Name: David, RuzzelS.
Section:CPE-401
Date Assigned:September 12, 2023
Date Submitted:

EXPERIMENT 5
Linear Time Invariant
Discrete Time System

OBJECTIVES

1. To calculate and plot the response of LTI Systems to a particular input.
2. To investigate the result of convolution on two finite length sequences.
3. To perform stability test on a given DT system.

MATERIALS AND EQUIPMENT

Computer with installed Octave 5.2.0 / MATLAB 2018

AUDIO FILES

hello.wav

INTRODUCTION

A discrete-time system processes an input signal in the time-domain to generate an output signal with more desirable properties by applying an algorithm composed of simple operations on the input signal.

- A linear time-invariant (LTI) discrete-time system possesses the properties of linearity and time-invariance.
- The response of a discrete-time system to a unit sample sequence δ[n] is called the unit sample response or, simply, the ***impulse response denoted as h[n]***.
- An LTI discrete-time system is BIBO stable if and only if its impulse response h[n] sequence is absolutely summable, that is,

$$\sum_{n=-\infty}^{\infty} |h[n]| < \infty \qquad \text{Eq.(5.1)}$$

Given an LTI discrete-time system in the form,

$$a_0 y[n] + a_1 y[n-1] + \cdots + a_N y[n-N] = b_0 x[n] + b_1 x[n-1] + \cdots + b_M x[n-M] \qquad \text{Eq. (5.2)}$$

where x[n] and y[n] are the input and the output respectively. If we assume the system to be causal, then we can rewrite Eq.(5.2) as

$$y[n] = \sum_{k=0}^{M} \frac{b_k}{a_0} x[n-k] - \sum_{k=1}^{N} \frac{a_k}{a_0} y[n-k] \qquad \text{Eq.(5.3)}$$

provided $a_0 \neq 0$.

Impulse Response

The response of a discrete-time system to a unit sample sequence δ[n] is called the unit sample response or, simply, the ***impulse response denoted as h[n]***. We can determine the impulse response by applying an input of $\delta[n]$ to the system at rest and obtaining the response h[n].

If x[n]=δ[n], then, y[n]=h[n] , Eq.(5.3) can be modified as

$$h[n] = \sum_{k=0}^{M} \frac{b_k}{a_0} \delta[n-k] - \sum_{k=1}^{N} \frac{a_k}{a_0} h[n-k] \qquad \text{Eq.(5.4)}$$

h[n]=0 for n<0.

The command to compute the first **L** samples of the impulse response is **impz(num,den,L)**, where **num** contains the coefficients of x, similarly, **den** contains the coefficients of y. Alternatively, the impulse response can also be computed using the **filter** command **filter(num,den,D)**, where **D** is a unit sample sequence in this case. The **filter** command filters the data in vector **D** with the filter described by vectors **num** and **den** to create the filtered data **h**.

Program Expt5_1 given below computes and plots the impulse response, using **impz** command, of the system described by the equation

$$y[n] - 0.4y[n-1] + 0.75y[n-2] = 2.2403x[n] + 2.4908x[n-1] + 2.2403x[n-2] \qquad \text{Eq.(5.5)}$$

PROCEDURES

```
% Program Expt5_1
% Program to Compute 41 samples
% of the impulse response, h[n],of Eq(5.5)
clf;
L = 41;
n=0:L-1;
num = [2.2403 2.4908 2.2403];
den = [1 -0.4 0.75];
h = impz(num,den,L);
% Plot the impulse response
stem(n,h);
xlabel('Time index n'); ylabel('Amplitude');
title('Impulse Response'); grid;
```

STEP 1 Run program Expt5_1 and generate the impulse response of the discrete-time system of Eq.(5.5). What are the first five values of h[n]?

2.24030000000000

3.38692000000000

1.91484300000000

-1.77425280000000

-2.14583337000000

STEP 2 Using the **impz** command, obtain the first 40 samples of the impulse response of the LTI system defined by the following equations:

a) $y[n] + 0.71y[n-1] - 0.46y[n-2] - 0.62y[n-3] = 0.9x[n] - 0.45x[n-1] + 0.35x[n-2] + 0.002x[n-3]$

Write the first five values of h[n]:

0.9000

-1.0890

1.5372

-1.0323

0.7649

b) $y[n] = 3x[n] - x[n-1] + 2x[n-2] + x[n-3]$

Write the first five values of h[n]:

3

-1

2

1

0

STEP 3 Using the **filter** command, obtain the first 40 samples of the impulse response of the LTI system described in Step 2(a).

0.9000 -1.0890 1.5372 -1.0323 0.7649

Compare your result with that obtained using the **impz** command.

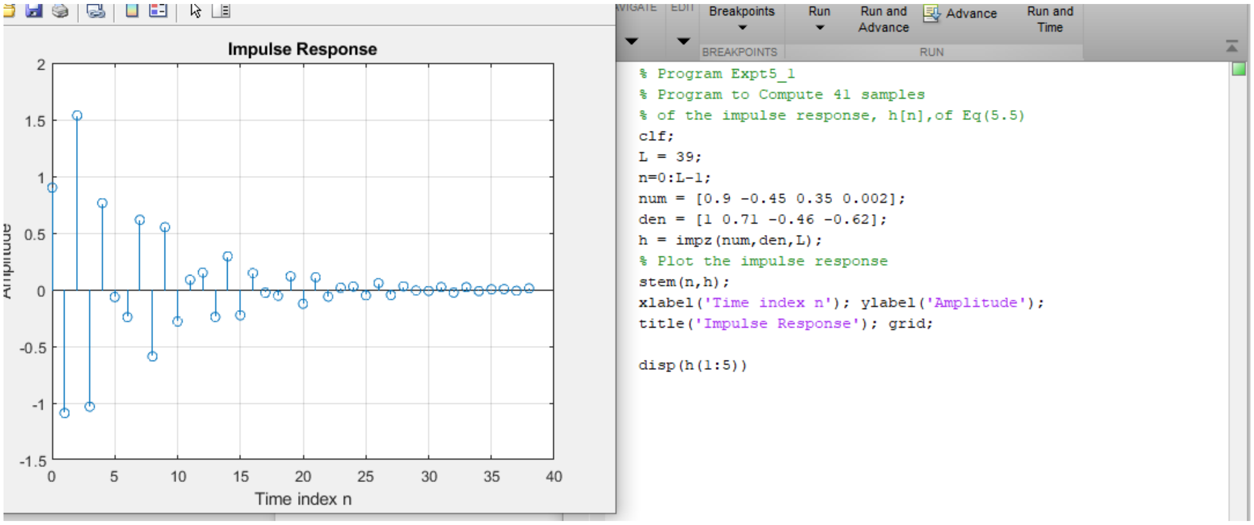
The values in impz is written in column while in filter they are given in row.

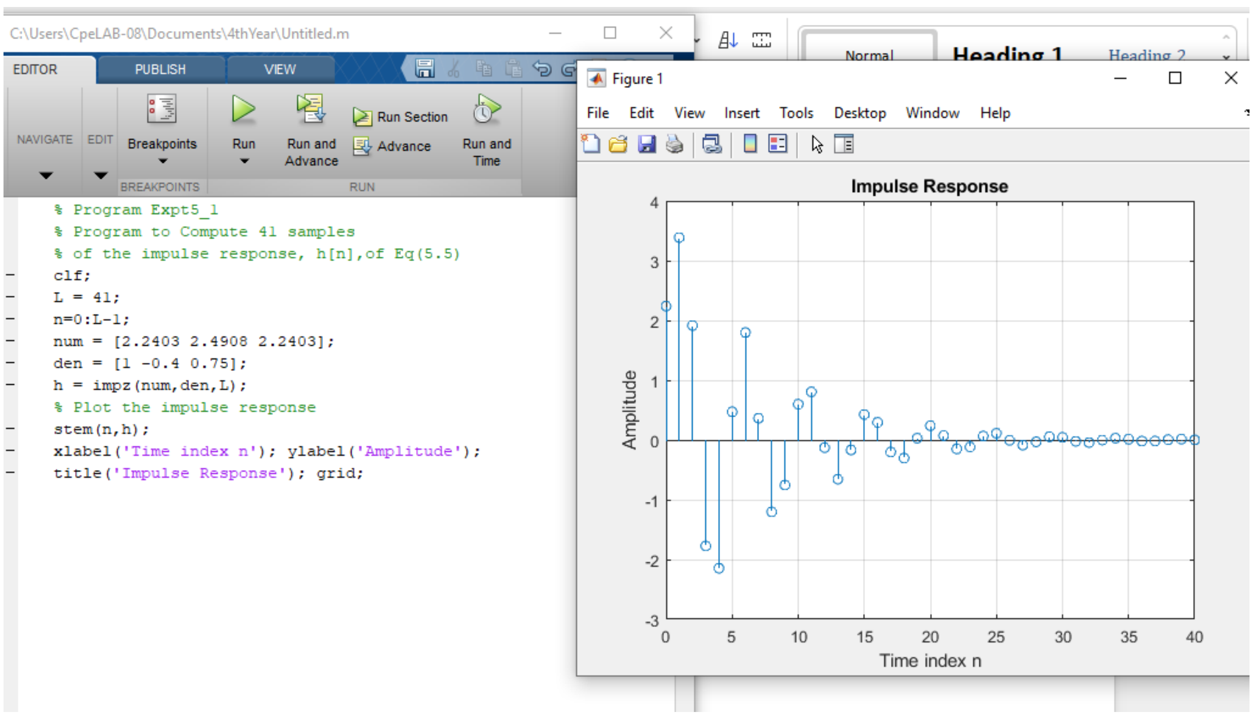
Attach the screenshot of the code and plot below:

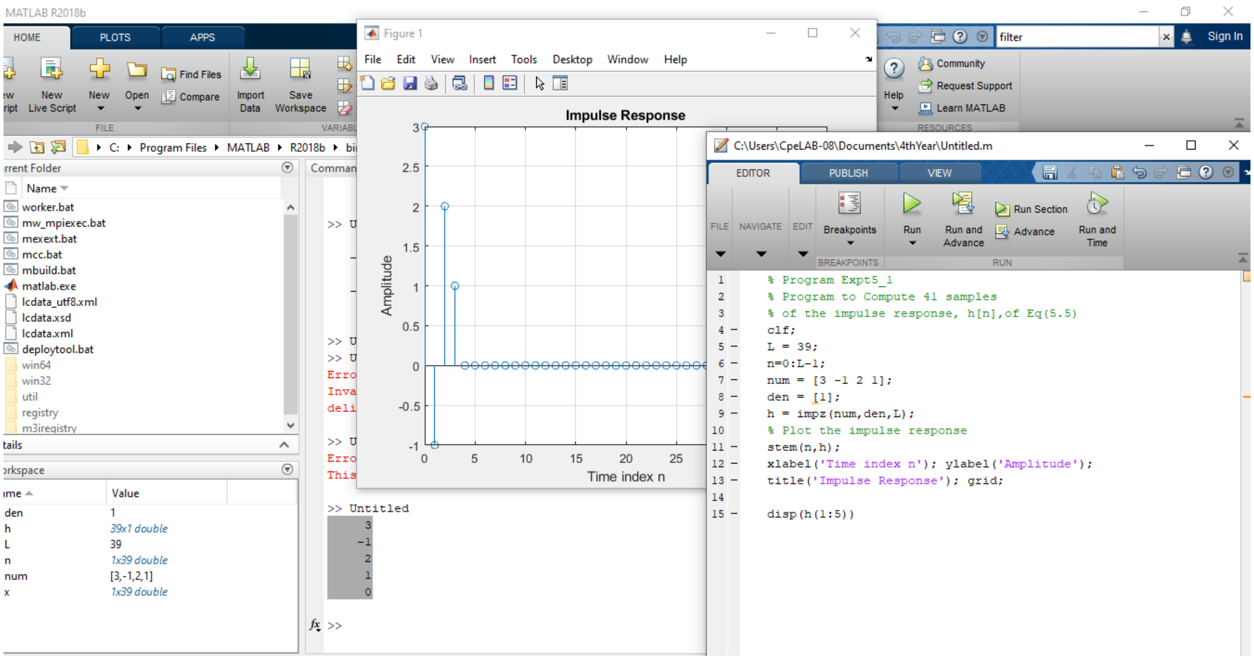
```
% Program Expt5_1
% Program to Compute 41 samples
% of the impulse response, h[n],of Eq(5.5)
clc;
clf;
L = 39;
n=0:L-1;
num = [0.9 -0.45 0.35 0.002];
den = [1 0.71 -0.46 -0.62];
x = zeros(1, L);
x(1)=1;

h = filter(num,den,x);
% Plot the impulse response
stem(n,h);
xlabel('Time index n'); ylabel('Amplitude');
title('Impulse Response'); grid;

disp(h(1:5))
```


Impulse Response
Time index n
% Program Expt5_1
% Program to Compute 41 samples
% of the impulse response, h[n],of Eq(5.5)
clf;
L = 39;
n=0:L-1;
num = [0.9 -0.45 0.35 0.002];
den = [1 0.71 -0.46 -0.62];
h = impz(num,den,L);
% Plot the impulse response
stem(n,h);
xlabel('Time index n'); ylabel('Amplitude');
title('Impulse Response'); grid;
disp(h(1:5))


C:\Users\CpeLAB-08\Documents\4thYear\Untitled.m
EDITOR
PUBLISH
VIEW
Breakpoints
Run
Run and Advance
Run Section
Advance
Run and Time
% Program Expt5_1
% Program to Compute 41 samples
% of the impulse response, h[n],of Eq(5.5)
clf;
L = 41;
n=0:L-1;
num = [2.2403 2.4908 2.2403];
den = [1 -0.4 0.75];
h = impz(num,den,L);
% Plot the impulse response
stem(n,h);
xlabel('Time index n'); ylabel('Amplitude');
title('Impulse Response'); grid;
Figure 1
File Edit View Insert Tools Desktop Window Help
Impulse Response
Amplitude
Time index n



Convolution Sum

The response $y[n]$ of a linear time-invariant discrete-time system characterized by an impulse response $h[n]$ to an input signal $x[n]$ is given by

$$y[n] = \sum_{k=-\infty}^{\infty} h[k]x[n-k]$$

Eq.(5.6)

which can be alternately written as

$$y[n] = \sum_{k=-\infty}^{\infty} h[n-k]x[k]$$

Eq.(5.7)

The sum in Eq.(5.6) and Eq.(5.7) is called the convolution sum of the sequence x[n] and h[n], and is represented as:

$$y[n] = h[n] * x[n] \qquad \text{Eq.(5.8)}$$

the notation $*$ denotes the convolution sum.

The convolution operation is implemented in Octave/Matlab by the command **conv**, provided the two sequences to be convolved are of finite length. For two sequences of length N1 and N2, **conv** returns the resulting sequence of length N1+N2-1.

```
% Program Expt5_2
%Convolution Sum using the conv and filter commands
clf;
h = [3 2 1 -2 1 0 -4 0 3];  % impulse response
x = [1 -2 3 -4 3 2 1];      % input sequence
y = conv(h,x);
ylength=length(h)+length(x)-1;
n = 0:ylength-1;
subplot(2,1,1);
stem(n,y);
xlabel('Time index n'); ylabel('Amplitude');
title('Output Obtained by Convolution'); grid;
x1 = [x zeros(1,8)];
y1 = filter(h,1,x1);
subplot(2,1,2);
stem(n,y1);
xlabel('Time index n'); ylabel('Amplitude');
title('Output Generated by Filtering'); grid;
```

STEP 4 Run Program Exp5_2 to generate y[n] obtained by the convolution of the sequences h[n] and x[n], and to generate y1[n] obtained by filtering the input, x[n], by the impulse response, h[n]. Is there any difference between y[n] and y1[n]?

None, they are the same.

If the length of the impulse response h[n] is N1 and the length of the input signal x[n] is N2, write the general formula for computing the length, N, of the output of **conv**?

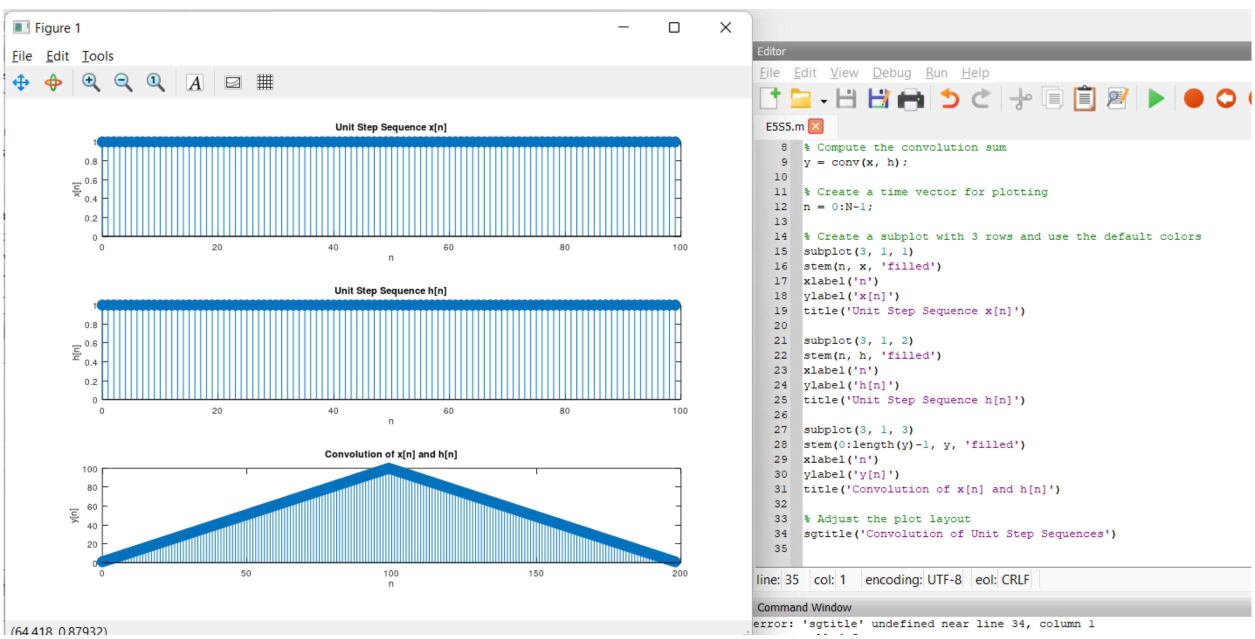
N1 + N2 -1

What is the reason for zero-padding x[n] to obtain x1[n]?

For the array to be equal in size.

STEP 5 Write a program that will generate the convolution sum of two unit step sequences of length 100 each. Make a subplot having 3 rows showing x[n], h[n] and y[n]. Use the proper label and title for each plot.

Attach the screenshot of the code and plot below:



Write your observation on the result.

Convolution output is a triangle, where the peak is in the center and the lowest are on the both sides. The amplitude output is 100, same with the input length and impulse response.

STEP 6 Read and store 'hello.wav' in **x** and convolve it with the impulse response, **h**, given below. Store the result of convolution in **y**. Use the sound command to listen to the original file and the result of convolution. Make a subplot composing of three rows showing x[n], h[n], and y[n].

Below is the Code to generate a certain impulse response, h[n].

```
k=zeros(1,round(fs*0.09));          %generate zeros; fs is the sampling frequency of the .wav file
h=[1,k,0.5,k,0.25,k,0.125,k,0.0625];    %impulse response
```

Write your observation by comparing the sound of the original wav file and the sound of the convolved signals.

After convolution occurs, the output has an echo like effect. Also it becomes louder than the original song.wav.

```
clf;
clear;
Fs = 44100;

% Read the audio file
[x, Fs] = audioread('song.wav');

k = zeros(1, round(Fs * 0.09));
h = [1, k, 0.5, k, 0.25, k, 0.125, k, 0.0625];

% Convolve the audio signal with the impulse response
y = conv(x, h);
ylength = length(x) - length(h) + 1;
n = 0:ylength - 1;

[x_wav, fs] = audioread('song.wav');
y_convolved = conv(x_wav, h);

% Play the original audio
sound(x_wav, fs);

% Pause to listen to the original audio, then play the convolved audio
pause(length(x_wav) / fs);
sound(y_convolved, fs);
```

THE MOVING AVERAGE FILTER SYSTEM

The three-point smoothing filter equation, $y[n] = \frac{1}{3}(x[n-1] + x[n] + x[n+1])$ is an LTI FIR system. Moreover, as *y*[*n*] depends on a future input sample *x*[*n* + 1], the system is non-causal. A causal version of the three-point smoothing filter is obtained by simply delaying the output by one sample period, resulting in the FIR filter described by

$$y[n] = \frac{1}{3}(x[n] + x[n-1] + x[n-2])$$

Generalizing the above equation we obtain

$$y[n] = \frac{1}{M}\sum_{k=0}^{M-1} x[n-k]$$

which defines a causal *M*-point smoothing FIR filter. The system of this generalized equation is also known as a *moving average filter* . We illustrate its use in eliminating high-frequency components from a signal composed of a sum of several sinusoidal signals. Below is the simulation of an M-point moving average filter.

```
% Program Expt5_3
% Simulation of an M-point Moving Average Filter
% Generate the input signal
n = 0:100;
s1 = cos(2*pi*0.05*n); % A low-frequency sinusoid
s2 = cos(2*pi*0.47*n); % A high frequency sinusoid
x = s1+s2;
% Implementation of the moving average filter
M = input('Desired length of the filter = ');
num = ones(1,M);
y = filter(num,1,x)/M;
% Display the input and output signals
clf;
subplot(2,2,1);
plot(n, s1);
axis([0, 100, -2, 2]);
xlabel('Time index n'); ylabel('Amplitude');
title('Signal #1');
subplot(2,2,2);
plot(n, s2);
axis([0, 100, -2, 2]);
xlabel('Time index n'); ylabel('Amplitude');
title('Signal #2');
subplot(2,2,3);
plot(n, x);
axis([0, 100, -2, 2]);
xlabel('Time index n'); ylabel('Amplitude');
title('Input Signal');
subplot(2,2,4);
plot(n, y);
axis([0, 100, -2, 2]);
xlabel('Time index n'); ylabel('Amplitude');
title('Output Signal');
axis;
```

STEP 7 Run the above program for M = 2 to generate the output signal with x[n] = s1[n] + s2[n] as the input.

Which component of the input x[n] is suppressed by the discrete time system simulated by this program (High Frequency or Low frequency) ? What type of filter is this?

High frequency, therefore low pass filter.

STEP 8 Run the same program for other values of filter length M. Write your observation below.

The amplitude of the output decreases as the M increases.

STABILITY OF LTI SYSTEMS

An LTI discrete-time system is BIBO stable if its impulse response is absolutely summable. It therefore follows that a necessary condition for an IIR LTI system to be stable is that its impulse response decays to zero as the sample index gets larger. Program Exp5_4 is a simple MATLAB program used to compute the sum of the absolute values of the impulse response samples of a causal IIR LTI system.

$$S[K] = \sum_{n=0}^{K} |h[n]|$$

If the value of $|h[K]|$ is smaller than 10^{-6}, then it is assumed that the sum $S(K)$ has converged and is very close to $S(\infty)$.

```
% Program Exp5_4
% Stability test based on the sum of the absolute
% values of the impulse response samples
clf;
num = [1 -0.8]; den = [1 1.5  0.9];
N = 200;
h = impz(num,den,N+1);
parsum = 0;
for k = 1:N+1;
    parsum = parsum + abs(h(k));
    if abs(h(k)) < 10^(-6), break, end
end
% Plot the impulse response
n = 0:N;
stem(n,h)
xlabel('Time index n'); ylabel('Amplitude');
% Print the value of abs(h(k))
disp('Value =');disp(abs(h(k)));
```

STEP9 Run program Expt5_4. What is the purpose of the command **for-end** in the code?

To make the code runs only runs when the value of h(k) is less than 10^(-6)

What is the purpose of the command **break** in the code? When will this be executed?

When the for loop reaches its condition, then the loop will end.

What is the discrete-time system equation whose impulse response is being determined by Expt5_4?

y[n] = ex[n]

Is this system stable?(Yes/No) **Yes**. Explain your answer. **As the sample increases the amplitude is becoming zero.**

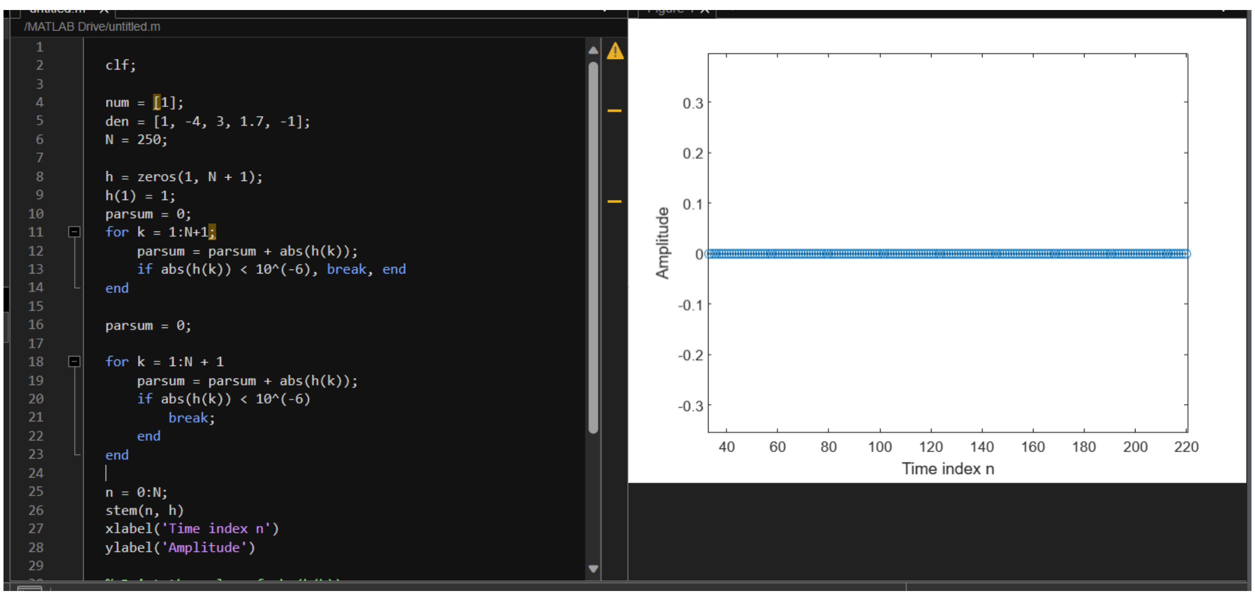
STEP10 Consider the following discrete-time system characterized by the difference equation:

$$y[n] = x[n] - 4\,x[n-1] + 3x[n-2] + 1.7\,y[n-1] - y[n-2].$$

Modify Program Expt5_4 to compute and plot 250 samples of the impulse response of the above system. Is this system stable? Explain.

Yes because the impulse response decays over time.

Attach the screenshot of the code and plot for steps 9 and 10



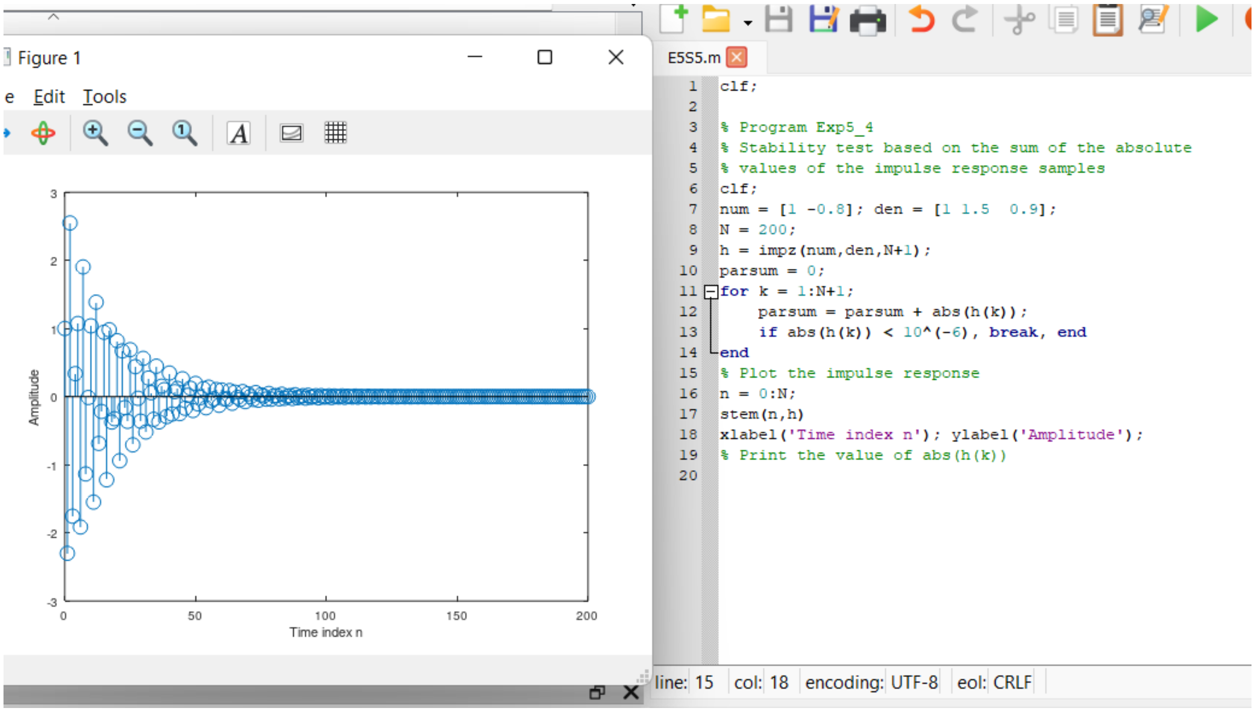


Figure 1
Edit
Tools
Amplitude
Time index n
E5S5.m
clf;
% Program Exp5_4
% Stability test based on the sum of the absolute
% values of the impulse response samples
clf;
num = [1 -0.8]; den = [1 1.5 0.9];
N = 200;
h = impz(num,den,N+1);
parsum = 0;
for k = 1:N+1;
parsum = parsum + abs(h(k));
if abs(h(k)) < 10^(-6), break, end
end
% Plot the impulse response
n = 0:N;
stem(n,h)
xlabel('Time index n'); ylabel('Amplitude');
% Print the value of abs(h(k))
line: 15
col: 18
encoding: UTF-8
eol: CRLF