Laboratorio di Telecomunicazioni - a.a. 2010/2011

Introduzione all'uso di MATLAB

L.Verdoliva

Questa breve introduzione ha come obiettivo quello di fornirvi gli elementi minimi del Matlab (MATrix LABoratory), di cui faremo uso per lo studio e la sperimentazione dei concetti di base dell'analisi numerica. Chi è interessato ad una guida completa può trovare il manuale, disponibile in formato elettronico, al sito: http://www.mathworks.com/access/helpdesk/help/techdoc/matlab.shtml.
Leggete questo documento e provate tutte le istruzioni che troverete di seguito.

1 Concetti base

Lanciando Matlab si apre la finestra operativa (*Command Window*), da cui è possibile digitare i comandi e lavorare in maniera interattiva. Per esempio, si può usare Matlab come se fosse una semplice calcolatrice, digitando

```
» 4+5*2
```

e premendo INVIO otterremo la risposta. In questo modo, si perde il risultato dell'operazione che però piò essere memorizzato in una variabile nel seguente modo:

```
» a=4+5*2
```

E' importante sottolineare che, a differenza di altri linguaggi di programmazione, non è necessario specificare il tipo della variabile usata, che risulta automaticamente definita in seguito all'assegnazione dei valori che deve assumere. Digitando il nome della variabile

```
» a
```

è possibile conoscerne il valore. Se si fa seguire "`;`" alla linea di comando si impedisce al Matlab di stampare a video l'output del comando stesso.
L'elenco di tutte le variabili usate viene automaticamente memorizzato da Matlab nel *Workspace*, che mostra per ogni variabile il nome, la dimensione e l'occupazione in memoria. In alternativa è anche possibile visualizzare l'insieme di tutte le variabili presenti in memoria, e la loro dimensione, attraverso il comando `whos` o cancellarle mediante `clear`.

```
» whos;
```
mostra l'insieme delle variabili definite, il loro tipo, e la loro occupazione di memoria. Da notare che il tipo assegnato per default alle variabili è double.

```
» clear a;
```
cancella la variabile `a` dalla memoria; `clear all` cancella tutte le variabili definite.

In realtà quest'ultima operazione può anche essere effettuata selezionando la variabile nella finestra del Workspace e schiacciando il pulsante `delete`.

2 Variabili e operazioni elementari

La variabile base in Matlab è la matrice, che può essere definita nel seguente modo:

» `A=[1,2,3; 4,5,6]`
: il carattere "," (virgola), che può anche essere sostituito da uno spazio, separa gli elementi di una stessa riga, mentre ";" (punto e virgola) separa le righe presenti. Quindi con questo comando si definisce una matrice `A`, 2×3, e si assegnano alla prima riga i valori $\{1\,2\,3\}$ e alla seconda $\{4\,5\,6\}$.

Per accedere ad un elemento della matrice basta fornire l'indice di riga e quello di colonna con il seguente comando

» `a=A(1,3)`
: definisce la variabile `a` e vi assegna il valore 3 (riga 1, colonna 3);

Due casi speciali di una matrice sono lo scalare (matrice costituita da un solo elemento) e il vettore (matrice costituita da una sola riga o una sola colonna).

» `b=3;`
: definisce la variabile `b` e vi assegna il valore 3.

» `c=[1,2,3,4,5,6];`
: definisce un vettore riga di sei elementi con valori 1,2,3,4,5 e 6; se al posto delle virgole si lasciano degli spazi si ottiene lo stesso risultato. Analogamente alle matrici con `c(4);` si accede al valore di posto 4.

» `c=[1;2;3;4;5;6];`
: definisce un vettore colonna di sei elementi.

Il comando `c=[1,2,3,4,5,6];` può essere abbreviato con `c=[1:1:6];` che genera tutti i numeri fra 1 e 6 (inclusi) con passo 1. Nel caso particolare in cui, come adesso, il passo sia unitario, lo si può omettere, e scrivere quindi `c=[1:6];`.

L'operatore ":" (colonna) è anche molto utile per estrarre una sottomatrice da una matrice. Supponiamo per esempio di aver definito una matrice D 9×8; per estrarre delle sottomatrici da D si possono usare i seguenti comandi:

» `D(2:5,1:3);`
: estrae una matrice 4×3 da D le cui righe vanno dalla seconda alla quinta, mentre le colonne dalla prima alla terza.

» `D(2,:)`
: estrae la seconda riga dalla matrice; per estrarre la seconda colonna basta digitare `D(:,2)`.

» `D(:)`
: fornisce un vettore colonna 72×1 costituito dalle colonne di `D` concatenate assieme.

Di seguito si elencano alcune operazioni molto utili per la definizione di matrici:

» `C=[A,B];`
definisce la matrice `C` affiancando le due matrici `A` e `B`, che devono avere lo stesso numero di righe.

» `C=[A;B];`
definisce la matrice `C` sovrapponendo le due matrici `A` e `B`, che devono avere lo stesso numero di colonne.

» `A=zeros(N,M);`
definisce una matrice di `N` righe ed `M` colonne con elementi tutti nulli. E' anche possibile definire una matrice quadrata, $N \times N$, di zeri con il comando `A = zeros(N)`.

» `A=ones(N,M);`
definisce una matrice di `N` righe ed `M` colonne con elementi tutti unitari. E' anche possibile definire una matrice quadrata, $N \times N$, di valori unitari con il comando `A = ones(N)`.

» `A=eye(N);`
definisce una matrice identità quadrata di `N` righe ed `N` colonne.

» `B=A.';`
definisce B come trasposta di A.

» `B=A';`
definisce `B` come hermitiana (trasposta e coniugata) di `A`. Per le matrici reali coincide con la trasposta.

Essendo la variabile base una matrice le operazioni aritmetiche sono automaticamente quelle definite su matrici. Questo significa che il comando `A*B` effettua la moltiplicazione matriciale tra `A` e `B`, e quindi il numero di colonne di `A` deve essere uguale al numero di righe di `B`. Si ricordi inoltre che tale operazione non gode della proprietà commutativa per cui bisogna stare molto attenti sia all'ordine con cui si realizza che alle dimensioni delle due matrici.

In ogni caso è possibile effettuare la moltiplicazione "puntuale" tra due matrici (delle stesse dimensioni) nel senso di moltiplicare elemento per elemento i valori che si trovano nella stessa posizione. Per fare ciò basta anteporre il "." (punto) al simbolo *. Questa regola vale in generale: ogni volta che si antepone il punto ad un operatore aritmetico l'operazione non è più di tipo matriciale. Fate attenzione allora: la notazione `A^2` equivale a realizzare la moltiplicazione matriciale di A con se stessa; se si vuole semplicemente elevare al quadrato tutti gli elementi di X bisogna scrivere `A.^2`.

Esistono moltissime funzioni che permettono di estrarre utili informazioni dalle matrici, di seguito se ne mostrano alcune:

» `[M,N]=size(A);`
restituisce in `M` ed `N` il numero di righe e di colonne della matrice `A`; con `size(A)` tali dimensioni vengono mostrate su schermo.

» `L=length(a);`
restituisce in `L` la lunghezza del vettore `a`.

» `B=fliplr(A);`
inverte l'ordine delle colonne della matrice `A`; `flipud(A)` opera in modo analogo sulle righe.

» `min(A)`
se `A` è un vettore ne restituisce il valore minimo; se è una matrice calcola i minimi lungo ogni colonna e li restituisce in un vettore riga. Per ottenere il minimo della matrice usare `min(min(A))` oppure `min(A(:))`. Se si vuole memorizzare tale valore in una variabile scrivere `m=min(min(A))`. Analogamente `max(A)` restituisce il massimo.

» `mean(A)`
restituisce il valor medio, funziona come `min(A)`.

» `B=cos(A);`
definisce `B` come la matrice contenente il coseno di ogni elemento di `A`, opera quindi punto per punto. Si noti come la funzione elementare coseno abbia per default un argomento che è di tipo matrice. Questa considerazione vale per numerose funzioni matematiche: `B=abs(A), B=log(A), B=sign(A), B=exp(A)`.

Se si vogliono salvare i dati memorizzati su di un file, basta digitare il comando

» `save dati A B;`
si crea nella directory corrente un file dal nome `dati.mat` che contiene i dati delle due matrici.

» `save dati;`
si crea nella directory corrente un file dal nome `dati.mat` che contiene tutte le variabili presenti nel Workspace.

Fate attenzione perché i dati vengono salvati nella directory corrente. Premete allora il tasto *Current Directory* e posizionatevi nella directory di lavoro prima di cominciare a lavorare. Le variabili salvate possono essere rielaborate nuovamente con il matlab una volta caricate in memoria con il comando

```
» load dati;
```

N.B. Un'utile funzionalità del Matlab è l'uso del tasto "freccia in alto" (↑), che permette di ridurre notevolmente i tempi di immissione dei dati: infatti premendo questo tasto si ottengono le istruzioni precedentemente digitate, che si possono eventualmente modificare; lo stesso comando può agire in maniera più selettiva: digitando una lettera o un gruppo di lettere e poi ↑, vengono riproposti solo i comandi che iniziano con la lettera o il gruppo di lettere digitati.

3 Operazioni logiche

Le operazioni logiche sono molto utili in Matlab dato che permettono spesso di evitare i cicli for. Supponiamo allora di aver definito il seguente vettore:

```
x=[1 2 -3 3 -2 1 4 0 -1];
```

L'operazione logica:

```
y=x>0;
```

fornisce un vettore in cui ogni elemento vale 1 (VERO) solo se i corrispondenti elementi del vettore sono maggiori di zero, altrimenti restituisce 0 (FALSO). In questo caso:

```
y=[1 1 0 1 0 1 1 0 0];
```

Allo stesso modo il comando

```
y=(x==0);
```

dà in uscita un vettore in cui ogni elemento vale 1 (VERO) laddove gli elementi del vettore sono pari a zero. A questo punto il vettore `y` può essere utilizzato come una maschera per ottenere un vettore `z` in cui sono stati annullati tutti gli elementi minori di zero nel seguente modo:

```
y=x>0;
z=x.*y;
```

Si può usare anche la funzione `find`, per determinare la lista degli indici in un vettore per cui una determinata condizione è verificata. Per esempio

```
i=find(x>0);
```

restituisce la lista degli indici dove il vettore `x` è maggiore di zero.

4 I grafici in Matlab

Vediamo adesso come si traccia un grafico in Matlab. Cominciamo col rappresentare una sequenza di punti definita mediante il vettore `x=[0 0.3 0.5 0.8 1 0.8 0.5 0.3 0]`, avendo definito l'asse dei tempi `t=[1:0.5:5]`:

» `stem(t,x)`
: rappresenta la sequenza di valori di `x` come impulsi centrati negli istanti definiti dal vettore `t`;

» `plot(t,x)`
: realizza il grafico interpolando linearmente i valori contenuti in `x`;

» `subplot(m,n,p)`
: consente di visualizzare più grafici all'interno della stessa figura. Suddivide la figura in `mxn` quadranti e inserisce il grafico corrente nella posizione `p`-esima.

» `axis([xmin xmax ymin ymax])`
: permette di stabilire il range mostrato sull'asse delle ascisse (da `xmin` a `xmax`) e su quello delle ordinate (da `ymin` a `ymax`).

» `title('Titolo della figura')`
: permette di inserire un titolo alla figura.

» `close`
: chiude la figura. Il comando `close(all)` chiude tutte le figure.

Una figura in Matlab può poi essere ulteriormente modificata premendo l'ultimo tasto presente nella finestra (*Show Plot Tools*).

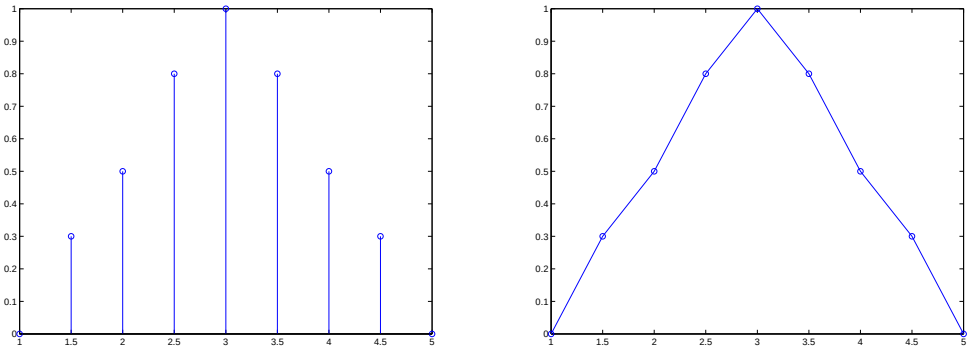

Figura 1: I comandi stem e plot

5 I costrutti di programmazione

Matlab è un linguaggio interpretato e non compilato, vale a dire che ogni istruzione che voi scrivete deve essere trasformata in comandi più semplici da un interprete al momento dell'esecuzione, e poi eseguita. Questo fatto si riflette in una maggiore lentezza di esecuzione rispetto ad un linguaggio compilato. Tuttavia, Matlab possiede istruzioni molto potenti per la manipolazione di vettori e matrici. Se riuscite a fare ricorso il più possibile a queste istruzioni, evitando di utilizzare cicli `for` che accedono singolarmente a ciascun elemento di un vettore, i tempi di esecuzione miglioreranno sensibilmente.
Come esempio, si considerino i seguenti tre metodi per sommare tutti gli elementi di una matrice `X`

1. ```
   [M,N] = size(X);
   Xsum=0;
   for i=1:M
     for j=1:N
        Xsum = Xsum + X(i,j);
     end
   end
   ```
2. `Xsum=sum(sum(X));`
3. `Xsum=sum(X(:))`.

Il primo modo è equivalente al codice della programmazione convenzionale (Fortran, Pascal, C) e fa uso di un doppio ciclo for per scorrere tutti gli elementi della matrice. Il secondo usa la funzione `sum` di Matlab che funziona come `min` e va quindi applicata due volte se applicata ad una matrice, a meno di non concatenare le righe di tale matrice in un vettore con l'operatore colonna. Quest'ultima soluzione è la più efficiente: basta un'unica chiamata a funzione e non sono presenti cicli for.

6 L'Help in linea del Matlab

L'uso del Matlab è estremamente facilitato dall'Help in linea che permette di conoscere quali funzioni sono disponibili nei toolbox, *cosa* fanno e *come* devono essere usate.
   ```

`» help`
: invoca l'help in linea di Matlab e restituisce la lista di argomenti sui quali è possibile avere informazioni; è una buona idea usarlo almeno una volta.

`» help help`
: spiega come usare l'help e indica altri utili comandi, es `lookfor`.

`» help <topic/command>`
: invoca l'help su un particolare argomento o comando, e suggerisce una lista di argomenti o comandi correlati con quello richiesto.

Provate a digitare `help` seguito dalle funzioni che avete già provato a usare, come `mean, min, max, ...` per verificarne il funzionamento ed avere altre utili indicazioni. Chiamate sempre l'help in linea quando avete dei dubbi su come deve essere usato un determinato comando.

7 Uso degli script e delle funzioni

Per eseguire un blocco di istruzioni è necessario creare uno *script* Matlab, cioè un file con estensione `.m` nel quale siano inserite tutte le istruzioni che si voglione eseguire. Questo può essere fatto creando un nuovo file dall'editor e digitando le istruzioni, premendo il tasto destro del mouse e selezionando *Create M-File*. Dopodichè é sufficiente lanciare lo script (scrivendone il nome senza estensione da riga di comando, oppure premendo `F5` dall'editor). Durante le esercitazioni provate sempre a creare degli script piuttosto che a digitare le istruzioni da linea di comando: in questo modo risparmierete tempo quando volete ripetere e/o correggere dei comandi.

In alternativa si possono creare delle vere e proprie funzioni che realizzano le operazioni di cui abbiamo bisogno e che non sono già presenti in Matlab. Facciamo subito un esempio scrivendo la funzione (banale) per il calcolo della somma degli elementi di una matrice.

```
function s=matsum(X)
% MATSUM somma tutti gli elementi di una matrice
% s=matsum(X)
% X: matrice di ingresso
% s: valore di uscita

% controllo sulle dimensioni della matrice
[M N] = size(X);
if( M == 1 & N == 1 )
  error ('errore!');
end
s=sum(X(:));
```

L'M-file che contiene la funzione deve avere il nome della funzione stessa, cioè in questo caso `matsum.m`. Per eseguire la funzione basta richiamarla da linea di comando, definendo opportunamente i parametri richiesti in ingresso (in questo caso solo `X`):

```
>> X = [1 2 3; 4 5 6; 7 8 9];
>> s = matsum(X);
```

Nella variabile `s` sarà memorizzato il risultato. Inoltre digitando `help matsum` verrà visualizzato tutto ciò che avete scritto nel commento usando il carattere % per descrivere il comportamento della funzione. Cercate quindi di documentare sempre e in modo appropriato le funzioni che scrivete in modo che possano essere utilizzate facilmente.

Provate adesso a scrivere uno script che effettua la somma degli elementi di una matrice e poi lanciate il programma da linea di comando. A questo punto osserverete che è possibile adottare anche questa nuova soluzione. Ma allora che differenza c'è nello scrivere una `function` piuttosto che uno `script`? Per comprendere cosa c'è di diverso, basta guardare nel workspace quali sono le variabili che sono state memorizzate nei due casi. Noterete come con lo script tutte le variabili coinvolte sono state memorizzate nel workspace, mentre con la funzione sono presenti solo le variabili in uscita. Questo significa che, in quest'ultimo caso, non avete accesso alle variabili definite nel corpo della funzione. Questa soluzione può sembrare restrittiva, in realtà, vi permette di poter utilizzare lo stesso nome per le varibili definite nelle funzioni senza che si crei alcun conflitto con variabili definite in altri programmi. Per esempio la variabile `M` può essere usata anche all'esterno della funzione con significato diverso, senza che sorga alcun problema.

7.1 Esercizi proposti

a) Scrivete una funzione in cui, data una matrice di valori positivi, reali e distinti tra loro, di dimensioni 16×16, volete conservare solo i 4 valori più grandi e annullare tutti gli altri. Il prototipo della funzione deve essere il seguente: `function y=elabora(x)`, dove `x` è la matrice in ingresso e `y` quella in uscita.

   Suggerimento. Usate la funzione `sort` di matlab che realizza l'ordinamento e gli operatori logici.

b) Scrivete una funzione che effettua l'operazione di saturazione dei dati (amplificatore reale), vale a dire se i dati contenuti in un vettore superano un certo valore limite (in modulo) essi vengono resi uguali proprio a tale valore. Il prototipo della funzione deve essere il seguente: `function y=clip(x, Limit)`, dove `x` è il vettore in ingresso, `Limit` è il valore limite e `y` il vettore in uscita. Per verificare la correttezza della funzione, scrivete uno script dal nome exe.m in cui definite il vettore di ingresso: $[-10\ \ 3\ \ -6\ \ 0\ \ 1\ \ -2\ \ 3\ \ 4\ \ -15\ \ 3\ \ 21]$, e il valore limite (in modulo) pari a 8, quindi chiamate le funzione `clip` e visualizzate il risultato.

8 Alcuni suggerimenti generali per l'uso di Matlab

1. Scrivete funzioni brevi, ma usate nomi significativi, anche se lunghi, per le variabili (in Matlab potete usare fino a 32 caratteri).

2. create un codice indentato e ben commentato, scoprirete presto quanto sia utile avere una buona documentazione dei programmi che si vogliono usare;

3. evitate i cicli for (quando è possibile);

4. se lavorate con matrici molto grandi, "preallocatele" in memoria prima di usarle, per esempio con il comando `A = zeros(M,N)`, per evitare che ci siano inefficienze durante l'esecuzione;

5. per la correzione degli errori (debugging) esaminate le variabili create nell'ambiente di lavoro con il comando `whos`;

6. fate attenzione al diverso significato delle parentesi tonde (per le operazioni aritmetiche) e quadre (per formare vettori o matrici);

7. attenzione anche all'uso delle lettere maiuscole e minuscole (per Matlab sono diverse);

8. usate `help`, potreste scoprire che la funzione che avete intenzione di scrivere esiste già!

9. infine, fate molta attenzione ai messaggi di errore che vi segnala il Matlab. Leggeteli sempre perché non solo vi danno informazioni sul tipo di errore, ma vi indicano anche il punto nel codice in cui il programma si blocca, e quindi verosimilmente è localizzato l'errore.

9 Come spostarsi in ambiente Matlab

Terminiamo questa breve introduzione elencando i principali comandi per la gestione dei file e delle directory:

`» cd`
: mostra la directory corrente.

`» dir`
: mostra il contenuto della directory corrente.

`» cd <nomedir>`
: porta nella nuova directory di lavoro nomedir.

`» delete <nomefile>`
: cancella il file nomefile dalla directory corrente.

`» what`
: elenca gli m-file presenti nella directory corrente.

`» diary('nomefile')`
: salva in nomefile tutto ciò che viene visualizzato su schermo.

`» exit`
: esce da Matlab, idem con quit.