EE320 Filtering and Source Separation Assignment

Liam Tang 202135591

January 26, 2024

Please Note

This document contains only the worked solutions for questions. The code used to implement the plots and audio outputs can be found in the folders attached where there are separate Matlab files for each question.

1 Analysis of Sinusoidal Interference

1.1 Question 1 (Time domain analysis of interference frequency)

To estimate the interference frequency f_0, the first 250 samples of x_1[n] were visually inspected using a stem plot. This sample range was strategically selected as the interference sinusoid was most dominant during this window. This can be visualised in Figure 1.

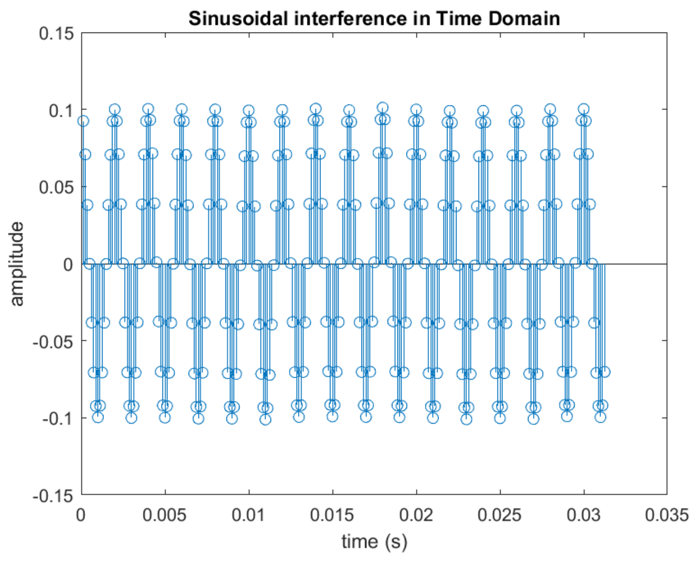

Figure 1: Interference Sinusoid stem plot

From Figure 1, the period was found to be T_0=*0.002s* by comparing the time difference between adjacent wave crests. The frequency was calculated, thus yielding:

$$f_0 = \frac{1}{T_0} = 500\,\text{Hz} \tag{1}$$

1.2 Question 2 (Frequency domain analysis of interference frequency)

Analysis of the interference frequency in the Fourier domain involved computing the Fourier transform of the signal and plotting the output. Figure 2 shows a spike at 500Hz and represents the interference frequency since it is present throughout the entire audio signal.

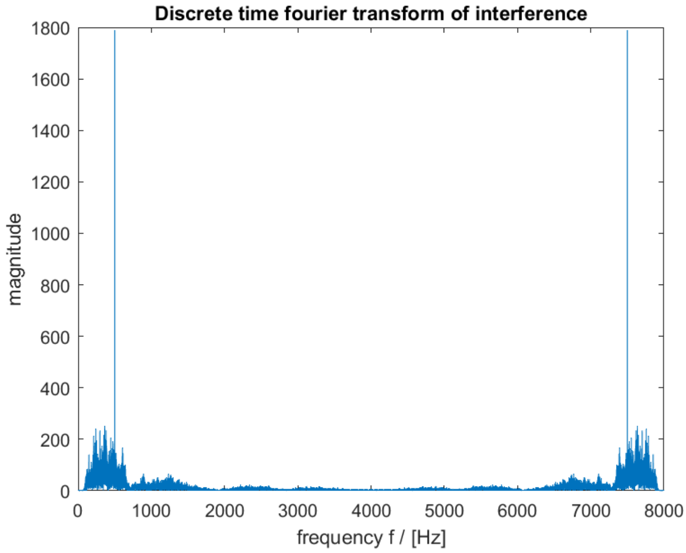

Figure 2: Fourier transform analysis

Since the analysis in the time and frequency domain align, the frequency of the interference sinusoid can be confidently confirmed as 500Hz.

2 Analysis of Delay/Angle of Arrival and Gain

2.1 Question 3 (Delay estimation in the time domain)

An estimation of the delay using time domain analysis was ascertained by calculating the difference in time between adjacent peaks from x_1[n] and x_2[n]. The plots shown in Figure 3 were used to obtain $\Delta t = 3.8 \times 10^{-4}\ s$.

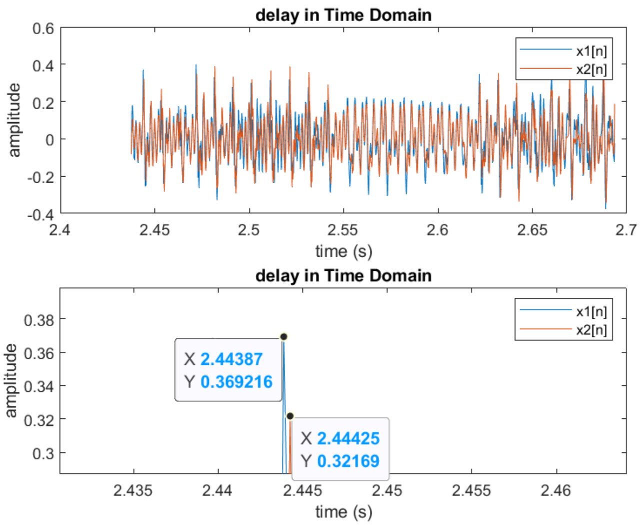

Figure 3: Time domain analysis of delay

2.2 Question 4 (Delay and gain analysis in the Fourier domain)

Written analysis of the delay and gain in the Fourier domain involved utilisation of Fourier transform property knowledge and consultation of the lecture notes.

$$G(\Omega) = \mid \frac{X_2(e^{j\Omega})}{X_1(e^{j\Omega})} \mid \tag{2}$$

Gathering an expression for $x_2[n]$:

$$x_1[n] \circ\!\!-\!\!\bullet X_1(e^{j\Omega)}$$
$$x_2[n] = \rho x_1[n-\tau]$$
$$\rho x_1[n-\tau] \circ\!\!-\!\!\bullet \rho e^{-j\Omega\tau} X_1(e^{j\omega})$$
$$\therefore x_2[n] = \rho e^{-j\Omega\tau} X_1(e^{j\Omega}) \tag{3}$$

Substituting (3) into (2):

$$G(\Omega) = \mid \frac{\rho e^{-j\Omega\tau} X_1(e^{j\Omega})}{X_1(e^{j\Omega})} \mid \tag{4}$$

$$G(\Omega) = \mid \rho e^{-j\Omega\tau} \mid$$

Where Gain $= \rho$ and $\angle = \Omega\tau$

2.3 Question 5 (Delay and gain estimation in the Fourier domain)

2.3.1 gain estimation

Estimating the gain in the Fourier domain involved the execution of an element-wise quotient of X2 and X1, leaving an array, X3, of the gains at each frequency. Samples in the 19501 to 21548 range were used as speaker 1 is more dominant during this window. Note that X1 and X2 are simply x1 and x2 but Fourier transformed such that $x_1[n] \circ\!\!-\!\!\bullet X_1(e^{j\Omega})$, and likewise for $x_2[n]$. The mean of this array gave $\rho_1 = 1.0934$.

An alternative method to this is to plot the magnitude response of X3 and assume the gain from this without consideration of outliers. This can be accomplished visually or by use of a histogram. This method could be preferred since taking the mean of the gain could capture outlying spikes. These spikes would be created when X1 values were close to zero, creating a large X3 value that could bias the final mean reading.

After testing the gains derived from both methods, it was evident that the mean() function approach produced more convincing results - hence its selection in the final solution.

2.3.2 delay estimation

To determine the delay, the phase angles of X3 were computed using the angle function in Matlab. This phase response can be visualised in Figure 4.

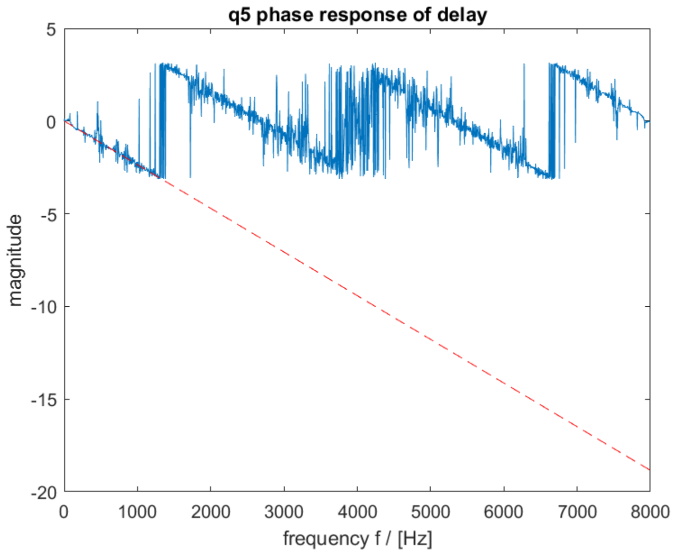

Figure 4: Phase response of delay

Finding the delay, τ_1, can be achieved by first inspecting the line of best fit and its intersection with the upper boundary x-axis. Using this method, a value of -18.8496 is obtained. By dividing by 2π and rounding to the nearest integer, the delay in terms of sampling periods can be determined.

$$\frac{-18.8496}{2\pi} = -3.00007 = 3\,delay\,samples \tag{5}$$

To find the delays in terms of time, the delay in samples can be normalised by dividing by the sampling frequency, f_s.

$$\tau_1 = \frac{3\,delay\,samples}{8000Hz} = 3.75 \times 10^{-4}s \tag{6}$$

It is worth noting that the delay here, estimated in the Fourier domain, is similar to that of the one estimated in the time domain in Question 3 ($\Delta t = 3.8 \times 10^{-4}\ s$)

2.3.3 angle calculation

Now that the time delay of signals received from speaker 1, $s_1[n]$, has been calculated, the angle v_1 can be calculated using formulae derived from the diagram shown in Figure 5.

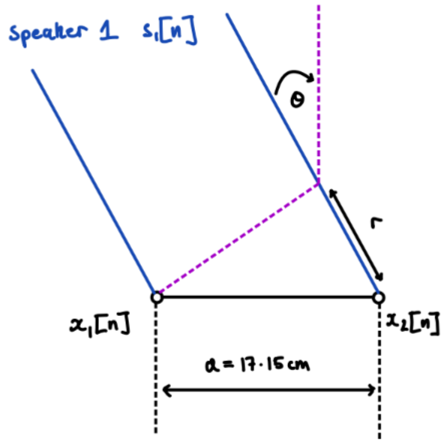

Figure 5: Diagram of delay

From this diagram, the following equations can be deduced.

$$r = v \times \tau_1 \tag{7}$$

$$\sin\theta = \frac{r}{d} \tag{8}$$

Plugging the known variables into (7), and assuming the speed of sound to be v=*331m/s*:

$$r = 331 \times 3.75 \times 10^{-4} = 0.124125m \tag{9}$$

Plugging known variables into (8):

$$\theta = \sin^{-1}(\frac{r}{d}) = \sin^{-1}(\frac{0.124125}{17.15 \times 10^{-2}}) = v_1 = 46.37° \tag{10}$$

2.4 Question 6 (Estimation for 2nd speaker)

The same approach in Question 5 can be used to estimate the delay, gain, and angle v_2 between $x_1[n]$ and $x_2[n]$ for speaker 2, $s_2[n]$. To specify it for speaker 2, a different sampling window was used where speaker 2 was more dominant than speaker 1. This was found to be in the range 24800 to 25824.

2.4.1 gain estimation

$$\rho_2 = 1.3153 \tag{11}$$

2.4.2 delay estimation

The phase response of the X3 quotient is illustrated in Figure 6.

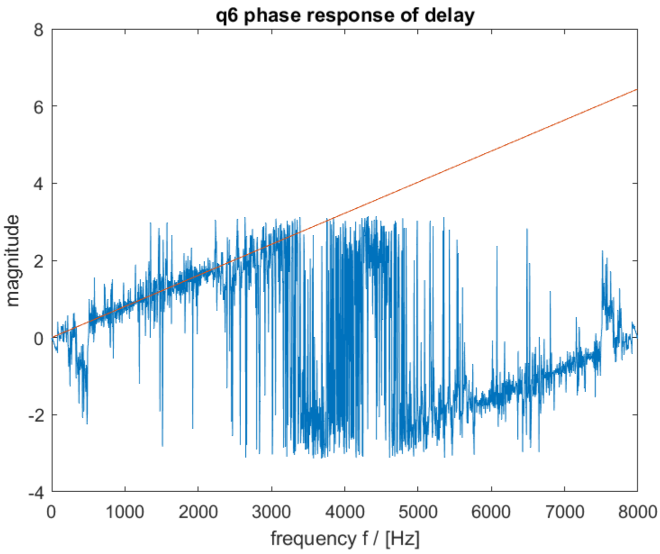

Figure 6: Diagram of delay

$$\tau_2 = 1\,delay\,sample = 1.25 \times 10^{-4} s \tag{12}$$

2.4.3 angle calculation

Although speaker 2 is in a different direction than speaker 1, relative to the microphones $x_1[n]$ and $x_2[n]$, the same notion and equations in Question 5 can be used.

$$\theta = \upsilon_2 = 13.96^\circ \tag{13}$$

3 Source Separation Design and Implementation

3.1 Question 7 (Estimation of relative transfer functions)

The matrix of relative transfer functions was produced by using the equation in Figure 7, taken from the assignment instructions document.

$$\begin{bmatrix} x_1[n] \\ x_2[n] \end{bmatrix} = \begin{bmatrix} \rho_{1,1}\delta[n-\tau_{1,1}] \\ \rho_{2,1}\delta[n-\tau_{2,1}] \end{bmatrix} * s_1[n]$$

Figure 7: Representation of 2 microphone signals receiving a single signal

Note that this equation only applies to a single signal. Using the same approach but tailoring it for 2 signals, $s_1[n]$ and $s_2[n]$, the transfer functions in the n and z domains are achieved such that H[n] $\circ\!\!-\!\!\bullet$H(z).

$$\begin{bmatrix} 1 & 1.0934[n-3.75\times10^{-4}] \\ 1.3153[n-1.25\times10^{-4}] & 1 \end{bmatrix} \circ\!\!-\!\!\bullet \begin{bmatrix} 1 & 1.0934z^{-3} \\ 1.3153z^{-1} & 1 \end{bmatrix} \tag{14}$$

To get it into H[n] form, the gains and delays were substituted into the form in Figure 7 but accounting for 2 singles received. Transformation into the H(z) form entailed replacing the [n-*delay in time*] with $z^{-(delay\ in\ samples)}$.

3.2 Question 8 (Construction of separation filters)

Using the identity G(z)=$H^{-1}(z)$ and the inverse matrix formula, the unmixing matrix G(z) was defined as (15).

$$H^{-1}(z) = G(z) = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \tag{15}$$

Substituting the gains rho_1 and rho_2 gave (16)

$$G(z) = \frac{1}{1 \times 1 - (1.0934z^{-3})(1.3153z^{-1})} \begin{bmatrix} 1 & -1.0934z^{-3} \\ -1.3153z^{-1} & 1 \end{bmatrix} \tag{16}$$

Replacing gain values with their respective variables (17)

$$G(z) = \frac{1}{1 \times 1 - (\rho_1 z^{-3})(\rho_2 z^{-1})} \begin{bmatrix} 1 & -\rho_1 z^{-3} \\ -\rho_2 z^{-1} & 1 \end{bmatrix} \tag{17}$$

Simplification leads to the final unmixing matrix (18)

$$G(z) = \begin{bmatrix} \frac{1}{1-\rho_1\rho_2 z^{-4}} & \frac{-\rho_1 z^{-3}}{1-\rho_1\rho_2 z^{-4}} \\ \frac{-\rho_2 z^{-1}}{1-\rho_1\rho_2 z^{-4}} & \frac{1}{1-\rho_1\rho_2 z^{-4}} \end{bmatrix} \tag{18}$$

Note that this form was preferred, rather than having explicit rho_1, rho_2 gain values, due to coding flexibility in Matlab. Also, it aligns with the code format issued in the assignment instructions.

```
%defining signals picked up by microphones
[x1,x2] = AssignmentScenario(202135591);

%define sample frequency
fs = 8000;

%defining rho values
rho1 = 1/1.0934;
rho2 = 1/1.3153;

y1 = filter(1,[1 0 0 0 -rho1*rho2],x2) + filter([0 0 0 -rho1],[1 0 0 0 -rho1
    *rho2],x1);
y2 = filter([0 -rho1],[1 0 0 0 -rho1*rho2],x2) + filter(1,[1 0 0 0 -rho1*
    rho2],x1);

sound(y1,fs);
%sound(y2,fs);
```

Listing 1: Separation Implementation

When run in matlab, the code displayed in Listing 1, performs successful separation of signals with the interference frequency still in the background. The reciprocal of gains ρ_1 ρ_2 are taken since gains greater than one lead to a *causal* system. Since the reciprocals are taken, the order of x1 and x2 operations are swapped.

3.3 Question 9 (Bonus – efficient implementation)

One strategy to implement the unmixing matrix more efficiently would be to use 1 large filter rather than 4 individual ones. This way, the computing speed would be much faster, contributing to a more efficient system.

4 Interference Suppression

4.1 Question 10 (Notch filter analysis)

The following equations (19) and (20) are used to determine the denominator and numerator coefficients.

$$Q(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{a_0 + a_1 z^{-1} + a_2 z^{-2}} \tag{19}$$

$$Q(z) = \frac{(1 - e^{j\Omega_0} z^{-1})(1 - e^{-j\Omega_0} z^{-1})}{(1 - \gamma e^{j\Omega_0} z^{-1})(1 - \gamma e^{-j\Omega_0} z^{-1})} \tag{20}$$

4.1.1 Denominator coefficients

Expanding denominator of (20):

$$(1 - \gamma e^{j\Omega_0} z^{-1})(1 - \gamma e^{-j\Omega_0} z^{-1}) \tag{21}$$
$$= 1 - 4\gamma(e^{j\Omega_0} + e^{-j\Omega_0})z^{-1} + \gamma^2 z^{-2} \tag{22}$$

Using the identity $cos(\Omega n) = \frac{1}{2}\left(e^{j\Omega_0} + e^{-j\Omega_0}\right)$:

$$= 1 - 4\gamma cos\Omega_0 z^{-1} + \gamma^2 z^{-2} \tag{23}$$

Comparing the coefficients of (19) with (23), the following can be deduced:

$$a_0 = 1, a_1 = -4\gamma cos\Omega_0, a_2 = \gamma^2 \tag{24}$$

4.1.2 Numerator coefficients

The same process used in the Denominator coefficients section can be applied to the numerators. However, it is noticed that the numerator is the same as the denominator given $\gamma = 1$.

$$\therefore\ b_0 = 1, b_1 = -4cos\Omega_0, b_2 = 1 \tag{25}$$

4.1.3 Analysis when $\gamma = 0.9$

Substituting $\gamma = 0.9$ into the denominator and numerator coefficients (24), (25):

$$a_0 = 1, a_1 = -3.6cos\Omega_0, a_2 = 0.81 \tag{26}$$

$$b_0 = 1, b_1 = -4cos\Omega_0, b_2 = 1 \tag{27}$$

4.1.4 Analysis when $\gamma = 0.99$

Substituting $\gamma = 0.99$ into the denominator and numerator coefficients (24), (25):

$$a_0 = 1, a_1 = -3.96cos\Omega_0, a_2 = 0.9801 \tag{28}$$

$$b_0 = 1, b_1 = -4cos\Omega_0, b_2 = 1 \tag{29}$$

4.1.5 Magnitude responses

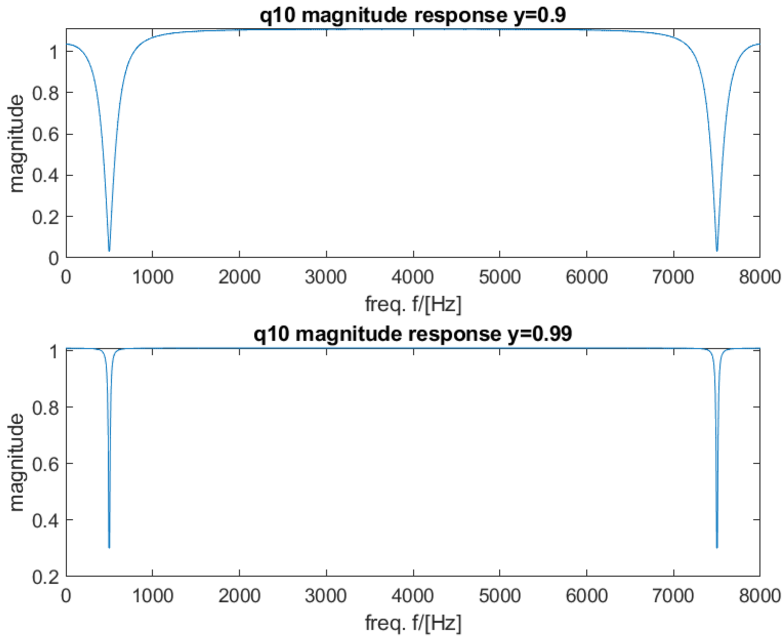

Figure 8: Magnitude Response Plot

4.2 Question 11 (Notch filter implementation and application)

After implementing the notch filter code for both γ values, both filters perfectly removed the sinusoidal interference. When comparing the filter performances, it was concluded that both audio signals sounded just as good to the naked human ear.

However, when inspecting the magnitude responses, as shown in Figure 8, the $\gamma = 0.99$ plot has a narrower stopband than the $\gamma = 0.9$. This could indicate that the $\gamma = 0.9$ filter removes wanted frequencies, as well as the 500Hz sinusoidal interference. Since the stopband of the $\gamma = 0.99$ filter is narrower, the final audio output will be finer since unnecessary frequencies will not be removed.

4.3 Question 12 (Filter sequence)

The notch filters and unmixing system can be swapped since all system blocks are linear time-invariant.