Lezione n.1 - Soluzioni

Laboratorio di Telecomunicazioni

L.Verdoliva

1 Generazione di segnali a tempo discreto

Di seguito trovate il codice che consente di generare un impulso triangolare compreso tra 0 e $2N-1$.

```
function [x,n]=triangolo(n1,n2,N);
% TRIANGOLO genera un impulso triangolare
% [x,n]=triangolo(n1,n2,N), N è la semidurata

n=[n1:.n2];
x=[1-abs(n-N)/N].*[(n>=0) & (n<=2*N)];
```

Se si inseriscono commenti subito dopo la definizione della funzione, essi compariranno nell'help in linea del Matlab Provate per esempio a digitare da linea di comando `help triangolo`, vedrete che verranno visualizzate le prime righe di commento consecutive presenti nel file `triangolo.m`. Per questo motivo è molto importante commentare bene ogni funzione che scrivete: convenzionalmente, la prima riga di help contiene il nome in maiuscolo della funzione ed una descrizione sintetica. Questo perché il comando `lookfor` effettua la ricerca di una funzionalità desiderata leggendo solo la prima riga di help (cioè la prima riga di commento) di ogni funzione nel path. È allora consigliabile seguire sempre questa convenzione quando si scrivono le proprie funzioni. La seconda riga, invece, generalmente presenta un esempio di chiamata della funzione.

Nella definizione della funzione triangolo è anche possibile sfruttare la funzione per il calcolo dell'impulso rettangolare e scrivere il codice seguente per generare i valori del segnale:

```
x=[1-abs(n-N)/N].*rett(n1,n2,2*N);
```

Una volta scritta la funzione, basterà digitare i comandi:

```
>> [x,n]=triangolo(-20,20,5);
>> stem(n,x,'filled');
```

per generare e diagrammare un impulso triangolare di semidurata 5 nella finestra temporale $(-20, 20)$. E' importante sottolineare che è anche possibile scrivere uno script dal nome `triangolo1.m` come di seguito:

```
clc; clear all; close all;
n1 = input('Istante iniziale della finestra temporale: ');
n2 = input('Istante finale della finestra temporale: ');
N  = input('Semidurata impulso triangolare: ');

n=[n1:n2];                                  % definizione finestra temporale
x=[1-abs(n-N)/N].*[(n>=0) & (n<=2*N)];      % generazione del segnale
stem(n,x,'filled');                         % visualizzazione
```

Provate a lanciare il programma dal prompt dei comandi, semplicemente digitando `triangolo1`. La differenza fondamentale tra uno script ed una funzione è che il primo condivide lo spazio di memoria con la command window di Matlab, quindi non prevede l'esistenza di parametri di ingresso e di uscita: tutte le variabili "visibili" nello script lo sono anche nella finestra di comando e viceversa. Usando una funzione invece lo scambio dei parametri è rigidamente vincolato dall'interfaccia.

Di seguito trovate il codice per la generazione del gradino unitario $x(n) = \mathrm{u}(n - n_0)$, dell'impulso unitario $x(n) = \delta(n - n_0)$ e dell' esponenziale monolatero $x(n) = a^{n-n_0}\,\mathrm{u}(n - n_0)$.

```
function [x,n]=gradino(n1,n2,n0);
% GRADINO genera un gradino unitario u(n-n0)
% [x,n]=gradino(n1,n2,n0)

n=[n1:n2];
x=(n>=n0);
```

```
function [x,n]=delta(n1,n2,n0);
% DELTA genera un impulso unitario delta(n-n0)
% [x,n]=delta(n1,n2,n0)

n=[n1:n2];
x=(n==n0);
```

```
function [x,n]=esponenziale(n1,n2,a,n0);
% ESPONENZIALE genera un impulso esponenziale monolatero
% [x,n]=esponenziale(n1,n2,a)

n=[n1:n2];
x=(a.^(n-n0)).*(n>=n0);
```

Chiaramente è possibile modificare il prototipo di queste funzioni se si vuole generale un segnale di ampiezza generica pari a A. Per poter verificare le proprietà di una sinusoide discreta, può essere conveniente scrivere una funzione per la generazione del segnale sinusoidale:

```
function [x,n]=sinusoide(n1,n2,A,v0,theta);
% SINUSOIDE genera un segnale sinusoidale
% [x,n]=sinusoide(n1,n2,A,v0,theta)

n=[n1:n2];
x=A*cos(2*pi*v0*n + theta);
```

A questo punto per generare e diagrammare la sinusoide tempo discreto: $x(n) = 3\cos(0.1\pi n + \pi/3)$, basta digitare i seguenti comandi:

```
>> [x,n]=sinusoide(0,10,3,1/20,pi/3);
>> stem(n,x,'filled');
```

Facciamo adesso le verifiche che vengono richieste dall'esercitazione sulle proprietà della sinusoide.

1. Per verificare che la sinusoide è periodica solo se la frequenza ν_0 è un numero razionale, si può scrivere il seguente script dal nome `verifica1.m`:

   ```
         % Script per la verifica della prima proprietà

   clc; clear all; close all;
   n1=-20; n2=20;
   v1=3/20; v2=3/(20*pi);
   [x1,nx1]=sinusoide(n1,n2,1,v1,0);
   [x2,nx2]=sinusoide(n1,n2,1,v2,0);

   subplot(2,1,1); stem(nx1,x1,'filled');
   title('sinusoide periodica');
   xlabel('n'); ylabel('x(n)');
   subplot(2,1,2); stem(nx2,x2,'filled');
   title('sinusoide non periodica');
   xlabel('n'); ylabel('x(n)');
   ```

   Per effettuare la verifica scrivete `verifica1` nel prompt dei comandi. Notate come dal grafico anche la seconda sinusoide sembra periodica; in realtà, se stampate a video i valori assunti dai due segnali, noterete come il primo segnale ha un periodo $N_1 = 20$ (pari proprio al denominatore di ν_1 ridotta ai minimi termini), mentre nel secondo segnale i valori assunti non si ripetono mai.

2. In modo analogo si può verificare che due sinusoidi che differiscono per un numero intero sono identiche. Ponendo nel codice di precedente `v1=0.3` e `v2=1.3` visualizzerete due sinusoidi con periodo 10.

3. Per verificare che incrementando la frequenza da 0 a $1/2$ aumenta la rapidità di variazione, mentre questa diminuisce se la frequenza passa da $1/2$ a 1, si può rappresentate graficamente la sinusoide per $\nu_0 = 0, \frac{1}{16}, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, \frac{7}{8}, \frac{15}{16}, 1$, scrivendo il seguente script:

```
      % Script per la verifica della terza proprietà

clear all; close all; clc;
n1=0; n2=16;
% verifichiamo che aumenta la frequenza per 0<v<1/2
v0=0;
for i=1:5
    [x,nx]=sinusoide(n1,n2,1,v0,0);
    figure(1);
    subplot(3,2,i); stem(nx,x,'filled');
    axis([n1 n2 -1 1]);
    title(sprintf('v0=%5.4f',v0));
    v0=2^(i-1)/16;
end

pause;
% verifichiamo che diminuisce la frequenza per 1/2<v<1
v0=1;
for i=1:5
    [x,nx]=sinusoide(n1,n2,1,v0,0);
    figure(2);
    subplot(3,2,6-i); stem(nx,x,'filled');
    axis([n1 n2 -1 1]);
    title(sprintf('v0=%5.4f',v0));
    v0=1-2^(i-1)/16;
end
```

Facciamo adesso alcune osservazioni sui comandi utilizzati.

- E' stato possibile generare più grafici nella stessa figura all'interno del ciclo `for` e inserire il titolo di ogni grafico con il valore della frequenza utilizzata attraverso il comando `sprintf`. Questo comando stampa delle variabili secondo il formato definito tra singoli apici e mette il risultato in una stringa. Per maggiori dettagli si usi l'help in linea.
- Il comando `pause` non fa altro che bloccare l'escuzione del codice, fin quando non viene premuto un pulsante da tastiera.

2 Suggerimenti

Alcuni suggerimenti per l'uso di Matlab.

- Le funzioni che scrivete in Matlab devono sempre trovarsi nella directory corrente affinché possiate utilizzarle. In realtà è possibile modificare il *path* del Matlab, ovvero aggiungere il percorso in cui si trovano le vostre funzioni al percorso di default in cui Matlab cerca la libreria standard. In questo modo non sarà necessario trovarsi nella directory corrente per richiamare le funzioni. Da *File* scegliete l'opzione *Set path* della tendina a menù, quindi inserite il percorso in cui si trovano le vostre funzioni e poi cliccate su *Add Folder*, infine salvate e uscite. (N.B. Non potete fare questa modifica in laboratorio, ma solo sul vostro PC).

- E' possibile realizzare il *debug* del codice che scrivete usando tasto debug dall'editor di testo.
- Matlab vi permette di effettuare l'indentazione automaticamente selezionando il testo desiderato, quindi premendo *smart indent* dal menù text.