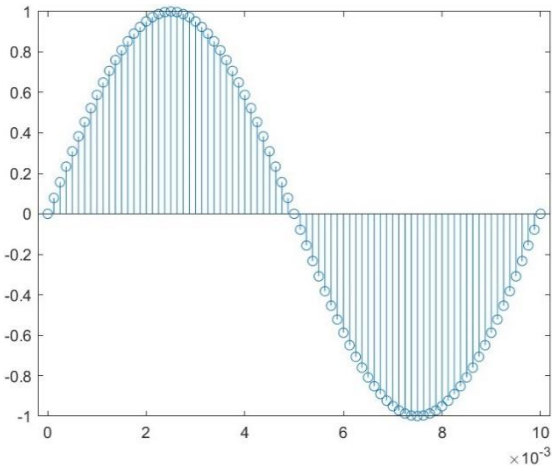
Lab 3: Aliasing in Signal Sampling

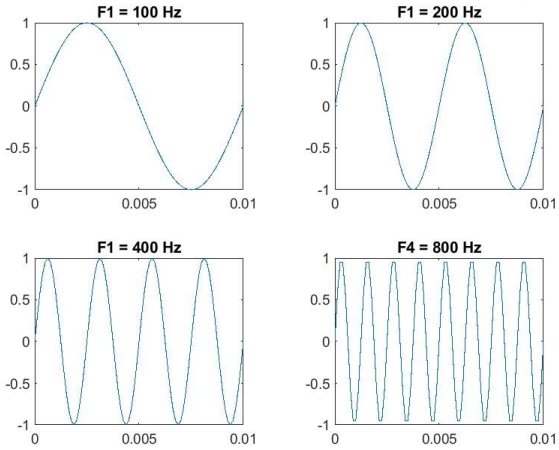
Professor: Dr. Mohamed Elamien
Names: Abdurahman Butt and Erion Keka
Student Numbers: 400435085 and 400435050
Submission Date: 2024-11-21

Part 1 - Aliasing in the Telephone System:

1. The graph shown beside, obtained through running the MATLAB code in the lab manual shows a sinusoid signal with a frequency of 100 Hz, sampled at a frequency of 8000 Hz. It has the form $x(t) = \sin(2\pi ft + \phi)$ where ϕ is equal to zero. The output demonstrates one full cycle of the sinusoid matching with T = (1 / f) = (1 / 100 Hz) = 10 ms. The plot is created utilizing the stem function in MATLAB.



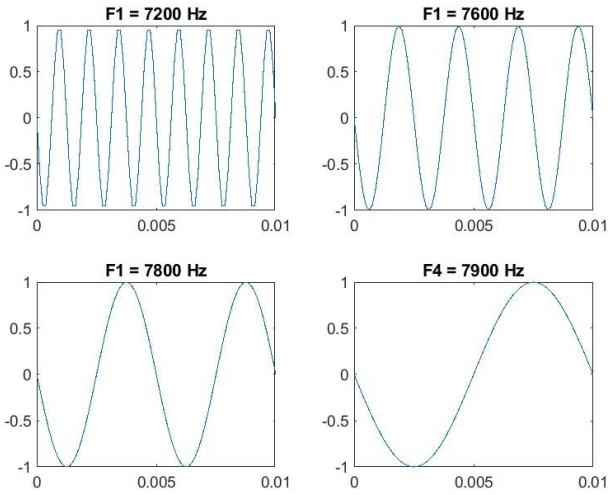
2. After changing the MATLAB program to allow for four plots of different frequencies and an audio output featuring 2-second tone segments of each frequency, we obtain the output on the right. The audio output consists of a humming sound that changes pitch every two 2 seconds. Every 2 seconds, the pitch increases due to the increase in the frequency. The code for this program is include below.



Question2.m

```
% Different frequencies for different plots
f1 = 100;
f2 = 200;
f3 = 400;
f4 = 800;

% Sampling frequency and interval
fs = 8000;
Ts = 1/fs;

% Set time duration of plot, i.e., 10 msec.
tfinalplot = 10e-3;

% Make the time vector for the plot
nplot=0:Ts:tfinalplot;

% Play the spurt for 2 seconds
tfinal = 2;
nsound=0:Ts:tfinal;

% First Frequency
subplot(2,2,1);
xnT1 = sin(2*pi*f1*nsound);
plot(nplot,xnT1(1:length(nplot)));
title('F1 = 100 Hz');

% Second Frequency
subplot(2,2,2);
xnT2 = sin(2*pi*f2*nsound);
plot(nplot,xnT2(1:length(nplot)));
title('F1 = 200 Hz');

% Third Frequency
subplot(2,2,3);
xnT3 = sin(2*pi*f3*nsound);
plot(nplot,xnT3(1:length(nplot)));
title('F1 = 400 Hz');

% Fourth Frequency
subplot(2,2,4);
xnT4 = sin(2*pi*f4*nsound);
plot(nplot,xnT4(1:length(nplot)));
title('F4 = 800 Hz');

% Save xnT as a wav sound file, soundfile.wav.
audiowrite('Question2.wav', cat(2, xnT1, xnT2, xnT3, xnT4), fs);

% Saving the graph as .jpg
exportgraphics(gcf, 'Question2.jpg');
```

3. After changing the frequencies within the MATLAB, we obtain the output on the right. When the input frequency of the sinusoid is increased, the output frequency of the sinusoid is decreased. This is apparent in the audio file as well, unlike in question two, the pitch decreases every 2 seconds. Since the input frequency is changing whilst all else is held constant, we are experiencing aliasing. We are sampling much slower than the Nyquist rate. The Nyquist rate states that we must sample at least two times the highest frequency to avoid aliasing. We are sampling at a rate of 8000 Hz which is far from two times our frequencies of 7200, 7600, 7800, and 7900 Hz thus, resulting in aliasing.

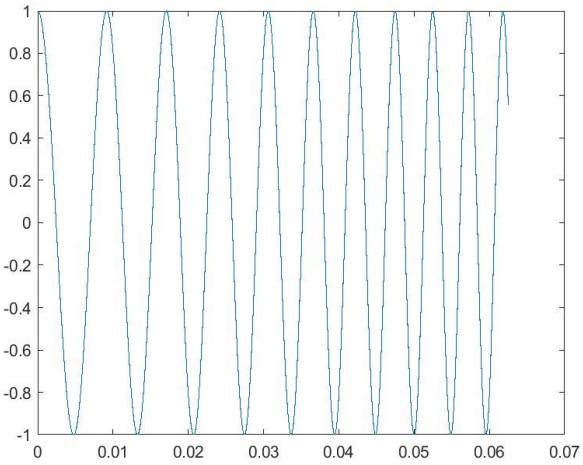


Question3.m

```
% Different frequencies for different plots
f1 = 7200;
f2 = 7600;
f3 = 7800;
f4 = 7900;

% Sampling frequency and interval
fs = 8000;
Ts = 1/fs;

% Set time duration of plot, i.e., 10 msec.
tfinalplot = 10e-3;

% Make the time vector for the plot
nplot=0:Ts:tfinalplot;

% Play the spurt for 2 seconds
tfinal = 2;
nsound=0:Ts:tfinal;

% First Frequency
subplot(2,2,1);
xnT1 = sin(2*pi*f1*nsound);
plot(nplot,xnT1(1:length(nplot)));
title('F1 = 7200 Hz');

% Second Frequency
subplot(2,2,2);
xnT2 = sin(2*pi*f2*nsound);
plot(nplot,xnT2(1:length(nplot)));
title('F1 = 7600 Hz');

% Third Frequency
subplot(2,2,3);
xnT3 = sin(2*pi*f3*nsound);
plot(nplot,xnT3(1:length(nplot)));
title('F1 = 7800 Hz');

% Fourth Frequency
subplot(2,2,4);
xnT4 = sin(2*pi*f4*nsound);
plot(nplot,xnT4(1:length(nplot)));
title('F4 = 7900 Hz');

% Save xnT as a wav sound file, soundfile.wav.
audiowrite('Question3.wav', cat(2, xnT1, xnT2, xnT3, xnT4), fs);

% Saving the graph as .jpg
exportgraphics(gcf, 'Question3-1.jpg');
```

4. The performance of the telephone system would degrade significantly if anti-aliasing pre-filtering was not used. The sampling rate implemented within telephone systems is based on the fact that most human voice energy is typically below 3.5 KHz which is rounded up to 4 KHz to capture more voices. By the Nyquist rate, the input is sampled at a rate of 8 KHz. The use of pre-filtering allows for telephones systems to filter out signals that are above 3.5 KHz. Without this filtering, input signals above 4 KHz which will cause aliasing to occur as it we are not ensuring the Nyquist frequency. In the above experiments, we noted that with our sampling rate of 8 KHz, the input frequencies of 7200, 7600, 7800, and 7900 Hz will create aliasing in the output signal. However, if the filter was applied in our experiments, we would be able to avoid the aliasing by filtering out the higher frequencies and solely allow frequencies that can be sampled accurately.

Part 2 - Aliasing of a Frequency Chirp Signal:



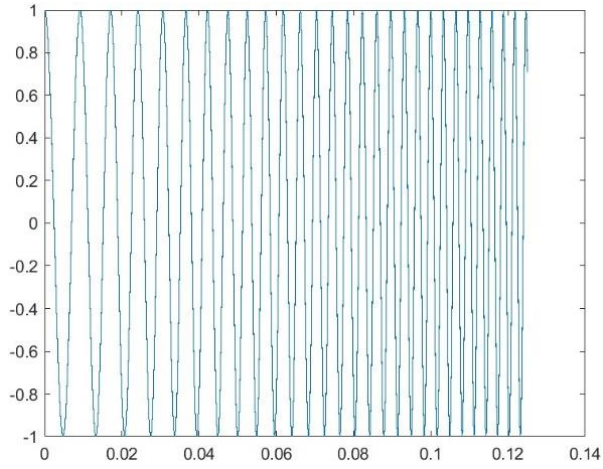
1. The graph shown beside, obtained through running the MATLAB code in the lab manual shows the signal with a frequency of 100 Hz and μ = 2000. The signal was sampled at a frequency of 32000 Hz. The signal has the following form: $c(t) = \cos(\pi\mu t^2 + 2\pi f_1 t + \phi)$. The graph shows an oscillation of the signal until reaching the end point found by doing $\frac{\mu}{f_s}$. The outputted audio file for this part features a sound whose pitch increases throughout the entire audio clip.

Question4.m

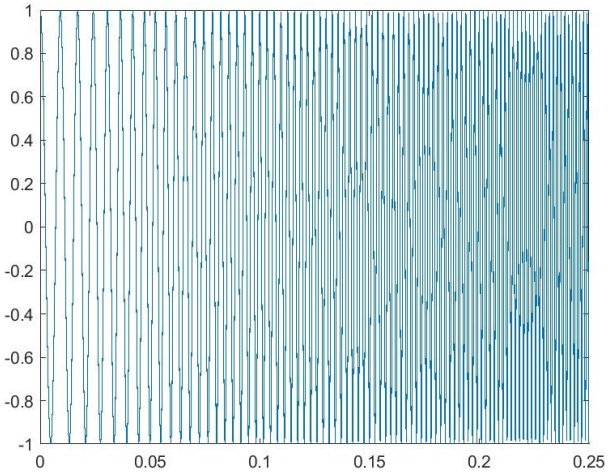
```
% Assigning frequency and samples
f = 100;
u = 2000;

% Sampling frequency and interval
fs = 32000;
Ts = 1/fs;
t = 0: Ts : 8;

% Set time duration of plot
tfinalplot = u/fs;

% Make the time vector for the plot
nplot=0:Ts:tfinalplot;

% Play the spurt for 8 seconds
tfinal = 8;
nsound=0:Ts:tfinal;

% Sampling the Function
cnT = cos(pi*u*t.^2 + 2*pi*f*t);

% Make the plot
plot(nplot,cnT(1:length(nplot)));

% Save cnT as a wav sound file, soundfile.wav.
audiowrite('Question4.wav', cnT', fs);

% Saving the graph as .jpg
exportgraphics(gcf, 'Question4.jpg');
```

2.

a. **16 KHz:** When changing the sampling frequency 16 KHz, we obtain the output on the right where we see that the graph features far more oscillations compared to the previous graph. This is due to reducing the sampling rate which affects the resolution in turn, creating these rapid oscillations. The number of periods in the output graph has doubled with the halving of the sampling frequency. In the audio output, we can hear that the pitch begins low then increases to its peak then decreasing again such that the pitch to begin the video is the same as that towards the end of the clip.



b. **8 KHz:** When changing the sampling frequency to 8 KHz, we obtain the output on the right. Once again, we can notice that output features far more oscillations compared to the previous two graphs. This again is due to the decrease in the sampling frequency. The number of periods in graph has once again been double from the first part as the sampling frequency has been halved once again. From the audio clip, we can hear that much like the previous part, the audio clip begins at a lower pitch then increases to its peak pitch; however, in this audio clip, this increase and decreasing occurs twice rather than once as compared to part a.



c. **Explanation:** From the lab manual, it can be noted that by taking the derivative of the signal's phase, we obtain the following function, $f(t) = \mu t + f_1$, that describes the frequency increasing linearly with time beginning at the frequency, f_1. Based on the values provided, we can solve the equation to find that our frequencies increase linearly from 100 Hz up to 16 100 Hz. From this, we should be sampling at a frequency of 32.2 KHz based on the Nyquist rate. However, initially, we sample at a frequency of 32 KHz which more closely aligns with the Nyquist rate but, in the next parts, our sampling frequency is reduced to 16 KHz and 8KHz which are much lower than the Nyquist frequency. As a result of the decreased sampling frequency, we hear the pitch decrease in the audio clips. Moreover, in the 16 KHz experiment, we only hear one pitch decrease compared to the two pitch decreases in the 8 KHz experiment. This occurs due to the halving of the sampling frequency thus, in the 16 KHz experiment, we obtain a

single audible cycle whereas, in the 8 KHz experiment, there's two audible cycles. Furthermore, over a telephone connection that includes the anti-aliasing filtering, the aliasing experienced would not be present. The anti-aliasing filtering is implemented to ensure that there are not frequencies that do not align with the sampling rate. Finally, by experimenting with various values of f_1, f_s, and μ, we can conclude that f_1 is responsible for speed at which the output increases or decreases. The sampling frequency, f_s, is responsible for maximum pitch level and its duration. Then, μ is found to be responsible for determining how frequently the signal oscillates.