Matlab. Prime nozioni.

Alvise Sommariva

Università degli Studi di Padova
Dipartimento di Matematica

7 marzo 2017

Introduzione

Il proposito di questa prima lezione è mostrare le prime istruzioni in Matlab per poter effettuare alcuni semplici programmi numerici.

L'utente che non intenda utilizzare Matlab può considerare l'installazione di Octave, un ambiente numerico che presenta così tante similitudini da essere utilizzato con codici MATLAB con alte probabilitá di non presentare alcun errore di sintassi.

Per dettagli tecnici si consulti

http://octave.sourceforge.net/

o la guida a Matlab (in inglese)

http://it.mathworks.com/moler/chapters.html

Il Workspace di Matlab

Tanto l'ambiente Matlab quanto Octave, presentano una shell detta workspace in cui possono essere eseguite le prime operazioni.

A tal proposito una volta lanciati gli ambienti gli ambienti Matlab/Octave

- cliccando su un'icona di Matlab/Octave oppure
- lanciandoli dalla shell di Linux,

in questi si possono effettuare i primi calcoli, utilizzando la workspace come fosse una calcolatrice.

I numeri in Matlab/Octave

Matlab/Octave utilizza di default una rappresentazione numerica in doppia precisione, cioè via 64 bit, seguendo lo standard IEEE floating-point.

Questo permette di rappresentare numeri macchina in valore assoluto nell'intervallo [realmin, realmax].

Per illustrare questo, una volta raggiunta la workspace di Matlab/Octave, vediamo quanto siano `realmin` e `realmax`, digitando

```
>> realmin
ans =
      2.2250738585072e-308
>> realmax
ans =
     1.7976931348623e+308
>>
```

I numeri in Matlab/Octave

Una quantità rilevante è la precisione di macchina, cioè la minima costante eps tale che se $x \in [\text{realmin}, \text{realmax}]$ e $\texttt{fl}(x)$ è la rappresentazione di x tramite numeri macchina allora

$$\texttt{fl}(x) = x \cdot (1 + \delta), \quad \text{con } |\delta| \leq u$$

cioè , con facili conti,

$$\frac{\texttt{fl}(x) - x}{x} = \delta, \quad \text{con } |\delta| \leq u.$$

In altri termini eps è il più grande errore relativo compiuto nell'approssimare un numero reale che sta in [realmin, realmax] con un numero macchina.

I numeri in Matlab/Octave

In Matlab/Octave risulta

```
>> eps
ans =
       2.2204e-16
>>
```

Nota.

Si osservi che `eps` *non è il piccolo numero positivo rappresentabile in Matlab/Octave, visto che tale quantità è* `realmin`*. Lo si può dedurre pure dal fatto che pure* `eps/2` *è rappresentabile, come si vede da*

```
>> eps/2
ans =
    1.1102e-16
>>
```

I numeri in Matlab/Octave

Matlab/Octave dispone di altre quantità interessanti predefinite.

- `pi` che è il floating point rappresentante π.
- `Inf` che rappresenta infinito.
- `NaN` che rappresenta not a number.

```
>> 34/0
ans =
    Inf
>> 0/0
ans =
    NaN
>> Inf/Inf
ans =
    NaN
>> Inf-Inf
ans =
    NaN
>> Inf^Inf
ans =
    Inf
```

Operazioni aritmetiche e funzioni elementari predefinite

Tra i numeri macchina è possibile fare le usuali operazioni aritmetiche

$+$	addizione
$-$	sottrazione
$*$	prodotto
$/$	divisione
$\wedge$	potenza

Operazioni aritmetiche e funzioni elementari predefinite

In Matlab/Octave sono predefinite le seguenti funzioni elementari

`abs`	valore assoluto
`sin`	seno
`cos`	coseno
`tan`	tangente
`cot`	cotangente
`asin`	arco seno
`acos`	arco coseno
`atan`	arco tangente
`sinh`	seno iperbolico
`cosh`	coseno iperbolico
`tanh`	tangente iperbolica
`asinh`	arco seno iperbolico
`acosh`	arco coseno iperbolico
`atanh`	arco tangente iperbolica
`sqrt`	radice quadrata
`exp`	esponenziale
`log 2`	logaritmo base 2
`log10`	logaritmo base 10
`log`	logaritmo naturale
`fix`	arrotondamento verso 0
`round`	arrotondamento verso l'intero più vicino
`floor`	arrotondamento verso $-\infty$
`ceil`	arrotondamento verso $+\infty$
`sign`	segno
`rem`	resto della divisione

Il Workspace di Matlab

Nel workspace di Matlab/Octave, digitiamo

```
>> a=atan(0.2);
```

In questa prima riga, Matlab/Octave ha

- valutato l'arcotangente di 0.2;
- ha assegnato tale valore alla variabile "a";
- la presenza del ";" alla fine dell'istruzione, ha concluso l'operazione senza scrivere nel workspace un'approssimazione del valore della variabile "a".

Il Workspace di Matlab e il comando di assegnazione

Non convinti di questo, scriviamo

```
>> a=atan(0.2)
a =
    0.1974
>>
```

In queste poche righe, Matlab/Octave ha

- valutato l'arcotangente di 0.2;
- ha assegnato tale valore alla variabile "a";
- l'assenza del ";" alla fine dell'istruzione, ha concluso l'operazione scrivendo nel workspace un'approssimazione del valore della variabile "a" con alcune cifre decimali.

Il Workspace di Matlab e il comando di assegnazione

Vediamo un secondo esempio, con le funzioni `floor`, `ceil`, `round`.

```
>> a=4.65;
>> b=ceil(a) % intero piu' grande piu' vicino.
b =
     5
>> c=round(a) % arrotondamento agli interi
c =
     5
>> d=floor(a) % intero piu' piccolo piu' vicino.
d =
     4
>>
```

Il Workspace di Matlab e il comando di assegnazione

```
>> a=-4.65;
>> b=ceil(a)
b =
    -4
>> c=round(a)
c =
    -5
>> d=floor(a)
d =
    -5
>>
```

Esercizio

Come funziona il comando `fix`*, qualora applicato a* 4.65 *e* -4.65*?*

Il Workspace di Matlab e il comando di assegnazione

Vediamo un terzo esempio, con le operazioni aritmetiche

```
>> a=5; b=pi;
>> s=a+b
s =
    8.1416
>> z=a-b
z =
    1.8584
>> t=a*b
t =
   15.7080
>> u=a/b % Attenzione al segno di divisione /, non e' \
     !!!!
u =
    1.5915
>> a=5; b=2; a^b
ans =
    25
>>
```

Il formati numerici di Matlab/Octave

Matlab/Octave dispone di diversi formati numerici, che visualizzano, qualora necessario, i numeri macchina in modi diversi.

- `format short`: notazione decimale con 4 cifre dopo la virgola;
- `format short e`: notazione esponenziale con 4 cifre dopo la virgola;
- `format short g`: la migliore delle precedenti;
- `format long`: notazione decimale con 15 cifre dopo la virgola in doppia precisione, e 7 cifre dopo la virgola in singola precisione.
- `format long e`: notazione esponenziale con 15 cifre dopo la virgola in doppia precisione, e 7 cifre dopo la virgola in singola precisione.
- `format long g`: la migliore delle precedenti.

Il formati numerici di Matlab/Octave

Consideriamo di seguito come operino i *formati numerici*, su uno stesso numero. Partiamo esaminando i formati di tipo short.

```
>> a=1/7;
>> format short; a
a =
    0.1429
>> format short e; a
a =
   1.4286e-01
>> format short g; a
a =
      0.14286
```

Ad ogni riga di comando si dice al workspace di esporre di default all'utente la variabile in un certo formato numerico (in questi casi di tipo *short*).

Il formati numerici di Matlab/Octave

Per quanto riguarda i formati di tipo long.

```
>> format long; a
a =
   0.142857142857143
>> format long e; a
a =
     1.428571428571428e-01
>> format long g; a
a =
          0.142857142857143
>>
```

Nuovamente, ad ogni riga di comando si dice al workspace di esporre di default all'utente la variabile in un certo formato numerico (in questi casi di tipo *long*).

Singola e doppia precisione

Nonostante non sia indicato utilizzare la singola precisione, a volte per capire alcuni problemi numerici, può tornare utile il comando `single` che converte un numero in precisione singola.
In Matlab (ma non in Octave 3.8!!)

```
>> format long e
>> pi
ans =
     3.141592653589793e+00
>> pi_single=single(pi)
pi_single =
    3.1415927e+00
>> errore_assoluto=abs(pi-pi_single)
errore_assoluto =
    8.7422777e-08
>> errore_relativo=abs(pi-pi_single)/abs(pi)
errore_relativo =
    2.7827534e-08
>>
```

Singola e doppia precisione

Nota.

In Octave 3.8 il comando "`single`*" presenta stranezze*

```
octave:1> format long e
octave:2> pi
ans =     3.14159265358979e+00
octave:3> pi_single=single(pi)
pi\_single =     3.14159274101257e+00
octave:4> pi-pi_single
ans =     0.00000000000000e+00
octave:5>
```

Ovviamente da

$$3.14159265358979e+00 \neq 3.14159274101257e+00$$

non può essere che i due numeri sono uguali e quindi sicuramente

$$3.14159265358979e+00 - 3.14159274101257e+00 \neq 0.$$

Singola e doppia precisione

Per aggirare l'ostacolo usare `double(single(pi))` *come nell'esempio più sotto*

```
octave:7> format long e
octave:8> pi
ans =     3.14159265358979e+00
octave:9> pi_single=double(single(pi))
pi_single =     3.14159274101257e+00
octave:10> pi-pi_single
ans =    -8.74227801261895e-08
octave:11>
```

Gestione dell'output su video: il comando disp

Il comando disp serve per visualizzare una stringa o il valore di una variabile.

A tal proposito cominciamo con il visualizzare le stringhe. Lo faremo tramite esempi. Le stringhe vengono descritte come testo alfanumerico tra due apostrofi del tipo '(apostrofo *dritto*). Possono contenere o meno spazi.

```
>> disp('pippo_pluto_e_paperino')
pippo_pluto_e_paperino
>> disp('pippo pluto e paperino')
pippo pluto e paperino
>>
```

Gestione dell'output su video: il comando disp

Un classico problema è quello di rappresentare stringhe contenenti proprio "'". Lo si supera digitando tali apostrofi due volte.

```
>> disp('L''accento viene rappresentato correttamente')
L'accento viene rappresentato correttamente
>>
```

Se la stringa è immagazzinata in una variabile, è comunque possibile rappresentare il contenuto della variabile.

```
>> s='Frankestein Junior';
>> disp(s)
Frankestein Junior
>>
```

Gestione dell'output su video: il comando disp

Il comando disp può pure descrivere quantità numeriche, sia che siano quantità numeriche che valori di variabili.

```
>> disp(pi+1)
    4.141592653589793e+00
>> s=pi+1;
>> disp(s)
    4.141592653589793e+00
>>
```

Si possono visualizzare più dati in unico comando disp.

```
>>   disp(['pi vale ',num2str(pi),' in formato short'])
pi vale 3.1416 in formato short
>>
```

Nella precedente riga si è usato num2str che converte un numero in una stringa di testo.

Gestione dell'output su video: il comando disp

A priori il comando num2str potrebbe non essere chiaro e quindi un utente potrebbe desiderare un aiuto.

Per tutte le funzioni, Matlab/Octave dispone di un aiuto, tramite il comando .

Così, nel caso di num2str abbiamo

```
>> help num2str
 num2str Convert numbers to a string.
    T = num2str(X) converts the matrix X into a string
        representation T with about 4 digits and an
        exponent if required.  This is useful for
        labeling plots with the TITLE, XLABEL, YLABEL,
        and TEXT commands.
    ...
    See also int2str, sprintf, fprintf, mat2str.
    ...
>>
```

Gestione dell'output su video: il comando fprintf

Un comando simile a disp per visualizzare un insieme di dati di output con un certo formato è pure fprintf.

```
>> fprintf('Frankestein Junior')
Frankestein Junior>>
```

Notiamo che non ha mandato a capo dopo *Junior*. Per farlo, basta aggiungere il descrittore di formato \n.

```
>> fprintf('Frankestein Junior \n')
Frankestein Junior
>>
```

Se lo avessimo usato pure all'inizio, avremmo ottenuto

```
>> fprintf('\n Frankestein Junior \n')

 Frankestein Junior
>>
```

saltando così pure una riga.

Gestione dell'output su video: il comando fprintf

Il descrittore \t permette invece di fare un *tab*, ovvero spostare la stringa verso destra.

```
>> fprintf('\t Frankestein Junior \n')
        Frankestein Junior
>>
```

Nel caso dei numeri, esistono due descrittori, %f e %e, che rappresentano i numeri in formato decimale ed esponenziale.

```
>> s=10.123456789;
>> fprintf('la variabile s vale %1.5e \n', s)
la variabile s vale 1.01235e+01
>> fprintf('la variabile s vale %2.5f \n', s)
la variabile s vale 10.12346
>>
```

Come si vede, il valore della variabile s è stato espresso

- nel primo caso con una cifra prima e 5 dopo la virgola,
- nel secondo caso con due cifre prima e 5 dopo la virgola.

Definizione di una funzione

In Matlab l'utente può definire una funzione scrivendo un M-file, cioè un file con l'estensione *.m*.
Per scrivere una funzione si digiti nell'editor di Matlab o per Octave con l'editor preferito

```
function y=fun1(x)
y=5+sin(x);
```

e lo si salvi come `fun1.m`.
Di conseguenza

```
y=fun1(pi);
```

assegna alla variabile di input x il valore π e alla variabile di output y il valore $5 + \sin(\pi)$.

Definizione di una funzione

Nota. (Sulla funzione `fun`)

Si osservi che sulle vecchie versioni di Matlab si poteva definire anche `fun.m`*. Nelle nuove versioni, dal 2016, ciò non è possibile visto che esiste una funzione di Matlab con tale nome.*

Definizione di una funzione

Nota.

Ovviamente Matlab segnala errore se alla variabile di output y non è assegnato alcun valore.
Per convincersene si scriva la funzione

```
function y=fun1(x)
z=5+sin(x);
```

e da shell si esegua il comando

```
y=fun1(pi);
```

come risultato Matlab avvisa su shell

```
Warning: One or more output arguments not assigned
    during call to 'fun1'.
```

in quanto la variabile di output y *non è stata assegnata.*

Definizione di una funzione

Nota.

- *Le funzioni si possono salvare ovunque. L'importante è che siano visibili dalla directory in cui si trova attualmente è Matlab/Octave.*
- *Per verificarlo basta scrivere* `ls` *nel workspace di Matlab e verificare che la funzione considerata sia ivi presente.*

  Altrimenti basta cambiare directory con il comando `cd` *fino a trovare la cartella con la funzione desiderata.*

  Alternativamente si può modificare la cartella attuale del workspace in Matlab, cliccando sul simbolo di cartella con la freccia verso il basso, presente sopra il workspace.

Definizione di una funzione

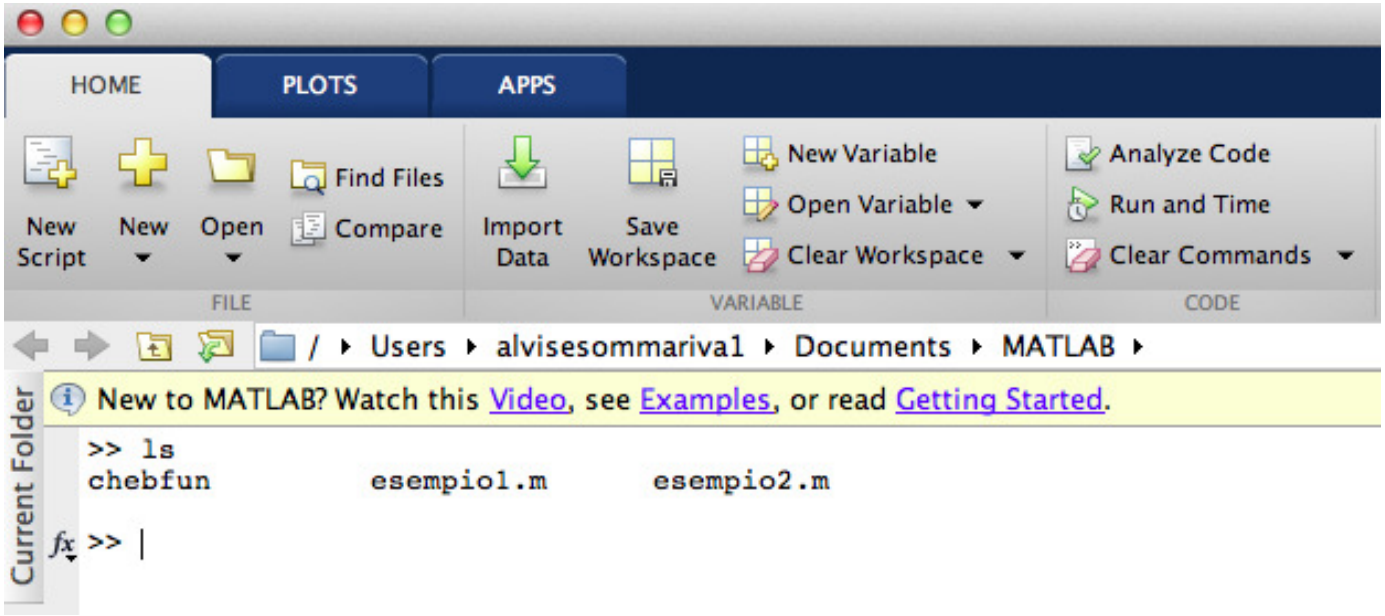

Figura : Esempio di files visibili da una una directory. La directory attuale di Matlab è quella nel campo ▶ *Users* . . . ▶ *MATLAB* ▶ e contiene la cartella `chebfun` e i files `esempio1.m`, `esempio2.m`

Definizione di una funzione

Nota.

Le funzioni predefinite da Matlab sono visibili da qualsiasi directory di lavoro.

Quindi se il file fattoriale.m creato dall'utente è nella cartella PROGRAMMI e viene chiamato dalla funzione binomiale.m che fa parte di una cartella esterna ALTRO (ma non di PROGRAMMI), Matlab segnala l'errore compiuto.

Se invece binomiale.m chiama la funzione Matlab predefinita factorial, la funzione binomiale.m viene eseguita perfettamente.

Definizione di una funzione

Nota. (Variabili locali)

L'uso delle variabili è locale alla funzione. In altre parole se scriviamo

```
s=fun1(pi);
```

durante l'esecuzione della funzione di "`fun1`*", che ricordiamo era*

```
function y=fun1(x)
y=5+sin(x);
```

viene assegnata alle variabili x, y una allocazione di memoria locale che viene rilasciata quando si esce da "`fun1`*". Uno degli effetti è che il programma*

```
>>y=2*pi;
>>x=fun1(y)
```

viene eseguito correttamente nonostante ci sia un'apparente contrasto tra le "x" ed "y" della parte nella workspace di Matlab con le variabili "x" ed "y" della funzione "`fun1`*", che peraltro hanno un significato diverso (alla "x" del programma viene assegnata in "*`fun1`*" la variabile locale "y"!).*

Un esempio di funzione

Nota. (Più variabili in input e output)

Spesso risulta necessario avere più variabili di input o di output in una funzione. Per capirne la sintassi si consideri l'esempio

```
function [s,t] = fun2(x,y)
  s=(x+y);
  t=(x-y);
```

Quindi

```
>> [s,t]=fun2(5,3)
s =
     8
t =
     2
>>
```

Un esempio di funzione

Dato un triangolo rettangolo di cui sono noti i cateti, vogliamo calcolare la sua area e l'ipotenusa. Scriviamo la seguente funzione,

```
function [area,ipot]=parametri_triangolo_rettangolo(
    cateto1,cateto2)
area=cateto1*cateto2/2;
ipot=sqrt(cateto1^2+cateto2^2); % IPOTENUSA CON TEOR.
    PITAGORA.
```

che salviamo nel file parametri_triangolo_rettangolo.m. Il simbolo % permette di fare commenti sul programma.

Un esempio di funzione

Supposto che il file suddetto sia visibile dal workspace (controllare con `ls` e qualora necessario cambiare directory con `cd`!),

```
>> cateto1=3;
>> cateto2=4;
>> [area,ipotenusa]=parametri_triangolo_rettangolo(
    cateto1,cateto2);
>> area
area =
     6
>> ipotenusa
ipotenusa =
     5
>>
```

Il comando input

In molte funzioni può essere utile interagire con l'utente per richiedere determinate quantità.

Il comando input scrive un testo a video, chiedendo all'utente di inviare un valore e premere il tasto "return". Vediamone un esempio.

```
>> a=input('Scrivi un numero intero positivo: ');
Scrivi un numero intero positivo: 5
>> b=input('Scrivi un altro numero intero positivo: ');
Scrivi un altro numero intero positivo: 6
>> fprintf('Il prodotto dei numeri inseriti e'': %8.0f
    \n',a*b);
Il prodotto dei numeri inseriti e':       30
>>
```