Lab 3

Kevin Le

McMaster University

ELECENG 3TP3

November 17th, 2023

Aliasing in the Telephone System:

1. The output of the given code is a sampled sine wave (discrete time). The plot shows one period of the sine wave. The points hold the shape of the graph well because there are 80 samples over one period of the wave. The sampling frequency is much higher than the frequency of the wave.

```
% 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);
% save the graph.
exportgraphics(gcf, 'graph1.jpg');
```

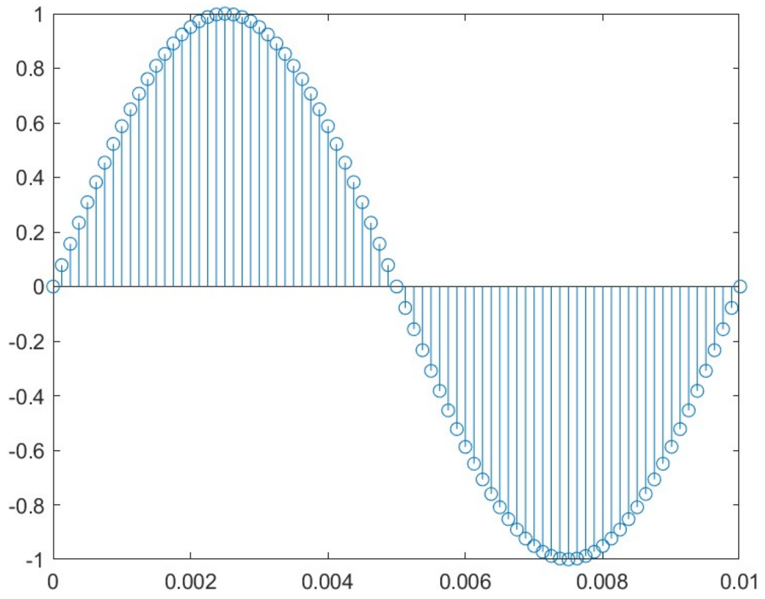

Figure 1: Stem plot of a sampled sinusoid signal. Signal has frequency of 100Hz, sampled at 8000Hz, time duration is 10ms.

2. Listening to the output sound file, I first heard a deep, low pitch sound, then a sound with a little higher pitch, then an even higher pitch sound, and finally a sound with the highest pitch. The four sounds are increasing in pitch, with a discrete change every 2 seconds. This makes sense because the original frequencies are 100Hz, 200Hz, 400Hz and 800Hz and the sampled and reconstructed signals also have the same frequencies. As the frequency increases, the sound gets higher pitched.

```
clc; clear all;
% 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;
% Make the time vector for replayed sound spurt
% Play the spurt for 2 seconds
tfinal = 2;
nsound=0:Ts:tfinal;
%vector for the four 2-second tone segments
xnT_concat = [];
frequencies = [100 200 400 800];
for i = 1:4
   % Use sinusoid frequency f = 100, 200, 400, 800 Hz
   f = frequencies(i)
   % Sample the sinusoid.
   xnT = sin(2*pi*f*nsound);
   xnT_concat = [xnT_concat xnT];

   % Make the plot
   my_title = [int2str(f) ' Hz'];
   subplot(2,2,i);
   plot(nplot, xnT(1:length(nplot)));
   title(my_title);
end
% Save xnT as a wav sound file, soundfile.wav.
audiowrite('soundfile.wav', xnT_concat, fs);
% Uncomment/edit this next line to save the graph.
exportgraphics(gcf, 'graph_2.jpg');
```

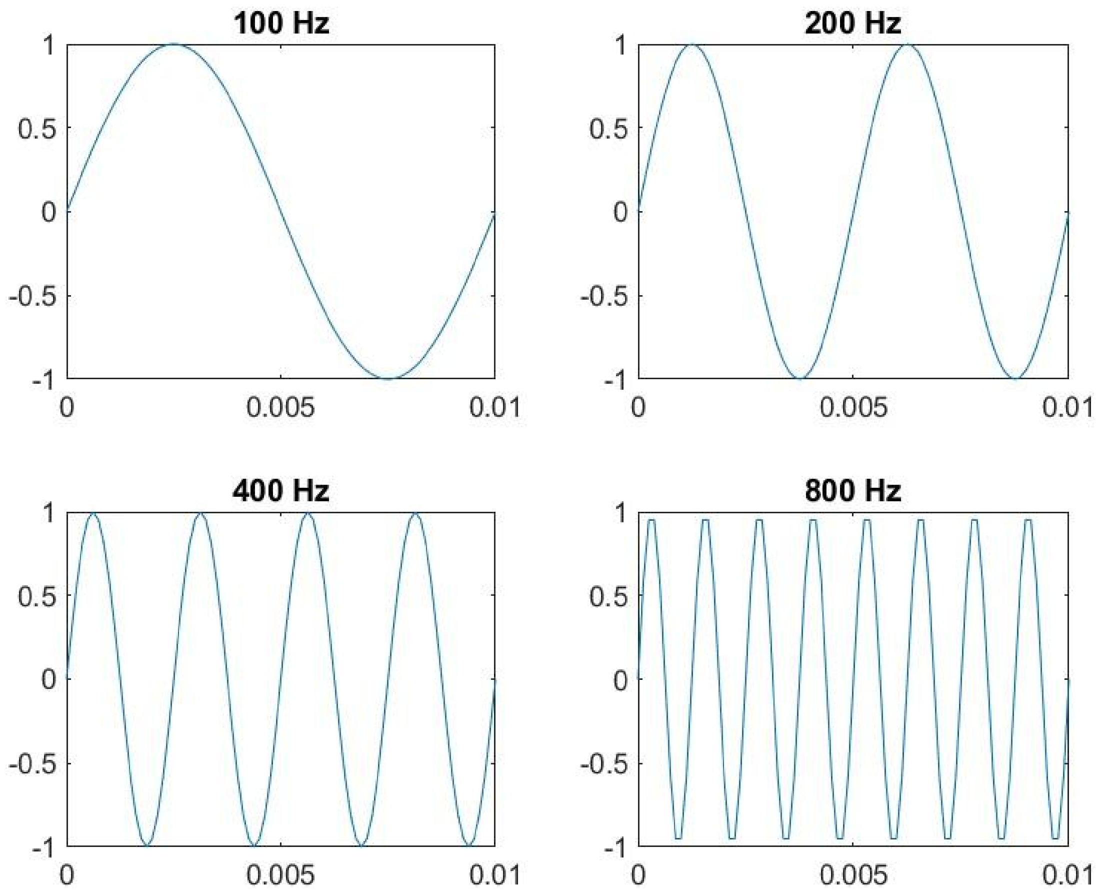

Figure 2: Graphs of sampled sine waves, converted back to analog using the "plot" command with frequencies 100 Hz, 200 Hz, 400 Hz and 800 Hz.

3. The sound of this audio file is the same as the previous but in the reverse order. The sound starts high pitch and decreases in pitch every 2 seconds. The output frequency seems to decrease as the input frequency increases. This may seem odd because the frequencies 7200 Hz, 7600 Hz, 7800 Hz and 7900 Hz are much higher than the previous frequencies and it should be expected that the sounds are much higher pitched. However, if we examine the output graphs, the waves seen on the graph match with what is heard. It can also be noticed that the graphs for 100 Hz and 7900 Hz, 200 Hz and 7800 Hz, 400 Hz and 7600 Hz, and 800 Hz and 7200 Hz are the same but with a 180 degree phase shift. The pairs of frequencies add to 8000 Hz, which is the sampling frequency. According to the Nyquist Sampling Theorem, the sampling rate must be at least 2 times the maximum frequency of an input signal. In this example 8000 Hz is much less than 2 times 7200 Hz, 7600 Hz, 7800 Hz and 7900 Hz. As a result, the sampled and output signal does not represent the input signal. For the system to follow the Nyquist Sampling Theorem, the maximum frequency of the input signal must be 4000 Hz (half of f_s = 8000 Hz). As the input frequency increases above 4000 Hz, the sampled and output frequency begins to decrease, not able to sufficiently represent the input signal, and worse, frequencies above 4 kHz produce an alias signal, identical to a signal between 0 kHz and 4 kHz.

```matlab
clc; clear all;
% 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;
% Make the time vector for replayed sound spurt
% Play the spurt for 2 seconds
tfinal = 2;
nsound=0:Ts:tfinal;
%vector for the four 2-second tone segments
xnT_concat = [];
frequencies = [7200 7600 7800 7900];
for i = 1:4
   % Use sinusoid frequency f = 7200, 7600, 7800, 7900 Hz
   f = frequencies(i);
   % Sample the sinusoid.
   xnT = sin(2*pi*f*nsound);
   xnT_concat = [xnT_concat xnT];

   % Make the plot
   my_title = [int2str(f) ' Hz'];
   subplot(2,2,i);
   plot(nplot, xnT(1:length(nplot)));
   title(my_title);
end
% Save xnT as a wav sound file, soundfile.wav.
audiowrite('soundfile_2.wav', xnT_concat, fs);
% Uncomment/edit this next line to save the graph.
exportgraphics(gcf, 'graph_3.jpg');
```

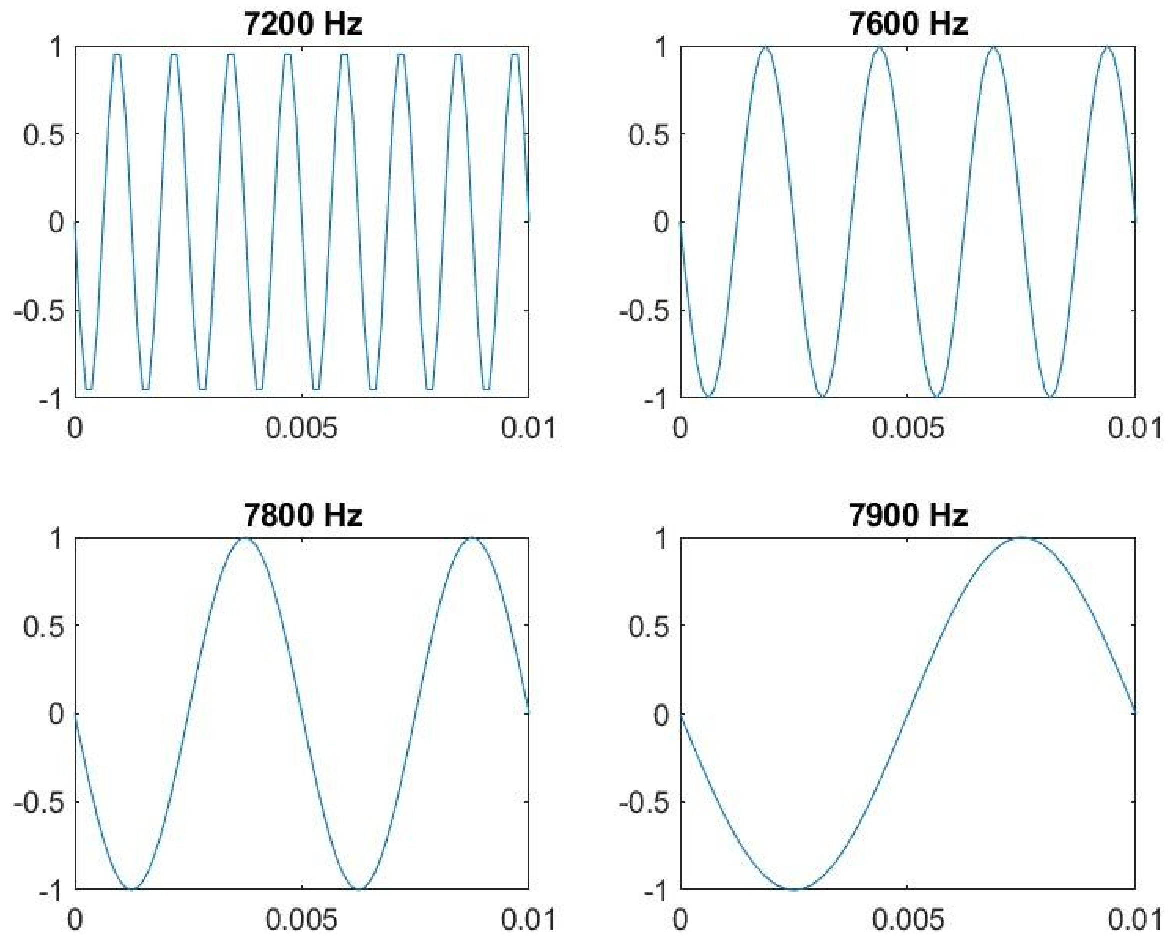

Figure 3: Graphs of sampled sine waves converted back to analog using the "plot" command, with original frequencies of 7200 Hz, 7600 Hz, 7800 Hz and 7900 Hz.

4. Because human voice is typically under 3.5 kHz, telephone systems use the sampling rate 8 kHz.According to the Nyquist sampling theorem, any sound with frequencies under 4 kHz, including human voice, can be properly transmitted. However, sounds above 4 kHz can also be passed through, but as seen in the above experiment, the other end of the telephone won't receive that actual signal but rather a signal identical to a regular signal under 4 kHz. This causes normal signals at the output to be mixed with undesired signals, causing high levels of distortion. This is what would happen without anti-aliasing pre-filtering. An anti-aliasing pre-filter, acting as a low-pass filter for the system, prevents frequencies above 4 kHz to be passed into the system while only allowing desired signals, signals under 4 kHz (such as human voice) to be passed in. By minimizing frequencies above 4 kHz into the system, this minimizes alias frequencies caused by signals with frequencies above 4 kHz. If this filter was in place for the above experiments, there won't be a difference for the frequencies 100 Hz, 200 Hz, 400 Hz and 800 Hz, however, the frequencies 7200 Hz, 7600 Hz, 7800 Hz and 7900 Hz gets filtered out, and no signal will be seen at the output, or at worst, the output signal will have a significantly reduced amplitude, making the produced sounds much more difficult to hear.

Aliasing of a Frequency Chirp Signal:

1. Listening to the output audio, I hear a sound that starts with a low pitch and continuously increases in pitch over the 8 seconds. This makes sense because the signal being sampled has a starting frequency of 100 Hz and increases in frequency over time. This is the reason for the increase in pitch of the sound. The rate of increase is 2000 Hz per second, so at 8 seconds, the frequency is 2000 * 8 + 100 = 16,100 Hz. The Nyquist sampling rate for this signal is 16,100 * 2 = 32,200 Hz ≈ 32 kHz. The signal is sampled at the Nyquist sampling frequency, therefore, the sound output sounds as expected. The output graph also looks correct, with the signal showing an increase in frequency over time. Because there are 2000 samples, the graph stops at 2000/32000 Hz = 0.0625 seconds.

```
clc; clear all;
% Sampling frequency and interval
fs = 32000;
Ts = 1/fs;
% Starting frequency and increase slope
f1 = 100;
mu = 2000;
% Make the time vector for replayed sound spurt
% Play the spurt for 2 seconds
tfinal = 8;
nsound=0:Ts:tfinal;
% Make the time vector for the plot
nplot=0:Ts:(2000-1)*Ts;
% Sample the sinusoid.
cnT = cos(pi*mu*nsound.^2 + 2*pi*f1*nsound);
% Make the plot
plot(nplot, cnT(1:2000));
title('32 kHz Sampling Frequency');
% Save cnT as a wav sound file, soundfile.wav.
audiowrite('soundfile_chirp_1.wav', cnT, fs);
% Uncomment/edit this next line to save the graph.
exportgraphics(gcf, 'graph_chirp_1.jpg');
```

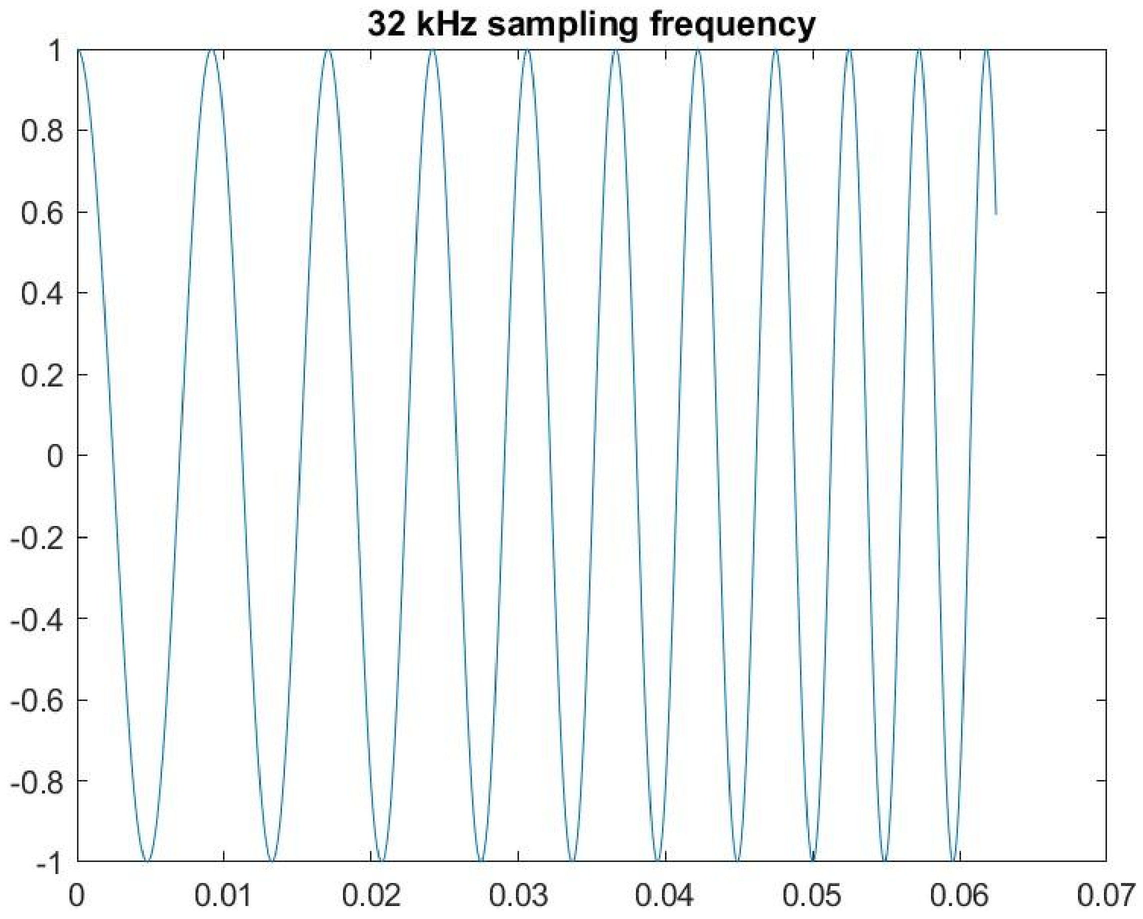

Figure 4: Reconstruction of the first 2000 samples of a frequency chirp signal. f_1 = 100 Hz, μ = 2000, f_s = 32 kHz.

2. Repeating the experiment with a sampling frequency of 16 kHz instead of 32 kHz, I hear the same increase in pitch over time as with the 32 kHz sampling rate, however, at 4 seconds, the pitch begins to decrease. The first 4 seconds with 16 kHz sampling rate sounds indistinguishable from 32 kHz sampling rate, with the same level of pitch and rate of increase. However, in the final 4 seconds, the pitch begins to decrease, seemingly with the same rate of decrease as the rate of increase, back to the sound which it started with. This also makes sense because at 4 seconds, the frequency of the input signal is 2000 * 4 + 100 = 8100 Hz, and it must be sampled with a frequency of 8100 * 2 = 16,200 Hz ≈ 16 kHz. Beyond 4 seconds, the frequency of the input signal exceeds the constraint of 8000 Hz to satisfy the Nyquist theorem, and we immediately notice aliasing in effect. Repeating the same experiment with a sampling frequency of 8 kHz, I hear an increase in pitch in the first 2 seconds, with the same rate of increase as that of the 16 kHz and 32 kHz signals, but then begins to decrease back to the original sound from 2 to 4 seconds. Between 4 and 8 seconds, the same thing happens again. pitch increases from 4 to 6 seconds and decreases from 6 to 8 seconds. This makes sense because at 2 seconds, the frequency of the input signal is 2000 * 2 + 100 = 4100 Hz. The Nyquist sampling rate for this signal is 4100 * 2 = 8200 Hz ≈ 8 kHz. An input beyond 4000 Hz begins to have aliasing because the sampling rate is insufficient for that frequency. What is interesting is that as the input frequency increases beyond 8 kHz (like from 4 to 8 seconds), it sounds

just as if the output frequency increases and peaks at 4 kHz and decreases again, like a triangle wave. Over a telephone connection, with sampling frequency of 8 kHz and an anti-aliasing filter present, this frequency chirp signal would sound like a pitch increase for 2 seconds, until the frequency of the signal is around 4 kHz, then the sound would stop. Experimenting with a lower f_s like 4 kHz, the triangle wave behavior is present, but this time goes through 4 cycles instead of 2 cycles for f_s = 8 kHz and 1 cycle for f_s = 16 kHz. Changing f_s back to 8 kHz and increasing μ to 4000, the output signal has the same triangle wave behavior as f_s = 4 kHz and μ = 2000 (same frequency, 4 cycles in 8 seconds). The difference is that the maximum frequency for f_s = 8 kHz and μ=4000 is 4 kHz, and the maximum frequency for f_s = 4 kHz and μ = 2000 is 2 kHz. This makes sense because doubling μ would half the time the input signal takes to reach the maximum frequency, doubling the number of cycles. The value of f_1 seems to give a phase shift to the triangle wave. Increasing f_1 makes the sound start at a higher pitch, but the rate of change remains the same and the triangle wave goes through the same number of cycles.

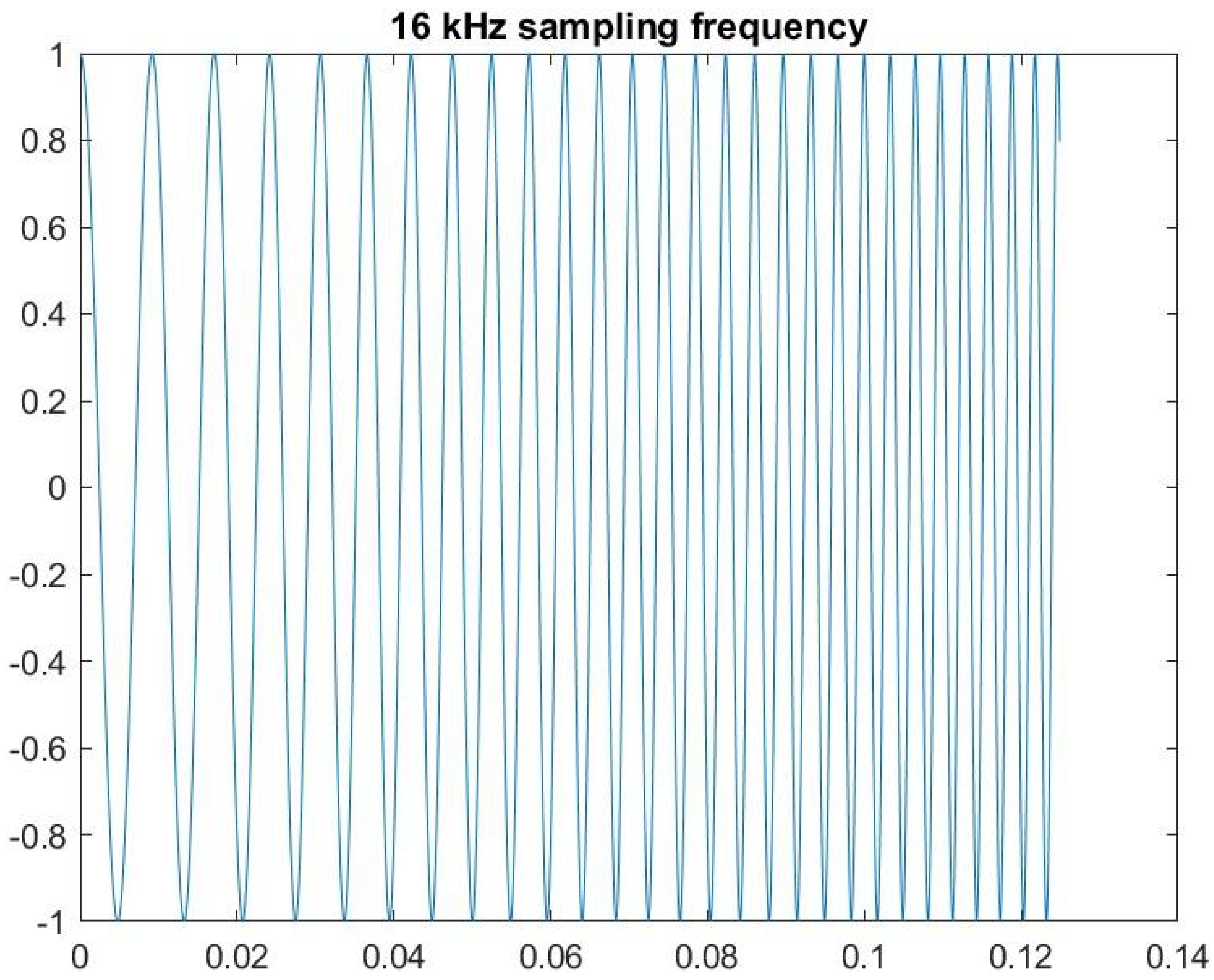

Figure 5: Reconstruction of the first 2000 samples of a frequency chirp signal. f_1 = 100 Hz, μ = 2000, f_s = 16 kHz.

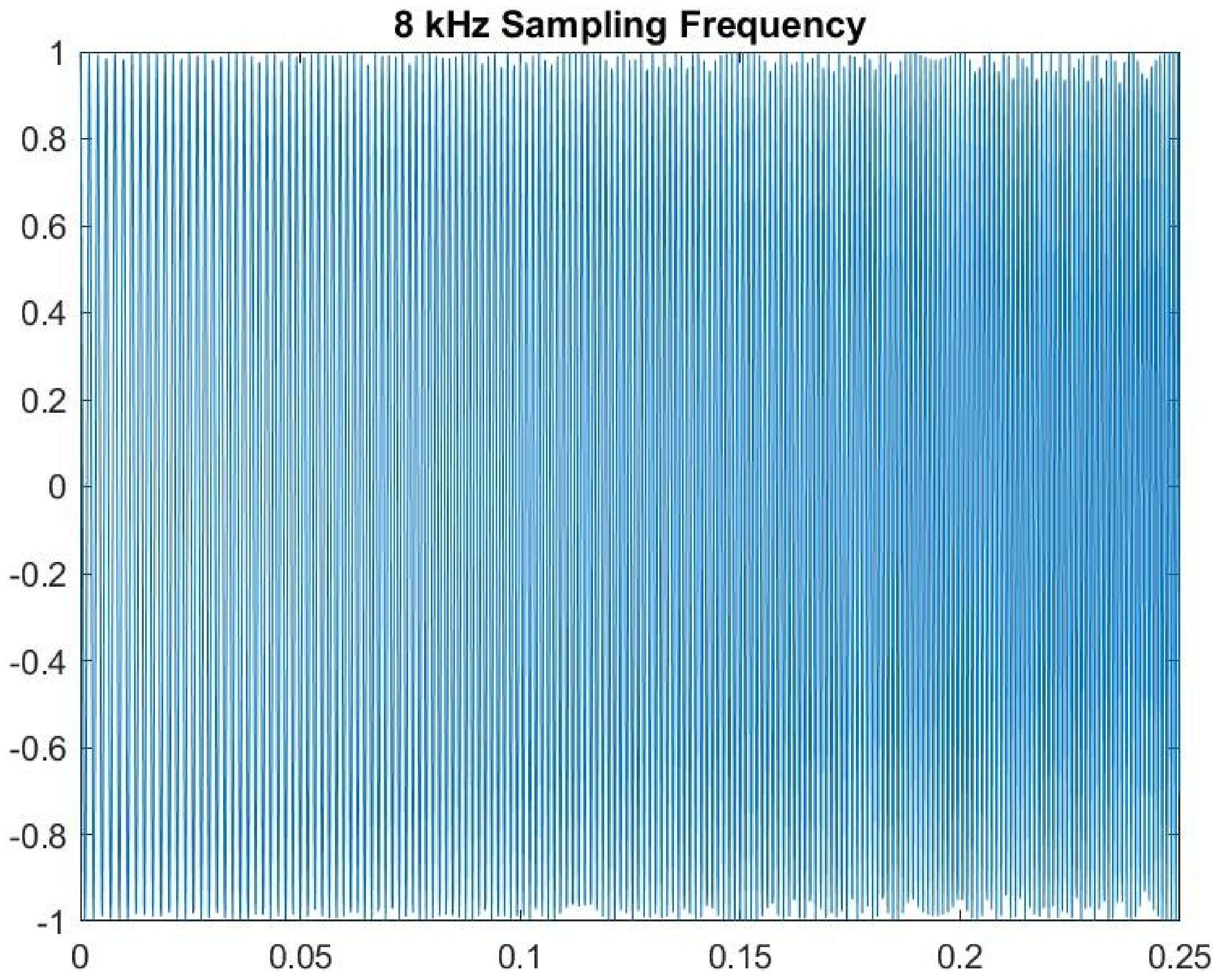

Figure 6: Reconstruction of the first 2000 samples of a frequency chirp signal. f_1 = 100 Hz, μ = 2000, f_s = 8 kHz.