Multi-Rate Processing

# Introduction

This project has three objectives

1. To develop experience performing digital signal processing
2. To solidify understanding of multi-rate processing, interpolation and decimation, and poly-phase filters
3. To design an appropriate digital filter

# Filter Design

We were required to design a filter with a small ripple in the pass-band (< .001 dB), and a sharp peak side-lobe level (<-80dB). We were interpolating by a factor of 3/2, thus the more restrictive LPF required was one with a normalized frequency of 1/3 (0.33). To achieve this in a non-ideal scenario I gave myself some wiggle room on the pass (0.31) and stop (0.35) bands. I used the default FIR filter design method provided by the Matlab tool (Equiripple).

To determine the order I had the tool generate a minimal filter, which used 248 taps. For convenience in coding I then manually designed a filter with the next length divisible by 6 (thus it later divided evenly into the different polyphaser filters), 252. In setting the number of taps I wasn’t able to be as specific about the pass-band ripple or the stop-band peak, but the filter didn’t appear to change too much. Upon inspection the side-lobe peak was about 85 dB, and the pass-band ripple about 5x10-4 dB (.0005 dB), thus well within the requirements. Note that for some reason Matlab defines the order as the number of taps less one.

See the Figures 1-6 for details of use of the Matlab filter design tool:

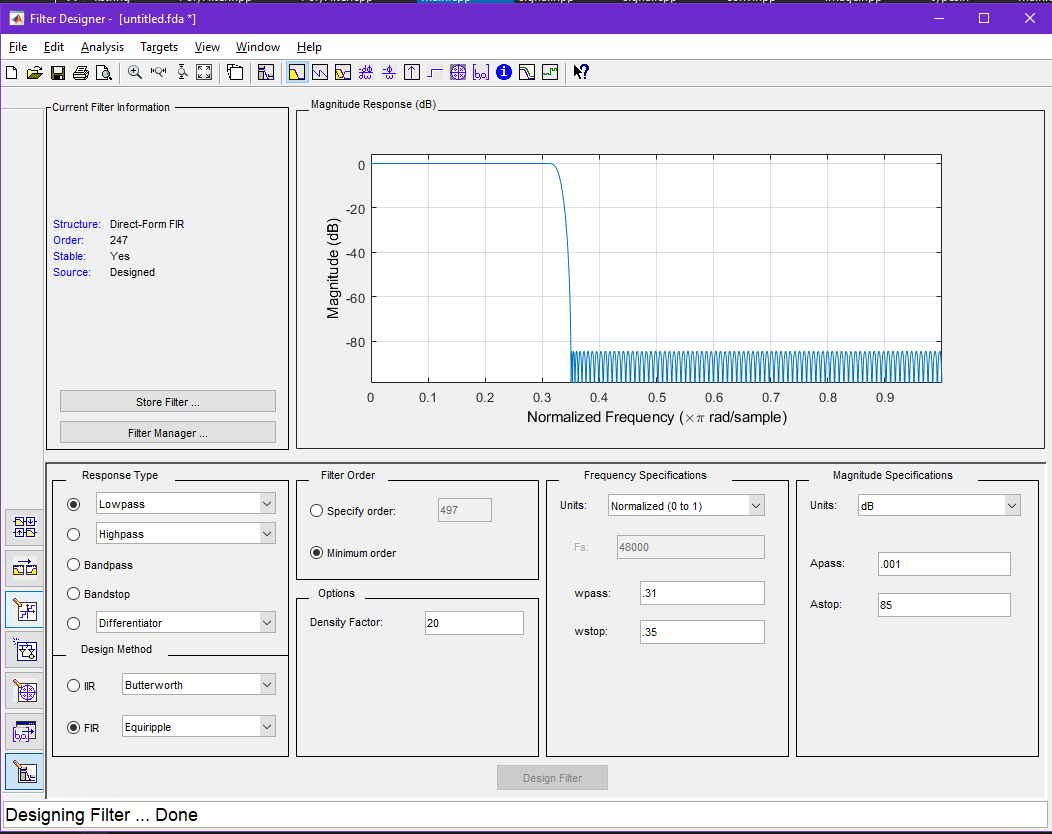


Figure - Minimal Order Filter Design Specifications

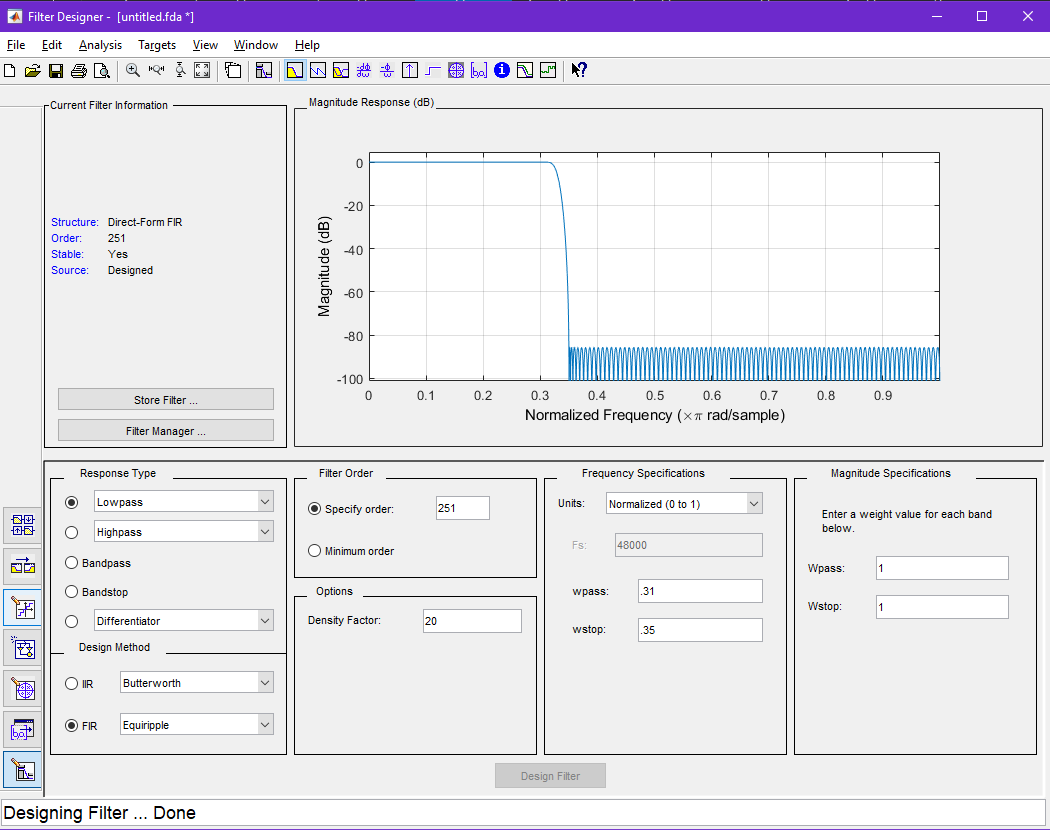


Figure - Modulus 6 Order Filter Design Specifications

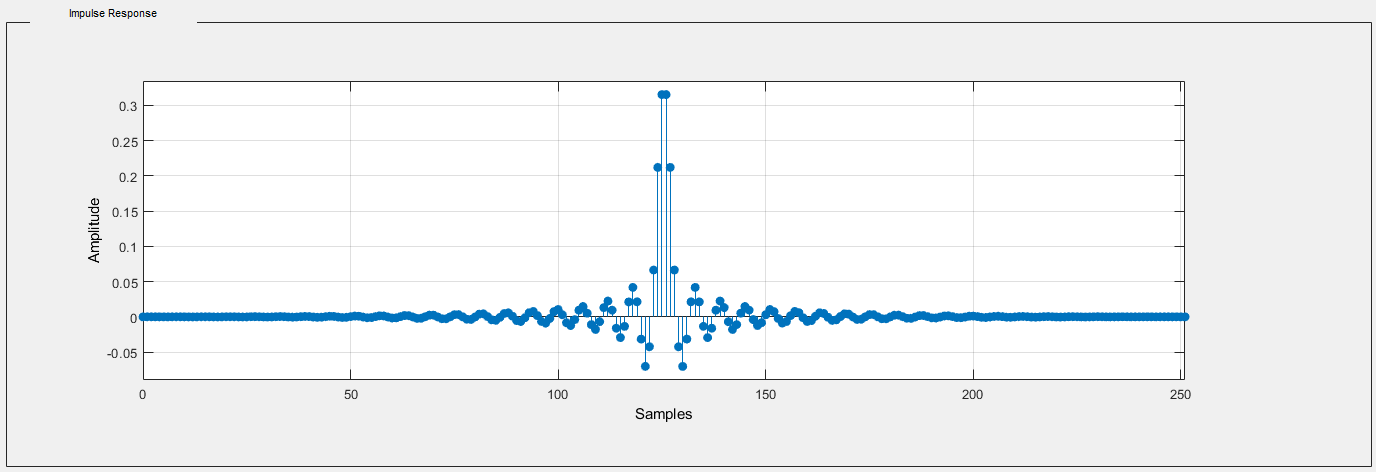


Figure - Filter Impulse Response

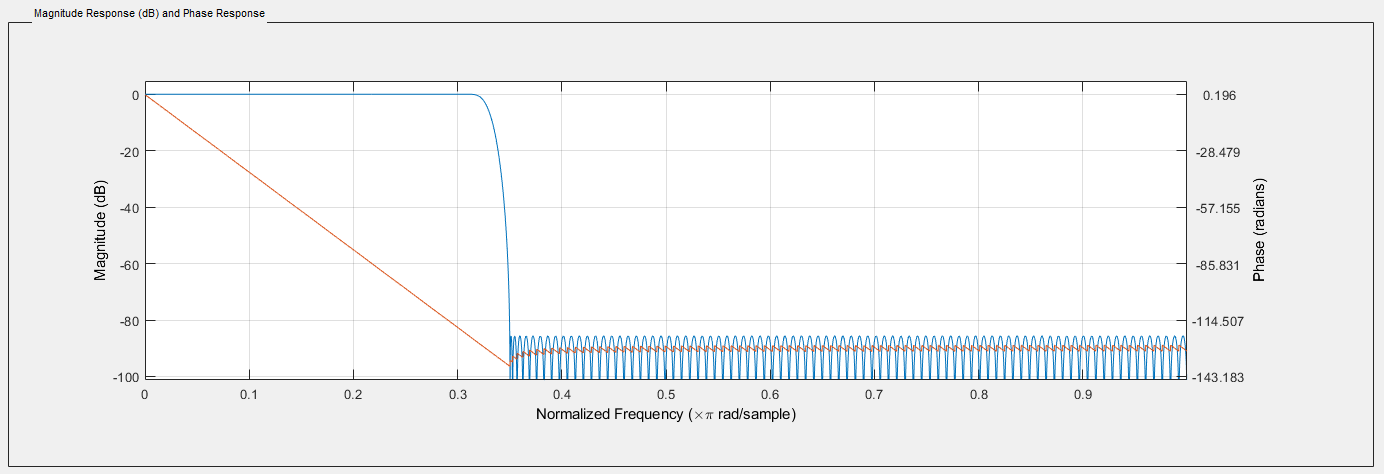


Figure - Filter Transfer Function

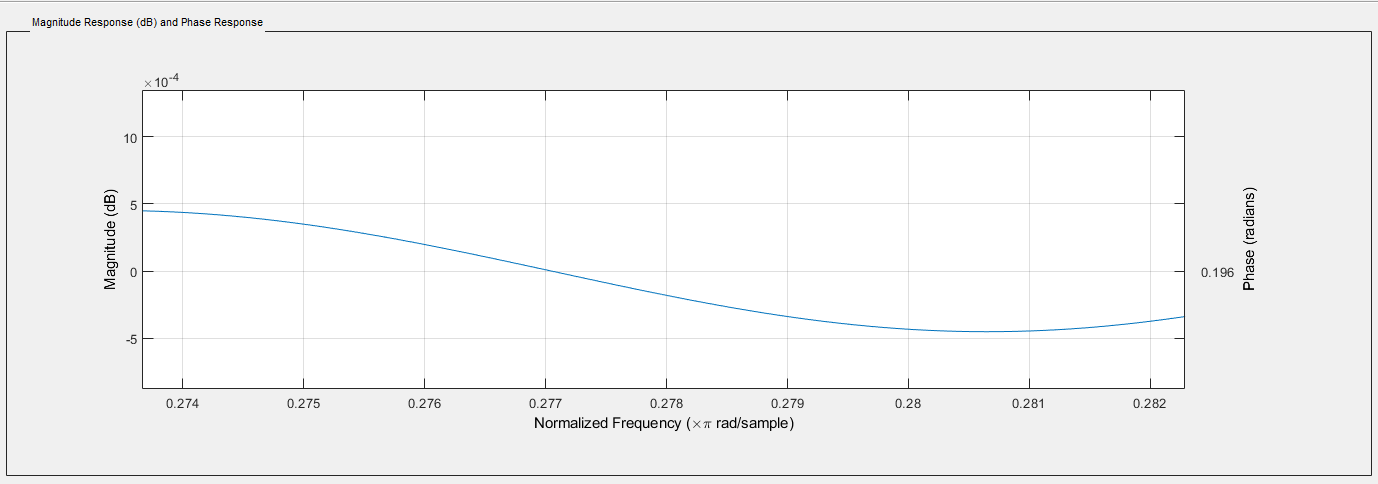


Figure - Filter Passband Ripple Zoomed In

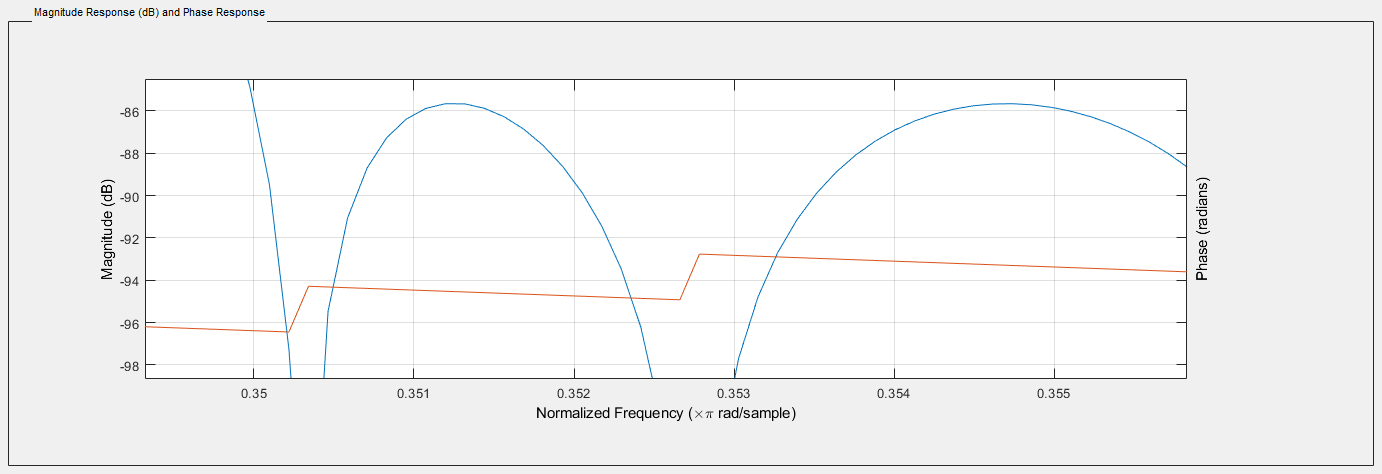


Figure - Filter Side-Lobe Peak Zoomed In

Finally to maintain consistent amplitude of the original signal as a result of the interpolation process the filter was scaled to have a gain of 3.

# Polyphase Filter Design

Much of the effort of designing the polyphase filter was derived in class lectures. In class we derived a 2/3 decimator, so it was a simple restructure to make a 3/2 interpolator.

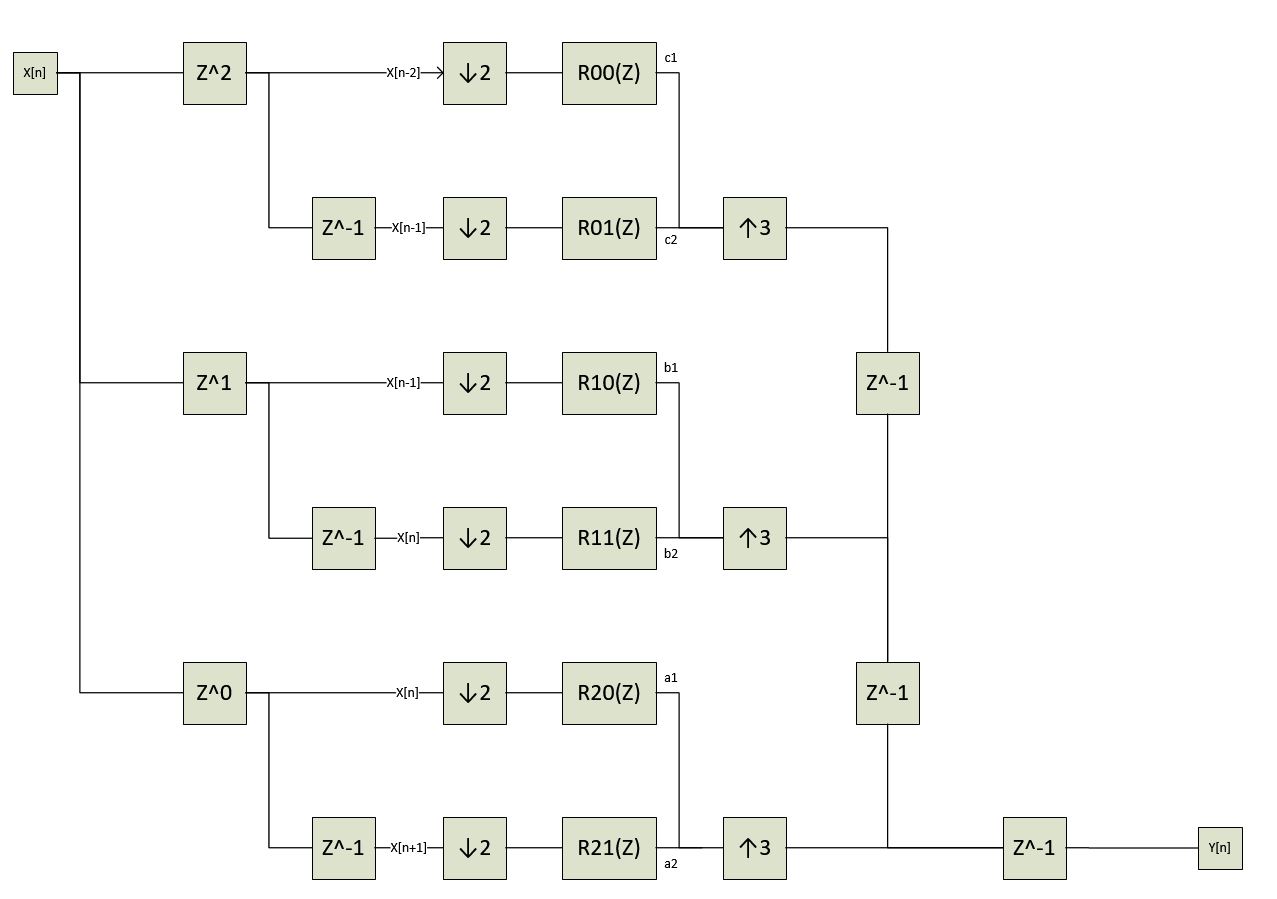


Figure - Polyphase flow (Image made by peer Taylor Peterson, without whom this report would have a scanned hand-drawing instead due to lack of experience with Visio and lack of time)

In class we derived the number of MPUs to be and the number of APUs . N is the side of our filter, 252. Thus we conclude this system would require 127 MPUs and 125 APUs. Of note (from the assignment description), a 3/2 interpolator given an input of 11.025 kHz would output a signal of 16.538 kHz.

In the code you will see me reference Rud. The indices u and d correspond with the graph above. Rud[n] is defined such that the coefficients of h[n] (as defined in the previous section) are allocated as follows:

**{ R00[0], R01[0], R10[0], R11[0], R20[0], R21[0], R00[1], R01[1], R10[1], R11[1], R20[1], R21[1], R00[2], … }**

# Conclusion

Images can be filtered using 2D convolution. There are many types of filters with a wide variety of applications. Convolution is at the center of digital signal and image processing.

Appendix: C++ and Matlab Code

# signal.hpp

# signal.cpp

# main.cpp

# matlab code