Assignment-1 Documentation

-Amruta Vidwans and Milap Rane

Class Definition:

combfilt.h:

We define a class CombFilt, having the components g(Gain) and tau (Delay Time), which are the parameters required by both IIR and FIR comb filter. We decided to use a single filter class object for both FIR and IIR Filter implementations, for simplicity. CombFilt class was created with FIR and IIR as filtering functions in it. The g and tau are declared outside any scope, making their scope "private".

Error_t init ():
This defines an initialization of the Filter Parameters, to g=0.5 and tau = 0.5 sec, which are defined as default values for the filter.

static Error_t create (CombFilt *&pCombFiltIf)
and
static Error_t destroy (CombFilt *&pCombFiltIf):
These do the task of assigning and freeing memory locations for instances of this particular class..

void CombFilt::GetFiltVar(float fg_ent, float ftau_ent,float fFs):
We define a public member function GetFiltVar (that stands for Get Filter Variables), which helps the user access the Filter parameters g and tau.It takes in the value ftau_ent in secs, and also takes the sampling frequency fFs as a parameter. The Multiplication (ftau_ent*fFs) gives us tau in samples. Since the user inputs tau in seconds, we convert it into samples.

void createBuffer(int iNumChannels), void clearBufer(int iNumChannels)
and void destroyBuffer(int iNumChannels):
These are used to assign, clear(i.e set to zero) and freememory locations for the Delayline buffer within the Comb Filtering functions.They take the number of channels in the audio file as a parameter.

void FIRCombFilt(float **ppfAudioData,float **ppfFiltAudio,int iNumChannels ,int iInFileLength)

and

void IIRCombFilt(float **ppfAudioData,float **ppfFiltAudio,int iNumChannels ,int iInFileLength):

These are the main functions within which the Comb Filtering process is done.It accepts for parameters, two double pointers, one concerning the input(**ppfAudioData) and the other concerning the output(**ppfFiltAudio).Also, input values accepted are the number of channels and the iInFileLength, which for a block by block analysis shall be equal to Block length.

Running the executable:

The user need to provide atleast the audio path along with the executable. The rest of the parameters are taken as default values. The user has to specify the arguments in the order:
"Path/FileName.wav" gain delay "FilterType"
The FilterType should be FIR or IIR. Example of how to call the executable is as follows:
MUSIC8903Exec /Users/Desktop/test.wav 0.9 10 IIR

The delay can be as along as the memory allows. If the delay is greater than the input signal then we get back the original signal.
The interface works for all channel configurations and sample rates.

Comparison Of Outputs of Two Files, with the MATLAB Implementation:

The audio files used are sinewave4410.wav , and sinewav441.wav, two puretone sinewaves with sampling rate of 44100 Hz of 4410Hz and 441Hz,of length 10s each.

These values are found to be very similar to each other, such that they have overlapping figures. The Red* part is the C++ implemented values and the black part is the MATLAB Implementation based plot of obtained values. The smaller amplitude signal is input signal. An analysis of the difference functions shows us that the difference between the values obtained through these implementations are in the range of 0 - 5 * 10^-6, which is close to zero(very small).The Common parameters are gain g=0.5, and tau=10.

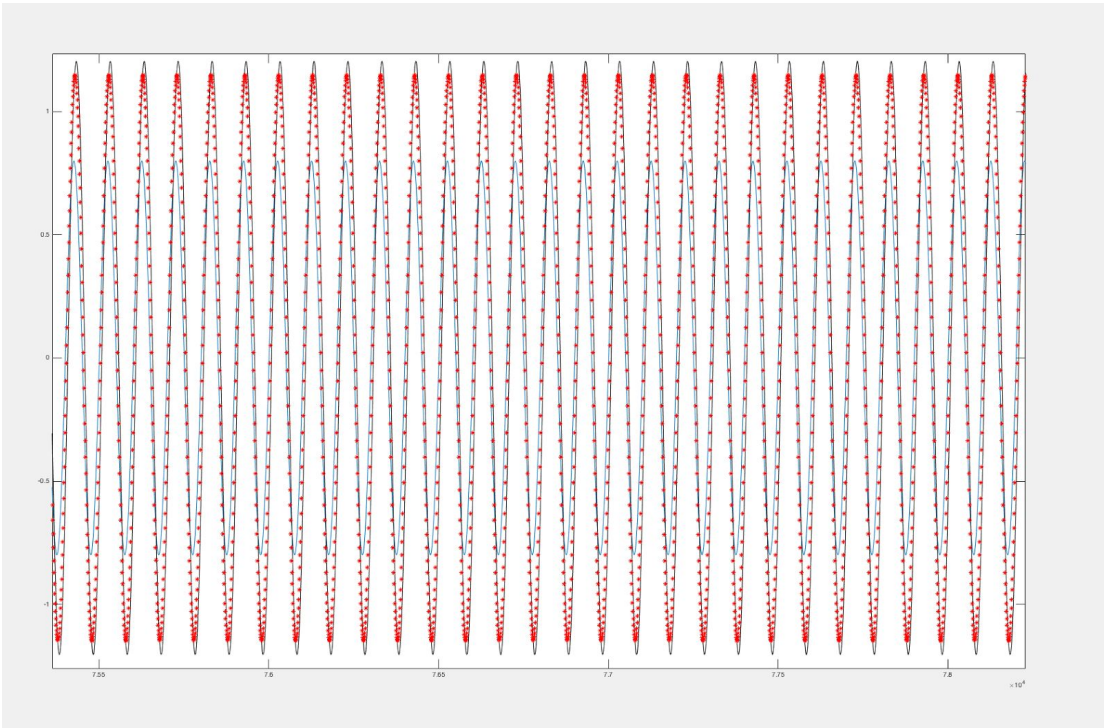

Fig-1: Sinewav4410.wav C++ vs MATLAB IIR Comb Filter Implementations

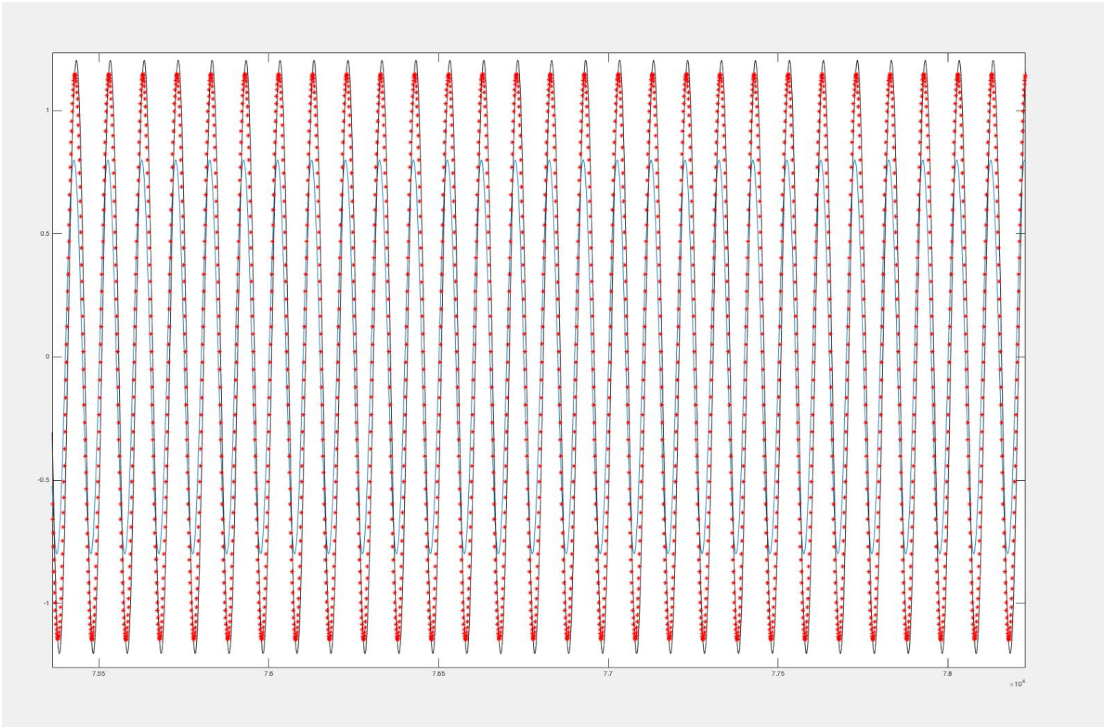

Fig-2: Sinewav4410.wav C++ vs MATLAB IIR Comb Filter Implementations

Apart from the specified test cases, we added another test where we check for different sampling rate for both IIR and FIR filter.