Bangladesh University of Engineering and Technology

Department of Electrical and Electronic Engineering (EEE)

Course No.: EEE 312 **Section:** A2

Experiment No. 4

Assignment – 4: Frequency domain analysis of DT signals and systems

Instructions: *Include your Matlab code snippets and all necessary command window output+plots in your report.*

1. Consider the filters given below. For each of them, the poles and zeroes have been pointed out. Comment on whether the filters are **low-pass, high-pass, bandpass, or notch**.

i. $H_1(z)$: poles $[-0.25 \pm j0.74, -0.18 \pm j0.4, -0.16 \pm j0.13]$, zeros $[1]^{\times 6}$.

ii. $H_2(z)$: poles $\left[0.86e^{\pm j22.66^\circ},\ 0.67e^{\pm j10.48^\circ}\right]$, zeros $\left[1e^{\pm j76.35^\circ},\ 1e^{\pm j124.45^\circ}\right]$.

iii. $H_3(z)$: poles $[0.16 \pm j0.96,\ 0.35 \pm j0.81,\ 0.57 \pm j0.8]$, zeros $[\pm 1,\ 0.08 \pm j0.99,\ 0.64 \pm j0.77\ ]$.

iv. $H_4(z)$: poles $\left[0.98e^{\pm j35^\circ}\right]$, zeros $\left[1e^{\pm j35^\circ}\right]$.

2. Take two examples of a signal $x(n) \xleftrightarrow{DFT,\ N\ point} X(k)$ from your choice and show that **Parseval's Theorem** holds true:

$$\sum_{n=0}^{N-1} |x(n)|^2 = \frac{1}{N}\sum_{k=0}^{N-1} |X(k)|^2$$

3. Two continuous-time signals are given:

$$p(t) = 4\sin(100\pi t) + 3\cos(300\pi t) + \sin(800\pi t)$$

$$q(t) = 3\cos(300\pi t) + \sin(800\pi t)$$

Use a sampling frequency of $2000\ Hz$. For each mentioned signal, iteratively find the **lowest number of DFT points** that will satisfy the two conditions below:

(a) Only the main frequency representative peaks [three for $p(t)$ and two for $q(t)$] are visible within the $[0, \pi]$ frequency range.

(b) Each peak is located within 1.5% error of its actual representative frequency.

4. Record your voice signal, $s(n)$ and corrupt is with coloured noise, $w(n)$:

$$y(n) = s(n) + w(n)$$

Compare the power spectral density (**PSD**) of the corrupted signal, $|Y(k)|^2$, with the original voice signal's PSD, $|S(k)|^2$. Consider the coloured noise signal to be each of these: (i) Car noise, (ii) Classroom/Office Noise, (iii) Ceiling fan noise, (iv) Traffic noise. [Source of the noise can be found in the internet or you may record them separately]. From the obtained PSDs, compare $r_{yy}(m)$ and $r_{ss}(m)$ as well.

5. Implement an **FDM** system with the following protocol:

(i) $m_1(t)$ and $m_2(t)$ are voice recorded signals from you and your partner respectively (both saying the same sentence).

(ii) The sampling frequency found by "`audioread()`" will be too high for MATLAB to perform modulation. So you need to use a lower sampling frequency to accommodate for storage, which is effectively down sampling the audio. [The process for down sampling is depicted afterwards.]

(iii) $c_1(t)$ and $c_2(t)$ are carriers of $5kHz$ and $10kHz$.

(iv) Use a sampling frequency of $30kHz$ or $40kHz$. (This is the new F_s for down sampling. This also need to be greater than the carriers).

(v) As channel characteristics, add some **5~10 dB SNR** based gaussian noise after summing the two modulated signals.

(vi) Visualize each step of constructing the FDM signal. Use proper bandpass and lowpass **rectangular** filters to finally extract $m_1(t)$ and $m_2(t)$.

(vii) Listen to the extracted signals $m_1(t)$ and $m_2(t)$, as well as compare their sounds to the original versions. You will need `ifft()` to convert fourier transformed signals back to time domain signals.

(viii) For down sampling, study the code below and take code segments which are necessary for your implementation:

```
clc
close all
clear all

[m,fs] = audioread("Test0.mp3") ;   % sampling freqency from ".mp3"/".wav"
m = reshape(m,[1 length(m)]);
fs_lower = 40000;                   % lower sampling frequency: downsampling
to=length(m)/fs;
t_lower = [0:1/fs_lower:to-1/fs_lower];
t=[0:1/fs:to-1/fs];
m1 = interp1(t,m,t_lower);          % new audio clip

sound(m1,fs_lower)                  % sounds the same

fc = 5000;
c=cos(2*pi*fc*t_lower);
u=m1.*c;    % DSB -AM modulated Signal

N=2^15;
M=fft(m1,N);
M=M/fs_lower;%scaling

U=fft(u,N);
U=U/fs_lower;%scaling
fn = [0:1/N:1-1/N]*fs_lower-fs_lower/2;
subplot(211),plot(fn,abs(fftshift(M)))
subplot(212),plot(fn,abs(fftshift(U)))
```