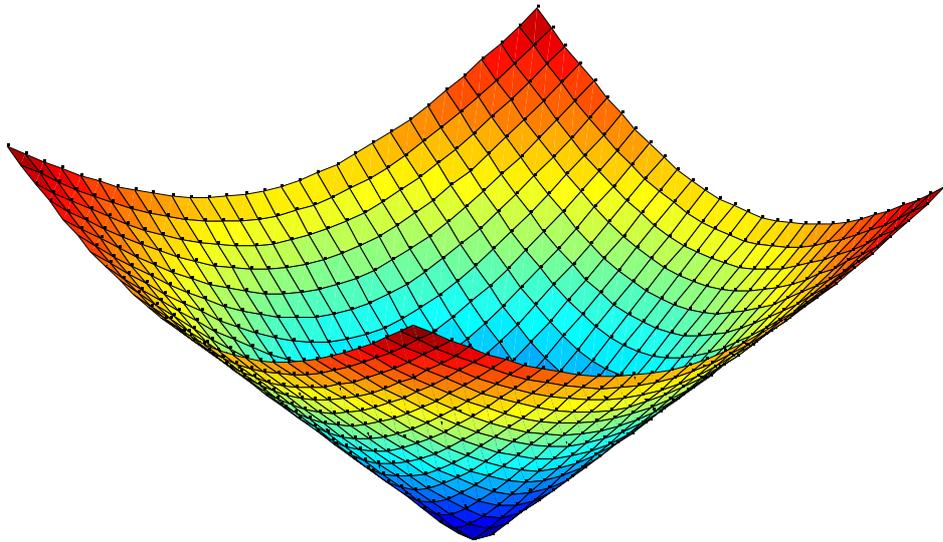

Mini Manuale Matlab



Antonio Salvato

Questo documento è stato redatto interamente utilizzando il software OpenOffice.org, disponibile gratuitamente in rete all'indirizzo www.openoffice.org.

Le immagini sono state generate attraverso il software Matlab, www.mathworks.com.

Matlab® è un marchio registrato dalla MathWorks Inc.

Rev 1.0 - Ottobre 2003

In copertina: rappresentazione grafica della funzione $\sqrt{x^2 + y^2}$

Introduzione

Questa piccola guida è un breve resoconto di quanto visto nel corso dei primi due anni di Ingegneria Elettronica in relazione a Matlab, il potente e diffusissimo software di computo tecnico sviluppato dalla americana MathWorks.

Il presente documento NON vuole essere assolutamente sostitutivo di alcuna guida ufficiale o altra opera edita da esperti disponibile in commercio, dato che il sottoscritto NON è un esperto ma un semplice utente di Matlab alle prime armi.

Questo documento vuole invece essere una breve lettura utile per compiere i primi passi nell'ambiente di lavoro di Matlab ed apprendere la sintassi di base dei comandi di maggior utilizzo nell'ambito dei corsi del primo e secondo anno di Ingegneria Elettronica e Informatica.

A.S.

Indice Generale

Parte 1: Introduzione a Matlab.....	3
Che cos'è Matlab?.....	3
Launch Pad/Workspace.....	4
Command History/Current Directory.....	4
Command Window.....	4
Gli M-File.....	4
Parte 2: Vettori e Matrici.....	6
Introduzione.....	6
Tre comandi basilari.....	6
CD.....	6
DIR.....	6
HELP.....	6
Scalari, Vettori e Matrici.....	7
Operazioni di base.....	9
Somma e Differenza.....	10
Moltiplicazione e Divisione.....	10
Trasposizione.....	11
Elevamento a potenza.....	12
Operazioni avanzate.....	12
INV.....	13
EIG.....	13
SIZE.....	13
EYE.....	13
Numeri Complessi.....	13
Parte 3: Polinomi.....	15
Operazioni sui Polinomi.....	15
ROOTS.....	15
POLY.....	15
CONV.....	16
DECONV.....	16
Scomposizione in fratti semplici.....	16
RESIDUE.....	17
Parte 4: Control System Toolbox.....	18
Le variabili Sistema: ss,tf,zpk.....	18
Operazioni sui sistemi dinamici.....	21
SSDATA.....	21
TFDATA.....	22
ZPKDATA.....	22
MINREAL.....	22
FEEDBACK.....	22
DCGAIN.....	22
Diagrammi di Bode, Nyquist, Nichols.....	23

PZMAP.....	23
BODE.....	23
BODEMAG.....	23
NYQUIST.....	23
NICHOLS.....	24
RLOCUS.....	24
Risposta nel dominio del tempo.....	24
IMPULSE.....	24
STEP.....	24
LSIM.....	25
LTI Viewer.....	25
SISO Design Tool.....	26
Parte 5: Diagrammi 2D.....	30
Definizione di intervalli e funzioni.....	30
Linspace.....	31
PLOT.....	31
GRID.....	33
BOX.....	33
FIGURE.....	33
Parte 6: Diagrammi 3D.....	34
Definizione di domini a 2 e 3 dimensioni.....	34
MESHGRID.....	34
Funzioni reali di due variabili reali.....	35
Funzioni vettoriali di variabili reali.....	36
Rappresentazione grafica di funzioni.....	37
MESH.....	37
SURF.....	38
SHADING.....	38
CONEPLOT.....	39
STREAMLINE.....	40

Parte 1: Introduzione a Matlab

Che cos'è Matlab?

Matlab è un linguaggio software ad alte prestazioni che integra funzioni di calcolo matematico, visualizzazione grafica e programmazione. Esso è implementato in un potente ambiente di lavoro ad interfaccia grafica, avente una struttura modulare che ne rende notevolmente espandibili le funzionalità.

È possibile infatti installare *moduli* aggiuntivi, detti **Toolbox**, che forniscano caratteristiche specifiche, non presenti nel set di funzioni originali di Matlab.

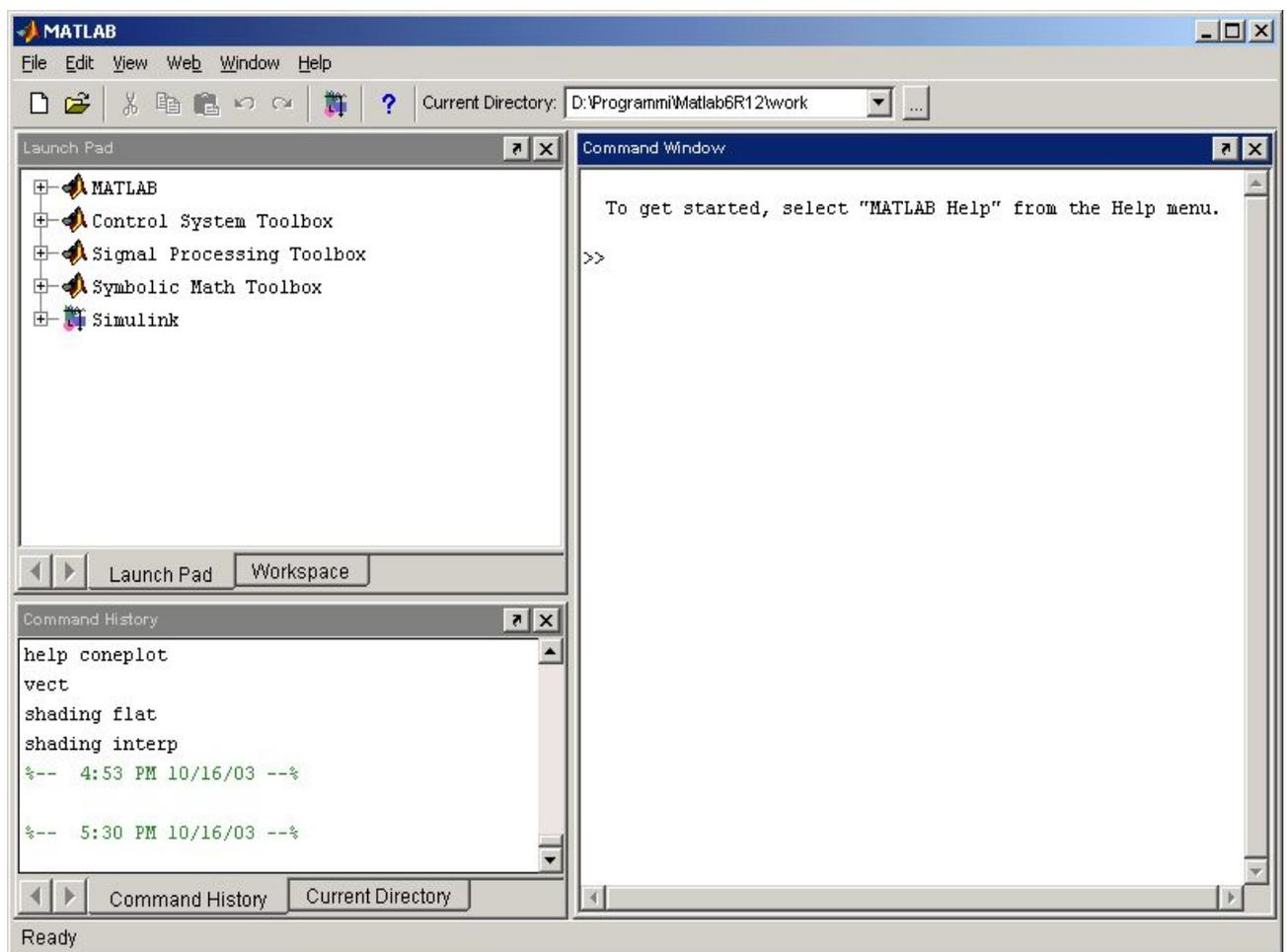


Fig. 1 - L'ambiente di lavoro Matlab

Possiamo indifferentemente parlare di “linguaggio Matlab” o “ambiente di lavoro Matlab”, dato che il linguaggio e l'ambiente di lavoro godono di una notevole integrazione.

L'ambiente di lavoro è suddiviso (in modalità di visualizzazione di Default, attivabile

mediante il menu View) in tre aree principali: Launch Pad/Workspace, Command History/Current Directory e Command Window.

Launch Pad/Workspace

Mediante il Launch Pad è possibile accedere in modo rapido alle funzioni offerte da uno specifico Toolbox installato. Ad esempio dalla struttura ad albero è possibile lanciare direttamente gli strumenti Lti Viewer e Siso Design Tool presenti nel Control System Toolbox.

Il Workspace è invece un potente strumento di monitoraggio delle variabili utilizzate nel corso delle operazioni effettuate dall'utente. Possiamo ottenere informazioni sul tipo e sull'occupazione di memoria di una certa variabile e addirittura accedere per via grafica direttamente al contenuto della variabile stessa.

Command History/Current Directory

La finestra Command History è in pratica una lista di tutti i comandi digitati, la “storia” dei comandi per l'appunto, con utilissime funzioni di copia e incolla.

Tramite la finestra Current Directory possiamo spostarci invece tra le cartelle del nostro Hard Disk come faremmo con un qualsiasi File Manager. Di Default essa visualizza il contenuto della cartella */work* contenuta nella directory principale di Matlab. Nella cartella */work* possiamo conservare tutti gli M-File (vedi più avanti) da noi creati ed accedervi in modo rapido.

Command Window

Questa finestra è, intuitivamente, il posto dove andremo a digitare i nostri comandi. In sostanza è la *shell* testuale dell'ambiente di lavoro ed è senz'altro lo strumento più potente che abbiamo a disposizione per usufruire in modo completo della enorme mole di programmi presenti in Matlab.

Gli M-File

Un M-File è un file, appunto, in cui è possibile salvare intere sequenze di comandi, anche molto lunghe. Sul nostro hard disk un M-File presenterà un'estensione “.m”.

È possibile dunque eseguire in modo rapido sequenze di azioni semplicemente lanciando l'M-File opportunamente creato. Per farlo basta spostarsi nella cartella che lo contiene mediante la shell testuale, attraverso il comando “cd”, e digitare il nome completo dell'M-File, senza l'estensione “.m”.

Poiché Matlab offre funzionalità di programmazione un M-File può contenere anche dei veri e propri programmi e non solo semplici “liste” di comandi indipendenti. Possiamo in questo modo aggiungere altre funzioni, anche complesse, a quelle che già ci vengono

offerte in partenza, allo scopo di risolvere problemi specifici.

Nella **Parte 2** di questa guida verranno elencati una serie di comandi di base che servono per compiere azioni elementari all'interno dell'ambiente di lavoro e per manipolare semplici entità matematiche quali scalari, vettori e matrici.

Nella **Parte 3** verrà introdotta la notazione per definire polinomi ed effettuare operazioni tra essi.

Nella **Parte 4** verranno esposte alcune caratteristiche del Control System Toolbox, un Toolbox che implementa potenti strumenti per l'analisi e la sintesi di sistemi dinamici lineari e non lineari.

Nella **Parte 5** saranno analizzati gli strumenti di base per la creazione di diagrammi bidimensionali, utili per rappresentare ad esempio funzioni reali di una variabile reale.

Nella **Parte 6** infine verranno presentati alcuni comandi relativi alla creazione di diagrammi in tre dimensioni, utili per rappresentare ad esempio funzioni reali di due variabili o campi vettoriali.

Parte 2: Vettori e Matrici

Introduzione

In questa sezione verranno presentati dapprima comandi che consentono di effettuare alcune semplici ma utili operazioni come spostarsi tra le directory del nostro hard disk. Verrà subito dopo illustrata la sintassi per dichiarare variabili ed effettuare semplici operazioni tra varie entità matematiche.

Risulta curioso sapere che la struttura matematica su cui si fonda Matlab è la matrice: in Matlab anche uno scalare è una matrice ...di dimensioni 1x1!

La *struttura dati astratta matrice* è in pratica in Matlab il tipo fondamentale. Non a caso Matlab è l'acronimo di **MAT**rix **LAB**oratory.

Tre comandi basilari

CD

uso:

```
cd nome_directory
cd ..
cd
```

Consente di spostarsi tra le cartelle sull'hard disk. L'attributo *nome_directory* è la cartella in cui ci si vuole spostare. È necessario specificare il percorso completo solo se la cartella desiderata non si trova all'interno di quella in cui si è attualmente.

L'attributo “..” consente di spostarsi nella cartella posta al livello immediatamente superiore a quello attuale.

Se non vengono specificati attributi il comando restituisce semplicemente il percorso della cartella attuale.

DIR

uso:

```
dir
```

Elenca i files contenuti nella cartella in cui ci si trova.

HELP

uso:

```
help
help nome_directory
help nomecomando
```

Se non viene specificato alcun attributo visualizza la lista di tutti i gruppi di

programmi presenti in matlab e nei toolbox installati. Ogni gruppo di comandi è situato in una directory. Per visualizzare la lista dei comandi presenti in un gruppo occorre specificare l'attributo *nomedirectory* che rappresenta il nome della directory relativa al gruppo desiderato.

Se si specifica infine l'attributo *nomecomando*, che rappresenta il nome di un generico comando in un generico gruppo, ne vengono visualizzati tutti i dettagli (significato, sintassi, parametri, ecc.).

Ad esempio digitando:

```
help matlab\graph2d
```

si visualizza la lista dei comandi presenti nel gruppo *graph2d*.
Invece digitando:

```
help plot
```

si visualizzano informazioni dettagliate sul comando PLOT.

Scalari, Vettori e Matrici

Per dichiarare una variabile reale ed assegnarle contemporaneamente un valore basta scrivere, per esempio:

```
x = 2.45;
```

Abbiamo dichiarato la variabile *x* e le abbiamo assegnato il valore 2.45.

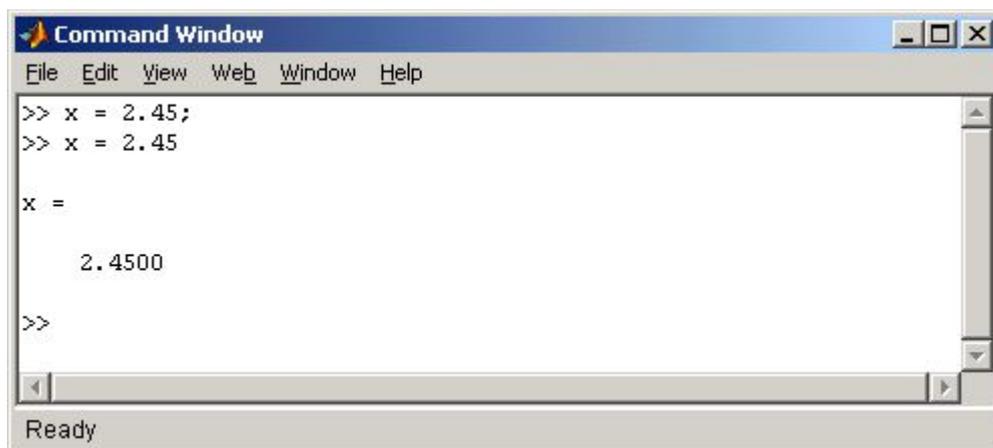


Fig. 2 - Dichiarazione di una variabile reale...

L'omissione del punto e virgola fa sì che a schermo venga stampato esplicitamente il contenuto della variabile *x*.

Come abbiamo detto Matlab manipola solo matrici, ed infatti un'ispezione al Workspace ci dice che x è una matrice di dimensioni 1×1 .

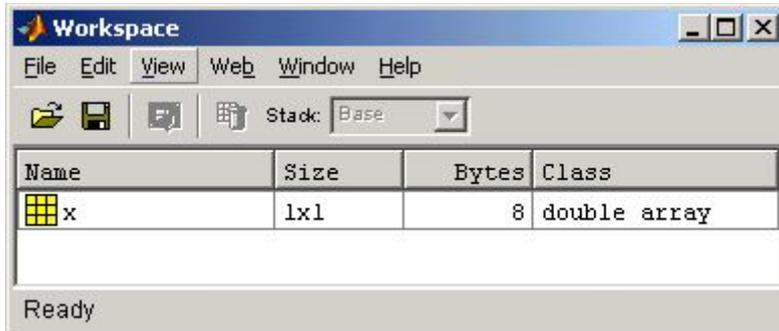


Fig. 3 - ...e i suoi dettagli: nome, dimensione, tipo.

Da notare l'icona posta a sinistra di x ...è una matrice! :-)

Possiamo di dichiarare un vettore:

```
a = [1 2 3];
```

o una matrice:

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

utilizzando la notazione riportata negli esempi.

Nella dichiarazione di una matrice il carattere “;” serve a separare le righe, mentre lo spazio “ ” serve a separare le colonne. Il tutto è racchiuso tra parentesi quadre “[]”.

Ovviamente un vettore è una particolare matrice avente una delle due dimensioni unitaria, così come lo scalare, che ha tutte e due le dimensioni unitarie. Quindi la notazione usata per dichiarare una matrice è quella più generale.

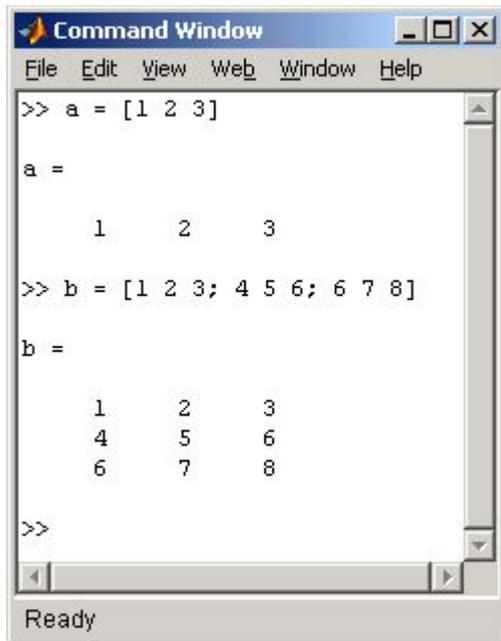


Fig. 4 - Dichiarazione di un vettore riga e di una matrice...

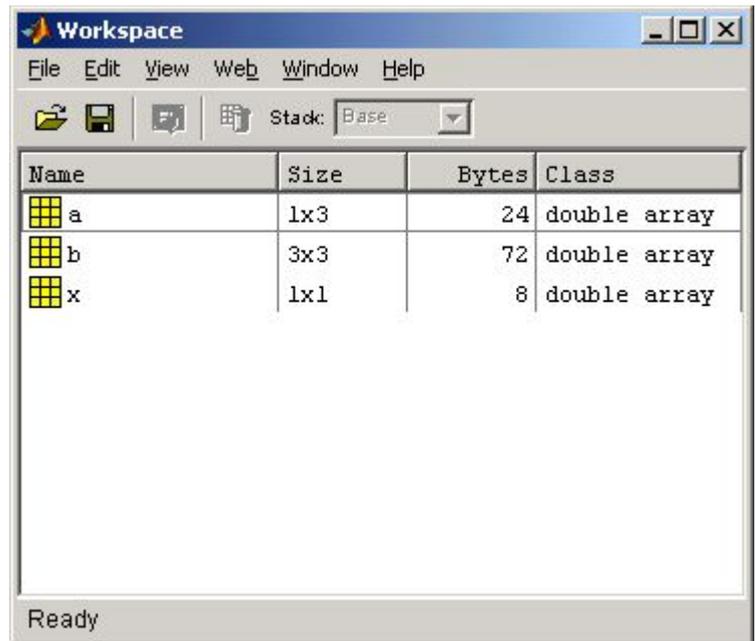


Fig. 5 - ...e i relativi dettagli nel Workspace.

Si osservi nella figura a sinistra l'effetto dell'omissione del punto e virgola, ossia la visualizzazione esplicita del contenuto di a e b.

Si guardi poi cosa è accaduto nel Workspace, a destra.

Operazioni di base

In Matlab gli operatori fondamentali sono i seguenti:

+	addizione
-	sottrazione
*	moltiplicazione
/	divisione
^	Elevamento a potenza
'	trasposizione

In operazioni tra scalari(matrici 1x1) il significato degli operatori riportati è quello che ben conosciamo dalla matematica, ad eccezione di quello di trasposizione “'” che non ha alcun effetto(o meglio, da come risultato lo scalare stesso a cui viene applicato).

In operazioni tra matrici e vettori bisogna stare più attenti, poiché devono essere

ovviamente rispettate le regole relative all'algebra delle matrici.

Chiariremo quanto detto mediante esempi.

Somma e Differenza

Per sommare due matrici di uguali dimensioni A e B, e mettere il risultato in una terza matrice C basta scrivere:

$$\begin{aligned} A &= [1 \ 2; 3 \ 4]; \\ B &= [5 \ 6; 7 \ 8]; \\ C &= A + B; \end{aligned}$$

La matrice C sarà dunque la seguente:

$$C = \begin{bmatrix} 6 & 8 \\ 10 & 12 \end{bmatrix}$$

Non è possibile invece ottenere un risultato se si tenta la somma tra matrici di dimensioni diverse.

$$\begin{aligned} A &= [1 \ 2; 3 \ 4]; \\ B &= [5 \ 6; 7 \ 8; 9 \ 10]; \\ C &= A+B; \end{aligned}$$

errore!

Quanto detto sopra vale pari pari per l'operazione di differenza.

Moltiplicazione e Divisione

La moltiplicazione tra due matrici è intesa invece *righe per colonne*. Possiamo effettuare il prodotto tra due opportune matrici A e B scrivendo:

$$\begin{aligned} A &= [1 \ 2; 3 \ 4]; \\ B &= [5 \ 6 \ 7; 8 \ 9 \ 10]; \\ C &= A*B; \end{aligned}$$

Il risultato è pertanto:

$$C = \begin{bmatrix} 21 & 24 & 27 \\ 47 & 54 & 61 \end{bmatrix}$$

Non è possibile invece valutare il prodotto $B \cdot A$, a meno che, come è noto, A e B non siano matrici quadrate di uguali dimensioni.

La moltiplicazione di una matrice per uno scalare è sempre possibile ed è commutativa, ovviamente.

L'operatore “/” consente di moltiplicare una matrice per l'inverso di uno scalare, oppure per la matrice inversa di un'altra matrice (*prodotto a destra*, righe per colonne).

Ad esempio scrivere:

$$A/B;$$

è analogo a scrivere:

$$A \cdot \text{inv}(B)$$

Diremo anche che “/” tra due matrici opportune rappresenta l'operatore di *divisione a destra*.

Possiamo utilizzare l'operatore “\” tra due matrici per effettuare la *divisione a sinistra*:
Quindi:

$$A \backslash B;$$

è analogo a:

$$\text{inv}(B) \cdot A;$$

Trasposizione

Per trasporre una matrice basta scrivere:

$$\begin{aligned} A &= [1 \ 2; 3 \ 4]; \\ B &= A'; \end{aligned}$$

La matrice B sarà:

$$B = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

La trasposizione è sempre possibile.

Elevamento a potenza

L'operatore di elevamento a potenza “^” può essere utilizzato in due modi.

1) Se A è una matrice quadrata possiamo scrivere:

$$B = A^n;$$

ed ottenere la moltiplicazione di A per se stessa n volte:

$$B = \underbrace{A \cdot A \cdot \dots \cdot A}_{n \text{ volte}}$$

2) Se A è una matrice qualsiasi, di dimensioni l x m possiamo scrivere:

$$B = A.^n;$$

Facendo precedere l'operatore “^” da un punto in pratica eleviamo a potenza n-esima i singoli elementi a_{ij} della matrice A. La matrice risultato è posta in B:

$$b_{ij} = (a_{ij})^n \quad i = 1, \dots, l \quad j = 1, \dots, m$$

Operazioni avanzate

Se x è una variabile scalare possiamo effettuare su di essa operazioni più complesse utilizzando le funzioni matematiche disponibili in Matlab.

Sarà possibile ad esempio estrarre la radice quadrata di x, oppure valutarne il logaritmo naturale:

$$y = \text{sqrt}(x);$$

$$g = \text{log}(x);$$

Abbiamo posto la radice quadrata di x in y e il logaritmo naturale di x in g.
Per una lista completa di funzioni matematiche digitare:

```
help elfun
```

al prompt dei comandi.

Se invece abbiamo a che fare con matrici, risultano utili le funzioni descritte di seguito.

INV

uso:

```
B = inv(A);
```

Valuta l'inversa di una matrice. A è la matrice da invertire. Il risultato è messo in B. Si possono invertire solo matrici quadrate, ovviamente.

EIG

uso:

```
[V,D] = eig(A);
```

Calcola gli *autovalori* e gli *autovettori* di una matrice quadrata.

D è una matrice diagonale contenente (nella diagonale e in ordine, da sinistra a destra) gli autovalori di A.

V è una matrice avente per colonne (in ordine di associazione ai corrispondenti autovalori, da sinistra verso destra) gli autovettori di A.

SIZE

uso:

```
size(A)
```

Stampa a schermo le dimensioni (numero righe - numero colonne) della matrice A.

EYE

uso:

```
A = eye(N);
```

Crea una matrice identità (matrice quadrata con tutti 1 sulla diagonale principale) e la pone in A.

Numeri Complessi

Matlab opera senza problemi anche con variabili complesse. Possiamo dichiararne una scrivendo, ad esempio:

```
p = 5 + 7i;
```

$$z = 5 + 7j;$$

Si ha $p = z$: l'operatore immaginario è indifferentemente i o j .

Su un numero complesso è possibile compiere tutte le operazioni di base, nonché quelle avanzate¹, valide per un numero reale. La sintassi è esattamente la stessa.

¹ Nell'interpretazione corretta dei risultati derivanti dalla applicazione di funzioni matematiche elementari (o composte) a numeri complessi sarebbe opportuno far riferimento alla teoria delle funzioni complesse.

Parte 3: Polinomi

Come sappiamo un polinomio è univocamente definito se si conoscono i suoi coefficienti: per definire un polinomio in Matlab basta definire un vettore che li contenga. Il grado del polinomio è automaticamente fissato dalla dimensione del vettore.

Ad esempio, digitando:

```
a = [1 2 5 3];
```

avremo definito il polinomio:

$$a(s) = s^3 + 2s^2 + 5s + 3$$

Il vettore contiene 4 elementi. Il grado del polinomio è $3 = 4 - 1$.

In generale se il vettore contiene n elementi il polinomio è di grado $n - 1$. Ciò è sempre vero perchè qualora nel polinomio qualche coefficiente sia nullo, bisognerà scriverlo esplicitamente nel vettore e quindi, seppur nullo, esso contribuirà alla dimensione del vettore.

Da questo punto in poi (fino alla fine della Parte 3) useremo semplicemente il termine "polinomio" per indicare un vettore che ne contenga i coefficienti.

Operazioni sui Polinomi

ROOTS

uso:

```
roots(a);
```

Calcola le radici del polinomio a, supposto di grado n, e le visualizza a schermo. Le radici sono in numero pari ad n e possono essere, come sappiamo dalla matematica reali o complesse, a coppie coniugate.

POLY

uso:

```
a = poly(b);
```

Trova i coefficienti di un polinomio, a partire dalla conoscenza delle sue radici. b è un vettore contenente n radici note, reali o complesse, a coppie coniugate. a è il vettore risultato che contiene i coefficienti del polinomio calcolato. In realtà ad n radici corrispondono infiniti polinomi di grado n, tutti differenti per un semplice fattore moltiplicativo: il comando POLY fornisce pertanto il

polinomio monico (cioè quello avente coefficiente della massima potenza di s pari a 1).

CONV

USO:

$$c = \text{conv}(a,b);$$

Effettua la moltiplicazione tra i due polinomi a e b .
 c è il polinomio prodotto. Se a è di grado m e b è di grado n allora c sarà di grado $m + n$.

DECONV

USO:

$$[q,r] = \text{deconv}(a,b)$$

Effettua la divisione tra i due polinomi a e b .
 q è il polinomio quoziente. r è il polinomio resto.

Scomposizione in fratti semplici

Supponiamo di avere una funzione razionale fratta, costituita cioè dal rapporto tra due polinomi:

$$\frac{n(s)}{d(s)} = \frac{s^m + b_1 s^{(m-1)} + \dots + b_{m-1} s + b_m}{s^n + a_1 s^{(n-1)} + \dots + a_{n-1} s + a_n}$$

Supponiamo di scomporre il polinomio a denominatore nel prodotto:

$$d(s) = (s - p_1) \dots (s - p_n)$$

dove p_1, \dots, p_n sono le n radici del polinomio a denominatore, dette anche *poli* della funzione razionale fratta.

Possiamo scrivere:

$$\frac{n(s)}{d(s)} = \frac{r_1}{(s - p_1)} + \dots + \frac{r_n}{(s - p_n)} + K(s)$$

ossia *scomporre in fratti semplici* la funzione $n(s)/d(s)$.

r_1, \dots, r_n sono gli n *residui* associati agli n poli della funzione r . fratta e $K(s)$ è la *parte intera*. Quest'ultima è un polinomio diverso da zero solo se m è maggiore o uguale a n .

In Matlab è possibile usare il comando RESIDUE per effettuare la scomposizione in fratti

semplici.

RESIDUE

uso:

```
[r,p,k] = residue(n,d);
```

Calcola i residui della funzione razionale fratta avente per numeratore il polinomio n e per denominatore il polinomio d .

r è il vettore dei residui.

p è il vettore dei poli.

k è il vettore dei coefficienti della eventuale parte intera.

Parte 4: Control System Toolbox

Le variabili Sistema: ss,tf,zpk

Tra gli innumerevoli Toolbox disponibili in Matlab, il Control System Toolbox è senz'altro di notevole utilità per chi si trova ad affrontare problemi di analisi dei sistemi dinamici LTI (lineari tempo-invarianti) o di sintesi di sistemi di controllo.

Questo Toolbox introduce in Matlab due importanti tipi di variabile:

- `ss` = variabile sistema dinamico in *spazio di stato*
- `tf` = variabile *funzione di trasferimento* nel *dominio di Laplace*

Ricordiamo che un sistema dinamico LTI presenta nello spazio di stato una rappresentazione (rappr. l-s-u) del tipo:

$$\begin{aligned}\dot{x} &= Ax + Bu \\ y &= Cx + Du\end{aligned}$$

dove A è la *matrice dinamica*, B è la *matrice degli ingressi* e C è la *matrice delle uscite*.

Nel dominio di Laplace invece la f.d.t. è del tipo²:

$$W(s) = \frac{s^m + b_1 s^{(m-1)} + \dots + b_{m-1} s + b_m}{s^n + a_1 s^{(n-1)} + \dots + a_{n-1} s + a_n} = K \frac{(s - z_1) \dots (s - z_m)}{(s - p_1) \dots (s - p_n)}$$

ossia una funzione razionale fratta con m zeri ed n poli. K è il guadagno della funzione di trasferimento (non è il guadagno statico!).

Per creare in Matlab un oggetto sistema in spazio di stato, basta definire le matrici A, B, C, D (cosa che sappiamo già fare) e scrivere:

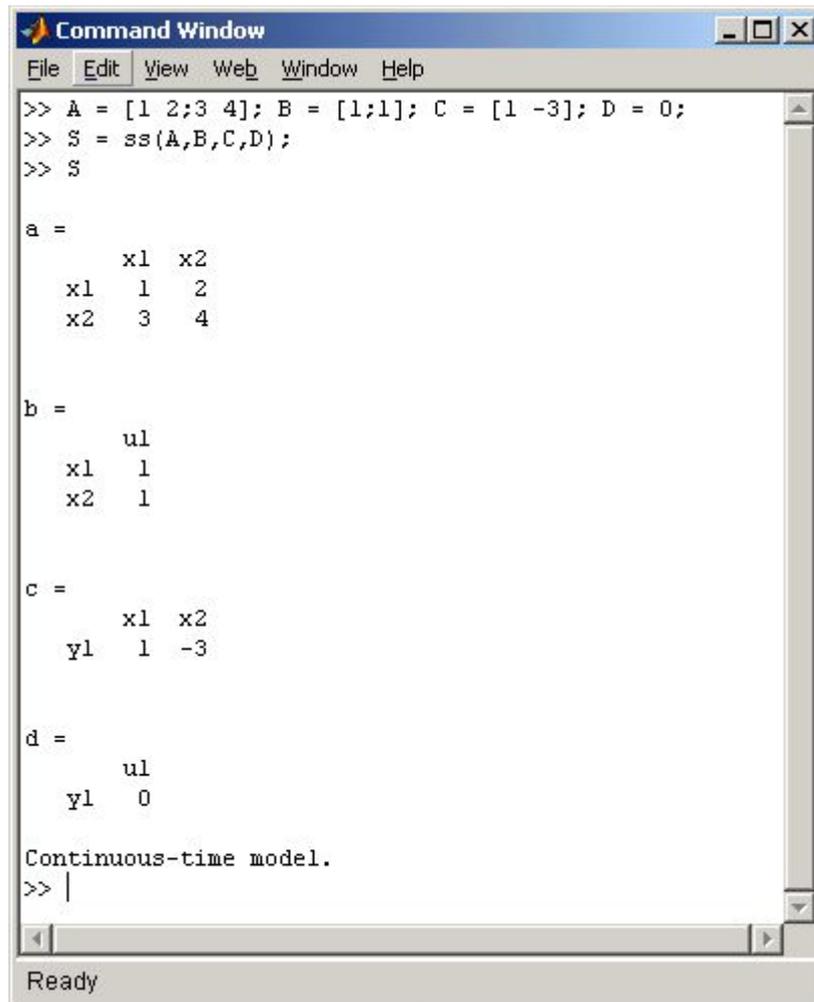
```
S = ss(A,B,C,D);
```

Per specificare direttamente la funzione di trasferimento scriviamo invece:

```
W = tf(n,d);
```

dove n e d sono i vettori che contengono i coefficienti del polinomio a numeratore e a denominatore della f.d.t..

² La f.d.t. è in realtà una funzione razionale fratta solo SE il sistema è SISO, altrimenti sarà, in caso di sistema MIMO, un vettore o una matrice di funzioni razionali fratte. Considereremo per semplicità solo sistemi SISO.



```

Command Window
File Edit View Web Window Help
>> A = [1 2;3 4]; B = [1;1]; C = [1 -3]; D = 0;
>> S = ss(A,B,C,D);
>> S

a =
      x1  x2
x1      1   2
x2      3   4

b =
      u1
x1      1
x2      1

c =
      x1  x2
y1      1  -3

d =
      u1
y1      0

Continuous-time model.
>> |
Ready

```

Fig. 6 - Definiamo le matrici A,B,C,D e dichiariamo la variabile sistema S.

Possiamo ricavare la f.d.t a partire dalla i-s-u [e viceversa] scrivendo:

$$W = \text{tf}(S); \quad [S = \text{ss}(W);]$$

È possibile definire una f.d.t. anche come somma, differenza, prodotto o divisione di altre f.d.t.. Ad esempio, detti A, B e C tre oggetti di tipo tf possiamo definire una W come segue:

$$W = \text{tf}(((A+B)*(B+C))/(C*B));$$

È anche possibile l'elevamento a potenza mediante l'operatore “^”.
Ad esempio:

$$W = \text{tf}((A^2)/B);$$

La f.d.t. W sarà pari al quadrato di A diviso per B.

```

Command Window
File Edit View Web Window Help
>> A = tf([1 1],[1 5 2])

Transfer function:
      s + 1
-----
s^2 + 5 s + 2

>> B = tf(1,[1 2])

Transfer function:
      1
-----
s + 2

>> W = tf((A^2)/B)

Transfer function:
      s^3 + 4 s^2 + 5 s + 2
-----
s^4 + 10 s^3 + 29 s^2 + 20 s + 4

>> |
Ready
    
```

Fig. 7 - Definiamo una f.d.t. facendo uso degli operatori che conosciamo.

Un modo alternativo, e forse più naturale, per specificare una f.d.t è dato nel seguente esempio:

```
s = tf('s');
```

```
W = (s+1)/(s^2+3*s+1);
```

In pratica si dichiara la variabile funzione di trasferimento $H(s) = s$ così come mostrato nel primo comando e poi si dichiara la f.d.t. desiderata, scrivendola esplicitamente così come faremmo su un pezzo di carta.

Esiste infine un ulteriore tipo di variabile di sistema dinamico:

- zpk = variabile funzione di trasferimento in forma zeri-poli-guadagnoK

Mediante esso è possibile specificare una f.d.t. indicandone gli zeri, i poli e il guadagno K. Per esempio, mediante il comando:

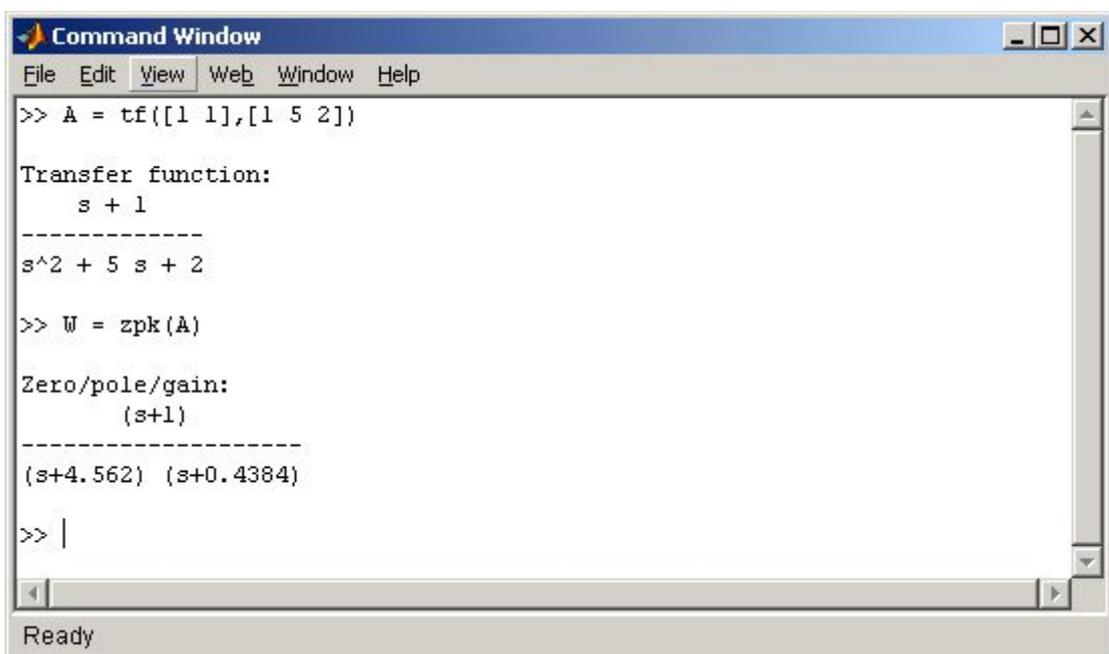
```
W = zpk([-1 -3],[0 -5 7],10);
```

definiamo una f.d.t. con due zeri (in -1 e in -3), tre poli (in 0, -5 e 7), e $K = 10$.

Possiamo altresì “convertire” un oggetto di tipo `tf` o `ss` in un altro di tipo `zpk`.
Es:

```
A = tf(a,b);
```

```
W = zpk(A);
```



```

Command Window
File Edit View Web Window Help
>> A = tf([1 1],[1 5 2])

Transfer function:
   s + 1
-----
 s^2 + 5 s + 2

>> W = zpk(A)

Zero/pole/gain:
   (s+1)
-----
(s+4.562) (s+0.4384)

>> |
Ready

```

Fig. 8 - Conversione di un oggetto `tf` in un oggetto `zpk`.

Operazioni sui sistemi dinamici

Vediamo ora quali manipolazioni possiamo effettuare su un sistema dinamico definito come su esposto. Indicheremo con `SYS` una generica variabile sistema indifferentemente di tipo `ss`, `tf`, `zpk`.

SSDATA

uso:

```
[A,B,C,D] = SSDATA(SYS);
```

Estrae le matrici `A`, `B`, `C`, `D` del sistema `SYS` provvedendo automaticamente alla conversione in oggetto `ss` qualora `SYS` non lo sia già.

TFDATA

uso:

```
[NUM,DEN] = tfdata(SYS,'v');
```

Estrae da una funzione di trasferimento il polinomio a numeratore, NUM, e quello a denominatore, DEN. Il tag 'v' serve a far sì che NUM e DEN siano dei vettori di coefficienti e non Array di celle generici. Se SYS non è di tipo tf il comando provvede automaticamente alla conversione.

ZPKDATA

uso:

```
[Z,P,K] = zpkdata(SYS,'v');
```

Estrae da una funzione di trasferimento gli zeri (nel vettore Z), i poli (nel vettore P) e il guadagno K. Il tag 'v' serve a far sì che Z,P e K siano dei vettori di coefficienti e non Array di celle generici. Se SYS non è di tipo zpk il comando provvede automaticamente alla conversione.

MINREAL

uso:

```
MSYS = minreal(SYS);
```

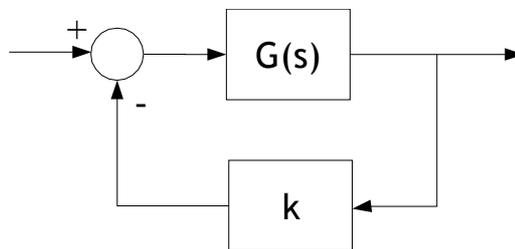
Effettua tutte le cancellazioni polo-zero nella f.d.t. SYS, fornendone dunque l'espressione semplificata.

FEEDBACK

uso:

```
W = feedback(G,k);
```

Crea il sistema a retroazione (negativa di default) avente G come *f.d.t. di catena diretta* e k come *costante di retroazione*.



Se $k < 0$ la retroazione è positiva.

DCGAIN

uso:

```
Kp = dcgain(SYS);
```

Estrae il guadagno statico³ del sistema SYS.

Diagrammi di Bode, Nyquist, Nichols

Non mancano nel C.S.Toolbox tutti gli strumenti grafici della teoria dei sistemi quali i *diagrammi di Bode* o i *diagrammi polari*. È possibile inoltre visualizzare direttamente sul *piano di Gauss* i poli e gli zeri di una f.d.t. nonché valutare il *luogo delle radici* di un *sistema a ciclo chiuso* a partire dalla conoscenza della f.d.t. di *guadagno di anello*. Di seguito sono riportati i comandi relativi.

PZMAP

uso:

```
pzmap(SYS);
```

Fornisce la rappresentazione sul piano complesso dei poli e degli zeri del sistema SYS.

BODE

uso:

```
bode(SYS);
bode(SYS,{wmin,wmax});
```

Traccia i diagrammi di Bode dei Moduli e delle Fasi del sistema SYS. Si tratta dei diagrammi di Bode *reali*. Non è disponibile invece una funzione che tracci quelli asintotici. Si può scegliere l'intervallo di frequenze desiderato specificando la pulsazione minima, wmin, e quella massima, wmax, in rad/sec.

BODEMAG

uso:

```
bodemag(SYS);
bodemag(SYS,{wmin,wmax});
```

Traccia il solo diagramma di Bode dei Moduli. L'uso è identico al comando BODE.

NYQUIST

uso:

```
nyquist(SYS);
nyquist(SYS,{wmin,wmax});
```

³ $K_p = \lim_{s \rightarrow 0} W(s)$

Traccia il diagramma polare (o *di Nyquist*) del sistema SYS. Si può scegliere l'intervallo di frequenze desiderato specificando la pulsazione minima, w_{min} , e quella massima, w_{max} , in rad/sec.

NICHOLS

uso:

```
nichols(SYS);  
nichols(SYS, {wmin, wmax});
```

Traccia il *diagramma di Nichols* del sistema SYS. w_{min} e w_{max} sono gli estremi dell'intervallo di frequenze desiderato, in rad/sec.

RLOCUS

uso:

```
rlocus(G);  
rlocus(G, k);
```

Traccia il luogo delle radici della funzione di trasferimento a ciclo chiuso del sistema a retroazione negativa (se $k > 0$) $W = G / (1 + kG)$.

G è la f.d.t. di catena diretta del sistema retroazionato.

k è la costante di retroazione. Nel caso in cui k non è specificato il comando assume $k = 1$.

Se $k < 0$ il sistema a ciclo chiuso è a retroazione positiva.

Risposta nel dominio del tempo

Di un sistema dinamico ci interesserà senz'altro valutare la risposta a determinati tipi di ingresso. Abbiamo a disposizione alcuni comandi che consentono di valutare immediatamente la *risposta impulsiva* e la *risposta indiciale*, nonché altre funzioni per "testare" un sistema con un qualsiasi tipo di ingresso da noi definito.

IMPULSE

uso:

```
impulse(SYS);  
[y, t, x] = impulse(SYS);
```

Diagramma la risposta impulsiva nell'uscita del sistema SYS.

Se si specificano i parametri di uscita (y, t, x) il comando non disegna alcun diagramma ma produce il vettore contenente i campioni della risposta nell'uscita, y, nello stato, x, e il vettore dei campioni della variabile indipendente t (tempo).

STEP

uso:

```
step(SYS);
[y,t,x] = step(SYS);
```

Diagramma la risposta indiciale nell'uscita del sistema SYS.

Se si specificano i parametri di uscita (y,t,x) il comando non disegna alcun diagramma ma produce il vettore contenente i campioni della risposta nell'uscita, y, nello stato, x, e il vettore dei campioni della variabile indipendente t(tempo).

LSIM

uso:

```
lsim(SYS,u,t);
lsim(SYS,u,t,x0);
```

Diagramma la risposta del sistema SYS al segnale di ingresso u(t) i cui campioni sono contenuti nel vettore u. Il vettore t contiene invece i campioni della variabile indipendente t del segnale di ingresso⁴. u e t hanno uguale dimensione. Il parametro x0 specifica le condizioni iniziali. Esso è un vettore di dimensione n, dove n è l'*ordine del sistema* SYS. In caso di omissione di x0 esso è posto automaticamente a 0 dal comando.

LTI Viewer

Tutti i comandi incontrati finora sono potentemente implementati nell'intuitiva interfaccia grafica fornita dal tool **LTI Viewer**.

Basta digitare:

```
ltiview(SYS);
```

per avere a portata di clic qualsiasi informazione sul sistema SYS.

⁴ Per definire intervalli e rappresentare segnali vedere la Parte 4.

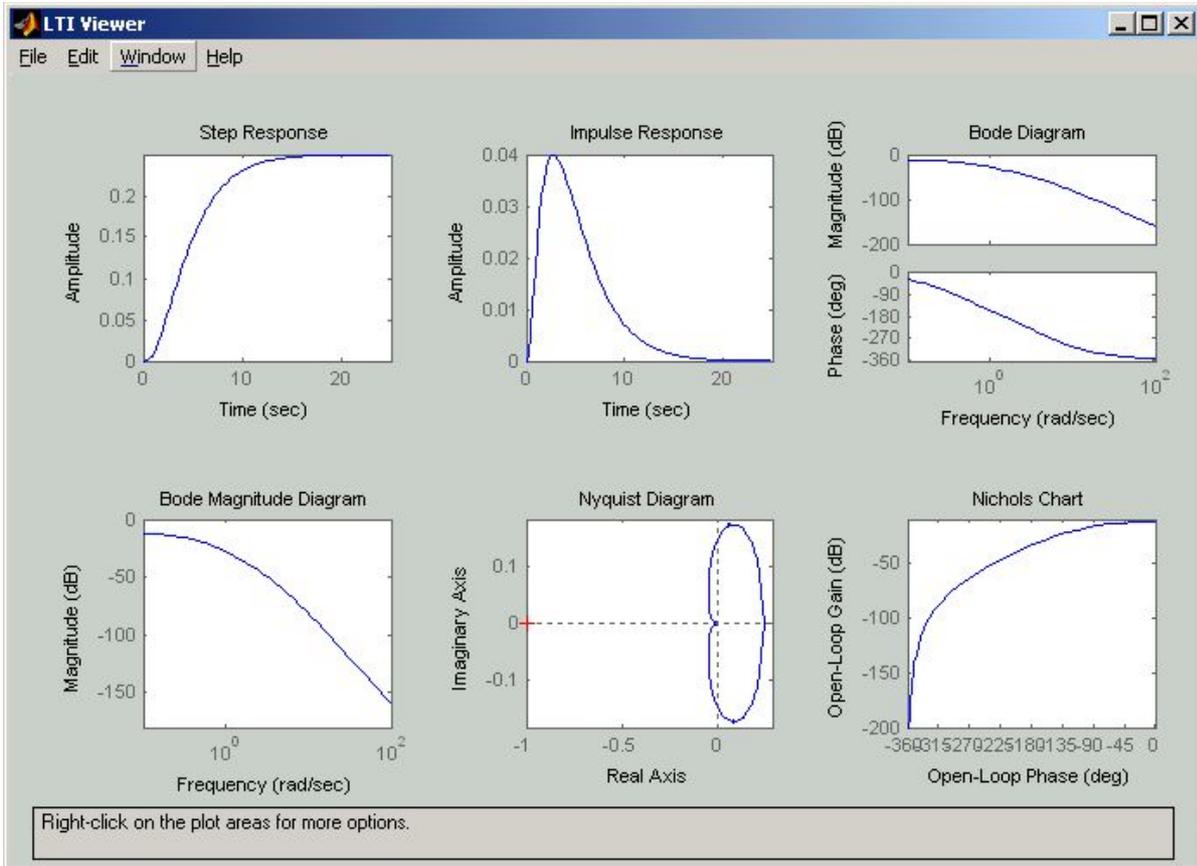


Fig. 9 - L'interfaccia grafica di LTI Viewer. Si possono visualizzare i parametri caratteristici dei singoli grafici (tempo di assestamento, margini di stabilità, ecc.) utilizzando il tasto destro su ognuno di essi.

Navigando i menù in alto e utilizzando il tasto destro è possibile accedere a tutte le opzioni relative ai singoli diagrammi, visualizzare contemporaneamente i dati di più sistemi dinamici, ingrandire i singoli grafici e visualizzarne i parametri caratteristici.

SISO Design Tool

Un altro strumento di estrema utilità, questa volta nell'ambito del progetto di sistemi di controllo, è il SISO Design Tool.

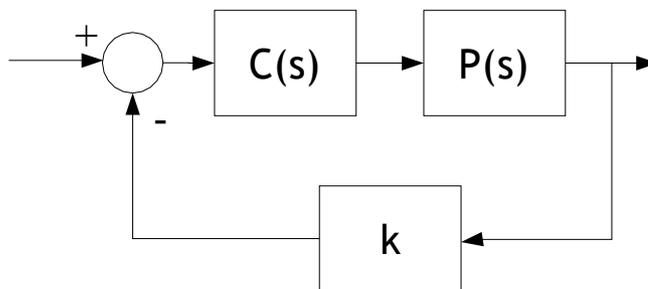


Fig. 10 - Un sistema di controllo con retroazione.

Obiettivo del progettista è sintetizzare la funzione di trasferimento $C(s)$ mediante opportune tecniche, nota la f.d.t. dell'impianto da controllare $P(s)$.

Definita in Matlab la f.d.t. $P(s)$ basta digitare al prompt:

```
sisotool(P);
```

Viene aperta una finestra divisa in diverse sezioni, caratterizzata dalla presenza di alcuni menù specifici e di una barra degli strumenti(toolbar) posta in alto.

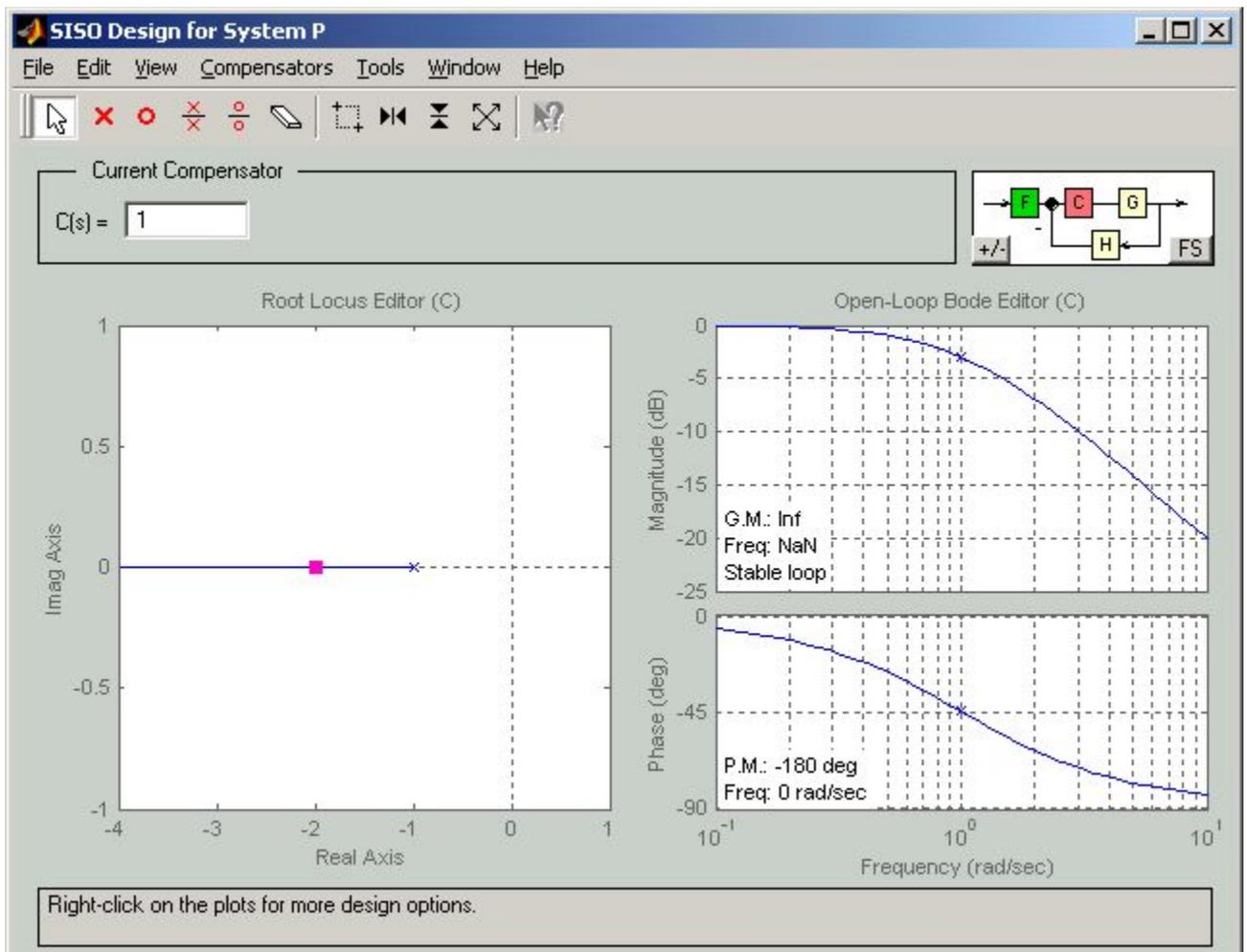


Fig. 11 - L'interfaccia grafica del SISO Design Tool. Possiamo manipolare molto facilmente il controllore per via grafica (sul piano di Gauss) o analitica e verificare istantaneamente il comportamento del sistema a ciclo chiuso.

Nel box in alto a destra la f.d.t. $P(s)$ è rappresentata dal blocco G . H è il blocco di retroazione che di default è pari alla costante 1. C è il controllore la cui f.d.t. è mostrata per esteso immediatamente a sinistra.

Nell'area al centro viene automaticamente mostrato il luogo delle radici del sistema a ciclo chiuso. È possibile aggiungere sul piano di Gauss direttamente poli e zeri al

controllore $C(s)$ per via grafica utilizzando i pulsanti nella toolbar in alto.

Tramite i menù è possibile visualizzare qualsiasi informazione o diagramma relativo ad ognuna delle f.d.t. $C(s)$, $P(s)$, e $H(s)$ e al sistema a ciclo chiuso $W(s)$.

Tra essi vi è anche la *Carta di Nichols*, fondamentale per soddisfare le specifiche in *banda passante* e *modulo di risonanza* nel sistema a ciclo chiuso.

Attraverso il comando Export dal menù File possiamo esportare direttamente nello stack di memoria di Matlab(Workspace) le variabili relative alle f.d.t. di *Sensitività diretta* e *Sensitività complementare*, risp. $S(s)$ e $T(s)$, e tante altre f.d.t. quali il guadagno di anello o la f.d.t. a ciclo chiuso, in modo da poter effettuare manipolazioni dirette su queste ultime attraverso la shell testuale.

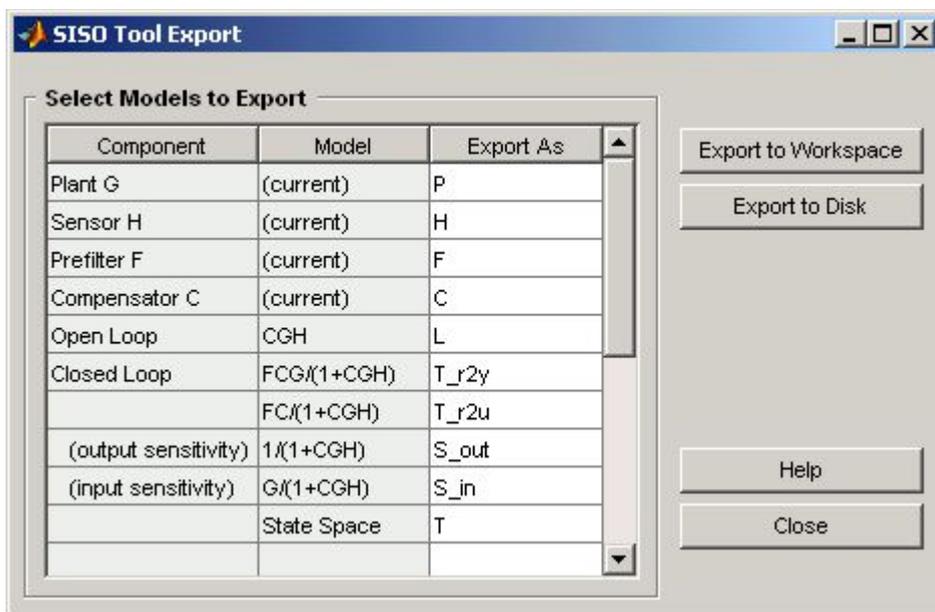


Fig. 12 - Possiamo esportare una qualsiasi f.d.t. nel Workspace e utilizzarla a nostro piacimento mediante il prompt dei comandi.

Ancora, possiamo valutare i diagrammi tipici(Bode, Nichols, risposta indiciale, ecc.) in relazione a funzioni di trasferimento che abbiano ingressi e uscite posti in punti specifici dello *schema a blocchi*. In questo modo possiamo ad esempio valutare la *risposta a disturbi* costanti in *catena diretta* o in *catena di retroazione*, il cui andamento è di notevole importanza per le prestazioni del sistema a ciclo chiuso.

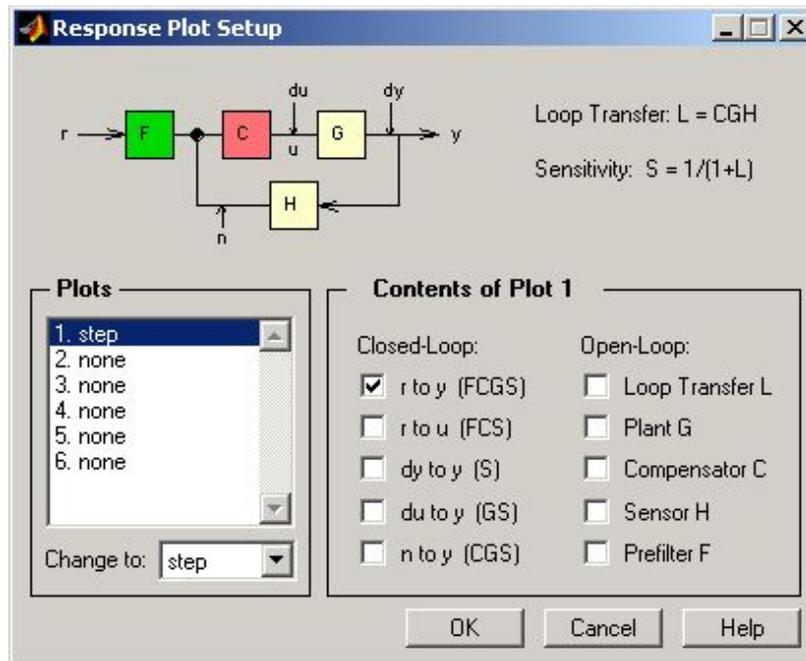


Fig. 13 - La finestra ci consente di specificare un punto di ingresso e uno di uscita nello schema a blocchi e di effettuare diverse operazioni sulla f.d.t. corrispondente.

Come sempre infine si possono settare indipendentemente tutte le opzioni relative ai singoli diagrammi e visualizzare su di essi i parametri caratteristici.

Parte 5: Diagrammi 2D

Definizione di intervalli e funzioni

Per diagrammare una generica *funzione di una variabile reale* $f(t)$, rappresentabile come sappiamo mediante una coppia di assi cartesiani, occorre definire in Matlab l'intervallo dei valori della *variabile indipendente* t per i quali effettivamente vogliamo graficare la $f(t)$.

Matlab in realtà valuta una generica funzione $f(t)$ per un numero finito di punti all'interno dell'intervallo di valori di t specificato. I valori della $f(t)$ così ottenuti vengono poi *interpolati*.

Occorre quindi specificare di un intervallo(della variabile t) quanti punti andranno effettivamente ad essere considerati per l'interpolazione. Maggiore è tale numero, maggiore sarà la precisione del diagramma ma di converso maggiore sarà il tempo richiesto per il calcolo.

A tale scopo possiamo usare la sintassi presentata nell'esempio qui di seguito:

```
t = -10:0.01:10;
```

Mediante questo comando abbiamo specificato che l'intervallo da considerare per la variabile t è $[-10,10]$. Inoltre è stato scelto un *passo di campionamento* pari a 0.01: i punti considerati per l'interpolazione saranno tutti quelli distanziati tra loro di 0.01 a partire dall'estremo sinistro dell'intervallo. Nel caso dell'esempio essi saranno dunque $-10, -9.99, -9.98, \dots, 9.98, 9.99, 10$ (notiamo che 0.01 è contenuto un numero finito di volte in $[-10,10]$, per cui tra i punti da interpolare saranno inclusi anche gli estremi $-10, 10$).

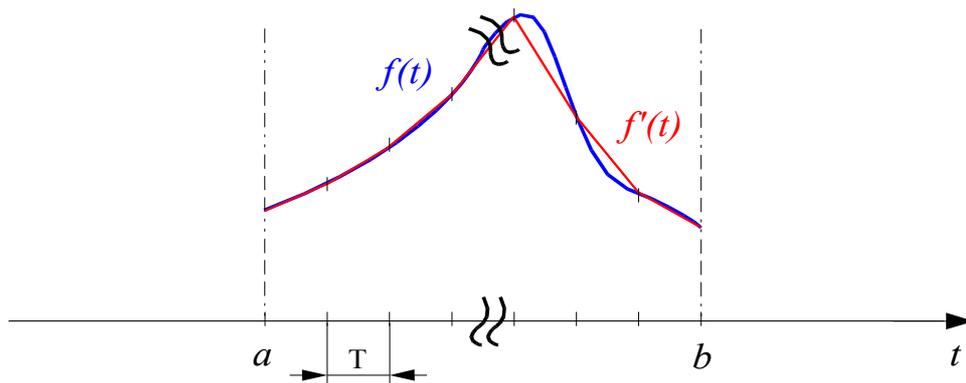


Fig. 14 - Una generica funzione $f(t)$ con la corrispondente approssimazione mediante interpolazione $f'(t)$ con passo di campionamento pari a T

t è in conclusione il vettore dei campioni della variabile indipendente.

Più in generale scriveremo dunque:

```
t = tmin:T:tmax;
```

dove tmin e tmax sono gli estremi dell'intervallo e T è il passo di campionamento.

Possiamo definire un intervallo anche utilizzando il comando Linspace:

Linspace

uso:

```
t = linspace(a,b,n);
```

Crea un vettore contenente n valori reali equispaziati compresi tra a e b, a partire da a.

Per definire i valori della funzione f(t) basta scrivere:

```
f = f(t);
```

dove f è il vettore dei campioni della variabile dipendente ed ha dimensione identica al vettore t, mentre f(t) è una espressione matematica in cui compare la variabile t. Matlab supporta tutte le funzioni elementari che conosciamo e le rispettive inverse. f(t) può essere anche una funzione composta mediante funzioni elementari. Per un elenco completo delle funzioni matematiche supportate da Matlab digitare:

```
help elfun
```

al prompt dei comandi.

Per diagrammare infine la funzione f(t) basta usare il comando PLOT, di seguito descritto.

PLOT

uso:

```
plot(t,f);
plot(t1,f1,t2,f2,...,tn,fn);
plot(t1,f1,'s1',t2,f2,'s2'...,tn,fn,'sn');
```

Stampa il diagramma di f in funzione di t; f e t devono avere la stessa dimensione.

È possibile stampare i grafici di più funzioni f1,...,fn in funzione dei rispettivi intervalli t1,...,tn in una stessa finestra utilizzando la seconda sintassi.

I parametri 's1',..., 'sn' sono *stringhe*⁵ che consentono di impostare attributi quali il colore o lo spessore del grafico delle rispettive funzioni. Digitando "help PLOT" al prompt di Matlab si può ottenere la lista dei valori possibili per il parametro 's'.

In figura è riportato un esempio di quanto detto finora.

```

Command Window
File Edit View Web Window Help
>> t = -10:0.01:10;
>> y1 = (exp(-abs(t))).*sin(10*t);
>> y2 = exp(-abs(t));
>> plot(t,y1,t,y2)
>> grid
>>
Ready
    
```

Fig. 15 - Definizione della funzione $f(t)=e^{-|t|}\sin(10t)$

Il diagramma stampato dal comando PLOT viene proposto in una nuova finestra.

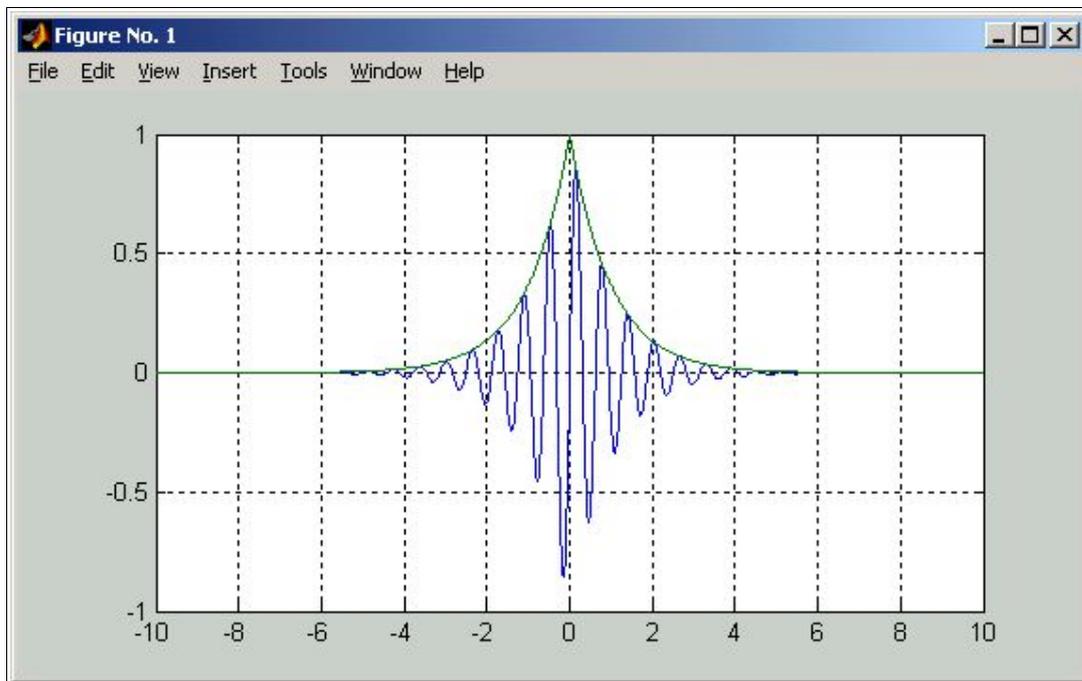


Fig. 16 - Il grafico della funzione $f(t)=e^{-|t|}\sin(10t)$

Riportiamo di seguito alcuni comandi utili per modificare un determinato grafico una

⁵ In Matlab una stringa va racchiusa tra gli apici "". Ad esempio 'Oggi è Domenica' è una stringa in Matlab.

volta stampato.

GRID

uso:

```
grid on;  
grid off;
```

Attiva(on) o disattiva(off) la griglia tarata sul grafico nella finestra di output corrente.

BOX

uso:

```
box on;  
box off;
```

Attiva(on) o disattiva(off) il bordo tarato sul grafico nella finestra di output corrente(selezionata).

Per creare una nuova finestra di output basta usare invece il comando FUGURE.

FIGURE

uso:

```
figure;
```

Aggiunge una finestra di output e la rende corrente.



Fig. 17 - Varie finestre di output grafico. Quella attiva è la No. 3.

Parte 6: Diagrammi 3D

Definizione di domini a 2 e 3 dimensioni

Possiamo a questo punto estendere i concetti presentati per la definizione in Matlab di intervalli di numeri reali, rappresentabili mediante segmenti, alla definizione di domini in \mathbb{R}^2 e in \mathbb{R}^3 , rappresentabili rispettivamente mediante parti di piano e parti di spazio.

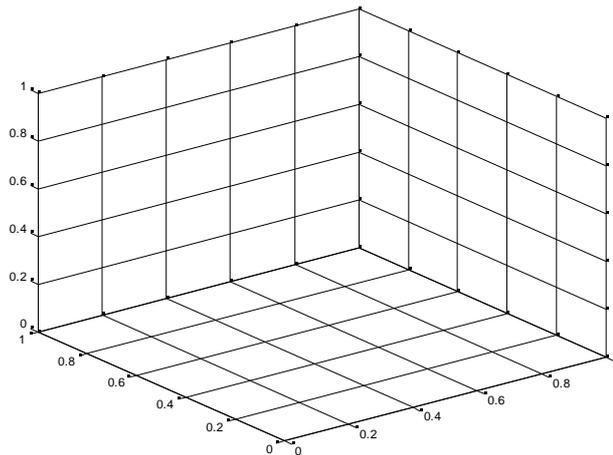


Fig. 18 - Un dominio tridimensionale.

Il comando MESHGRID rende notevolmente agevoli queste operazioni.

MESHGRID

USO:

```
[X,Y] = meshgrid(x,y);
[X,Y,Z] = meshgrid(x,y,z);
```

Definisce domini a due e tre dimensioni.

- Per creare un dominio bidimensionale occorre specificare solo le matrici X e Y ed i vettori x e y.

Le matrici X e Y sono create a partire dai vettori x e y che costituiscono degli intervalli di valori monodimensionali: le righe della matrice X sono copie del vettore x e le colonne della matrice Y sono copie del vettore y.

I vettori x e y possono essere definiti mediante il comando Linspace.

Ad esempio se scriviamo:

```
x = linspace(-1,1,5);
y = linspace(-2,2,10);
[X,Y] = meshgrid(x,y);
```

otterremo le seguenti matrici X e Y:

$$X = \begin{bmatrix} -1 & -0.5 & 0 & 0.5 & 1 \\ -1 & -0.5 & 0 & 0.5 & 1 \\ -1 & -0.5 & 0 & 0.5 & 1 \\ -1 & -0.5 & 0 & 0.5 & 1 \\ -1 & -0.5 & 0 & 0.5 & 1 \\ -1 & -0.5 & 0 & 0.5 & 1 \\ -1 & -0.5 & 0 & 0.5 & 1 \\ -1 & -0.5 & 0 & 0.5 & 1 \\ -1 & -0.5 & 0 & 0.5 & 1 \\ -1 & -0.5 & 0 & 0.5 & 1 \end{bmatrix} \quad Y = \begin{bmatrix} -2 & -2 & -2 & -2 & -2 \\ -1.5556 & -1.5556 & -1.5556 & -1.5556 & -1.5556 \\ -1.1111 & -1.1111 & -1.1111 & -1.1111 & -1.1111 \\ -0.6667 & -0.6667 & -0.6667 & -0.6667 & -0.6667 \\ -0.2222 & -0.2222 & -0.2222 & -0.2222 & -0.2222 \\ 0.2222 & 0.2222 & 0.2222 & 0.2222 & 0.2222 \\ 0.6667 & 0.6667 & 0.6667 & 0.6667 & 0.6667 \\ 1.1111 & 1.1111 & 1.1111 & 1.1111 & 1.1111 \\ 1.5556 & 1.5556 & 1.5556 & 1.5556 & 1.5556 \\ 2 & 2 & 2 & 2 & 2 \end{bmatrix}$$

La “sovrapposizione” di X e Y costituisce una griglia di punti che sono le *coppie ordinate* del dominio bidimensionale da noi definito. In corrispondenza di tali coppie ordinate potrà essere valutata una qualsiasi funzione reale di 2 variabili.

Se i vettori x e y sono uguali possiamo evitare di specificarli entrambi, scrivendo ad esempio:

```
[X,Y] = meshgrid(x);
```

in cui abbiamo indicato solo il vettore x. L'effetto sulle matrici X e Y resta comunque quello relativo all'uso tradizionale di MESHGRID.

- Analogamente se specifichiamo anche la matrice Z e il rispettivo vettore z, possiamo definire un dominio tridimensionale. Questa volta dunque la griglia di punti si estenderà non più sul piano ma nello spazio. È effettivamente un reticolo tridimensionale di punti, un insieme di *triple ordinate*. Le matrici X,Y,Z saranno in realtà degli *array a tre dimensioni*.

Funzioni reali di due variabili reali

Per definire una funzione reale di due variabili reali, una volta definitone il dominio bidimensionale mediante il comando MESHGRID basta usare la sintassi:

```
Z = f(X,Y);
```

in cui f è una generica espressione matematica (una qualsiasi composizione di funzioni elementari supportate da Matlab) che contenga le matrici X e Y. Z sarà un array a tre dimensioni contenente i punti dello spazio che matematicamente rappresentano la funzione:

$$z = f(x, y)$$

Un esempio chiarirà meglio quanto detto.

Per definire la funzione di due variabili:

$$z = e^{(x^2 + y^2)}$$

in relazione al dominio:

$$x, y \in [-5, 5]$$

basterà scrivere in Matlab:

```
x = linspace(-1,1,100);  
y = linspace(-2,2,200);  
[X,Y] = meshgrid(x,y);  
Z = exp(X.^2 + Y.^2);
```

Anche in questo caso, come ci si potrà rendere conto, Matlab compie un'operazione di interpolazione. La funzione Z viene valutata solo per quei punti che effettivamente costituiscono la griglia di punti del dominio bidimensionale dichiarato. I punti intermedi non vengono invece calcolati. All'atto della rappresentazione di Z, mediante uno dei comandi che vedremo di seguito, i punti intermedi saranno comunque disegnati sullo schermo utilizzando per la tecnica dell'interpolazione.

È evidente che quanto più fitta è la griglia (quanto più grande è il numero di punti che la costituisce) più accurato sarà il grafico della funzione Z ma al contempo più elevati saranno i tempi di calcolo.

La risoluzione (numero di punti) della griglia potrà essere variata indirettamente accrescendo il numero di punti dei vettori x e y come già sappiamo fare ad esempio mediante il comando Linspace.

Funzioni vettoriali di variabili reali

Il caso di una funzione vettoriale di tre variabili reali, e quindi di un vettore a tre dimensioni, include anche quello di funzioni vettoriali di due o una variabile reale. Presenteremo dunque solo tale caso (f.v. di 3 var. reali).

Una funzione vettoriale di un vettore a tre dimensioni è rappresentata in forma compatta da un'espressione matematica del tipo:

$$\mathbf{F} = \mathbf{F}(x, y, z)$$

dove \mathbf{F} è un vettore. In forma estesa possiamo scrivere:

$$\mathbf{F} = [u, v, w] = [u(x, y, z), v(x, y, z), w(x, y, z)]$$

dove u, v, w sono le componenti scalari di \mathbf{F} e sono funzioni dei numeri reali x, y, z . In Matlab per specificare \mathbf{F} , o meglio le sue componenti scalari basta scrivere:

```
[U,V,W] = [u(X,Y,Z),v(X,Y,Z),w(X,Y,Z)];
```

dove X, Y, Z sono state definite mediante il comando MESHGRID e u, v, w sono delle generiche espressioni matematiche (composizioni delle funzioni matematiche elementari supportate da Matlab) contenenti X, Y, Z .

Ad esempio per definire la funzione:

$$\mathbf{F} = [u, v, w] = [\sin x, \cos y, \sin x \cos y]$$

basterà scrivere:

```
x = linspace(-1,1,100);
y = linspace(-2,2,200);
z = linspace(-1,1,100);
[X,Y,Z] = meshgrid(x,y,z);
[U,V,W] = [sin(X),cos(Y),(sin(X)).*(cos(Y))];
```

Rappresentazione grafica di funzioni

Non ci resta a questo punto che conoscere i comandi che ci consentono in Matlab di rappresentare graficamente i tipi di funzioni che abbiamo introdotto sopra.

Il seguente gruppo di comandi serve a visualizzare funzioni reali di due variabili reali.

MESH

uso:

```
mesh(Z)
```

Disegna a schermo una funzione di 2 variabili $Z = f(X, Y)$ utilizzando la tecnica di visualizzazione *mesh surface* e utilizzando la *colormap* di default "jet".

Esempio:

```
x = linspace(-3,3,20);
[X,Y] = meshgrid(x);
```

```
Z = exp(sqrt(X.^2 + Y.^2));  
mesh(Z)
```

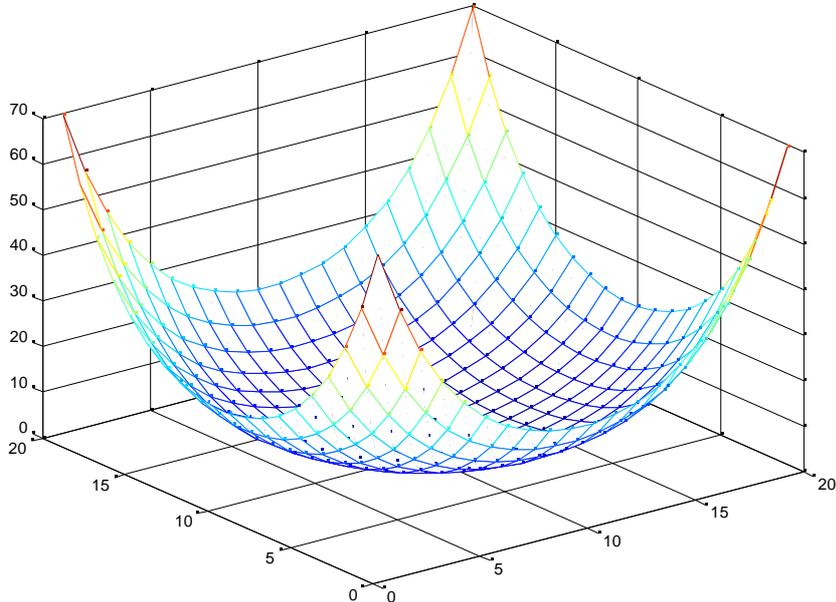


Fig. 19 - Il grafico della funzione $z = e^{\sqrt{x^2 + y^2}}$ disegnato attraverso il comando *MESH*.

SURF

USO:

```
surf(Z);
```

Disegna a schermo una funzione di 2 variabili $Z = f(X, Y)$ utilizzando la tecnica di visualizzazione *colored surface* e utilizzando la *colormap* di default “jet”. Di default la superficie disegnata presenta un modo di ombreggiatura di tipo “faceted”, ossia con mesh nere sovrapposte.

SHADING

USO:

```
shading tipo;
```

Consente di cambiare il modo di ombreggiatura.

L'attributo *tipo* rappresenta il modo di ombreggiatura scelto tra i seguenti:

- | | |
|----------|---|
| flat: | ombreggiatura costante ad aree |
| interp: | ombreggiatura ad interpolazione bilineare (Gouraud) |
| faceted: | ombreggiatura costante ad aree con mesh sovrapposte |

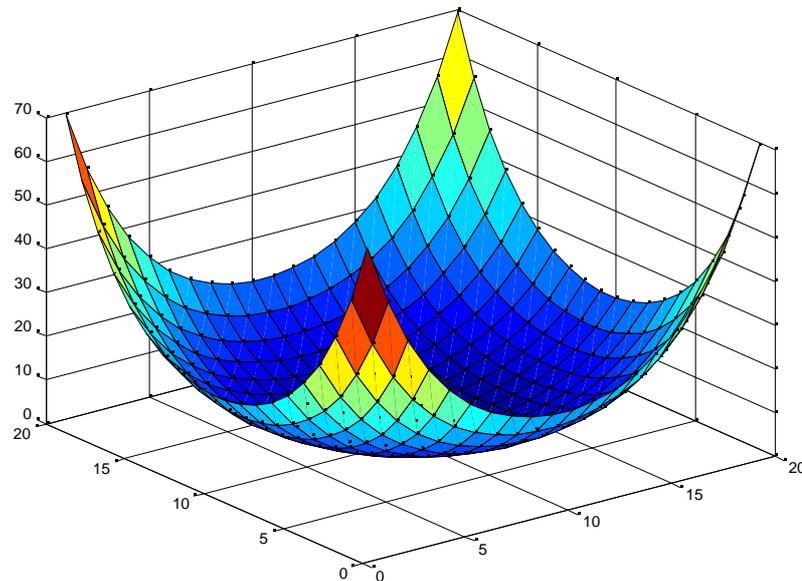


Fig. 20 - Il grafico della funzione $z = e^{\sqrt{x^2 + y^2}}$ disegnato attraverso il comando *MESH*.

I comandi qui di seguito invece ci permettono di rappresentare graficamente funzioni vettoriali.

CONEPLOT

uso:

```
coneplot(X,Y,Z,U,V,W,Cx,Cy,Cz);
coneplot(.....,'quiver')
```

Disegna schermo una funzione vettoriale delle tre variabili reali x,y,z usando una rappresentazione a coni tangenti alle linee di campo della funzione.

Il numero di coni disegnati può essere impostato mediante i parametri Cx,Cy,Cz .

X,Y,Z costituiscono le matrici rappresentative del dominio tridimensionale;

U,V,W costituiscono le matrici rappresentative delle componenti scalari della funzione vettoriale;

Cx,Cy,Cz costituiscono le matrici rappresentative del numero e della posizione dei coni lungo x,y e z rispettivamente. Vanno definite mediante il comando *MESHGRID*.

'quiver' disegna frecce anzicchè coni.

Esempio:

```
x = linspace(-3,3,20);  
[X,Y,Z] = meshgrid(x);  
[Cx,Cy,Cz] = meshgrid(linspace(-1,1,5));  
U = sin(X);  
V = cos(Y);  
W = Z;  
coneplot(X,Y,Z,U,V,W,Cx,Cy,Cz)
```

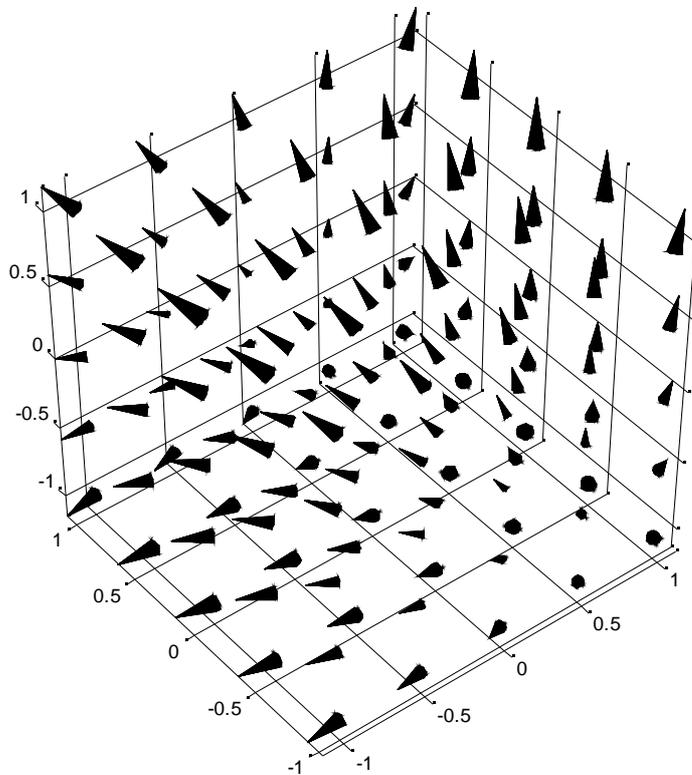


Fig. 21 - Esempio di rappresentazione di funzione vettoriale mediante CONELOT.

STREAMLINE

uso:

```
streamline(X,Y,Z,U,V,W,Sx,Sy,Sz);
```

Disegna schermo una funzione vettoriale delle tre variabili reali x,y,z usando una rappresentazione mediante linee di campo.

I parametri X,Y,Z,U,V,W sono gli stessi del comando CONEPLLOT.

Sx,Sy,Sz costituiscono le matrici rappresentative del numero e della posizione dei punti da interpolare per disegnare le linee di campo lungo x,y e z rispettivamente. Vanno definite mediante il comando MESHGRID.

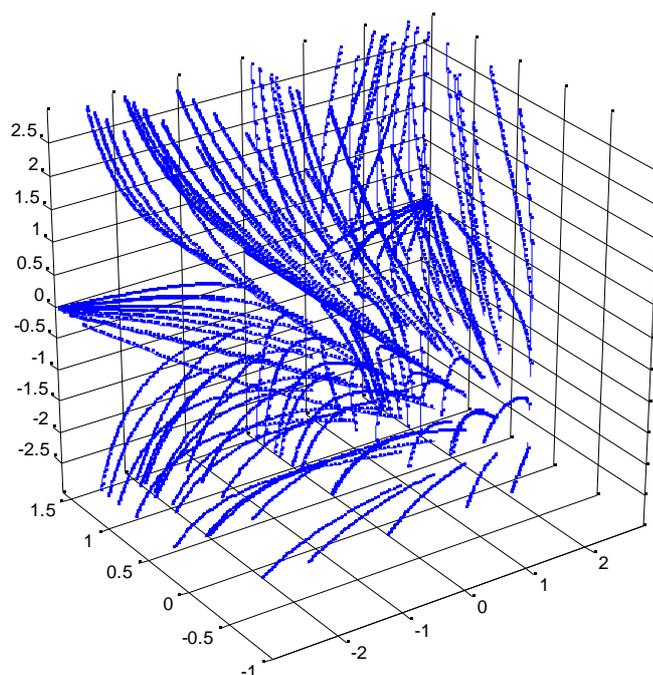


Fig. 22 - Rappresentazione della stessa funzione mediante linee di flusso attraverso il comando STREAMLINE.