Digital Communication

Lab Manual #3

Sampling and Quantization

Objective :
To Learn how to digitize a signal: Sampling is used to discretize it's time and Quantization is to discretize it's amplitude .

Methodology:

An analogue signal is characterized by the fact that its attributes (like: amplitude, frequency and phase) can take any value over a continuous range. On the other hand, digital signals can take only discrete and finite values. One can convert an analogue signal to a digital signal by sampling and quantizing (collectively called analog-to-digital conversion, or ADC).
It is typically more efficient to process the resulting discrete signals by digital signal processors. The processed signals are then converted back into analog signals using a reconstruction or interpolation operation (called digital-to-analog conversion, or DAC).

Sampling:

In **signal processing**, **sampling** is the reduction of a continuous **signal** to a discrete **signal**. A common example is the conversion of a sound wave (a continuous **signal**) to a sequence of samples (a discrete-time **signal**). A **sample** is a value or set of values at a point in time and/or space.
Suppose signal x(t) **in figure on top left** has the highest frequency component f_m. According to the Nyquist Theorem, the sampling rate(no of samples per second) must be at least $2f_m$, or twice the highest analogue frequency component. If the sampling rate is less than $2f_m$, some of the highest frequency components in the analogue input signal may not be correctly represented in the digitized output.

How to do Sampling:

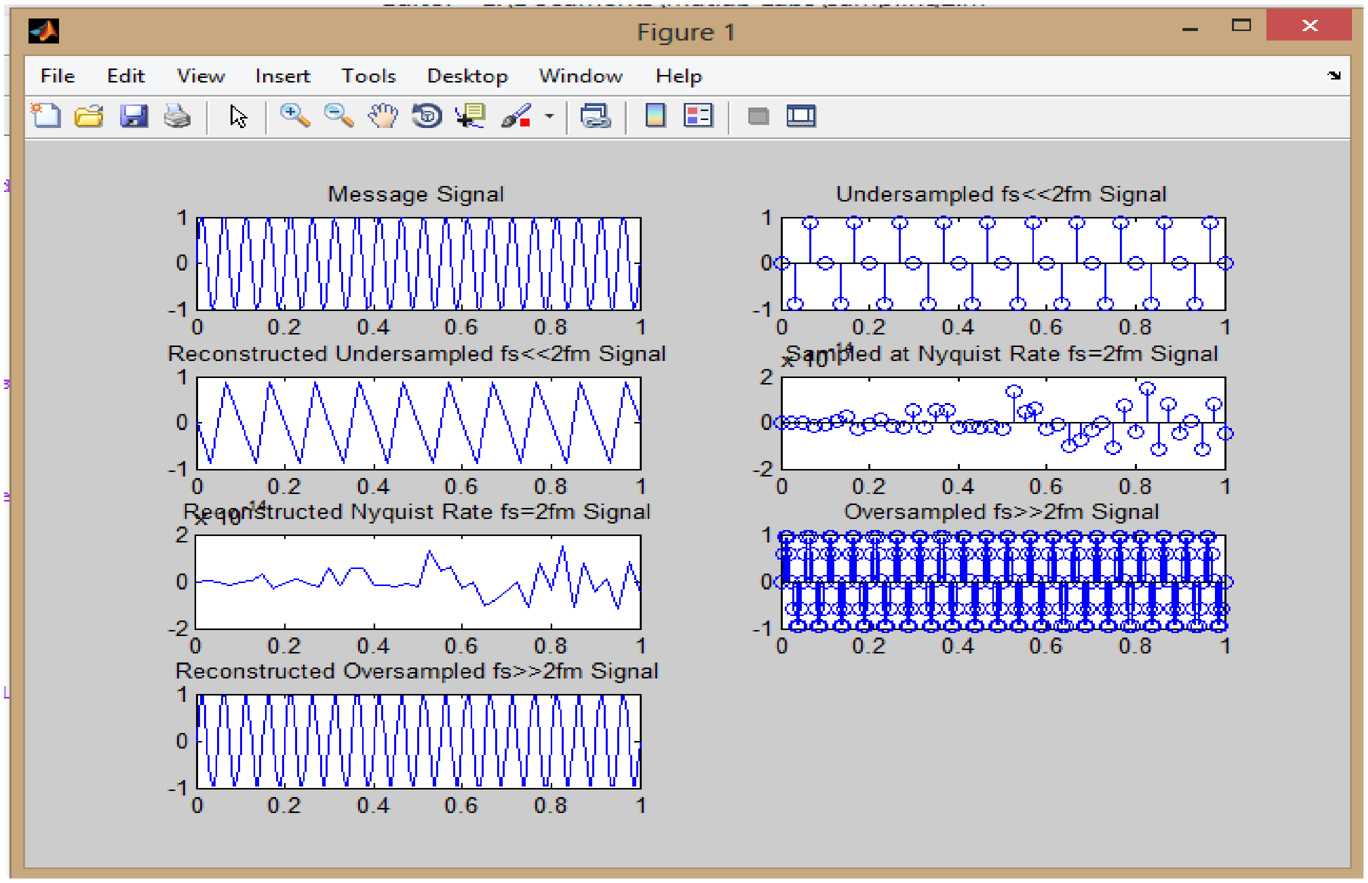
To sample this continuous-time signal x(t)is to represent x(t) at a discrete number of points, making $\mathbf{t = nT_s}$, where T_s is the sampling period(time to take one sample). This signal can be reconstructed from its sampled values

$$\mathbf{y(n) = x(nT_s)}$$

Here y will be the sampled signal of x. Sampled at rate f_s=1/T_s

If $f_s < 2f_m$ this will cause a distorted signal.For $f_s=2f_m$ we will still not get the perfect shape as of our original signals.At $f_s>2f_m$ we can recreate the original signal.See Example given below.

As the image below shows that the signal reconstructed under and equal to Nyquist rate does not exactly recreate the original signal.but the reconstruction of signal from the data that is sampled at even slightly greater value than Nyquist perfectly represents the original signal.



TASK 1:

Recreate the figure given above to Prove Nyquist theorem using Matlab.

Quantization:

Quantization is discretizing the amplitude we drop the continues values of amplitude in a specified step and assign them same value from that range. ***Quantization causes data loss.***

In order to process the sampled signal digitally, the sample values have to be quantized to a finite number of levels along y-axis, and each value can then be represented by a string of bits. For example, if the signal is quantized to N different levels, then log2(N) bits per sample are required.

$$\mathbf{n = \log_2 N}$$

where N = No of levels of quatization

n = No of bits used to represent each signal value.

Notice that to quantize a sample value is to round it to the nearest point among a finite set of permissible values. Therefore, a distortion will inevitably occur. This is called quantization noise (or error).

Quantization can be classified as uniform and non-uniform. In the case of uniform quantization, the quantization regions are chosen to have equal length. However, in non-uniform quantization, regions of various lengths are allowed. Non-uniform quantization can be implemented through compression-expansion (or companding) of the signal, and this is commonly used (as in telephony) to maintain a uniform signal-to-quantization noise ratio over the full dynamic range of the signal (refer to your textbook for more details).

Steps for Uniform Quantization of sine wave (or any other signal) stored in x:

1: Create Bins Along y-axis(Amplitude).

I. By calculating the step size for each bin

$$\mathbf{\Delta = (max(x)\text{-}min(x))\ /no.\ of\ bins}$$

II. now your first level is

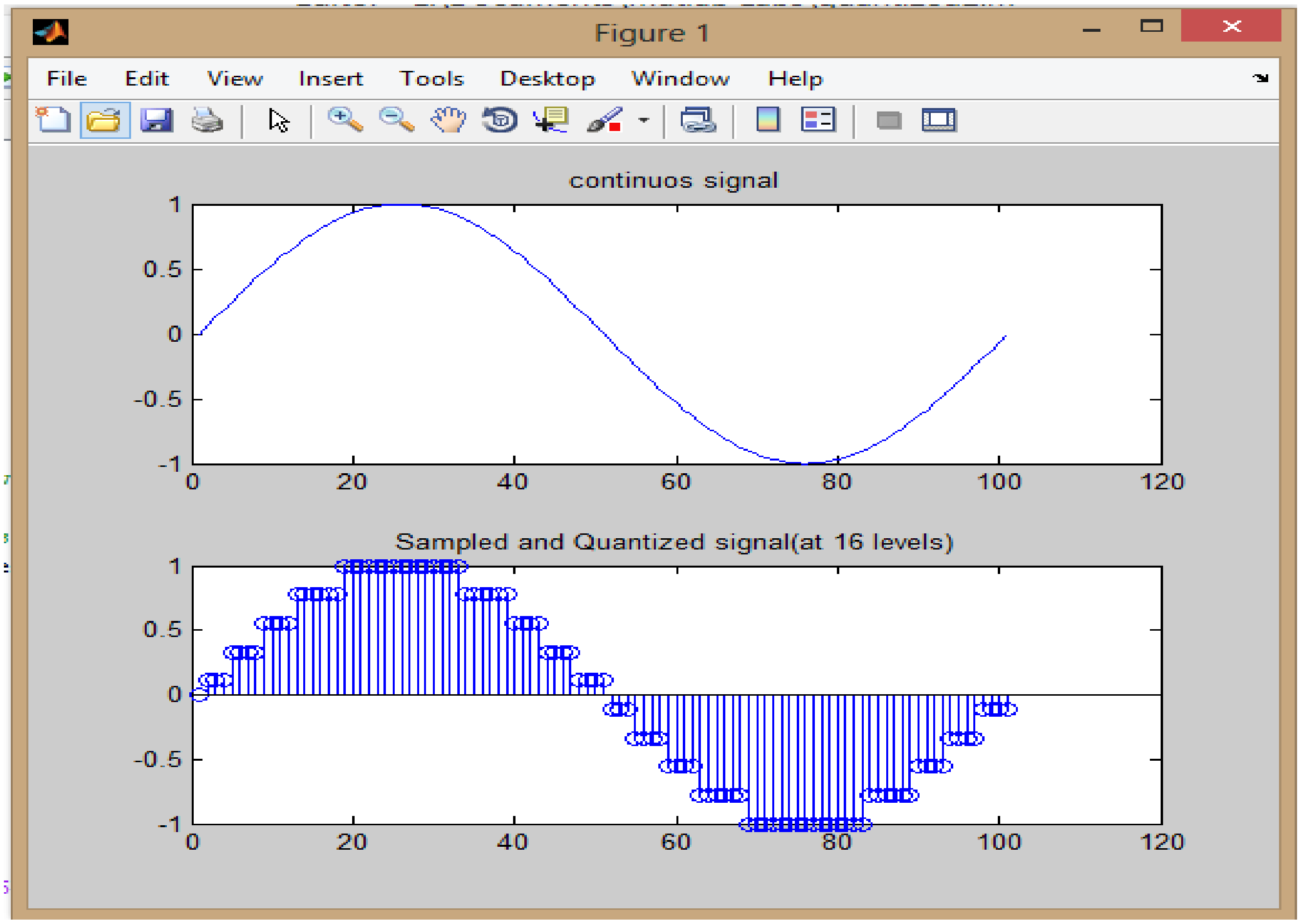
$$\mathbf{step_{j+1} = step_j + \Delta}$$

use loop to access all the levels

2: For every value of the signal x check in which bin/level It lies and than assign it lowest , highest or middle value of that bin/level.

3: Plot the reassigned values of the quantized Signal.

For a simple sine wave Output should be:

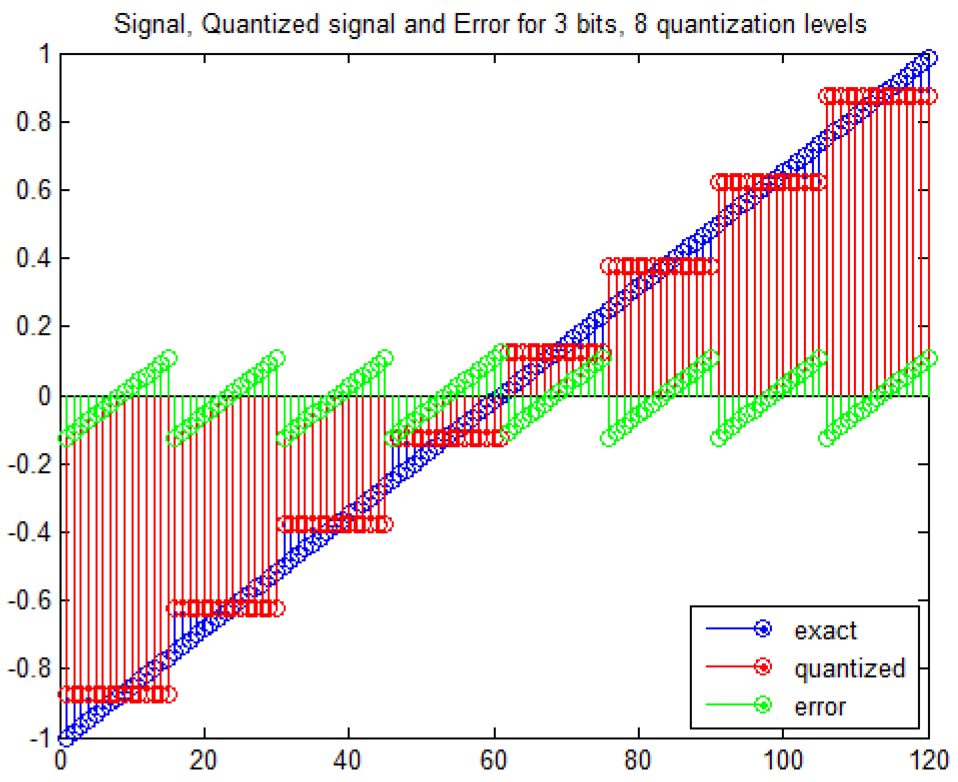


Quantization Error:

When a signal is digitized, quantization errors occur. This is demonstrated below where the input signal, x, is rounded to one of 8 quantization levels.

In this diagram the sampled (but not yet quantized) signal is in blue (i.e., x[n]), the sampled and quantized signal (i.e., $x_q[n]$) is in red, and the quantization error (i.e., e[n]) is shown in green.
To calculate e[n] use subtraction:

$$\mathbf{e[n] = x[n] - x_q[n]}$$



What is companding?
Companding refers to a technique for compressing and then expanding (or decompressing) an analog or digital signal. It is a combination of the words "compressing" and "expanding." This twin-sequential process is non-linear overall but linear over short periods of time. Data is compressed before being transmitted. Then, it is expanded at the receiving end using the same non-linear scale to restore it to its original form.

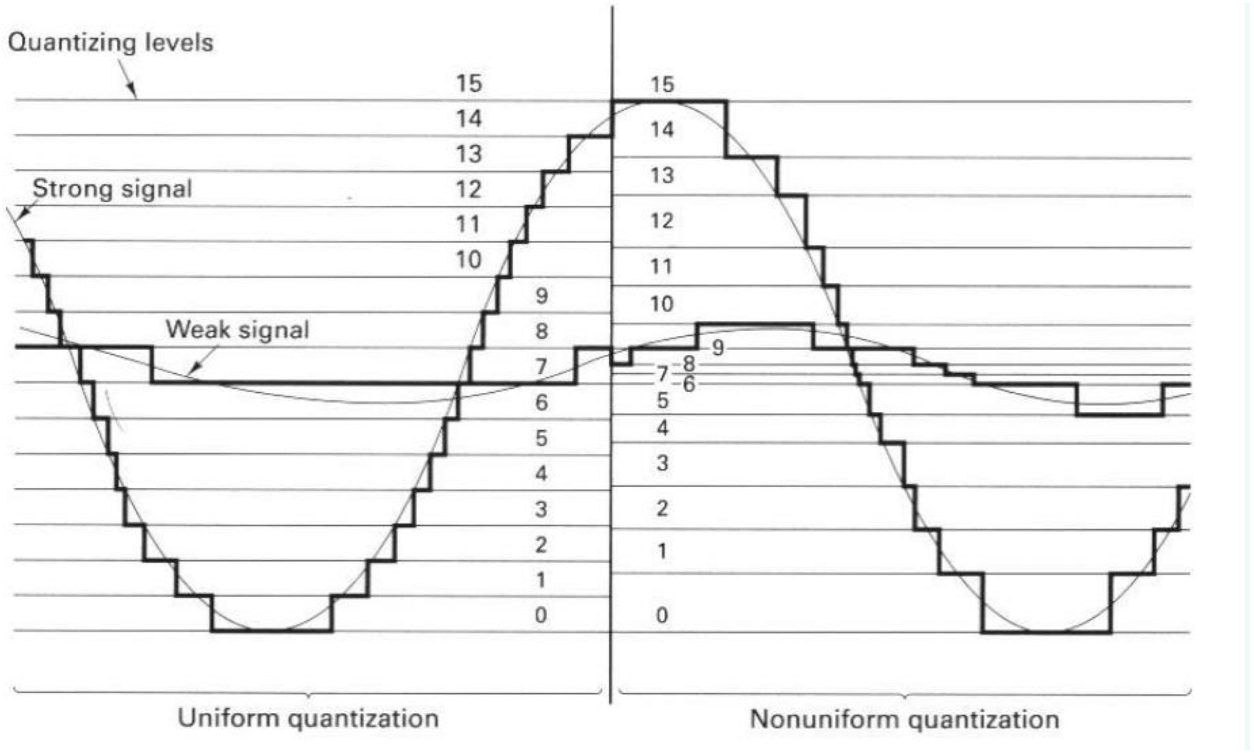
Non Uniform Quantization/Companding:
Definition:
Nonuniform quantizers have unequally spaced levels
1) The spacing can be chosen to optimize the Signal-to-Noise Ratio for a particular type of signal

2) It is characterized by:

- Variable step size
- Quantizer size depend on signal size

Why Non Uniform Quantization:
In the following figure right in non-uniform quantization of both weak and a strong signal whereas left side is uniform quantization. As we can see for weak signal/small amplitude uniform quantization(same level size for all signals/amplitudes) had made only two level due to which we have completely lost the original shape of signal. But it can define strong signal perfectly.

Now if we increase the number of levels than it will be a solution for weak signal but strong signal is too large and cause an increased numbers of bits.
Tradeoff to these problems is using non uniform quantization in which we use small step size on weak signal/small amplitudes and step size starts to increase as we move from low amplitude to high amplitude.



A Practical Example:

Consider the situation where we have an analog voice signal. This voice signal tends to stay within certain amplitude limits, although on occasion there are some high-amplitude moments. We can say, for instance, that most of the analog signal is between 0V and 5V, although on occasion the signal can get as high as 20V. It would seem that we want to spend more of our resources in the bottom quarter of our range, and not worry so much about the upper parts of the range.

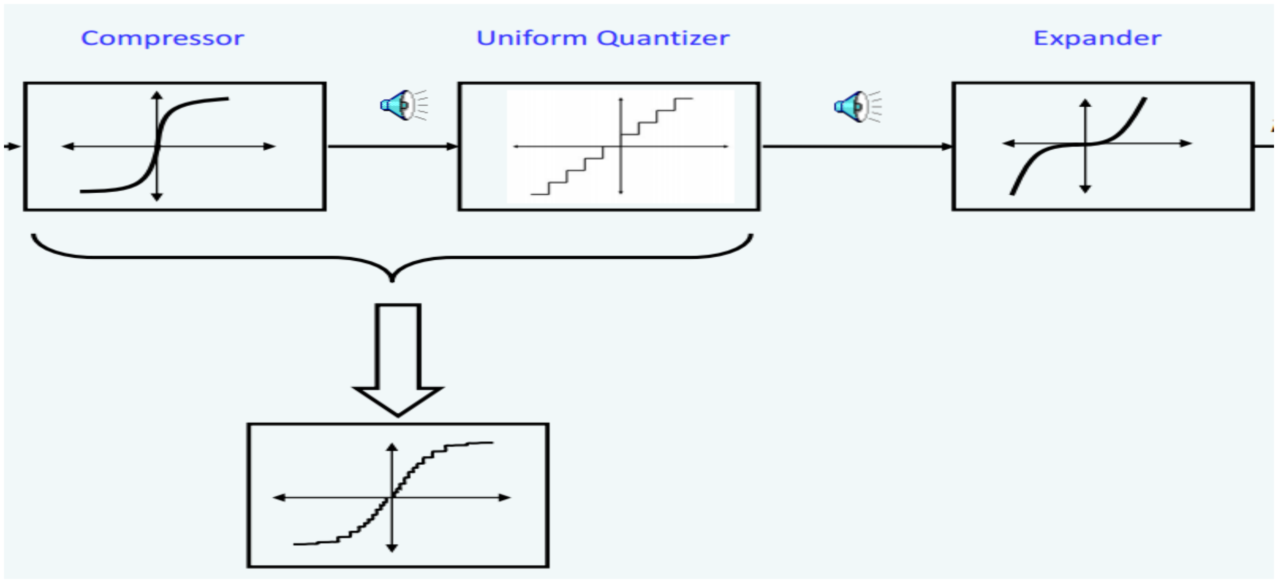
If our sampler has a resolution of 5 bits (32 levels), then a full 3/4ths, (24) of our levels are essentially a waste of space! Now, what if there was a way to "spend" more of our levels, and use more of our bits, between 0V and 5V, and not spend many levels at all on the upper 15V of our range. There is a way to do this, and it's called **Non-Uniform Quantization**

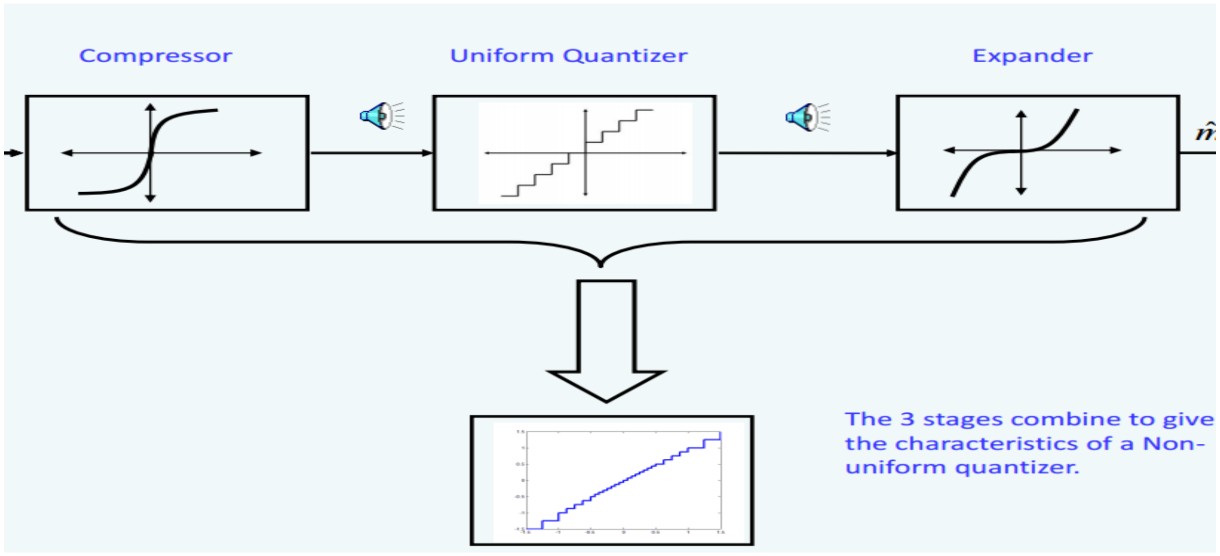
Implementation of Non-Uniform Quantization:

Non-uniform quantization is achieved by, first passing the input signal through a “compressor”. The output of the compressor is then passed through a uniform quantizer.

The combined effect of the compressor and the uniform quantizer is that of a nonuniform quantizer.

At the receiver the voice signal is restored to its original form by using an expander. This complete process of Compressing and Expanding the signal before and after uniform quantization is called Companding.




Compressor
Uniform Quantizer
Expander
The 3 stages combine to give the characteristics of a Non-uniform quantizer.

Matlab Functions we will be using:
We will be using following functions during this lab.
Quantiz:

You can use your own code too.
index = quantiz(sig,partition) returns the quantization levels in the real vector signal sig using the parameter partition. partition is a real vector whose entries are in strictly ascending order.
If partition has length n, index is a vector whose kth entry is

- 0 if sig(k) ≤ partition(1)
- m if partition(m) < sig(k) ≤ partition(m+1)
- n if partition(n) < sig(k)

[index,quants] = quantiz(sig,partition,codebook) is the same as the syntax above, except that codebook prescribes a value for each partition in the quantization and quants contains the quantization of sig based on the quantization levels and prescribed values. codebook is a vector whose length exceeds the length of partition by one.quants is a row vector whose length is the same as the length of sig. quants is related to codebook and index by

```
quants(ii) = codebook(index(ii)+1);
```

where ii is an integer between 1 and length(sig).

Example for 4 level quantization

```
t=0:0.001:1;
x=sin(2*pi*t*5);
partition=-1:2/4:1;
out=quantiz(x,partition);
plot(out)
```

Compand() for companding:
out = compand(in,Mu,v,'**mu/compressor**') implements a µ-law compressor for the input vector in. Mu specifies µ, and v is the input signal's maximum magnitude. out has the same dimensions and maximum magnitude as in.

out = compand(in,Mu,v,'**mu/expander**') implements a µ-law expander for the input vector in. Mu specifies µ and v is the input signal's maximum magnitude. out has the same dimensions and maximum magnitude as in.

out = compand(in,A,v,'**A/compressor**') implements an A-law compressor for the input vector in. The scalar A is the A-law parameter, and v is the input signal's maximum magnitude. out is a vector of the same length and maximum magnitude as in.

out = compand(in,A,v,'**A/expander**') implements an A-law expander for the input vector in. The scalar A is the A-law parameter, and v is the input signal's maximum magnitude. out is a vector of the same length and maximum magnitude as in.

TASK 2:

Digitize a sin wave of frequency 10 by sampling according to Nyquist, Quantize using 16 levels and encode by assigning binary codes.

TASK 3:

I. Apply Uniform and non Uniform quantization on a sound signal and compare their Quantization error.

II. Apply A-law and Mu-law for non-uniform quantization and compare their Quantization error.