Minhaj Shah / 400119266
Jack Wawrychuk / 400145293

3DY4 Group 14 Lab 1

1. One very interesting characteristic about square waves is the difference in behaviour in the time and frequency domain. In the time domain, the pattern of a square wave can visibly and intuitively be easily determined. There are constant repetition of the high and low portions for each duty cycle. What is very interesting, is examining the behaviour of square waves in the frequency domain. From the plots generated, it can be seen that the amplitude of the square wave drastically decreases as frequency increases. This behaviour connects directly from the definition of a Fourier Transform. Any periodic waveform can be constructed from a series of sinusoids. A square wave in particular is no different. Square waves are mathematically equivalent to the sum of a sine wave at the same fundamental frequency with an infinite series of odd-multiple frequency sine waves at diminishing amplitudes. This is exactly the reason why we see this behaviour in the frequency spectrum.

2. In this section a bandpass filter was implemented using the scipy function firwin. Firwin takes a series of inputs then computes the coefficients of a finite impulse response filter. In our implementation four parameters needed to be passed, the number of taps, the range of frequencies to be filtered, the type of filter (eg bandpass vs bandstop) and the window to be used. The biggest difference between using firwin for a lowpass filter and a bandpass filter is that instead of passing one frequency a range needs to be passed, such that every frequency outside that range will be filtered out. For our test case three signals with frequency of 1Hz, 10Hz and 30Hz were generated then added together to create a multi-tone signal. This signal was then put through our bandpass filter. The values corresponding to the band edges were calculated by dividing the desired frequency by the nyquist rate. For example if we only wanted the 10Hz signal to appear in the band, we could set a lower limit of 9Hz/50 = 0.18, and an upper limit of 11Hz/50 = 0.22, making the function call signal.firwin(N_taps, [0.18,0.22],pass_zero='bandpass', window=('hann')).

3. Implementing the functionality required for this portion of the Lab, definitely came with some challenges. After understanding the task, the first step was to create our own function which could preform convolution. Due to the nature of this computation, a pair of nested loops were utilized to iterate through indices to track resultant summations. An if statement with multiple conditions was used to ensure that no invalid indices were attempted. Verification became a challenge to ensure our function worked in all scenarios. The solution we came up with was to first import convolve from numpy, which would preform convolution as well. By comparing outputs at smaller inputs, the functionality of our function was deemed correct. During this process, the most optimal loop bounds were considered. Since our computation involves a finite sequence, we were able to ensure that the most inner loop runs approximately to the size of the impulse function which in comparison to the input, is significantly smaller. After this decision was made and the respective portion of the equation was altered, we had to test this function. Instead of jumping immediately to the configuration of blocks, a good way to test this was to incorporate the function into the filter_single_pass function. After confirming that the audio file sounded as expected, the block problem was then tackled. By creating a condition which breaks if the loop bound becomes equal to the length of the input, x, this would ensure that at each execution of our convolution function, only a block size amount of data would be computed. This would also handle edge cases where the block size is not the same at the very last iteration due to it not being a multiple of the total input size.