ELE 301 Fall 2011
Laboratory No. 1

1 Summary

There are two objectives to this lab: (a) to practice some basic MATLAB commands , and (b) to become familiar with the generation and display of simple signals in MATLAB. During the lab you will use MATLAB to generate and listen to some simple audio frequency tones.

2 Representing Signals in MATLAB

2.1 Sampling

To deal with continuous time signals we will have to work with samples of the signals. Let $x(t), t \in [0, T]$, be a continuous time signal defined over the time interval $[0, T]$. Fix an integer $N > 1$ and set $\Delta = T/N$. Then provided N is sufficiently large the signal x can be adequately represented by the set of N uniformly spaced samples $s(k) = x(t_k)$ where $t_k = k\Delta$, $k = 0, 1, \ldots, N-1$.

The sampling frequency in the above representation is $F_s = 1/\Delta = N/T$. Roughly, Shannon's sampling theorem says that if the sample frequency is more than twice the highest frequency present in the signal, then the samples adequately represent the original signal. (We will do more on this later.)

A sampled signal can be stored as the elements of a $1 \times N$ matrix. For example, the signal $x(t) = \sin(2t), t \in [0, 10]$, can be represented in MATLAB as a $1 \times N$ matrix as follows:

```
Ts=0.1;
N=100;
w0=2;
t=(0:1:N-1)*Ts;
x=sin(t*w0);
```

Here, `Ts` is the intersample time Δ (So 1/Ts is the sample frequency); `N` is the total number of samples; `w0` is the frequency of the sine wave in radians; `t` is a $1 \times N$ matrix indexed from 1 to N containing the sample times 0, Ts, 2Ts, ... (N-1)Ts; `x` is a $1 \times N$ matrix containing the samples of the signal $\sin(2t)$ at the sample times.

2.1.1 Plotting a signal

You can plot the signal with the commands:

```
plot(t,x);
grid
xlabel('time - secs')
ylabel('signal x')
title('Plot of x vs t')
```

This yields the plot shown in Figure 1.

You can also use the plot command to plot several signals on the same graph and you can control the colors of the lines as well as their type. For example,

Figure 1: Example of a simple MATLAB plot

```
plot(t,x,'-',t,y,'--');
grid
xlabel('time - secs')
ylabel('signal x')
title('Plot of x and y vs t')
```

will plot the signals `x` and `y` on the same graph. The first signal will be plotted with a solid line and the second signal will be plotted in a dashed line. Use the help facility to learn all the color and line type controls.

Sometimes you want to plot more than one signal on the same page but on different graphs. In this case you can use the `subplot` command to set up an array of plots. For example,

```
subplot(2,1,1), plot(t,x,'g');
grid
xlabel('time - secs')
ylabel('signal x')
title('Plots of x and y')

subplot(2,1,2), plot(t,y,'r');
grid
xlabel('time - secs')
ylabel('signal x')
```

will plot the signals `x` and `y` on a 2 by 1 grid of separate plots. The first signal will be plotted in a green line and the second in a red line.

You can also plot each group of signals on a completely new page using the `figure` command. For example,

```
figure (1)
plot(t,x);
grid
xlabel('time - secs')
ylabel('signal x')
figure(2)
plot(t,y);
grid
xlabel('time - secs')
ylabel('signal y')
```

will plot the signal `x` on the first figure window and the signal `y` on the second figure window.

2.1.2 Listening to a signal

The command `sound(x,Fs)` will output the signal x with sample frequency `Fs` to the computer sound system. This works fine for listening to short signals but is not of high quality.

A better but slower method is to first save the signal as a .wav file (a microsoft format for sound files), and then listen to the signal with a separate audio player program. This will yield much better quality reproduction. We will do this in Lab two.

2.2 Writing an M-file

The best way to use MATLAB (and this is the way you should use it!) is to first write a MATLAB program, called an M-file, and then type the name of the file at the MATLAB prompt. This allows you to make changes, fix bugs, etc., in an efficient manner. Once you have a correct program you can use it just like a simple MATLAB command. You can even call your M-files from within other M-files.

To write a M-file you use any text editor to create a file containing MATLAB commands and then store the file as an ASCII file with a .m extension. If you pull down the file menu in MATLAB you can start the editor directly from the menu. Of course your M-files have to be stored in a location that is in the MATLAB path so that MATLAB knows where to look for them. By default, MATLAB always looks first in the working directory. So if you store your M-files in your working directory, MATLAB will always find them. If you want to call an M-file from another directory, then you have to add that directory to the MATLAB Path. You can do this with the commands:

```
>>P=path;
>>path(P,'E:\userid\path_to_file');
```

If MATLAB encounters a % symbol on any input line it ignores the rest of that line. So the % symbol can be used to add comments to your programs.

A simple program might look like this:

```
clear % clear memory
clf reset % clear all figures


T=10; % signal duration
Fs=8000; % sampling frequency
t=0:1/Fs:T; % time axis

x=exp(-2*t).*sin(2*pi*f*t);
plot(t(1:n),x(1:N),'r')
grid
xlabel('time-secs')
ylabel('signal value - volts')
title('A Plot of a Simple Signal')


sound(x,Fs) % play the signal
```

ELE 301
Laboratory No. 1

3 Lab Procedure

This part of the handout needs to be completed and handed in.

3.1 Task 1: Pure Tones

The musical notes of the western music scale are grouped into *octaves* with each octave containing 12 notes. The octave containing middle C covers the frequency range from 220Hz to 440Hz. Within each octave the notes are logarithmically spaced so that jumping one octave doubles the frequency. Thus the frequency of each note is $2^{1/12}$ times the frequency of the note below it. The frequencies of the notes in the middle C octave are shown in table 1.

A	220Hz
As	$2^{1/12}$*A $\approx$ 233Hz
B	$2^{1/12}$*As $\approx$ 247Hz
C	$2^{1/12}$*B $\approx$ 262Hz
Cs	$2^{1/12}$*C $\approx$ 277Hz
D	$2^{1/12}$*Cs $\approx$ 294Hz
Ds	$2^{1/12}$*D $\approx$ 311Hz
E	$2^{1/12}$*Ds $\approx$ 330Hz
F	$2^{1/12}$*E $\approx$ 349Hz
Fs	$2^{1/12}$*F $\approx$ 370Hz
G	$2^{1/12}$*Fs $\approx$ 392Hz
Gs	$2^{1/12}$*G $\approx$ 415Hz

Figure 2: Table 1.

A signal model for a pure tone of frequency f Hz is the sinusoidal signal $x(t) = A\sin(2\pi ft + \phi)$. Here A is the amplitude of the tone, f is the frequency in Hz, $2\pi f$ is the frequency in rad/sec, and ϕ is the phase.

1. Write an M-file called `tone.m` to generate a pure tone of 3 secs duration at middle C. Use a sampling frequency 8 KHz. Your program should play the signal through the sound system using the `sound` command and it should plot the first N signal samples vs time (use N=300). Start your program by clearing the workspace and then assign values to the following variables:

   `Fs` - sampling frequency in Hz
   `f` - tone frequency Hz
   `w` - tone frequency rad/sec
   `T` - duration of signal
   `t` - vector of sample times
   `Amp` - amplitude of tone
   `ph` - phase of tone
   `x` - the vector of signal values
   `N` - no. of samples to be plotted

   Make sure your plot displays a grid, has the axes correctly labelled, and is given a title.

2. What do you get if you try to plot to whole signal vs time. Explain.

3. How does changing the value of `Amp` effect the sound of the signal?

4. How does changing the value of `ph` effect the sound of the signal?

5. What happens if you decease the sampling frequency to 1 KHz? Is this still theoretically adequate? Does it drastically alter the quality of the results?

6. Demonstate your program to the TA and get the TA to grade it. Congratulations, you have just completed a working MATLAB program.

3.2 Task 2: Chords

Now on to bigger and better programs.

One of the most fundamental operations on signals is signal addition or superposition. This simply corresponds to combining signals by adding their values at each time. For example, a musical chord results when several (appropriate) notes are played concurrently. The perceived signal is the sum of the individual signals. In the case of K notes (K is usually 3), the composite signal can be written as

$$x(t) = \sum_{n=1}^{K} A_n \sin(2\pi f_n t + \phi_n)$$

Modify your program `tone.m` so that it will generate K single tone signals each with its own frequency, amplitude and phase. This is simple to do: just change the assignment statement for `Amp`, `ph`, and `f` to be vector assignments instead of constants, and generate the signals as the inner part of a loop (if you are clever you can do it without the loop). Your program should plot each signal (the first N samples) vs time on the same page but on different graphs and in different colors, and play each signal in succession. It should also plot on a separate page the graph the sum of the signals and play this as well.

Demonstrate your program with K=3, Amp=[4 4 4], ph=[0 0 0], and f=[262 330 392]. This is the C Major chord. The TA will give you a grade for this program.

3.3 Task 4: Harmonics

Lets continue with the idea of signal superposition to show that complex signals can be constructed in this fashion.

When a particular note is played on a musical instrument not only is the fundamental frequency (tone) generated but also higher harmonics of the fundamental, i.e., pure tones at the frequencies $2f, 3f, 4f, \ldots$. These harmonics give the instrument a richer sound.

1. Copy your M-file `tone.m` into a new file `harmon.m` and modify the new file to generate a signal of 1.5 seconds duration that is the sum of the tone middle C and its first 10 harmonics. Use a sampling frequency of 16 KHz. Allow for the possiblity that the harmonics have different amplitudes and phases. You can do this by using one matrix to represent the amplitudes and another to represent the phases of the component signals.

   Your program should:

   - plot the pure tone and the sum of the tone and its harmonics on the same graph vs time (again just the first N samples). Remember to label the axes and the give the plot a title.
   - play both signals using the sound command.
   - play a signal that consists of the pure tone for 1.5 secs concatenated with the tone plus harmonics for 1.5 secs, concatenated with just the harmonics for 1.5 secs. This will allow you to hear the effects of the harmonics very clearly. Can you tell the pitch with just the harmonics? Try dropping the first few harmonic as well, can you still tell the pitch? Can you suggest why this may be so?

   You can experiment with different amplitudes and phases to see the effect on the composite signal appearence and sound. Demostrate you program to the TA using

   ```
   Amp = [ 1 0 1/3 0 1/5 0 1/7 0 1/9 0 ]/pi

   Ph = (pi/2)*[ 1 1 1 1 1 1 1 1 1 1 ]
   ```

2. In this case what are the theoretical constraints on the sampling frequency?

3.4 Task 4: Phase Error

The human ear is remarkably tolerant of differences in the phase of pure tones. To investigate this effect, try this experiment. The program `randph.m` generates and plays two signals. First it generates a signal at the frequency of middle C with ten harmonics. Then it generates a second signal using the same harmonic amplitudes but the phases are random numbers in the range $[-\pi, \pi]$. You can do this with the command `rand`. Each time `rand` is called it generates a random number between 0 and 1. Both signals are plotted on the same graph so that you can see the effect of the random phase. The program then randomly selects one of the signals and plays it. It then waits 4 seconds, again randomly selects a signal and plays it. It then asks you to say if the two signals it played were the same or different.

Try it out - see how many correct answers you can give on ten successive attempts. Write the outcome of each of your attempts here:

What conclusion can you draw about the tolerence of the human ear to differences in the phase of harmonic components?