

Laboratorio 1 – Introduzione a Matlab[®] - Octave

©2009 - Questo testo (compresi i quesiti ed il loro svolgimento) è coperto da diritto d'autore. Non può essere sfruttato a fini commerciali o di pubblicazione editoriale. Non possono essere ricavati lavori derivati. Ogni abuso sarà punito a termine di legge dal titolare del diritto. This text is licensed to the public under the Creative Commons Attribution-NonCommercial-NoDerivs2.5 License (<http://creativecommons.org/licenses/by-nc-nd/2.5/>)

Matlab[®] è un software per il calcolo numerico che fornisce un ambiente per varie applicazioni scientifiche e ingegneristiche. Matlab[®] è un software commerciale a pagamento. Maggiori informazioni possono essere reperite all'indirizzo <http://www.mathworks.it/>.

GNU Octave è un linguaggio ad alto livello, principalmente pensato per il calcolo scientifico. Octave non possiede una interfaccia grafica di default come Matlab[®], ma è possibile installarla separatamente. Senza interfaccia l'inserimento dei comandi avviene da linea di comando. Octave è essenzialmente compatibile con Matlab[®], salvo per alcune piccole differenze che saranno ampiamente affrontate durante il corso. È un software libero che è possibile scaricare all'indirizzo <http://www.gnu.org/software/octave/>.

Tutte i laboratori possono indifferentemente essere svolti in Matlab[®] o Octave. Ulteriori informazioni relative ai software sono disponibili sul sito del corso, al seguente indirizzo:

<http://www2.mate.polimi.it:8080/CN/CNEA/>

Interfaccia Grafica di Matlab[®]. È costituita principalmente da quattro ambienti. Nel **workspace** sono rappresentate tutte le variabili memorizzate, il loro valore e tipo. La finestra **current directory** rappresenta una finestra sulla cartella in cui si sta lavorando e mostra tutti i file presenti nella cartella stessa. La **command history** contiene lo storico di tutti i comandi digitati. L'ambiente principale è la **command window** in cui vengono inseriti i comandi. Per quanto riguarda Octave, a meno che non si installi una interfaccia grafica, c'è solo la command window.

Matlab[®] è l'acronimo di **Matrix Laboratory**, per cui tutte le variabili in Matlab[®] sono considerate matrici. In particolare gli scalari sono considerati matrici 1×1 , i vettori riga sono matrici $1 \times n$, i vettori colonna sono matrici $n \times 1$, dove n è la lunghezza del vettore. Octave si comporta nello stesso modo.

In questo laboratorio vengono fornite le nozioni necessarie per cominciare ad usare Matlab[®] o Octave. Tutti i comandi presentati hanno la stessa sintassi sia in Matlab[®] che in Octave.

Assegnazione di scalari. Cominciamo con l'assegnare il valore 2.45 alla variabile **a**:

```
>> a = 2.45
a =
    2.45
```

assegniamo ora il valore 3.1 alla variabile **A**. Osserviamo che Matlab[®] fa distinzione tra le lettere maiuscole e le lettere minuscole.

```
>> A = 3.1
A =
    3.1
```

Le variabili sono sovrascrivibili, cioè se ora assegniamo ad **A** un nuovo valore:

```
>> A = 7.2
A =
    7.2
```

il precedente valore 3.1 viene definitivamente perso. Osserviamo che possiamo far seguire un comando da una virgola, senza rilevare nessuna differenza; tale virgola è però necessaria per separare più comandi scritti sulla stessa linea.

```
>> a = 1.2,
a =
    1.2
>> a = 1.7, a = 2.45
a =
    1.7
a =
    2.45
```

Se invece si fa seguire il comando da un punto e virgola, Matlab[®] non visualizzerà sulla finestra di comando il risultato dell'operazione; il punto e virgola può essere usato per separare due comandi sulla stessa riga.

```
>> a = 1.2;
>> a = 1.7; a = 2.45
a =
    2.45
```

Per sapere quali sono le variabili dell'utente attualmente in memoria si utilizza il comando:

```
>> who
Your variables are:
A  a
```

Per sapere quali sono le variabili in memoria definite dall'utente è anche possibile utilizzare il comando `whos`. Quest'ultimo, a differenza di `who`, mostra anche la dimensione, l'occupazione di memoria in numero di bytes e il tipo della variabile.

```
>> whos
Name      Size      Bytes  Class  Attributes

A         1x1         8  double
a         1x1         8  double
```

Le variabili possono essere cancellate utilizzando il comando `clear`. Possiamo ad esempio cancellare la variabile `A` digitando:

```
>> clear A
```

o tutte le variabili con il comando `clear`:

```
>> clear
```

Per ripulire la finestra grafica dalle istruzioni precedenti è possibile usare il comando

```
>> clc
```

oppure

```
>> home
```

Per conoscere la funzionalità di una istruzione sconosciuta si usa il comando

```
>> help istruzione
```

Digitando soltanto il comando **help**, viene mostrato a schermo un elenco di tutti i pacchetti disponibili in Matlab®.

La seguente tabella riassume le principali funzioni di gestione dell'ambiente e la loro azione:

comando	azione
clear	Cancella tutte le variabili
clear var	Cancella la variabile <code>var</code>
clc	Cancella tutte le istruzioni a schermo e blocca la barra di scorrimento
home	Cancella tutte le istruzioni a schermo
help istruzione	Fornisce le funzionalità e le modalità d'uso di <code>istruzione</code>
who	Elenca le variabili in memoria
whos	Elenca le variabili, il loro tipo e la dimensione di memoria occupata

Variabili predefinite. Alcune variabili, proprie di Matlab®, non necessitano di alcuna definizione. Tra queste ricordiamo:

```
>> pi: il numero  $\pi = 3.141592653589793\dots$ ;
```

```
>> i: l'unità immaginaria  $\sqrt{-1}$ ;
```

```
>> exp(1): il numero di Nepero  $e = 2.718281828459046\dots$ 
```

Attenzione! le variabili predefinite possono essere ridefinite, ovvero:

```
>> a = 5 + 2 * i
a =
    5.0000 + 2.0000i
>> i = 2
i =
     2
>> a = 5 + 2 * i
a =
     9
```

Istruzione format. Permette di modificare il formato di visualizzazione dei risultati ma NON la precisione con cui i calcoli vengono condotti. Il comando ha la sintassi

```
>> format tipo
>> 1/7
```

e produce i risultati elencati nella seguente tabella, in base al `tipo` usato

tipo	
rat	1/7
short	0.1429
short e	1.4286e - 01
short g	0.142856
long	0.142857142857143
long e	1.428571428571428e - 01
long g	0.142857142857143

Il comando `format`, senza ulteriori specifiche, seleziona automaticamente il formato più conveniente per la classe delle variabili in uso. In Octave il comando `format` si usa esattamente nello stesso modo, è però possibile che fornisca risultati leggermente diversi da quelli di Matlab®. **Assegnazione di vettori.** Riportiamo alcuni metodi per l'inserimento di vettori.

- Si possono costruire vettori riga elencando gli elementi separati da uno spazio o da una virgola, come nella tabella seguente

comando	risultato
<code>>> b = [1 5 9 4]</code>	<code>b = (1, 5, 9, 4)</code>
<code>>> c = [1, 5, 9, 4]</code>	<code>c = (1, 5, 9, 4)</code>

- Per costruire dei vettori riga con elementi equispaziati, il comando generico è

```
>> b = [ primo elemento : passo: ultimo elemento ]
```

Alcuni esempi sono raccolti nella tabella sottostante:

comando	risultato
<code>>> d = [1 : 1 : 4]</code>	<code>d = (1, 2, 3, 4)</code>
<code>>> g = [1 : 4]</code>	<code>g = (1, 2, 3, 4)</code>
<code>>> r = [1 : 0.1 : 2]</code>	<code>r = (1, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2)</code>
<code>>> s = [0 : -0.3 : -1]</code>	<code>s = (0, - 0.3, - 0.6, - 0.9)</code>

Si può utilizzare anche il comando `linspace` per definire un vettore riga di elementi equispaziati. Al comando devono essere forniti come parametri di ingresso i due estremi dell'intervallo e il numero di elementi del vettore.

```
>> p = linspace( primo elemento, ultimo elemento, numero elementi)
```

Nell'esempio il vettore `p` è costituito da 6 elementi equispaziati nell'intervallo `[0, 5]`:

```
>> p = linspace( 0, 5, 6)
p =
    0    1    2    3    4    5
```

- I vettori colonna vengono costruiti elencando gli elementi separati dal punto e virgola, come nell'esempio:

```
>> q = [ 1; 2; 3; 4 ]
q =
1
2
3
4
```

Assegnazione di matrici. È buona norma utilizzare per il nome delle variabili per le matrici le lettere maiuscole. Combinando i comandi visti per definire i vettori riga e colonna si ottengono immediatamente le istruzioni per definire le matrici, per esempio:

```
>> H = [ 1 2 3 ; 2 4 7 ; 1 4 3 ]
H =
1 2 3
2 4 7
1 4 3
```

Operazioni su vettori e su matrici.

- **Trasposizione** L'operazione di trasposizione si realizza aggiungendo un apice all'istruzione di assegnazione:

```
>> a = [ 1 : 4 ];
>> b = a'
b =
1
2
3
4
```

Naturalmente applicando l'operazione di trasposizione a tutte le modalità viste per definire dei vettori riga, si ottengono dei vettori colonna.

- **Operazioni algebriche** Le principali operazioni algebriche tra matrici e vettori sono elencate nella tabella sottostante. Dati

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

operazione	azione
$a + b$	Somma di due vettori della stessa dimensione
$H + G$	Somma di due matrici della stessa dimensione
$a - b$	Sottrazione di due vettori della stessa dimensione
$H - G$	Sottrazione di due matrici della stessa dimensione
$H * G$	Prodotto algebrico righe per colonne. Il risultato è una matrice 3×3
$G * a$	Prodotto algebrico righe per colonne. Il risultato è un vettore 3×1
$a' * G$	Prodotto algebrico righe per colonne. Il risultato è un vettore 1×3
$a' * b$	Prodotto algebrico righe per colonne. Il risultato è una matrice 3×3
$a * b'$	Prodotto algebrico righe per colonne. Il risultato è uno scalare
$3 * G$	Prodotto di uno scalare per una matrice
$3 * b$	Prodotto di uno scalare per un vettore
<code>cross(a,b)</code>	Prodotto vettore. a e b devono necessariamente avere solo tre elementi

Osserviamo che nel prodotto righe per colonne le dimensioni dei vettori o delle matrici utilizzate devono essere compatibili.

- **Operazioni puntate (o elemento per elemento)** In Matlab[®] sono definite anche delle operazioni particolari che agiscono sulle matrici e sui vettori elemento per elemento.

- Somma o sottrazione di un valore scalare ad ogni elemento di una matrice (vettore). La matrice (vettore) risultante ha le stesse dimensioni della matrice (vettore) di partenza.

```
>> f = [ 1 2 3 ; 2 4 6 ; 3 6 9 ];
>> l = f + 3
l =
    4     5     6
    5     7     9
    6     9    12
```

- Prodotto elemento per elemento di due matrici (vettori). Date due matrici (vettori) il loro prodotto elemento per elemento è una matrice (vettore) delle stesse dimensioni e i cui elementi sono il prodotto degli elementi di posto corrispondente delle matrici (vettori) iniziali. (Attenzione alla compatibilità, in questo caso le due matrici (vettori) devono avere le stesse dimensioni!). Per esempio

```
>> b = [2 5 3 2];
>> c = [1 2 4 2];
>> f = b .* c
f =
    2   10   12    4
```

mentre se i due vettori non hanno le stesse dimensioni si ottiene un messaggio di errore, come nel seguente esempio

```
>> b = [2 5 3 2];
>> a = [ 3 5 6];
>> f = a .* b
??? Error using ==> times
Matrix dimensions must agree
```

- Divisione elemento per elemento di due matrici (vettori). Date due matrici (vettori) la loro divisione elemento per elemento è una matrice (vettore) delle stesse dimensioni e i cui elementi sono il rapporto degli elementi di posto corrispondente delle matrici (vettori) iniziali. Per esempio

```
>> b = [2 5 3 2];
>> c = [1 2 4 2];
>> format rat
>> f = b ./ c
f =
    2  5/2  3/4  1
```

- Elevamento a potenza di una matrice (vettore) elemento per elemento. La matrice (vettore) risultante ha le stesse dimensioni della matrice (vettore) di partenza e i suoi elementi sono ottenuti elevando alla potenza richiesta gli elementi della matrice (vettore) di partenza. Per esempio

```
>> a = [1 2 3 4];
>> b = a.^2
b =
    1    4    9   16
```

Si osservi che tutte le operazioni elemento per elemento sono contraddistinte dal “.” che precede il simbolo dell’operazione, con l’eccezione dell’operazione di somma e sottrazione

Nella tabella sottostante sono riassunte le operazioni “elemento per elemento”. Dati

```
>> a = [ 1 3 5 6 ];
>> b = [ 4 6 4 8 ];
```

operazione	azione
$b + 3$	Somma lo scalare 3 a tutti gli elementi di b
$a - 3$	Sottrae lo scalare 3 a tutti gli elementi di a
$a .* b$	Moltiplica gli elementi di a e b di posto corrispondente
$a ./ b$	Divide ogni elemento di a per il corrispondente elemento di b
$a .^n$	Eleva alla potenza n tutti gli elementi di a

Funzioni matematiche elementari. Le funzioni matematiche elementari restituiscono matrici o vettori della stessa dimensione della variabile cui è applicata la funzione. Data una matrice A :

funzione	azione
abs(A)	Valore assoluto degli elementi di A
sqrt(A)	Radice quadrata degli elementi di A
exp(A)	Funzione esponenziale di ogni elemento di A
log(A)	Logaritmo naturale di ogni elemento di A
log10(A)	Logaritmo in base 10 di ogni elemento di A
log2(A)	Logaritmo in base 2 di ogni elemento di A
sin(A)	Seno di ogni elemento di A
cos(A)	Coseno di ogni elemento di A
tan(A)	Tangente di ogni elemento di A
asin(A)	Arco seno di ogni elemento di A
acos(A)	Arco coseno di ogni elemento di A
atan(A)	Arco tangente di ogni elemento di A
sinh(A)	Seno iperbolico di ogni elemento di A
cosh(A)	Coseno iperbolico di ogni elemento di A
tanh(A)	Tangente iperbolica di ogni elemento di A

Osserviamo che per le funzioni trigonometriche i valori di A devono essere espressi in radianti.

Esercizio: calcolare le seguenti espressioni matematiche e confrontarne il risultato

- $\frac{e^5 + \sin^3(\pi)}{\sqrt{\ln 30 - 10}} = -18.1973$
- $e^{\log_2 50} + e^{\log_{10} 40} + e^{\ln 30} = 317.5134$

Una volta definiti $x = 4$ e $y = 5$ calcolare

- $2x \ln(|y| + 1) - y \ln(x + 2) = -5.3753$
- $\arctan\left(\frac{x}{y}\right) - \sin^2(x\sqrt{|y|}) = 0.4611$

Funzioni per la gestione di vettori e matrici. Nella tabella successiva è riportato un elenco delle più importanti funzioni Matlab® e Octave per la gestione di vettori e matrici, con alcuni esempi di applicazione. Alcuni comandi sono indifferentemente utilizzabili sia per i vettori che per le matrici. In caso contrario è esplicitamente dichiarato. Data una matrice A e un vettore b:

funzione	azione
size(A)	restituisce un vettore di due elementi, il primo è il numero di righe di A, il secondo il numero di colonne di A
size(A, 1)	restituisce il primo elemento di size(A), cioè il numero di righe di A
size(A, 2)	restituisce il secondo elemento di size(A), cioè il numero di colonne di A
length(b)	restituisce la lunghezza del vettore b (solo per vettori)
end(b)	restituisce l'ultimo elemento del vettore b (solo per vettori)
max(b)	restituisce il più grande elemento di b
min(b)	restituisce il più piccolo elemento di b
max(A)	restituisce un vettore riga contenente il più grande elemento di ogni colonna di A
min(A)	restituisce un vettore riga contenente il più piccolo elemento di ogni colonna di A
sum(b)	restituisce uno scalare pari alla somma di tutti gli elementi di b
sum(A)	restituisce un vettore i cui elementi sono la somma degli elementi di colonna di A
diag(A)	Estrae la diagonale principale di A (solo per matrici quadrate)
diag(A, 1)	Estrae la sopradiagonale di ordine 1 di A (solo per matrici quadrate)
diag(A, -1)	Estrae la sottodiagonale di ordine 1 di A (solo per matrici quadrate)
diag(b)	Costruisce una matrice quadrata diagonale con gli elementi di b sulla diagonale principale
tril(A)	Crea una matrice triangolare inferiore con elementi coincidenti con i corrispondenti elementi di A (solo per matrici quadrate)
triu(A)	Crea una matrice triangolare superiore con elementi coincidenti con i corrispondenti elementi di A (solo per matrici quadrate)
A(1,1)	Estrae l'elemento di posto (1,1) di A
A(:,1)	Estrae la prima colonna di A
A(1,:)	Estrae la prima riga di A
A(1:3,:)	Estrae le prime tre righe di A
A(1:3,1:3)	Estrae la sottomatrice costituita dalle prime tre righe e tre colonne di A

Merita una descrizione a parte, la funzione `norm`. Essa ha due parametri in input: un vettore v e un numero intero n oppure `inf`. Se viene passato solo il vettore, calcola la norma 2 (*norma euclidea*) di v , definita come $\|v\|_2 = \sqrt{\sum_{i=1}^{\text{length}(v)} v_i^2}$. Se viene passato un numero intero n , calcola la norma n di v definita come $\|v\|_n = \left(\sum_{i=1}^{\text{length}(v)} |v_i|^n\right)^{\frac{1}{n}}$. Infine se viene passato come secondo argomento `inf`, calcola la norma infinito di v definita come $\|v\|_\infty = \max_{1 \leq i \leq \text{length}(v)} |v_i|$.

operazione	azione
<code>norm(v)</code>	Calcola la norma euclidea (norma 2) di v
<code>norm(v,1)</code>	Calcola la norma 1 di v
<code>norm(v,2)</code>	Calcola la norma euclidea (norma 2) di v
<code>norm(v,inf)</code>	Calcola la norma infinito di v

Funzioni per definire vettori o matrici particolari.

Dati `Nrighe` e `Ncolonne` il numero di righe e il numero di colonne della variabile vettoriale che vogliamo costruire, la tabella seguente riassume i comandi per la costruzione di matrici particolari:

funzione	azione
<code>zeros(Nrighe, Ncolonne)</code>	Costruisce una matrice (vettore) di tutti 0
<code>zeros(Nrighe)</code>	Costruisce una matrice quadrata di tutti 0
<code>ones(Nrighe, Ncolonne)</code>	Costruisce una matrice (vettore) di tutti 1
<code>ones(Nrighe)</code>	Costruisce una matrice quadrata di tutti 1
<code>eye(Nrighe)</code>	Costruisce una matrice quadrata con elementi pari ad 1 sulla diagonale (matrice identità)
<code>rand(Nrighe, Ncolonne)</code>	Costruisce una matrice (vettore) di elementi casuali compresi tra 0 ed 1
<code>rand(Nrighe)</code>	Costruisce una matrice quadrata di elementi casuali compresi tra 0 ed 1

Esercizio: quadrato magico di ordine 4.

$$A = \begin{bmatrix} 16 & 2 & 3 & 13 \\ 5 & 11 & 10 & 8 \\ 9 & 7 & 6 & 12 \\ 4 & 14 & 15 & 1 \end{bmatrix}$$

a) Inserire la matrice.

o I modo)

```
>> A = [16 2 3 13; 5 11 10 8; 9 7 6 12; 4 14 15 1]
```

```
A =
```

```
16     2     3     13
 5     11    10     8
 9     7     6     12
 4     14    15     1
```

o II Modo)

```
>> A = [ 16 2 3 13
>>      5 11 10 8
>>      9 7 6 12
>>      4 14 15 1 ]
```

```
A =
```

```
16     2     3     13
 5     11    10     8
```

```

    9    7    6   12
    4   14   15    1

```

- o III Modo) Usare il comando `magic(4)` che costruisce la matrice magica di ordine 4.

```

>> A = magic(4)
A =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1

```

- b) Sommare per colonne.

```

>> sum(A,1)
ans =
    34    34    34    34

```

- c) Sommare per righe.

- o I Modo)

```

>> sum(A,2)
ans =
    34
    34
    34
    34

```

- o II Modo)

```

>> A = A'
A =
    16     5     9     4
     2    11     7    14
     3    10     6    15
    13     8    12     1
>> sum(A,1)
ans =
    34    34    34    34

```

Il comando `A'` costruisce la matrice trasposta della matrice `A`. Osserviamo che il comando `sum(A)` calcola la somma per colonne degli elementi della matrice `A`.

- d) Sommare gli elementi della diagonale principale.

- o I Modo) Scriviamo la somma degli elementi richiesti. Con il comando `A(n,m)` otteniamo il valore dell'elemento di `A` posizionato sull'`n`-esima riga e sull'`m`-esima colonna.

```
>> A(1,1) + A(2,2) + A(3,3) + A(4,4)
ans =
    34
```

oppure, utilizzando il comando `sum` visto per i vettori:

```
>> sum([A(1,1) A(2,2) A(3,3) A(4,4)])
ans =
    34
```

o II Modo) Cominciamo con l'estrarre la diagonale principale:

```
>> a = diag(A)
a =
    16
    11
     6
     1
```

oppure

```
>> a = diag(A,0)
a =
    16
    11
     6
     1
```

Una volta estratta la diagonale, possiamo usare il comando `sum` visto nel caso dei vettori.

```
>> sum(a)
ans =
    34
```

L'operazione si può eseguire in una sola riga di comando scrivendo:

```
>> sum(diag(A))
ans =
    34
```

f) Calcolare $A*A$, A^2 , $A.*A$ e $A.^2$.

```
>> A * A
ans =
    345    257    281    273
    257    313    305    281
    281    305    313    257
    273    281    257    345
>> A^2
ans =
    345    257    281    273
```

```

    257  313  305  281
    281  305  313  257
    273  281  257  345
>> A .* A
ans =
    256     4     9   169
     25    121    100    64
     81     49     36   144
     16    196    225     1
>> A.^2
ans =
    256     4     9   169
     25    121    100    64
     81     49     36   144
     16    196    225     1

```

i) Calcolare la radice quadrata degli elementi di A:

```

>> sqrt(A)
ans =
    4.0000    1.4142    1.7321    3.6056
    2.2361    3.3166    3.1623    2.8284
    3.0000    2.6458    2.4495    3.4641
    2.0000    3.7417    3.8730    1.0000

```

oppure

```

>> A.^(1/2)
ans =
    4.0000    1.4142    1.7321    3.6056
    2.2361    3.3166    3.1623    2.8284
    3.0000    2.6458    2.4495    3.4641
    2.0000    3.7417    3.8730    1.0000

```

Esercizio: costruire la matrice

```

>> A =
     2     0    10     0    40
     5     2     0    10     0
    10     5     2     0    10
     0    10     5     2     0
    40     0    10     5     2

```

- sommare tutti gli elementi della matrice A.
- estrarre tutte le sottomatrici principali di ordine 3 dalla matrice A.