M202161029_亚历克上

EE 5410 Digital Signal Processing

MATLAB Exercise

Telephone Touch-Tone Signal Encoding and Decoding

Question# 01 Create a file named “tones.m” with the following MATLAB code:

Answer:

Here we have create file named “tones.m” in MATLAB editor shown in below figure:

```
tones.m   tones3.m   tones2.m   tone
clear all;
fs = 4000;
Ts = 1/fs;
t = [0:Ts:2];
f1 = 400;
f2 = 600;
x=[cos(2*pi*f1*t), cos(2*pi*f2*t)];
soundsc(x,fs);
```

a) **What is the duration of the signal?**
Answer: signal duration is 2 seconds.

b) **What do you hear?**
Answer: ac sound with the frequency of f1 and f2 is 679 and 770.

Question# 02 Create a file named "tones2.m" with the following MATLAB code:

Answer: According to the question we have created a file named tones2.m as shown in below:

Editor - C:\Users\Alec\Documents\GitHub\Master

tones.m | tones3.m | tones2.m

```
clear all;
x=[];
fs = 4000;
Ts = 1/fs;
t = [0:Ts:0.0205];
f = 400;
for i=1:100
  x=[x,cos(2*pi*f*t)];
end
soundsc(x,fs)
```

Also we created another file named "tones3.m" as shown in below figure:

tones.m | tones3.m

```
clear all;
fs = 4000;
Ts = 1/fs;
t = [0:Ts:2.05];
f = 400;
x=cos(2*pi*f*t);
soundsc(x,fs)
```

a) **State one difference between the two signals.**
   Answer: the difference is observation period. tones2.m file has less observation time then tones3.m file.
b) **Explain the difference.**
   Answer: tones2.m has less duration and if the blank vector from tones2.m is removed then we can hear only beep sound.

Question# 03 Create a file named “tone.m” with the following MATLAB code: and Note that tone is a user-defined MATLAB function. Try the following commands: help tone, x=tone(100,0.1) and y=tone(1000,0.01). Describe the usage for each of the three commands.

Answer:

tones.m | tones3.m | tones2.m | tone.m | test_signal_receiver.

```
function x = tone(frequency, observation_length)
% x=tone(frequency, observation_length) is used to generate
% a sinusoidal signal x with frequency and observation
% length specified in the arguments.
fs = 4000;
Ts = 1/fs;
t = [0:Ts:observation_length];
x = cos(2*pi*frequency*t);
```

- The function of command “help tone” is: it is used to generate a sinusoidal signal x with frequency and observation length specified in the arguments.
- Like: x= tone(frequency, observation length)
- x=tone(100,0.1) after running this command we have generated the 397 columns as output.
- **y=tone(1000,0.01):** this command generates 41 columns.

Outputs:

Editor - C:\Users\Alec\Documents\GitHub\Masters-2021\SECOND YEAR\Advanced in DSP\tone.m

tones.m | tones3.m | tones2.m | tone.m | test_signal_receiver.m | test_sampling2.m | test_linear_equation.m | test_Gaussian.m | test_ex_dft.m

```
function x = tone(frequency, observation_length)
% x=tone(frequency, observation_length) is used to generate
% a sinusoidal signal x with frequency and observation
% length specified in the arguments.
fs = 4000;
Ts = 1/fs;
t = [0:Ts:observation_length];
x = cos(2*pi*frequency*t);
```

Command Window

```
>> x=tone(100,0.1)

x =

  Columns 1 through 14

    1.0000    0.9877    0.9511    0.8910    0.8090    0.7071    0.5878    0.4540    0.3090    0.1564    0.0000   -0.1564   -0.3090   -0.4540

  Columns 15 through 28

   -0.5878   -0.7071   -0.8090   -0.8910   -0.9511   -0.9877   -1.0000   -0.9877   -0.9511   -0.8910   -0.8090   -0.7071   -0.5878   -0.4540

  Columns 29 through 42

   -0.3090   -0.1564   -0.0000    0.1564    0.3090    0.4540    0.5878    0.7071    0.8090    0.8910    0.9511    0.9877    1.0000    0.9877

  Columns 43 through 56

    0.9511    0.8910    0.8090    0.7071    0.5878    0.4540    0.3090    0.1564   -0.0000   -0.1564   -0.3090   -0.4540   -0.5878   -0.7071

  Columns 57 through 70

   -0.8090   -0.8910   -0.9511   -0.9877   -1.0000   -0.9877   -0.9511   -0.8910   -0.8090   -0.7071   -0.5878   -0.4540   -0.3090   -0.1564

  Columns 71 through 84

    0.0000    0.1564    0.3090    0.4540    0.5878    0.7071    0.8090    0.8910    0.9511    0.9877    1.0000    0.9877    0.9511    0.8910

  Columns 85 through 98
```

Question# 04 Write a MATLAB function named dtmfdial.m, to implement a DTMF dialer based on the frequency table in Figure 1. A skeleton of dtmfdial.m is given as follows:

Answer: Function named "dtmfdial.m":

Editor - C:\Users\Alec\Documents\GitHub\Masters-2021\SECOND YEAR\Advanced in ...

ex10_8.m | ex7_6.m | ex7_2_3.m | ex7_2_2.m | dtmfdial.m

```
%DTMFDIAL Create a DTMF tone
%usage: xx=dtmfdial(keyName)
% keyName = character which is one of the valid key names
% xx = signal vector that corresponds to the DTMF
dtmf.keys = ['1','2','3';
 '4','5','6';
 '7','8','9';
 '*','0','#'];
ff_cols = [1209,1336,1477];
ff_rows = [697;770;852;941];
dtmf.colTones = ones(4,1)*ff_cols;
dtmf.rowTones = ff_rows*ones(1,4);
```

(i) The input to the function is one of the valid key names.
Answer:
The input function has been already generated as show in above.

(ii) The output should be a vector of samples at sampling frequency 8000 Hz containing the DTMF tone. Each DTMF signal is the sum of a pair of unity amplitude sinusoidal signals and the time duration is 0.2s.
Answer: we have implemented the changes in code according to question so here is our new code:

```
%DTMFDIAL Create a DTMF tone
%usage: xx=dtmfdial(keyName)
% keyName = character which is one of the valid key names
% xx = signal vector that corresponds to the DTMF
dtmf.keys = ['1','2','3';
 '4','5','6';
 '7','8','9';
 '*','0','#'];
ff_cols = [1209,1336,1477];
ff_rows = [697;770;852;941];
dtmf.colTones = ones(4,1)*ff_cols;
dtmf.rowTones = ff_rows*ones(1,4);

dur1 = 0.20;
dur2 = 0.05;

tt = 0:1/fs:dur1;
xx = 0;
for ii = 1:length(keyName)
    keyName = keyName(ii);

    [r,c] = find(dtmf.keys==keyName);

    if (numel(ff_rows) == 0 | numel(c) == 0)
        continue
    end;
    tone = cos(2*pi*dtmf.rowTones(r,c)*tt) + cos(2*pi*dtmf.colTones(r,c)*tt);

    xx = [xx, zeros(1), tone];
end
```

Output key:

When we press any on command window, the number of columns are generated as shown in below:

```
>> dtmfdial('1','1')

ans =

         0         0    2.0000   -0.3029   -1.5212    0.5293    0.4560    0.0460    0.3952   -1.1235   -0.4769    1.9187
```

(iii) The frequency information is given in two 4 3 × matrices, namely, dtmf.colTones and dtmf.rowTones. To translate a key into the correct locations of the two matrices, the find function can be used. An example of using find is:

[ii,jj] = find('3'==dtmf.keys)

Answer:

```matlab
function xx=dtmfdial(keyName)
%DTMFDIAL Create a DTMF tone
%usage: xx=dtmfdial(keyName)
% keyName = character which is one of the valid key names
% xx = signal vector that corresponds to the DTMF
dtmf.keys = ['1','2','3';
 '4','5','6';
 '7','8','9';
 '*','0','#'];
%The following commands respods to the
ff_cols = [1209,1336,1477];
ff_rows = [697;770;852;941];
% below commands enlarge the vector into matrix
dtmf.colTones = ones(4,1)*ff_cols;
dtmf.rowTones = ff_rows*ones(1,4);

 [ii,jj]=find (keyName==dtmf.keys);
 t=0:0.001:0.2;

xx = cos(2*pi*t*ff_cols(jj))+cos(2*pi*t*ff_rows(ii));
soundsc(xx)
```

We have translated the key into the location of two correct matrix as shown in below figure:

```
>> xx=dtmfdial('1')

xx =

  Columns 1 through 9

    2.0000   -0.0721   -1.6565    0.1428    0.7509   -0.0353    0.3892   -0.2484   -1.3588

  Columns 10 through 18

    0.5892    1.8266   -0.8012   -1.6553    0.7216    0.9441   -0.2990    0.0207   -0.3624

  Columns 19 through 27

   -0.8832    1.0274    1.3556   -1.4183   -1.3200    1.3316    0.8572   -0.7346   -0.1949

  Columns 28 through 36

   -0.2027   -0.3970    1.1578    0.7190   -1.7737   -0.7126    1.7978    0.4653   -1.1873

  Columns 37 through 45

   -0.1511    0.1360   -0.0618    0.9913    0.0915   -1.7927    0.0248    1.9801   -0.1599
```

(iv) Play the sound of the DTMF tone using soundsc.

After running the code, there is a beep sound " soundsc".

Question#05: One simple way to implement a band-pass FIR filter is to use the following impulse response:

h [n] =1/L Cos(wn), 0≤n<L

where ω is the center frequency of the band-pass filter and L is the filter length. Use MATLAB to generate a band-pass filter with π =ω 0.2 .

Answer:

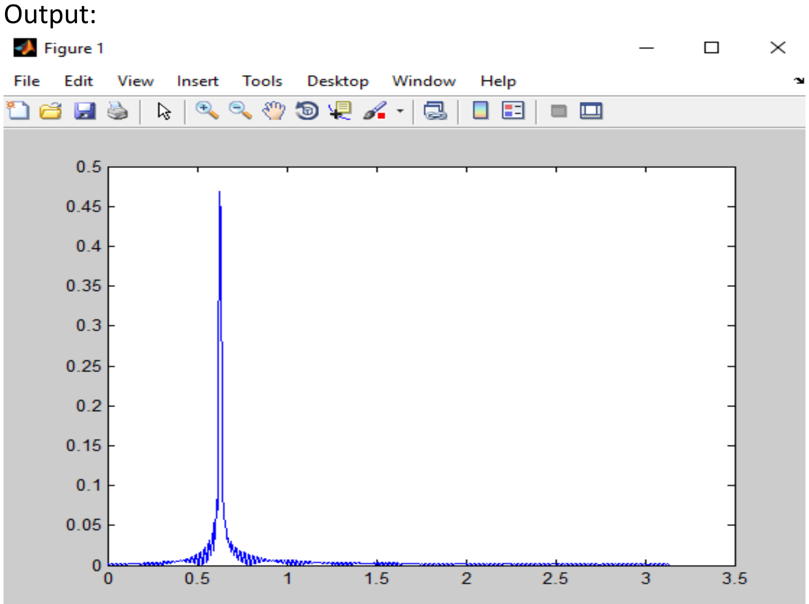
a) Try the cases of 50 =L and 500 =L . Plot the magnitudes of the frequency spectra of the two filters using freqz. An example of using freqz is:
 [a,b] = freqz(h); %h is the impulse response
 plot(b,abs(a));

Here L=500

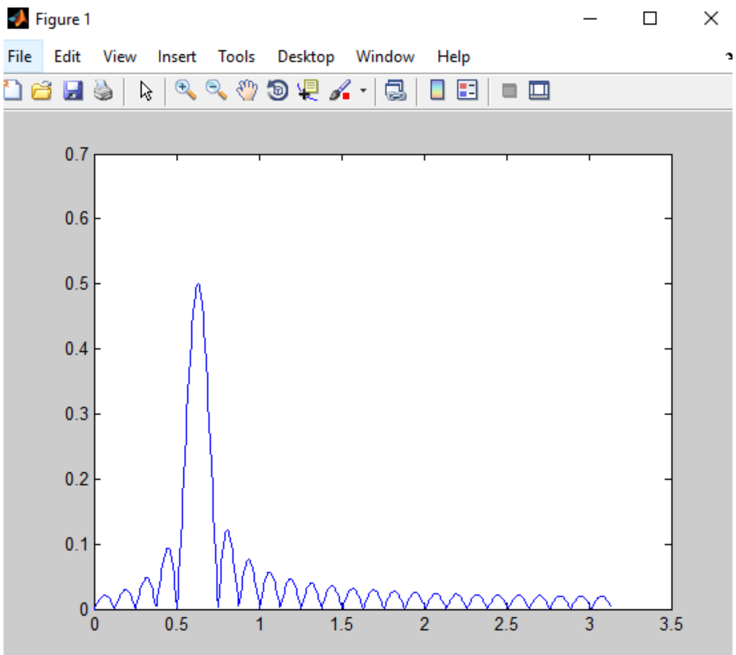
```
n=0:1:l-1;
l=500; %lenght
w=0.2*pi; %center frequency of bandpass filter
h=1./l*cos(w*n);

[a,b] = freqz(h); %h is the impulse response
plot(b,abs(a));
```

Output:



When L=50 then output is:



b) Compute the energies of h[n] for L=50 and L=500 The energy of is defined as:

$$E_h = \sum_{n=0}^{L-1} |h[n]|^2$$

Answer: Here is the code at L=50:

```
n=0:1:l-1;
l=50; %lenght
w=0.2*pi; %center frequency of bandpass filter
h=1./l*cos(w*n);

E=sum((abs(h)).^2)
```

Output at L=50:

```
>> question5

E =

    0.0100
```

Output at L=500:

```
E =

   1.0000e-04
```

c) Which filter will give a better DTMF decoding performance, h [n] with L=50 or L=500 ?
Answer: When the filter length is increased, precision and response is affected affecting performance.

Question# 06 Write a MATLAB function named dtmfdetect.m, to implement a DTMF encoder and decoder in a noisy environment. The requirements of the dtmfdetect function are given as follows:

(i) The input to the function consists of one of the valid key names, filter length of the band-pass filters and noise power. That is, dtmfdetect('1',50,1) will generate a DTMF tone '1' with L=50 and the tone is corrupted by a zero-mean white Gaussian noise with power of 1. The output will show the result of the detection, namely, displaying a message of The detected key is 1.

(ii) Each DTMF signal is the sum of a pair of unity amplitude sinusoidal signals and the time duration is 0.2s with sampling frequency fs=8000 .

(iii) To add a zero-mean white Gaussian noise to the noise-free DTMF tone, you can use the randn command. An example of using randn is:
noise = sqrt(0.1)*randn(1,10);

(iv) To detect the DTMF tone frequencies, you first need to pass the signal to a filter bank of 7 band-pass filters whose center frequencies are 697 Hz, 770 Hz, 852 Hz, 941 Hz, 1209 Hz, 1336 Hz and 1477 Hz. The DTMF tone can then be deduced from the two outputs with the largest energy. An example of producing the output signal given the input and FIR filter coefficients is

```
y=conv(x,h);  % x is the input and h is the filter  % impulse response
```

An example of computing the energy of a signal is

```
energy = sum(y.*y);
```

Answer:

Editor - C:\Users\Alec\Documents\GitHub\Masters-2021\SECOND YEAR\Advanced in DSP\dtmfdetect.m

+13 | example3_13_simplified.m | ex7_6.m | ex7_2_3.m | ex7_2_2.m | dtmfdetect.m

```
%DTMFDIAL   Create a DTMF tone
%usage: xx=dtmfdial(keyName)
% keyName = character which is one of the valid key names
% xx = signal vector that corresponds to the DTMF

dtmf.keys = ['1','2','3';
    '4','5','6';
    '7','8','9';
    '*','0','#'];
ff_cols = [1209,1336,1477];
ff_rows = [697;770;852;941];
dtmf.colTones = ones(4,1)*ff_cols;
dtmf.rowTones = ff_rows*ones(1,4);
fs = 8000;
t = [0:1/fs:0.2];
[ii,jj]=find(keyName==dtmf.keys);
noise = sqrt(noise_level)*randn(size(t));
xx = cos(2*pi*dtmf.colTones(ii,jj)*t)+cos(2*pi*dtmf.rowTones(ii,jj)*t)+noise;
n=[0:L-1];
h1 = 1/L*cos(2*pi*ff_cols(1)*n/fs);
h2 = 1/L*cos(2*pi*ff_cols(2)*n/fs);
h3 = 1/L*cos(2*pi*ff_cols(3)*n/fs);
h4 = 1/L*cos(2*pi*ff_rows(1)*n/fs);
h5 = 1/L*cos(2*pi*ff_rows(2)*n/fs);
h6 = 1/L*cos(2*pi*ff_rows(3)*n/fs);
h7 = 1/L*cos(2*pi*ff_rows(4)*n/fs);
o1 = conv(h1,xx);
o2 = conv(h2,xx);
o3 = conv(h3,xx);
o4 = conv(h4,xx);
```

a) Try your dtmfdetect function with various keys, different L (50 =L and 500=L) and noise powers (σ^2 = 0, σ^2 =1 and σ^2 =50). For each key, perform 10 trials and record the number of correct detection with the following table.

Key	$L = 50$ $\sigma^2 = 0$	$L = 500$ $\sigma^2 = 0$	$L = 50$ $\sigma^2 = 1$	$L = 500$ $\sigma^2 = 1$	$L = 50$ $\sigma^2 = 50$	$L = 500$ $\sigma^2 = 50$
1	1	1	1	1	4	1
2	2	2	2	2	2	3
3	3	3	3	3	3	3
4	4	4	4	4	4	4
5	5	5	5	5	2	5
6	6	6	6	6	4	6
7	7	7	7	7	7	7
8	8	8	8	8	8	8
9	9	9	9	9	9	9
0	0	0	0	0	0	0
*	*	*	*	*	*	*

#	#	#	#	#	#	#