LABORATORIO 1----- 04/03/2005

Argomenti trattati:

- **Note introduttive a MATLAB**
- **Scalari e vettori in MATLAB**
- **Istruzioni if, while, for…**
- **Grafico di funzioni**

Alcune informazioni su MATLAB

MATLAB e` **uno strumento per il calcolo scientifico** utilizzabile in modo a più livelli, dalla calcolatrice tascabile, alla simulazione ed analisi di sistemi complessi.
Il nome **MATLAB** e` un'abbreviazione di **MAT**RIX**-LAB**ORATORY: la struttura di base e` la matrice ovvero ogni quantità (variabile) viene trattata come una matrice di dimensioni $m \times n$. Uno scalare reale e` una matrice 1×1.

Qual e` il vantaggio di lavorare in un ambiente in cui la matrice e` l'elemento di base del calcolo?

- non e` necessario dichiarare esplicitamente all'inizio del lavoro una variabile matrice in termini delle sue dimensioni e del tipo dei suoi coefficienti (interi, reali, complessi…) → notevole semplificazione
- e` già predefinito un ampio insieme di matrici elementari (matrice identità, matrice nulla…) → matrici più complesse possono essere costruite rapidamente partendo da queste matrici fondamentali
- sono predefiniti vari operatori algebrici di uso comune, quali ad esempio somma, prodotto, elevamento a potenza, nonché il calcolo del determinante o del rango di una matrice → un'efficiente implementazione di queste operazioni influisce notevolmente sui tempi di calcolo.

Inoltre sono predefinite numerose funzioni primitive, dette built-in functions. Esse permettono di risolvere in modo rapido ed efficiente problemi complessi, ad esempio il calcolo degli autovettori ed autovalori di una matrice.

Le raccolte di funzioni predefinite dedicate ad uno specifico argomento vengono dette toolboxes. La finanza, la statistica, l'analisi dei segnali e delle immagini sono alcuni dei campi a cui sono dedicati dei toolboxes di MATLAB.

Dove trovare dispense e informazioni su MATLAB?

- sul sito ufficiale di MATLAB www.mathworks.com

sono disponibili numerosi manuali (in inglese) sia introduttivi che dedicati più approfonditamente ad aspetti specifici (programmazione, grafica, toolboxes)

- sui siti di numerose università ed industrie sono riportati tutorial esempi di problemi di interesse pratico studiati con l'uso di MATLAB

Per iniziare…

All'avvio di MATLAB appare il prompt, ovvero la linea da cui digitare le istruzioni >>

Il comando >>demo mostra degli esempi significativi di possibili applicazioni del software.

L' help di MATLAB permette di ottenere informazioni dettagliate su qualsiasi comando. Ad esempio: >>help sqrt

Il solo comando >>help elenca gli argomenti per i quali e` disponibile la guida suddivisi in grandi aree tematiche (funzioni elementari, trattamento di matrici, grafica…)

Alcuni trucchi utili…

- e` possibile richiamare "storicamente" i comandi precedentemente digitati nella sessione di lavoro usando i tasti ↑,↓

- e` possibile spostarsi lungo la linea di comando corrente e modificare la riga scritta utilizzando i tasti →,←
- e` possibile completare un'istruzione già precedentemente digitata scrivendone le prime lettere e utilizzando poi il tasto ↑.

Alcuni comandi da conoscere

- l'istruzione >> diary mywork.dat apre il file di testo mywork.dat nel quale viene trascritto (a partire da quel momento) il flusso delle istruzioni digitali (e` una cronaca del lavoro svolto). L'istruzione >>diary off interrompe la scrittura della cronaca e chiude il file mywork.dat
- l'istruzione >>whos elenca le variabili attualmente attive in memoria e da` alcune informazioni importanti sulle loro caratteristiche
- l'istruzione >>save area.mat permette di salvare nel file binario area.mat il contenuto di *tutte* le variabili attive in memoria in quel momento.
- l'istruzione >>save area.mat z x le variabili x e z
- l'istruzione >>load area.mat ricarica le variabili salvate nel file area.mat e le rende attive in memoria (verificare con whos!)
- il comando >>quit termina la sessione di lavoro e chiude MATLAB.

Scalari e vettori in MATLAB

In Matlab non e` necessario definire e dichiarare le variabili. Per default, tutte le variabili vengono trattate in **doppia precisione**.

Iniziamo ad usare Matlab come una calcolatrice, ad esempio scriviamo z=3*2, assegnando così alla variabile z il valore 6.
Se scrivessimo solamente 3*2, il valore 6 verrebbe assegnato alla variabile **ans** (abbreviazione di answer). Tale variabile contiene sempre l'ultimo valore non esplicitamente assegnato dall'utente ad una variabile.

Variabili predefinite sono **pi** (pigreco), **i,j** (unità immaginarie), **eps** (epsilon macchina). Ogni variabile può essere tuttavia sovrascritta, ad esempio posso assegnare pi=5 (attenzione!). Per cancellare il valore di una variabile (o se è predefinita portarla al suo valore iniziale) usiamo il comando clear. Ad esempio

```
>>pi
   3.1416
>>pi=5;
>> clear pi
>> pi
     3.1416
```

Il comando clear all cancella il valore di tutte le variabili (provare ad usare tale comando in combinazione con il comando whos che elenca le variabili presenti nello spazio di lavoro).

Per introdurre un vettore colonna basta riportare fra parentesi quadre i valori della componenti del vettore stesso separati da un punto e virgola, ad esempio: v=[1;2;3]. Per introdurre un vettore riga e` sufficiente riportare i valori separati da spazi bianchi o virgole, ad esempio: w=[1 2 3] oppure w=[1,2,3].

Il comando v=[1:10] genera un vettore riga di dieci componenti dato dai valori 1,2,…,10. Il comando v=[1:.5:10] genera un vettore riga di venti componenti dato dai valori 1,1.5,2,2.5….,10, ovvero con passo 0.5.
La sintassi generale e` v=[valore_iniz:passo:valore_finale]. Il passo può essere anche negativo, ad ex. v=[10:-.5:1];

Per **accedere alla componente di un vettore**, ad esempio alla terza, e assegnare alla variabile z tale valore, scriviamo z=v(3). **Attenzione: la numerazione inizia da 1 e non da zero!**
Nota: esistono in Matlab le parole chiave start e end per accedere rispettivamente al primo e all'ultimo elemento di un vettore. Ad ex. v(start) equivale a v(1), mentre v(end) equivale a v(10) se v ha dieci elementi.

Matlab produce un messaggio di errore quando si cerca di accedere ad una componente non definita, ad esempio se v ha dieci elementi e vogliamo accedere a v(11), oppure se vogliamo accedere a v(0).

Per controllare la dimensione di una variabile, usiamo il comando size, ad esempio size(v). Questo comando e` anche utile quando Matlab segnala un conflitto di dimensioni fra quantità che si vogliono manipolare.

Il comando zeros(n,1) produce un vettore colonna di dimensione n con elementi tutti nulli. Il comando zeros(1,n) produce un vettore riga di dimensione n con elementi tutti nulli. Il comando ones(n,1) genera un vettore colonna con tutte le componenti pari a 1.

Operazioni su vettori

- **il modulo di un vettore v** e` dato da $\|\mathbf{v}\|=(\mathbf{v},\mathbf{v})^{1/2}$ e viene calcolato con il comando norm(v) (equivalente alla norma 2 del vettore: norm(v,2)). Per calcolare la norma infinito di un vettore, il comando e` norm(v,inf).
- **trasposizione** di un vettore **w** di dimensione $(n\text{x}1)$, w': $(1\text{x}n)$
- **elevamento a potenza componente per componente:** ex. vogliamo calcolare il cubo di ciascuna componente. Usiamo la sintassi v.^3, il cui risultato fornisce il vettore $(v_1^3, v_2^3, \ldots, v_n^3)$

Operazioni fra vettori della stessa dimensione: siano **v, w** vettori riga di R^n, con componenti $\{v_i\}$ e $\{w_i\}$, rispettivamente. Definiamo:

- **somma algebrica**

  $\mathbf{v+w}=(v_1+w_1, \ldots, v_n+w_n)$. In MATLAB tale operazione si esegue scrivendo v+w.

- **prodotto scalare tra due vettori**

  $(\mathbf{v},\mathbf{w})=(v_1w_1+v_2w_2+\ldots+v_nw_n)$

In MATLAB tale operazione tra vettori della stessa dimensione si esegue scrivendo v*w'

- **prodotto componente per componente (attenzione: differente dal prodotto scalare!)**

  (v,w)=(v_1w_1, v_2w_2,… ,v_nw_n)

  In MATLAB tale operazione si esegue scrivendo v.*w

Un scalare e` un vettore di dimensione 1. Quindi abbiamo: la somma a+b, la sottrazione a-b, il prodotto a*b, la divisione a/b, la potenza a^b.

Istruzioni if, while, for…

```
if espressione1
    istruzione1
elseif espressione2
    istruzione 2
else
    istruzione 3
end
```

```
for variabile = espressione
    istruzione
    ...
    istruzione
end
```

```
while espressione
      istruzione
end
```

ESEMPI

```
n=5;
a = zeros(n,1);
for i = 1:n
      if (i==1)
        a(i) = 1/i;
      else
```

```
        a(i) = 1/(i-1);
      end;
end;


n=5;
a = zeros(n,1);
i=1;
while(i<=n)
      if (i==1)
        a(i) = 1/i;
      else
        a(i) = 1/(i-1);
      end;
      i=i+1;
end;
```

Funzioni definite come stringa

Esiste in Matlab una sintassi particolare che permette di **definire una funzione in modo simbolico**. Per esempio, definiamo:
f='sin(x)+x.^2' (attenzione alla sintassi con gli apici e i punti e attenzione all'operazione di elevamento a potenza componente per componente!)
Tale funzione è una stringa, ovvero ad essa non sono associati dei valori numerici (verificare con whos f).
Se ora vogliamo associare ad essa dei valori numerici, scriviamo per esempio (si noti il fatto che f=f(x))

```
x=0:0.01:2*pi;
y=eval(f);
```

Il comando eval permette di **assegnare ad f dei valori numerici** in corrispondenza degli elementi del vettore x. Tali valori numerici vengono conservati nel vettore y (verificare con whos y).

Grafico di funzioni

Dati due vettori x,y di uguali dimensioni, il comando plot(x,y) traccia un grafico della funzione y=y(x) basandosi sui valori rispettivi contenuti nei due vettori (provare a scrivere plot(x,y,'o') per convincersi che la funzione viene definita per punti!).

Ex:.

```
x=0:pi/10:pi;
y1=sin(x); y2=cos(x) ;
plot(x,y,'r',x,y2,'g')
```

Esercizio

1. Definire le seguente variabili:
   - x=vettore di estremi 0 e 10, con passo 0.1
   - fun = 'exp(x)'
2. Stabilire il tipo degli oggetti definiti:
   - Che tipo di oggetto e` x ?
   - Che tipo di oggetto e`fun ?
   - Che tipo di oggetto e`eval(fun) ?
3. A partire dalle precedenti definizioni, quali fra le seguenti istruzioni producono un grafico della funzione y=exp(x), nell intervallo [0,10] ? Verificare con MATLAB le risposte.
   - plot(x,y)
   - plot(x,fun)
   - plot(eval(x),eval(fun))
   - plot(x,eval(fun))