

Il software MATLAB (acronimo di MATrix LABoratory) è uno strumento di simulazione per la generazione ed il trattamento dei segnali. MATLAB integra funzioni per il calcolo, la memorizzazione e la visualizzazione dei segnali con un linguaggio di programmazione orientato alla manipolazione di oggetti matematici.

MATLAB come calcolatrice

MATLAB mette a disposizione un'interprete di comandi che permette di effettuare un'ampia varietà di operazioni matematiche. Le operazioni disponibili sono +, -, *, /, ^ (elevamento a potenza), le funzioni trigonometriche `cos()`, `sin()`, `tan()`, . . . le funzioni `exp()`, `log()`, `log10()`. Queste operazioni possono essere applicate a espressioni numeriche (MATLAB come "calcolatrice avanzata").

```
>> 1+log10(10^2)
```

```
ans =  
     3
```

```
>> 2.5*(sin(pi/2)+3)
```

```
ans =  
    10
```

Attenzione: gli angoli sono sempre essere espressi in radianti.

Variabili in MATLAB

Le operazioni matematiche mostrate si possono applicare direttamente a numeri, oppure a variabili. Inoltre, il risultato di un'operazione si può assegnare ad una variabile.

```
>> a=5*3+12
```

```
a =  
    27
```

In MATLAB non è necessario dichiarare le variabili e la loro dimensione prima di poterle utilizzare. Le variabili sono identificate da nomi alfanumerici di lunghezza qualunque, e sono *case sensitive*.

Le variabili in MATLAB sono **sempre** matrici di numeri (da qui il nome MATrix LABoratory!).

I valori scalari e i vettori sono quindi casi particolari di matrici. Gli elementi delle matrici sono numeri reali o complessi, memorizzati con precisione double (8 byte). L'unità immaginaria si indica con **j** o **i** (se non vengono utilizzati come nomi di variabili).

Assegnamento di un valore scalare complesso:

```
>> b=3+2*j
```

```
b =  
    3.0000 + 2.0000i
```

Assegnamento di un vettore riga (gli elementi, racchiusi tra parentesi quadre possono essere separati da spazi e/o virgole):

```
>> c=[1, 2, 0]
```

```
c =  
    1  2  0
```

Assegnamento di una matrice (spazi e/o virgole separano elementi di una riga, e ";" separano le diverse righe):

```
>> A=[11 3 2; 5 1 11; 9 6 9; 2 15 19]
A =
    11     3     2
     5     1    11
     9     6     9
     2    15    19
```

Gli elementi racchiusi tra le parentesi quadre, e separati da virgole o “;”, possono essere numeri, variabili scalari o matrici stesse (i blocchi accostati devono essere coerenti). Ad esempio:

```
>> B=[c;A]
B =
     1     2     0
    11     3     2
     5     1    11
     9     6     9
     2    15    19
```

Riferimento a elementi o sottoinsiemi di una matrice

E' possibile accedere all'elemento di riga **r** e colonna **c** della matrice **A** utilizzando il comando **A(r,c)**.

E' anche possibile accedere a gruppi di elementi di un vettore:

- **B(2,3)** è l'elemento (2,3) della matrice **B**.
- **B(:,1)** è la prima colonna della matrice **B**.
- **B(2:4,3)** è un vettore colonna costituito dagli elementi 2, 3 e 4 della terza colonna di **B**.
- **B(2,3)=7** è il comando che assegna il valore 7 all'elemento (2,3) della matrice **B**.

Attenzione: in MATLAB gli indici partono sempre da 1.

In MATLAB `è possibile accedere agli elementi di una matrice utilizzando un solo indice e contando gli elementi progressivamente lungo le colonne a partire dalla prima.

```
A=[1 2 3; 4 5 6; 7 8 9; 10 11 12]
A =
     1     2     3
     4     5     6
     7     8     9
    10    11    12
```

```
>> A(3,2)
ans =
     8
```

```
>> A(7)
ans =
     8
```

Operazioni sulle matrici

Le operazioni matematiche elementari definite dai simboli **+**, **-**, *****, **/**, **^** si riferiscono a operazioni matriciali, che sono quindi eseguibili solo quando le dimensioni delle matrici sono congruenti. Eccezione a questa regola sono le operazioni tra uno scalare e una matrice.

Per poter applicare queste operazioni ripetutamente e singolarmente a ogni elemento della matrice basta anteporre “.” al simbolo dell'operazione.

```
>> C=[3 4; 5 7]
```

```
C =  
     3     4  
     5     7
```

```
>> C^2
```

```
ans =  
    29    40  
    50    69
```

```
>> C.^2
```

```
ans =  
     9    16  
    25    49
```

Le funzioni matematiche (ad esempio `sin()`, `log()`, ...) sono applicate ad ogni singolo elemento della matrice.

ATTENZIONE all'operatore `'`:

- `A'` restituisce l'hermitiana (trasposta e complessa coniugata) di `A`.
- `A.'` restituisce la trasposta di `A`.

Informazioni sulle variabili e sui comandi MATLAB

In MATLAB è possibile ottenere informazioni sulle variabili in memoria.

Il comando `size(A)` restituisce le dimensioni della variabile `A`.

```
>> size(A)
```

```
ans =  
     4     3
```

Il comando `whos` permette di ottenere informazioni su tutte le variabili attualmente in memoria:

```
>> whos
```

Name	Size	Bytes	Class
A	4x3	96	double array
B	5x3	120	double array
C	2x2	32	double array
a	1x1	8	double array
b	1x1	16	double array (complex)
c	1x3	24	double array

Grand total is 36 elements using 296 bytes

Le variabili possono essere rimosse dalla memoria utilizzando il comando `clear`: il comando `clear A` cancella dalla memoria la variabile `A` mentre `clear` cancella tutte le variabili.

Per ottenere informazioni su un qualsiasi comando od operazione MATLAB è possibile usare il comando `help`:

```
>> help sqrt
```

```
SQRT    Square root.
```

```
    SQRT(X) is the square root of the elements of X. Complex  
    results are produced if X is not positive.
```

See also SQRTM. See also SQRTM.

Generazione e visualizzazione di segnali in MATLAB

Un segnale è essenzialmente una funzione di una (o più) variabili indipendenti. In particolare, i **segnali continui** sono funzioni di una variabile che varia con continuità in un intervallo mentre i **segnali discreti** sono funzioni di una variabile che assume valori discreti a intervalli prefissati.

In MATLAB possono essere rappresentate solo sequenze, (segnali discreti), come vettori.

I segnali continui che assumono infiniti valori **non** possono essere invece rappresentati in senso stretto ma possono essere approssimati discretizzando la variabile indipendente, con passo sufficientemente piccolo (nel seguito del corso quest'affermazione diventerà più chiara).

Per poter definire una sequenza con passo arbitrario si può usare la seguente notazione (già utilizzata per estrarre sottoinsiemi da una matrice):

inizio:passo:fine

Ad esempio, volendo rappresentare un segnale nell'intervallo temporale di 2s, da -1s a 1s, con passo di 4 ms, la variabile indipendente (tempo) si rappresenta in MATLAB tramite il vettore:

```
>> t=-1:0.004:1;
```

Attenzione: “;” fa sì che l'istruzione venga eseguita senza la visualizzazione del risultato. Utilizzando un passo negativo si possono ottenere sequenze di valori decrescenti:

```
>> v=3:-0.5:1
```

```
v =  
    3.0000  2.5000  2.0000  1.5000  1.0000
```

Se, infine, il passo non viene specificato, è assunto uguale ad 1:

```
>> v=1:7
```

```
v =  
    1  2  3  4  5  6  7
```

E' quindi possibile costruire il segnale $s(t)$ una sinusoide di ampiezza $A=4$, frequenza $f_0=5$ Hz e fase $\varphi=0$ radianti [$s(t) = A \sin(2\pi f_0 t + \varphi)$]:

```
>> s=4*sin(2*pi*5*t);
```

Per visualizzare il segnale si usa la funzione **plot(t,s)** (si veda la figura 1).

plot crea il grafico dei punti individuati da **t** (asse delle ascisse) ed **s** (asse delle ordinate).

Pur essendo **s** una sequenza, l'onda viene rappresentata in forma continua, interpolando tra i due valori adiacenti per gli istanti di tempo mancanti.

plot(t,s,'o') visualizza i soli campioni della sequenza in funzione dei valori di **t** (figura 2). Si analizzi l'help della funzione **plot** per capire il significato dei diversi parametri.

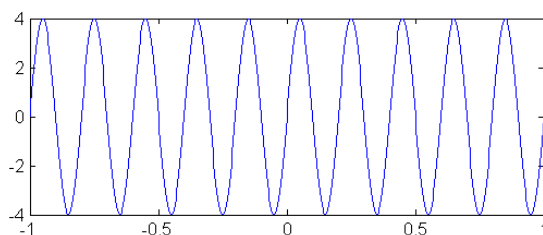


Figura 1

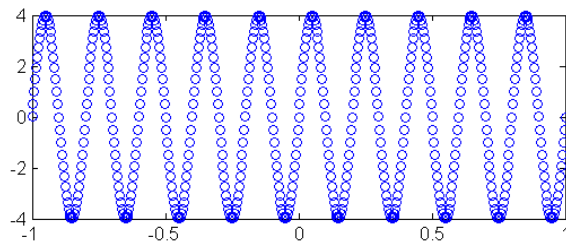


Figura 2

Esercizio: esponenziali complessi

MATLAB visualizza i numeri complessi come parte reale + parte immaginaria. Se non sono state assegnate, le variabili **i** e **j** rappresentano l'unità immaginaria. Alternativamente l'unità immaginaria può essere scritta come **sqrt(-1)**.

```
>> t=0:0.001:1;           % asse temporale [s]
>> f=4;                   % frequenza [Hz]
>> s=exp(j*2*pi*f*t);     % esponenziale complesso
```

La parte reale e la parte immaginaria hanno un andamento sinusoidale (Formola di Eulero):

$$A \exp[j(2\pi f_0 t + \varphi)] = A \cos(2\pi f_0 t + \varphi) + j \sin(2\pi f_0 t + \varphi)$$

I segnali complessi possono essere visualizzati plottando separatamente la parte reale e la parte immaginaria (Figura 3):

```
>> plot(t,real(s),'b',t,imag(s),'r')           % Figura 3a
>> comet(real(s),imag(s))
```

Il comando **comet(x,y)** realizza un plot animato del vettore **y** rispetto al vettore **x**. Nel caso in esame i punti si dispongono lungo un cerchio con raggio pari al modulo dell'esponenziale complesso (1 in questo caso).

```
>> plot3(t,real(s),imag(s))                   % Figura 3b
>> xlabel('Tempo (s)'); ylabel('Parte reale'); zlabel('Parte Imm');
```

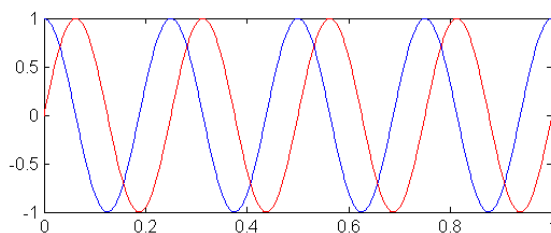


Figura 3a

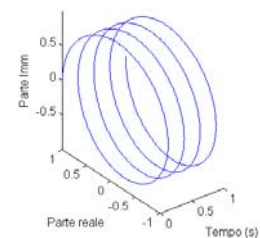


Figura 3b

Esercizio

In MATLAB è anche possibile ascoltare i segnali generati utilizzando il comando **sound(s,Fs)** e **soundsc(s,Fs)** (questa secondo comando normalizza il segnale in modo che la sua ampiezza massima sia in modulo pari a 1), con passo di campionamento **1/Fs**.

Utilizzando il comando **sound** ed il segnale **s=cos(2*pi*f*t)**, sperimentare le variazioni del suono con la frequenza (per accordare uno strumento: f=110Hz corrisponde alla nota LA).

Funzioni definite dall'utente

Ad una sequenza di comandi MATLAB può essere associato un nome: in questo modo l'insieme delle funzioni disponibili può essere esteso con funzioni definite dall'utente. La sequenza di istruzioni deve essere scritta in un file di testo (obbligatoriamente con l'estensione **.m**): verrà eseguita ogni volta che

si digiterà al prompt di MATLAB il nome del comando (che **corrisponde** al nome del file senza l'estensione). Questi files possono essere scritti con un qualsiasi editor di testo; MATLAB fornisce già un editor adatto allo scopo.
Esistono due tipi di files .m:

scripts

- sono sequenze di comandi senza nessuna intestazione;
- producono esattamente lo stesso effetto dell'esecuzione al prompt dei comandi contenuti, riga per riga: tutte le variabili generate rimangono nel workspace al termine dell'esecuzione;
- si lanciano digitando nome_file al prompt dei comandi;

functions

- hanno intestazione `function [out1,out2...]=nome_file(in1,in2...);`
- tutte le variabili generate all'interno della funzione (tranne ovviamente quelle restituite) sono locali: vengono cancellate al termine dell'esecuzione;
- si lanciano dal prompt dei comandi tramite:
`[var1,var2...]=nome_file(par1,par2...);`

Attenzione: le funzioni definite dall'utente per poter essere eseguite devono trovarsi nella directory corrente (comando cd) o nel path di MATLAB (comando path).

Esercizio (file rect.m)

Si scriva una funzione (**rect.m**) che restituisca in uscita un rettangolo di estremi **A** e **B**, rispetto al vettore delle ascisse **t**. La funzione ha 3 parametri in ingresso (un vettore, due valori scalari) e 1 in uscita (un vettore).

```
function y=rect(t,A,B)
% y=rect(t,A,B)
% Crea nell'intervallo temporale t un rettangolo
% di ampiezza 1 che ha inizio nell'istante A e termina
% nell'istante B.

% Inizializzazione dell'uscita: un vettore di zeri con le stesse
% dimensioni del vettore t che definisce l'asse dei tempi
y=zeros(size(t));
% Ricerca dei valori di t compresi tra a e B
set=find((t>=A) & (t<=B));
% Assegnamento dell'ordinata del rettangolo
y(set)=1;
return
```

Attenzione: i commenti in MATLAB sono preceduti dal simbolo %.

Si utilizzi la nuova funzione **rect()** per generare e visualizzare un rettangolo di ampiezza 3 e di estremi -2 s 1 s. L'asse temporale, discretizzato con un passo di 10 ms, ha estremi -3s, 3s.

```
>> t=-3:0.01:3;
>> R=3*rect(t,-2,1);
>> plot(t,R);
>> axis([-3 3 -1 4]);
```

MATLAB: linguaggio di programmazione

L'ambiente MATLAB possiede un completo linguaggio di programmazione, compresi i comandi che consentono di controllare il flusso dell'elaborazione come salti condizionali e cicli.

Ciclo for:

```
for variabile=[insieme di valori]
istruzioni...
```

end

Ad ogni ripetizione del ciclo la variabile assume un valore estratto dal vettore [insieme di valori]: non è necessario che questo vettore sia costituito da elementi interi e non è necessario che gli elementi siano equispaziati.

Ciclo while:

```
while (espressione logica)
istruzioni...
end
```

Salto condizionale if:

```
if (espressione logica)
istruzioni...
else
istruzioni...
end
```

Gli operatori logici in MATLAB sono:

- Uguale ==
- Diverso ~=
- Disuguaglianze <, >, <= e >=
- OR logico | e AND logico &

MATLAB: visualizzazione

Ogni istruzione grafica viene eseguita nella finestra attiva. Per creare più finestre, si utilizza il comando **figure**, che crea una nuova finestra e la rende attiva.

Per fissare gli estremi di ascissa e ordinata mostrati nella finestra grafica, si utilizza il comando:

```
axis([xmin xmax ymin ymax])
```

Infine è possibile aggiungere del testo agli assi e alla figura stessa mediante i comandi:

```
xlabel('testo') , ylabel('testo') , title('testo')
```

Più grafici possono coesistere in un'unica finestra grafica di MATLAB che può essere suddivisa in più aree grafiche. Il che permette di inserire più grafici nella stessa finestra è

```
subplot(nx,ny,ng)
```

dove **nx** ed **ny** rappresentano, rispettivamente, il numero di divisioni verticali e orizzontali della finestra, e **ng** rappresenta l'indice del grafico attivo tra i (nx*ny) grafici possibili. I grafici sono ordinati per righe e, ad esempio, per dividere la finestra in 4 quadranti e selezionare il grafico in basso a sinistra si scriverà:

```
subplot(2,2,3)
```

Talvolta può essere molto utile sovrapporre un grafico ad un altro già esistente senza che quest'ultimo venga cancellato. Questo può essere fatto utilizzando il comando **hold on** (**hold off** disattiva questa funzionalità) prima di plottare il grafico da sovrapporre.

Convoluzione di segnali

Il risultato $c(t)$ della convoluzione tra due segnali $a(t)$ e $b(t)$ (o di due sequenze a_n e b_n) è dato da:

$$c(t) = \int_{-\infty}^{+\infty} a(\tau)b(t-\tau)d\tau \qquad c_n = \sum_{m=-\infty}^{+\infty} a_m b_{m-n}$$

In MATLAB esiste il comando **conv** che realizza la convoluzione tra due segnali.

```
>> c=conv(a,b)*dt;
```

Il fattore **dt** serve se si vuole confrontare il risultato numerico ottenuto su segnali discreti con la convoluzione di segnali continui.

Per poter rappresentare il segnale **c** è necessario associare al segnale stesso un opportuno asse temporale **tc**. Ovviamente gli assi temporali relativi ai segnali **a**, **b** e **c**, rispettivamente **ta**, **tb** e **tc**, sono tutti discretizzati con lo stesso passo temporale **dt**. L'ordinata memorizzata nel primo campione del vettore **c**, **c(1)**, corrisponde all'istante **ta(1)+tb(1)** pertanto l'asse dei tempi **tc** risulta definito come segue:

```
>> tc=(ta(1)+tb(1))+[0:length(c)-1]*dt;
```

Negli esempi seguenti, poichè entrambi i segnali **a** e **b** sono definiti a partire dal tempo $t = 0$ s, la definizione di **tc** si semplifica nel seguente modo:

```
>> tc=[0:length(c)-1]*dt;
```

Esercizio

Utilizzando la funzione MATLAB **rect.m** si generi un rettangolo da 0 a 100ms, passo di campionamento $dt = 1$ ms. Si generi un'altra sequenza di 1000 campioni (stesso passo di campionamento) con degli impulsi (di ampiezza casuale) nei campioni 200, 750 e 800. Si disegnano i singoli segnali e la loro convoluzione.

Traccia del programma

```
dt=0.001;
N=1000;
t=[0:(N-1)]*dt;
a=rect(t,0,0.1);
b=zeros(1,N);
b(200)=1;
b(750)=-0.3;
b(800)=0.8;
c=conv(a,b)*dt;
ta=[0:length(a)-1]*dt;
tb=[0:length(b)-1]*dt;
tc=[0:length(c)-1]*dt;
plot(.....)
```

DTMF (Dual Tone Multi Frequency, file tono.m)

La seguente funzione restituisce il segnale generato da un telefono a toni. Ogni tasto genera la somma di due sinusoidi.

```
function S=tono(TASTO,T);
%
% Calcola il segnale S di durata T (sec) generato da un telefono
% a toni quando viene premuto TASTO (variabile stringa).
% Ogni tasto è la somma di due sinusoidi. Le frequenze delle
% sinusoidi ed i valori che può assumere la variabile
% tasto sono riportati di seguito.
% Il segnale e' campionato a 8192 Hz.
%
%      freq[Hz]   1209 1336 1477
%      697        1    2    3
%      770        4    5    6
```

```

%      852      7      8      9
%      941      *      0      #
%
% Esempio: s=tono('3',.5);

FC=8192;           % frequenza di campionamento
t=0:1/FC:T;       % asse dei tempi

fr = [697 770 852 941]; % frequenze delle righe
fc = [1209 1336 1477]; % frequenze delle colonne

if      TASTO=='1', r=1; c=1;
elseif TASTO=='2', r=1; c=2;
elseif TASTO=='3', r=1; c=3;
elseif TASTO=='4', r=2; c=1;
elseif TASTO=='5', r=2; c=2;
elseif TASTO=='6', r=2; c=3;
elseif TASTO=='7', r=3; c=1;
elseif TASTO=='8', r=3; c=2;
elseif TASTO=='9', r=3; c=3;
elseif TASTO=='*', r=4; c=1;
elseif TASTO=='0', r=4; c=2;
elseif TASTO=='#', r=4; c=3;
else disp('tasto non valido'); return;
end

S=[sin(fr(r)*2*pi*t)+sin(fc(c)*2*pi*t)]/2;
return

```

Per esempio lanciare i comandi

```
>> s=tono('3',.5); soundsc(s,8192);
```