MANUALE MATLAB

A cura di Giuseppe Ciaburro

http://www.ciaburro.it

info@ciaburro.it

Indice

Figura 1: Giuseppe Ciaburro

Capitolo 1

Introduzione

1.1 Matlab

Che cosa è Matlab? MATLAB è un linguaggio ad alto rendimento per la computazione tecnica. Esso integra il calcolo, la visualizzazione e la programmazione in un ambiente di facile impiego in cui i problemi e le soluzioni sono espressi in notazione matematica familiare. E' strutturato secondo il seguente schema:

- Matematica e calcolo.
- Sviluppo di procedura.
- Modellistica, simulazione e prototyping.
- Analisi di dati,esplorazione e visualizzazione.
- Disegno industriale e scientifico.
- Sviluppo di applicazione, compreso la costruzione grafica dell' interfaccia di utente.

MATLAB è un sistema interattivo in cui l'elemento di base è un array quindi non richiede il dimensioning.Ciò permette la risoluzione di molti problemi di calcolo tecnici, in particolare quelli con le formulazioni vettorali e matriciali, attraverso algoritmi molto più semplici e snelli rispetto a quelli che sarebbero necessari in un programma in linguaggio scalare non interattivo quali C o il fortran. Il nome MATLAB corrisponde al laboratorio della matrice.MATLAB era originalmente scritto per fornire facile accesso al software delle matrici;si è sviluppato dal LINPACK e dal EISPACK, che rappresentano insieme la punta del progresso software per il calcolo delle matrici. MATLAB si è evoluto durante gli anni con input da molti utenti.In ambienti universitari è l'attrezzo didattico standard per corsi introduttivi e corsi avanzati, nella matematica, nell'ingegneria e nella scienza. MATLAB caratterizza una famiglia delle soluzioni application-specific denominate toolboxes. Molto utile per la maggior parte degli utenti di MATLAB, toolboxes, fornisce le basi per applicare la tecnologia specializzata.I toolboxes sono collezioni complete di funzioni MATLAB (M-files)che estendono l'ambiente di MATLAB per risolvere particolari categorie di problemi.Gli ambienti in cui i toolboxes sono disponibili sono:

- elaborazione dei segnali,
- sistemi di controllo,
- reti neurali,
- logica incoerente,
- wavelets,
- simulazione e molti altri.

1.2 Per iniziare

Iniziare con MATLAB

Questo tutorial è teso ad aiutare chi inizia ad imparare MATLAB. Contiene un numero elevato di esempi così da poter da subito utilizzare MATLAB ,e dovrebbe poi essere utile per il seguito. Per attivare MATLAB su un PC o Mac, basta un duplice scatto sull'icona di MATLAB. Per attivare MATLAB su un sistema UNIX, digitare matlab al prompt del sistema operativo. Per uscire da MATLAB in qualsiasi istante, digitare QUIT al prompt di MATLAB. Se si necessita di più assistenza, digitare HELP al prompt di MATLAB, o cliccare sul menu dell'HELP su un PC o Mac.

Matrici e Magic Squares

Il migliore modo per iniziare con MATLAB è quello di imparare a maneggiare le matrici, questa sezione mostra come fare . Una matrice è in MATLAB, un ordine rettangolare di numeri,significato speciale qualche volta è adottato per le $1 - by - 1$ matrici che sono scalari, e per matrici con solamente una riga o colonna che sono vettori. MATLAB ha altri modi di immagazzinare dati numerici e dati non numerici, ma all'inizio, di solito è meglio pensare a tutto come una matrice. Le operazioni in MATLAB sono destinate ad essere più naturali possibile. Dove gli altri linguaggi di programmazione lavorano con numeri uno alla volta, MATLAB permette di lavorare facilmente e rapidamente con matrici di interi.

Immettere le matrici

Si possono registrare matrici in MATLAB in molti modi diversi.

- introdurre un elenco esplicito di elementi.
- caricare matrici da files di dati esterni.
- generare matrici utilizzando la funzione built-in.
- creare matrici con le proprie funzioni in M-files.

Cominciamo a registrare la matrice come un elenco dei suoi elementi. Si seguano a tal proposito solamente alcune convenzioni di base:

- Separare gli elementi di una riga con spazi vuoti o virgole.
- Usare un punto e virgola ";" per indicare la fine di ciascuna fila.
- Racchiudere l'elenco intero di elementi con parentesi quadrate , [ ].

Per registrare la matrice di Dürer, semplicemente digitare:

```
A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
```

MATLAB espone a video solo la matrice digitata,

```
A = 16  3  2 13
 5 10 11  8
 9  6  7 12
 4 15 14  1
```

Questo procedimento associa precisamente i numeri a porzioni di memoria. Una volta fornita la matrice, essa è registrata automaticamente nel workspace di MATLAB. Ora si può indicarla semplicemente come A.Quindi si può sommare, trasporre, etc .Si è probabilmente già consapevoli delle proprietà speciali di una matrice magic square con i vari modi di sommare i suoi elementi. Se si prende la somma lungo qualsiasi riga o colonna, o lungo una delle due diagonali principali, si ottiene sempre lo stesso numero. Verificare tutto ciò usando MATLAB. Il primo tentativo è il seguente:

```
sum(A)
```

MATLAB risponde con:

```
ans = 34 34 34 34
```

Quando non si specifica una variabile di output, MATLAB usa la variabile ans, per immagazzinare i risultati di un calcolo. Si è calcolato un vettore fila che contiene certamente le somme delle colonne di A; ogni colonna ha la stessa somma, la somma magica, 34. Cosa si può dire sulle somme delle righe? MATLAB ha una preferenza per lavorare con le colonne di una matrice, così il modo più facile per ottenere le somme delle righe è quello di trasporre la matrice,si calcoli la colonna somma della trasposta, e poi si trasponga il risultato. L'operazione di trasposizione è denotata da un apostrofo '. Così:

```
A'
```

determina il risultato seguente:

```
ans = 16 5 9 4
      3 10 6 15
      2 11 7 14
      13 8 12 1
```

e

```
sum(A ') '
```

produce un vettore colonna che contiene le somme delle righe:

```
ans = 34 34 34 34
```

La somma degli elementi della digonale principale è ottenuta facilmente con l'aiuto della funzione diag che estrae solo gli elementi dellla diagonale.

```
diag(A)
```

produce:

```
ans = 16 10 7 1
```

e

```
sum(diag(A))
```

produce:

```
ans = 34
```

L'altra diagonale, l'antidiagonale così chiamata non è così importante MATLAB non ha così matematicamente, una funzione fatta apposta . Ma una funzione originariamente creata per uso in grafica, fliplr riporta una matrice da sinistra a destra.

```
sum(diag(fliplr(A)))

ans = 34
```

Si è così verificato che la matrice di Dürer è davvero una magic square e, nel processo,si sono utilizzate operazioni sulle matrici di MATLAB. Le sezioni seguenti continuano a usare questa matrice per illustrazioni supplementari .

Capitolo 2

Fondamenti di Matlab

2.1 Capacità di Matlab

Pedici

L'elemento in riga i e colonna j di A è denotato con A(i,j). Per esempio:

```
A(4,2)
```

è il numero nella quarta riga e seconda colonna. Per la nostra magic square, `A(4,2)` è 15. Così è possibile calcolare la somma degli elementi nella quarta colonna di A digitando:

```
A(1,4) + A(2,4) + A(3,4) + A(4,4)
```

Questo produce:

```
ans = 34
```

ma non è il modo più elegante di sommare una singola colonna. È anche possibile per assegnare gli elementi di una matrice utilizzare un singolo pedice, A(k). Questo è il modo solito di citare vettori riga e vettori colonna. Ma si può anche applicare ad una matrice bidimensionale in questo caso la matrice è considerata come un vettore colonna, formato dalle colonne della matrice originale. Così, per la nostra magic square, A(8) è un altro modo di assegnare il valore 15 immagazzinato nella posizione A(4,2), cioè quarta riga seconda colonna. Se si tenta di usare il valore di un elemento della matrice al di fuori di essa, si commette un errore:

```
t = A(4,5)
```

Index exceeds matrix dimensions (Messaggio di errore)

D'altra parte, se si immagazzina un nuovo valore in un elemento della matrice, c'è un aumento dell'ordine per accomodare il nuovo venuto:

```
A(4,5) = 17

A = 16  3  2 13
     0  5 10 11
     8  0  9  6
     7 12  0  4
    15 14  1 17
```

L'operatore : (due punti)

Il due punti,:, è uno dei più importanti operatori di MATLAB.
Si trova in molte forme diverse.
L'espressione:

```
1:10
```

è un vettore riga che contiene i numeri interi da 1 a 10

```
1 2 3 4 5 6 7 8 9 10
```

Per ottenere una sequenza, si deve specificare un incremento. Per esempio:

```
100:-7:50
```

è

```
100 93 86 79 72 65 58 51
```

cioè una sequenza di numeri da 100 a 50 con passo uguale a -7;
mentre

```
0:pi/4:pi
```

è

```
0 0.7854 1.5708 2.3562 3.1416
```

Le espressioni sottoscritte che coinvolgono i due punti assegnano porzioni di una matrice.

```
A(1:k,j)
```

rappresenta i primi k elementi della colonna jth di A. Così:

```
Sum(A(1:4,4))
```

calcola la somma della quarta colonna. Ma c'è un modo migliore . Il due punti assegna tutti gli elementi in una riga o colonna di una matrice e la keyword END assegna l'ultima riga o colonna. Così

```
sum(A(:,end))
```

calcola la somma degli elementi nell'ultima colonna di A.

```
ans = 34
```

Perchè la somma della magic square è uguale a 34? Se i numeri interi da 1 a 16 sono ordinati in quattro gruppi con somme uguali, quella somma deve essere:

```
sum(1:16)/4
```

che, chiaramente, è

```
ans = 34
```

Se si ha accesso al Symbolic Math Toolbox si può scoprire che la somma per un n-by-n magic square è $(n^3 + n)/2$.

La funzione magic

La funzione magic di MATLAB costruisce magic square di qualsiasi dimensione.Non a caso, questa funzione è chiamata magic.

```
B = magic(4)

B = 16 2 3 13
    5 11 10 8
    9 7 6 12
    4 14 15 1
```

2.2 Operazioni

In MATLAB sono definite le seguenti operazioni aritmetiche:

1) addizione +
2) sottrazione -
3) moltiplicazione *
4) divisione /
5) elevamento a potenza ^

Quando l'espressione da valutare è troppo lunga per stare su di un'unica riga di comando,è possibile utilizzare un carattere di continuazione dato da . . . (tre punti). Ad esempio,

```
>> 1 + 1/4 + ...
1/8 ans = 1.375
```

E' possibile verificare che poichè in MATLAB non vi è alcuna distinzione tra variabili intere, reali o complesse, il risultato dell'operazione di elevamento a potenza nel caso di esponenti frazionari può non corrispondere a quello naturalmente atteso. Ad esempio, volendo calcolare la radice cubica di -5, si ottiene:

```
>> (-9)^(1/5)
ans = 1.2555 + 0.9122i
```

e non un numero reale. Il problema è dovuto al fatto che MATLAB, lavorando in notazione complessa, calcola come prima radice cubica di -9 una delle due con parte immaginaria non nulla. Anche in MATLAB è possibile alterare le precedenze classiche delle operazioni aritmetiche mediante l'uso opportuno delle parentesi tonde. Ad esempio, se nel caso precedente si fosse scritto (- 9) 1 /5 si sarebbe ottenuto il valore -l.8000 ossia, giustamente, - 9/5. Per quanto riguarda i vettori, le operazioni elementari si estendono (quando ben definite) in modo del tutto naturale, con l'eccezione delle operazioni di divisione e di elevamento a potenza. Ad esempio,

```
>> a = [1:4];
>> b = [1:3];
>> c = [3 2 6 -1] ;
>> a+c        (somma di vettori riga)
ans =
    4      4     9      3

>> a-c        (differenza di vettori riga)

ans =
    -2     0     -3     5

>> a+b
??? Errar using ==> + Matrix  dimension must agree.

>>a*c
??? Errar using ==> * Inner matrix  dimension must agree.
```

Le ultime istruzioni ci indicano che, le operazioni fra vettori sono valide solo se le dimensioni sono consistenti. Infatti, dato un vettore riga s di dimensione n (ossia una matrice 1 * n) ed un vettore colonna d di dimensione m, si potrà eseguire il prodotto scalare s*d solo se m=n (ottenendo in tal caso uno scalare).

```
>> s=[1:4];
>> d=[1;2;3;4];
```

```
>> s*d
ans=
   30
```

Quindi d*s fornirà una matrice m*n.

```
>> s=[1:4];
>> d=[1;2;3;4];
>> M=d*s
M=
    1     2     3     4
    2     4     6     8
    3     6     9    12
    4     8    12    16
    5    10    15    20
```

2.3 Espressioni

Come altri linguaggi di programmazione, MATLAB prevede espressioni matematiche, ma queste espressioni coinvolgono matrici intere. Il blocco di espressioni è:

- Variabili
- Numeri
- Operatori
- Funzioni

Variabili

MATLAB non richiede alcuna dichiarazione del tipo della dimensione. Quando MATLAB incontra un nome variabile nuovo, crea automaticamente la variabile e stanzia l'ammontare adatto in deposito. Se la variabile già esiste, MATLAB cambia i suoi contenuti e, se necessario, effettua un nuovo deposito. Per esempio

```
elenco = 30
```

crea una 1-by-1 matrice di nome elenco e immagazzina il valore 30 come suo singolo elemento. I nomi delle variabili consistono di una lettera, seguita da qualsiasi numero di lettere, cifre o sottolinea. MATLAB usa solamente i primi 31 caratteri di un nome di una variabile. MATLAB è case sensitive; distingue tra uppercase e lowercase . A ed a non sono la stessa variabile. Per vedere la matrice assegnata a ciascuna variabile, semplicemente digitare il nome della variabile.

Numeri

MATLAB usa notazione decimale e convenzionale, con un punto decimale e opzionale ed indicando quantità positive o il segno meno, per numeri negativi. La notazione scientifica usa la lettera e per specificare la potenza decimale.I numeri immaginari usano i o j, come un suffisso. Degli esempi di numeri legali sono:

```
3                      -99                    0.0001

9.6397238        1.60210e-20              6.02252e23

1i                -3.14159j                    3e5i
```

Tutti i numeri che usano il format `long` specificato dall'IEEE sono immagazzinati internamente con lo standard floating-point. I numeri floating-point hanno una precisione limitata di 16 cifre decimali significative e una serie limitata da `10^-308` a `10^308+`. (Il computer di VAX usa una configurazione diversa, ma la sua precisione e serie sono quasi le stesse.)

Operatori

Le espressioni usano operatori di aritmetica familiari e le regole della precedenza.

```
+   Addizione
-   Sottrazione
*   Moltiplicazione
/   Divisione
\   Left division
```

(sono descritte nella sezione su matrici e Algebra Lineare in MATLAB)

```
^   potenza
'   Asse coniugato complesso trasposto
()  Valutazione specifiche di ordine.
```

Funzioni

MATLAB prevede un gran numero di funzioni matematiche standard,incluso abs, sqrt, exp, e sin. La radice quadrata o il logaritmo di un numero negativo non è un errore; il risultato complesso è prodotto automaticamente. MATLAB prevede anche funzioni matematiche molto più avanzate, incluso Bessel e gamma function. La maggior parte di queste funzioni accettano argomenti complessi. Per un elenco delle funzioni matematiche elementari, digitare:

```
help elfun
```

Per un elenco più avanzato delle funzioni matematiche e delle matrici digitare:

```
help specfun
help elmat
```

Alcune delle funzioni, come sqrt e sin, sono calcolate al momento. Loro sono parte del cuore di MATLAB quindi sono molto efficienti, ma i dettagli della computazione non sono prontamente accessibili. Le altre funzioni, come gamma e sinh sono perfezionate in M-files. Si può vedere il codice originario che le valuta ed eventualmente anche cambiarlo . Molte funzioni speciali prevedono valori di costanti utili.Infinito è generato dividendo un valore diverso da zero per zero, o valutando bene espressioni matematiche e definite che tendono all'infinito. Not-a-number è generato tentando di valutare espressioni tipo 0/0 o Inf-Inf che non hanno valori matematici ben definiti. I nomi delle funzione non sono riservati. È possibile utilizzare alcuni di tali nomi con una nuova variabile, come:

```
eps = 1.e-6
```

e poi usare quel valore in calcoli susseguenti. La funzione originale può essere ripristinata con :

```
clear eps
```

Alcuni nomi sono associati a precisi valori, come ad esempio:

pi : 3.14159265...

i : unità Immaginaria,

j : come i

eps : Floating-point con precisione, 2 -52

realmin : il più piccolo floating-point, 2 -1022

realmax : il più grande floating-point,(2-e)2 1023

Inf : Infinito

NaN : Not-a-number.

Abbiamo già ha visto molti esempi di espressioni di MATLAB. Qui ci sono altri esempi, ed i valori che ne risultano.

```
rho = (1+sqrt(5)) /2 rho = 1.6180

a = abs(3+4i) a = 5

z = sqrt(besselk(4/3,rho-i)) z = 0.3730+ 0.3214i

huge= l'exp(log(realmax)) huge = 1.7977e+308

toobig = il pi*huge toobig = Inf.
```

2.4 Lavorare con le matrici

Questa sezione presenta altri modi di creare matrici.

Generazione di matrici

MATLAB prevede quattro funzioni che generano matrici di base.
Degli esempi sono:

```
Z = zeros(2,4) Z =
    0 0 0 0
    0 0 0 0

F = 5*ones(3,3) F =
    5 5 5
    5 5 5
    5 5 5

N = fix(10*rand(1,10)) N =
    4 9 4 4 8 5 2 6 8 0

R = randn(4,4) R =
    1.0668 0.2944 -0.6918 -1.4410
    0.0593 -1.3362 0.8580  0.5711
   -0.0956  0.7143 1.2540 -0.3999
   -0.8323  1.6236 -1.5937 0.6900

zeros All zeros

ones All ones

rand Uniformly distributed random elements

randn Normally distributed random elements.
```

Manipolare le matrici

Poniamo l'attenzione su alcune funzioni che consentono di manipolare matrici e vettori.

• abs
Applicata ad una matrice reale, produce la matrice dei valori assoluti. Applicata ad un numero complesso, ne calcola il modulo. Esempio:

```
z=3+i*4

z =

    3.0000+ 4.0000i

abs(z)

ans =

      5
```

• tril (oppure triu)

Estraggono da una matrice la parte triangolare inferiore(superiore). Indicando un numero positivo o negativo come secondo argomento è possibile estrarre le altre diagonali della matrice. Ad esempio:

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

A =

    1     2     3
    4     5     6
    7     8     9

triu(A)

ans =

    1     2     3
    0     5     6
    0     0     9

triu(A,-1)

ans =

    1     2     3
    4     5     6
    0     8     9

triu(A,1)

ans =

    0     2     3
    0     0     6
    0     0     0
```

• diag

Applicata ad una matrice ne estrae la diagonale,applicata ad un vettore crea una matrice diagonale.

```
A=[5 6 ; 7 8]

A =

     5     6
     7     8

diag(A)

ans =

     5
     8
```

```
diag(ans)

ans =

     5     0
     0     8
```

• sum

Applicata ad una matrice fornisce un vettore che contiene le somme per colonna degli elementi della matrice; applicata ad un vettore fornisce uno scalare dato dalla somma degli elementi del vettore.

```
A=[0 1 2 ;3 4 5 ;6 7 8 ]

A =

     0     1     2
     3     4     5
     6     7     8

sum(A)

ans =

     9    12    15

B=[0 1 2]

B =

     0     1     2

sum(B)

ans =

     3
```

• max (oppure min)

Se applicata ad un vettore calcola il massimo (rispettivamente il minimo) degli elementi di un vettore. Se applicata ad una matrice produce un vettore che contiene il massimo degli elementi della matrice per colonne.

• norm

Calcola la norma 2 di un vettore o di una matrice. Inoltre, norm (A, inf) calcola la norma infinito di A, mentre norm (A, 1) calcola la norma 1 di A.

• cond

Calcola il numero di condizionamento in norma 2 di una matrice A.

• eig

calcola gli autovalori di una matrice quadrata. Se richiamata nel modo seguente `[X, D] =eig (A)` produce una matrice diagonale D che contiene gli autovalori di A e la matrice X le cui colonne sono gli autovettori di A.

```
A=[0 1 2 ;3 4 5 ;6 7 8 ]; [X,D]=eig(A)

X =

    0.7997   -0.1648    0.4082
    0.1042   -0.5058   -0.8165
   -0.5913   -0.8468    0.4082


D =

   -1.3485         0         0
         0   13.3485         0
         0         0    0.0000
```

Funzioni intrinseche matematiche

Riportiamo un elenco di alcune funzioni matematiche predefinite in MATLAB. Molte di queste funzioni, presentate qui nella loro versione scalare, possono essere applicate anche a variabili matriciali. Indicando con x ed y due numeri reali e z un numero complesso, ricordiamo le seguenti funzioni:

```
sqrt(x)       \sqrt{x}

round(x)      arrotondamento: x = 3.6 ----> 4

fix(x)        troncamento:    x = 3.6 ----> 3

sign(x)       segno di x(vale 1,0 o -1)

sin(x)        sinx

cos(x)        cosx

tan(x)        tanx

sinh(x)       sinh x

cosh(x)       coshx

tanh(x)       tanhx

asin(x)       arcsinx

acos(x)       arccosx

atan(X)       arctanx

exp(x)        e^x

log(x)        log x (logaritmo naturale di x)
```

```
log10(x)      log10 x (logaritmo in base lodi x)

real(z)       parte reale di z

imag(z)       parte immaginaria di z

conj(z)       z* (complesso coniugato di z)
```

Load

Il comando load legge file binari che contengono matrici, generati da precedenti sessioni di MATLAB, o legge file di testo contenenti dati numerici. I file di testo dovrebbero essere organizzati come una tabella rettangolare di numeri,separata da spazi vuoti con una riga per linea, e un numero uguale di elementi in ciascuna fila. Per esempio,se al di fuori di MATLAB si crea un file di testo che contiene queste quattro linee:

```
16.0   3.0  2.0  13.0
 5.0  10.0 11.0   8.0
 9.0   6.0  7.0  12.0
 4.0  15.0 14.0   1.0
```

Immagazzinando poi il file sotto il nome matrice.dat, attraverso il comando

```
load  matrice.dat
```

si legge il file e si crea un variabile, matrice, contenente la nostra matrice d'esempio.

M-Files

Si possono creare matrici utilizzando M-files che sono file di testo contenenti codici MATLAB.Per fare questo basta creare un file che contiene lo stesso statement che si desiderava digitare al prompt di MATLAB,e salvare il file sotto un nome che finisce in .m. Per accedere ad un editor di testo su un PC o Mac, scegliere OPEN o NEW dal menu file o pigiare il bottone adatto sul toolbar. Per accedere ad un editor di testo sotto UNIX, usare il simbolo ! seguito dal comando desiderato. Per esempio, creiamo un file che contiene queste cinque linee:

```
A = [...
16.0 3.0 2.0 13.0
5.0 10.0 11.0 8.0
9.0  6.0 7.0 12.0
4.0 15.0 14.0 1.0];
```

immagazziniamo il file sotto il nome matrice.m. Poi digitiamo lo statement

```
matrice
```

che legge il file e crea un variabile, A, contenente la nostra matrice d'esempio.

Concatenazione

La Concatenazione è il processo di congiunzione di piccole matrici per fare matrici più grandi.Infatti,la prima matrice è stata creata concatenando i suoi elementi individuali. La parentesi quadrata, [ ], è l'operatore della concatenazione.Per un esempio, si parta con la 4-by-4 magic square, A, e si formi:

```
B = [A A+32; A+48 A+16]
```

Il risultato è una matrice 8-by-8 , ottenuta congiungendo le quattro submatrici.

```
B =

   16  3  2 13 48 35 34 45
    5 10 11  8 37 42 43 40
    9  6  7 12 41 38 39 44
    4 15 14  1 36 47 46 33
   64 51 50 61 32 19 18 29
   53 58 59 56 21 26 27 24
   57 54 55 60 25 22 23 28
   52 63 62 49 20 31 30 17
```

Questa matrice mostra un altro modo per ottenere una magic square. I suoi elementi sono un riordinamento dei numeri interi 1:64, le somme delle colonne forniscono il valore corretto per una 8-by-8 magic square.

```
sum(B) ans = 260 260 260 260 260 260 260 260
```

Ma la somma delle righe , sum(B ') ', non fornisce lo stesso risultato.

Come cancellare Righe e Colonne

Si possono cancellare righe e colonne da una matrice utilizzando solo un paio di parentesi quadrate. Cominciamo con:

```
X = A;
```

Poi, per cancellare la seconda colonna di X, usare:

```
X(:,2) = [ ]
```

Questo cambia X in

```
X =
   16  2 13
    5 11  8
    9  7 12
    4 14  1
```

Se si cancella un singolo elemento da una matrice, il risultato non è più una matrice.Così,

```
X(1,2) = [ ]
```

produce un errore. Comunque,un singolo pedice cancella un singolo elemento, o sequenza di elementi, e rifoggia gli elementi che rimangono in un vettore riga. Quindi:

```
X(2:2:10) = [ ]
```

fornisce:

```
X = 16 9 2 7 13 12 1.
```

La Finestra di Comando

Finora, si è usato la linea di comando di MATLAB, digitando comandi espressioni, e vedendo i risultati stampati nella finestra di comando.Questa sezione descrive alcuni modi di alterare l'area della finestra di comando. Se il sistema permette di selezionare la fonte per la finestra di comando o lo stile dei caratteri, si raccomanda di utilizzare una fonte dall'ampiezza fissa,come Fixedsys o Courier, per fornire una spaziatura corretta.

Il Comando FORMAT

Il comando format controlla la configurazione numerica dei valori esposta da MATLAB. Il comando regola solamente come i numeri sono esposti, non come MATLAB li calcola o li salva. Di seguito sono indicati diversi formati,insieme con l'output che ne deriva, prodotto da un vettore x con componenti di dimensioni diverse.

```
x = [4/3 1.2345e-6]

format short
     1.3333 0.0000 1.3333 0.0000

format short e
     1.3333e+000 1.2345e-006

format short g
     1.3333 1.2345e-006

format long
     1.33333333333333 0.00000123450000

format long e
     1.333333333333333e+000 1.234500000000000e-006

format long g
     1.33333333333333 1.2345e-006

format bank
     1.33 0.00

format rat
     4/3 1/810045

format hex 3ff5555555555555 3eb4b6231abfd271 .
```

Se il più grande elemento di una matrice è più grande di 10^3 o più piccolo di 10^{-3}, MATLAB applica un fattore di scala comune per i format short e long. Inoltre, sopprime molte delle linee bianche che appaiono nell'output.Questo le lascia vedere più informazioni sullo schermo o finestra. Se si vuole più controllo sull'output, usare le funzioni sprintf e fprintf.

Eliminazione dell'output a video

Se si digita semplicemente uno statement e poi si digita RETURN o Enter, MATLAB automaticamente mostre i risultati sullo schermo.Se invece si termina la linea con un punto e virgola, MATLAB compie il calcolo ma non espone alcuno output. Questo è particolarmente utile quando si generano matrici grandi. Per esempio:

```
A = magic(100);
```

non mostra a video la matrice A.

Linee di Comando lunghe

Se un'asserzione non va bene su una linea,si usi l'operatore tre punti,..., seguito da Return o Enter per indicare che l'asserzione continua sulla prossima linea. Per esempio:

```
s = 1 -1/2 + 1/3 -1/4 + 1/5 - 1/6 + 1/7...
 - 1/8 + 1/9 - 1/10 + 1/11 - 1/12;
```

Spazi vuoti intorno ai seguenti operatori =, +, e - sono opzionali, ma migliorano la leggibilità.

Editing da Linea di comando

Le varie frecce e chiavi di controllo sulla tastiera permettono di richiamare, compilare, e correggere comandi digitati in precedenza. Per esempio, supponga di digitare per sbaglio la seguente riga:

```
rho = (1 + sqt(5)) /2
```

E' sbagliata l'ortografia di sqrt. MATLAB risponde con:

```
Undefined function or variable 'sqt'.
```

Invece di ribattere la linea intera, semplicemente pigi il tasto freccia su. Il comando errato è in questo modo richiamato e può essere corretto.Usi le frecce dx e sx per trasportare il cursore su t per inserire la r mancante . L'uso ripetuto della freccia su richiama le linee digitate in precedenza,digitando un carattere e poi il tasto freccia su trova una linea precedente che comincia con quel carattere. L'elenco dei comando di linea disponibile è diverso a seconda del computer. Sperimenti quale delle chiavi seguenti è disponibile sulla sua macchina. (Molte di queste chiavi saranno familiari agli utenti di EMACS.)

```
freccia su        oppure  ctrl-p  -Richiamano linea precedente

freccia giu       oppure  ctrl-n  -Richiamano linea seguente

freccia sx        oppure  ctrl-b  -Trasportano indietro di un carattere

freccia dx        oppure  ctrl-f  -Trasportano in avanti di un carattere

ctrl+freccia dx   oppure  ctrl-r  -Trasportano alla destra della parola

ctrl+freccia sx   oppure  ctrl-l  -Trasportano alla sinistra della parola

home              oppure  ctrl-a  -Porta all'inizio della linea

end               oppure  ctrl-e  -Porta alla fine della linea

esc               oppure  ctrl-u  -Ripulisce la linea

del               oppure  ctrl-d  -Cancellano il carattere sul cursore

backspace         oppure  ctrl-h  -Cancellano carattere dopo il cursore

                          ctrl-k  -Cancella fino alla fine della linea.
```

2.5 Operazioni sui file

• Salvare in un file

Vediamo come salvare dei dati su un archivio di testo.Si voglia salvare i risultati della funzione exp(x) entro l'intervallo(1; 2) con passo 0.1. Si costruisca dapprima il vettore seguente:

x = 1 : 0.1 : 2 ;

e quindi si valuta una matrice A nel modo seguente:

A = [x ;exp(x)];

Decidiamo che il nome esterno del file su cui si vuole salvare la matrice A sia file.txt ; esso dovra' essere posto entro apici perche' e' una stringa. Apriamo il file in scrittura con 'wt' assegnandogli il nome interno nomefile. L'istruzione e' la seguente:

nomefile = fopen ('file.txt' , 'wt');

Si deposita nell'archivio nomefile la matrice A scrivendo le due colonne di numeri decimali:

la prima colonna con 6 cifre di cui 2 decimali
la seconda colonna 8 cifre di cui 4 decimali.

Si osservi che fprintf e' un acronimo di file print formatted.

fprintf (nomefile , '%6.2g %8.4g\n' , A);

Si noti che per primo si mette il nome interno: nomefile per secondo si mette il formato per terzo si mette la matrice: A
Quindi si chiude il file con il comando:

fclose (nomefile);

Se si vuole analizzare il file cosi' creato lo si puo' fare con un editor di testo qualsiasi.

• Caricare un file

Vediamo come leggere dati da un archivio di testo.Il nome esterno del file sia "file.txt" che e' generato con un programma esterno.
Il nome deve essere posto entro apici perchè è una stringa. Per iniziare si apre il file assegnandogli un nome interno, ad esempio nomefile, in lettura ('r' che sta per "read").

nomefile = fopen ('file.txt' , 'r');

Si preleva dall'archivio nomefile la matrice A leggendo le due colonne di numeri in formato "%g".

A = fscanf (nomefile , '%g %g' , [2 inf]);
A = A';

Si noti che per primo si mette il nome interno: (nomefile) per secondo si mette il formato ('%g%g') per terzo si mette il numero di elementi per riga(2) e, non sapendo quante righe sono, si mette inf il tutto entro parentesi quadre per indicare la matrice da prelevare. Per ultimo si chiude il file:

fclose (nomefile);

Ricapitoliamo i comandi utilizzati:

aprire $\Longrightarrow$ fopen (...)
chiudere $\Longrightarrow$ fclose (...)
scrivere $\Longrightarrow$ fprintf (...)
leggere $\Longrightarrow$ fscanf (...)

2.6 Grafici

• Grafici

MATLAB ha estesioni facilitate per esporre vettori e matrici come grafici, così da poter annotare e stampare questi grafici. Questa sezione descrive alcune delle funzioni grafiche più importanti e contiene esempi di alcuni applicazioni tipiche.

• Creare un diagramma

La funzione PLOT ha forme diverse, dipendendo dagli argomenti di entrata. Se y è un vettore, plot(y) produce un grafico lineare degli elementi di y contro l'indice degli elementi di y. Se specifica due vettori come argomenti, plot(x,y) produce un grafico di y contro x. Per esempio, per diagrammare il valore della funzione seno da 0 a 2π, uso:

```
t = 0:pi/100:2*pi;

y = sin(t);

plot(t,y)
```

Ottenendo così il seguente tracciato:

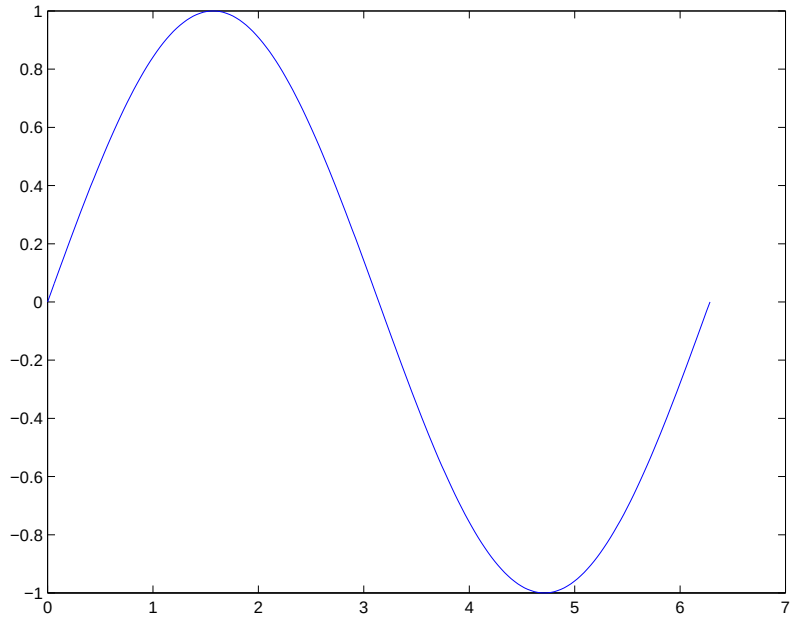

Figura 2.1: Funzione seno

Diverse coppie di x-y creano grafici multipli con una singola chiamata . MATLAB automaticamente traccia i diversi grafici attraverso un predefinito (ma settabile dall'utente) elenco di colori che permette

di distinguere ciascuna collezione di dati. Per esempio, queste asserzioni tracciano quattro funzioni di t, ciascuna curva in un colore diverso:

```
y2 =  sin(t - .40);

y3 =  sin(t - .8);

y4 =  sin(t - 1.2);

plot(t,y,t,y2,t,y3,t,y4)
```

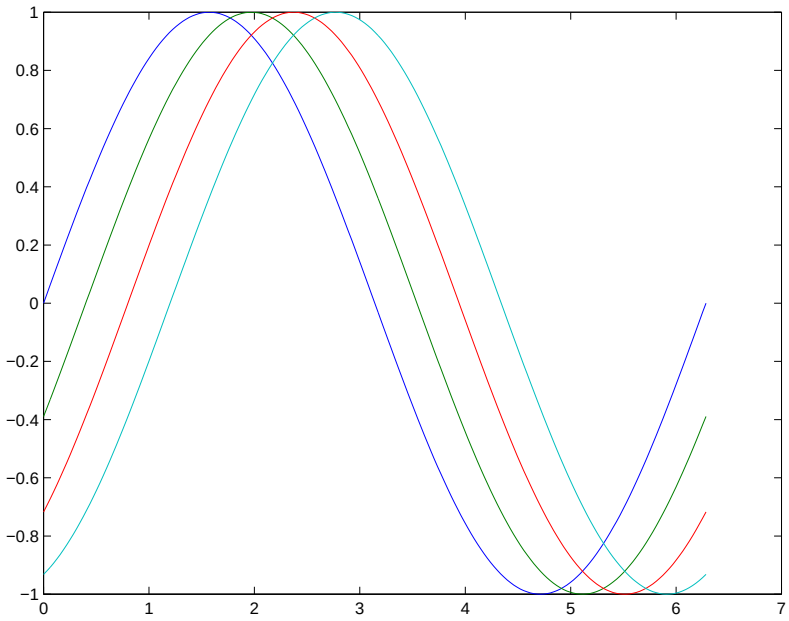

Figura 2.2: Grafici Multipli

È possibile specificare colore, stile della linea, e marcatori, con segnali positivi o cerchi, attraverso il seguente comando:

```
plot(x,y, 'marcatore-stile-colore')
```

dove il `marcatore-stile-colore` è costiutito da una sequenza di 1 , 2 , o 3 caratteri (separati da virgolette)che rappresentano un colore, uno stile di linea , ed un tipo di marcatore; tipi di colore sono:

'c', 'm', 'y', 'r', 'g', 'b', 'w', e 'k'

Questi corrispondono a cyan, magenta, giallo, rosso, verde, azzurro, bianco, e nero. Stringhe di Line-style sono:

'-' per la linea continua
'- -' per la linea tratteggiata

':' per una linea a puntini
'-.' per una linea a puntini e tratteggio
'none' senza linea.

Tipi di marcatore comuni sono:

'+', 'o', '*', 'x'

Per esempio, l'asserzione:

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

traccia in giallo una linea a puntini e pone un marcatore + a ciascun dato. Se specifica un tipo di marcatore ma non un linestyle, MATLAB disegna solamente il marcatore.

2.7 Operazioni sui grafici

• Finestre della figura

La funzione plot apre automaticamente una nuova finestra della figura se non c'è ne sono gia sullo schermo. Se una finestra della figura esiste, plot usa tale finestra di default. Per aprire una finestra nuova e renderla la finestra corrente digitare:

figure

Per trasformare una figura esistente in finestra corrente , digitare

figure(n)

dove n è il numero nella barra del titolo della figura.

• Aggiunta di un tracciato ad un Grafico Esistente

Il comando HOLD permette di aggiungere tracciati ad un grafico esistente. Quando si digita :

hold on

MATLAB non rimuove il grafico esistente; aggiunge i dati nuovi al grafico corrente, e riscala se necessario. Per esempio, questa asserzione prima crea un tracciato della funzione sin in colore blu e con il marcatore *, poi sovrappone un tracciato per la funzione cos in colore rosso e con il marcatore o:

```
t = 0:pi/100:2*pi;

y = sin(t);

plot(t,y,'*')

z=cos(t);

hold on

plot(t,z,'r:o')
```

Il comando hold on fa si che il primo tracciato sia combinato col secondo tracciato in figura .

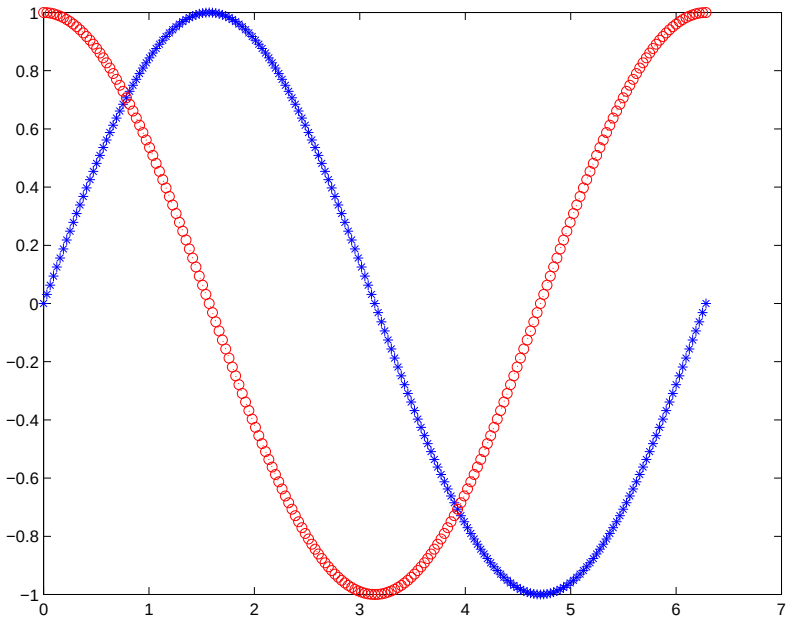

Figura 2.3: Marcatori

• Subplots

La funzione subplot permette di esporre tracciati multipli nella stessa finestra o li stampa sullo stesso pezzo di carta,digitando:

subplot(m,n,p)

trasforma la finestra della figura in un m-by-n matrice di piccoli subplots e seleziona il pth subplot come plot corrente.I tracciati (plot) sono numerati prima lungo la prima fila della finestra della figura, poi la seconda fila e così via. Per esempio, scomporre il tracciato di dati in quattro subregions diversi della finestra della figura, nell'esempio trattato si tracciano i grafici delle seguenti funzioni: sin, cos, tan, atan;

```
t = 0:pi/10:2*pi;
y = sin(t);
subplot(2,2,1)
plot(t,y)
y1=cos(t);
subplot(2,2,2)
plot(t,y1)
y2=tan(t);
subplot(2,2,3)
plot(t,y2)
y3=atan(t);
subplot(2,2,4)
plot(t,y3)
print -dbmp16m figura.bmp
```

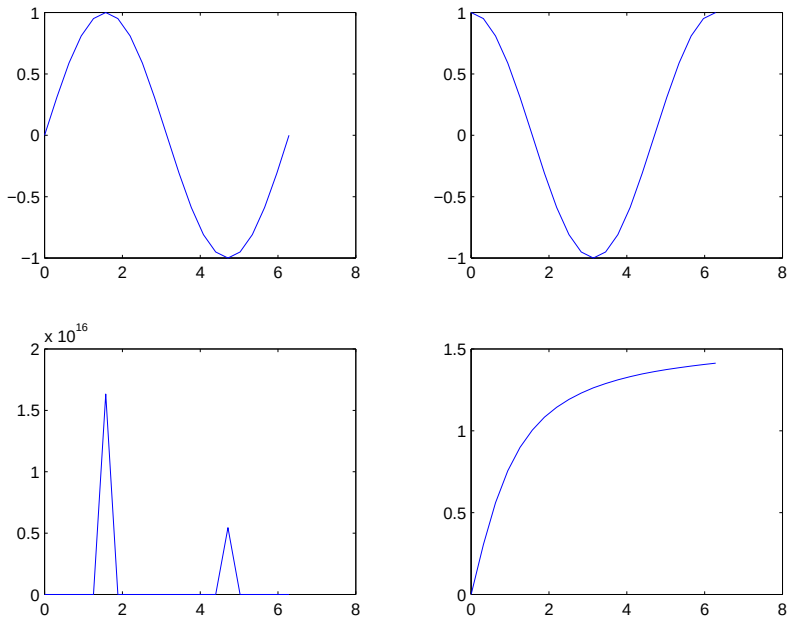

Figura 2.4: Subplot

L'ultima istruzione salva la figura nel file figura.bmp (24-bit .BMP file format) il quale potrà essere inserito in documenti html.

2.8 Dati immaginari e complessi

• Dati immaginari e Complessi

Quando i dati da plottare sono complessi,la parte immaginaria è ignorata eccetto quando il tracciato è dato da un singolo argomento complesso.Per questo caso speciale, il comando è shortcut per tracciare la parte reale contro la parte immaginaria. Perciò,

plot(Z)

dove Z è un vettore complesso o matrice, è equivalente a:

plot(real(Z),imag(Z))

Per esempio

```
t = 0:pi/8:2*pi;

plot(exp(i*t), ' -o ')
```

traccia un poligono di 16 lati con cerchietti ai vertici.

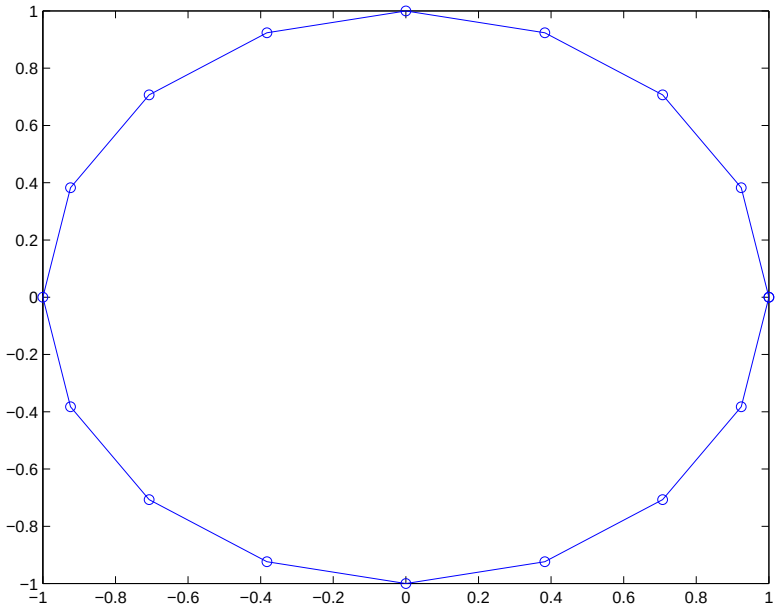

Figura 2.5: Dati Immaginari e complessi

2.9 Assi

• Assi

La funzione axis ha un numero di opzioni per personalizzare la misurazione in scala, l'orientamento, ed il rapporto d'aspetto dei tracciati. Ordinariamente,MATLAB trova i massimi e minimi dei dati e sceglie una plot-box adatta ed identifica gli assi con delle label. La funzione axis ha la priorità di default, per personalizzare i limiti degli assi si digita:

axis([xmin xmax ymin ymax])

axis accetta anche un numero di keywords per il controllo degli assi. Per esempio:

axis square

impone che i due assi abbiano la stessa lunghezza

axis equal

impone che gli incrementi per ogni marchio su x e y siano uguali. Così:

plot(exp(i*t))

seguito da axis square o axis equal, trasforma l'ovale in un corretto cerchio;

axis auto

restituisce l'asse in scala default, in maniera automatica.

• Titolo ed etichette degli assi

Le funzioni xlabel, ylabel, e zlabel aggiungono etichette agli assi x , y , z. La funzione title aggiunge un titolo in cima alla figura e la funzione text inserisce testo dovunque nella figura. Un sottoinsieme di notazione di Tex produce lettere greche, simboli matematici, e fonti alternate. L'esempio seguente usa `\leq per <=, \pi` per π, e `\it` per fonte corsiva.

```
t = -pi:pi/100:pi;

y = sin(t);
```

```
plot(t,y)

axis([-pi pi -1 1])

xlabel('-\pi \leq \itt \leq \pi')

ylabel('sin(t)')

title('Grafico della funzione sin')

text(1,-1/3,'\it{Funzione dispari}')
```

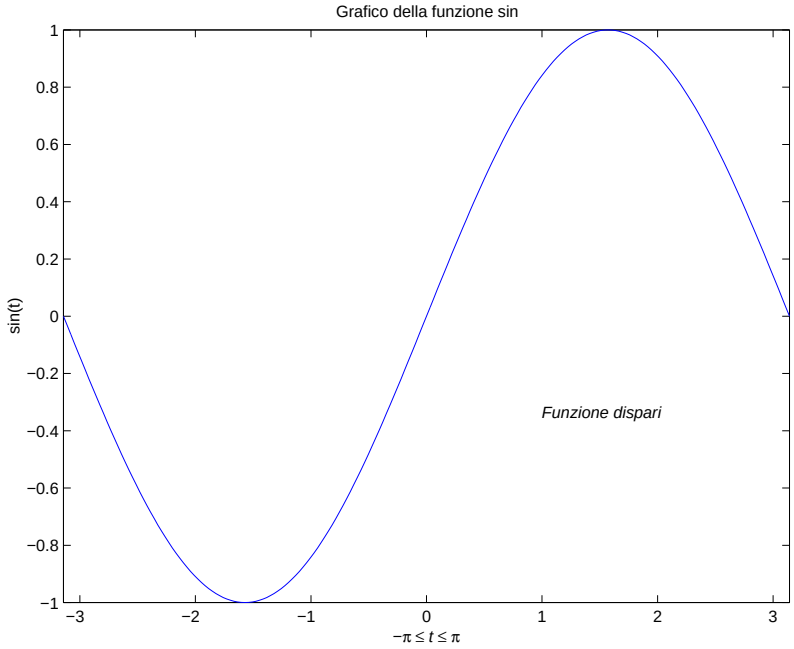

Figura 2.6: Assi,Titoli ed altro

2.10 Alfabeto greco e caratteri speciali

L'alfabeto greco viene generato dai comandi

α	\alpha	θ	\theta	o	o	τ	\tau
β	\beta	ϑ	\vartheta	π	\pi	υ	\upsilon
γ	\gamma	ι	\iota	ϖ	\varpi	ϕ	\phi
δ	\delta	κ	\kappa	ρ	\rho	φ	\varphi
ϵ	\epsilon	λ	\lambda	ϱ	\varrho	χ	\chi
ε	\varepsilon	μ	\mu	σ	\sigma	ψ	\psi
ζ	\zeta	ν	\nu	ς	\varsigma	ω	\omega
η	\eta	ξ	\xi				

Tabella 2.1: Lettere greche minuscole

mentre per le maiuscole si utilizzano i seguenti comandi:

Γ	\Gamma	Λ	\Lambda	Σ	\Sigma	Ψ	\Psi
Δ	\Delta	Ξ	\Xi	Υ	\Upsilon	Ω	\Omega
Θ	\Theta	Π	\Pi	Φ	\Phi		

Tabella 2.2: Lettere greche maiuscole

I caratteri speciali sono elencati in tabella 2.3.

$\pm$	\pm	$\mp$	\mp	$\times$	\times	$\circ$	\circ
$\cap$	\cap	$\cup$	\cup	∇	\nabla	$\|$	|
ℓ	\ell	$\Re$	\Re	$\Im$	\Im	∂	\partial
∞	\infty	$\exists$	\exists	$\forall$	\forall		
$\leq$	\le	$\subset$	\subset	$\subseteq$	\subseteq	$\geq$	\geq
$\ll$	\ll	$\supset$	\supset	$\supseteq$	\supseteq	$\gg$	\gg
$\in$	\in	$\equiv$	\equiv	$\sim$	\sim	$\approx$	\approx
$\neq$	\ne	$\propto$	\propto	$\div$	\div	$*$	\ast
$\rightarrow$	\to	$\mapsto$	\mapsto	$\Rightarrow$	\Rightarrow	$\Longrightarrow$	\Longrightarrow
$\sum$	\sum	$\int$	\int	$\prod$	\prod	$\oint$	\oint

Tabella 2.3: Simboli matematici

2.11 Immagini

• Mesh e superficie dei tracciati

MATLAB definisce una superficie dalle coordinate z dei punti su di una griglia nel piano x-y, usando linee diritte per connettere punti adiacenti.Le funzioni mesh e surf visualizzano superfici in tre dimensioni, mesh produce superfici wireframe colorando solamente le linee che connettono i punti definiti,Surf espone in colore, le linee che connettono e le facce della superficie.

• Funzioni di visualizzazione in due Variabili

Per esporre una funzione di due variabile, z = f (x,y), generare due matrici X e Y costituite di file e colonne ripetute , rispettivamente, sul dominio della funzione. Poi usare queste matrici per valutare e tracciare la funzione. La funzione meshgrid trasforma nel dominio specifico, un singolo vettore o due vettori x e y in matrici X e Y per usarle nella valutazione della funzione di due variabili. Le file di X sono copie del vettore x e le colonne di Y sono copie del vettore y. Per valutare la funzione bidimensionale sin(r)/r, lungo le due direzioni x e y si procede nel modo seguente:

```
[X,Y] = meshgrid(-8:.5:8);

R = sqrt(X.^2 + Y.^2) + eps;

Z = sin(R)./R;

mesh(X,Y,Z)
```

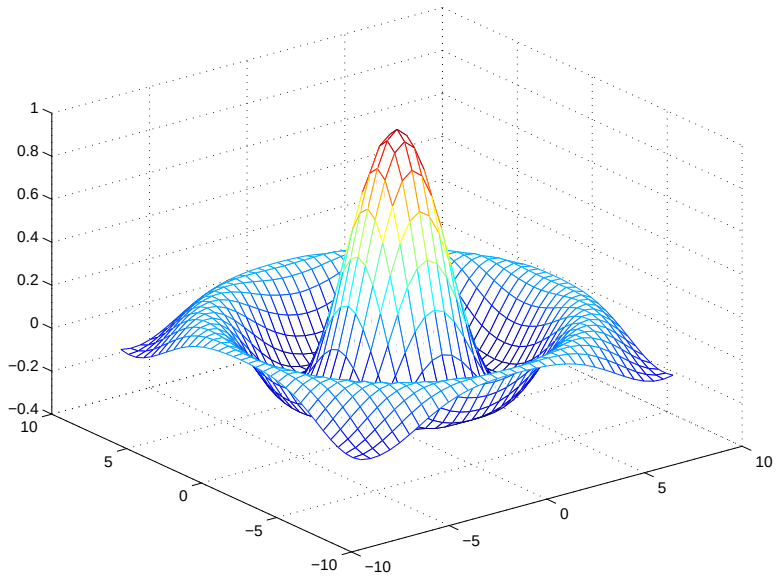

Figura 2.7: Funzione di due variabili

In questo esempio, R è la distanza dall' origine che è al centro della matrice. Aggiungere eps evita l'indeterminato 0/0 all'origine.

• Immagini

Array bidimensionali possono essere esposti come immagini, dove gli elementi dell'array determinano luminosità o colore delle immagini. Per esempio:

```
load durer
```

```
whos
```

mostra il file durer.mat contenuto nell'elenco delle demo costituito da una matrice 648-by-509, X, e una matrice 128-by-3, map. Gli elementi di X sono numeri interi tra 1 e 128, che servono come indice nella mappa colore, map. Quindi digitando:

```
image(X)
```

```
colormap(map)
```

```
axis image
```

si riproduce la corrosione di Dürer, un'immagine artistica di valore eccezionale. Un alto grado di risoluzione della scansione della magic square è disponibile in un altro file. Digitare:

```
load detail
```

e poi usare il tasto freccia su sulla tastiera per ripristinare l'immagine, colormap, e comandi dell'asse, cioè ripetere i seguenti comandi.

```
image(X)
```

```
colormap(map)
```

```
axis image
```

• Stampa dei grafici

L'opzione PRINT sul menu file ed il comando print stampa la figura creata da MATLAB. Il menu print apre una finestra di dialogo che offre le opzioni di stampa di default. Il comando print prevede più flessibilità nel tipo di output e permette di controllare la stampa da M-file. Il risultato può essere spedito direttamente alla stampante di default o può essere immagazzinato in un specifico file. Una varietà larga di configurazioni di output, incluso Poscript è disponibile. Per esempio, questa asserzione salva i contenuti della finestra della figura corrente come, color Encapsulated Level 2 PostScript nel file chiamato magicsquare.eps:

```
print  -depsc2  figura.eps
```

È importante sapere le capacità della stampante prima di usare il comando stampa. Per esempio, i file Level 2 Postscript generalmente sono più piccoli e sono più rapidi da stampare rispetto a Level 1 Postscript files. Comunque, non tutte le Stampanti postscript sostengono Level 2, è importante conoscere le potenzialità della propria stampante.

2.12 Grafici di matrici

Quando si invoca la funzione plot con argomento una singola matrice:

```
plot(Y)
```

Matlab traccia una curva per ogni colonna della matrice utilizzando come indice per l'asse x il numero di righe della matrice. Ad esempio:

```
Z=peaks;
```

fornisce una matrice 49*49 ottenuta valutando una funzione di due variabili. Il diagramma della matrice:

```
plot(Z)
```

produce un grafico di 49 linee, che di seguito riportiamo.

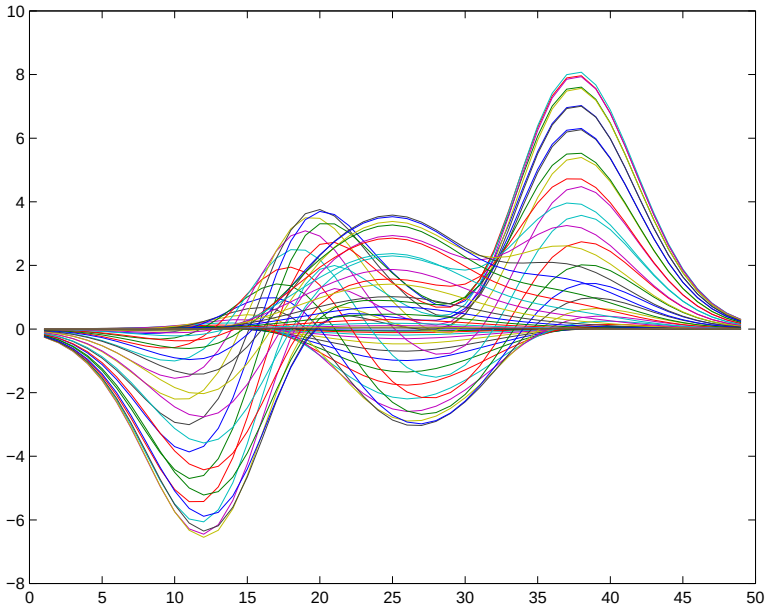

Figura 2.8: Grafici di matrici

Se il comando plot è usato con due argomenti e se X e Y hanno più di una riga o una colonna allora:

- Se Y è una matrice e X un vettore, plot(X,Y) traccia un diagramma delle righe o delle colonne di Y verso il vettore X, utilizzando colori o tipi di linea differenti per ogni curva.

- Se X è una matrice ed Y un vettore, plot(X,Y) traccia una curva per ogni riga o colonna di X rispetto ad Y.

Ad esempio, diagrammando la matrice peaks rispetto al vettore 1:length(peaks) si ottiene un diagramma ruotato rispetto a quello precedente:

```
Y=1:length(peaks);
plot(peaks,Y)
```

si ottiene:

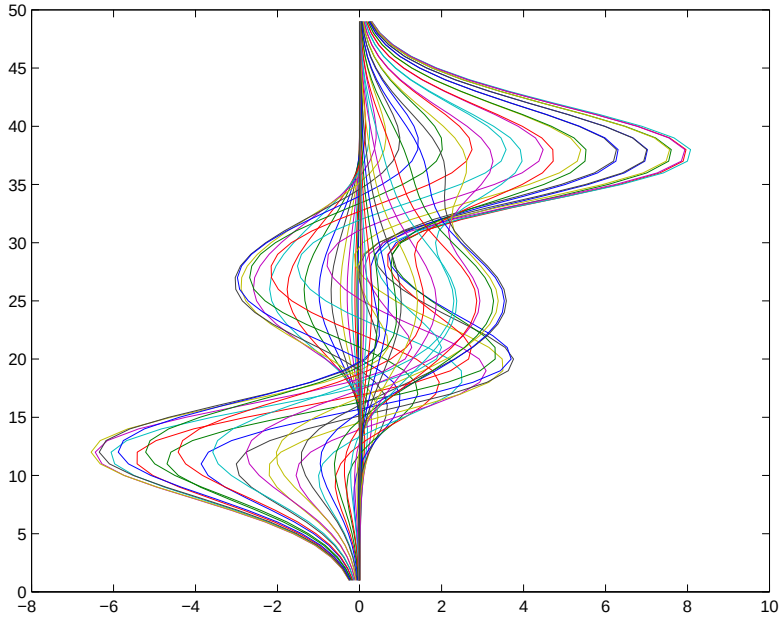

Figura 2.9: Grafici di matrici

Se infine X ed Y sono matrici delle stesse dimensioni allora plot(X,Y) fornisce un diagramma delle colonne di X rispetto alle colonne di Y.

2.13 Help e Documentazione Online

Ci sono molti modi diversi per accedere ad informazioni online sulle funzioni di MATLAB.

- Il comando HELP
- La finestra di aiuto
- Il MATLAB Help Desk
- Le pagine online
- Il Comando Help

Il comando Help è il modo più semplice per determinare la sintassi ed il comportamento di una particolare funzione. Informazioni sono esposte direttamente nella finestra di comando. Per esempio:

```
help magic
```

fornisce il seguente risultato:

```
MAGIC Magic square.
 MAGIC(N) is an N-by-N matrix constructed from
 the integers 1 through N^2 with equal row,
 column, and diagonal sums.
 Produces valid magic squares for N = 1,3,4,5....
```

Il MATLAB help usa caratteri maiuscoli per i nomi di funzioni e variabili per fare si che siano messe in rilievo dal resto del testo. Quando digitate il nome di una funzione ,comunque, utilizzate sempre i corrispondenti caratteri minuscoli perché MATLAB è case sensible e tutte le funzioni sono in lowercase.

Tutte le funzioni di MATLAB sono organizzate in gruppi logici, e la struttura dell'elenco di MATLAB è basata su questo raggruppamento. Per esempio, tutte le funzioni lineari algebriche risiedono nell'elenco matfun. Per elencare i nomi di tutti le funzioni in quell'elenco, con una breve descrizione digitare:

```
help matfun

Matrix functions - numerical linear algebra.

Matrix analysis. norm - Matrix or vector norm. normest - Estimate the matrix
2-norm ...
```

Il comando

```
help
```

fornisce tutti gli elenchi, con una descrizione della categoria della funzione rappresentata:

```
matlab/general matlab/ops ...
```

- La Finestra Help

In MATLAB è disponibile una finestra di aiuto, attivabile su PCs e Macs selezionando l'opzione Help Window nel menu Help,o cliccando sul punto interrogativo sulla sbarra del menu. È anche disponibile su tutti i computer digitando:

```
helpwin
```

Per usare la finestra d'aiuto su un particolare tema, digitare:

```
helpwin topic
```

La finestra d'aiuto accede alle stesse informazioni come il comando HELP, ma l'interfaccia della finestra prevede collegamenti agli altri temi.

• Il comando lookfor

Il comando lookfor permette di cercare funzioni attraverso una keyword. Ad esempio, MATLAB non ha una funzione chiamata `inverse`. Così la risposta alla prossima asserzione sarà:

```
help inverse

inverse.m not found.
```

Invece digitando:

```
lookfor inverse
```

cerca su una dozzina di matchs.Il risultato dipende da che toolboxes si è installato, si riceverà come risultato della ricerca dei link tipo:

```
INVHILB Inverse Hilbert matrix.

ACOSH Inverse hyperbolic cosine.

ERFINV Inverseof the error function.

INV Matrix inverse.

PINV Pseudoinverse.

IFFT Inverse discrete Fourier transform.

IFFT2 Two-dimensional inverse discrete Fourier transform.

ICCEPS Inverse complex cepstrum.

IDCT Inverse discrete cosine transform.
```

Aggiungendo -all al comando lookfor, come in:

```
lookfor -all
```

si ricercherà l'intero input,e non solo la prima linea .

• The Help Desk

L' Help Desk di MATLAB prevede l'accesso a una larga serie di aiuto e referenze immagazzinate su disco o CD-ROM. Molta della documentazione è posta sottoforma di HyperText (HTML) ed è accessibile attraverso un browser come Netscape o Microsoft Explorer. L'aiuto può essere richiesto su PCs e Macs selezionando il menu Help, o, su tutti i computer, digitando:

```
helpdesk
```

Tutti gli operatori e le funzioni di MATLAB hanno pagine di aiuto online in HTML. Queste pagine prevedono più dettagli ed esempi che il risultato del semplice comando help. Attraverso un motore di ricerca,si possono consultare tutti gli aiuti online .

• Il comando doc

Se si conosce il nome di una funzione specifica, si può vedere la sua pagina di referenza direttamente.Per ottenere la pagina di referenza per la funziona eval per esempio, digitare:

```
doc eval
```

Il comando doc apre il browser, se già non è aperto.

• Stampare le pagine del manuale

Versioni della documentazione online in inglese, così come il resto della documentazione MATLAB è anche disponibile in formato (PDF) attraverso l'hel desk.

• Link a MathWorks

Se il Suo computer è connesso ad Internet, la Scrivania d'Aiuto prevede un collegamento a MathWorks, la homepage di MATLAB. Lei può fare domande,suggerimenti, e riportare i possibili difetti. Lei può anche usare il Motore di Ricerca di MathWorks per consultare una base di dati .

2.14 L'Ambiente di MATLAB

L'ambiente di MATLAB conserva i valori delle variabili utilizzate durante una Sessione di MATLAB ed i file che contengono programmi e dati che vengono utilizzati tra sessioni diverse.

• Il Workspace

Il workspace è l'area di memoria accessibile dal prompt di MATLAB. I due comandi who e whos, mostrano i contenuti correnti del workspace. Il comando who dà un elenco corto, mentre whos dà anche taglia e informazioni del deposito. Di seguito è mostrata la produzione prodotta da whos su un workspace che contiene risultati di alcuni degli esempi in questo tutorial.

```
whos
Name Size Bytes Class
A 4x4 128 double array
D 5x3 120 double array
M 10x1 3816 cell array
S 1x3 442 struct array
h 1x11 22 char array
n 1x1 8 double array
s 1x5 10 char array
v 2x5 20 char array
Grand total is 471 elements using 4566 bytes.
```

Per cancellare tutta la variabile esistente dal workspace, utilizzare il comando clear.

• Il comando SAVE

Il comando SAVE conserva i contenuti del workspace in un file con estensione .mat che può essere letto col comando LOAD in una sessione successiva di MATLAB. Per esempio:

```
save sessione-odierna
```

salva il workspace intero nel file sessione-odierna.m. Se desidera può salvare solamente una certa variabile specificando il nome dopo il filename.Ordinariamente, la variabile è salvata in una configurazione binaria che può essere letta rapidamente (e accuratamente) da MATLAB. Se vuole accedere a questi file al di fuori di MATLAB, può specificare una configurazione alternativa.

```
-ascii Use 8-digit text format. -ascii -double Use 16-digit text format. -ascii
-double -tabs Delimit array elements with tabs. -v4 Create a file for MATLAB
version 4. -append Append data to an existing MAT-file.
```

Quando si salva il workspace in configurazione testo,si dovrebbe salvare una sola variabile alla volta. Se salva più di una variabile, MATLAB creerà il file di testo, ma non sarà capace di caricarlo facilmente in MATLAB.

• Il Percorso di Ricerca

MATLAB usa un percorso di ricerca,come un elenco ordinato di directory per determinare come eseguire le funzioni chiamate. Quando chiama una funzione standard, MATLAB esegue la prima funzione del M-file sul percorso che ha il nome specificato. Il comando:

```
path
```

mostra il percorso di ricerca su qualsiasi piattaforma. Su PCs e Macs, scelga Set Path dal menu FILE per vedere o cambiare il percorso.

• Manipolazione di file

I comandi dir , type, delete, e cd rappresentano una collezione di generici comandi di sistema per manipolare file. Questa tavola indica gli equivalenti di questi comandi negli altri sistemi operativi.

MATLAB	MS-DOS	UNIX	VAX/VMS
dir	dir	ls	dir
type	type	cat	type
delete	del o erase	rm	delete
cd	chdir	cd	set default

Per la maggior parte di questi comandi, può usare pathnames, wildcards, e drive-designators nel modo solito.

• Il Comando DIARY

Il comando diary crea un'agenda della sessione di MATLAB in un file del disco. Lei può vedere e compilare il file di testo con un qualsiasi word-processor . Per creare un file diary che contiene tutti i comandi digitare:

```
diary
```

Per salvare la sessione di MATLAB in un archivio con un particolare nome, uso:

```
diary filename
```

Per bloccare la registrazione della sessione, uso:

```
diary off
```

• Per lanciare Programmi Esterni

Il carattere punto di esclamazione! è una uscita ed indica che il resto della linea di input è un comando del sistema operativo . Questo è piuttosto utile per invocare utilities o per lanciare altri programmi senza lasciare MATLAB. Su VMS, per esempio:

```
!edt giuseppe.m
```

invoca un editor chiamato edt per un file chiamato giuseppe.m. Quando si esce dal programma esterno, il sistema operativo ritorna a MATLAB .

2.15 Ulteriori informazioni su Matrici ed Array

Questa seziona ci fornisce ulteriori informazioni sul modo di trattare matrici ed array , Focalizziamo ora l'attenzione sui seguenti argomenti:

1)Algebra Lineare

2)Arrays

3)Multivariate Data

- Algebra Lineare

Spesso i termini matrici ed array sono usati in modo interconnesso. Più precisamente, una matrice è un array numerico e due dimensioni che rappresenta una trasformazione lineare. Le operazioni matematiche definite sulle matrici sono il soggetto dell'algebra lineare.

La Dürer's magic square seguente:

```
A = 16 3 2 13
5 10 11 8
9 6 7 12
4 15 14 1
```

prevede molti esempi che danno un'idea delle operazioni sulle matrici in MATLAB. Lei già ha visto la matrice trasposta, A'. Aggiungendo una matrice alla sua trasposta si ottiene una matrice simmetrica.

```
A + A'
ans = 32    8   11   17
 8   20   17   23
11   17   14   26
17   23   26    2.
```

Il simbolo della moltiplicazione, *, denota la moltiplicazione fra matrici come prodotto righe per colonne. Moltiplicando una matrice alla sua trasposta si ottiene allo stesso modo una matrice simmetrica.

```
A'*A
ans = 378 212 206 360
212 370 368 206
206 368 370 212
360 206 212 378
```

Il determinante di questa particolare matrice risulta essere zero, indicando che la matrice è singolare.

```
d = det(A)
d = 0
```

Ponendo:

```
R = rref(A)
R =
1 0 0 1
0 1 0 -3
0 0 1 3
0 0 0 0
```

quando la matrice è singolare, non ha un inversa. Se si tenta di calcolare l'inversa con:

```
X = inv(A)
```

si otterrà un segnale di avvertimento del tipo:

```
Warning: Matrix is close to singular or badly scaled.
              Results may be inaccurate. RCOND = 1.175530e-017.
```

L'errore di Roundoff ha prevenuto l'algoritmo di inversione di matrice dallo scoprire la singolarità esatta. Gli autovalori della magic square sono interessanti.

```
e = eig(A)
e = 34.0000
 8.0000
 0.0000
-8.0000
```

Uno degli autovalori è zero che è un altra conseguenza della singolarità. Il più grande autovalore è 34, cioè pari alla somma magica. Questo perché il vettore unitario è un autovettore.

```
v = ones(4,1)
v = 1
1
1
1

A*v
ans =
34
34
34
34
```

Quando una magic square è scalata della sua somma magica,

```
 P = A/34
```

il risultato è una matrice le cui somme delle righe e delle colonne sono tutte uno.

```
P =
0.4706 0.0882 0.0588 0.3824
0.1471 0.2941 0.3235 0.2353
0.2647 0.1765 0.2059 0.3529
0.1176 0.4412 0.4118 0.0294.
```

Tali matrici rappresentano le probabilità della transizione in un processo di Markov. Potenze ripetute della matrice rappresentano passi ripetuti del processo. Ad esempio, la quinta potenza:

```
 P^5
```

è

```
0.2507 0.2495 0.2494 0.2504
0.2497 0.2501 0.2502 0.2500
0.2500 0.2498 0.2499 0.2503
0.2496 0.2506 0.2505 0.2493
```

Questo mostra che come k tende ad infinito, tutti gli elementi della kth potenza, P^k, tendono ad 1/4. Infine, i coefficienti nel polinomio della caratteristica:

```
poly(A)
```

sono:

```
1 -34 -64 2176 0
```

Questo indica che il polinomio della caratteristica

$$det(A - \lambda I) \tag{2.1}$$

è

$$\lambda^4 - 34\lambda^3 - 64\lambda^2 + 2176\lambda \tag{2.2}$$

Il termine costante è zero, perché la matrice è singolare, e il coefficiente del termine cubico è -34, perché la matrice è magica!

• ARRAY

Al di fuori del mondo dell'algebra lineare, le matrici possono essere viste come array bidimensionali. Le operazioni di aritmetica sugli arrays sono effettuate elemento per elemento. Questo vuole dire che somma e sottrazione si equivalgono per arrays e matrici ma le operazioni di moltiplicazione sono diverse. MATLAB usa un punto, o punto decimale, come parte della notazione per operazioni di moltiplicazione degli arrays.

La lista degli operatori è la seguente:

+	Somma
-	Sottrazione
*	Moltiplicazione elemento per elemento
/	Divisione elemento per elemento
\	Divisione sinistra elemento per elemento
^	Potenza elemento per elemento
'	Transposta

Se la Dürer magic square è moltiplicata per se stessa attraverso la moltiplicazione di arrays

```
A.*A
```

il risultato è un array che contiene il quadrato dei numeri interi da 1 a 16, in un ordine insolito.

```
ans = 256 9 4 169
25 100 121 64
81 36 49 144
16 225 196 1
```

Le operazioni sugli array sono utili per costruire tavole.
Definiamo n come il vettore colonna seguente:

```
n = (0:9)';
```

Poi

```
pows = [n n.^2 2.^n]
```

forma una tabella di quadrati e potenze di due.

```
pows = 0  0   1 1  1   2 2  4   4 3  9   8 4 16  16 5 25  32 6 36  64 7 49 128
8 64 256 9 81 512
```

Le funzioni matematiche elementari operano su arrays elemento per elemento.
Così risulta:

```
format short g x = (1:0.1:2)'; logs = [x log10(x)]
```

forma una tavola di logaritmi.

```
logs = 1.0 0 1.1 0.04139 1.2 0.07918 1.3 0.11394 1.4 0.14613 1.5 0.17609 1.6
0.20412 1.7 0.23045 1.8 0.25527 1.9 0.27875 2.0 0.30103
```

- Multivariate Data

MATLAB usa l'analisi orientata sulle colonne per dati statistici multivariate. Ogni colonna in una collezione di dati rappresenta una variabile e ciascuna fila un'osservazione. L'elemento (i,j)-esimo è la ith osservazione della jth variabile.
Come esempio, si consideri una collezione di dati con tre variabili:

- Pulsazioni
- Peso
- Ore di esercizi per settimana

Per cinque osservazioni, l'array che ne risulta è il seguente:

```
D = 72 134 3.2 81 201 3.5 69 156 7.1 82 148 2.4 75 170 1.2
```

La prima fila contiene le pulsazioni ,il peso ,le ore di esercizi per settimana per il paziente 1, la seconda fila contiene i dati per il paziente 2, e così via. Ora si possono applicare molte delle funzioni di analisi dei dati di MATLAB a questa collezione di dati. Per esempio, per ottenere la deviazione standard di ciascuna colonna:

```
mu = mean(D),sigma = std(D)
mu =
   75.8 161.8 3.48
sigma =
      5.6303 25.499 2.210
```

Per un elenco delle funzioni per l'analisi di dati, disponibili in MATLAB, digitare:

```
help datafun
```

Se si ha accesso allo Statistics Toolbox, si digiti:

```
help stats
```

2.16 Strutture di controllo del flusso

MATLAB ha cinque strutture di controllo di flusso:

- if statements
- switch statements
- for loops
- while loops
- break statements

Analizzamole singolarmente in modo dettagliato.

- if statement

La struttura if valuta un'espressione logica ed esegue un gruppo di asserzioni quando l'espressione è vera. L'else if opzionale e altre keywords provvedono per l'esecuzione di gruppi alternati di asserzioni. Una keyword END termina l'ultimo gruppo di asserzioni. I gruppi di asserzioni sono delineati da quattro keywords non sono previste parentesi. L'algoritmo di MATLAB per generare una magic square di ordine n coinvolge tre casi diversi: quando n è dispari, quando n è pari ma non divisibile per 4, o quando n è divisibile per 4. Questo è descritto dal costrutto seguente:

```
if rem(n,2) ~= 0
   M = odd_magic(n)
elseif rem(n,4) ~= 0
   M = single_even_magic(n)
else
   M = double_even_magic(n)
end
```

In questo esempio, i tre casi sono mutuamente esclusivi, ma se loro non lo fossero, la prima condizione vera sarebbe eseguita. Tutto ciò è importante per capire come gli operatori relazionali e le strutture if lavorano con le matrici. Quando si vuole controllare l'uguaglianza tra due variabile, usare:

```
if A == B, ....
```

Questo è legale in MATLAB, e fa quello che ci si aspetta quando A e B sono scalari. Ma quando A e B sono matrici, A == B non esamina se loro sono uguali, esamina solo dove loro sono uguali; il risultato è un altra matrice di 0 e di 1 che espone l'uguaglianza elemento per elemento. Infatti, se A e B non sono della stessa taglia, allora A == B è un errore. Il modo corretto per controllare l'uguaglianza tra due variabile è quello di usare la funzione isequal:

```
if isequal(A,B), ...
```

Qui c'è un altro esempio per enfatizzare questo punto. Se A e B sono scalari, il programma seguente mai arriverà alla situazione inaspettata. Ma per la maggior parte delle matrici, includendo le nostre magic square con colonne scambiate nessuna delle condizioni seguenti `A> B, A< B o A == B` è vero per tutti gli elementi e così l'altra clausola è eseguita.

```
if A > B
   'greater'
elseif A < B
```

```
   'less'
elseif A == B
   'equal'
else
   error('Unexpected situation')
end
```

Molte funzioni sono utili per ridurre i risultati di paragoni tra matrici e condizioni scalari attraverso l'uso dell'IF, includendo le seguenti:

```
isequal
isempty
all
any
```

• switch e case

L'asserzione switch esegue gruppi di asserzioni basati sul valore di un variabile o espressione. La keywords case e otherwise delinea i gruppi. Solamente il primo caso è eseguito. Ci deve essere sempre una fine per lo statement switch .

La logica dell'algoritmo delle magic square può essere descritta anche con:

```
switch (rem(n,4)==0) + (rem(n,2)==0)
   case 0
      M = odd_magic(n)
   case 1
      M = single_even_magic(n)
   case 2
      M = double_even_magic(n)
   otherwise
      error('This is impossible')
end
```

Diversamente dal linguaggio C, lo switch di MATLAB non sbaglia. Se la prima asserzione del case è vera, il restante case statement non è eseguito. Così, break statement non sono richiesti.

• for statement

Il loop for ripete un gruppo di asserzioni un numero fisso di volte . Un END statement termina le asserzioni.

```
for n = 3:32
   r(n) = rank(magic(n));
end
r
```

Il punto e virgola che termina l'asserzione sopprime la stampa ripetuta, e il termine r dopo il loop espone il risultato finale. È una buona idea per ordinare i loop al fine di una discreta leggibilità, identare il testo ad esempio:

```
for i = 1:m
   for j = 1:n
       H(i,j) = 1/(i+j);
```

```
   end
end
```

• while statement

Il ciclo while ripete un gruppo di asserzioni un numero indefinito di volte attraverso il controllo di una condizione logica. Un END delinea le asserzioni. Di seguito è riportato un programma completo, illustrando while, if, else, e end, il quale programma utilizza il metodo della bisezione per trovare uno zero di un polinomio.

```
a = 0; fa = -Inf;
b = 3; fb = Inf;
while b-a > eps*b
    x = (a+b)/2;
    fx = x^3-2*x-5;
    if sign(fx) == sign(fa)
       a = x; fa = fx;
    else
       b = x; fb = fx;
    end
end
x
```

Il risultato è la radice del polinomio $x^3 - 2x - 5$, vale a dire:

```
x = 2.09455148154233
```

• break statemant

L'asserzione break lascia che si esca velocemente da un loop FOR o WHILE.In un ciclo innestato,break esce dal ciclo interno solamente .Qui c'è un miglioramento sull'esempio fornito nella sezione precedente.

```
a = 0; fa = -Inf;
b = 3; fb = Inf;
while b-a > eps*b
   x = (a+b)/2;
   fx = x^3-2*x-5;
   if fx == 0
      break
   elseif sign(fx) == sign(fa)
      a = x; fa = fx;
   else
      b = x; fb = fx;
   end
end
x
```

2.17 Scripts e Funzioni

MATLAB è un linguaggio di programmazione potente, così come un ambiente computazionale interattivo.I Files che contengono codice MATLAB sono chiamati M-files. Dopo aver creato un M-file usando un qualsiasi editor di testo, tale file può essere usato come un comando od una funzione MATLAB. Ci sono due generi di M-file:

• Scripts che non accettano argomenti d'entrata o argomenti di uscita. Loro operano su dati nel workspace.

• Functions che possono accettare argomenti d'entrata e argomenti di uscita. Una variabile interna è locale alla funzione.

Se sei un nuovo programmatore di MATLAB,crea gli M-file di prova e posizionali nella directory corrente. In seguito gli M-file,potranno essere organizzati in altre directory e toolboxes personali che potrai aggiungere al percorso di ricerca di MATLAB. Se si duplicano nomi di funzioni, MATLAB esegue quello che viene prima nel percorso di ricerca. Per vedere i contenuti di un M-file, per esempio myfunction.m, posso digitare:

```
myfunction
```

• Script

Quando si richiama uno script, MATLAB esegue semplicemente i comandi presenti nel file.Gli Script possono operare su dati esistenti nel workspace, o loro possono creare dati nuovi su cui operare. Sebbene gli script non forniscano dati di output, qualsiasi variabile che loro creano rimane nel workspace, per essere usata in calcoli susseguenti. Inoltre, gli script possono produrre dei grafici, usando funzioni come plot.

Per esempio, creiamo un file chiamato rango.m che contiene questi comandi MATLAB:

```
% Determinazione del rango della magic squares
r = zeros(1,32);
for n = 3:32
   r(n) = rank(magic(n));
end
r
bar(r)
```

Digitando l'asserzione:

```
rango
```

imponiamo a MATLAB di eseguire i seguenti comandi:

1)calcola il rango delle prime 30 magic squares
2)traccia il grafico del risultato

Si ottiene così il grafico seguente:

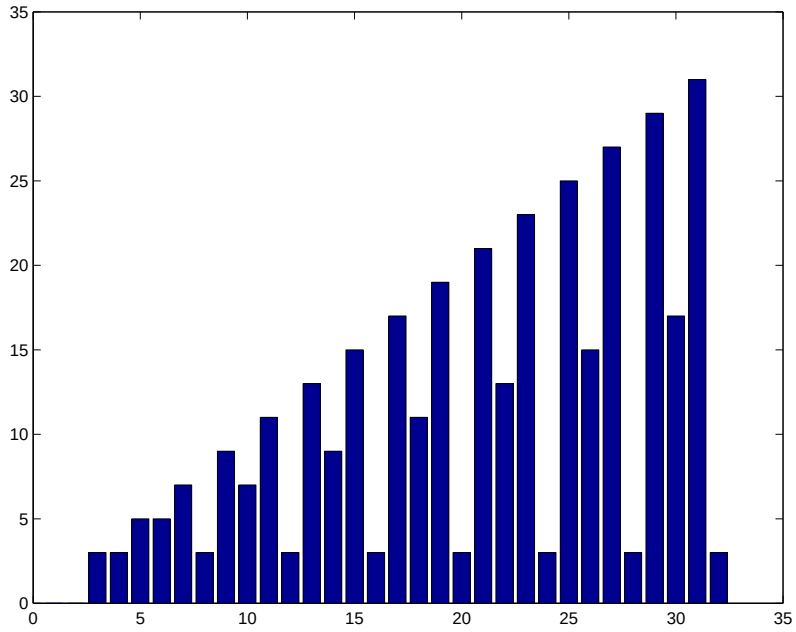

Figura 2.10: Rango delle magic squares

Dopo l'esecuzione del file , le variabili n e r rimangono nel workspace.

- Funzioni

Le Funzioni sono M-file che possono accettare argomenti d'entrata e forniscono argomenti di uscita. Il nome dell'M-file e della funzione deve essere lo stesso. Le Funzioni operano su variabili definite nel workspace proprio, separato dal workspace a cui si accede all'ingresso di MATLAB, cioè le variabili usate all'interno della funzione sono locali.

Vediamo come scrivere una una funzione con parametri in ingresso e parametri in uscita. Si voglia costruire una funzione che, dati i lati "a" e "b" di un rettangolo fornisca l'area "A", il perimetro "p" e la diagonale "d". Indichiamo con (a; b) la lista d'ingresso (parentesi tonde) e con [A; p; d] la lista d'uscita (parentesi quadre) (si noti la virgola fra i parametri).

```
function [ A , p , d ] =rettang ( a , b )
A = a * b;
p = 2 * ( a + b );
d =sqrt ( a^2 + b^2 );
```

Si noti che non c'e' confusione tra la lettera A e la a perche' MATLAB distingue le lettere maiuscole dalle minuscole. Questa function, salvata con il nome rettang.m puo' essere richiamata da un altro modulo o direttamente dalla finestra comandi ad esempio con il comando:

```
[area; perim; diag] =rettang(2; 3)
```

Quindi la sintassi e' la seguente:

```
function [lista d'uscita] = nome (lista d'ingresso)
```

Un buono esempio è fornito dalla funzione rank. L'M-file rank.m è disponibile nella directory toolbox/matlab/matfun

Si può lanciare il file digitando:

```
rank
```

Il file in questione è il seguente.

```
function r = rank(A,tol)
% RANK Matrix rank.
% RANK(A) provides an estimate of the number of linearly
% independent rows or columns of a matrix A.
% RANK(A,tol) is the number of singular values of A
% that are larger than tol.
% RANK(A) uses the default tol = max(size(A)) * norm(A) * eps.

s = svd(A);
if nargin==1
   tol = max(size(A)) * max(s) * eps;
end
r = sum(s > tol);
```

La prima linea di una funzione M-file comincia con la funzione keyword.La quale da il nome alla funzione ed ordina gli argomenti. In questo caso,ci sono due argomenti di input ed uno di output. Le righe seguenti,che iniziano con il simbolo %, rappresentano dei commenti per un aiuto, non vengono considerate nell'applicazione. Queste linee si stampano quando si digita:

```
help rank
```

La prima linea del testo di aiuto è la H1 line che MATLAB espone quando si ricerca aiuto digitando help on nella directory che contiene la funzione.
Il resto del file rappresenta codice eseguibile MATLAB definendo la funzione. La variabile s presente nel corpo della funzione, così come le variabili sulla prima fila r, A e tol, sono del tutto locali alla funzione; sono indipendenti e separate da qualsiasi variabile nel workspace di MATLAB. Questo esempio illustra un aspetto delle funzioni di MATLAB, che ordinariamente non si trova negli altri linguaggi di programmazione , un numero variabile di argomenti. La funzione rank può essere usata in molti modi diversi:

```
rank(A)
r = rank(A)
r = rank(A,1.e-6)
```

Molti M-file lavorano così. Se nessuno argomento di output è fornito, il risultato è immagazzinato in ans. Se il secondo argomento di input non è fornito, la funzione calcola un valore di default. Fra il corpo della funzione, due quantità chiamate nargin e nargout sono disponibili,le quali ci dicono il numero di argomenti di input e di output coinvolti in ciascun uso della funzione. La funzione rank usa nargin, ma non ha bisogno di usare nargout.

- Variabili globali

Se si vuole che più di una funzione condivida una singola copia di una variabile, basta semplicemente dichiarare la variabile come globale in tutte le funzioni. Si faccia la stessa cosa alla linea di comando

se si desidera che ad essa acceda anche il workspace. La dichiarazione globale deve essere posizionata prima che la variabile sia usata in una funzione. Sebbene non sia richiesto, usando lettere maiuscole per i nomi di variabili globali ci si aiuta a distingurle dalle altre variabili. Per esempio, creiamo un M-file chiamato falling.m:

```
function h = falling(t)
global GRAVITY
h = 1/2*GRAVITY*t.^2;
```

Poi interattivamente introduciamo le asserzioni:

```
global GRAVITY GRAVITY = 32;
y = falling((0:.1:5)');
```

Le due asserzioni globali danno il valore assegnato a GRAVITY al prompt di comando nella funzione. Si può cambiare poi GRAVITY interattivamente per ottenere soluzioni nuove senza compilare alcun file.

- Dualità Command/Function

Comandi di MATLAB sono :

```
load help
```

Molti comandi accettano modificatori che specificano operandi.

```
load giuseppe.dat
help magic
type rank
```

Un metodo alternato di approvvigionare i modificatori di comando fornisce una sequenza di argomenti di funzioni.

```
load('giuseppe.dat')
help('magic')
type('rank')
```

Questa è la "dualità tra command/function" di MATLAB. Qualsiasi comando della forma:

```
command argument
```

può essere scritto anche nella forma funzionale:

```
command('argument')
```

Il vantaggio dell'approccio funzionale viene quando l'argomento della sequenza è costruito dagli altri pezzi. L'esempio seguente processa files multipli, August1.dat, August2.dat e così via.In esso si usa la funzione int2str, che converte un numero intero in una sequenza di caratteri, per aiutare alla costruzione del nome del file.

```
for d = 1:31 s = ['August' int2str(n) '.dat'] load(s)
% Process the contents of the d-th file
end
```

• La funzione eval

La funzione eval lavora con variabili di testo per rendere più agevole l'uso delle macro. L'espressione o asserzione:

```
eval(s)
```

usa l'interprete di MATLAB per valutare l'espressione o eseguire l'asserzione contenuta nella sequenza del testo s. L'esempio della sezione precedente potrebbe essere fatto anche col codice seguente, sebbene questo risulta meno efficiente perché coinvolge il pieno interprete, non solo una chiamata alla funzione.

```
for d = 1:31
   s = ['load giuseppe' int2char(n) '.dat']
   eval(s)
   % Process the contents of the d-th file
end
```

• Vettorizzazione

È importante una vettorizzazione per ottenere la più alta velocità di MATLAB, nell'eseguire gli algoritmi contenuti negli M-file. Dove gli altri linguaggi di programmazione userebbero cicli for o do , MATLAB può usare operazioni su vettori o matrici. Un semplice esempio si ottiene creando una tavola di logaritmi.

```
x = 0;
for k = 1:1001
  y(k) = log10(x);
  x = x + .01;
end
```

Agli utenti di MATLAB esperti piace dire "La Vita è troppo corta per spenderla scrivendo dei cicli." Una versione vettorizzata dello stesso codice è:

```
x = 0:.01:10;
y = log10(x);
```

Per codici più complicati, le opzioni di vettorizzazione non sono sempre così ovvie. Quando la velocità è importante, si dovrebbe sempre cercare di vettorizzare i propri algoritmi.

• Preallocation

Se non si può vettorizzare un pezzo di codice, si può fare velocizzare un ciclo FOR preallocando qualsiasi vettore o array nei quali sono immagazzinati i risultati di output. Per esempio, questo codice usa la funzione zeros per preallocare il vettore creato nel ciclo FOR. Questo fa sì che il ciclo sia eseguito più velocemente.

```
r = zeros(32,1);
for n = 1:32
   r(n) = rank(magic(n));
end
```

Senza la preallocation nell'esempio precedente, l'interprete di MATLAB allarga il vettore r di un elemento ad ogni ciclo. La vector preallocation elimina questo passo ed il risultato è una esecuzione più veloce.

- Funzioni di funzione

Una classe di funzioni, chiamate "funzioni della funzione," lavora con funzioni nonlineari di una variabile scalare . Ovvero, una funzione lavora su un altra funzione. Le funzioni della funzione includono:

- Ricerche di zeri
- Ottimizzazioni
- Quadrature
- Equazioni differenziali ordinarie

MATLAB rappresenta le funzioni nonlineari attraverso una M-file funzione. Per esempio, qui c'è una versione semplificata della funzione humps dalla directory matlab/demos:

```
function y = humps(x)
y = 1./((x-.3).^2 + .01) + 1./((x-.9).^2 + .04) - 6;
```

Valutare questa funzione per un set di valori nell'intervallo seguente:

```
x = 0:.002:1;
y = humps(x);
```

Poi tracciare il grafico della funzione con:

```
plot(x,y)
```

si ottiene il seguente grafico:

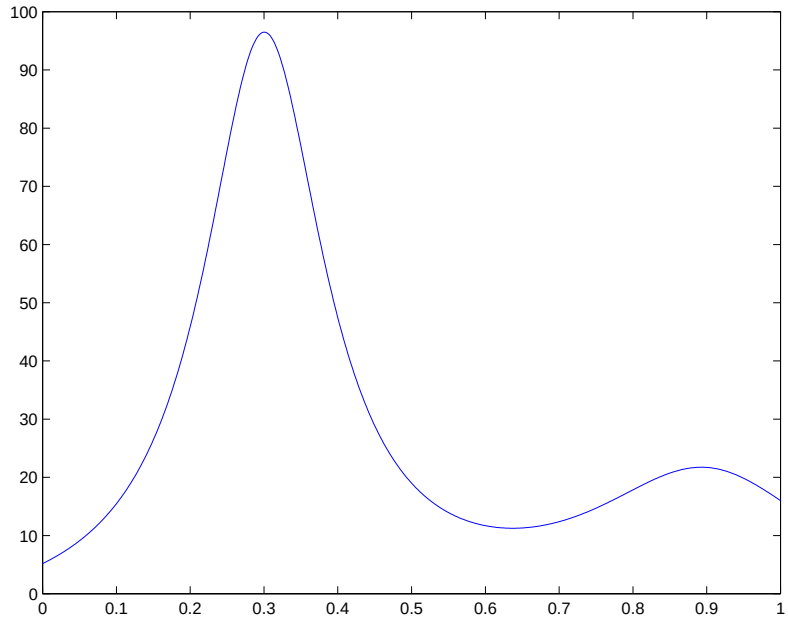

Figura 2.11: Funzione di funzione

Il grafico mostra che la funzione ha un minimo locale vicino ad x= 0.6. La funzione fmins trova il minimo valore di x . Il primo argomento di fmins è il nome del funzione di cui deve essere trovato il minimo, il secondo argomento è una supposizione grezza dell'ubicazione del minimo.

```
p = fmins('humps',.5)
p =
  0.6370
```

Per valutare la funzione in corrispondenza del minimo digitare:

```
humps(p)
ans = 11.2528
```

Analisti numerici usano i termini quadratura ed integrazione per distinguere tra approssimazione numerica di un integrale definito ed integrazione numerica di equazioni differenziali. Le routine di quadratura di MATLAB sono quad e quad8. L'asserzione:

```
Q = quad8('humps',0,1)
```

calcola l'area sotto la curva nel grafico e fornisce

```
Q = 29.8583
```

Finalmente, il grafico mostra che la funzione non èmai zero su questo intervallo. Così, se si cerca uno zero con

```
z = fzero('humps',.5)
```

lo si troverà fuori dell'intervallo

```
z = -0.1316.
```

Capitolo 3

Calcolo numerico

3.1 Polinomi

In Matlab un polinomio è rappresentato da un vettore. Per creare un polinomio è sufficiente immettere i coefficiente del polinomio in un vettore in ordine decrescente. Per esempio, consideriamo il seguente polinomio:

$$x^5 + 10x^4 + 0.6x^3 + 12x^2 - 7x + 5 \tag{3.1}$$

Per inserirlo nel Matlab, è sufficiente registrarlo come un vettore nel modo seguente:

```
x = [1 10 0.6 12 -7 5]

     x =
          1  10  0.6  12  -7  5
```

Il Matlab può interpretare un vettore di lunghezza n+1 come un polinomio di ordine n. Per i coefficienti non presenti nel polinomio si deve inserire lo zero nella corrispondente posizione del vettore. Per esempio,

$$x^5 - 7x + 5 \tag{3.2}$$

in Matlab deve essere rappresentato come:

```
y = [1 0 0 0 -7 5]
```

Si può calcolare il valore del polinomio usando la funzione polyval. Per esempio, per trovare il valore del precedente polinomio per s=2,

```
z = polyval([1 0 0 0 -7 5],2)

     z =
          23
```

E' anche possibile estrarre le radici di un polinomio. Ciò è utile quando si ha un ordine elevato del polinomio, come

$$x^5 + 10x^4 + 0.6x^3 + 12x^2 - 7x + 5 \tag{3.3}$$

Per trovare le radici basta semplicemente inserire il seguente comando:

```
roots([1 10 0.6 12 -7 5])


     ans =
        -10.0662
     -0.3251 + 1.1240i
     -0.3251 - 1.1240i
      0.3582 + 0.4843i
      0.3582 - 0.4843i
```

La moltiplicazione tra due polinomi è data dalla convoluzione dei loro coefficienti. La funzione usata dal Matlab è conv.

```
x = [1 2]; y = [1 4 8]; z = conv(x,y)

     z =
          1   6   16   16
```

La divisione di due polinomi viene realizzata con la funzione deconv che restituisce il resto e il risultato. Dividere z per y equivale ad ottenere x, secondo quanto fatto sopra:

```
[xx, R] = deconv(z,y)

     xx =
          1   2

     R =
          0   0   0   0
```

Come si può vedere, questo è il polinomio x cercato. Se z non fosse un multiplo di y, il vettore del resto avrebbe alcuni elementi diversi da zero.

Per sommare due polinomi dello stesso ordine, come già visto, basta eseguire z=x+y (i due vettori devono avere la stessa lunghezza). Nel caso generale,e cioè quando i due polinomi sono di grado diverso, puoi usare la funzione da definire, polyadd.

```
function[polinom]=polyadd(polin1,polin2)
%La funzione polyadd(polin1,polin2) somma due polinomi con dimensioni diverse
if length(polin1) < length(polin2)
  corto=polin1;
  lungo=polin2;
else
  corto=polin2;
  lungo=polin1;
end diff=length(lungo)-length(corto); if diff > 0
  polinom=[zeros(1,diff),corto]+lungo;
else
  polinom=lungo+corto;
end
```

Per usare polyadd, copia la funzione in un m-file, così da poterla utilizzare come una qualsiasi funzione del toolbox del Matlab. Assumendo che la funzione sia stata creata, il comando è copletato aggiungendo i polinomi x e y (osserva che in questo esempio i polinomi hanno grado diverso):

```
z = polyadd(x,y)

     x =
          1   2
```

```
y =
     1  4  8

z =
     1  5  10
```

3.2 Risoluzione di sistemi di equazioni lineari

In questa lezione impareremo a risolvere un sistema di equazioni lineari con matlab.Supponiamo di integrare un sistema di $N \times N$ equazione e cioè un sistema di N equazione in N incognite, ed indichiamo con A la matrice dei coefficienti delle incognite e con B in vettore contenente i termini noti; ad esempio se il sistema è il seguente:

$$\begin{aligned} 2x + 3y &= 11 \\ x + 5y &= 16 \end{aligned} \tag{3.4}$$

allora sarà:

$$\begin{aligned} A &= [2\ 3; 1\ 5]; \\ B &= [11\ 16]; \end{aligned} \tag{3.5}$$

Il sistema precedente in notazione matriciale viene indicato come:

$$A * X = B \tag{3.6}$$

dove X rappresenta il vettore soluzione e cioè il vettore:

$$X = [x\ y]; \tag{3.7}$$

La soluzione del sistema (3.6) è data da:

$$X = inv(A) * B; \tag{3.8}$$

L'm-file che effettua tale operazione è riportato di seguito:

```
%-------------------------------------------------------------
%Soluzione di un sistema lineare a Cura di Giuseppe Ciaburro
%-------------------------------------------------------------
%Sia A la matrice dei coefficienti delle incognite e
%B il vettore termine noti%Bisogna prima immettere nel promt di matlab A e B
%Valutiamo l'inversa di A
C=inv(A);
%Determiniamo la trasposta
C=C';
%Calcoliamo il vettore soluzione
X=B*C;
disp('Il vettore delle soluzioni è')
X
```

Come si vede è necessario effettuare la trasposta della matrice inversa perchè matlab esegue solo il prodotto di un vettore per una matrice e non il prodotto di una matrice per un vettore .

Effettuiamo allora tali calcoli con matlab:

```
C=inv(A)

C =

    0.7143   -0.4286
   -0.1429    0.2857
```

effettuiamone la trasposta:

```
C=C'

C =

    0.7143   -0.1429
   -0.4286    0.2857
```

Calcoliamo infine il vettore soluzione:

```
X=B*C

X =

    1.0000    3.0000
```