Introduzione al MATLAB©

Lucia Gastaldi

Dipartimento di Matematica,
http://dm.ing.unibs.it/gastaldi/

15 gennaio 2008

Outline

MATLAB = MATrix LABoratory

MATLAB è un linguaggio di supporto per il calcolo scientifico.

Calcolo scientifico:

si occupa dello sviluppo, della implementazione e dell'analisi degli algoritmi numerici utili per l'impiego di modelli matematici.

MATLAB è un linguaggio orientato alle matrici con importanti aggiunte per l'analisi dei dati e per la visualizzazione scientifica.
L'unità base dei dati è un **array**.

Proprietà di MATLAB

- È un interprete di comandi.
- È un linguaggio di programmazione.
- Ha una buona potenzialità grafica.
- Versioni per Unix/Linux, Windows, Mac.
- I files Matlab sono portabili da una piattaforma all'altra.

TOOLBOX e SIMULINK

- **TOOLBOX** = Librerie per applicazioni specifiche
  - statistica,
  - reti neurali,
  - ottimizzazione,
  - analisi di immagini,
  - controllo e identificazione di sistemi,
  - logica fuzzy,
  - equazioni alle derivate parziali,
  - matematica finanziaria,
  - ...
- **SIMULINK** = Programmazione grafica per agevolare la modellizzazione e la simulazione di sistemi complessi

Per maggiori dettagli: `www.mathworks.com`

Avviare MATLAB

Selezionare l'icona di MATLAB 7 nel desktop.

Si apre una finestra divisa in tre parti.

Command Window

È la finestra principale, dove compaiono un po' di informazioni sulla versione e sull'help in linea, poi c'e' il **prompt**

```
                    < M A T L A B >
          Copyright 1984-2005 The MathWorks, Inc.
          Version 7.1.0.183 (R14) Service Pack 3
                      August 02, 2005


To get started, select MATLAB Help or Demos from the Help menu.

>>
```

Le finestre laterali

A sinistra della finestra principale ci sono due finestre più piccole.

Finestra superiore

Si può scegliere fra **Workspace** e **Current Directory**.

- **Workspace**: compare il nome delle variabili, con il valore, la dimensione, il numero di Bytes occupati, ed il tipo.
  NB Per scegliere quale informazioni vedere cliccare su View e scegliere le colonne.
- **Current Directory**: compare la lista dei file contenuti nella directory corrente con il tipo (file o folder) e la data dell'ultimo aggiornamento.
  NB Il nome della **Current Directory** si legge nella barra degli strumenti in alto.

Command History:

contiene tutti i comandi che vengono eseguiti nella finestra principale.

Menù principale

- **File**: gestisce i file di Matlab. Si apre un Menù con la possibilità di aprire files (nuovi o già esistenti), di stampare e di definire le preferenze.
- **Edit**: soliti comandi per editare i file.
- **View**: configurazione delle finestre.
- **Graphics**: gestisce i grafici.
- **Debug**: per trovare gli errori nel programma.
- **Desktop**: configurazione del desktop di MATLAB.
- **Window**: per navigare nelle finestre.
- **Help**: aiuto in linea (**molto utile**).

Help

Help Navigator

- **Contents**: elenco del contenuto di MATLAB ad albero.
- **Index**: indice alfabetico.
- **Search**: ricerca per parole chiave.
- **Demos**: esempi.
- **Favorites**: link ai servizi e supporto su Web.

La finestra di destra contiene le informazioni desiderate e vi si naviga come su Web.

Assegnazione di variabili scalari

```
>> a=1.54
```

- a nome della variabile (max 31 caratteri alfanumerici, il primo dei quali non deve essere un numero)
- `1.54` valore numerico assegnato alla variabile.

Nomi delle variabili

I nomi delle variabili **non** devono contenere spazi e caratteri speciali come:

- simboli di operazione: -,=,+,*;
- apostrofi
- punteggiatura
- slash e backslash

Il comando `>> a=1.54` *produce*

```
>> a =
    1.5400
```

Il comando `>> a=1.54;` *non produce risposta*

Variabile di default

```
>> 1.67
```

produce

```
>> ans =
        1.6700
```

`ans` è il nome della variabile di default.

Per vedere il contenuto di una variabile

Visualizzo il contenuto della variabile a

```
>> a
```

produce

```
a =

    1.5400
```

Visualizzo il contenuto della variabile ans

```
>> ans
```

produce

```
ans =

    1.6700
```

Formato di rappresentazione dei numeri `short`

```
>> c=0.456723
c =
    0.4567
```

Il numero è stato rappresentato con 5 cifre

```
>> format short e
>> c
c =
    4.5672e-01
```

Forma esponenziale con 5 cifre per la mantissa

Formato di rappresentazione dei numeri `long`

```
>> format long e
>> c
c =
     4.567230000000000e-01
```

Forma esponenziale con 16 cifre per la mantissa

```
>> format long
>> c
c =
    0.456723000000000
```

Il numero è rappresentato con 15 cifre

Formati disponibili

Variabile	**Significato**
`FORMAT`	Default.
`FORMAT SHORT`	Virgola fissa scalata con 5 cifre.
`FORMAT LONG`	Virgola fissa scalata con 15 cifre.
`FORMAT SHORT E`	Forma esponenziale con 5 cifre di mantissa.
`FORMAT LONG E`	Forma esponenziale con 15 cifre di mantissa.
`FORMAT SHORT G`	Sceglie la rappresentazione migliore con 5 cifre.
`FORMAT LONG G`	Sceglie la rappresentazione migliore con 15 cifre.

Esempio

$1.5 \Rightarrow$

format short	1.5000
format long	1.50000000000000
format short e	1.5000e+00
format long e	1.50000000000000e+00
format short g	1.5
format long g	1.5

Variabili predefinite

Variabile	**Significato**
`ans`	valore ultima operazione eseguita e non assegnata ad una variabile
`i,j`	unità immaginaria, $\sqrt{-1}$
`pi`	approssimazione di π
`eps`	precisione macchina
`realmax`	massimo numero macchina positivo rappresentabile
`realmin`	minimo numero macchina positivo rappresentabile
`Inf`	∞, ossia un numero maggiore di `realmax`
`NaN`	Not a Number (0/0, `Inf`/`Inf`, ...)
`computer`	tipo di computer
`version`	versione di MATLAB

Variabili predefinite

Il contenuto di queste variabili può essere variato con una semplice operazione di assegnazione:

```
>> pi=18
pi =
    18
```

Per riassegnare alla variabile `pi` il valore π:

```
>> clear pi
>> pi
ans =
    3.1416
```

clear

Per cancellare il contenuto della variabile a: `>> clear a`

Per cancellare il contenuto di tutte le variabili: `>> clear`

Area di lavoro **WORKSPACE**

Le variabili vengono memorizzate nell'area di lavoro **Workspace**.
La finestra **Workspace** contiene la lista della variabili e le seguenti informazioni:

- **Name**: nome della varabile.
- **Value**: valore assegnato alla varabile.
- **Size**: dimensione come array (righe per colonne).
- **Bytes**: occupazione di memoria in termini di **bytes**.
- **Class**: il tipo di variabile char, double, sparse, cell, struct, uint8.

Di default, Matlab lavora con variabili in **doppia precisione**.
Ogni numero memorizzato in doppia precisione occupa 8 Bytes.
Le variabili scalari sono viste come **array** di dimensione `1x1` (una riga e una colonna).

Lettere maiuscole e minuscole sono considerate diverse sia nei comandi che nei nomi delle variabili.

Operazioni aritmetiche

^ potenza
* prodotto
/ divisione
+ somma
- differenza

Es: per calcolare $x = \dfrac{3 + 5^3 - 2/3}{4(5 + 2^4)}$

```
>> x=(3+5^3-2/3)/(4*(5+2^4))
```

- Sono osservate le precedenze classiche dell'artimetica
- Per alterare le precedenze si utilizzano esclusivamente le parentesi **tonde**

Per spezzare le righe

Il comando

```
>> b=1+1/2+5/3+1/4+23/6+...
2/9+1/10;
```

permette di spezzare un'istruzione troppo lunga

Array

Il linguaggio MATLAB lavora con un solo tipo di oggetti: **l'array di MATLAB**.

Tutte le variabili di Matlab, inclusi scalari, vettori, matrici, stringhe, celle (cell arrays), strutture e oggetti sono memorizzati in Matlab come **array**.

Ogni array contiene le seguenti informazioni:

- Il tipo
- La dimensione
- I dati associati all'array
- Se la variabile è reale o complessa, nel caso di array numerico
- Gli indici e gli elementi diversi da zero, nel caso di array sparse,
- Il numero di campi e il nome dei campi, nel caso di una struttura o oggetto.

Assegnazione di array - vettori riga e colonna

```
>> a=[1 2 3 4];
>> a=[1,2,3,4];
>> a=(1:4);
```

Modi equivalenti per generare un array 1x4, 1 riga e 4 colonne, vettore riga, contenente i numeri da 1 a 4

```
>> a
a =
     1     2     3     4
```

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

Per generare un array 4x1, 4 righe e 1 colonna, vettore colonna

Assegnazione di array - matrici

```
>> c=[5 3 4; 2 4 -2]
c =
     5     3     4
     2     4    -2
```

Per generare un array 2x3, matrice 2 righe e 3 colonne

```
>> d(3,4)=3.

d =
     0     0     0     0
     0     0     0     0
     0     0     0     3
```

Genera una matrice 3x4, che ha tutti elementi nulli tranne quello di posto 3,4

Lo spazio o la virgola separano elementi sulla stessa riga.
Il punto e virgola separa le righe.

Dimensioni di un array

Il comando **size** fornisce le dimensioni di una matrice.

```
>> size(c)

ans =
     2     3
```

produce il vettore riga di due elementi contenenti il numero di righe e di colonne di `c`.

Il comando **length** fornisce la lunghezza di un vettore.

```
>> length(a)

ans =
     4
```

produce un numero pari alla lunghezza del vettore `a`.

```
length(c)=max(size(c))
```

Come accedere agli elementi di `array`

```
>> a(2)
ans =
     2
```

Per accedere ad un elemento di un vettore

```
>> c(2,1)
ans =
     2
```

Per accedere ad un elemento di una matrice

Come modificare un elemento di un array

```
>> b(3)=5
b =
     1
     2
     5
     4
```

Per modificare un elemento di un vettore. Se non si utilizza il **;** *viene visualizzato l'array completo*

```
>> c(1,3)=18
c =
     5     3    18
     2     4    -2
```

Per modificare un elemento di una matrice.

Operazioni standard dell'algebra lineare

\+ somma di vettori o matrici (elemento per elemento)
\- differenza di vettori o matrici (elemento per elemento)
$*$ prodotto tra vettori e/o matrici (righe per colonne)

Sono le operazioni dell'algebra lineare; quindi:

- per somma e differenza: gli operandi devono avere le stesse dimensioni
- per il prodotto: il numero delle colonne della prima matrice deve essere uguale al numero delle colonne della seconda matrice.

Operazioni su array

```
>> a1+b
```

entrambi vettori colonna 4x1

```
ans =
     2
     4
     8
     8
```

```
>> a-b
??? Error using ==> -
Matrix dimensions must agree.
```

`a` =*vettore riga* 1x4
`b` =*vettore colonna* 4x1

```
>> a*b
ans =
    36
```

(1x4)(4x1) *prodotto scalare*

```
>> c*d'
ans =
   358
   -14
```

(2x3)(3x1) *prodotto matrice vettore*

```
>> d*c
??? Error using ==> *
Inner matrix dimensions must agree.
```

(3x1)(2x3) *prodotto non possibile*

Operazioni punto

Le operazioni **punto** agiscono su array che abbiano le stesse dimensioni:

`.*` prodotto elemento per elemento
`./` divisione elemento per elemento
`.^` potenza elemento per elemento

```
>> a1b=a1.*b
a1b =
     1
     4
    15
    16
```

$$(\texttt{a1b})_i = (\texttt{a1})_i * \texttt{b}_i$$

$$\text{con a1} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} \text{ e b} = \begin{bmatrix} 1 \\ 2 \\ 5 \\ 4 \end{bmatrix}$$

Trasposizione di vettore

```
>> a'
ans =
     1
     2
     3
     4
```

Il vettore trasposto di `a` *viene memorizzato nella variabile* `ans`

```
>> a1=a'
```

Il vettore trasposto di `a` *viene memorizzato nella variabile* `a1`

... trasposizione di matrici:

```
>> c1=c'
c1 =
     5     2
     3     4
     4    -2
```

La notazione due punti

La **notazione due punti** **:** serve per creare vettori, sottomatrici e per il ciclo di tipo **for**

```
Vettore=Inizio:Passo:Fine
```

Come creare vettori a valori equispaziati

Dati due numeri interi `j` e `k`, il comando

```
>> j:k
```

produce il vettore contenente i numeri interi da `j` a `k` compresi.

Il vettore è vuoto se `j>k`.

Esempio

```
>> x = 1:7
x =
     1     2     3     4     5     6     7
```

Come creare vettori a valori equispaziati

Vettore=Inizio:Passo:Fine

Dati tre numeri reali `i`, `j` e `k`, il comando

```
>> j:i:k
```

produce il seguente vettore

`[j,j+i,j+2i, ...,j+mi]` essendo $\texttt{j+mi} \leq \texttt{k}$ e $\texttt{j+(m+1)i} > \texttt{k}$

ossia, partendo dal valore `j`, si somma `i` fino a raggiungere un valore tale che sommando `i` si ottiene un valore **maggiore** a `k`.

Il vettore è vuoto se $\texttt{i} = 0$, se $\texttt{i} > 0$ e $\texttt{j} > \texttt{k}$, oppure se $\texttt{i} < 0$ e $\texttt{j} < \texttt{k}$.

Nel caso in cui $\texttt{i} = 1$ allora i due comandi

```
>> j:i:k        >>j:k
```

danno lo stesso risultato.

Esempi

```
>> x = 1:2:15
x =
     1     3     5     7     9    11    13    15
```

```
>> y=1:9
y =
     1     2     3     4     5     6     7     8     9
```

```
>> z=10:-2:2
z =
    10     8     6     4     2
```

```
>> w=1:0.33:3
w =

    1.0000    1.3300    1.6600    1.9900    2.3200    2.6500
```

Come estrarre righe e colonne di una matrice

Dato un array A:

- `A(:,j)` è la j-esima colonna di A;
- `A(i,:)` è la i-esima riga di A;
- `A(:,j:k)` è la sottomatrice di A che contiene le colonne di A dalla j-esima alla k-esima;
- `A(:)` fornisce tutti gli elementi di A, vista come una singola colonna

Esempi

```
>> A=hilb(5)
A =
    1.0000    0.5000    0.3333    0.2500    0.2000
    0.5000    0.3333    0.2500    0.2000    0.1667
    0.3333    0.2500    0.2000    0.1667    0.1429
    0.2500    0.2000    0.1667    0.1429    0.1250
    0.2000    0.1667    0.1429    0.1250    0.1111
```

```
>> A(2,:)
ans =
    0.5000    0.3333    0.2500    0.2000    0.1667
```

```
>> A(:,3)
ans =
    0.3333
    0.2500
    0.2000
    0.1667
    0.1429
```

```
>> A(1:2,1:3)
ans =
    1.0000    0.5000    0.3333
    0.5000    0.3333    0.2500
```

Esercizi

Esercizio 1

Costruire un vettore `b` di $N > 10$ componenti in modo che valga:

$$b_i = (-1)^{i+1} \quad \text{ossia} \quad b = (1, -1, 1, -1, \ldots, -1^{N+1}).$$

Modificare il vettore b in modo che le componenti multiple di 3 abbiamo valore 0 cioè $b_{3i} = 0$ per $i = 1, \ldots, N/3$.
Modificare il vettore b in modo che valga $b_{10} = 100$.

Esercizio 2

Costruire una matrice di numeri casuali di dimensione 10×10 con il comando

```
A=rand(10)
```

Estrarre nella matrice B le colonne pari e nella matrice C la sottomatrice principale di dimensione 5×5 che si ottiene eliminando le ultime 5 righe e colonne da A.

linspace

Se il passo non è intero, è preferibile il comando **linspace** per creare un vettore di punti equispaziati in un intervallo:

`linspace` (*Inizio, Fine, Numero di Punti*)

```
>> a=0; b=1; n=8;
>> x=linspace(a,b,n)

x =
  Columns 1 through 7

         0   0.1429   0.2857   0.4286   0.5714   0.7143   0.8571

  Column 8

  1.0000
```

Il vettore ha componenti:

$$x(i) = a + (i-1)\frac{b-a}{n-1}.$$

Funzioni matematiche predefinite

```
>> y=cos(pi/4)+sin(pi/4)

y = 1.4142
```

Funzione	**Significato**
`sin, cos, tan`	seno, coseno, tangente
`asin, acos, atan`	arcoseno, arcocoseno, arcotangente
`exp`	esponenziale
`sinh, cosh`	seno iperbolico, coseno iperbolico
`tanh`	tangente iperbolica
`log, log2, log10`	logaritmo in base e, in base 2 e in base 10
`sqrt`	radice quadrata
`abs`	valore assoluto
`sign`	funzione segno

Come assegnare una funzione: **inline**

`f=inline(expr,arg1,arg2,...,argn)`

dichiara una **function** di nome `f`:
la stringa `expr` contiene l'espressione di `f`
`arg1,arg2,...,argn` sono i nomi delle variabili da cui dipende `f`.

Esempio

```
>> f=inline('x.^2.*atan(x)')        produce

f=
     Inline function:
     f(x)=x.^2.*atan(x)

>> g=inline('sqrt(x.^2+y.^2)','x','y')        produce

g=
     Inline function:
     g(x,y)=sqrt(x.^2+y.^2)
```

Come valutare una funzione

Per valutare `f` nel punto `x`:

```
>> x=1.718;
>> y=f(x);
```

Per valutare `g` nel punto (a, b):

```
>> a=1.71; b=2.23;
>> z=g(a,b);
```

Il nome della variabile in cui si calcola il valore di una funzione assegnata come `inline` non deve essere necessariamente uguale al nome delle variabili usate nella assegnazione della funzione

Si può valutare la funzione anche in un vettore di punti:

```
>> x1=[0 1]; y1=[1 2]
>> g(x1,y1)
ans=
        1    2.2361
```

Esercizio

Problema 1:

valutare $f(x) = x^2 \cos(x)$ sull'intervallo $I = [-1, 2]$ e rappresentarla graficamente.

- Definire una griglia sull'intervallo $I = [-1, 2]$, ovvero scegliere un insieme discreto di punti rappresentativo per I:
  `>> x=linspace(-1,2,50);`
  Crea un vettore riga di 50 elementi, contenente i valori di 50 punti equispaziati in I
- Definire la funzione e valutarla:
  `>> f=inline('x.^2.*cos(x)'); y=f(x);`
  x è un vettore, si vuole calcolare $y_i = x_i^2 cos(x_i)$ per ogni i, quindi si devono usare le operazioni .
- Rappresentare i punti (x_i, y_i) su di un piano cartesiano:

  `>> plot(x,y)`

plot

plot(x,y, *'color_linestyle_marker'*)

```
>> plot(x,y,'m-*')
```

color: `c,m,y,r,b,g,w,k`

linestyle: `-,--,:,-.,none`

marker: `+,o,*,.,x,square`

Per disegnare 2 o piú coppie di vettori sullo stesso grafico:

```
>> g=inline('sin(x).*exp(x)');
>> yg=g(x);
>> plot(x,y,'b:',x,yg,'r-');
```

oppure

```
>> plot(x,y,x,yg)
>> legend('f','g')
```

fplot

Per fare il grafico di una funzione f su un intervallo $[a, b]$ si può usare il comando `fplot` con la seguente sintassi

```
fplot(f,[a,b])
```

oppure

```
fplot('stringa',[a,b])
```

dove `stringa` contiene l'espressione della funzione.

Se la funzione è assegnata con un M-file di tipo function la sintassi è:

```
fplot(@f,[a,b])
fplot('f',[a,b])
```

Esempi sull'uso di**fplot**

Negli esempi seguenti, consideriamo la funzione $f : [-2, 2] \to \mathbb{R}$ data da

$$f(x) = \frac{1}{1+x^2} \quad x \in [-2, 2].$$

Funzione di tipo **inline**

Definizione	`fun=inline('1./(1+x.^2)')`
Valutazione	`y=fun(1.3)`, `y=fun(x)`, `y=fun(a)`
Grafica	`fplot(fun,[-2,2])`

Funzione assegnata con una **function**

Definizione	`function y=fun(x)` `y=1./(1+x.^2);`
Valutazione	`y=fun(1.3)`, `y=fun(x)`, `y=fun(a)`
Grafica	`fplot(@fun,[-2,2])`