Laboratorio di Fondamenti di Automatica
Prima esercitazione

Introduzione a MATLAB

Premessa

- Scopo di quest'esercitazione di laboratorio:
  - introdurre l'ambiente di calcolo MATLAB, con particolare riferimento al suo uso nel contesto dell'Automatica.
- Contenuto dell'esercitazione:
  - descrizione generale di MATLAB,
  - comandi di base,
  - comandi relativi a vettori, matrici e polinomi,
  - principali funzioni predefinite,
  - comandi grafici;

Descrizione generale di MATLAB

- MATLAB (MATrix LABoratory) è un ambiente (software) per il calcolo composto da
  - un nucleo computazionale 'general purpose';
  - un insieme di 'toolbox' che lo adattano a diversi campi applicativi (controllo, elaborazione dei segnali, statistica, chimica, finanza, ...);
  - un interprete per eseguire comandi e/o programmi (m-file) in un linguaggio procedurale;
  - molte altre cose (strumenti per creare GUI, per disegnare schemi a blocchi, per generare codice C, ...) che non tratteremo.

Un breve inciso

- MATLAB è un software proprietario prodotto da The Mathworks. Esistono alternative open-source, tra cui si segnala in particolare Scilab (www.scilab.org).
- Tra MATLAB e Scilab vi sono alcune differenze sintattiche ma non ve ne sono di sostanziali.
- In questo corso si adopera MATLAB. Tuttavia, tutto quel che si fa qui si può fare anche con Scilab. Chi è interessato è perciò invitato a scaricare Scilab (per Windows o Linux) dal sito citato, installarlo e usarlo (e magari studiarlo, visto che l'open source serve anche a questo).
- Detto ciò, torniamo a MATLAB.

Lanciate MATLAB e attendete il prompt (»).

Comandi di base

- MATLAB può fare da 'calcolatrice', nel senso che dal suo prompt è possible valutare espressioni numeriche. Esempio:

```
>>1+2
ans =
     3
```

- Il risultato viene scritto nella variabile **ans**. E' ovviamente possibile assegnarlo a un'altra variabile (per la quale non occorre dichiarazione). Esempio:

```
>>pippo=sqrt(-1);
>>pippo
pippo =
       0 + 1.0000i
```

- Si noti che il punto e virgola non fa stampare l'output del comando e che MATLAB lavora indifferentemente con numeri reali e complessi.

Definizione di variabili

- E’ possibile definire variabili e espressioni non numeriche piu’ complesse.

  Esempio:

```
>>a=4; b=2;
>>a*b
ans =
     8
```

Il Workspace

- Ogni variabile definita in questo modo viene conservata in memoria (nel lessico MATLAB, nel **workspace**).
- Il comando ***whos*** mostra una lista delle variabili definite e le loro dimensioni in memoria (esiste anche **who**, che non mostra le dimensioni):

```
>>whos
  Name        Size                Bytes  Class
  ans         1x1                     8  double array
  a           1x1                     8  double array
  b           1x1                     8  double array
  Grand total is 3 elements using 24 bytes
```

- Per cancellare una variabile (es. b):

```
>>clear b
```

- Per cancellare tutto il contenuto del workspace:

```
>>clear
```

Vettori e matrici

- Per definirli li si mette tra quadre ricordando che la virgola o lo spazio sono il separatore di colonna entro una riga, il punto e virgola separa le righe e l'apice postfisso traspone.
- E' permesso raggruppare (sotto)vettori e (sotto)matrici.

Esempio:

```
>>A=[1 2;3,4];
>>b=[5 6]';
>>c=[7;8];
>>d=[9 10,11 12];
>>[A b,c;d]
ans =
      1     2     5     7
      3     4     6     8
      9    10    11    12
```

Provate esempi simili (5 min) e accertatevi di aver capito.

Operazioni principali su vettori e matrici

- Somma e prodotto: **+** e *****
- Prodotto elemento per elemento: **.*** (***** preceduto da un punto)
- Determinante (di matrice quadrata **M**): **det(M)**
- Inversa (di matrice quadrata **M**): **inv(M)**
- Autovalori (di matrice quadrata **M**): **eig(M)**, che invocata con due lvalue, ad esempio **[A,a]=eig(M)**, fornisce gli autovalori (in una matrice diagonale **a**) e gli autovettori (nelle colonne di una matrice **A**)

Esempi:

```
>>M=[1 2;2 1];
>>B=inv(M);
>>B*M
ans =
      1     0
      0     1
```

```
>>[A,a]=eig(M);
>>inv(A)*M*A
ans =
     -1     0
      0     3
```

Operazioni principali su vettori, matrici e polinomi

- Matrice diagonale con elementi **m1...mn**: **diag([m1,... mn])**.
- Matrice identità di ordine **n**: **eye(n)**
- Polinomio caratteristico (di matrice quadrata **M**): **poly (M)**, che ritorna i coefficienti del polinomio come un vettore ordinato per potenze decrescenti della variabile del polinomio (questo è il modo standard per rappresentare i polinomi in MATLAB).
- Radici di un polinomio (già che ci siamo): **roots(p)**, dove **p** è il vettore riga dei coefficienti.
- Prodotto di convoluzione di due vettori **v1** e **v2** (cioè vettore dei coefficienti del polinomio prodotto di quelli che hanno **v1** e **v2** per vettori dei coefficienti): **conv(v1,v2)**.
- Facciamo alcuni esempi e commentiamoli.

Operazioni principali su vettori, matrici e polinomi

- Creiamo la matrice

```
>>A=[1 2;2 1];
```

- Calcoliamone gli autovalori come radici del polinomio caratteristico e creiamo la relativa matrice diagonale:

```
>>v=roots(poly(A)); (v è il vett. degli autovalori)
>>D=diag(v);
```

- Calcoliamo ora una matrice **M** che diagonalizza **A**:

```
>>[M,mv]=eig(A); (mv è la m. diag. degli autovalori)
```

- Verifichiamo:

```
>>inv(M)*A*M-mv
ans =
 1.0e-015 *
      0          0
      0    0.4441
```

... come si vede, è praticamente la matrice nulla.

Operazioni principali su vettori, matrici e polinomi

- Capiamo bene la distinzione tra vettori riga e colonna:

```
>>r=[1 2 3]; c=[4 5 6]';
>>r*c
ans =
     32
>>c*r
ans =
      4     8    12
      5    10    15
      6    12    18
>>r.*c
??? Error using ==> .*
Matrix dimensions must agree.
>>r.*c'
ans =
      4    10    18
```

Operazioni principali su vettori, matrici e polinomi

- Deteminamo le radici del polimimio $(1+x)(3x+x^2-5)$:

```
>>roots(conv([1 1],[1 3 -5]))
ans =
    -4.1926
    -1.0000
     1.1926
```

- Risolviamo il sistema lineare nelle incognite **a**, **b**, **c** le cui equazioni sono **a+c=2**, **2a+b+3=0**, **a-b+c=0**:

```
>>M=[1 0 1;2 1 0;1 -1 1];
>>x=inv(M)*[2 -3 0]' (x conterrà a, b, c in colonna)
x =
    -2.5000
     2.0000
     4.5000
```

Operazioni principali su vettori, matrici e polinomi

- Accesso a parti di matrici e vettori (si noti che gli indici sono basati a 1):

```
>>A=[1 2 3;4 5 6;7 8 9];
>>A(1,:) (prima riga)
ans =
      1     2     3
>>A(:,3) (terza colonna)
ans =
      3
      6
      9
>>A(2:3,:) (due righe contigue)
ans =
      4     5     6
      7     8     9
```

Operazioni principali su vettori, matrici e polinomi

```
>>A(2:3,1:2) (sottomatrice contigua)
ans =
      4     5
      7     8
>>A([1 3],[1 3]) (sottomatrice non contigua)
ans =
      1     3
      7     9
```

Provate esempi simili (5 min) e accertatevi di aver capito.

Principali funzioni predefinite

- Matematiche:
  - **sqrt, sin**, **cos**, **tan**, **atan**, **exp**, **log**, **log10**, **abs**, **angle**, **real**, **imag**, ...;
  - nota: l'unità angolare è il radiante.
- Relative alle stringhe:
  - **sprintf**, ...;
  - non le trattiamo perché servono essenzialmente nei programmi (m-file).
- Specializzate, ovvero provenienti da un toolbox:
  - vedremo più avanti quelle utili per l'analisi e la sintesi dei sistemi di controllo.
- MATLAB ha un potente sistema di help dove tutto è documentato (**help nomeComando**).

Principali funzioni predefinite

- Esempi:

```
>>p=[1 1 1];
>>r=roots(p)
r =
   -0.5000 + 0.8660i
   -0.5000 - 0.8660i
>>[abs(r) angle(r)*180/pi real(r) imag(r)]
ans =
    1.0000  120.0000   -0.5000    0.8660
    1.0000 -120.0000   -0.5000   -0.8660
```

Principali funzioni predefinite

- In MATLAB i vettori posono anche essere definiti come **valoreIniziale:valoreIncremento:valoreFinale**, il che è utile quando li si usa per esprimere funzioni (per esempio segnali).
- Tutte le funzioni matematiche operano anche sui vettori (attenzione al punto nei prodotti elemento per elemento).
- Esempio: calcoliamo la funzione $\mathbf{y=\sin(0.1t^2)}$ per t da 0 a 20 a passi di 0.01.

```
>>t=0:0.01:20;
>>y=sin(0.1*t.^2);
```

- Esistono anche **linspace** (poco usato) e **logspace**, che definiscono vettori spaziati linearmente o logaritmicamente:

```
>>v=logspace(-2,3,100);   (100 punti tra 1e-2 e 1e3)
>>logspace(0,1,3) (guardare il risultato per capire)
```

Comandi grafici

- Il principale comando per i grafici 2D è **plot.**
- Non perdiamo tempo con le sue moltissime opzioni (si veda **help plot**); un po' le vedremo strada facendo.
- Alcuni esempi:

```
>>x=0:pi/100:2*pi;
>>y1=sin(2*x);
>>y2=cos(x).*exp(-x);
>>plot(x,y1,x,y2);
>>figure(2);
>>plot(x,y1,'r',x,y2,'b:');
```

Comandi grafici

- Proseguiamo imparando a impostare gli assi e suddividere i grafici (e anche qualcos'altro):

```
>>figure(1);
>>axis([x(1) x(end) -2 2]); (notare x(end))
>>figure(2);
>>clf (ripulisce la figura)
>>clc (ripulisce la finestra comandi)
>>subplot(211); plot(x,y1);
>>subplot(212); plot(x,y2);
```

- I tre numeri argomento di **subplot** sono il numero di righe e colonne in cui suddividere la finestra e la posizione del grafico su cui si vuole agire (1 è il primo in alto a sinistra e quindi la numerazione procede per righe).

Provate esempi simili (5 min) e accertatevi di aver capito.