Rapida Introduzione all'uso del Matlab

Ottobre 2002

Tutti i tipi di dato utilizzati dal Matlab sono in forma di array.
I vettori sono array monodimensionali, e così possono essere viste le serie temporali, le matrici sono array bidimensionali.
È possibile inserire i dati in Matlab tramite alcune funzioni che permettono di leggere i file (come ad esempio *textread*, che permette di leggere file di testo, *imread* per leggere le immagini o *fread* che permette di leggere file binari), tramite *l'import wizard* che riesce a importare automaticamente una grande quantità di tipi di dati (l'*import wizard* viene fatto partire selezionando *Import Data* dal menù *File*) o manualmente.
In particolare è possibile assegnare a delle variabili i valori che ci interessano o inserendoli singolarmente o utilizzando delle funzioni apposite.

Per creare una variabile è sufficiente digitarne il nome e assegnarle un valore.

Digitando a=3 al prompt (>>) della command window viene creata una variabile di tipo *double* (double sta per double precision, un numero double occupa 8 byte di memoria). La variabile viene memorizzata nel *workspace* che è l'insieme di variabili create in una sessione di lavoro. È sempre possibile vedere il contenuto del workspace utilizzando l'apposita finestra *Workspace Browser*, altrimenti si può digitare la parola *whos* al prompt.
Nelle versioni più recenti di Matlab tramite il Workspace Browser è possibile vedere il contenuto delle variabili "cliccandoci" sopra due volte oppure usare il tasto destro del mouse per aprire un menu (*context menu*) che permette ad esempio di farne un grafico.
È importante osservare che Matlab è *case sensitive*, quindi distingue tra lettere minuscole e maiuscole.

Le operazioni matematiche sono eseguibili solo su dati di tipo *double*, se dobbiamo lavorare su un'immagine che può essere memorizzata come matrice di dati di tipo intero (es. int8 intero a 8 bit, int16 intero a 16 bit) bisogna prima convertire i dati in double (con apposite funzioni di conversione).

Matlab lavora anche con numeri complessi. I simboli "*i*" e "*j*" indicano l'unità immaginaria. Cediamo l'inserimento di un numero complesso nella command window:

```
>> c=3+4j

c =

   3.0000 + 4.0000i
```

Se vogliamo inserire un numero in forma esponenziale ad esempio 45×10^{-1} basta digitare…

```
>> b=45e-1

b =

    4.5000
```

Vediamo come inserire alcuni dati e come è possibile manipolarli dalla command window.

Inserimento vettore
I valori possono essere inseriti singolarmente

```
>> vett=[1 4 5 8 -10 11.5]

vett =

    1.0000    4.0000    5.0000    8.0000  -10.0000   11.5000
```

Si deve notare che dopo l'inserimento viene effettuato l'eco sullo schermo.
Per eliminarlo basta far seguire all'espressione inserita il simbolo ";"

```
>> vett=[1 4 5 8 -10 11.5];
```

Per inserire un vettore ordinato si può utilizzare la seguente forma…

```
>> vett=[inizio:incremento:fine]
```

es.

```
>> vett=[-10:2:10]

vett =

   -10    -8    -6    -4    -2     0     2     4     6     8    10
```

N.B. L'incremento può essere non intero e anche negativo. Come default vale 1.

Estrazione di una porzione del vettore

Possiamo ottener il valore i-esimo del vettore vett, digitando vett(i)

Se vogliamo estrarre ed assegnare ad una nuova variabile, scalar, il valore del quinto elemento di vett,

```
>> scalar=vett(5)

scalar =

    -2
```

Se vogliamo creare un altro vettore costituito da una porzione di vett, bisogna digitare vett_new=vett(i:k) dove i e k sono gli indici rispettivamente del primo e dell'ultimo elemento

```
>> vett_new=vett(3:6)

vett_new =

    -6    -4    -2     0
```

Inserimento matrici

Per creare manualmente dalla command window una matrice si possono inserire inserire i valori delle righe separati dal ; oppure andando a capo…

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

mat =

     1     2     3
     4     5     6
     7     8     9


>> mat2=[ 10 9 8 7
6 5 4 3
2 1 0 -1]

mat2 =

    10     9     8     7
     6     5     4     3
```

```
    2     1     0    -1
```

Le matrici possono essere create combinandone altre in vario modo…

```
>> row_mat=[mat mat2]

row_mat =

    1     2     3    10     9     8     7
    4     5     6     6     5     4     3
    7     8     9     2     1     0    -1
```

…oppure…

```
>> column_mat=[mat ; mat]

column_mat =

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

la stessa cosa si può fare partendo da vettori (che sono matrici 1xN o Nx1)

```
>> mat_from_vett=[vett ; vett]

mat_from_vett =

  -10    -8    -6    -4    -2     0     2     4     6     8    10
  -10    -8    -6    -4    -2     0     2     4     6     8    10
```

Come per i vettori è possibile estrarre un elemento dalle matrici o diverse porzioni, come righe, colonne.

Se mat è la nostra matrice l'elemento dell'i-esima riga e della j-esima colonna è

mat(i,j)

```
>> mat(2,3)
```

```
ans =

    6
```

N.B. in questo caso non ho assegnato a nessuna variabile il valore di mat(2,3) e il matlab ha assegnato automaticamente il valore alla variabile *ans* (answer). La variabile ans può essere usata in calcoli successivi.

Si possono estrarre porzioni della matrice, indicandole con mat(i:j,k:l), cioè gli elementi con indice di riga compreso tra i e j e indice di colonna tra k e l.

```
>> sub_mat=mat(1:2,2:3)

sub_mat =

    2     3
    5     6
```

Tramite il simbolo ":" si selezionano tutti gli elementi di riga o colonna: ad esempio la riga i-esima di mat è mat(i,:), idem per le colonne

```
>> mat(3,:)

ans =

    7     8     9
>> row_mat(:,2:4)

ans =

    2     3    10
    5     6     6
    8     9     2
```

altrimenti si possono inserire in un vettore gli indici degli elementi da estrarre…

```
>> row_mat(:,[1 4 7])

ans =

    1    10     7
    4     6     3
    7     2    -1
```

Si possono fare operazioni di assegnamento avendo cura di utilizzare operandi di uguale dimensione.

```
>> row_mat(1:2,[4 5])=mat(1:2,2:3)
```

Per eliminare una colonna o una riga basta fare un assegnamento con un array vuoto

```
>> mat(:,2)=[ ]
```

```
mat =

    1    3
    4    6
    7    9
```

Operazioni sugli array

In matlab sono definiti i seguenti **operatori aritmetici**

+ Addizione
- Sottrazione
* Moltiplicazione
/ Divisione destra
\ Divisione sinistra
^ Elevamento a potenza
‘ Trasposizione con coniugazione

Le dimensioni delle matrici devono essere congruenti con l’operazione effettuata, l’unica eccezione è rappresentata dalle operazioni tra matrici con uno scalare: in questo caso l’operazione con lo scalare è applicata ad ogni elemento della matrice.

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

```
mat =

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

```
>> 10*mat

ans =

    10    20    30
    40    50    60
    70    80    90
```

L'operazione di divisione può servire per la soluzione di un sistema di equazioni lineari. La soluzione del sistema AX=B è data da X=A\B, mentre il sistema XA=B è X=B/A.

Gli operatori ***, \ , / ,** se preceduti da un punto, lavorano singolarmente su elementi corrispondenti degli array (quindi lavorano con array di uguale dimensione) mentre con **.^** si indica l'elevamento a potenza di ogni singolo elemento dell'array.

Per effettuare la trasposizione di una array coniugando contemporaneamente i suoi elementi bisogna applicare l'operatore " ' " .

```
>> A=[1 3+j -j;6.4-5j 8.3 -1]

A =

   1.0000            3.0000 + 1.0000i       0 - 1.0000i
   6.4000 - 5.0000i   8.3000           -1.0000

>> A'

ans =

   1.0000            6.4000 + 5.0000i
   3.0000 - 1.0000i   8.3000
      0 + 1.0000i  -1.0000
```

Facendolo precedere dal punto (.') si ottiene invece la trasposizione senza coniugazione.

Operatori di confronto

Gli operatori di confronto sono

<	minore	>=	maggiore o uguale
<=	minore o uguale	= =	uguale
>	maggiore	~ =	diverso

Questi operano su elementi corrispondenti di matrici di uguale dimensione, confrontando quindi elemento per elemento. La solita eccezione è data dal confronto con uno scalare. Il risultato del confronto è pari a 1 se la condizione è verificata mentre è pari a 0 se è falsa.

```
>> A=[1 3; 4 6; 7 9];
>> B=[3 5; -1 5; 0 11];
>> A>B



ans =

    0     0
    1     1
    1     0

>> B>0

ans =

    1     1
    0     1
    0     1
```

Operatori logici

Gli operatori logici sono

&	AND
|	OR
~	NOT

Anche gli operatori logici lavorano su elementi corrispondenti degli array.
Il valore ottenuto dall'applicazione di AND è pari a 1 se entrambi gli operandi sono diversi da zero, mentre OR da come risultato 1 se almeno uno è diverso da zero.
L'operatore NOT nega l'operando al quale è applicato: quindi un operando diverso da zero diventa zero e ogni operando pari a 0 diviene 1.

```
>> A=[1 19 0 3 7];
```

```
>> B=[0 -2  0 0 7];

>> A&B

ans =

    0     1     0     0     1

>> A|B

ans =

    1     1     0     1     1

>> ~A

ans =

    0     0     1     0     0
```

Creazione di matrici e vettori tramite funzioni speciali

Il matlab prevede alcune funzioni per la generazione di matrici.

La funzione *ones(M,N)* genera una matrice M x N formata da elementi uguali a 1. Nel caso in cui inseriamo solo un argomento la matrice di uscita sarà quadrata.

```
>> ones(3,1)

ans =

    1
    1
    1

>> ones(3)

ans =

    1     1     1
    1     1     1
    1     1     1
```

Nello stesso modo funzionano, *zeros* che genera matrici di zeri, *rand* e *randn* che generano matrici composte da numeri casuali, rispettivamente uniformemente e normalmente distribuiti.
La funzione *eye* genera una matrice composta da zeri e con tutti elementi uguali a 1 sulla diagonale principale.

Funzioni di Matlab

In Matlab sono definite molte funzioni matematiche elementari e complesse.
Ad esempio *sin* che calcola il seno, *sqrt* che fornisce la radice quadrata di un numero, *mean* che calcola la media degli elementi di un vettore, *eig* che fornisce gli autovalori e gli autovettori di una matrice.
Questi esempi si prestano ad una prima classificazione delle funzioni presenti in Matlab: ad esempio funzioni come *sin* o *sqrt* lavorano essenzialmente su valori scalari e se applicati a vettori o matrici forniscono in uscita il risultato calcolato per ogni elemento. Funzioni come *mean* o similmente *max* vengono applicate a vettori (riga o colonna non importa) e se applicate a matrici forniscono il risultato dell'operazione sulle singole colonne.

Quindi possiamo definirci ad esempio un insieme di valori sui quali calcolare una funzione, metterli in forma vettoriale o matriciale e poi inserirlo come argomento.
Ad esempio possiamo definirci un intervallo temporale

```
>> tempo=[0:0.1:1]

tempo =

    0    0.1000    0.2000    0.3000    0.4000    0.5000    0.6000    0.7000    0.8000
0.9000    1.0000
```

e poi calcolare il valore della funzione seno in ogni istante

```
>> y=sin(tempo)

y =

    0    0.0998    0.1987    0.2955    0.3894    0.4794    0.5646    0.6442    0.7174
0.7833    0.8415
```

Il valore medio della nostra variabile y è facilmente calcolabile come

```
>> m=mean(y)

m =

    0.4558
```

Una lista delle funzioni matematiche elementari presenti in Matlab può essere ottenuta digitando

```
>> help elfun
```

mentre digitando

```
>> help specfun
>> help elmat
```

si avrà una lista di funzioni matematiche più avanzate e di funzioni che operano su matrici.
Matlab possiede molte altre funzioni che coprono una vastissima gamma di applicazioni che vanno dalla generazione di grafici 2D e 3D, all'elaborazione delle immagini e dei segnali, dalla creazione di reti neurali a risoluzione di problemi inerenti all'identificazione dei sistemi.
Molte di queste funzioni sono presenti nella versione base di Matlab, mentre altre sono acquistabili separatamente e sono contenute in Toolbox specifici per le diverse applicazioni.
Per avere una lista di tutte le tipologie di funzioni presenti al momento sul computer si può digitare help

```
>> help

HELP topics:

matlab\general       -  General purpose commands.
matlab\ops           -  Operators and special characters.
matlab\lang          -  Programming language constructs.
matlab\elmat         -  Elementary matrices and matrix manipulation.
matlab\elfun         -  Elementary math functions.
matlab\specfun       -  Specialized math functions.
matlab\matfun        -  Matrix functions - numerical linear algebra.
matlab\datafun       -  Data analysis and Fourier transforms.
etc…
```

La lista (che continua) è formata da una divisione delle funzioni per argomento. Per saperne di più su un singolo argomento si può digitare help seguito dal nome opportuno ad esempio

```
>> help datafun

  Data analysis and Fourier transforms.

  Basic operations.
    max       - Largest component.
    min       - Smallest component.
    mean      - Average or mean value.
    median    - Median value.
    std       - Standard deviation.
    var       - Variance.
    etc…
```

Una descrizione delle singole funzioni si ottiene digitando help seguito dal nome della funzione…..
Un help più evoluto è consultabile tramite il menù help presente nella finestra di Matlab oppure attraverso un'esplorazione del contenuto della Launch Pad, presente nelle versioni più recenti.

Operazioni dalla riga di comando

Tramite la riga di comando è possibile inserire variabili e applicare delle funzioni o operatori a variabili esistenti nel workspace. Dopo ogni istruzione, che può essere l'assegnamento di una variabile o l'esecuzione di una funzione, si può premere Invio, andando quindi alla riga successiva oppure inserire un'altra istruzione dopo la virgola o il punto e virgola.

```
>> vett1=[0 1 0], vett2=[1 4 6]'

vett1 =

    0     1     0


vett2 =

    1
    4
    6
```

Per continuare in una nuova riga l'inserimento di un'istruzione si possono usare tre punti prima dell'Invio.

```
>> vett=[1 2 3 ...
4 5 6]

vett =

    1    2    3    4    5    6
```

Istruzioni di controllo

Matlab utilizza istruzioni di controllo simili a quelle usate in molti linguaggi di programmazione. Vediamone alcune.

Ciclo for

```
for indice=inizio:incremento:fine

istruzioni

end
```

esempio

```
>> for i=1:10, x(i)=1/i^2; end;
```

l'indice i scorre i primi 10 elementi del vettore x e assegna loro il valore 1/i^2

Diversi cicli for possono essere annidati. Un esempio utile per lavorare sui singoli elementi di una matrice

```
for i=1:10
        for j=1:15
                mat(i,j)=i+j
        end
end
```

Ciclo while

```
while espression
```

istruzioni
end

esegue le istruzioni un numero indefinito di volte fintanto che l'espressione è vera.
Un'espressione può essere formata da un numero, una variabile, o una funzione di queste e fornisce un risultato.
Se il risultato è formato da un elemento diverso da zero, allora l'espressione è considerata vera,
Ad esempio l'espressione "3" è sempre vera mentre "0" è falsa.
L'espressione può comprendere confronto tra numeri o variabili oppure funzioni più complesse che in ogni caso forniscono un risultato in forma di numero o di array.
Per essere vera un'espressione di questo tipo deve fornire un risultato i cui elementi sono tutti diversi da zero.

Esempio.

Definito

>> a=[1 2 3 6 -1]

l'espressione a>0 da come risultato[1 1 1 1 0] ed è quindi falsa.
Mentre a>-2 che da come risultato [1 1 1 1 1] è vera.

>>
i=1;
while i<=10

x(i)=i.^2;
i=i+1;
end

break

Il comando break determina la fine dell'esecuzione del ciclo for o while all'interno del quale viene chiamato.
Se usato all'interno di cicli innestati, determina la fine del ciclo più interno.

If

if espressione

istruzioni

elseif espressione

istruzione

else

istruzione

end

Le istruzioni sono eseguite se l’espressione è vera.
I comandi else e elseif sono opzionali.