ENGN4537/6537 Digital Signal Processing

School of Engineering, Australian National University

Second Semester, 2023

MATLAB Project

[illegible], 20 October 2023

DIGITAL SIGNAL PROCESSING OF AUDIO SIGNALS

Important Notes:

- For assessment purposes, students need to submit a **zip folder** that contains the **final report (pdf)**, **MATLAB code (.m) files, generated audio (.wav) files, and other supporting materials**.
  - All **requested plots** and **MATLAB (.m) functions** should be also included **in the final report** as instructed in the questions.
- For each question, provide a short statement explaining your approach to solving the problem and a short statement explaining the outcome (either desired or undesired outcomes).
- Please use the following filename for the zip folder:
  - uXXXXXXX_Givenname_Surname_ENGNX537_Project
- Please use the following filename for the final report:
  - uXXXXXXX_Givenname_Surname_ENGNX537_Project_Report

  For example:
    - u1234567_Sheldon_Cooper_ENGN6537_Project_Report (for ENGN4537 students)
    - u7654321_Leonard_Hofstadter_ENGN4537_Project_Report (for ENGN6537 students)

- Assessment: ENGN4537 students are evaluated based on a total of 100 raw marks and ENGN6537 students are evaluated based on a total of 130 raw marks.
- This project accounts for **30%** of the overall course grade (for both ENGN4537 and ENGN6537 students).
- Students must score a **minimum of 50% in this project to pass this course.**

  (i.e., ENGN4537 students must score a minimum of 50 marks out of 100 marks and ENGN6537 students must score a minimum of 65 marks out of 130 marks)
- **Any late submissions after the due date will receive a score of 0.** For exceptional cases which require an extension, students must email the course convenor at the earliest with valid reasons.
- Apart from this Project Manual, to complete this project, students should also download the provided audio files and MATLAB files from WATTLE:
  - DSP_Speech.wav
  - DSP_Music.wav
  - DSP_Noise.wav
  - DSP_HRIR.zip (or HRIR_1.mat and HRIR_2.mat)
  - Handout for Windowed FIR LPF Design

Q1 – INTRODUCTION TO MATLAB, PART 1 (Read and Play Audio Files)

[5 Marks for ENGN4537 students]

[10 Marks for ENGN6537 students]

In this section, you will pra[illegible] write a .wav audio file and obtain related information from the audio file. Download a [illegible] clip given by **"DSP_Speech.wav"**. This is a .wav (short form for "Windows Audio Vide[illegible]LAB can read. The MATLAB functions of interest for this type of audio file are ***audioread(), audioplayer(), soundsc(), audiowrite(), and audioinfo()***.

Note that please use MATLAB versions newer than R2014b. Herein below are some instructions on the MATLAB functions of interest:

- The ***audioread*** function will read any audio file and return to you a vector that is (1) sampled data and (2) its sample frequency. If the audio clips are MONO (a single audio channel), the ***audioread*** will return a vector that is N x 1 in dimensions (where N is the number of samples). However, most of the audio clips available to you are STEREO (left and right channels), so in this case, the ***audioread*** will return a vector that is N x 2 in dimensions. Type:

  >> help **audioread**

  in the command window to learn how to use it.

- The ***audioplayer*** is an object in MATLAB for playing audio. Realize that the correct sample frequency must be known to play the music correctly. Note that you need to use ***play*** to play the audio samples in the ***audioplayer*** object if you want to play back the music. Type:

  >> help **audioplayer**

  in the command window to learn how to use it.

- The ***audiowrite*** function will allow you to create a ".wav" file that can be played in any music player (like Windows Media Player). Type:

  >> help **audiowrite**

  in the command window to learn how to use it.

- The ***audioinfo*** returns information about the contents of the audio file. Type:

  >> help **audioinfo**

  in the command window to learn how to use it.

Use the built-in function *audioinfo()* to obtain information about the provided audio file **"DSP_Speech.wav"**.

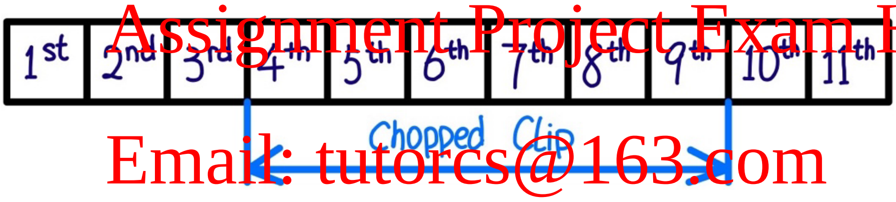
Q 1.1 Answer the following questions using the information obtained from *audioinfo()*. **[2 Marks]**

- **Q 1.1.1** What is the number of bits per sample (i.e., bit-depth) of the provided audio clip?
- **Q 1.1.2** What is the sampling rate of the provided audio clip?
- **Q 1.1.3** What is the [illegible] of the provided audio clip?
- **Q 1.1.4** What is the [illegible]ded audio clip?
- **Q 1.1.5** What is the [illegible]ples in the provided audio clip?

Use the function *audioread*[illegible]ltiple input parameters such that you can read the provided audio clip **"DSP_Speech.w**[illegible]n the 4th second to the 9th second (i.e., resulting in a new 6-second audio clip). Note tha[illegible]y the *audioread()* in Question 1.2.

Q 1.2 What is the command you use in this question? (i.e., write down the one-line code for this question) **[1 Mark]**

Use *audioplayer()* and *play()* to play the chopped audio clip.



Q 1.3 What are the commands you use in this question? (i.e., write down the two-line codes for this question) **[1 Mark]**

Q 1.4 What is the command for stopping the audio playback? **[1 Mark]**

- The sample rate is an important quantity for audio, which has a great influence on its quality. The sample rate of the speech clip can be changed using ***resample*** (Resample uniform or nonuniform data to a new fixed rate) function. Type:

  >> help **resample**

  in the command window to learn how to use it.

For ENGN6537 students, use the built-in function *resample()* to answer the following questions.

Q 1.5 **(ENGN6537 Students ONLY)** Resample the vector of the **"DSP_Speech.wav"** audio clip at 4 times the original rate and use *audioplayer()* and *play()* to play it. State the difference between it and the original audio file. **[2.5 Marks]**

Q 1.6 **(ENGN6537 Students ONLY)** Resample the music vector at 1/4 times the original rate and use *audioplayer()* and *play()* to play it. State the difference between it and the original audio file. **[2.5 Marks]**

Q2 – INTRODUCTION TO MATLAB, PART 2 (Collect Voice Data)

[10 Marks for ENGN4537 students]

[10 Marks for ENGN6537 students]

In Question 2, you will coll[illegible]own voice in a .wav file that will be used in the following questions of this project.

Use the built-in function *au*[illegible]ord your voice when you speak the requested phrase below. Then, store the audio record[illegible]The audio recording should have the following specifications:

- It has a duration of **12 seconds**.
- It has a sampling frequency of **48,000Hz**.
- It has **1 audio channel**.
- It has a bit-depth of **16 bits per sample**.

Besides, you need to follow the following guidelines and requirements when you collect your voice data.

- Before you start recording, you MUST play a provided background audio file (i.e., **'DSP_Noise.wav'**).
- To play the background noise, you need to use to separate device other than the laptop/desktop that you will be using to record your voice. It is because some built-in sound cards may not support you to record and play music simultaneously.
- For example, you can use your mobile phone to play the background noise and use your laptop to record your speech.
- Make sure that the provided background noise can be heard from the beginning to the end of the recording.
- Make sure that the provided background noise has a proper volume. It is neither too low to sense the noise, nor too high to sense the human speech.
- In the audio record, you need to say the following phrase:

    I really like DSP, and my Uni-ID is uXXXXXXX.
    For example, "I really like DSP, and my Uni-ID is u7654321."

- You may make several attempts before you achieve all the requirements above.
- Lastly, you need to store the audio recording in the following requested format:

    DSP_Givenname.wav
    For example, DSP_Sheldon.wav

Important Notes:

- After using *audiorecorder()* to collect your voice data, you should use *audioplayer()* or *soundsc()* to double-check the quality of your recording.
- Make sure you set the correct sampling frequency for *audiorecorder()* function, as its default sampling frequency is not the requested value.
- After you save the r[illegible]mber to use another audio player app (e.g., Windows Media Player) to check the [illegible]rding.

Q 2.1 Briefly describe how [illegible] voice in MATLAB? **[10 Marks]**

- How many step(s) d[illegible]
- Which built-in function(s) did you use?
- How did you choose the appropriate input parameter(s) for each function?

Remember to attach your "DSP_Givenname.wav" file in the project submission.

Q3 – DISCRETE FOURIER TRANSFORM (DFT)

[20 Marks for ENGN4537 students]

[20 Marks for ENGN6537 students]

This section helps you und[illegible] of Discrete Fourier Transform (DFT) and its application to analyse the frequency conte[illegible]inuous signal should be sampled before taking the DFT. The number of samples has a gr[illegible]FT result because the sampled signal should be long enough to contain the complete inf[illegible]al signal. Consider the continuous signal:

$$x(t) = 1.2\cos(2\pi f_1 t) + 2\sin(2\pi f_2 t),$$

where $f_1 = 150$ Hz and f_2 is determined by **the first three digits of your U number** (for example, if your U number is 6200000, then f_2 is 620 Hz). Sample $x(t)$ at $f_s = 4800$ Hz, where f_s is the sampling frequency.

Q 3.1 Generate 7200 samples of $x(t)$ and plot the signal. The x-axis should denote the time in seconds calculated based on f_s and the number of samples. Attach the plot in your final report. **[1 Mark]**

Q 3.2 Read the following instructions carefully. Then, write a MATLAB function called $dft1$ that will use the following definition to create a plot of the magnitude of the DFT of an input signal (use *for* loop to calculate DFT point by point). This plot should display frequency content (magnitude of the DFT) from 0 Hz to $f_s/2$ Hz. The function has two inputs (the signal and its sample frequency), one output vector (the magnitude of the DFT that lies between 0 Hz and $f_s/2$ Hz), and it will also produce the DFT magnitude plot. **[5 Marks]**

The DFT transforms a sequence of N complex numbers $\{x_n\}$ into another sequence of complex numbers $\{X_k\}$, which is defined by

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i\frac{2\pi}{N}kn}$$

Reason for including only half of the bandwidth: Recall that approximately the 2nd half of the DFT is a mirror image of the first half, and that the usable frequency information lies between frequencies from 0 Hz to $f_s/2$ Hz. It means that you would have approximately N/2 samples in our plot.

Design steps for Question 3.2 are:

- Determine the number of samples of the signal (N).
- Compute the DFT of the signal.
- Compute the magnitude of the DFT of the signal.
- Determine the frequency resolution of the plot, $\Delta f = f_s / N$.
- Create a frequency vector from 0 Hz to $f_s/2$ Hz, using an increment of Δf Hz.
- Determine the length of this frequency vector (call it N1).
- Create a new vector that consists of the first N1 values of the DFT magnitude.
- Plot the frequency vector against the first N1 values of the DFT magnitude.
- Provide proper labels for the axes of the plot.

Q 3.3 Write a MATLAB function called $dft2$ that will use MATLAB in-built function *fft()* to create a plot of the magnitude of the DFT of an input signal. The design requirements and steps are same as Question 3.2. **[2 Marks]**

The MATLAB in-built function for computing the DFT of a given sequence is called *fft()*. Type:

\>> help **fft**

in the command window to

Note that the output of in-bu l be multiplied by a scaling factor 2/N to restore the amplitude of the original signal in tim the transform length of *fft* function.

Q 3.4 Here we will use MA and *toc* to compute the running time of $dft1$ and $dft2$. Take the 7200 samples of $x(t)$ as the input of function $dft1$ and $dft2$.

Type:

\>> help **tic**

\>> help **toc**

in the command window to learn how to use them.

Fill in the information below (you should take average of at least 5 runs): **[2 Marks]**

- The running time of $dft1$: ____________s
- The running time of $dft2$: ____________s

Then, answer the following question. **[2 Marks]**

- Which dft function has the lower running time and why?

Q 3.5 Use your $dft2$ function to analyse the frequency content of 7200 samples of $x(t)$. Attach the DFT magnitude plot in your final report. **[2 Marks]**

Q 3.6 Using $dft2$ generate the magnitude plots for the following cases and attach them in your report: **[3 Marks]**

- Case-1: Generate **32** samples of $x(t)$ and perform its **32**-point DFT
- Case-2: Pad zeros to the 32-sample $x(t)$ to get a vector of length **512**. Then perform its **512**-point DFT
- Case-3: Generate **512** samples of $x(t)$ and perform its **512**-point DFT

Then, answer the following questions: **[3 Marks]**

- Do these three plots show the frequency content of $x(t)$ clearly compared to the plot in Question 3.5?
- How do the sample size and zero padding affect the plots?

Q4 – CONVOLUTION

[5 Marks for ENGN4537 students]

[10 Marks for ENGN6537 students]

DFT is a powerful tool that ... ns. One of them is to calculate the linear convolution of two sequences. For example, DI ... calculate the received signal at the human ears from a given source based on the linear c ... the original signal and the Head-Related Impulse Responses (**HRIRs**) from the given so...

According to the convolution property, the circular convolution of two finite-length sequences can be done by taking a DFT of each sequence, multiplying pointwise, and then performing an inverse DFT. Besides, zero-padding can make the circular convolution behave like linear convolution.

Q 4.1 Consider the two provided inputs: Sequence x_1 is the speech signal **"DSP_Speech.wav"**, sequence x_2 is given by 'HRIR_1.mat', which is an impulse response recorded from a loudspeaker to an in-ear microphone. You can use MATLAB inbuilt function *load()* to load '*.mat' data into your workspace.

Write a code to calculate the linear convolution between x_1 and x_2 using DFT. Then use MATLAB in-built function *conv* to check if you get the correct results.

Type:

>> help **conv**

in the command window to learn how to use it.

Attach the plots of the convolution output from both methods in your final report. Remember to briefly describe how did you implement the two convolution methods in MATLAB. **[2 Marks]**

Q 4.2 Load 'HRIR_2.mat' and convolve with the speech signal **"DSP_Speech.wav"** using MATLAB in-built *conv* function. Combine the results from Question 4.1 as channel 1 with the resultant vector from Question 4.2 as channel 2 to form a 2-channel vector. Using '*audiowrite()*', save this 2-channel vector as a '.wav' file named **'Binaural_Surname.wav'**. **[3 Marks]**

- Listen to **'Binaural_Surname.wav'** and comment on the difference compared to **"DSP_Speech.wav"**.
- Generate the time-domain plots of 'HRIR_1.mat' and 'HRIR_2.mat'. Attach them in your final report.
- Based on the listening experience and the plots, identify which of 'HRIR_1.mat' and 'HRIR_2.mat' corresponds to the impulse response of the **RIGHT EAR**.

Q 4.3 **(ENGN6537 Students ONLY)** Consider a signal of 2048 samples and a filter of 2048 samples. To perform the filtering of the signal using linear convolution by means of DFT, is zero-padding required? If yes, what is the length of the zero padding? If no, justify your answer. **[5 Marks]**

Q5 – SHORT-TIME FOURIER TRANSFORM (STFT)

[15 Marks for ENGN4537 students]

[15 Marks for ENGN6537 students]

The frequency content of s[illegible]ghly time variant. Therefore, it is important to address the problem of representing the [illegible]um of a signal. This can be done as a simple extension of the DFT introduced in the prev[illegible]lying a window "sliding" along time domain signal. This is theoretically referred to as t[illegible]er Transform (STFT). In MATLAB STFT is illustrated by a "**spectrogram**", which sho[illegible] frequencies in time. This information is very useful in the analysis of a signal since it gives a sort of signature in the time and frequency domain. The MATLAB in-built function for generating the STFT of a given signal is called ***spectrogram()*** and this function shows how the various frequency components of a signal evolve with time.

Type:

>> help **spectrogram**

in the command window to learn how to use it.

Q 5.1 Use your $dft2$ function to analyse the frequency content of your voice recording from Question 2 (i.e., "DSP_Givenname.wav"). Attach this plot in your final report. **[1 Mark]**

The noise signal '**DSP_Noise.wav**' has a harmonic structure, i.e., a set of peaks will appear in its frequency spectrum reflecting multiple tones. Let us denote the fundamental frequency of the signal as f_0, then the second and the third harmonics are given by $2f_0$ and $3f_0$.

Q 5.2 Using your $dft2$ function, estimate f_0 of the noise signal by analysing its spectrum. Attach the plot and note f_0 in Hz in your final report. **[2 Marks]**

Q 5.3 Based on the observations from Question 5.1 and Question 5.2, distinguish the frequency contributions of your voice and the noise signal. Note them in your final report. **[1 Mark]**

Q 5.4 Use MATLAB's ***spectrogram()*** function to analyse your recorded audio file (DSP_Givenname.wav). Attach the plot in your final report and discuss the output compared to that of Question 5.1. **[3 Marks]**

- Annotate (in MATLAB, using text boxes/arrows in the figure window) where you believe your own voice starts and stops in the plot.
- Use your name as the plot title.

The ***spectrogram()*** function allows us to input parameters like ***window*** function, ***overlap length***, ***DFT size*** and ***sampling frequency*** to control its working. It can also provide the outputs $[S, F, T]$, where S is the 2-D STFT matrix representing the frequency spectrum over time, F is the vector of frequencies in Hz at which the STFT is evaluated and T is the vector of time values in seconds corresponding to the centres of the data segments used to compute the STFT.

Q 5.5 Answer the following questions regarding ***spectrogram()***: **[3 Marks]**

- Which of the mentioned input parameter(s) directly influence the size of vector F?
- Which of the mentioned input parameter(s) directly influence the size of vector T?
- If the overlap length is 50% of window size, find the relation between the input parameter(s) and the resolution of vector T.

Q 5.6 Without looking at [illegible] MATLAB function called $FindSignalStart(x)$ that will determine when the speec[illegible]egins in a recorded audio signal (DSP_Givenname.wav). **[2 Marks]**

>> $[y, StartTime] = Find$[illegible]

where x is the input vector of samples of recorded audio signal. The output y contains the speech samples that is the same as input but with the first non-speaking samples removed, i.e., the spoken word begins right at the beginning of y. The output $StartTime$ is the time (in seconds) corresponding to the starting instant of your speech.

Note that:

- The MATLAB ***find*** command may be useful in determining where a threshold value is first exceeded.
- Do **NOT** use a loop in the function $FindSignalStart(x)$.

Hint: It is easier to do this task in frequency domain by using $[S, T, F] = spectrogram(x)$. Think about how your voice changes the frequency spectrum. There are several methods to achieve the task. Any logical method with correct output is acceptable. Listen to your result to check if your code work.

Q 5.7 Without looking at any plots, write a MATLAB function called $FindSignalStop(x)$ that will determine when the speech signal ends in a recorded audio signal (DSP_Givenname.wav). There may be some dead time after the final word is spoken. **[2 Marks]**

>> $[y, EndTime] = FindSignalStop(x)$

where x is the input vector of samples of recorded audio signal. The output y contains the speech samples same as input but with the last non-speaking samples removed, i.e., the spoken word ends right at the ending of y. The output $EndTime$ is the time (in second) corresponding to the ending instant of your speech.

Note that

- You should use the same ideas you had for the $FindSignalStart(x)$ function.
- The ***flipud*** or ***fliplr*** function may be useful here.
- Do **NOT** use a loop in this function.

Q 5.8 Do the following steps and attach the spectrogram plot: **[1 Mark]**

- Step 1: Use your $FindSignalStart$ function and $FindSignalStop$ function to cut your recorded audio file (DSP_Givenname.wav).
- Step 2: Save your new voice signal as DSP_ Chopped_Givenname.wav.
- Step 3: Play this file to check.

Your voice should start right at the beginning and end right at the ending.

Q6 – FILTERING

[15 Marks for ENGN4537 students]

[25 Marks for ENGN6537 students]

In this section, we try to cle ng filtering and acquire desired clean speech only.

Familiarize with the MATL FVTool is a Graphical User Interface (GUI) that allows you to analyse digital filters).

>> $fvtool\ (b, a)$ launches t n Tool and computes the Magnitude Response for the filter defined by numerator and denominator coefficients in vectors b and a, respectively. The tool also provides other helpful visualizations like phase response, impulse response, pole-zero plots, etc.

>> $fvtool(b, a, x)$ will display the magnitude response of the filter and the frequency content of the input signal x.

Q 6.1 Consider the three filters below:

$$H_1(\Omega) = \frac{0.9504 - 1.8484\ e^{-j\Omega} + 0.9504\ e^{-2j\Omega}}{1 - 1.8484\ e^{-j\Omega} + 0.9009\ e^{-2j\Omega}}$$

$$H_2(\Omega) = \frac{0.0945 - 0.129\ e^{-j\Omega} + 0.0945\ e^{-2j\Omega}}{1 - 1.6244\ e^{-j\Omega} + 0.6832\ e^{-2j\Omega}}$$

$$H_3(\Omega) = \frac{0.2346 - 0.3187\ e^{-j\Omega} + 0.2346\ e^{-2j\Omega}}{1 + 0.4419\ e^{-j\Omega} + 0.2231\ e^{-2j\Omega}}$$

Use *fvtool* to generate filter magnitudes response plots of the three filters. Attach the plots in your final report. Based on these plots, comment on how you think they would perform on filtering the fundamental component of the noise signal '**DSP_Noise.wav**' used during your voice recording. **[6 Marks]**

Q 6.2 Pick the most efficient filter from Question 6.1 to filter your original voice recording from Question 2 (i.e., DSP_Givenname.wav) using MATLAB in-built function ***filter***. **[4 Marks]**

Type:

>> help **filter**

in the command window to learn how to use it.

If the filtered signal is called y, run $fvtool(b, a, y)$ to see the shape of the frequency spectrum of the filtered signal. Since the process of filtering is done using convolution in the time domain, in the frequency domain we are multiplying the frequency content of the signal by the frequency response of the filter.

Attach the plot in your final report. Listen to the filtered signal y, and comment on the effect of the filter.

Now, let us consider the following filter:

$$H_4(\Omega) = \frac{1 - 1.93\, e^{-j\Omega} + 1\, e^{-2j\Omega}}{1 - 1.91\, e^{-j\Omega} + 0.98\, e^{-2j\Omega}}$$

Q 6.3 Do the following steps: **[5 Marks]**

- Step 1: Filter your o[illegible]g from Question 2 (i.e., DSP_Givenname.wav) using $H_4(\Omega)$.
- Step 2: Listen to the [illegible]omment on the effect of the filter.
- Step 3: Using *fvtool* [illegible]ro plot of $H_4(\Omega)$. Attach this plot in your final report.
- Step 4: Using the po[illegible]he sound of the filtered audio signal from Step 1.

Q 6.4 **(ENGN6537 Students ONLY)** Design a **low-pass** filter to filter out the noise (includes all the harmonics) in your original recording (DSP_Givenname.wav). **[5 Marks]**

- You are allowed to use MATLAB inbuilt function ***lowpass()*** or inbuilt app **filter designer/builder** or **any MATLAB tools** to design the required low-pass filter.
- Use this filter to filter your recording signal, and save the result as 'DSP_ LP_Givenname.wav'.
- Listen to the filtered signal. Comment on the effectiveness of your filter.

It is encouraged to submit your filter even if it may not be able to do a perfect job.

Q 6.5 **(ENGN6537 Students ONLY)** Design **band-pass** filter(s) to filter out the noise (includes all the harmonics) in your original recording (DSP_Givenname.wav). **[5 Marks]**

- This may be achieved by one band-pass filter or multiple filters.
- Use the filter(s) you designed to filter your recording signal.
- Save the final result as 'DSP_ BP_Givenname.wav'.
- Listen to the filtered signal. Comment on the difference between this band pass filter(s) and the low pass filter in Question 6.4. Which one do you prefer?

Q7 – MUSIC MODULATION (Amplitude Modulation) AND FILTER DESIGN (A Windowed, Finite Impulse Response Filter)

[30 Marks for ENGN4537 students]

[40 Marks for ENGN6537 students]

In order to facilitate the tran[illegible] modulation and demodulation are usually applied in the field of engineering. In this sect[illegible]l be amplitude modulated and then demodulated using FIR low-pass filter. Download t[illegible] **"DSP_Music.wav"** from the course website. If you read in the signal using ***audioread***, then run the ***fvtool*(1,1,*a*)** command (if *a* is the input signal), then *fvtool* will display the frequency content of the signal *a*. The music is given in Stereo way (2 channels), use **the first channel** to complete the rest of the questions.

__Q 7.1__ Amplitude modulation (AM) is a modulation technique commonly used in electronic communication. In this question, you should apply the AM technique by multiplying the music clip with a sine wave whose frequency is **9 kHz**.

Listen to the modulated music and answer the following questions.

Q 7.1.1 Write a MATLAB code file to plot the modulated music signal in the time domain. Then, use ***dft2()*** to plot its frequency and phase components in the frequency domain. Redo the process above for the original music signal (i.e., plot its time-domain figure, and frequency and phase responses in the frequency domain). Attach your plots (i.e., you should attach 6 plots in Question 7.1.1). **[3 Marks]**

Q 7.1.2 Compare the frequency-domain plots between the original signal and the modulated signal. After the Amplitude Modulation (AM) above, how are the frequency components of the original music clip changed? How are the phase components of the original music clip changed? **[1 Mark]**

Q 7.1.2 (a) What is the Fourier Transform (FT) of the modulation signal (i.e., the sine wave whose frequency is 9 kHz)? **[1 Mark]**

Q 7.1.2 (b) Use the results from Question 7.1.2 (a) to explain the results from Question 7.1.2 (i.e., explain why the frequency and phase components of the original music clip are changed in this way). **[2 Marks]**

Q 7.1.3 Compare the time-domain plots between the original signal and the modulated signal. Listen to the two music clips. How is the modulated music changed in time domain? **[1 Mark]**

To restore the original music, the modulated signal has to be demodulated first.

Demodulation is the inverse process of modulation, so a sine wave same as that in Question 7.1 is multiplied by the modulated music signal. Use ***dft2*** to analyse the frequency content of the demodulated signal (attach the plot). Afterwards, design a windowed FIR low pass filter to remove or attenuate the high-frequency part in the signal, while keeping as much frequency content of the actual signal as possible, to maximize the quality of the filtered signal. Listen to the demodulated music and the goal is to filter the signal so you cannot hear the annoying tones.

Design constraints:

- Your filter should have as small a transition width as possible.
- It must have fewer than 200 coefficients.
- It must suppress side lobes (stopband attenuation) > 50 dB.

Q 7.2 Explain how the mod [illegible] red. Explain why multiplying the same sine wave can restore the original music. Besides, [illegible] ce between the original music and the restored music? What would happen to the magnit [illegible] music this time? **[3 Marks]**

Q 7.3 Describe how you [illegible] (s) (refer to 'Handout for Windowed FIR LPF Design' in MATLAB project folder in [illegible]

Q 7.4 Answer the following questions.

Q 7.4.1 Which kind of filter should we use? A low-pass filter (LPF) or a high-pass one (HPF)? Briefly explain your choice. **[1 Mark]**

Q 7.4.2 Why should we use a FIR filter, not an IIR filter, in Question 7? Do some desktop research and briefly compare the benefits and drawbacks of FIR and IIR filters. **[1 Mark]**

Q 7.4.3 Why should we use a causal filter, not a non-causal one? Do some desktop research and briefly compare the benefits and drawbacks of causal and non-causal filters. **[1 Mark]**

Q 7.4.4 In Step 5 of the provided "Handout for Windowed FIR LPF Design" file, you are asked to shift the impulse response $h_2[n]$ to the right by (N-1)/2. If possible, plot the phase response of the impulse response before and after shifting (i.e., plot the phase response of $h_2[n]$ and $h[n]$). If not, briefly explain why you cannot plot the phase response(s). Attach your plots in the final report. **[2 Marks]**

Q 7.4.5 Using the results from Question 7.4.4, explain how Step 5 would make changes to the filter's output in detail. Note that you should explain it in both frequency domain and time domain. Besides, you should justify any differences between the two phase-response plots with respect to the number of samples shifted (i.e., (N-1)/2) in Step 5 of the 'Handout for Windowed FIR LPF Design' PDF. **[3 Marks]**

Hint for Question 7.4.4 & 7.4.5:

In these questions, you need to use the **Fourier Transform's Properties** to explain theoretically what kind of changes Step 5 would cause to the filter in the **frequency domain**, especially to the **filter's phase response**.

Besides, you should explain whether these changes are visible in MATLAB's plots or not. If yes, please attach the related plots (before and after Step 5) to your report. If not, please explain why.

You can search how MATLAB handles **causal/non-causal** systems for more information.

Q 7.4.6 **(ENGN6537 Students ONLY)** In Step 3 of the provided "Handout for Windowed FIR LPF Design" file, you are asked to select a suitable window function $w[n]$. Explain why we should use a window function to chop the IIR filter and make it becomes a FIR filter. (**Hint**: Learn the characteristics of a window function.) **[10 Marks]**

Q 7.5 Create a plot for the magnitude of the frequency response of your filter (in dB) using *fvtool*, and on the same plot, display the frequency content of the corrupted signal (i.e., the demodulated signal). Turn in this plot. **[1 Mark]**

Q 7.6 Fill in the information below from your design. Note that you should attach related plots in your final report to briefly explain how you obtain the information below. **[2 Marks]**

- Passband edge frequ ______ Hz
- Transition width = _
- Window type: _____ mber of coefficients: ______________
- Stop band attenuatic esign process): ____________ dB

Q 7.7 Fill in the information below from your frequency response magnitude plot. Again, you should attach related plots to briefly explain how you obtain the information below. **[1 Mark]**

- Actual stop band attenuation: _____________ dB
- -3dB Bandwidth: _______________ Hz

Filter the corrupted signal (i.e., the demodulated signal) with your filter. Listen to the filtered signal, you should hear the music, but NOT the annoying tones.

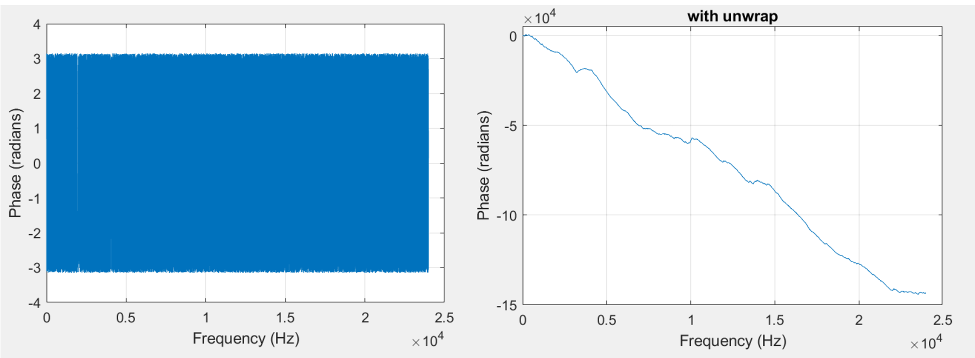
Q 7.8 Create a plot for the magnitude of the frequency response of your filter (in dB) using *fvtool*, and on the same plot, display the frequency content of the filtered signal (recall that *fvtool* will allow you to plot both). Turn in this plot. **[1 Mark]**

Q 7.9 Describe how well your filter seems to work, both audibly and based on the plot you just created. **[2 Marks]**

Hint for Question 7:

In Question 7, you are asked to compare the phase response of the original music clip with that of the modulated music clip (the one from Q7.1). For a better visualization, we suggest you apply the MATLAB function **'unwrap()'** on the phase response plots obtained from your **'dft2()'** function.

The following two example plots illustrate the differences of the 'Phase Response Plot' before and after using the **'unwrap()'** function. The left plot is the phase response plot directly obtained from your **'dft2()'** function, where the phase angles are limited within the range of $[-\pi, \pi]$. It is difficult to observe the trend of phase change from this plot. Therefore, we suggest you apply the **'unwrap()'** function to the original phase response, which results in the right-hand side (RHS) figure below. In the RHS plot, the phase response becomes a continuous curve, which is much better for observation and comparison.



Congratulations!

You have completed the MatLab Project for this course.

Remember to include all the requested files in your final submission.