National Institute of Technology
Calicut

Department of Electronics and Communication Engineering

Digital Signal Processing LAB

Week 2 Report

Date: 07–08–2025
Instructor: Dr. Sakthivel V

Submitted by:
Muhammed Nihal MP — B230437EC
Mummana Jagdeesh — B231113EC

Aim

To analyze the effects of different sampling rates on the representation and reconstruction of a sine wave. This includes demonstrating undersampling, critical sampling, and oversampling conditions using MATLAB simulations.

Theory

Sampling is the process of converting a continuous-time signal into a discrete-time signal by taking values of its amplitude at uniform intervals. The frequency at which these samples are taken is called the sampling frequency (f_s). According to the Nyquist-Shannon Sampling Theorem, to perfectly reconstruct a signal, the sampling frequency must be at least twice the maximum frequency component present in the signal.

Types of Sampling

- **Undersampling:** Sampling at a frequency less than twice the signal frequency ($f_s < 2f$). This leads to aliasing, where high-frequency components are incorrectly represented.
- **Critical Sampling (Nyquist rate):** Sampling exactly at twice the signal frequency ($f_s = 2f$). This is the minimum requirement for perfect reconstruction.
- **Oversampling:** Sampling at a frequency significantly higher than the Nyquist rate ($f_s > 2f$). This results in smoother signal representation and greater robustness against noise.

Equations

- Continuous signal: $x(t) = A\sin(2\pi ft)$
- Discrete sampled signal: $x[n] = A\sin(2\pi fnT_s)$, where $T_s = 1/f_s$
- Nyquist criterion: $f_s \geq 2f$

MATLAB Code and Explanation

Continuous-Time Sine Wave Generation

MATLAB Code for Continuous Sine Wave

```
clc;
clear all;
close all;
t=0:0.0001:0.1;
A=10;
f=50;
```

```
subplot(3,3,1);
x=A*sin(2*pi*f*t);
ylim([-20 20]);
plot(t,x);
grid on;
title("Sine wave");
xlabel("t");
ylabel("f(t)");
subplot(3,3,2);
fs1=50;
ts=1/fs1;
n=0:1/fs1:0.1;
x1=A*sin(2*pi*f*n);
ylim([-20 20]);
stem(n,x1);
grid on;
title("fs=50Hz");
xlabel("t");
ylabel("f(t)");
subplot(3,3,3);
fs1=100;
ts=1/fs1;
n=0:1/fs1:0.1;
x1=A*sin(2*pi*f*n);
ylim([-20 20]);
stem(n,x1);
grid on;
title("fs=100Hz");
xlabel("t");
ylabel("f(t)");
subplot(3,3,4);
fs1=200;
ts=1/fs1;
n=0:1/fs1:0.1;
x1=A*sin(2*pi*f*n);
ylim([-20 20]);
stem(n,x1);
grid on;
title("fs=200Hz");
xlabel("t");
ylabel("f(t)");
subplot(3,3,5);
fs1=400;
ts=1/fs1;
n=0:1/fs1:0.1;
x1=A*sin(2*pi*f*n);
ylim([-20 20]);
```

```
stem(n,x1);
grid on;
title("fs=400Hz");
xlabel("t");
ylabel("f(t)");
subplot(3,3,6);
fs1=500;
ts=1/fs1;
n=0:1/fs1:0.1;
x1=A*sin(2*pi*f*n);
ylim([-20 20]);
stem(n,x1);
grid on;
title("fs=500Hz");
xlabel("t");
ylabel("f(t)");
subplot(3,3,7);
fs1=1000;
ts=1/fs1;
n=0:1/fs1:0.1;
x1=A*sin(2*pi*f*n);
ylim([-20 20]);
stem(n,x1);
grid on;
title("fs=1000Hz");
xlabel("t");
ylabel("f(t)");
subplot(3,3,8);
fs1=2000;
ts=1/fs1;
n=0:1/fs1:0.1;
x1=A*sin(2*pi*f*n);
ylim([-20 20]);
stem(n,x1);
grid on;
title("fs=2000Hz");
xlabel("t");
ylabel("f(t)");
subplot(3,3,9);
fs1=4000;
ts=1/fs1;
n=0:1/fs1:0.1;
x1=A*sin(2*pi*f*n);
ylim([-20 20]);
stem(n,x1);
grid on;
title("fs=4000Hz");
```

```
xlabel("t");
ylabel("f(t)");
```

Explanation: This code generates a continuous sine wave of 50 Hz using a high-resolution time vector. The high resolution ensures a smooth and accurate representation of the wave.