Manuale di MATLAB

(livello base)

1 Panoramica introduttiva

Il MATLAB (acronimo delle parole inglesi *MATrix LABoratory*) è un software basato sulla manipolazione di matrici molto utilizzato nel campo della ricerca scientifica, non solo matematica, a causa della sua grande portabilità (infatti è disponibile sia per grandi workstation che per comuni PC), unita ad una notevole facilità d'uso e alle potenzialità di calcolo. Inoltre l'uso del MATLAB è reso facile dalla presenza di un manuale dei comandi in linea, che può essere invocato tramite il comando `help`, e dalla presenza del comando `demo` che presenta numerosi e significativi esempi di applicazioni di tutte le funzioni MATLAB.
Per lanciare il MATLAB in ambiente Windows o Mac si deve effettuare un doppio click sull'icona del programma. A questo punto compare il prompt del software

```
>>
```

mentre per uscire si deve digitare exit oppure quit.
Il comando `help` come già detto fornisce tutte le informazioni relative ad un particolare comando oppure una lista di tutti gli argomenti per i quali è presente un aiuto. La sintassi del comando è semplice:

```
>>help
```

fornisce la lista di tutti gli argomenti per i quali è presente un aiuto, ad esempio:

```
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.
...
```

Oppure si utilizza la sintassi:

```
>>help comando
```

per visualizzare la guida relativa a *comando*. Ad esempio, digitando:

```
>>help load
```

viene visualizzata la guida relativa al comando `load`.
Per avere una dimostrazione interattiva delle funzionalità del MATLAB basta digitare il comando `demo`; comparirà un *menù ad albero* dal quale si potranno scegliere una lista di dimostrazioni da visualizzare.
Il MATLAB è un interprete di comandi le cui istruzioni sono del tipo:

```
>>variabile = espressione
```

oppure

```
>>variabile
```

In quest' ultimo caso, quando cioè un'istruzione è costituita solo dal nome di una variabile viene interpretata come la visualizzazione del valore di tale variabile. Vediamo i seguenti esempi:

```
>>b=5;
>>b
b =
     5

>>b=5
b =
     5
```

Nel primo caso dopo l'assegnazione è stato posto il punto e virgola ";" ed il valore 5 è stato registrato nella variabile b; digitando "b" il MATLAB visualizza il contenuto della variabile b. Nel secondo caso l'assegnazione non è stata seguita dal ";" per cui dopo l'invio il MATLAB visualizza automaticamente il contenuto della variabile b. Stesso risultato si sarebbe ottenuto digitando:

```
>>b=5,
b =
     5
```

vale a dire facendo seguire all'assegnazione la virgola ",".
Il MATLAB può essere utilizzato anche coma una semplice calcolatrice; a tale scopo basta digitare al prompt dei comandi la serie di calcoli da effettuare e premere invio:

```
>>3+4
ans =
     7
```

Il MATLAB registra il risultato dell'espressione numerica nella variabile `ans` (abbreviazione per la parola inglese *answer*): tale variabile è una variabile d'appoggio nel MATLAB e contiene la risposta più recente (most recent answer). Ciò vuol dire che il risultato di una nuova espressione numerica è registrato automaticamente nella variabile `ans` sovrascrivendo il valore precedente della stessa. Ogni espressione introdotta viene interpretata e calcolata. Ogni istruzione può essere scritta anche su due righe purché prima di andare a capo vengano scritti 3 punti "..." *.* Più espressioni possono essere scritte sulla stessa riga purché siano separate da una virgola o dal punto e virgola. Se una riga di un file MATLAB inizia con `%` allora tale riga viene considerata come un commento. Il MATLAB fa distinzione tra lettere minuscole e maiuscole (vale a dire, con terminologia inglese, che è *case sensitive*); quindi, se abbiamo definito una variabile `A` e facciamo riferimento a questa scrivendo `a` essa non viene riconosciuta.

1.1 MATLAB Workspace

Tutte le variabili inserite sono registrate nel *workspace*; dopo la serie di comandi appena inseriti il *MATLAB workspace* contiene le variabili: `b` ed `ans`. I comandi `who` e `whos` si utilizzano per interrogare il workspace, vale a dire per ottenere informazioni sulle variabili in esso contenute. In particolare, il comando `who` determina la visualizzazione dell'elenco del nome delle variabili presenti nel workspace; il comando `whos` determina la creazione di un elenco alfabetico (sono poste per prime le variabili con la maiuscola) delle variabili presenti nel workspace, con l'aggiunta di informazioni sul tipo e sulla dimensione di memoria occupata. Ad esempio, dopo aver digitato i precedenti comandi:

```
>> who
Your variables are:
ans  b
```

digitando, invece:

```
>> whos
  Name      Size                    Bytes  Class
```

```
  ans         1x1                            8  double array
  b           1x1                            8  double array
Grand total is 2 elements using 16 bytes
```

L'output dell'ultimo commando permette di mettere in evidenza una peculiarità del MATLAB: *tutte le variabili hanno un'organizzazione matriciale*, vale a dire un numero è una matrice 1x1.

1.2 Alcune semplici operazioni

La Figura 1 mostra la schermata di avvio della versione 7.1 di MATLAB; con l'aiuto di questa figura è possibile svolgere una rapida carrellata di semplici operazioni di frequente utilizzo.

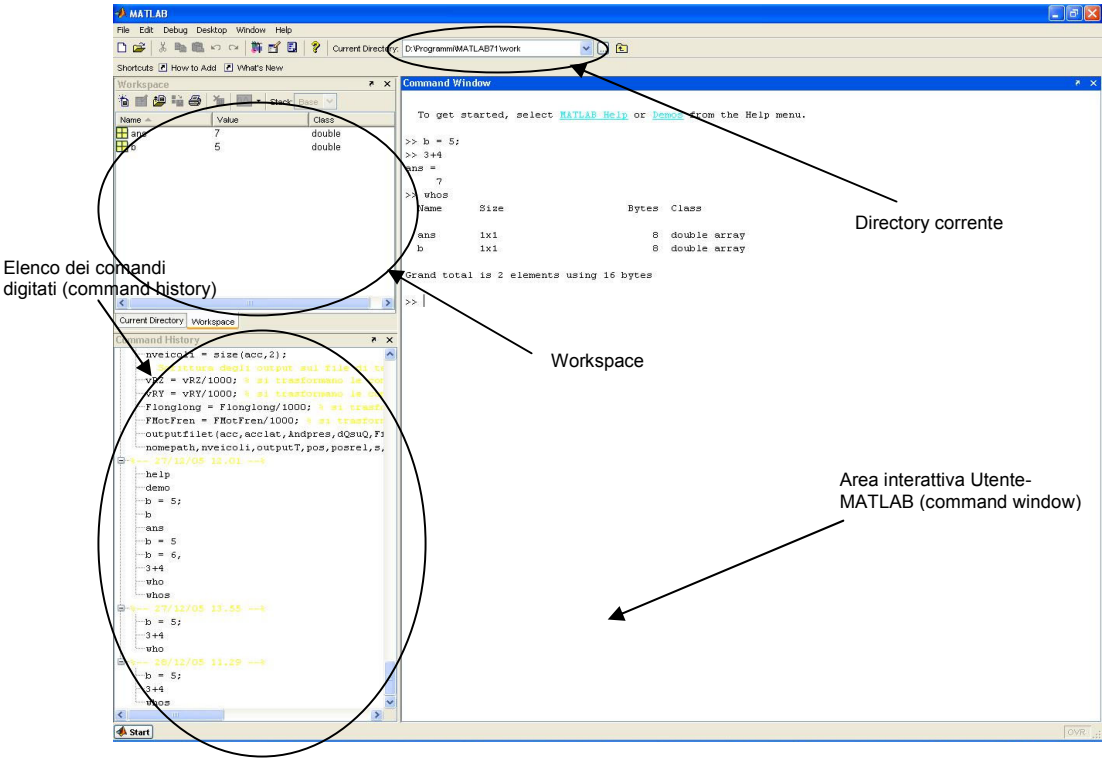

Figura 1. Ambiente MATLAB (MATLAB desktop environment).

1.2.1 Cambio della directory di lavoro

La directory di default, a meno di non cambiare le impostazioni predefinite del MATLAB, è: <dir installazione>\work. Per il MATLAB 7.1 è, ad esempio: C:\Programmi\MATLAB71\work. Per cambiarla basta semplicemente editare il contenuto della directory corrente. Oppure, in alternativa, ma questo modo di procedere è più utile in sede di programmazione, si può utilizzare il comando cd (*change directory*):

```
>> cd C:\Mia directory
>> cd(VariabileConMiaDirectory)
```

Nel primo caso si è inserito manualmente il nome della directory di lavoro preferita dall'Utente, nel secondo, utilizzato tipicamente in ambito di programmazione, il nome della directory è registrato nella variabile stringa `VariabileConMiaDirectory`.

1.2.2 Elenco comandi digitati

La *command history* contiene l'elenco dei comandi inseriti manualmente dall'Utente. Per richiamarli vi si

può accedere direttamente con il mouse, mediante *drag&drop* o doppio click oppure utilizzando le frecce su (↑) e giù (↓) del cursore; oppure, per ricercare i comandi inseriti che iniziano con la "a", basta digitare a e premere più volte la freccia in su ↑ per avere un elenco di comandi inseriti che iniziano con la a.

1.2.3 Salvataggio, caricamento e cancellazione dati

Sono operazioni tipiche che servono a registrare (comando save) o a prelevare (comando load) dati da disco ed a cancellare dati dalla memoria volatile (comando clear). L'utilizzo dei comandi save e load è molto simile:

`SAVE FILENAME` salva tutte le variabili del *workspace* nel file FILENAME.mat (all'interno della directory corrente).

`SAVE` da solo crea un file chiamato matlab.mat e vi registra tutte le variabili contenute nel *workspace*.

`SAVE FILENAME X` salva solo la variabile `X`.

`SAVE FILENAME X Y Z` salva le variabili X, Y, Z. si può usare l'asterisco '*' per salvare tutte le variabili che cominciano con una data combinazione di lettere, ad esempio, `SAVE FILENAME X Y*` salva nella directory corrente il file FILENAME.mat contenente le variabili X e tutte quelle che cominciano con Y.

Durante la programmazione, di solito conviene utilizzare una variabile contenente il nome del file dove si vuole registrare il contenuto delle variabili; in tal caso, il commando `save (NFileSalvaDati)` salva le variabili del *workspace* nel file contenuto nella variabile `NFileSalvaDati`. Allo stesso modo funziona il comando `load`.

Per cancellare le variabili contenute nel *workspace* si utilizza il comando clear:

`CLEAR` cancella tutte le variabili dal *workspace*.

`CLEAR X Y` cancella le variabili X ed Y; anche qui si può usare l'asterisco '*' con funzioni del tutto simili al caso precedente.

`CLEAR GLOBAL` rimuove solo le variabili globali (vedere …).

`CLEAR ALL` cancella tutto: variabili, variabili globali, punti di interruzione nel debug, …

2 Creazione delle variabili

La prima cosa da imparare del MATLAB è come manipolare le matrici che costituiscono la struttura fondamentale dei dati. Una *matrice* è una tabella di elementi caratterizzata da due dimensioni: il numero delle righe e quello delle colonne. I vettori sono matrici aventi una delle dimensioni uguali a 1. Infatti, come ben noto, esistono due tipi di vettori: i *vettori riga* aventi dimensione 1 × n, e i *vettori colonna* aventi dimensione n × 1. I dati scalari sono matrici di dimensione 1×1. Le matrici possono essere introdotte in diversi modi, per esempio possono essere assegnate esplicitamente, o caricate da file di dati esterni, utilizzando funzioni predefinite. Per esempio, l'istruzione:

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

assegna alla variabile A una matrice di tre righe e tre colonne. Gli elementi di una riga della matrice possono essere separate da virgole o dallo spazio, mentre le diverse righe sono separate da un punto e virgola. Se alla fine dell'assegnazione viene messo il punto e virgola allora la matrice non viene visualizzata sullo schermo. In generale, se vogliamo assegnare ad A una matrice ad *m* righe ed *n* colonne la sintassi è la seguente:

```
>>A = [riga 1;riga 2;…;riga m];
```

(è chiaro che gli elementi delle diverse righe devono essere gli stessi, vale a dire i diversi vettori riga devono avere lo stesso numero di colonne o, che è lo stesso, medesima dimensione.

Per definire un vettore riga x, basta, ad esempio, digitare:

```
>> x = [1 -2 +5];
```

Alternativamente:

```
>> x = [1,-2,+5];
```

Per definire, invece, un vettore colonna basta separare le diverse componenti di un vettore mediante un punto e virgola:

```
>> y = [1;-2;+8];
```

È possibile inserire anche un ritorno a capo anziché il punto e virgola e definire in modo ugualmente corretto il vettore colonna.
Un vettore colonna si può anche definire utilizzando l'operatore trasposto complesso ^{H}, in MATLAB ('), a partire da un vettore riga (il paragrafo 3 parla più diffusamente dei comuni operatori MATLAB):

```
>> a = [1 3 7]'
a =
     1
     3
     7
```

Una matrice od un vettore può anche essere assegnato dinamicamente cambiando le sue dimensioni:

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

Stesso modo di procedere si può adottare per un vettore.
L'elemento della riga *i* e della colonna *j* viene denotato con A(i, j), per accedere ad un elemento della matrice basta scrivere:

```
>> A(2,1)
ans =
     4
```

Per accedere ad elementi di un vettore, sia esso riga o colonna, si può usare sia la notazione con due indici come per le matrici, ove l'indice *i* sarà 1 per i vettori riga o, alternativamente sarà 1 l'indice *j* per i vettori colonna; oppure, più semplicemente utilizzando una notazione con un solo indice che sta ad indicare la componente del vettore cui ci si vuole riferire[1]:

```
>> b(2)
ans =
     8
```

Se C è una matrice non ancora inizializzata allora l'istruzione

```
>> C(3,2)= 1
```

fornisce come risposta

```
C =
0     0
0     0
0     1
```

[1] Evidentemente, accedendo ad una componente che non esiste si provocherà un errore.

cioè il programma assume come dimensioni per C dei numeri sufficientemente grandi affinché l'assegnazione abbia senso. Se ora si pone:

```
>> C(1,3)= 2
```

si ha:

```
C =
0     0     2
0     0     0
0     1     0
```

In MATLAB gli indici devono essere strettamente positivi, eventuali valori negativi o nulli producono errore.

2.1 Definizione di vettori mediante l'operatore :

L'operatore due punti (:) può essere utilizzato convenientemente per definire vettori le cui diverse componenti sono legate tra loro; ad esempio, per definire un vettore v di 5 componenti, con valori delle componenti che vanno da 1 a 5 basta digitare:

```
>> v = 1:5
v =
     1     2     3     4     5
```

Oppure, alternativamente:

```
>> v = [1:5]
v =
     1     2     3     4     5
```

In generale, l'operatore : si utilizza in istruzioni del tipo: ind1:incr:ind2, dove se incr > 0 allora ind1 < ind2 per non avere un vettore vuoto, altrimenti può anche risultare ind1 > ind2. Di fatto:
ind1:incr:ind2 è equivalente a ind1,ind1+incr,ind1+2*incr,…,ind2.
L'operatore due punti (:) può essere utilizzato anche per accedere ad una parte di matrici o vettori; ad esempio:

```
>> v(2:4)
ans =
     2     3     4
>> v(4:-1:2)
ans =
     4     3     2
```

Infine, utilizzando l'operatore due punti senza specificare gli indici ind1 e ind2 si richiede implicitamente di accedere a tutti gli elementi di una data dimensione:

```
>> A(:,1)
ans =
     1
     4
     7
>> A(2,:)
ans =
     4     5     6
```

Combinando quanto finora visto dovrebbe essere possibile comprendere la seguente assegnazione:

```
>> A(5,:) = v(5:-1:3)
```

```
A =
     1     2     3
     4     5     6
     7     8     9
     0     0     0
     5     4     3
```

Nella quale si è aumentata la dimensione della matrice A e si è associata alla sua quinta riga gli elementi del vettore v presi dal quinto al terzo in ordine decrescente. Si noti una sottigliezza:

```
>> A(4,:) = v(5:-1:2)
```

produce un errore perché si è tentato di associare ad un vettore riga contenente 3 componenti –A(4,:)– un vettore riga contenente 4 componenti –v(5:-1:2)–. Invece risulta:

```
>> A(4,1:4) = v(5:-1:2)
A =
     1     2     3     0
     4     5     6     0
     7     8     9     0
     5     4     3     2
     5     4     3     0
```

la quale è corretta perché si è ridefinita la matrice A.
Questi modi di procedere sono validi anche per le matrici pluridimensionali, che però non vengono qui trattate.

2.2 Indicizzazione lineare

In precedenza si è detto che per accedere ad una componente di un vettore si può utilizzare un solo indice; tale modo di procedere può essere applicato anche alle matrici e risulta molto potente in quanto consente di eseguire operazioni anche complesse in una sola istruzione di codice; digitando:

```
>> A(6)
ans =
     8
```

Vale a dire si è acceduti all'elemento di posizione 3,2[2]. Nell'accesso ad un solo indice, il MATLAB fornisce come risposta l'elemento che si ottiene contando a partire da quello di posto (1,1) scendendo per le diverse colonne a partire dalla prima; digitando:

```
>> A(7);
```

si registra nella variabile ans il valore `3`. In pratica, in una matrice mxn per accedere all'elemento di posto *(i,j)*, basta usare l'indice I = i+(j-1)*m.
L'indicizzazione lineare può essere utilizzata anche per accedere a più elementi di una matrice contemporaneamente; per accedere agli elementi in posizione (2,2) e (2,3) basta digitare:

```
>> A([5 8])
ans =
     5     6
```

[2] Ci si sta riferendo all'espressione di `A` data al paragrafo 2.

2.3 Sottomatrici e concatenazione

Data la matrice `A`, l'operazione `A(v_row,v_col)` seleziona una sottomatrice di `A` corrispondente alle righe indicata nel vettore `v_row` ed alle colonne indicate in `v_col`. Consideriamo la seguente matrice `A`:

```
A =
      0.95013       0.7621      0.61543      0.40571     0.057891
      0.23114      0.45647      0.79194      0.93547      0.35287
      0.60684     0.018504      0.92181       0.9169      0.81317
      0.48598      0.82141      0.73821      0.41027    0.0098613
       0.8913       0.4447      0.17627      0.89365      0.13889
```

Il comando `A(2:4,1:3)` seleziona la sottomatrice di `A` corrispondente alle righe dalla 2 alla 4 ed alle colonne dalla 1 alla 3:

```
      0.23114      0.45647      0.79194
      0.60684     0.018504      0.92181
      0.48598      0.82141      0.73821
```

Si sarebbe potuto ottenere lo stesso risultato mediante i comandi:

```
A([2 3 4],[1 2 3]);
```

oppure:

```
v_row = 2:4; v_col = 1:3;
A(v_row,v_col)
```

Prima si è visto un esempio di concatenazione; più precisamente valgono le due semplici definizioni:

concatenazione orizzontale: Se le B_i sono matrici aventi tutte lo stesso numero di righe (possono essere anche vettori colonna), la matrice A può essere definita a partire da tali matrici per concatenazione orizzontale, vale a dire nei due modi seguenti equivalenti:

`A = [B1,B2,…Bn]` oppure `A = [B1 B2 … Bn];`

concatenazione verticale: Se le B_i sono matrici aventi tutte lo stesso numero di colonne (possono essere anche vettori riga), la matrice A può essere definita a partire da tali matrici per concatenazione verticale:

`A = [B1;B2;…Bn]`

Qualora le precedenti condizioni sulle B_i non fossero valide ci sarebbe un errore.
Esempio:

```
>> B1 = [5 4 9];
>> B2 = [2 5 7];
>> A1 = [B1 B2]
A1 =
     5     4     9     2     5     7
>> A2 = [B1;B2]
A2 =
     5     4     9
     2     5     7
```

2.4 Matrici elementari

In MATLAB sono definite alcune funzioni per la creazione rapida di matrici elementari; nel seguito si

forniscono solo gli utilizzi più semplici di tali funzioni per informazioni più avanzate si rimanda alla guida delle rispettive funzioni. Le funzioni descritte nel seguito sono utilissime per preallocare vettori e matrici.

2.4.1 zeros

Tale funzione crea una matrice di zeri di ordine mxn:

```
>> Z = zeros(3,5)
Z =
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
```

Per creare una matrice di zeri quadrata di ordine n basta scrivere ad esempio: `Z = zeros(3);`

2.4.2 ones

Questa funzione crea una matrice di uno (1) di ordine mxn:

```
>> O = ones(2,4)
O =
     1     1     1     1
     1     1     1     1
```

Per creare una matrice quadrata si procede come per la precedente funzione. È evidente che per creare una matrice piena dello scalare s, basta scrivere (si veda il paragrafo 3 per una rapida carrellata circa i simboli degli operatori matematici utilizzati in MATLAB):

```
>> S = 8*ones(2,4)
S =
     8     8     8     8
     8     8     8     8
```

2.4.3 eye

Per creare la matrice identità di ordine n si utilizza la funzione `eye`:

```
>> I = eye(3)
I =
     1     0     0
     0     1     0
     0     0     1
```

2.4.4 rand e randn

Generano matrici o vettori i cui elementi sono distribuiti randomicamente secondo una distribuzione uniforme (`rand`) e gaussiana o normale (`randn`). Il loro utilizzo è del tutto simile a quello delle funzioni precedenti.

2.4.5 linspace e logspace

Sono funzioni utilizzate per creare vettori i cui elementi sono spaziati linearmente (`linspace`) o logaritmicamente (`logspace`); in particolare:

`LINSPACE (X1, X2, N)` genera un vettore riga ad N componenti in cui la prima componente è X1 e l'ultima (l'N-esima) è X2, i valori delle altre componenti si ottengono per spaziatura lineare

tra X1 ed X2, vale a dire che la i-esima componente valrà X1+(i-1)*(X2-X1)/(N-1). Se N vale 1, linspace restituisce X2.

`LOGSPACE (X1, X2, N)` genera un vettore riga di N componenti i cui valori sono spaziati logaritmicamente tra 10^X1 e 10^X2. Risulta $\frac{v(i+1)}{v(i)} = \frac{v(j+1)}{v(j)}$

2.5 Informazioni sulle matrici

La tabella seguente raccoglie le modalità più utilizzate per ottenere informazioni dalle matrici (o dai vettori qualora applicate a vettori):

Per conoscere…	Funzione	Risultato restituito
La dimensione di una matrice	`size`	numero di righe e colonne della matrice
La dimensione massima di una matrice	`length`	dimensione massima della matrice
se due matrici sono uguali	`isequal`	1 se le matrici sono uguali, 0 altrimenti
il numero di dimensioni di una matrice	`ndims`	dimensioni della matrice (`length`(size(A))
il numero di elementi di una matrice	`numel`	m·n
se una matrice è vuota	`isempty`	1 se la matrice è vuota, 0 altrimenti.

2.6 Variabili speciali

La tabella seguente raccoglie alcune delle variabili speciali utilizzate dal MATLAB, per saperne di più digitare help elmat.

`ans`	risposta più recente
`pi`	pi greco -arcocoseno di -1-
`i e j`	unità immaginarie
`inf`	infinito, vale a dire il più grande numero positivo rappresentabile
`NaN`	Il risultato dell'operazione non è un numero, Not a Number

La costante `inf` è ottenuta come risultato di una divisione per zero oppure il calcolo del logaritmo di zero o se il risultato è un overflow. La costante `NaN` invece è ottenuta come risultato di operazioni matematicamente non definite come 0/0 oppure $\infty - \infty$. Come accade per la maggior parte dei linguaggi di programmazione anche in MATLAB è possibile definire variabili il cui nome è una costante predefinita, quindi, per esempio, è possibile usare la variabile `i` come indice intero, ma ciò è sconsigliabile.

3 Operatori matematici

In MATLAB sono definite le seguenti operazioni su matrici e vettori:

operatore	significato	scalare	matrice
+	somma	*a+b*	$(A+B)_{ij}=A_{ij}+B_{ij}$
–	sottrazione	*a-b*	$(A-B)_{ij}=A_{ij}-B_{ij}$
*	moltiplicazione	*a*b*	$A*B$[3]
^	elevamento a potenza	*a^b*	A^s[4]
/	divisione a destra	*a/b*	$A/B=A*B^{-1}$
\	divisione a sinistra	*b/a*	$A\backslash B = A^{-1}*B$
.*	prodotto elemento per elemento	*a*b=a.*b*	$(A.*B)_{ij}=A_{ij}*B_{ij}$

[3] Prodotto matriciale, il numero di colonne di A deve essere uguale al numero di righe di B.
[4] Nell'elevamento a potenza, uno dei due operatori deve essere uno scalare.

./	divisione a destra elem per elem	*a./b = a/b*	$(A./B)_{ij} = A_{ij}/B_{ij}$
\.	divisione a sinistra elem per elem	*a.\b = a\b*	$(A.\backslash B)_{ij} = B_{ij}/A_{ij}$
.^	elevamento a potenza elem per elem	*a.^b = a^b*	$(A.\hat{}B)_{ij} = A_{ij}\hat{}B_{ij}$

3.1 Operatori logici e relazionali

La tabella seguente mostra i simboli ed i significati delle operazioni relazionali più comunemente utilizzate:

Operatori	==	~=	>	<	>=	<=
Significato	uguale	diverso da	maggiore	minore	maggiore o uguale	minore o uguale

Ogni operazione logica restituisce 1 se è vera 0 altrimenti. Per eseguire un'operazione logica tra matrici è necessario che le loro dimensioni siano le stesse: $(A\mathbf{op}B)_{ij}=A_{ij}\mathbf{op}B_{ij}$. Un'operazione logica tra matrici e scalari è equivalente a: $(A\mathbf{op}b)_{ij}=A_{ij}\mathbf{op}b$.
Gli operatori logici principali, invece, sono:

Operatori	&	|	~
Significato	and	or	not

Per saperne di più digitare help ops. Gli operatori logici che coinvolgono gli scalari sono && (and) e || (or).

4 Funzioni matematiche elementari

Nel seguito si riporta una breve carrellata delle funzioni matematiche elementari più comunemente utilizzate, il MATLAB fornisce un elenco completo di tali funzioni digitando al prompt help elfun.

4.1 Funzioni trigonometriche

Gli argomenti delle funzioni trigonometriche devono essere espressi in radianti, così come sono in radianti i risultati delle funzioni trigonometriche inverse.:

Funzione	**Significato**	**Funzione**	**Significato**
`sin(x)`	seno	`asin(y)`	arcoseno
`cos(x)`	coseno	`acos(y)`	arcocoseno
`tan(x)`	tangente	`atan(y)`	arcotangente

4.2 Funzioni esponenziali

Funzione	**Significato**
`exp(x)`	e^x
`log(x)`	ln(x)
`log10(x)`	$\log_{10}(x)$
`log2(x)`	$\log_2(x)$
`sqrt(x)`	$\sqrt{}$
`x^y`	x^y

4.3 Funzioni varie

Altre funzioni comunemente utilizzate sono:

Funzione	Significato
`abs(x)`	|x|
`imag(x)`	Im(x)
`real(x)`	Re(x)
`sign(x)`	segno(x)
`rem(x/y)`	resto della divisione
`round(x)`	arrotondamento di x[5]
`floor(x)`	arrotondamento verso l'intero più piccolo
`ceil(x)`	arrotondamento verso l'intero più grande.

5 Funzioni di vettore

Le precedenti funzioni si possono applicare sia a scalari che a vettori nonché a matrici; il loro utilizzo più frequente è in relazione agli scalari; le funzioni di questo paragrafo si possono applicare anche a matrici, ma è più comune il loro utilizzo per i vettori ed a questi ci si riferirà esplicitamente per semplicità.

Funzione	Significato
`max(x)`	massimo elemento di un vettore
`min(x)`	minimo elemento di un vettore
`sum(x)`	somma degli elementi di un vettore
`cumsum(x)`	somma cumulativa.
`sort(x)`	ordinamento di un vettore
`length(x)`	dimensione del vettore x
`norm(x)`	norma di x

Per esempio per determinare il massimo elemento di una matrice A si deve scrivere `max (max (A))`, `max(A)` fornirebbe, infatti, un vettore riga con i valori massimi di tutte le colonne di A. Le funzioni `max` e `min` possono fornire in uscita anche l'indice della componente massima (o minima) del vettore. La sintassi in questo caso è la seguente:

```
>> [massimo,k]=max(x);
>> [minimo,k]=min(x);
```

La variabile k contiene, rispettivamente, l'indice dov'è contenuto il massimo ed il minimo valore di x; risulta massimo = x(k) e minimo = x(k), rispettivamente.

6 Funzioni di matrici

Le più utili funzioni di matrici sono riportate nella tabella seguente:

Funzione	Utilizzo
`eig(·)`	Calcolo autovalori ed autovettori
`inv(A)`	inversa di A
`det(A)`	determinante di A
`cond(A)`	numero di condizionamento[6]
`rank(A)`	rango della matrice
`tril(x)`	estrazione della parte triangolare inferiore

[5] round di 2.5 da come risultato 3. Risulta sempre floor(x)≤round(x)≤ceil(x).
[6] È utile per valutare la singolarità di una matrice; alti valori del numero di condizionamento indicano una matrice quasi singolare, vale a dire non con rango massimo..

`triu(A)`	estrazione della parte triangolare superiore
`diag(A)`	estrazione della diagonale di una matrice[7]

La funzione `eig` può avere un diverso numero di input e di output:
`E = EIG(X)` fornisce in output un vettore contenente gli autovalori della matrice quadrata `X`.
`[V,D] = EIG(X)` fornisce in output una matrice diagonale `D` contenente gli autovalori di `X` ed una matrice piena `V` le cui colonne sono i corrispondenti autovettori, cosicché: $X \cdot V = V \cdot D$.
`[V,D] = EIG(A,B)` fornisce una matrice diagonale `D` degli autovalori della matrice $A^{-1} \cdot B$ ed una matrice piena V contenente gli autovettori corrispondenti della matrice $A^{-1} \cdot B$ cosicché $A \cdot V = B \cdot V \cdot D$.

7 Le stringhe

La stringa è una matrice di caratteri; il tipo di dato associato al carattere in MATALB è il `char`. Un esempio di variabile stringa è dato dalla seguente assegnazione:

```
S = 'esempio di stringa';
```

basta quindi far seguire all'uguale una serie di caratteri contenuti tra apici. Nel caso in cui la stringa abbia un apostrofo si procede come di seguito:

```
S = 'stringa con l''apostrofo';
```

vale a dire che per indicare un apostrofo si usano due apici. Le stringhe definite precedentemente sono vettori di caratteri anziché di numeri, pertanto valgono le regole viste precedentemente sull'accesso e sulla manipolazione:

```
>> S(9:11)
ans =
con
>> S([1 4 7])
ans =
sia
>> S(9:11) = []
S =
stringa  l'apostrofo
```

Nell'ultimo esempio si è usato il vettore vuoto `[]`, il risultato è stato, di fatto, quello di cancellare i caratteri compresi tra la posizione 9 e la 11.

```
>> S+5
ans =
  Columns 1 through 12
   120   121   119   110   115   108   102    37    37   113    44   102
  Columns 13 through 20
   117   116   120   121   119   116   107   116
```

Sommando (o sottraendo) ad una stringa uno scalare si ottiene un vettore numerico contenente la codifica ASCII dei caratteri precedentemente costituenti la stringa più (o meno, rispettivamente) lo scalare. Per visualizzare i caratteri associati a tale sequenza numerica basta utilizzare il comando `char`.

[7] Applicando tale funzione ad un vettore si ottiene una matrice diagonale avente sulla diagonale principale gli elementi di tale vettore; se x è un vettore, allora x = diag(diag(x)).

7.1 Funzioni correlate alle stringhe

Se `X` è un vettore numerico, la funzione `char` restituisce una stringa delle stesse dimensioni di `X`, ma con caratteri corrispondenti alla codifica ASCII di `X`; chiaramente se X contiene un valore superiore a 255 allora l'output non sarà di alcuna utilità, vale a dire non sarà alcun carattere.
La funzione inversa di `char` è `double`: se `S` è una stringa, l'istruzione `X = double(S)` restituisce un vettore delle stesse dimensioni di `S`, ma numerico, vale a dire con i valori numerici associati ai caratteri di `S`, sempre secondo la codifica ASCII.
Prima si è detto che le stringhe possono essere manipolate in maniera del tutto simile alle matrici; esse hanno però funzioni speciali per la concatenazione.

- *Concatenazione orizzontale*: se le S_i sono stringhe, esse si possono concatenare orizzontalmente nei due seguenti modi equivalenti: S = [S1, S2, … Sn]; S = `strcat`(S1, S2, …, Sn);
- *Concatenazione verticale*: se le S_i sono stringhe tutte con lo stesso numero di caratteri (spazi compresi), esse si possono concatenare verticalmente nei due seguenti modi equivalenti: S = [S1; S2; … Sn]; S = `strvcat`(S1, S2, …, Sn)[8];

Per capire se una variabile è numerica o di caratteri (vale a dire è una stringa) è utile la funzione `ischar` che restituisce `1` se l'argomento è una stringa, `0` altrimenti.

8 Esercizi

1) Definire una matrice nulla di ordine (3,5). Definire ora un vettore riga a ed assegnarlo alla prima riga di A (senza far cambiare le dimensioni di A). Detta B la trasposta di A, estrarne la sottomatrice C di ordine (3,3) corrispondente alle righe 2:4 di B.
2) Utilizzando le matrici A e B del punto precedente, definire una nuova matrice D di ordine 5,3 piena di "uno" ed effettuare le tre moltiplicazioni: M1 = D*A; M2 = A*D; e M3 = moltiplicazione elemento per elementi tra D e B. Verificare che le dimensioni delle matrici sono (rispettivamente): (5,5), (3,3), (5,3).
3) Salvare nel workspace le sole matrici M1, M2 ed M3.

[8] In realtà, la funzione `strvcat` consente il concatenamento verticale anche di stringhe aventi un numero diverso di caratteri: in tal caso l'istruzione `strvcat` aggiungerà un opportuno numero di spazi fittizi.