Ruiyan Guo 400256752
Xinyu Chen 400221680

fmMonobasic Take Home Exercise

In this section, the built-in method firwin is replaced by my own method `myCoeff(Fs, Fc, Ntaps)` , it creates the coefficients for mono audio filtering. The built-in method lfilter is replaced by my own method `mylfilter(coeff, data)`,which is the single pass filter implementation.

fmMonoblock Take Home Exercise

Same as the MonoBasic part, for mono audio filtering, the built- in method firwin is replaced by my own method `myCoeff(Fs, Fc, Ntaps)`. For block processing purposes, the lfilter is replaced by my own method `mylfilter_w_block(coeff, data, size, buffer)`, where the data is the demodulated fm data, and buffer is the array for state saving.

To implement the functionality of demodulator, my own function `myDemod(I, Q, prev_i = 0.0, prev_q = 0.0)` is introduced. For the inputs, I is the in-phase signal, Q is the quadrature phase signal, prev_i and prev_q are for state saving purposes, where they store the previous values of I and Q. The function has three returns, fm_demod, prev_i, and prev_q. fm_demod is the array containing the demodulated data, prev_i / prev_q carry the state saving values to the next block.

experiment.cpp Take Home Exercise

In this section, the estimatePSD function provided in the python code is translated into c++. The function `std::vector<float> estimatePSD(std::vector<float> &samples, std::vector<float> &freq, int nfft, float Fs)` is added. The function returns the frequency vector and PSD estimate.

In addition to the estimatePSD function, an assisting function is also written: `std::vector<std::complex<float>> slicing(std::vector<std::complex<float>> &arr, int a, int b)`. This function can slice out the portion of data from a vector, input a and b are the starting and ending position. The function returns the vector that contains the slice of data we need.

Inside the main function, the PSD estimate plot is added into the logVector, with the name “demod_psd”.