ELECENG 3TP3

Lab3 Report

Xu Xikai –C02-400440917-XU503

Nov 11, 2024

Q1. Aliasing on a sinusoid

The graphic shows a Sinusoid diagram made from impulse signal, each sample point of the signal represents a discrete sample value of the sine wave at a given time interval. The points in the graph are arranged within a fixed sampling interval, capturing the shape of the sine wave at these specific sampling moments. The signal appears to be composed of discrete points rather than a continuous curve.

Code to reverse the process

% Do a plot of a sampled sinusoid with frequency f = 100 Hz

f = 100;

% 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;

% Sample the sinusoid.

xnT = sin(2\*pi\*f\*nplot);

% Make the plot

%stem(nplot, xnT);

figure;

plot(nplot \* 1e3, xnT, '-o');

title('xuxikai400440917');

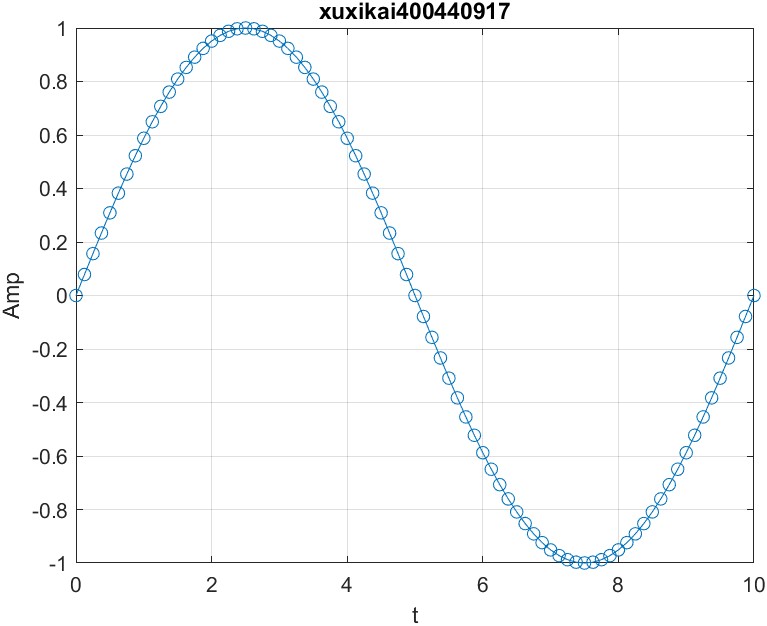
xlabel('t');

ylabel('Amp');

grid on;

% Uncomment/edit this next line to save the graph.

exportgraphics(gcf, 'milestone1.jpg');



Q2. Integration implimentation

Code:

frequencies = [100, 200, 400, 800];

fs = 8000;

Ts = 1 / fs;

tfinalplot = 10e-3

figure;

for i = 1:length(frequencies)

    f = frequencies(i);

    nplot = 0:Ts:tfinalplot;

    xnT = sin(2 \* pi \* f \* nplot);

    subplot(2, 2, i);

    plot(nplot \* 1e3, xnT);

    title(['f = ' num2str(f) ' Hz']);

    xlabel('t');

    ylabel('Amp');

    grid on;

end

saveas(gcf, 'milestone2s.jpg');

sound\_vector = [];

tfinal = 2;

for i = 1:length(frequencies)

    f = frequencies(i);

    nsound = 0:Ts:tfinal;

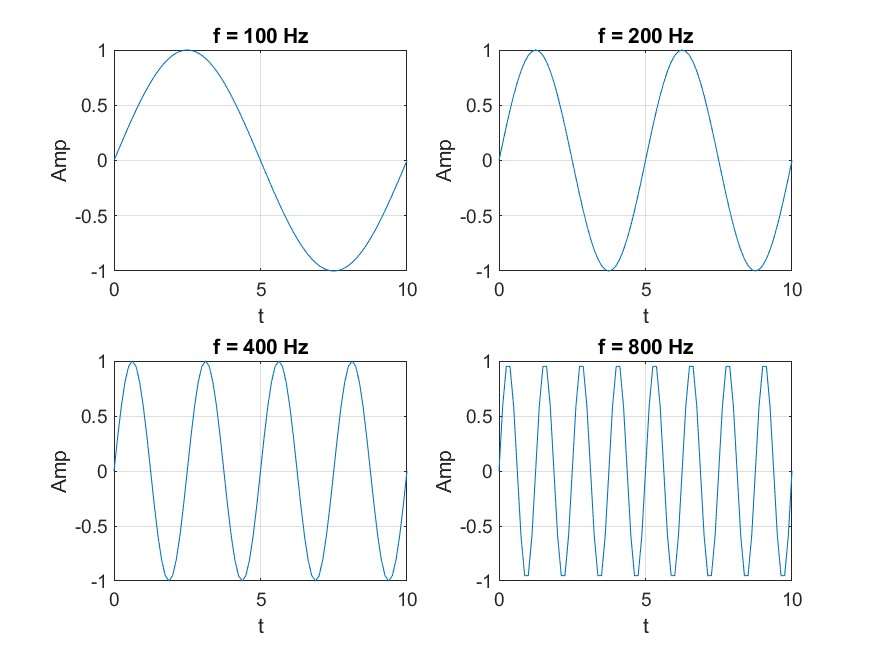
    xnT = sin(2 \* pi \* f \* nsound);

    sound\_vector = [sound\_vector xnT];

end

audiowrite('milestone2.wav', sound\_vector, fs);

sound(sound\_vector, fs);



The sound goes from inaudible to audible at the second wave frequency, then gradually increases in frequency until the audio is no longer audible.

Q3. Switch the frequency to 7200, 7600, 7800, 7900

Code: frequencies = [7200, 7600, 7800, 7900];

fs = 8000;

Ts = 1 / fs;

tfinalplot = 10e-3;

figure;

for i = 1:length(frequencies)

    f = frequencies(i);

    nplot = 0:Ts:tfinalplot;

    xnT = sin(2 \* pi \* f \* nplot);

    subplot(2, 2, i);

    plot(nplot \* 1e3, xnT);

    title(['f = ' num2str(f) ' Hz']);

    xlabel('t');

    ylabel('Amp');

    grid on;

end

saveas(gcf, 'milestone3s.jpg');

sound\_vector = [];

tfinal = 2;

for i = 1:length(frequencies)

    f = frequencies(i);

    nsound = 0:Ts:tfinal;

    xnT = sin(2 \* pi \* f \* nsound);

    sound\_vector = [sound\_vector xnT];

end

audiowrite('milestone3.wav', sound\_vector, fs);

sound(sound\_vector, fs);

As input frequencies like 7200, 7600, 7800, and 7900 Hz exceed the nyquist limit (4000 Hz for a sampling rate of 8000 Hz), **aliasing** occurs, causing higher frequencies to be "folded back" and heard as lower tones. This results in distorted sounds. In contrast, lower frequencies like 100, 200, 300, and 400 Hz are accurately sampled and sound clear because they are below the nyquist limit. This demonstrates how sampling rate affects the accuracy of frequency representation.

Q4. Filtering effect

Without anti-aliasing pre-filtering, a telephone system would suffer from aliasing, causing high-frequency components to fold back and distort the signal. Anti-aliasing filters prevent this by removing frequencies above the nyquist limit before sampling, ensuring only representable frequencies are captured. In your experiments, with filtering in place, you would hear clear and accurate sounds without distortion, as high frequencies would be filtered out before sampling.

Q5: Aliasing of a Frequency Chirp Signal

Code:

f1 = 100;

mu = 2000;

fs = 32000;

tmax = 8;

phi = 0;

t = 0:1/fs:tmax;

c = cos(pi \* mu \* t.^2 + 2 \* pi \* f1 \* t + phi);

figure;

plot(t(1:2000), c(1:2000));

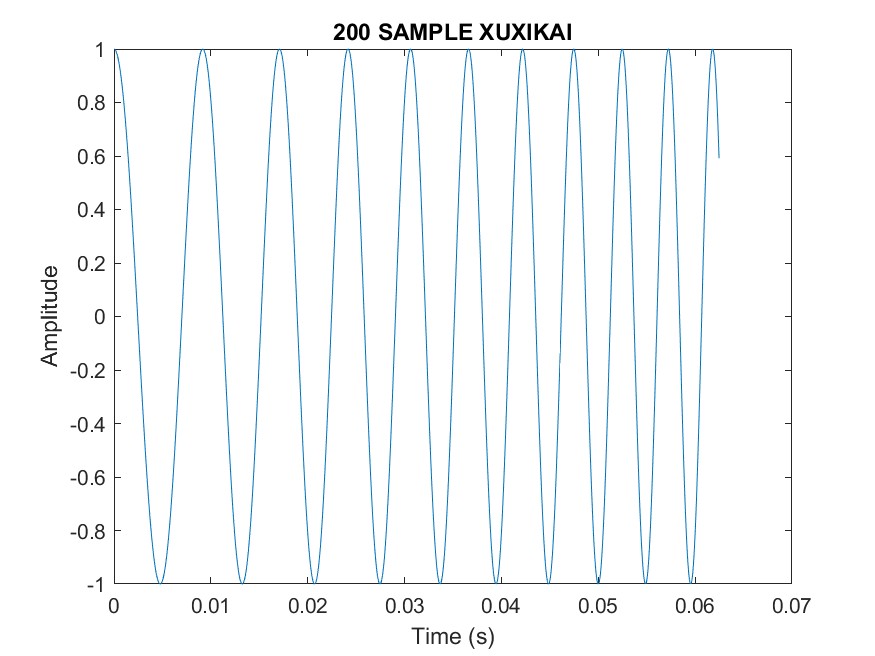
title('200 SAMPLE XUXIKAI');

xlabel('Time (s)');

ylabel('Amplitude');

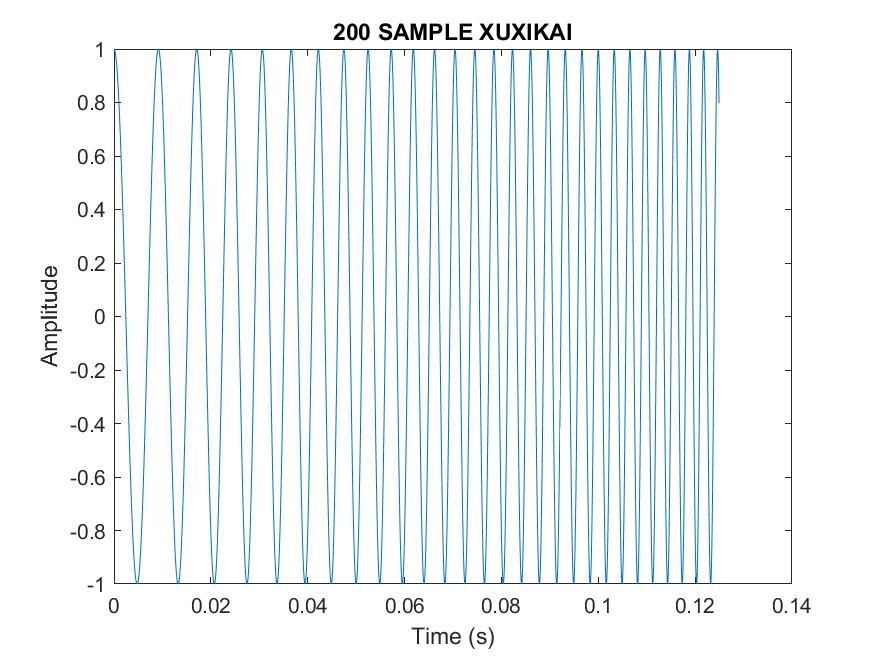
audiowrite('32kHz.wav', c, fs);

sound(c, fs);



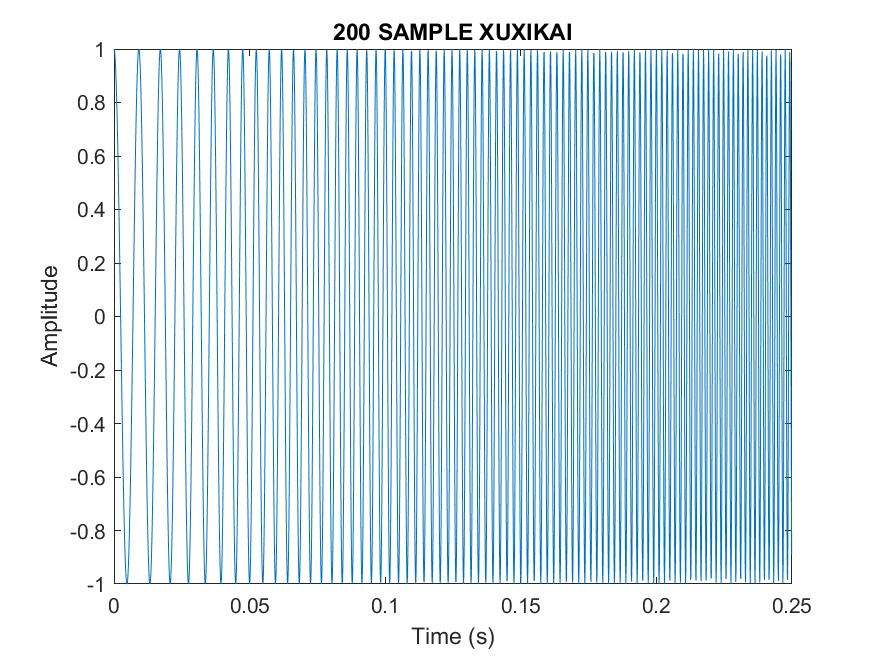
The plot will show the sampled frequency chirp. Since the sampling frequency is high, the chirp signal will be accurately represented in both the plot and the playback. I observed that the signal has an increasing frequency with time. For the soundFx, when played back, I heard a smooth, rising tone, where the pitch increases continuously as the frequency of the chirp increases.

fs = 16000; change to 16000Hz sampling



The plot will still show the frequency chirp, but the resolution is lower due to the reduced sampling rate. For the soundFx, I heard a bit distortion, which @ the end of the sound wave, it no longer peached. Since the sampling rate is not high enough to represent the increasing frequency accurately, higher frequencies fold back into the audible range, causing a "warped" sound. The higher-frequency components are not represented correctly, resulting in a distorted tone.

fs = 8000; change to 8000Hz sampling



The plot will show a very aliased signal, where high-frequency components are severely distorted. This happens because the sampling rate is insufficient to represent the signal's frequency components correctly. I heard significant distortion @ mid of the sound wave. The chirp will not sound smooth, as the high-frequency components will be aliased and produce lower, incorrect frequencies. This is what would happen if the signal were transmitted over a telephone network without anti-aliasing pre-filtering.

If the signal is passed through a telephone network and anti-aliasing pre-filtered, the high frequencies are removed before sampling, thus preventing aliasing from occurring. As a result, the signal will be cleaner and the pitch will rise smoothly, as it does when sampling at 32 kHz. The anti-aliasing filter prevents distortion caused by insufficient sampling rate, ensuring that high frequency components are not mistakenly folded back into the low frequency range.

Higher values for μ will cause the signal's frequency to increase more rapidly. Reducing the sampling rate below the nyquist frequency for these cases will lead to more pronounced aliasing and distortion. For both any types of freq index, if the signal’s frequency exceeds half the sampling rate, aliasing will occur. This is why a high sampling rate is crucial for accurate signal representation, particularly for signals with rapidly changing frequencies like chirps.