

INTRODUCTION

to

ADAPTIVE SIGNAL PROCESSING

Marc Moonen

Department of Electrical Engineering · ESAT/SISTA

K.U. Leuven, Leuven, Belgium

marc.moonen@esat.kuleuven.ac.be

Chapter 1

Introduction

1.1 Linear filtering and adaptive filters

Filters are devices that are used in a variety of applications, often with very different aims. For example, a filter may be used, to reduce the effect of additive noise or interference contained in a given signal so that the useful signal component can be discerned more effectively in the filter output. Much of the available theory deals with *linear filters*, where the filter output is a (possibly time-varying) linear function of the filter input. There are basically two distinct theoretical approaches to the design of such filters :

1. The ‘*classical*’ approach is aimed at designing frequency-selective filters such as lowpass/bandpass/notch filters *etc.* For a noise reduction application, for example, it is based on knowledge of the gross spectral contents of both the useful signal and the noise components. It is applicable mainly when the signal and noise occupy clearly different frequency bands. Such classical filter design will not be treated here.
2. ‘*Optimal filter design*’, on the other hand, is based on optimization theory, where the filter is designed to be “best” (in some sense). If the signal and noise are viewed as stochastic processes, based on their (assumed available) statistical parameters, an optimal filter is designed that, for example, minimizes the effects of the noise at the filter output according to some statistical criterion. This is mostly based on minimizing the mean-square value of the difference between the actual filter output and some desired output, as illustrated in **Figure 1.1**. It may seem odd that the theory requires a desired signal - for if such a thing is available why bother with the filter? Suffice it to say that it is usually possible to obtain a signal that, whilst it is not what really is required, is sufficient for the purpose of controlling the adaptation process. Examples of this will be given further on - albeit in the context of ‘adaptive filtering’ where we do not assume knowledge of the stochastic parameters but which is based on a very similar idea. The theory of optimal filter design dates back to the work of Wiener in 1942 and Kolmogorov in 1939. The resulting solution is often referred to as the *Wiener filter*¹. Adaptive

¹Basic Wiener filter theory will be reviewed in Chapter 2.

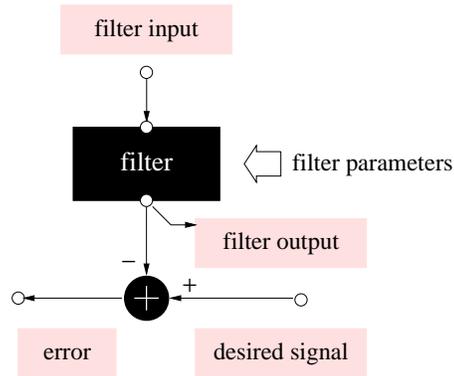


Figure 1.1: Prototype Wiener filtering scheme

filter theory largely builds on this work.

Wiener filter design may be termed ‘a priori’ design, as it is based on a priori statistical information. It is important to see, however, that a Wiener filter is only optimal when the statistics of the signals at hand truly match the a priori information on which the filter design was based. Even worse, when such a priori information is not available, which is usually the case, it is not possible to design a Wiener filter in the first place. Furthermore the signal and/or noise characteristics are often *nonstationary*. In this case, the statistical parameters vary with time and, although the Wiener theory still applies, it is usually very difficult to apply it in practice. An alternative method, then, is to use an *adaptive filter*, that in a sense is *self-designing*.

An adaptive filter has an *adaptation algorithm*, that is meant to monitor the environment and vary the filter transfer function accordingly. The algorithm starts from a set of initial conditions, that may correspond to complete ignorance about the environment, and, based in the actual signals received, attempts to find the optimum filter design. In a stationary environment, the filter is expected to *converge*, to the Wiener filter. In a nonstationary environment, the filter is expected to *track* time variations and vary its filter coefficients accordingly.

As a result, there is no such thing as a unique optimal solution to the adaptive filtering problem. Adaptive filters have to do without a priori statistical information, but instead usually have to draw all their information from only one given realization of the process, *i.e.* one sequence of time samples. Nevertheless, there are then many options as to what information is extracted, how it is gathered, and how it is used in the algorithm. In the stationary case, for example, ergodicity may be invoked to compute the signal statistics through time averaging. Time averaging obviously is no longer a useful tool in a nonstationary environment. Given only one realization of the process, the adaptation algorithm will have to operate with ‘instantaneous’ estimates of the signal statistics. Again, such estimates may be obtained in various ways. Therefore, we have a ‘*kit of tools*’ rather than a unique solution. This results in a variety of algorithms. Each alternative algorithm offers desirable features of its own.

A wealth of adaptive filtering algorithms have been developed in the literature. A selection will be presented in this book. The choice of one solution over another is determined not only by *convergence* and *tracking* properties, but also by *numerical sta-*

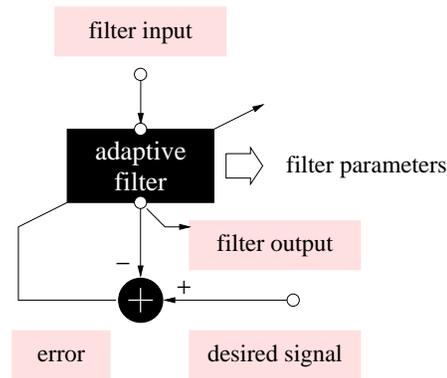


Figure 1.2: Prototype adaptive filtering scheme

bility, *accuracy* and *robustness*, as well as *computational complexity* and, sometimes, amenability to *hardware implementation*. The latter property refers to the structure of the algorithm, where, for example, modularity and inherent parallelism are desirable in view of VLSI implementation. We will come back to such aspects further on.

1.2 Prototype adaptive filtering scheme

The prototype adaptive filtering scheme is depicted in **Figure 1.2**, which is clearly similar to the Wiener filtering scheme of Figure 1.1. The basic operation now involves two processes :

1. a *filtering* process, which produces an output signal in response to a given input signal.
2. an *adaptation* process, which aims to adjust the filter parameters (filter transfer function) to the (possibly time-varying) environment. The adaptation is steered by an error signal that indicates how well the filter output matches some desired response. Examples are given in the next section. Often, the (average) square value of the error signal is used as the optimization criterion.

Depending on the application, either the filter parameters, the filter output or the error signal may be of interest.

An adaptive filter may be implemented as an analog or a digital component. Here, we will only consider *digital filters*. When processing analog signals, the adaptive filter is then preceded by analog-to-digital convertors². Similarly, a digital-to-analog conversion may be added whenever an analog output signal is needed, see **Figure 1.3**. In most subsequent figures, the A/D and D/A blocks will be omitted, for the sake of conciseness.

As will be explained in subsequent chapters, a pragmatic choice is to use a *FIR filter* (finite impulse response filter), where the filter output is formed as a linear combination

²For convenience, we assume that the analog-to-digital convertors include any signal conditioning such as anti-aliasing filters.

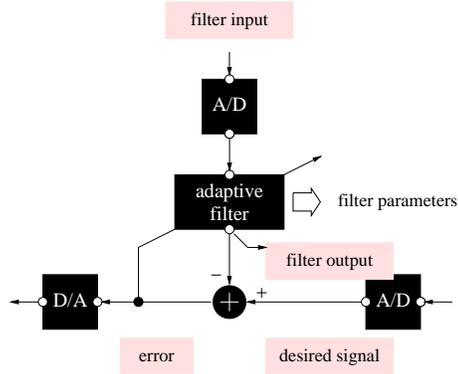


Figure 1.3: Prototype adaptive digital filtering scheme with A/D and D/A

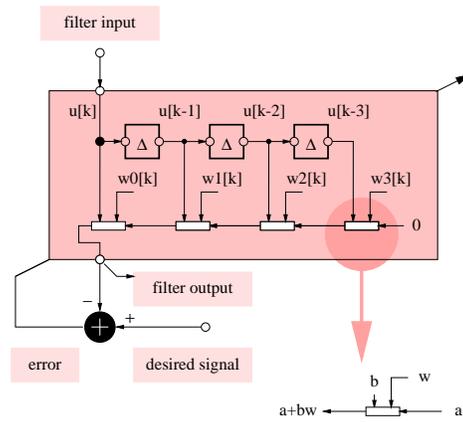


Figure 1.4: Prototype adaptive FIR filtering scheme

of delayed input samples, *i.e.*

$$y_k = w_0 \cdot u_k + w_1 \cdot u_{k-1} + \dots + w_{N-1} \cdot u_{k-N+1}$$

with y_k the filter output at time k , u_k the filter input at time k , and w_i , $i = 0 \dots N-1$ the filter weights (to be ‘adapted’), see **Figure 1.4**. This choice leads to tractable mathematics and fairly simple algorithms. In particular, the optimization problem can be made to have a cost function with a single turning point (unimodal). Thus any algorithm that locates a minima, say, is guaranteed to have located the global minimum. Furthermore the resulting filter is unconditionally stable³. The generalization to adaptive *IIR filters* (infinite impulse response filters), **Figure 1.5**, is nontrivial, for it leads to stability problems as well as non-unimodal optimization problems. Here we concentrate on adaptive FIR filter algorithms, although a brief account of adaptive IIR filters is to be found in Chapter 9. Finally, non-linear filters such as Volterra filters and neural network type filters will not be treated here, although it should be noted that some of the machinery presented here carries over to these cases, too.

³That is the filtering process is stable. Note however that the adaptation process can go unstable. This is discussed later.

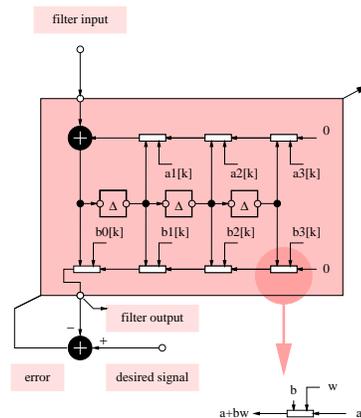


Figure 1.5: Prototype adaptive IIR filtering scheme

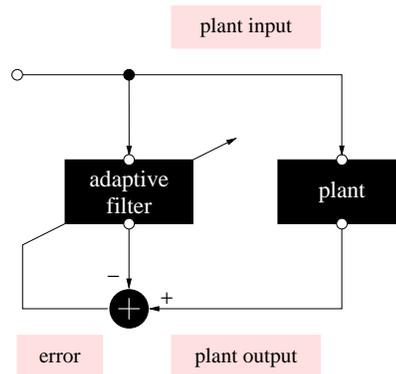


Figure 1.6: System identification scheme

1.3 Adaptive filtering applications

Adaptive filters have been successfully applied in a variety of fields, often quite different in nature. The essential difference arises in the manner in which the desired response is obtained (see figure 1.2). In the sequel, a selection of applications is given. These can be grouped under different headings, such as *identification*, *inverse modeling*, *interference cancellation*, etc. We do not intend to give a clear classification here, for it would be rather artificial (as will be seen).

1.3.1 Identification applications

It is instructive to start with the system identification application of adaptive filtering (even though this is already away from the noise/interference reduction we mentioned initially). This is because in such cases it is clear what the desired signal is. The general identification mechanism is depicted in **Figure 1.6**. The adaptive filter is used to provide a model that represents the best fit to an unknown plant. The plant and the adaptive filter are driven by the same input. The plant output is the desired filter response. The quantities of interest here are the filter parameters, *i.e.* the model itself. In control ap-

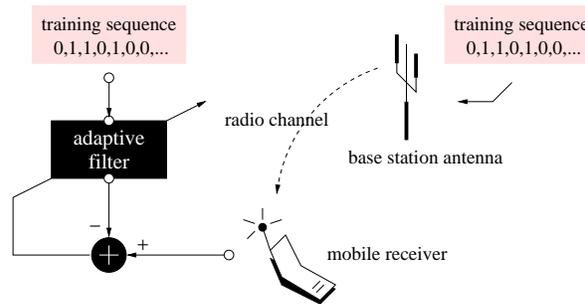


Figure 1.7: Radio channel identification in mobile communications

plications, the plant may be any production process. The adaptive filter then provides a mathematical model which may be used for control system design purposes.

A specific identification application is depicted in **Figure 1.7**, and involves modelling a radio channel. In mobile communications, channel dispersion introduces intersymbol interference, which means that the signal which is received, by a mobile (say), is a filtered version of the original transmitted symbol sequence⁴. To undo the channel distortion, equalization is used at the receiver. One solution to this problem involves the use of a so-called maximum likelihood symbol sequence estimator (Viterbi decoder), which requires an (FIR) model of the channel distortion. Such a model is obtained here by adaptive identification. To this aim, a so-called training sequence is transmitted periodically (interleaved with the information symbols). The training sequence is defined in advance and so is known to the receiver and is used by the adaptive filtering algorithm as the desired signal. The receiver thus *sees* the channel (plant) output and *knows* the corresponding channel (plant) input, hence has everything it needs to do the system identification (cf. Figure 1.6). This channel model is then used in the equalizer. A radio channel is often highly time-varying, so that channel identification has to be repeated continuously. The GSM system for mobile telephony, for example, is based on burst mode communication, where bursts of 148 bits each are transmitted. Each burst has a 26 bit training sequence, which is used to update the channel model. The training bits thus constitute a significant portion (roughly 17%) of the transmitted sequence. Current research is focused on the design of algorithms that operate with fewer training bits, or even without them. The latter class of algorithms is referred to as ‘blind identification/equalization’.

1.3.2 Echo cancellation

A related application is echo cancellation, which brings us back to the original noise/-interference cancellation application. The prototype scheme is depicted in **Figure 1.8**. Here the echo path is the ‘plant’ or ‘channel’ to be identified. The goal is to subtract a synthesized version of the echo from another signal (for example, picked up by a microphone) so that the resulting signal is ‘free of echo’ and really contains only the signal of interest.

A simple example is given in **Figure 1.9** to clarify things. This scheme applies to

⁴Implementation details are omitted here. Transmission involves pulse shaping and modulation. Reception involves demodulation, synchronisation and symbol rate (or higher) sampling.

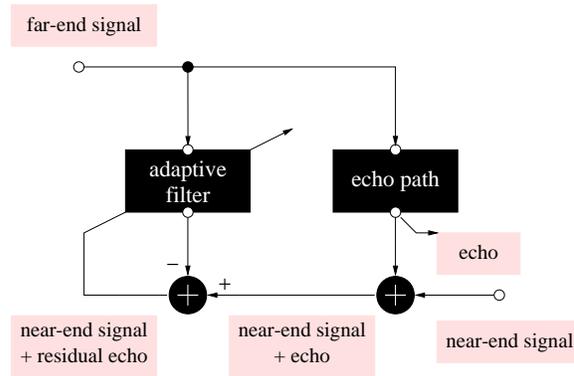


Figure 1.8: Echo cancellation scheme

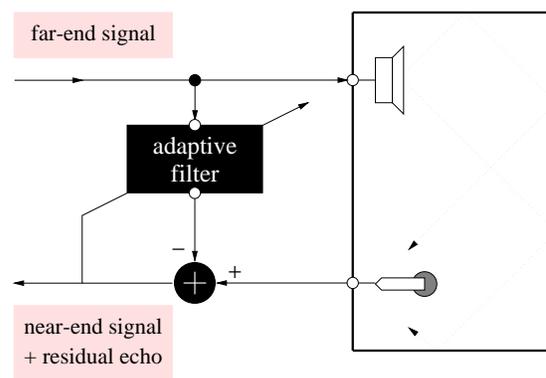


Figure 1.9: Acoustic echo cancellation

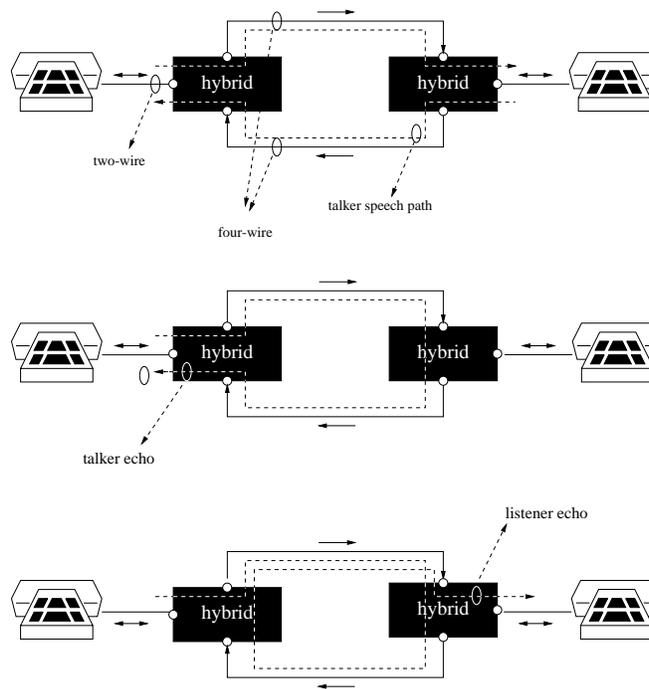


Figure 1.10: Line echo cancellation in a telephone network

hands-free telephony inside a car, or teleconferencing in a conference room. The far-end signal is fed into a loudspeaker (mounted on the dashboard, say, in the hands-free telephony application). The microphone picks up the near-end talker signal as well as an echoed version of the loudspeaker output, filtered by the room acoustics. The desired signal (see Figure 1.8 again) thus consists of the echo (cf. ‘plant output’) as well as the near-end talker signal. This is the main difference with the ‘original’ identification scheme (Figure 1.6). It is assumed that the near-end signal is statistically independent of the far-end signal, which results in the adaptive filter trying to model the echo path as if there were no near-end signal. When this is not the case (*i.e.* nearly always!), the filter weights are adjusted principally in those periods when only the far-end party is talking. In these periods, the error signal is truly a residual echo signal, and hence may indeed be fed back to adjust the filter. Recall that the adaptive filter has an adaptation and a filtering process. The filtering process is run continuously, even in the presence of the near-end talker, to remove the echo. It only the adaptation of the filter weights that gets switched off. Such a scheme clearly requires an extra circuit that can detect when the near-end talker is speaking but we will not discuss this here.

Echo cancellers are also used at other places in telephone networks. A simplified long-distance telephone connection is shown in **Figure 1.10**. The connection contains two-wire segments in the so-called subscriber loops (connection to the central office) and some portion of the local network, as well as a four-wire segment for the long-distance portion. In the four-wire segment, a separate path is provided for each direction of transmission, while in the two-wire segments bidirectional communication is carried on one and the same wire pair. The connection between the two-wire circuit and the four-wire circuit is accomplished by means of a *hybrid transformer* or *hybrid*. Suffice it to say that imperfections such as impedance mismatches cause echoes to be gener-

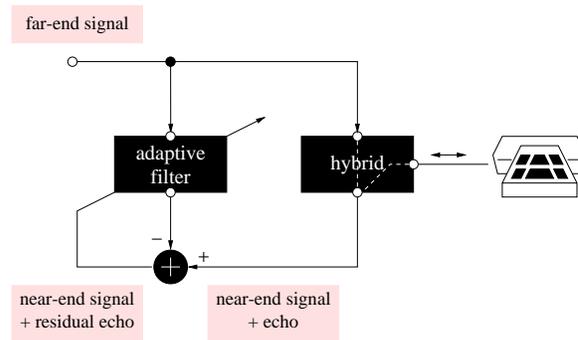


Figure 1.11: Line echo cancellation in a telephone network

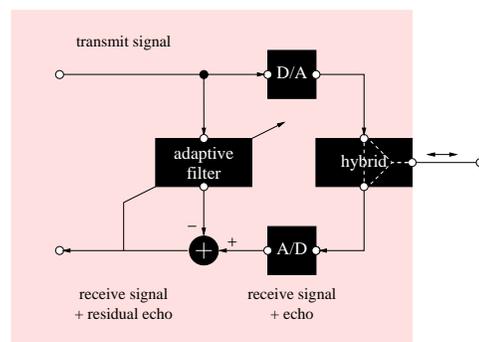


Figure 1.12: Echo cancellation in full-duplex modems

ated in the hybrids. In the figure, two echo mechanisms are shown, namely talker echo and listener echo. Echoes are noticeable when the echo delay is significant (*e.g.*, in a satellite link with 600 ms round-trip delay). The longer the echo delay, the more the echo must be attenuated. The solution is to use an echo canceler, as illustrated in **Figure 1.11**. Its operation is similar to the acoustic echo canceler of Figure 1.9.

It should be noted that the acoustic environment is much more difficult to deal with than the telephone one. While telephone line echo cancellers operate in an albeit a priori unknown but fairly stationary environment, acoustic echo cancellers have to deal with a highly time-varying environment (moving an object changes the acoustic channel). Furthermore, acoustic channels typically have long impulse responses, resulting in adaptive FIR filters with a few thousands of taps. In telephone line echo cancellers, the adaptive filters the number of taps is typically ten times less. This is important because the amount of computation needed in the adaptive filtering algorithm is usually proportional to the number of taps in the filter - and in some cases to the square of this number.

Echo cancellation is applied in a similar fashion in full-duplex digital modems, **Figure 1.12**. ‘Full-duplex’ means simultaneous data transmission and reception at the same ‘full’ speed. Full duplex operation on a two-wire line, as in Figure 1.12, requires the ability to separate a receive signal from the reflection of the transmitted signal, which may be achieved by echo canceling. The operation of this echo canceler is similar to the operation of the line echo canceler.

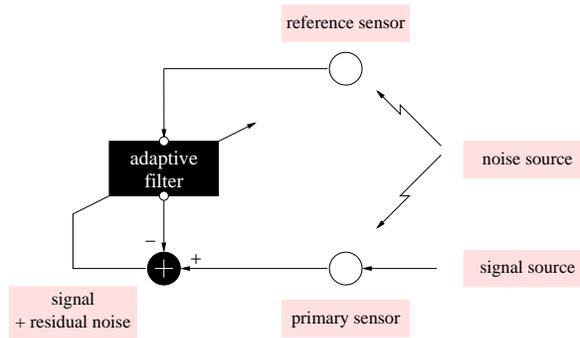


Figure 1.13: Interference cancellation scheme

1.3.3 Interference/noise cancellation

Echo cancellation may be viewed as a special case of the general interference/noise cancellation problem. The scheme is shown in **Figure 1.13**. Roughly speaking, the primary input (desired signal) consists of signal plus noise, and the reference input consists of noise alone. The noise in the reference signal is generally different from the noise in the primary signal (*i.e.* the transfer functions from the noise source to the different sensors are not identical), but it is assumed that the noise in the different sensors are *correlated*. The object is to use the reference sensor signal to reduce the noise in the primary sensor - assuming that the signal and noise are uncorrelated as in the echo cancellation problem. In the identification context, the adaptive filter may be viewed here as a model for the ‘noise-source-to-primary-sensor’ transfer function, times the inverse of the ‘noise-source-to-reference-sensors’ transfer function.

An example is shown in **Figure 1.14**, where the aim is to reduce acoustic noise in a speech recording. The reference microphones are placed in a location where there is sufficient isolation from the source of speech. Acoustic noise reduction is particularly useful when low bit rate coding schemes (*e.g.* LPC) are used for the digital representation of the speech signal. Such coders are very sensitive to the presence of background noise, often leading to unintelligible digitized speech.

Another ‘famous’ application of this scheme is main electricity interference cancellation (*i.e.* removal of 50-60 Hz sinewaves), where the reference signal is taken from a wall outlet.

1.3.4 Inverse modelling

The next class of applications considered here may be termed inverse modelling applications, which is again related to the original identification/modelling application. The general scheme is shown in **Figure 1.15**. The adaptive filter input is now the output of an unknown plant, while the desired signal is equal to the plant input. Note that if the plant is invertible, one can imagine an ‘inverse plant’ (dashed lines) that reconstructs the input from the output signal, so that the scheme is comparable to the identification scheme (Figure 1.6). Note that it may not always be possible to define an inverse plant and thus this scheme may not work. In particular, if the plant has a zero in its z -domain transfer function *outside* the unit circle, the inverse plant will have a pole *outside* the

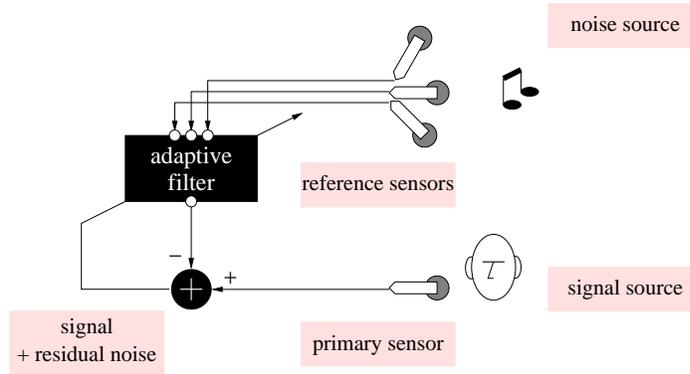


Figure 1.14: Acoustic interference cancellation example

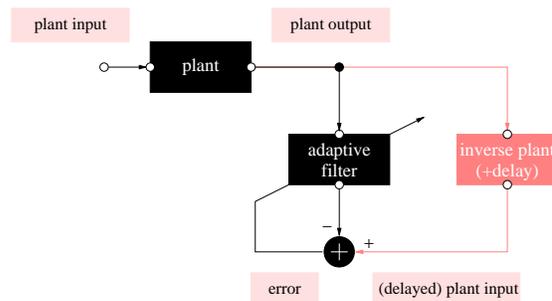


Figure 1.15: Inverse modelling scheme

unit circle and hence will be unstable.

The mobile communications application of Figure 1.7 can be reconsidered here as an inverse modelling scheme. Instead of first computing a channel model and then designing an equalizer, one can also try and compute a suitable equalizer right away. This is indicated in **Figure 1.16**. Again, a training sequence is transmitted, but now the adaptive filter immediately tries to reconstruct this sequence, *i.e.* invert/equalize the channel distortion⁵. Clearly, the scheme is a special case of Figure 1.15.

Such an equalizer structure may easily be switched into so-called ‘decision-directed mode’, after training. If, after initial training, a sequence of information bits is transmitted, the output of the adaptive filter is expected to be ‘close’ to the transmitted sequence. A *decision device* or *slicer* may be used that picks the nearest bit (or symbol) at each sampling instance, to remove the residual noise, see **Figure 1.17**. If the average probability of error is small, the output of the slicer may be used as the desired signal for further filter adaptation, *e.g.* to track modest time variations in the channel characteristic.

A recent trend in mobile communications research is towards the use of multiple antenna (antenna array) base stations. With a single antenna, the channel equalization problem for the so-called uplink (mobile to base station) is essentially the same as

⁵This type of equalization will be used to illustrate the theory in Chapter 4.

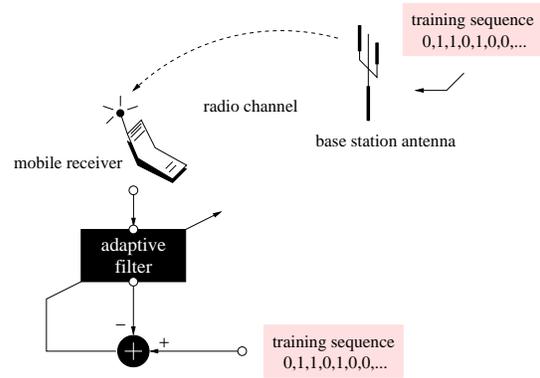


Figure 1.16: Channel equalization in mobile communications (training mode)

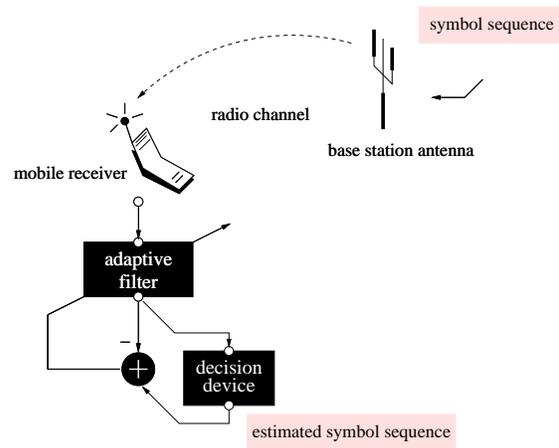


Figure 1.17: Channel equalization (decision-directed mode)

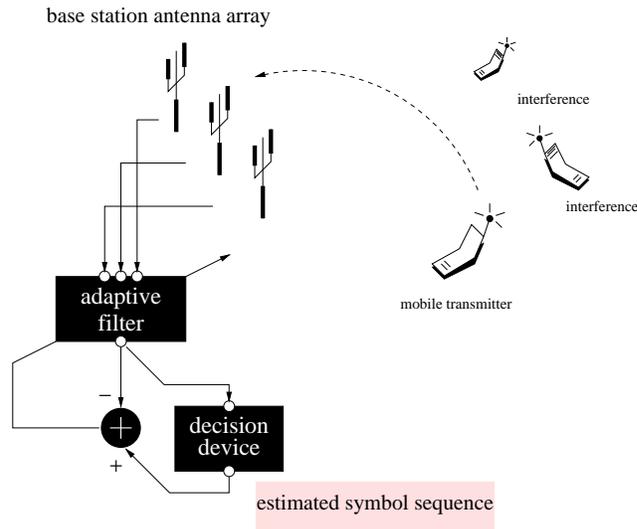


Figure 1.18: Channel equalization and interference cancellation

for the downlink (base station to mobile, see Figures 1.7, 1.16 and 1.17). However, when base stations are equipped with antenna arrays additional flexibility is provided, see **Figure 1.18**. Equalization is usually used to combat intersymbol interference, but now, in addition, one can attempt to reduce *co-channel interference* from other users, *e.g.* those occupying adjacent frequency bands. This may be accomplished more effectively by means of antenna arrays. The adaptive filter now has as many inputs as there are antennas. For narrow band applications⁶, the adaptive filter computes an optimal linear combination⁷ of the antenna signals. The multiple physical antennas and the adaptive filter may then be viewed as a ‘virtual antenna’ with a beam pattern (directivity pattern) which is a linear combination of the beampatterns of the individual antennas. The (ideal) linear combination will be such that a narrow main lobe is steered in the direction of the selected mobile, while ‘nulls’ are steered in the direction of the interferers⁸. For broad band applications or applications in multipath environments (where each signal comes with a multitude of reflections from different locations), the situation is more complex and higher-order FIR filters are used. This is a topic of current research.

1.3.5 Linear prediction

In many applications, it is desirable to construct the desired signal from the filter input signal, for example consider

$$d_k = u_{k+1}$$

⁶A system is considered to be narrowband when the signal bandwidth is much smaller than the RF carrier frequency.

⁷*i.e.* zero-order multi-input single output FIR filter.

⁸A null is a direction where the beam pattern has zero gain. Beamforming will be used to illustrate the theory in Chapters 3 and 6.

with d_k the desired signal at time k (see also Figure 1.4). This is referred to as the *forward linear prediction problem*, *i.e.* we try to predict one time step forward in time. When

$$d_k = u_{k-N}$$

we have the *backward linear prediction problem*. Here we are attempting to ‘predict’ a previously observed data sample. This may sound rather pointless since we know what it is. However, as we shall see backward linear prediction plays an important role in adaptive filter theory.

If e_k is the error signal at time k , one then has (for forward linear prediction)

$$e_k = u_{k+1} - w_0 \cdot u_k - w_1 \cdot u_{k-1} - \cdots - w_{N-1} \cdot u_{k-N+1}$$

or

$$u_{k+1} = e_k + w_0 \cdot u_k + w_1 \cdot u_{k-1} + \cdots + w_{N-1} \cdot u_{k-N+1}$$

This represents a so-called *autoregressive (AR) model* of the input signal u_{k+1} (similar equations hold for backward linear prediction). AR modelling plays a crucial role in many applications, *e.g.* in *linear predictive speech coding (LPC)*. Here a speech signal is cut up in short segments (‘frames’, typically 10 to 30 milliseconds long). Instead of storing or transmitting the original samples of a frame, (a transformed version of) the corresponding AR model parameters together with (a low fidelity representation of) the error signal is stored/transmitted.

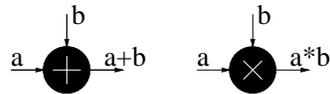
1.4 Signal Flow Graphs

Here, and in subsequent chapters, to avoid the use of as many mathematical formulas as possible, we make extensive use of *signal flow graphs*. Signal flow graphs are rigorous graphical representations of algorithms, from which a mathematical algorithm description (whenever needed) is straightforwardly derived. Furthermore, by virtue of the graphical presentation they provide additional insight in the structure of the algorithm, which is often not so obvious from its mathematical description.

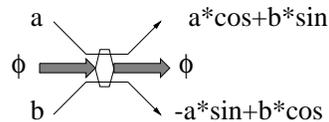
As indicated by the example in **Figure 1.19** (some of the symbols will be explained in more detail in later chapters) a signal flow graph is equivalent to a set of equations in an infinite time loop, *i.e.* the graph specifies a sequence of operations which are executed for each time step k ($k = 1, 2, \dots$).

Signal flow graphs are commonly used to represent simple filter structures, such as in Figures 1.4 and 1.5. More complex algorithms, such as in Figure 1.21 and especially Figure 1.22 (see below), are rarely specified by means of signal flow graphs, which is unfortunate for the advantages of using signal flow graphs are most obvious in the case of complex algorithms

multiplication/addition



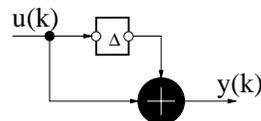
orthogonal transformations



delay elements



example



means:

for $k=1,2,\dots$
 $y(k)=u(k)+u(k-1)$
 end

Figure 1.19: SFG building blocks

It should be recalled that a signal flow graph representation of an algorithm is not a description of a hardware implementation (e.g. a systolic array). Like a circuit diagram the signal flow graph defines the data flow in the algorithm. However, the cells in the signal flow graph represent mathematical operators and not physical circuits. The former should be thought of as instantaneous input-output processors that represent the required mathematical operations of the algorithm. Real circuits on the other hand require a finite time to produce valid outputs and this has to be taken into account when designing hardware.

1.5 Preview/overview

In this section, a short overview is given of the material presented in the different chapters. A few figures are included containing results that will be explained in greater detail later, and hence the reader should not yet make an attempt to fully understand everything. To fix thoughts, we concentrate on the acoustic echo cancellation scheme, although everything carries over to the other cases too.

As already mentioned, in most cases the FIR filter structure is used. Here the filter output is a linear combination of N delayed filter input samples. In **Figure 1.20** we have $N = 4$, although in practice N may be up to 2000. The aim is to adapt the tap weights w_i such that an accurate replica of the echo signal is generated.

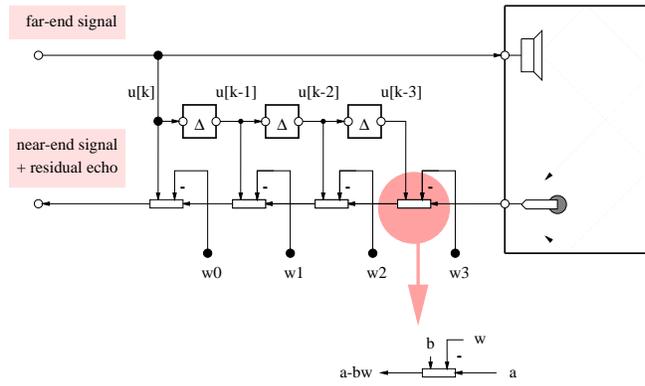


Figure 1.20: Adaptive FIR filtering

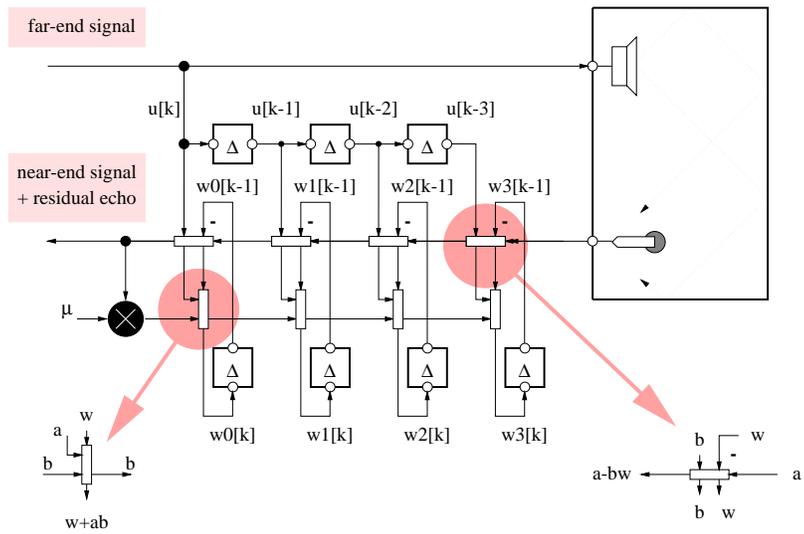


Figure 1.21: LMS algorithm (Chapter 2)

Many of the adaptation schemes we will encounter take the form

$$\mathbf{w}(k) = \mathbf{w}(k-1) + (\text{correction})$$

where

$$\mathbf{w}^T(k) = [w_0(k) \quad w_1(k) \quad w_2(k) \quad \cdots \quad w_{N-1}(k)]$$

contains the tap weights at time k . We have already indicated that the error signal often used to steer the adaptation, and so one has

$$\mathbf{w}(k) = \mathbf{w}(k-1) + \underbrace{(d_k - \mathbf{u}_k^T \mathbf{w}(k-1))}_{e_k} \cdot (\text{correction})$$

where

$$\mathbf{u}_k^T = [u_k \quad u_{k-1} \quad u_{k-2} \quad \cdots \quad u_{k-N+1}]$$

This makes sense since if the error is small then we do not need to alter the weight; on the other hand, if the error is large, we will need a large change in the weights. Note that the correction term has to be a vector - $\mathbf{w}(k)$ is one! Its role is specify the ‘direction’ in which one should move in going from $\mathbf{w}(k-1)$ to $\mathbf{w}(k)$. Different schemes have different choices for the correction vector. A simple scheme, known as the *Least Mean Squares (LMS) algorithm* (Widrow 1965), uses the FIR filter input vector as a correction vector, *i.e.*

$$\mathbf{w}(k) = \mathbf{w}(k-1) + \mu \cdot \mathbf{u}_k \cdot (d_k - \mathbf{u}_k^T \mathbf{w}(k-1))$$

where μ is a so-called step size parameter, that controls the speed of the adaptation. A signal flow graph is shown in **Figure 1.21**, which corresponds to Figure 1.20 with the filter adaptation mechanism added. The LMS algorithm will be derived and analysed in Chapter 3. Proper tuning of the μ will turn out to be crucial to obtain stable operation. The LMS algorithm is seen to be particularly simple to understand and implement, which explains why it is used in so many applications.

In Chapter 2, the LMS algorithm is shown to be a ‘pruned’ version of a more complex algorithm, derived from the least squares formulation of the optimal (adaptive) filtering problem. The class of algorithms so derived, is referred to as the class of *Recursive Least Squares (RLS) algorithms*. Typical of these algorithms is an adaptation formula that differs from the LMS formula in that a ‘better’ correction vector is used, *i.e.*

$$\mathbf{w}(k) = \mathbf{w}(k-1) + \mathbf{k}_k \cdot (d_k - \mathbf{u}_k^T \mathbf{w}(k-1))$$

where \mathbf{k}_k is the so-called *Kalman gain vector*, which is computed from the autocorrelation matrix of the filter input signal (see Chapter 2).

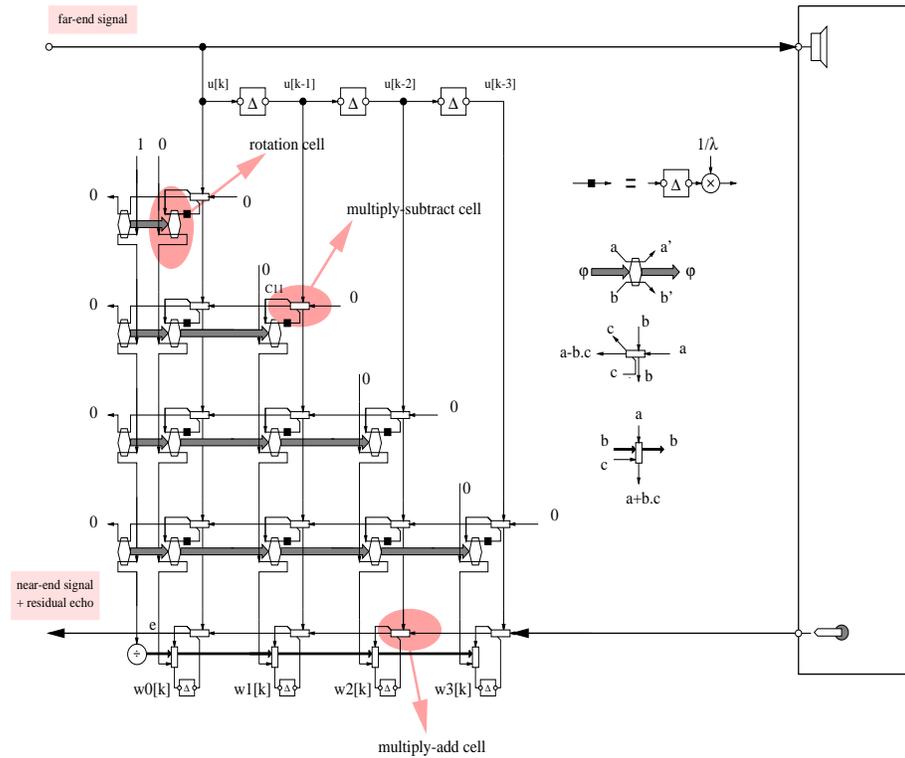


Figure 1.22: Square-root RLS algorithm (Chapter 3)

One particular RLS algorithm is shown in **Figure 1.22**. Compared with the LMS algorithm (Figure 1.21), the filter/adaptation part (bottom row) of this algorithm is roughly unchanged, however an additional triangular ‘network’ has been added to compute the Kalman gain vector, now used in the adaptation. This extra computation is the reason that the RLS algorithms have better performance. The RLS algorithm shown here is a so-called *square-root algorithm*, and will be derived in **Chapter 4**, together with a number of related algorithms.

Comparing Figure 1.21 with Figure 1.22 reveals many issues which will be treated in later chapters. In particular, the following issues will be dealt with :

Algorithm complexity: The LMS algorithm of Figure 1.21 clearly has less operations (per iteration) than the RLS algorithm of Figure 1.22. The complexity of the LMS algorithm is $\mathcal{O}(N)$, where N is the filter length, whereas the complexity of the RLS algorithm is $\mathcal{O}(N^2)$. For applications where N is large (e.g. $N = 2000$ in certain acoustic echo control problems), this gives LMS a big advantage. In **Chapter 5**, however, it will be shown how the complexity of RLS-based adaptive FIR filtering algorithms can be reduced to $\mathcal{O}(N)$. Crucial to this complexity reduction is the particular shift structure of the input vectors \mathbf{u}_k (cfr. above).

Convergence/tracking performance: Many of the applications mentioned in the previous section, operate in unknown and time-varying environments. Therefore, speed of convergence as well as tracking of the adaptation algorithms is crucial.

Note however that tracking performance is not the same as convergence performance. Tracking is really related to the estimation of non-stationary parameters - something that is strictly outside the scope of much of the existing adaptive filtering theory and certainly this book. As such we concentrate on the convergence performance of the algorithms but will mention tracking where appropriate.

The LMS algorithm, *e.g.*, is known to exhibit slow convergence, while RLS-based algorithms perform much better in this respect. An analysis is outlined in Chapter 3.

If a model is available for the time variations in the environment, a so-called *Kalman filter* may be employed, Kalman filters are derived in Chapter 6. All RLS-based algorithms are then seen to be special instances of particular Kalman filter algorithms.

Further to this, in Chapter 7 several routes are explored to improve the LMS's convergence behaviour, *e.g.*, by including signal transforms or by employing frequency domain as well as subband techniques.

Implementation aspects: Present-day VLSI technology allows the implementation of adaptive filters in application specific hardware. Starting from a signal flow graph specification, one can attempt to implement the system directly assuming enough multipliers, adders, *etc.* can be accommodated on the chip. If not, one needs to apply well known VLSI algorithm design techniques such as 'folding' and 'time-multiplexing'.

In such systems, the allowable sampling rate is often defined by the longest computational path in the graph. In **Figure 1.23**, *e.g.*, (which is a re-organized version of Figure 1.20), the longest computational path consists of four multiply-add operations and one addition. To shorten this ripple path, one can apply so-called *retiming* techniques. If one considers the shaded box (subgraph), one can verify that by moving the delay element from the input arrow of the subgraph to the output arrow, resulting in **Figure 1.24**, the overall operation is unchanged. The ripple path now consists of only two multiply-add operations and one addition, which means that the system can be clocked at an almost two times higher rate. Delay elements that break ripple paths are referred to as *pipeline delays*. A graph that has ripple paths that consist of only a few arithmetic operations is often referred to as a 'fully pipelined' graph

In several chapters, retiming as well as other signal flow graph manipulation techniques are used to obtain fully pipelined graphs/systems.

Numerical stability: In many applications, adaptive filter algorithms are operated over a long period of time, corresponding to millions of iterations. The algorithm is expected to remain stable at all time, and hence numerical issues are crucial to proper operation. Some algorithms, *e.g.* *square-root RLS algorithms*, have much better numerical properties, compared with non-square-root RLS algorithms (Chapter 4).

In Chapter 8 ill-conditioned adaptive filtering problems are treated, leading to so-called 'low rank' adaptive filtering algorithms, which are based on the so-called singular value decomposition (SVD).

Finally in Chapter 9, a brief introduction to adaptive IIR (infinite impulse response) filtering is given, see Figure 1.5.

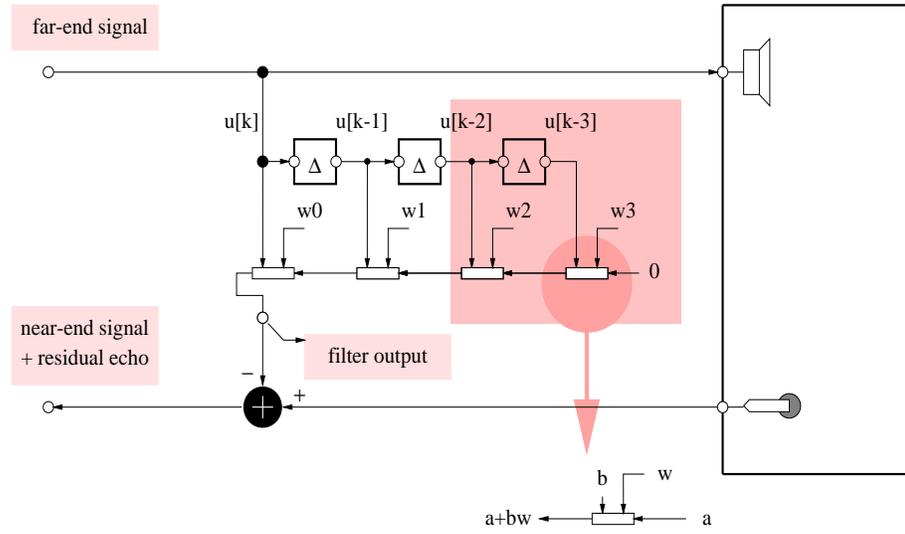


Figure 1.23: FIR filter

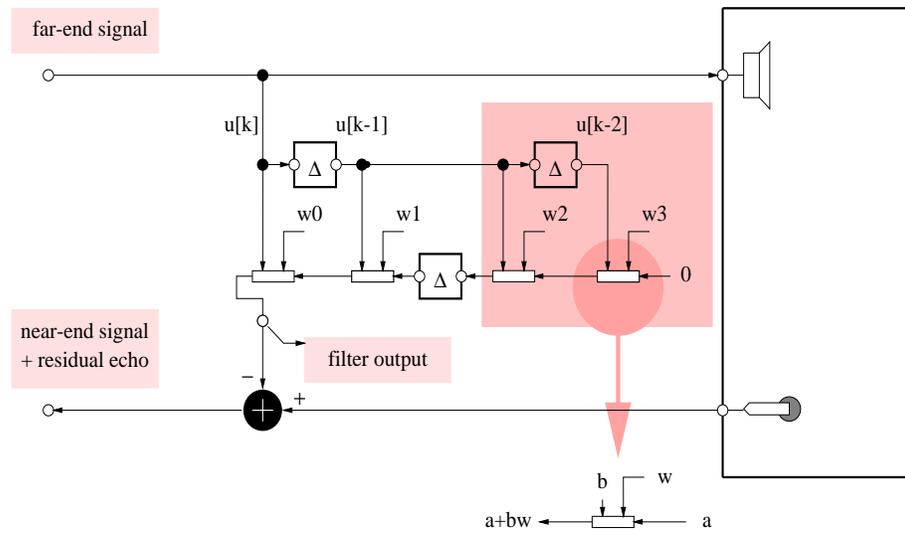


Figure 1.24: Retimed FIR filter