Aarhus Institue of Technology

Morten Hgsberg: 201704542
Sune Dyrbye: 201205948
Yehven Parolia: 20112870

Date: November 30, 2021

Introduction

From 2009 to 2013 the Kelper Space Telescope looked almost continualy at one point in space. Its goal was to measure the magnitude of a large number of stars with a high frequency for 4 years in order to examine the exoplanets (planets orbiting stars other than the sun) similar to Earth.

This report examines the time series of data taken by the Kepler Space Telescope in order to exaimne the effects of Finite Impulse Responce (FIR) and Infinite Imuplse Responce (IIR) filters. The data is choosen to be a different type of signal from what we usually see, and it is therefore not expected that any results of scientific relevance is acheived.

In the following we will examine the design of a FIR filter, and how FIR and IIR filters affects the time series.

Time series and spectrum

The data series we examine is shown in Figure 1. The time axis is in bariocentric julian days (BJD) with an offset of 2454833 days (bringing 0 to 1/1/2009). The y-axis is photometric flux, meassured in $\frac{e^-}{s}$. It can be seen that the data undergoes a general drift, increasing in magnitude as time passes.

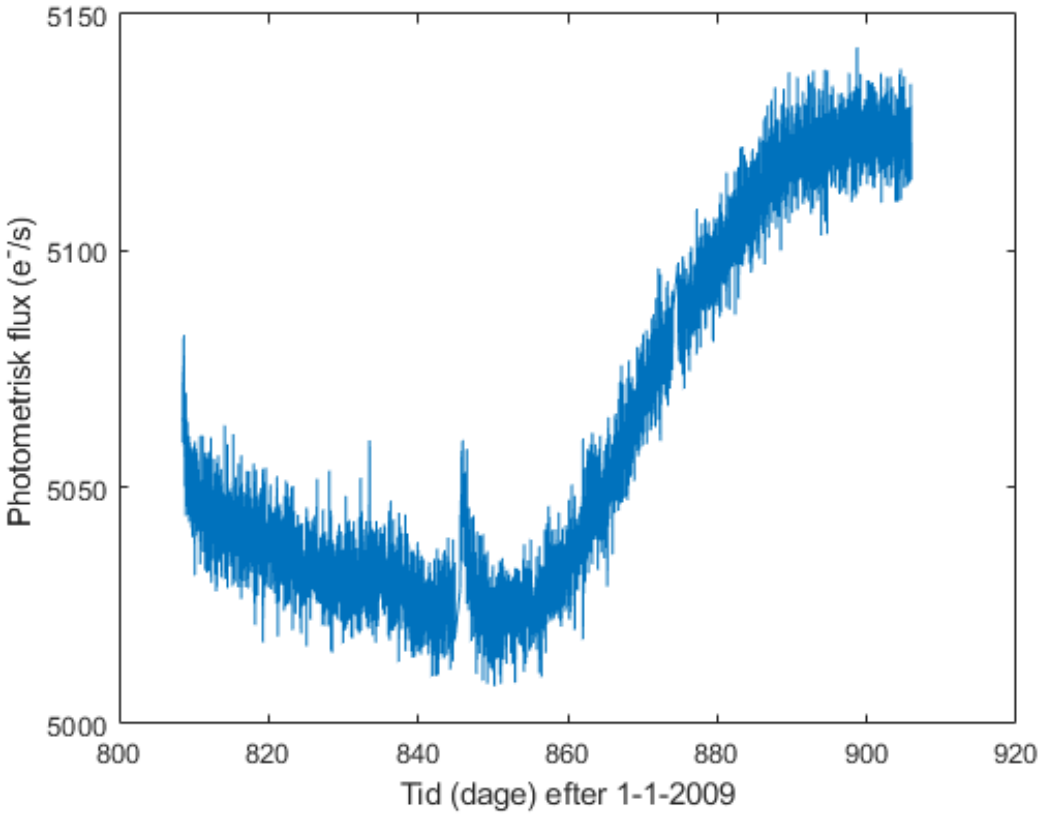

Figure 1: Kepler time series

The discrete fourier transform (DFT) of the timeseries is displayed in Figure 2. Note that the

frequency is in $\frac{1}{day}$ rather than the usual Hz. This is due to the timeseries being sampled in fractions of days (arround once every half hour (or 0.00056 Hz)).
It can be seen in the figure that there is a rather large component with a low frequency. This is probably caused by the general drift in the data.

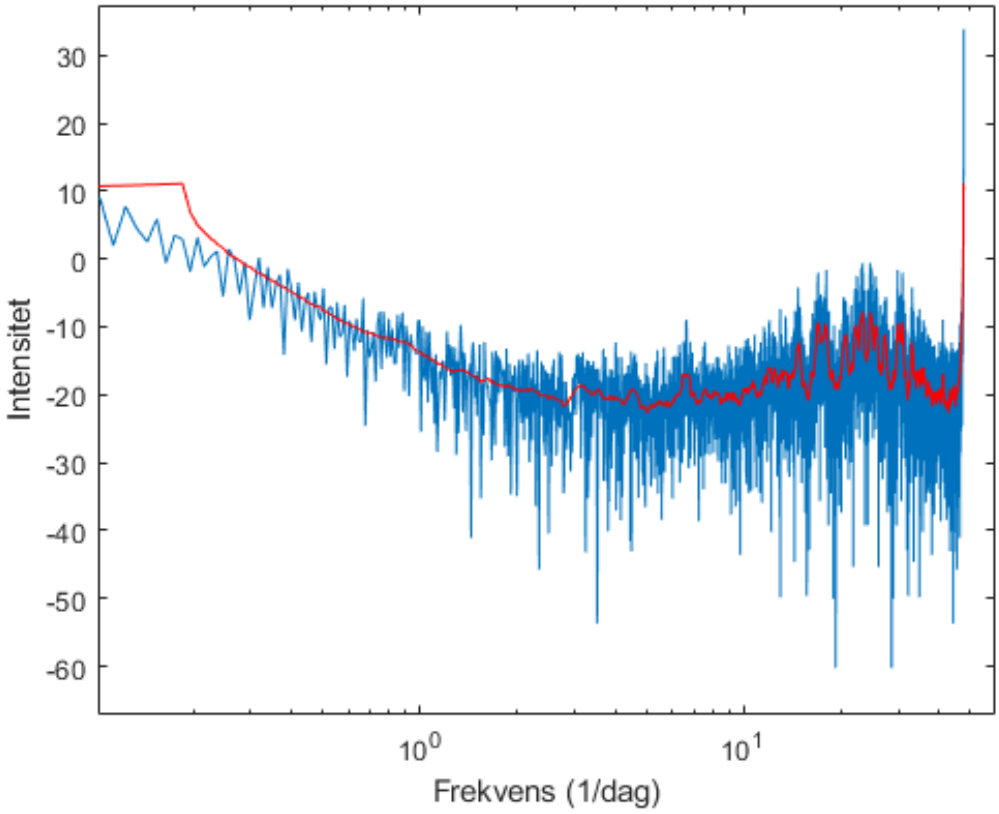

Figure 2: Spectrum of the simeseries and the same spectrum smoothed vit a binsize of 35

Figure 2 also shows a smoothed spectrum, with a binsize of 35. It is clear that there is some irregularity in the low frequencies, probably a result of the smoothing including parts beyond the spectrum.

The smoothed spectrum removes a large amount of (presumably) noise from the spectrum and highligts a number of peaks. As the timeseries is from a planet survey, it is probable that one or more of these peaks are caused by an eclipse.

Filtre

This project utilizes two types of filter a FIR (finite inpulse responds) filter and an IIR (infinite impulse responds) filter. The FIR filter is designed and implemented in matlab using the window method.

FIR Filter Design

First step in designing a FIR filter is to design an ideal IIR filter before trucating it with by multiplying the IIR filter with a finite length window function.

By using our spectral analysis from the earlier sections, we qualitatively decided to make the cutoff frequence $f_c = 10$. The sample frequency f_s is given from our dataset, $f_s = 47.7774$.

Lastly, the filters made order $M = 250$ thus using 250 filter coefficients.

Resolution

Next step in designing our filter, we determine the frequency resolution which provides specifications for the FIR transfer function.

$$f_{res} = \frac{f_s}{M}$$

Transfer function

Using f_{res} and f_c, we can determine which frequency bin corresponds to frequencies below f_c. This must be in done in integer values i.e. rounded to closes integer value.

$$f_{bin} = \left\lfloor \frac{f_c}{f_{res}} \right\rfloor = 52$$

In this case the bin number corresponds to f_c is 52, which we design our lowpass filter around see Figure 3.

These specification help determine which frequencies should be passed and which should be removed. In this case we remove everything above frequency 10.

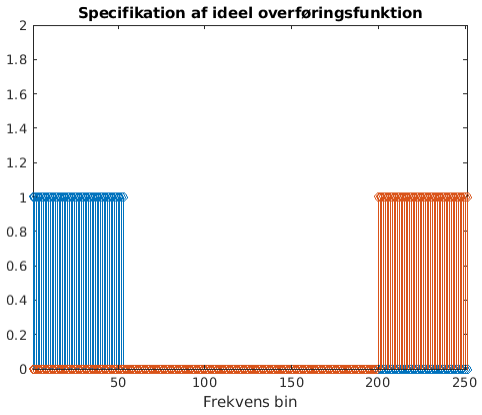

Figure 3: Specification for our transfor function

In matlab we can now make our transfer function h using the follwing code. Which when combined with, in our case a hanning window function serves to become our filter which we can run our data set through.

matlabStuff/filter.m

```
1 H_left  = [1 ones(1,freq_bin_round) zeros(1,(M/2)-freq_bin_round)];
2 H_right = fliplr(H_left(2:end));
3 h = fftshift(real(ifft(H)));
4 w_hanning = hanning(M+1)';
5 h_win = h.*w_hanning;
6
7 H_without_win = fft(h,f_sample_round);
8 H_with_win = fft(h_win,f_sample_round);
```

Figure 4: matlab code for making a transfer function

Now that we have our transfer function, we can apply it to our data to see if we can reduce the potential noice in data. we can plot its coefficients together with the window function, along with the resultat transfer function to see if it does indeed "cover" the peak around $10Hz$.

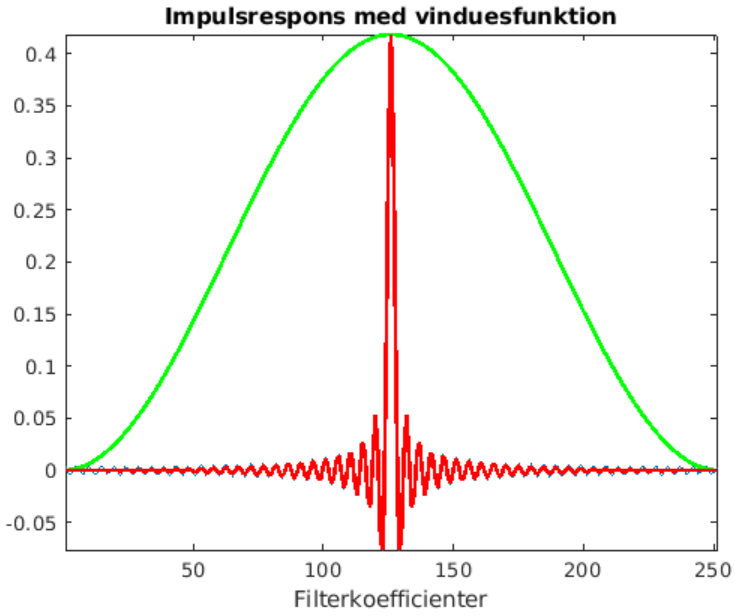

Figure 5: Impulseresponds overlayed with the window function, showing its koefficent values as functions of index numbers

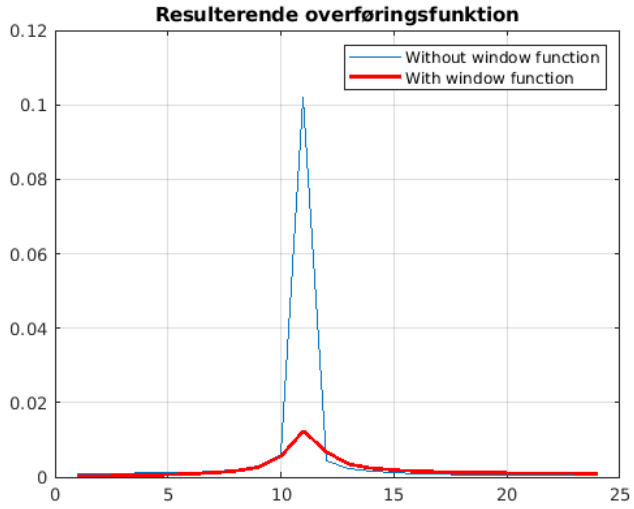

Figure 6: Resulting transfer function overlayed with and without the window function

Applying the FIR filter

With all of the preperation work done, we can now apply our filter to our dataset to see the results.

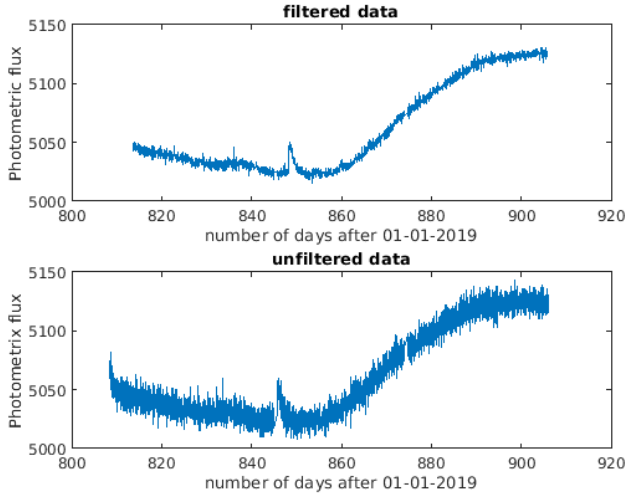

Figure 7: Side by side comparison of the data before and after the filter has been applied

Its clear fron this comparison that we have manage to remove a some of the signal noice, but since this data set does not show anything usefull (the spike is a kalibration issue), we wont be able to tell anything from this filtering.

IIR Filter

For the IIR filter the "butter" function has been used too design the transfer function and thus produce the filtering coefficients.

Once this has been done everything plays out much like the FIR filter, we can plot our transfer function and the hanning window function to see the coefficients and their values as functions of indexing.

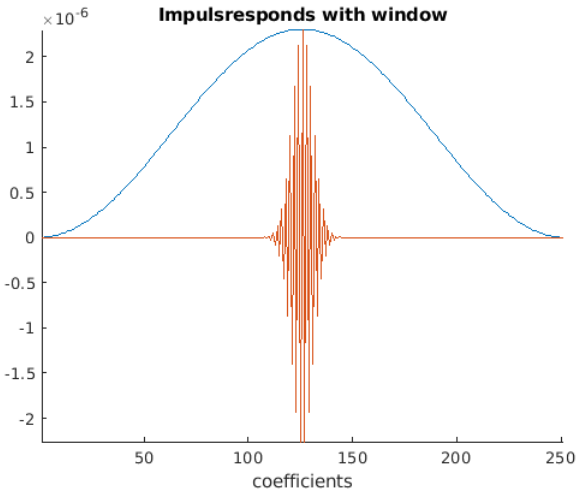

Figure 8: Impulse responds with window function for the IIR filter

Here we can see that the new Impulseresponds does not tapper at all compared to the FIR filter which did tapper off toward the center.

Figure 9: Resulting transfer function overlayed with and without the window function

unlike the FIR filter which produce a simple peak, and reduced peak for our transfer function the IIR butterfilter, has produced a resulting transfer function that seem to cover the entire spectre of frequencies.

Note also that when the window function is applied to it, it is substantially reduced, over the entire spectre as well.

Finally lets see what the result off applying this filter to our original data.

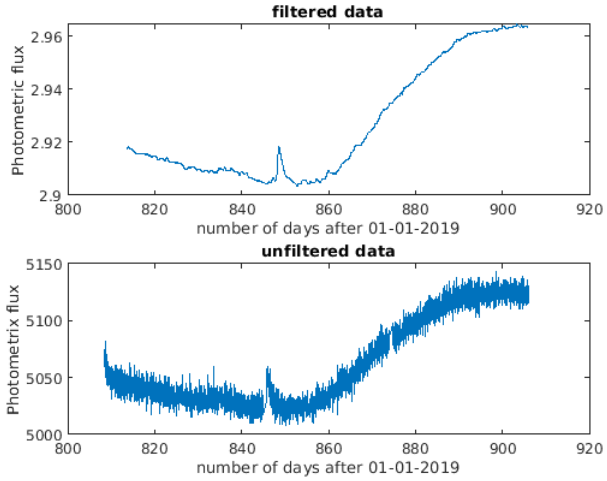

Figure 10: Data comparison for original data and IIR filtered data

As we can see in Figure 10 this has removed any hint of noice, but if there were significant data in this dataset it might have remove that as well. IIR could therefore at least in this case be too strict of a filter to use and the FIR filter might therefore be prefered.

Experimenting with different cutoff frequencies and orders

If we try with a different cutoff frequency, compared to what was seen in Figure 7 we can see that lowering the frequency cutoff for our lowpass filter means that we are removing more potential noice, but also have a higher chance of removing vital data points.

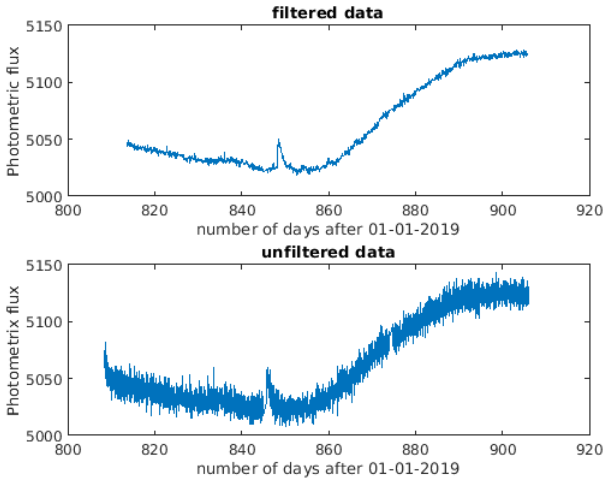

Figure 11: Data comparison for original data and $f_c = 5$ instead of the origianl $f_c = 10$

Its not shown here but not that it will in general lead to a wider Impulseresponds and vice versa for higher frequence cutoffs.

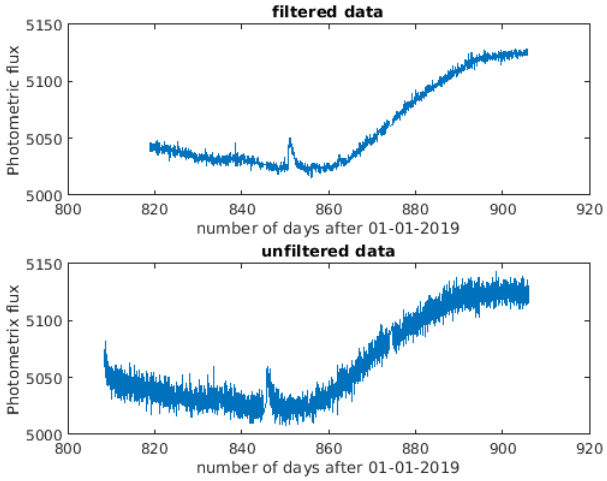

Figure 12: Data comparison for original data and higher order filtered data

Here we dont see much difference, maybe because the filter order is alread very high, but we can see that the data is a little more refined that for Figure 7. Obviously if we had a lower order filter the data would be less refined once filtered.

Group Delay

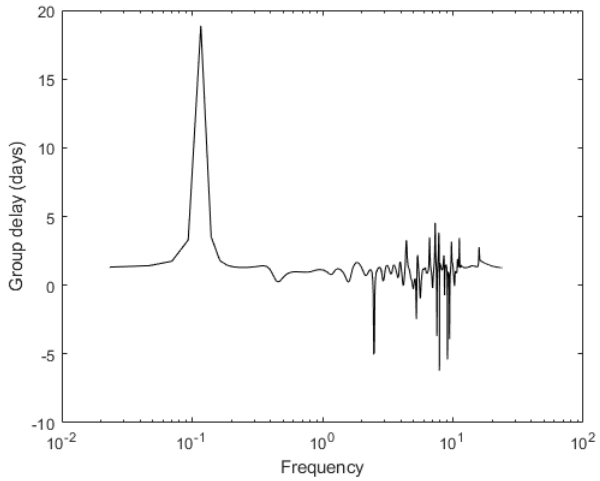

Figure 13: Data comparison for original data and higher order filtered data

We calculate the group delay for the original IIR filter using matlabs *grpdelay* function. This is shown in Figure 13. It shows the delay through the IIR filter. Note the massive delay around 0,1. The negative delay is interpreted as the filter anticipating the value of the signal.

Assignment 3.15

Given a signal amplitude vector x = [3 -1 0.6 -0.8 -2 0 0.9]

We first calculate the DFT values by matlab function fft(x).

The results if fft(x) can been seen in figure 14 blue dots. Note that only the real values (Amplitudes) are shown for more information see the Matlab file opgave_315.m.

Now we calculate the twiddle-factor for an N=8 point DFT. $W_8 = e^(-j*(2*pi*m*n/8)$ where n and m are the matrix notations corresponding to the column, row notation.

We now calculate the needed 8 twiddle-factors corresponding to the 8 different position on the unit circle.
${W_8}^0 = 1;$
${W_8}^1 = exp((-j*(2*pi/8)));$
${W_8}^2 = exp((-j*(2*2*pi/8)));$
${W_8}^3 = exp((-j*(2*3*pi/8)));$
${W_8}^4 = exp((-j*(2*4*pi/8)));$
${W_8}^5 = exp((-j*(2*5*pi/8)));$
${W_8}^6 = exp((-j*(2*6*pi/8)));$
${W_8}^7 = exp((-j*(2*7*pi/8)));$

Remember that ${W_8}^n = {W_8}^{n-8}$ for $n >= 8$ and so on. The full matrix W can bee seen in the matlab file.

We now calculate the product $x*W$ and get the DFT values also seen in the figure 14 red dots.

We observe that the two different methods of calculating DFT values are not the same, this is expected as the vector is a size of 7 elements and the DFT calculated manually is of N=8.

Assignment P6.4

Given a filter described by the following equation:

a) $y(n) = x(n) - y(n-2)$

Transferring into Z domain:

$Y(z) = X(z) - Y(z)*z^(-2) -> Y(z)*(1+z^(-2)) = X(z)$

Transfer function calculated as H(z) = Y(z)/X(z):

$H(z) = Y(z)/X(z) = 1/(1+z^(-2)) -> z^2/(z^2+1)$

The following transfer functions can be expressed in polar as well as in rectangular coordinates:

For polar substitute z for $e^(jw)$ we get:

$H(w) = 1/(1+e^(-j*w*2))$

For rectangular expand e^jw to cos(w)+jsin(w):

$1/(1+sin(2w)+jcos(2w))$

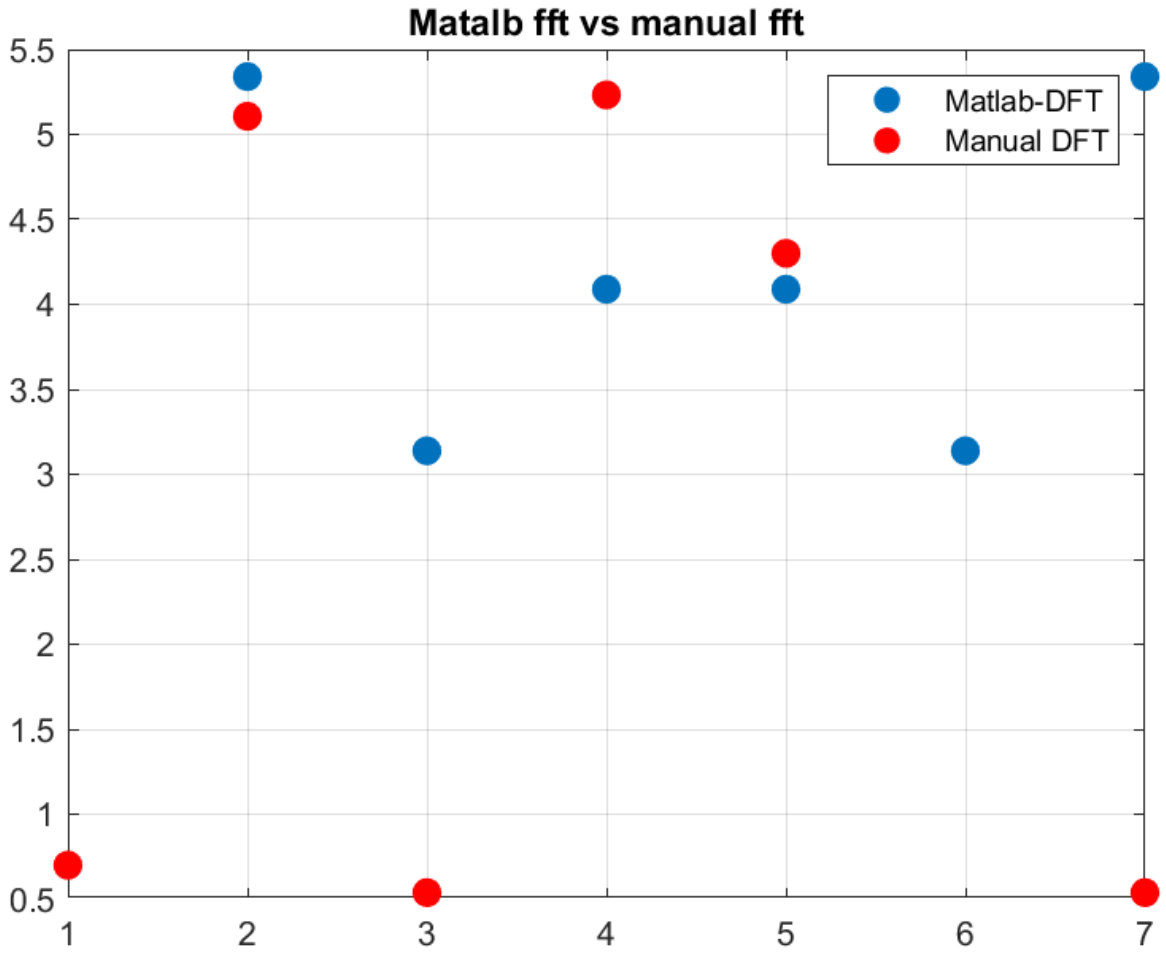

Figure 14: Comparison of the matlab DFT (blue) and manual DFT values (red) real values

Given a filter described by the following equation:

b) $y(n) = x(n) + 3x(n-1) + 2x(n-2) - y(n-3)$

Transferring into Z domain:

$Y(z) = X(z) + X(z)3z^(-1) + X(z)2z^(-2) - Y(z)z^(-3)$

$Y(z)(1 + z^(-3)) = X(z)(1 + 3z^(-1) + 2z^(-2))$

Transfer function calculated:

$H(z) = (1 + 3z^(-1) + 2z^(-2))/(1 + z^(-3))$

For polar substitute we get:

$H(w) = (1 + 3e^(-jw) + 2e^(-2jw))/(1 + e^(-3jw))$

For rectangular substitution:

$H(w) = (2 + cos(w) + jsin(w))/(2 * cos(w) - 1)$

Given a filter described by the following equation:

c) $y(n) = x(n) + x(n-1) + x(n-3) + x(n-4) + y(n-2)$

Transferring into Z domain:

$Y(z) = X(z) + X(z)z^(-1) + X(z)z^(-3) + X(z)z^(-4) - Y(z)z^(-2)$

$Y(z)(1 + z^(-2)) = X(z)(1 + z^(-1) + z^(-3) + z^(-4))$

Transfer function calculated:

$H(z) = (1 + z^(-1) + z^(-3) + z^(-4))/(1 + z^(-2))$

For polar substitute we get:

$H(w) = (1 + e^(-jw) + e^(-3jw) + e^(-4jw))/(1 + e^(-2jw))$

For rectangular substitution:

$H(w) = (1+cos(w)-jsin(w)+cos(3w)-jsin(3w)+cos(4w)-jsin(4w))/(1+cos(2w)-jsin(2w))$