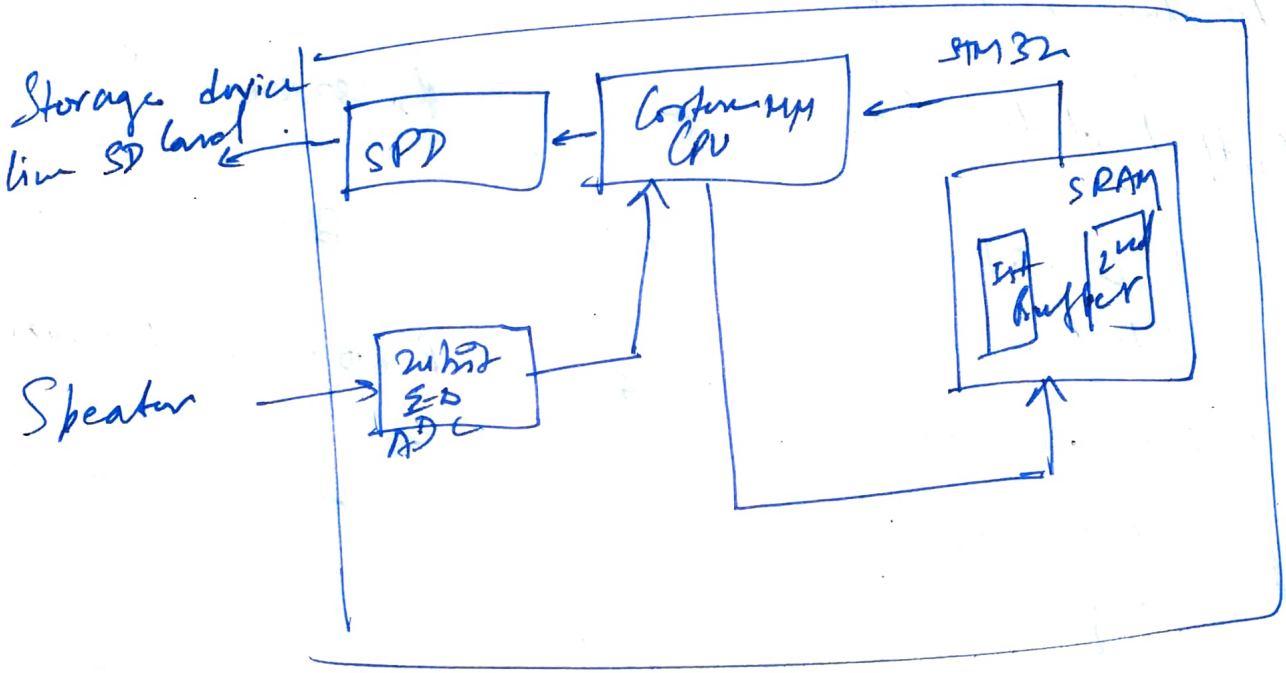
Programmable Embedded Systems

Pratyush Patwal
18EE35014

Q1:- Interfacing of 24bit ADC (sigma delta modulator type) with STM32 microcontroller where the specific signal is will processed.

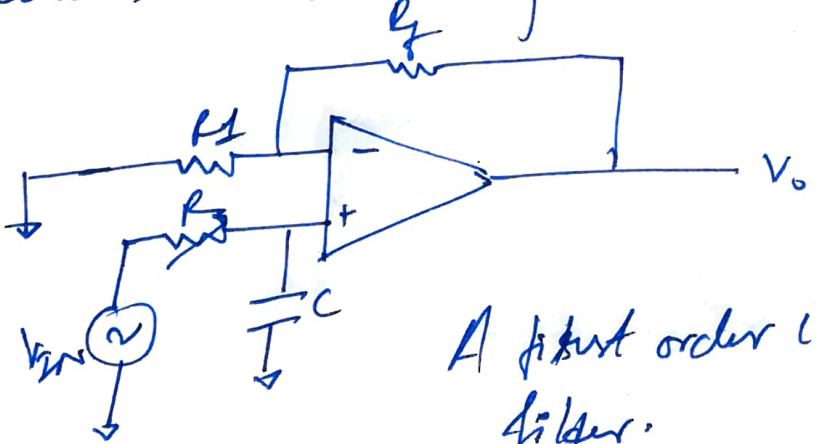


Q1. Before the ADC we use analog signal processing circuits. Mainly we need the following:

(1) Avoid aliasing:

Aliasing is ~~the~~ when frequency signal folds back its baseband of the ADC. For this anti aliasing filter is used. This is basically done to restrict the signal bandwidth to $\frac{f_s}{2}$. $B = \frac{f_s}{2}$ where f_s is the sampling frequency.

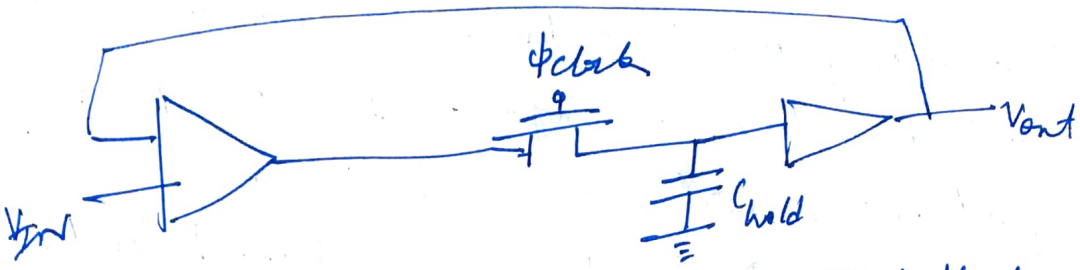
Anti-aliasing filter is nothing but a LPF. This can be realised using a butterworth filter



A first order LPF Butterworth filter.

(2) Sample and Hold

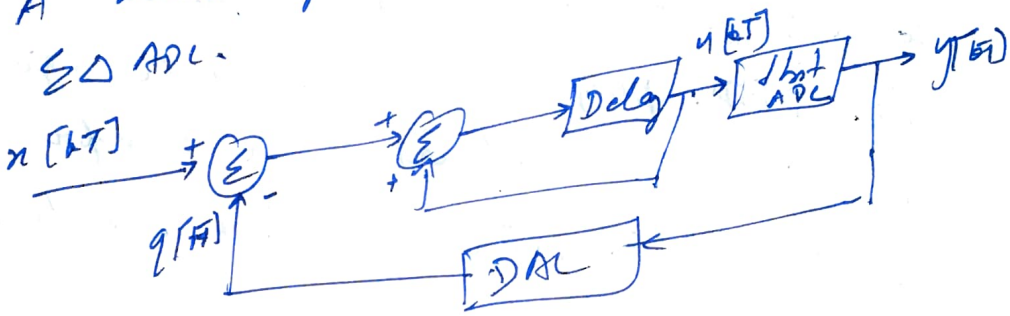
It is used to sample the input analog signal & hold for sometime while ADC converts it to digital equivalent.



A simplest Sample & hold with feedback.

Signed Delta ADC

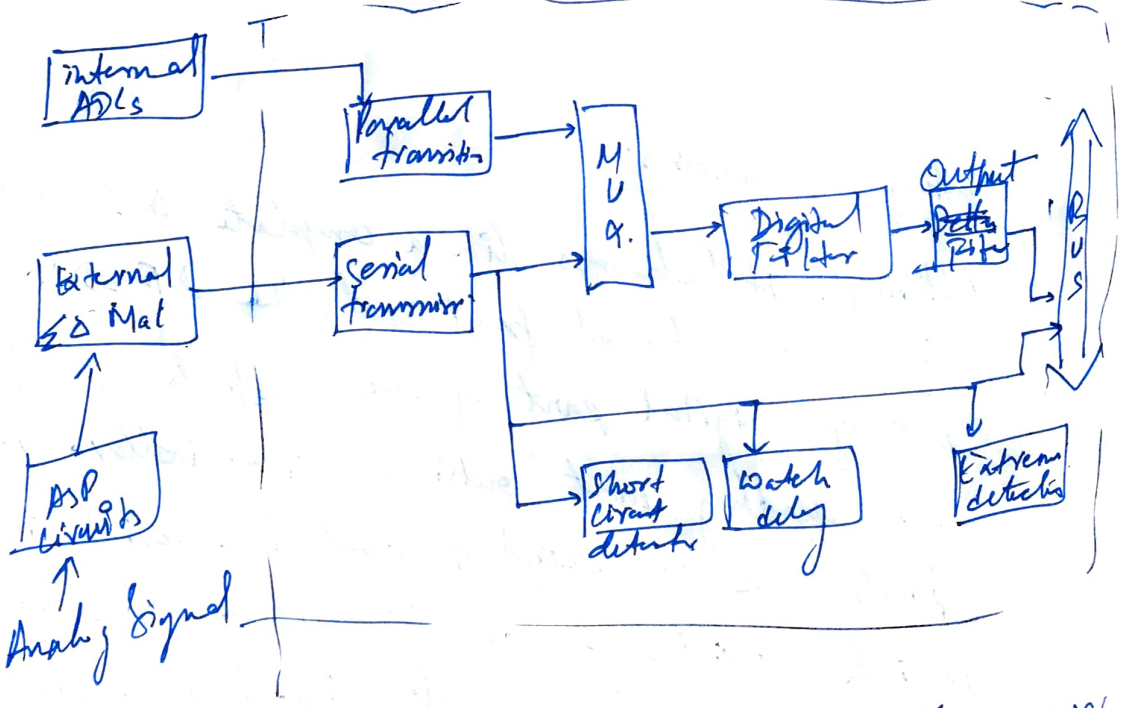
A block diagram representing 1st order $\Sigma\Delta$ ADC.



Interfacing details:-

Although it behaves like a complete A-D converter with an external front end, the DF-SDM is purely digital part of the complete setup with its analog part outside microcontroller. Both parts are connected through the serial interface which provides 1 bit $\Sigma\Delta$ stream. The average value of stream is analog i/p. This connection is as easy as only 1/2 wires are needed.

the analog part is outside the microcontroller & that is done to propose as internal digital part with wide range of features.
the DFSDM (Digital Filter for Sigma Delta Modulator) represents the digital part which is connected to analog part by fast serial interface. The external analog circuit is provided separately according to user needs, like galvanic isolation for motor control or metering applications.
The DFSDM peripherical performs a digital signal processing from external data therefore it offers a ratio solution between speed & resolution.



Block Diagram of interfacing of a ΣΔ modulator / ADC with STM32 microcontroller.

Q2. Bill of Materials:

	Parts	qty	Cost
1.	Sine filter + integrators.	4	Rs. 200×4 = Rs 800
2.	Serial trasceivers	8	Rs 290×8 = Rs 2320
3.	Analog watch dogs	4	Rs. 450×4 = Rs. 1800
4.	Output data units	4	Rs. 180×4 = Rs. 720
5.	Short circuit detectors	5	Rs 280×5 = Rs. 1400
6.	Extreme detectors	4	Rs 600×4 = Rs 2400
7.	Parallel data o/p registers	8	Rs. 250×8 = Rs. 2000
8.	DSP unit	1	Rs. 2000

Total = Rs. 13,440.

Q3. (1) Read data from memory

(2) Memory manger checks if copy in cache

if Yes — { cache hit

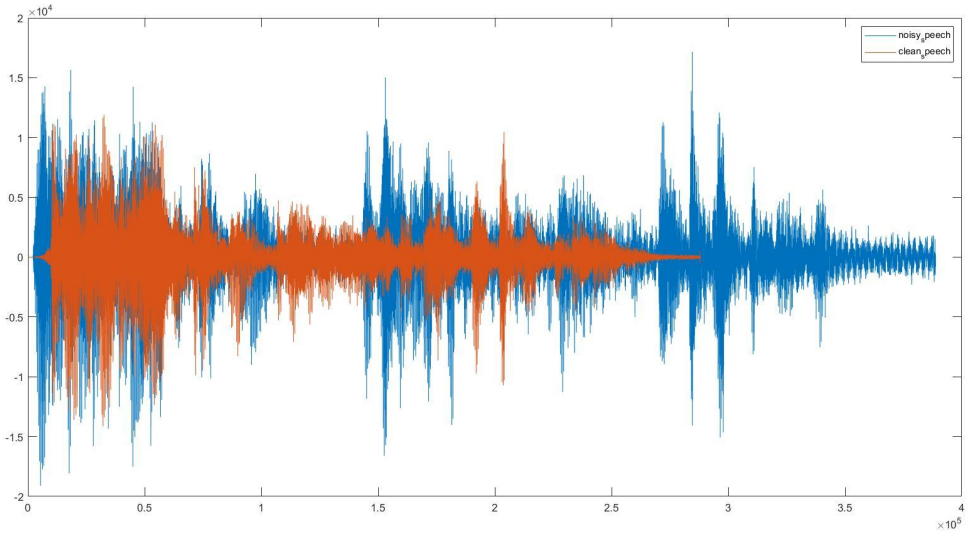
copy read in registers }.

else { cache miss:

cache is updated with row of memory from RAM }.

Q4.

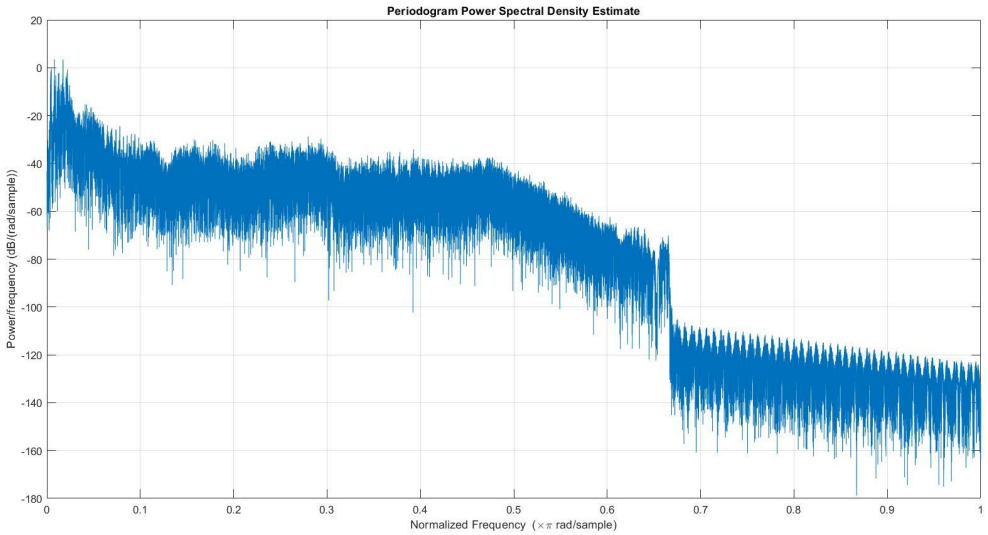
At first, the two given signals were converted to mp3 files using a VLC media player. Then the mp3 files were analysed using FFT and PSD functions of MATLAB.



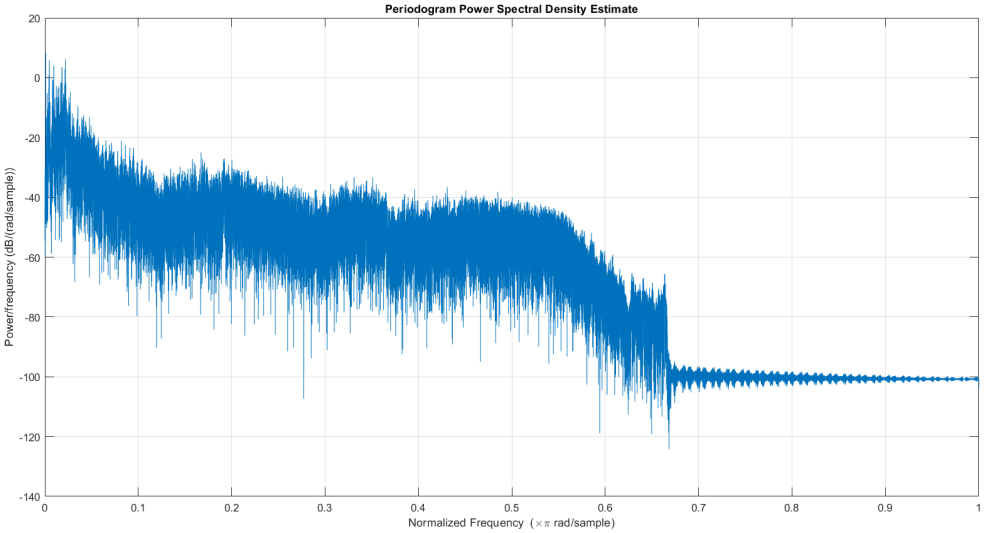
Plots of noisy and clean signals

Blue - noisy signal

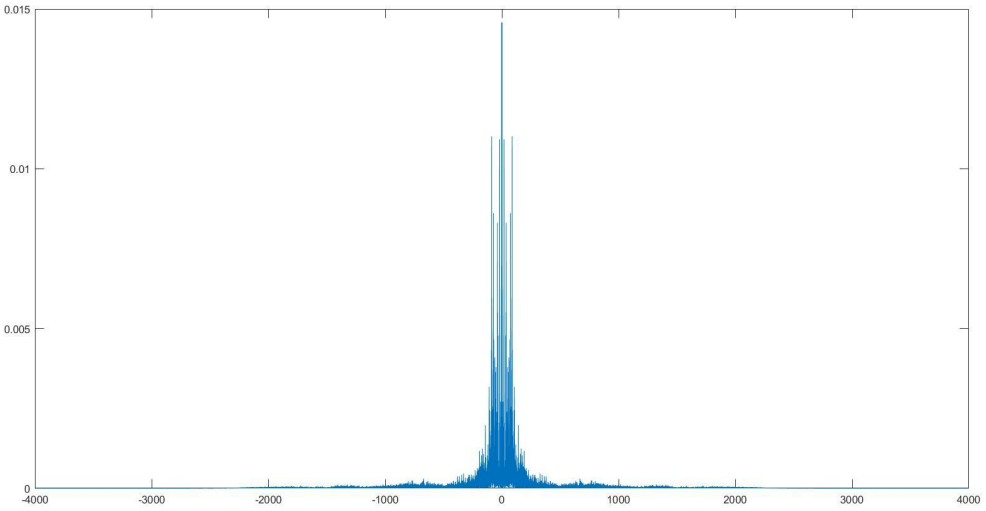
The red - clean signal



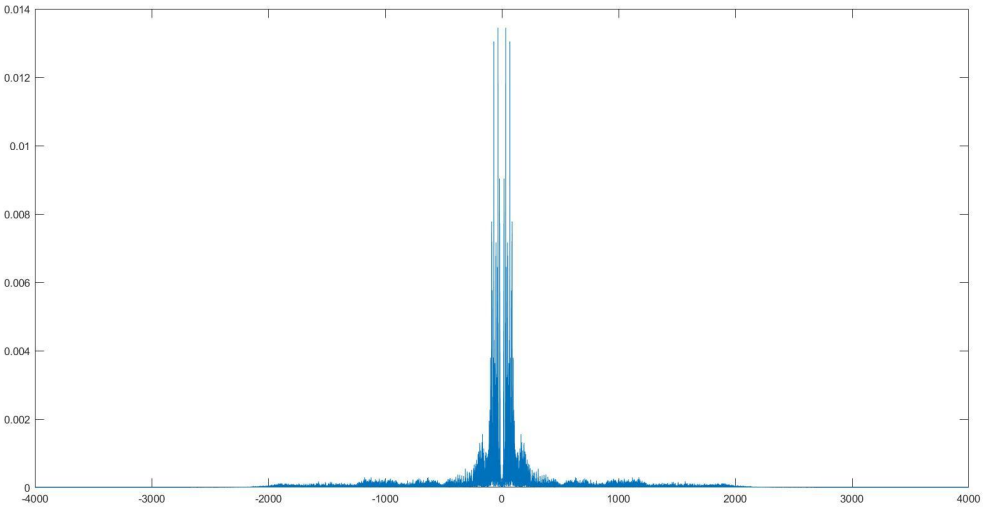
PSD OF CLEAN_SPEECH.mp3



PSD OF NOISY_SPEECH.mp3



FFT OF NOISY SPEECH



FFT of CLEAN SPEECH

CODE TO generate PSD AND FFTs

PSD CODE:

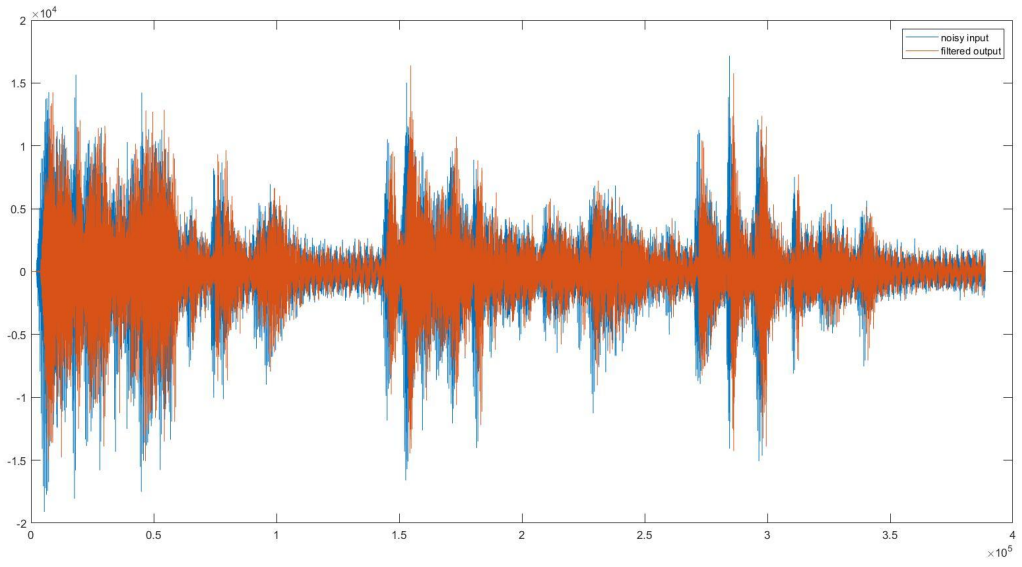
```
%Getting periodograms
%Reading two files
[y1,Fs1]  = audioread('clean_speech_mp3.mp3');
[y2,Fs2] = audioread('noisy_speech_mp3.mp3');
figure(1)
periodogram(s1);
figure(2)
periodogram(s2);
```

FFT CODE:

```
A=audioread('clean_speech.m4a');
fs = 8000;

scaler = 2^15;
A = scaler * A;
```

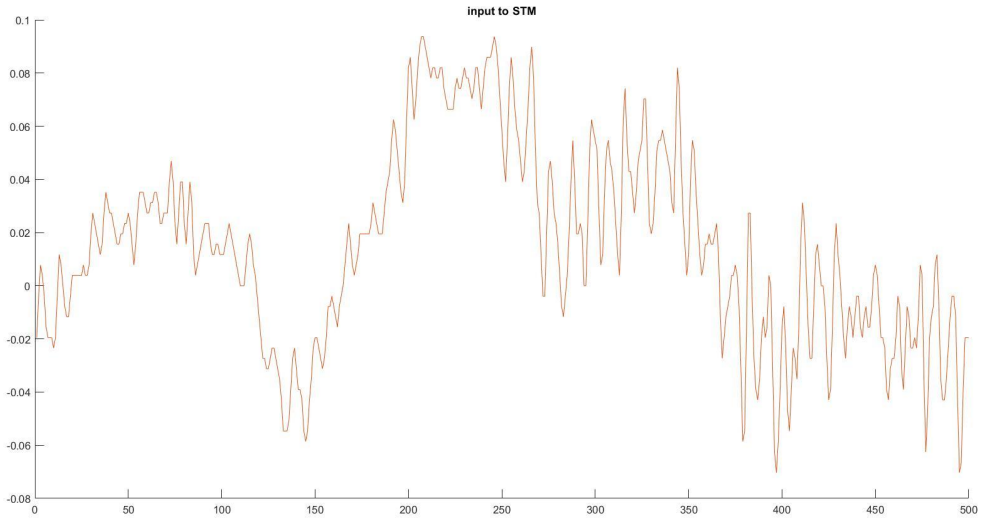
```
sig = audioread('noisy_speech.m4a');
sig = scaler * sig;

figure(2)
hold off
z=A/max(A);
N = length(z);
df = fs / N;
w = (-(N/2):(N/2)-1)*df;
y = fft(z(:, 1), N) / N;
y2 = fftshift(y);
plot(w, abs(y2));
```

Now we do COMB FILTERING ON MATLAB:

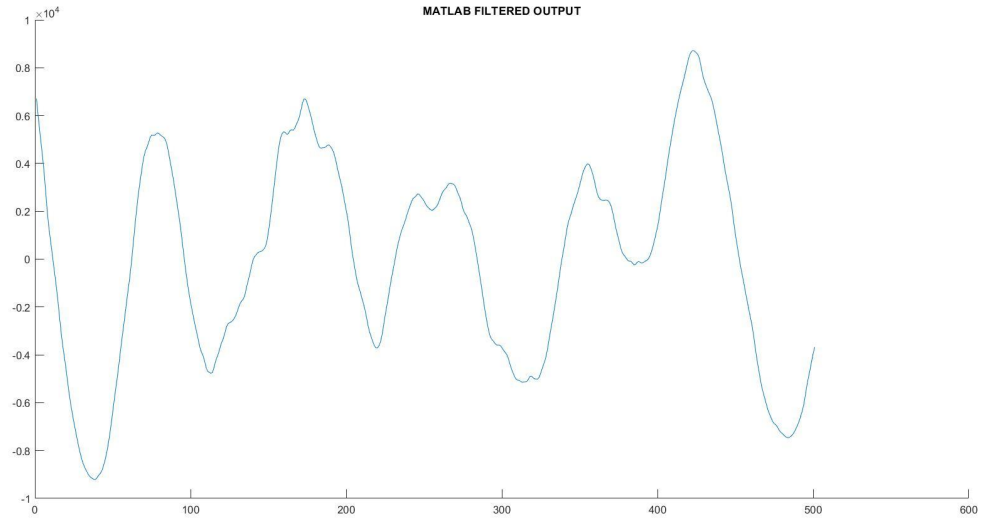


Red - MATLAB filtered output
Blue - input noisy signal.

Let’s take a look at the zoomed in versions

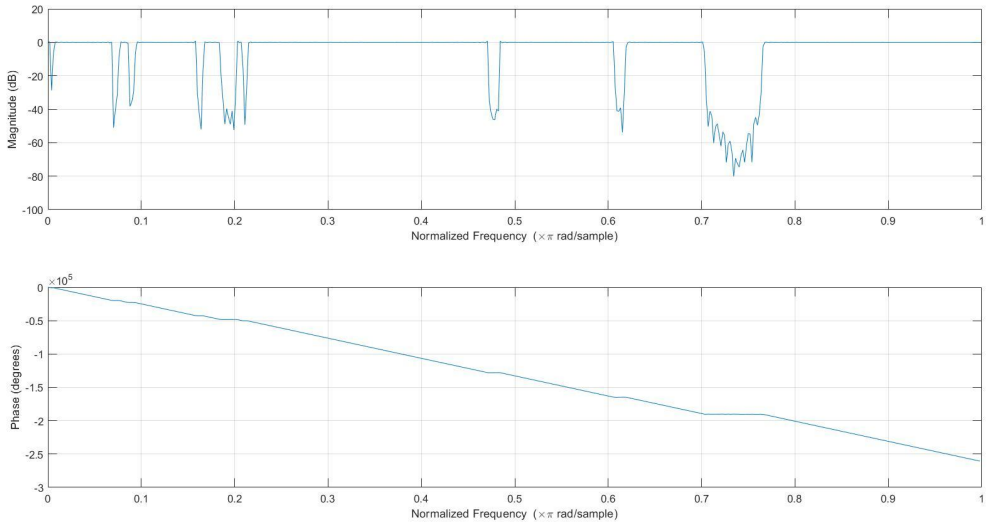


INPUT NOISY SIGNAL

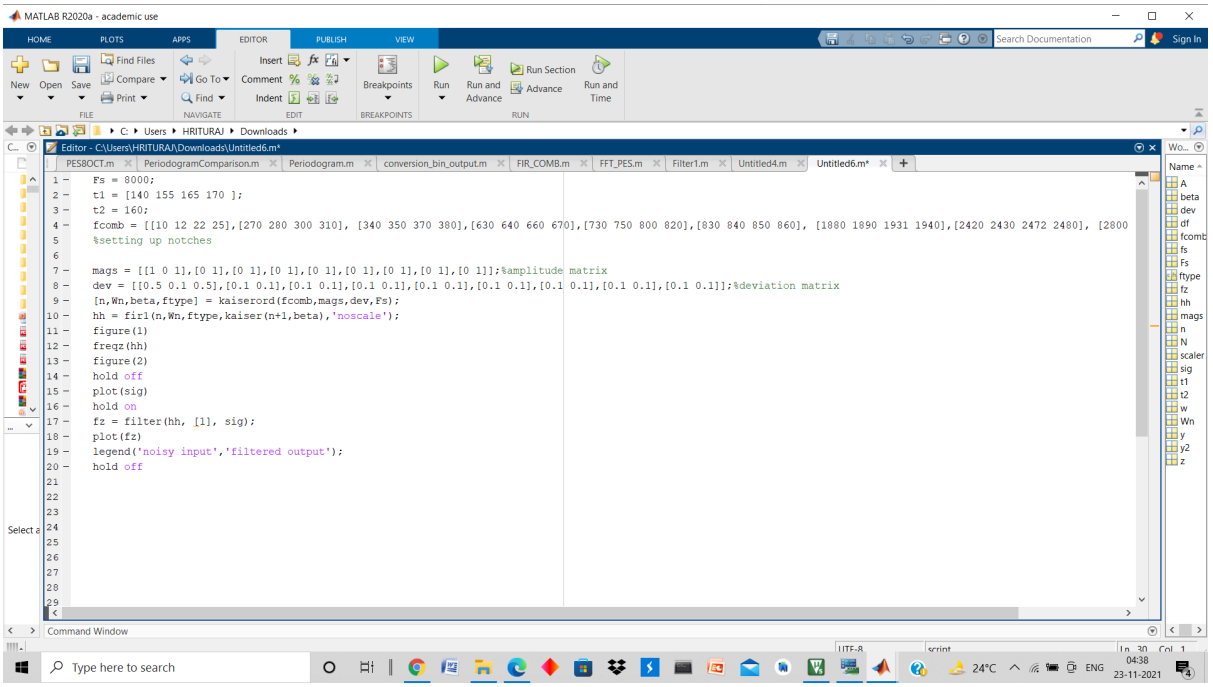


MATLAB FILTERED OUTPUT

We notice that high-frequency components disappear hence filtering takes place.



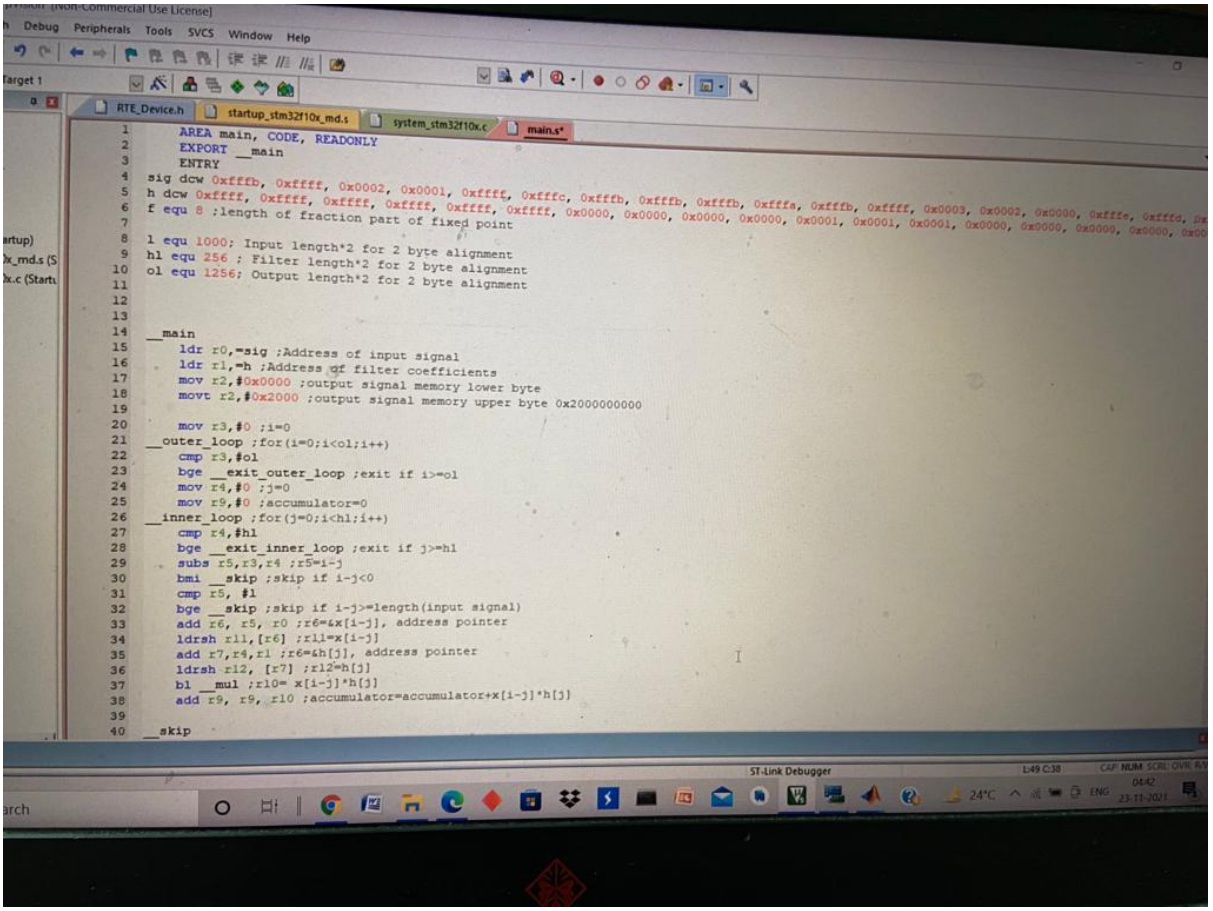
Magnitude and Phase response of Comb filter

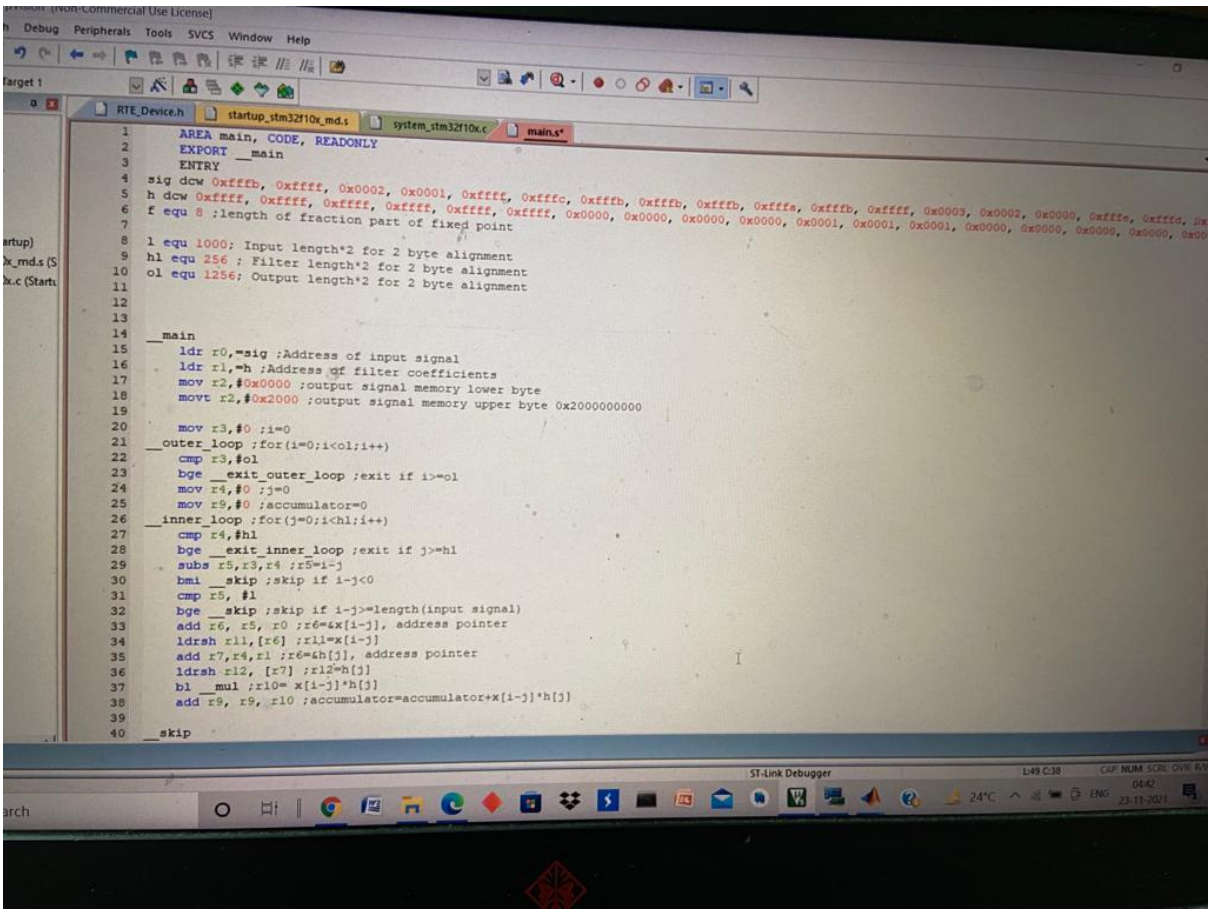


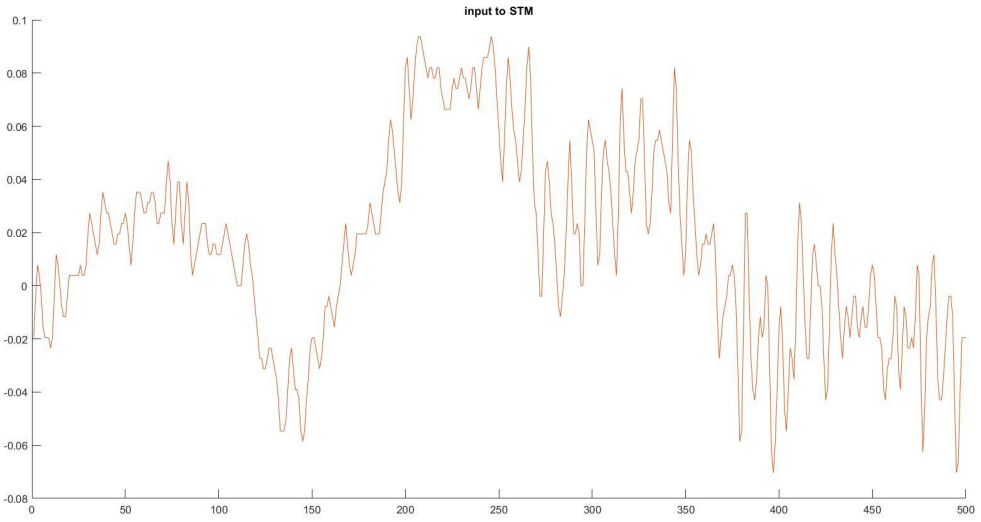
COMB FILTERING CODE ON MATLAB

Q5 and Q6

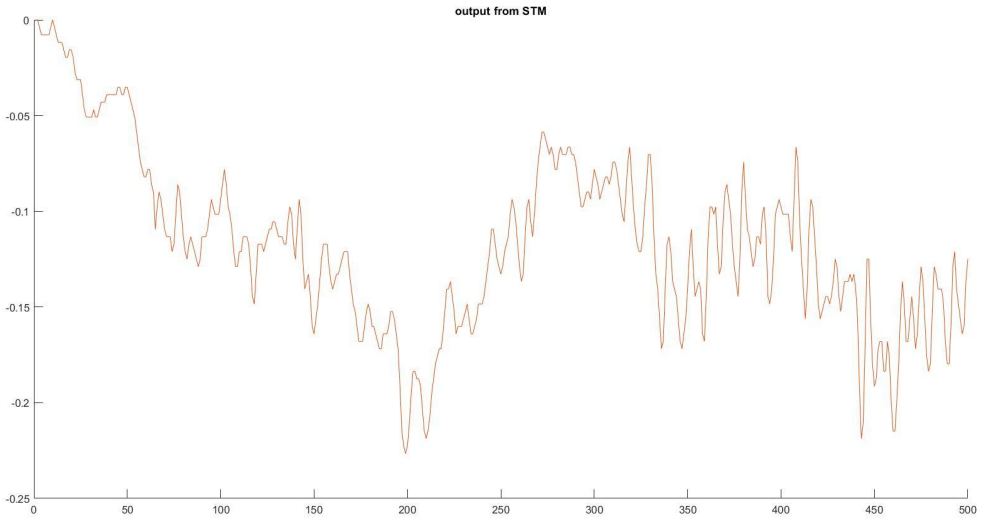
ARM code for generic FIR filter
The coefficients and input signal has been set according to our current needs.







INPUT NOISY SIGNAL



OUTPUT FROM STM

We take notice of the fact that the filtering in MATLAB resulted in a much cleaner version of the speech as compared to filtering on STM.