**Signals and Systems Homework #4 Matlab**

**Delivery list**

***Due Friday 11:59 pm by emailing, June 5, 2020.***

1. All Matlab codes
2. A file containing plots and your answers (Word or PDF file format is preferred).
3. Output audio files with the correct file name (see below).

Put all files in a zipped single file with the name of your student ID.

* 1. **(20%)Windowing effect** means the real-world time-domain waveform is observed with a limited time period. Thus its Fourier representation should take this observation “WINDOW” into account. This assignment is related to the example in the Chapter 4 classnote.
     1. To evaluate this effect, we need to develop DTFT function. Instead of using Matlab’s fft, we can implement DTFT according to the definition. The code is as follows. Do you know **why** it can be implemented like this? (可用矩陣方式解釋)

%%%% This is an example to do DTFT function X = DTFT(x, n, W)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% X = DTFT values computed at W (1xNw)

% x = Time sampled signal (1xNn)

% n = sample time vector (1xNn)

% W = frequency location vector (1xNw)

% Nw: The length of W

% Nn: The length of n

X\_tmp = exp(-1j\*(W.' \* n)) \* x.'; X = X\_tmp.';

End

* + 1. Plot the DTFT (magnitude only) of *y*(*t*) for the following 3 cases: *M* = **8, 20, 100.**

*M* is the window size

𝑦(𝑛) = 𝑥[𝑛]𝑤[𝑛]. (1)

𝑥[𝑛]

7𝜋

= cos (

16

9𝜋

𝑛) + cos (

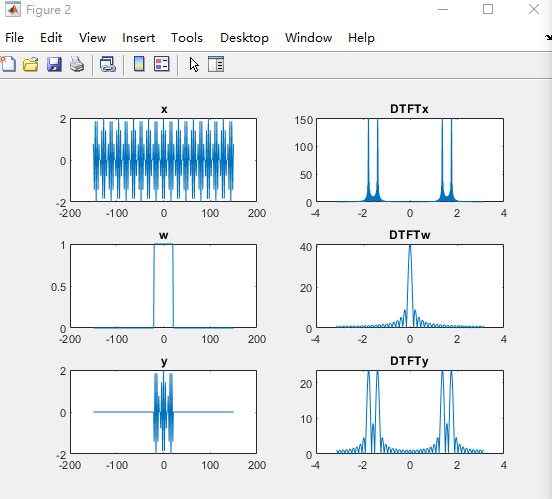
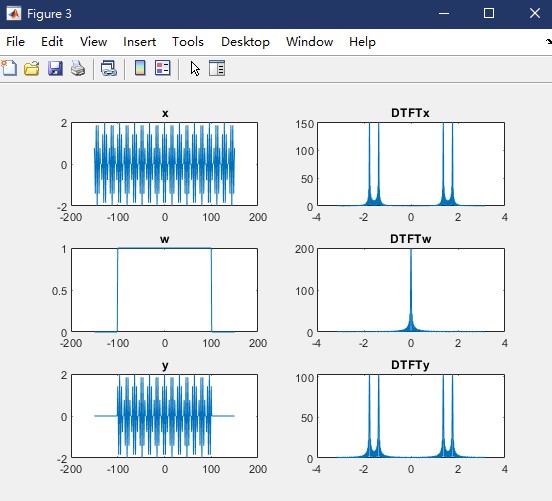
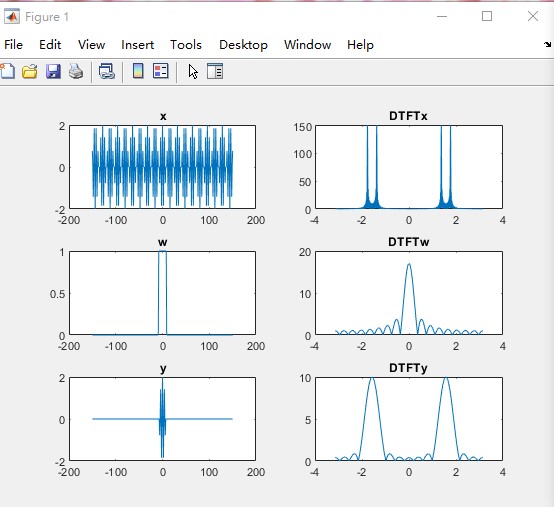
16

𝑛) (2)

𝑤[𝑛] = {1, |𝑛| ≤ 𝑀

0, |𝑛| > 𝑀

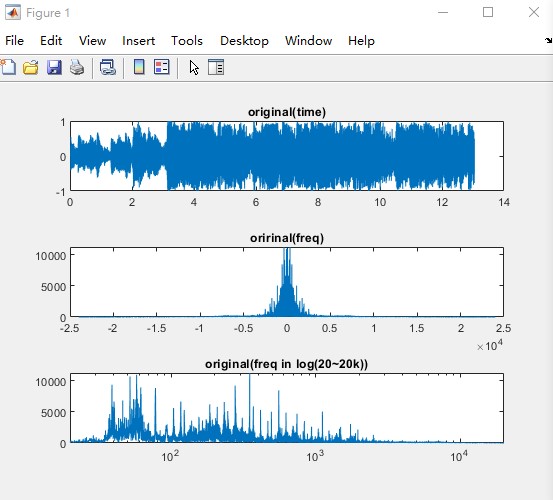
(3)

**Plot the x, y, and w** with three different M values. Paste your results with correct time and frequency axis here.

* 1. (80%) Matlab assignments to understand how to process an audio signal (e.g. music) using digital signal processing. For the following questions, please refer to the corresponding Matlab’s exemplary codes. What you need to do is to COMPLETE the codes and GENERATE all results (either figures or audio outputs).

(CAUTION: MAKE SURE the volume is moderately low to avoid hurting your hearing. 請注意測試輸出檔案時, 小心音量控制, 避免傷到聽力, 此音訊最大音量振幅為1, 人的靈敏度可以聽到0.01-0.001的聲音，請小心測試).

* + 1. Read the music (2-channel stereo music) using Matlab' audioread. For simplicity, we will process **ONE** channel only (i.e., the first channel. The first row in the raw data. This is MONO tone).
       1. PLOT the time waveform and its frequency-domain representation using FFT. Please also indicate the **correct** time and frequency index based on the sampling rate and data length. (Paste your plots here)
       2. Write this MONO music to a new file and save it as .OGG format using Matlab’s audiowrite. (Output a file named as org+your student ID.ogg)

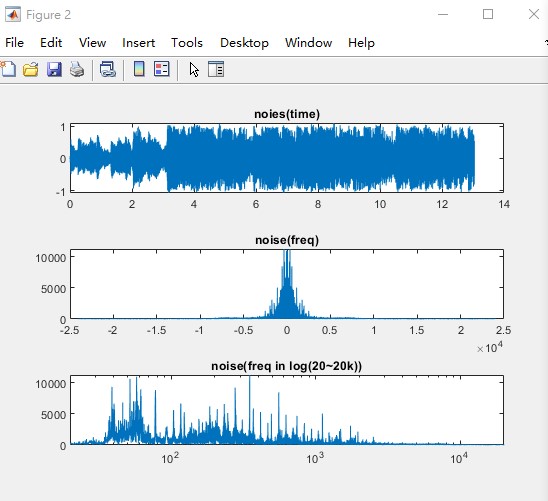


* + 1. Noise the music (To test your hearing sensitivity).
       1. Add a Gaussian white noise (using ‘randn’**, See HW#1**) to the original signal such that the signal-to-noise ratio is 30 dB. The maximum signal amplitude is assumed to be 1. PLOT and compare the frequency-domain representation with and without adding noise. (Paste your plots here)
       2. Same as (i). What do you hear? (可以中英文白話敘述)

A little bit like what you can hear when tuning to an empty FM frequency on radio.

* + - 1. Output a file named as noise30dB+your student ID.ogg
      2. How high the SNR is so that you hear the same quality as the original music? (這部分只要回答SNR多少妳/你就感覺不到noise 的影響, 請實測看看並注意音量控制小心影響聽力)

I did this in a relatively noisy place and 40dB SNR is small enough for me to neglect



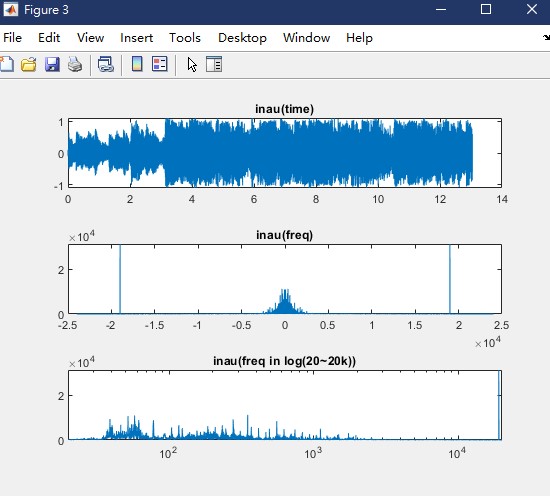
* + 1. Watermark the music (To test your hearing frequency band or generate inaudible signals). Add a single tone (i.e., a single frequency, cosine signal) to the original data so that you **CANNOT** hear this single tone. That is, you hear exactly the same as the original music.
       1. How to add this single tone and what is the frequency of it? PLOT and compare the frequency-domain representation with and without adding noise. (Paste your plots here). Note that aliasing might be present when the frequency is higher than half of the sampling rate.

SNR = 30;

noise\_var = 10^(-SNR/10);

ys\_noise = ys + sqrt(noise\_var)\*randn(N,1);

* + - 1. Output a file named as inaudible+your student ID.ogg

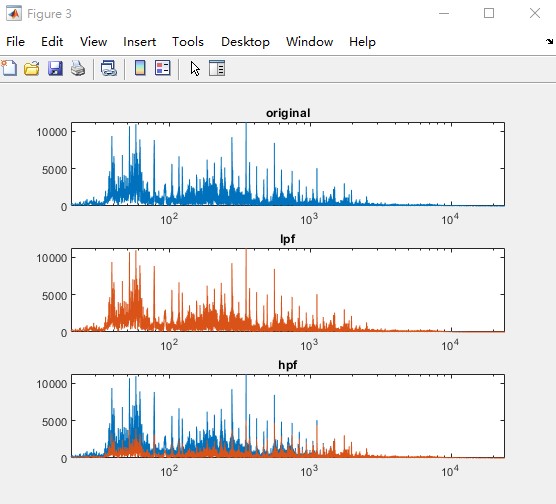


\* inaudible watermark frequency at 19kHz

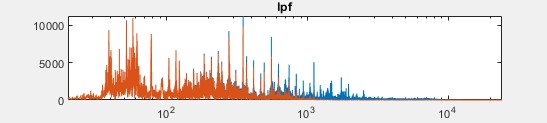
\*\*Don’t know why the original code is at 40Hz which aliasing must happen(@ 8kHz)

* + 1. Filter the music. Apply the low-pass filter and high-pass filter to listen what is the difference
       1. Use Matlab’s fir1 to generate a low-pass filter with a 64-order FIR type and a variable cut-off frequency (You need to **specify** what the cut-off frequency is). Plot and compare the frequency presentation before and after filtering (Paste your result here). Output the audio file called lpf+your student ID.ogg
       2. Use the same function fir1 to generate a high-pass filter with a 64-order FIR type and a variable cut-off frequency. You need to **specify** what the cut-off frequency is. Plot (Paste your result here) and output the audio file called hpf+your student ID.ogg.

(v) State what is the difference of the filtered music between low-pass and high- pass filtering? (可以中英文白話敘述)



\*note that since human's sensitivity of sound @ different freq is different, lpf with cutpff freq @ 10k is sensible for human ears but can't be plot clearly via matlab, so i changed it to 1K just for plotting \*all these plot with semilogx (x = 20~20k)



~~And this stuff sounds blur af~~

(v) I’ve been using or playing with equalizer for music player for a better music listening experience since I was 10, and cuz I’m pretty used to it, I would just say the high freq is small after lpf and low freq is small after hpf. Tbh, I’m pretty sure this irresponsible answer doesn’t reach the requirement, so I guess I’ll share some experience about it instead.

I personally like the feeling of bass, so applying a LPF sounds pretty reasonable, but It doesn’t always work like that.

Human’s hearing sensitivity differs according to different frequencies, it reaches its maximum at around 10^3Hz since most vocal are around this frequency.

Which means, in order to make us feel the bass drum(or “kick”), the low frequency of music are normally high already. Boosting low freq via an equalizer will easily cause audio waveform to be clipped, which leads to voice crack. Or you’ll feel the sound being compressed\* when a loud bass kicked, if you’re audio devices are not well enough.

\*this compress means sth like DRC - dynamic range compression, not audio file compression

Kicks are usually at around 100Hz, so enhancing everything below it isn’t the thing since sounds below 50Hz are usually some annoying humming noise, so I usually get rid of them to make sure I don’t feel stuffy and dizzy.

Following is the current setting on my phone



There’re still other things I can share about music filters or sound effects etc, but I guess I’ll stop here.

* + 1. Downsample the music. The original music has the sampling rate of 48kHz.
       1. We are trying to make the sampling rate “HALF” (i.e., 24kHz). What should it be done without aliasing? (hint: apply an anti-aliasing filtering before downsampling. A factor of 2 means sampling every other sample.)

Complete the code and output the file called ds2+your student ID.ogg.

Downsampling by 2 will cause frequency above 0.5 of the highest frequency (0.25 of the original sample rate) to overlap or aliasing

Applying a LPF can make sure to reduce aliasing, but it leads to losing of high frequency, which in this case is apparent for me.

* + 1. Compress the music. Use the above filtering or downsampling or **other skills** you can “compress” the music as much as possible in terms of file size without losing the audio quality too much. Complete the code and output the file called compress+your student ID.ogg. Specify your compression ratio by comparing the output file size with the original .ogg file size (i.e., the file generated in (a)(ii))

First I simply redo the downsampling and anti-aliasing process again.

I output two compressed file

“compress\_0710807.ogg”

“compress\_0710807\_defaultQuality.ogg”

First is compressed by setting the quality parameter to 0, and it output a file of 53.5kB

Second one is just in case adjusting quality parameter is illegal, I first output a tmp ogg file with quality set to 5, and re-read that file into matlab and output with default ogg setting.

The reason I do that is because ogg is out put has a VBR or variable Bit Rate, which means the encoder will judge whether a part of the audio has the signal that worth giving a larger bitrate to preserve it, I first output with a lower quality parameter to get rid of the unnecessary thing in the audio with the aid of the OGG encoding algorithm itself, then re-load that file and output again with default settings, which should have a lower average bitrate since most thing that takes extra bitrate have already been removed.

org\_0710807.ogg------ 187kB

“compress\_0710807.ogg” ------53.5kB(28.6%)

“compress\_0710807\_defaultQuality.ogg” ------81.7kB(43.7%)