Riassunto di MATLAB

Riccardo Montagnin

Indice

Capitolo 1

Utility generali

All'interno di MATLAB esistono vari comandi che aiutano l'utente a districarsi tra le varie sue funzionalità. Tra i più importanti ricordiamo sicuramente:

- **help**
  Questo comando permette di visualizzare la schermata di aiuto per una data funzionalità, e può essere utilizzato in due modi differenti:

  1. `help`: elenca tutti i topic principali di aiuto nella Finestra Comandi. Ogni topic principale corrisponde al nome di una cartella nel percorso di ricerca di MATLAB.
  2. `help nome`: visualizza il testo di aiuto per la funzionalità specificata da `nome`, come una funzione, un metodo, una classe, una toolbox o una variabile.

- **lookfor**
  Questo comando viene utilizzato quando si vuole trovare il nome di una funzione che svolga un dato compito. Viene utilizzata nei seguenti modi:

  1. `lookfor topic`: ricerca la stringa topic all'interno della prima riga di commento (H1) dei programmi MATLAB trovati all'interno del percorso di ricerca. Per ogni file per il quale si è trovata una corrispondenza, MATLAB mostra l'intera riga H1.
  2. `lookfor topic -all`: ricerca una corrispondenza per la stringa topic nell'intero primo blocco di commento dei programmi nel percorso di ricerca.

Capitolo 2

Utility numeriche

I comandi più utili dal punto di vista puramente numerico sono i seguenti.

2.1 Costanti macchina

- `realmin`: mostra il più piccolo numero macchina rappresentabile.
  Valore: `2.2250738585072 e-308`
- `realmax`: mostra il più grande numero macchina rappresentabile.
  Valore: `1.79769313486232 e+308`
- `eps`: mostra la precisione di macchina, ovvero la più piccola quantità di modulo diverso da 0 che, sommato ad 1, restituisce un risultato diverso da 1. Esso è anche il più grande errore relativo compiuto nell'approssimazione di un numero reale all'interno dell'intervallo `[realmin, realmax]`.
  Valore: `2.2204 e-16`

2.2 Costanti numeriche

- `pi`: rappresenta l'approssimazione di π.
- `Inf`: rappresenta l'approssimazione di ∞.
- `NaN`: Not a Number, rappresenta un risultato non computabile, come quello derivante da una espressione come 0/0 o qualsiasi altra forma indeterminata.

2.3 Operazioni numeriche

Operazioni aritmetiche

- `+` : somma.
- `-` : differenza.
- `*` : prodotto.
- `/` : quoziente.
- `^` : potenza.

Funzioni elementari Sia `x` una qualsiasi espressione, costante, o un qualsiasi vettore. Allora in MATLAB esistono le seguenti.

- `abs(x)` : valore assoluto.
- `sin(x)` : seno.
- `cos(x)` : coseno.
- `tan(x)` : tangente.
- `cot(x)` : cotangente.
- `asin(x)` : arco seno.
- `acos(x)` : arco coseno.
- `atan(x)` : arco tangente.
- `sinh(x)` : seno iperbolico.
- `cosh(x)` : coseno iperbolico.
- `tanh(x)` : tangente iperbolica.
- `asinh(x)` : arco seno iperbolico.
- `acosh(x)` : arco coseno iperbolico.
- `atanh(x)` : arco tangente iperbolica.
- `sqrt(x)` : radice quadrata.
- `exp(x)` : esponenziale.
- `log2(x)` : logaritmo in base 2.
- `log10(x)` : logaritmo in base 10.
- `log(x)` : logaritmo naturale (base e).
- `fix(x)` : arrotondamento verso lo 0.
- `round(x)` : arrotondamento verso l'intero più vicino (per eccesso o difetto).
- `floor(x)` : arrotondamento verso $-\infty$ (per difetto).
- `ceil(x)` : arrotondamento verso $+\infty$ (per eccesso).
- `sign(x)` : segno
  Valore: `-1` se l'elemento è < 0, `0` se l'elemento è 0, `+1` se l'elemento è > 0).
- `rem(x)` : resto di una divisone.

Capitolo 3

Formati di visualizzazione dei numeri

3.1 Come MATLAB memorizza i numeri

MATLAB dispone di vari formati numerici che visualizzano, quando necessario, i numeri macchina in modi diversi:

- `format short`: notazione decimale con 4 cifre dopo la virgola.
- `format short e`: notazione esponenziale con 4 cifre dopo la virgola.
- `format long`: decimale con 15 cifre dopo la virgola in doppia precisione, e 7 cifre dopo la virgola in singola precisione.
- `format long e`: notazione esponenziale con 15 cifre dopo la virgola in doppia precisione, e 7 cifre dopo la virgola in singola precisione.
- `format long g`: la più compatta tra `format long` e `format long e`.

Oltre a questi esistono due comandi per trasformare i numeri da doppia a singola precisione e viceversa:

- `single(x)`: converte un numero in precisione singola.
- `double(x)`: converte un numero in precisione doppia.

Capitolo 4

Comandi di output

MATLAB dispone di due principali comandi di output a video: `disp` e `frpint`.

4.1 Output su video

4.1.1 Il comando `disp`

Il comando `disp` serve per visualizzare una stringa o il valore di una variabile.
Le stringhe che si vogliono visualizzare devono essere incluse tra due apostrofi semplici `'`.

Esempio di utilizzo con una stringa semplice:
```
» disp('Questa stringa verrà visualizzata a video')
```

Per visualizzare il valore di una variabile è necessario utilizzare la funzione `num2str(x)` che converte il valore di `x` in stringa. Inoltre per concatenare le due stringhe bisogna trattare la loro unione come un vettore.

Esempio di utilizzo con una stringa ed una variabile:
```
» disp(['Il valore di pi è:  ', num2str(pi), 'in formato short.'])
```

4.1.2 Il comando `fprint`

Questo comando server per visualizzare un insieme di dati di output con un certo formato. Esso inoltre ha una gestione migliore della concatenazione tra stringhe e numeri.

Esempio di utilizzo con una stringa semplice:
```
» fprint('Questa stringa verrà visualizzata a video')
Questa stringa verrà visualizzata a video»
```

All'interno di esso possono essere usati diversi caratteri speciali:

- `\t`: viene usato per inserire una tabulazione verso destra.
- `\n`: viene usato per inserire una nuova riga.

Esempio di utilizzo con una caratteri speciali:
```
» fprint('Questa stringa verrà visualizzata a video \n')
Questa stringa verrà visualizzata a video
»
```

E' anche possibile inserire numeri all'interno di una stringa, e definire la loro visualizzazione mediante la seguente struttura:

`%3$0-12 .5bu`

Dove i campi hanno il seguente significato (quelli in rosso sono quelli obbligatori:

1. %: simbolo obbligatorio per identificare che si vuole rappresentare un numero.
2. 3$: identificatore della posizione dell'argomento nella funzione di input.
   N.B. Obbligatorio se si vogliono inserire più numeri in una stringa.
3. 0-: flags, possono essere zero o più tra i seguenti:
   - '-': giustifica il testo a sinistra.
   - '+': stampa sempre il segno (+ o -) per qualsiasi valore.
   - ' ': inserisce uno spazio bianco prima del valore.
   - '0': inserisci degli 0 per riempire la lunghezza del campo.
   - '#': modifica la conversione numerica selezionata.
4. 12: lunghezza del campo. Indica il numero minimo di caratteri da stampare.
5. .5: precisione.
   - Per %f (floating point) o %e (esponenziale), indica il numero di cifre da tenere dopo la virgola.
   - Per %g (il più compatto tra %f e %e), indica il numero di cifre significative da considerare.
6. b: sottotipo.
7. u: carattere di conversione.

Esempio di utilizzo con una caratteri speciali:

```
» a = 10.123456789;
» b = 15.123456789;
» fprint('La variabile b vale %2$1.5f, mentre a vale %1$2.5e \n', a, b)
La variabile b vale 1.51235e+01, mentre a vale 10.12346
»
```

4.2 Output su file

All'interno di MATLAB è possibile anche eseguire stampe su file. Per fare ciò è sufficiente fare come segue.

1. Creare una variabile che contenga il file aperto tramite il comando `fopen('file_name', 'mode')`.
   Esempio:
   `fileID = fopen('file.txt', 'w');` // 'w' sta per 'write' (scrittura)
2. Eseguire la stampa sul file tramite il comando `fprintf(file_id, espressione [, variabili])`.
   Esempio:
   `fprintf(fileID,'%6s %12s \r\n','x','exp(x)');`
3. Chiudere il file tramite il comando `fclose(file_id)`.
   Esempio:
   `fclose(fileID);`

Capitolo 5

Le funzioni definite dall'utente

5.1 Definire una funzione

In MATLAB un utente può definire una funzione scrivendo un M-file, ovvero un file con estensione *.m*.
Per creare una funzione la sintassi da utilizzare all'interno del file *.m* è la seguente:

```
function [y1, ..., yN] = myfunc(x1, ..., xN)
```

Dove:

1. `function` è una keyword obbligatoria.
2. `y1, ..., yN` sono i parametri di output della funzione, che possono avere nomi arbitrari.
3. `myfunc` è il nome della funzione che si vuole creare.
4. `x1, ..., xN` sono i parametri di input della funzione, che possono avere nomi arbitrari.

Una volta scritta questa intestazione il corpo della funzione viene scritto sotto di essa.
Per richiamare la funzione definita, è sufficiente utilizzare la stessa segnatura senza la keyword `function`.

N.B. Si è soliti salvare la funzione in un file *.m* con lo stesso nome della funzione.

Capitolo 6

Istruzioni condizionali, cicli for e while

Come all'interno di molti linguaggi di programmazione, anche nel linguaggio MATLAB è possibile utilizzare le istruzioni condizionali e i cicli for e while.

6.1 Istruzioni condizionali

L'istruzione condizionale principalmente utilizzata è l'istruzione `if`:

```
if  espressione
        istruzioni
elseif  espressione
        istruzioni
else
        istruzioni
end
```

Oltre all'istruzione `if` esiste anche quella di `switch`:

```
switch  espressione_di_switch
    case  espressione_di_case
            istruzioni
    case  espressione_di_case
            istruzioni
                ...
    otherwise
            istruzioni
end
```

Gli operatori di confronto sono i seguenti:

- `==` : uguale.
- `~=` : non uguale.
- `<` : minore.
- `>` : maggiore.
- `<=` : minore o uguale.
- `>=` : maggiore o uguale.

Inoltre più espressioni logiche possono essere combinate tra loro mediante i seguenti:

- `&&` : and.
- `||` : or.
- `~`: not
- `&` : and componente per componente.
- `|` : or componente per componente.

6.2 Cicli `for` e `while`

6.2.1 Il ciclo `for`

Il ciclo `for` itera una porzione di codice, al variare di certi indici. La sua scrittura è la seguente:

```
for  variabile = vettore
          istruzioni
end
```

Esempio:

```
» s=0;
for  j = 1:10
     s=s+j;
end
```

6.2.2 Il ciclo `while`

Il ciclo `while`, a differenza di quello `for`, viene utilizzato quando **non è noto a priori** il numero di volte che il ciclo dovrà iterare.
La sua scrittura è la seguente:

```
while  espressione_logica
          istruzioni
end
```

Esempio:

```
» s=1;
while  s>0
     s=rand(1)-0.5;
end
```

Capitolo 7

I vettori

All'interno di MATLAB vengono spesso utilizzati i **vettori**, a tal punto che quasi tutte le funzioni e le operazioni disponibili sono definite anche su essi. Vediamo quindi come poter creare e trattare i vettori.

7.1 Creare un vettore riga

Per creare un vettore **riga** è sufficiente utilizzare la seguente segnatura:

```
v = [x1 ...  xN]
```

dove `x1, ..., xN` sono le componenti che si vogliono inserire nel vettore.
Esempio:

```
» v = [1 2 3]
v =
   1    2    3
»
```

7.2 Aggiungere elementi ad un vettore

Dato `v = [x1 ...  xN]` un vettore qualsiasi, per aggiungere elementi a v si utilizza la seguente scrittura:

```
v = [v xN+1 ...  xM
```

dove `xN+1, ..., xM` sono le componenti aggiuntive che si vogliono inserire.
Esempio:

```
» v = [1 2 3];
» v = [v 9 10]
v =
   1    2    3    9    10
»
```

7.3 Selezionare gli elementi di un vettore

Per selezionare il **j-esimo** elemento di un vettore `v = [x1 ... xN]` qualsiasi si utilizza la seguente segnatura:

`v(j)`

Esempio:

```
» v = [5 4 9]
v =
   5    4    9
» v(2)
ans =
   4
```

Per selezionare invece l'**ultima** componente di un vettore si utilizza

`v(end)`

Esempio:

```
» v = [5 4 9]
v =
   5    4    9
» v(end)
ans =
   9
```

Per conoscere la lunghezza di un vettore, il comando da utilizzare è:

`length(v)`

Esempio:

```
» v = [5 4 9]
v =
   5    4    9
» length(v)
ans =
   3
```

7.4 Creare un vettore colonna

Per creare un vettore **colonna** si usa la seguente segnatura:

`v = [x1 ; ... ; xN]`

dove `x1, ..., xN` sono le componenti da inserire nel vettore.
Alternativamente, si può creare un vettore colonna partendo da un vettore riga tramite la funzione di **trasposizione**.
Sia `x = [x1 ... xN]` un vettore riga, allora

$$y = v'$$

crea il vettore `y = [x1 ; ... ; xN]`, ovvero la trasposizione di x.

7.5 Creare un vettore equi-spaziato

Per creare un vettore `v` equi-spaziato in MATLAB è disponibile il seguente comando:

```
v = (a:h:b)
```

Dove:

- `a` è il punto iniziale.
- `h` è la spaziatura.
- `b` è il punto finale.

Se invece è noto il punto di inizio `a`, il punto di fine `b` e il numero di punti totali `N` si può utilizzare il comando seguente:

```
v = linspace(a:b:N)
```

7.6 Operazioni con i vettori

All'interno di MATLAB sono definite le seguenti funzioni elementari con i vettori:

- `+` : addizione
- `-` : sottrazione
- `.*` : prodotto puntuale
- `./` : quoziente puntuale
- `.^`: potenza puntuale

Oltre ad esse sono definite anche tutte le funzioni elementari viste con gli scalari nella **Sezione 2.3**.

Capitolo 8

Disegnare le funzioni

8.1 Il comando `plot`

All'interno di MATLAB è possibile disegnare il grafico delle funzioni, attraverso il comando `plot`. La segnatura di questo comando è la seguente:

```
plot(X, Y)
```

Questo crea un grafico 2D dei dati in `Y` contro i corrispondenti dati in `X`.

- Se `X` e `Y` sono entrambi vettori, devono avere lunghezza uguale. Disegnerà Y contro X.
- Se `X` e `Y` sono entrambi matrici, allora devono avere dimensione uguale. La funzione disegnerà le colonne di `Y` contro le colonne di `X`.
- Se uno tra `X` e `Y` è un vettore e l'altro una matrice, allora la matrice deve avere dimensioni tali per cui una delle sue dimensioni è uguale alla lunghezza del vettore.
  Se il numero di righe della matrice è uguale alla lunghezza del vettore, allora `plot` disegnerà ogni colonna della matrice contro il vettore. Se il numero di colonne della matrice è uguale alla lunghezza del vettore, disegnerà invece ogni riga della matrice contro il vettore.
  Se la matrice è quadrata, allora disegnerà ogni colonna contro il vettore.
- Se uno tra `X` e `Y` è uno scalare e l'altro è una matrice o un vettore, allora disegna dei punti. Per vedere tali punti bisogna però specificare un simbolo di marcatura, per esempio, `plot(X, Y, 'o'`.

Oltre a questo, con il comando

```
plot(X, Y, LineSpec)
```

è possibile impostare delle lo stile, il tratto e il colore della linea.

Con

```
plot(X1,Y1,...,Xn,Yn)
```

è invece possibile disegnare diverse coppie `X`, `Y` utilizzando gli stessi assi per le varie linee.

N.B. Per vedere tutte le opzioni della funzione `plot` è utile il comando `help plot`.