ELECENG 3TP3 Signals and Systems

Lab 3: Aliasing in Signal Sampling

Section C01, Instructor: Prof. Jun Chen

Khaled Hassan

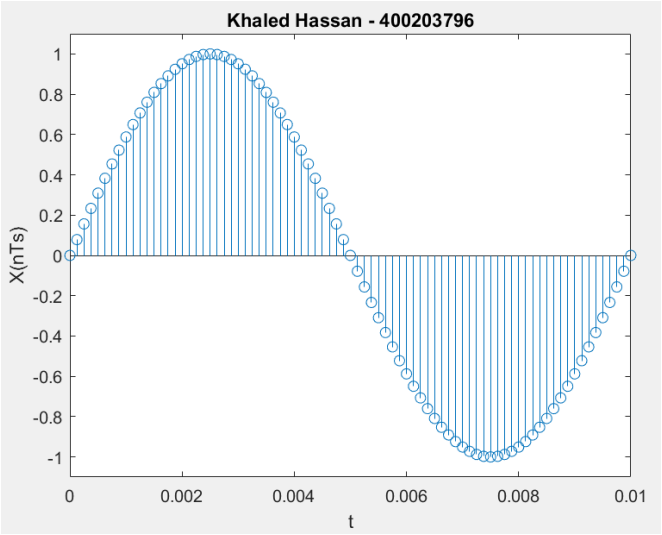
Hassak9

400203796

Aliasing in the Telephone System

Note: The code used is provided as screenshots in the body of the report (as it looks cleaner), and as text in the Appendix.

1.



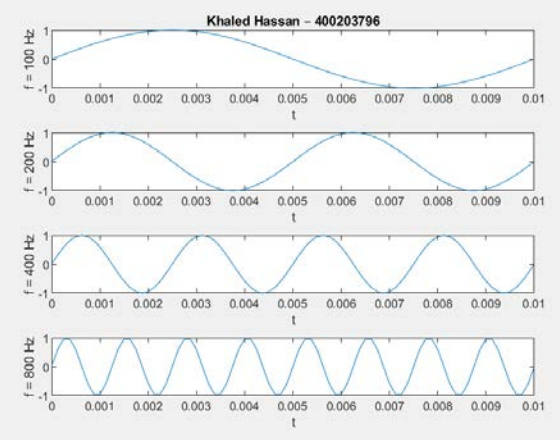

```
>> %plot of a sampled sinusoid with frequency f = 100 Hz
f = 100;
% Sampling frequency and interval
fs = 8000;
Ts = 1/fs;

% 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);

%%print -djpg graph1.jpg

title("Khaled Hassan - 400203796");
axis([0 0.01 -1.1 1.1]);
ylabel("X(nTs)");
xlabel("t");
```

Description:

The above code generates the figure on its left, which represents a discrete-time (DT) representation, derived from sampling the continuous-time (CT) function $x(t) = (2\pi ft + \phi)$, where the phase shift ϕ is set to 0, the amplitude is 1, the period is 10 and the frequency is 100 Hz. This function can be thought of as an audio signal being sent through a telephone system.

The samples were taken at a frequency of 8 KHz (Nyquist sampling rate for human voice), which results in a 0.125 ms interval between any 2 consecutive samples. This sampling frequency is high enough to yield a fairly accurate DT representation of the CT function, which is why this Nyquist rate is used in telecommunications. Using the MATLAB plot command instead of stem results in a straight-line, CT representation of the original function. Essentially, that would result in the original graph of the signal. Since the sampling frequency is high in the stem plot, we could also think of it as if we connected the consecutive samples with a straight line.

2.



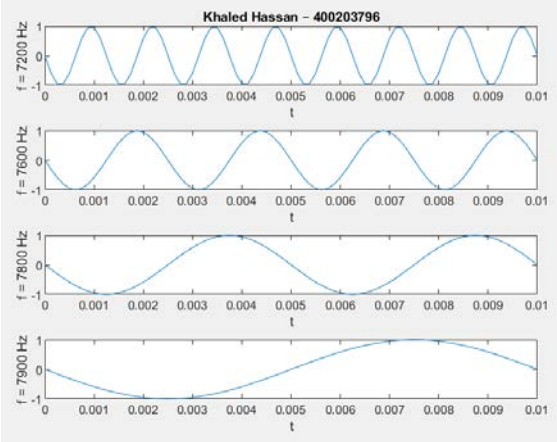

```
>> % Use sinusoid frequency f = 100, 200. 400. 800 Hz
f1= 100;
f2 = 200;
f4 = 400;
f8 = 800;

% 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 plots
nplot=0:Ts:tfinalplot;

% Make the time vector for replayed sound spurt
% Play the spurt for 2 seconds
tfinal = 2;
nsound=0:Ts:tfinal;

% Sample the sinusoids
xnT1 = sin(2*pi*f1*nsound);
xnT2 = sin(2*pi*f2*nsound);
xnT4 = sin(2*pi*f4*nsound);
xnT8 = sin(2*pi*f8*nsound);

% Make the plot
subplot(4, 1, 1);
plot(nplot, xnT1(1:length(nplot)));
xlabel('t');
ylabel('f = 100 Hz');
title('Khaled Hassan - 400203796');

subplot(4, 1, 2);
plot(nplot, xnT2(1:length(nplot)));
xlabel('t');
ylabel('f = 200 Hz');

subplot(4, 1, 3);
plot(nplot, xnT4(1:length(nplot)));
xlabel('t');
ylabel('f = 400 Hz');

subplot(4, 1, 4);
plot(nplot, xnT8(1:length(nplot)));
xlabel('t');
ylabel('f = 800 Hz');

xnT = [xnT1 xnT2 xnT4 xnT8];

% Save xnT as a wav sound file, Q2soundfile.wav.
audiowrite('Q2soundwavefile.wav', xnT, fs);
```

The code above was used to generate the 4 subplots above. Here, the same function as in question 1 was used, but with 4 different frequencies: 100 Hz, 200 Hz, 400 Hz and 800 Hz. The subplots show that as frequency increases, the period decreases proportionally resulting in more cycles occurring within the same time interval.

The audio file created from concatenating the 4 functions, in increasing frequency order was 8 seconds long, with each 2-second interval representing the function with a specific frequency. As such, the first 2 seconds represented the function with f = 100 Hz, which resulted in a low-pitch, deep beep-resembling sound. The next 2 seconds represented the function with f = 200 Hz, which was a higher-pitched sound, closer to a beep than before. The next 2 seconds represented the function with f = 400 Hz, an even higher-pitched, sound that distinctively

resembled a beep. During the last 2 seconds, the loudest-pitch beep was heard. As a result, we can conclude that the pitch of an audio signal is directly proportional to its frequency.

3.




```
>> % Use sinusoid frequency f = 100, 200. 400. 800 Hz
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 plots
nplot=0:Ts:tfinalplot;

% Make the time vector for replayed sound spurt
% Play the spurt for 2 seconds
tfinal = 2;
nsound=0:Ts:tfinal;

% Sample the sinusoids
xnT1 = sin(2*pi*f1*nsound);
xnT2 = sin(2*pi*f2*nsound);
xnT3 = sin(2*pi*f3*nsound);
xnT4 = sin(2*pi*f4*nsound);

% Make the plot
subplot(4, 1, 1);
plot(nplot, xnT1(1:length(nplot)));
xlabel('t');
ylabel('f = 7200 Hz');
title('Khaled Hassan - 400203796');

subplot(4, 1, 2);
plot(nplot, xnT2(1:length(nplot)));
xlabel('t');
ylabel('f = 7600 Hz');

subplot(4, 1, 3);
plot(nplot, xnT3(1:length(nplot)));
xlabel('t');
ylabel('f = 7800 Hz');

subplot(4, 1, 4);
plot(nplot, xnT4(1:length(nplot)));
xlabel('t');
ylabel('f = 7900 Hz');

xnT = [xnT1 xnT2 xnT3 xnT4];
% Save xnT as a wav sound file, Q3soundfile.wav.
audiowrite('Q3soundwavefile.wav', xnT, fs);
```

This code was used to generate the above subplot, and an audio file. As for the subplot, the frequencies used this time were f = 7200 Hz, 7600 Hz, 7800 Hz and 7900 Hz. The 4 subplots represent the same 4 subplots from question 2, except in decreasing frequencies: the 1st subplot represents x(t) (from q1) at f = 800 Hz, the 2nd one at f = 400 Hz, the 3rd at f = 200 Hz and the 4th one at f = 100 Hz. This is further supported by the audio file, which resembles the highest-pitched beep at the end of Q2's audio file first, followed by a lower pitched, lower frequency beep for the next 2 seconds, and so on. Therefore, Q3soundwavefile.wav represents the same functions as in Q2soundwavefile.wav, but concatenated in decreasing order of frequencies (100, 200, 400, 800 Hz) instead of increasing.

This occurs due to aliasing. For a signal of bandwidth B, the minimum sampling rate to avoid aliasing is $F_{smin} - B > B \Rightarrow F_{smin} > 2B$. E.g. for f = 7900 Hz and a sampling rate $F_s = 8\,KHz$:

F = 7900 Hz > (8KHz) / 2 => 7900 > 4000 Hz. (so aliasing occurs!)

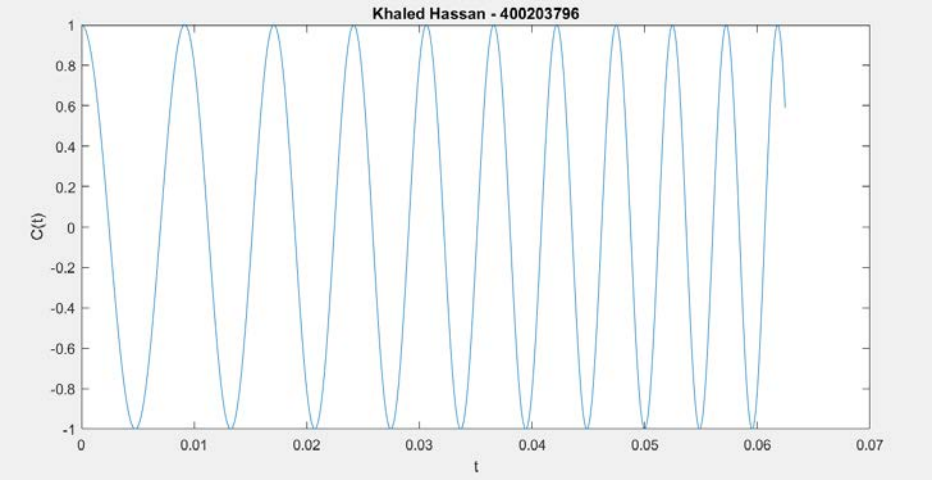
Aliasing will occur at $F_s - f = 8000 - 7900 = 100\,Hz$, which is the frequency of the equivalent graph and 2-second audio signal from Q2; the one with f = 100 Hz. Aliasing occurs at 800 Hz, 400 Hz and 200 Hz for frequencies 7200 Hz, 7600 Hz and 7800 Hz respectively.

4.

Anti-aliasing pre-filtering involves using a low-pass filter before ADC (Analog to Digital Conversion), to effectively truncate the portions of the signal that would cause aliasing if they were sampled. The performance of a phone system will be worse of there was no anti-aliasing, as aliasing causes distortions in the values carried by the signal, the actual audio being transmitted for the call to occur. If filtering was used in the above questions, only frequencies up till half of the sampling frequency would be sampled.

Aliasing of a Frequency Chirp Signal

1.



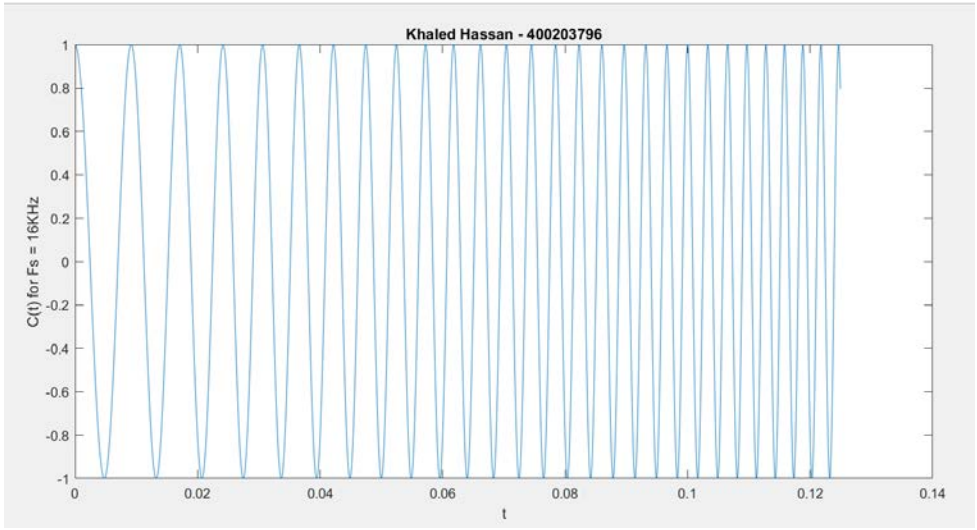

```
>> %plot of a sampled sinusoid with frequency f = 100
f = 100;
u = 2000;
% Sampling frequency and interval
fs = 32000;
Ts = 1/fs;
nSamples = 2000;
perT = 8;

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

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

% Sample the sinusoid.
cnT = cos(pi * u * nplot .^ 2 + 2 * pi * f * nplot);

% Make the plot
plot(nplot(1:2000), cnT(1:2000));

title("Khaled Hassan - 400203796");
ylabel("C(t)");
xlabel("t");

% Save xnT as a wav sound file, Q1Bsoundfile.wav.
audiowrite('Q1Bsoundwavefile.wav', cnT, fs);
```

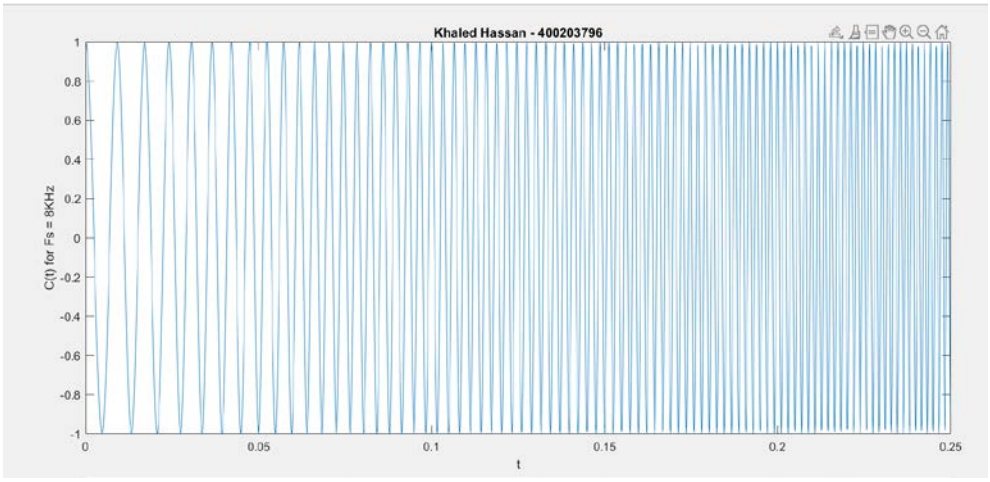
The code on the right was used to generate the plot on the left, as well as an audio file. As the graph demonstrates, the period of the wave generated decreases with time. Consequently, the frequency is increasing. The amplitude of the graph remains unaffected at 1. This is expected, as the given function is, $c(t) = \cos(\pi\mu t^2 + 2\pi f_1 t + \phi)$. Taking the 1st order

derivative of its phase with respect to time [d/dt($\pi\mu t^2 + 2\pi f_1 t + \phi$)], we can find its frequency, which turns out to be $f(t) = \mu t + f_1$. This function represents the frequency of the signal , f(t) with time. As the equation shows, it is a linear function (as it is in the form y = mx+b), and its initial frequency is f_1, following a slope of μ. In this question, f_1 was set at 100 Hz and μ was 2000, which indicates that the frequency increases from 100 Hz by 200 Hz every second.

The audio file produced was an 8 second representation of c(t). Initially, the frequency of the function was 100 Hz, and it increased by 2 KHz every second. This translated to a deep, low-pitched sound that quickly experiences a sharp increase in sharpness and pitch (due to the increase in frequency) as time progressed.

2.






```
>> %plot of a sampled sinusoid with frequency f = 100 Hz
f = 100;
u = 2000;
% Sampling frequency and interval
fs = 16000;
%% fs = 8000;
Ts = 1/fs;
nSamples = 2000;
perT = 8;

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

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

% Sample the sinusoid.
cnT = cos(pi * u * nplot .^ 2 + 2 * pi * f * nplot);

% Make the plot
plot(nplot(1:2000), cnT(1:2000));

title("Khaled Hassan - 400203796");
ylabel("C(t) for Fs = 16KHz");
%% ylabel("C(t) for Fs = 8KHz");

xlabel("t");

% Save xnT as a wav sound file, Q2Bsoundfile.wav.
audiowrite('Q2Bsoundwavefile.wav', cnT, fs);

% Save xnT as a wav sound file, Q2Csoundfile.wav.
%%audiowrite('Q2Csoundwavefile.wav', cnT, fs);
```

The code here was used to generate both plots (some parts were commented/uncommented). The upper graph represents the code with the sampling frequency F_S = 16 KHz, and the lower one is with F_S = 8 KHz. In both cases, μ was kept constant at 2000 and f_1 at 100 Hz. In both cases, aliasing occurs.

In the first graph, F_S = 16 KHz, and the time required to acquire the first 2000 samples was double that required in question 1 since the time interval between samples doubles. The audio file was similar to the one created in question 1; the sound starts off with a low pitch as frequency starts at 100 Hz. Then, the frequency of the signal gets progressively higher (increases due to a slope of 2KHz)., and as a result, so do the pitch and tone of the chirp. The main difference here is that at t = 4 seconds, the frequency peaks begins to reduce back to 100 Hz. As a result, the tone and pitch of the chirp also reduce. At the end of the audio file, we have the same pitch as in the beginning. This is due to aliasing; due to F_S being equal to 16 KHz, as the frequency of the audio signal increases, it passes the maximum bandwidth before which aliasing occurs. This bandwidth F_{Smin} = 16 / 2 KHz = 8 KHz. About halfway through the 8 second interval, when the signal frequency starts to reduce again. The frequency of the signal is equal to 8100 Hz, which has passed that threshold. As a result, the signal is being under sampled and aliasing occurs.

Similarly, for the second graph, F_S = 8 KHz, the time required to acquire the first 2000 samples is double that of the previous system, and quadruple that of question 1 of this section. The audio signal features an increase in frequency from f_1=100 Hz, followed by a decrease starting at around 2 seconds. In other words, it involves what happened when F_S = 16 KHz twice over half the time. Again, sampling also occurs since F_{Smin} = 8 / 2 KHz = 4 KHz, which is passed at around t = 2 s (at this point, f = 4100 Hz > 4 KHz => aliasing occurs). Without anti-aliasing pre-filtering circuitry, aliasing is highly likely to occur. If the telephone connection did perform anti-aliasing pre-filtering, aliasing would obviously be avoided. The average human voice can send sounds of frequencies up to 4 KHz. The Nyquist rate, which is 2 times that, is the standard sampling rate used in such telephone systems, would eliminate the chances of aliasing occurring as the frequency of the signal being sent through the system will probably never exceed half of the threshold.

Experimenting with other values was done by changing one at a time and keeping others constant. For different sampling frequencies, F_S, the maximum threshold before which sampling occurred is directly proportional; doubling F_S doubled the threshold, and vice versa. As μ, the rate of change of the frequency, changed, so too did the time at which aliasing occurred. They are inversely proportional, since a higher μ means that aliasing occurs quicker, at a lower time. Similarly, f_1 was also inversely proportional to the time at which aliasing occurred, but the effect was much slower due to f_1 being constant whereas μ is a multiplying factor.

Appendix A: Code Used

Aliasing in the Telephone System

1.

```
%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);
%% plot(nplot, xnT);

%%print -djpg graph1.jpg

title("Khaled Hassan - 400203796");
axis([0 0.01 -1.1 1.1]);
ylabel("X(nTs)");
xlabel("t");
```

2.

```
% Use sinusoid frequency f = 100, 200. 400. 800 Hz
f1= 100;
f2 = 200;
f4 = 400;
f8 = 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 plots
nplot=0:Ts:tfinalplot;

% Make the time vector for replayed sound spurt
% Play the spurt for 2 seconds
tfinal = 2;
nsound=0:Ts:tfinal;

% Sample the sinusoids
xnT1 = sin(2*pi*f1*nsound);
xnT2 = sin(2*pi*f2*nsound);
xnT4 = sin(2*pi*f4*nsound);
xnT8 = sin(2*pi*f8*nsound);

% Make the plot
subplot(4, 1, 1);
plot(nplot, xnT1(1:length(nplot)));
xlabel('t');
ylabel('f = 100 Hz');
title('Khaled Hassan – 400203796');

subplot(4, 1, 2);
plot(nplot, xnT2(1:length(nplot)));
xlabel('t');
ylabel('f = 200 Hz');

subplot(4, 1, 3);
plot(nplot, xnT4(1:length(nplot)));
xlabel('t');
ylabel('f = 400 Hz');

subplot(4, 1, 4);
plot(nplot, xnT8(1:length(nplot)));
```

```
xlabel('t');
ylabel('f = 800 Hz');

xnT = [xnT1 xnT2 xnT4 xnT8];

% Save xnT as a wav sound file, Q2soundfile.wav.
audiowrite('Q2soundwavefile.wav', xnT, fs);
' '
```

3.

```
% Use sinusoid frequency f = 7200, 7600. 7800. 7900 Hz
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 plots
nplot=0:Ts:tfinalplot;

% Make the time vector for replayed sound spurt
% Play the spurt for 2 seconds
tfinal = 2;
nsound=0:Ts:tfinal;

% Sample the sinusoids
xnT1 = sin(2*pi*f1*nsound);
xnT2 = sin(2*pi*f2*nsound);
xnT3 = sin(2*pi*f3*nsound);
xnT4 = sin(2*pi*f4*nsound);

% Make the plot
subplot(4, 1, 1);
```

```
plot(nplot, xnT1(1:length(nplot)));
xlabel('t');
ylabel('f = 7200 Hz');
title('Khaled Hassan – 400203796');

subplot(4, 1, 2);
plot(nplot, xnT2(1:length(nplot)));
xlabel('t');
ylabel('f = 7600 Hz');

subplot(4, 1, 3);
plot(nplot, xnT3(1:length(nplot)));
xlabel('t');
ylabel('f = 7800 Hz');

subplot(4, 1, 4);
plot(nplot, xnT4(1:length(nplot)));
xlabel('t');
ylabel('f = 7900 Hz');

xnT = [xnT1 xnT2 xnT3 xnT4];
% Save xnT as a wav sound file, Q3soundfile.wav.
audiowrite('Q3soundwavefile.wav', xnT, fs);
```

Aliasing of a Frequency Chirp Signal

1.

```
%plot of a sampled sinusoid with frequency f = 100 Hz
f = 100;
u = 2000;
% Sampling frequency and interval
fs = 32000;
Ts = 1/fs;
nSamples = 2000;
perT = 8;

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

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

% Sample the sinusoid.
cnT = cos(pi * u * nplot .^ 2 + 2 * pi * f * nplot);

% Make the plot
plot(nplot(1:2000), cnT(1:2000));

title("Khaled Hassan - 400203796");
ylabel("C(t)");
xlabel("t");

% Save xnT as a wav sound file, Q1Bsoundfile.wav.
audiowrite('Q1Bsoundwavefile.wav', cnT, fs);
```

2.

```
%plot of a sampled sinusoid with frequency f = 100 Hz
f = 100;
u = 2000;
% Sampling frequency and interval
%% fs = 16000;
fs = 8000;
Ts = 1/fs;
nSamples = 2000;
perT = 8;

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

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

% Sample the sinusoid.
cnT = cos(pi * u * nplot .^ 2 + 2 * pi * f * nplot);

% Make the plot
plot(nplot(1:2000), cnT(1:2000));
```

```
title("Khaled Hassan - 400203796");
%% ylabel("C(t) for Fs = 16KHz");
ylabel("C(t) for Fs = 8KHz");

xlabel("t");

%% Save xnT as a wav sound file, Q2Bsoundfile.wav.
%% audiowrite('Q2Bsoundwavefile.wav', cnT, fs);

% Save xnT as a wav sound file, Q2Csoundfile.wav.
audiowrite('Q2Csoundwavefile.wav', cnT, fs);
```