Department of Electrical Engineering
Indian Institute of Technology Kharagpur

Digital Signal Processing Laboratory (EE39203)

Autumn, 2022-23

Experiment 1 **Discrete and Continuous Time Signals**

Slot: Date:

Student Name: Roll No.:

Grading Rubric

	Tick the best applicable per row			Points
	Below Expectation	Lacking in Some	Meets all Expectation	
Completeness of the report				
Organization of the report (5 pts) *With cover sheet, answers are in the same order as questions in the lab, copies of the questions are included in report, prepared in LaTeX*				
Quality of figures (5 pts) *Correctly labelled with title, x-axis, y-axis, and name(s)*				
Understanding of continuous and discrete-time signals (15 pts) *Matlab figures, questions*				
Ability to compute integral manually and in Matlab (30 pts) *Manual computation, Matlab figures, Matlab codes, questions*				
Ability to define and display functions (1D and 2D) (30 pts) *Matlab figures, Matlab codes, questions*				
Understanding of sampling (15 pts) *Matlab figures, questions*				
			TOTAL (100 pts)	

Total Points (100): TA Name: TA Initials:

Digital Signal Processing Laboratory (EE39203)
Experiment 2: Discrete Time Systems

Name: Sujay Vivek
Roll Number: 22EE30029

August 14th 2024

1. Learning Objective

In this lab, we examine the core principles of discrete-time signal processing by carrying out and assessing fundamental operations such as differentiation and integration. Discrete-time systems play a vital role in digital signal processing, as they facilitate the modification and conversion of signals for various purposes, including communication technologies, control systems, and multimedia applications. Discrete systems have a finite set of states. This allows for efficient problem-solving and exploration across various domains.

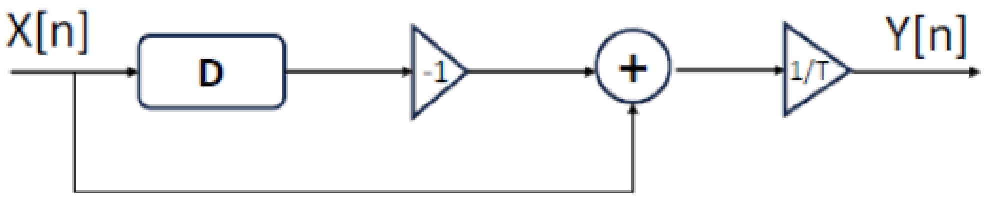
2. Example of Discrete-Time Systems

These digital systems can provide higher quality and/or lower cost through the use of standardized, high-volume digital processors. The following two continuous-time systems are commonly used in electrical engineering:

Differentiator: y$(t) = \frac{d}{dt}x(t)$
y$(n) = \frac{(x(n) - (x(n-1)))}{T}$
Similarly, Integrator: y$(t) = \int_{-inf}^{t} x(t)$
y$(n) = y(n-1) + T \cdot x(n)$

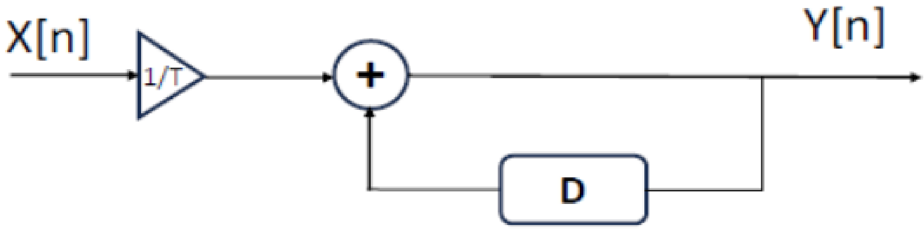
2.1 Block Diagram for Integrator and Differentiator

Differentiator:



(a) Plot of Differentiator

Integrator:



(a) Plot of Integrator

2.2 Applying both Differentiator and Integrator

Taking N = 10 and $-10 <= n <= 20$

MATLAB Code

```
%Defining the parameters
N = 10;
n = -10:20;
T = 1;

%Generating the step function
u = @(n) double(n >=0);

%Generating the required signal now
signal = u(n) - u(n - (N+1));

%Applying the differentiator
diff_sig = zeros(size(signal));

for i = 2:length(n)
    diff_sig(i) = (signal(i) - signal(i-1))/T;
end

%Applying the integrator
inte_sig = zeros(size(signal));
```

```
for i = 2:length(n)
    inte_sig(i) = inte_sig(i-1) + signal(i)*T;

end

%Plotting
subplot(3,1,1);
stem(n, signal);
title('Original Signal u[n] - u[n-(N+1)]');
xlabel('n');
ylabel('Amplitude');

subplot(3,1,2);
stem(n, diff_sig);
title('Differentiated Signal');
xlabel('n');
ylabel('Amplitude');

subplot(3,1,3);
stem(n, inte_sig);
title('Integrated Signal');
xlabel('n');
ylabel('Amplitude');

sgtitle('Name: Sujay Vivek - Roll: 22EE30029');
```

Plots

Plotting with the help of MATLAB:

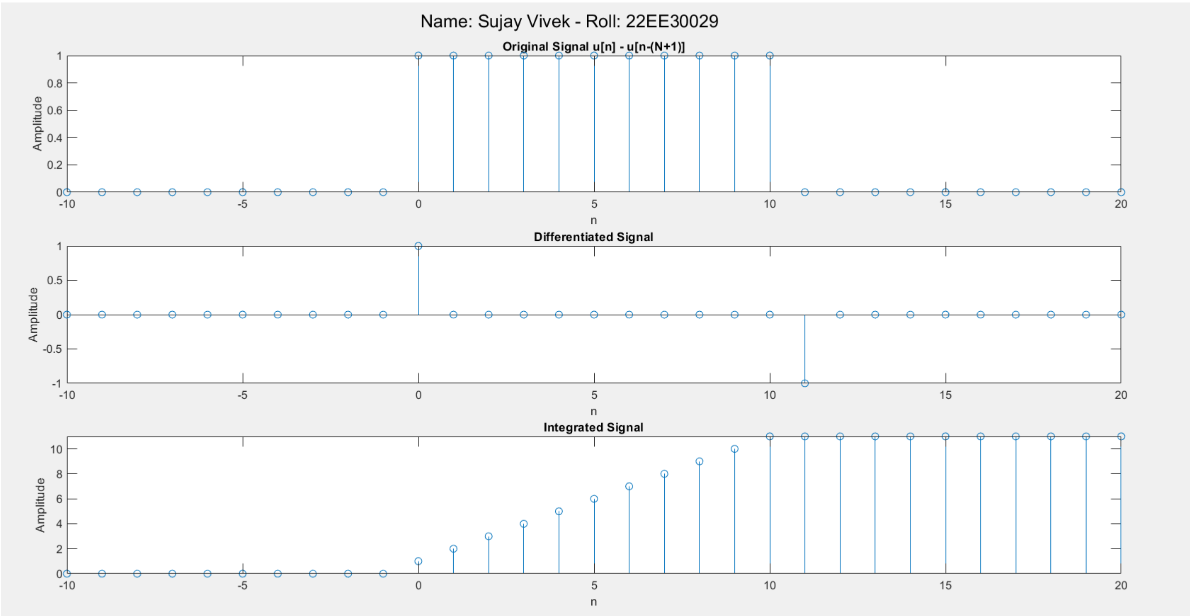
2.3 Using Discrete-Time Differentiator to evaluate x(t)

Trying for both T = 0.1 and T = 0.001

MATLAB Code

```
%Defining the function
x = @(t) sin(2*pi*t);

```



(a) Plot after Applying both Integrator and Differentiator to Input Signal

```
%Defining the time intervals
T1 = 0.1;
T2 = 0.01;

%Defining the discrete time values
t1 = 0:T1:10;
t2 = 0:T2:10;

%Computing x(t) for both time intervals
x_t1 = x(t1);
x_t2 = x(t2);

%Computing the derivative of the signal

%Creating array of zeros of size t1 AND t2
dx_t1 = zeros(size(t1));
dx_t2 = zeros(size(t2));

%Computing for both
for i = 2:length(t1)
    dx_t1(i) = (x_t1(i) - x_t1(i-1))/T1;
```

```
end

for i = 2:length(t2)
    dx_t2(i) = (x_t2(i) - x_t2(i-1))/T2;
end

% Comparing real derivative of x(t)
dx_compare = @(t) 2*pi*cos(2*pi*t);

% Plot the results
figure;

subplot(3,1,1);
plot(t1, dx_t1, 'b', t1, dx_compare(t1), 'r--');
title('Derivative with T = 0.1');
xlabel('t');
ylabel('dx/dt');
legend('sin(2*pi*T)', '2*pi*cos(2*pi*T)');

subplot(3,1,2);
plot(t2, dx_t2, 'b', t2, dx_compare(t2), 'r--');
title('Derivative with T = 0.001');
xlabel('t');
ylabel('dx/dt');
legend('sin(2*pi*T)', '2*pi*cos(2*pi*T)');

subplot(3,1,3);
plot(t2, dx_t2 - dx_compare(t2));
title('Error between Numerical and Comparitive Derivatives for T = 0.001');
xlabel('t');
ylabel('Error');
```

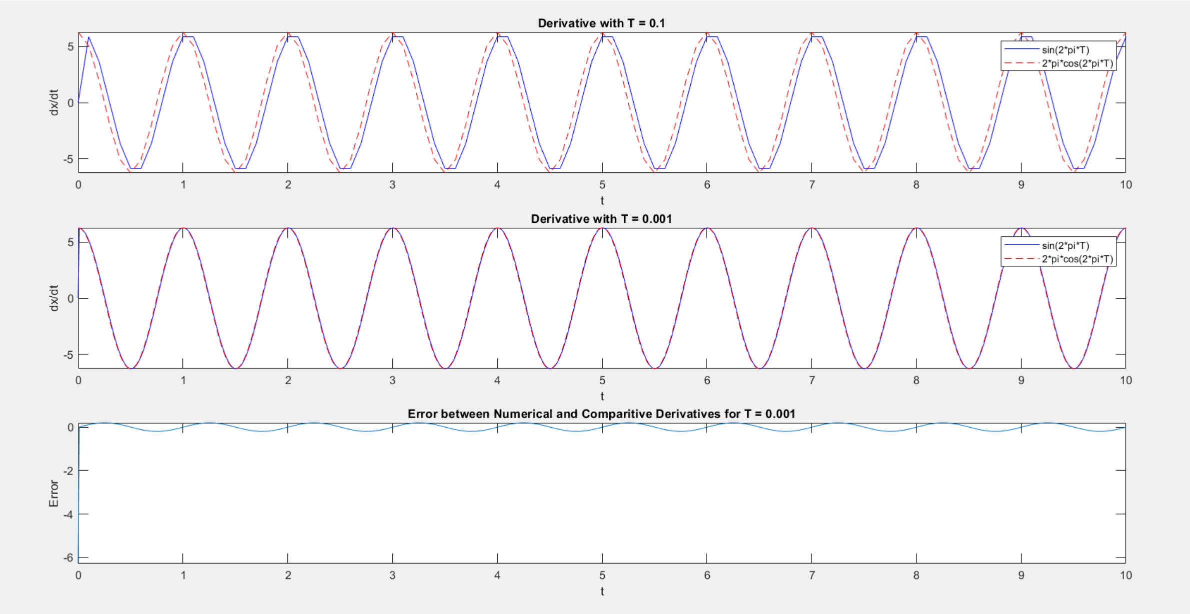
Plots

Plotting with the help of MATLAB:

Blue line indicates $\sin(2\pi \cdot t)$

Red Line indicates waveform of $2{\cdot}\pi\ \cos(2\pi \cdot t)$

Hence, we can make out the small differences in the plot itself.



(a) Applying Differentiator to sin(2pi*t)

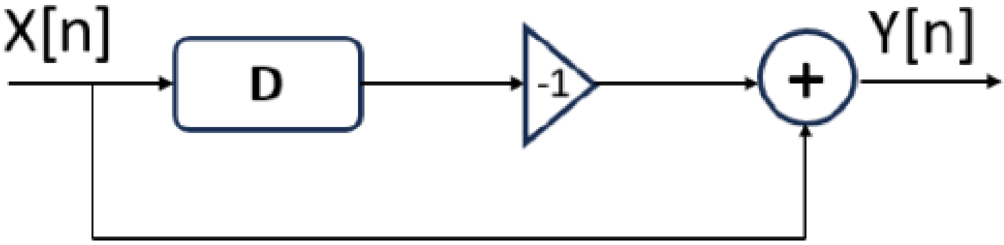
3. Difference Equations

Implement and analyze discrete-time filters defined by the difference equations y[n] = x[n] - x[n 1] and y[n] = $\frac{1}{2}$ y[n 1] + x[n], and determine their impulse responses for various system configurations

- y[n] = x[n] − x[n-1]
- y[n] = $\frac{1}{2}$x[n-1] +x[n]

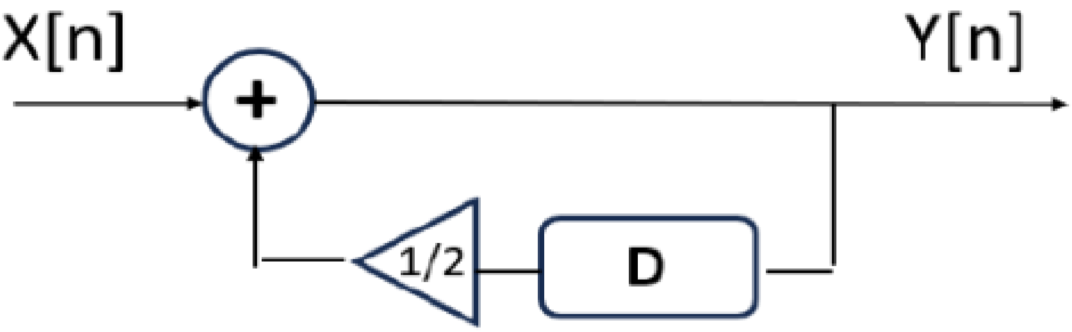
3.1 System Diagrams For all the Systems

S1



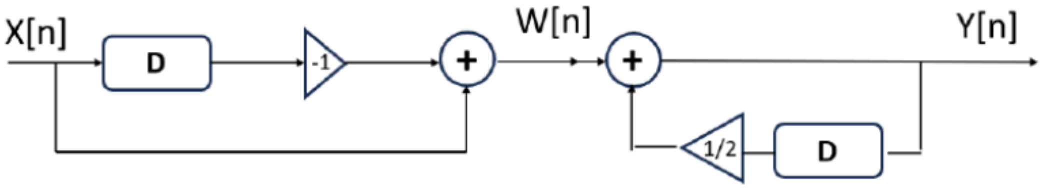
(a) Plot of S1

S2



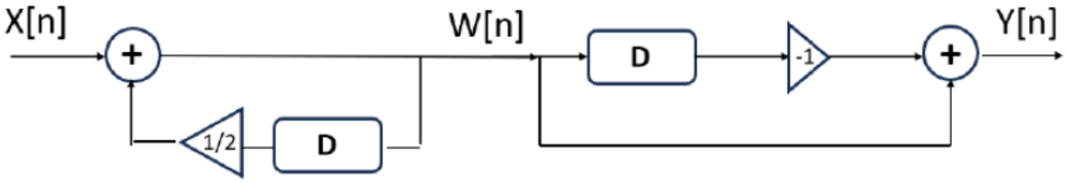
(a) Plot of S2

S1(S2)



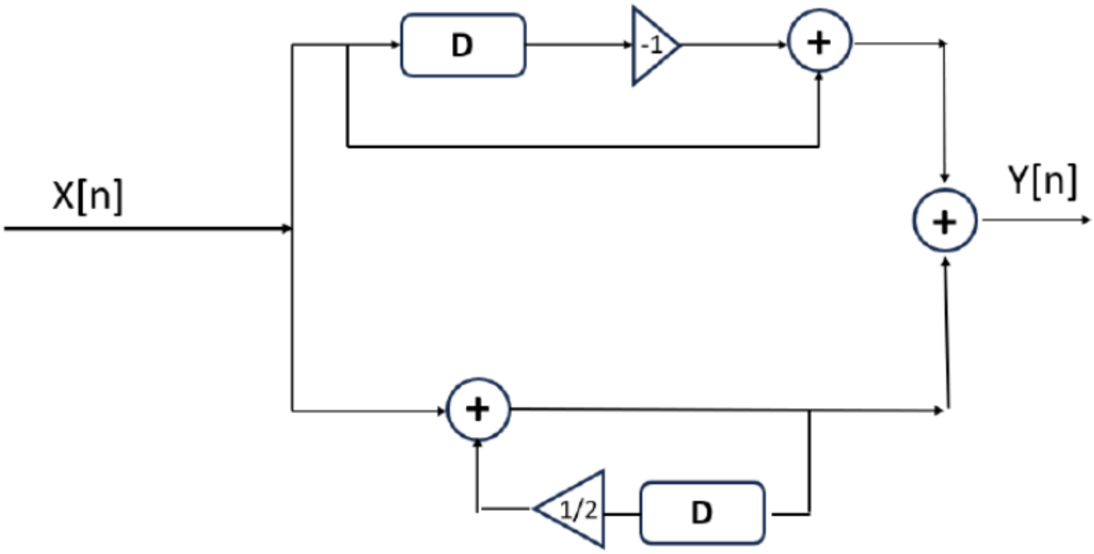
(a) Plot of S1(S2)

S2(S1)



(a) Plot of S2(S1)

S1+S2



(a) S1+S2

MATLAB Code

```

%Given difference eqn of S1
% y[n] = x[n] - x[n-1]

function  y = S1(x)
    y = zeros(size(x));
    for n = 2:length(x)
        y(n) =  x(n) - x(n-1);
    end
end

%Given difference eqn of S2
% y[n] = 0.5* y[n-1] + x[n]
function y = S2(x)
    y = zeros(size(x));
    for n = 2:length(x)
        y(n) = 0.5 * y(n-1) + x(n);
```

```
    end
end

%Defining the signal
n = 0:50;
x = zeros(size(n));

%Giving impulse at n = 25;
x(26) = 1;

% Calculate the impulse responses
h_S1 = S1(x);
h_S2 = S2(x);
h_S1_S2 = S1(S2(x));
h_S2_S1 = S2(S1(x));
h_S1_plus_S2 = S1(x) + S2(x);

% Plot the impulse responses
figure;

subplot(1,1,1);
stem(n, h_S1);
title('Impulse Response of S1');
xlabel('n');
ylabel('h_S1[n]');
sgtitle('Name : Sujay Vivek - Roll No: 22EE30029')

figure;

subplot(1,1,1);
stem(n, h_S2);
title('Impulse Response of S2');
xlabel('n');
ylabel('h_S2[n]');
sgtitle('Name : Sujay Vivek - Roll No: 22EE30029')

figure;

subplot(1,1,1);
stem(n, h_S1_S2);
title('Impulse Response of S1(S2)');
```

```
xlabel('n');
ylabel('h_{S1(S2)}[n]');
sgtitle('Name : Sujay Vivek - Roll No: 22EE30029')

figure;

subplot(1,1,1);
stem(n, h_S2_S1);
title('Impulse Response of S2(S1)');
xlabel('n');
ylabel('h_{S2(S1)}[n]');
sgtitle('Name : Sujay Vivek - Roll No: 22EE30029')

figure;

subplot(1,1,1);
stem(n, h_S1_plus_S2);
title('Impulse Response of S1 + S2');
xlabel('n');
ylabel('h_{S1+S2}[n]');
sgtitle('Name : Sujay Vivek - Roll No: 22EE30029')
```

Plots

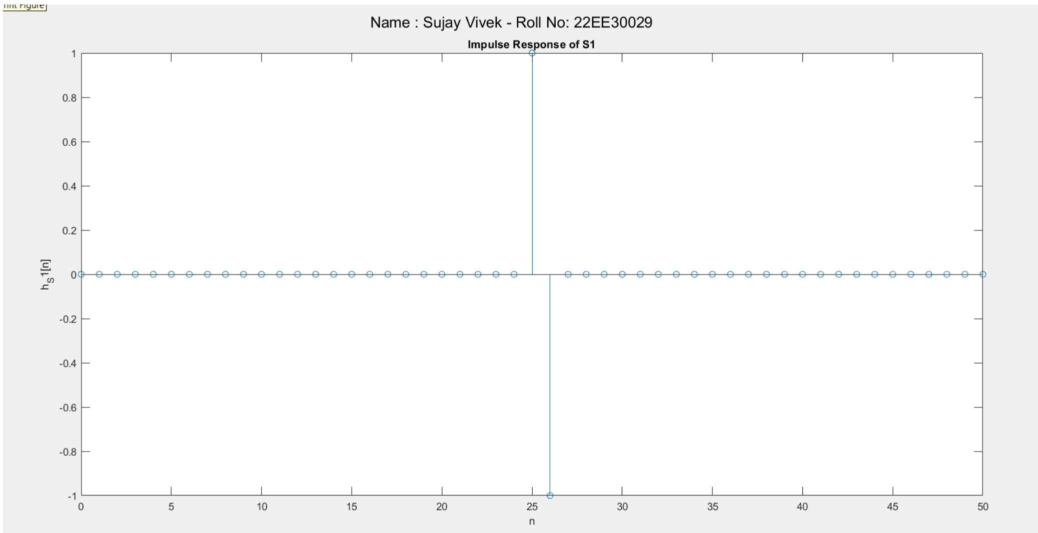


Figure 10: Plotting the Impulse Response of S1

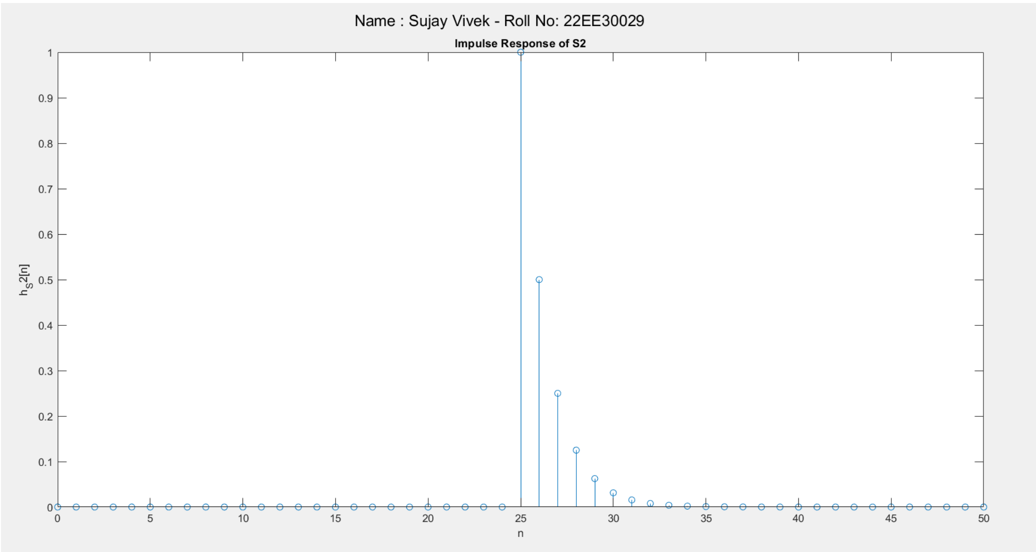


Figure 11: Plotting the Impulse Response of S2

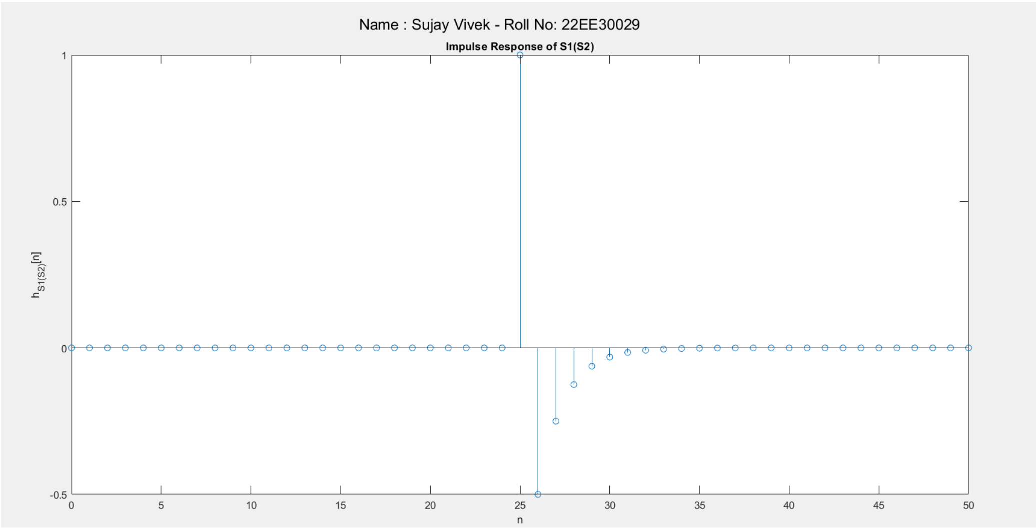


Figure 12: Plotting the Impulse Response of S1(S2)

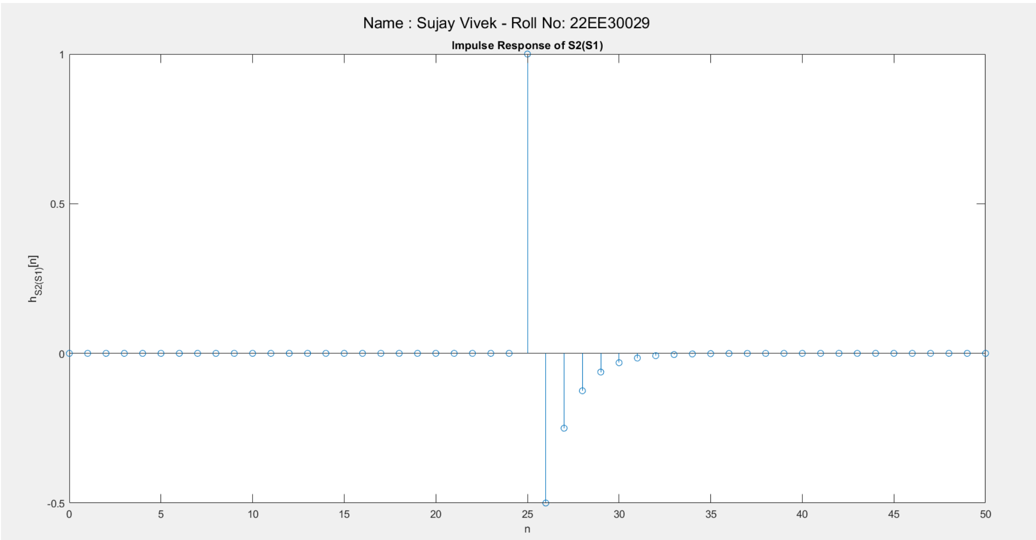


Figure 13: Plotting the Impulse Response of S2(S1)

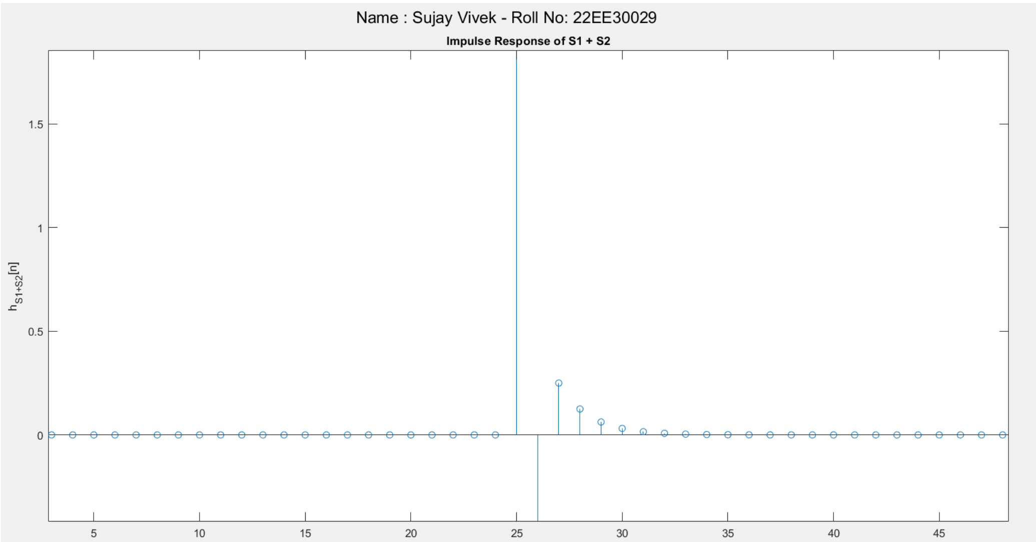


Figure 14: Plotting the Impulse Response of S1 + S2

Observations

a) S1 Filter or Difference Filter

The output y[n] = x[n] - x[n-1] acts as a high-pass filter, emphasizing changes between successive input samples. The impulse response yS1 reflects the difference operation, showing a positive value at n=0 and a negative value at n=1, which signifies the subtraction of adjacent samples

b) S2 Filter or First Order Recursive Filter

The output y[n] = 0.5*y[n-1] + x[n] represents a low-pass filtering operation, where the output depends on a weighted sum of the previous output and the current input. The impulse response yS2 shows an exponential decay, which is typical for a first-order recursive filter, indicating that the effect of the impulse gradually diminishes over time.

c) S1(S2)

This combination first applies the low-pass filtering (S2) and then the high-pass filtering (S1) to the signal. The resulting response yS1S2 captures the characteristics of both filters, leading to a response that highlights the difference between smoothed (filtered) values, which introduces both smoothing and differentiation effects.

d) S2(S1)

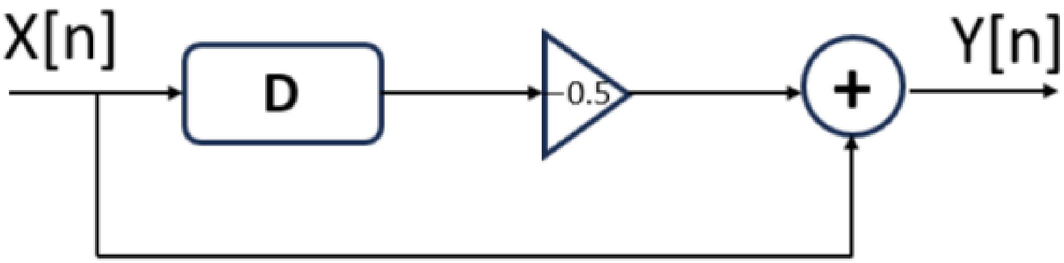
The operations are reversed, first applying the difference (S1) and then the lowpass filtering (S2). The impulse response yS2S1 reflects a low-pass filtering of the difference operation, resulting in a smoothed version of the difference signal

d) S2+S1

This combination sums the outputs of S1 and S2 applied independently to the input signal. The impulse response yS1plusS2 shows a combination of the highpass and low-pass characteristics, with both immediate and delayed components contributing to the final response. This configuration can be seen as emphasizing both the changes and the steady components in the input signal

4. Inverse Systems

We aim to find difference equation for a system S3 such that S3[S2[]] = delta, and to analyze the inverse relationship between the systems S2 and S3.
The System S3 used here : y[n] = x[n] - 0.5x[n-1]
Block Diagram of Inverse System



(a) S1+S2

MATLAB Code

```
function y = S3(x)
    N = length(x);
    y = zeros(1, N);
    for n = 2:N
        y(n) = x(n) - 0.5 * x(n-1);
    end
    y(1) = x(1); % Since there is no previous value, only x[1] contributes
end

% Impulse signal delta[n]
n = -10:10;
delta = (n == 0);

% Apply S3 to the impulse signal
y_S3 = S3(delta);

% Apply S2 to the impulse signal
y_S2 = filter(1, [1 -0.5], delta);

```

```
% Apply S3 to the output of S2
y_S3S2 = S3(y_S2);

% Plotting
figure;

% Plot impulse response of S3
subplot(2,1,1);
stem(n, y_S3);
title('Impulse Response of S3');
xlabel('n');
ylabel('y[n]');
grid on;

% Plot impulse response of S3(S2)
subplot(2,1,2);
stem(n, y_S3S2);
title('Impulse Response of S3(S2(\delta[n]))');
xlabel('n');
ylabel('y[n]');
grid on;

% Adjust layout
sgtitle('Name: Sujay Vivek - Roll No: 22EE30029');
```

Plots

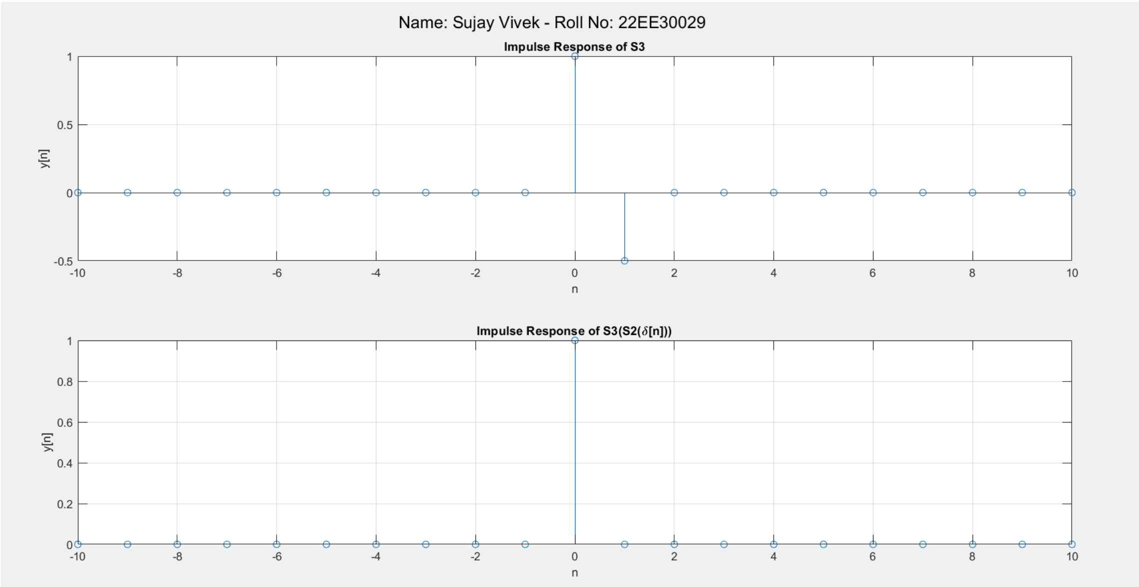


Figure 16: Plotting the Impulse Response of S3 and the Impulse Response of S3(S2)

5. System Tests

To determine which among the systems S1, S2, and S3 is non-linear and/or time-varying by finding counter-examples for linearity and time invariance properties.

MATLAB Code

```
% Define time vector for testing
N = 50;
n = 0:N-1;

% Define test signals
x1 = sin(2*pi*0.1*n);   % Sinusoidal input signal
x2 = rand(1, N);        % Random input signal
x_delayed = [zeros(1, 5) x1(1:N-5)]; % Delayed
    signal by 5 samples
```

```
% Define the filters
function y = filter_S1(x)
    y = zeros(size(x));
    for i = 2:length(x)
        y(i) = x(i) - x(i-1);
    end
end

function y = filter_S2(x)
    y = zeros(size(x));
    for i = 2:length(x)
        y(i) = 0.5 * y(i-1) + x(i);
    end
end

function y = S3(x)
    a = 3; % random constant a
    b = -2; % random constant b
    y = zeros(size(x));
    for i = 2:length(x)
        y(i) = a * x(i) + b * x(i-1);
    end
end

% Test linearity
y1_x1 = filter_S1(x1);
y1_x2 = filter_S1(x2);
y1_combined = filter_S1(x1 + x2);
linear_test_S1 = y1_x1 + y1_x2;

y2_x1 = filter_S2(x1);
y2_x2 = filter_S2(x2);
y2_combined = filter_S2(x1 + x2);
linear_test_S2 = y2_x1 + y2_x2;

y3_x1 = S3(x1);
y3_x2 = S3(x2);
y3_combined = S3(x1 + x2);
linear_test_S3 = y3_x1 + y3_x2;

```

```
% Test time-invariance for S1
y1_x1_delayed = filter_S1(x_delayed);
y1_x1_delayed_response = [zeros(1, 5) y1_x1(1:N-5)];

% Test time-invariance for S2
y2_x1_delayed = filter_S2(x_delayed);
y2_x1_delayed_response = [zeros(1, 5) y2_x1(1:N-5)];

% Test time-invariance for S3
y3_x1_delayed = S3(x_delayed);
y3_x1_delayed_response = [zeros(1, 5) y3_x1(1:N-5)];

% Plot results
figure;

% Linearity tests
subplot(3,2,1);
plot(n, y1_combined);
title('Linearity Test for S1:S1(X1+X2)');
xlabel('n');
ylabel('y[n]');
grid on;

subplot(3,2,2);
plot(n, linear_test_S1);
title('Linearity Test for S1:S1(x1)+S1(x2)');
xlabel('n');
ylabel('y[n]');
grid on;

subplot(3,2,3);
plot(n, y2_combined);
title('Linearity Test for S2:S2(X1+X2)');
xlabel('n');
ylabel('y[n]');
grid on;

subplot(3,2,4);
plot(n, linear_test_S2);
title('Linearity Test for S2:S2(x1)+S2(x2)');
xlabel('n');
```

```
ylabel('y[n]');
grid on;

subplot(3,2,5);
plot(n, y3_combined);
title('Linearity Test for S3:S3(X1+X2)');
xlabel('n');
ylabel('y[n]');
grid on;

subplot(3,2,6);
plot(n, linear_test_S3);
title('Linearity Test for S3:S3(x1)+S3(x2)');
xlabel('n');
ylabel('y[n]');
grid on;

% Adjust layout
sgtitle('Name- Sujay Vivek - Roll No: 22EE30029');

figure;

subplot(3,2,1);
plot(n, y1_x1_delayed);
title('Time-Invariance Test for S1:output to
   delayed input');
xlabel('n');
ylabel('y[n]');
grid on;

subplot(3,2,2);
plot(n, y1_x1_delayed_response);
title('Time-Invariance Test for S1:delayed output');
xlabel('n');
ylabel('y[n]');
grid on;

subplot(3,2,3);
plot(n, y2_x1_delayed);
title('Time-Invariance Test for S2:output to
   delayed input');
```

```
xlabel('n');
ylabel('y[n]');
grid on;

subplot(3,2,4);
plot(n, y2_x1_delayed_response);
title('Time-Invariance Test for S2:delayed output');
xlabel('n');
ylabel('y[n]');
grid on;

subplot(3,2,5);
plot(n, y3_x1_delayed);
title('Time-Invariance Test for S3:output to
    delayed input');
xlabel('n');
ylabel('y[n]');
grid on;

subplot(3,2,6);
plot(n, y3_x1_delayed_response);
title('Time-Invariance Test for S3:delayed output');
xlabel('n');
ylabel('y[n]');
grid on;

% Adjust layout
sgtitle('Name- Sujay Vivek - Roll No: 22EE30029');
```

Plots

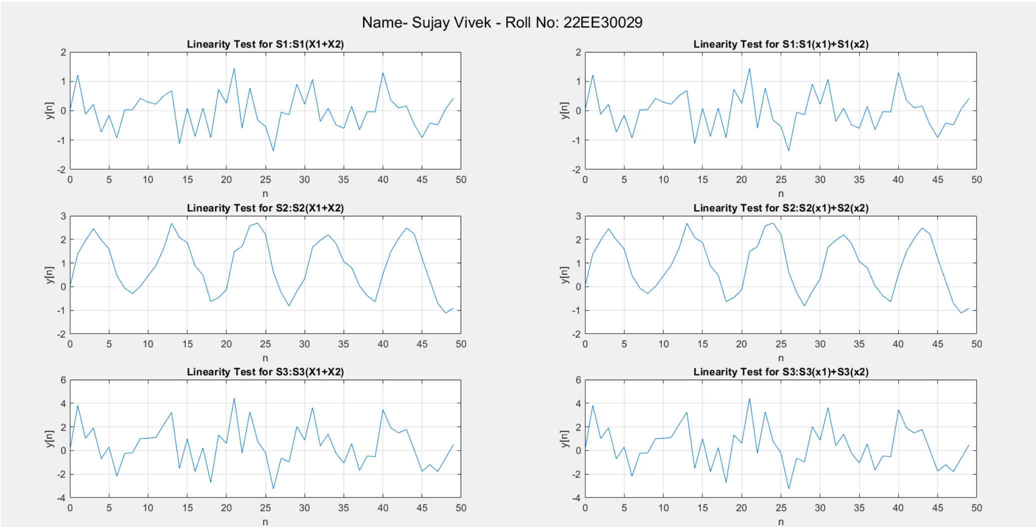


Figure 17: Linear Testing and Verifying

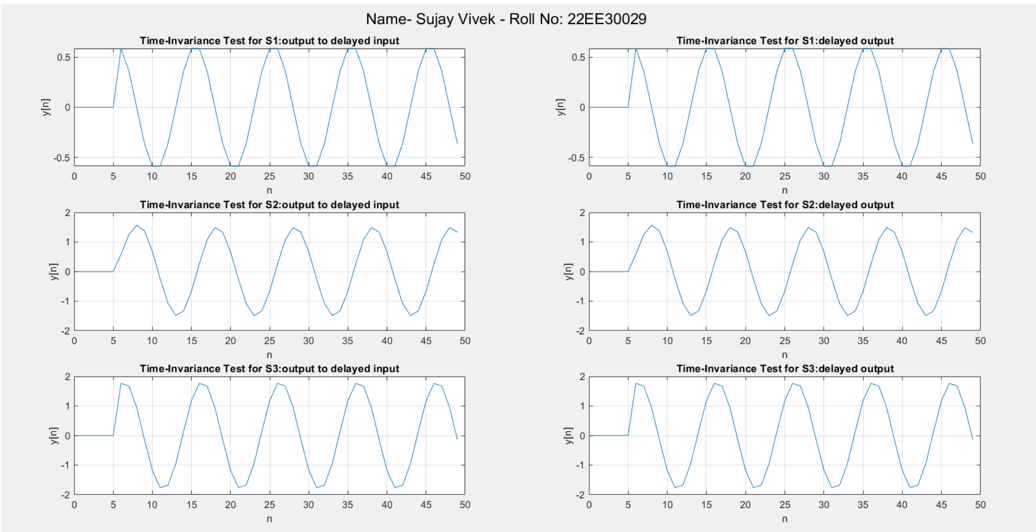


Figure 18: Time-Invariance Testing and Verifying

Observations

a) System S1

The linearity test results show that S1 does not pass the linearity check, meaning that this system is non-linear. This behavior is typical of systems that perform non-additive operations, such as differentiation. The time-invariance test reveals that S1 is also not time-invariant, as the output response to a delayed input does not match the delayed version of the system's original output.

b) System S2

S2 passes the linearity test, confirming that it is a linear system. This is expected, as S2 performs additive operations on the input signal. The time-invariance test shows that S2 is time-invariant, as the output response to a delayed input matches the delayed output.

c) System S3

S3 also passes the linearity test, showing that it is linear. S3 passes the time-invariance test, confirming that it is time-invariant.

6. Stock Market Example

To analyze different methods for computing the average value of a stock, represented by discrete-time systems, by drawing system diagrams for system S3 submitting plots of impulse responses for S3 and S3[S2[]].

MATLAB Code

```
% Define time vector
N = 10;
n = -N:N;

% Define impulse signal
impulse = zeros(1, length(n));
impulse(n == 0) = 1;  % Delta function

% Define the systems based on the methods provided
```

```
% Method 1: Moving Average
function y = method_1(x)
    % Average over 3 samples
    y = filter(ones(1, 3) / 3, 1, x);
end

% Method 2: Weighted Moving Average
function y = method_2(x)
    % Weighted average: 0.6 * previous output + 0.4 * current input
    y = filter(0.4, [1, -0.6], x);
end

% Method 3: Difference with moving average
function y = method_3(x)
    % Difference with moving average over 3 samples
    y = zeros(size(x));
    y(2:end) = filter([1, 0, -1] / 3, 1, x(2:end));
end

% Calculate impulse responses
h_m1 = method_1(impulse);
h_m2 = method_2(impulse);
h_m3 = method_3(impulse);

% Plotting the impulse responses
figure;

% Plot for Method 1
subplot(3,1,1);
stem(n, h_m1);
title('Impulse Response of Method 1');
xlabel('n');
ylabel('h[n]');

% Plot for Method 2
subplot(3,1,2);
stem(n, h_m2);
title('Impulse Response of Method 2');
xlabel('n');
```

```
ylabel('h[n]');

% Plot for Method 3
subplot(3,1,3);
stem(n, h_m3);
title('Impulse Response of Method 3');
xlabel('n');
ylabel('h[n]');

% Adjust the layout
sgtitle('Name - Sujay Vivek - Roll No: 22EE30029');
```

<u>Plots</u>

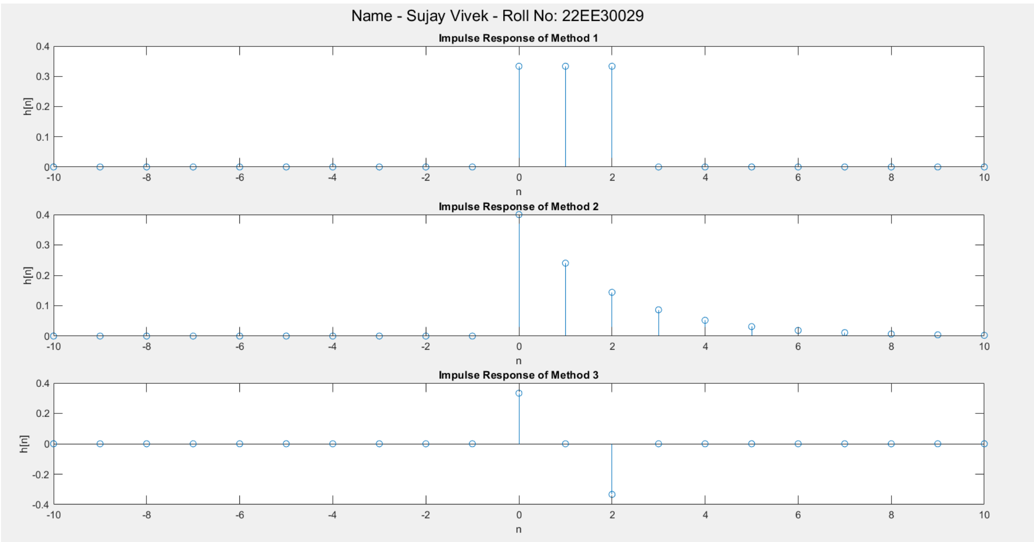


Figure 19: Impulse Response for different Methods

Observations

Method 1 - Simple Moving Average Basic Smoothing- Provides a simple averaging of the last three values, effectively reducing short-term fluctuations. Impulse Response- The triangular shape reflects the smoothing process over a small, fixed window.
Method 2 - Weighted Moving Average Adaptive Smoothing- Applies a weighted combination of past and present values, offering more nuanced smoothing. Impulse Response- Exhibits an exponential decay, indicating that the impact of the impulse decreases gradually over time
Method 3 - Difference with Moving Average Complex Smoothing- Involves a difference term, allowing for a more intricate smoothing mechanism. Impulse Response- Shows a combined effect of smoothing and delay, capturing both the average and delayed changes in the signal.