EE2016 – Microprocessors Theory + Lab

MIDSEM EXAMINATION – PART C

IMPLEMENTATION OF 5 AND 32 – TAP FIR FILTERS

REPORT

NAME : Manikandan Sritharan

ROLL NUMBER: EE19B038

MOBILE NUMBER: 8667258283

META INFORMATION ABOUT THE SIMULATION

Sampled Input Signal is stored from 0x0060.

Coefficients of the filter are stored from 0x018E.

Output values are stored from 0x01AE. Since each output is 16 bits, it is stored in the following format in memory:

HIGH (output), LOW (output)

That is the higher 8 bits will be written in the memory location first, followed by the lower 8 bits.

The simulation takes in 300 sample points. (There isn't enough memory in ATMEGA8 to implement the filter for 1000 sample points. Since the concept remains the same, I have implemented the filter for 300 sample points)

The text file input_signals_and_coeffs.txt contains the samples used for the simulation and the filter coefficients (in hexadecimal).

The text outputs.txt contains the output signal for various inputs used in the simulation (in hexadecimal). Scaling factor for each output is also mentioned.

The python notebook, Data_cleaning.ipynb contains two blocks of code for the python scripts used in the project to edit the data.

The MATLAB codes used are also attached.

The MATLAB files, Freq_Res_5.m and Freq_Res_32.m contains the code for the generation of the Magnitude and Phase Response of the corresponding FIRs.

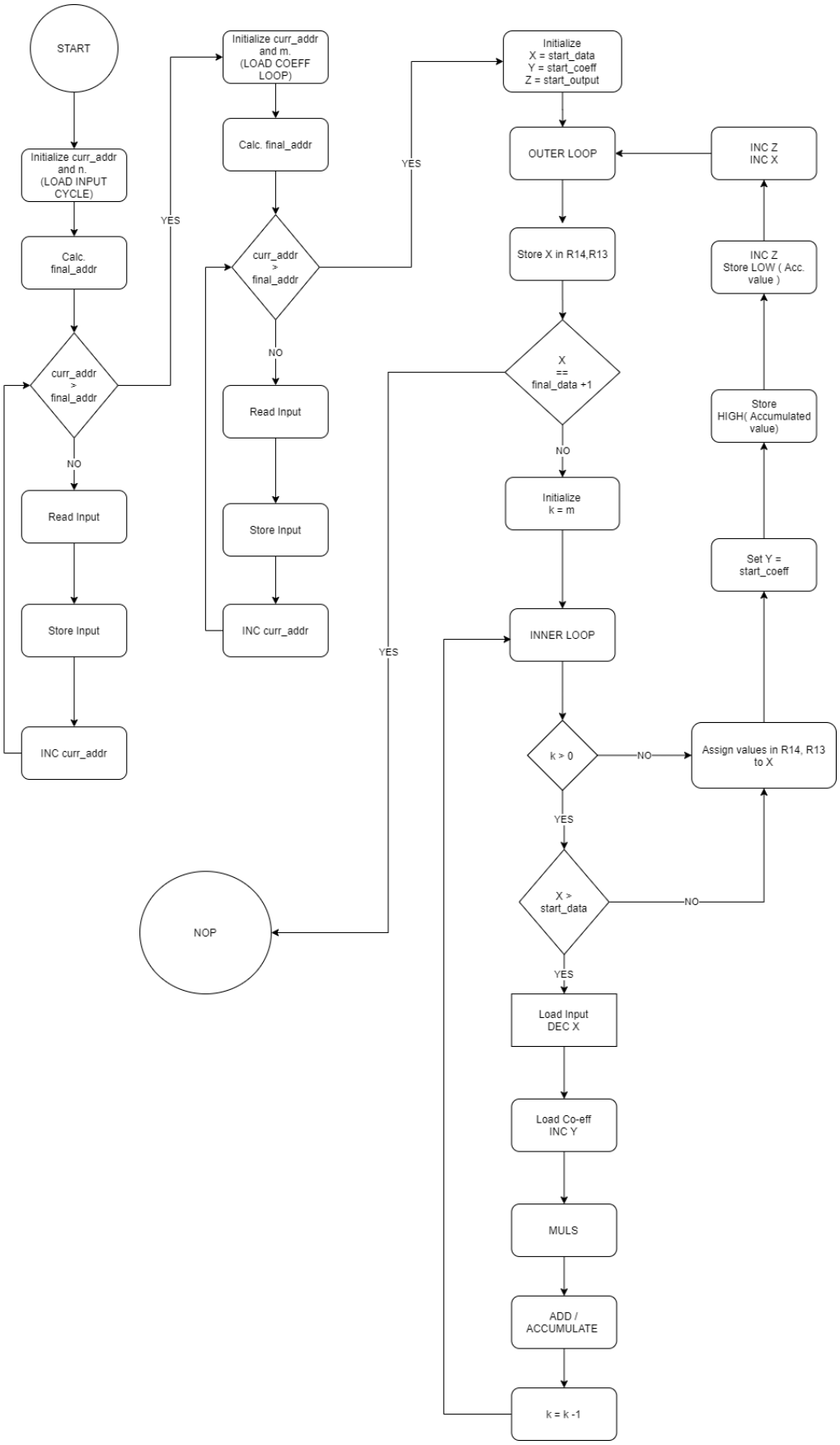
To stimulate the filter for an input, detailed instructions are given under "Data Generation and Cleaning".

The Input and Output plots for various inputs can be found under "Results".

The main.asm file is used for both 5 – tap and 32 – tap filters. The same code works for both filters with the help of "Switches". See "Data Processing – Assembly Program".

Do not forget to scale the inputs before feeding into the system and also, to scale down the outputs before plotting.

Algorithm for m – tap FIR filter with n sample inputs



Data Generation and Cleaning

GENERATION OF TEST SIGNALS:

1) DC:

Following MATLAB Script can be used to generate a DC signal with Amplitude = 1.

```
n = 300; % Number of Sample Points
y = ones([n,1]);
```

2) Sinusoidal signal of Frequency= 1800 Hz.

Following MATLAB Script can be used to generate a sine signal with Amplitude = 1, Frequency= 1800.

```
ti = 0; % start_time
tf = 2*pi; % end_time
n = 300; % Number of Sample points
f = 1800; % Frequency
dt = (tf - ti)/n;
t = ti : dt : tf - dt;
y = sin(2*pi*f*t);
```

3) White noise.

Following MATLAB Script can be used to generate a noise signal with a power level of -3 db.

```
n = 300; % Number of sample points
y = wgn(n,1,-3);
```

PRE-PROCESSING OF INPUT SIGNAL:

The input is appropriately scaled (**Scaling Factor** is carefully chosen, in order to avoid overflow during the processing of the input signal). The input, is then, converted into 2-digit hexadecimal numbers.

```
% scaling the output by 25.
y = y*25;
% representing in 2 digit hexadecimal format.
y = int32(y);
y_hex = dec2hex(y,2);
```

We then, add “0x” before each number and “,” after each element in `y_hex`. Remove line breaks. This can be achieved by using the following python script:

```
f = open("input.txt", "r")
f1 = open("output.txt", "w")
for S in f:
   f1.write(" 0x")
   f1.write(S[0:2])
   f1.write(",")
f.close()
f1.close()
```

PRE-PROCESSING OF FILTER COEFFICIENTS:

The Coefficients are scaled by 100 and rounded off. They are, then, converted to 2-digit hexadecimal equivalent.

Following is the MATLAB code used for the 5 – tap FIR Filter:

```
b = [0,0,0,0,0];
b(1)=-0.389015680;
b(2)=0.1001390904;
b(3)=0.6933953286;
b(4)=0.1001390904;
b(5)=-0.389015680;
b = b*100; % scaling by 1e2
b = int32(b); % rounding off
b_hex = dec2hex(b); % 2-digit hexadecimal representation
```

Following is the MATLAB code used for 32 – tap FIR filter.

```
% Filter coefficients for 32 tap filter
b = [-0.09844;-0.07398;0.05484;-0.12421;
    0.005203;-0.00293;0.045132;0.102960;
    0.012101;0.008764;0.003020;-0.123872;
    0.0035353;-0.14988;-0.242838;0.41265549;
    0.412655;-0.2428388;-0.14988;0.003535;
    -0.123872;0.003020;0.0087646;0.01210138;
    0.10296005;0.045132;-0.0029318;0.0052031;
    -0.12421140;0.05484441;-0.0739811;-0.0984459];
b = b*100; % scaling by 1e2
b = int32(b); % rounding off
b_hex = dec2hex(b); % 2-digit hexadecimal representation
```

PROCESSING OF OUTPUT SIGNAL:

Output from the processing is stored in DATA IRAM. Copy the data and save it in output_pre.txt.
A sample of the data copied is shown below:

```
data 0x037C  fc c9  üÉ
data 0x037E  0b ac  .¬
data 0x0380  fa e9  úé
data 0x0382  f8 3b  ø;
data 0x0384  0a ad  ..
```

Use the following python script to process output_pre.txt.

```
f = open("output_pre.txt", "r")
f1 = open("output.txt", "w")
for S in f:
    f1.write("\"")
    f1.write(S[13:15])
    f1.write(S[16:18])
    f1.write("\";")
f.close()
f1.close()
```

A sample of the contents of output.txt :

```
“fcc9”;“0bac”;“fae9”;“f83b”;“0aad”;
```

Copy and paste the contents of output.txt into an array Y in MATLAB. Run the following code.

```
y = zeros([Y.length(),1]);
for i = 1:Y.length()
    y(i) = nhex2dec(Y(i),16);
    y(i) = y(i)/1e2; %scaled down by 100.
end
```

Function definition of nhex2dec();

```
function [x]=nhex2dec(hexstring,n)
% hexstring : hex representation of two's complement of xmydec=hex2dec(hexstring);
% x : input decimal number
% n : number of bits to perform 2's complements
x = hex2dec(hexstring);
x = x - (x >= 2.^(n-1)).*2.^n;
```

y **now contains the processed output signal.**

Data Processing – Assembly Program

Input Format:

For an **m – tap filter**, N input values to the .asm file have been given in the following format:

```
NUM1 : .DB  val1,val2…,valN;  N sample points
NUM2:  .DB  b(0), b(1), …, b(m-1);  m filter coefficients
```

Modifications to be made before executing .asm file:

LN 44, 89 – Switches.
In order to go from 5 – tap to 32 – tap and vice – versa, make appropriate changes at these lines.
Un-comment the required choice and comment the other one.

Choose NUM1 statement depending on the input.
Choose NUM2 statement depending on the type of FIR filter, you want to stimulate.

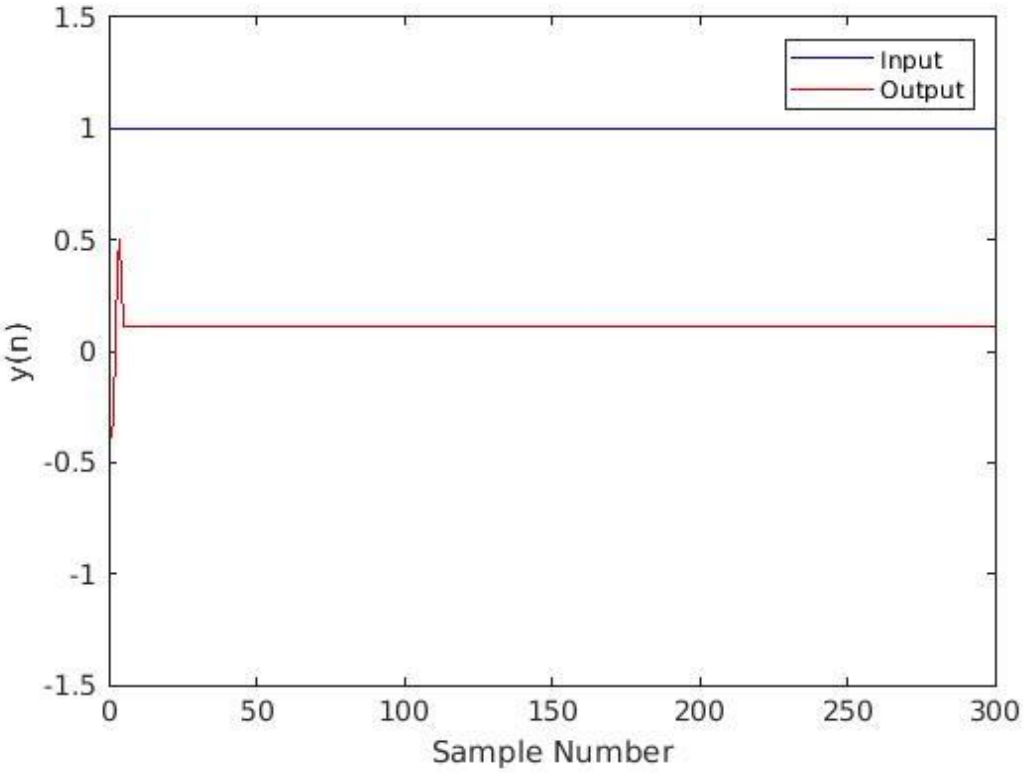
To Note:

Let the stimulated FIR filter be a m-tap filter.
The range of 2 – digit signed hexadecimal is -128 to 127.
An upper bound of the output is m*Amp(Input).
In order to avoid overflow, we take a Scaling factor, roughly equal to [127/m].
So, input signals to the 5 – tap and 32 – tap filters are not the same for the same signal.
In other words, we have to scale them back after processing, to compare them.

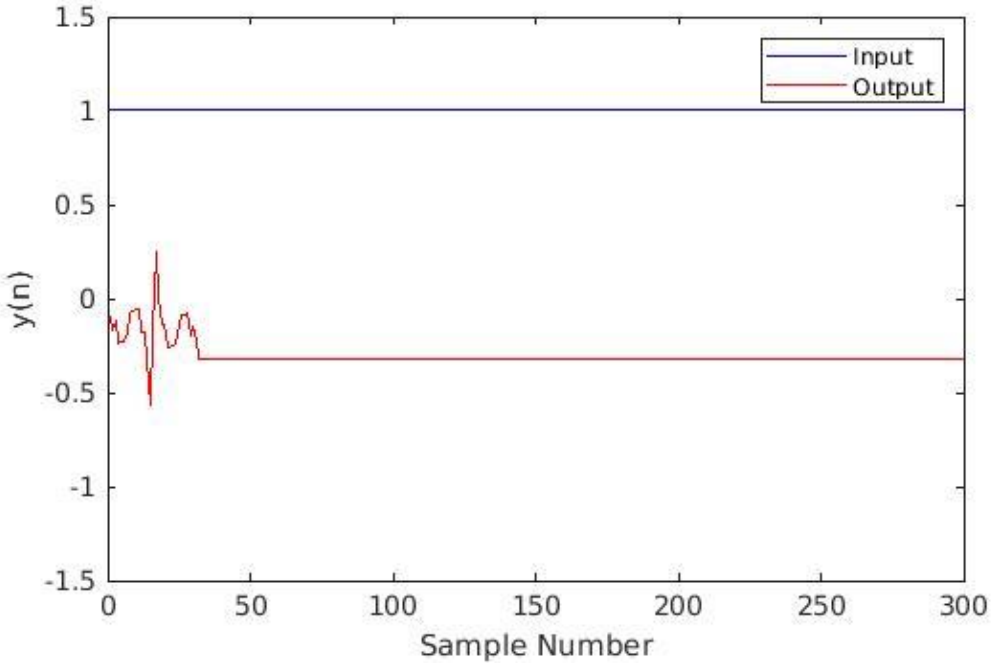
Results

1) DC – u(t) signal:

DC signal – 5 tap FIR filter – Input /Output plot

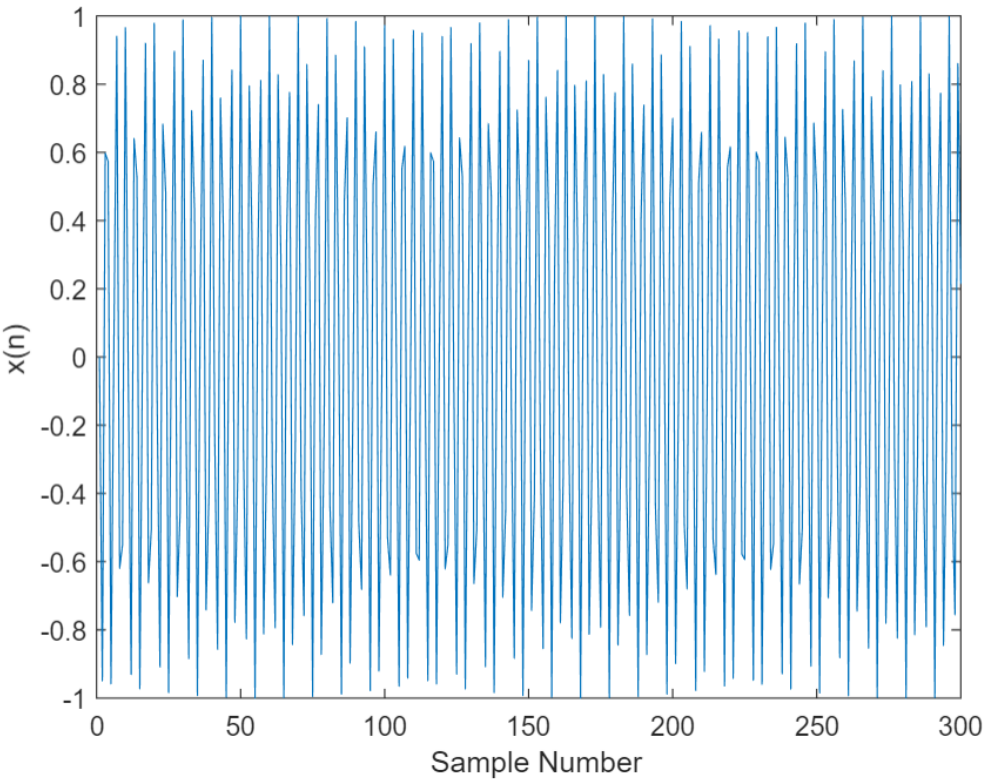


DC signal – 32 tap FIR filter – Input / Output plot

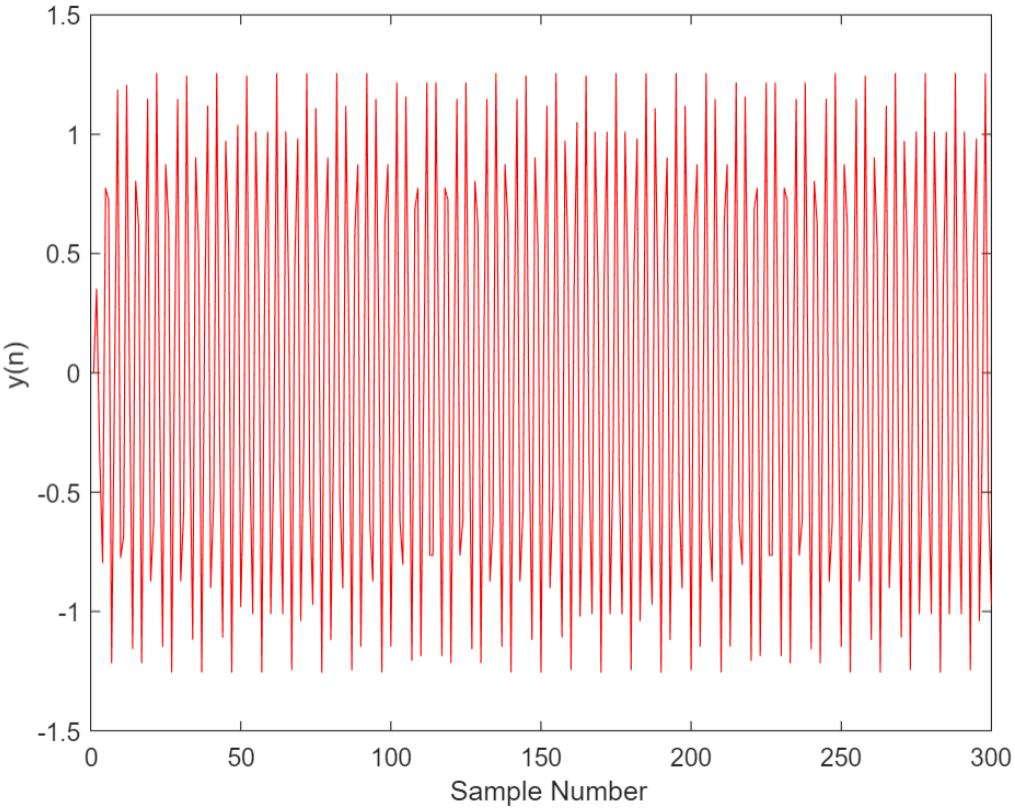


2) Sinusoidal signal of frequency 1800 Hz.

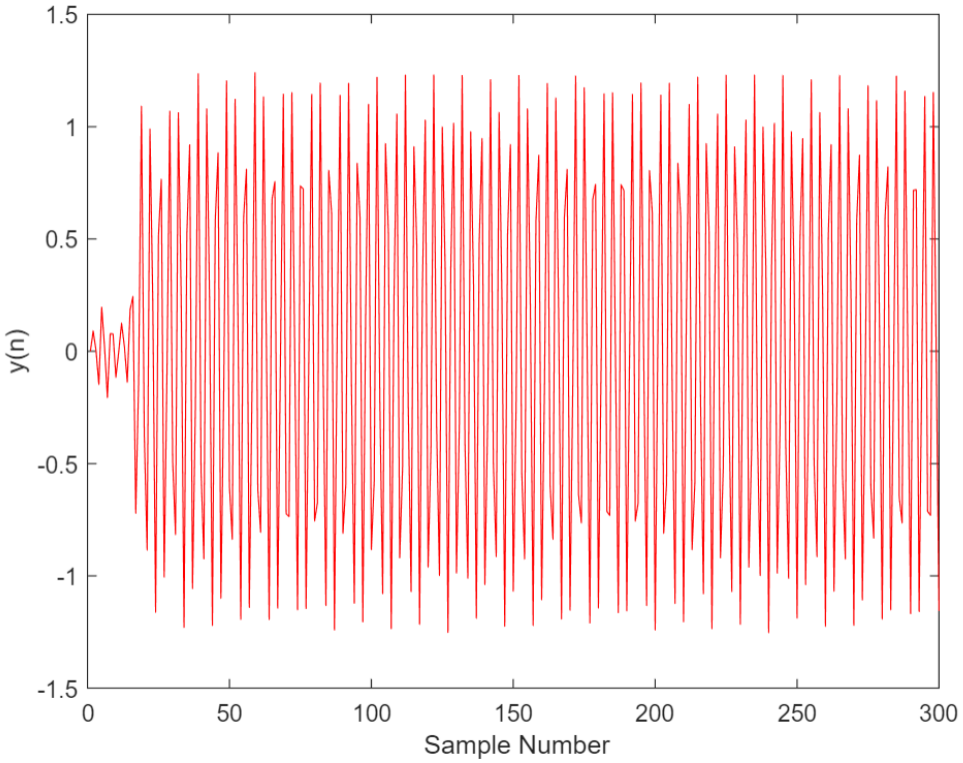
AC Input Signal



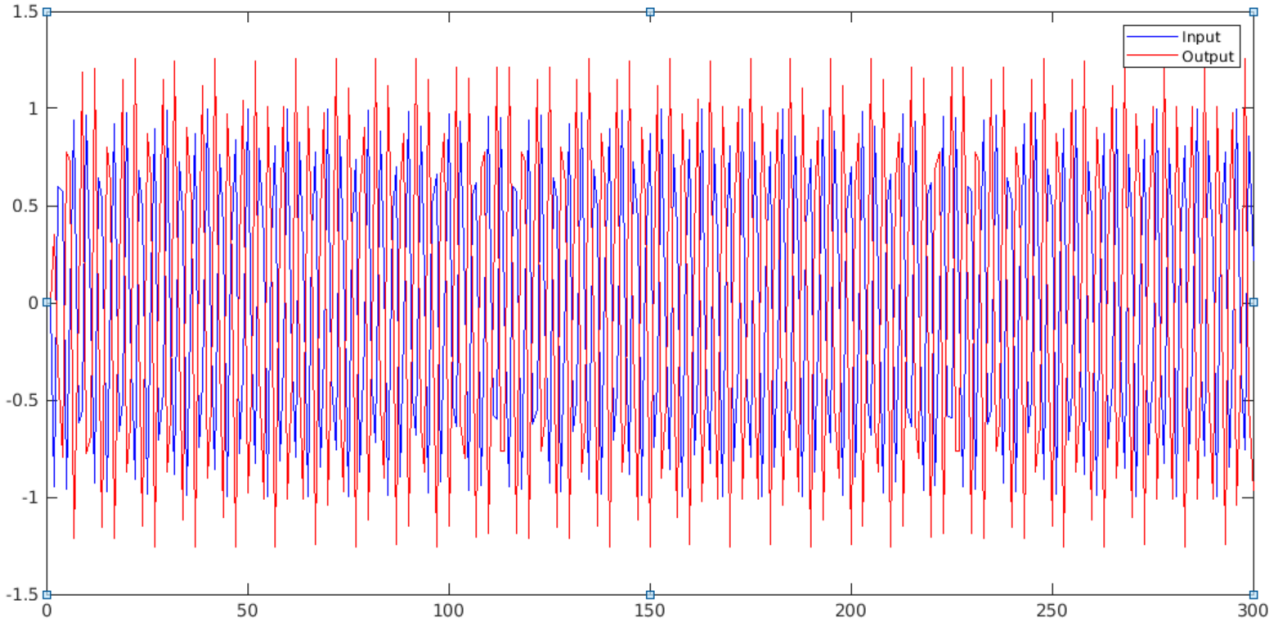
AC Signal – 5 tap FIR filter – Output plot



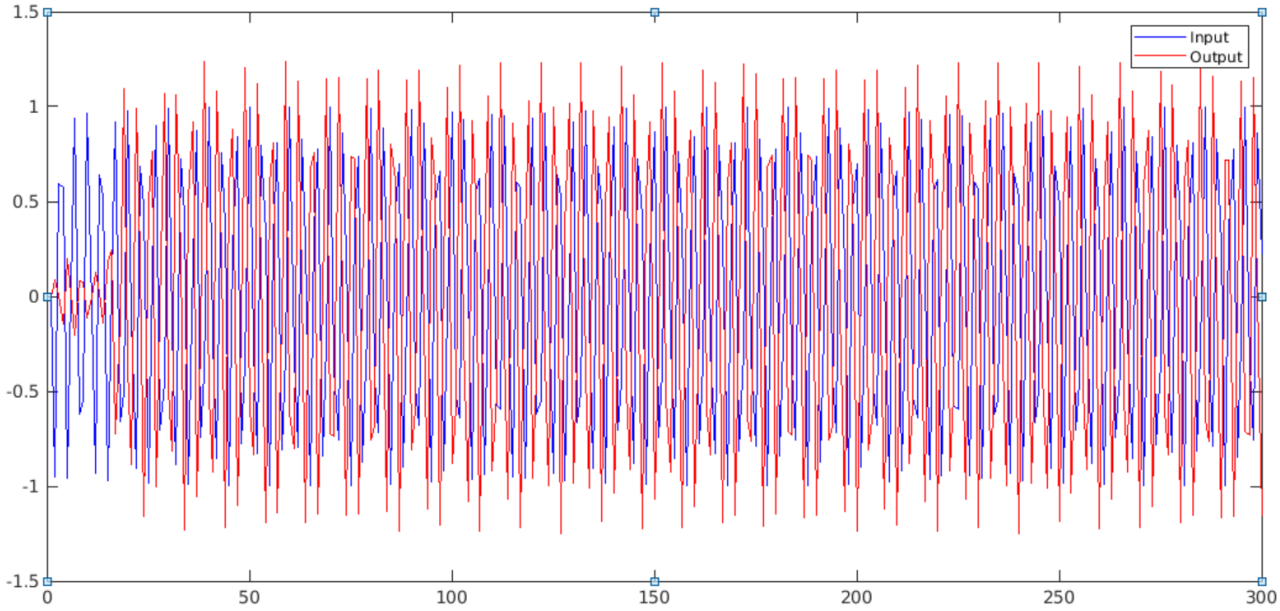
AC Signal – 32 tap FIR filter – Output plot



AC Signal – 5 tap FIR filter – I/O plot

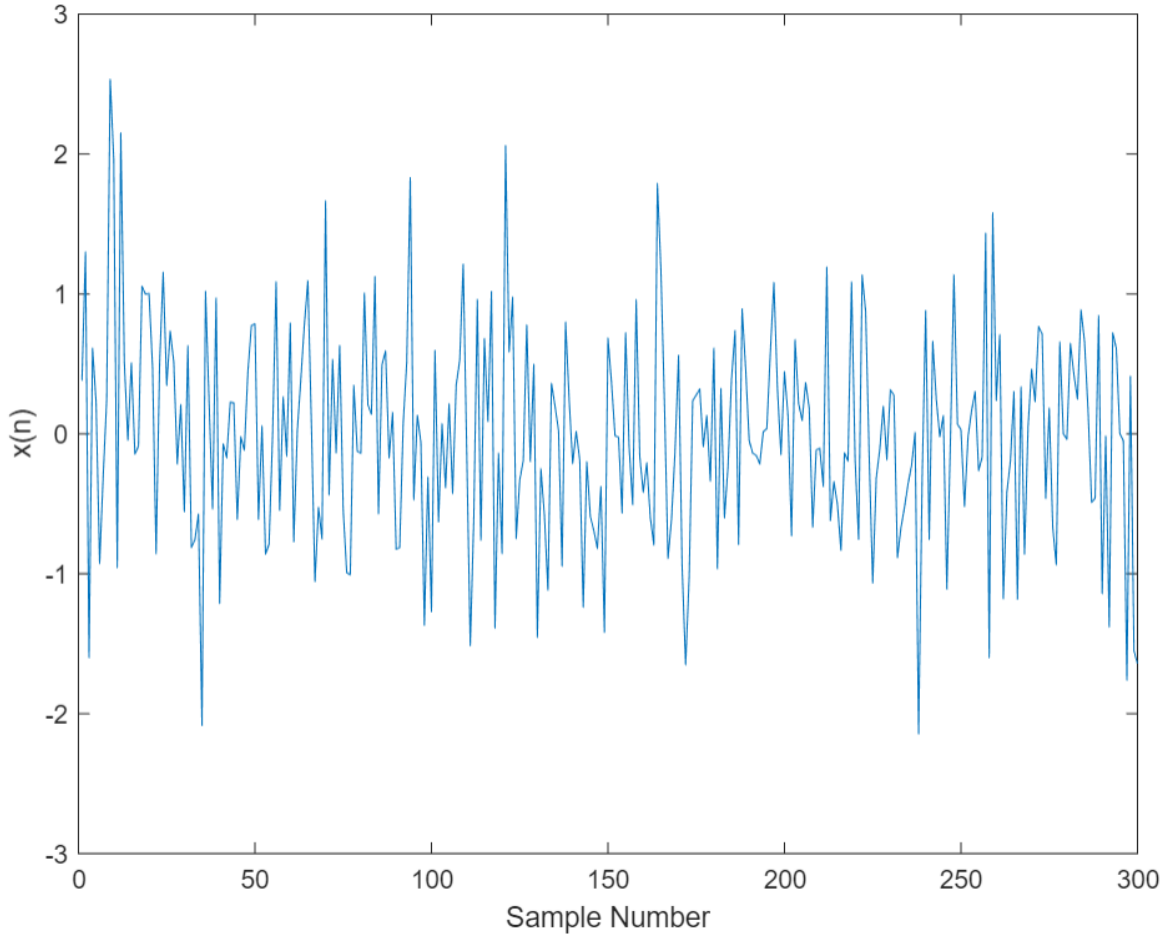


AC Signal – 32 tap FIR filter – I/O plot

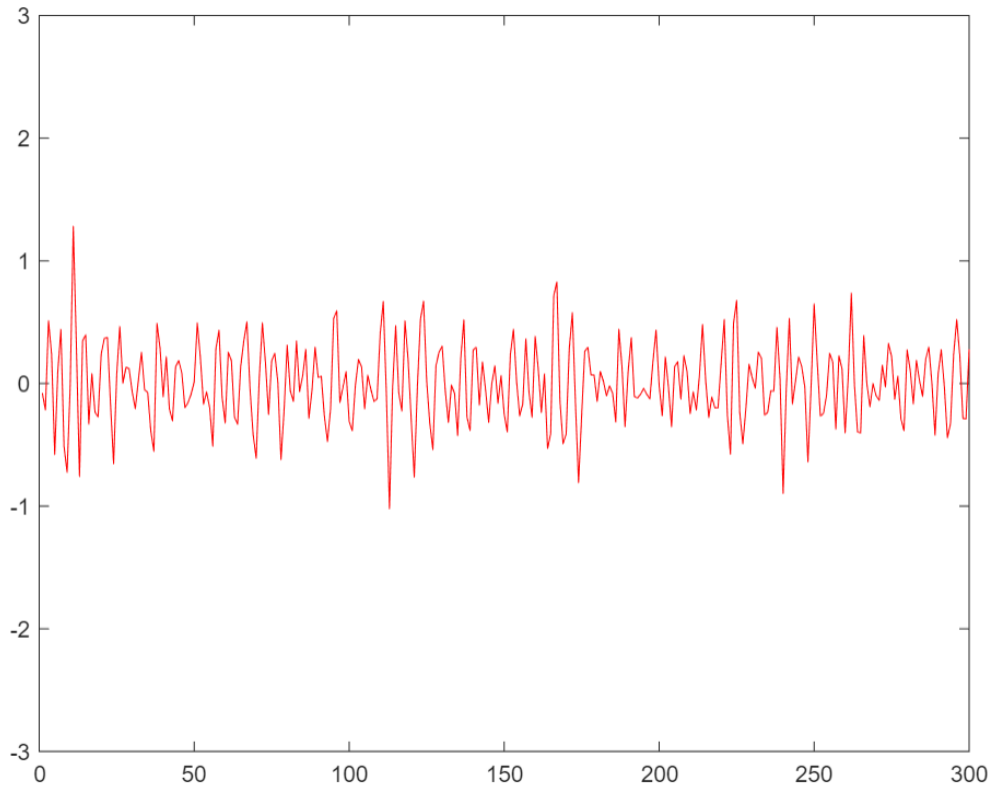


3) White Noise:

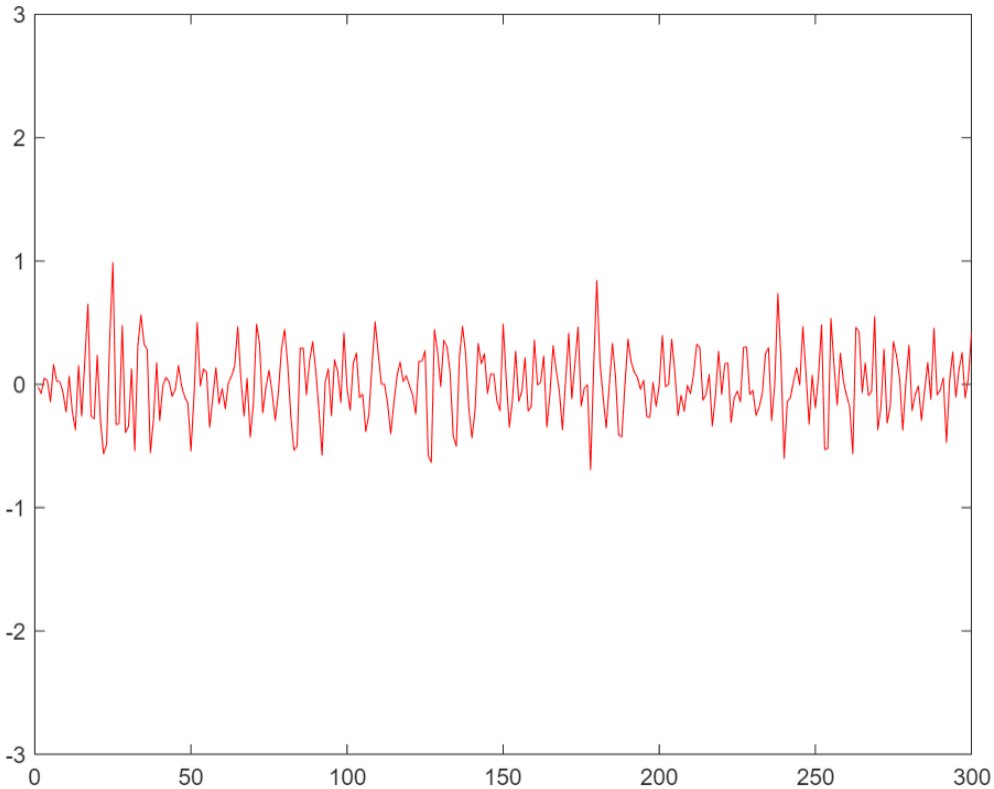
White Noise – Input Signal plot



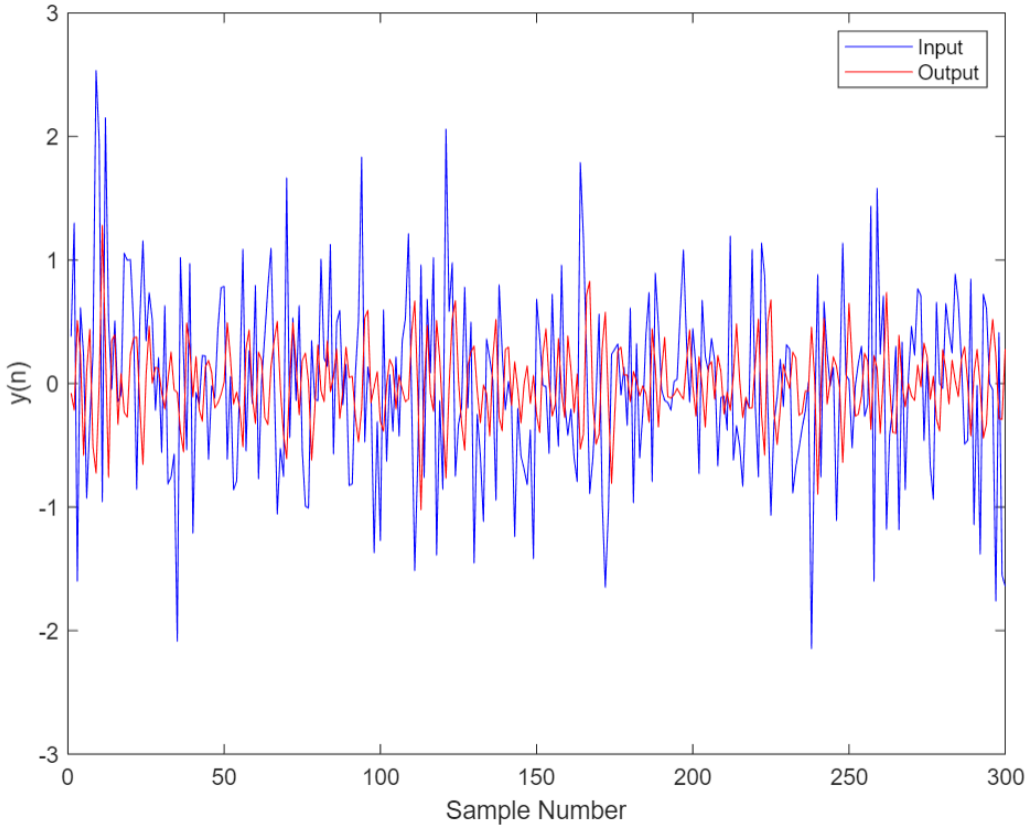
White Noise – 5 tap FIR filter – Output plot



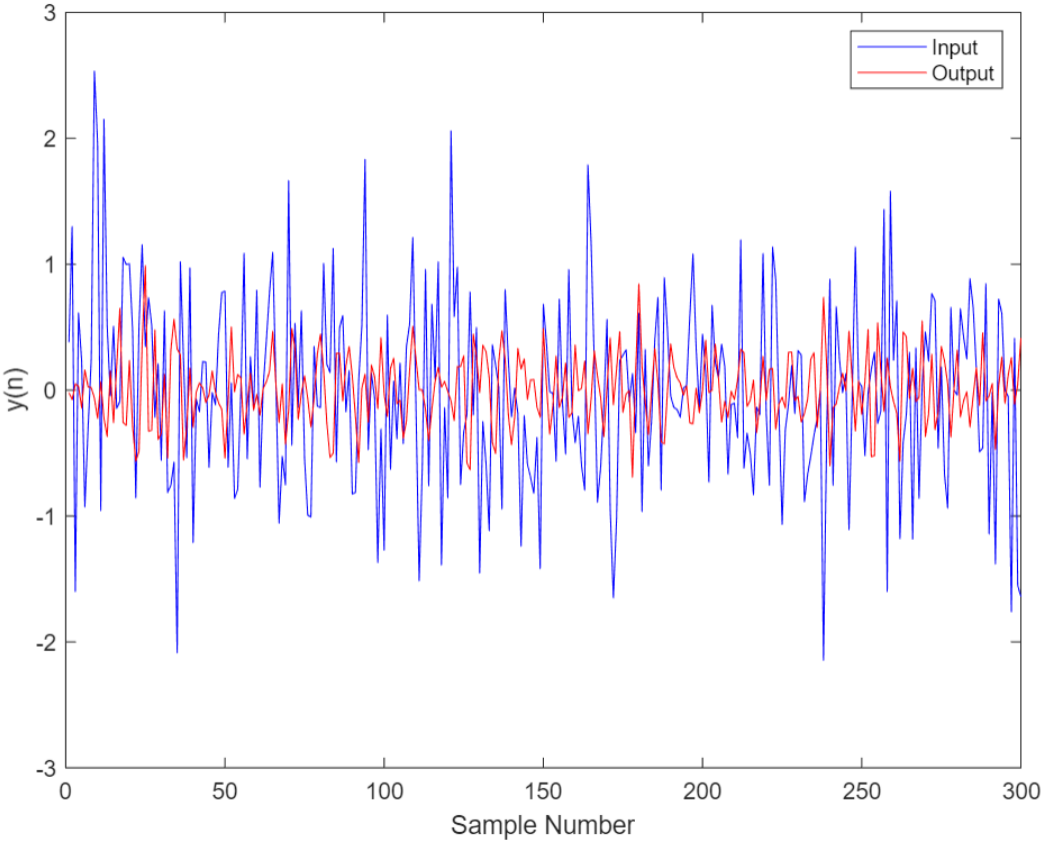
White Noise – 32 tap filter – Output plot



White Noise – 5 tap filter – I/O plot



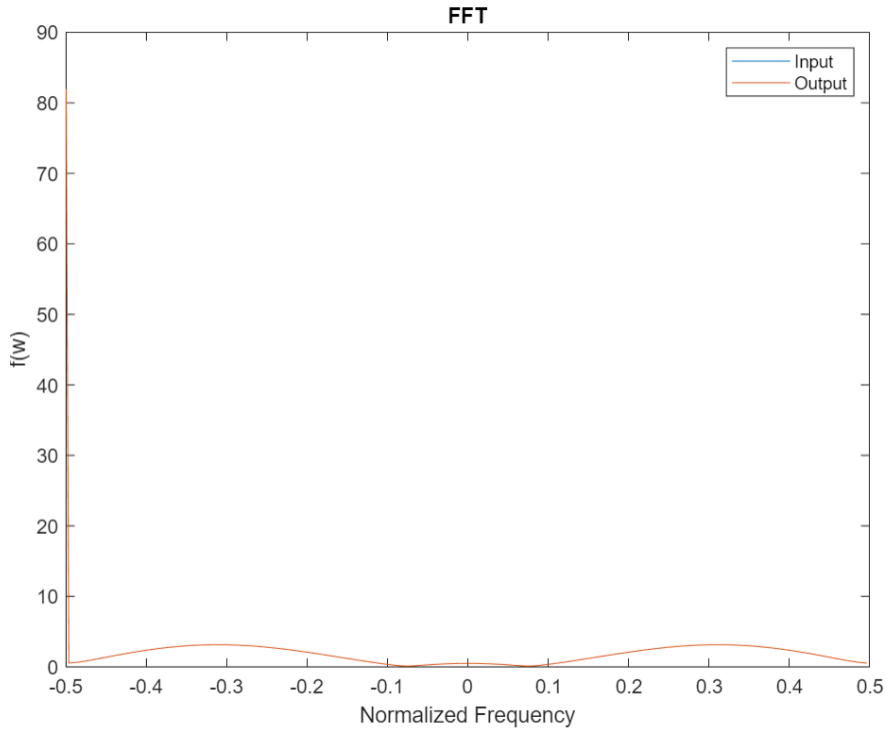
White Noise – 32 tap filter – I/O plot



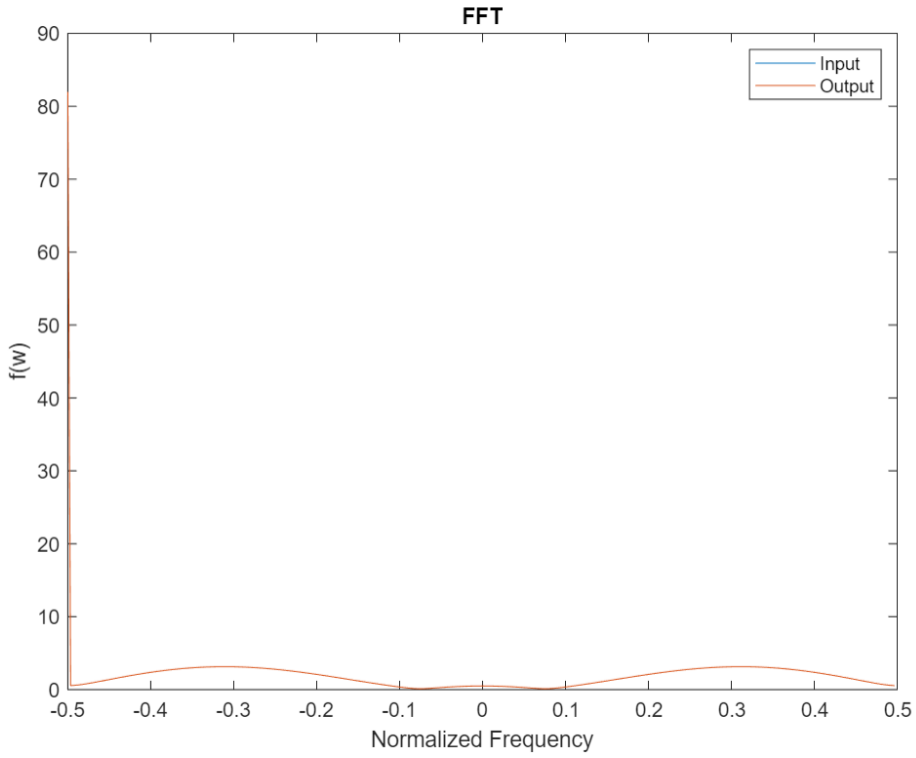
Frequency response of Input and Output Signals (Using FFT)

1) DC – Input

DC – 5 – tap

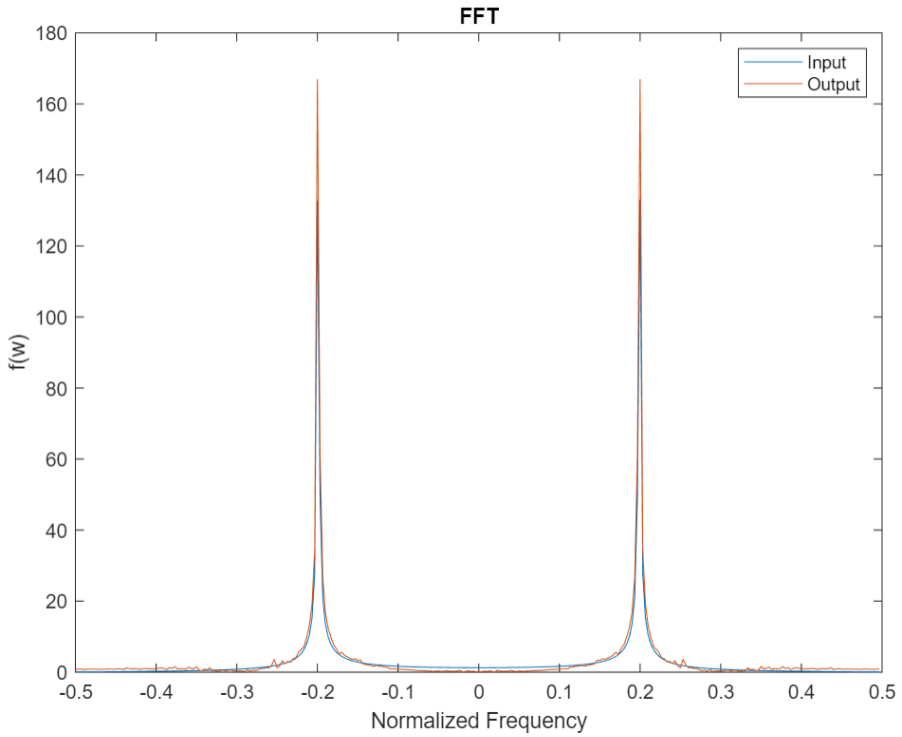


DC – 32 – tap

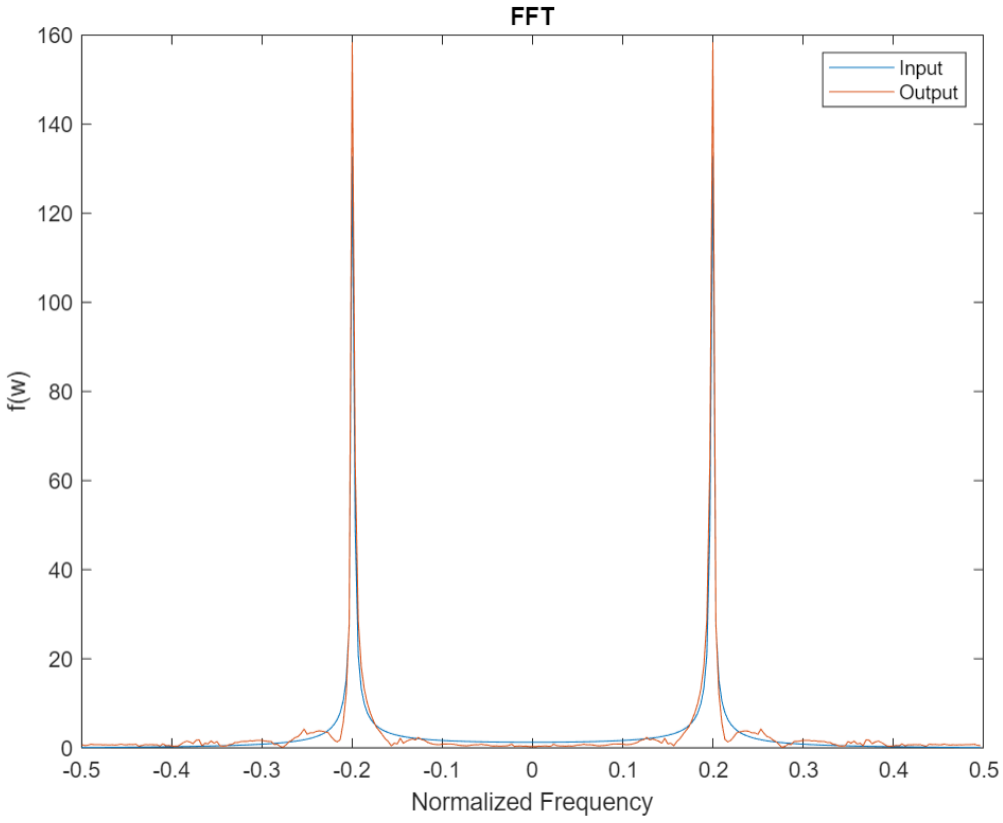


2) AC – input

AC – 5 – tap

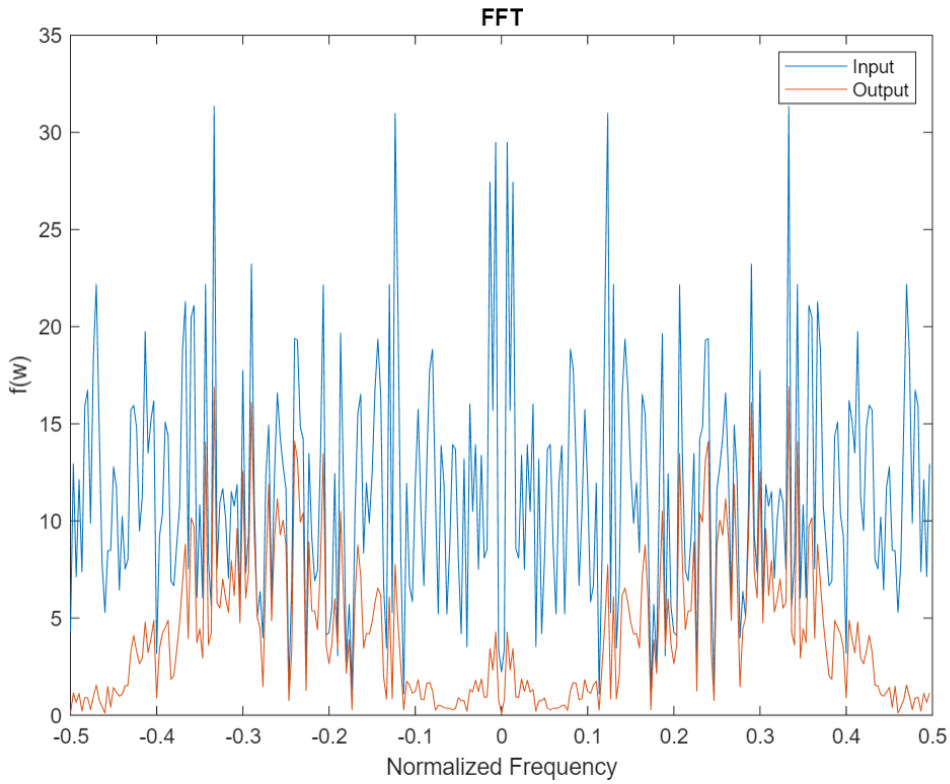


AC – 32 – tap

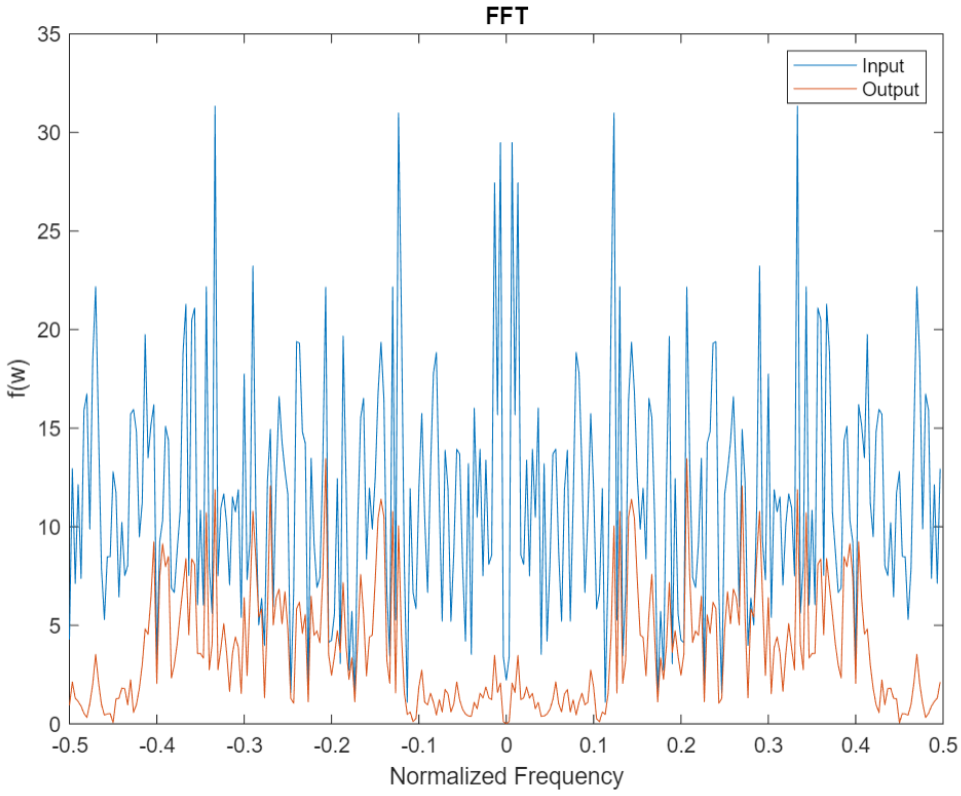


3) White Noise:

White Noise – 5 – tap

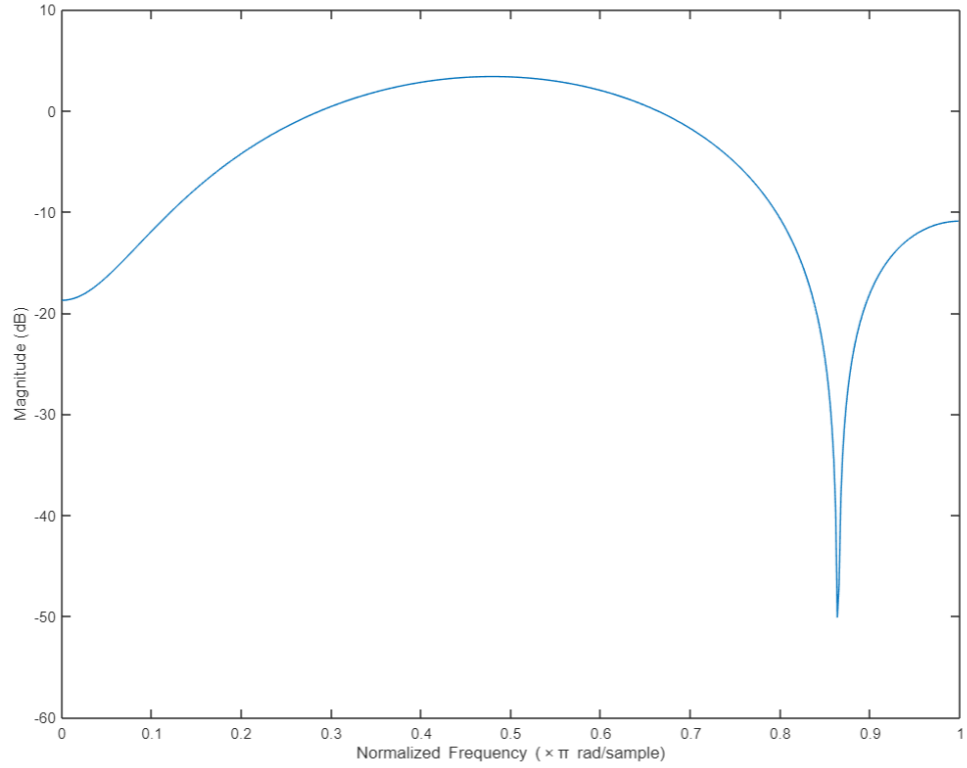


White Noise – 32 – tap

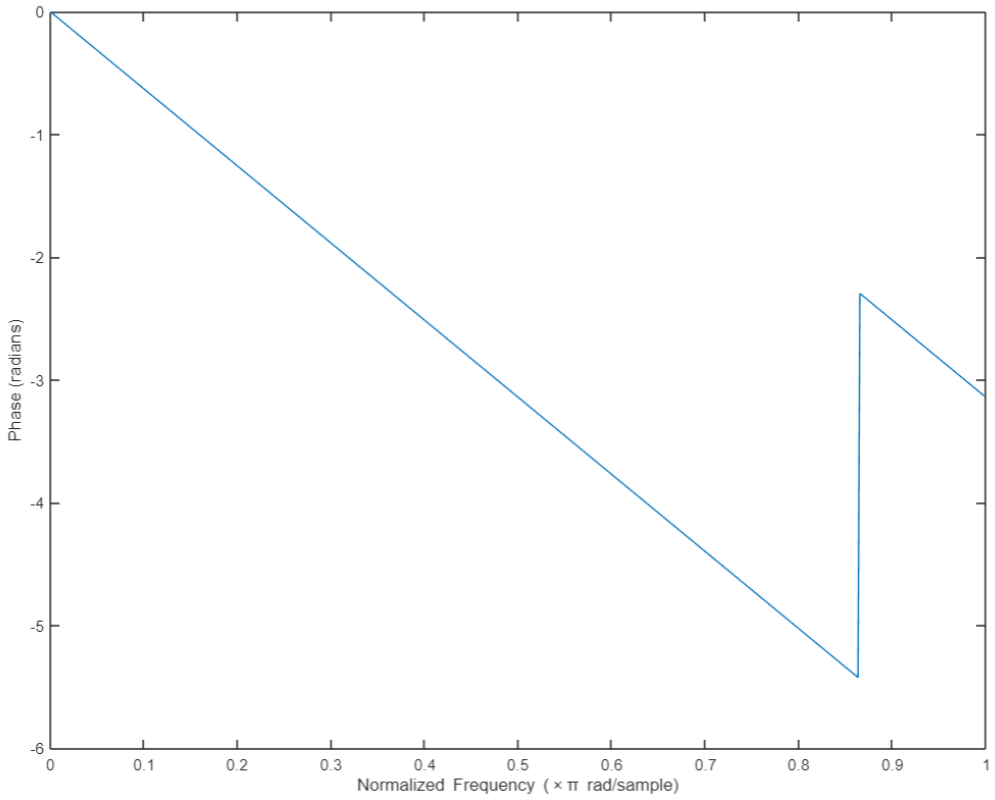


FREQUENCY RESPONSE OF THE GIVEN 5 – TAP FIR FILTER

Magnitude Response:

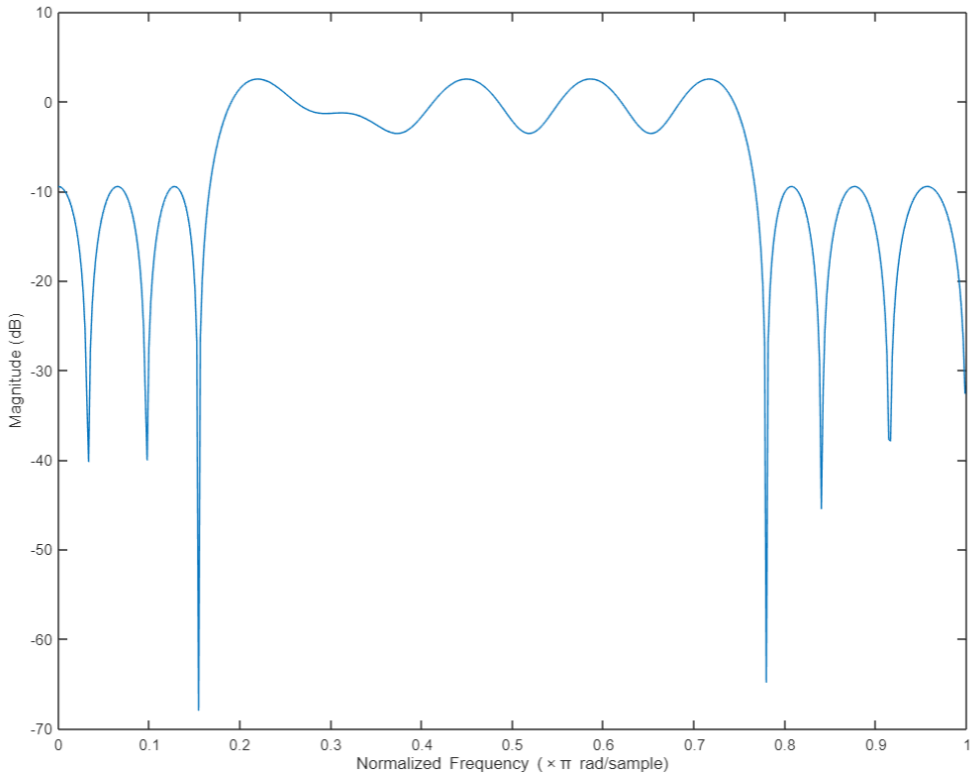


Phase Response:



FREQUENCY RESPONSE OF THE GIVEN 32 – TAP FIR FILTER

Magnitude Response:



Phase Response: