Chapter 5.

Channel Estimation for Broadcasting Receivers

Contents

5.1. A Review of Previous Contributions

Prompted by the increasing popularity of the OFDM modulation for digital broadcasting and wireless networking, the topic of coherent demodulation and channel estimation using pilots received considerable attention with both academic and industrial research communities. First results began to be published in the early 90's.

The first explicit results were presented in 1991 in [35] for a wireless system with trellis-coded modulation (TCM) and rate-compatible puncturing (RCP). The proposed channel estimation strategy was to employ separable Wiener filters in the frequency domain, first along the s axis, then along the k axis, the aim being to reduce the complexity, the data and coefficients storage, as well as the delay, compared to a 2D Wiener filter. An essential advantage of this solution is that both interpolation and noise reduction is performed with the same polyphase Wiener filter.

A similar technique is also reported in [10] for a rectangular pilot grid. In this case, the order in which the interpolation/estimation is performed is not important, although in the paper the estimation in time direction (s axis) is performed first. In addition to [35], this paper takes the residual frequency shift and the asymmetric nature of the channel impulse response (CIR) into account, which appear as additional incremental rotations in the frequency domain, along the s and k axes respectively.

Unfortunately, both [35] and [10] lack an informative performance analysis with respect to filter sizes and channel statistics. This would be highly useful in that it would enable designers to quickly select the appropriate filter sizes and coefficients based on the desired performance and the given channel properties. This present thesis also addresses these shortcoming and shows how to optimally design the separable polyphase Wiener filters. The solution of using separable Wiener filters in the frequency domain is very scalable and efficient, having also a complexity which does not depend on the symbol size. With careful design and optimized coefficients, it can achieve excellent performance and the lowest implementation cost.

A different approach is taken in [92], where the channel impulse response (CIR) is estimated in time domain. The application scenario is also different, a symbol containing only pilots. A noisy CIR is obtained by converting the pilot carriers to time domain through an inverse DFT. The actual CIR estimation is performed by multiplying the noisy CIR with a special square matrix **Q** of size N×N, whose elements depend on the channel statistics. The **Q** estimation matrix can be derived for MMSE (minimum mean square error) or LS (least squares) estimation. The MMSE estimator depends on the actual channel statistics (noise and delay profile) and achieves better performance than the LS estimator. The latter is also equivalent to a zero-forcing estimator, as it only performs interpolation between the pilot samples.

Furthermore, [92] proposes a reduced-complexity version of these estimators, which exploits the fact that the CIR is limited to L samples, which reduces the size of the **Q** matrix from N×N to L×L. In comparing the two estimators, the paper incorrectly states that the complexity of the MMSE estimation is higher that that of the LS estimation. Although the expression of **Q** in the case of MMSE is indeed more complex than for LS, this does not translate in a higher hardware complexity since the matrix is computed only once at design time and its size is the same for both MMSE and LS. There is one sense in which it can be said that the MMSE estimation is more complex than LS, namely when different matrix coefficients are stored in order to provide optimal accuracy for different delay profiles and noise variances. The latter two have to be estimated and the appropriate coefficients selected from a look-up table.

In [79] channel estimation through interpolation between pilot carriers in frequency domain is discussed. The channel delay is considered to have a negative exponential profile and the analyzed interpolation types are zero-order hold and linear. As expected, the performance of these channel estimators are rather modest and the noise affecting the pilots is not attenuated at all. Interpolation performance is given in terms of average symbol error probability (SEP) versus oversampling factor for different QAM modulation types.

Another discussion of the 2D Wiener estimator in frequency domain is presented in [34]. The general case of an arbitrary pilot grid is treated first from a theoretical point of view. For a rectangular grid, the authors show that there is virtually no degradation when employing two separable 1D filters. Estimation performance is given in terms of MSE versus pilots SNR with the number of filter taps as a parameter.

In [18] the authors extend the idea in [92] by proposing a method for reducing the complexity of the MMSE estimator. The complexity reduction is achieved through a low-rank approximation using the singular value decomposition (SVD) of the channel autocorrelation matrix. This approximation leads to an irreducible error floor of the estimator, which can be reduced by increasing the estimator complexity.

Although the authors argue that this solution has a complexity which is comparable with that of a time-domain-only estimator, they fail to mention that the most costly operation is the multiplication with the matrix $\mathbf{U}$ of the singular vectors and its inverse. This matrix is unitary, i.e. $\mathbf{U}^H\mathbf{U} = \mathbf{I}$. Unfortunately, the elements of the matrix depend on the actual channel statistics, which requires the storage of a large number of coefficients in a practical implementation. Moreover, the cost of the matrix multiplication is $\mathcal{O}(n^2)$ with respect to the symbol size. While arguably offering a lower performance, the solution presented in [92], being based on using FFT and IFFT, has a complexity of $\mathcal{O}(n \log n)$ and is independent on the channel statistics. Moreover, the FFT of the OFDM demodulator can be reused for channel estimation as well.

The problem of the pilot spacing in block-based estimators/interpolators is dealt with in [64]. The authors show that the best estimation results for Gaussian noise are obtained when the pilots are equally spaced. When no noise is present, if the number of pilots is at least the length of the channel impulse response, perfect estimation can be achieved regardless of the pilots position. Assuming no a priori knowledge of the channel, the expression of the ML estimator is derived for equally-spaced pilots. Moreover, the paper proves that the estimation performance in the case of an invariant multi-path channel is the same for rectangular pilot grids with $D_s = 1$ (pilots every D_k carriers in all symbols) or $D_k = 1$ (pilots in all carriers, every D_s symbols).

Another channel estimation solution that relies on the singular value decomposition of the channel correlation in frequency domain, similarly to [18], is presented in [55]. In addition to [18], this estimator also exploits the correlation along the time axis in order to achieve better performance through additional filtering. Knowing that in frequency domain the correlation along the time axis s is independent of the correlation along the time axis k, the two estimation problems can be separated. The additional filtering/estimation in time domain is performed in [55] between the two matrix multiplications, the solution being straightforward.

In [36] an estimator for a comb-type (rectangular pilot grid with $D_s = 1$) pilot arrangement is considered. The channel estimation for the pilot positions and the interpolation are performed in two separate steps, which is not the most efficient solution. In this chapter, we will also show how to combine these into a single filtering operation. The estimation uses exactly the same SVD like in [18], while the interpolation types are limited to linear and second-order. Before and after interpolation, an incremental rotation of the complex samples is performed in order to center the complex spectrum, which is the CIR in time domain, according to the duality principle. Thus, the samples have a slower variation and the interpolation errors are reduced.

A solution for channel estimation using separable Wiener filters in the frequency domain is presented in [80], with application to DBV-T. The filter in time direction has two taps and is optimized for a single Doppler profile and guard interval. The filter in frequency direction has 12 taps and is adaptive with respect to the channel power delay profile. As the latter is asymmetrical, the resulting filters have complex coefficients, which increases the number of multiplications by four. The estimator performance is expressed through BER-vs-SNR curves for a DVB-T system in 2K mode with 1/2 code rate and QPSK modulation.

A very informative treatment of the block-based channel estimation in frequency direction using windowed DFT estimation is found in [99]. The estimation is performed on a symbol-by-symbol basis by interpolation between pilots using DFT. Rectangular, Hamming, and Hanning window types are investigated, using a 5-tap channel with an exponential power delay profile. The difference between the different window types is the achieved interpolation error floor, with the Hanning window offering the best performance. The ultimate cause of the interpolation error is the leakage of the channel taps energy into neighboring samples.

Without leakage, i.e. when the tap delays are exact multiples of the sampling period, no interpolation error would occur. In [99], performance is assessed using MSE-vs-SNR and SER-vs-SNR curves. Comparisons with a robust Wiener interpolator show only minimal performance degradations. The Wiener interpolator was designed in the assumption that all channel taps in the guard interval have the same power (shifted rectangular spectrum of the frequency-domain CTF).

In [54] another solution for robust channel estimation is presented. By robust it is understood that the estimator is insensitive to channel statistics (power delay profile and Doppler spread). Using a 2D pilot grid, the interpolation is performed block-wise using 2D FFT, filtering, and IFFT. Unlike, previous solutions, the processing is done block-wise along the continuous time axis also, not only along frequency axis. Regardless of the achieved estimation performance, the inherent complexity of a 2D FFT transform renders this solution impractical for actual implementations. Moreover, due to the block nature of the processing, the estimates for the OFDM cells at the margins of the block will have a larger error.

An interesting comparison between the LS and the MMSE estimators for block-based interpolation/estimation is offered in [63]. In this approach, the time-domain channel impulse response (CIR) is estimated. The LS estimate views the CIR as a deterministic but unknown vector, whereas the MMSE estimate views it as a random vector whose particular realization is to be estimated. The MMSE estimator adopts a Bayesian approach and relies on prior information regarding the channel statistics, i.e. power delay profile and noise.

The estimates are first obtained for arbitrary pilot positions. If the pilots are equally spaced, the expression of the ML estimator is reduced to that obtained in [64], which is essentially a DFT of the pilots. The paper provides also complexity and performance analyses. As expected, the performance analysis shows that the estimation performance is reduced at the block margins. A solution suggested by the authors is to decrease the pilot spacing at the margins in order to compensate for the performance loss. This solution, however, leads to an increase in complexity, while offering only a marginal improvement in the overall BER.

In [13] the problem of block-based estimation/interpolation is revisited. The estimation and the interpolation are performed in separate steps. Both LS and MMSE methods are considered for the estimation. The interpolation is performed either in frequency domain, using linear, second-order, low-pass, and spline interpolation, or in time domain using DFT interpolation. Low-pass interpolation is shown to offer the lowest interpolation error. However, no complexity analysis is performed.

The topic of performance evaluation of channel estimation algorithms is also dealt with in [16] in the context of 4G systems. The cases analyzed are limited to estimation through interpolation using separable 1D filtering. Three interpolation techniques are considered: linear, cubic spline, and FFT interpolation. The performance is expressed through BER-vs-SNR plots.

In [83] the CIR is estimated by filtering the FFT of the pilot cells using adaptive Wiener filters. The number of filters is exactly the length of the estimated CIR or channel taps, which are assumed uncorrelated. The paper discusses first the performance of MMSE estimators for finite and infinite filter length, which have the disadvantage that they require the channel statistics. For the adaptive estimators using the NLMS or the RLS algorithm, these statistics are not required and their performance exceeds the performance of an ML estimator, coming close to a matched MMSE one.

5.2. Performance Requirements

In the following, we intend to evaluate how the estimation performance affects the overall decoding performance, expressed as BER vs. SNR. The accuracy of the channel estimate is expressed as the estimate noise variance (power), denoted by σ_H^2. We also denote the noise power of the received signal by σ_N^2. The performance of the channel estimator is given by two factors. First, assuming that the received signal is noiseless, the channel estimate will still have a residual error, which is caused by imperfect interpolation. We denote this error by $\sigma_{N,min}^2$. Second, the estimator can be designed so that σ_H^2 is less than σ_N^2. We denote by G the noise attenuation of the estimate: $G = \sigma_N^2/\sigma_H^2$.

As the average signal power is one, the noise variance can be also expressed as the inverse of the signal-to-noise ratio SNR. Thus, we have that $\sigma_H^2 = 1/SNR_H$ and $\sigma_N^2 = 1/SNR_N$. Moreover, the noise variance can be also thought of as the mean square error (MSE) of the estimate: $\sigma_H^2 = MSE_H$.

Considering the interpolation error $MSE_{H,min}$ and the estimator gain G, the plot of the MSE_H against SNR_N have a shape like in **Figure 5.1**. It is worth mentioning that the estimator gain depends on the estimator design alone, whereas the interpolation error depends also on the spectrum of the bi-dimensional channel transfer function. The latter will be higher for high Doppler spreads and long channel impulse responses.

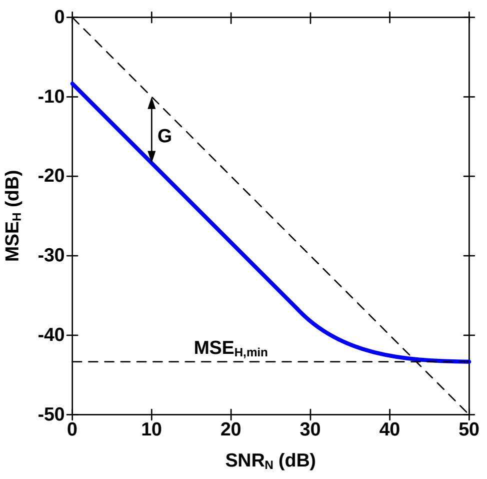


Figure 5.1.: Typical MSE_H vs. SNR_N profile

In order to asses the influence of the estimation error quantitatively, the simulation test bench in **Figure 5.2** has been set up. The entire chain has been coded in object-oriented Matlab and implements the 2K and 8K modes of the DVB-T standard. The test bench has a user data path and a reference data path, with independent AWGN sources on each, which allows to evaluate the effects of data noise and channel estimate error separately. It is essential that the two multi-path channel models are identical, so that the user data and the references experience identical time-variant multi-path delays.

For the following simulations we consider a multi-path channel with 8 consecutive discrete taps having the same average power and a spacing equal with the sampling period. The Doppler profile is Jakes, with a discrete symbol Doppler rate of 1/64. We define the symbol Doppler rate as the Doppler rate normalized to the symbol period, i.e. $T = T_U + T_G$. The DVB-T chain operates in the 2K mode, with a relative guard interval length of 1/32. All three QAM modulation types are considered. For each parameter combination, 1024 (2^{10}) OFDM symbols are simulated. It is important that the number of symbols per simulation is significantly larger than the Doppler symbol period. In our case this number is 64 (2^6), therefore there are 16 (2^4) Doppler symbol periods per simulation.

As a first step in our analysis, we investigate the effect of noise on the data samples and the channel estimate independently. First, we assume an ideal (noiseless) channel estimate. The

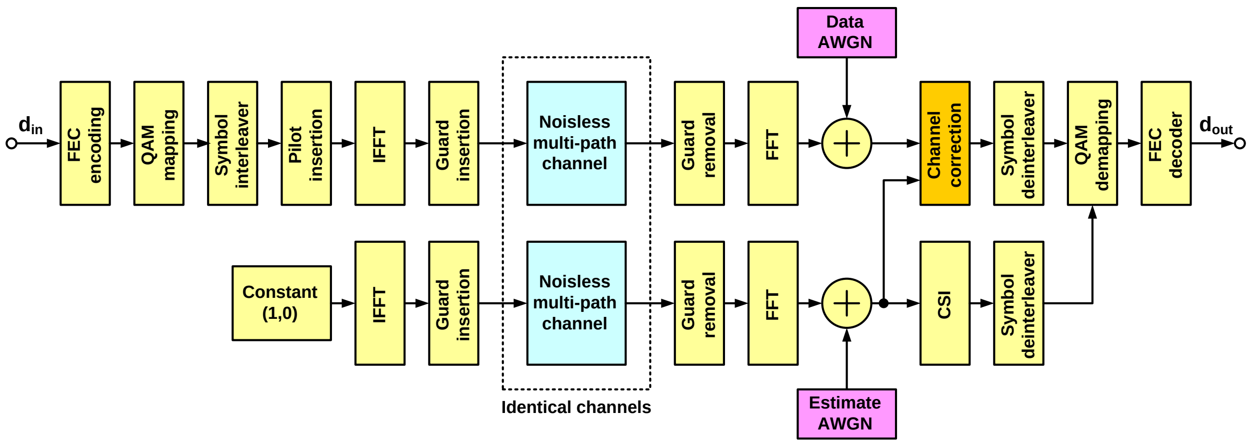


Figure 5.2.: Simulation test bench for evaluating the influence of the channel estimation performance on the overall BER

results are shown in **Figure 5.3** for 4-QAM, 16-QAM, and 64-QAM and an SNR range from -2 dB to 26 dB. Considering a QEF (quasi-error-free) limit at a BER of 10^{-4}, the corresponding SNR thresholds are 12.5 dB, 18 dB, and 23 dB respectively.

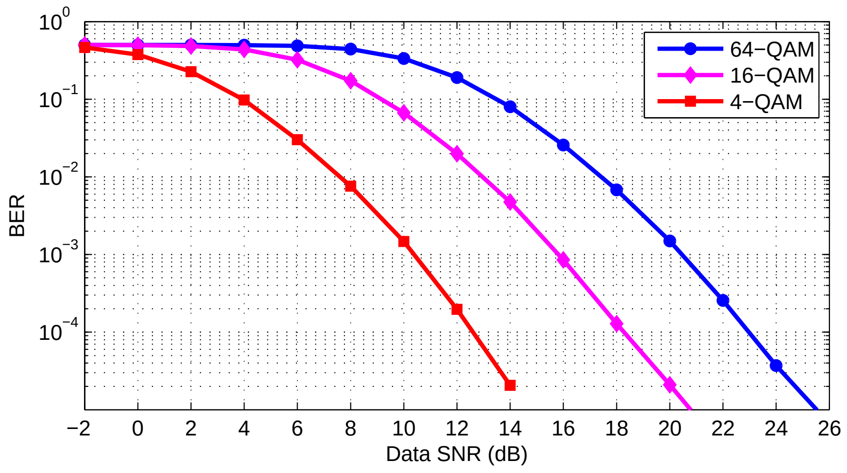


Figure 5.3.: BER vs. data SNR for noisless channel estimation

Second, we assume that the data samples are not affected by noise and that the channel estimate is affected by errors which are modeled as noise. If the received data and pilot samples are not affected by noise, the channel estimation errors occur only because of non-ideal interpolation. The results, shown in **Figure 5.4**, are almost identical to those in **Figure 5.3**. If we consider a QEF limit of 10^{-4}, the results show that the SNR of the channel estimate must be at least 24 dB. If this condition is met, any BER degradation above 10^{-4} will be caused by the data noise rather than the channel estimate.

We investigate now the effect of the channel estimate gain G on the overall BER. The resulting BER vs. SNR plots are show in **Figure 5.5** for four values of G: 0 dB (no gain), 3 dB, 6 dB,

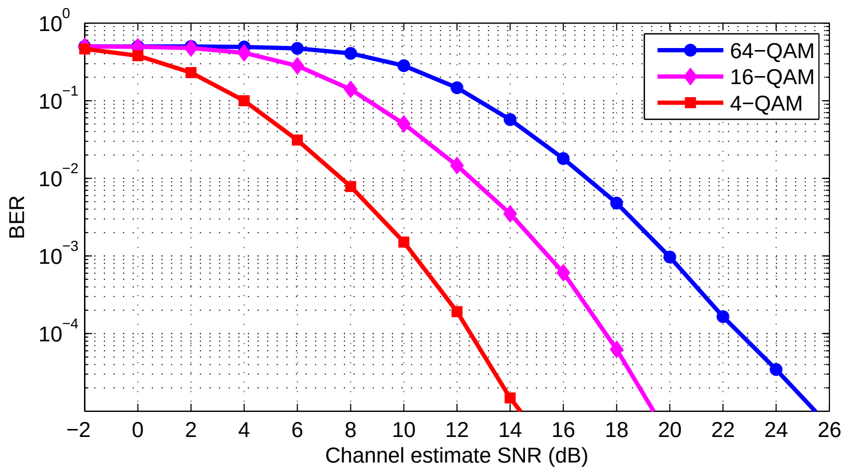


Figure 5.4.: BER vs. channel estimate SNR for noiseless received data

and noiseless estimate (infinite gain), for 64-QAM modulation only. These plots show that the optimization of the channel estimation can lead to a a tenfold improvement of BER compared to the case in which channel estimation is performed by interpolation only (zero gain).

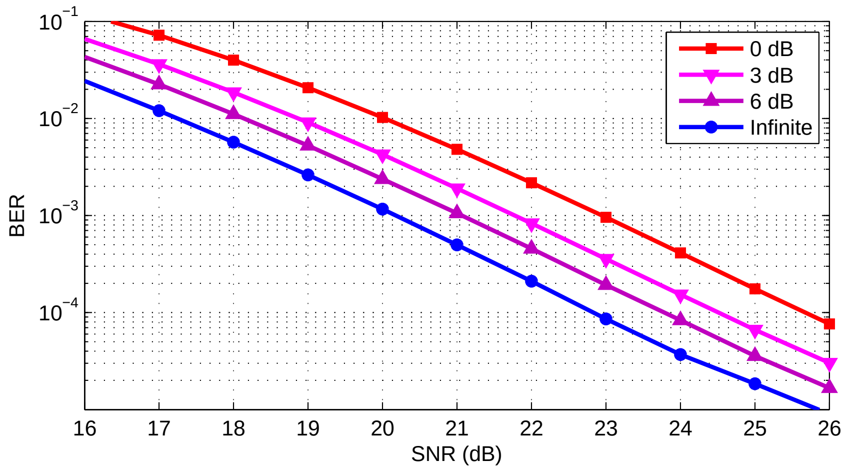


Figure 5.5.: BER improvement for various SNR gains of the channel estimate

The actual dependence of BER on the estimator gain G is shown in **Figure 5.6** for SNR values from 19 dB to 22 dB. The results indicate that for high estimation gains the BER can be improved by up to 10 times, as it can also be seen in **Figure 5.5**. The improvement is roughly the same for all SNR values. The most significant BER improvements are achieved for gains up to 6 dB (4 times). Increasing the estimation gain further incurs a higher estimation complexity and achieves only a minor performance improvement. This is exactly the reason why the reference pilots are transmitted with a boosted power (16/9 for DVB-T and 2 for DRM). Increasing the pilot power by 2 directly improves the estimate SNR with 3 dB, which in turn results in a BER improvement by approx. 2.5 times.

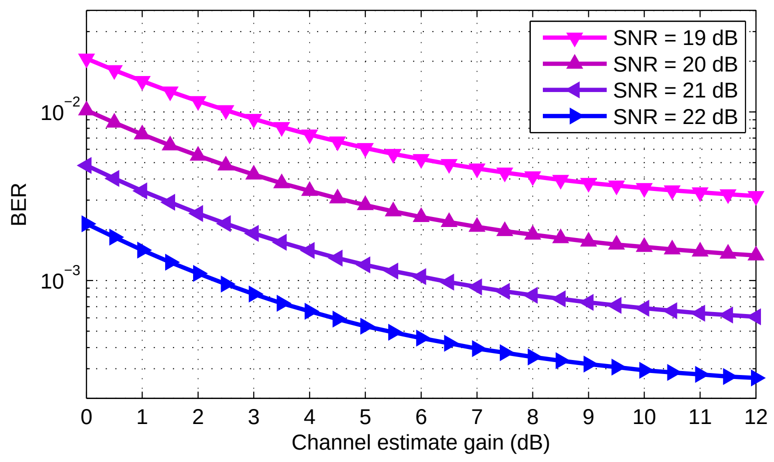


Figure 5.6.: Performance improvement with estimate gain for various SNR's

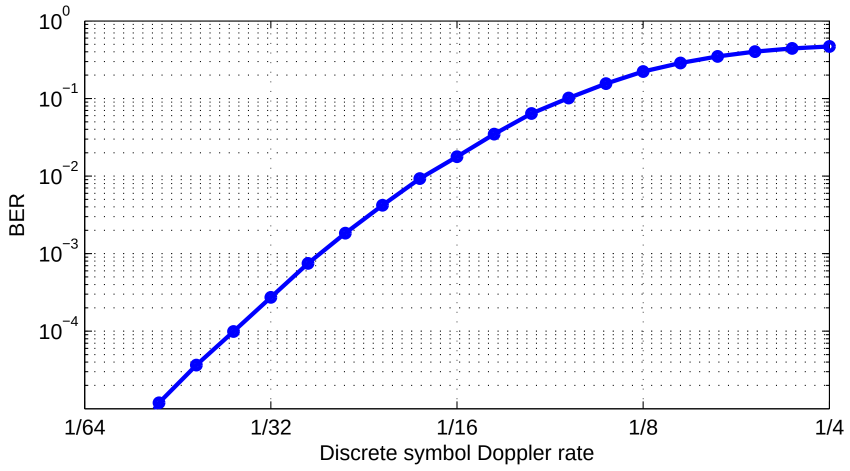


Figure 5.7.: BER vs. symbol Doppler rate for 64-QAM modulation

In OFDM systems, one significant source of errors is the ICI (inter-carrier interference) due to Doppler spread. We have shown in **Section 2.3** that the amount of ICI depends on the discrete symbol Doppler rate (discrete Doppler rate divided by the total symbol period T). For the same discrete Doppler rate, larger OFDM symbols will exhibit larger ICI values. For lower values, the errors resulting from ICI can be treated as additive noise. **Figure 5.7** shows the BER versus the symbol Doppler rate. For a QEF BER of 10^{-4}, the maximum symbol Doppler rate lies below 1/32. For a pilot spacing $D_s = 4$, the resulting oversampling factor is at least $32/(2 \cdot D_s) = 4$. This insights can be exploited in order to increase the gain of the channel estimator in the time direction.

5.3. Estimation Algorithms and Architectures

The channel estimator produces estimates of the bi-dimensional channel transfer function $\tilde{H}(s,k)$ based on the received values of the scattered reference pilots. For all existing OFDM broadcasting applications with coherent demodulation, the tilted pilot pattern shown in **Figure 3.2b** is preferred, with the spacing parameters D_s and D_k. The bi-dimensional pilot grid is used for sampling the time-variant CTF:

$$\hat{H}(s,k) = \frac{z_{s,k}}{p_{s,k}} \tag{5.1}$$

where $z_{s,k}$ is the noisy received value and $p_{s,k}$ is the known pilot value for OFDM cell (s,k), i.e. at symbol s and carrier k.

5.3.1. Estimation through 2D Filtering

The ideal solution for channel estimation is to employ a 2D Wiener filter directly in frequency domain and perform interpolation/smoothing between pilots, as shown in **Figure 5.8a**. The filter size is of course finite in the s and k direction. If we denote the number of taps along the two axes by M_s and M_k, both even, the expression of the channel estimate can be written as:

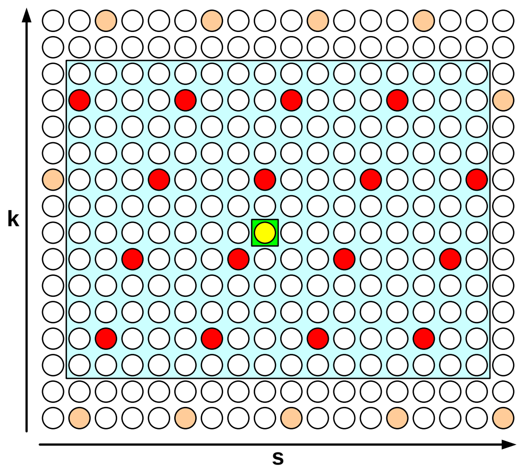
$$\tilde{H}(s,k) = \sum_{i=-M_s/2}^{M_s/2-1} \sum_{j=-M_k/2}^{M_k/2-1} h_{s,k}(i,j)\hat{H}\left(\left(\left\lfloor \frac{s}{D_s} \right\rfloor - i\right) D_s, \left(\left\lfloor \frac{k}{D_k} \right\rfloor - j\right) D_k\right) \tag{5.2}$$

A number of N_sN_k snapshots are used for computing the estimate $\tilde{H}(s,k)$ for each cell. Obviously, the filter coefficients depend on the relative position of the cell on the sampling grid. It can be seen that for such a grid there are D_sD_k distinct relative positions, therefore an equal number of coefficients sets will be needed. **Figure 5.8b** shows the grouping of the OFDM cells according to the coefficients sets used for their computation. Such a rectangular group contains D_sD_k cells for which distinct coefficients sets are used. Altogether, the number of coefficients that need to be stored is:

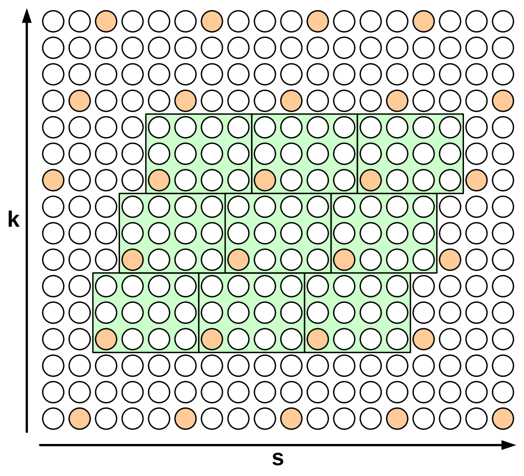
$$N_{coeff} = N_s \cdot N_k \cdot D_s \cdot D_k \tag{5.3}$$

In the case of DVB-T, where $D_s = 4$ and $D_k = 3$, if we consider a relatively modest filter with 4×4 coefficients, the total number of coefficients is 192. Depending on the actual hardware architecture and required throughput, such a large number of coefficients may not be realizable. If a sequential architecture is used, the coefficients can be stored in a ROM and accessed sequentially. A more serious disadvantage of a 2D estimator is the number of complex MAC operations required for computing an estimate sample, which is N_sN_k in this case.

The complexity and the number of coefficients can be reduced if the 2D filter is replaced with two separable filters, for the s and k axis respectively. Since there is no correlation between the



(a) Pilots used for computing the channel estimate



(b) Cell grouping according to the coefficients set used for estimation

Figure 5.8.: 2D channel estimation in frequency domain

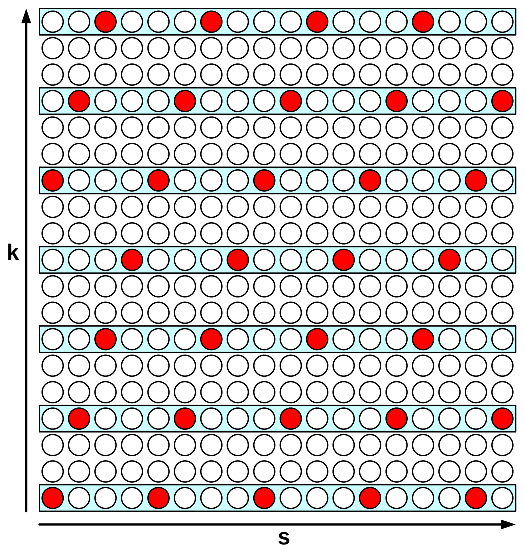
variations along the two axes, there will not be any performance degradation. Indeed, since the causes of two variations are completely independent, the associated statistical processes are completely uncorrelated. In this case, the 2D autocorrelation function can be expressed as the product of the autocorrelation functions in the s and k axes.

5.3.2. Estimation through 1D Filtering

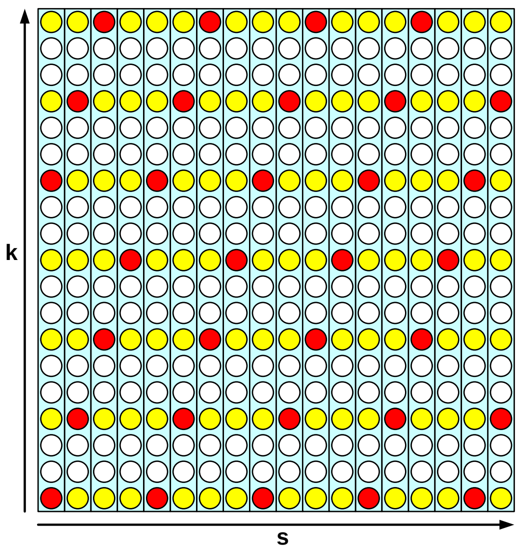
If a rectangular pilot grid (**Figure 3.2a**) is used, the order of the two estimation steps is irrelevant. For a tilted grid (**Figure 3.2b**), however, the interpolation in the s axis must be performed first in order to reduce the pilot spacing in the k axis from D_sD_k to D_k. The original D_sD_k spacing is usually not sufficient to fulfill the sampling theorem along the k axis. The interpolation process for a tilted pilot grid, using two separable 1D filters is shown in **Figure 5.9**.

From an estimation/filtering point of view, there is an important difference between the handling of the two axes. Along the s axis the estimation is performed continuously as symbols are received, whereas along the k axis the number of carriers is limited. Thus, estimation in k axis is performed block-wise using either a filter, like for estimation in the s axis, or a block-based estimator. In this work, we consider only separable filter-based estimation.

A good overview of various block-based interpolation methods is presented in [63]. The two main classes of estimators are the ML (maximum likelihood), which considers the CTF a deterministic but unknown vector, and the MMS (minimum mean square), which treats the CTF as the realization of a random process with known statistics. Essentially, the ML estimation performs just interpolation. For equally-spaced pilots this is equivalent to an DFT interpo-



(a) First along the symbol s axis



(b) Then along the carrier k axis

Figure 5.9.: Channel estimation using separable filters

lator. The MMS estimators, however, also perform noise reduction and thus estimation gain. Their drawback is that the optimum solution depends on the channel SNR and auto-correlation function, the latter being the Fourier transform of the power-delay profile.

The principle of 1D estimation through filtering is shown in **Figure 5.10** for a pilot spacing of 3 and a filter size of 4. Much like for the 2D case, the estimate has a similar expression:

$$\tilde{H}(s) = \sum_{i=-M_s/2}^{M_s/2-1} h_s(i)\hat{H}\left(\left(\left\lfloor \frac{s}{D_s} \right\rfloor - i\right) D_s\right) \tag{5.4}$$

The same expression applies to the estimation along the k axis, by simply replacing D_s with D_k and M_s with M_k. This equation is equivalent to a polyphase implementation of an interpolation filter. For each of the D_s possible phases there is an optimized set of coefficients. The theory behind polyphase interpolation and its adaptation for channel estimation, i.e. Wiener interpolators, is presented in **Section 5.4**. Instead of estimate the channel at the pilot position and then interpolate, it is more convenient to perform both operation in one step. The theory of Wiener interpolators shows that the polyphase coefficients can be obtained as solutions of a set of Wiener-Hopf equations.

It must be mentioned here that since the pilot samples and the CTF are complex functions, the estimators will also have complex coefficients in the most general case. However, if the signal spectrum is symmetrical, the imaginary part of the autocorrelation function is zero and the resulting coefficients will be real. This is desirable since the number of stored coefficients and MAC operations is reduced by half.

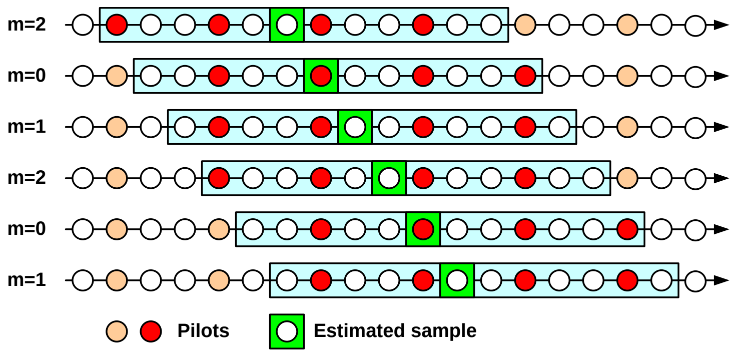


Figure 5.10.: Stream-based estimation using polyphase filters

5.3.3. Estimation along the Symbol Axis

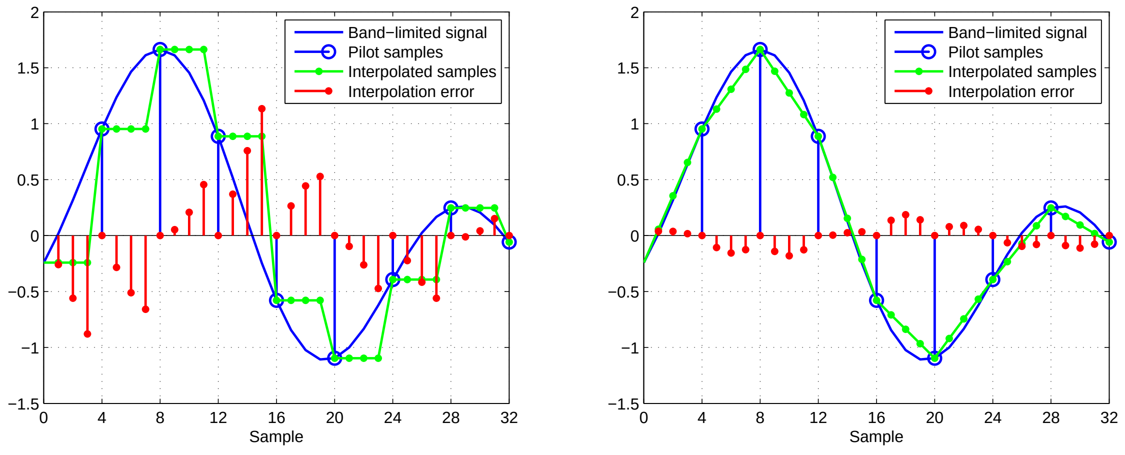
The CTF variations along the s axis are produced by the Doppler fading of the multi-path channel taps, which are in turn caused by the movement of the receiver or the ionospheric layers. Depending on the propagation conditions, various spectral profiles of these variations result, as introduced in **Subsection 2.1.2**: Gaussian, Jakes, or flat band-limited (rectangular). All these idealized profiles are symmetrical, although additional frequency shifts may destroy this symmetry.

Perfect channel estimation requires the exact knowledge of the Doppler spectrum. In order to maximize the estimator gain, the coefficients need to be optimized for the actual Doppler spread and SNR. It is therefore recommended to have a Doppler frequency and an SNR estimator, whose outputs can be used to select the appropriate set of coefficients from among different sets optimized for different Doppler frequencies and SNR's. Although an interesting topic in itself, adaptive estimators are outside the scope of this thesis.

Due to the algorithmic group delay of the filter, $G_s = M_s D_s/2 - 1$, the same number of data symbols must be buffered so that they remain aligned with the channel estimates. This is the toughest constraint in a channel estimator since the amount of memory needed to store G_s symbols can be very large. More exactly, the number of stored complex samples is $G_s N_{act}$, where $N_a ct$ is the number of active carriers per OFDM symbol. Memory is also needed for the taps of the multiple parallel filters, i.e. for storing the pilots for each carrier that contains pilots. The number of complex samples to be stored is the number of carriers with scattered pilots multiplied by the number of taps M_s.

In the case of DVB-T in 8K mode, $N_{act} = 6816$ and $D_s = 4$. Assuming a 4-tap filter, the resulting size for the group delay compensation memory is $6816 \times 7 = 47712$ complex samples. The size of the pilots memory is $2272 \times 4 = 9088$, which is significantly less than what is required for group delay compensation.

Having established the fact that the number of filter taps must be kept to a minimum, we can now also regard linear interpolation and zero-order hold as worthwhile alternatives to real channel estimation. In order to minimize memory requirements we can only perform interpolation at this stage, with no noise reduction. The latter will be implemented in the subsequent estimation stage only, i.e. along the k axis. In **Figure 5.11** we show how the process of zero-order hold and linear interpolation between pilots for a signal with a bandwidth of $1/8$ and a pilot spacing $D_s = 4$.



(a) Zero-order hold (b) Linear interpolation

Figure 5.11.: Interpolation between pilots

In deciding which of the two interpolation types is the acceptable minimum, the interpolation error has to be compared with the error caused by the inter-carrier interference due to the Doppler effect itself, so that the former will not exceed the latter. From the plot in **Figure 5.7**, which show the BER versus symbol Doppler rate, one can see that a quasi error free BER of 10^{-4} can be achieved only for symbol Doppler rates below $1/32$. Form **Figure 5.4** it can be seen that the same BER can be achieved if the channel estimate SNR (-MSE in dB) lies below 25 dB. Interpolation error has been evaluated in **Figure 5.37** for various type of interpolation filters. At a symbol Doppler rate of $1/32$, the interpolation MSE is -30 dB for zero-order hold and -70 dB for linear interpolation. It can now be seen that even in the case of a zero-order hold interpolation the effect on the BER are lower than that of the ICI caused by the same Doppler fading.

We can therefore safely state that increasing the number of taps of the interpolation filter along the s axis does not achieve any BER improvement. However, if a Wiener interpolator is used instead of a regular interpolator, an estimation gain can be achieved. According to **Figure 5.28**, the latter does increase with the number of taps and will contribute to a lower BER, as shown **Figure 5.5**.

The possibility to increase the number of taps can be easily exploited in various OFDM standards, which specify various symbol sizes, for instance 2K and 8K in DVB-T. In this case, the number of taps and the delay compensation memory will be sized for the largest symbol. For smaller symbol sizes, the same memory can be used to compensate a proportionally larger

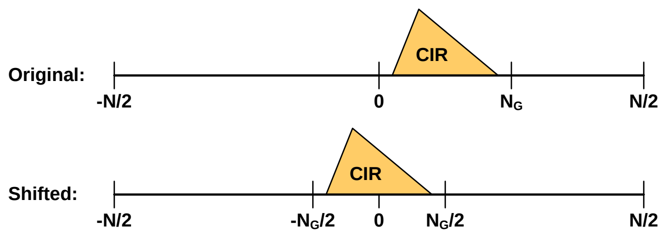


Figure 5.12.: Centering the CTF spectrum before estimation

group delay, which allows the use of an estimation filter with more taps at no extra cost. For instance, a linear interpolation filter in the 8K mode can be replaced with an 8-tap Wiener estimator in the 2K mode, which can easily achieve an estimation gain of at least 6 dB and maximally 9 dB, especially at the relatively low bandwidths (¡ 1/32) at which it operates.

5.3.4. Estimation along the Carrier Axis

The CTF variations along the k axis result from the multi-path profile of the channel. It has been shown in **Subsection 2.1.5** that the complex CTF (channel transfer function) in frequency domain is the Fourier transform of the complex CIR (channel impulse response) in time domain. According to the duality principle, if the CTF is considered as the signal of interest, the multipath power delay profile is its power spectral density. Intuitively, the shorter the CIR, the smoother the CTF. If the CIR consists of two peaks with a spacing of L samples, the CTF is a sine with L periods in the whole symbol.

In order to avoid ISI (inter-symbol interferences), the guard interval N_G must be chosen larger than the CIR length L and the timing synchronization circuit has to ensure that the entire CIR falls between 0 and N_G, as shown in **Figure 5.12**. Since all CIR samples have positive delays, the spectrum of the CTF will be asymmetrical. Thus, the bandwidth seen by the estimator is $2N_G/N$ in the worst case. In order to reduce the requirements for the estimator, the spectrum can be easily shifted, as shown in **Figure 5.12**, the most straightforward solution being to shift with exactly $N_G/2$, which reduces the worst-case bandwidth by two.

The amount of shifting can be either fixed, with half of the guard interval length N_G, or adaptive, depending on the actual CIR profile. Adaptive shifting is especially attractive when the CIR is significantly shorter than the guard interval, i.e. $L << N_G$. **Figure 5.13** shows an example where $L = N_G/4$ and $N_G = N/4$. With adaptive shifting the estimator sees a much lower signal bandwidth compared with fixed shifting with $N_G/2$, which can be used to improve the estimator gain. Adaptive shifting can be also achieved by using fixed shifting and designing the timing synchronization block so that the CIR is centered within the $[0; N_G]$ interval.

Frequency shift is performed by incrementally rotating the complex samples, the rotation angle/phase is generated by a phase accumulator, as shown in **Figure 5.14**. If the symbol size N

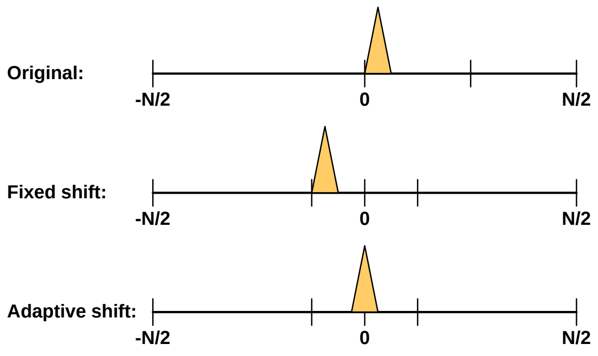


Figure 5.13.: Fixed and adaptive spectrum shifting

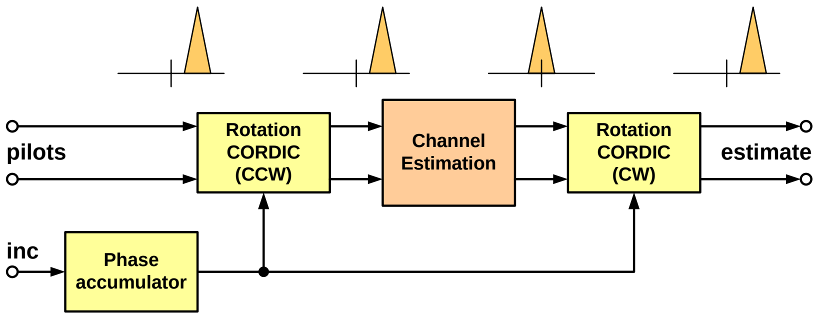


Figure 5.14.: Channel estimation including spectrum shifting

is a power of two, i.e. $N = 2^B$, the phase accumulator is exactly B-bit wide and the increment is the number of samples with which the spectrum is shifted. After estimation, exactly the same rotation needs to be performed in the opposite direction in order to shift the spectrum back to its original position

As it contains two CORDIC rotators, the solution is relatively complex. Considering that the channel estimate needs to be converted to polar coordinate for performing channel correction, the circuit can be simplified by processing the CTF in polar coordinates from the beginning. Thus, the configuration in **Figure 5.14** can be simplified as in **Figure 5.15**. Note that the amplitude is not affected by the frequency shift and therefore no additional processing is needed. The incremental rotation is applied to the phase part only, and consists in a simple subtraction of the accumulator phase. However, the phase has to be unwrapped before estimation to avoid wrong results, since the phase takes only values in range $[-\pi; \pi)$. The unwrapping process relies on the fact that the phase jumps cannot exceed π, according to the sampling theorem. The unwrapping is trivial and consists only in discarding a number of MSB's.

It must be mentioned here that channel correction is usually not implemented using a complex division, as this would be much too complex. Instead, both the complex incoming data and the channel estimate are first converted from cartesian to polar complex coordinates using CORDIC blocks in vectoring mode. Thus, the complex division can be implemented using only a real division stage and a subtractor, as shown in **Figure 5.16**. If the channel estimation is

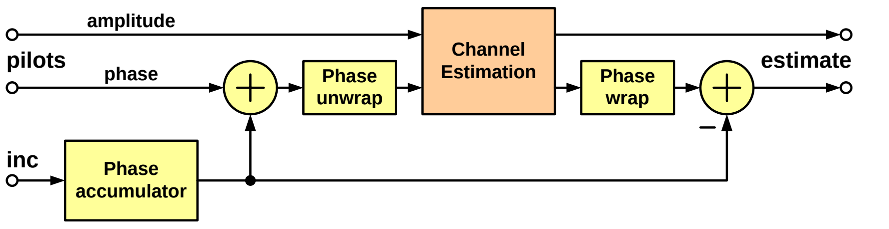


Figure 5.15.: Channel estimation using polar complex representation

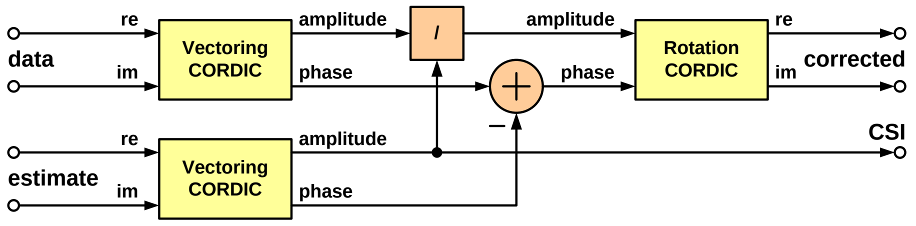


Figure 5.16.: Implementation solution for the channel correction stage

performed using a polar representation as well, the corresponding CORDIC stage can be saved. Moreover, an additional benefit is that the CSI (channel state information) is obtained for free, since the CSI can be approximated with the absolute value of the CTF estimate.

For OFDM systems that support different guard interval lengths, the incremental rotation for spectrum shifting needs to be changed accordingly, i.e. exactly with $N_G/2$. One example is the DVB-T standard, which specifies four possible guard bands, of relative length 1/4, 1/8, 1/16, and 1/32. The pilot spacing D_k must never exceed the inverse thereof, so that the sampling theorem along the k axis is fulfilled. In DVB-T for instance, this condition is fulfilled for all guard bands since $D_k = 3$. It can be also shown that the maximum allowed discrete bandwidth has in this case the following values: 3/4, 3/8, 3/16, 3/32. One needs to bear in mind that the spectrum is usually asymmetrical. Knowledge of the maximum bandwidth allows the optimization of the estimator, so that higher gains can be obtained even when no runtime bandwidth estimate is available.

Potentially, very high estimation gains can be achieved, especially for lower bandwidths and a large number of estimation filter taps. For each bandwidth there is a limit of the achievable gain, as shown in **Figure 5.40**, after which any increase in the number of taps will not improve the estimator gain. This maximum gain, however, increases for lower bandwidths. Practical filter sizes are 8 and 16, since they can achieve very good estimation gains with reasonable complexity. The design of such a filter is treated in more detail in **Section 5.4**, together with a performance analysis under various conditions.

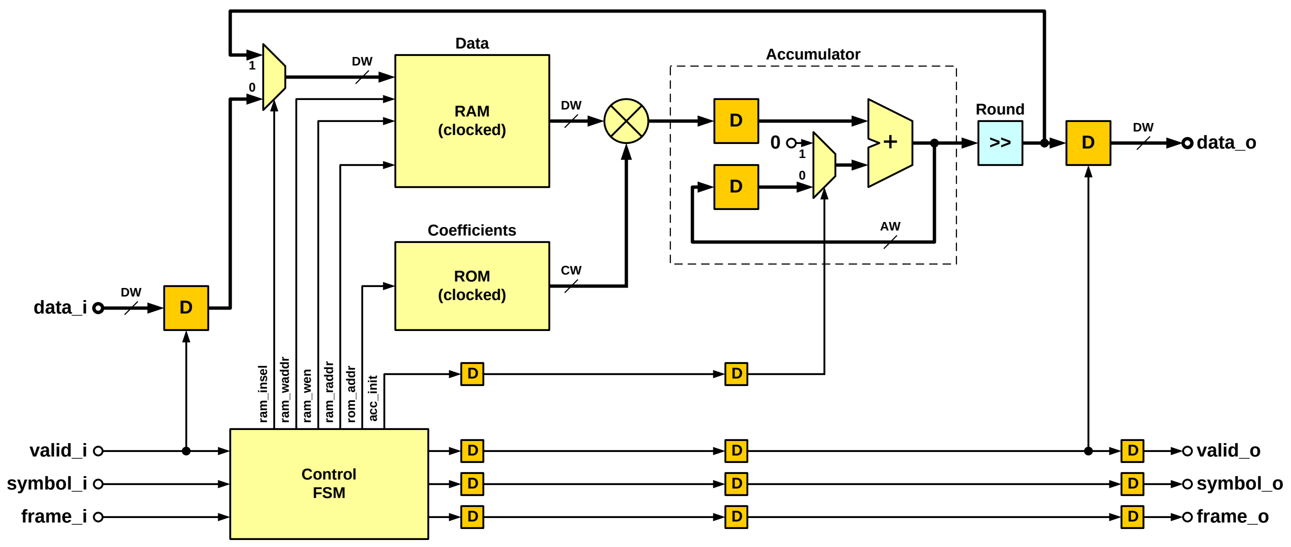


Figure 5.17.: Generic sequential channel estimation architecture

5.3.5. Channel Estimation Architecture

In **Figure 5.17** we present a generic sequential architecture for channel estimation. Note here the striking similarity with the sequential bi-quad filter architecture presented in **Figure 4.21**. The architecture consists of a single MAC (multiply accumulate) unit controlled by a state machine and performs interpolation (estimation) first along the time axis, then along the frequency axis, as shown in **Figure 5.9**, using polyphase filters. The filter coefficients are stored in a ROM, which can be also a RAM programmable by a controller. The pilots and the intermediate interpolation results are stored in RAM. The control state machine consists of counters that keep track of the pilot positions and arithmetical blocks for computing the correct memory addresses of data and coefficients.

The estimator block receives only the active OFDM cells, i.e. belonging to active subcarriers. The protocol, shown in **Figure 5.18**, is unidirectional and consists of three additional signals: `valid`, which qualifies cell data as valid and marks the availability of new data, `symbol`, which marks the first cell in the symbol, and `frame`, which marks the first cell of the first symbol in an OFDM frame.

In **Figure 5.18**, for illustration purposes, a new cell is transferred every four clock cycles, with a dead interval at the end of the symbol, there are four active cells per symbol, and a frame consists of three symbols. In a real case, the number of active cells per symbol is of the form $nD_k + 1$, i.e. the first and the last active subcarriers contain scattered pilots, and the number of symbols per frame is of the form nD_s, so that the pilot pattern is not disturbed at frame boundaries.

The proposed channel estimation in **Figure 5.17** generates a channel estimate for every input cell. If the cell is a pilot, it is stored in memory and used for estimation, first along the time

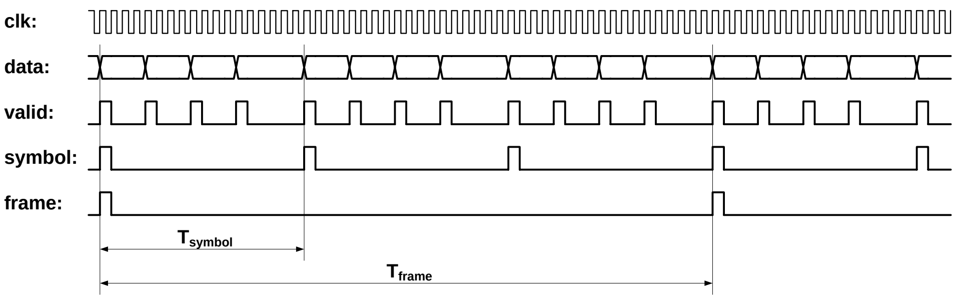


Figure 5.18.: Protocol for transferring OFDM cell data between modules

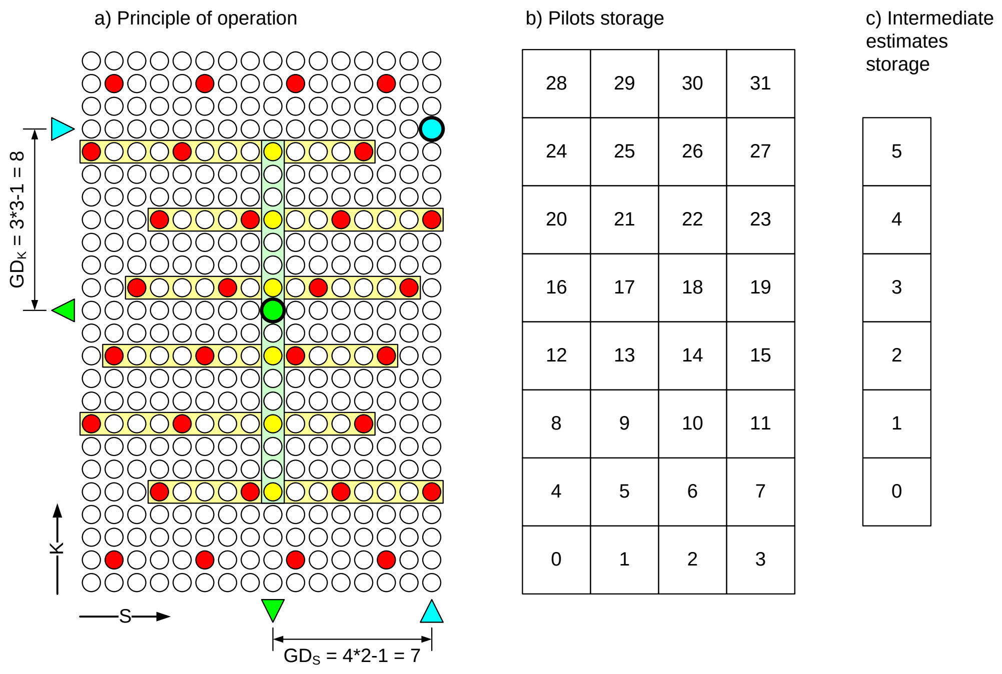


Figure 5.19.: Operation and memory map for the proposed channel estimation architecture

axis s. This estimation is performed only for cells that belong to subcarriers which contain scattered pilots, i.e. every D_k cells. The intermediate estimation results are stored back into memory, in another region than the pilots.

Figure 5.19 shows the principle of operation of the channel estimator for the following case: pilot grid with $D_s = 4$ and $D_k = 3$, and number of filter taps $M_s = 4$ and $M_k = 6$ respectively. The coordinates of the current input cell are marked with triangles pointed towards the cell, whereas the current output estimate is marked with triangles pointed towards outside. Thus, the group delay in both directions can be clearly seen. Along the s axis, the group delay is $GD_s = M_s D_s/2 - 1$ symbols. Along the k axis, the group delay is $GD_k = M_k D_k/2 - 1$ cells. For each subcarrier with pilots, a number of M_s pilots have to be stored. The number of intermediate estimates to be stored is always M_k, since the final estimation is only performed for a single symbol. Memory locations are accessed in both cases as circular buffers. The overall

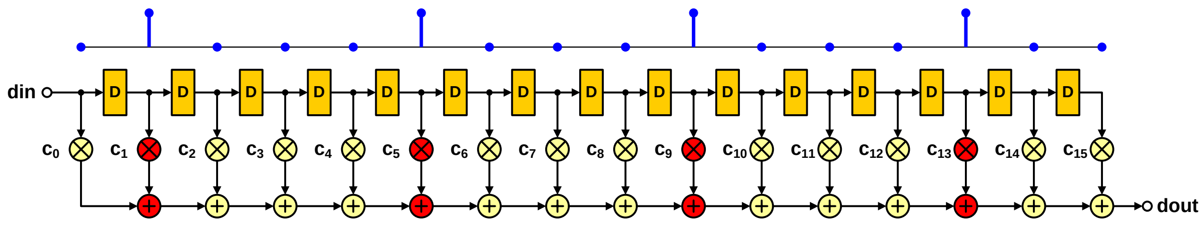


Figure 5.20.: Regular FIR interpolation filter of length 16

memory requirements are relatively low compared with the group delay compensation, where entire symbols have to be stored.

It is also important to determine the minimum required clock frequency for such a sequential architecture. We need to bear in mind that the clock frequency should be a multiple of the fundamental sampling rate of the OFDM system, which leads to a simplified implementation. For the subcarriers with pilots, the number of clocks needed to compute a channel estimate is $M_s + M_k$, which is the worst case timing, since both estimations have to be performed. This is 10 clock cycles in our example, which for a sample period of 64/7 MHz in the case of 8 MHz channels leads to a clock frequency of about 90 MHz. This is easily attainable in any technology, including FPGA.

Since the proposed channel estimation architecture uses separable polyphase filters for interpolation/estimation in both directions, we also present here the principles of polyphase decomposition. At the first glance, one would be tempted to use the textbook principle for interpolation, i.e. inserting zeros followed by filtering using a low-pass FIR filter. Such a FIR filter is shown in **Figure 5.20**, together with the samples with zeros in between. The filter size is 16, for an interpolation factor of 4. One can readily seen that three fourths of the multiplications are with zero and do not contribute to the final result. The efficiency is the inverse of the interpolation factor, and thus becomes very low for large interpolation factors.

As an intermediate step towards the elimination of all trivial multiplications, the above FIR filter is decomposed into 4 smaller filters of length 4, one for each interpolation phase. The number of phases is equal with the interpolation factor. Each new input sample is fed to all four filters, while the output is selected through a switch (multiplexor), synchronous with the input switch. So far, only a rearrangement of the elements of the original FIR filter has been performed, with no complexity reduction.

Figure 5.22 shows the first simplification, which consist in discarding the input switch and using a single delay line for all 4 filters. This is possible since the 4 delay lines in **Figure 5.21** have always the same content. Moreover, this enables the input rate to be reduced to the original non-upsampled rate, while the output is still operated with the higher upsampled rate.

In **Figure 5.22**, the multipliers work with the slow input rate, therefore they are idle for 3 cycles out of 4. This leads to a further simplification, shown in **Figure 5.23**, whereby the same 4 multipliers are shared for all 4 phases. For each multiplier there is a coefficient multiplexor,

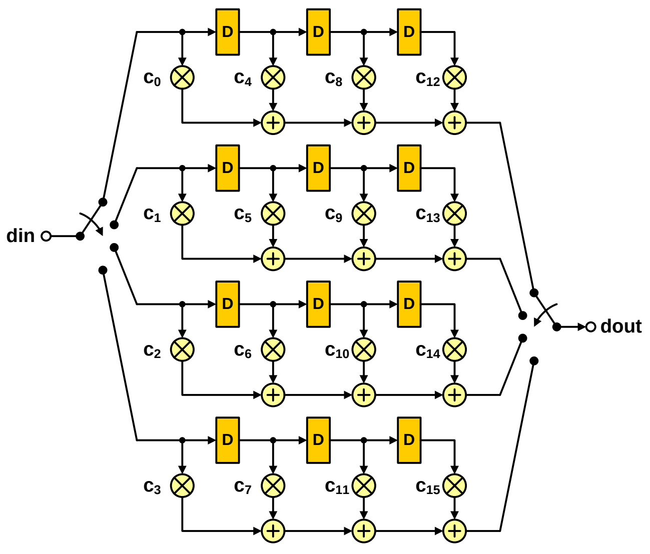


Figure 5.21.: Polyphase filter bank decomposition

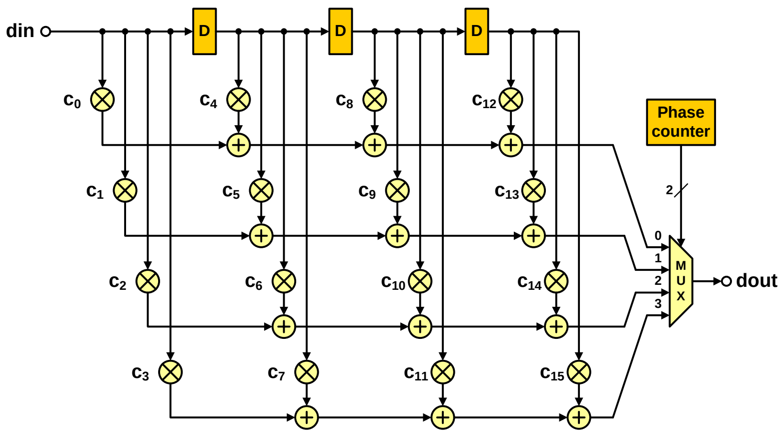


Figure 5.22.: Polyphase filter with separate multipliers

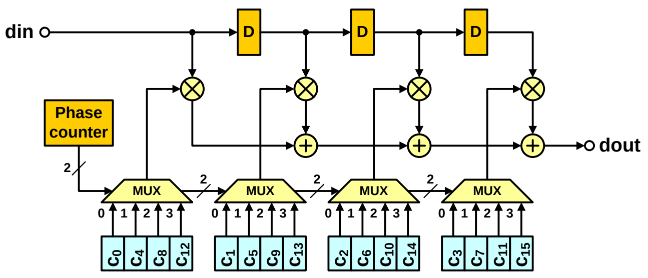

din
D
D
D
Phase counter
2
MUX
0 1 2 3
c_0 c_4 c_8 c_{12}
c_1 c_5 c_9 c_{13}
c_2 c_6 c_{10} c_{14}
c_3 c_7 c_{11} c_{15}
dout

Figure 5.23.: Polyphase filter with shared multipliers

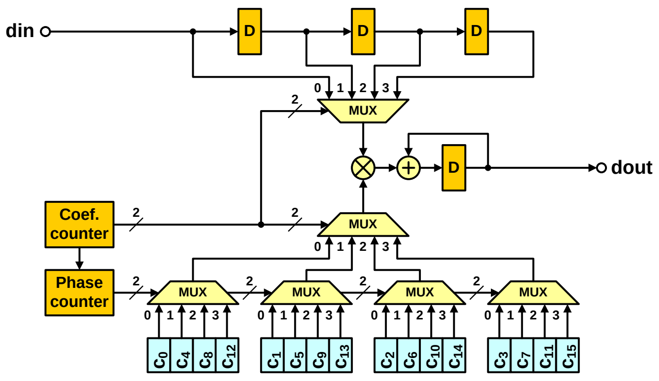

din
D
D
D
0 1 2 3
MUX
2
dout
Coef. counter
Phase counter
MUX
c_0 c_4 c_8 c_{12}
c_1 c_5 c_9 c_{13}
c_2 c_6 c_{10} c_{14}
c_3 c_7 c_{11} c_{15}

Figure 5.24.: Polyphase filter with a single MAC unit

which selects the appropriate coefficient for the desired phase. All coefficient multiplexor are synchronous, being controlled by the same phase from a phase counter.

If the output upsampled rate is lower than the clock rate, a single MAC unit can be used for performing all multiplications and additions, as shown in **Figure 5.24**. The only constraint is that the ratio between the clock rate and the output sample rate must be larger than the interpolation filter size.

5.4. Algorithms for One-Dimensional Estimation and Interpolation

One-dimensional estimation consists of two main operations: noise reduction of the pilot samples followed by interpolation between these samples. The straightforward approach would be to employ a Wiener filter followed by an interpolation filter. However, this solution requires a large number of MAC operations, and also has a higher group delay. The latter issue is especially inconvenient for estimation along the time axis, where any group delay has to be compensated by an equal delay of all data samples, which requires extra memory. The solution is to use an asymmetrical Wiener filters which perform both noise reduction and interpolation in one step.

This section starts with a short presentation of the Wiener filter theory and their performance for different application scenarios. The second subsection introduces various interpolation types and presents their performance, with emphasis on the signal-match interpolator, as presented in [66]. Finally, we introduce the Wiener estimation filters and analyze their performance in various scenarios. It is interesting to mention that when the noise is zero, the Wiener estimator is reduced to a signal-matched interpolation filter. Likewise, when the interpolation factor is one, the estimator is reduced to a regular Wiener filter.

5.4.1. Wiener Filters

Problem Formulation and Its Solution

We have a signal $x(n)$ with known statistics, affected by additive white Gaussian noise $w(n)$ with known variance σ_w^2. Thus, the received signal has the expression:

$$y(n) = x(n) + w(n) \tag{5.5}$$

We need to design a FIR filter of size M that estimates the signal optimally, in the sense of minimizing the mean-square error between the estimate $\tilde{x}(n)$ and the transmitted signal $x(n)$, i.e. $E\{|\tilde{x}(n) - x(n)|^2\}$. Designing the filter means to determine the filter coefficients c_m, for $m = 0 \ldots M-1$. According to the well-known Wiener theory, the optimal coefficients can be determined by solving the Wiener-Hopf equation:

$$\mathbf{R_{yy}c} = \mathbf{r_{xy}} \tag{5.6}$$

where $\mathbf{R_{yy}}$ is the $M \times M$ autocorrelation matrix of the noisy signal $y(n)$, $\mathbf{c}$ is the column vector with the M coefficients to be determined, and $\mathbf{r_{xy}}$ is the correlation vector between the original signal $x(n)$ and the one affected by noise $y(n)$. We also denote with $\mathbf{r_{yy}}$ the autocorrelation vector of $y(n)$.

$$\mathbf{r_{yy}} = \Big[r_{yy}(0), r_{yy}(1), \ldots, r_{yy}(M-1)\Big]^{\mathrm{T}} \tag{5.7}$$

$$\mathbf{r_{xy}} = \Big[r_{xy}(0), r_{xy}(1), \ldots, r_{xy}(M-1)\Big]^{\mathrm{T}} \tag{5.8}$$

The autocorrelation matrix $\mathbf{R_{yy}}$ is a Toeplitz matrix obtained from the $\mathbf{r_{yy}}$ vector, each diagonal having all its elements identical. For real signals, the matrix is symmetrical with respect to the main diagonal. In its expanded form, the Wiener-Hopf equation has the following expression:

$$\begin{bmatrix} r_{yy}(0) & r_{yy}(1) & r_{yy}(2) & \cdots & r_{yy}(M-1) \\ r_{yy}^*(1) & r_{yy}(0) & r_{yy}(1) & \ddots & \vdots \\ r_{yy}^*(2) & r_{yy}^*(1) & r_{yy}(0) & \ddots & r_{yy}(2) \\ \vdots & \ddots & \ddots & \ddots & r_{yy}(1) \\ r_{yy}^*(M-1) & \cdots & r_{yy}^*(2) & r_{yy}^*(1) & r_{yy}(0) \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{M-1} \end{bmatrix} = \begin{bmatrix} r_{xy}(0) \\ r_{xy}(1) \\ r_{xy}(2) \\ \vdots \\ r_{xy}(M-1) \end{bmatrix} \tag{5.9}$$

The above matrix equation represents a linear system of M equations with the M filter coefficients as unknowns. For a certain $\mathbf{r_{xx}}$ and $\mathbf{r_{xy}}$, the Wiener theory guarantees that the resulting filter is optimal for those conditions.

In our case, the channel only introduces noise, without affecting the signal spectrum through frequency selectivity ((**5.5**)). The necessary information is represented by the signal statistics, i.e. autocorrelation function or power density spectrum, and the variance of the additive white Gaussian noise. Besides, the noise is uncorrelated with the signal, therefore $\mathbf{r_{xw}}$ is almost null. The autocorrelation vector $\mathbf{r_{yy}}$ and the correlation vector $\mathbf{r_{xy}}$ have the following expressions:

$$\mathbf{r_{yy}} = \mathbf{r_{xx}} + \sigma_w^2[1, 0, \ldots, 0]^{\mathrm{T}} \tag{5.10}$$

$$\mathbf{r_{xy}} = \mathbf{r_{xx}} + \underbrace{\mathbf{r_{xw}}}_{\approx 0} \tag{5.11}$$

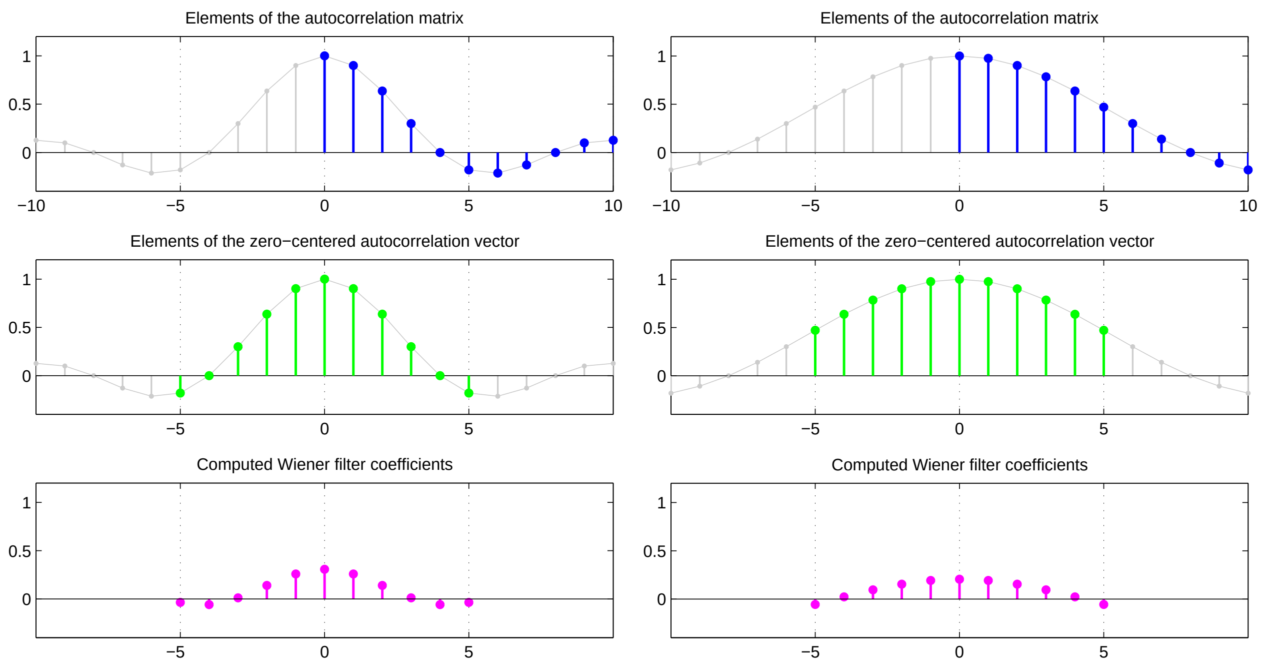
where $\mathbf{r_{xx}}$ is the autocorrelation of the original signal $x(n)$ and σ_w^2 is the AWGN variance.

Figure 5.25 shows the elements of the correlation matrix and the correlation vector, as well as the computed filter coefficients for a 11-tap symmetrical Wiener filter, considering a band-limited signal with discrete bandwidth of 1/4 (**Figure 5.25a**) and 1/8 (**Figure 5.25b**). For both filters an SNR = 10 dB has been considered. It can be seen that the autocorrelation of the band-limited signal is a *sinc* function, whose main lobe has a width of 8 and 16 respectively.

Such a filter is optimal in that it minimizes the MSE of the estimate. However, this is true only if the actual signal and channel statistics match those for which it has been designed. In practice, this requires perfect knowledge of the signal autocorrelation and noise variance, which is usually not possible. Robust filters should not exhibit too much performance degradation when there is a parameter mismatch.

Matched Wiener Filter Performance

First, we investigate the performance of the Wiener filter when both the bandwidth and the SNR are matched. Performance is expressed in terms of MSE of the filtered signal as a function of the matched bandwidth and SNR, with the filter size M as the only parameter. In the



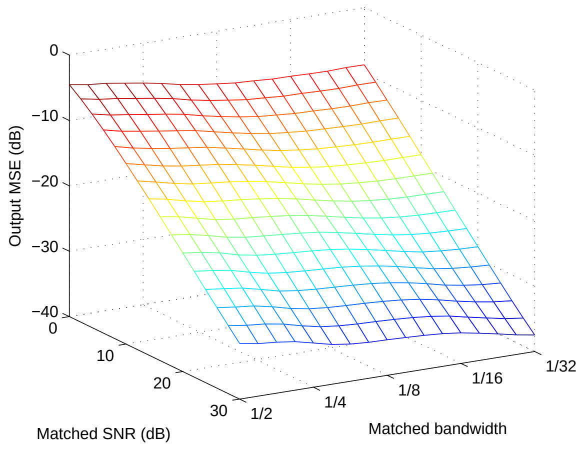
(a) Bandwidth = 1/4 (b) Bandwidth = 1/8

Figure 5.25.: Coefficients computation for a 11-tap symmetrical Wiener filter

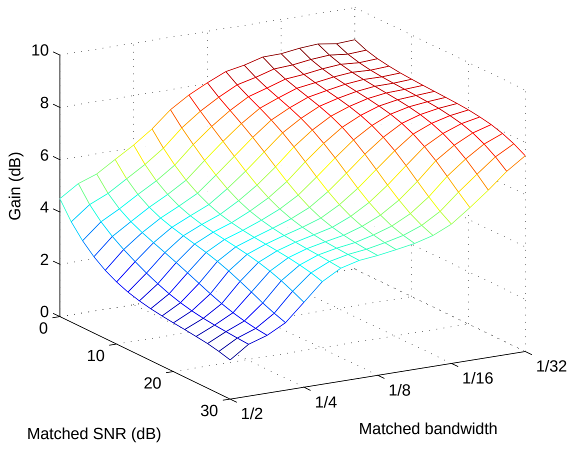
following we consider only symmetrical filters with an odd number of taps. **Figure 5.26a** and **Figure 5.26b** show the MSE and the SNR gain of a 7-tap matched Wiener filter. The discrete bandwidth covers four octaves, from 1/64 to 1/2, while SNR covers three decades, from 0 to 30 dB, the former being considered relative to the Nyquist limit of $f_s/2$.

The only independent parameter of the filter is its length or the number of taps. **Figure 5.27** shows the gain of a matched Wiener filter with sizes between 3 and 11 for an SNR of 10 dB and 20 dB respectively. A few conclusions can be drawn. First, the filter gain increases with decreasing bandwidths and saturates to a value which depends on the filter length. The more taps the filter has, the higher the maximum gain. Second, the maximum gain depends on the filter size alone. Only the bandwidth at which the gain saturates depends on the SNR. Moreover, the gain itself tends to saturate with the number of filter taps.

In order to evaluate the maximum achievable filter gain as a function of the number of taps, we performed a simulation where we chose a bandwidth of 1/256 and an SNR of 10 dB. The gain vs. filter length plots are shown in **Figure 5.28** for filter lengths between 3 and 23. It can be clearly seen that the maximum gain depends linearly on the filter size. Doubling the number of taps results in a doubling of the filter gain. However, as the number of taps increases the maximum gain is achieved for increasingly lower bandwidths.



(a) Output MSE



(b) SNR gain

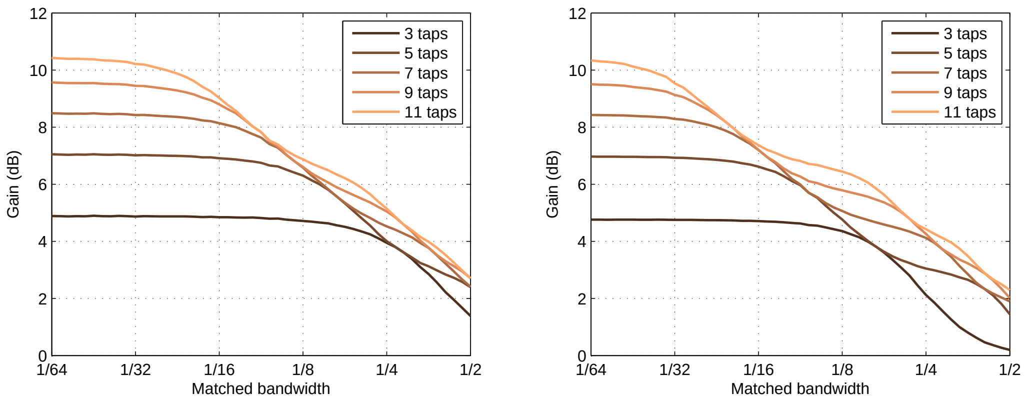
Figure 5.26.: Output MSE and SNR gain for a 7-tap matched Wiener filter

Mismatched Wiener Filter Performance

In the following, we want to investigate how the filter performs if the actual signal statistics and noise do not match those for which the filter has been designed. Like in the previous analysis, we consider a band-limited random signal affected by additive white Gaussian noise. First, we analyze the influence of the bandwidth mismatch on the filter performance, considering the SNR is matched and treating the SNR and the filter size as parameters. In order to gain an understanding of how the bandwidth mismatch affects the filter performance we simulated the dependence of the MSE and filter gain on the designed filter bandwidth and the actual signal bandwidth. The results are presented in **Figure 5.29** for a 7-tap filter and an SNR of 10 dB.

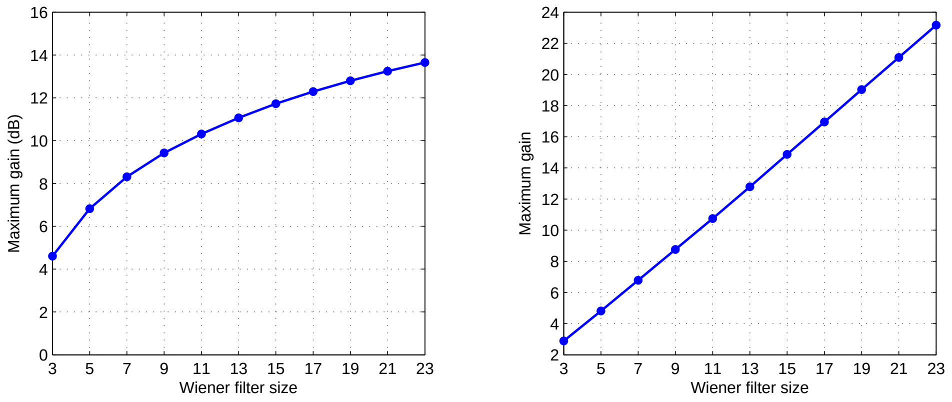
It can be seen that the minimum achievable MSE decreases with decreasing bandwidths. For actual signal bandwidths below the designed value, the MSE remains almost constant and the potential for even smaller MSE remains unexploited. However, if the designed bandwidth is exceeded, a rapid degradation of the MSE can be observed. Therefore, when designing a Wiener filter it is better to choose a larger bandwidth instead of a smaller one. In order to evaluate how the bandwidth mismatch affects the filter gain for various filter sizes and SNR's, we analyzed a combination of two filter sizes (5 and 9 taps) and two SNR's (10 and 20 dB). The gain plots are shown in **Figure 5.30** for all four cases.

For a given filter size, the mismatch problem is aggravated at higher SNR's. The same deviation from the designed bandwidth towards higher values will result in a higher gain loss. Moreover, for the same designed bandwidth the gain is lower at higher SNR's. In the case of channel estimation for OFDM, an estimate gain of 6 dB is sufficient. Therefore, a rule of a thumb which provides good results in most cases is that the Wiener designed bandwidth should lie between 1/4 and 1/8. Corner cases are the stationary reception, when the gain can be maximized using



(a) SNR = 10 dB (b) SNR = 20 dB

Figure 5.27.: Gain vs. bandwidth for a matched Wiener filter



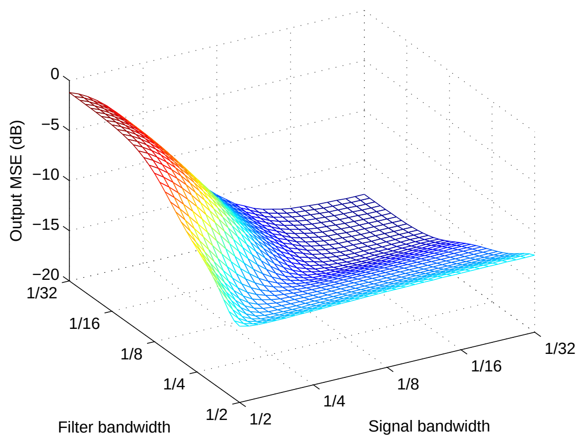
(a) Gain in dB (b) Absolute gain

Figure 5.28.: Gain of a matched Wiener filter vs. filter size

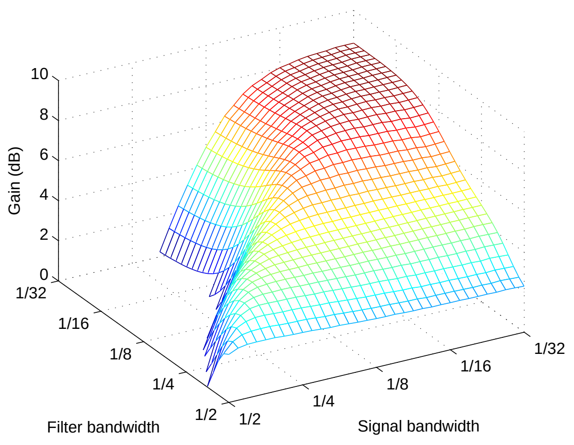
a long filter designed with a very low bandwidth, or very high Doppler spreads, when the designed bandwidth must be maximized, at the expense of a reduced estimation gain. It is impossible to achieve a high estimate gain at a high Doppler spread.

Having investigated the effects of the bandwidth mismatch on the filter gain, we go on to investigate the effects of SNR mismatch as well. The bandwidth is assumed perfectly matched. First, we plot the dependence of the gain on the designed and actual SNR at a given bandwidth. **Figure 5.31** shows the 2D plots of the output MSE and SNR gain for a bandwidth of 1/8, when the actual and the designed SNR's vary from 0 to 40 dB.

In order to better evaluate the effects of the SNR mismatch on the filter gain, we analyze the dependence of the gain on the actual signal SNR with the filter SNR as a parameter. **Figure 5.32** shows two such plots for a 7-tap Wiener filter, considering a matched bandwidth

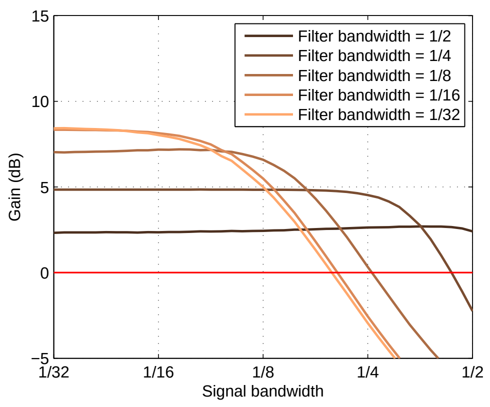


(a) Output MSE

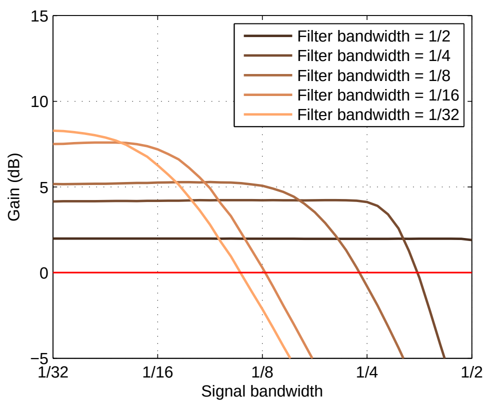


(b) SNR gain

Figure 5.29.: Output MSE and SNR gain for a 7-tap Wiener filter with bandwidth mismatch

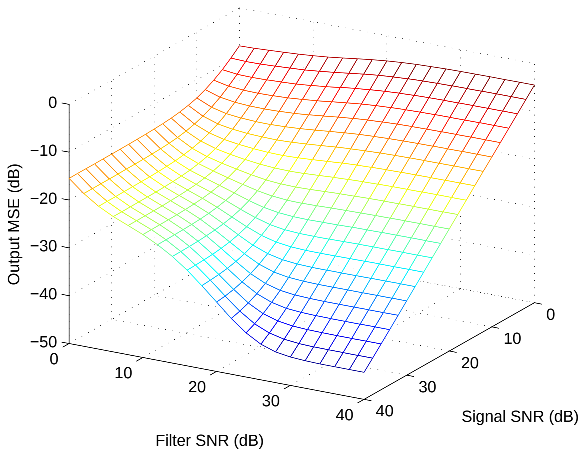


(a) SNR = 10 dB

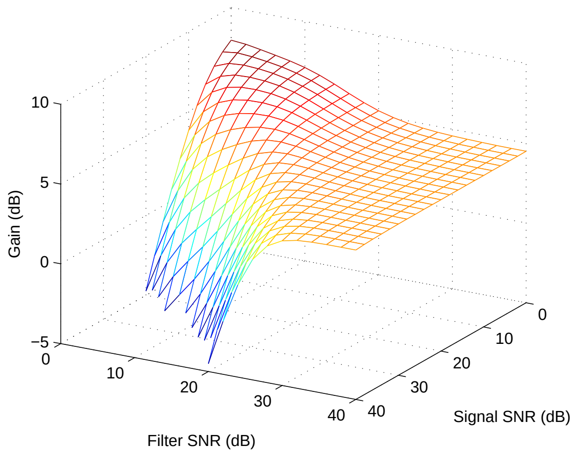


(b) SNR = 20 dB

Figure 5.30.: Gain of a 7-tap Wiener filter with bandwidth mismatch



(a) Output MSE



(b) SNR gain

Figure 5.31.: Output MSE and SNR gain for a 7-tap Wiener filter with SNR mismatch

of 1/4 and 1/16 respectively. From these plots, we can conclude that for typical SNR's in range 10 ... 20 dB, a very good trade-off is to choose a filter SNR of 10 dB. If an SNR estimator is available, the best coefficients set can be selected dynamically during operation.

5.4.2. Signal-Matched Interpolation

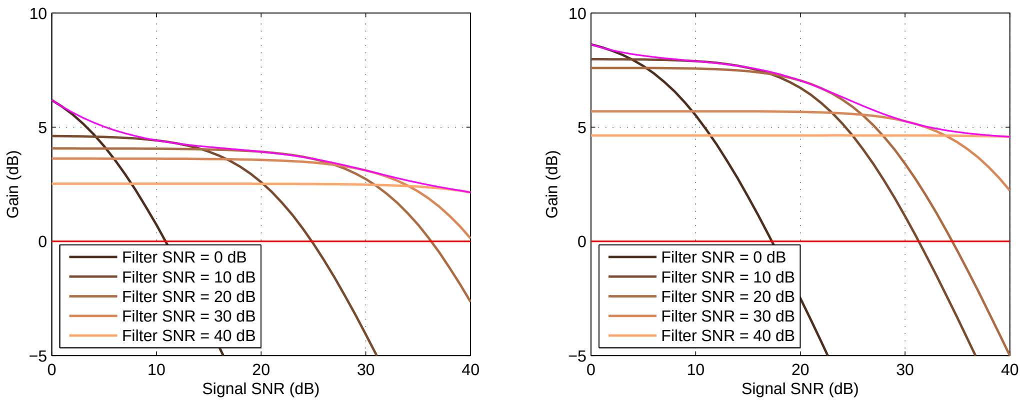
We consider a sequence $y(n)$ which has $L-1$ consecutive zeros every L samples, i.e. samples are non-zero for $n = mL$, where $m \in \mathbb{Z}$ and L is an integer called interpolation factor. Practically, $y(n)$ is a sampled version of an unknown band-limited sequence $x(n)$

$$y(n) = \begin{cases} x(n) & \forall n = mL \\ 0 & \forall n \neq mL \end{cases} \tag{5.12}$$

The task of the interpolation filter is to reconstruct the original sequence $x(n)$ from the samples of $y(n)$. The filter must be designed so that the output sequence $\tilde{x}(n)$ approximates $x(n)$ as close as possible in the sense of minimizing the mean-squared error $E\{|\tilde{x}(n) - x(n)|^2\}$. We denote with $\Delta x(n)$ the interpolation error $\tilde{x}(n) - x(n)$. The block diagram of the system and the associated signals are shown in **Figure 5.33**.

We assume a symmetrical FIR interpolation filter with a length $N = 2ML + 1$. The output will be

$$\tilde{x}(n) = \sum_{k=-ML}^{+ML} h(k)y(n-k) \tag{5.13}$$



(a) Bandwidth = 1/4 (b) Bandwidth = 1/16

Figure 5.32.: Gain of a 7-tap Wiener filter with SNR mismatch

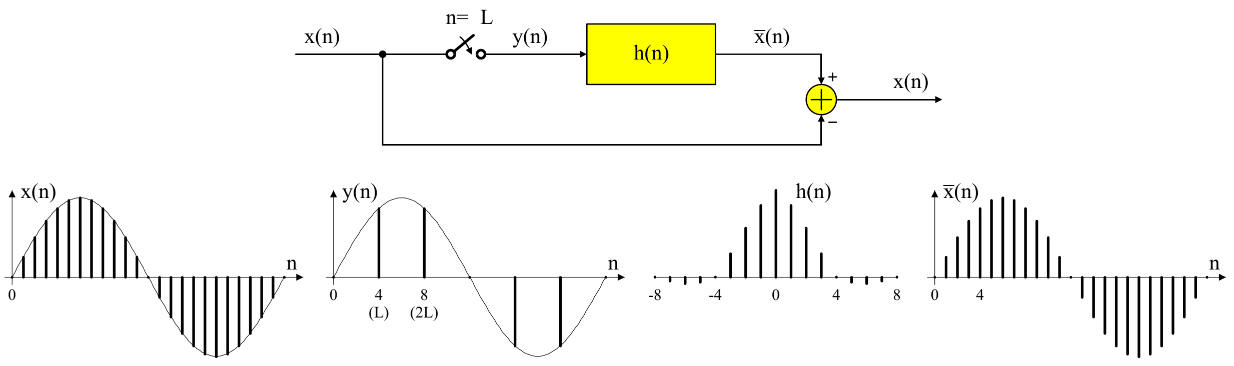


Figure 5.33.: Block diagram of the system and the associated signals.

where $h(k)$ is the impulse response or the coefficient vector of the filter. In this form, the filter performs $2ML$ multiply-accumulate operations per output cycle. As only $1/L$ of the samples to be filtered are non-zero, most multiplications are trivial, which results in a very inefficient solution if (**5.13**) is implemented directly.

Using a polyphase approach will increase the computation efficiency significantly by eliminating the trivial multiplications. In the following, we show how the polyphase decomposition is performed and how the optimal coefficients can be determined when we know the spectrum of the initial sequence $x(n)$. In doing this, we follow the approach presented in [66].

Knowing that $y(n) = 0$ for $n \neq mL$ and considering the first and the last coefficient are zero, $h(-ML) = h(ML) = 0$, we get for $n = mL + \phi$ with $\phi = 0, \ldots, L-1$

$$\tilde{x}(mL+\phi) = \sum_{m_1=-M}^{M-1} h(m_1L+\phi)y((m-m_1)L) \tag{5.14}$$

For each ϕ the output $\tilde{x}(n)$ is obtained using $2M$ samples of $y(n)$ and $h(n)$. For simplification we introduce the following notations:

$$x_\phi(n) = \begin{cases} x(n) & n = mL+\phi \\ 0 & n \neq mL+\phi \end{cases} \tag{5.15a}$$

$$\tilde{x}_\phi(n) = \begin{cases} \tilde{x}(n) & n = mL+\phi \\ 0 & n \neq mL+\phi \end{cases} \tag{5.15b}$$

where m is an integer and $\phi = 0, 1, \ldots, L-1$ can be thought as a phase index.

Using the polyphase decomposition, $x(n)$ and $\tilde{x}(n)$ can be written as:

$$x(n) = \sum_{\phi=0}^{L-1} x_\phi(n) \tag{5.16a}$$

$$\tilde{x}(n) = \sum_{\phi=0}^{L-1} \tilde{x}_\phi(n) \tag{5.16b}$$

We define now the corresponding L error sequences as:

$$\Delta x_\phi(n) = \tilde{x}_\phi(n) - x_\phi(n) \tag{5.17}$$

Likewise, the overall error can be written as:

$$\Delta x(n) = \sum_{\phi=0}^{L-1} \Delta x_\phi(n) \tag{5.18}$$

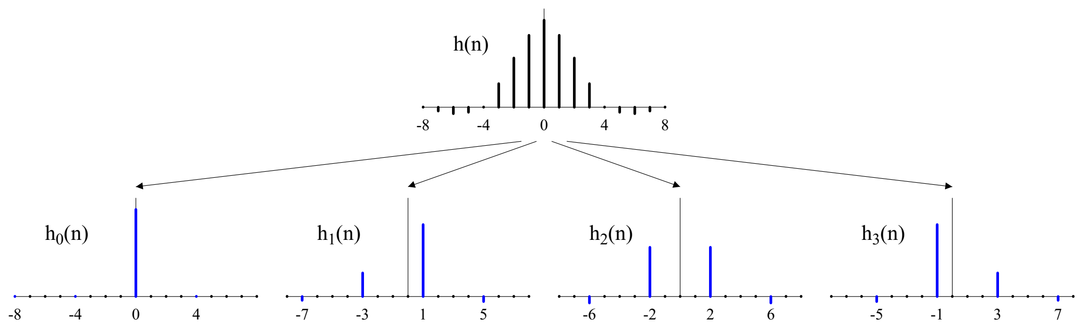


Figure 5.34.: Polyphase decomposition of the impulse response for $L = 4$ and $M = 2$.

Each error sequence $\Delta x_\phi(n)$ depends only on a subset of the $h(n)$, hereafter denoted with $h_\phi(n)$. Thus, the minimum of $E\{|\Delta x(n)|^2\}$ is obtained by minimizing each $E\{|\Delta x_\phi(n)|^2\}$ separately. The decomposition of $h(n)$ into its L polyphase subsets $h_\phi(n)$ of length $2M$ is defined by the equation below. **Figure 5.34** shows such a decomposition for $L = 4$ and $M = 2$.

$$h_\phi(n) = \begin{cases} h(n) & n = mL + \phi \\ 0 & n \neq mL + \phi \end{cases} \tag{5.19}$$

Thus, the final impulse response is the sum of its polyphase components.

$$h(n) = \sum_{\phi=0}^{L-1} h_\phi(n) \tag{5.20}$$

Using a polyphase decomposition, the design of an interpolating filter of degree $2ML$ with impulse response $h(n)$ has been split up into designing L subfilters of degree $2M$ with impulse response $h_\phi(n)$. Each subfilter calculates an estimate of $x(mL+\phi)$ using the samples of $y(n) = x(mL)$, as it can be seen in **Figure 5.35**. Besides reducing the order of the problem, the polyphase decomposition led to a more efficient implementation since all L subfilters operate now at the non-interpolated lower sampling rate. Thus, the number of MAC operations is reduced by L.

In order to minimize $E\{|\Delta x_\phi|\}$, we write the sequence Δx_ϕ explicitly for those values of k where the samples are non-zero. According to (**5.14**) and (**5.15**) we get

$$\Delta x_\phi(mL+\phi) = \sum_{m_1=-M}^{M-1} h_\phi(mL+\phi) y((m-m_1)L) - x_\phi(mL+\phi) \tag{5.21}$$

$$= \sum_{m_1=-M}^{M-1} h_\phi(mL+\phi) x((m-m_1)L) - x(mL+\phi) \tag{5.22}$$

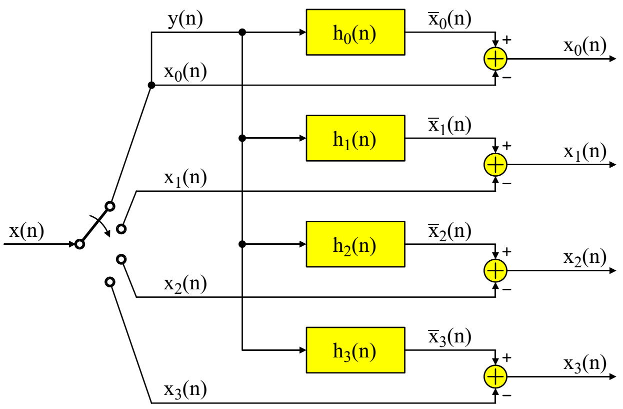


Figure 5.35.: Polyphase interpolator error model for $L = 4$ and $M = 2$.

It can be readily seen that $\Delta x_0(n) = 0$ if

$$h_0(n) = \begin{cases} 1 & k = 0 \\ 0 & k \neq 0 \end{cases} \tag{5.23}$$

that is, the first filter leaves the signal unchanged.

The minimization of the other $E\{|\Delta x_\phi|^2\}$ depends on the statistics of the input sequence $x(n)$ and has been presented in detail in [66]. If we denote with $r(n)$ the autocorrelation function of the original signal $x(n)$, the condition for minimizing $E\{|\Delta x_\phi|^2\}$ resembles the Wiener-Hopf equation in the Wiener filtering theory:

$$\mathbf{R}\mathbf{h}_\phi = \mathbf{r}_\phi \tag{5.24}$$

where

$$\mathbf{M} = \begin{bmatrix} r(0) & r(L) & r(2L) & \cdots & r((2M-1)L) \\ r(L) & r(0) & r(L) & \ddots & \vdots \\ r(2L) & r(L) & r(0) & \ddots & r(2L) \\ \vdots & \ddots & \ddots & \ddots & r(L) \\ r((2M-1)L) & \cdots & r(2L) & r(L) & r(0) \end{bmatrix} \tag{5.25}$$

$$\mathbf{h}_\phi = \Big[h(-ML+\phi), h(-(M-1)L+\phi), \ldots, h(\phi), \ldots, h((M-1)L+\phi)\Big]^{\mathrm{T}} \tag{5.26}$$

$$\mathbf{r}_\phi = \Big[r(-ML+\phi), r(-(M-1)L+\phi), \ldots, r(\phi), \ldots, r((M-1)L+\phi)\Big]^{\mathrm{T}} \tag{5.27}$$

For $M = 4$, the equation becomes:

$$\begin{bmatrix} r(0) & r(L) & r(2L) & r(3L) \\ r(L) & r(0) & r(L) & r(2L) \\ r(2L) & r(L) & r(0) & r(L) \\ r(3L) & r(2L) & r(L) & r(0) \end{bmatrix} \begin{bmatrix} h(-2L+\phi) \\ h(-L+\phi) \\ h(\phi) \\ h(L+\phi) \end{bmatrix} = \begin{bmatrix} r(-2L+\phi) \\ r(-L+\phi) \\ r(\phi) \\ r(L+\phi) \end{bmatrix} \tag{5.28}$$

The matrix $\mathbf{R}$ is the Toeplitz matrix of the sampled autocorrelation function $h(mL)$ and does not depend on ϕ. Since the signal is real, the autocorrelation function is symmetrical, i.e. $r(n) = r(-n)$. Equation (**5.24**) leads to:

$$\mathbf{h}_\phi = \mathbf{R}^{-1}\mathbf{r}_\phi \tag{5.29}$$

The calculation of $h(n)$ involves a one-time inversion of the $\mathbf{R}$ matrix of size $2L \times 2L$ and $L-1$ multiplications of the inverted matrix with the $\mathbf{r}_\phi$ vector. Further simplifications are possible by exploiting the symmetry of the autocorrelation function [66]. However, since the coefficient calculation is only done once at design time, these simplifications are not important in most cases.

It suffices to say that for the impulse responses of the L subsystems, the following symmetry relationship holds:

$$h_\phi(n) = h_{L-\phi}(-n) \tag{5.30}$$

which reduces the number of vectors $h_\phi(n)$ to be calculated by a factor of 2. Besides, it results from (**5.20**) that the impulse response of the interpolating filter is symmetrical, $h(n) = h(-n)$, i.e. the ideal interpolator has linear phase.

Interpolation Performance Analysis

In investigating the performance of the signal-match interpolator, we adopt the same strategy as in the case of the Wiener filter, considering a random signal with a band-limited rectangular spectrum. The interpolation filter itself is also matched to such a signal, only the bandwidths being different. Unlike for the Wiener filter, there is no SNR to be optimized. Instead, an additional parameter is the interpolation factor, which in our analysis is fixed to four. The following analysis is limited to three octaves, from 1/16 to 1/2, as shown in **Figure 5.36a** where the interpolation MSE is plotted for a 4-tap filter. Note that the first diagonal in **Figure 5.36a** is the MSE in the case of a perfect match, which is also shown in **Figure 5.36b** as a thinner line that marks the MSE floor.

It can be seen that if the actual signal bandwidth is larger than the designed filter bandwidth, the error tends to be some 10 dB above that of a matched filter. For lower signal bandwidths, the error remains essentially constant. The interpolation error is low enough for all designed

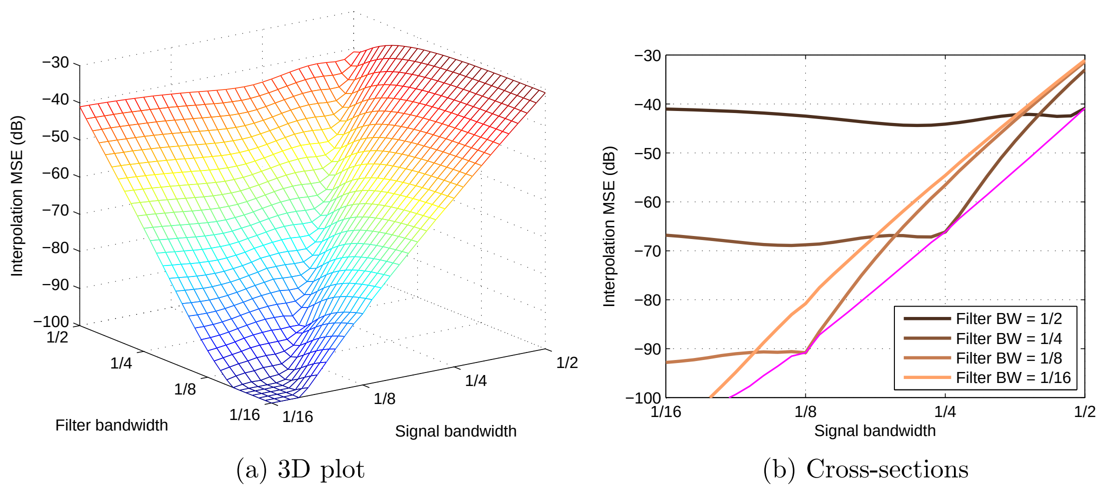


Figure 5.36.: Interpolation error in the presence of bandwidth mismatch for a signal-matched interpolation filter with four taps

bandwidths. Even for a designed bandwidth of 1/2, the interpolation error lies below -40 dB, which ensures that the performance is not affected even in the case of a 64-QAM modulation. A low MSE can be modeled as an additive Gaussian noise, a value of -40 dB being equivalent with an SNR of 40 dB. According to **Figure 5.4**, this will not affect the performance for any QAM modulation type.

It is therefore readily apparent that adapting the filter coefficients to the signal bandwidth does not bring any performance gain since the error floor is very low for any set of coefficients. That is why, if only interpolation needs to be performed, it is recommended to use an interpolation filter that does not depend on the signal statistics. For this purpose, we propose here the use of Lagrange interpolation filters. In **Figure 5.37** we present their interpolation errors for various input signal bandwidths, i.e. before interpolation.

The particular case of the linear interpolation (2 taps) is also the degenerated Lagrange polynomial of first degree. Besides the Lagrange interpolation filter with 2, 4, and 6 taps, we also consider the zero-order hold interpolation. Although the latter has the worst performance of all existing interpolators, it has one important advantage, the absence of a group delay. This can make its choice appealing for interpolation along the time axis. When the Doppler spread is low, the bandwidth of the interpolated signal is also low and the interpolation error is kept to an acceptable level. Since the filter does not introduce group delay, no additional storage is required for group delay compensation.

5.4.3. Wiener Interpolation

Intuitively, the estimation process can be better understood as a sequence of two operations. First, the channel is estimated at the pilot positions using a classical Wiener filter for smoothing

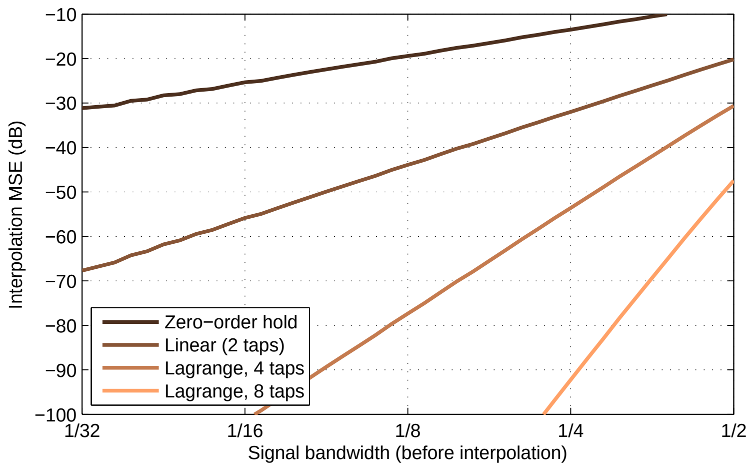


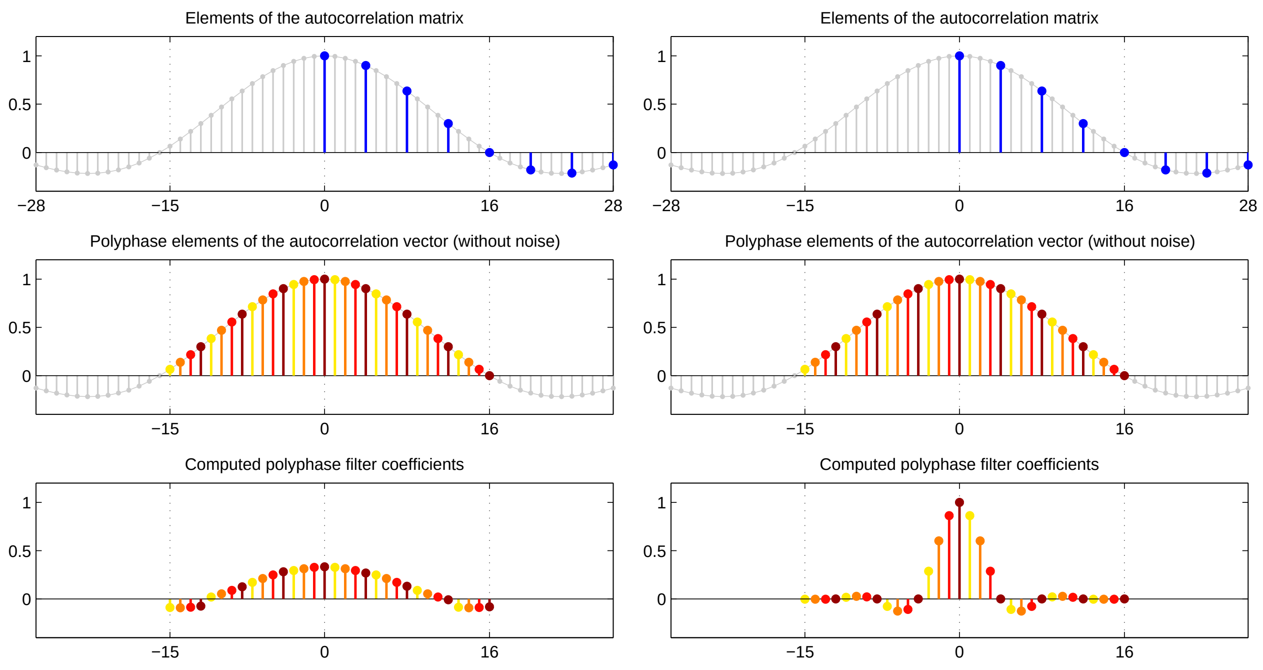
Figure 5.37.: Interpolation error for various interpolation filters

(noise reduction). Second, the estimates for all other positions are obtained through interpolation. However, this approach involves a large number of computations and introduces a significant group delay. The latter should be kept to a minimum, especially for estimation along the time axis, since all data symbols need to be delayed accordingly.

The optimum strategy is to combine the two operations int a single one. The solution to this problem can be derived by noting that the coefficients for the Wiener and the interpolation filters result as solutions of a Wiener-Hopf equation. Compare for this purpose (**5.6**) for the Wiener filter with (**5.24**) for the signal-matched interpolation filter. This is actually the reason why we went on to present the theory behind the signal-matched interpolation filter, not especially its interpolation performance, which is not essential.

Now (**5.24**) can be simply expanded by adding the noise variance σ_w^2 to the expression of $\mathbf{r}_\phi$ in (**5.27**), the same like in (**5.10**), for each polyphase set. In the limit case when the noise variance is zero, the Wiener interpolator is reduced to a signal-matched interpolator. The computed polyphase coefficients for an 8-tap filter are shown in **Figure 5.38a** for an SNR of 10 dB and in **Figure 5.38b** for a noiseless signal (SNR $= \infty$). In this example an interpolation factor of 4 and a signal bandwidth of 1/4 have been considered.

The MSE of such a Wiener interpolation filter, optimized for a bandwidth of 1/4, is shown in **Figure 5.39a** as a function of the SNR, together with the gain of Wiener filters optimized for 1/2 and 1/8 respectively. The thin diagonal designates the unity (0 dB) gain line. The SNR gain of the filter, which is the distance to this line, must always be positive. Otherwise, if the MSE vs. SNR characteristic crosses the unity gain line, the filter actually worsens the SNR and it would be better to use a regular interpolator in this case. The MSE characteristic of such an interpolator is shown in **Figure 5.39b** for the same SNR range. As expected, simple interpolation achieves no SNR gain and the MSE curve is very close to the unity gain diagonal. Actually, there is a slight gain, which is due to the fact that the interpolation is a low-pass

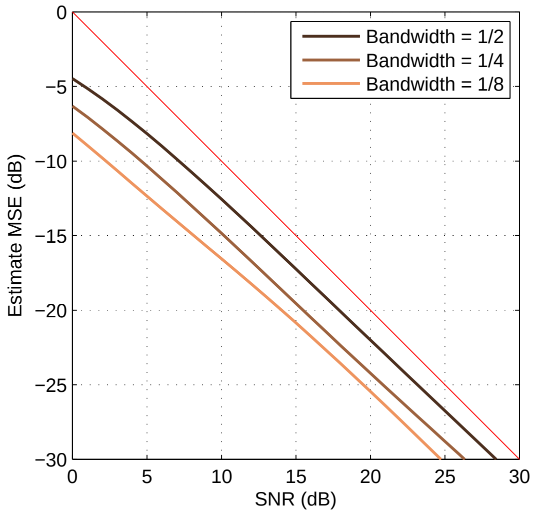


(a) Wiener interpolation, SNR = 10 dB (b) Signal-matched interpolation (SNR = ∞)

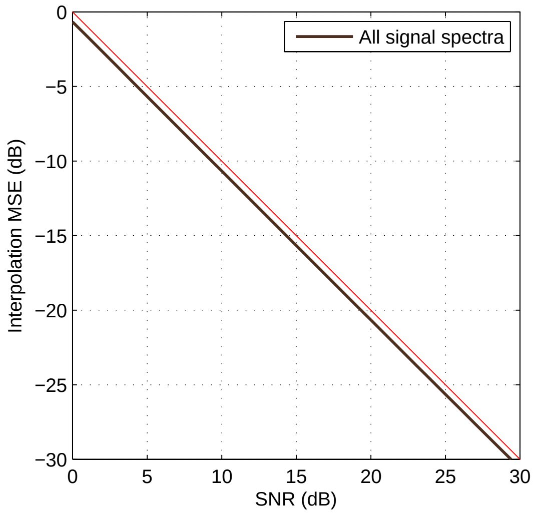
Figure 5.38.: Coefficients computation for an 8-tap Wiener interpolator

filtering which slightly attenuates the high-frequency noise components.

In order to be able to choose the appropriate filter length for a specific scenario, we also present the maximum gain of a Wiener interpolation filter as a function of the number of taps, at various bandwidths. The results are shown in **Figure 5.40** for all even number of taps from 4 to 32. It can be clearly seen that the gain saturates with the number of taps, with the saturation value depending on the bandwidth.



(a) Wiener interpolation/estimation



(b) Interpolation only

Figure 5.39.: Estimation MSE vs. received SNR

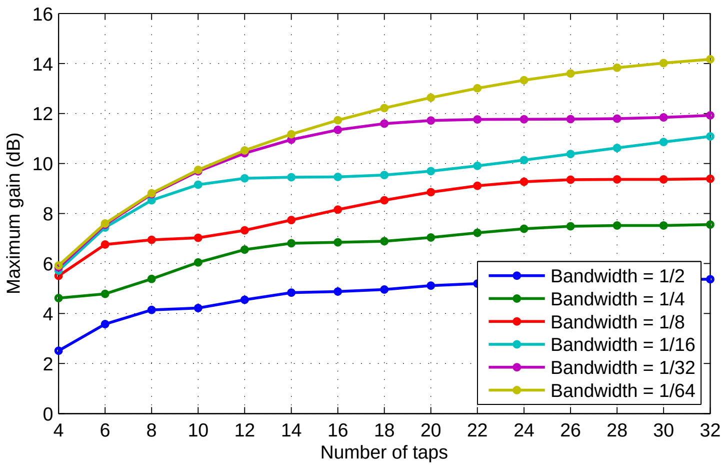


Figure 5.40.: Dependence of the filter gain on the filter size