School of Electronic and Electrical Engineering

UNIVERSITY OF LEEDS

ELEC1401 Communications Networks and Signals

MATLAB Sheet II

This practice sheet has been designed to help you better understand the sinusoidal signals, the sampling procedure, and the Nyquist rate.

The tasks in this worksheet are for learning purposes, the marks associated with the tasks are for guidance only and they will not contribute to your final module mark. However, completing these tasks is essential to ensure that you understand the material as Matlab will be assessed in the January assessment (Assessment 2). You are expected to finish the tasks by the end of Unit 2 of this module. The sheet includes empty boxes below each task to write your answers, this should be useful when you review the Matlab sheet before the final assessment. You might find some of the tasks already explained in the screencasts, they are repeated here for completeness and for you to try it yourself if you have not done so.

You need headphones for some of the tasks. Please bring your own headphones if you come to the lab.

MATLAB Help: In MATLAB you can get help in various ways. The simplest way, if you know the exact name of a particular function, is to type "help name_of_function" in the MATLAB command line. Alternatively, you can browse MATLAB help to find more about what you are looking for. In any of the problems below, if anything is unclear, or you don't know how to use a function, you should first try MATLAB Help. MATLAB commands and built-in functions appear in red in this sheet.

****

Section 1: Sinusoidal functions and their harmonics

Recommended time for completing the tasks in this section: 30 minutes

Suppose t = 0:1/44100:1;

a. Try this

   >> f = 1000; y = sin(2*pi*f*t); p= audioplayer(y,44100); play(p)

   and listen to the generated signal. How does the tonality of the produced sound change when you change f between 50 and 20,000? Do they sound natural to you?

b. Now, try this

```
>> A=0; t=0:1/44100:1; for i =500:2500
A = A + 1/sqrt(i)*sin(2*pi*i*t);
end;
>> p= audioplayer(A, 44100); play(p)
```

Did it become any more pleasant to listen to? Plot A versus t.

c. Now let's look at a real note. Use audioread function to load the piano_A4.wav file given on Minerva. You have to first download the file to your Current MATLAB folder. Now, let's try this:

```
[y,Fs] =audioread('piano_A4.wav');
```

Check the Help manual for audioread. Find out what y and Fs are.

Using `p= audioplayer(y, Fs); play(p)`, listen to the file. How long is it?

Think of y as a function of time. Suppose time 0 corresponds to the first sample value y(1). To what time would the second sample value, y(2), correspond? How about the third sample?

Task 1: Can you write a variable t that represents the times at which the original musical note has been sampled. Write t in the box below. Plot y versus t on screen.

[1 mark]

These marks are for guidance only and they will **not** contribute to your final module mark

d. Now try

```
specgram(y,400,Fs)
```

Note: For some Matlab versions, this command will not work. You can then use:

```
spectrogram(y,hanning(400),300,400,Fs,'yaxis')
```

You may need to install 'Signal Processing Toolbox' for this to work.

The resulting graph will show you at each point of time which frequencies are contributing most to the generated sound (zoom in to see better what those frequencies are). Here, colour 'red' indicates a large amplitude.

Task 2: Which (fundamental) frequency and its harmonics seem to be the main contributors to note A4?

[1 mark]

Section 2: Sampling

Recommended time for completing the tasks in this section: 60 minutes

In this section, we first try to understand the sampling rate, and the minimum sampling rate we need for various signals. We try different signals and sample them at different rates, and by listening to the sampled version, we judge when we start losing quality.

a. Let's start with a single tone. Put your headphones on and try this

```
>> Fs = 2500; t=0:1/Fs:1; y=sin(999*2*pi*t); p= audioplayer(y,Fs); play(p)
```

What is the frequency of the signal you are listening to? What is its amplitude? What is the sampling rate? What is the duration of this signal? Write them below:

Frequency:
Amplitude:
Sampling rate:
Signal duration:

b. Start changing the sampling rate in the above line. Try Fs = 4000 and gradually come down. Try other points in between/below/above these values. See at what point you hear a different tonality from the original signal (let's say the one sampled at a high rate, e.g., 4000/s, in our case). Based on your observations, see if you can find a **cut-off sampling rate** below which you hear a different tonality from the original signal, but above which the tonality remains unchanged. Note that the amplitude/volume may change as well. But, we are mostly concerned with the tone/pitch that you are hearing. Complete the table below. Try sampling rates above that rate and see if they are any different.

Fs	Is tonality different?
4000	
3000	
2500	
1500	
2200	
1800	
Try other values to narrow down Fs	

Can you find a cut-off rate? Write it below.

Cut-off rate for the 999 Hz signal:

c. Now, change the frequency of the tone in part (a) to 799 Hz and 1499 Hz, and, in each case, repeat part (b). Note that the change in tonality may happen sooner or later than what you observed in part (b), so you may need to consider different values of Fs than we have in the table. Also note that MATLAB may not accept Fs to be less than 1000. Write the corresponding cut-off rate that you find for each frequency below:

Cut-off rate for the 799-Hz signal:

Cut-off rate for the 1499-Hz signal:

<u>Task 3:</u> Based on the above observations, find a relationship between the minimum acceptable sampling rate (the cut-off rate) and the frequency of the original signal (denote it by f), and write it below. Do you remember what this rate is called?

[1 mark]

minimum acceptable sampling rate (the cut-off rate) =

d. Now let’s apply what you found in part (c) to an audio signal. Use audioread to read out the file voice48k.wav provided on Minerva. Make sure you save the file in your Current Folder. Now, type this:

```
>> [y, Fs] = audioread(‘voice48k.wav’);
```

What is Fs? Listen to this file by the following commands:

```
>> p = audioplayer(y(1:1:length(y)),Fs); play(p)
```

In the above line, y represents a sampled version of the original audio signal with a sampling rate Fs. Suppose, instead of sampling at the rate Fs, you want to sample the file at a rate Fs/2. What would be the sampling period in that case? What would you play instead of y itself? Convince yourself that the following line would do the job for you:

```
>> p= audioplayer(y(1:2:length(y)),Fs/2); play(p)
```

This is called downsampling, in the above, with a factor 2. We want to downsample the original signal with different factors, and listen to it, and find out at what point the quality starts to drop and at what point the quality becomes too poor.

Task 4: Can you modify the above line of code to play the downsampled version of y by a factor n. Write your working code in the below box:

[1 mark]

e. Use your new line of code to listen to the downsampled version of the original signal. Change the parameter n and watch for the quality of signal. It would help if you make a table like the one below. At what sampling rate, the quality starts to drop? At what sampling rate you cannot recognize the speaker or what he says? It would help if you alternate between the original signal and the downsampled version to better feel the difference. Summarize your observations below.

n	Fs	Quality
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

Task 5: Based on the above observations, what is the minimum sampling rate you suggest for your car radio?

[1 mark]

minimum acceptable sampling rate for an audio signal (in a car radio) =

f. Optional: Repeat the same process with the two music pieces provided: fluteSA.wav and Beethoven192k.wav. What do you observe now?