Practical FIR Filter Design in MATLAB®

Revision 1.1

Ricardo A. Losada

The MathWorks, Inc.

3 Apple Hill Dr. Natick, MA 01760, USA

January 12, 2004

Abstract

This tutorial white-paper illustrates practical aspects of FIR filter design and fixed-point implementation along with the algorithms available in the Filter Design Toolbox and the Signal Processing Toolbox for this purpose.

The emphasis is mostly on lowpass filters, but many of the results apply to other filter types as well.

The tutorial focuses on practical aspects of filter design and implementation, and on the advantages and disadvantages of the different design algorithms. The theory behind the design algorithms is avoided except when needed to motivate them.

Contents

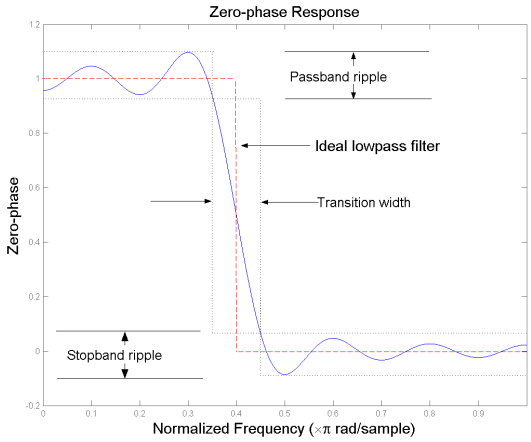

Figure 1: Illustration of the typical deviations from the ideal lowpass filter when approximating with an FIR filter, $\omega_c = 0.4\pi$.

1 Ideal lowpass filter

The ideal lowpass filter is one that allows through all frequency components of a signal below a designated cutoff frequency ω_c, and rejects all frequency components of a signal above ω_c.

Its frequency response satisfies

$$H_{\mathrm{LP}}(e^{j\omega}) = \begin{cases} 1, & 0 \le \omega \le \omega_c \\ 0, & \omega_c < \omega \le \pi \end{cases} \tag{1}$$

The impulse response of the ideal lowpass filter (1) can easily be found to be [1]

$$h_{\mathrm{LP}}[n] = \frac{\sin(\omega_c n)}{\pi n}, \quad -\infty < n < \infty. \tag{2}$$

2 FIR lowpass filters

Because the impulse response required to implement the ideal lowpass filter is infinitely long, it is impossible to design an ideal FIR lowpass filter.

Finite length approximations to the ideal impulse response lead to the presence of ripples in both the passband ($\omega < \omega_c$) and the stopband ($\omega > \omega_c$) of the filter, as well as to a nonzero transition width between the passband and stopband of the filter (see Figure 1).

2.1 FIR filter design specifications

Both the passband/stopband ripples and the transition width are undesirable but unavoidable deviations from the response of an ideal lowpass filter when approximating with a finite impulse response. Practical FIR designs typically consist of filters that meet certain design specifications, i.e., that have a transition width and maximum passband/stopband ripples that do not exceed allowable values.

In addition, one must select the filter order, or equivalently, the length of the truncated impulse response.

A useful metaphor for the design specifications in FIR design is to think of each specification as one of the angles in a triangle as in Figure 2.

The metaphor is used to understand the degrees of freedom available when designating design specifications. Because the sum of the angles is fixed, one can at most select the values of two of the specifications. The third specification will be determined by the design algorithm utilized. Moreover, as with the angles in a triangle, if we make one of the specifications larger/smaller, it will impact one or both of the other specifications.

As an example, consider the design of an FIR filter that meets the following specifications:

Specifications Set 1

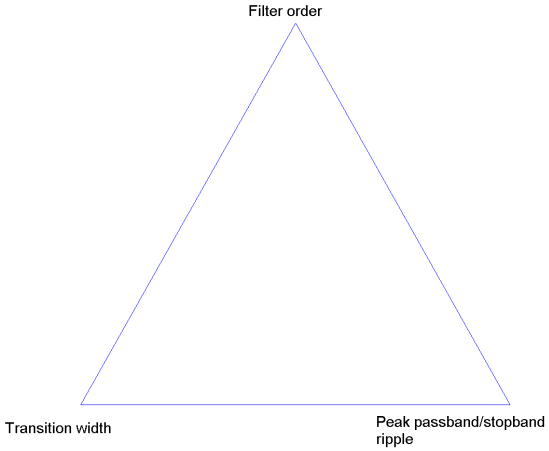

Figure 2: FIR design specifications represented as a triangle.

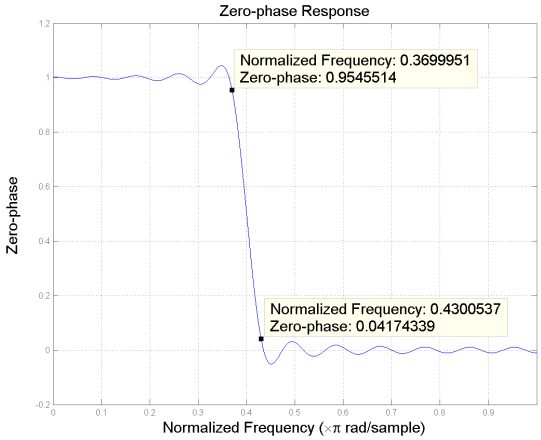

Figure 3: Kaiser window design meeting predescribed specifications.

1. Cutoff frequency: 0.4π rad/sample

2. Transition width: 0.06π rad/sample

3. Maximum passband/stopband ripple: 0.05

The filter can easily be designed with the truncated-and-windowed impulse response algorithm implemented in `fir1` (or using `fdatool`) if we use a Kaiser window. The zero-phase response of the filter is shown in Figure 3. Note that since we have fixed the allowable transition width and peak ripples, the order is determined for us.

Close examination at the passband-edge frequency, $\omega_p = 0.37\pi$[1] and at the stopband-edge frequency $\omega_s = 0.43\pi$ shows that the peak passband/stopband ripples are indeed within the allowable specifications. Usually the specifications are exceeded because the order is rounded to the next integer greater than the actual value required.

[1]The passband-edge frequency is the boundary between the passband and the transition band. If the transition width is TW, the passband-edge frequency ω_p is given in terms of the cutoff frequency ω_c by $\omega_p = \omega c - \mathrm{TW}/2$ Similarly, the stopband-edge frequency is given by $\omega_s = \omega c + \mathrm{TW}/2$.

3 Optimal FIR designs with fixed transition width and filter order

While the truncated-and-windowed impulse response design algorithm is very simple and reliable, it is not optimal in any sense. The designs it produces are generally inferior to those produced by algorithms that employ some optimization criteria in that it will have greater order, greater transition width or greater passband/stopband ripples. Any of these is typically undesirable in practice, therefore more sophisticated algorithms come in handy.

Optimal designs are computed by minimizing some measure of the deviation between the filter to be designed and the ideal filter. The most common optimal FIR design algorithms are based on fixing the transition width and the order of the filter. The deviation from the ideal response is measured only by the passband/stopband ripples. This deviation or error can be expressed mathematically as [2]

$$E(\omega) = H_a(\omega) - H_{\mathrm{LP}}(e^{j\omega}), \quad \omega \in \Omega$$

where $H_a(\omega)$ is the zero-phase response of the designed filter and $\Omega = [0, \omega_p] \cup [\omega_s, \pi]$. It is still necessary to define a measure to determine "the size" of $E(\omega)$ - the quantity we want to minimize as a result of the optimization. The most often used measures are the $\mathcal{L}_\infty$-norm ($\|E(\omega)\|_\infty$

- minimax designs) and the $\mathcal{L}_2$-norm ($\|E(\omega)\|_2$ - least-squares designs).

In order to allow for different peak ripples in the passband and stopband, a weighting function, $W(\omega)$ is usually introduced,

$$E_W(\omega) = W(\omega)[H_a(\omega) - H_{\mathrm{LP}}(e^{j\omega})], \quad \omega \in \Omega$$

3.1 Linear-phase designs

A filter with linear-phase response is desirable in many applications, notably image processing and data transmission. One of the desirable characteristics of FIR filters is that they can be designed very easily to have linear phase. It is well known [3] that linear-phase FIR filters will have impulse responses that are either symmetric or antisymmetric. For these types of filters, the zero-phase response can be determined analytically [3], and the filter design problem becomes a well behaved mathematical approximation problem [4]: Determine the best approximation to a given function - the ideal lowpass filter's frequency response - by means of a polynomial - the FIR filter - of given order -the filter order -. By "best" it is meant the one which minimizes the difference between them - $E_W(\omega)$ - according to a given measure.

The `remez` function implements an algorithm developed in [5] that computes a solution to the design problem for linear-phase FIR filters in the $\mathcal{L}_\infty$-norm case. The design problem is essentially to find a filter that minimizes the *maximum* error between the ideal and actual filters. This type of design leads to so-called equiripple filters, i.e. filters in which the peak deviations from the ideal response are all equal.

The `firls` function implements an algorithm to compute solution for linear-phase FIR filters in the $\mathcal{L}_2$-norm case. The design problem is to find a filter that minimizes the energy of the error between ideal and actual filters.

3.1.1 Equiripple filters

Linear-phase equiripple filters are desirable because they have the smallest maximum deviation from the ideal filter when compared to all other linear-phase FIR filters of the same order. Equiripple filters are ideally suited for applications in which a specific tolerance must be met. For example, if it is necessary to design a filter with a given minimum stopband attenuation or a given maximum passband ripple.

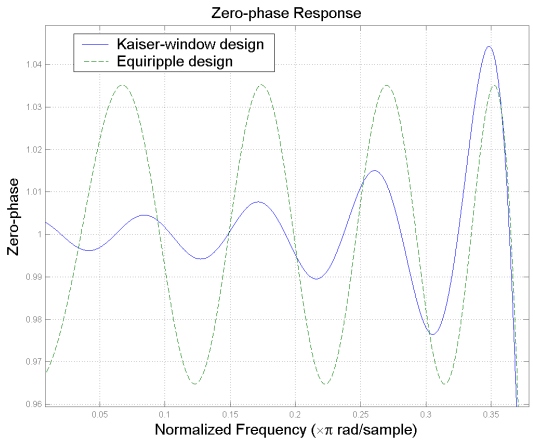

Figure 4: Passband ripple for of both the Kaiser-window-designed FIR filter and the remez-designed FIR filter.

For example the Kaiser-window design of Section 2.1 was of 42nd order. With this same order, an equiripple filter (with fixed transition width) can be designed that is superior to the Kaiser-window design:

```
br = remez(42,[0 0.37 0.43 1],[1 1 0 0]);
```

Figure 4 shows the superposition of the passband details for the filters designed with the Kaiser window and with the `remez` function. Clearly the maximum deviation is smaller for the `remez` design. In fact, since the filter is designed to minimize the maximum ripple (minimax design), we are guaranteed that no other linear-phase FIR filter of 42nd order will have a smaller peak ripple for the same transition width.

3.1.2 Least-squares filters

Equiripple designs may not be desirable if we want to minimize the energy of the error (between ideal and actual filter) in the passband/stopband. Consequently, if we want to reduce the energy of a signal as much as possible in a certain frequency band, least-squares designs are preferable.

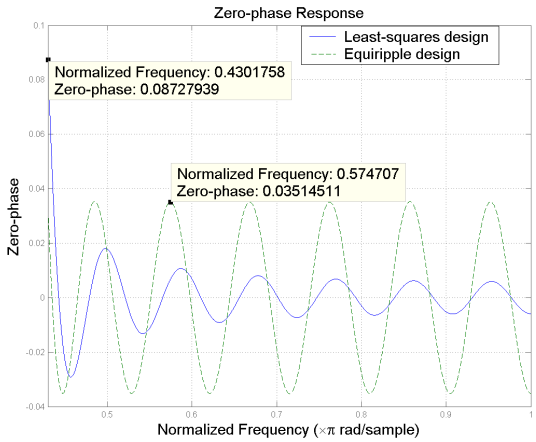

Figure 5: Comparison of an optimal equiripple FIR design and an optimal least-squares FIR design. The equiripple filter has a smaller peak error, but larger overall error.

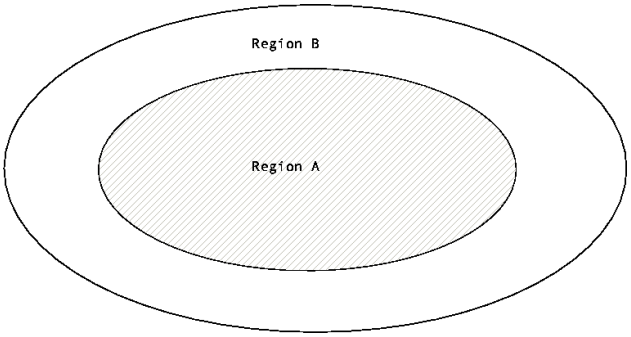

Figure 6: Depiction of the solution space for linear-phase and nonlinear-phase FIR filters for a given set of specifications. Region A represents the set of all linear-phase FIR filters that meet the specifications. Region B represents the set of all linear and nonlinear-phase FIR filters that meet the specifications.

For example, for the same transition width and filter order as the equiripple filter designed in Section 3.1.1, a least-squares FIR design can be computed from

```
bls = firls(42,[0 0.37 0.43 1],[1 1 0 0]);
```

The stopband energy for this case is given by

$$E_{\text{sb}} = \frac{1}{2\pi}\int_{0.43}^{\pi} |H_a(e^{j\omega})|^2 d\omega$$

where $H_a(e^{j\omega})$ is the frequency response of the filter.

In this case, the stopband energy for the equiripple filter is approximately 1.7608e-004 while the stopband energy for the least-squares filter is 3.3106e-005. (As a reference, the stopband energy for the Kaiser-window design for this order and transition width is 6.1646e-005).

So while the equiripple design has less peak error, it has more "total" error, measured in terms of its energy. The stopband details for both equiripple design and the least-squares design is shown in Figure 5.

3.2 Nonlinear-phase designs

One of the advantages of FIR filters, when compared to IIR filters, is the ability to attain exact linear phase in a straightforward manner. As we have already mentioned, the linear-phase characteristic implies a symmetry or antisymmetry property for the filter coefficients. Nevertheless, this symmetry of the coefficients constraints the possible designs that are attainable. This should be obvious since for a filter with $N+1$ coefficients, only $N/2+1$ of these coefficients are freely assignable (assuming N is even). The remaining $N/2$ coefficients are immediately determined by the linear-phase constraint.

One can think of this as reducing the search space for an optimal solution. The idea is depicted in Figure 6 [7]. Region A in the graph represents the set of all linear-phase FIR filters that meet a given set of specifications. This set contains both the optimal equiripple and the optimal least-squares filters we have mentioned so far. Region B represents the set of all FIR filters that meet a set of specifications, regardless of their phase characteristic. Clearly Region B contains Region A.

3.2.1 Minimum-phase designs

If one is able to relax the linear-phase constraint (i.e. if the application at hand does not require a linear-phase characteristic), it is possible to design minimum-phase equiripple filters that are superior to optimal equiripple linear-phase designs based on a technique described in [8].

For example, the following minimum-phase design has

both smaller peak passband ripple and smaller peak stopband ripple than the linear-phase equiripple design of Section 3.1.1:

```
bm = gremez(42,[0 0.37 0.43 1],...
          [1 1 0 0],[1 10],'minphase');
```

It is important to note that this is not a totally unconstrained design. The minimum-phase requirement restricts the resulting filter to have all its zeros on or inside the unit circle.[2]

3.2.2 More general nonlinear-phase designs

We have just stated that minimum-phase designs are not completely unconstrained due to the requirement on the loci of the zeros. A general nonlinear-phase design algorithm is provided in the `firlpnorm` function.

Consider the following specifications:

Specifications Set 2

1. Cutoff frequency: 0.375π rad/sample
2. Transition width: 0.15π rad/sample
3. Maximum passband ripple: 0.008
4. Maximum stopband ripple: 0.0009

A 30th order FIR equiripple filter (with nonlinear phase) can be designed to meet that set of specs with `firlpnorm`,

```
blp = firlpnorm(30,[0 .3 .45 1],[0 .3 ...
      .45 1],[1 1 0 0],[1 1 10 10]);
```

This contrasts with a 37th order filter if we require linear phase. By comparison, a minimum-phase equiripple filter designed using `gremez` as described above also requires a 30th order filter to meet the specifications which is quite remarkable considering the minimum-phase constraint.

The fact that two different nonlinear-phase filters of the same order meet the same specifications illustrates the difficulty associated with nonlinear-phase designs in general. There is no longer a unique optimal solution to a given design problem. Figure 7 shows the virtually identical magnitude responses. In contrast, Figure 8 shows the remarkably different impulse responses.

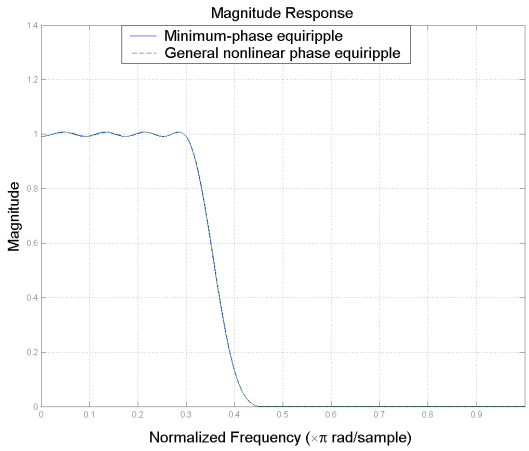

Figure 7: Magnitude responses of a minimum-phase equiripple 30th order filter and a general nonlinear-phase equiripple filter of the same order. Both filters are designed to meet the same specs.

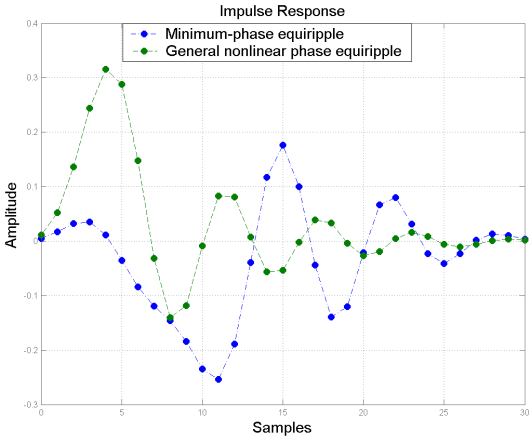

Figure 8: Impulse response comparison for an equiripple minimum-phase filter and a nonlinear-phase equiripple filter with virtually the same magnitude response.

[2] Given any linear-phase FIR filter with nonnegative zero-phase characteristic, it is possible to extract the minimum-phase spectral factor using the firminphase function.

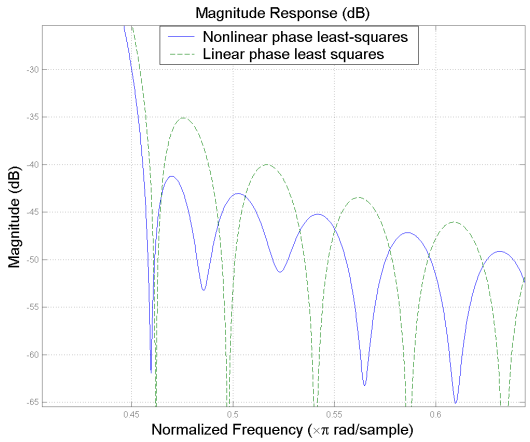

Figure 9: Stopband details of a nonlinear-phase least-squares filter and a linear-phase least-squares filter of the same order. The nonlinear-phase filter provides a smaller transition width and a larger stopband attenuation.

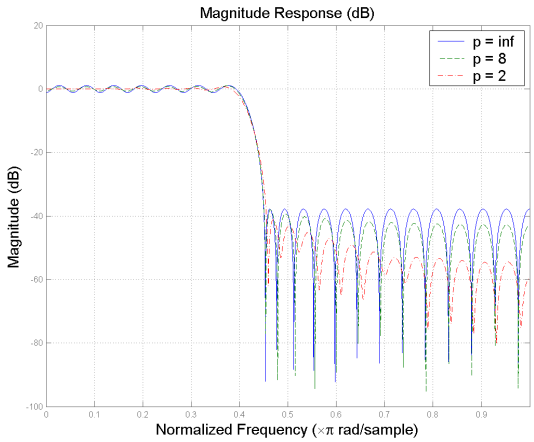

Figure 10: Optimal $\mathcal{L}_p$ norm designs for different values of p. All filters have the same order and transition width.

`firlpnorm` also provides the ability to select a different norm for the optimization. While the default optimization is for the $\mathcal{L}_\infty$ norm, any norm between (and including) $\mathcal{L}_2$ and $\mathcal{L}_\infty$ is possible.

By the arguments given above, it is possible to attain a superior design using `firlpnorm` instead of `firls` for the same filter order, provided linear phase is not a requirement. For example,

```
b = firlpnorm(40,[0 .4 .45 1],[0 .4 .45 1],...
    [1 1 0 0],[1 1 10 10],[2 2]);
b2 = firls(40,[0 .4 .45 1],[1 1 0 0],[1 20]);
```

yields a smaller transition width and a larger stopband attenuation for the nonlinear-phase case (with approximately the same peak passband ripple). The stopband details are shown in Figure 9.

Because it is possible to choose the $\mathcal{L}_p$ norm with which to optimize, `firlpnorm` is very flexible and allows for the designer to reach a compromise between equiripple and least-squares designs. This is illustrated in Figure 10.

3.2.3 A word on practical implementation

Because of the symmetry in the coefficients, some practical implementations will allow for a linear-phase response using roughly half the number of multipliers. This is particularly true with FPGAs and specialized hardware. The end result is that it may very well be possible to stick to a linear-phase design and achieve a more efficient implementation than comparable nonlinear-phase designs.

4 Optimal equiripple designs with fixed transition width and peak passband/stopband ripple

We have seen that the optimal equiripple designs outperform Kaiser-window designs for the same order and transition width. The differences are even more dramatic when the passband ripple and stopband ripple specifications are different. The reason is that the truncated-and-windowed impulse response methods always give a result with approximately the same passband and stopband peak ripple. Therefore, always the more stringent peak ripple constraint is satisfied, resulting in exceeding (possibly significantly) all other ripple constraints at the expense of

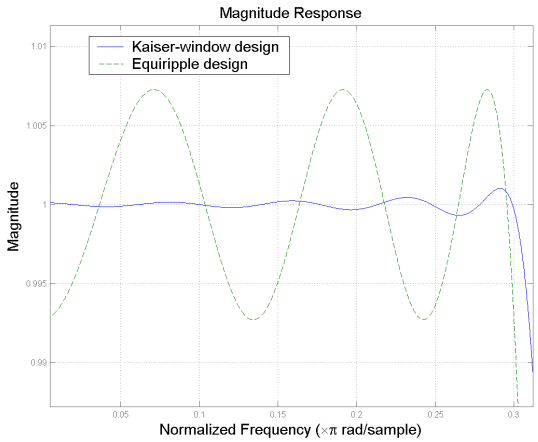

Figure 11: Passband ripple details for both the Kaiser-window-designed FIR filter and the remez-designed FIR filter. The Kaiser-window design over-satisfies the requirement at the expense of increase number of taps.

unnecessarily large filter order.

To illustrate this, we turn to a different equiripple design in which both the peak ripples and the transition width are fixed. Referring back to the triangle in Figure 2, this means the resulting filter order will come from the design algorithm.

Consider once again the Specifications Set 2. The `gremez` function can be used to design this filter

```
b = gremez('minorder',[0 .3 .45 1],...
    [1 1 0 0],[.008 .0009]);
```

resulting in a filter of 37th order (38 taps). By comparison, a Kaiser-window design requires a 50th order filter (51 taps) to meet the same specifications. The passband details can be seen in Figure 11. It is evident that the Kaiser-window design over-satisfies the requirements significantly.

4.1 Minimum-phase designs with fixed transition width and peak passband/stopband ripple

The same procedure to design minimum-phase filters with fixed filter order and fixed transition width can be used to design minimum-phase filters with fixed transition width and peak passband/stopband ripple. In this case, rather than obtaining smaller ripples, the benefit is meeting the same transition width and peak passband/stopband ripples with a reduced filter order.

As an example, consider the following specifications set:

Specifications Set 3

1. Cutoff frequency: 0.13π rad/sample
2. Transition width: 0.02π rad/sample
3. Maximum passband ripple: 0.01
4. Maximum stopband ripple: 0.001

The minimum order needed to meet such specifications with a linear-phase FIR filter is 262. This filter must be the result of an optimal equiripple design. If we relax the linear-phase constraint however, the `gremez` function can design a minimum-phase FIR filter that meets the specifications set with 216th order:

```
bgm = gremez('minorder',[0 .12 .14 1],...
      [1 1 0 0],[0.01 0.001],'minphase');
```

5 Optimal equiripple designs with fixed peak ripple and filter order

So far we have illustrated equiripple designs with fixed transition width and fixed order and designs with fixed transition width and fixed peak ripple values. The Filter Design Toolbox also provides algorithms for designs with fixed peak ripple values and fixed filter order [6]. This gives maximum flexibility in utilizing the degrees of freedom available to design an FIR filter.

We have seen that, when compared to Kaiser-window designs, fixing the transition width and filter order results in an optimal equiripple design with smaller peak ripple values, while fixing the transition width and peak ripple values results in a filter with less number of taps. Naturally, fixing the filter order and the peak ripple values should result in a smaller transition width.

To verify this, we use the `firceqrip` function,

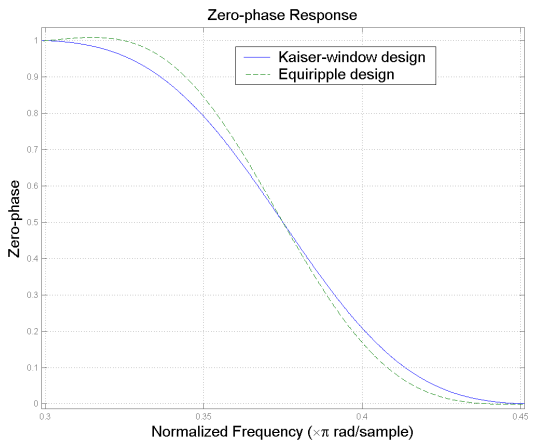

Figure 12: Comparison of a Kaiser-window-designed FIR filter and an optimal equiripple FIR filter of the same order and peak ripple values. The equiripple design results in a reduced transition-width.

```
bc = firceqrip(50,0.375,[0.008 0.0009]);
```

The comparison of this new design with the Kaiser-window design is shown in Figure 12. The transition width has been reduced from 0.15π to approximately 0.11π.

5.1 Minimum-phase designs with fixed peak ripple and filter order

Once again, if linear-phase is not a requirement, a minimum-phase filter can be designed that is a superior in some sense to a comparable linear-phase filter. In this case, for the same filter order and peak ripple value, a minimum-phase design results in a smaller transition width than a linear-phase design.

For example, compared to the 50th order linear-phase design `bc`, the following design has a noticeably smaller transition width:

```
bcm=firceqrip(50,0.375,[0.008 0.0009],'min');
```

6 Other equiripple designs

For specific design problems, further equiripple design options are available in the Filter Design Toolbox. Notably, the constrained-band design - where one can fix the filter order along with the peak ripple and the beggining/end of a given band (passband or stopband)- and the sloped stopband design, where the stopband is no longer equiripple, but rather has a predetermined slope.

6.1 Constrained-band equiripple designs

Sometimes when designing lowpass filters for decimation it is necessary to guarantee that the stopband of the filter begins at a specific frequency value and that the filter provide a given minimum stopband attenuation.

If the filter order is fixed - for instance when using specialized hardware - there are two alternatives available in the Filter Design Toolbox for optimal equiripple designs. One posibility is to fix the transition width, the other is to fix the passband ripple.

For example, the design Specifications Set 2 call for a stopband that extends from 0.45π to π and provide a minimum stopband attenuation of approximately 60 dB. For illustration purposes, suppose the filter order available is 40 (41 taps). The `firceqrip` function can design this filter if we also fix the passband ripple to 0.008. The result will be a filter with the smallest possible transition width for any linear-phase FIR filter of that order that meets the given specifications.

```
bc = firceqrip(40,0.45,[0.008 0.0009],...
     'stopedge');
```

If in contrast we want to fix the transition width, we can use the `gremez` function. The result in this case will be a filter with the smallest possible passband ripple for any linear-phase FIR filter of that order that meets the given specifications.

```
bg = gremez(40,[0 .3 .45 1],[1 1 0 0],...
     [1 0.0009],{'w','c'});
```

The passband details of the two filters are shown in Figure 13. Note that both filters meet the Specifications Set 2 because the order used (40) is larger than the minimum order required (37) by an equiripple linear-phase filter to

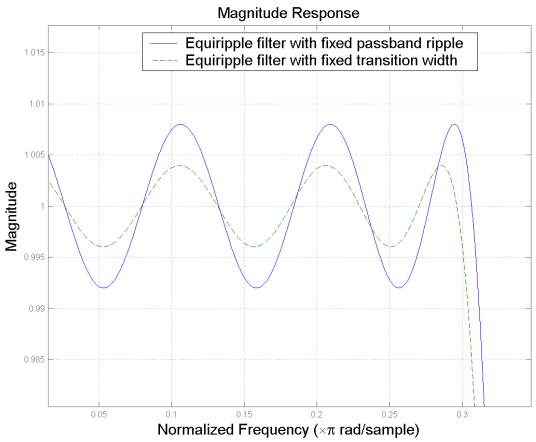

Figure 13: Comparison of two optimal equiripple FIR filters of 40th order. Both filters have the same stopband-edge frequency and minimum stopband attenuation. One is optimized to minimize the transition width while the other is optimized to minimize the passband ripple.

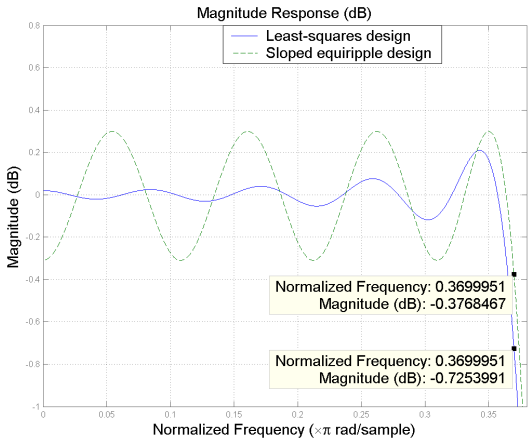

Figure 14: Passband details of a sloped optimal equiripple FIR design and an optimal least-squares FIR design. The equiripple filter has a smaller peak error or smaller transition width depending on the interpretation.

meet such specifications. The filters differ in how they "use" the extra number of taps to better approximate the ideal lowpass filter.

6.2 Sloped equiripple filters

An alternative to using least-squares designs is to design optimal equiripple filters but allowing for a slope in the stopband of the filter. This has the advantage (over least-squares designs) that the passband can remain equiripple, thus minimizing the input signal fluctuations in that region.

While one can achieve sloped stopbands using the `remez` or `gremez` methods by utilizing the weights, `firceqrip` provides the best control and easiest way to do this (at the expense of not having full control over the transition width). Using `firceqrip` one can specify the desired slope (in dB per frequency unit) for the stopband.

For example, the following design,

```
bf = firceqrip(42,0.4346,[0.035],[0.03],...
     'slope',40,'stopedge');
```

results in a stopband energy of approximately 3.9771e-005, not much larger that the least-squares design of Section 3.1.2, while having a smaller transition width (or peak passband ripple - depending on the interpretation). The passband details of both the least-squares design and the sloped equiripple design are shown in Figure 14 (in dB). The stopband details are shown in Figure 15 (also in dB).

7 Advanced design algorithms - interpolated FIR filters

For any given FIR design algorithm, if the peak ripple specifications remain the same, the filter order required to meet a given specifications set is inversely proportional to the transition width allowed.

When the transition width is small, such as in the Specifications Set 3, the filter order required may be quite large. This is one of the primary disadvantages of FIR filters. We have already seen that relaxing the linear-phase requirement results in a significant savings in the number of filter coefficients.

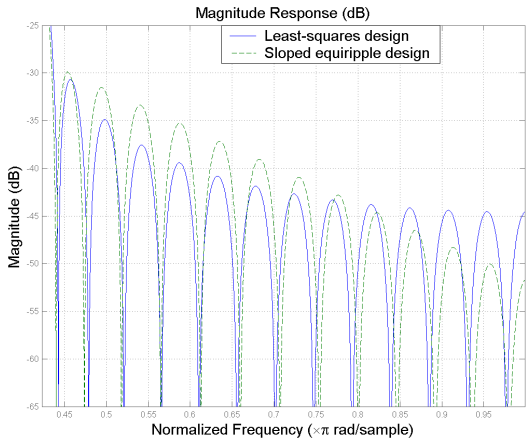

Figure 15: Stopband details of a sloped optimal equiripple FIR design and an optimal least-squares FIR design. The overall error of the equiripple filter approaches that of the least-squares design.

The so-called *interpolated FIR* (IFIR) approach [9],[10],[11] yields linear-phase FIR filters that can meet the given specifications with a reduced number of multipliers.

The idea is rather simple. Since the length of the filter grows as the transition width shrinks, we don't design a filter for a given (small) transition width. Rather, we design a filter for a multiple L of the transition width. This filter will have a significantly smaller length than a direct design for the original (small) transition width. Then, we *upsample* the impulse response by a factor equal to the multiple of the transition width, L. Upsampling will cause the designed filter to compress, meeting the original specifications without introducing extra multipliers (it only introduces zeros, resulting in a larger delay). The price to pay is the appearance of spectral replicas of the desired filter response within the Nyquist interval. These replicas must be removed by a second filter (called in this context the interpolation filter or image suppressor filter) that is cascaded with the original to obtain the desired overall response. Although this extra filter introduces additional multipliers, it is possible in many cases to still have overall computational savings relative to conventional designs. The implementation is shown in Figure 16.

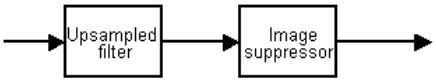

Figure 16: The IFIR implementation. An upsampled filter is cascaded with an image suppressor filter to attain an overall design with a reduced computational cost.

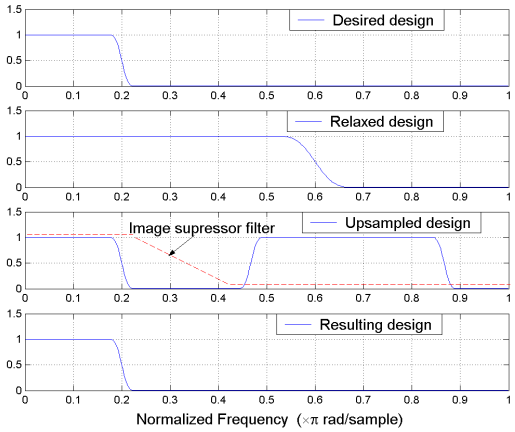

Figure 17: Illustration of the IFIR design paradigm. Two filters are used to attain stringent transition width specifications with reduced total multiplier count when compared to a single filter design.

The idea is depicted by example in Figure 17 for the case of an upsampling factor of 3. The "relaxed" design is approximately of one third the length of the desired design, if the latter were to be designed directly. The upsampled design has the same transition width as the desired design. All that is left is to remove the spectral replica introduced by upsampling. This is the job of the image suppressor filter.

As an example of the computational cost savings, consider once again the design Specifications Set 3. The number of multipliers required for a single linear-phase design was 263. An IFIR design can attain the same specs with 127 multipliers when using an upsampling factor of 6:

```
[bup,bimg]=ifir(6,'low',[.12 .14],[.01 .001]);
```

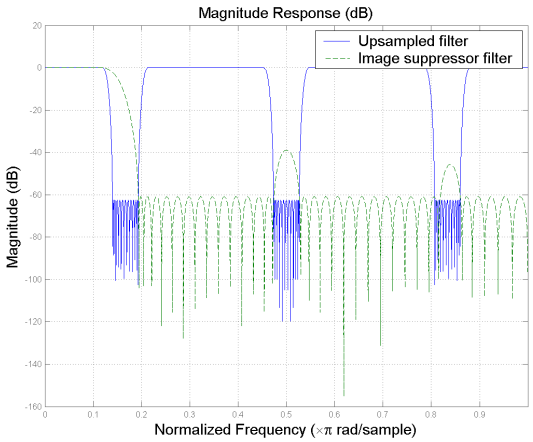

Figure 18: Magnitude response of the upsampled filter and the image suppressor filter in an IFIR design.

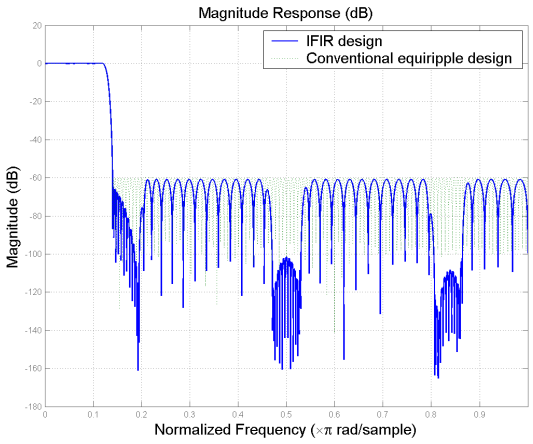

Figure 19: Overall magnitude response of an IFIR design and a conventional equiripple design. The IFIR implementation requires 127 multipliers vs. 263 for the conventional implementation.

The response of the upsampled filter and the image suppressor filter is shown in Figure 18. The overall response, compared to a single linear-phase equiripple design is shown in Figure 19.

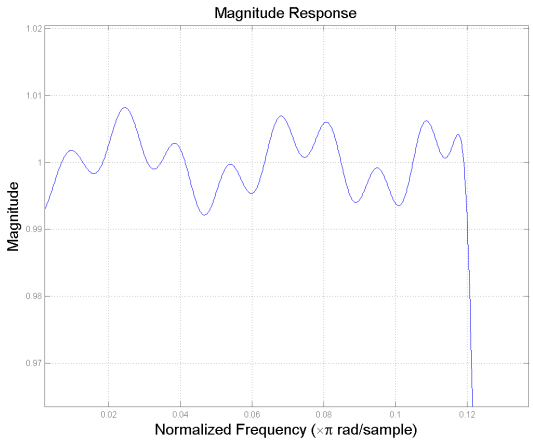

Figure 20: Passband details of an IFIR design revealing a rather chaotic behavior of the ripple.

7.1 Further IFIR optimizations

A drawback in the IFIR design is that the passband ripples of the two filters are combined in a disorderly fashion. In the worst case scenario, they can add up, requiring the design to ensure that the sum of the two peak passband ripples does not exceed the original set of specifications. Close inspection of the passband of the overall design in the previous example, shown in Figure 20, reveals a rather chaotic behavior (but certainly within spec.) of the ripple.

Further optimized designs, [2], [12], attain a much cleaner passband behavior by jointly optimizing the design of the two filters to work better together. This results in a filter that can meet the specifications set with an even further reduction in the number of multipliers. The savings are especially significant for the image suppressor filter, which is greatly simplified by this joint optimization.

Utilizing this joint optimization, the Specifications Set 3 can be met with only 74 multipliers, once again for an upsampling factor of 6. The filter can be designed using the `'adv'` flag in the `ifir` function.

The manner in which the two filters work together is best described by looking at their magnitude responses, shown in Figure 21. By pre-compensating for a severe

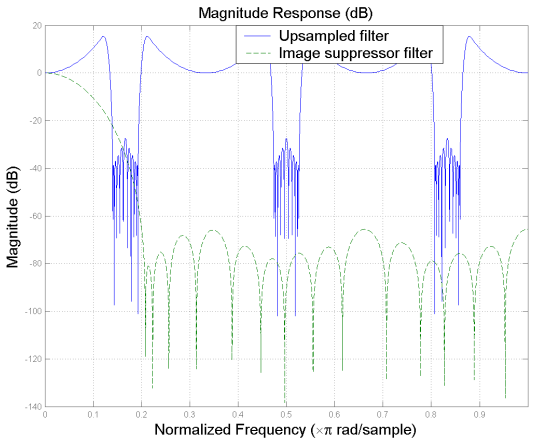

Figure 21: Magnitude response of the upsampled filter and the image suppressor filter in an optimized IFIR design. The two filters are jointly optimized in the design to achieve a specifications set with a reduced number of multipliers.

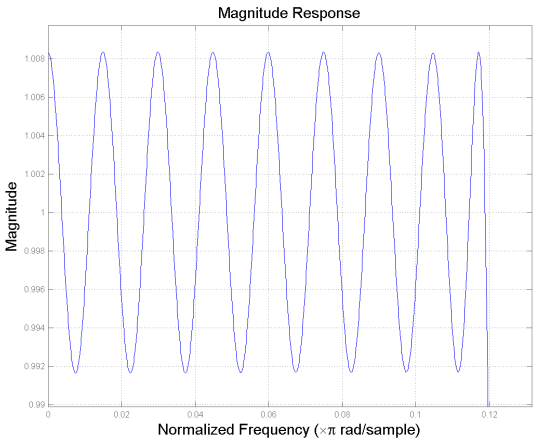

Figure 22: Passband details of an optimized IFIR design. The optimized design exhibits nice equiripple behavior.

"droop" in the image suppressor filter, a flat passband can be achieved with dramatic savings in the number of multipliers required for the image suppressor filter. Out of the 74 multipliers required, 29 are for the image suppressor filter and 45 for the upsampled filter. By contrast, in the previous IFIR design, 78 of the 127 multipliers correspond to the image suppressor filter, while 49 correspond to the upsampled filter.

The passband details of the overall design show a nice equiripple behavior, hinting at a much better optimized design. The passband details are shown in Figure 22.

7.2 Multirate implementation of IFIR design

When designing an IFIR filter, the upsampling factor L used must be such that the (normalized) stopband-edge frequency ω_s satisfies $L\omega_s < \pi$. This implies that the bandwidth of the output signal would be reduced by a factor of L.

It is convenient from a computational cost perspective to reduce the sampling frequency of the filtered signal, since at that point the Nyquist criterion is being unnecessarily oversatisfied. Subsequent processing of the filtered signal without reducing its sampling rate would incur in unnecessary (and expensive) redundant processing of information.

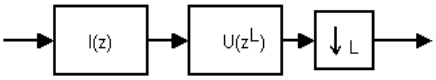

Figure 23: Cascading an IFIR implementation with a downsampler.

The idea is to downsample the filtered signal by a factor of L to match the reduction in bandwidth due to filtering. If we denote by $I(z)$ the image suppressor filter and by $U(z^L)$ the upsampled filter, we would have a cascade of these two filters and a downsampler as shown in Figure 23. Using the Noble identities, we can "commute" the downsampler and $U(z^L)$ to obtain the implementation shown in Figure 24. The combination of $I(z)$ and the downsampler form a decimator which can be implemented efficiently in polyphase form.

8 Interpolation filter design

In the context of multirate signal processing, interpolation usually refers to band-limited interpolation. Band-limited

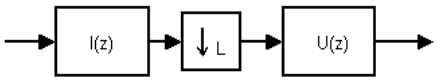

Figure 24: Interchange of the downsampler and the upsampled filter using the Noble identities.

interpolation is based on the notion of an underlying band-limited continuous-time signal that is being sampled.

Ideal band-limited interpolation will take a digital (sampled) signal and produced an interpolated signal that will be identical to the signal that would be obtained by sampling the underlying continuous-time signal at a higher rate.

Ideal band-limited interpolation can be accomplished by means of upsampling and using an ideal lowpass filter. Especially interesting is a time-domain interpretation of the ideal interpolator, which leads naturally to polyphase implementations.

8.1 Ideal band-limited interpolation in the frequency domain

As we have already mentioned, the key concept of bandlimited-interpolation is that a signal to be interpolated is a sampled version of a band-limited continuous time signal.

Denote the continuous-time signal by $x_c(t)$ and suppose its spectrum is zero for all $|f| > f_{\max}$. Its frequency spectrum $X(2\pi jf)$ is shown in Figure 25.

If the signal is sampled at $f_s = 2f_{\max}$, we obtain the signal

$$x_T[n] = \{x_c(nT)\}, \quad T = \frac{1}{f_s}.$$

Its spectrum, $X_T(e^{2\pi jf/f_s})$ is shown in Figure 26.

Now suppose the continuous-time signal was sampled at a rate $f_s' = Lf_s = 2Lf_{\max}$. The sampled signal at the higher rate,

$$x_{T'}[m] = \{x_c(mT')\}, \quad T' = \frac{1}{f_s'} = \frac{T}{L},$$

where $m = Ln + k$, $k = 0, \ldots, L-1$, will have a spectrum $X_{T'}(e^{2\pi jf/f_s'})$ as shown in Figure 27 for the case $L = 2$.

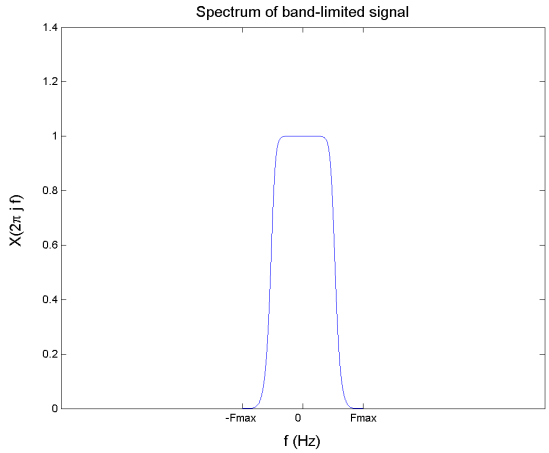

Figure 25: Spectrum of band-limited continuous-time signal.

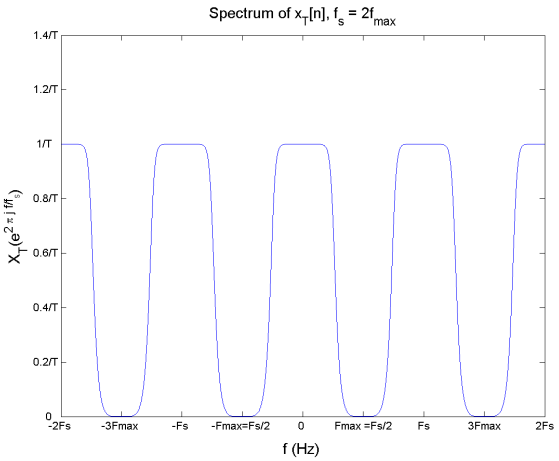

Figure 26: Spectrum of sampled signal with $f_s = 2f_{\max}$.

The job of the ideal interpolation filter should now be clear from the frequency domain standpoint. Take the discrete-time signal with spectrum $X_T(e^{2\pi jf/f_s})$ and *digitally* produce the discrete-time signal $X_{T'}(e^{2\pi jf/f_s'})$ that would have been obtained from sampling the original continuous-time signal at rate $f_s' = Lf_s$.

The response of the ideal interpolation filter is shown in Figure 28. Clearly it is a lowpass filter with periodicity

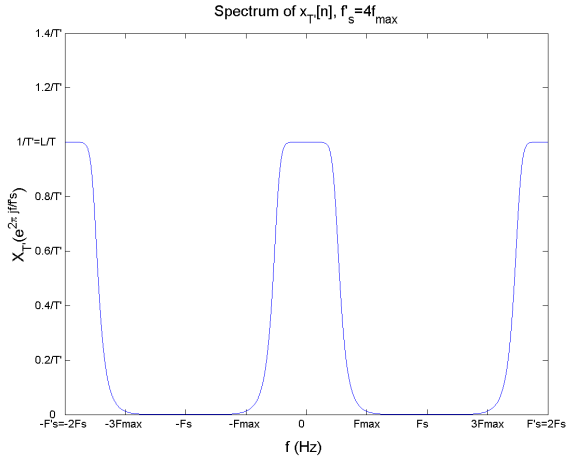

Figure 27: Spectrum of sampled signal with $f_s' = 4f_{\max}$.

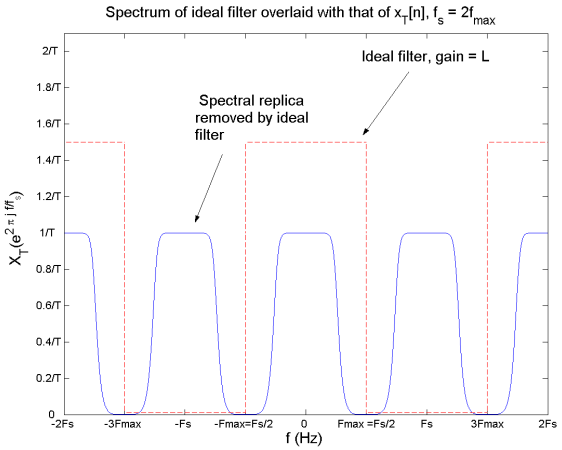

Figure 28: Ideal interpolation filter overlaid with spectrum of sampled signal with $f_s = 2f_{\max}$.

f_s', i.e. it must be operating at the high sampling rate. For this reason, it is necessary to upsample the input signal[3] by inserting an appropriate amount of zeros between samples in order to feed the interpolation filter a signal at the correct rate.

More precisely, the response of the ideal filter $H_D(e^{2\pi jf/f_s'})$ for the general case of interpolation by a factor of L is given by

$$H_D(e^{2\pi jf/f_s'}) = \begin{cases} L, & |f| \le \frac{f_s}{2} \\ 0, & \frac{f_s}{2} < |f| \le \frac{f_s'}{2} \end{cases} \tag{3}$$

The impulse response of the ideal interpolation filter can be found from the inverse DTFT [1],

$$h_D[m] = \frac{L}{f_s'} \frac{\sin(\pi f_s T' m)}{\pi T' m} \tag{4}$$

If we use the fact that $f_s' = 1/T'$ and $T' = \frac{1}{Lf_s}$ we have

$$h_D[m] = \frac{\sin(\pi m/L)}{\pi m/L}, \quad -\infty < m < \infty \tag{5}$$

As expected for an ideal lowpass filter, it takes an infinite impulse response to realize it. Further insight for the ideal interpolation filter will be given in Section 8.2 where we analyze things in the time-domain.

[3] Although this is not necessary in practice where efficient algorithms are used.

8.2 Ideal band-limited interpolation in the time domain

Once again, the key idea of ideal band-limited interpolation is to *digitally* produce a signal that would be exactly the same as a signal we had obtained by sampling a band-limited continuous time signal at the higher sampling rate.

The situation in the time domain is depicted in Figure 29.

Assuming the Nyquist sampling criterion has been satisfied, i.e. the continuous-time signal is band-limited and has been sampled at a rate $f_s = \frac{1}{T} \ge 2f_{\max}$, no information has been lost from the continuous-time signal $x_c(t)$. Therefore it should be possible to somehow recreate *any* instantaneous value $x_c(t_0)$ of the continuous-time signal from the sampled signal $x_T[n]$.

Looking at Figure 29, we can see that the job of the 5-fold interpolator is to take every input sample $x_T[n]$ and produce 5 output samples $\{x_{T'}[m]\}$, $m = 5n+k$, $k = 0,\ldots,4$ as follows (note that $T = 0.5$ and $T' = T/5 = 0.1$):

- $x_{T'}[5n] = x_T[n]$
- $x_{T'}[5n+1] = x_T[n+\frac{1}{5}]$
- $x_{T'}[5n+2] = x_T[n+\frac{2}{5}]$

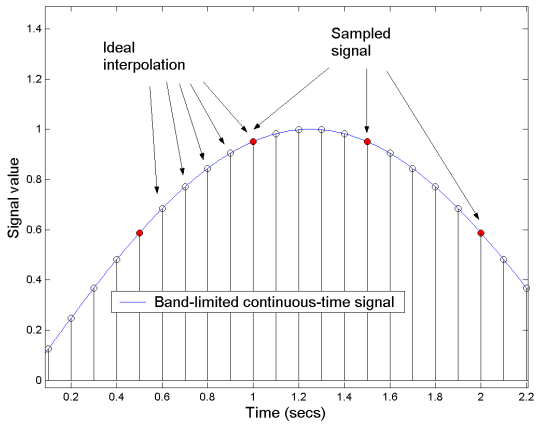

Figure 29: Illustration of ideal band-limited interpolation in the time domain.

- $x_{T'}[5n+3] = x_T[n+\frac{3}{5}]$
- $x_{T'}[5n+4] = x_T[n+\frac{4}{5}]$

In general, the ideal interpolator consists of a bank of L filters which will *fractionally advance* the input signal by a factor $k/L, k = 0,\ldots,L-1$. The outputs of the filters are then interleaved (i.e. only one filter needs to operate per high rate output sample) to produce the interpolated signal.

The L filters that comprise the filter bank are the fractional advance filters $H_k(z)$,

$$H_k(z) = z^{k/L}, \quad k = 0,\ldots,L-1.$$

Evaluating on the unit circle, we have

$$H_k(e^{j\omega}) = e^{j\omega k/L}, \quad k = 0,\ldots,L-1$$

so that each filter $H_k(e^{j\omega})$ is allpass, i.e. $|H_k(e^{j\omega})| = 1$ and has linear phase, $\arg\{H_k(e^{j\omega})\} = \omega k/L$.

Herein lies the impossibility of designing these filters. We cannot design them as FIR filters because no FIR filter can be allpass (except for a pure delay). We cannot design them as IIR filters, because no stable IIR filter can have linear phase. However, it is clear how we want to approximate the ideal interpolation filter bank.

FIR approximations can produce the exact linear phase, while approximating an allpass response as best possible. On the other hand, IIR approximations will be exactly allpass, while trying to produce the required phase.

It is insightful to realize that the filters comprising the filter bank are the polyphase components of the ideal interpolation filter derived in (5)! Thus this view of the ideal interpolator has the efficient polyphase structure "built-in".

Indeed, the impulse response of each fractional advance filter in the filter bank is given by the inverse DTFT,

$$\begin{aligned} h_k[n] &= \frac{1}{2\pi}\int_{-\pi}^{\pi} e^{j\omega k/L} e^{j\omega n} d\omega \\ &= \frac{\sin\left(\pi\frac{Ln+k}{L}\right)}{\pi\frac{Ln+k}{L}} \end{aligned}$$

which corresponds to the L decimated sequences of the ideal impulse response by again writing uniquely $m = Ln+k$, $k = 0,\ldots,L-1$ in (5).

8.3 Design of FIR interpolation filters

While interpolation filters are simply lowpass filters that can be designed with the various techniques outlined previously, the polyphase filters that compose the ideal interpolation filter give some insight on things to be looking for when designing interpolation filters.

Consider an interpolation by a factor of L. The ideal L polyphase filters will have a group-delay given by

$$-\tfrac{k}{L}, \quad k = 0,\ldots,L-1$$

For simplicity, consider an FIR approximation to the ideal interpolation filter where the order is of the form $N = 2LM$. Then each polyphase filter will have order $N/L = 2M$.

Note that the ideal interpolation filter is infinitely non-causal. After finite length truncation, it is possible to make the approximation causal by delaying by half the filter order, $N/2$. However, because we will implement in efficient polyphase form, we can make each polyphase component causal by delaying it by M samples.

The delay will mean the introduction of a phase component in the response of each polyphase component. So that instead of approximating the ideal fractional advance $e^{j\omega k/L}$ the polyphase components will approximate

$e^{j\omega(k/L-M)}$. The group-delay will consequently be of the form

$$-\frac{d\phi(\omega)}{d\omega} = -\frac{d\omega(k/L-M)}{d\omega} = M - k/L.$$

A problem that arises is that even though the FIR approximation to the ideal interpolation filter is symmetric and thus has linear phase, the polyphase components are not necessarily symmetric and thus will not necessarily have exact linear phase. However, for each non symmetric polyphase filter, there is a mirror image polyphase filter which will have the exact same magnitude response with a mirror image group-delay that will compensate any phase distortion.

8.3.1 Nyquist FIR filters

When we analyzed the behavior of the ideal interpolation filter in the time-domain, we saw that for every input sample, L samples are produced including one that is exactly the same as the input sample. This exact copy is "produced" by the polyphase filter that has allpass magnitude and zero phase (i.e. the case k=0). In practice, this is the only polyphase filter that can be designed exactly, albeit with a group-delay of M rather than zero.

Roughly speaking, a Nyquist filter is one for which one of its polyphase components is a pure delay and thus leaves the input signal unchanged (except for a possible delay). When designing an interpolation filter, it is desirable for it to be a Nyquist filter since this will ensure that even a nonideal filter will allow the input samples to pass through unchanged. It can also be computationally advantageous since one of the polyphase subfilters will have no multipliers.

8.3.2 Halfband filters

Nyquist filters are also called Lth-band filters because the passband of their magnitude response occupies roughly $1/L$ of the Nyquist interval. In the special case of an interpolation by a factor of 2, the filters are known as *halfband* filters. Halfband filters are commonly used when interpolating (or decimating) by a factor of 2.

The cutoff frequency for a halfband filter is always 0.5π. Moreover, the passband and stopband ripples are identical, limiting the degrees of freedom in the design. The function `firhalfband` designs FIR halfband filters. The specifications set still follows the triangle metaphor shown in Figure 2, taking into account the limitations just described.

The following three function calls design three equiripple linear-phase halfband filters using a different pair of specifications in each case from the three available -order (N), transition-width (TW), and peak passband/stopband ripple (R)- :

```
b1=firhalfband(102,.47);            % N and TW
b2=firhalfband(102,.01,'dev');      % N and R
b3=firhalfband('minorder',.47,.01);% TW and R
```

To analyze how the design compares to the ideal interpolation filter, we can create an FIR interpolator object and look at its polyphase subfilters, for example if we use the third filter, `b3`,

```
h = mfilt.firinterp(2,2*b3);
polyphase(h)
```

The magnitude and group-delay responses for the polyphase components of this filter are shown in Figures 30 and 31. Note that $M = N/2L$ is 16.5 in this case, so that the group-delays are exactly $M - k/L$, $k = 0, 1$. The only deviation from an ideal filter (ignoring an overall delay of M samples) comes from the fact that one of the polyphase subfilters is not perfectly allpass.

8.3.3 Other Nyquist filters

Nyquist filters are characterized in the time-domain by their impulse response being exactly equal to zero every L samples (except the exact middle sample of the impulse response). This is precisely why we get a polyphase subfilter that is a perfect allpass delay and allows the samples to be interpolated to pass through the filter unchanged.

Designing a filter that is both a lowpass and simultaneously satisfies the just mentioned time-domain characteristic is not a trivial task except for the case of window-based designs, [13], [14].

Nevertheless, the advantage of conventional optimal equiripple designs over a Nyquist window-based design is not as clear in this case as it is with any conventional lowpass filter. We illustrate by example: consider a Kaiser window Nyquist filter design with a stopband attenuation

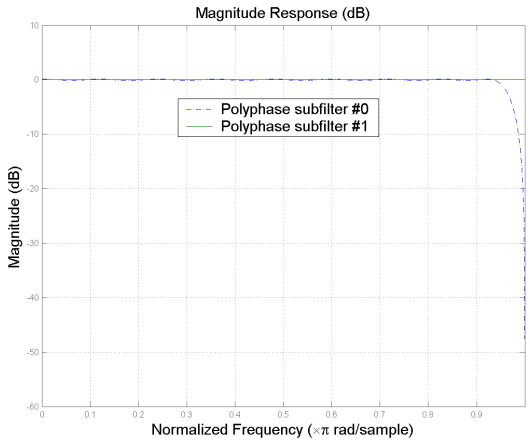

Figure 30: Magnitude response for polyphase subfilters of a halfband FIR filter. Ideally, both subfilters would be perfectly allpass.

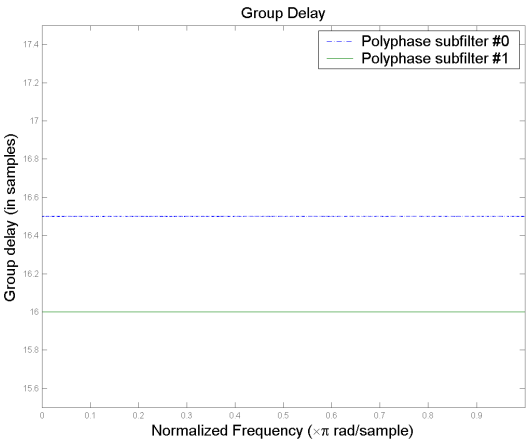

Figure 31: Group-delay response for polyphase subfilters of a halfband FIR filter. If the 16.5 samples delay -introduced for causality reasons- is ignored, the group-delay behaves exactly as the ideal interpolation filter, with an offset of $1/L$ between the group-delay of each subfilter ($L = 2$ in this example).

of 40 dB. Nyquist filters are often designed in terms of their roll-off factor, ρ, due to their applications in communications.[4] The roll-off factor is related to transition-width simply by $\mathrm{TW} = \rho\pi/L$. In this example, $\rho = 0.1$ and $L = 4$ thus the transition-width is 0.025π.

```
b1 = firnyquist('minorder',4,.1,.01); % L=4
```

The resulting filter is of 90th order. If we design an equiripple filter of the same order and same attenuation, we obtain a filter with a smaller transition width, but that does not satisfy the time-domain requirement.

```
b2 = firceqrip(90,.25,[.01 .01]);
```

The magnitude responses of the polyphase subfilters for the Nyquist window-based design are shown in Figure 32. For comparison, the magnitude responses for the optimal equiripple design are shown in Figure 33. Note the better approximation to allpass filters in the Nyquist design compared to the equiripple design (albeit for a slightly smaller interval - this is the tradeoff).

Similarly, if we compare the group-delay response of the polyphase subfilters, the Nyquist design once again better approximates the ideal constant group-delay as compared to the equiripple design. The group-delay responses for the polyphase subfilters of the Nyquist design are shown in Figure 34. The group-delay responses for the polyphase subfilters of the equiripple design are shown in Figure 35.

9 Design of perfect-reconstruction two-channel FIR filter banks

A two-channel filter bank is shown in Figure 36. Filters $H_0(z)$ and $H_1(z)$ are called the analysis filters while $G_0(z)$ and $G_1(z)$ are the synthesis filters.

The filter bank is called perfect reconstruction if the end-to-end system acts as a delay, i.e. if the output signal is simply a delayed (and possibly scaled) version of the input.

[4] The well-known raised-cosine filter is a special case of a Nyquist filter. In fact, the same reason that raised-cosine filters are common, i.e. to achieve zero intersymbol-interference with a non ideal filter, is why they are able to interpolate without affecting the input samples - namely the fact that the impulse response becomes zero exactly at the right time.

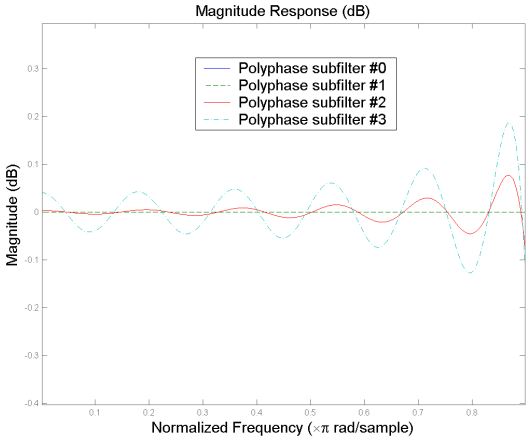

Figure 32: Magnitude response for polyphase subfilters of a Nyquist FIR filter designed with the window method. The polyphase subfilters better approximate allpass filters than a comparable equiripple design for the bulk of the frequency band.

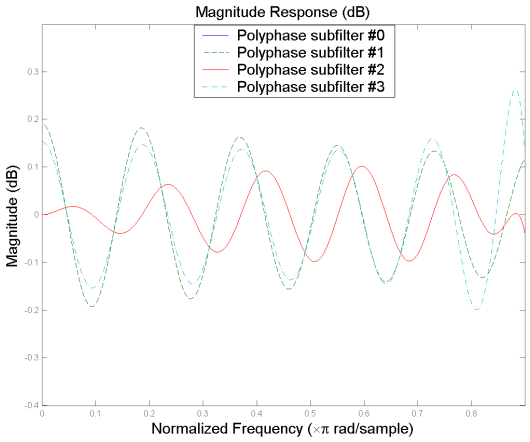

Figure 33: Magnitude response for polyphase subfilters of a optimal equiripple lowpass FIR filter. None of the subfilters behaves as a perfect allpass, an indication that this is not a Nyquist filter.

It is well-known, [10], [15], [16], that perfect reconstruction can be achieved if

$$\tfrac{1}{2}G_0(z)H_0(-z)+\tfrac{1}{2}G_1(z)H_1(-z)=0$$

and

$$\tfrac{1}{2}G_0(z)H_0(z)+\tfrac{1}{2}G_1(z)H_1(z)=z^{-k}.$$

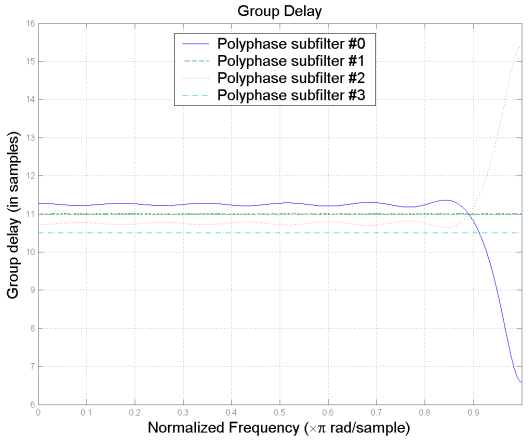

Figure 34: Group-delay response for polyphase subfilters of a Nyquist FIR filter of order 90 and $L=4$.

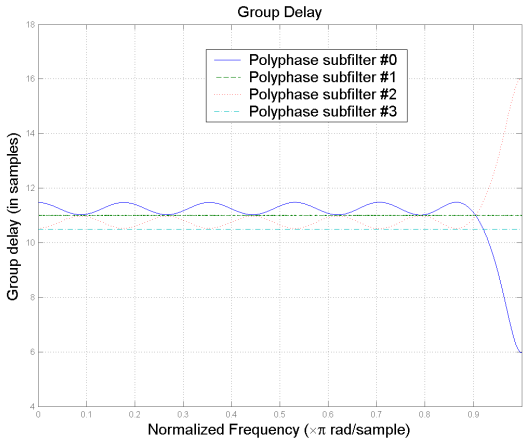

Figure 35: Group-delay response for polyphase subfilters of a conventional equiripple lowpass design that could be used for interpolation with $L=4$.

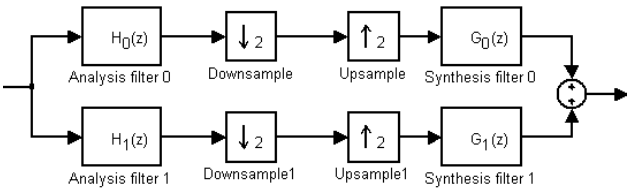

Figure 36: Two-channel subband coding filter bank.

Starting with a prototype lowpass filter $H(z)$ of odd order N, the following selection for the filters results in perfect reconstruction using *solely* FIR filters [15],

$$H_0(z) = H(z) \qquad H_1(z) = z^{-N}H_0(-z^{-1})$$
$$G_0(z) = 2z^{-N}H_0(z^{-1}) \qquad G_1(z) = 2z^{-N}H_1(z^{-1})$$

This type of perfect reconstruction filter bank is called an *orthogonal* filter bank or a *power-symmetric* filter bank [15].

The function `firpr2chfb` designs equiripple FIR filters $H_0(z), H_1(z), G_0(z), G_1(z)$ such that the filter bank is an orthogonal perfect reconstruction filter bank. The parameters to specify are simply the filter order N and the passband-edge frequency ω_p. A prototype lowpass filter is designed from which the four required filters are obtained. For example,

```
[h0,h1,g0,g1] = firpr2chfb(99,.45);
```

Alternatively, the peak stopband ripple can be specified. As usual, we can obtain minimum-order designs by specifying both the passband-edge frequency and the peak stopband ripple. In all cases, the design specifications apply to the prototype filter $H(z)$,

```
[h0,h1,g0,g1]=firpr2chfb(99,1e-3,'dev');
[h0,h1,g0,g1]=firpr2chfb('minorder',.45,1e-3);
```

The power-symmetric term is used because for these designs, the following holds:

$$|H_0(e^{j\omega})|^2 + |H_1(e^{j\omega})|^2 = 1, \forall\omega.$$

We can look at the magnitude-squared responses of $H_0(z), H_1(z)$ using `fvtool`. The magnitude-squared responses are shown in Figure 37 for $N = 19$ and $\omega_p = .45\pi$.

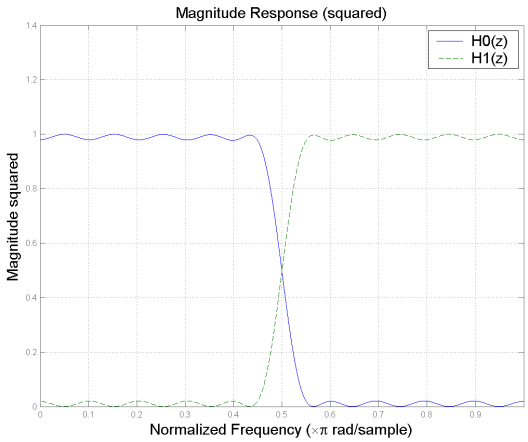

Figure 37: Magnitude-squared responses of the analysis filters in an FIR perfect reconstruction filter bank. The two filters are power-complementary.

Notice how where one filter's ripple rises the other filter's ripple declines to add up to one.

Increasing the filter order (and possibly the passband-edge frequency) improves the lowpass/highpass separation provided by the analysis filters but doesn't have an effect on the perfect reconstruction characteristic of the overall system.

10 Implementing an FIR filter using fixed-point arithmetic

Several factors have to be taken into account when implementing an FIR filter using fixed-point arithmetic. For one thing, the coefficients have to be quantized from double-precision floating point in which they are designed into fixed-point representation with usually a smaller number of bits. We must make sure we make the most of the limited number of bits we have. Furthermore, performing the arithmetic in fixed-point will introduce further quantization errors when actually filtering with the quantized coefficients. Once again, we must make sure we minimize these quantization errors as much as the hardware at hand allows us.

10.1 Some notation

First we will like to introduce the notation used in the Filter Design Toolbox to represent fixed-point numbers. Consider a register used to store a fixed-point number,

$$\underbrace{\square\square\square\ldots\ \square}_{B-\text{bits}}\quad \begin{matrix} b_0\, b_1\, b_2 & & b_{B-1} \end{matrix}$$

The register has B bits (it has a wordlength of B), the value of the kth bit is given by b_k which can obviously be only 0 or 1. A two's complement fixed-point number stored in such a register has a value given by

$$\text{value} = -b_0 2^{B-L-1} + \sum_{k=1}^{B-1} b_k 2^{B-L-k-1} \qquad (6)$$

where L is a positive or negative integer to be described now.

From (6), we can see that the value of a fixed-point number is determined by assigning weights of the form 2^{-m} to each bit. The leftmost bit, b_0 has the largest weight, 2^{B-L-1}, this bit is called the most-significant bit (MSB). The rightmost bit, b_{B-1}, has the smallest weight, 2^{-L}, which is why it is called the least-significant bit (LSB).

Given the bit values, b_k, the pair $\{B,L\}$ completely characterizes a fixed-point number, i.e. suffices in determining the value that the bits represent. We call such a pair the format of a given quantity, and store it in a two-element vector, `[B,L]`.

10.2 Quantizing the coefficients

Consider the following filter

```
b=gremez('minorder',[0 .11 .14 1],...
     [1 1 0 0],[.01 .0001]);
```

The filter has an attenuation of 80 dB and a its largest coefficient is 0.1206.

The first thing to do is check if there are enough bits available to represent the coefficients and provide the required dynamic range. A good rule of thumb [17] is to assume 5dB/bit for the dynamic range[5]. In this example we need at least 16 bits in order to provide the 80 dB of attenuation.

[5]Note that the usual 6db/bit rule doesn't apply because quantization error for the filter coefficients tends to be correlated, especially at the extremes of the impulse response.

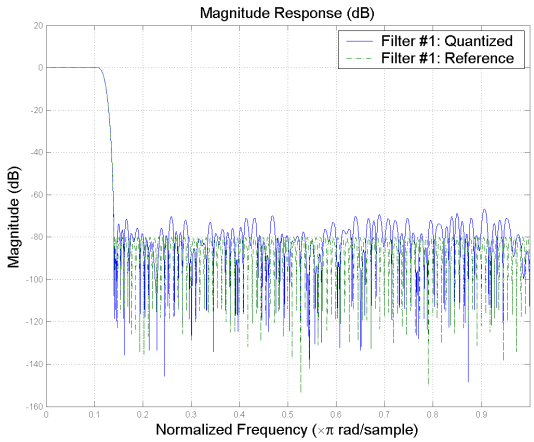

Figure 38: Magnitude response of the filter quantized with [16,15] format.

It is not sufficient to simply say we are going to use 16 bits. For example, the following code creates a fixed-point FIR filter using 16 bits to represent the coefficients in fractional format:

```
Hq=qfilt('fir',{b},...
   'CoefficientFormat',[16,15]);
```

The magnitude response of the quantized filter is shown in Figure 38. For comparison purposes, the nonquantized magnitude response is also shown. Note that the stopband attenuation for the quantized response is significantly less than 80 dB at various frequency bands. The problem is the poor utilization of the available range for the `[16,15]` format as shown in Figure 39.

To make the most of the 16 bits, there are two equivalent approaches we can take. If we want to use `[16,15]` format, we can scale the coefficients by multiplying them by a factor of 8 to make the largest coefficients as close to 1 as possible without overflowing.

Alternatively, we can use `[16,18]` format so that the quantization range becomes $[0.125, 0.125)$. The magnitude response using this format is shown in Figure 40. The improvement over the first case is evident.

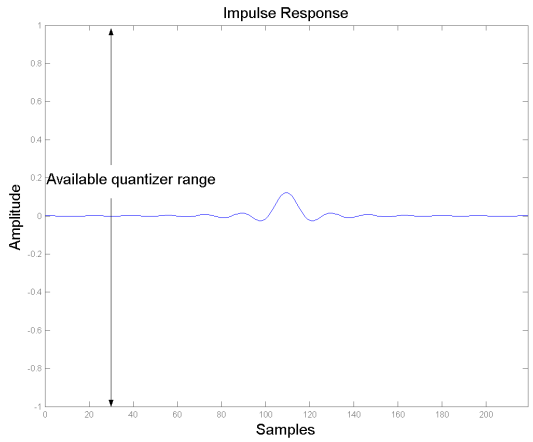

Figure 39: Impulse response of filter to be quantized shown relative to the available range for the coefficient format selected.

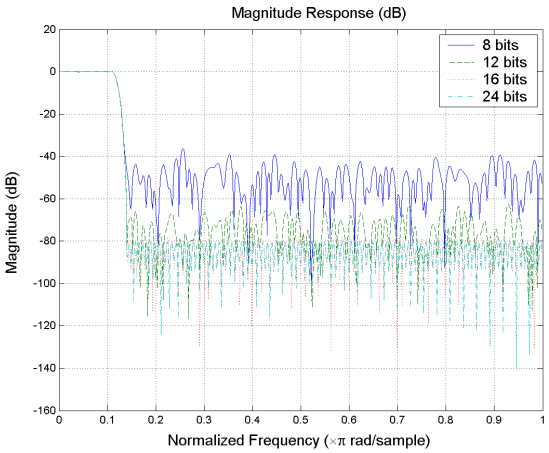

Figure 41: Magnitude responses for various quantizations of a filter with 80 dB stopband attenuation.

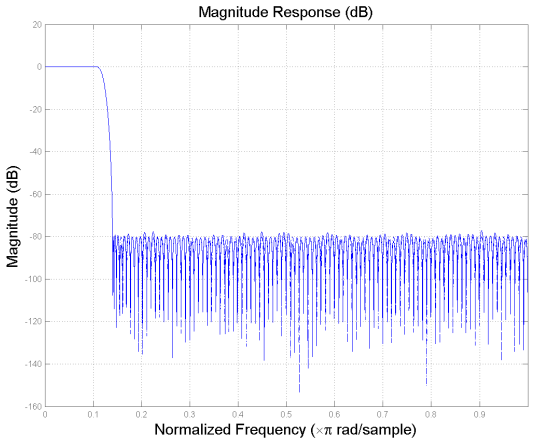

Figure 40: Magnitude response of the filter quantized with [16,18] format.

Note that whether we scale the coefficients and use `[16,15]` or we don't scale and we use `[16,18]`, the actual stored value (the binary bits) of each coefficient is the same. However, in the former case, the filter now has a gain of 18 dB due to the multiplication by eight. But this can be compensated at the end, by moving the binary point 3 bits to the left, without changing the bits.

To emphasize the point regarding the need to use both the right number of bits and use them wisely, we present the magnitude response of four different quantizations of the same filter. In all cases, the format has been selected to cover the range $[0.125, 0.125)$. The responses are shown in Figure 41. Notice that if you have fewer than 16 bits available, you might as well redesign the filter, since you have many more multipliers than you can use[6]. On the other hand, increasing the precision to 24 bits provides only modest improvements in this case.

10.3 Fixed-point filtering

Quantizing the coefficients correctly is not the only thing we need to worry about when implementing an FIR filter with fixed-point arithmetic. Suppose we want to implement this filter using the Direct-form structure. The structure is shown as a reference in Figure 42 for 5 coefficients. For the example at hand, we have 16 bit coefficients, and suppose we need to filter 16-bit data that is well scaled in

[6] If the specification is changed from 80 dB to 60 dB, 178 multipliers are required as opposed to 220. If it is reduced to 40 dB, 134 multipliers are required. Of course it is not a given that the application can allow this change in specifications. The point is having less than 16 bits makes it unfeasible to attain 80 dB.

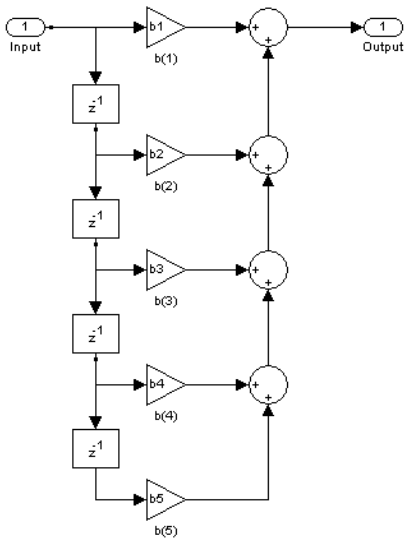

Figure 42: Direct-form implementation of an FIR filter with 5 coefficients.

the $[-1,1)$ range. We can generate random data with that characteristic as follows[7]:

```
q=quantizer([16,15],'RoundMode','round');
xq=randquant(q,1000,1);
```

We will use `[16,18]` format for the coefficients for illustration purposes. Since the input is already quantized, we don't need an input quantizer or a multiplicand quantizer,

```
Hq=qfilt('fir',{b},...
   'CoefficientFormat',[16,18]);
set(Hq,'InputFormat','none')
set(Hq,'MultiplicandFormat','none');
```

For reference, the other parameters are set by default as follows:

```
OutputFormat  = [16  15]
ProductFormat = [32  30]
SumFormat     = [32  30]
```

[7] In order to reproduce the results, one can reset the seed of the random number generator prior to generating the random vector: rand('seed',0);

however, we will temporarily set them to `'none'` to have a reference to compare to:

```
set(Hq,'OutputFormat','none');
set(Hq,'ProductFormat','none');
set(Hq,'SumFormat','none');
yi=filter(Hq,xq);
```

The quantity `yi` represents the "ideal" output. This is the best output we can hope to compute. Aside from using the 16-bit quantized coefficients, all computations are performed with double-precision arithmetic. Having `yi` provides a nice reference signal to compare to.

Now we set the parameters back to their default values, except the product format is not accurate for this case. The multiplication of a `[16,18]` coefficients with a `[16,15]` input sample results in a `[32,33]` product. On a DSP processor, we have two 16-bit registers being multiplied and the result stored in a 32-bit product register. The correct setting for the `ProductFormat` is `[32,33]`:

```
set(Hq,'OutputFormat',quantizer([16,15]));
set(Hq,'ProductFormat',quantizer([32,33]));
set(Hq,'SumFormat',quantizer([32,30]));
yq=filter(Hq,xq);
```

An extremely useful tool to monitor what has happened is `qreport(Hq)`,

	Max	Min	NOv	NUn	NOps
Coefficient	0.12	-0.026	0	0	220
Input	0.999	-0.999	0	0	1e3
Output	0.474	-0.536	0	2	1e3
Multiplicand	0.999	-0.999	0	0	22e3
Prod	0.12	-0.12	0	0	22e3
Sum	0.527	-0.537	0	0	22e3

which in this case reports that no overflows have occurred. To measure how good the output is, we compute the energy of the error and the maximum error,

```
norm(yi-yq,2)
ans =
       0.00054794884123692
norm(yi-yq,inf)
ans =
     3.05137364193797e-005
```

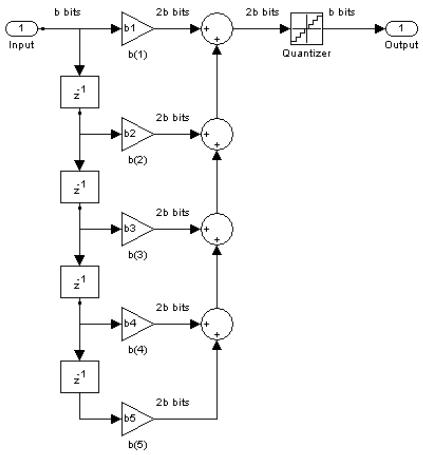

Figure 43: Model showing the quantization noise by reducing the number of bits from the adders to the output.

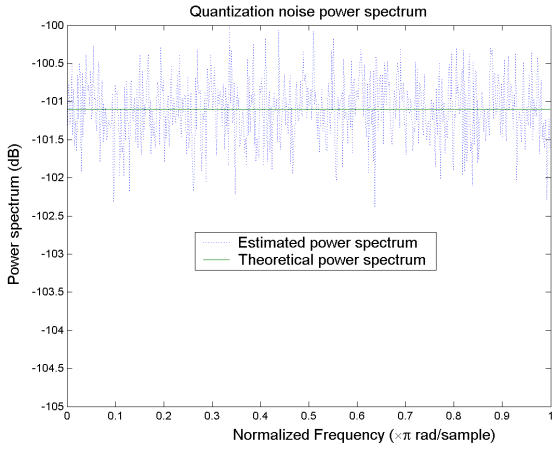

Figure 44: Theoretical and estimated power spectrum of the quantization noise.

Looking at Figure 42, one can see there is clearly a source of error when moving the data from the set of adders (what would be the accumulator in a DSP processor) to the output. Indeed, the wordlength is being reduced from 32 to 16 bits. A model of what is happening is shown in Figure 43.

The theoretical power spectrum of the quantization noise at the output of the filter corresponding to the model in Figure 43 is given by

$$S_y(\omega) = |H_n(e^{j\omega})|^2 \sigma_x^2$$

where $H_n(e^{j\omega})$ is the transfer function from the noise-source to the output -in this case simply one-, and σ_x^2 is the power spectrum of the noise source -in this case it is constant and equal to the variance of the noise[8]

$$\sigma_x^2 = \frac{2^{2(1-b)}}{12}$$

where b is the number of bits. So in this case, the theoretical power spectrum is constant and for 16 bits it is,

$$S_y(\omega) = 10\log_{10}\frac{2^{2(-15)}}{12} = -101.100811159671\text{dB}$$

[8]Strictly speaking, this formula is approximate because the signal at the accumulator does not cover the entire range $[-1,1)$ and because we are not quantizing an analog signal, rather we are reducing the number of bits in an already quantized signal.

An estimate of the noise power spectrum can be computed with the `nlm` function,

```
[H,w,Pnn]=nlm(Hq,512,100);
```

A plot of `Pnn` (in dB) compared to the theoretical power spectrum is shown in Figure 44.

If the quantization noise shown in Figure 43 is the only noise in the system, we should be able to get an output that exactly matches `yi` by setting the output format to be the same as the sum format (one can think of it as the ability to "look inside the accumulator"),

```
set(Hq,'OutputFormat',quantizer([32,30]));
yq=filter(Hq,xq);
norm(yi-yq,2)
ans =
      2.02838467848398e-006
norm(yi-yq,inf)
ans =
      7.98609107732773e-008
```

While the error has clearly been reduced, there is still some left, indicating some roundoff still present in the system. This is confirmed by looking at the power spectrum for the noise using `nlm`. The plot of the power spectrum is shown in Figure 45. The noise is obviously less

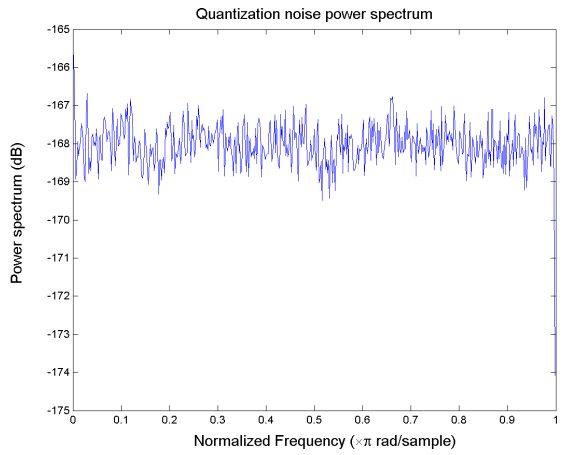

Figure 45: Noise power spectrum when making the output format equal to the sum format.

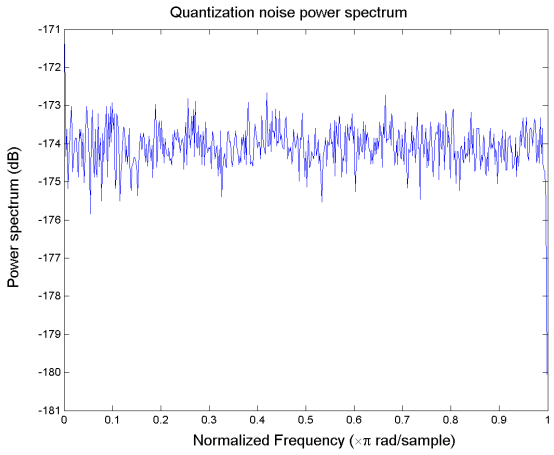

Figure 46: Noise power spectrum when setting both the sum format and the output format to [32,31].

than before (about -168 dB), which is consistent with the smaller errors we computed. To find the source of the error it is simply a matter of looking at the discrepancy between the product format and the sum format.

The sum format is set to `[32,30]` so that the three least significant bits from the product register are basically being lost. We may be tempted to make the sum format the same as the product format, but overflows occur left and right,

```
set(Hq,'SumFormat',quantizer([32,33]));
yq=filter(Hq,xq);
Warning: 1944 overflows in QFILT/FILTER.
```

The problem is that for additions, in general *k* bits are not enough to always store the result of adding two quantities with *k* bits each. Overflow *might* occur, and when adding so many numbers (220 in this example) chances are very high that it will occur. So it is preferable to live with some roundoff error, rather than to overflow (the two-norm of the error is a whooping 2.09011261755715, while the infinity norm is 0.285711827455089).

We can follow a trial-and-error procedure reducing the sum format to `[32,32]`, `[32,31]`, etc. until no overflow occurs. However, a better way is to go back to the `[32 30]` setting, filter, and look at the report given by `qreport`. For this example, `qreport` shows that the maximum and minimum sum values are 0.527 and -0.5357 respectively. Therefore, a format of `[32,31]` will be the optimal setting to minimize quantization noise while not overflowing.

```
set(Hq,'SumFormat',quantizer([32,31]));
set(Hq,'OutputFormat',quantizer([32,31]));
yq=filter(Hq,xq);
norm(yi-yq,2)
ans =
     7.53800283935414e-007
norm(yi-yq,inf)
ans =
     2.93366611003876e-008
```

Once again, the better results are confirmed by `nlm` which now shows a power spectrum for the noise of -174 dB. The power spectrum plot is shown in Figure 46.

10.3.1 Using an accumulator with extended precision

The results obtained previously are the best we were able to obtain with a 32-bit accumulator such as that available in some early DSP processors. Modern DSP processors provide an accumulator with extended preci-

sion, so-called *guard bits*, typically 40 bits when the data wordlength is 16 bits.

If such an accumulator is available, we can get better results once again if we use the extra bits wisely. For instance, the following setting for the sum format will not do,

```
set(Hq,'SumFormat',quantizer([40,31]));
set(Hq,'OutputFormat',quantizer([40,31]));
```

because no overflow occurred with the `[32,31]` setting anyway. So throwing extra bits does no good (the errors are exactly the same as for the `[32,31}` case). However, if we set the LSB weighting the same as for the product format, namely, if we use the following setting,

```
set(Hq,'SumFormat',quantizer([40,33]));
set(Hq,'OutputFormat',quantizer([40,33]));
```

the errors between "ideal" and actual become exactly zero. Of course, in this example it was not necessary to have a full 40-bit accumulator to achieve an output exactly equal to what we have called ideal. Once again, from the report generated with `qreport` it was evident that a setting of `[34,33]` for both sum and output would have done.

In an actual DSP processor the output is not of the same width as the accumulator, so realistically we need to set the output format back to either 16 bits or 32 bits in this example. Assuming we have 32 bits for the output, we can once again determine the best possible output setting by using `qreport`. In this case, `[32,31]` is the best setting because the minimum value reported at the output is -0.5357. The two-norm and infinity-norm of the errors are

```
norm(yi-yq,2)
ans =
     6.82098421980174e-009
norm(yi-yq,inf)
ans =
     3.49245965480804e-010
```

which compare favorably with 7.53800283935414e-007 and 2.93366611003876e-008 respectively (which were the best we could do for a 32-bit output with a 32-bit accumulator).

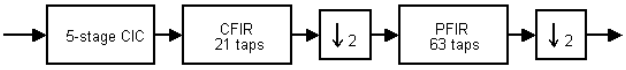

Figure 47: Block diagram of the decimation part of the DDC.

11 A design example

In this section we present an example of designing two FIR filters for use on a digital down-converter (DDC) to be used to downconvert a GSM signal. The hardware to work with is a Graychip 4016 multi-standard quad DDC chip [18].

Roughly speaking, a digital down-converter has two main parts. The first section, which consists of a numerically-controlled oscillator (NCO) and a mixer is used to "bring" an IF signal down to baseband. The second section is a (multistage) decimator used to isolate the desired signal.

In this design example we concentrate on the second part, i.e., we assume the signal has been moved to baseband in a satisfactory manner.

For decimation purposes, the 4016 provides for a multistage approach consisting of 3 FIR filters. Of the three filters, one is a cascaded integrator-comb (CIC) 5-stage decimator and two are programmable decimate-by-two FIR filters.

The multistage decimator block diagram is shown in Figure 47. The 5-stage CIC filter takes the high-rate input signal and decimates it by a programmable factor. The CIC filter is followed by a 21-tap compensation FIR (CFIR) filter that equalizes the "droop" due to the CIC filter and provides further lowpass filtering and decimation by 2. The CFIR is followed by a 63-tap programmable FIR (PFIR) filter that is used for a final decimate-by-2.

One thing to note is that in a multistage decimator one would always put the simplest filter first (that is, working at the highest rate), and would progressively increase the complexity of the filters in subsequent stages. This is exactly what happens here, the CIC filter is attractive at high rates because it provides multiplierless operation. The filter provides (coarse) lowpass filtering using adders and delays. The filter is not without its drawbacks though, its magnitude response is very far from ideal and exhibits a "droop" in the passband which progressively attenuates

signals. The CFIR filter is also relatively simple, having only 21 taps. Its primary mission is to compensate for the droop from the CIC filter. The PFIR filter is the most complex of the three, requiring 63 multiplications per sample, which is why it operates at the lowest rate.

It is worth pointing out that this is a good example of designs that require a fixed filter order. Also, both the CFIR and PFIR are linear-phase filters by construction, the designer can specify only half of its multipliers. Linear phase is usually a desirable characteristic in data transmission. The available wordlength for the coefficients of both the CFIR and PFIR filters is 16 bits.

11.1 Using the 4016 for GSM

The 4016 is programmable so that it can be used with multiple standards. To this extent, the decimation factor of the CIC filter can be selected as well as the coefficients for both the CFIR and the PFIR filters.

For the particular case of GSM, we have the following requirements [18]

- Input sample rate: 69.333248 MHz
- CIC decimation factor: 64
- CFIR input sample rate: 1.083332 MHz
- PFIR input sample rate: 541.666 kHz
- PFIR output sample rate: 270.833 kHz
- Passband width: 80 kKz
- Passband ripple: less than 0.1 dB peak to peak

The CIC filter has 5 stages and a decimation factor of 64. To view the magnitude response of this filter, we can simply create a CIC decimation object and use `fvtool`,

```
Hcic=mfilt.cicdecim(64,1,5);
fvtool(Hcic)
```

The magnitude response is shown in Figure 48. The filter exhibits a $|\sin(x)/x|^5$ shape. It also has a large DC gain (more than 180 dB), that has to be compensated for. To compensate for this large gain, the 4016 provides a power-of-two scaling prior to data entering the filter, in order to avoid overflows.

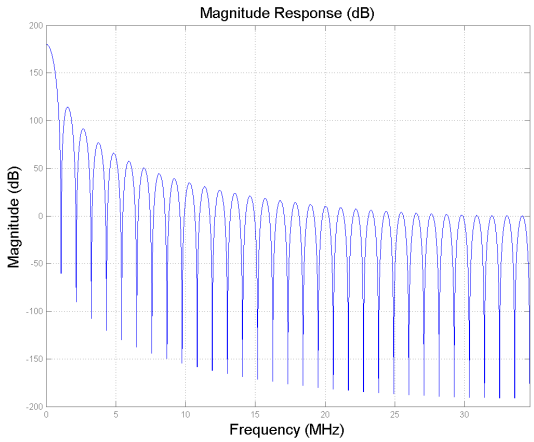

Figure 48: Magnitude response of 5-stage CIC decimator.

11.1.1 Designing the CFIR filter

Since the overall passband that is desired is 80 kHz, it is worthwhile to look at the CIC response in this band to get an idea of what the CFIR filter must compensate for. The passband details of the CIC filter are shown in Figure 49. The filter shows a droop with an attenuation of about 0.4 dB at 80 kHz. This is far more than the allowable peak to peak ripple.

We want to design an optimal equiripple filter to make the most of the 21 taps available. Since only 11 coefficients are actually freely specifiable, we are constrained to a linear-phase design.

We choose to use the `firceqrip` function for the following important reasons:

- It allows for compensation of responses of the form $|\frac{\sin(x)}{x}|^N$.
- The filter order is specifiable.
- It allows for a slope in the stopband, which we will use to attenuate spectral replicas of the PFIR filter that follows.
- We can constrain the peak passband and stopband ripples.

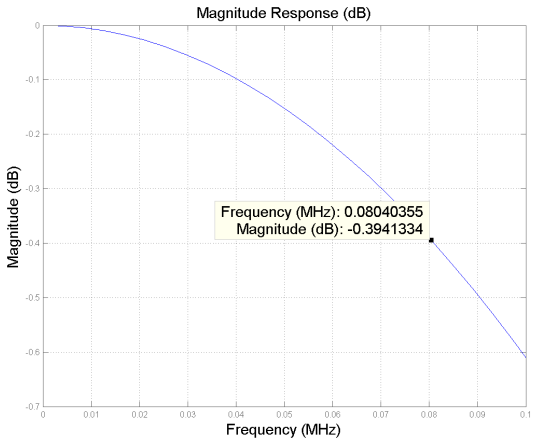

Figure 49: Passband details of scaled 5-stage CIC decimator.

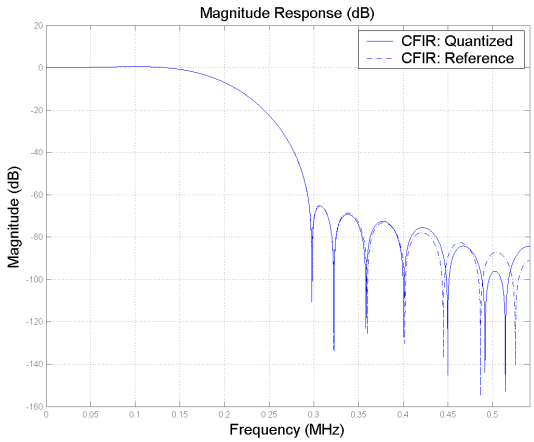

Figure 50: Magnitude response of CFIR filter.

- Instead of the cutoff frequency, we can specify the passband-edge frequency. In this particular case, since the passband is the interval $[0, 80\text{kHz}]$, we want to compensate for the CIC droop in the passband only.

The filter order is determined for us by the hardware. For the passband-edge frequency, we select 80 kHz, since this the final passband of interest. We choose a very small passband ripple, 0.01 dB, in order for the overall ripple to be way within spec, keeping in mind there is still the PFIR filter to follow which will add its own passband ripple. The stopband attenuation is selected as 40 dB with a 60 dB slope to provide adequate attenuation of the PFIR spectral replicas. Because this is a 5-stage CIC, the droop is of the form $|\frac{\sin(x)}{x}|^5$, so we select 5 as the sinc power to compensate for. Finally, the sinc frequency factor is chosen as 0.5.

```
N = 20; % Filter order
Npow = 5; % Sinc power
w = 0.5; % Sinc frequency factor
Apass = 5.7565e-004; % 0.01 dB
Astop = 0.01; % 40 dB
Aslope = 60; % 60 dB slope
Fpass = 80/541.666; % Passband-edge
cfir = firceqrip(N,Fpass,[Apass, Astop],...
       'passedge','slope',...
       Aslope,'invsinc',[w,Npow]);
Hcfir = mfilt.firdecim(2,cfir);
```

The magnitude response of the CFIR filter is shown in Figure 50 quantized to 16 bits. Without zooming in, it is hard to see the passband inverse-sinc response. We can see however, as expected, the large transition width along with the sloped stopband. Since the largest coefficient of the CFIR filter is 0.37, we use a `[16,16]` format to make the most of the 16 bits available.

To get an idea of the combined filter CIC*CFIR, we overlay the magnitude response of each of these filters, along with the combined magnitude response of the two. This is shown in Figure 51. We can see the spectral replicas of the CFIR filter centered around the frequency it is operating at, 1.083332 MHz. It is hard to see the sinc-compensation in this plot. For this we zoom in further. The zoomed-in plot is shown in Figure 52. The plot covers approximately the band $[0, 120\text{kHz}]$. It is evident from the plot that the combined response is virtually flat in the passband (up to 80 kHz).

11.1.2 Designing the PFIR filter

An overlay of the GSM spectral mask requirements [18] with the combined response of the CIC filter and the CFIR

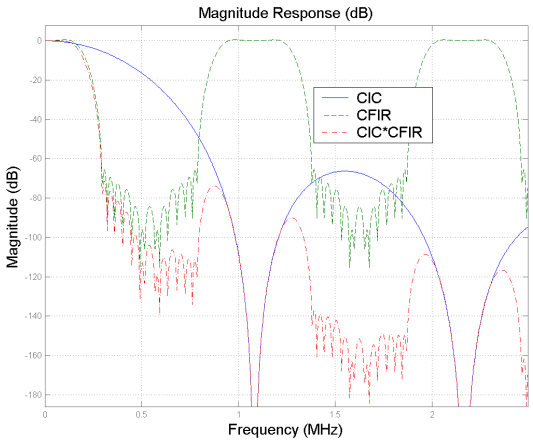

Figure 51: Magnitude response of CIC filter and CFIR filter overlayed, along with the combined response of the two.

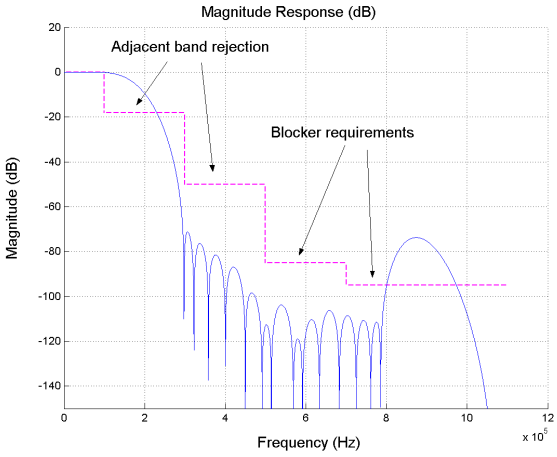

Figure 53: Magnitude response of combined CIC and CFIR filters overlaid with the GSM spectral mask requirements. Clearly the combination of these two filters does not meet the GSM requirements.

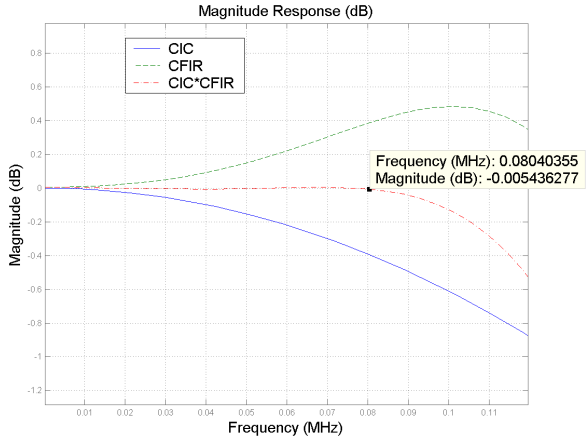

Figure 52: Passband details for the magnitude response of CIC filter and CFIR filter overlayed, along with the combined response of the two.

filter is shown in Figure 53. It is evident from the plot that the combination of these two filters is not sufficient to meet the GSM requirements for either adjacent band rejection or blocker requirements. The combined filter still has a transition band that is too large, due to the large transition band from the CFIR filter.

The PFIR filter is intended to be used to do the extra work required to meet the GSM specifications. It is a linear-phase FIR filter consisting of 63 taps. The design gets a little tricky though. We know that the passband-edge is 80 kHz, and the first adjacent band is at 100 kHz. If we design a simple lowpass filter with `remez` or `gremez` as follows:

```
N = 62;
Fs=541666;
F=[0 80e3 100e3 541666/2]/(Fs/2);
A = [1 1 0 0];
W = [1 1];
pfir= gremez(N,F,A,W);
Hpfir = mfilt.firdecim(2,pfir);
```

The passband ripple requirement is not quite met. We can alter the weights to get better passband ripple, but we must be careful not to violate the adjacent band specifications. A setting of `W = [10, 1];` would do the trick, but with significantly less adjacent band attenuation. A compromise can be achieved by setting up the design as a lowpass with two separate stopband regions, each one with a different weight to be used in the optimization:

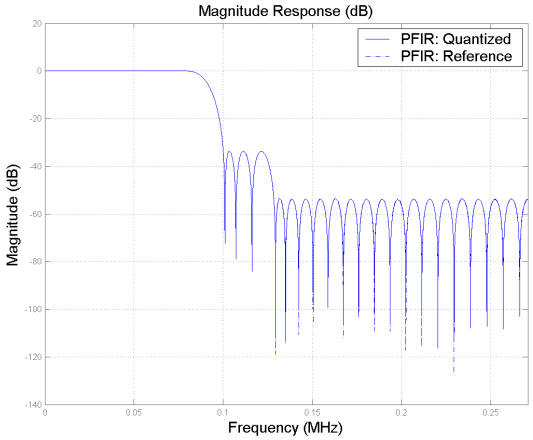

Figure 54: Magnitude response of PFIR filter.

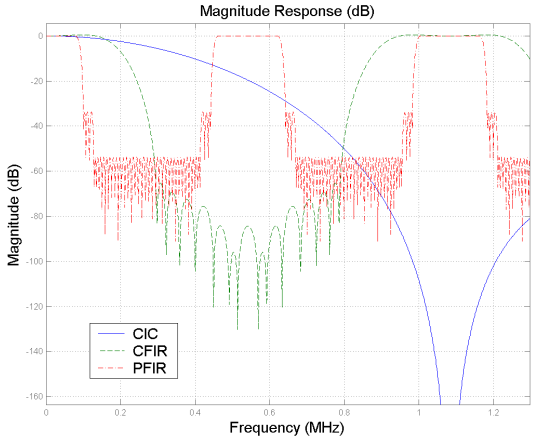

Figure 55: Magnitude response of CIC, CFIR, and PFIR filters.

```
N = 62;
Fs=541666;
F=[0 80e3 100e3 122e3 132e3 541666/2]/(Fs/2);
A = [1 1 0 0 0 0];
W = [10 1 10];
pfir= gremez(N,F,A,W);
Hpfir = mfilt.firdecim(2,pfir);
```

The quantized PFIR filter is shown in Figure 54. The maximum coefficient is 0.3378 so once again we use the `[16,16]` format. The reference (nonquantized) filter is also shown, but it is practically indistinguishable from the quantized response. The different attenuation in the two stopbands due to the different weighting is evident. The passband ripple is kept small in order to not exceed the allowed peak to peak ripple.

A plot showing the magnitude responses of all three filters, CIC, CFIR, and PFIR, is shown in Figure 55. Notice that the sloped stopband of the CFIR filter provides maximum attenuation when the spectral replicas of the PFIR filter occur.

The overall response of the combination CIC*CFIR*PFIR is shown in Figure 56. The GSM spectral mask requirements are now easily met as is clearly shown in the figure. The passband details are shown in Figure 57. The requirement that the peak to peak ripple be less than 0.1 dB is easily met. Clearly the design could be further tuned to provide a smaller transition width at the expense of larger peak to peak passband ripple and/or lesser adjacent band rejection.

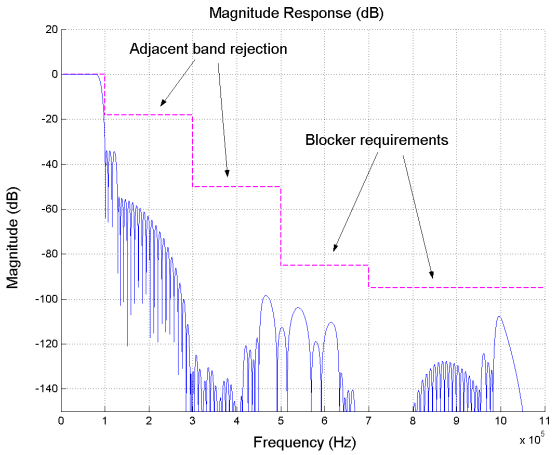

Figure 56: Combined response of CIC, CFIR, and PFIR filters, along with GSM spectral mask requirement.

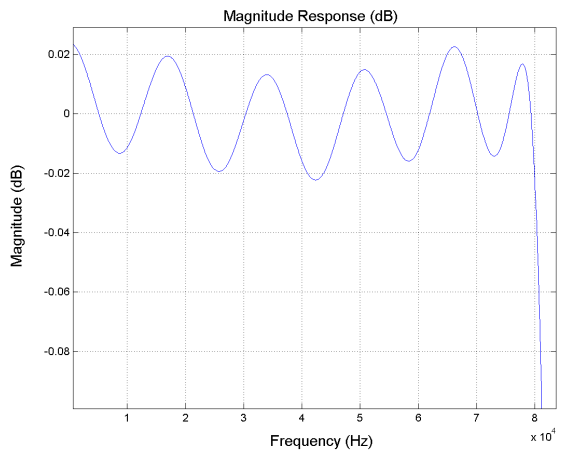

Figure 57: Passband details of combined magnitude response of the CIC, CFIR, and PFIR filters.

A Revision history

Revision 1.1 - Corrected section on design of perfect-reconstruction two-channel filter banks.
Revision 1.0 - Initial release.

References

[1] S. J. Orfanidis, *Introduction to Signal Processing,* Prentice Hall, Upper Saddle River, 1996.

[2] T. Saramaki. "Finite impulse response filter design," Chapter 4 in *Handbook for Digital Signal Processing,* S. K. Mitra and J. F. Kaiser eds. John Wiley and Sons, New York, 1993.

[3] L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing,* Prentice Hall, Englewood Cliffs, 1975.

[4] E. W. Cheney, *Introduction to Approximation Theory,* American Mathematical Society, Rhode Island, 1998.

[5] J. H. McClellan, T. W. Parks, and L. R. Rabiner, "A computer program for designing optimum FIR linear phase digital filters," *IEEE Trans. Audio Electroacoust.* AU-21, 506-526, 1973.

[6] I. W. Selesnick and C. S. Burrus, "Exchange algorithms that complement the Parks-McClellan algorithm for linear-phase FIR filter design," *IEEE Trans. on Circuits and Systems II,* 44(2):137-142, February 1997.

[7] Suggested by R. A. Vargas, in private communication, 2001.

[8] O. Hermann, and H. W. Schüssler, "Design of nonrecursive digital filters with minimum phase," *Electron. Lett. 6*,pp. 329-330, 1970.

[9] Y. Neuvo, C.-Y. Dong, and S. K. Mitra, "Interpolated finite impulse response filters," *IEEE Trans. on Acoust. Speech and Signal Proc.,* vol. ASSP-32, pp. 563-570, June 1984.

[10] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*, Prentice Hall Signal Proc. Series, Englewood Cliffs, 1993.

[11] R. Lyons, "Interpolated narrowband lowpass FIR filters," *IEEE Signal Proc. Mag.,* pp. 50-57, January, 2003.

[12] T. Saramaki, Y. Neuvo, and S. K. Mitra, "Design of computationally efficient interpolated FIR filters," *IEEE Trans. on Circuits and Systems,* vol. 35, N0. 1,pp. 70-88, January 1988.

[13] T. Saramaki, "A class of FIR Nyquist (*N*th-band) filters with zero intersymbol interference," *IEEE Trans. Circuits Syst.* CAS-34, pp. 1182-1190, 1987.

[14] P. P. Vaidyanathan and T. Q. Nguyen, "Eigenfilters: a new approach to least squares FIR filter design and applications including Nyquist filters," *IEEE Trans. Circuits Syst.* CAS-34, pp. 11-23, 1987.

[15] S. K. Mitra, *Digital Signal Processing. A Computer-Based Approach,* 2nd Ed., McGraw-Hill, New York, 2001.

[16] N. J. Fliege, *Multirate Digital Signal Processing*, John Wiley & Sons Ltd. Chichester, 1994.

[17] f. harris, “Multirate Signal Processing in Transmitter & Receiver Designs,” UCLA Extension course, November, 2000.

[18] Graychip, Inc., “GC4016 MULTI-STANDARD QUAD DDC CHIP DATA SHEET,” Revision 1.0, August 27, 2001.