Minhaj Shah / 400119266
Jack Wawrychuk / 400145293

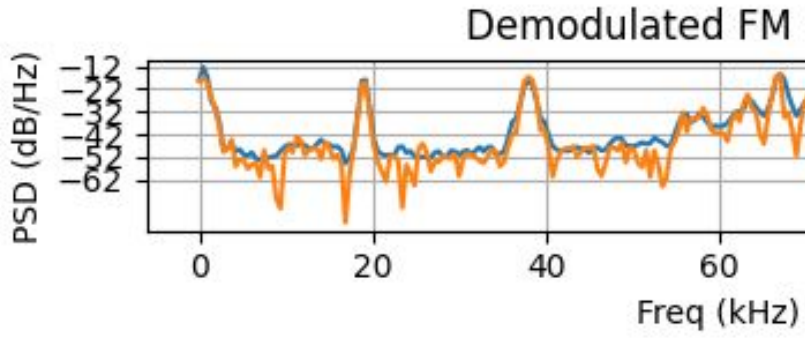
3DY4 Group 14 Lab 3

1. In this portion of the lab, we had to replace the methods from Scipy, *lfilter* and *firwin*, without our own methods from Lab 1. This task was straightforward as these functions were already made robust during the previous labs. The more interesting part was the implementation for the "computational-friendly" version of *fmDemodArctan*. We decided to complete this during this Lab. By implementing the equation from the resource provided, a successful computation was able to be executed to replace the inefficient Arctan function. The function is called *fmDemodFriendly*. By examining the plots and the wav files produced using this function, it was confirmed that this function was indeed working. Due this strategy producing an approximation and not an exact output to Arctan, there was some noticeable discrepancies in the audio, but overall, it sounded good. This function came with some good performance upgrades, as the execution time of the program noticeably decreased as well.

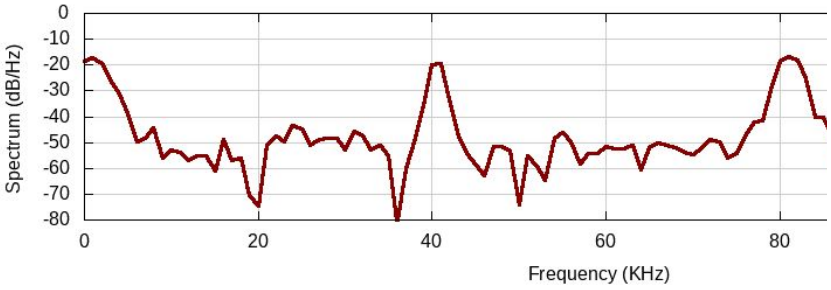
2. This task involved also implementing our own functions from Lab 1 to replace the Scipy methods. This task was also straightforward due to the same reasons mentioned above. The computational friendly versin of *fmDemodArctan* was also successfully implemented for *fmMonoBlock* as well. No changes needed to be made to the implementation due to the fact that the previous phase is still returned from the function so continuity for the next process could still be met. The program preforms well, run time wise, and the resulting wav audio sounds appropriate.

3. This final exercise is where things got really interesting. Implementing the *estimatePSD* function in C++ was a nice learning experience. Since libraries such as numpy were not available and all the handy Python tricks, it resulted in us understanding the inner workings of those functions by building them from scratch. We brought back the slice function we made from the old lab as well as it became handy extracting subvectors. The real challenge was to ensure the function was working accurately. By changing the GNU configuration in example.gnuplot, the plots were able to be generated which allowed us to visualize the power spectra. To more accurately compare the accuracy between the Python and C++ implementations of *estimatePSD*, a common input was needed to visualize the resulting outputs. A list in Python was created of size 5120 elements. This was initialized to some sample data copied from the output of *fm_demod* from a

particular iteration. By then copying the same 5120 elements, and initializing a vector of the same size with the same elements, both functions could be ran using the same input. Below are the figures from this test. The first one is the output of the PSD plot in orange. The second plot, is the result from GNU in the C++ implementation with the same input. It can accurately be seen that the implementation in C++ was done correctly as the results are very similar.



Python estimatePSD Plot for Idetnical Test Input



C++ estimatePSD Plot for Idetnical Test Input