

I Lezione: Il programma MATLAB

In questa sezione introdurremo in maniera molto breve il programma di simulazione MATLAB (una abbreviazione di MAtrix LABoratory) che è stato usato per lo sviluppo delle esercitazioni presentate in questa raccolta. Il programma è disponibile nei computer del laboratorio di interfaccoltà e della rete del Dipartimento di Ingegneria Elettrica, Gestionale e Meccanica dell'Università di Udine.

1 Introduzione

MATLAB è un programma interattivo di calcolo che permette di risolvere problemi numerici senza che sia necessario scrivere esplicitamente una procedura in un linguaggio di programmazione ad alto livello. MATLAB si basa sul calcolo matriciale, e possiamo pensare ad esso come ad un sistema efficace per accedere ad una libreria di procedure di calcolo numerico molto sofisticate, con in più la possibilità di rappresentare graficamente i nostri risultati. Ha una interfaccia molto semplice, ed un singolo comando permette di risolvere problemi come l'inversione di una matrice o la soluzione di un sistema di equazioni differenziali. Inoltre, una successione di comandi può essere raccolta in un file ed essere eseguita invocando, da dentro il programma MATLAB, il nome del file stesso: è poi possibile definire delle funzioni che forniscono dei valori in uscita, di modo che MATLAB risulta esso stesso programmabile e fornisce la possibilità di effettuare simulazioni e calcoli, anche molto complessi, in modo semplice e diretto.

Una guida completa a MATLAB è al di là degli scopi di queste note: si ricorda comunque che esso fornisce un sistema di aiuto in linea cui conviene ricorrere molto spesso. Dopo essere entrati in MATLAB, il comando `help` mostra l'elenco delle procedure per le quali è disponibile una descrizione; per avere informazioni specifiche su di un comando, occorre digitare `help nome_comando`. Ad esempio, con il comando `help exp` si ottengono informazioni sull'uso della funzione `exp` per il calcolo della funzione esponenziale. Per una dimostrazione delle potenzialità di MATLAB, si dia il comando `demo`. Le versioni più recenti di MATLAB permettono di accedere ad un sistema integrato, con una finestra di comandi, un editor, ecc. Una voce specifica `Help` nel menù, permette di accedere ad un sistema ipertestuale di aiuto in linea.

2 Matrici in MATLAB

MATLAB lavora essenzialmente con un unico tipo di dati: le matrici. Queste sono vettori bidimensionali, con m righe e n colonne in generale, i cui singoli elementi sono variabili reali oppure complesse. Una matrice può essere creata introducendo esplicitamente i suoi valori usando un comando di assegnazione, oppure come risultato di comandi o funzioni predefiniti. Ad esempio, il comando

```
a=[1 2 3; 4 5 6]
```

crea la matrice **a** di dimensioni 2×3 . Si noti che il punto e virgola separa una riga della matrice dalla riga successiva. Le variabili complesse sono gestite in maniera molto intuitiva: Il comando `a=[3+i*2 4]` crea un vettore con due elementi complessi.

MATLAB fornisce l'eco del risultato di ogni comando, a meno che questo non venga fatto seguire da punto e virgola. Ad esempio, il comando

```
a=[1 2 3; 4 5 6];
```

crea, come prima, la matrice **a**, senza che i suoi valori vengano stampati immediatamente dopo sul terminale. La possibilità di sopprimere l'eco è particolarmente utile quando si scrivono programmi che contengono sequenze di comandi, oppure quando le dimensioni di una matrice siano particolarmente grandi.

Le seguenti operazioni fra matrici sono disponibili in MATLAB:

+	addizione
-	sottrazione
*	moltiplicazione
^	elevazione a potenza
'	trasposto coniugato
\	divisione a sinistra
/	divisione a destra.

Ad esempio, date le matrici **A** e **b**, il comando `x=A\b` fornisce, nel vettore **x**, la soluzione del sistema di equazioni $Ax = b$. Analogamente, `x=b/A` è la soluzione di $xA = b$ (nel caso di matrice **A** rettangolare, MATLAB fornisce la soluzione del sistema ai *minimi quadrati*).

Le operazioni introdotte sono operazioni *matriciali*: qualora si vogliano effettuare operazioni sulle matrici *elemento per elemento*, gli operatori precedenti devono essere preceduti da un punto. Per esempio, il comando `[1 2].*[1 3]` fornisce il vettore `[1 6]`, mentre l'ommissione del punto darebbe luogo ad un messaggio di errore, dato che la dimensione delle matrici non è compatibile con l'operazione di moltiplicazione matriciale.

MATLAB permette di creare vettori o matrici con una notazione conveniente per specificare gli intervalli.

Ad esempio, il comando `a=1:0.1:10;` crea un vettore di 91 elementi, spazati uniformemente con passo 0.1, il cui primo elemento è il valore 1 e l'ultimo elemento è il valore 10. Per accedere all'elemento *i*-esimo, si usa la notazione `a(i)`, mentre per selezionare gli elementi dall'*i*-esimo al *j*-esimo si usa la notazione `a(i:j)`. Ad esempio, il comando `b=a(2:30)` crea un vettore **b** con gli elementi di **a** dal 2 al 30. Si noti che gli indici dei vettori in MATLAB iniziano da 1.

Più in generale, se **a** è una matrice $m \times n$, il comando `b=a(1:2, 2:3)` crea una matrice **b** di dimensioni 2×2 che contiene le righe dalla 1 alla 2 e le colonne dalla 2 alla 3 della matrice **a**.

3 Strutture di controllo

In MATLAB è possibile utilizzare strutture di controllo analoghe a quelle che sono disponibili nei linguaggi di programmazione (`if`, `for`, `while`).

Struttura if. Il comando:

```
if t>0
    x=1;
elseif t==0
    x=0.5;
else
    x=0;
end
```

fornisce, nella variabile `x`, il valore della funzione gradino a tempi continui, calcolata in `t`.

Struttura for. I comandi

```
[m n]=size(a);
for i=1:m
    for j=1:n
        c(i,j)=2*a(i,j);
    end
end
c
```

creano e visualizzano la matrice `c` ottenuta moltiplicando per 2 gli elementi della matrice `a`, come se si fosse scritto `c=2*A`. Si noti che MATLAB permette di moltiplicare una matrice per uno scalare senza ricorrere a notazioni particolari.

Struttura while. La sequenza di comandi

```
sum=1; add=x;
while abs(add)>=1e-3
    sum=sum+add;
    add=x*add;
end
```

calcola la somma della serie geometrica $\sum_{k=0}^{\infty} x^k$, $|x| < 1$, fermandosi quando il valore assoluto del termine corrente è minore di 10^{-3} .

I seguenti operatori di relazione e logici sono definiti in MATLAB:

<	minore
>	maggiore
<=	minore uguale
>=	maggiore uguale
==	uguale
~=	diverso
&	AND
	OR
~	NOT

Si noti la differenza fra il simbolo usato per l'assegnazione (=) ed il simbolo usato per il confronto (==).

4 Funzioni predefinite

Le funzioni in MATLAB operano su scalari o sui singoli elementi delle matrici. In quest'ultimo caso, il risultato è una matrice della stessa dimensione di quella cui è stata applicata la funzione. Ad esempio, il comando `b=sqrt(a)` crea una matrice `b` i cui elementi sono la radice quadrata degli elementi di `a`.

Fanno eccezione le funzioni tipo `max`, `min`, `sum`, `prod`, `mean`, `std`, etc., che hanno come argomento una matrice e ritornano un vettore. Ad esempio, `sum(a.^2)` dà come risultato un vettore che contiene la somma dei quadrati degli elementi di ciascuna colonna di `a`.

Di seguito, si dà un breve elenco delle funzioni che possono essere utilizzate all'interno di MATLAB. Per esempi ulteriori e spiegazioni, si utilizzi, come al solito, il comando `help nome_funzione`.

Funzioni per la creazione di matrici speciali (`help elmat`).

<code>eye</code>	matrice identità
<code>zeros</code>	matrice di zeri
<code>ones</code>	matrice di uno
<code>diag</code>	crea una matrice diagonale
<code>triu</code>	parte triangolare superiore
<code>tril</code>	parte triangolare inferiore
<code>rand</code>	genera numeri casuali
<code>magic</code>	genera un quadrato magico
<code>toeplitz</code>	crea una matrice di Toeplitz

Funzioni scalari (help elfun; help specfun).

sin	asin	exp	abs	round	besselj
cos	acos	log (log naturale)	sqrt	floor	erf
tan	sign	log10 (log base 10)	real	imag	gamma
atan	angle	log2 (log base 2)	rem (resto)	ceil	gcd

Funzioni con ingresso vettoriale (help datafun).

max	sum	median	any	cross
min	prod	mean	all	dot
sort	find	std	cov	corrcoef

Funzioni matriciali (help matfun).

eig	autovalori e autovettori
chol	fattorizzazione di cholesky
svd	singular value decomposition
inv	inversa
lu	fattorizzazione LU
qr	factorizzazione QR
expm	esponente di matrice
sqrtm	radice quadrata di matrice
poly	polinomio caratteristico
det	determinante
size	dimensioni
norm	norma-1, norma-2, norma- ∞
rank	rango
reshape	cambia le dimensioni di una matrice
rot90	rotazione di 90 gradi degli elementi

Elaborazione del segnale help datafun.

filter	filtraggio 1D
filter2	filtraggio 2D
conv	convoluzione 1D
conv2	convoluzione 2D
fft	calcolo DFT 1D via FFT
fft2	calcolo DFT 2D via FFT
ifft	DFT 1D inversa
ifft2	DFT 2D inversa
fftshift	posiziona lo zero dello spettro
quad8	calcolo numerico di integrali
roots	radici di un polinomio

5 Visualizzazione grafica e stampa

MATLAB può produrre sia grafici di funzioni monodimensionali che curve di livello e grafici di funzioni a più dimensioni. Per una dimostrazione delle potenzialità grafiche di MATLAB, si può dare il comando `demo`.

Se `x` e `t` sono due vettori della stessa dimensione, il comando `plot(t,x)` apre una finestra con il grafico di `x` in funzione di `t`. Ad esempio, i comandi

```
t=0:0.01:6.28; x=sin(t); y=cos(t); plot(t,x,t,y);
```

creano una nuova finestra con il grafico delle funzioni $x(t) = \sin(t)$ e $y(t) = \cos(t)$ fra 0 e 2π (approssimativamente). Si rifletta sui comandi utilizzati: con il primo, si crea un vettore `t` i cui elementi contengono, con passo 0.01, i valori da 0 a 6.28. Gli indici del vettore vanno da 1 a `length(t)` (la funzione `length` dà in uscita il numero di elementi nel vettore stesso). Il secondo e il terzo comando creano due nuovi vettori, con i valori delle funzioni seno e coseno calcolate nei punti di `t`. Infine, il comando `plot` produce i grafici, ottenuti unendo con tratti rettilinei le coppie $(t(i), x(i))$ $(t(i), y(i))$, $i=1:\text{length}(t)$. È possibile definire il titolo del grafico, le caratteristiche degli assi, posizionare del testo all'interno del grafico: a questo proposito, con il comando `help`, si vedano le descrizioni dei comandi `plot`, `title`, `xlabel`, `ylabel`, `gtext`, `text`, `zoom`.

Il comando `stem(t,x)` può essere utilizzato per graficare segnali a tempo discreto: anziché unire le coppie di valori $(t(i), x(i))$ con un tratto rettilineo, il comando disegna, in corrispondenza ad ogni `t(i)`, una barra verticale di lunghezza proporzionale al valore `x(i)`.

L'esecuzione di un comando `plot` cancella il grafico precedentemente disegnato nella finestra grafica. Per aprire una nuova finestra grafica, si deve usare il comando `figure`. Con `figure(n)` si rende attiva la finestra grafica numero `n`.

È inoltre possibile creare una copia su file del grafico della finestra grafica corrente, contenente i comandi necessari per la successiva stampa: il comando `print`, la cui documentazione è ottenibile con il comando `help print`, permette di gestire la maggior parte delle stampanti e dei linguaggi di descrizione grafica presenti nel mercato. Ad esempio, per salvare il grafico sul file `nomefile.eps` in formato Encapsulated PostScript (EPSF), si deve dare il comando

```
print -deps nomefile.eps
```

6 M-files

Come accennato nell'introduzione, è possibile raggruppare una sequenza di comandi MATLAB in un file esterno, creato con un editore di testi qualsiasi, che deve avere estensione `.m`. Se il nome del file è `nome_file.m`, dando il comando `nome_file` dall'interno di MATLAB, verrà eseguita la sequenza dei comandi contenuti nel file stesso. Tutti i listati della presente raccolta di esercitazioni sono contenuti in file `.m`. È opportuno far precedere i comandi di un M-file dal

comando `clear` che annulla lo spazio delle variabili create fino a quel momento in MATLAB: questo affinché i risultati od i nomi di variabili precedentemente utilizzati non interferiscano con quelli usati nel file `.m`. Per vedere quali sono, in ogni momento, le variabili definite in MATLAB e la loro dimensione, si usino i comandi `who` e `whos`. È inoltre conveniente poter aggiungere dei commenti nel file: a questo scopo, si può far precedere una riga dal carattere `%`, che forza MATLAB a non interpretare i caratteri successivi. Se un comando si estende oltre la lunghezza di una riga, esso può essere continuato nella riga seguente interrompendo con tre caratteri di interpunzione:

```
variabile_lunga= ...  
matrice_A.^2;
```

7 Funzioni esterne

È possibile aggiungere nuove funzioni al vocabolario di MATLAB. I comandi relativi ad una funzione devono essere contenuti in un file esterno, ancora con estensione `.m`. La prima riga del file deve contenere la lista dei parametri della funzione (che possono essere sia scalari che matrici) e la parola chiave `function`. Ad esempio, volendo scrivere una funzione che calcola la media, la potenza e la deviazione standard campionarie dei valori contenuti in una matrice, possiamo creare il file `stats.m`:

```
% STATS: calcola la media, la potenza e deviazione standard campionarie della  
% matrice di ingresso
```

```
function [m,p,s] = stats(x)  
    n = prod(size(x));  
    m = sum(sum(x)) / n;  
    p = sum(sum(x.^2)) / n;  
    s = sqrt(p - m^2);  
end
```

Per calcolare media, potenza e deviazione standard della matrice `a`, e porre il risultato nelle variabili `media`, `pot`, `stdv`, possiamo dunque invocare la funzione con il comando

```
[media,pot,stdv]=stats(a).
```

Si possono avere un qualsiasi numero di argomenti di uscita e di ingresso: questi ultimi vengono passati alla funzione *per valore* e non possono essere modificati al suo interno. Se la funzione ha un solo argomento di uscita, le parentesi quadre possono essere omesse. Ad esempio, la funzione precedente potrebbe essere riscritta come:

```
% STATS: calcola la media, la potenza e deviazione standard campionarie della
% matrice di ingresso
```

```
function y = stats(x)
    n = prod(size(x));
    y(1) = sum(sum(x)) / n;
    y(2) = sum(sum(x.^2)) / n;
    y(3) = sqrt(y(2) - y(1)^2);
end
```

e darebbe come risultato un vettore con i valori desiderati. Le variabili utilizzate all'interno della funzione e che non compaiono nella lista dei parametri sono *locali* alla funzione stessa, e possono pertanto avere lo stesso nome di variabili definite in altre procedure senza che si abbiano pericolose interferenze.

Si noti che le prime due linee del file sono dei commenti: esse vengono visualizzate da MATLAB, nel caso specifico, se si dà il comando `help stats`. In realtà, molte delle funzioni predefinite di MATLAB sono esse stesse definite all'interno di M-files usando la sintassi appena vista.

8 Esercizi per la prima lezione

All'apertura del programma, MATLAB si posiziona in una directory di default, evidenziata nella barra sopra la finestra dei comandi. Alternativamente, il comando `pwd` stampa la directory corrente, per modificare la quale si può usare il comando `cd nomedirectory`. Può risultare conveniente modificare la directory di default, per utilizzare una propria directory di lavoro. A questo proposito, si può scrivere un file speciale, `startup.m`, che va posto nella directory di default e viene eseguito da MATLAB ad ogni apertura del programma: ponendo nel file `startup.m` il comando `cd nomedirectory`, si può cambiare la directory di lavoro automaticamente ad ogni apertura del programma. MATLAB cerca gli M-file con il codice delle funzioni nella directory corrente e in un percorso di altre directory prestabilite. Per visualizzare il percorso, si usi il comando `path`. Per aggiungere directory personali al percorso, si può usare il comando `addpath`, che ovviamente può essere incluso nel file `startup.m`. In alternativa, si può modificare il path utilizzando la voce *Set Path* nel menu *File*.

Gli esercizi della prima lezione riguardano la definizione di semplici funzioni utili nell'analisi dei sistemi di elaborazione del segnale e delle comunicazioni elettriche. A titolo di esempio, si riporta di seguito il listato del file `rect.m` per il calcolo della funzione $\text{rect}(t)$.

```
function y=rect(t)
% Uso: y=rect(t)
% Restituisce un vettore i cui elementi contengono i valori
```

```

% della funzione rect nei punti del vettore t.

z=find(abs(t)>0.5); % trova gli indici corrispondenti a abs(t)>0.5
                    % dove rect(t) vale 0.
nz=find(abs(t)<0.5); nz5=find(abs(t)==0.5); y(z)=zeros(size(z));
y(nz)=ones(size(nz));
y(nz5)=0.5*ones(size(nz5)); % si impone il valore 0.5 in corrispondenza
                             % delle discontinuita'

```

Per utilizzare tale funzione, si scriva il listato con un qualsiasi editor di testo (ad esempio quello di MATLAB, se presente, accessibile con `edit`), e lo si salvi in un file `rect.m` in una directory nel path di MATLAB (ad esempio la directory corrente). Con il comando `help rect` si visualizzano i commenti posti fra l'intestazione e il corpo della funzione. Per visualizzare un grafico di $\text{rect}(t)$ possiamo usare i comandi

```

t=-1:0.01:1;
y=rect(t);
plot(t,y);
axis([-1,1,-0.25,1.25]);
xlabel('t');
ylabel('rect(t)');

```

Esercizi

1. Scrivere una funzione `sinc.m` per il calcolo della funzione $\text{sinc}(t)$, e si disegnino i grafici di $\text{sinc}(t)$, $\text{sinc}(t/2)$, $\text{sinc}(t - 0.5)$, in un opportuno range di valori di t ;
2. Scrivere una funzione `triangle.m` per il calcolo del segnale $\text{triangle}(t)$, e si disegnino i grafici di $\text{triangle}(t)$, $\text{triangle}(t/2)$, $\text{triangle}(t - 0.5)$, in un opportuno range di valori di t ;
3. Scrivere una funzione `trap.m` per il calcolo del segnale $\text{trap}(B_M, b_m, t)$, con andamento a trapezio isoscele centrato nell'origine, base maggiore B_M , base minore b_m e altezza unitaria;
4. Disegnare il grafico del segnale $x(t) = e^{-t} \text{rect}(t - 0.5)$.
5. Disegnare, usando il comando `stem`, il grafico della funzione $\text{sinc}(t - 2T)$, $t \in Z(T)$, $T = 0.5$;
6. Scrivere una funzione per il calcolo del segnale a tempo continuo e periodico

$$\text{sinc}_N(t) = \frac{\sin \pi t}{N \sin(\pi t/N)}.$$

Il periodo del segnale è N per N dispari e $2N$ per N pari.