Concept of digitization and sampling rate for cosine and voice signals

Nowadays we listen to music mostly from computers or handheld electronic devices which save audio in digital format (ones and zeros). But our voice or the music in its natural form cannot be saved in computer or digital media like pen drive, CD ROM etc. So how do we save voice or music in the format of zeros and ones? What is the process involved in converting voice into a series of ones and zeros? At the end of this experiment you'll understand this.

Aims of the experiment:

1. Discrete time representation of signals.
2. Sampling rate selection.
3. Effect of under sampling (aliasing).
4. Discretization of amplitude of signal (quantization).
5. Impact of bits per sample on signal shape.

Apparatus:

1. A PC installed with Scilab 5.5.0 or above.
2. A PC headset (containing headphone and microphone, 3.5 mm jack connector) – for voice sampling and quantization.

Theory:

1. Sampling:

This is the first step in converting the audio signal into digital signal. When we say sampling in signal processing, we mean that we are converting a continuous time signal into a discrete time signal. For example, in Fig. 1, the signal shown in green color is a continuous time signal i.e. the signal is available without break in time domain. But the same signal when sampled, as shown in red color in Fig. 1, has values only at certain points in time i.e. discrete in time. These discrete points are called sample points and we have to choose the points appropriately so that the discrete points when joined together form the original signal reliably. The frequency with which these samples are taken from time domain is called sampling frequency denoted as f_s (samples/sec).

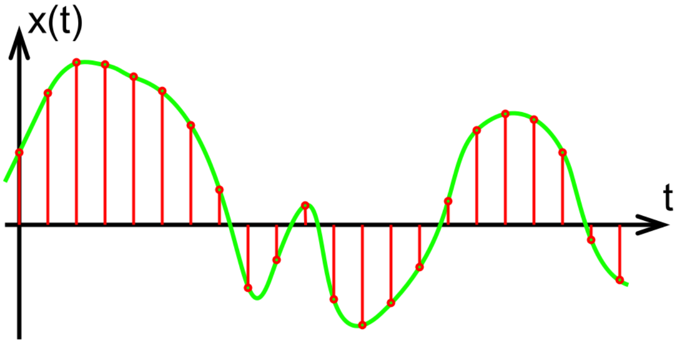


Figure 1 – Sampling a continuous time signal

One main problem that stems when sampling the signal with inappropriate f_s is “aliasing”. When a sampled signal is interpolated (joining the sample points together), the reconstructed signal may resemble a lower frequency signal if sampling frequency is not properly chosen. An example of this is shown in Fig. 2 and 3. In fig. 3, blue color plot shows the continuous time signal, red color plot shows the sampled discrete time signal and the black color plot the reconstructed signal. The top left graph in fig. 3 shows the reconstruction from a sampled signal with $f_s = 2f_m$. The reconstructed signal resembles a triangle but it gives the two peaks (positive and negative peak of the cosine signal) completely, so we can reconstruct the original continuous time signal. In the same fig. 3, top right graph shows the reconstruction of sampled signal with $f_s = f_m$ where we only get the positive peaks. When interpolating this signal, we get a DC (direct current, zero Hertz frequency) component, whereas the original signal has 1 *Hz* frequency. This is called aliasing where 1 *Hz* signal appears as 0 *Hz* signal. One way to avoid aliasing is to sample the signal with $f_s \geq 2f_m$.

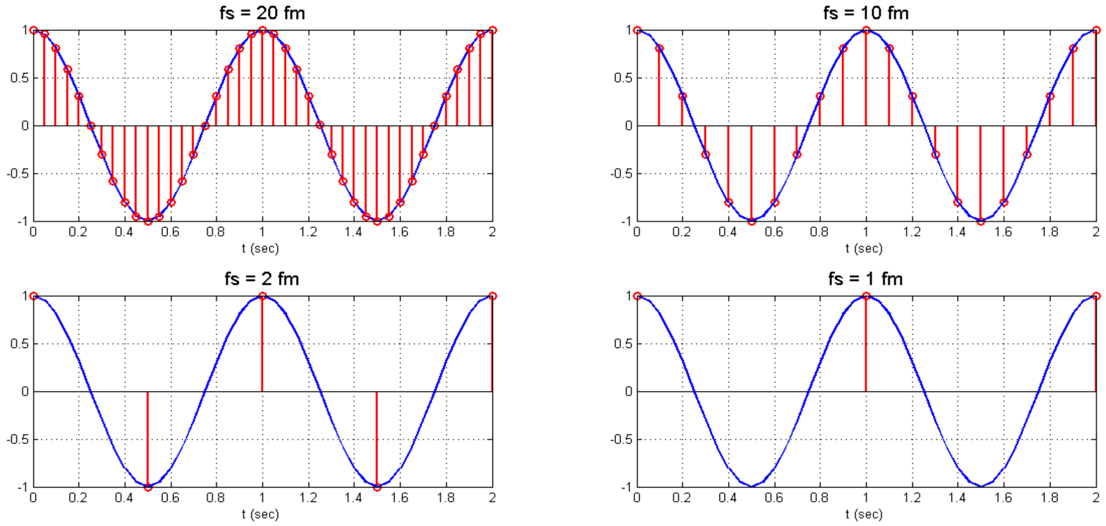


Figure 2 – Sampling a cosine signal of frequency f_m=1 *Hz* with various sample frequency f_s.

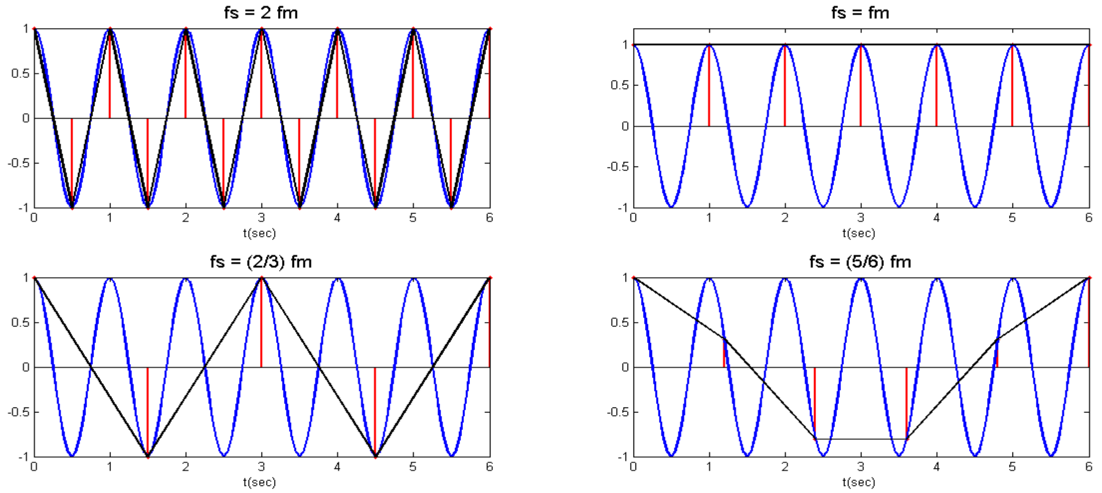


Figure 3 – Reconstruction of signal from sampled signal.

We've seen the sampling of a cosine signal. What do we do when we sample a signal that has more than one cosine signal? How do we avoid aliasing in sampling composite signals? An example is shown below. In eqn. (2), a composite signal with 1000 *Hz* and 3000 *Hz* frequency component is shown. If we choose the sampling frequency $f_s = 2 \text{ x } 1000 = 2000$ (eqn. (3)), and then convert the continuous time signal in eqn. (2) to discrete time signal by substituting $t = nT_s$, and then following the simple trigonometric identity in eqn. (8), we obtain eqn. (9) where the 1000 *Hz* signal appears as $5cos(\pi n)$ and the also the 3000 *Hz* appears as $6cos(\pi n)$. This is aliasing where 3000 *Hz* signal appears as 1000 *Hz* signal. Appropriate selection of sampling frequency f_s rectifies the problem. Here we need to select $f_s \geq 2\ f_{m2} = 2 \text{ x } 3000 = 6000$.

$$x(t) = 5\cos(2000\pi t) + 6\cos(6000\pi t) \tag{1}$$

$$x(t) = 5\cos(2\pi(1000)t) + 6\cos(2\pi(3000)t) \tag{2}$$

$$f_s = 2f_{m1} = 2 * (1000) = 2000 \tag{3}$$

$$t = nT_s, T_s = \frac{1}{f_s} \tag{4}$$

$$x(nT_s) = 5\cos(2\pi(1000)nT_s) + 6\cos(2\pi(3000)nT_s) \tag{5}$$

$$x(n) = 5\cos(2\pi(\frac{1000}{2000})n) + 6\cos(2\pi(\frac{3000}{2000})n) \tag{6}$$

$$x(n) = 5\cos(\pi n) + 6\cos(3\pi n) \tag{7}$$

$$x(n) = 5\cos(\pi n) + 6\cos((2\pi + \pi)n) \tag{8}$$

$$x(n) = 5\cos(\pi n) + 6\cos(\pi n) \tag{9}$$

2. Quantization:

In the previous sections we’ve seen the process of converting continuous time signal into discrete time signal through sampling. Now we’ll see another process known as quantization. In sampling process, we took the signal values at appropriate time points. Those signal values can be ranging from -∞ to ∞ and it may take any value in between there extremes (even fractional values). This is shown in fig. 4. The samples shown as red dots are sampled signal values. In the same figure, quantization levels are shown by dotted gray line and we’ll limit the signal value to one of these levels. This is done because if we limit the number of levels in signal, we can index the values (e.g. 0 to 7 in the fig. 4 shown as 000 to 111 in the y-axis) and this can be converted to binary representation. If we increase the number of levels in y-axis, we also increase the number of bits needed to represent the levels. Ultimately, it is these ones and zeros that gets saved in CD ROM and pen drives that we hear as audio signal. If we observe the figures 6 to 8, we’ll see that as number of quantization levels is increased, the quantized signal closely follows the original signal. This reduces the quantization error. As quantization error is reduced, the original signal can be reconstructed with less error. This error in quantized signal sounds like noise and because of this, it is otherwise called quantization noise. This is given as σ_q^2 in table 1. A measure of signal quality can be measured based on σ_q^2. This is SQNR (signal to quantization noise ratio) and is calculated as,

$$\text{SQNR} = \frac{S}{\sigma_q^2} \tag{10}$$

Where, S is the power of the cosine signal and σ_q^2 is the mean square value of quantization error.

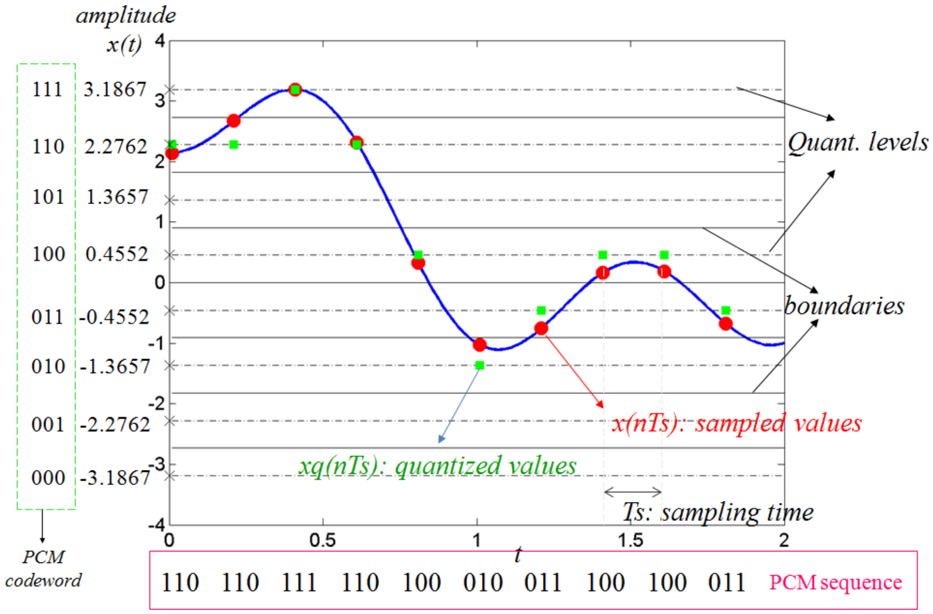


Figure 4 – Quantization, process of discretizing values taken by signal.

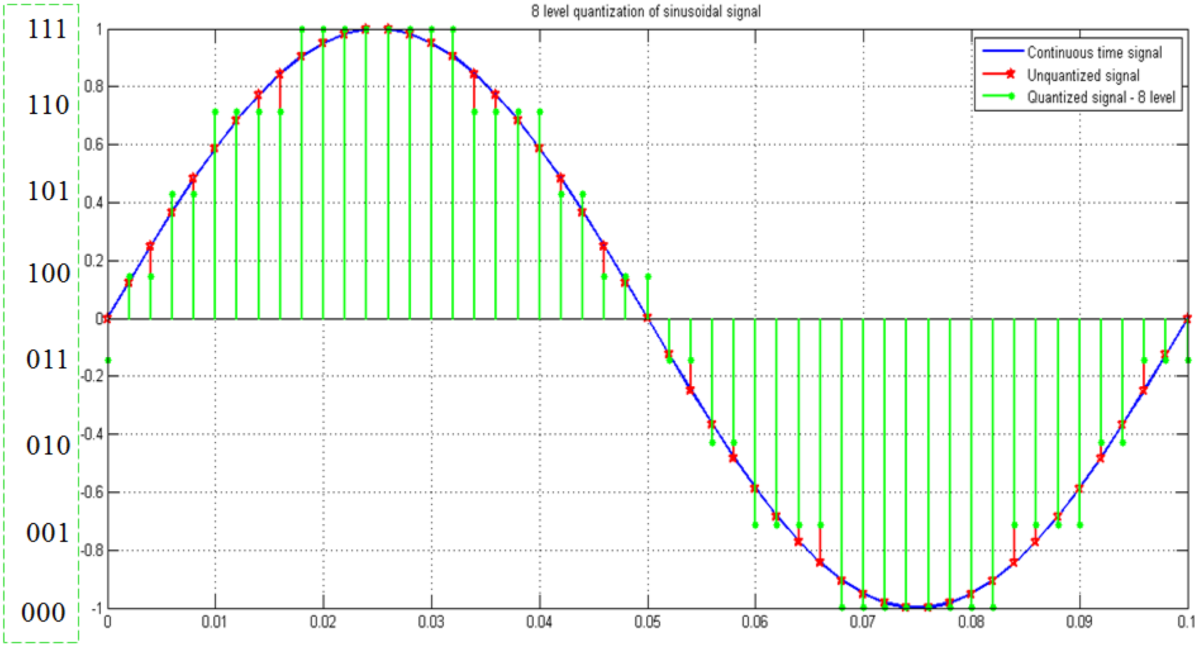


Figure 5 – 8 level quantization of sinusoidal signal and its binary representation

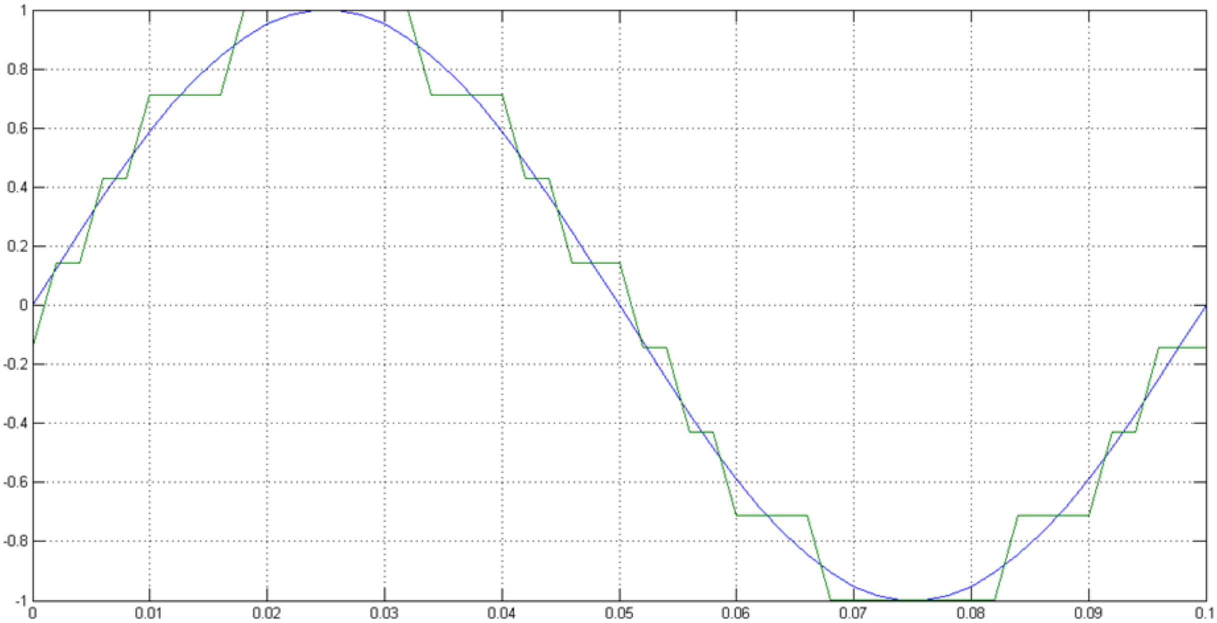


Figure 6 – 8 level quantization of sinusoidal signal shown approximating the original signal.

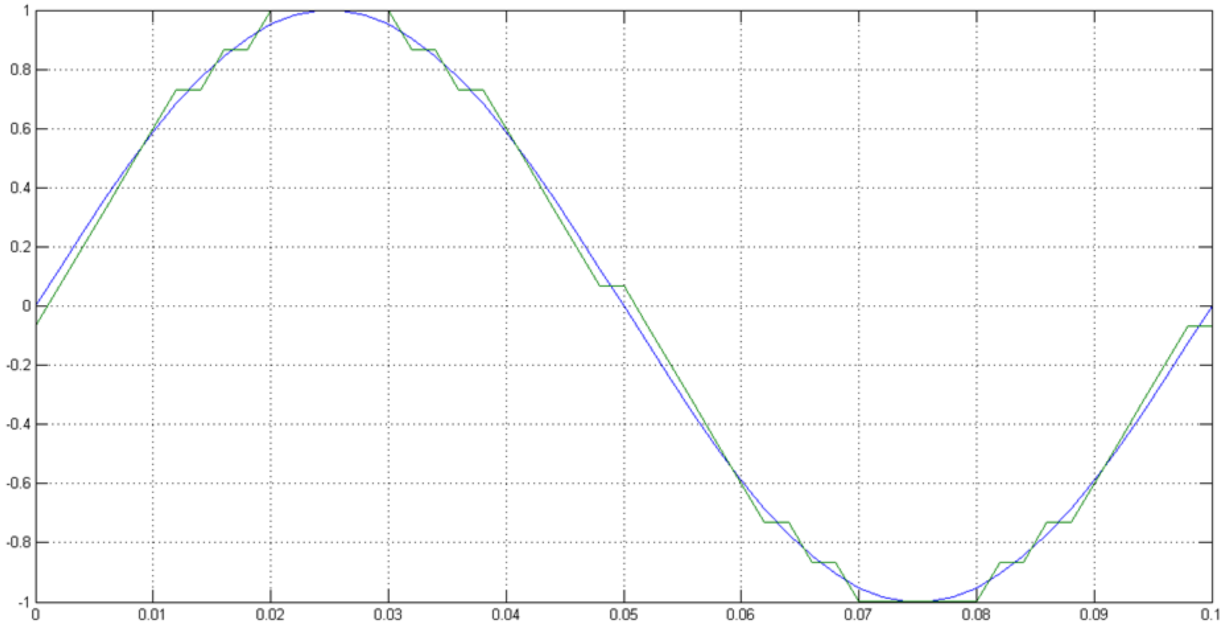


Figure 7 – 16 level quantization of sinusoidal signal shown approximating the original signal.

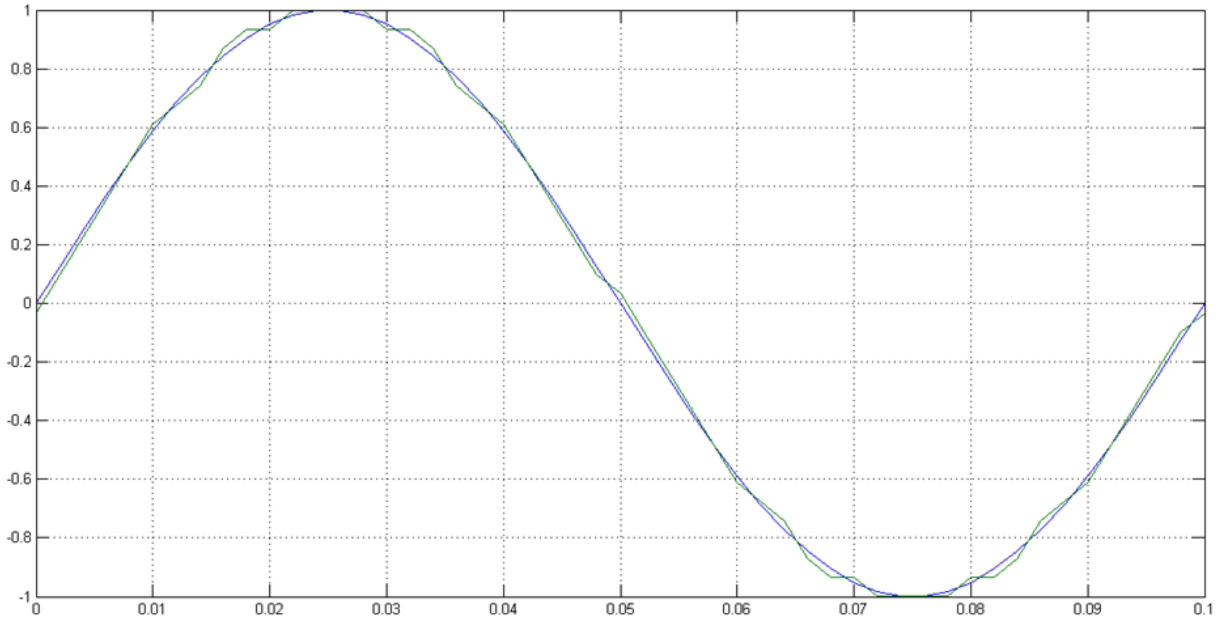


Figure 8 – 32 level quantization of sinusoidal signal shown approximating the original signal.

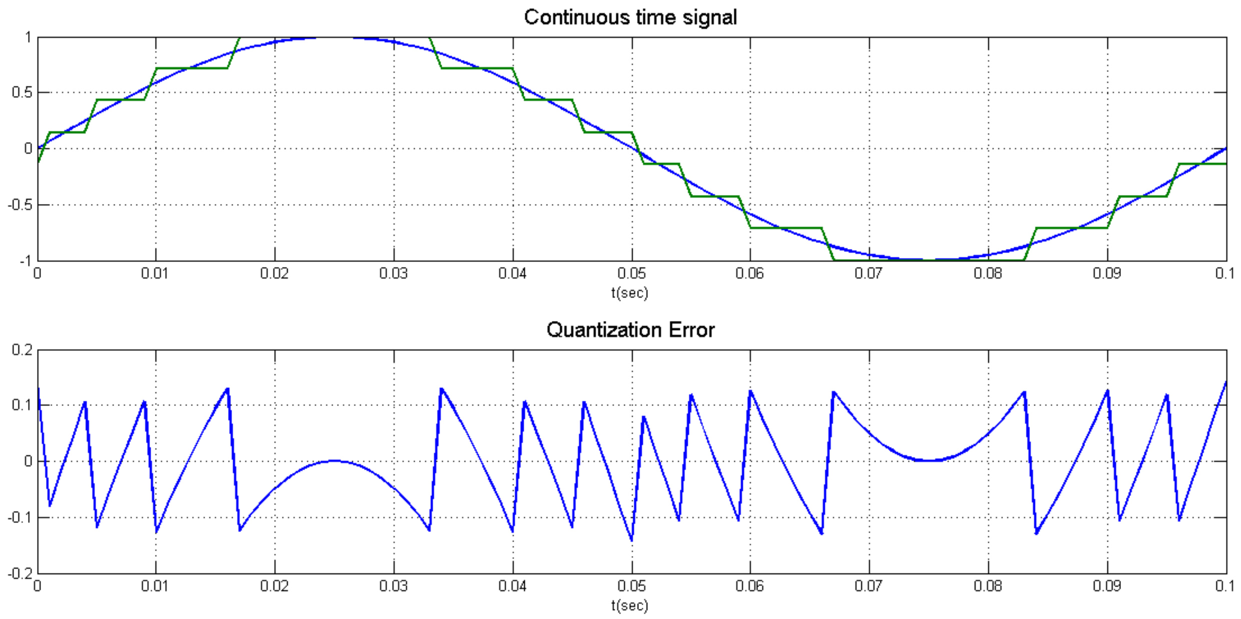


Figure 9 – quantization error of 8 level quantized signal.

Useful Scilab functions:

1. *cos(x)* – Generates a cosine signal for the values in array ‘*x*’. Output is also an array.
2. *quantize(y,b)* – Function for quantizing the values in array ‘*y*’. ‘*b*’ decides the number of quantization levels.
3. *mtlb_var(x,1,2)* – Provides the mean square value of the zero mean signal ‘*x*’.
4. *sound(y,f_s)* – Plays the values of array ‘*y*’ at sampling rate ‘f_s’ as audio.
5. *wavread(“example.wav”)* – Reads the audio in file “*example.wav*” and saves it in an array. The input to this function is an audio file saved in “*.wav*” format.
6. *intdec(y,ratio)* – Down samples the signal in ‘*y*’ by the ratio given in ‘*ratio*’ where *ratio* ≤ 1. Output of this function is down sampled array.

Procedure:

1. Sampling cosine signal:

1. In Scilab, generate a cosine signal of frequency f_m with sampling frequency $f_s \geq 2f_m$.

2. Take a specific number of bits per sample ‘***bits_per_sample***’ and quantize the generated sinusoidal signal with Scilab “*quantize( )*” function.
3. Plot the original sinusoidal signal from step 1 and quantized sinusoidal signal from step 2 in a single figure (overlay the plots).
4. Calculate the difference between the original signal from step 1 and the quantized signal from step 2. This gives the quantization error. From quantization error, find the root mean square (RMS) value using *‘mtlb_var( )’* function in Scilab.
5. Vary the number of bits per sample ‘***bits_per_sample***’ and observe the effect of this on the quantization error and the RMS value of quantization error.
6. Find the ratio of signal power of original cosine signal and the RMS value of quantization error. This measure gives a figure of merit called Signal to Quantization Noise Ratio (SQNR). Find the value of SQNR for different values of ‘***bits_per_sample***’.

2. Voice signal:

1. Record your voice using the microphone attached to the computer. Save the recorded voice in ‘*.wav*’ file format.
2. Open Scilab and by using the function ‘*wavread( )*’ from Scilab, read the file that contains your voice. Save the output of ‘*wavread( )*’ command to an array named ***‘voice’***.
3. Select a sampling frequency ‘f_s’ and number of bits per sample ‘***bits_per_sample***’.
4. Provide the sampling frequency ‘f_s’ and ***‘voice’*** values to Scilab’s ‘*intdec( )*’ function. Save the output of Scilab’s ‘*intdec( )*’ function to a variable named ‘***sampled_voice***’.
5. Now using Scilab’s ‘*quantize( )*’ function, quantize the ‘***sampled_voice***’ and save it in ‘***sampled_quantized_voice***’.
6. Using Scilab’s ‘*sound( )*’ function, play the ‘***sampled_quantized_voice***’.
7. Now calculate the difference between ‘***sampled_voice***’ and ‘***sampled_quantized_voice***’. After this, find the RMS error value.
8. For various values of ‘f_s’ and ‘***bits_per_sample***’, do the above procedure and observe the voice quality and RMS error value.

Results and Calculation:

1. Cosine signal
    a. Quantization noise:

Number of bits/sample	**σ_q^2 (RMS error)**	**SQNR (dB)**
6		
5		
4		
3		

2. Voice signal
    a. Quantization noise:

Number of bits/sample	**RMS error**
8	
6	
4	
3	

Assignment

Concept of digitization and sampling rate for cosine and voice signals

1. Find the appropriate sampling frequency for the following signals:
    a. $s(t) = A_m cos(2\pi t)$
    b. $s(t) = A_m cos(20\pi t)$
    c. $s(t) = A_1 cos(25\pi t) + A_2 cos(50\pi t)$
    d. $s(t) = \sum_{m=1}^{4} A_m cos(2\pi f_m t)$, where $\max(f_1, f_2, f_3, f_4) = f_3$.
2. Generate a cosine signal of amplitude $A_m = 1, f_m = 2000\ Hz$. Change the sampling frequency and observe the sound (see the table in lab manual).
3. Generate a cosine signal of amplitude $A_m = 1, f_m = 10000\ Hz$. Change the number of bits per sample (bits_per_sample) and observe the sound (see the table in lab manual). What happened when number of bits per sample is reduced?
4. Record your voice signal in the PC with Audacity and save it in .wav format. Change the sampling frequency from 8000 samples per second (sps) to 1000 sps in steps of 1000 (see the table in lab manual). What is the observation for reduced sampling frequency?
5. Repeat 4 now by varying bits per sample with sampling frequency $f_s = 8000$ sps (see the table in lab manual).
6. A signal is sampled with sampling frequency f_s = 100 sps. That signal is available for a duration of 2 sec. How many samples of the signal will be there in this duration?
7. 1024 samples of a signal are available. If sampling frequency f_s = 100 sps, how may seconds the signal will last? If we consider f_s = 1024 sps, what is the new duration of the signal?
8. Variance of a signal x_i is defined as $\sigma^2 = (\sum_{i=1}^{N}(x_i - m)^2)/N$. Where m is the average value of x_i and N is the length of the signal x_i. x_i is given in the below tabulation. Calculate the variance of x_i using the above formula in Scilab. Verify the result using *mtlb_var(x,1,2)* Scilab function.

i	1	2	3	4	5	6	7	8	9	10
x_i	11	55	29	20	55	99	96	100	97	97

Demo of
Concept of digitization and sampling rate for cosine and voice signals

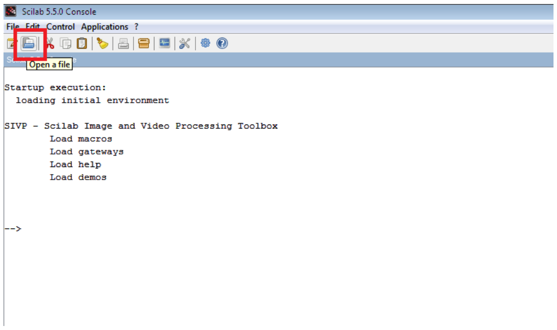
This file gives the demo of how to do the sampling of cosine and voice signal. The softwares and equipment required are:

1. Scilab 5.5.0.
2. Audacity.
3. Headphones with built-in microphone.

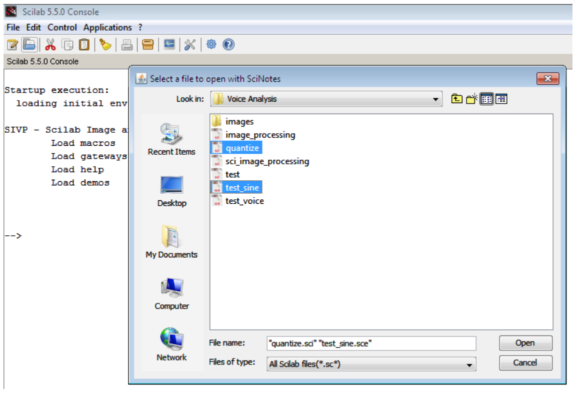
Steps involved in cosine signal sampling:

Step 1: Open Scilab 5.5.0 from the desktop.

Step 2: Open the files “test_sine.sce” and “quantize.sci” using the Scilab 5.5.0.



```
Startup execution:
  loading initial environment

SIVP - Scilab Image and Video Processing Toolbox
       Load macros
       Load gateways
       Load help
       Load demos



-->
```



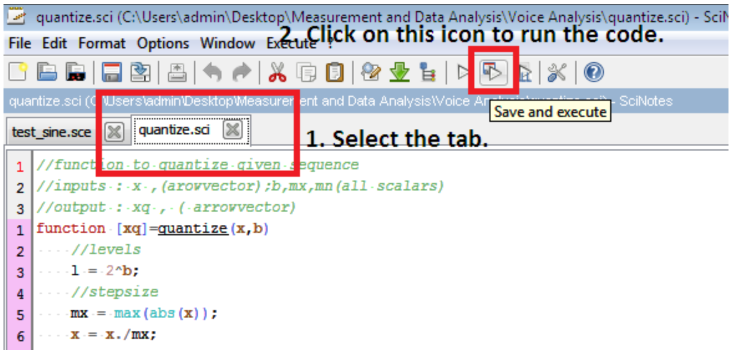
```
Startup execution:
  loading initial env

SIVP - Scilab Image a
       Load macros
       Load gateways
       Load help
       Load demos



-->
```

Look in: Voice Analysis

File name: "quantize.sci" "test_sine.sce"

Files of type: All Scilab files(*.sc*)

Step 3: Click on the "quantize.sci" function file in Scilab 5.5.0. Run the code for one time by pressing "F5" function key or by pressing the "Save and execute" icon available in the Scilab as shown below.

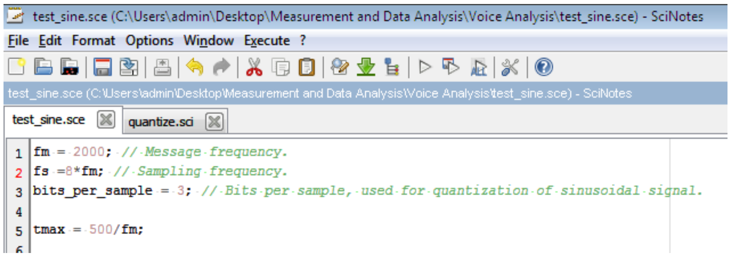

quantize.sci (C:\Users\admin\Desktop\Measurement and Data Analysis\Voice Analysis\quantize.sci) - SciN
File Edit Format Options Window Execute ?
Save and execute
test_sine.sce
quantize.sci

2. Click on this icon to run the code.

1. Select the tab.

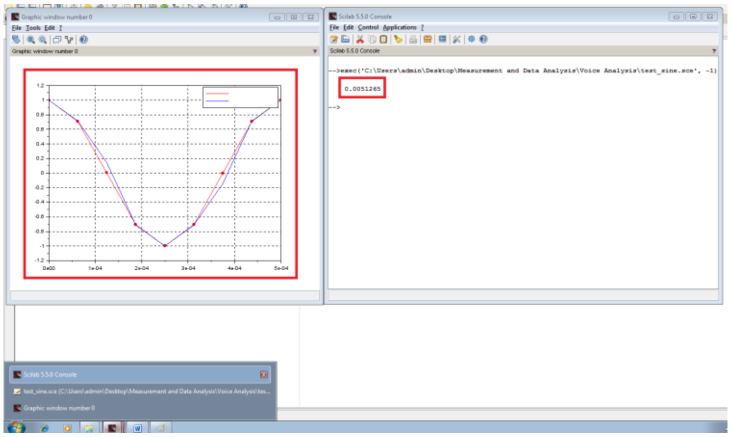
```
//function to quantize given sequence
//inputs : x , (arowvector);b,mx,mn(all scalars)
//output : xq , ( arrowvector)
function [xq]=quantize(x,b)
    //levels
    l = 2^b;
    //stepsize
    mx = max(abs(x));
    x = x./mx;
```

Step 4: Switch the tab to "test_sine.sce" file. Change the message frequency, sampling frequency and bits per sample as desired using the variables shown below.


test_sine.sce (C:\Users\admin\Desktop\Measurement and Data Analysis\Voice Analysis\test_sine.sce) - SciNotes
File Edit Format Options Window Execute ?
test_sine.sce
quantize.sci

```
fm = 2000; // Message frequency.
fs =8*fm; // Sampling frequency.
bits_per_sample = 3; // Bits per sample, used for quantization of sinusoidal signal.

tmax = 500/fm;
```

Step 5: Press "F5" or click "Save and execute" icon to run the code. Listen to the sound of the sinusoidal signal you hear from Scilab. Go to the Scilab console window and note down the RMS error as shown below. Also view the quantized signal plotted in the figure.



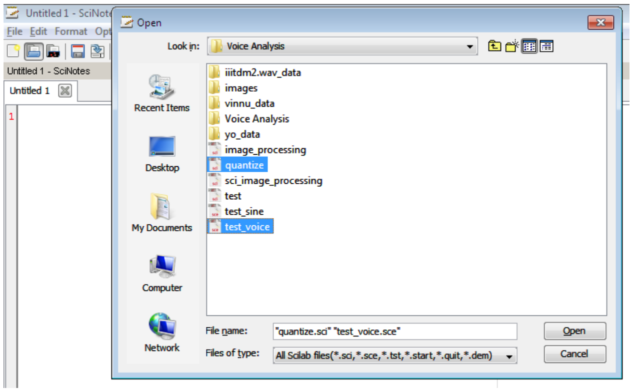
```
-->exec('C:\Users\admin\Desktop\Measurement and Data Analysis\Voice Analysis\test_sine.sce', -1)

 0.0051265

-->
```

Step 6: Repeat Step 4 and Step 5 for various values of sampling frequencies and bits per sample values (quantization experiment).

Steps involved in Voice recording and sampling:

Step 1: Open Scilab 5.5.0 from the desktop.

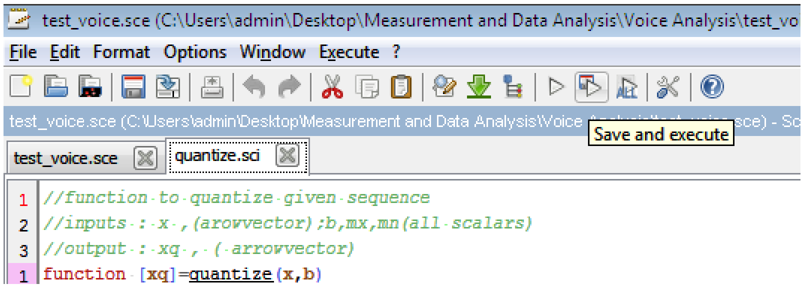
Step 2: Open the files “test_voice.sce” and “quantize.sci” using the Scilab 5.5.0 as shown below.



File name: "quantize.sci" "test_voice.sce"

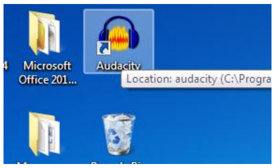
Files of type: All Scilab files(*.sci,*.sce,*.tst,*.start,*.quit,*.dem)

Step 3: Click on the “quantize.sci” function file in Scilab 5.5.0. Run the code for one time by pressing “F5” function key or by pressing the “Save and execute” icon available in the Scilab as shown below.

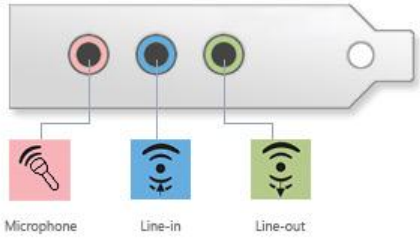

test_voice.sce (C:\Users\admin\Desktop\Measurement and Data Analysis\Voice Analysis\test_vo
File Edit Format Options Window Execute ?
Save and execute
test_voice.sce
quantize.sci

```
//function to quantize given sequence
//inputs : x , (arowvector) ;b,mx,mn(all scalars)
//output : xq , ( arrowvector)
function [xq]=quantize(x,b)
```

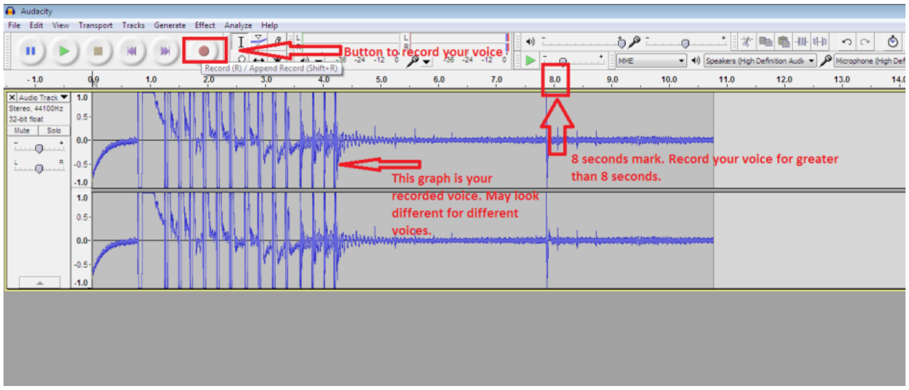
Step 4: Click on “Audacity” icon on the desktop to open “Audacity”.



Step 5: Connect your headphone with inbuilt microphone with the system by connecting appropriate pins of the headphone and microphone into the system sockets.



Step 6: Go to audacity and record your voice for minimum of 8 seconds like shown in the below figure.

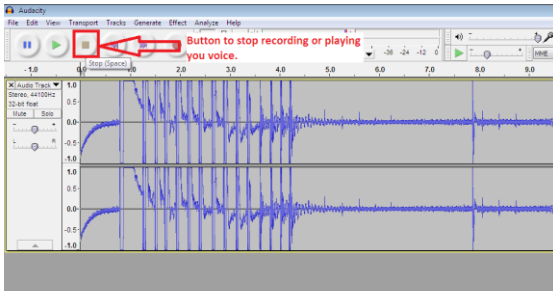

Audacity
File Edit View Transport Tracks Generate Effect Analyze Help
Record (R) / Append Record (Shift+R)
8.0
Audio Track
Stereo, 44100Hz
32-bit float
Mute Solo

Button to record your voice

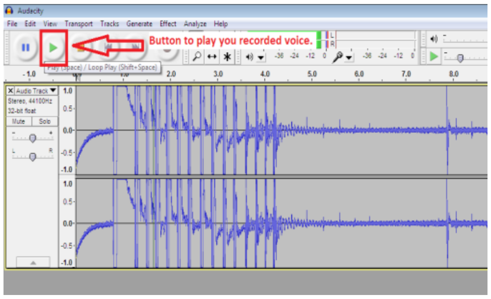
8 seconds mark. Record your voice for greater than 8 seconds.

This graph is your recorded voice. May look different for different voices.

Step 7: Stop recording your voice and check your recorded voice by playing it like shown below. You should be able to hear your voice clearly while you play it. Otherwise record it again by closing Audacity and following Step 6.

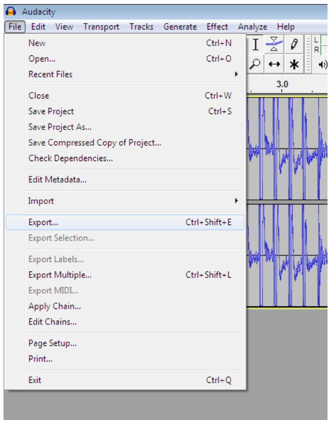

Audacity
File Edit View Transport Tracks Generate Effect Analyze Help
Stop (Space)

Button to stop recording or playing you voice.

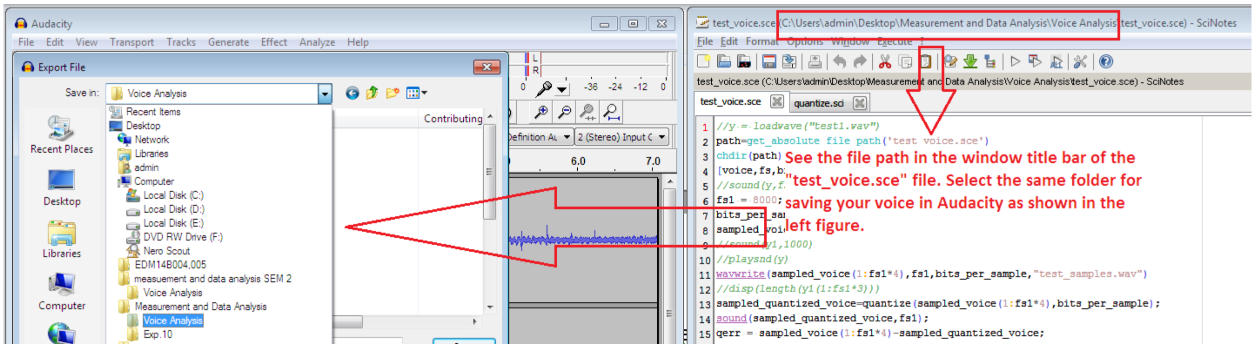

Audacity
File Edit View Transport Tracks Generate Effect Analyze Help
Play (Space) / Loop Play (Shift+Space)

Button to play you recorded voice.

Step 8: Save your recorded voice by clicking on the “Export” option available in the “File” menu item as shown below.



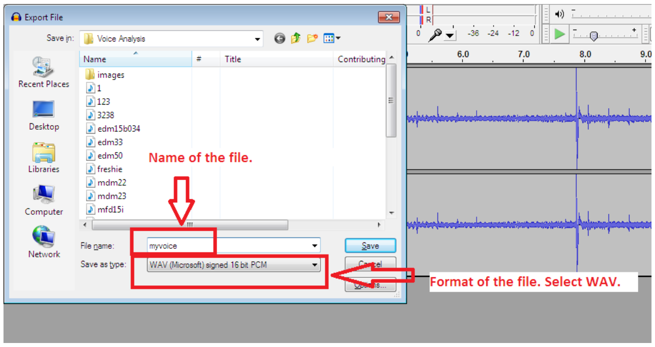
Step 9: Select the folder where you want to save the recorded voice as shown in the below figure. The right side figure is the Scilab window. The left side one is Audacity "Export" page dialog box.



```
//y = loadwave("test1.wav")
path=get_absolute file path('test voice.sce')
chdir(path)
[voice,fs,b
//sound(y,f
fs1 = 8000;
bits_per_sa
sampled_voi
//sound(y1,1000)
//playsnd(y)
wavwrite(sampled_voice(1:fs1*4),fs1,bits_per_sample,"test_samples.wav")
//disp(length(y1(1:fs1*3)))
sampled_quantized_voice=quantize(sampled_voice(1:fs1*4),bits_per_sample);
sound(sampled_quantized_voice,fs1);
qerr = sampled_voice(1:fs1*4)-sampled_quantized_voice;
```

See the file path in the window title bar of the "test_voice.sce" file. Select the same folder for saving your voice in Audacity as shown in the left figure.

Step 10: After selecting the folder, give your recorded voice a name. For example, "myvoice". Save the file in .wav format as shown below.



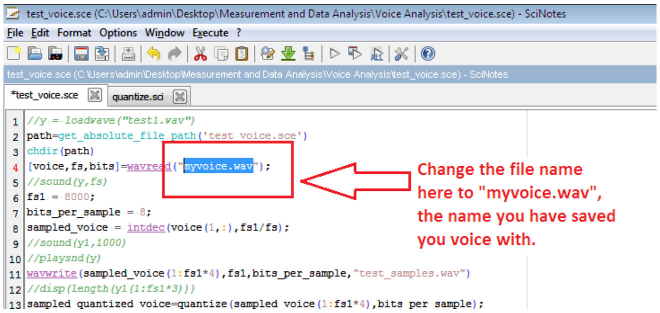
File name: myvoice

Save as type: WAV (Microsoft) signed 16 bit PCM

Name of the file.

Format of the file. Select WAV.

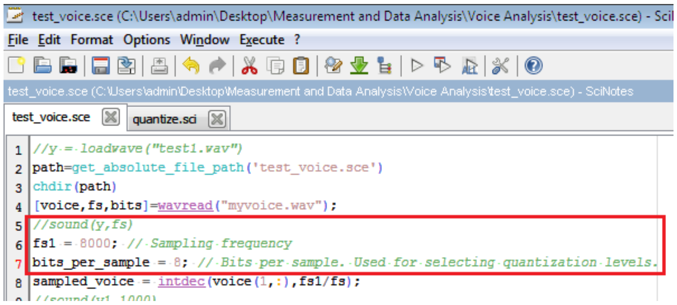
Step 11: Now go to Scilab software. Go to the "test_voice.sce" code file. Give the name of the file you have saved in Step 10 here.


test_voice.sce (C:\Users\admin\Desktop\Measurement and Data Analysis\Voice Analysis\test_voice.sce) - SciNotes
File Edit Format Options Window Execute ?
*test_voice.sce
quantize.sci

```
//y = loadwave("test1.wav")
path=get_absolute_file path('test voice.sce')
chdir(path)
[voice,fs,bits]=wavread("myvoice.wav");
//sound(y,fs)
fs1 = 8000;
bits_per_sample = 8;
sampled_voice = intdec(voice(1,:),fs1/fs);
//sound(y1,1000)
//playsnd(y)
wavwrite(sampled_voice(1:fs1*4),fs1,bits_per_sample,"test_samples.wav")
//disp(length(y1(1:fs1*3)))
sampled quantized voice=quantize(sampled voice(1:fs1*4),bits per sample);
```

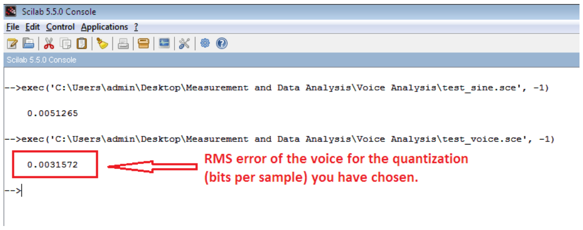
Change the file name here to "myvoice.wav", the name you have saved you voice with.

Step 12: Change the sampling frequency and bits per sample parameters in the scilab file as shown below to your desired values.


test_voice.sce (C:\Users\admin\Desktop\Measurement and Data Analysis\Voice Analysis\test_voice.sce) - Scil
File Edit Format Options Window Execute ?
test_voice.sce
quantize.sci

```
//y = loadwave("test1.wav")
path=get_absolute_file_path('test_voice.sce')
chdir(path)
[voice,fs,bits]=wavread("myvoice.wav");
//sound(y,fs)
fs1 = 8000; // Sampling frequency
bits_per_sample = 8; // Bits per sample. Used for selecting quantization levels.
sampled_voice = intdec(voice(1,:),fs1/fs);
```

Step 13: Press “F5” or click on “Save and execute” icon to run the code. Listen to you voice and note down the RMS error that displayed in the Scilab console as shown below.



```
-->exec('C:\Users\admin\Desktop\Measurement and Data Analysis\Voice Analysis\test_sine.sce', -1)

    0.0051265

-->exec('C:\Users\admin\Desktop\Measurement and Data Analysis\Voice Analysis\test_voice.sce', -1)

    0.0031572

-->
```

RMS error of the voice for the quantization (bits per sample) you have chosen.

Step 14: Repeat Step 12 to Step 13 for different values of sampling frequency and bits per sample values.