Laboratorio di Matlab

Alessandro Formaglio

Dipartimento di Ingegneria dell'Informazione,

Università di Siena

alex@dii.unisi.it

http://www.dii.unisi.it/∼control/MatLab/LabMatlab.html

9 Luglio 2009

Tipi di dato multidimensionali

cellarray

- Matrici i cui elementi sono *array*; ogni elemento puo' avere dimensioni diverse.
- Definizione di un *cell array*:

  `C = {A1 A2;B1 B2}` assegna alla variabile `C` un cellarray di 2×2 elementi corrispondenti alle matrici `A1,A2,B1,B2`
- Indicizzazione:

  `D = C{i,j}` assegna alla variabile `D` l'elemento (`i,j`) di `C`
- Concatenazione orizzontale e verticale:

  `D = [C {A3; B3}]` assegna alla variabile `D` la concatenazione orizzontale di `C` con il cellarray `{A3;B3}`

  `D = [C;{A3 B3}]` assegna alla variabile `D` la concatenazione verticale di `C` con il cellarray `{A3 B3}`

strutture

- Matrici i cui elementi sono accessibili attraverso *campi testuali*; ogni campo può essere di tipo diverso e con dimensioni diverse
- Definizione di una struttura vuota (non obbligatoria):
  `S = struct('nome',{},'cognome',{},'matricola',{});`
  assegna alla variabile `S` una struttura vuota caratterizzata dai tre campi:
  `nome  cognome  matricola`
- Assegnazione di valori ai campi:

```
>> S.nome = 'mario';
>> S.cognome = 'rossi';
>> S.matricola = 652;
```

Attenzione: se la struct è vuota i primi elementi vanno inseriti come `S(1).nome`, `S(1).cognome` e `S(1).matricola`

```
>> S
S =
        nome: 'mario'
     cognome: 'rossi'
   matricola: 652
```

- Restituzione valori:

  la notazione `struttura.campo` restituisce il valore della struttura per il campo specificato:

```
>> a = S.matricola
a =
     652
```

M-files

- Matlab può eseguire sequenze di comandi da *file*. Questi file sono chiamati M-file perché hanno estensione '**.m**'.

- La maggior parte del codice che produrrete sarà creato e gestito tramite M-files.

- Due tipi di M-files:
  - *script files*
  - *function files*

script files

- Uno *script file* consiste di una sequenza di comandi Matlab.

- Se creo un file `prova.m`, e lo eseguo come un comando al prompt invocando:
  `>> prova` (N.B. il nome del file senza l'estensione)
  saranno eseguite tutte le istruzioni contenute nel file `prova.m`.

- Le variabili generate da uno script file sono **globali** e quindi andranno a finire direttamente nel workspace della corrente sessione di Matlab, con le *naturali conseguenze*.

- Un M-files può richiamare un'altro M-files, oppure può richiamare se stesso in modo *ricorsivo*.

gestione M-files

stringhe: `s = 'hello world';` quello che sta tra gli apici definisce la stringa che viene assegnata ad `s`

```
>> s
s = hello world
```

disp: i comandi `disp('hello world'), disp(s);` producono lo stesso risultato, cioè mostrano a schermo una stringa:

```
hello world
```

error: il comando `error(s);` mostra a schermo la stringa `s` ed interrompe l'esecuzione dell'M-file

input: il comando `d = input(s);` mostra a schermo la stringa `s` ed aspetta la digitazione di un valore fino a che non viene premuto il tasto invio ↩. Tale valore viene assegnato alla variabile `d`. Per input di tipo stringa il comando diventa `d = input(s,'s');` (cfr. `scanf` del linguaggio C)

stringhe

- `S = 'sequenza di caratteri'` definisce un vettore di caratteri assegnato alla variabile `S`.
- Funzioni correlate:
  - `S = char(X);`
    ritorna una stringa di caratteri associati ai valori di `X`, in codifica ASCII.
    Es.

```
>> char(123)
   "{"
```

  - `X = double(string);`
    ritorna i valori numerici associati alla stringa S, in codifica ASCII
    Es.

```
>> double('{')
   "123"
```

- `S = [S1 S2 ... Sn];`
  concatenazione orizzontale di n stringhe

- `S = strcat(S1,S2,...,Sn);`
  concatenazione orizzontale di n stringhe

- `S = strvcat(S1,S2,...,Sn);`
  concatenazione verticale di n stringhe

- `S = {S1 S2 ... Sn};`
  crea un vettore le cui componenti sono n stringhe

- `ischar(S);`
  restituisce 1 se `S` è una stringa, 0 altrimenti

- `S(v_char);`

  restituisce i caratteri della stringa S nelle posizioni indicate dal vettore `v_char`

- `S = sprintf(format,A);`

  genera stringhe (cfr. `printf` del linguaggio C).

  Es. 1 :

```
>> s = sprintf('La matrice ha dimensioni %dx%d.',2,3)
   s = La matrice ha dimensioni 2x3
```

  Es. 2 :

```
>> s = sprintf('%s\n','hello','word')
   s =
       hello
       word
```

Esercizio

Creare uno *script file* che richiede:

1. Digitazione in ingresso del nome e cognome di 4 persone (usare `input`) e salvarli in una struttura indicizzata con i campi `nome` e `cognome`.
2. Concatenare i cognomi verticalmente
3. Concatenare le iniziali delle persone orizzontalmente, seprate da virgole
4. Plottare a schermo i risultati preceduti da spiegazioni testuali

Istruzioni di controllo di flusso

for

while

if

case

for

sintassi:

```
. . .
for variabile = espressione
istruzioni
end
. . .
```


Esempio 1:

```
n = 10;
x = [];
for i = 1:n
    x = [x,i^2];
end
```

Esempio 2:

```
passo = 0.1;
t = 0.1:passo:100;
f = log(t);

for i = 2:length(t)
    dfdt(i-1) = (f(i)-f(i-1))/passo;
end
```


while

sintassi:

```
. . .
while relazione
istruzioni
end
. . .
```


Esempio 3:

```
n = 0;
a = 6;
while n < exp(a)
   n = n+1;
end
```

Esempio 4:

```
n = 6;
A = magic(n);
B = zeros(n);
C = [];
k = 6;
while k > 0
    B(6-k+1,:) = A(k,:);
    C = [C A(:,k)];
    k = k-1;
end
```

if

sintassi:

```
. . .
if relazione 1
istruzioni 1
elseif relazione 2
istruzioni 2
else
istruzioni 3
end
. . .
```


Esempio 5:

```
k = input('Inserisci un numero maggiore di cento :');
str = ischar(k);
if k > 100 & ~str
    disp('bravo!');
    delta = 100-k
elseif k > 50 & ~str
    disp('ci sei quasi!')
    delta = 100-k
elseif k > 0 & ~str
    disp('mica tanto vicino!');
    delta = 100-k
elseif str
    disp('ho detto NUMEROOOOO');
else
    disp('allora sei duro!');
    delta = 100-k
end
```

case

sintassi:

```
. . .
switch espressione di switch
case case valore 1
istruzioni
case {case valore 2, case valore 3, case valore 4,...}
istruzioni
otherwise
istruzioni
end
. . .
```


Esempio 6:

```
k = input('Inserisci un numero da 1 a 3:');
switch k
case 1
    disp('bravo!');
case {2,3}
    disp('esagerato!')
otherwise
    disp('Ho detto da 1 a 3 !!!');
end
```

Esercizio 5

1. Generare una matrice random 100×100 utilizzando il comando `rand`

2. Estrarne la diagonale principale utilizzando un doppio ciclo `for` ed assegnarla ad un vettore `v1`

3. Estrarne la contro diagonale utilizzando `while` ed assegnarla ad un vettore `v2`

Function

- Sono script file con dichiarazione di funzione

- Prendono parametri in *ingresso* e restituiscono parametri in *uscita*:

```
function [o1,o2,...] = function_name(i1,i2,...);
...
istruzioni
...
```

- Al nome del file viene associato un comando Matlab

  Attenzione: alla discrepanza tra *nome file* e *nome function*!!!

Esempio:

```
function [s,d] = sommadiff(x,y);
s = x+y;
d = x-y;
```

Se il file viene salvato come `sommadiff.m`, posso utilizzare tale funzione nel seguente modo:

```
>> [a,b] = sommadiff(3,5)
>> a = 8
>> b = -2
```

Attenzione: perchè Matlab riconosca il nome di una function come comando, è necessario che il file si trovi nella *current directory*

Esercizio 6

Creare una funzione che dato in ingresso un numero $n \in \mathbb{N}$, produce l'n-esimo termine della serie di Fibonacci.

Esercizio 7

Scrivere una funzione che prende in ingresso una matrice M quadrata e restituisce il determinante di M (naturalmente la funzione Matlab predefinita `det` può essere utilizzata solo per controllare i risultati). La seguente formula indica come calcolare ricorsivamente il determinante di una matrice:

$$det(M) = \sum_{j=1}^{n} (-1)^{i+j} m_{ij} det(M_{ij})$$

dove M_{ij} è la matrice M decurtata della i-esima riga e della j-esima colonna.