**FIR filter design**

**Filter design by windowing method**

1. **LPF design**

For this exercise, I created a function called LPF which takes as input parameters Fc, Fs, N, window type and implements equation 6.11

1. **Effect of longer filter length**

When it comes to the magnitude response, near the cutoff frequency the transition region is much more abrupt as the number of filter taps increases. However, this comes at the cost of a larger pass-band ripple as N increases. Moreover, with the increase of N we can observe that the side-lobes get narrower and they decrease in magnitude much faster than in the first case. When N increases, the number of ripples increases.

1. **Characteristics of windowing functions**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Filter type | Peak side-lobe amplitude [dB] | Approximate main-lobe width [π] | Peak approximation error δ - linear | Peak approximation error δ [dB] |
| Rectangular | -13.26 | 0.017 | 0.099 | -20.087 |
| Hamming | -42.31 | 0.025 | 0.0023 | -52.765 |
| Hanning | -31.46 | 0.027 | 0.006 | -44.436 |
| Bartlett | -26.43 | 0.025 |  |  |
| Tukey | -15.12 | 0.022 | 0.079 | -22.047 |

1. **Effect of Windowing on Low-pass Filter Design**

In order to measure the peak approximation error, I represented the magnitude response of the filter on a linear scale and in the passband I measured the maximum ripple that occurs. Then I calculated the corresponding value in dB. It can be observed that the Hamming window has the lowest peak approximation error.

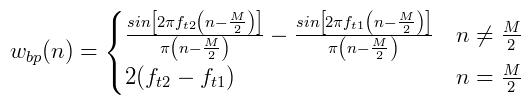
1. **HPF design**

The HPF has all zeroes, except for one, which is in 0, placed on the unit circle, whereas the LPF has from the previous exercise has only two zeroes on the unit circle, four within the unit circle, one of which is in the origin. The HPF has also a zero at plus infinity.

Relationship between the poles and zeroes of a LPF and HPF: multiplying with ejw in frequency domain is equivalent to a rotation in the z plane with π, which determines the mirroring of the unit circle, together with the poles and zeroes, with respect to the y axis.

1. **Band-pass Filter Design**

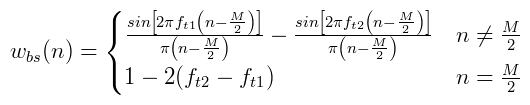
A band-pass filter is obtained by cascading a LPF with fc1 with a HPF with fc2, with fc2<ffc1. The impulse response of the BPF can be obtained by convolving the impulse response of the LPF with the HPF. However, I implemented the transfer function given below:



For both band pass and band stop, the filter order needs to be even (an odd filter length). After computing the impulse response of the BPF, I multiplied it with the Hamming window and then I normalized the magnitude response such that it has the magnitude of approximately 0 dB in the pass-band.

1. **Band-stop Filter Design**

A band-stop filter is obtained by cascading a HPF with fc1 with a LPF with fc2, with fc2>ffc1. The impulse response of the BSF can be obtained by adding the impulse response of the LPF with the HPF. However, I implemented the transfer function given below:



For both band pass and band stop, the filter order needs to be even (an odd filter length). After computing the impulse response of the BSF, I multiplied it with the Hamming window and then I normalized the magnitude response such that it has the magnitude of approximately 0 dB in the pass-band.

1. **Kaiser Window**

For this exercise, I created a function as described in the script that implements equations 6.20-6.22 and applies the computed parameters to the formula described in equation 6.18, obtaining in this way the coefficients of the impulse response of the Kaiser window.

**Filter Design by Optimization**

1. **Low-pass Filter (LPF) Design**

For this exercise I chose the maximum value for K, namely K=16 in order to have a more densely populated frequency grid. In order to generate the frequency grid w and the desired amplitude response, I created a function:

function [pulsation,amplitude]=frequency\_grid(f\_pass,f\_stop,f\_sampling,N,K)

which calculates the total number of points, which is M\*K, where M=N-1, creates a vector called w which contains M\*K equally distributed points in the interval 0:π , then computes wp and ws and determines afterwards the number of points that fall in the passband and stopband, ignoring thus the points that fall in the transition band. With this information, the function creates 2 vectors called passband and stopband which contain a number of equally distributed points equal with the number of points in the passband and stopband respectively, also associating this values with the desired amplitude response of the desired filter type.

The output of this function become the input of the design\_filter function as described in the script, where in the first step I compute Q and L according to the filter type, then I determine Wq and dq, I compute the U matrix and afterwards, having determined all the unknowns, I compute p. Then, according to the filter type, I compute the bm vector and based on this vector I determine the impulse response h for the first half of the filter. Yet again, taking into consideration the filter type, which determines the fact that the impulse response is either symmetric or antisymmetric, I create the entire impulse response by merging the already obtained h with its flipped version 9(using the fliplr command in Matlab), the latter being multimplied with a 1 or -1 for a symmetric or antisymmetric impoulse response. Then I normalized the magnitude response such that it has the magnitude of approximately 0 dB in the pass-band and plotted the results using fvtool.

1. **High-pass Filter (HPF) Design**

Analogous to point 1.

1. **Differentiator Design**

Analogous to point 1, with the mention that I ignored the first element in the weight function because of the < sign in the desired amplitude response, which implies ignoring also the first element for D and w because all vectors must be of the same length

**Filter Design by Frequency Sampling**

For this exercise, I created a function called amplitude\_response which takes Fs, Fp, Fstop and N as input parameters and outputs A, the desired amplitude response of the filter. First, it generates the k vector, the number of points (samples) corresponding to the upper half of the unit circle, based on the parity of N and then it starts creating the amplitude response, taking into consideration also if one of the wk fall within the transition band: if only one value falls within the transition band, the corresponding amplitude response for that particular frequency is 0.4, and if two frequencies fall within the transition band, the corresponding amplitude response values are 0.59 and 0.11.

The amplitude response vector thus determined is one of the inputs of the design\_filter function, which in the first step determines the magnitude response for the upper half of the unit circle by multiplying the amplitude response vector with exp(-j\*π\*k\*(m-1)/M). Additionally, for type 3 and type 4 filters, one additional multiplication with 1j is done. Afterwards, the magnitude response for the lower half of the unit circle is determined by complex conjugating the values corresponding to the vector determined previously for the upper half of the unit circle.

In the end, the impulse response of the desired filter is obtained by taking the real part of the IFFT applied to the vector of magnitude response for the whole unit circle.

Comparing the amplitude response of the filters obtained through the frequency sampling method with the amplitude response of the filters obtained through the optimization method, we observe that in the first case the amplitude response at the desired frequencies is guaranteed to be the same with the specifications, whereas for the latter case this does not hold true. The impulse response has the same shape, but the amplitude of the spikes varies between the two cases.