Chapter 8 The FIR Filters

In this chapter

Ideal Filters

FIR Filter design

Windowing

Introduction

The Digital Filters are classified as, IIR or Infinite Impulse Response filters and FIR or Finite Impulse Response filters. We have discussed the IIR filters in the previous chapter and designed a few, including; the high-pass, low-pass, band-pass and band-stop filters. In all, the computation time was fast, as it did not take much to calculate few poles and few zeros, but the phase response was generally poor. The initial transient response in IIR filters lingers on for a long time. The filters require values from the past output and some parts of the initial values are always present in the current output, in theory at least. Though, in practice, the fraction grows less and less and eventually reaches 0, but it might take a very long time before the floating-point arithmetic precision gives way to the accuracy of the calculated numbers. The linear phase response makes the FIR filters suitable candidates where the phase information has an important role to play.

We now discuss a different method of obtaining a filter Transfer Function that would simply rely upon the values from the past input but not the past outputs, as such; there is a uniform phase delay with no distortion. The linear phase is a definite plus, where phase delay is critical and this is the Fourier Transform method of designing digital filters. The figures and the tables in this chapter are being created with the help of Scilab and Matlab scripts and listings of the instructions are provided in Appendix A and B. It is recommended that the reader should exercise the instructions in the scripts and gain confidence in filter design method by seeing the response interactively. The starting point for the FIR filter design is the ideal frequency response.

Ideal Filters

The Figure 8.1 is a repeat of the ideal filter response that we have seen before. Notice the rectangular shape and the sharp cutoff point that marks the transition from the pass-band to the stop-band, there is practically no transition-band and the gain is either 0 or 1. Such an ideal filter, if we could achieve somehow, would solve all our filtering problems, but in reality (as you will see later), we cannot achieve such perfection. We can only try to get closer. In FIR filters, we seek the function that closely matches the ideal response.

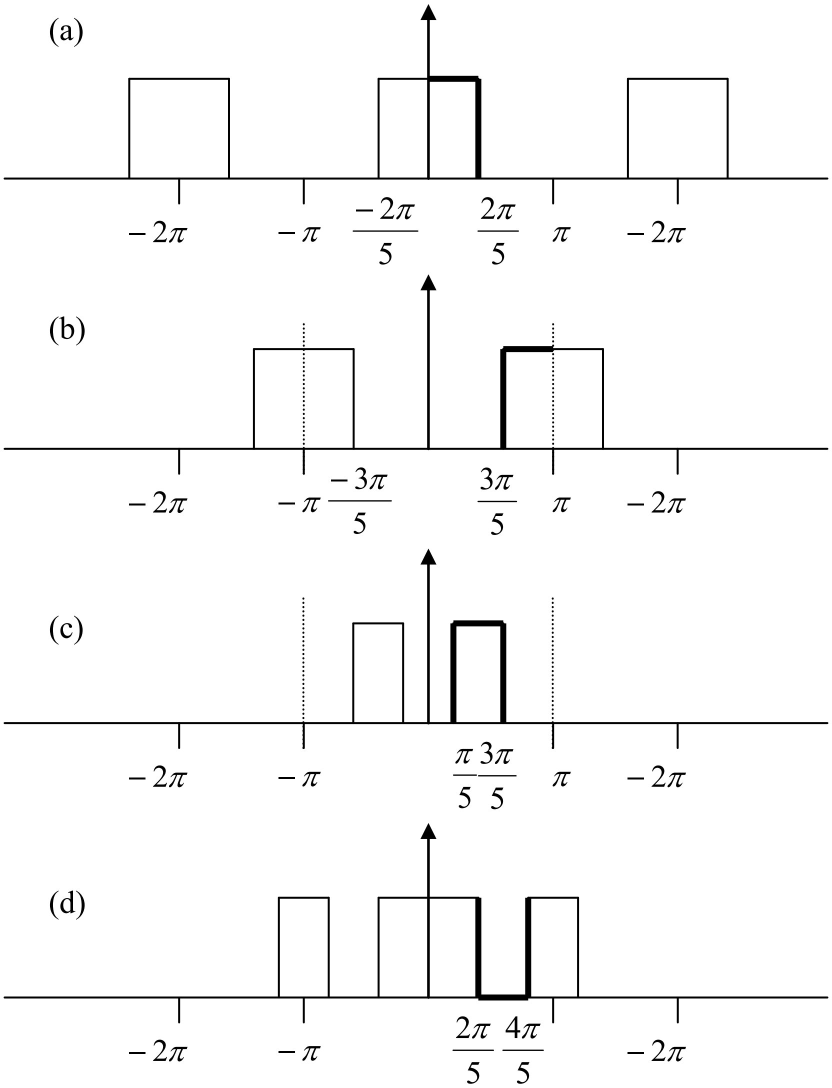
Let’s work backward and think; if there was a frequency response, close to the ideal response, then there must have been a time based impulse response that generated the ideal frequency response. The Fourier Transform of the ideal response would have impulse response as its coefficients, and this is basis of Fourier Transform method of designing digital filters. Once we know the impulse response, it is easier to formulate a filter by performing convolution of the input samples with the impulse response, just as we did in the previous section of the IIR filter design.

The objective is to find the coefficients of the Fourier series that generated the ideal response (or close to it) and that Fourier series is the desired filter Transfer Function.

Let's chose a low pass filter as shown in the Figure 8.2. (As usual, we will be denoting the discrete sampling frequency with the symbol Ω.) Ignoring for a moment the negative frequencies, such an ideal frequency response has a well-defined period $-\pi$ to $+\pi$ with the pass band region from $-\Omega$ to $+\Omega$ (where Ω is the normalized sampling frequency of radians per sample), center on 0. This is a perfect low pass filter with the cutoff sampling frequency of $+\Omega$. The values within the pass band region are 1 and outside this region, they are 0, thus forming a function with the values as shown in the following series,

0,0,0,..0,0,0,1,1,1…1,1,1,0,0,0..0,0,0

What function has such a rectangular shape? To answer this question, let's revisit the Fourier Transform and produce the coefficients that form the output.



π

****** Insert Figure 8.1 here ******

Figure 8.1. The ideal frequency response; a) low pass filter, b) high pass filter, c) band pass filter, d) band stop filter

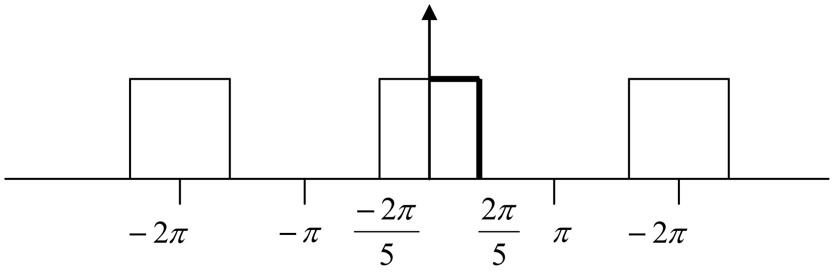
The Fourier series representation of the ideal frequency response

A discontinuous function such as the one in Equation 8.2 may be expressed in terms of its Fourier series. The function itself is a periodic function and according to the definition, a periodic function $f(x)$ may be represented as an infinite series of sin and cos, whose amplitudes (or coefficients) c_n may be computed from the function itself, as shown in Equation 8.1 and 8.2. (See chapter one for a discussion on extracting the Fourier Coefficients.)

$$f(x) = \sum_{n=-\infty}^{\infty} c_n e^{j\omega x} \tag{8.1}$$

The coefficients,

$$c_n = \frac{1}{2\pi}\int_{0}^{\pi} f(t) e^{-j\omega x} dx \tag{8.2}$$



****** Insert Figure 8.2 here ******

Figure 8.2. The ideal low pass filter with the cutoff frequency $-\Omega = \frac{-2\pi}{5}$ to $+\Omega = \frac{2\pi}{5}$

In digital filters, we have a reverse situation, the frequency response $H(\Omega)$ is a periodic function, and the coefficients are time based. The Discrete Time Fourier Transform of the ideal filter Transfer Function of the Figure 8.2 is given in Equation 8.3.

$$H(\Omega) = \sum_{n=-\infty}^{\infty} h_n e^{-j\Omega n} \qquad 8.3$$

The coefficients h_n are obtained using the inverse Discrete Time Fourier Transform,

$$h(n) = \frac{1}{\Omega_{-c} - \Omega_c} \int_{\Omega_{-c}}^{\Omega_c} H(\Omega) e^{j\Omega n} d\Omega \qquad 8.4$$

Notice; the periodic response function *h(n)* of Equation 8.4 has a period of $-\pi$ to $+\pi$, but the pass band region from $\Omega_{-c} = -2\pi/5$ to $\Omega_c = 2\pi/5$ (example form Figure 8.1) is the only nonzero part, thus, the integral is performed over the pass band region only. In our case $\Omega_c = -\Omega_c$, substituting the value of the Transfer Function (between Ω_{-c} to Ω_c) and integrating to obtain the expression for the impulse response *h(n)* we get,

$$h(n) = \frac{1}{2\pi} \int_{-\Omega c}^{\Omega_c} 1 \bullet e^{jn\Omega} d\Omega = \frac{1}{2\pi} \frac{e^{jn\Omega}}{jn} \Bigg]_{-\Omega c}^{\Omega_c}$$

$$h(n) = \frac{1}{n\pi} \left(\frac{e^{jn\Omega} - e^{-jn\Omega}}{2j} \right) = \frac{\sin(n\Omega_c)}{n\pi}$$

$$h(n) = \frac{\Omega_c}{\pi} \times \frac{\sin(n\Omega_c)}{n\Omega_c}$$

Substituting the value of h_n in Equation 8.3, we get the Transfer Function $H(\Omega)$ as

$$H(\Omega) = \sum_{n=-\infty}^{\infty} \left(\frac{\Omega_c}{\pi} \times \frac{\sin(n\Omega_c)}{n\Omega_c} \right) e^{-j\Omega n} \qquad 8.5$$

The coefficients *h(n)* of the Equation 8.5 are equivalent of the time based impulse response of the ideal frequency response. According to Fourier, we need infinite numbers of these coefficients to truly represent the given periodic function.

From a practical standpoint, the number of coefficients is a matter of preference; since, a filter with infinite coefficients is inconceivable, we limit ourselves to a finite number *N*.

We will see the consequences of choosing a higher or lower value later in the section, but next, we discuss its magnitude response obtained from the filter Transfer Function.

Filter Transfer Function

A Transfer Function of a filter is a function of frequency and it is the Fourier Transform of the filter's impulse response. It is also the ratio of the Transform of the output over the Transform of the input. A plot of the Transfer Function vs. frequency variable Ω shows how the output magnitude varies with the different input frequencies. Taking the Fourier Transform of the impulse response of an ideal filter, we describe the Transfer Function as,

$$H(\Omega) = \frac{Y_z}{X_z} = \sum_{n=-\infty}^{\infty} h_n e^{-jn\Omega}$$

The ideal frequency response required infinite summation, but suppose we truncate the series to a finite $2N+1$ number of terms. The Transfer Function with these reduced numbers evaluated is given in Equation 8.6.

$$H(\Omega) = \sum_{n=-N}^{N} h_n e^{-j\Omega n}$$

$$H(\Omega) = h_{-N}e^{j\Omega N} + h_{-(N-1)}e^{j\Omega(N-1)} \cdots h_0 e^0 \cdots h_{(N-1)}e^{-j\Omega(N-1)} + h_N e^{-j\Omega N} \qquad 8.6$$

The Equation 8.6 is simply a polynomial in h_n evaluated on the unit circle. (See Appendix B Chapter 8, fig8_17 through fig8_21 for the Scilab instructions to perform the evaluation), But we could also use the following derivative and obtain a close form expression of the filter Transfer Function.

Substituting the equality,

$$h_{-n} = h_n$$

$$e^{-j\Omega n} + e^{j\Omega n} = 2\cos(\Omega n)$$

We obtain the following equation for the Transfer Function,

$$H(\Omega) = h(0) + 2h_1 \cos\Omega + 2h_2 \cos 2\Omega + \cdots + 2h_{n-1}\cos(N-1)\Omega + 2h_N \cos N\Omega$$

$$H(\Omega) = h(0) + 2\sum_{k=1}^{N} h(n)\cos k\Omega \qquad 8.7$$

There is no imaginary component in the Equation 8.7 that means the filter has linear **phase response,** and the Transfer Function magnitude is simply,

$$|H(\Omega)| = h(0) + 2\sum_{k=1}^{N} h(n)\cos k\Omega$$

Substituting the values for h_n computed with the Equation 8.7 we obtain an expression for the amplitude response.

With the knowledge of filter coefficients and the design of magnitude response, we are ready to implement FIR filters.

FIR Filter design

We will be discussing the four different kinds of filters, the low-pass, the high-pass, the band-pass and the band-stop filters. The frequency response of each is depicted in the Figure 8.1. Implementing a filter is simply a matter of finding the right impulse response and developing a convolution algorithm that would convolve the input samples with the appropriate impulse response, (similar to the IIR filter of Chapter 7). Thus, the quest is to obtain the right impulse response. We begin with the implementation of a low-pass filter.

Low-pass Filters

A filter is known by its frequency response and is characterized by its cutoff frequencies. From implementation point of view, the number of coefficients or filter-taps matter as they affect the speed of execution. Though, choosing a large number of filter coefficients brings the filter response close to the ideal response but it may slow down the execution. In our example filter, we chose 21 coefficients. The cutoff frequency is determined with respect to the sampling frequency. If the sampling rate is10k samples per second then a cutoff frequency of 2.5khz means $\Omega_C = 0.25\pi$.

As usual, we will perform convolution of the 21-points (of the impulse response) with the k_{th} input samples $x[k]$ to produce the k_{th} filter output $y[k]$. (the reason for picking an odd number is to have a symmetry on 0).

$$y[k] = \sum_{n=-10}^{10} h_n x[k-n]$$

The first step in filter design is to compute the impulse response corresponding to the Frequency response. We have already developed an expression for the low-pass filter by taking the inverse Fourier Transform as shown in the derivative of Equation 8.5. Ideally, we should evaluate the equation for $n = -\infty$ to $n = +\infty$, but for 21 coefficients, we would use $n = -10$ to $n = +10$. For $n \neq 0$ the evaluation is straightforward but to avoid the divide by 0 error (in case of *n=0*), the L'Hospital's rule could be used;

$$h[0] = \frac{(d/dn)\Omega_C \sin(n\Omega_C)}{(d/dn)n\pi} = \frac{\Omega_C \cos(n\Omega_C)}{\pi} = \frac{\Omega_C}{\pi} \qquad \text{for} \qquad n = 0$$

$$h[n] = \frac{\Omega_C}{\pi} \times \frac{\sin(n\Omega_C)}{n\Omega_C} \qquad \text{for} \qquad n \neq 0 \qquad\qquad 8.8$$

The impulse response given in Equation 8.8 is of the form $\sin x/x$, commonly known as sinc function. Rectangular functions of either time or frequency domain usually transform into sinc functions, when Transformed from one domain to another. The public domain software 'Scilab' has a built-in sinc function that you can use to compute the coefficients of the desired filter. The Coefficients given in Table 8.1 were generated with the help of the Scilab program and the listing is provided in Appendix B.

Example 8.1. Design a low pass filter with the cutoff frequency of 1/8th of the Nyquist frequency.

We merely have to find the filter coefficients corresponding to the cutoff frequency. It should be noted that the actual cutoff frequency is 2 times the value of the desired frequency, since we measure cutoff frequency with respect to the actual sampling frequency. Substituting the value of $\Omega_C = 0.25\pi$ and evaluating the Equation 8.1 for $n = -10$ to $n = +10$, we get 21 coefficients as described in the Table 8.1.

The Table 8.1 describes the filter coefficients $h(n)$ for $n = -10$ to $n = +10$. See Appendix B Chapter 8 for the Scilab listings to generate the coefficients.

h(-10)=h(10)	.0318310
h(-9)=h(9)	.0250088
h(-8)=h(8)	0
h(-7)=h(7)	-.0321542
h(-6)=h(6)	-.0530516
h(-5)=h(5)	-.0450158
h(-4)=h(4)	0
h(-3)=h(3)	.0750264
h(-2)=h(2)	.1591549
h(-1)=h(1)	.2250791
h(0)	.25

Convolving the impulse response with the input samples produces the filter output,

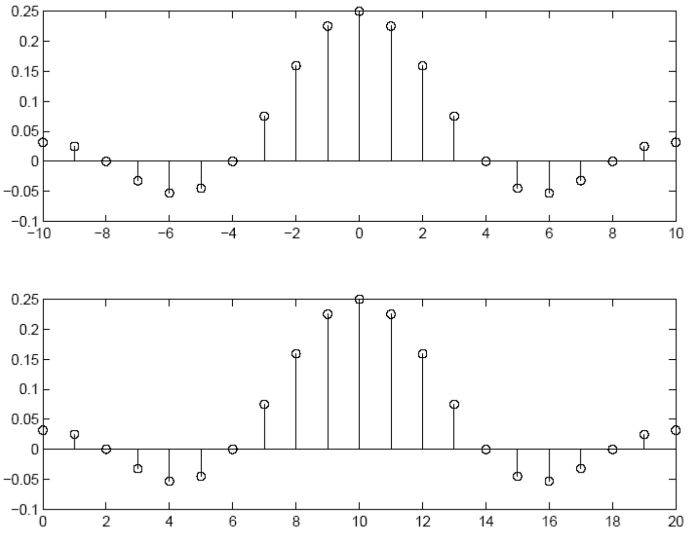
$$y[k] = \sum_{n=-10}^{10} h_n x[k-n] \qquad 8.9$$

If *k=0* is the current instance then the Equation 8.9 is expanded as,

$$y[0] = h_{-10}x[-10] + h_{-9}x[-9]\cdots h_0 x[0]\cdots + h_{10}x[10] + h_9 x[9]$$

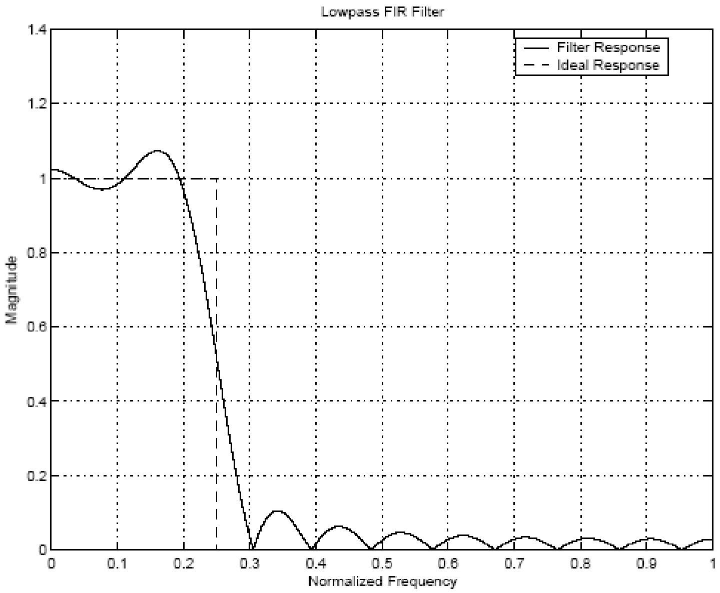
You may recognize that the filter output of Equation 8.8 is non-causal, as it requires values from the future. One way to solve the problem is to shift the filter coefficients so that they start from 0, instead of –10, as shown in the Figure 8.3. Being a linear and time invariant system, the impulse response in the future is the same as impulse response in the past. Our modified causal filter is,

$$y[k] = \sum_{n=0}^{20} h_n x[k-n]$$



****** Insert Figure 8.3 here ******
Figure 8.3. Impulse response of ideal low pass filter with cutoff frequency of 1/8th of the Nyquist, a) non-causal, b) causal

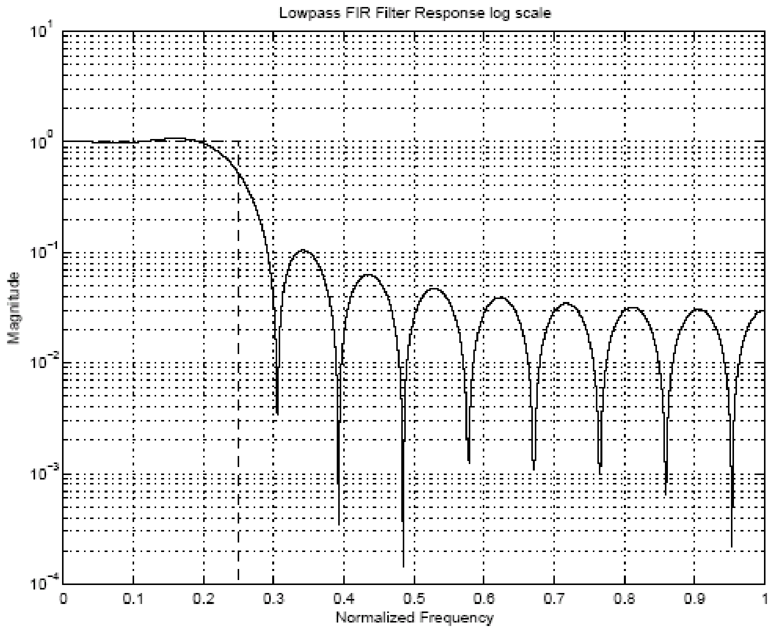
The Figure 8.4 is a plot of the frequency response of the low pass filter of Example 8.1, together with the ideal frequency response for a comparison.



****** Insert Figure 8.4 here ******

Figure 8.4. Magnitude response of the 21-tap low pass filter (cutoff frequency $+\Omega = \frac{2\pi}{5}$) compared to the ideal filter.

A logarithmic bode plot of the frequency response of Example 8.1 is presented in the Figure 8.5 to highlight the side lobe amplitude The normal plot shows the pass band ripples overshoot of 7% near the transition band.



******* Insert Figure 8.5 here *******
Figure 8.5. Logarithmic plot compared to the normal plot of the frequency response of the low pass filter of Example 8.1, showing the ripple affect.

Decimation Filters

Sometimes we acquire signals with a far higher rate then our processing or storage can manage. It would be wise to pass the data through a low-pass filter before throwing away the unwanted, simply because there are higher frequency components that may produce aliasing effect on the remaining data.

Decimation is the process of reducing the sampling rate and the Decimation factor is the ratio of the input rate to the output rate. This ratio must be an integer since we cannot throwaway part of the value. If there is a need for fractional reduction then interpolation must be performed before decimating the data. For example, to change the sample rate of a signal from 48 kHz to 44.1 kHz see Part 4: Resampling.)

2.1.5 Which signals can be downsampled?

> A signal can be downsampled (without doing any filtering) whenever it is "oversampled", that is, when a sampling rate was used that was greater than the Nyquist criteria required. Specifically, the signal's highest frequency must be less than half the *post-decimation* sampling rate. (This just boils down to applying the Nyquist criteria to the input signal, relative to the new sampling rate.)

In most cases, though, you'll end up lowpass-filtering your signal prior to downsampling, in order to enforce the Nyquist criteria at the *post-decimation* rate. For example, suppose you have a signal sampled at a rate of 30 kHz, whose highest frequency component is 10 kHz (which is less than the Nyquist frequency of 15 kHz). If you wish to reduce the sampling rate by a factor of three to 10 kHz, you must ensure that you have no components greater than 5 kHz, which is the Nyquist frequency for the reduced rate. However, since the original signal has components up to 10 kHz, you must lowpass-filter the signal prior to downsampling to remove all components above 5 kHz so that no aliasing will occur when downsampling.

This combined operation of filtering and downsampling is called *decimation*.

2.1.6 What happens if I violate the Nyquist criteria in downsampling or decimating?

You get aliasing--just as with other cases of violating the Nyquist criteria. (Aliasing is a type of distortion which cannot be corrected once it occurs.)

2.2 Multistage

2.2.1 Can I decimate in multiple stages?

Yes, so long as the decimation factor, M, is not a prime number. For example, to decimate by a factor of 15, you could decimate by 5, then decimate by 3. The more prime factors M has, the more choices you have. For example you could decimate by a factor of 24 using:

- one stage: 24
- two stages: 6 and 4, or 8 and 3
- three stages: 4, 3, and 2
- four stages: 3, 2, 2, and 2

2.2.2 Cool. But why bother with all that?

If you are simply downsampling (that is, throwing away samples without filtering), there's no benefit. But in the more common case of decimating (combining filtering and downsampling), the computational and memory requirements of the filters can usually be reduced by using multiple stages.

2.2.3 OK, so how do I figure out the optimum number of stages, and the decimation factor at each stage?

That's a tough one. There isn't a simple answer to this one: the answer varies depending on many things, so if you really want to find the optimum, you have to evaluate the resource requirements of each possibility.

However, here are a couple of rules of thumb which may help narrow down the choices:

- Using two or three stages is usually optimal or near-optimal.
- Decimate in order from the largest to smallest factor. In other words, use the largest factor at the highest sampling rate. For example, when decimating by a factor of 60 in three stages, decimate by 5, then by 4, then by 3.

The multirate book references give additional, more specific guidance.

2.3 Implementation

2.3.1 How do I implement decimation?

Decimation consists of the processes of lowpass filtering, followed by downsampling.

To implement the filtering part, you can use either FIR or IIR filters.

To implement the downsampling part (by a downsampling factor of "M") simply keep every *Mth* sample, and throw away the *M-1* samples in between. For example, to decimate by 4, keep every fourth sample, and throw three out of every four samples away.

2.3.2 That almost sounds too easy...

Beauty, eh? ;-)

2.3.3 If I'm going to throw away most of the lowpass filter's outputs, why bother to calculate them in the first place?

You may be onto something. In the case of FIR filters, any output is a function only of the past inputs (because there is no feedback). Therefore, you only have to calculate outputs which will be used.

For IIR filters, you still have to do part or all of the filter calculation for each input, even when the corresponding output won't be used. (Depending on the filter topology used, certain feed-forward parts of the calculation can be omitted.),. The reason is that outputs you *do* use are affected by the feedback from the outputs you *don't* use.

The fact that only the outputs which will be used have to be calculated explains why decimating filters are almost always implemented using FIR filters!

2.4 FIR Decimators

2.4.1 What *computational* savings do I gain by using a FIR decimator?

Since you compute only one of every M outputs, you save M-1 operations per output, or an overall "savings" of (M-1)/M. Therefore, the larger the decimation factor is, the larger the savings, percentage-wise.

A simple way to think of the amount of computation required to implement a FIR decimator is that it is equal to the computation required for a non-decimating N-tap filter operating at the *output* rate.

2.4.2 How much *memory* savings do I gain by using a FIR decimator?

None. You still have to store every input sample in the FIR's delay line, so the memory requirement is the same size as for a non-decimated FIR having the same number of taps.

2.4.3 How do I design a FIR decimator?

Just use your favorite FIR design method. The design criteria are:

1. The passband lower frequency is zero; the passband upper frequency is whatever information bandwidth you want to preserve after decimating. The passband ripple is whatever your application can tolerate.
2. The stopband lower frequency is half the output rate minus the passband upper frequency. The stopband attenuation is set according to whatever aliasing your application can stand. (Note that there will always *be* aliasing in a decimator, but you just reduce it to a negligible value with the decimating filter.)
3. As with any FIR, the number of taps is whatever is required to meet the passband and stopband specifications.

2.4.4 How do I implement a FIR decimator?

A decimating FIR is actually the same as a regular FIR, except that you shift M samples into the delay line for each output you calculate. More specifically:

1. Store M samples in the delay line.

2. Calculate the decimated output as the sum-of-products of the delay line values and the filter coefficients.
3. Shift the delay line by M places to make room for the inputs of the next decimation.

Also, just as with ordinary FIRs, circular buffers can be used to eliminate the requirement to literally shift the data in the delay line.

2.4.5 Where can I get source code to implement a FIR decimator in C?

Right here. Our "multirate_algs" package includes a decimation routine. You can download it from dspGuru's DSP Algorithm Library.

2.4.6 Where can I get assembly code to implement a FIR decimator?

The major DSP vendors provide examples of FIR decimators in their data books and application notes; check their web sites.

2.4.7 How do I test a FIR decimator?

You can test a decimating FIR in most of the ways you might test an ordinary FIR:

1. A special case of a decimator is an "ordinary" FIR. When given a value of "1" for M, a decimator should act exactly like an ordinary FIR. You can then do impulse, step, and sine tests on it just like you can on an ordinary FIR.
2. If you put in a sine whose frequency is within the decimator's passband, the output should be distortion-free (once the filter reaches steady-state), and the frequency of the output should be the same as the frequency of the input, in terms of absolute Hz.
3. You also can extend the "impulse response" test used for ordinary FIRs by using a "fat impulse", consisting of M consecutive "1" samples followed by a series of "0" samples. In that case, if the decimator has been implemented correctly, the output will not be the literal FIR filter coefficients, but will be the sum of every subset of M coefficients.
4. You can use a step response test. Given a unity-valued step input, the output should be the sum of the FIR coefficients once the filter has reached steady state.

High-pass filters

The high-pass filter frequency response is as shown in the Figure 8.1.b. The only non-zero values are from $-\pi$ to $-\Omega_1$ and from Ω_1 to π; this is as if the low-pass filter is subtracted from 1.

$$H_{HP}(\Omega) = 1 - H_{LP}(\Omega) \qquad 8.10$$

The corresponding impulse response is,

$$h_{hp}(n) = \delta(n) - h_{lp}(n)$$

Where $\delta(n) = 1$ when n=0 and 0 otherwise.

Substituting the value of *h(n)* from the derivative of Equation 8.10 we get high pass filter coefficients,

$$h_{hp}(n) = 1 - \frac{\Omega_1}{\pi} \times \frac{\sin(n\Omega_1)}{n\Omega_1} \qquad \text{for } n = 0$$
$$h_{hp}(n) = -\frac{\Omega_1}{\pi} \times \frac{\sin(n\Omega_1)}{n\Omega_1} \qquad \text{for } n \neq 0$$

The following is an example of high pass filter design.

Example 8.2. Design a high pass filter with the cutoff frequency of 3/5th of the Nyquist frequency.

Computed values of 21 coefficients for a high cutoff frequency of 0.6π ($\Omega_1 = 0.3\pi$) are shown in the Table 8.2 for $n = -10$ to $n = +10$.

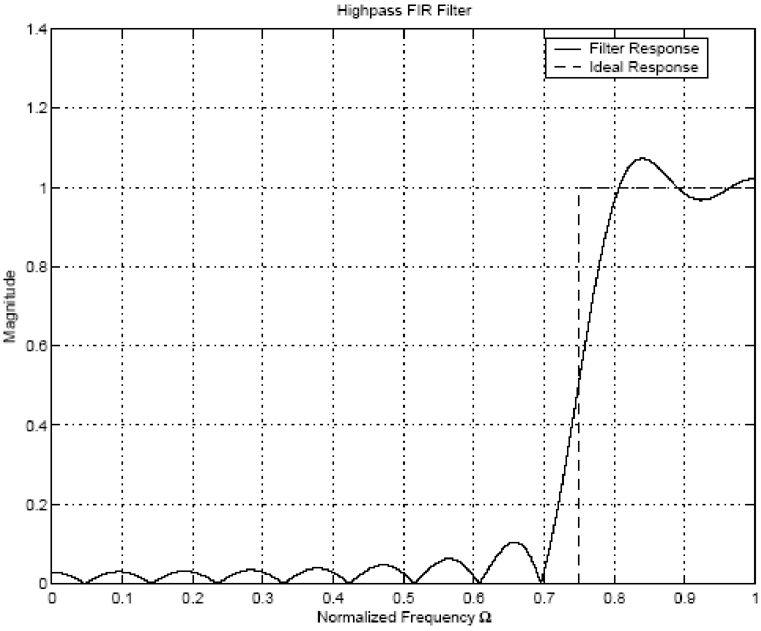
Table 8.2.The filter coefficients for the high pass filter with the cutoff frequency of 0.6π.

h(-10)=h(10)	0
h(-9)=h(9)	.0336367
h(-8)=h(8)	-.0233872
h(-7)=h(7)	-.0267283
h(-6)=h(6)	.0504551
h(-5)=h(5)	0
h(-4)=h(4)	-.0756827
h(-3)=h(3)	.0623660
h(-2)=h(2)	.0935489
h(-1)=h(1)	-.3027307
h(0)	.4

The convolution equation of the high pass filter coefficients with the input samples is presented in Equation 8.11.

$$y[k] = \sum_{n=0}^{20} h_{HP} x[k-n] \qquad 8.11$$

The frequency response is as shown in the Figure 8.6.



****** Insert Figure 8.6 here ******
Figure 8.6. Magnitude response of the 21-tap high pass filter (cutoff frequency $+\Omega = \frac{3\pi}{5}$) compared to the ideal filter.

Band-pass Filters

The ideal Band-pass filter frequency response is shown in the Figure 8.1.c. The band pass center frequency is Ω_0 and $\Omega_1 = \Omega_0 - \Omega_C, \Omega_2 = \Omega_0 + \Omega_C$. The cutoff frequency Ω_C refers to the low pass filter of Figure 8.1.a. The corresponding impulse response is obtained from the appropriate integration of Equation 8.4,

$$h(n) = (\frac{1}{\Omega_1 - \Omega_2} \int_{-\Omega_{-1}}^{-\Omega_2} H(\Omega) e^{j\Omega n} d\Omega) + (\frac{1}{\Omega_2 - \Omega_1} \int_{\Omega_{-2}}^{\Omega_2} H(\Omega) e^{j\Omega n} d\Omega)$$

$$h_{BP}(n) = \frac{\Omega_1}{\pi} \times \frac{\sin(n\Omega_1)}{n\Omega_1} - \frac{\Omega_2}{\pi} \times \frac{\sin(n\Omega_2)}{n\Omega_2} \qquad 8.12$$

The following example demonstrates the design of a band pass filter using filter coefficients of Equation 8.12.

Example 8.3. Design a Band pass filter with the low cutoff frequency 2/5th of the sampling frequency and the high cutoff frequency 3/5th of the sampling frequency.

For the band pass filter of ($\Omega_1 = 0.4\pi$) and the high cutoff frequency of ($\Omega_2 = 0.6\pi$) are shown in the Table 8.3 for $n = -10$ to $n = +10$. (See Appendix B for the Scilab listing to generate the coefficients).

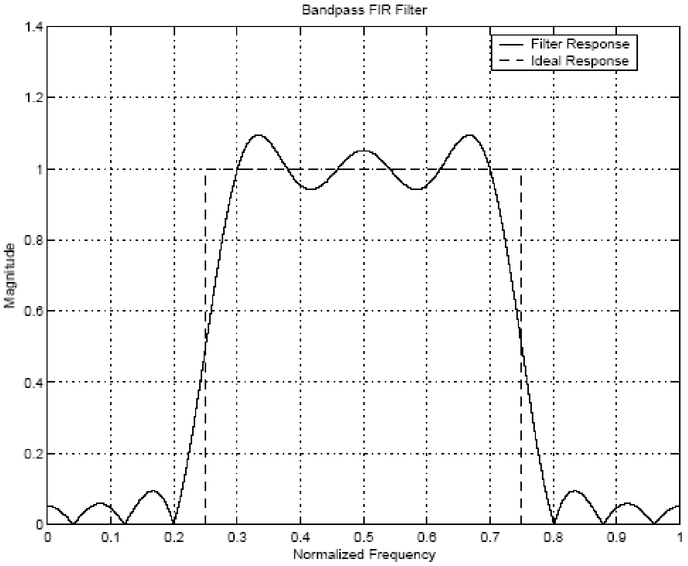
Table 8.3. The filter coefficients for a band pass filter for $n = -10$ to $n = +10$.

h(-10)=h(10)	-.0318310
h(-9)=h(9)	-.0250088
h(-8)=h(8)	0
h(-7)=h(7)	.0321542
h(-6)=h(6)	.0530516
h(-5)=h(5)	.0450158
h(-4)=h(4)	0
h(-3)=h(3)	-.0750264
h(-2)=h(2)	-.1591549
h(-1)=h(1)	-.2250791
h(0)	-.2

And the filter output is,

$$y[k] = \sum_{n=0}^{20} h_{BP} x[k-n] \qquad 8.13$$

The frequency response of the Equation 8.12 is as shown in the Figure 8.7.



****** Insert Figure 8.7 here ******
Figure 8.7. Magnitude response of the 21-tap band pass filter of Example 8.3

Band-stop Filters

We can obtain an expression for a band-stop filter, similar to band-pass filter by the appropriate integration of Equation 8.4,

$$h(n) = (\frac{1}{\Omega_1 - \Omega_2} \int_{-\Omega_{-1}}^{-\Omega_2} H(\Omega) e^{j\Omega n} d\Omega) + (\frac{1}{\Omega_2 - \Omega_1} \int_{\Omega_{-2}}^{\Omega_2} H(\Omega) e^{j\Omega n} d\Omega)$$

$$h_{BS}(n) = \frac{\Omega_1}{\pi} \times \frac{\sin(n\Omega_1)}{n\Omega_1} - \frac{\Omega_2}{\pi} \times \frac{\sin(n\Omega_2)}{n\Omega_2} \qquad 8.14$$

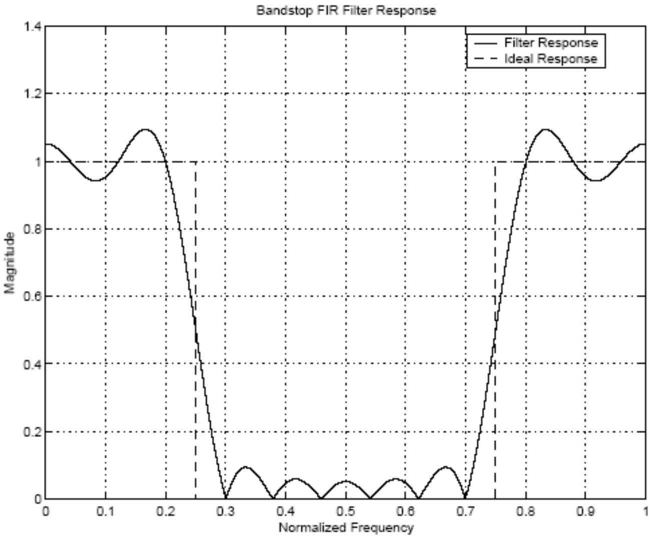
Example 8.4. Design a Band stop filter with the low cutoff frequency 2/5th of the sampling frequency and the high cutoff frequency 4/5th of the sampling frequency.

For the band pass filter of ($\Omega_1 = 0.4\pi$) and the high cutoff frequency of ($\Omega_2 = 0.8\pi$) are shown in the Table 8.3 for $n = -10$ to $n = +10$. Please see Appendix B for the Scilab listing to generate the coefficients.

Table 8.4. The filter coefficients for a band pass filter for $n = -10$ to $n = +10$.

h(-10)=h(10)	0
h(-9)=h(9)	-.0128481
h(-8)=h(8)	-.0612286
h(-7)=h(7)	.0699755
h(-6)=h(6)	.0192721
h(-5)=h(5)	0
h(-4)=h(4)	-.0289082
h(-3)=h(3)	-.1632762
h(-2)=h(2)	.2449143
h(-1)=h(1)	.1156328
h(0)	-.4

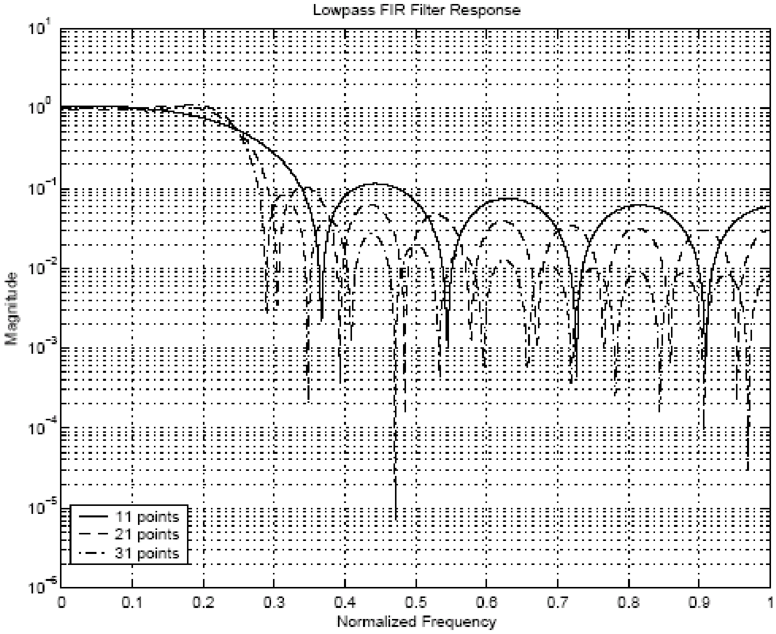
A plot of the filter Transfer Function of the Equation 8.14 is shown in the Figure 8.8.



****** Insert Figure 8.8 here ******

Figure 8.8. Magnitude response of the 21-tap band stop filter (low cutoff frequency $+\Omega = \frac{2\pi}{5}$), (high cutoff frequency $+\Omega = \frac{4\pi}{5}$), compared to the ideal filter

The FIR filters derived from the ideal response always produce ripples in the pass band region. These ripples are the result of truncating the infinite coefficients Fourier series while ignoring sinusoids in the process of summation. Increasing the number of filter coefficients improves the ripples somewhat (see Figure 8.9 for a comparison of 11, 21 and coefficients Transfer Function). These ripples are unavoidable, but there are ways to improve the design and that is topic of discussion of our next section.



******* Insert Figure 8.9 here *******

Figre 8.9. The logarithmic response of 11-tap, 21-tap and 31-tap Rectangular Window

Windowing

Frequency domain multiplication is akin to time domain convolution; similarly, time domain multiplication is the same as frequency domain convolution. Performing Fourier Transform on a set of data is like having a window to a different domain. From the window of convolution, we see frequencies and through frequency domain multiplication, we see time series. We see one through the other. Imagine (when performing the infinite summation) as if you are looking through a convolving window, and you see all the frequencies that are present in the impulse response. But a truncated series would miss out some frequencies and that shows as ripples in the Transfer Function. It has the same effect as multiplying an infinite impulse response with a rectangular window of 1's and 0's. In our example filters of 21 coefficients, we were forced to truncate the infinite number of coefficients into a fixed number of 21 values. We were by default using a rectangular window of twenty-one 1's and rest 0's as shown in the Figure 8.10.

0,0,0,..0,0,0,1,1,1…1,1,1,0,0,0..0,0,0

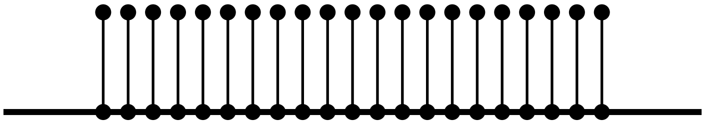
The Fourier series equivalent of the ideal frequency response of Figure 8.1b is given as,

$$g(\Omega) = \sum_{k=-\infty}^{k=\infty} c_k e^{jk\Omega}$$

The coefficients c_k represent the amplitudes of the impulse response. But for practical consideration, the infinite summation must be truncated to a finite value,

$$g(\Omega) = \sum_{k=-N}^{k=N} c_k e^{jk\Omega}$$

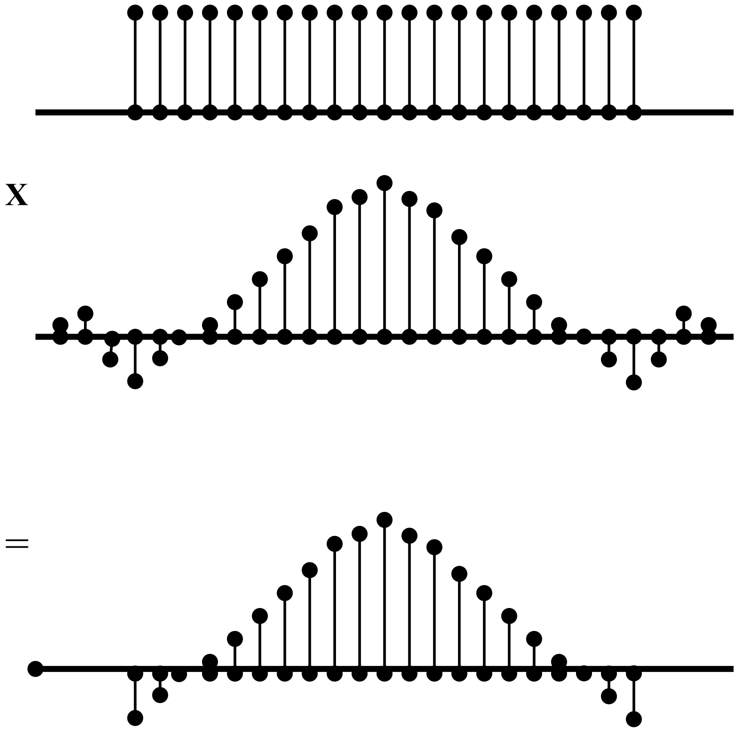
The affect of multiplying the c_k values with a series of (2N+1) non-zero terms can be described graphically, as shown in the Figure 8.11. The number of sample points (2N+1) merely indicates a symmetric window with N number of sample points on each side of the center.



******* Insert Figure 8.10 here *******

Figure 8.10. Discrete time rectangular window of 21 points

The window determines how much of the impulse response can be seen. Looking from the multiplication window, an infinite length impulse response *h(n)* (essentially, an inverse Fourier Transform of an ideal frequency response) when multiplied with a rectangular window *w(n)* produces a truncated impulse response *t(n)*. This is a time domain multiplication, but the effect is same as frequency domain convolution and that is what we obtained when we realized the Transfer Function through a truncated series of impulse response, see the Figure 8.11. The result of chopping of the ends of the infinite series shows up as ripples in the Transfer Function near the Transition band as well as side lobes in the stop band.



******* Insert Figure 8.11 here *******
Figure 8.11. Time domain multiplication of the Rectangular Window with the infinite impulse response of the ideal infinite Rectangular Window

Ideally, if the rectangular convolving window were as large as the frequency response window, there wouldn't be any distortion of the resulting Transfer Function. Looking from another angle, the process of obtaining the actual impulse response $H_A(\Omega)$ is as if we have done a frequency domain convolution of a rectangular convolving window $W(\Omega)$ with an ideal frequency response $H_I(\Omega)$.

$$H_A(\Omega) = H_I(\Omega) * W(\Omega)$$

Besides rectangular window, other functions produce coefficients that when multiplied by the impulse response of the ideal filters improve the filter performance, as shown in the next section.

The Rectangular Window

We characterize a rectangular window as a series of (2N+1) non-zero terms. The values are either 1 (inside the window) or 0 (outside the window). It is convenient to have a window symmetric on 0, thus we have N sample points on each side. We can always shift the window so that it begins with 0 to make it a causal window.

0,0,0,..0,0,0,1,1,1…1,1,1,0,0,0..0,0,0

For 2N+1 number of 1s, the window becomes,

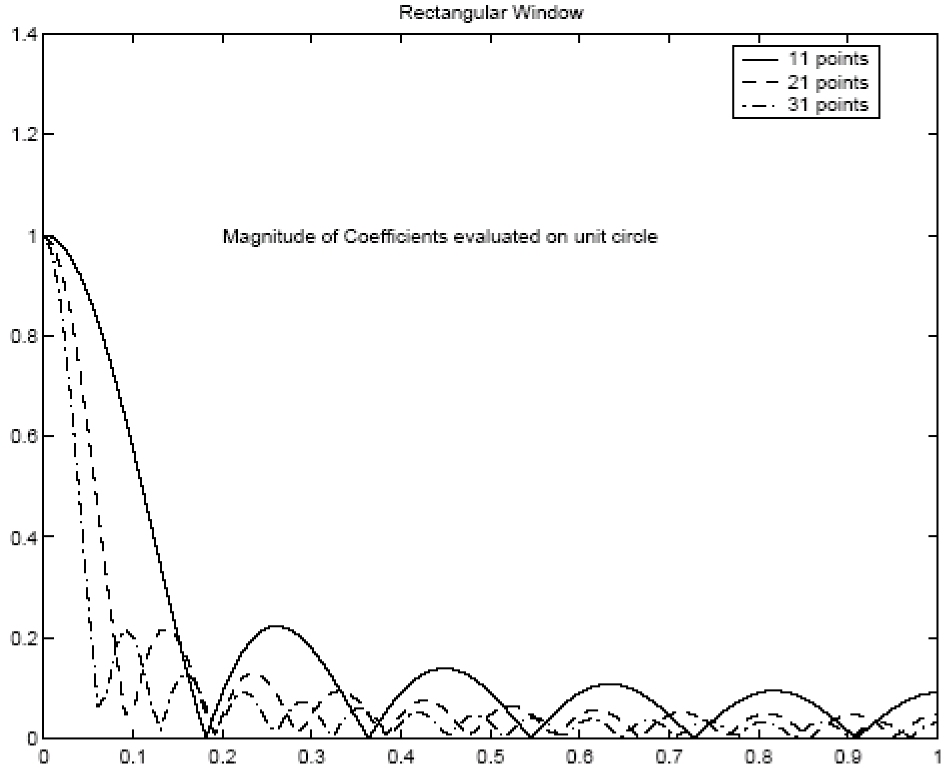
$$a_k = \begin{cases} 1 & \left| k \le |N| \right| \\ 0 & \left| k > |N| \right| \end{cases} \qquad 8.16$$

The Fourier Transform of the rectangular window of Equation 8.16 is,

$$W_R(\Omega) = \sum_{k=-N}^{k=N} e^{jk\Omega} = e^{-jN\Omega} + e^{-j(N-1)\Omega} + \cdots + e^{j(N-1)\Omega} + e^{jN\Omega} \qquad 8.17$$

The Equation 8.17 is a polynomial evaluated on a unit circle whose coefficients are 1 and the Figure 8.5 is its frequency response plot where N=10. The Scilab instructions for evaluating the Transfer Function magnitude of Equation 8.17 are provided in Appendix B Chapter 8 fig8_12.

The Figure 8.12 is the magnitude of Equation 8.17 plotted for different values of N. Notice, the reduction in transition width as we increase the number of coefficients. But the side lobe amplitude remains same. The side lobe amplitude is specific to a window function and is independent of the number of coefficients in the window. For rectangular window the first side lobe amplitude is around 13 dB, which is about 1/5th of its pass band magnitude. In most situations it is probably not an acceptable value, but compare to other windows the rectangular window offers a steeper transition width.



******* Insert Figure 8.12 here *******

Figure 8.12. The plot showing the magnitude response of the Rectangular Window for different number of filter coefficients

There is an approximate relationship between the transition widths and the number of coefficients, and for a rectangular window, it is $(2N+1)\times\Delta f \approx 1.9$. Two aspects are important when specifying a filter,

1) The transition width.
2) The side lobe amplitude.

The number of coefficients affects the transition width, but the side lobe amplitude is fixed for a window type.

Example 8.5:

Suppose a low pass filter is specified as,

$$
\begin{array}{ll}
0.8 \le \left|H(e^{j\omega})\right| \le 1.2 & 0 \le \omega \le 0.18\pi \\
\left|H(e^{j\omega})\right| \le 0.2 & 0.22 \le \omega \le \pi
\end{array}
$$

The requirements of magnitude drop of 0.2 is within the reach of the rectangular window, and we can compute the number of coefficients for the transition width of 0.04 as,

$$(2N+1) \times 0.04 \approx 1.9$$

$$N \approx 23$$

The magnitude summation of the Equation 8.17 can also be expressed as,

$$W_R(\Omega) = \frac{e^{j(N+\frac{1}{2})\Omega} - e^{-j(N+\frac{1}{2})\Omega}}{e^{j\Omega/2} - e^{-j\Omega/2}} =$$

$$W_R(\Omega) = \frac{\sin(N+1/2)\Omega}{\sin(\Omega/2)} \qquad 8.18$$

If we were to *normalize* the window function to a unit area (meaning the sum of the coefficients to be equal to 1) then each coefficient is an equal weight of $\frac{1}{2N+1}$ and the Equation 8.18 becomes

$$a_k = \begin{cases} \frac{1}{2n+1} & |k \le N| \\ 0 & |k > N| \end{cases} \qquad 8.19$$

And the corresponding impulse response of Equation 8.19 becomes,

$$W_R(\Omega) = \frac{\sin(N+1/2)\Omega}{(1/2N+1)\sin(\Omega/2)}$$

All the filters being discussed in the previous section were designed using the rectangular window, as the coefficients were generated with the values from the ideal rectangular transfer function. Next, we discuss other window functions that have better suppression in the stop band region compare to the rectangular window.

Power Window

Shaping the rectangular window with a power function multiplier can make a simple design improvement. In fact, any function whose values tapers off towards the end can serve the purpose of a window shaper and provide a performance improvement. A function that puts increasingly extra weight towards the end is,

.

$$a_k = \begin{cases} (1-(k/N)^2 & \left|k \le |N|\right| \\ 0 & \left|k > |N|\right| \end{cases} \qquad 8.20$$

The maximum weight of the power function of Equation 8.20 is 1 when k=0 and for k=N the power function tapers off to 0 values.

$$\begin{array}{ll}
W_{PW}(0) = 1 & W_{PW}(0) = 1 \\
W_{PW}(1) = .99 & W_{PW}(11) = .99 \\
W_{PW}(2) = .96 & W_{PW}(12) = .96 \\
W_{PW}(3) = .91 & W_{PW}(13) = .91 \\
W_{PW}(4) = .84 & W_{PW}(14) = .84 \\
W_{PW}(5) = .75 & W_{PW}(15) = .75 \\
W_{PW}(6) = .64 & W_{PW}(16) = .64 \\
W_{PW}(7) = .51 & W_{PW}(17) = .51 \\
W_{PW}(8) = .36 & W_{PW}(18) = .36 \\
W_{PW}(9) = .19 & W_{PW}(19) = .19 \\
W_{PW}(10) = 0 & W_{PW}(20) = 0
\end{array}$$

$$h(n) = \frac{1}{2\pi}\int_{-\Omega_1}^{\Omega_1}(1-(n/N)^2)\bullet 1 \bullet e^{jn\Omega}d\Omega = \frac{(1-(n/N)^2)}{2\pi}\left.\frac{e^{jn\Omega}}{jn}\right]_{-\Omega_1}^{\Omega_2}$$

$$h(n) = \frac{(1-(n/N)^2)}{n\pi}\left(\frac{e^{jn\Omega} - e^{-jn\Omega}}{2j}\right) = \frac{(1-(n/N)^2)\sin(n\Omega_1)}{n\pi}$$

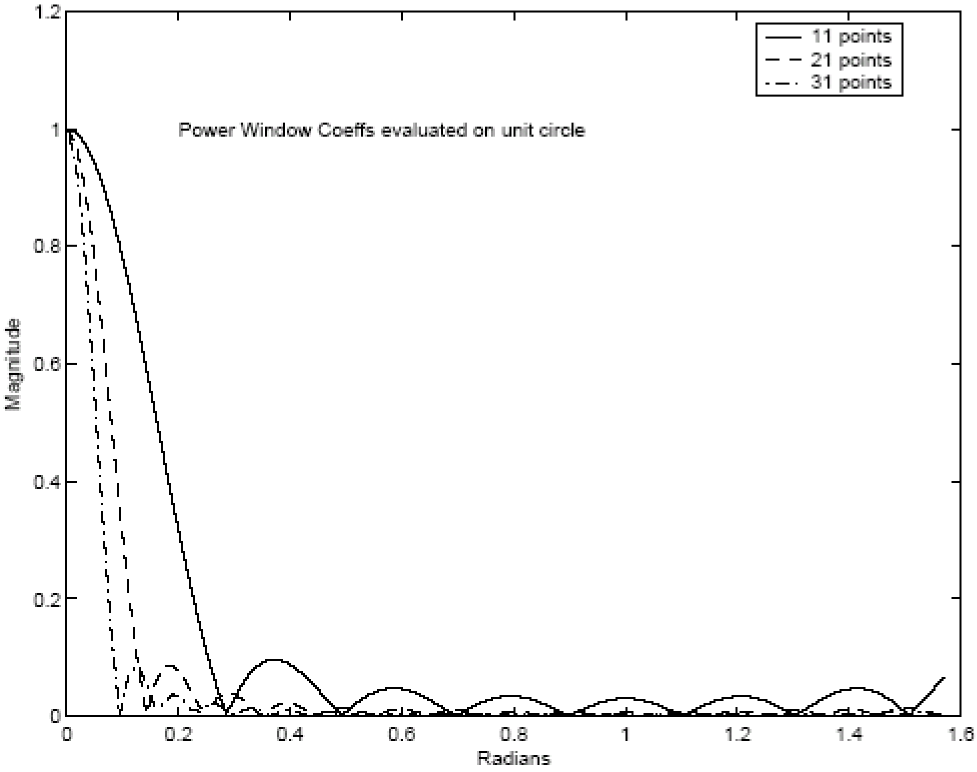
$$h(n) = \frac{\Omega_1(1-(n/N)^2)}{\pi} \times \frac{\sin(n\Omega_1)}{n\Omega_1} \qquad 8.21$$

The Table 8.5 shows the calculated values of 21 coefficients of the power window.

Table 8.5. Coefficients for a 21-tap low pass filter (cutoff frequency $\Omega_C = 0.4$).

h(-10)=h(20)	0
h(-9)=h(19)	-.0063910
h(-8)=h(18)	-.0084194
h(-7)=h(17)	.0136314
h(-6)=h(16)	.0322913
h(-5)=h(15)	0
h(-4)=h(14)	-.0635734
h(-3)=h(13)	-.0567530
h(-2)=h(12)	.0898070
h(-1)=h(11)	.2997034
h(0)	.4

The Figure 8.13 is the magnitude of Equation 8.21 plotted for different values of *N* and the Scilab instructions for evaluating the Transfer Function magnitude are provided in Appendix B Chapter 8 fig8_13. There is some improvement in the side-lobe amplitude as it dropped down to nearly 21 dB but the transition width goes up from 1.9 to 2.8 for the same number of coefficients. It only indicates that in order to meet the requirement of steeper roll-off of frequency response we must increase the number of coefficients.



******* Insert Figure 8.13 here *******
Figure 8.13. The plot showing the magnitude response of the Power Window for different number of filter coefficients

The following example illustrates the difference between rectangular window and the power window for the number of coefficients required for the same transition width as in rectangular window.

Example 8.5:

Suppose a low pass filter is specified as,

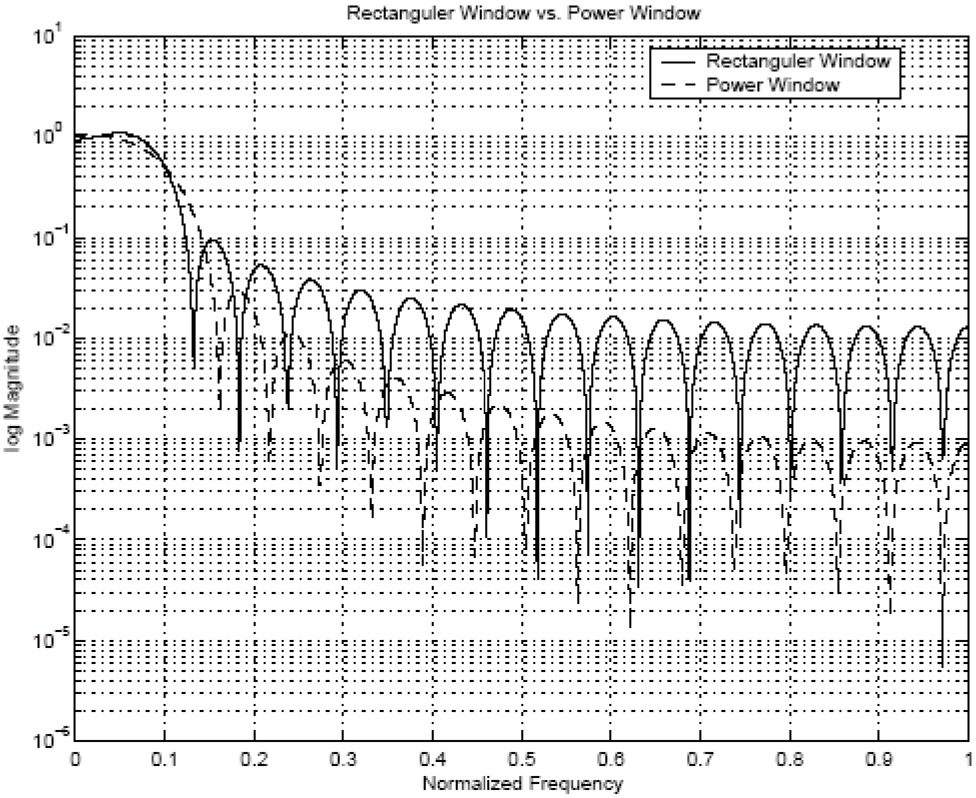
$$
\begin{aligned}
0.1 \le \left|H(e^{j\omega})\right| &\le 1.1 \qquad\qquad 0 \le \omega \le 0.18\pi \\
\left|H(e^{j\omega})\right| &\le 0.1 \qquad\qquad 0.22 \le \omega \le \pi
\end{aligned}
$$

The requirements of magnitude drop of 0.1 is within the reach of the power window, and to achieve the transition width of 0.04, we get,

$(2N+1)\times 0.04 \approx 2.8$

$N \approx 35$

The Figure 8.14 is the plot of the filter response of Example 8.1 using Hamming Window coefficients (together with rectangular window for comparison). There is a 3% overshoot near the transition band but the filter has a steeper response compare to other window functions, see the Figure 8.21.



******* Insert Figure 8.14 here *******

Figure 8.14. Figure showing reduction in the side lobe amplitude with the Power Window compared to the Rectangular Window.

Von Hann Window

A more severe weighting average that would put extra weight at the end is a cosine function, given in Equation 8.22. This is the Von Hann window or the raised cosine window,

$$a_k = \begin{cases} \dfrac{1+\cos \pi k/N}{2} & |k \leq N| \\ 0 & |k > N| \end{cases} \qquad 8.22$$

The following table provides the values of Equation 8.22 for *k*=0 through 9, and being a symmetric window the values for *k* = -1 to –9 are same as *k*=1 through 9.

$W_{PW}(0) = 1$

$W_{PW}(1) = .977486$

$W_{PW}(2) = .91214782$

$W_{PW}(3) = .81038122$

$W_{PW}(4) = .678$

$W_{PW}(5) = .54$

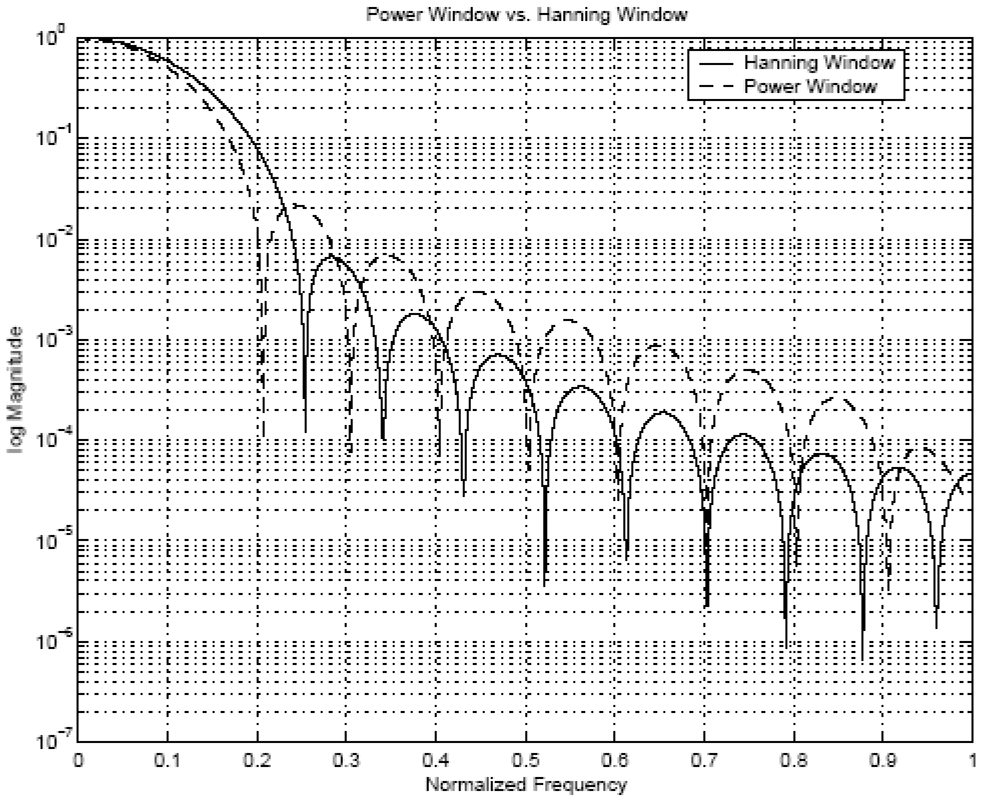
$W_{PW}(6) = .3978$

$W_{PW}(7) = .2696$

$W_{PW}(8) = .172$

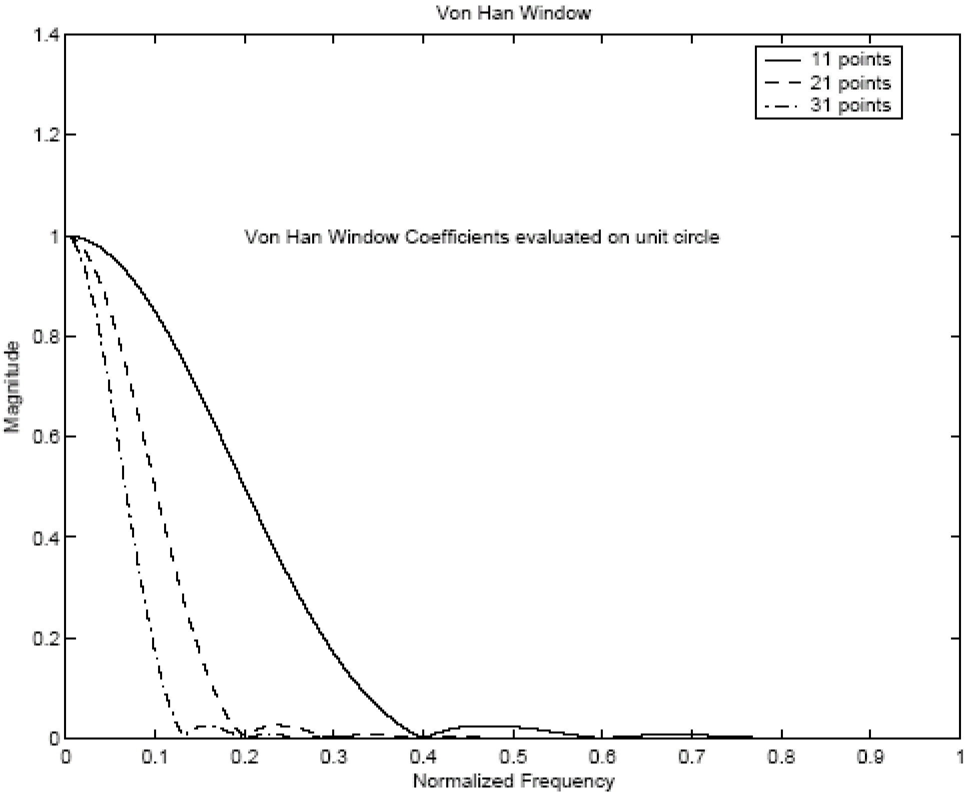
$W_{PW}(9) = .1025$

$W_{PW}(10) = 0.08$



******* Insert Figure 8.15 here *******
Figure 8.15. Figure showing reduction in the side lobe amplitude with the Von Hann Window compared to the Rectangular Window.

The magnitude of Equation 8.22 plotted for different values of *N* is presented in Figure 8.16. (The Scilab instructions are provided in Appendix B Chapter 8 fig8_16.) The side lobe amplitude drops down to nearly 31 dB but compare to rectangular window the transition width went up from 1.9 to 4.2.



******* Insert Figure 8.16 here *******

Figure 8.16. The plot showing the magnitude response of the Von Hann window for different number of filter coefficients

The filter specifications of the Example 8.5 are met with 52 coefficients for Von Window compare to 23 coefficients of the rectangular window, but on the bright side, we do get a suppression of 31dB in the stop band region.

$$(2N+1)\times 0.04 \approx 4.2$$

$$N \approx 52$$

Hamming Window

The Hamming Window, also known as cosine window with platform is given as

$$a_k = \begin{cases} 0.54 + (1-0.54)\cos \pi k/N & |k \le N| \\ 0 & |k > N| \end{cases} \qquad 8.23$$

The following table provides the values of Equation 8.23 for *k*=0 through 9,

$W_{PW}(0) = 1$

$W_{PW}(1) = .9755$

$W_{PW}(2) = .9045$

$W_{PW}(3) = .7938$

$W_{PW}(4) = .6545$

$W_{PW}(5) = .5$

$W_{PW}(6) = .3454$

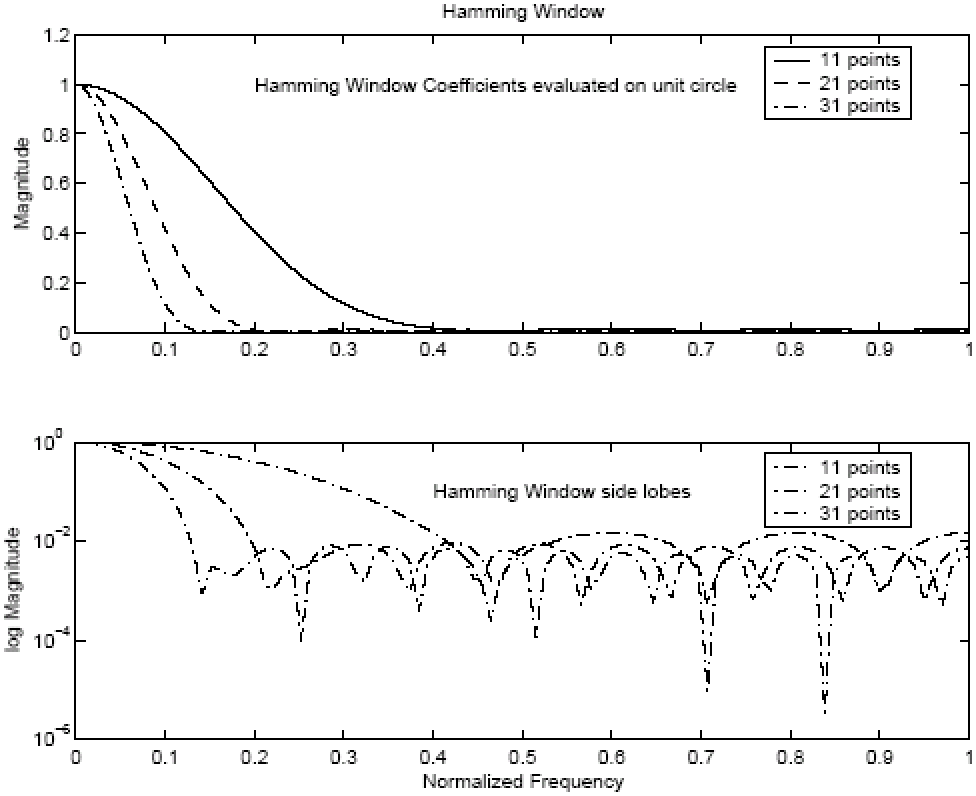
$W_{PW}(7) = .206$

$W_{PW}(8) = .095$

$W_{PW}(9) = .0244$

$W_{PW}(10) = 0$

The Figure 8.17 shows the magnitude of the Equation 8.23 plotted for different values of *N* with the help of Scilab program, (see Appendix B Chapter 8 fig8_17). The side lobe amplitude drops down to nearly 41 dB but compare to rectangular window the transition width goes up from 1.9 to 4.4.



******* Insert Figure 8.17 here *******

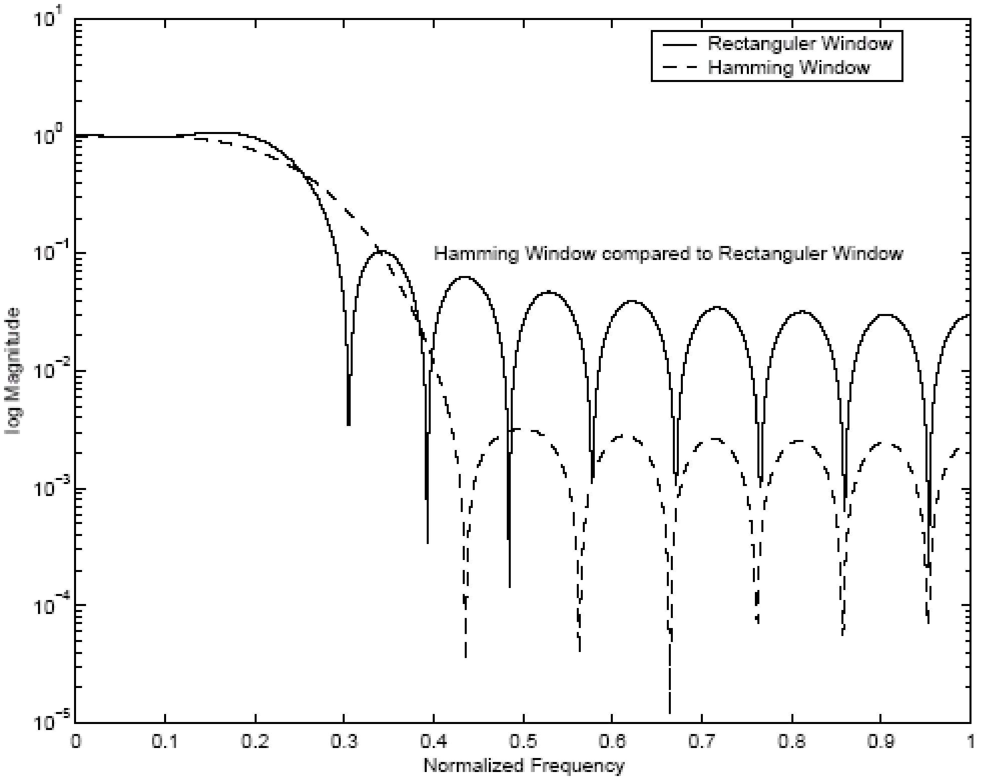
Figure 8.17. The plot showing the magnitude response of the Hamming Window for different number of filter coefficients

Compare to rectangular window a transition width of 0.4 requires,

$$(2N+1)\times 0.04 \approx 1.8$$

$$N \approx 55$$

A plot of the filter response of example 1, modified by the Hamming window is presented in the Figure 8.18 (Together with rectangular window for comparison.) There is a 0.03% overshoot near the transition band and the side lobe suppression is more aggressive, but there is an increase in the width of the roll off of transition band. (See the Figure 8.21 for comparison.)



******* Insert Figure 8.18 here *******

Figure 8.18. Figure showing reduction in the side lobe amplitude with the Hamming window compared to the rectangular window.

Kaiser Window

The Kaiser window's weighting function has a flexible design option. You can specify a range of allowable ripples in the side lobes, (those were fixed in the other window options). The function is given as,

$$a_k = \begin{cases} \dfrac{I_0\left[\beta\sqrt{1-(n/N)^2}\right]}{I_0(\beta)} & \left|\begin{matrix} k \le N \\ k > N \end{matrix}\right| \\ 0 & \end{cases} \qquad 8.23$$

$$I_0 = 1 + \sum_{m=1}^{M}\left(\frac{(x/2)^m}{m!}\right)^2$$

The parameter I_0 is the modified Bessel function of the first kind and zero order. The Equation 8.23 approximates the Bessel Function where summation limit M is an arbitrary limit. We can stop computing the Bessel terms when the value becomes less then a prescribed low limit. The Scilab program for computing Kaiser Window coefficients using Equation 8.23 is presented in Appendix B Chapter 8, (see function Kaiser_coeff()).

The β value specifies its frequency response in terms of the main lobe width and the side lobe level. Large value of β corresponds to a wider main lobe with smaller side lobe levels. The normal range of β is $4 \le \beta \le 9$. The parameter M is chosen such that the last term in series is less than 1×10^{-8}. The following table was computed using the value $\beta = 2\pi$. (See Appendix B chapter 8 fig8_21.

$W_{KR}(0) = 1$

$W_{KR}(1) = .9801$

$W_{KR}(2) = .9216$

$W_{KR}(3) = .8281$

$W_{KR}(4) = .7056$

$W_{KR}(5) = .5625$

$W_{KR}(6) = .4096$

$W_{KR}(7) = .2601$

$W_{KR}(8) = .1296$

$W_{KR}(9) = .0361$

$W_{KR}(10) = 0$

R.W Hamming has summarized the empirical relationship between the parameter β and the stop band ripples α in dB as follows,

$$\beta = \begin{vmatrix} 0.1102(\alpha - 8.7) & & \alpha > 50 \\ 0.5842(\alpha - 21)^{0.4} \_0.07886(\alpha - 21) & & 21 \le \alpha \le 50 \\ 0 & & \alpha < 21 \end{vmatrix}$$

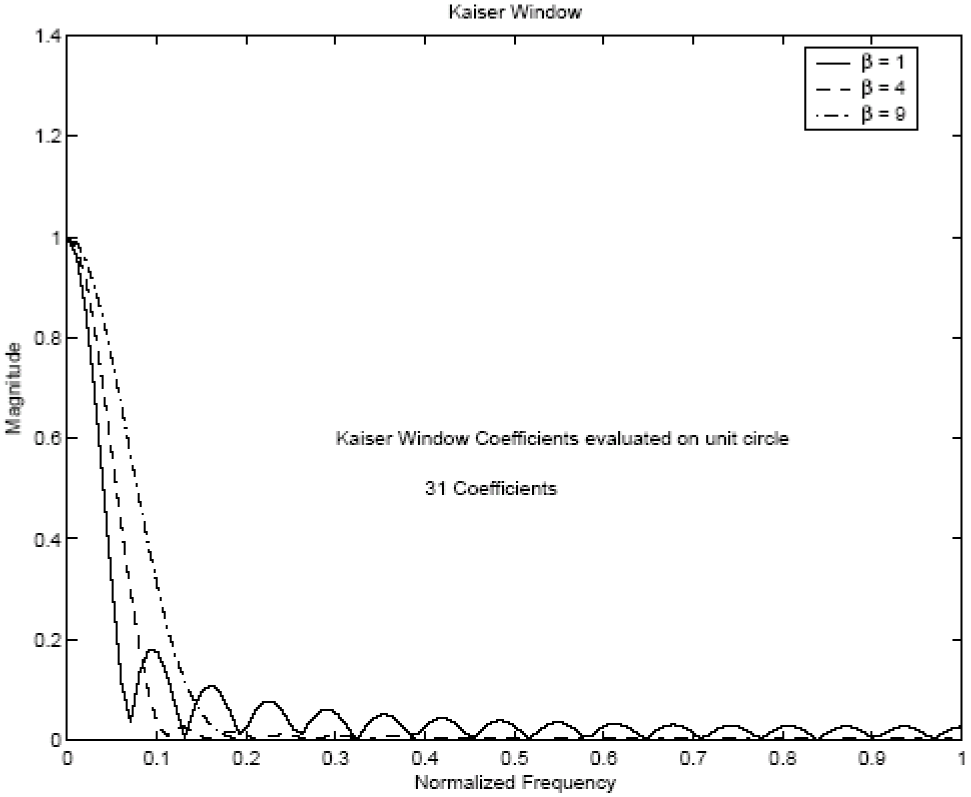
And the relationship between the transition width and the number of coefficients for is given as,

$$(2N+1) = \frac{\alpha - 7.95}{14.36\Delta f}$$

For a stop band ripple of 0.01 and transition width of $0.01f$ we could calculate the number of coefficients as,

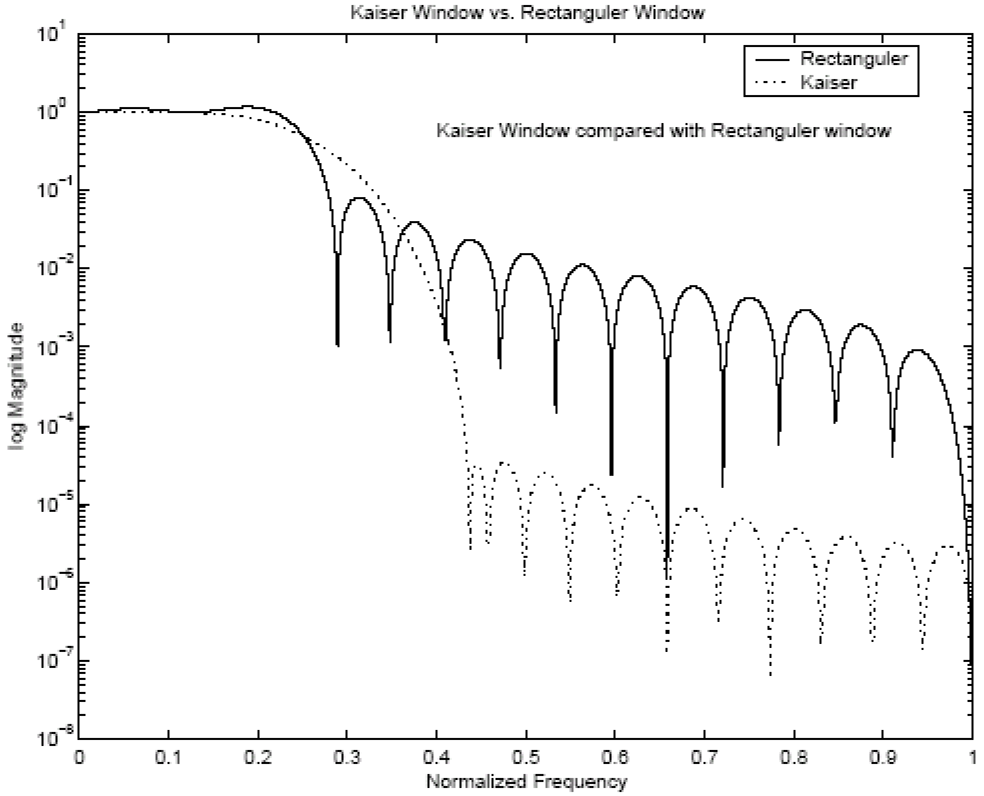
$$\alpha = 20\log(0.01) = -40$$
$$(2N+1) = \frac{40 - 7.95}{14.36 \times 0.01} = 223$$

The Figure 8.19is the magnitude of Equation 8.23 plotted for different values of *N*. The relationship between the number of coefficients and the transition width for the $\beta = 2\pi$ is given as, $(2N+1) \times \Delta f \approx 0.44$.



******* Insert Figure 8.19here *******
Figure 8.19.The plot showing the magnitude response of the rectangular window for different number of filter coefficients

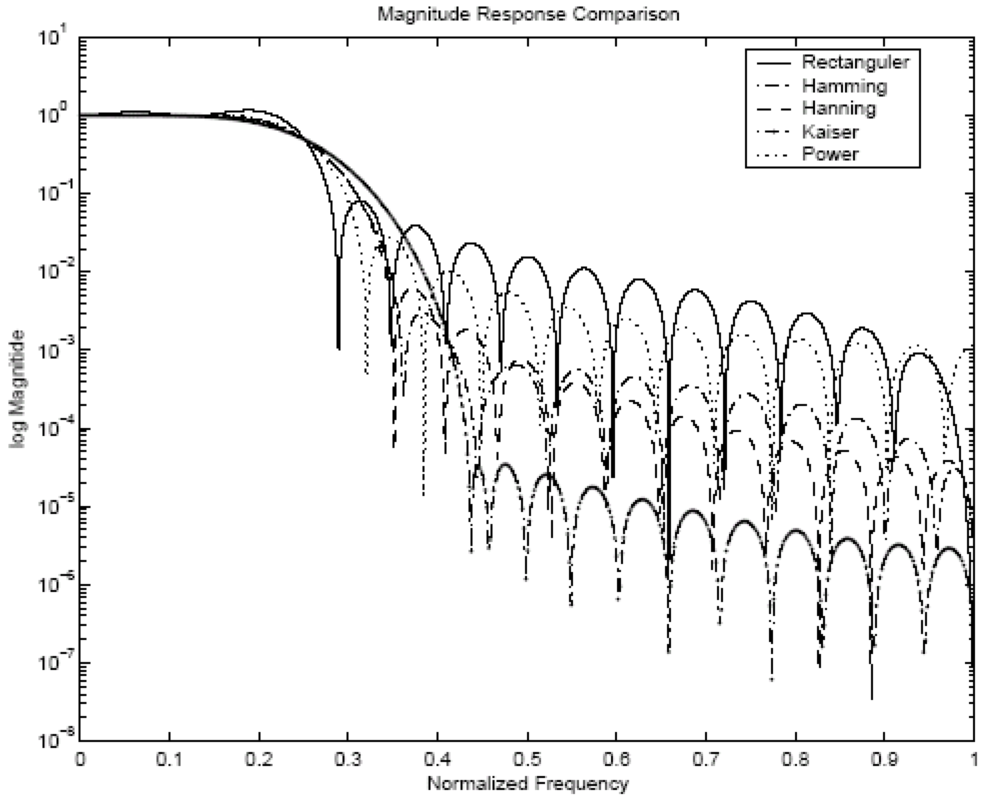
The Figure 8.20 is a plot f filter response of the Kaiser Window function of Equation 8.23 compare to the rectangular window.



******* Insert Figure 8.20ere *******

Figure 8.20. Figure showing reduction in the side lobe amplitude with the Kaiser window compared to the rectangular window.

A comparison of window functions shows that applying window function is a better alternate than increasing the coefficients of a rectangular window. The Figure 8.21 is a Bode plot of the response of the four Window functions as mentioned in the previous section.



******* Insert Figure 8.21 here *******

Figure 8.21. The comparison of the side lobe amplitudes of the Power Window, Von Hann Window, Hamming Window and the Kaiser Window

Conclusion

We have studied the Fourier Transform method of designing filters, commonly known as Finite Impulse Response (FIR) filters. The method provides filters that do not depend upon values from past, and the impulse response has a finite duration, thus, providing a linear phase response compare to the infinite impulse response method where the initial transients stay on for a long time.