Francesca Mazzia
Dipartimento Interuniversitario di Matematica
Università di Bari

MATLAB: lezione introduttiva

MATLAB (MATrix LABoratory): PSE per il scientifico ad alte prestazioni e la visualizzazione, integra analisi numerica, calcolo con matrici, grafica. È dotato di un particolare linguaggio di programmazione di tipo interpretativo che consente di sviluppare le proprie applicazione.

Ogni istruzione viene interpretata ed eseguita immediatamente.
Entrati in ambiente Matlab, compare il prompt:

```
>>
```

il che significa che il calcolatore è pronto a ricevere le istruzioni Matlab ed ad eseguirle. Se vogliamo eseguire la somma 3+2 basta scrivere:

```
>> 3+2
```

e premere il tasto invio. Otteniamo immediatamente la risposta:

```
ans =
      5
>>
```

e il calcolatore è di nuovo pronto a ricevere un nuovo comando. Il risultato dell'operazione eseguita è memorizzato nella *variabile* ans. Le variabili nei linguaggi di programmazione sono dei contenitori di valori. Se vogliamo assegnare il risultato dell'operazione alla variabile di nome var basta scrivere:

```
>>  var = 3+2
        var =
            5
```

Per evitare la visualizzazione del risultato dell'istruzione eseguita si scrive il simbolo ; alla fine della riga, per esempio:

```
>> var = 3+2;
```

Questa istruzione si chiama *istruzione di assegnazione*. Possiamo assegnare a una variabile il valore di una espressione più complicata. Per far questo dobbiamo sapere a quali simboli corrispondono le principali operazioni aritmetico-logiche. In Matlab questi sono:

$+$	addizione
$-$	sottrazione
$*$	moltiplicazione
$/$	divisione
^	elevamento a potenza
&	and logico
|	or logico
~	not logico
$==$	uguale
~ $=$	diverso
$<$	minore
$<=$	minore uguale
$>$	maggiore
$>=$	maggiore uguale

Le espressioni aritmetiche e logiche seguono le classiche regole di precedenza, che è possibile modificare utilizzando le parentesi tonde.

Per avere informazioni su tutte le operazioni che possono essere eseguite in Matlab si può digitare sul pulsante `help` e quindi su Help Window, appare una finestra con tutte le informazioni sul Matlab divise per capitoli. Si può usare anche il comando `help`. Eseguimo i seguenti comandi:

```
>> help
>> help help
>> help('=')
>> help('*')
```

Possiamo ora eseguire:

```
>> var = 5*3/(3+2)^2
var  =
       0.6000
```

È anche possibile calcolare i valori delle principali funzioni. Ad esempio sono predefinite le funzioni in una variabile:

abs(x)	valore assoluto di x,
log(x)	logaritmo naturale,
exp(x)	esponenziale con base e,
sin(x)	seno,
cos(x)	coseno,
tan(x)	tangente.

È possibile quindi eseguire espressioni del tipo:

```
>>  var = (5* 3/(3+cos(5)))^log(2) * 100
var  =
     286.6104
```

Se eseguiamo il comando

```
>>  format short e
```

e rieseguimo l'operazione

```
>>  var = (5* 3/(3+cos(5)))^log(2) * 100
var  =
     2.8661e+02
```

La variabile var = 2.8661e+002 è ora una variabile reale espressa in notazione esponenziale, essa è equivalente ad $2.8661 \cdot 10^2$.

Anche per queste funzioni è possibile usare il comando help. Si provi ad eseguire:

```
>> help cos
>> help tan
>> help log
```

Oltre alle variabili, sono di particolare utilità anche le *costanti*. Le costanti sono dei contenitori di valori che non cambiano nel corso dell'esecuzione di un algoritmo. In Matlab vi sono delle costanti predefinite, come, ad esempio, la costante `pi` che contiene il valore di π:

```
>>  pi
ans  =
     3.1442
```

Le variabili possono essere scalari oppure vettori o matrici. I vettori e le matrici sono variabili che contengono un certo numero di componenti. Ad esempio dopo l'istruzione:

```
>> A = [ 3, 5, 6];
```

la variabile A sarà un vettore di tre elementi con A(1)=3, A(2)=5, A(3)=6. Gli elementi di un vettore si indirizzano specificando l'indice tra parentesi tonde. Se vogliamo sapere la somma degli elementi di A possiamo eseguire:

```
>> A(1) + A(2) + A(3)
ans =
     14
```

Dopo l'istruzione:

```
>>  A = [  3, 5, 7
           8, 9, 10
           2, 3, 5 ] ;
```

la variabile A sarà una matrice con tre righe e tre colonne con elementi A(1,1)=3, A(1,2)=5, ..., A(2,1)=8, ..., A(3,3)=5. Ogni elemento di una matrice viene individuato da una coppia di indici, il primo indice è detto indice di riga, il secondo indice di colonna.

Il Matlab consente anche di memorizzare una sequenza di istruzioni in un file.

I nomi dei file Matlab, detti anche M-file, devono avere l'estensione .m, cioè devono essere del tipo *nomefile.m*. Essi possono essere scritti utilizzando un editore di testi richiamabile con il comando:

```
>> edit
```

oppure selezionando con il mouse `file` e `new` se il file è nuovo o `open` se il file è stato scritto precedentemente.

Gli M-file possono essere di due tipi *script* e *function*. Gli M-file di tipo script contengono semplicemente una sequenza di istruzioni Matlab. Essi utilizzano tutte le variabili già definite, dette variabili globali, le quali possono essere elencate con il comando:

```
>> who
```

oppure:

```
>> whos
```

Gli M-file di tipo function, invece, definiscono una nuova funzione Matlab. Essi accettano dei dati di input e restituiscono dei dati di output come risultato della loro elaborazione. La prima istruzione in un M-file di tipo function deve essere:

$$\text{function } [\ o_1, o_2, \ldots, o_n] = \text{nome-function}(i_1, i_2, \ldots, i_m)$$

dove $o_1, o_2, \ldots, o_n$ sono le variabili di output e $i_1, i_2, \ldots, i_m$ sono le variabili di input.

Quando abbiamo terminato di lavorare con il Matlab possiamo uscire con il comando:

```
>>  quit
```

Quando si esce dal Matlab tutte le variabili definite vanno perdute. È possibile tuttavia salvarne il contenuto in un file mediante il comando:

```
>>  save nome-file
```

Se nome-file non è specificato il contenuto è salvato nel file matlab.mat. Se nome-file è seguito da un elenco di variabili vengono memorizzate solo le variabili considerate. Quando si entra in Matlab è possibile caricare le variabili precedentemente memorizzate in un file mediante il comando:

```
>>  load nome-file
```

Anche in questo caso se nome-file non è specificato si leggono le variabili del file matlab.mat.

Eseguiamo ora la seguente istruzione

```
>> 2+4
```

sul video compare

```
ans =

     6
```

Se ora eseguiamo i comandi `who` e `whos` otteniamo

```
>> who

Your variables are:

ans

>> whos
  Name      Size         Bytes  Class

  ans       1x1              8  double array

Grand total is 1 elements using 8 bytes
```

In Matlab possiamo usare anche variabili complesse

```
>> y= sqrt(-1)

y =

        0 + 1.0000i
```

Se rieseguimo `whos`

```
>> whos
  Name      Size         Bytes  Class

  ans       1x1              8  double array
  x         1x1              8  double array
  y         1x1             16  double array (complex)

Grand total is 3 elements using 32 bytes
```

Il comando `format` permette di cambiare il modo in cui il numero viene visualizzato, per esempio possiamo visualizzare la costante `pi`

```
>> pi

ans =

    3.1416

>> format long
>> pi

ans =

   3.14159265358979

>> format long e
>> pi

ans =
```

```
    3.141592653589793e+000

>> format short e
>> pi

ans =

  3.1416e+000
```

Una costante importante è la costante `eps` che contiene la precisione di macchina:

```
>> eps

ans =

  2.2204e-016
```

Altre costanti collegate alla rappresentazione dei numeri sono:

```
>> realmin

ans =

  2.2251e-308

>> realmax

ans =

  1.7977e+308
```

Il Matlab utilizza l'aritmetica IEEE e permette di rappresentare infinito e NaN:

```
>> 1/0
Warning: Divide by zero.
```

```
ans =

   Inf

>> x=1/0
Warning: Divide by zero.

x =

   Inf
```

Ci sono alcune function matlab che permettono di verificare se un numero è finito o se è uguale ad NaN. Eseguima le seguenti operazioni:

```
>> isfinite(x)

ans =

     0

>> 0/0
Warning: Divide by zero.

ans =

   NaN

>> y=0/0
Warning: Divide by zero.

y =

   NaN

>> isnan(y)

ans =
```

1

Il Matlab consente di scrivere i file di tipo script e di tipo function utilizzando la sequenza, la selezione e la ripetizione. Per la selezione si usa la parola chiave `if`. Eseguiamo:

```
>> help if

 IF IF statement condition.
    The general form of the IF statement is

       IF expression
         statements
       ELSEIF expression
         statements
       ELSE
         statements
       END

    The statements are executed if the real part of the expression
    has all non-zero elements. The ELSE and ELSEIF parts are optional.
    Zero or more ELSEIF parts can be used as well as nested IF's.
    The expression is usually of the form expr rop expr where
    rop is ==, <, >, <=, >=, or ~=.

    Example
       if I == J
         A(I,J) = 2;
       elseif abs(I-J) == 1
         A(I,J) = -1;
       else
         A(I,J) = 0;
       end

    See also RELOP, ELSE, ELSEIF, END, FOR, WHILE, SWITCH.
```

Per i cicli di ripetizione possiamo utilizzare l'istruzione `for` e l'istruzione `while`

```
>> help for

 FOR    Repeat statements a specific number of times.
    The general form of a FOR statement is:

       FOR variable = expr, statement, ..., statement END

    The columns of the expression are stored one at a time in
    the variable and then the following statements, up to the
    END, are executed. The expression is often of the form X:Y,
    in which case its columns are simply scalars. Some examples
    (assume N has already been assigned a value).

         FOR I = 1:N,
             FOR J = 1:N,
                 A(I,J) = 1/(I+J-1);
             END
         END

    FOR S = 1.0: -0.1: 0.0, END steps S with increments of -0.1
    FOR E = EYE(N), ... END  sets E to the unit N-vectors.

    Long loops are more memory efficient when the colon expression appears
    in the FOR statement since the index vector is never created.

    The BREAK statement can be used to terminate the loop prematurely.

    See also IF, WHILE, SWITCH, BREAK, END.


>> help while

 WHILE  Repeat statements an indefinite number of times.
    The general form of a WHILE statement is:
```

```
WHILE expression
  statements
END
```

The statements are executed while the real part of the expression has all non-zero elements. The expression is usually the result of expr rop expr where rop is ==, <, >, <=, >=, or ~=.

The BREAK statement can be used to terminate the loop prematurely.

For example (assuming A already defined):

```
E = 0*A; F = E + eye(size(E)); N = 1;
while norm(E+F-E,1) > 0,
   E = E + F;
   F = A*F/N;
   N = N + 1;
end
```

See also FOR, IF, SWITCH, BREAK, END.