Next Up Previous Contents
Next: 4 Adaptive Filters Applications **Up:** 2 Introduction to Adaptive **Previous:** 5 Nonlinear Filters **Contents**

3 Basic Adaptive Filter Model

The optimization problem on which all ASPT functions are designed to solve is shown in Fig. 2.6. The adaptive filter is the dotted box in this figure and consists of two parts. The filter part and the update part. The filter part (labeled "Adjustable filter" in Fig. 2.6), can be based on any of the filter structures mentioned in Section 2.2. The function of the filter part is to calculate the filter output signal $y(n)$ as shown in Sections 2.2.1 to 2.2.4. The set of filter coefficients are continuously adjusted by the update part. The update part (labeled "Adaptive algorithm" in Fig. 2.6) is responsible for adjusting the filter coefficients so that the filter output $y(n)$ becomes as close as possible to a desired signal $d(n)$. In most cases, the update part changes the filter coefficients in small steps to minimize a certain function of the error signal $e(n)$, defined as the difference between the desired signal $d(n)$ and the filter output $y(n)$,

$$e(n) = d(n) - y(n).$$

The function to be minimized is often referred to as the performance functions, also known as the performance index. The performance function can be chosen based on statistic or deterministic approaches.

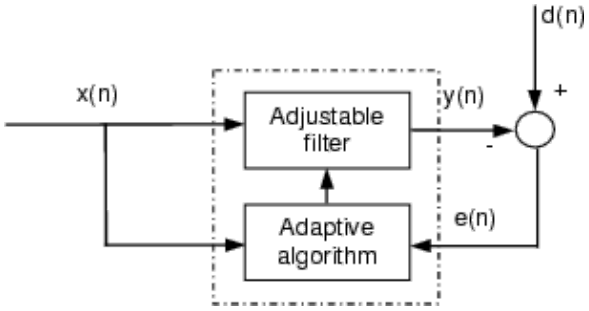

Figure 2.6: Block diagram of the general adaptive filtering problem.

The most commonly used statistical performance function is the mean square of the error signal $\zeta(n) = E\{e^2(n)\}$, where $E\{\cdot\}$ is the expectation operator. In this case, the update part of the adaptive filter adjusts the filter coefficients to minimize the mean square value of the error signal. On achieving this goal, and in ideal situations, the statistical average (mean value) of the error approaches zero, and the filter output approaches the desired signal. The adaptive filter has converged to its optimum solution in the mean square sense. When the input $x(n)$ and desired $d(n)$ are stationary signals, this optimization process leads to the well known Wiener filter [11]. The Least Mean Square (LMS) algorithms is a good example of a practical algorithm based on this statistical approach. The exact details on how the coefficients are adjusted define the time it takes to reach the final solution (the conversion time) and the difference between this final solution and the optimum solution (the final misadjustment). Many ASPT functions are based on this statistical framework. Examples are `asptlms()` (Section 4.9), `asptnlms()` (Section 4.11), `asptleakynlms()` (Section 4.7), `asptvsslms()` (Sections 4.10) for time domain sample per sample transversal and linear combiner filters. For time domain sample per sample recursive filters `aspteqerr()` (Section 6.2), and `asptouterr()` (Section 6.3). For block processing transversal filter `asptbfdaf()` (Section 4.2), `asptpbfdaf()` (Section 4.12), and `asptrcpbfdaf()` (Section 4.13). And finally for lattice filters `asptlmslattice()` (Section 5.4), `asptlbpef()` (Section 5.2), and `asptlfpef()` (Section 5.3). A commonly used deterministic performance function is the weighted sum of the squared value of the previous error signal samples $\zeta(n) = \Sigma_{k=1}^{n} \lambda^{n-k} e^2(k)$, where λ is a constant close to, but less than one, and $k = 1, 2, \cdots, n$. This choice puts more emphasis on recent observed error samples and gradually forgets about the past samples, a good reason for calling the parameter λ the forgetting factor. Minimizing $\zeta(n)$ leads to an optimum set of filter coefficients for the given set of data that makes the filter output $y(n)$ as close as possible to the desired signal $d(n)$ in the least squares sense. It is worth mentioning, however, that if the set of data satisfy certain statistical properties,

and a large data length is used, the optimum filter coefficients obtained from this deterministic optimization approaches the Wiener (statistical) solution [4]. The deterministic approach mentioned above is the basis for the Recursive Least Squares (RLS) algorithm and its derivatives known for fast convergence and fast tracking properties. Examples of ASPT functions that are based on the deterministic framework are `asptrls()` (Section 4.16). And finally for lattice filters `asptrlslattice()` (Section 5.5), `asptrlslbpef()` (Section 5.7), and `asptrlslfpef()` (Section 5.8). ASPT functions are all designed to match the model shown in Fig. 2.6. Besides parameters specific for each algorithm, the adaptive algorithms take the input $x(n)$, desired $d(n)$, and the filter coefficients vector from previous iterations as input parameters and provide the filter output $y(n)$, the error signal $e(n)$, and the updated filter coefficients as output parameters. For sample per sample algorithms such as the `asptlms()`, the adaptive function is called each sample in a loop to process each pair of input and desired samples. For block processing functions such as the `asptbfdaf()`, the adaptive algorithm is called each B samples, where B is the block length to process B input and B desired samples and provides B output and B error samples. The filter coefficients are then updated each B samples, the frequency of calling the adaptive algorithm. This closely simulates real time implementations and provides insight into real time implementations performance. More details on the operation of adaptive algorithms, their characteristics, and properties are given in the reference page of each algorithm. For theoretical background on those algorithms, the reader is referred to one of the classical text books mentioned at the end of this document.

Next Up Previous Contents

Next: 4 Adaptive Filters Applications **Up:** 2 Introduction to Adaptive **Previous:** 5 Nonlinear Filters **Contents**