*Robert Valencia*

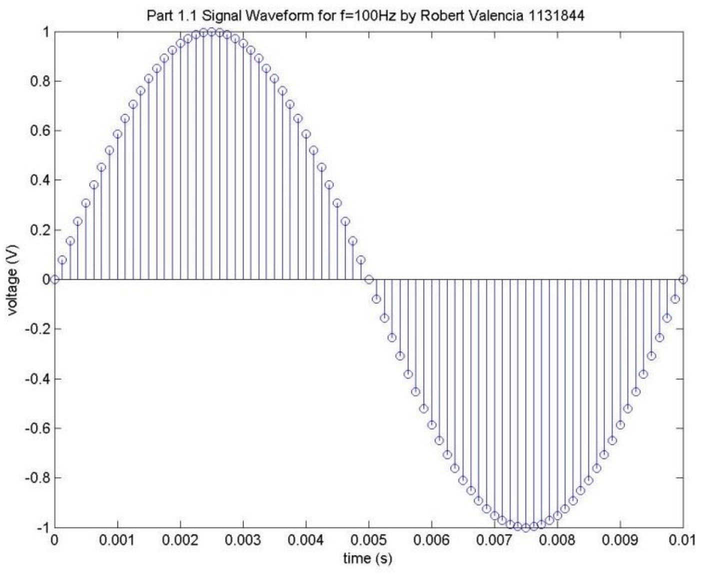
# EE 3TP4 Lab 3

**Part 1.1**

<u>MATLAB code</u>

```
%Part 1.1
%sinusoid frequency
f=100;
%sampling frequency and interval
fs=8000;
Ts=1/fs;
%final time value/duration
tEnd=10e-3;
%time vector
tVector=0:Ts:tEnd;
%sinusoid sample
xs=sin(2.*pi.*f.*tVector);
%plot
figure(1);
subplot(1,1,1);
stem(tVector, xs);
title('Part 1.1 Signal Waveform for f=100Hz by Robert Valencia’)
xlabel('time (s)');
ylabel('voltage (V)');
%print plot
print ('-djpeg', 'plot_1_1.jpeg');
```

<u>Output</u>



**1.1** From the plot shown above, it can be seen that the signal is more than adequately sampled for it to retain the correct signal frequency and shape representation. This is due to the fact that the sampling frequency is much higher than 5-10 times the maximum frequency in the signal.

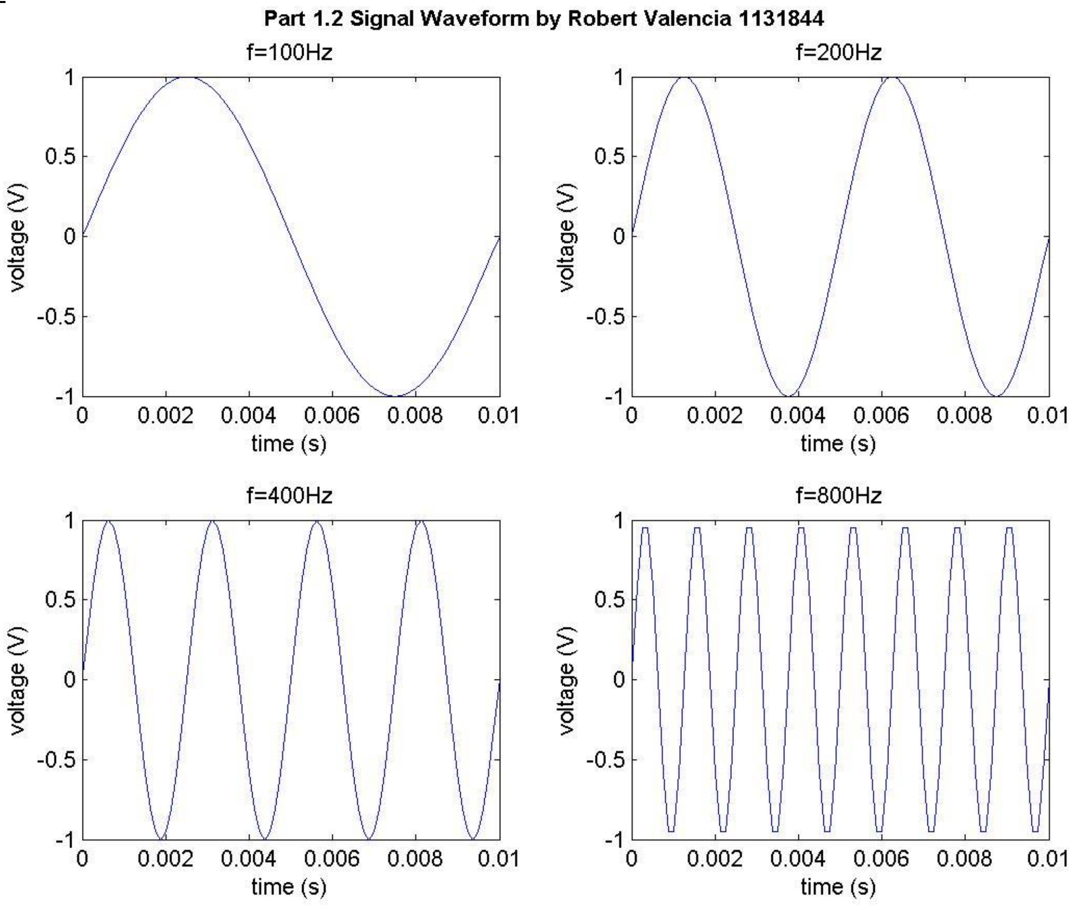
**Part 1.2**

<u>MATLAB code</u>

```
%Part 1.2
%sinusoid frequency
f=100;
%sampling frequency and interval
fs=8000;
Ts=1/fs;
%sampled signal final time value/duration
tEnd=10e-3;
%sampled signal time vector
tVector=0:Ts:tEnd;
%sound spurt final time value/duration
tEndSSpurt=2;
%sound spurt time vector
tVectorSSpurt=0:Ts:tEndSSpurt;
%sinusoid sample
xs100=sin(2.*pi.*f.*tVectorSSpurt);
%plot
figure(2);
subplot(2,2,1);
plot(tVector, xs100(1:length(tVector)));
title('f=100Hz')
xlabel('time (s)');
ylabel('voltage (V)');
%save as a wav sound file
wavwrite(xs100', fs, 8, 'sound_1_2_100.wav');

%sinusoid frequency
f=200;
%sampling frequency and interval
fs=8000;
Ts=1/fs;
%sampled signal final time value/duration
tEnd=10e-3;
%sampled signal time vector
tVector=0:Ts:tEnd;
%sound spurt final time value/duration
tEndSSpurt=2;
%sound spurt time vector
tVectorSSpurt=0:Ts:tEndSSpurt;
%sinusoid sample
xs200=sin(2.*pi.*f.*tVectorSSpurt);
%plot
subplot(2,2,2);
plot(tVector, xs200(1:length(tVector)));
title('f=200Hz')
xlabel('time (s)');
ylabel('voltage (V)');
%save as a wav sound file
wavwrite(xs200', fs, 8, 'sound_1_2_200.wav');

%sinusoid frequency
f=400;
%sampling frequency and interval
fs=8000;
Ts=1/fs;
```

```
%sampled signal final time value/duration
tEnd=10e-3;
%sampled signal time vector
tVector=0:Ts:tEnd;
%sound spurt final time value/duration
tEndSSpurt=2;
%sound spurt time vector
tVectorSSpurt=0:Ts:tEndSSpurt;
%sinusoid sample
xs400=sin(2.*pi.*f.*tVectorSSpurt);
%plot
subplot(2,2,3);
plot(tVector, xs400(1:length(tVector)));
title('f=400Hz')
xlabel('time (s)');
ylabel('voltage (V)');
%save as a wav sound file
wavwrite(xs400', fs, 8, 'sound_1_2_400.wav');

%sinusoid frequency
f=800;
%sampling frequency and interval
fs=8000;
Ts=1/fs;
%sampled signal final time value/duration
tEnd=10e-3;
%sampled signal time vector
tVector=0:Ts:tEnd;
%sound spurt final time value/duration
tEndSSpurt=2;
%sound spurt time vector
tVectorSSpurt=0:Ts:tEndSSpurt;
%sinusoid sample
xs800=sin(2.*pi.*f.*tVectorSSpurt);
%plot
subplot(2,2,4);
plot(tVector, xs800(1:length(tVector)));
title('f=800Hz')
xlabel('time (s)');
ylabel('voltage (V)');
%save as a wav sound file
wavwrite(xs800', fs, 8, 'sound_1_2_800.wav');

%main plot title
ha = axes('Position',[0 0 1 1],'Xlim',[0 1],'Ylim',[0
1],'Box','off','Visible','off','Units','normalized', 'clipping' , 'off');
text(0.5, 1,'\bf Part 1.2 Signal Waveform by Robert
Valencia','HorizontalAlignment','center','VerticalAlignment', 'top');
%print plots
print -djpeg plot_1_2.jpeg;

%concatenate the four 2-second tone segments
xsConcantenated=xs100+xs200+xs400+xs800;
%save as a wav sound file
wavwrite(xsConcantenated', fs, 8, 'sound_1_2_Concatenated.wav');
```

Output



**1.2.** From the plots shown above, it can be seen that the signals are more than adequately sampled for them to retain the correct signal frequency and shape representations. This is due to the fact that the sampling frequency is higher than 5-10 times the maximum frequencies of each the signal.

The individual sound files created have increasing pitches that corresponds to the increasing signal frequencies, which is what is expected. When the signals are concatenated, the sound produced contains a combination of all the pitches created by all of the component signals, each with their own unique frequencies.

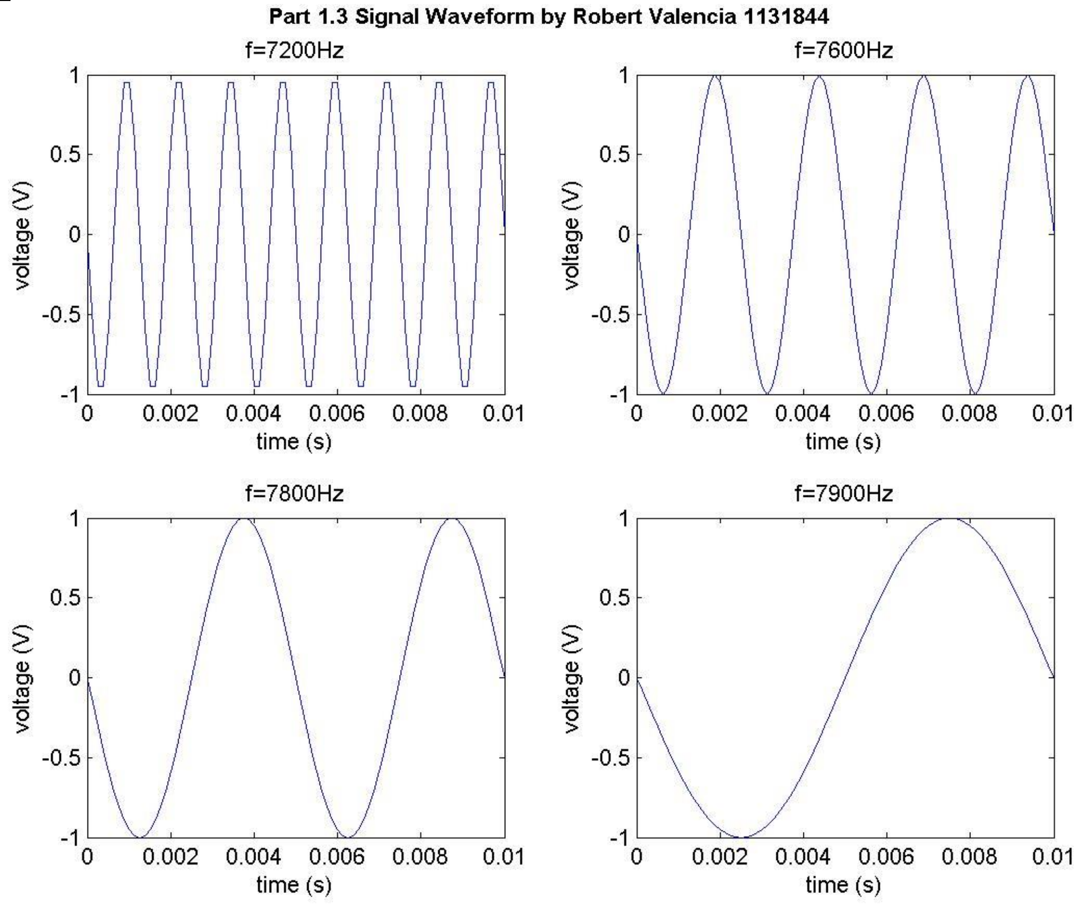
**Part 1.3**

MATLAB code

```
%Part 1.3
%sinusoid frequency
f=7200;
%sampling frequency and interval
fs=8000;
Ts=1/fs;
%sampled signal final time value/duration
tEnd=10e-3;
%sampled signal time vector
tVector=0:Ts:tEnd;
%sound spurt final time value/duration
tEndSSpurt=2;
%sound spurt time vector
tVectorSSpurt=0:Ts:tEndSSpurt;
%sinusoid sample
xs7200=sin(2.*pi.*f.*tVectorSSpurt);
%plot
figure(3);
subplot(2,2,1);
plot(tVector, xs7200(1:length(tVector)));
title('f=7200Hz')
xlabel('time (s)');
ylabel('voltage (V)');
%save as a wav sound file
wavwrite(xs7200', fs, 8, 'sound_1_3_7200.wav');

%sinusoid frequency
f=7600;
%sampling frequency and interval
fs=8000;
Ts=1/fs;
%sampled signal final time value/duration
tEnd=10e-3;
%sampled signal time vector
tVector=0:Ts:tEnd;
%sound spurt final time value/duration
tEndSSpurt=2;
%sound spurt time vector
tVectorSSpurt=0:Ts:tEndSSpurt;
%sinusoid sample
xs7600=sin(2.*pi.*f.*tVectorSSpurt);
%plot
subplot(2,2,2);
plot(tVector, xs7600(1:length(tVector)));
title('f=7600Hz')
xlabel('time (s)');
ylabel('voltage (V)');
%save as a wav sound file
wavwrite(xs7600', fs, 8, 'sound_1_3_7600.wav');

%sinusoid frequency
f=7800;
%sampling frequency and interval
fs=8000;
```

```
Ts=1/fs;
%sampled signal final time value/duration
tEnd=10e-3;
%sampled signal time vector
tVector=0:Ts:tEnd;
%sound spurt final time value/duration
tEndSSpurt=2;
%sound spurt time vector
tVectorSSpurt=0:Ts:tEndSSpurt;
%sinusoid sample
xs7800=sin(2.*pi.*f.*tVectorSSpurt);
%plot
subplot(2,2,3);
plot(tVector, xs7800(1:length(tVector)));
title('f=7800Hz')
xlabel('time (s)');
ylabel('voltage (V)');
%save as a wav sound file
wavwrite(xs7800', fs, 8, 'sound_1_3_7800.wav');

%sinusoid frequency
f=7900;
%sampling frequency and interval
fs=8000;
Ts=1/fs;
%sampled signal final time value/duration
tEnd=10e-3;
%sampled signal time vector
tVector=0:Ts:tEnd;
%sound spurt final time value/duration
tEndSSpurt=2;
%sound spurt time vector
tVectorSSpurt=0:Ts:tEndSSpurt;
%sinusoid sample
xs7900=sin(2.*pi.*f.*tVectorSSpurt);
%plot
subplot(2,2,4);
plot(tVector, xs7900(1:length(tVector)));
title('f=7900Hz')
xlabel('time (s)');
ylabel('voltage (V)');
%save as a wav sound file
wavwrite(xs7900', fs, 8, 'sound_1_3_7900.wav');

%main plot title
ha = axes('Position',[0 0 1 1],'Xlim',[0 1],'Ylim',[0
1],'Box','off','Visible','off','Units','normalized', 'clipping' , 'off');
text(0.5, 1,'\bf Part 1.3 Signal Waveform by Robert
Valencia','HorizontalAlignment','center','VerticalAlignment', 'top');
%print plots
print -djpeg plot_1_3.jpeg;

%concatenate the four 2-second tone segments
xsConcantenated=xs7200+xs7600+xs7800+xs7900;
%save as a wav sound file
wavwrite(xsConcantenated', fs, 8, 'sound_1_3_Concatenated.wav');
```

Output



**1.3** From the plots shown above, it can be seen that the signals are severely inadequately sampled, causing them to have incorrect signal frequency and shape representations, a phenomenon known as aliasing. This is due to the fact that the sampling frequency is much lower than the maximum frequencies of each the signal.

The individual sound files created have decreasing pitches that corresponds to the increasing signal frequencies, which is not what is expected, due to aliasing. When the signals are concatenated, the sound produced contains a combination of all the pitches created by all of the component signals, each with their own unique frequencies. Also, due to aliasing, the concatenated sound created from the higher frequency signals also sounds the same as the concatenated sound created from the lower frequency signals from section **1.2.**

**1.4.** Without anti-aliasing pre-filtering, signal representation for both frequency and waveform shape will be incorrect, causing some high frequency signals to sound like low frequency signals and vice versa, reducing voice conversation quality. Aside from that, pre-filtering also eliminates/reduces high frequency external noise, so, without it, external noise coming from non-human sources will be prominent in the signal, diluting the focus on the human speech, which, again, decreases the conversation quality. With all the proper pre-filtering, high frequency noises will be eliminated from the signal, focusing the transmitted signal on frequencies generated by human voices. Also, with proper anti-aliasing filters and adequate sampling,  signals of human voices will be properly represented for both frequency and waveform shape, enabling a high-quality voice conversation.

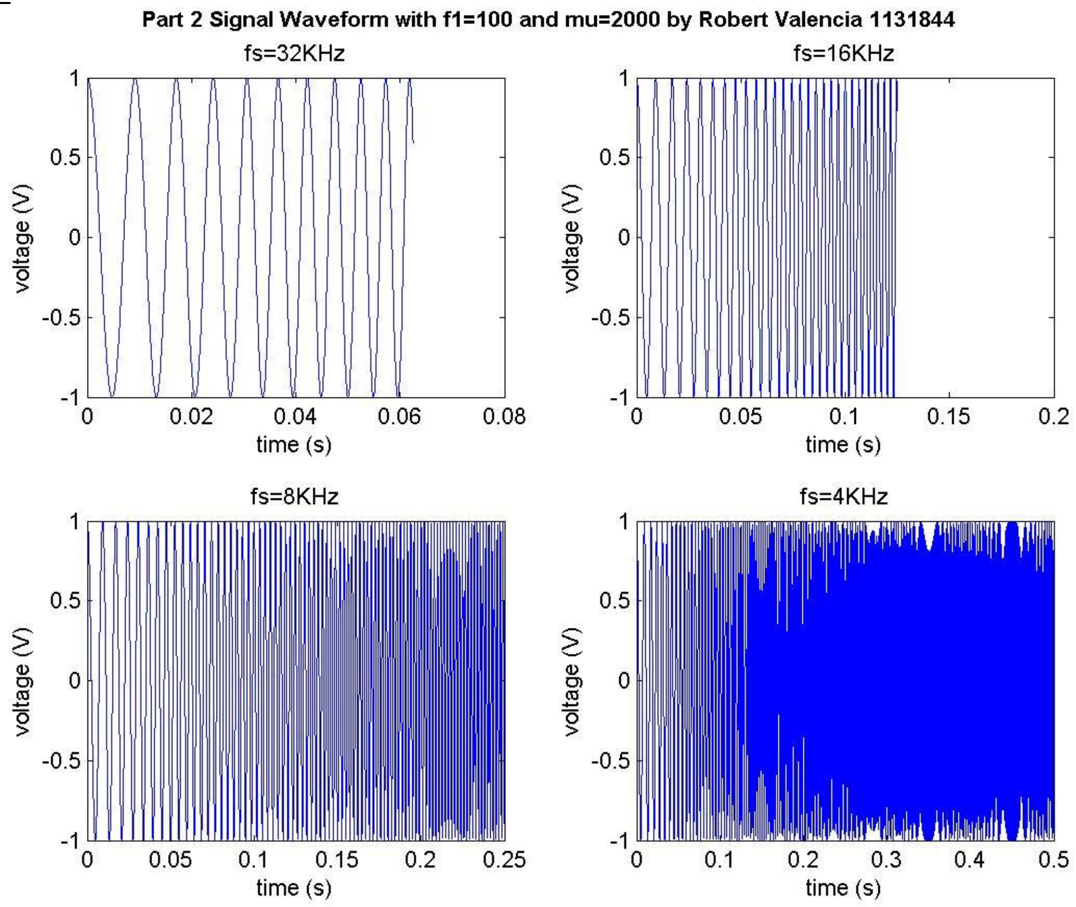
**Part 2a**

<u>MATLAB code</u>

```
%Part 2a with f1=100 and mu=2000
%initial sinusoid frequency
f1=100;
%frequency increase rate
mu=2000;
%sampling frequency and interval
fs=32000;
Ts=1/fs;
%sample final time value/duration
tEnd=8;
%sample time vector
tVector=0:Ts:tEnd;
%sinusoid sample
csa32K=cos(pi.*mu.*tVector.*tVector+2.*pi.*f1.*tVector);
%plot
figure(3);
subplot(2,2,1);
plot(tVector(1:2000), csa32K(1:2000));
title('fs=32KHz')
xlabel('time (s)');
ylabel('voltage (V)');
%save as a wav sound file
wavwrite(csa32K', fs, 8, 'sound_2a_32K.wav');

%initial sinusoid frequency
f1=100;
%frequency increase rate
mu=2000;
%sampling frequency and interval
fs=16000;
Ts=1/fs;
%sample final time value/duration
tEnd=8;
%sample time vector
tVector=0:Ts:tEnd;
%sinusoid sample
csa16K=cos(pi.*mu.*tVector.*tVector+2.*pi.*f1.*tVector);
%plot
subplot(2,2,2);
plot(tVector(1:2000), csa16K(1:2000));
title('fs=16KHz')
xlabel('time (s)');
ylabel('voltage (V)');
%save as a wav sound file
wavwrite(csa16K', fs, 8, 'sound_2a_16K.wav');

%initial sinusoid frequency
f1=100;
%frequency increase rate
mu=2000;
%sampling frequency and interval
fs=8000;
Ts=1/fs;
%sample final time value/duration
```

```
tEnd=8;
%sample time vector
tVector=0:Ts:tEnd;
%sinusoid sample
csa8K=cos(pi.*mu.*tVector.*tVector+2.*pi.*f1.*tVector);
%plot
subplot(2,2,3);
plot(tVector(1:2000), csa8K(1:2000));
title('fs=8KHz')
xlabel('time (s)');
ylabel('voltage (V)');
%save as a wav sound file
wavwrite(csa8K', fs, 8, 'sound_2a_8K.wav');

%initial sinusoid frequency
f1=100;
%frequency increase rate
mu=2000;
%sampling frequency and interval
fs=4000;
Ts=1/fs;
%sample final time value/duration
tEnd=8;
%sample time vector
tVector=0:Ts:tEnd;
%sinusoid sample
csa4K=cos(pi.*mu.*tVector.*tVector+2.*pi.*f1.*tVector);
%plot
subplot(2,2,4);
plot(tVector(1:2000), csa4K(1:2000));
title('fs=4KHz')
xlabel('time (s)');
ylabel('voltage (V)');
%save as a wav sound file
wavwrite(csa4K', fs, 8, 'sound_2a_4K.wav');

%main plot title
ha = axes('Position',[0 0 1 1],'Xlim',[0 1],'Ylim',[0
1],'Box','off','Visible','off','Units','normalized', 'clipping' , 'off');
text(0.5, 1,'\bf Part 2 Signal Waveform with f1=100 and mu=2000 by Robert
Valencia 1131844 ','HorizontalAlignment','center','VerticalAlignment',
'top');
%print plots
print -djpeg plot_2a.jpeg;
```

Output


Part 2 Signal Waveform with f1=100 and mu=2000 by Robert Valencia 1131844
fs=32KHz
fs=16KHz
fs=8KHz
fs=4KHz
voltage (V)
time (s)

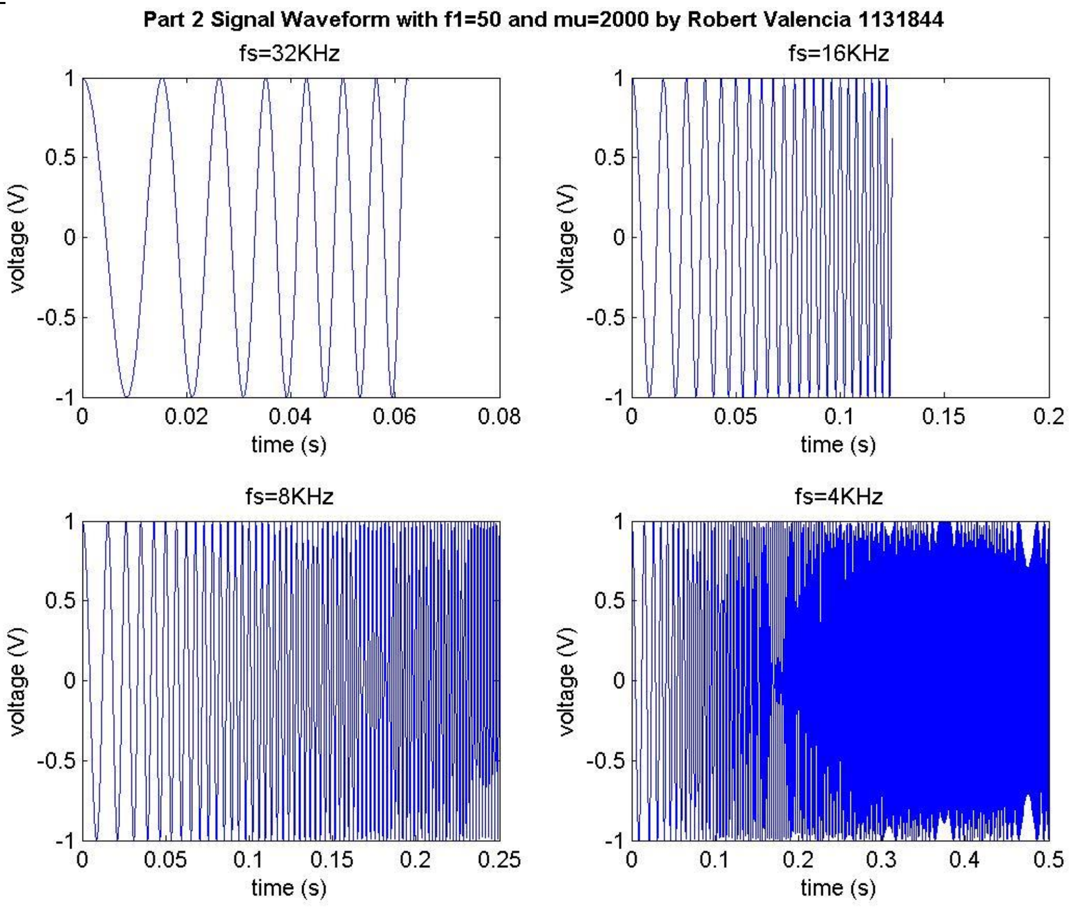
**Part 2b**

MATLAB code

```
%Part 2b with f1=50 and mu=2000
%initial sinusoid frequency
f1=50;
%frequency increase rate
mu=2000;
%sampling frequency and interval
fs=32000;
Ts=1/fs;
%sample final time value/duration
tEnd=8;
%sample time vector
tVector=0:Ts:tEnd;
%sinusoid sample
csb32K=cos(pi.*mu.*tVector.*tVector+2.*pi.*f1.*tVector);
%plot
figure(4);
subplot(2,2,1);
plot(tVector(1:2000), csb32K(1:2000));
title('fs=32KHz')
xlabel('time (s)');
ylabel('voltage (V)');
%save as a wav sound file
wavwrite(csb32K', fs, 8, 'sound_2b_32K.wav');

%initial sinusoid frequency
f1=50;
%frequency increase rate
mu=2000;
%sampling frequency and interval
fs=16000;
Ts=1/fs;
%sample final time value/duration
tEnd=8;
%sample time vector
tVector=0:Ts:tEnd;
%sinusoid sample
csb16K=cos(pi.*mu.*tVector.*tVector+2.*pi.*f1.*tVector);
%plot
subplot(2,2,2);
plot(tVector(1:2000), csb16K(1:2000));
title('fs=16KHz')
xlabel('time (s)');
ylabel('voltage (V)');
%save as a wav sound file
wavwrite(csb16K', fs, 8, 'sound_2b_16K.wav');

%initial sinusoid frequency
f1=50;
%frequency increase rate
mu=2000;
%sampling frequency and interval
fs=8000;
Ts=1/fs;
%sample final time value/duration
```

```
tEnd=8;
%sample time vector
tVector=0:Ts:tEnd;
%sinusoid sample
csb8K=cos(pi.*mu.*tVector.*tVector+2.*pi.*f1.*tVector);
%plot
subplot(2,2,3);
plot(tVector(1:2000), csb8K(1:2000));
title('fs=8KHz')
xlabel('time (s)');
ylabel('voltage (V)');
%save as a wav sound file
wavwrite(csb8K', fs, 8, 'sound_2b_8K.wav');

%initial sinusoid frequency
f1=50;
%frequency increase rate
mu=2000;
%sampling frequency and interval
fs=4000;
Ts=1/fs;
%sample final time value/duration
tEnd=8;
%sample time vector
tVector=0:Ts:tEnd;
%sinusoid sample
csb4K=cos(pi.*mu.*tVector.*tVector+2.*pi.*f1.*tVector);
%plot
subplot(2,2,4);
plot(tVector(1:2000), csb4K(1:2000));
title('fs=4KHz')
xlabel('time (s)');
ylabel('voltage (V)');
%save as a wav sound file
wavwrite(csb4K', fs, 8, 'sound_2b_4K.wav');

%main plot title
ha = axes('Position',[0 0 1 1],'Xlim',[0 1],'Ylim',[0
1],'Box','off','Visible','off','Units','normalized', 'clipping' , 'off');
text(0.5, 1,'\bf Part 2 Signal Waveform with f1=50 and mu=2000 by Robert
Valencia 1131844 ','HorizontalAlignment','center','VerticalAlignment',
'top');
%print plots
print -djpeg plot_2b.jpeg;
```

Output


Part 2 Signal Waveform with f1=50 and mu=2000 by Robert Valencia 1131844
fs=32KHz
fs=16KHz
fs=8KHz
fs=4KHz
voltage (V)
time (s)

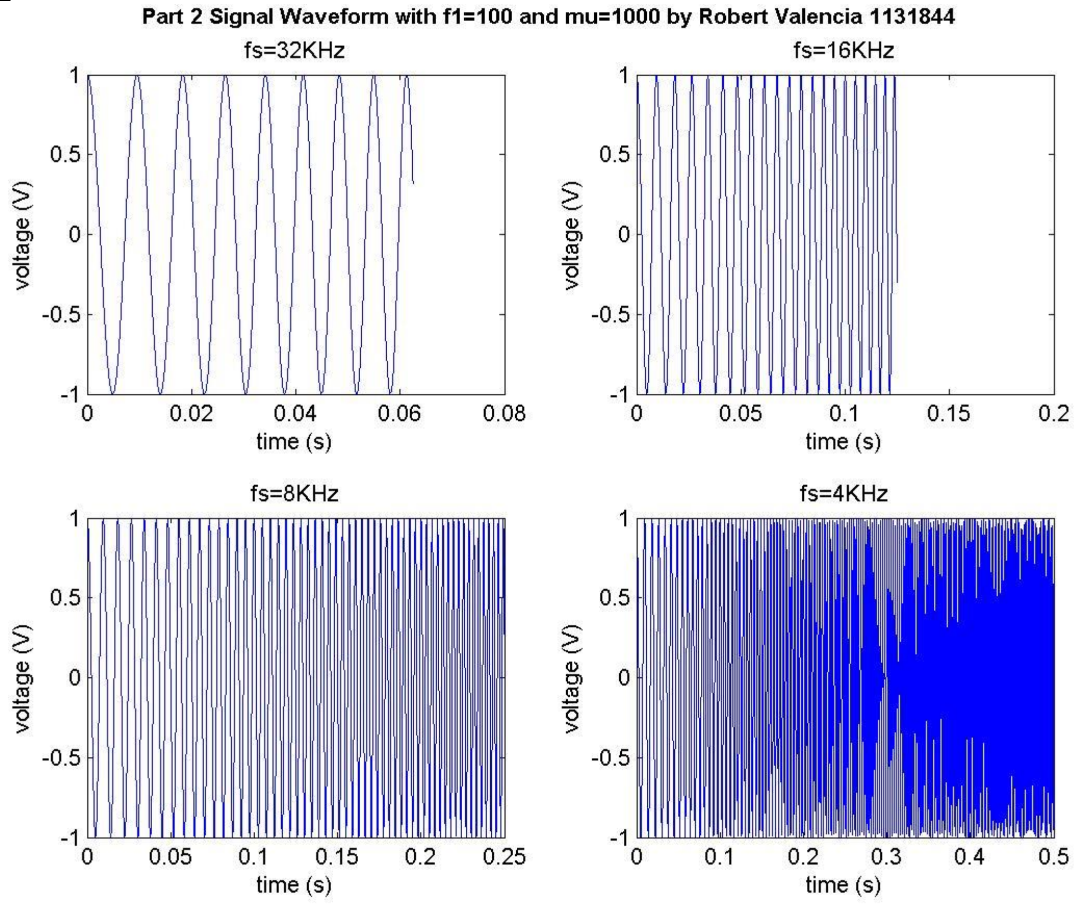
**Part 2c**

<u>MATLAB code</u>

```
%Part 2c with f1=100 and mu=1000
%initial sinusoid frequency
f1=100;
%frequency increase rate
mu=1000;
%sampling frequency and interval
fs=32000;
Ts=1/fs;
%sample final time value/duration
tEnd=8;
%sample time vector
tVector=0:Ts:tEnd;
%sinusoid sample
csc32K=cos(pi.*mu.*tVector.*tVector+2.*pi.*f1.*tVector);
%plot
figure(5);
subplot(2,2,1);
plot(tVector(1:2000), csc32K(1:2000));
title('fs=32KHz')
xlabel('time (s)');
ylabel('voltage (V)');
%save as a wav sound file
wavwrite(csc32K', fs, 8, 'sound_2c_32K.wav');

%initial sinusoid frequency
f1=100;
%frequency increase rate
mu=1000;
%sampling frequency and interval
fs=16000;
Ts=1/fs;
%sample final time value/duration
tEnd=8;
%sample time vector
tVector=0:Ts:tEnd;
%sinusoid sample
csc16K=cos(pi.*mu.*tVector.*tVector+2.*pi.*f1.*tVector);
%plot
subplot(2,2,2);
plot(tVector(1:2000), csc16K(1:2000));
title('fs=16KHz')
xlabel('time (s)');
ylabel('voltage (V)');
%save as a wav sound file
wavwrite(csc16K', fs, 8, 'sound_2c_16K.wav');

%initial sinusoid frequency
f1=100;
%frequency increase rate
mu=1000;
%sampling frequency and interval
fs=8000;
Ts=1/fs;
%sample final time value/duration
```

```
tEnd=8;
%sample time vector
tVector=0:Ts:tEnd;
%sinusoid sample
csc8K=cos(pi.*mu.*tVector.*tVector+2.*pi.*f1.*tVector);
%plot
subplot(2,2,3);
plot(tVector(1:2000), csc8K(1:2000));
title('fs=8KHz')
xlabel('time (s)');
ylabel('voltage (V)');
%save as a wav sound file
wavwrite(csc8K', fs, 8, 'sound_2c_8K.wav');

%initial sinusoid frequency
f1=100;
%frequency increase rate
mu=1000;
%sampling frequency and interval
fs=4000;
Ts=1/fs;
%sample final time value/duration
tEnd=8;
%sample time vector
tVector=0:Ts:tEnd;
%sinusoid sample
csc4K=cos(pi.*mu.*tVector.*tVector+2.*pi.*f1.*tVector);
%plot
subplot(2,2,4);
plot(tVector(1:2000), csc4K(1:2000));
title('fs=4KHz')
xlabel('time (s)');
ylabel('voltage (V)');
%save as a wav sound file
wavwrite(csc4K', fs, 8, 'sound_2c_4K.wav');

%main plot title
ha = axes('Position',[0 0 1 1],'Xlim',[0 1],'Ylim',[0
1],'Box','off','Visible','off','Units','normalized', 'clipping' , 'off');
text(0.5, 1,'\bf Part 2 Signal Waveform with f1=100 and mu=1000 by Robert
Valencia 1131844 ','HorizontalAlignment','center','VerticalAlignment',
'top');
%print plots
print -djpeg plot_2c.jpeg;
```

Output



**2.1 and 2.2.** Based on the sound files and the plots, it can be observed that the higher the sampling frequency, the more frequency components can be captured and the better the signal is represented due to a better sampling rate that enables more accurate waveform frequency and shape representations. When the sampling frequency is too low for the signal, aliasing will occur, causing the signal to be misrepresented due to insufficient sampling. This can be seen by altering the sampling frequencies, as shown in the previous 3 sets of 4 plots. Regardless of what mu or f1 value is chosen, the same principles apply, albeit with different sampling frequencies. When all the proper filtering is in place, a well-represented signal will be transmitted, creating a high-quality voice to be clearly heard while other noise sources are minimized.