funzioni utili per l'analisi, il progetto e la simulazione di sistemi dinamici con controllo a retroazione.
- Il CONTROL TOOLBOX consente di "trattare" sia sistemi continui che sistemi discreti, anche se i primi vengono sempre tradotti in un equivalenti discreti durante le simulazioni.
- I modelli che vengono utilizzati per la rappresentazione dei sistemi dinamici sono tre:
 - Modello di stato: definito attraverso le 4 matrici [A, B, C, D]

$$\text{del sistema} \quad \begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases}$$

- Funzioni di trasferimento

$$Y(s) = H(s)U(s) \quad \text{con} \quad H(s) = \frac{N(s)}{D(s)}$$

N(s) e D(s) sono rappresentati mediante vettori riga dei coefficienti dei polinomi, in ordine decrescente

- Poli-zeri:

$$H(s) = \frac{Z(s)}{P(s)} = \frac{(s + z_1)(s + z_2) \dots (s + z_{n-1})(s + z_n)}{(s + p_1)(s + p_2) \dots (s + p_{m-1})(s + p_m)}$$

Z e P sono rappresentati tramite vettori colonna delle radici

N.B. Le tre rappresentazioni valgono sia per sistemi continui che per sistemi discreti.

CONVERSIONI

- Esiste la possibilità di passare da una rappresentazione all'altra mediante funzioni di conversione:

$[n,d] = \text{ss2tf}(a,b,c,d,iu)$ converte da spazio-stato a funz. di trasf.
iu rappresenta l'ingresso considerato
(caso MIMO, altrimenti va posto a 1)

$[z,p,k] = \text{ss2zp}(a,b,c,d,iu)$ da spazio stato a poli-zeri e guadagno

$[a,b,c,d] = \text{tf2ss}(n,d)$ da funz. di trasf. a spazio stato

$[a,b,c,d] = \text{zp2ss}(z,p,k)$ da poli-zeri e guadagno a spazio-stato

$[n,d] = zp2tf(z,p,k)$ da poli-zero e guadagno a funz. di trasf.

- Si può passare da sistemi continui a sistemi a segnali campionati e viceversa:

$[ad,bd]=c2d(a,b,Ts)$ da matrici a e b del sistema continuo a corrispondenti discrete. Usa l'approx di Padè e può dare risultati imprecisi. Ts è il periodo di campionamento in secondi

$[ad,bd,cd,dd]=c2dm(a,b,c,d,Ts, 'metodo')$
analogo a c2d, permette di stabilire il metodo di discretizzazione (zoh, foh, tutsin, prewarp, matched - vedere help)

$[a,b] = d2c(ad,bd,Ts)$ da discreto a continuo

N.B. regola "mnemonica" per ricordarsi le funzioni: il 2 va letto come "to" ovvero d2c va letto come "discrete to continuos" e analogamente tutte le altre funzioni di conversione.

OPERAZIONI SUI BLOCCHI

- Consentono di calcolare la rappresentazione complessiva di sistemi connessi:
 - append
 - connect
 - parallel →parallelo di due sistemi dinamici
 - series →serie di due sistemi dinamici

Vedere l'help per l'uso

- Esiste anche una funzione che consente di creare un blocco del II ordine:

- `[n,d]=ord2(omega,csi)` crea una f.d.t. del II ordine con omega e csi scelti

ANALISI DI SISTEMI

- `obsv(a,c)`, `ctrb(a,b)` danno rispettivamente le matrici di osservabilità e controllabilità per il sistema dato.
- `obsvf(a,b,c)`, `ctrbf(a,b,c)` trasformano rispettivamente il sistema originario in uno equivalente in forma di osservazione e di controllo
- La risposta a segnali canonici e non e la risposta frequenziale possono essere facilmente ottenute con le seguenti funzioni (la sintassi cambia leggermente da versione 3.5 a 4x... meglio consultare l'help in linea):

`impulse`
`step`
`lsim`
`bode`
`nyquist`

- Esempio risposta al gradino (per un sistema rappresentato in spazio-stato):

`[y,x]=step(a,b,c,d,iu,t)` `t` è il vettore contenente il supporto della risposta. L'uscita è `y`, lo stato `x`

- Esempio di risposta frequenziale (per un sistema rappresentato con f.d.t.)

`[mag,phase]=bode(num,den,w)`
`w` è il vettore delle frequenze a cui calcolare la risposta, `mag` è il modulo, `phase` la fase.

N.B.1: Mentre per le risposte temporali è opportuno equispaziare linearmente gli istanti in cui si va a calcolare la risposta, per le risposte frequenziali conviene una equispaziatura logarimica delle frequenze, che corrisponde ad una lineare sui grafici. Per questo di usa la seguente istruzione per generare il vettore w:

`w= logspace[d1,d2,n]` Crea un vettore di n valori da 10^{d1} a 10^{d2} .
Se n è omesso, vale 50.

N.B.2. Tutte le funzioni viste valgono per i sistemi continui. Le analoghe discrete hanno lo stesso nome preceduto da "d".

`dimpulse, dstep, dlsim, dbode, dnyquist`

Va tenuto conto, comunque, che la frequenza va da 0 a π .

- `gram` e `dgram` consentono di calcolare il gramiano di raggiungibilità e di osservabilità per un sistema continuo e discreto, rispettivamente. Vedere `help` per la sintassi.
- `lqr`, `lqe`, `dlqr`, `dlqe`, `margin`, `place`, `acker` e altre funzioni consentono di calcolare i guadagni della matrice di reazione allo stato per posizionare gli autovalori del sistema a catena chiusa in posizioni ottime o desiderate...

ESEMPI

- Si trovano nella directory `/home/oboe/esempi_matlab`
- Ci sono anche due eseguibili per PC che possono essere utili a chi ha una vecchia versione di PC-Matlab: `egaepson.com` e `egalaser.com`. Questi due programmi consentono la copia di uno schermo VGA su stampante `epson-compatible` e `hp-laserjet-compatible`, rispettivamente.
- `contdig.m` fa vedere come si possa usare MATLAB per la simulazione di un sistema a dati campionati, con retroazione dall'uscita e regolatore proporzionale

```

% questo e` un m-file di prova.... %
echo on
% Vogliamo definire un sistema continuo, discretizzarlo %
% e poi provarne la risposta al gradino, sia a catena %
% aperta che a catena chiusa %

% Definiamo il sistema continuo come uno del secondo %
% ordine del tipo: %
%
%          %
%      W(s) =      1      %
%          -----      %
%      s^2 + 2*csi*wn s + wn^2      %
%
% con la funzione ord2(wn,csi) %

wn=2.4;
csi=0.4;

[num,den] = ord2(wn,csi)

pause

% vediamo il diagramma di bode di questa funzione... %

% anzitutto creiamo un insieme di frequenze alle quali %
% calcolare la f.d.t., che conviene siano spaziate %
% logarithmicamente (uso la funzione %
%     nome_var=logspace(d1,d2) o %
%     nome_var=logspace(d1,d2,n) %
%
%          %
%     per generare n elementi equispaziati log %
%     da 10^d1 a 10^d2 (n=50 di default) %

w=logspace(0,2,100);

% calcoliamo ora la risposta frequenziale %

[modulo,fase]=bode(num,den,w);

pause
% visualizziamo il risultato.... %

subplot(211),loglog(w,modulo),title('modulo')
xlabel('freq [rad]'), ylabel('quello che mi pare')

subplot(212),semilogx(w,fase),title('fase')

% potevamo ottenere il risultato anche dalla %
% rappresentazione di stato %

% si noti che e` possibile trasformare da funz. di trasf%
% a spazio stato con la funzione tf2ss %

```

```

[A,B,C,D]=tf2ss(num,den);

% N.B. e` [possibile anche il viceversa...           %

[modulo1,fase1]=bode(A,B,C,D,1,w);

% N.B. si deve specificare anche quale ingresso si usa. %

pause
% visualizziamo il risultato....                       %

subplot(211),loglog(w,modulo1),title('modulo_A,B,C,D')
xlabel('freq [rad]'), ylabel('quello che mi pare')

subplot(212),semilogx(w,fase1),title('fase')

pause
%
% Vediamo ora la risposta al gradino del sistema %

y=step(A,B,C,D,1);

subplot(111) %questo riporta la visualizzazione a posto..%

plot(y)

pause

% proviamo ora a discretizzare il sistema.           %

% fissiamo il tempo di campionamento T               %

T=0.1;

% Discretizziamo ricorrendo all funzione c2d          %

[Ad,Bd]=c2d(A,B,T);

% questa discretizzazione trasforma A in expm(A*T)... %
% che equivale a supporre uno zer-holder in ingresso %

% N.B. Si possono usare altri tipi di discretizzazione%
% con la funzione c2dm (es con prewarping...)        %

% le matrici Cd e Dd sono uguali a C e D (provare..) %

Cd=C;
Dd=D;

pause

% proviamo ora la risposta al gradino del sistema discreto%

yd = dstep(Ad,Bd,Cd,Dd,1);

```

```

plot(yd);

pause

% Come si puo` retroazionare il sistema???      %

% anzitutto, bisogna tenere conto che c'e` uno zero-holder %
% quindi c'e` un ritardo di un campione tra ingresso e uscita %
% reazionata....                                %

% si deve predisporre un vettore per l'ingresso, uno per
% l'uscita e un vettore temporale...

time=0:0.1:10;

out=zeros(size(time));

in=out;

in(10:101)=ones(1,92); % questo e` un gradino che parte da 1 s

% uso un controllo proporzionale

kp = 1;

ref(1)=kp*(in(1)-out(1));

x(:,1)=zeros(2,1);

% si organizza la simulazione come un ciclo      %

for j=1:101

    x(:,j+1) = Ad*x(:,j) + Bd*ref(j);
    out(j) = Cd*x(:,j);
    ref(j+1) = kp*(in(j)-out(j));

end %for

plot(time,out),title('risposta al gradino a catena chiusa')

end;

```