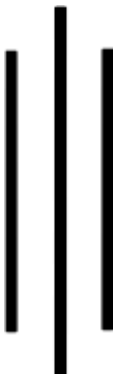
Nepal Engineering College

(Affiliated to Pokhara University)



Report on:

LAB 1: FAMILIARIZATION WITH BASIC DISCRETE-TIME SIGNAL.

Submitted By:

Name: Dipson Thapa

CRN: 020-313

Date: May 26, 2025

Submitted To:

Department Of:

Computer Engineering

FAMILIARIZATION WITH BASIC DISCRETETIME SIGNAL.

Objective

In this lab session, we will familiarize ourselves with basic discrete-time signals using MATLAB. We will write MATLAB programs to plot continuous-time sinusoidal signals of different frequencies and study the different properties of discrete-time sinusoidal signal. We will also write programs to plot various graphs that will illustrate Fourier series theory, some elementary signals as well as know about script and function m-files.

Theory

In signal processing, understanding the behavior of discrete-time signals is fundamental. This lab focuses on plotting and analyzing continuous-time sinusoidal signals and observing their discrete-time counterparts in MATLAB. By working with signals of varying frequencies, students become familiar with how waveform characteristics like amplitude, frequency, and phase affect the visual and analytical properties of a signal. Additionally, MATLAB's built-in functions allow for efficient generation, plotting, and manipulation of such signals, helping students grasp the underlying concepts.

A key part of this experiment is the study of sampling frequency and its effect on signal reconstruction. According to the Nyquist theorem, a signal must be sampled at least twice its frequency (Fs = 2F) to be accurately reconstructed. Sampling at a lower rate causes aliasing—where different signals become indistinguishable—leading to distortion. Oversampling (e.g., Fs = 10F) produces a much smoother signal and helps reduce quantization noise. By visualizing these scenarios in MATLAB, we gain a deeper understanding of the critical role that sampling rate plays in digital signal processing.

In real-world applications such as audio processing, telecommunications, and biomedical signal analysis, proper sampling and reconstruction are vital to ensuring data integrity and signal quality. Engineers rely on these principles to design systems that accurately capture and reproduce signals without distortion or information loss.

Mathematical Questions Including Plots

1. Continuous-time sinusoidal signal

Care must be taken while plotting continuous-time signal in MATLAB because all signals in MATLAB are discrete-time, but they will look like continuous-time signals if the sampling rate is much higher than the Nyquist rate. For example, if we wish to plot continuous-time sinusoid, $x\ t\ Asin\ 2\pi Ft\ \theta$, we must take sampling frequency, $F_s \geq 2F$. The MATLAB code to plot above signal would be:

```
F = 3; A = 2; th = 0;   % Three parameters frequency, amplitude and phase of sinusoid.
Fs = 20*F;              % Taking sampling rate is equal to twenty times the frequency.
Ts = 1/Fs; T=1/F;       % Time period of sampled sequence and sinusoid.
t = 0:Ts:3*T;           % Defining time t from zero to 3 time the time period of sinusoid.
xt = A*cos(2*pi*F*t + th);% Defining the signal x(t)
plot(t,x);              % Plotting the signal t Vs x(t)
title('Continuous time sinusoid);
xlabel('t'); ylabel('x(t)'); grid
```

1a. Create and run the m-file MATLAB Code:

```
F=3;
A=2;
th = 0;
Fs = 20*F;
Ts = 1/Fs;
T= 1/F;
t = 0:Ts:3*T;
xt = A*cos(2*pi*F*t+th);
figure;
plot(t,xt);
title('CRN: 020-313 – Continuous time sinusoid');
xlabel('t');ylabel('x(t)');
grid on;
```

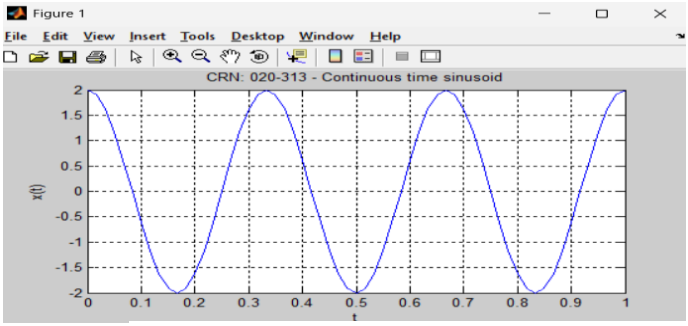


Figure 1Continous-time sinuosoidal signal

This figure shows the MATLAB code and corresponding plot of a continuous-time sinusoidal signal with a frequency of 3 Hz and amplitude of 2.The signal is sampled at 20 Hz and plotted over a time span of 3 periods.

The output plot displays a smooth sinusoidal waveform labeled with time (t) and amplitude (x(t)).

1b. Study the behavior for different sampling frequencies

We varied the sampling frequency as follows:

* Fs = 2*F (Nyquist rate): The waveform was just adequately captured.

```
F=3;
A=2;
th = 0;
Fs = 2*F;
Ts = 1/Fs;
T= 1/F;
t = 0:Ts:3*T;
xt = A*cos(2*pi*F*t+th);
figure;
plot(t,xt);
title('CRN: 020-313 – Continuous time sinusoid');
xlabel('t');ylabel('x(t)');
grid on;
```

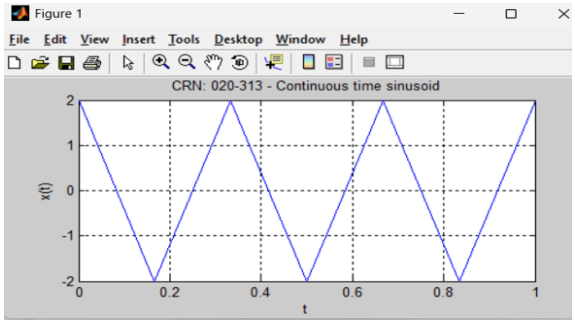


*Figure 2 CT-sinusoidal signal at Fs=2*F*

*Fs = 10*F (Oversampling): The waveform appeared very smooth.

```
F=3;
A=2;
th = 0;
Fs = 10*F;
Ts = 1/Fs;
T= 1/F;
t = 0:Ts:3*T;
xt = A*cos(2*pi*F*t+th);
figure;
plot(t,xt);
title('CRN: 020-313 – Continuous time sinusoid');
xlabel('t');ylabel('x(t)');
grid on;
```

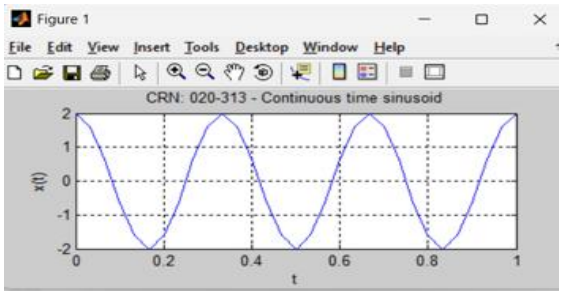


*Figure 3 CT-sinusoidal signal at Fs = 10*F*

*Fs = F (Under-sampling/Aliasing): The waveform appeared distorted due to aliasing.
Plots clearly showed how insufficient sampling leads to signal distortion.

```
F=3;
A=2;
th = 0;
Fs = F;
Ts = 1/Fs;
T= 1/F;
t = 0:Ts:3*T;
xt = A*cos(2*pi*F*t+th);
figure;
plot(t,xt);
title('CRN: 020-313 – Continuous time sinusoid');
xlabel('t');ylabel('x(t)');
grid on;
```

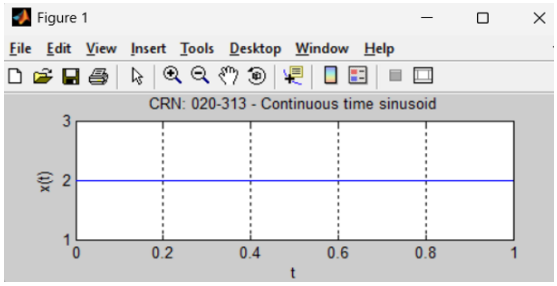


Figure 4 CT-sinusoidal signal at Fs = F

This figure shows how different sampling frequencies affect the shape of a continuoustime sinusoid. At Nyquist rate, the signal is adequately captured; oversampling makes it smooth, and under sampling results in aliasing and distortion. **1c. Plot different frequencies and fixed number of periods**

We tested different values of frequency:

* F = 1: Low frequency, longer wavelength.

```
F=1;
A=2;
th = 0;
Fs = 20*F;
Ts = 1/Fs;
T= 1/F;
t = 0:Ts:3*T;
xt = A*cos(2*pi*F*t+th);
figure;
plot(t,xt);
title('CRN: 020-313 – Continuous time sinusoid');
xlabel('t');ylabel('x(t)');
grid on;
```

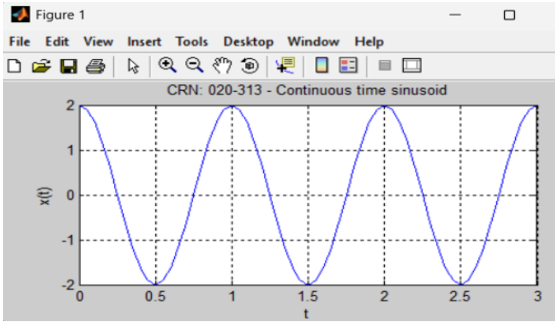


Figure 5 CT-sinusoid signal at F=1

* F = 5: Moderate frequency, shorter wavelength.

```
F=5;
A=2;
th = 0;
Fs = 20*F;
Ts = 1/Fs;
T= 1/F;
t = 0:Ts:3*T;
xt = A*cos(2*pi*F*t+th);
figure;
plot(t,xt);
title('CRN: 020-313 – Continuous time sinusoid');
xlabel('t');ylabel('x(t)');
grid on;
```

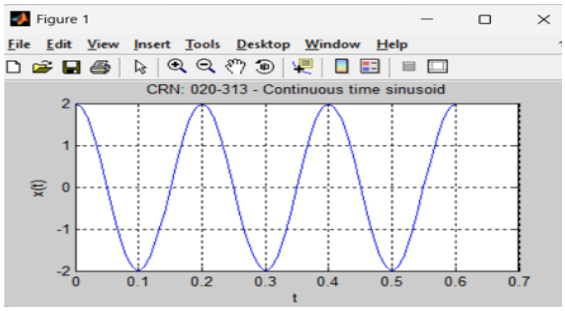


Figure 6 CT-sinusoid signal at F=5

* F = 10: High frequency, very tight oscillations.

```
F=10;
A=2;
th = 0;
Fs = 20*F;
Ts = 1/Fs;
T= 1/F;
t = 0:Ts:3*T;
xt = A*cos(2*pi*F*t+th);
figure;
plot(t,xt);
title('CRN: 020-313 – Continuous time sinusoid');
xlabel('t');ylabel('x(t)');
grid on;
```

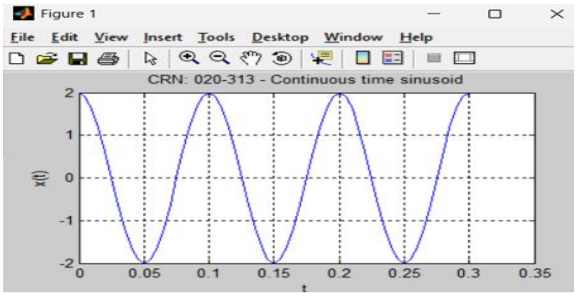


Figure 7 CT=sinusoid signal at F=10

By keeping the time duration fixed, higher frequencies resulted in more cycles in the same duration.

This plot demonstrates how varying the frequency of a sinusoid affects its appearance. As frequency increases, the number of oscillations within a fixed time window increases, resulting in tighter waveforms.

1d. Use help for plot, subplot, axis

We explored the following MATLAB commands:

- `help plot`: Provided detailed usage of the `plot` function.

- `help subplot`: Showed how to organize multiple plots in a single figure window.

- `help axis`: Explained how to set axis limits and aspect ratio manually.

Question 2: Discrete-Time Sinusoidal Signal

A discrete-time sinusoidal signal is expressed as,

$$x\,n \quad A\cos 2\pi f n \quad \theta$$

Where n is an integer variable. For x[n] to be periodic, its frequency f must be rational number i.e. f = k/N, where N is the fundamental period.

Write a MATLAB program to plot discrete-time sinusoid with frequency f = 3/10.

```
A=1;f=3/10;
theta=0;n=0:40;
x=A*cos(2*pi*f*n+theta);
stem(n,x);
title('CRN: 020-313 – Discrete-time sinusoid f = 3/10');
xlabel('n'); ylabel('x[n]');
grid on;
```

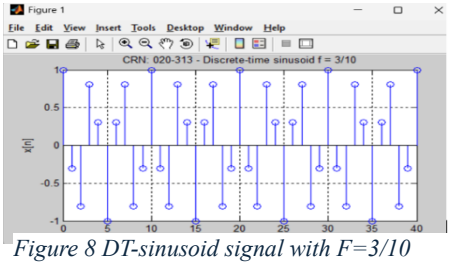


Figure 8 DT-sinusoid signal with F=3/10

2i. Periodicity with rational frequency

A discrete-time sinusoid is periodic only if f is rational (e.g. 3/10).

```
f1=3/10;n=0:39;
x1=cos(2*pi*f1*n);
f2=sqrt(2)/10;
x2=cos(2*pi*f2*n);
subplot(2,1,1);
stem(n,x1);
title('CRN: 020-313 – Periodic Signal (f = 3/10)');
xlabel('n'); ylabel('x[n]');
grid on;

subplot(2,1,2);
stem(n,x2);
title('CRN: 020-313 – Aperiodic Signal (sqrt (2)/10)');
xlabel('n'); ylabel('x[n]');
grid on;
```

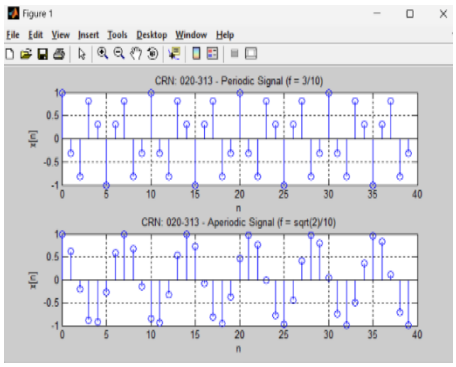


Figure 9 Perioidic and Aperiodic signals

This figure illustrates that discrete-time sinusoids are periodic only when the frequency is a rational number, resulting in repeating patterns.

2ii. Identical signals with frequency difference of integer multiple of 2π

f1 and f2 with f1 - f2 = integer will produce identical signals.

```
f1=3/10;n=0:49;
x1=cos(2*pi*f1*n);
f2=f1+1;
x2=cos(2*pi*f2*n);
subplot(2,1,1);
stem(n,x1);
title('CRN: 020-313 – Signal with f1 = 0.25');
xlabel('n'); ylabel('x1[n]');
grid on;

subplot(2,1,2);
stem(n,x2);
title('CRN: 020-313 – Signal with f1 = 1.25(f1 + 1)');
xlabel('n'); ylabel('x2[n]');
grid on;
```

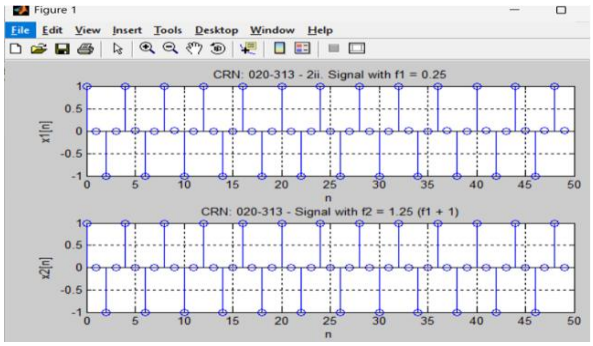


Figure 10 Identical signals with frequency difference

The figure shows that discrete-time sinusoids differing by an integer multiple of 2π are indistinguishable, as they yield identical sequences.

2iii. Maximum oscillation at ω = π

The fastest discrete oscillation occurs when ω = π (e.g. x[n] = cos(πn) = (-1)^n).

```
n=0:20;
x=cos(pi*n);
stem(n,x);
title('CRN 020-313 – 2iii. Maximum Oscillation at \omega = \pi');
xlabel('n'); ylabel('x[n]');
grid on;
```

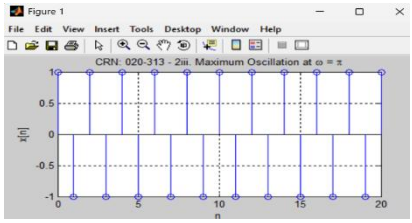


Figure 11 DT sinusoidal signal at max oscillation

This plot shows that when ω = π, the sinusoid oscillates at its maximum rate, alternating between 1 and -1, represented by the sequence (−1)^n.

Question 3: Elementary Discrete-Time Signals

Plot the basic discrete-time signal using MATLAB.

i. Unit Impulse sequence ii. Unit step sequence iii.

Unit Ramp sequence iv. Sinusoidal sequence

v. Exponential sequence

Type “help function” in command window to learn about user-defined function in MATLAB and perform the followings:

Question 3#: Create function m-files of each signals and plot the sequences. Also plot impulse sequence and ramp sequence using step sequence function and step sequence using impulse sequence function.

```
function x = unit_impulse(n)
% CRN: 020-313 - Unit Impulse Function
x = (n == 0);

function x = unit_step(n)
% CRN: 020-313 - Unit Step Function
x = double(n >= 0);

function x = unit_ramp(n)
% CRN: 020-313 - Unit Ramp Function
x = n .* (n >= 0);

function x = sinusoidal_seq(n, f)
% CRN: 020-313 - Sinusoidal Sequence Function
x = sin(2*pi*f*n);

function x = exponential_seq(n, a)
% CRN: 020-313 - Exponential Sequence Function
x = a.^n;
```

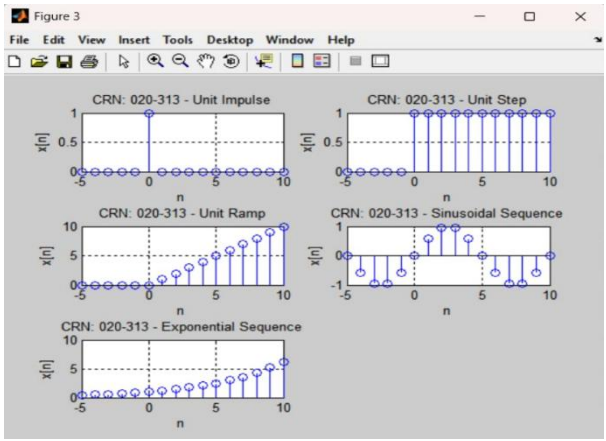


Figure 12 impulse, step, ramp, sin_seq and exp signals

```
% Question 3#: Elementary Discrete-Time Signals
% CRN: 020-313

n = -5:10;

% i. Unit Impulse Sequence
x_impulse = unit_impulse(n);

% ii. Unit Step Sequence
x_step = unit_step(n);

% iii. Unit Ramp Sequence
x_ramp = unit_ramp(n);

% iv. Sinusoidal Sequence
f = 0.1; % Frequency in cycles/sample
x_sin = sinusoidal_seq(n, f);

% v. Exponential Sequence
a = 1.2;
x_exp = exponential_seq(n, a);

% Plot All Signals
figure;
subplot(3,2,1);
stem(n, x_impulse); title('CRN: 020-313 - Unit Impulse'); xlabel('n'); ylabel('x[n]'); grid on;

subplot(3,2,2);
```

```
stem(n, x_step); title('CRN: 020-313 - Unit Step'); xlabel('n'); ylabel('x[n]'); grid on;

subplot(3,2,3);
stem(n, x_ramp); title('CRN: 020-313 - Unit Ramp'); xlabel('n'); ylabel('x[n]'); grid on;

subplot(3,2,4);
stem(n, x_sin); title('CRN: 020-313 - Sinusoidal Sequence'); xlabel('n'); ylabel('x[n]'); grid on;

subplot(3,2,5);
stem(n, x_exp); title('CRN: 020-313 - Exponential Sequence'); xlabel('n'); ylabel('x[n]'); grid on;

% Derived Signals from Functions

% Ramp from Step
ramp_from_step = cumsum(x_step);

% Step from Impulse
step_from_impulse = cumsum(x_impulse);

% Impulse from Step (1st difference)
impulse_from_step = [x_step(1), diff(x_step)];

figure;
subplot(3,1,1);
stem(n, ramp_from_step); title('CRN: 020-313 - Ramp from Step'); xlabel('n'); ylabel('x[n]'); grid on;

subplot(3,1,2);
stem(n, step_from_impulse); title('CRN: 020-313 - Step from Impulse'); xlabel('n'); ylabel('x[n]'); grid on;

subplot(3,1,3);
stem(n, impulse_from_step); title('CRN: 020-313 - Impulse from Step'); xlabel('n'); ylabel('x[n]'); grid on;
```

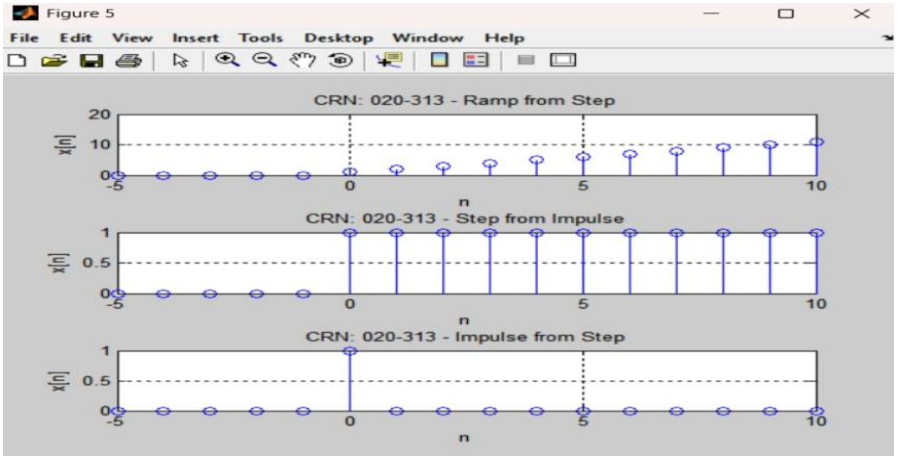


Figure 13 Elementary Discrete Time Signals

These plots show how some signals like ramp and step can be generated from impulse functions, reinforcing the interrelationship between elementary sequences.

4. Fourier series

Fourier series theory states that a periodic wave can be represented as a summation of sinusoidal waves with different frequencies, amplitudes and phase values.

a) Write a program that plots the signal x(t) $x(t) = \sum_{\substack{n=1 \\ (n\ odd)}}^{N} \frac{\sin 2\pi nt}{n}$ for N = 9.

```
% Lab 1 - Question 4#: Fourier Series Approximation
% CRN: 020-313
t = 0:0.001:1; % Time vector
%% a. Fourier Series Approximation for N = 9
N = 9;
x = zeros(size(t));
for n = 1:2:N % Odd harmonics only
    x = x + sin(2*pi*n*t)/n;
end

figure;
plot(t, x);
title('CRN: 020-313 - Fourier Series Approximation (N = 9)');
xlabel('t'); ylabel('x(t)');
grid on;
```

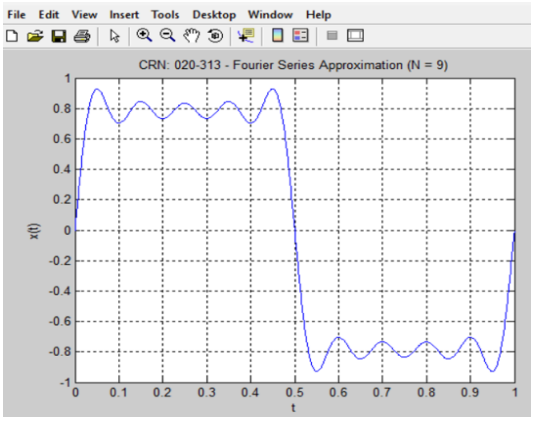


Figure 14 Fourier Series Approximation (N=9)

b. Write a program that plots the signal x(t) in step (a) but with N = 99 and N = 999.

```
 %% b. Fourier Series Approximation for N = 99 and N = 999
t = 0:0.001:1; % Time vector
% N = 99
N = 99;
x_99 = zeros(size(t));
for n = 1:2:N
    x_99 = x_99 + sin(2*pi*n*t)/n;
end

% N = 999
N = 999;
x_999 = zeros(size(t));
for n = 1:2:N
    x_999 = x_999 + sin(2*pi*n*t)/n;
end

figure;
subplot(2,1,1);
plot(t, x_99);
title('CRN: 020-313 - Fourier Series Approximation (N = 99)');
xlabel('t'); ylabel('x(t)');
grid on;

subplot(2,1,2);
plot(t, x_999);
title('CRN: 020-313 - Fourier Series Approximation (N = 999)');
xlabel('t'); ylabel('x(t)');
grid on;
```

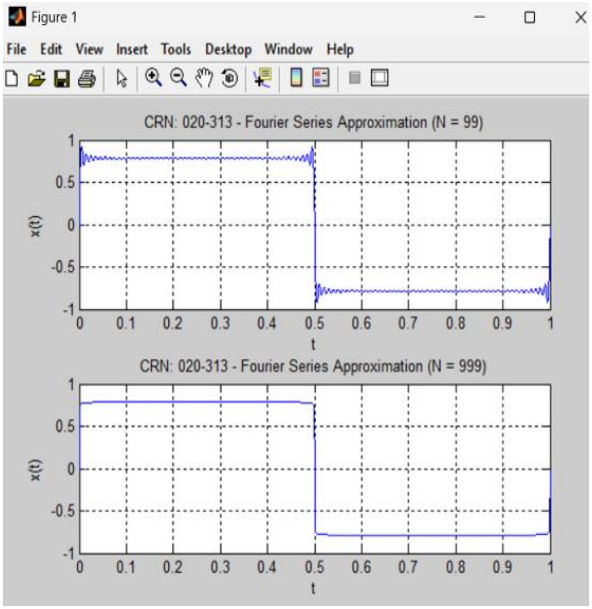


Figure 15Fourier Series Approximation(N = 99 & N = 999)

c. What do you conclude from steps (a) & (b)?

As N increases, the approximation becomes closer to a true square wave. Higher harmonics sharpen the edges, and the waveform shows Gibbs phenomenon near discontinuities."-

d. Write a program to plot square wave. (hint: use square() function)

```
%% d. Square Wave Using Built-in Function
t = 0:0.001:1; % Time vector
f = 5; % Frequency in Hz
x_sq = square(2*pi*f*t);

figure;
plot(t, x_sq);
title('CRN: 020-313 - Square Wave Using square()');
xlabel('t'); ylabel('x(t)');
grid on;
```

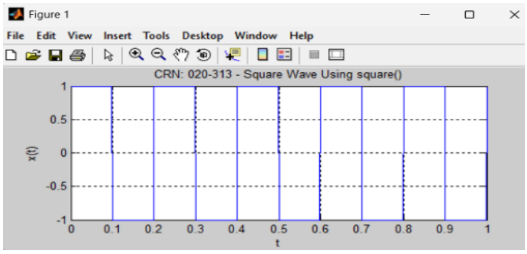


Figure 16 Square Wave

These figures demonstrate how increasing the number of harmonics (N) in a Fourier series improves the approximation of a square wave. Higher N values result in sharper transitions and more accurate waveforms but introduce Gibbs phenomenon near discontinuities.

5. Other Signals:

A normally distributed random signal of length N can be generated using the MATLAB command randn(1,N):

a)Write a program to generate a normally distributed random signal of length 100 samples.

```
%% a. Normally Distributed Random Signal (Length = 100)
N = 100;
x_rand = randn(1, N);
figure;
plot(x_rand);
title('CRN: 020-313 - Normally Distributed Random Signal (N =
100)');
xlabel('Sample Index'); ylabel('Amplitude');
grid on;
```

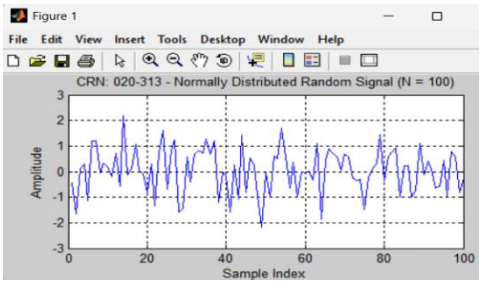


Figure 17 Random Signal

This plot represents a signal with samples drawn from a normal distribution, commonly used to model random noise in systems.

b)Write a program to generate noisy sinusoidal signal. (hint: add sine wave and noise)

```
%% b. Noisy Sinusoidal Signal
n = 0:N-1;
f = 0.05; % Frequency in cycles/sample
x_clean = sin(2*pi*f*n);        % Clean sine wave
noise = 0.3 * randn(1, N);      % Gaussian noise
x_noisy = x_clean + noise;      % Add noise to signal

figure;
plot(n, x_noisy);
title('CRN: 020-313 - Noisy Sinusoidal Signal');
xlabel('n'); ylabel('x[n]');
grid on;
```

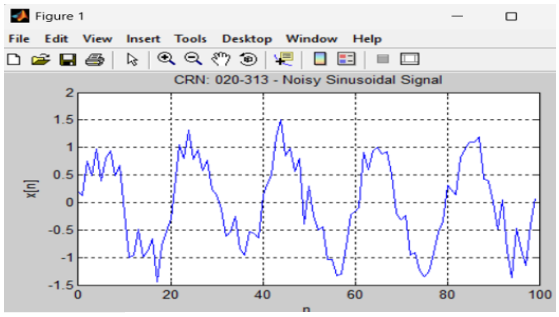


Figure 18 Noisy Sinusoidal Signal

This figure shows a sinusoidal waveform corrupted by random noise, demonstrating how noise affects signal clarity and can be used for noise analysis.

c)Compute and plot even and odd parts of the signal obtained in b.

```
% c. Even and Odd Parts of the Noisy Signal
x_rev = fliplr(x_noisy);              % Time-reversed version
x_even = 0.5 * (x_noisy + x_rev);     % Even part
x_odd  = 0.5 * (x_noisy - x_rev);     % Odd part
subplot(2,1,1);
plot(n, x_even);
title('CRN: 020-313 - Even Part of Noisy Signal');
xlabel('n'); ylabel('x_{even}[n]');
grid on;

subplot(2,1,2);
plot(n, x_odd);
title('CRN: 020-313 - Odd Part of Noisy Signal');
xlabel('n'); ylabel('x_{odd}[n]');
grid on;
```

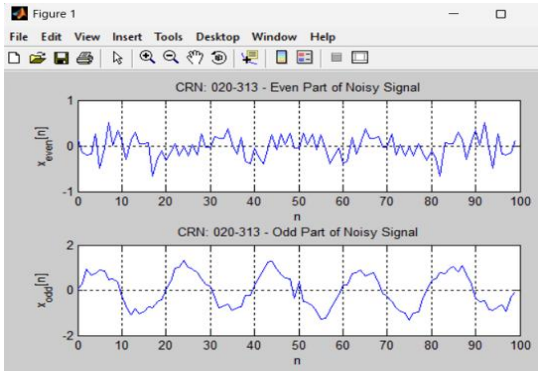


Figure 19 Even and Odd parts of Noisy signal

This plot separates a signal into its even and odd components, helping to understand signal symmetry and their use in signal decomposition.

Conclusion

This lab offered valuable hands-on experience with MATLAB for visualizing and understanding key signal processing concepts. By generating and analyzing both discrete and continuous-time signals, we gained practical insights into the effects of sampling rates, signal periodicity, and basic signal construction. Exploring topics like the Fourier series and the impact of noise helped bridge the gap between theory and real-world applications. Overall, the simulations made complex ideas more intuitive and highlighted the importance of these fundamentals in modern signal processing.