INTRODUZIONE A MATLAB

M.R. Russo

Università degli Studi di Padova
Dipartimento di Matematica Pura ed Applicata

A.A. 2010/2011

INDICE

- Gestione stringhe
- Gestione output avanzata
- Gestione input avanzata
- Salvataggio e richiamo dati

Stringhe

- Le stringhe sono matrici i cui elementi sono caratteri (es. a-z, A-Z, ...)
- Le stringhe sono racchiuse tra apici (' ')
- Sono tipicamente utilizzate per visualizzare il nome delle variabili ed il relativo valore.
- Si possono manipolare facilmente ed è possibile costruire frasi concatenando tra di loro delle matrici.

Creazione di una stringa

```
>> % stringa tra apici
>> nome = 'Mario'
nome =
Mario

>> % stringa tra apici
>> cognome = 'Rossi'
cognome =
Rossi
```

Concatenazione di stringhe

Uso operatore delle matrici

```
>> nc = [nome,' ',cognome]
nc =
Mario Rossi
```

Uso operatore dei vettori

```
>> length(nc)
ans =
    11
>> nc(7:11)
ans =
Rossi
```

Concatenazione di stringhe

```
>> a='Mi'
a =
Mi
>> b='chiamo'
b =
chiamo
>> c='Roberto'
c =
Roberto
>> d=[a,' ',b,' ',c]
d =
Mi chiamo Roberto
```

Concatenazione di stringhe

```
>> whos
  Name      Size              Bytes  Class

  a         1x2                   4  char array
  b         1x6                  12  char array
  c         1x7                  14  char array
  d         1x17                 34  char array
```

Matlab utilizza 2 byte per ogni carattere.

Da numero a stringa

Il comando num2str converte un numero in una stringa

```
>> msg1 = [’There are ’,num2str(100/2.54),’ inches in a meter’]
msg1 =
    There are 39.3701 inches in a meter
>> disp(['pi = ',num2str(pi)]);
pi = 3.1416
>> disp(['num. iter = ',num2str(i),' valore approssimato = ',num2str(x(i))]);
num. iter = 9 valore approssimato = 0.5
```

Funzioni per la manipolazione di stringhe

abs	Converte una stringa nel corrispondente valore numerico
isstr	vero se la variabile è una stringa
setstr	converte valore numerico in stringa
str2mat	Crea una matrice testo
lower	converte in minuscole
upper	converte in maiuscole
strcmp	confronta 2 stringhe
int2str	converte intero in stringa
num2str	converte numero in stringa
sprintf	converte numero in stringa (come C)
str2num	converte stringa in numero
sscanf	converte stringa in numero (come C)
dec2hex	conversione decimale-esadecimale
hex2dec	conversione esadecimale-decimale
hex2num	conversione esadecimale a floating point
base2dec	conversione da base B a decimale
dec2base	conversione da decimale a base B

Gestione Output

Per visualizzare un insieme di dati di output con un certo formato si utilizzano i comandi fprintf e sprintf

fprintf(*fid*,'*formato*',*variabili*)

- *formato* è una stringa di testo che tramite l'uso di caratteri speciali indica il tipo di formato dell'output,
- *variabili* è una lista opzionale di variabili, separate da virgole, che hanno un corrispondente all'interno della stringa formato,
- *fid* è un identificatore opzionale del file al quale l'output è inviato.

Gestione Output su video

Il comando sprintf ha la seguente sintassi

str=sprintf('*formato*',*variabili*)

indirizza su una stringa di testo di nome *str* le variabili indicate, con il formato definito.
Per visualizzare il tutto si usa il comando

>>disp(str)

Gestione Output

Il '*formato'* è una stringa che contiene i caratteri che si vogliono visualizzare e, nelle posizioni in cui si vuole venga inserito il valore, deve essere indicato uno dei formati preceduti dal carattere %. Tali codici di formati sono abitualmente seguiti da due interi separati da un punto (%6.3). Il primo numero indica quante colonne si desiderano impegnare in uscita ed il secondo il numero di cifre della parte frazionaria. I formati \n \t servono per organizzare le linee dell'output.

Gestione Output

Esempi:

```
>> n=4; x=1234.5467; err=1.345677e-4;
>> st=sprintf('Iter = %4.0f \t Valore x = %10.7f
\t Errore = %9.2e', n, x, err);
>> disp(st);
Iter =   4   Valore x = 1234.5467000 Errore =
  1.35e-004
>>disp(['lunedi'' ','martedi'' ','mercoledi'' '])
lunedi' martedi' mercoledi'
>> disp(['lunedi''   ';'martedi''  ';'mercoledi''']);
lunedi'
martedi'
mercoledi'
```

Codici di formato

La seguente tabella mostra alcuni codici di formato e relativo significato.

Codice di formato	Significato
%s	formato stringa
%d	formato decimale
%g	seleziona il formato per numeri interi, fixed point o esponenziali
%f	formato fixed point del numero (esempio 1343.675432)
%e	formato esponenziale del numero (esempio 1.34376e+003)
\n	inserisce carattere di ritorno a capo
\t	inserisce carattere di tabulazione

Gestione Output: esempi

```
% visualizza i valori di una funzione
t=0 :pi/10 :2*pi ;
y=2*sin(t).*cos(t) ;
funz=[t;y];
fprintf('-----------------------------------\n');
fprintf('Valori di sin(t)*cos(t) tra 0 e 2*pi \n');
fprintf('-----------------------------------\n');
fprintf('%4.2f %10.6f\n',funz) ;
```

% Si osservi che MatLab accede alla matrice funz per colonne, quindi il contenuto di funz in uscita sarà trasposto, la prima riga (t) sara' visualizzata come prima colonna.

Gestione Output: esempi

```
-------------------------------------
Valori di sin(t)*cos(t) tra 0 e 2*pi
-------------------------------------
0.00   0.000000
0.63   0.951057
1.26   0.587785
1.88  -0.587785
2.51  -0.951057
3.14  -0.000000
3.77   0.951057
4.40   0.587785
5.03  -0.587785
5.65  -0.951057
6.28  -0.000000
```

Gestione Output: esempi

Il seguente codice

```
% tabsincos.m
% costruisce la tabella dei valori di seno e coseno
% in 5 punti equispaziati di (0,pi)
x=linspace(0,pi,5);
s=sin(x);
c=cos(x);
disp('-----------------------------------');
fprintf('k \t x(k) \t sin(x(k)) \t cos(x(k))');
disp('-----------------------------------');
fprintf('%d\t %3.2f\t %8.5f\t %8.5f\n', [1:5;x;s;c]);
```

Gestione Output: esempi

Produce la seguente tabella

k	x(k)	sin(x(k))	cos(x(k))
1	0.00	0.00000	1.00000
2	0.79	0.70711	0.70711
3	1.57	1.00000	0.00000
4	2.36	0.70711	-0.70711
5	3.14	0.00000	-1.00000

Gestione Output su file

Per scrivere su file un insieme di dati di output con un certo formato si utilizzano i comandi
fopen, fprintf, fclose:

fid=fopen('*stringa*','w')
fid è una variabile che identifica il file e '*stringa*'
definisce il nome del file, apre il file in scrittura ('w')

fprintf(fid,'*formato*',variabili)
dove fid è l'identificatore del file di uscita, scrive nel file fid il valore delle variabili con il formato assegnato.

fclose(fid)
dove fid è la variabile che identifica il file, chiude il file.

Gestione Output

```
>> A = [ 1 4 7
         2 5 8
         3 6 9];
>> fid = fopen('file.dat','w');
>> fprintf(fid,'%5.0f', A);
>> fclose(fid);
>> type file.dat

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

% La matrice A viene memorizzata per colonne e le sue componenti stampate su di una riga

Gestione Output

Per gestire in uscita matrici di dimensioni *n x m*

```
>> A=[1 4 6 5 8; 6 7 4 3 1; 8 7 3 2 1];
>> [n,m] = size(A);
>> fid = fopen('matrice.out','w');
>> for i=1:n
>>     for j=1:m
>>         fprintf(fid,'%8.4f\t',A(i,j));
>>     end
>> fprintf(fid,'\n');
>> end
>> fclose(fid);
```

Gestione Input da tastiera

Il comando
variabile=input('*stringa*')
attende un dato in ingresso da tastiera, dopo aver visualizzato la stringa di caratteri *stringa*, e lo assegna a variabile. Utilizzabile per assegnare una sola variabile.

```
>> nmax = input(’ Numero massimo di iterazioni ’)
Numero massimo di iterazioni 40
nmax =
40
>>nomefile= input('Nomefile = ')
Nomefile = 'dati.in'  % Per aprire file deve essere una stringa
nomefile =
dati.in
```

Gestione Input

Per leggere da un file un insieme di dati si utilizzano fopen, fscanf, fclose:

```
fid=fopen('stringa')        % Apre il file

var=fscanf(fid,'formato',size)
```

fid identifica il file, legge tutti i dati contenuti nel file identificato da fid, convertendoli in base al formato specificato e memorizzandoli nella variabile var, size indica la dimensione della variabile var e può essere scritto come

```
nval            % legge nval numeri, li memorizza in vet.colonna
[nrighe, ncol]  % legge nrighe*ncol numeri memorizzandoli
                % in una matrice che ha tale dimensione

fclose(fid)                 % Chiude il file
```

Gestione Input: esempi

```
>> type dati.in
  0.10 2
  0.20 3
  0.30 4
>> f1 = fopen('dati.in');
>> vet = fscanf(f1,'%g %g',2)        % legge la prima riga
vet =
      0.1000
      2.0000
>> vet1 = fscanf(f1,'%g %g',2)       % legge la seconda riga
vet1 =
      0.2000
      3.0000
>> vet2 = fscanf(f1,'%g %g',2)       % legge la terza riga
vet2 =
      0.3000
      4.0000
>> fclose(f1)
```

Gestione Input: esempi

```
>> type FILEOUT.DAT      % contiene una matrice 3x3
     1    2   3
     4    5   6
     7    8   9
>> f4 = fopen('FILEOUT.DAT');
>> A = fscanf(f4,'%g %g %g',[3 inf]);
>> fclose(f4)
```

Legge per righe 3 elementi fino alla fine del file, A sarà una matrice di dimensione 3 righe x 3 colonne, bisogna trasporre la matrice per avere le componenti corrispondenti a quelle della matrice del file.

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

Salvataggio e richiamo dati

Spesso si hanno dati raccolti con altri programmi che si vorrebbero analizzare mediante MatLab. Il metodo più semplice consiste nell'avere questi dati in un file in formato ASCII, come nel prossimo esempio:

```
1    2          -5
2    0.2500     -9
3    0.0740     -23
4    0.0310     -53
5    0.0160     -105
6    0.0090     -185
7    0.0050     -299
8    0.0030     -453
9    0.0020     -653
10   0.0020     -905
```

Salvataggio e richiamo dati

Questi dati possono essere salvati su un file (per esempio dati.dat) e caricati in MatLab con il comando load.

In questo esempio, in MatLab viene creata una matrice dati di dimensione 10x3

```
>> load dati.dat
>> whos
  Name        Size              Bytes  Class

  dati       10x3               240  double array

Grand total is 30 elements using 240 bytes
```

Salvataggio e richiamo dati

```
>>M= load('dati.dat')
M =

    1.0000    2.0000   -5.0000
    2.0000    0.2500   -9.0000
    3.0000    0.0740  -23.0000
    4.0000    0.0310  -53.0000
    5.0000    0.0160 -105.0000
    6.0000    0.0090 -185.0000
    7.0000    0.0050 -299.0000
    8.0000    0.0030 -453.0000
    9.0000    0.0020 -653.0000
   10.0000    0.0020 -905.0000
```

Salvataggio e richiamo dati

Tutte le variabili definite o calcolate in una sessione di lavoro possono essere salvate mediante il comando save

```
>> save                          % salva tutte le variabili nel
 Saving to: Matlab.mat           % file Matlab.mat

>> save prova                    % salva nel file prova.mat
```

Salvataggio e richiamo dati

Per salvare soltanto alcuni valori (per esempio una tabella di dati) e non tutta la sessione di lavoro si utilizza il comando save con la sintassi

save *nomefile variabili formato*

formato è un parametro opzionale.
Il formato -ascii consente di salvare il file in modalità testo; se tale parametro è omesso il file viene salvato in formato binario.
Se *nomefile* è privo di estensione, allora il file viene automaticamente salvato con l'estensione .mat

Salvataggio e richiamo dati

Esempio:

```
% salvatabella.m
n=input('fornisci il numero dei valori:');
x=linspace(0,pi,n);
s=sin(x);
c=cos(x);
v=(1:n);
save tabella.dat v x s c -ascii
```

Salvataggio e richiamo dati

Per visualizzare una tabella di valori precedentemente salvata con il comando save si utilizza il comando load

load *nomefile formato*

Il formato -ascii è obbligatorio se nomefile è privo di estensione.

```
% veditabella.m
load tabella.dat
A=tabella;
disp('-----------------------------------');
fprintf('k\t x(k)\t sin(x(k))\t cos(x(k))\n');
disp('-----------------------------------');
fprintf('%d\t %3.2f\t %8.5f\t %8.5f\n',A);
```