



Laboratorio di Matlab

Rudy Manganelli

Dipartimento di Ingegneria dell'Informazione,
Università di Siena

manganelli@dii.unisi.it

<http://www.dii.unisi.it/~control/MatLab/LabMatlab.html>

Febbraio-Marzo 2009

Start up

MATLAB = Matrix Laboratory

- aprire e chiudere Matlab
- interfaccia grafica
- directory path:

```
>> cd
```

```
>> cd ..
```

```
>> cd ('directory_path')
```

```
>> cd directory_path
```

oppure usare il *current directory browser*



- assegnazione di uno scalare

```
>> a = 13;
```

```
>> a = 13
```

```
a =
```

```
    13
```

- workspace

- visualizzazione delle variabili del workspace:

- >> who → per visualizzare tutte le variabili

- >> whos → per visualizzare tutte le variabili con descrizione di struttura

- eliminazione delle variabili dal workspace:

- >> clear → per eliminare tutte le variabili dal workspace

- >> clear a; → per eliminare la variabile a dal workspace

- oppure usare il *workspace browser*

- **ans:**

`ans` è il nome della risposta più recente (*most recent answer*).

Variabile che vive quindi a tutti gli effetti nel workspace!

- **richiamo comandi:**

`↑` richiama l'ultimo comando digitato

`string + ↑` richiama l'ultimo comando digitato che inizia con tale stringa

oppure usare il *command history browser*

- **salvataggio dati:**

`save filename var` crea il file di nome *filename.mat* e salva sul file la variabile `var` ed i dati ad essa relativi

`save('filename', 'var1', 'var2', ...)` crea il file di nome *filename.mat* e salva sul file le variabili ed i dati ad esse relative



- caricamento dati:

`load 'filename.mat'` carica nel workspace tutti i dati contenuti nel file *filename.mat*

(in alternativa si può anche usare il comando `load filename.mat`)

- cancellazione dati:

`delete 'filename.mat'` cancella dalla directory corrente il file *filename.mat*

Attenzione: cancella qualsiasi file!!!

- `clc`:

`clc` è il comando per pulire la finestra di comando.

Vettori e Matrici

```
>> V = [13 4 6] vettore riga
```

```
V =
```

```
    13     4     6
```

```
>> V = [13; 4; 6] vettore colonna
```

```
V =
```

```
    13
```

```
     4
```

```
     6
```



```
>> M = [13 4 6; 10 1 9; 17 18 11] matrice
```

oppure

```
>> M = [13 4 6; ↵  
10 1 9; ↵  
17 18 11]
```

M =

```
    13     4     6  
    10     1     9  
    17    18    11
```

Attenzione: *i nomi delle variabili sono case sensitive: si distingue cioè tra lettere maiuscole e minuscole!!!*

Esercizio: Provare a generare matrici, vettori, scalari ...

OPS: Operatori e caratteri speciali

operatori aritmetici

operatore	scalare	matrice
+	$a + b$	$(A + B)_{ij} = A_{ij} + B_{ij}$
-	$a - b$	$(A - B)_{ij} = A_{ij} - B_{ij}$
*	$a * b$	$A * B$
/	$a/b = \frac{a}{b}$	$A/B = A * B^{-1}$
\	$a \backslash b = \frac{b}{a}$	$A \backslash B = A^{-1} * B$
.*	non interessante	$(A .* B)_{ij} = A_{ij} * B_{ij}$
./	non interessante	$(A ./ B)_{ij} = \frac{A_{ij}}{B_{ij}}$
.\	non interessante	$(A . \backslash B)_{ij} = \frac{B_{ij}}{A_{ij}}$

Attenzione: *Prodotti matriciali righe per colonne: dimensioni concordi!!!*



operatori relazionali

- **Non esistono variabili booleane**
- **0 : FALSO**
- Tutto ciò che è $\neq 0$ è **VERO**

operatori	<code>==</code>	<code>~=</code>	<code>></code>	<code><</code>	<code>>=</code>	<code><=</code>
significato	$=$	\neq	$>$	$<$	\geq	\leq

- Ogni operatore restituisce **1** se la relazione è vera, **0** se è falsa.
- Tra matrici le dimensioni devono essere concordi, $(A \text{ op } B)_{ij} = A_{ij} \text{ op } B_{ij}$.
- Matrici e scalari, $(A \text{ op } b)_{ij} = A_{ij} \text{ op } b$

operatori logici

simbolo	&		~
significato	and	or	not

- $a \text{ op } b$ restituisce **1** se è vero, **0** se è falso
- Tra matrici le dimensioni devono essere concordi, $(A \text{ op } B)_{ij} = A_{ij} \text{ op } B_{ij}$.
- Matrici e scalari, $(A \text{ op } b)_{ij} = A_{ij} \text{ op } b$

Questi sono solo alcuni operatori. Per vedere la lista completa, digitare nel prompt dei comandi:

```
help ops
```

Esercizio: Provare ad utilizzare gli operatori ...

Funzioni matriciali

matrici elementari

- $A = \text{zeros}(m, n)$ matrice di 0 con m righe ed n colonne
- $A = \text{ones}(m, n)$ matrice di 1 con m righe ed n colonne
- $A = \text{eye}(n)$ matrice identità di dimensione $n \times n$

informazioni sulle matrici

- $[m, n] = \text{size}(A)$ ritorna le dimensioni m, n della matrice $A \in \mathbb{R}^{m \times n}$
- $M = \text{length}(A)$ ritorna la dimensione massima di $A \in \mathbb{R}^{m \times n}$
- $\text{isequal}(A, B)$ ritorna 1 se $A = B$, 0 altrimenti



manipolazione di matrici

- $A = \text{diag}(v)$;

$$A = \begin{pmatrix} v_1 & 0 & \cdots & 0 \\ 0 & v_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & v_n \end{pmatrix}$$

- $a = \text{min:passo:max}$

Definisce un vettore a di $n = \frac{\text{max}-\text{min}}{\text{passo}} + 1$ elementi
con $a(1) = \text{min}$, $a(n) = \text{max}$

- **sottomatrici**

$A(v_row, v_col)$ seleziona la sottomatrice corrispondente agli indici di riga e di colonna definiti rispettivamente nei vettori v_row e v_col

Ad es. $A(1:4, 2:3)$ seleziona la sottomatrice 4×2 che ha elementi nelle



righe 1 : 4 e colonne 2 : 3

- $B = A'$ assegna a B il trasposto di A, cioè $B = A^T$.

- **concatenazione orizzontale:**

$A = [B1, B2, \dots, Bn]$ oppure $A = [B1 \ B2 \ \dots \ Bn]$ assegna alla matrice A la concatenazione orizzontale delle matrici $B1, B2, \dots, Bn$

- **concatenazione verticale:**

$A = [B1; B2; \dots; Bn]$ assegna alla matrice A la concatenazione verticale delle matrici $B1, B2, \dots, Bn$

- **Esempio:**

```
>> B1 = [13 4 6];
```

```
>> B2 = [18 11 7];
```

```
>> A1 = [B1 B2]
```

```
A1 =
```



```
13  4  6  18  11  7
```

```
>> A2 = [B1;B2]
```

```
A2=
```

```
13   4   6
18  11   7
```

- **sotto-assegnazione:**

```
>> A2(1,:) = [1 1 1]
```

```
A2=
```

```
1   1   1
18  11  7
```

indicizzazione

$\mathbf{A}(i, j)$	Seleziona l'elemento di posto (i, j)
$\mathbf{A}(:, j)$	Seleziona tutta la colonna j – <i>esima</i> della matrice \mathbf{A}
$\mathbf{A}(i, :)$	Seleziona tutta la riga i – <i>esima</i> della matrice \mathbf{A}
$\mathbf{A}(i, \text{end})$	Seleziona l'ultimo elemento della riga i – <i>esima</i> della matrice \mathbf{A}
$\mathbf{A}(\text{end}, j)$	Seleziona l'ultimo elemento della colonna j – <i>esima</i> della matrice \mathbf{A}
$\mathbf{A}(:, \text{end})$	Seleziona l'ultima colonna della matrice \mathbf{A}
$\mathbf{A}(\text{end}, :)$	Seleziona l'ultima riga della matrice \mathbf{A}
$\mathbf{A}(i, :) = []$	Elimina la riga i – <i>esima</i> della matrice \mathbf{A}
$\mathbf{A}(:, j) = []$	Elimina la colonna j – <i>esima</i> della matrice \mathbf{A}
$\mathbf{A} = []$	Crea una matrice vuota di dimensioni 0×0



variabili speciali

variabile	significato
<code>ans</code>	risposta più recente
<code>pi</code>	π
<code>i</code>	unità immaginaria
<code>inf</code>	infinito
<code>NaN</code>	not a number, per operazioni indefinite (tipicamente $\frac{0}{0}$)

Per saperne di più ...

```
help elmat
```



funzioni elementari

trigonometriche ed iperboliche

<code>sin(x)</code>	<code>asin(y)</code>
<code>cos(x)</code>	<code>acos(y)</code>
<code>tan(x)</code>	<code>atan(y)</code>
	<code>atan2(y, x)</code>

<code>sinh(x)</code>	<code>asinh(y)</code>
<code>cosh(x)</code>	<code>acosh(y)</code>
<code>tanh(x)</code>	<code>atanh(y)</code>

Attenzione: nelle funzioni trigonometriche x è espresso in **radianti!!!**

Altre funzioni

istruzione	significato
<code>exp(x)</code>	e^x
<code>log(x)</code>	$\ln(x)$
<code>log10(x)</code>	$\log_{10}(x)$
<code>log2(x)</code>	$\log_2(x)$
<code>sqrt(x)</code>	\sqrt{x}
<code>xey</code>	$x \times 10^y$
<code>x^y</code>	x^y

istruzione	significato
<code>abs(x)</code>	$ x $
<code>imag(x)</code>	$\text{Im}(x)$
<code>real(x)</code>	$\text{Re}(x)$
<code>conj(x)</code>	\bar{x}
<code>sign(x)</code>	segno (x)
<code>floor(x)</code>	$\lfloor x \rfloor$
<code>ceil(x)</code>	$\lceil x \rceil$

Per saperne di più ...

`help elfun`

Help

- `>> help`
- `>> help \toolbox_name`
- `>> help ops`
- `>> help elmat`
- `>> help elfun`
- `>> help function_name` (ritorna la sintassi esatta)
- `help html` con browser.

Esercizio 1

1. Definire una matrice A , 3×5 di zeri (utilizzando la funzione `zeros`). Definire un vettore a di dimensione 1×5 e assegnarlo alla prima riga di A . Definire una matrice B come la trasposta di A ed estrarne la sottomatrice C , 3×3 composta dalla 2°, 3° e 4° riga di B .

2. Siano A, B quelle definite al punto 1. Definire una nuova matrice D , 5×3 diversa dalla matrice nulla. Effettuare le tre moltiplicazioni:

$$M1 = D * A$$

$$M2 = A * D$$

$M3$ = moltiplicazione elemento per elemento tra D e B .

Verificare con la funzione `size` che le dimensioni siano:

$$M1 \rightarrow 5 \times 5; M2 \rightarrow 3 \times 3; M3 \rightarrow 5 \times 3.$$

Salvare il workspace con le sole matrici $A, B, C, M1, M2, M3$.



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 \leftarrow . 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"
```



- $S = [S1 \ S2 \ \dots \ Sn]$;
concatenazione orizzontale di n stringhe

- $S = \text{strcat}(S1, S2, \dots, Sn)$;
concatenazione orizzontale di n stringhe

- $S = \text{strvcat}(S1, S2, \dots, Sn)$;
concatenazione verticale di n stringhe

- $S = \{S1 \ S2 \ \dots \ Sn\}$;
crea un vettore le cui componenti sono n stringhe

- $\text{ischar}(S)$;
restituisce 1 se S è una stringa, 0 altrimenti

- `iscellstr(S)` ;
restituisce 1 se S è un vettore di stringhe, 0 altrimenti

- `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 2

Creare uno *script file* che richiede:

1. Digitazione in ingresso di nome e cognome (usare `input`) e salvarli in una struttura con i campi `nome` e `cognome`.
2. Concatenare nome e cognome verticalmente e orizzontalmente nelle quattro combinazioni possibili assegnandole a quattro diverse variabili
3. Generare i due vettori dei codici ASCII relativi alle lettere del nome e cognome
4. Utilizzando la function predefinita `flipdim(v, 2)` rigirare questi ultimi e utilizzando `char` calcolarne le stringhe associate `s1` e `s2`.

5. In uscita il file deve fare il display di due frasi :

```
il mio nome = s1      il mio cognome = s2
```

6. fare in oltre la media dei codici dei caratteri nome e cognome e generare la stringa `s3` che contiene il carattere relativo.

