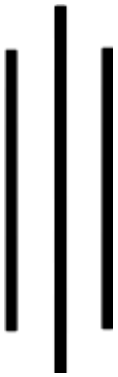
Nepal Engineering College

(Affiliated to Pokhara University)



Report on:

LAB 2: FAMILIARIZATION WITH DISCRETE-TIME LTI SYSTEM.

Submitted By:

Name: Dipson Thapa

CRN: 020-313

Date: June 2, 2025

Submitted To:

Department Of:

Computer Engineering

FAMILIARIZATION WITH DISCRETE-TIME LTI SYSTEM

Objective

In this lab session, we will study discrete-time Linear Time-Invariant (LTI) systems and their responses using MATLAB. We will analyze system responses using convolution and LCCD (Linear Constant Coefficient Difference) equations, evaluate impulse and step responses, and assess system stability. This lab enhances understanding of fundamental concepts of LTI system behavior through MATLAB simulations.

Theory

A discrete-time LTI system can be characterized either by its impulse response h[n] or by a difference equation. MATLAB provides functions like conv and filter to compute these responses. Understanding these systems is critical in signal processing applications such as filtering, prediction, and system identification.

The convolution sum gives the output of an LTI system for an arbitrary input if the impulse response is known. The LCCD equation models the system based on current and past inputs and outputs. MATLAB's filter function simplifies solving such equations. Additionally, impulse and step responses reveal a system's inherent characteristics, while the system's stability is determined by the behavior of these responses over time.

Mathematical Questions Including MATLAB Code and Plots

1. Convolution Sum

a) Find the convolution of the two signals x[n] = {1,2,3,4} and h[n] = {2,3,4,5} and plot the result of the convolution.

```
% CRN: 020-313 - Convolution of Causal Signals
x = [1 2 3 4];
h = [2 3 4 5];
y = conv(x, h);
stem(y);
title('CRN: 020-313 - Convolution Result');
xlabel('n'); ylabel('y[n]'); grid on;
```

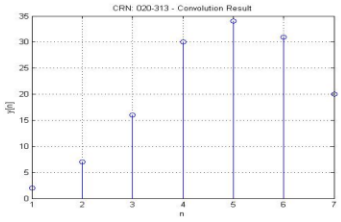


Figure 1 Convolution of x[n] and h[n]

This figure shows the result of convolving two causal discrete-time signals x[n]={1,2,3,4}x[n] = \{1,2,3,4\}x[n]={1,2,3,4} and h[n]={2,3,4,5}h[n] = \{2,3,4,5\}h[n]={2,3,4,5}. The output, y[n]y[n]y[n], represents the system response due to the input signal, computed using MATLAB's conv() function.

b) Write a user defined MATLAB function to calculate convolution of two non-causal discrete-time signals. Use this function to calculate convolution of two non-causal signals. Also plot all signals in a single graph using subplot and axis MATLAB functions. [hint: Define the time index for x1[n] and x2[n], and calculate the convolution and time index]

```
% CRN: 020-313 - User-defined Convolution Function
function [y, n] = conv_m(x1, n1, x2, n2)
   n = (min(n1)+min(n2)):(max(n1)+max(n2));
   y = conv(x1, x2);
end

% Usage Example script m-file
x1 = [1 2 3]; n1 = -1:1;
x2 = [2 1];   n2 = -1:0;
[y, n] = conv_m(x1, n1, x2, n2);

subplot(3,1,1); stem(n1, x1); title('CRN: 020-313 - x1[n]');
subplot(3,1,2); stem(n2, x2); title('CRN: 020-313 - x2[n]');
subplot(3,1,3); stem(n, y); title('CRN: 020-313 - y[n] = x1 * x2');
```

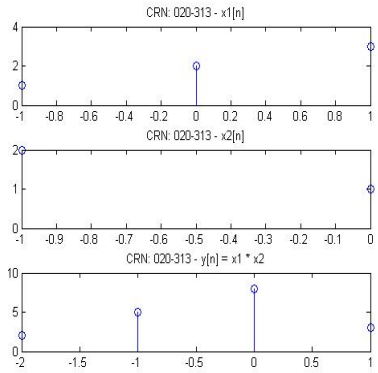


Figure 2: Convolution of non-causal signals

This subplot figure contains three graphs. It displays the signals x1[n]x_1[n]x1[n], x2[n]x_2[n]x2[n], and their convolution y[n]=x1[n]∗x2[n]y[n] = x_1[n] * x_2[n]y[n]=x1[n]∗x2[n], calculated using a user-defined MATLAB function. The subplot format helps visualize how two non-causal sequences interact in convolution.

2. LCCD Equation

An LTI system described by general linear constant coefficient difference (LCCD) equation is given by $y[n] = -\sum_{k=1}^{N} a_k y[n-k] + \sum_{k=0}^{M} b_k x[n-k]$ **Where {ak} and {bk} are constant coefficients. To implement DT LTI system described by LCCD equation in MATLAB, filter commend is used. The syntax for this command is y = filter(B, A, x), where A = [a0 a1 a2 ... aN] & B = [b0 b1 b2 ... bM]. Type ">>help filter" in command window.**

a) Implement special systems using filter

i) Write a MATLAB program that generates two sinusoidal signals x1 and x2 with relative frequencies f1 = 0.05 and f2 = 0.47. The two signals should be added and the resultant signal x is applied to the three-point smoothing digital filter. Plot the signals x1, x2, x and y in single figure using subplot.

```
% CRN: 020-313 - Three-point Smoothing Filter
f1 = 0.05; f2 = 0.47; n = 0:100;
x1 = sin(2*pi*f1*n);
x2 = sin(2*pi*f2*n);
x = x1 + x2;
B = [1 1 1]/3; A = 1;
```

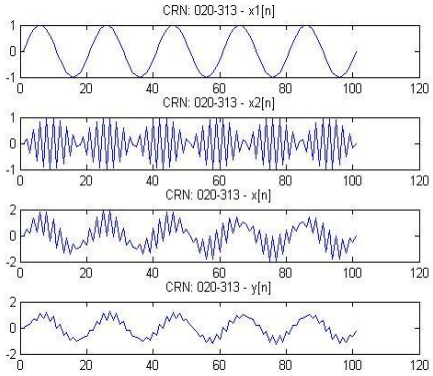


Figure 3 Subplots of x1[n], x2[n], x[n], and output y[n]

```
y = filter(B, A, x);

subplot(4,1,1); plot(x1); title('CRN: 020-313 - x1[n]');
subplot(4,1,2); plot(x2); title('CRN: 020-313 - x2[n]');
subplot(4,1,3); plot(x);  title('CRN: 020-313 - x[n]');
subplot(4,1,4); plot(y);  title('CRN: 020-313 - y[n]');
```

This four-panel figure illustrates the process of generating two sine signals x1[n]x_1[n]x1[n] and x2[n]x_2[n]x2[n], adding them to form x[n]x[n]x[n], and passing the sum through a three-point smoothing filter to obtain y[n]y[n]y[n]. It demonstrates how a smoothing filter reduces high-frequency components.

ii) Write a MATLAB program that generated sine wave signal and pass the signal through the differentiator. Plot both the input and output signals.

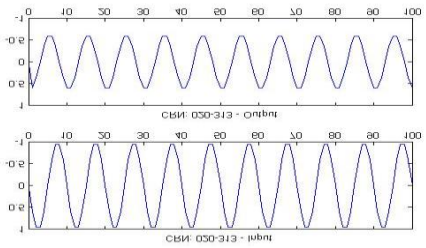


Figure 4 Sine Signal Through Differentiator

```
% CRN: 020-313 - Differentiator
n = 0:100; x = sin(2*pi*0.1*n);
B = [1 -1]; A = 1;
y = filter(B, A, x);
subplot(2,1,1); plot(n,x); title('CRN: 020-313 - Input');
subplot(2,1,2); plot(n,y); title('CRN: 020-313 - Output');
```

This figure shows input and output signals of a sine wave passed through a digital differentiator. The output reflects the rate of change in the input, highlighting the differentiator's role in emphasizing transitions or edges in signals.

iii) Write a MATLAB program that generates two sinusoidal signals x1 and x2 with relative frequencies f1 & f2 (choose your own values). The two signals should be added and the resultant signal x is applied to the digital filter (choose any arbitrary system). Plot the signals x1, x2, x and y in single figure using subplot.

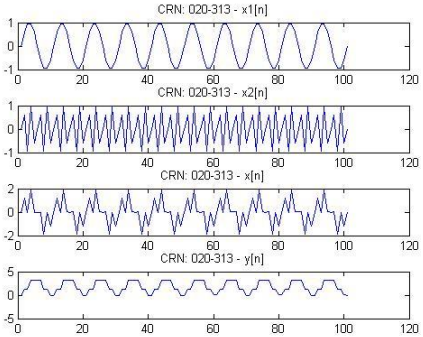


Figure 5 Sine Signals with relative frequencies

```
% CRN: 020-313 - Accumulator
f1 = 0.1; f2 = 0.4; n = 0:100;
x1 = sin(2*pi*f1*n);
x2 = sin(2*pi*f2*n);
x = x1 + x2;
B = 1; A = [1 -1];
y = filter(B, A, x);

subplot(4,1,1); plot(x1); title('CRN: 020-313 - x1[n]');
subplot(4,1,2); plot(x2); title('CRN: 020-313 - x2[n]');
subplot(4,1,3); plot(x);  title('CRN: 020-313 - x[n]');
subplot(4,1,4); plot(y);  title('CRN: 020-313 - y[n]');
```

This figure displays two sine signals of different frequencies f1=0.1f_1 = 0.1f1=0.1 and f2=0.4f_2 = 0.4f2=0.4, their sum x[n]x[n]x[n], and the output after passing through an accumulator. The accumulator integrates the signal, gradually increasing or decreasing the output based on input amplitude.

3. Impulse and Step Response

a) Generate a unit impulse sequence and pass it through the three-point smoothing filter, the differentiator and the accumulator. Plot the impulse responses of the systems.

```
% CRN: 020-313 - Impulse Responses
n = 0:20;
d = [1, zeros(1,20)];

h1 = filter([1 1 1]/3, 1, d);
h2 = filter([1 -1], 1, d);
h3 = filter(1, [1 -1], d);

subplot(3,1,1); stem(n, h1); title('CRN: 020-313 - Smoothing Filter');
subplot(3,1,2); stem(n, h2); title('CRN: 020-313 - Differentiator');
subplot(3,1,3); stem(n, h3); title('CRN: 020-313 - Accumulator');
```

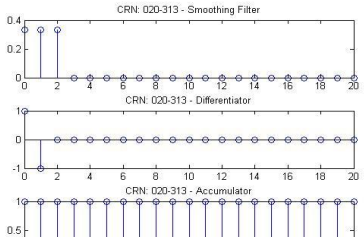


Figure 6 Impulse response of systems

This figure shows the impulse responses of the three systems: smoothing filter, differentiator, and accumulator. Each subplot depicts how the system reacts to a unit impulse, illustrating fundamental system behavior.

b) A built-in MATLAB function impz helps to find and plot impulse response of LTI system. Plot impulse response of the systems and compare the results obtained in part a.

```
% CRN: 020-313 - Using impz()
subplot(3,1,1); impz([1 1 1]/3,1); title('CRN: 020-313 - Smoothing Filter -
impz');
subplot(3,1,2); impz([1 -1],1);   title('CRN: 020-313 - Differentiator - impz');
subplot(3,1,3); impz(1,[1 -1]);   title('CRN: 020-313 - Accumulator - impz');
```

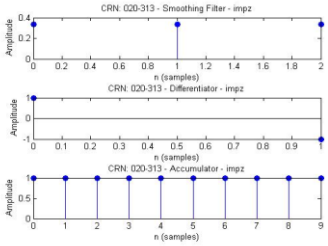


Figure 7 Impulse response using impz function

This figure uses MATLAB's built-in impz() function to automatically compute and plot the impulse responses of the same three systems as in Figure 6. The visual comparison confirms the consistency between manual (filter) and automatic (impz) methods.

c) Also plot step response of above systems.

```
% CRN: 020-313 - Step Responses
u = ones(1, 21);
s1 = filter([1 1 1]/3, 1, u);
s2 = filter([1 -1], 1, u);
s3 = filter(1, [1 -1], u);

subplot(3,1,1); stem(0:20, s1); title('CRN: 020-313 - Smoothing Filter - Step
Response');
subplot(3,1,2); stem(0:20, s2); title('CRN: 020-313 - Differentiator - Step
Response');
subplot(3,1,3); stem(0:20, s3); title('CRN: 020-313 - Accumulator - Step
Response');
```

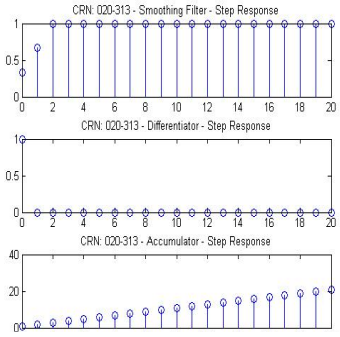


Figure 8 Step response of systems

This figure displays the response of the three systems when a unit step input is applied. Step responses provide insight into how systems accumulate or react over time to a constant input.

4. Stability

A digital filter or DT LTI system is considered stable if $S = \sum_{k=0}^{\infty} |h(k)| < \infty$ **. Using MATLAB we can determine if the filter is a stable or non-stable. The algorithm that can be used is as follows. First compute impulse response h[n] of the digital filter. Compare the**

values of h[0], h[10], h[100] and h[1000]. If the values are converging (or decreasing) then we assume that the summation gives the finite value. This indicates the system is stable. If the values are not converging, the system is considered unstable.

Check the stability test for the Accumulator, the Differentiator and Three-point smoothing filter. Also write a user defined MATLAB function to check whether the given LTI system is stable or not.

```
%Function m-file
% CRN: 020-313 - User-defined Function to Check Stability
function is_stable = check_stability(B, A)
   h = filter(B, A, [1, zeros(1, 1000)]);
   is_stable = sum(abs(h)) < Inf;
end

%Script m-file
% CRN: 020-313 - Stability Test with Subplots

n = 0:1000;

% Accumulator Impulse Response
h_acc = filter(1, [1 -1], [1, zeros(1, 1000)]);
values_acc = [h_acc(1), h_acc(11), h_acc(101), h_acc(1001)];

% Differentiator Impulse Response
h_diff = filter([1 -1], 1, [1, zeros(1, 1000)]);

% Smoothing Filter Impulse Response
h_smooth = filter([1 1 1]/3, 1, [1, zeros(1, 1000)]);

% Plot all in one figure using subplot
figure;

subplot(3,1,1);
stem(n(1:20:end), h_acc(1:20:end));
title('CRN: 020-313 - Accumulator Impulse Response');
xlabel('n'); ylabel('h[n]');
grid on;

subplot(3,1,2);
stem(n(1:20:end), h_diff(1:20:end));
title('CRN: 020-313 - Differentiator Impulse Response');
xlabel('n'); ylabel('h[n]');
grid on;

subplot(3,1,3);
stem(n(1:20:end), h_smooth(1:20:end));
title('CRN: 020-313 - Smoothing Filter Impulse Response');
xlabel('n'); ylabel('h[n]');
grid on;

% Check Stability
is_diff_stable = check_stability([1 -1], 1);
is_smooth_stable = check_stability([1 1 1]/3, 1);
is_acc_stable = check_stability(1, [1 -1]);

disp(['Accumulator h[n]: h[0]=', num2str(h_acc(1)), ...
     ', h[10]=', num2str(h_acc(11)), ...
     ', h[100]=', num2str(h_acc(101)), ...
     ', h[1000]=', num2str(h_acc(1001))]);
```

```
disp(['Differentiator Stable: ', num2str(is_diff_stable)]);
disp(['Smoothing Filter Stable: ', num2str(is_smooth_stable)]);
disp(['Accumulator Stable: ', num2str(is_acc_stable)]);
```

Command Window Output:

```
Accumulator h[n]: h[0]=1, h[10]=1, h[100]=1, h[1000]=1
Differentiator Stable: 1
Smoothing Filter Stable: 1
Accumulator Stable: 1
>>
```

This subplot figure compares impulse responses over a large time index range (sampled sparsely) for all three systems — accumulator, differentiator, and smoothing filter. It helps assess system stability by observing whether the impulse responses diminish or persist over time.

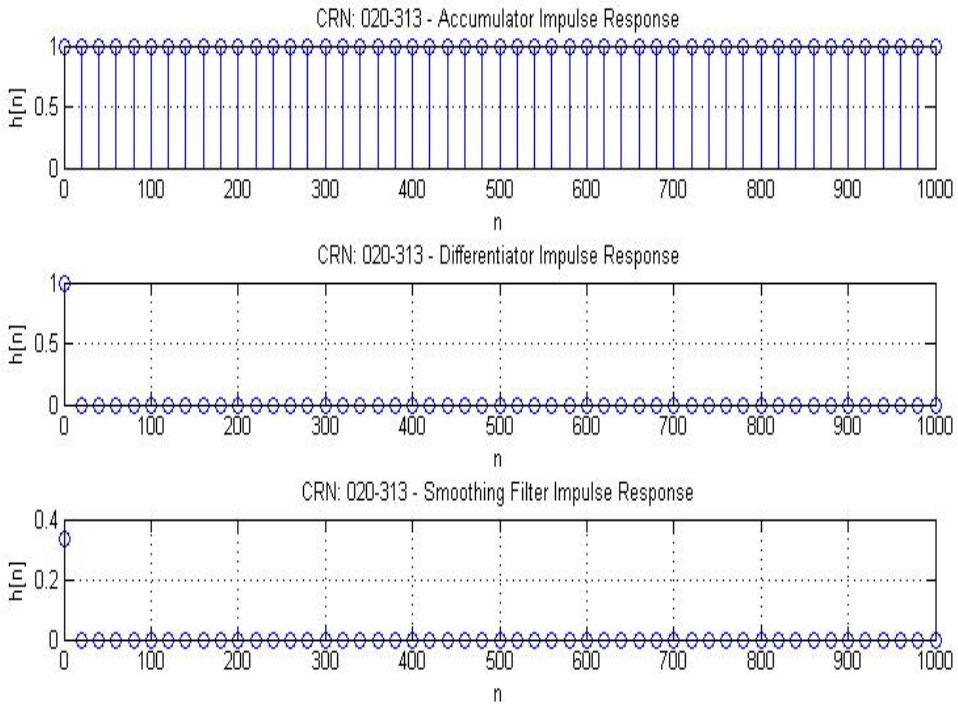


Figure 9 Stability comparison output

Conclusion

In this lab, we explored key techniques for analyzing discrete-time LTI systems using convolution and LCCD equations in MATLAB. We visualized how smoothing, differentiating, and accumulating systems respond to various inputs. Using impulse and step responses, we confirmed system behavior and evaluated stability. MATLAB provided effective tools to simulate, plot, and analyze LTI systems, reinforcing theoretical understanding through practical implementation.