***Experiment No: 2***

**Aim:** To study sampling and reconstruction of signal

**Objective:** Develop a program to sample a continuous time signal and convert it to Discrete Time Signal.

**Problem Definition:**

* Sample the input signal and display first 50 samples. Calculate data rate and bit rate.
* Reconstruct the original signal and display the original and reconstructed signals.
* Vary the sampling frequency and observe the change in the quality of reconstructed signal.

**Theory:**

In signal processing, sampling is the reduction of a continuous-time signal to a discrete-time 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.A sampler is a subsystem or operation that extracts samples from a continuous signal.A theoretical ideal sampler produces samples equivalent to the instantaneous value of the continuous signal at the desired points.

Sampling can be done for functions varying in space, time, or any other dimension, and similar results are obtained in two or more dimensions.

For functions that vary with time, let s(t) be a continuous function (or "signal") to be sampled, and let sampling be performed by measuring the value of the continuous function every T seconds, which is called the sampling interval or the sampling period. Then the sampled function is given by the sequence:

s(nT), for integer values of n.

The sampling frequency or sampling rate, fs, is the average number of samples obtained in one second (samples per second), thus

fs = 1/T.

Reconstructing a continuous function from samples is done by interpolation algorithms. The Whittaker–Shannon interpolation formula is mathematically equivalent to an ideal lowpass filter whose input is a sequence of Dirac delta functions that are modulated (multiplied) by the sample values. When the time interval between adjacent samples is a constant (T), the sequence of delta functions is called a Dirac comb. Mathematically, the modulated Dirac comb is equivalent to the product of the comb function with s(t). That purely mathematical abstraction is sometimes referred to as impulse sampling.

Most sampled signals are not simply stored and reconstructed. But the fidelity of a theoretical reconstruction is a customary measure of the effectiveness of sampling. That fidelity is reduced when s(t) contains frequency components whose periodicity is smaller than 2 samples; or equivalently the ratio of cycles to samples exceeds ½ (see Aliasing). The quantity ½ cycles/sample × fs samples/sec = fs/2 cycles/sec (hertz) is known as the Nyquist frequency of the sampler. Therefore, s(t) is usually the output of a lowpass filter, functionally known as an anti-aliasing filter. Without an anti-aliasing filter, frequencies higher than the Nyquist frequency will influence the samples in a way that is misinterpreted by the interpolation process.

**Steps:**

1. Take two fundamental frequencies F1=0.1 and F2=0.2.

2. Original signal values are calculated by using formulae:

sin2\*pi\*F1\*t(0 to 20.5 sec)+sin2\*pi\*F2\*t(0 to 20.5 sec) for the difference of 0.1 sec.

3. Output i.e total 201 values will be written into orig.txt file.

4. Sampling

a.Take 0 to 201 values.

b. for m=1 to 201 consider sampling period of N=10.

c. write output (sampled value=20) into sample.txt file.

5. Reconstruction

a. Replacing each sample by a sinc function, centered at time of the sample and scaled by the sample value x(nT) times 2fc/fs

b. adding all the sinc functions so created.

c. write output of reconstructed signal into recon.txt.

6. To display original, sampled and reconstructed waveform : use scilab

a. Copy contents of all three txt files(orig.txt, sample.txt, recon.txt) in excel sheets.

b. Read all excel files.

c. While displaying reconstructed signal, use first 200 values then next 200 upto 2001 values which we are getting in recon.txt file.

**Program:**

#include<stdio.h>

#include<math.h>

#include<graphics.h>

void main()

{

int gd = DETECT, gm, option;

int t, A, n, a;

float pi = 3.14, X = 0, y = 0, x[600], f = 10, r;

printf("Enter value of amplitude");

scanf("%d", &A);

printf("enter option");

scanf("%d", &option);

initgraph(&gd, &gm, NULL);

setbkcolor(WHITE);

switch (option)

{

case 1:

for (t = 0; t <= 500; t++)

{

if (t == 0)

{

x[t] = 1;

for (n = 0; n <= 100; n++)

{ y = (200 + n) - x[t] \* 100;

putpixel((X + 300), y, BLACK);

}

X++;

}

else

{

x[t] = 0;

for (n = 0; n <= 100; n++)

{ y = (200) - x[t] \* 100;

putpixel((X + 100), y, BLACK);

}

X++;

}

}

break;

case 2:

line(150, 200, 800, 200);

for (t = 0; t <= 500; t++)

{

if (t >= 0)

{

x[t] = 1;

for (n = 0; n <= 100; n++)

{ y = (200 + n) - x[t] \* 100;

putpixel((X + 300), y, BLACK);

}

X = X + 20;

} else {

x[t] = 0;

for (n = 0; n <= 100; n++)

{ y = (200) - x[t] \* 100;

putpixel((X + 100), y, BLACK);

}

X++;

}

}

break;

case 3:

line(150, 210, 800, 210);

for (t = 0; t <= 200; t++)

{

if (t >= 0)

{

x[t] = t;

a = 10 + t;

for (n = 0; n <= a; n++)

{ y = (200 + n) - x[t] \* 1;

putpixel((t + 300), y, BLACK);

}

t = t + 19;

} else {

x[t] = 0;

for (n = 0; n <= 100; n++)

{ y = (200) - x[t] \* 100;

putpixel((X + 100), y, BLACK);

}

X++;

}

}

break;

case 4:

line(150, 400, 800, 400);

r = 0.2;

for (t = 0; t <= 5; t++)

{

if (t >= 0)

{

x[t] = pow(r, t);

//printf("x[%d] = %f", t , x[t]);

a = 300 \* x[t];

for (n = 0; n <= a; n++)

{ y = (400 + n) - x[t] \* 300;

putpixel((X + 300), y, BLACK);

}

X = X + 19;

}

else

{

x[t] = 0;

for (n = 0; n <= 100; n++)

{ y = (200) - x[t] \* 100;

putpixel((X + 100), y, BLACK);

}

X++;

}

}

break;

default :

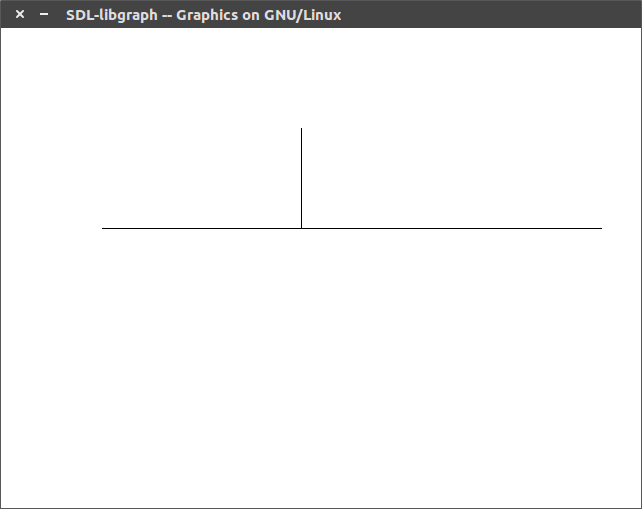
printf("Invalid grade\n" );

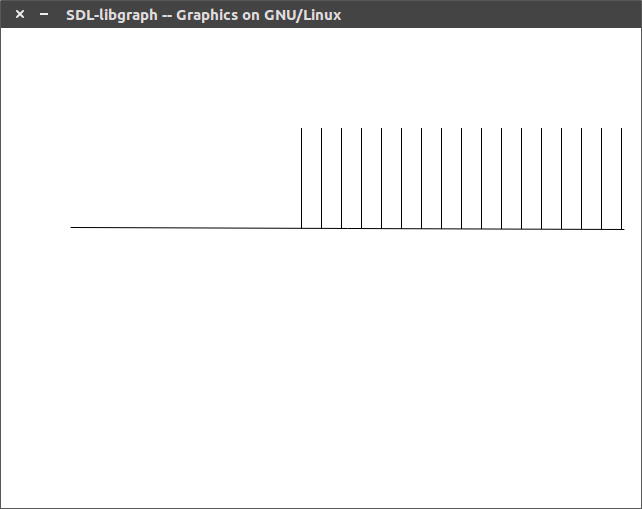
}

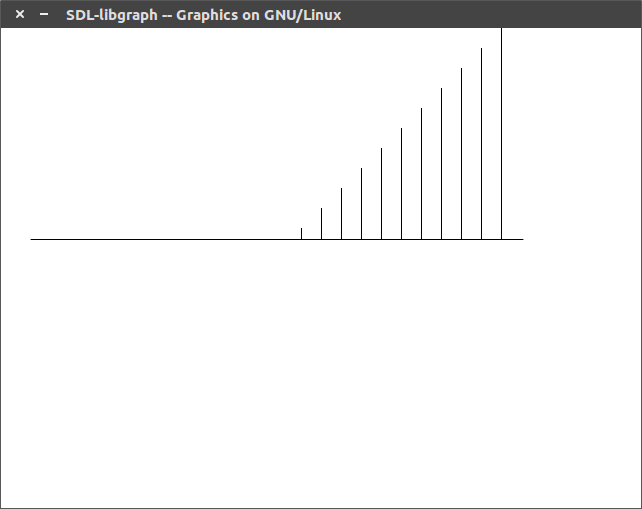
delay(50000);

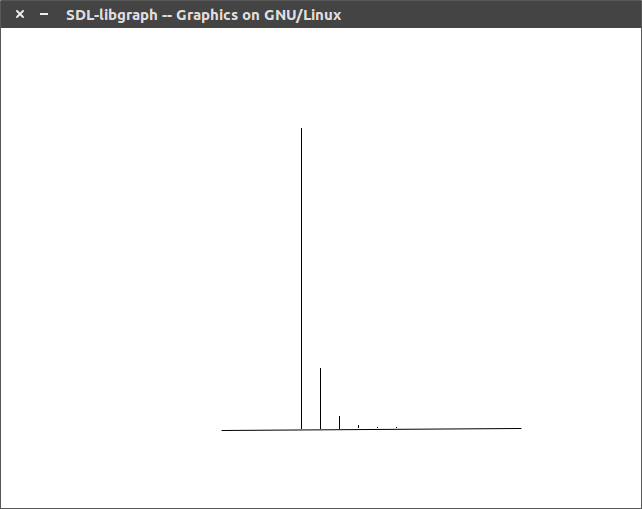
}

**Output:**





****

****

**Conclusion:** Thus, we developed a program to sample a continuous time signal and convert it to Discrete Time Signal & we displayed the result.