Pratyush Jaiswal
18EE30021

Date: __/__/__
Page: ____

Embedded Lab Report
Digital filter Design

AIM: Implement a filter (digital) of 6th order on an Atmega32.

Procedure:-

→ For getting the filter coefficients, fdatool of matlab is used. There a low pass filter of 6th order is created.

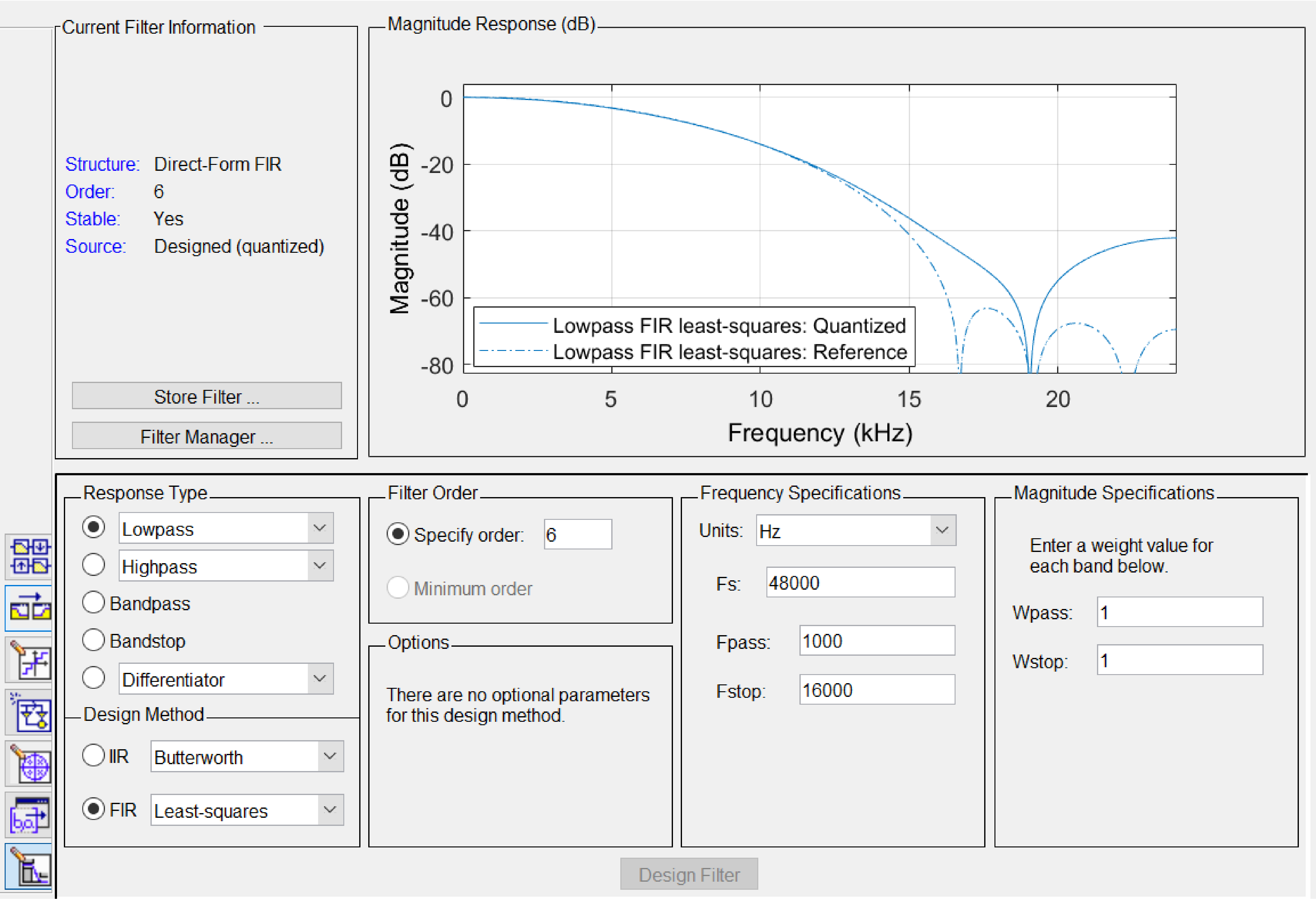
→ Then the input signal is created using two sine waves with one of the frequenices not allowed such that to be filtered.

→ The input is also normalised to an Integer between 0 & 127 (8 bits)

→ The algorithm proceeds like this:- first all values of coeffs are stored and one by one get multiplied with correspon-ding inputs & for each input value, an output is generated & stored. When the execution ends, we get the equal number of outputs as of the input.

```
;
; digital_filter.asm
;
; Created: 11-02-2021 22:58:47
; Author : Pratyush Jaiswal
;


;.INCLUDE "M32DEF.INC"
.ORG 0x00                               ; .ORG directive is used to indicate    ↪
  the begining of the address

LDI R16, HIGH(RAMEND)                   ; load R16 SFR with address of last     ↪
  SRAM location
OUT SPH, R16                            ; set upper byte of stack pointer to    ↪
  SRAM end
LDI R16, LOW(RAMEND)                    ; load R16 SFR with address of          ↪
  penultimate SRAM location
OUT SPL, R16                            ; set lower byte of stack pointer to    ↪
  SRAM end
LDI R16, 0xFF                           ; load R16 with value 0xFF
OUT DDRB, R16                           ; set PORTB as OUTPUT
LDI R16, 0x00

.EQU H0 = 3                             ; Setting the coefficients from the     ↪
  digital filter imported from matlab
.EQU H1 = 13
.EQU H2 = 29
.EQU H3 = 37
.EQU H4 = 29
.EQU H5 = 13
.EQU H6 = 3

Xin:
        ; Stroing the x-inputs
        .DB                                                                     ↪
          64,72,66,79,79,79,92,86,93,100,94,106,105,104,116,108,114,120,112,123 ↪
          ,120,117,127,118,122,126,116,125,120,115,124,112,114,116,104,112,105, ↪
          98,105,92,93,94,80,87,79,71,77,64,64,64,51


        .EQU Xlength = 50               ; storing the total length of the       ↪
          inputs

#define Yout 256                        ; defining for location at which the    ↪
  output is getting stored

#define RC R19                          ; defining a termporary variable
#define RX R20

#define RCount R18                      ; a counter for taking care of the      ↪
  number of inputs processed
```

```
#define AC1 R17                         ; accumulator(HIGH) where the final asnwer would be stored
#define AC0 R16                         ; accumulator(LOW)

LDI RCount, (XLength)                   ; storing the total number of inputs in the counter
LDI ZL, LOW(2*Xin)                      ; .DB stores the data in such a manner like 1st data in low bits of ROM location and another in high bits of ROM location
LDI ZH, HIGH(2*Xin)                     ; SO in each ROM location 2 data's stored hence Z is assigned as 2*Xin to locate the right data in the ROM memroy

LDI YL, LOW(Yout)                       ; to store the memeory location at which output is written
LDI YH, HIGH(Yout)

; for stroing the last 5 values of past inputs
; like for y[n], they will store: x[n-1], x[n-2], x[n-3], ...., x[n-6]
LDI R21, 0
LDI R27, 0
LDI R26, 0
LDI R25, 0
LDI R24, 0
LDI R23, 0
LDI R22, 0

start:
        ; initialising the accumulator value to zero
        LDI AC0, 0
        LDI AC1, 0

        ; Performing the multiplication between the coeffs and the corresponding x values and adding them to the accumulator

        ;h[0]*x[7]
        LDI RC, H0
        LPM R22, Z+
        MULS RC, R22
        ; adding the multiplication result to the accumulators
        ADD AC0, R0
        ADC AC1, R1

        ;h[0]*x[7]+h[1]*x[6]
        LDI RC, H1
        MULS RC, R23
        ; adding the multiplication result to the accumulators
        ADD AC0, R0
        ADC AC1, R1

        ;h[0]*x[7]+h[1]*x[6]+h[2]*x[5]
        LDI RC, H2
        MULS RC, R24
        ; adding the multiplication result to the accumulators
        ADD AC0, R0
        ADC AC1, R1

        ;h[0]*x[7]+h[1]*x[6]+h[2]*x[5]+h[3]*x[4]
        LDI RC, H3
        MULS RC, R25
        ; adding the multiplication result to the accumulators
        ADD AC0, R0
        ADC AC1, R1

        ;h[0]*x[7]+h[1]*x[6]+h[2]*x[5]+h[3]*x[4]+h[4]*x[3]
        LDI RC, H4
        MULS RC, R26
        ; adding the multiplication result to the accumulators
        ADD AC0, R0
        ADC AC1, R1

        ;h[0]*x[7]+h[1]*x[6]+h[2]*x[5]+h[3]*x[4]+h[4]*x[3]+h[5]*x[2]
        LDI RC, H5
        MULS RC, R27
        ; adding the multiplication result to the accumulators
        ADD AC0, R0
        ADC AC1, R1

        ;h[0]*x[7]+h[1]*x[6]+h[2]*x[5]+h[3]*x[4]+h[4]*x[3]+h[5]*x[2]+h[6]*x[1]
        LDI RC, H6
        MULS RC, R21
        ; adding the multiplication result to the accumulators
        ADD AC0, R0
        ADC AC1, R1

        ; left logical shifting the lower 8 bits and rolling the upper 8 bits for handling overflows
        LSL AC0
        ROL AC1


        ST Y+, AC1                      ; storing the output values and incrementing the location
        OUT PORTB, AC1                  ; outputting the data

        ; doing the shifting process or can say moving windows
        MOV R21, R27                    ; x[n-6] <- x[n-5]
        MOV R27, R26                    ; x[n-5] <- x[n-4]
        MOV R26, R25                    ; x[n-4] <- x[n-3]
        MOV R25, R24                    ; x[n-3] <- x[n-2]
        MOV R24, R23                    ; x[n-2] <- x[n-1]
        MOV R23, R22                    ; x[n-1] <- x[n]

        SBIW Z,0                        ; increasing the counter
            DEC RCount                  ; decreasing the counter managing the number of inputs
            BRNE start
            NOP
stop:
        RJMP stop
```



Current Filter Information

Structure: Direct-Form FIR
Order: 6
Stable: Yes
Source: Designed (quantized)

Response Type

● Lowpass
○ Highpass
○ Bandpass
○ Bandstop
○ Differentiator

Design Method

○ IIR Butterworth
● FIR Least-squares

Filter Order

● Specify order: 6
○ Minimum order

Options

There are no optional parameters for this design method.

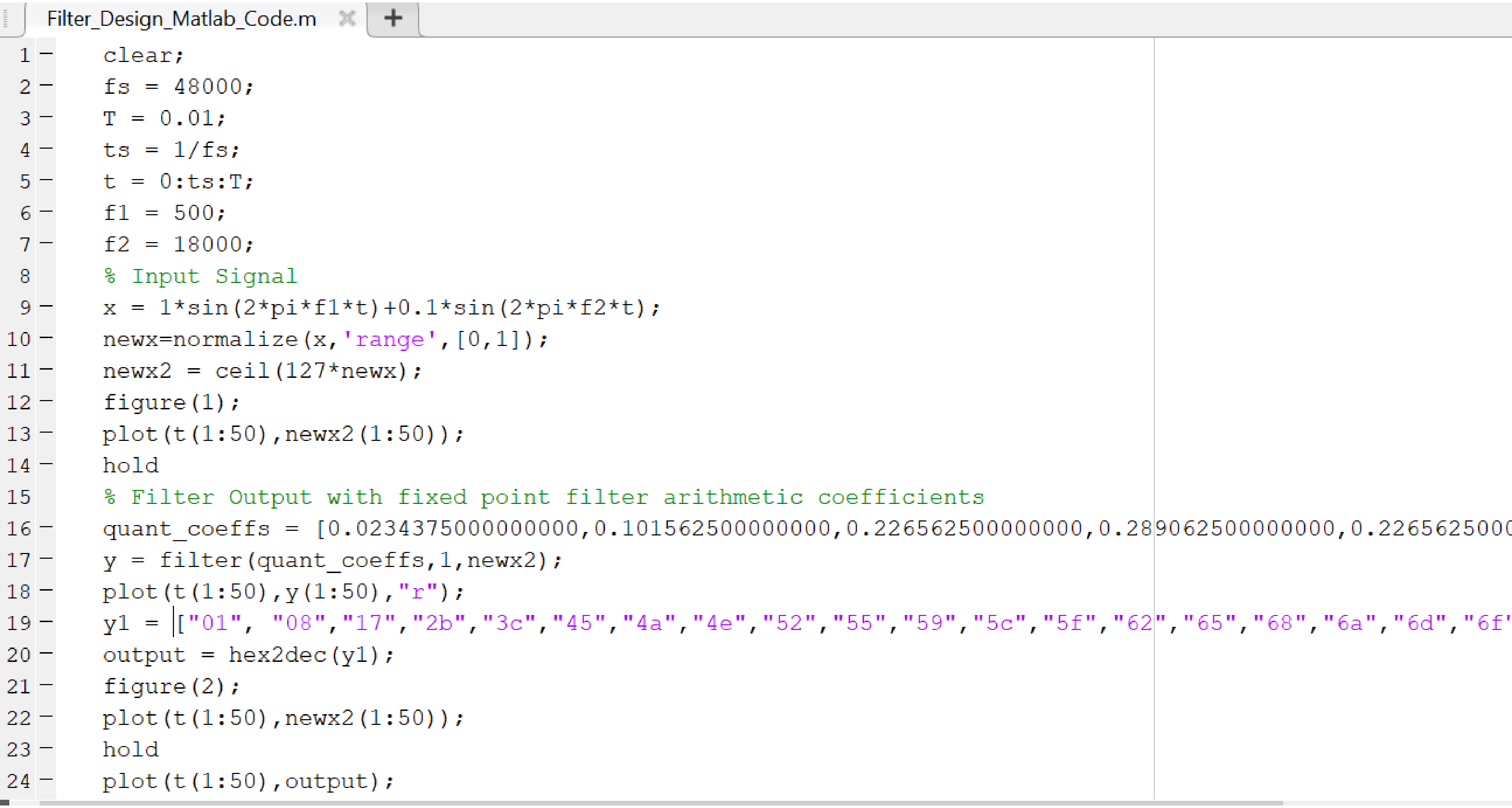
Frequency Specifications

Units: Hz
Fs: 48000
Fpass: 1000
Fstop: 16000

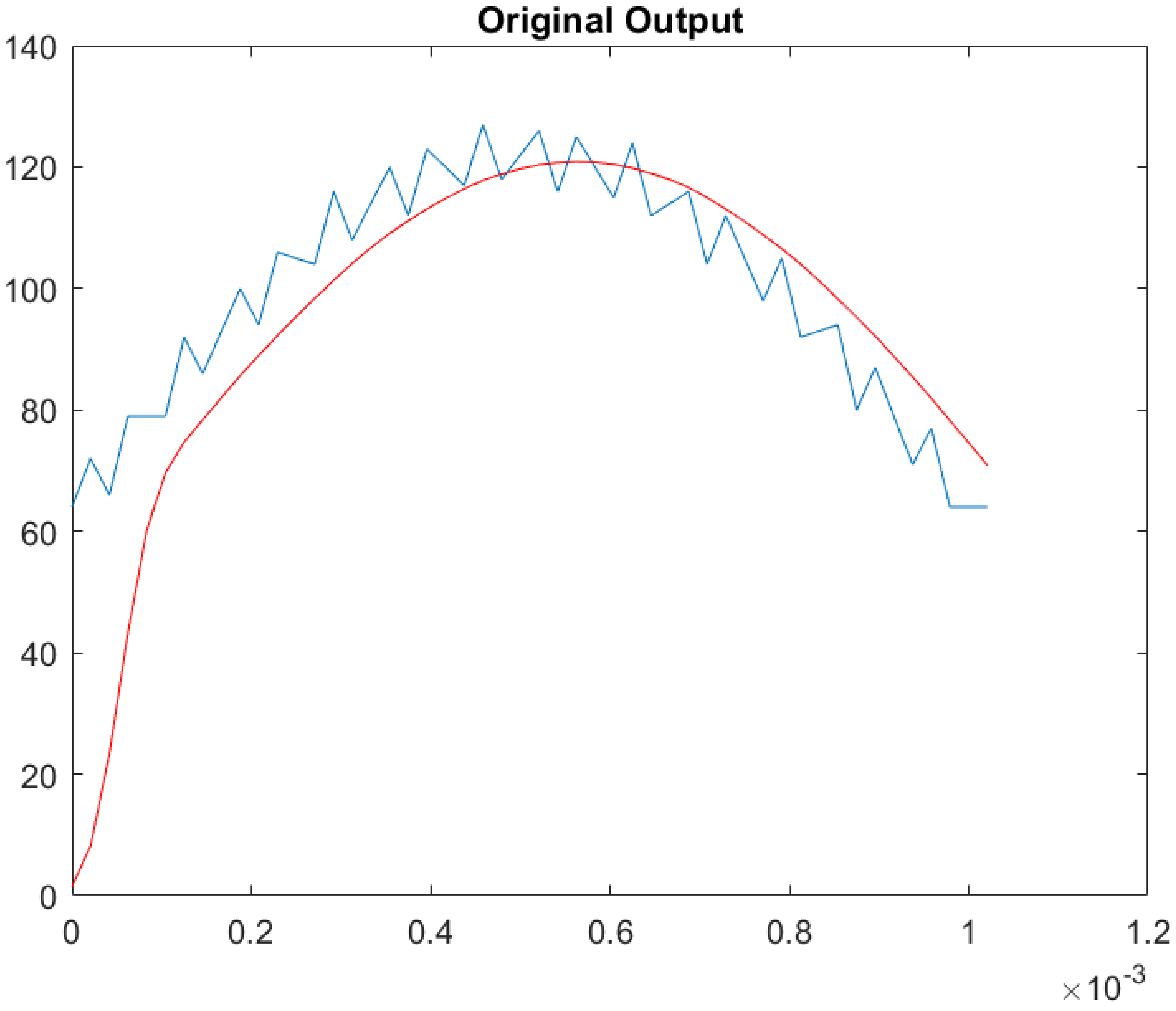
Magnitude Specifications

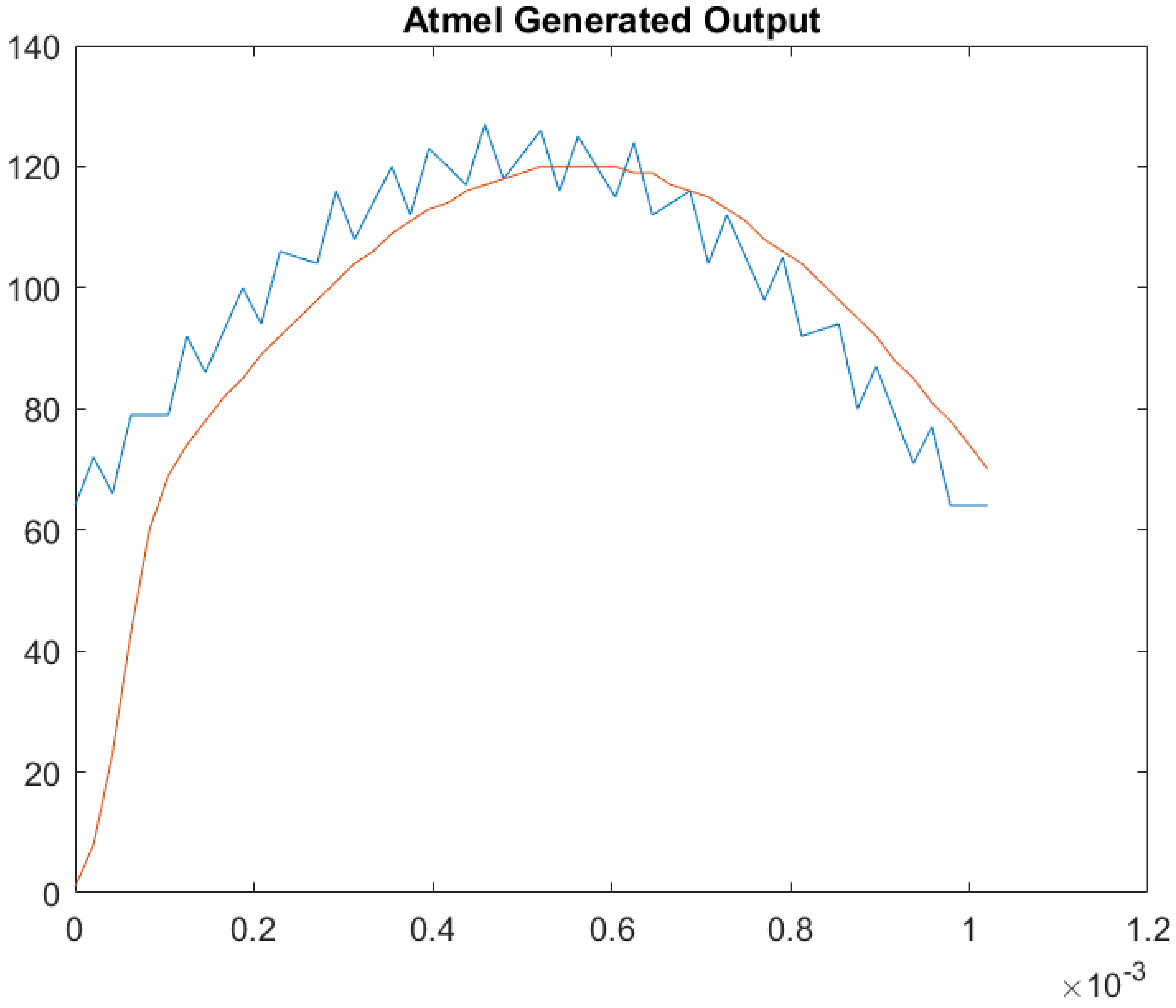
Enter a weight value for each band below.

Wpass: 1
Wstop: 1



```
clear;
fs = 48000;
T = 0.01;
ts = 1/fs;
t = 0:ts:T;
f1 = 500;
f2 = 18000;
% Input Signal
x = 1*sin(2*pi*f1*t)+0.1*sin(2*pi*f2*t);
newx=normalize(x,'range',[0,1]);
newx2 = ceil(127*newx);
figure(1);
plot(t(1:50),newx2(1:50));
hold
% Filter Output with fixed point filter arithmetic coefficients
quant_coeffs = [0.023437500000000,0.101562500000000,0.226562500000000,0.289062500000000,0.2265625000
y = filter(quant_coeffs,1,newx2);
plot(t(1:50),y(1:50),"r");
y1 = ["01", "08","17","2b","3c","45","4a","4e","52","55","59","5c","5f","62","65","68","6a","6d","6f"
output = hex2dec(y1);
figure(2);
plot(t(1:50),newx2(1:50));
hold
plot(t(1:50),output);
```


Original Output
140
120
100
80
60
40
20
0
0
0.2
0.4
0.6
0.8
1
1.2
×10-3



Atmel Generated Output

Discussion & Conclusion :-

- FIR Stands for Finite Impulsed Response and its transfer function is

$$y[n] = \sum_{i=0}^{6} C_i\, x[n-i] \;.$$

{for 6th order}.

- The fdatool provided by MATLAB, provides the coefficients whose values as in the range [-1, 1], those are multiplied by 128 to make them compatible for integer.
  Finally only the 8MSB bits AC are used which makes them scaled down to original by dividing 128.

- The final output is noise free filtered smooth signal.

- In the output it can be seen that the first few values have high error. This can be explained as for first few (7) inputs, the other coefficients don't come into play as their corresponding input values would be zero.
  This won't be a problem in large scale.