Group……………………………………………………………

LAB 1
Signal in Time Domain

Listen to sinusoidal signal

After we observe the effect of each variable to the shape of sinusoidal signals in the previous part, now we will listen to the sound of generated signals. Edit you code as below:

```
clear;
clc;
A=2;
Fs=8000;
F=400;
dur=1;
q=0;
n=0:1/Fs:dur;
y=A*cos(2*pi*F*n+q);
plot(n,y)
sound(y,Fs) % you can use wavplay(y,Fs) if function sound is not work
```

Change the value of A, Fs, F, dur, and q follow this table:

A	Fs	F	dur	q
1	8000	400	0.5	0.5*pi
2	8000	400	0.5	0.5*pi
2	8000	600	0.5	0.5*pi
2	8000	600	1	0.5*pi
2	8000	600	1	0

☺ Listen to each sound carefully and explain the effects of each constant variable (A, F, dur, q) to the perceived sound of these sinusoidal signals.

Let's create our MATLAB piano

We can generate sounds with different frequencies in the previous part. Why don't try to generate our own piano that can play a variety of notes as the real piano?

Before building our piano, a quick introduction to the frequency layout of the piano keyboard is needed.

On a piano, the keyboard is divided into octaves the notes in each octave being twice the frequency of the notes in the next lower octave.

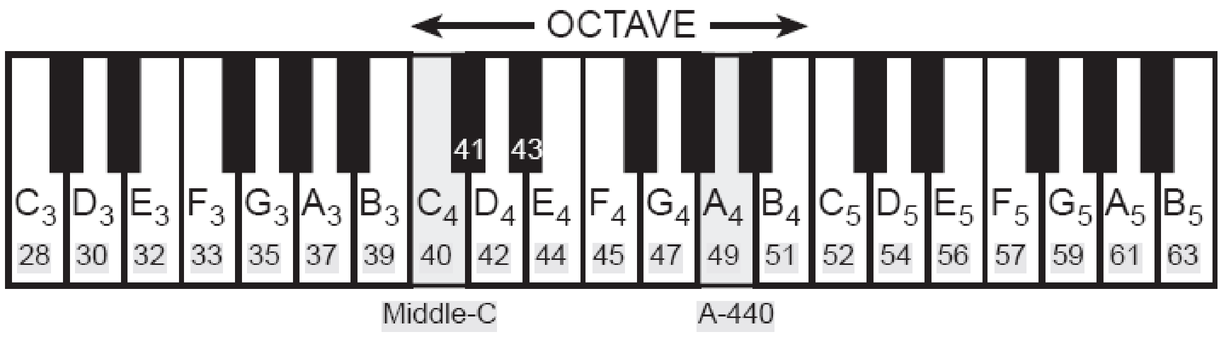

Figure 1

The reference note is A4 (A440) which is A above middle C and its frequency is 440 Hz. Each octave contains 12 notes (5 black keys and 7 white) and the ratio between the frequencies of the notes is constant between successive notes. Thus this ratio must be $2^{1/12}$. Hence we can calculate the frequency of each key *F(keynum)* by this equation:

$$F(keynum) = 440(2^{1/12})^{keynum-49} \tag{3}$$

Where n is the number of key.

For example, middle C is 9 keys below A-440, its frequency is approximately 261 Hz.

If you already understand the concept of piano keyboard frequency, let implement the function to calculate the frequency for each key and generate a pure sinusoidal signal of this frequency. -

Complete the following m-file function (note.m) to produce a desired note for the given duration.

note.m ☺

```
function tone = note(keynum,dur)
% NOTE Produce a sinusoidal waveform corresponding to a
% given piano key number
%
% usage: tone = note (keynum, dur)
% freq= the frequency calculated by the ratio from A440
% tone = the output sinusoidal waveform
% keynum = the piano keyboard number of the desired note
% dur = the duration (in seconds) of the output note
%
fs = 8000; %-- use 11025 Hz on PC/Mac, 8000 on UNIX
tt = 0:(1/fs):dur;
freq =……………………………………………..;%<=== FILL IN THIS LINE

tone =…………………………………………….;%<=== FILL IN THIS LINE
```

** You cannot run note.m because it's just a function not main program.

Now we can implement the main program for playing your song. The sequence of key numbers is assigned by variable keys. For each key, the main program will call function note.m to generate sinusoidal signal of the certain key number by sending the key number and duration as arguments.

Complete the m-file program calling note.m to play scale.

scale.m

```
%--- play_scale.m
%---
keys = [ 40 42 44 45 47 49 51 52 ];
%--- NOTES: C D E F G A B C
% key #40 is middle-C
% tone=output from calling note.m
dur = 0.25 * ones(1,length(keys));
fs = 8000; %-- use 11025 Hz on PC/Mac, 8000 on UNIX
xx = zeros(1,sum(dur)*fs+1);
n1 = 1;
for kk = 1:length(keys)
keynum = keys(kk);

tone = ……………………………………………………….;%<=== FILL IN THIS LINE
n2 = n1 + length(tone) - 1;
xx(n1:n2) = xx(n1:n2) + tone;
n1 = n2;
end
sound( xx, fs )
```

1. More interesting sound

In the previous part you can create the sound for each key by implementing the pure sinusoidal signal. Do you have any idea to create the more interesting sound such as the sound of piano when we press two keys at the same time?

Bell sound

The following codes are used for generating Bell sound. You can copy these two functions (bell.m, bellenv.m) to your MATLAB working directory.

bell.m

```
function xx = bell(ff, Io, tau, dur, fsamp)
%BELL produce a bell sound
%
% usage: xx = bell(ff, Io, tau, dur, fsamp)
%
% where: ff = frequency vector (containing fc and fm)
% Io = scale factor for modulation index
% tau = decay parameter for A(t) and I(t)
% dur = duration (in sec.) of the output signal
% fsamp = sampling rate
dt=1/fsamp;
tt = 0 : dt : dur;
xx= bellenv(tau, dur,fsamp).*(cos (2*pi*ff(1)*tt + Io.*bellenv(tau,
dur,fsamp).* cos(2*pi*ff(2)*tt)));
plot(xx);
sound(xx,fsamp)
```

bellenv.m

```
function yy = bellenv(tau, dur, fsamp);
%BELLENV produces envelope function for bell sounds
%
% usage: yy = bellenv(tau, dur, fsamp);
%
% where tau = time constant
% dur = duration of the envelope
% fsamp = sampling frequency
% returns:
% yy = decaying exponential envelope
%
% note: produces exponential decay for positive tau

dt=1/fsamp;
tt = 0 : dt : dur;
yy=exp(-tt/tau);
```

Call bell.m function by typing this command in the MATLAB workspace

```
>> xx = bell([220,440], 5, 2, 6, 11025);
```

☺
Can you hear the bell sound?
Can you see the different between shape of Bell signal and the pure sine signal?
How many frequencies contained in the Bell sound? What are they?

You may answer the first two questions easily. The different between shape of Bell signal and the pure sine signal can be seen from the graph of these two signals, but how to answer the last problem? Can you answer this question by considering the graph of bell signal?

To study the characteristic of signal we can do it in time domain as you do when you need to compare the shape of two signals. Moreover, we can analyze the characteristic of signal in frequency domain which represents the frequency components of a given signal. MATLAB provides tools for analyzing signal in frequency domain. One of these tools is spectrogram (*spectrogram* function).
Append this command in the end of bell.m function and call it again in MATLAB workspace. Explain what you see in this figure?

```
figure(2);
spectrogram(xx,128,120,128,fsamp);
```

You can use spectrogram function in scale.m, and the code in part 3 to represent the frequency content of the generated signals.