ELECENG 3TP3 – Signals and Systems

Lab 3: Aliasing in Signal Sampling

Instructor: Dr. Kiruba

Prepared and submitted by
Marryam Kamal – kamalm18 – 400446997

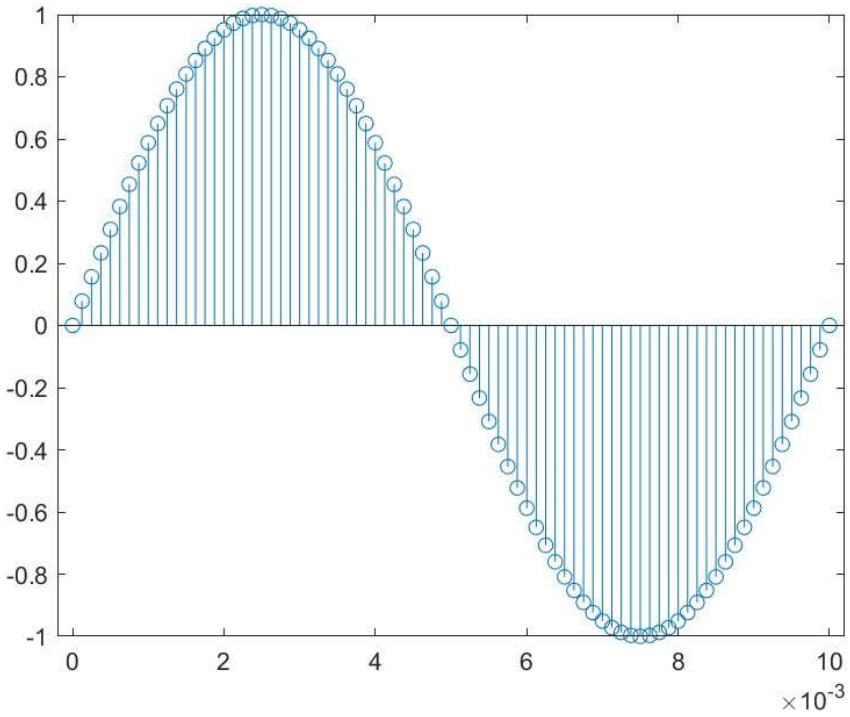
November 21st, 2024

Table of Contents

Part 1 – Aliasing in the Telephone System

1) The effects of aliasing on a sinusoid

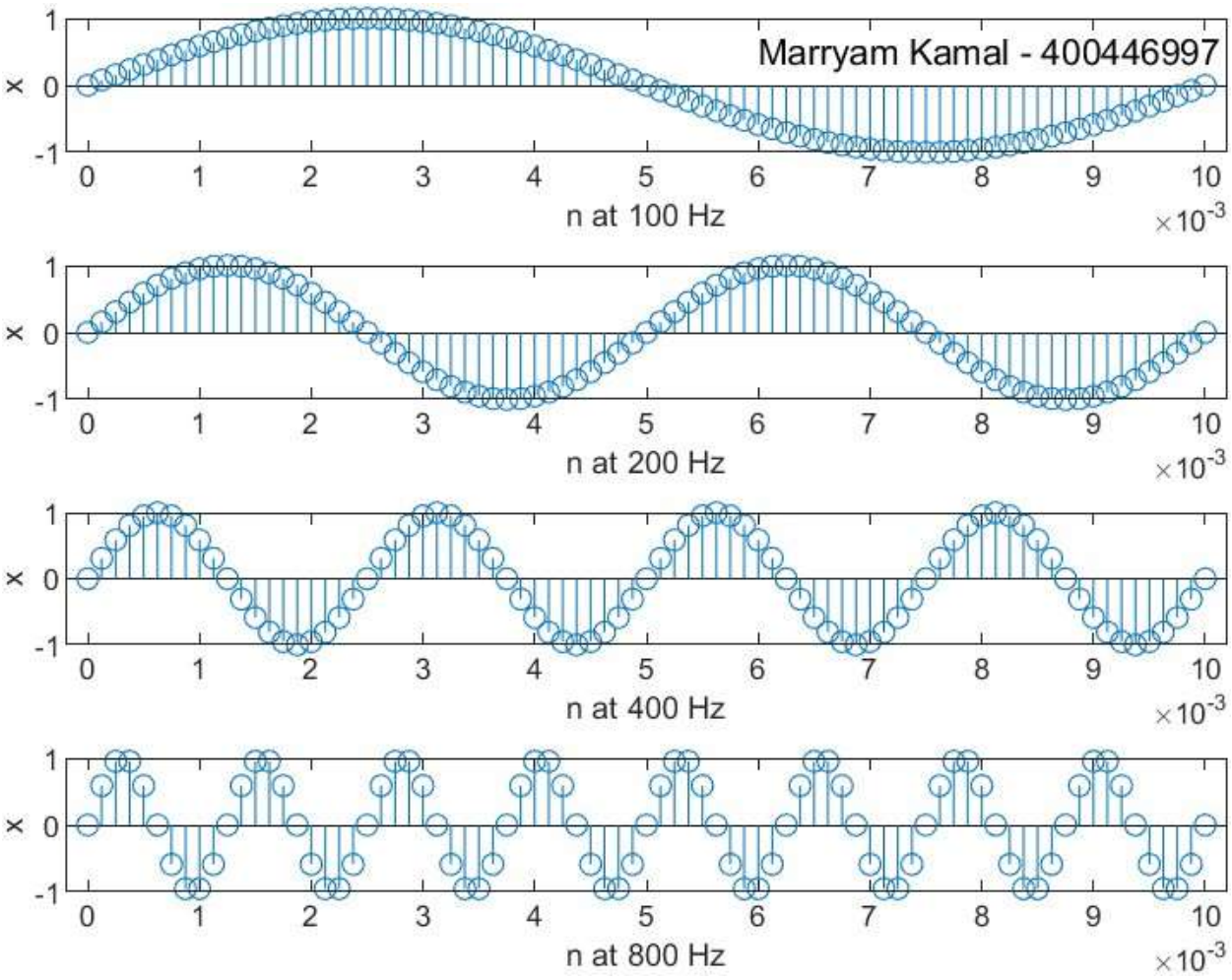
The output of the code displays the plot below, which shows a sinusoidal wave from 0 to 10ms, showing 8000/100 = 80 discrete samples.



2) Plotting f = 100, 200, 400, 800 Hz

Code:

```
function q2(f,x)
    fs = 8000;
    Ts = 1/fs;
    tfinalplot = 10e-3;
    nplot=0:Ts:tfinalplot;
    tfinal = 2;
    nsound=0:Ts:tfinal;
    xnT = sin(2*pi*f*nsound);
    subplot(4,1,x); stem(nplot, xnT(1:length(nplot)));
end

% Making the subplots
q2(100,1);
xlabel('n at 100 Hz'); ylabel('x');
q2(200,2);
xlabel('n at 200 Hz'); ylabel('x');
q2(400,3);
xlabel('n at 400 Hz'); ylabel('x');
q2(800,4);
xlabel('n at 800 Hz'); ylabel('x');
```

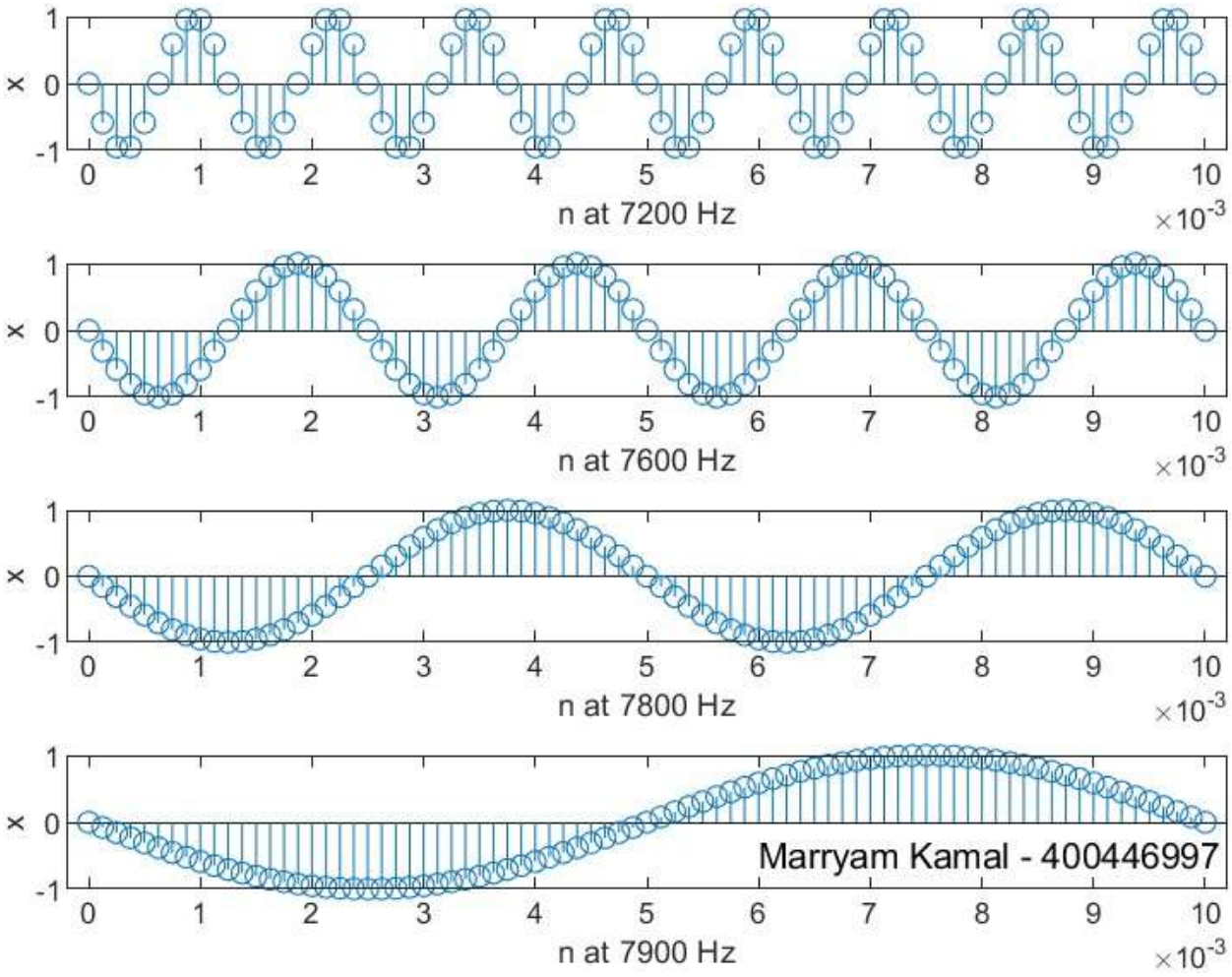


Code for concatenating the above into a single vector:

```
% Initialize the concatenated vector
xnT_combined = [];
tfinal = 2;
frequencies = [100, 200, 400, 800];
fs = 8000;
Ts = 1/fs;
nsound=0:Ts:tfinal;

for i = 1:length(frequencies)
    % Generate the sinusoidal tone
    f = frequencies(i);
    xnT = sin(2*pi*f*nsound);
    xnT_combined = [xnT_combined, xnT];
end

audiowrite('concatenated_sound_q2.wav', xnT_combined, fs);
```

When I play the sound, I hear a tone that increases in pitch every 2 seconds, with 100Hz being the lowest and 800Hz being the highest.

3) Plotting f = 7200, 7600, 7800, 7900 Hz



The same code was used to generate these graphs and the sound file as question 2.

At these frequencies, the sound appears to have the opposite effect as in question 2. The tone instead decreases in pitch, with 7200 Hz being the highest and 7900 Hz being the lowest. The Nyquist frequency of the signals is given as fs/2, which results in 8000/2 = 4000 Hz. Aliasing occurs when the input frequencies are higher than the Nyquist frequency, which is the case with the given input frequencies 7200-7900 Hz.

As a result, the signal "wraps around", resulting in very low output frequencies, comparable to those of inputs 200-800 Hz. Overall, the results are as follows:

	Input frequency	**Output frequency**
Below Nyquist frequency	Increasing	Increases as input increases
Above Nyquist frequency	Increasing	Decreases as input increases

4) Results of not using anti-aliasing filters

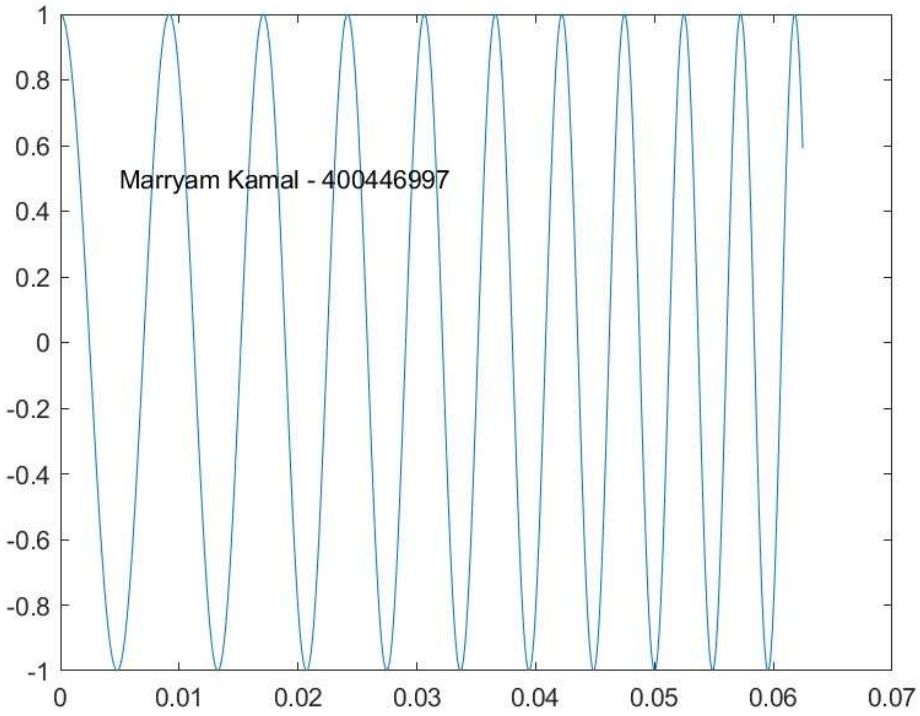
If anti-aliasing filters weren't applied to telephone systems, then the sampled signals will include unwanted low-frequency components that were not part of the original signal. These aliased frequencies distort the signal and make it impossible to reconstruct the original high-frequency content correctly for transmission to the other line.

Anti-aliasing filters are just low pass filters applied to reject input frequencies above the Nyquist frequency. By doing this, they ensure that only frequencies that can be accurately sampled are passed to the A/D and D/A converters.

If filtering is applied to the above experiment (f > 7200 Hz), then these signals would get filtered out, resulting in no sound being heard. This is because these input frequencies are above the Nyquist frequency. If it was applied to the experiment before this (f < 800 Hz), the sound would play as normal.

Part 2 – Aliasing of a Frequency Chirp Signal

1) Plotting fs = 32 kHz



Code:

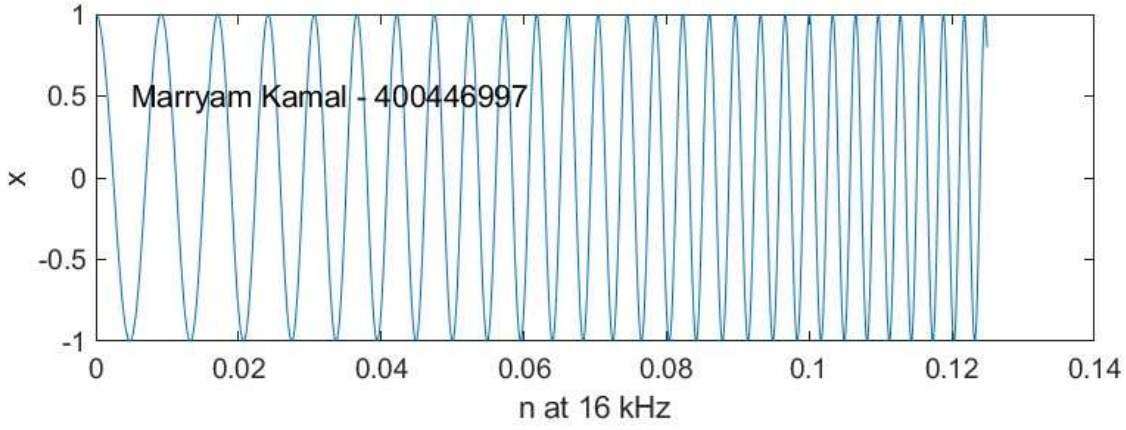
```
f = 100;
fs = 32000;
Ts = 1/fs;
tfinalplot = 8;

nplot=0:Ts:tfinalplot;
xnT = cos(pi*2000*nplot.^2 + 2*pi*f*nplot);
```

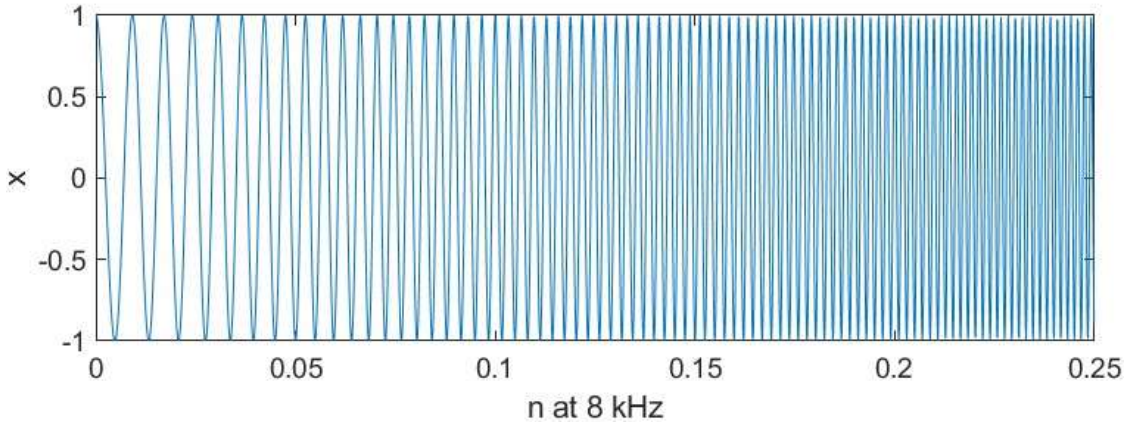
```
% Make the plot
plot(nplot(1:2000), xnT(1:2000));
text(0.005, 0.5,'Marryam Kamal - 400446997', 'FontSize', 10)
exportgraphics(gcf, 'q4_graph.jpg');
audiowrite('q4.wav', xnT', fs);
```

When I play the sound, I hear a tone that increases in pitch linearly in proportion to time, starting low and then ascending to a high pitch (high frequency) signal. This makes sense, as the frequency of the signal is increasing following the given function, the derivative the signal's phase.

$$f(t) = \mu t + f_1.$$

2) Plotting fs = 16, 8 kHz





The same code was used to generate these graphs and the sound file as question 1.

At 16 kHz, the audio sounded like the 32 kHz sample, however this time it appeared to "bounce back" once, resulting in a linearly increasing, followed by a decreasing pitch/frequency. This is a result of aliasing, occurring because the sampling frequency is now above the Nyquist frequency of the signal.

The Nyquist frequency, as discussed before, is equivalent to half the sampling frequency. In the 16 kHz case, the Nyquist limit would be 8 kHz. The maximum value the input frequency will reach can be easily calculated with the following equation, substituting μ = 2000, t = 8s and f1 = 100 Hz.

$$f(t) = \mu t + f_1.$$

The result is a max input of 16100 Hz, which greatly exceeds the Nyquist limit. This is why the audio appeared to wrap around and decrease in frequency halfway through. The same phenomenon would occur for the 8 kHz case, a result of aliasing in the signal.

As discussed in part 1, anti-aliasing filters would ensure that the "bouncing back" of sound isn't audible on the receiver's end.

Sampling frequency (fs)	**Nyquist frequency**	**Exceeding or within the limits?**	**Results and explanation**
10 kHz	5 kHz	Exceeding	Since 5000 < 16100, aliasing occurs and the signal rolls back into unwanted lower frequencies.
18 kHz	9 kHz	Exceeding	Since 9000 < 16100, aliasing occurs and the signal rolls back into unwanted lower frequencies.
40 kHz	20 kHz	Within limits	Since 20000 > 16100, aliasing doesn't occur and the signal plays as expected at a continuously increasing frequency.