Next Up Previous Contents
**Next:** 3 Basic Adaptive Filter **Up:** 2 Filter Structures supported **Previous:** 4 Lattice Filters **Contents**

## 5 Nonlinear Filters

In many applications linear filtering techniques do not provide satisfactory results and nonlinear filters must be used. For instance, in data communication applications it is known that the transmission errors on telephone lines at transmission rates higher than 4800 bits/s are caused almost entirely by nonlinear distortion. In applications such as subscriber lines that use much higher transmission rate, even a slight channel nonlinearity may cause problems for a linear echo canceler. The solution to such problems is to use nonlinear filters, equalizers, or controllers to manage the nonlinear systems at hand. Unlike linear systems that have a unique theory, there exists no unique method to model and characterize nonlinear systems. Different types of nonlinear systems have been developed to solve specific problems. The most commonly used nonlinear techniques are order statistics filters, polynomial filters, morphological filters, and homomorphic filters [7]. Of those techniques, the polynomial filters are supported by the current release of ASPT. The most widely used polynomial filters are the Volterra filters discussed below. A causal nonlinear time-invariant continuous system with finite memory can be modeled by a continuous Volterra series of finite order and finite memory as follows,

$$\begin{aligned} y(t) &= h_0 + \int_0^T h_1(t_1)x(t-t_1)dt_1 + \int_0^T\int_0^T h_2(t_1,t_2)x(t-t_1)x(t-t_2)dt_1dt_2 + \cdots \\ &\quad + \int_0^T \cdots \int_0^T h_k(t_1,\cdots,t_k)x(t-t_1)\cdots x(t-t_k)dt_1\cdots dt_k, \end{aligned}$$

where the multidimensional functions $h_k(t_1,\cdots,t_k)$ are the Volterra kernel functions that are assumed to be symmetric with respect to their variables. Similarly, a causal nonlinear time-invariant discrete system with finite memory can be modeled by a discrete Volterra series of finite order and finite memory as follows,

$$\begin{aligned} y(n) &= h_0 + \sum_{m_1=0}^{N-1} h_1(m_1)x(n-m_1) + \sum_{m_1=0}^{N-1}\sum_{m_2=0}^{N-1} h_2(m_1,m_2)x(n-m_1)x(n-m_2) + \cdots \\ &\quad + \sum_{m_1=0}^{N-1} \cdots \sum_{m_k=0}^{N-1} h_k(m_1,\cdots,m_k)x(n-m_1)\cdots x(n-m_k). \end{aligned}$$

One of the important properties of the Volterra series based filters is that the output of both the continuous and discrete Volterra filters are linear functions with respect to the coefficients. The nonlinearity of the filter is introduced by forming the cross products of the input signals. This linearity property allows applying much of the linear systems theory, including linear adaptive filtering, to the Volterra series based nonlinear filters. A disadvantage of the Volterra filters, however, is that they can not be used to model strong nonlinearity such as saturation and discontinuity. To illustrate, the second order Volterra filter is discussed in more details below. The output of the second order Volterra filter with symmetric coefficients is given by

$$y(n) = \sum_{m_1=0}^{N-1} w_1(m_1)x(n-m_1) + \sum_{m_1=0}^{N-1}\sum_{m_2=m_1}^{N-1} w_2(m_1,m_2)x(n-m_1)x(n-m_2),$$

where $N$ is the filter memory length. The linear relationship mentioned above allows writing equation (2.12) in vector notation as follows,

$$y(n) = \underline{\mathbf{w}}(n)^T\,\underline{\mathbf{x}}(n),$$

where $\underline{\mathbf{w}}(n)$ is the vector of filter coefficients including the linear kernel, $w_1$ and the nonlinear kernel, $w_2$. As an example, the coefficients vector of a second order Volterra filter of memory length 3 are given by

$$\underline{\mathbf{w}}(n) = [w_1(0), w_1(1), w_1(2), w_2(0,0), w_2(0,1), w_2(0,2), w_2(1,1), w_2(1,2), w_2(2,2)].$$

The vector $\underline{\mathbf{x}}(n)$ includes the current and past input samples as well as the cross products needed to calculate the nonlinear part of the filter output.

$$\underline{\mathbf{x}}(n) = [x(n), x(n-1), x(n-2), x^2(n), x(n)x(n-1), x(n)x(n-2), x^2(n-1), x(n-1)x(n-2), x^2(n-2)].$$

The total length of $\mathbf{w}(n)$ and $\mathbf{x}(n)$ for the second order Volterra filter is $N + sum(1:N)$ or $3 + (1+2+3) = 9$ in the above example. Besides calculating the filter output, an adaptive second order Volterra filters must also adjust the filter coefficients to optimize a certain performance index in some sense such that the filter output becomes as close as possible to the desired signal $d(n)$. Again, due to the linearity property mentioned above, methods used to adapt linear adaptive filters can readily be applied to adapt the coefficients of the Volterra adaptive filter. The current release of ASPT contains several functions that implement Volterra filters. For fixed filters, `sovfilt()` (Section 9.11) calculates the output of a second order Volterra filter. Second order Volterra adaptive filters can be implemented using `asptsovlms()` (Section 8.1), `asptsovnlms()` (Section 8.2), `asptsovrls()` (Section 8.3), `asptsovtdlms()` (Section 8.4), or `asptsovvsslms()` (Section 8.5).

Next Up Previous Contents

**Next:** 3 Basic Adaptive Filter **Up:** 2 Filter Structures supported **Previous:** 4 Lattice Filters **Contents**