3.5 Time-Varying Fractional Delay Filters

It is interesting to investigate the situation where the delay parameter of an FD filter is changed during operation, since in many applications the desired delay is not constant but varies with time. In this section we study the implementation of a variable-length digital delay line using FIR and IIR filters.

3.5.1 Consequences of Changing Filter Coefficients

The main issue is to understand what the change of the filter characteristics means for the output signal of the filter. We consider a single change at the coefficient set at time index $n = n_c$. There will be two kinds of consequences:

1) The output signal may suffer from a *transient phenomenon* that starts at $n = n_c$ if the state variables of the filter contain intermediate results related to the former coefficient set, or if they are cleared or part of the input data is otherwise missing. In the case of nonrecursive filters this problem does not exist when the filter has been implemented in a correct way, but in the case of recursive filters the occurrence of transients must be taken into account somehow.
2) The output signal of a time-varying FIR or IIR filter experiences a *discontinuity* at $n = n_c$, which is simply caused by the fact that after the coefficients have been changed the output signal will be a result of a different kind of filtering operation. However, the discontinuity can cause problems in some cases.

Both the transient signal and the discontinuity depend on the input signal of the filter and the magnitude of coefficient change: a small change in the values of the coefficients causes a transient with a smaller amplitude and a smaller discontinuity than a large change in coefficient values. The severity of the transients and discontinuity can be decreased by making the change in filter parameters smaller, e.g., by allowing the filter a reasonably long transition time when the values of the coefficients are gradually changed (by interpolation) from the initial to the target values.

3.5.2 Implementation of a Variable Delay Using FIR Filters

We first examine the realization of a variable-length digital delay line using FIR interpolating filters. Interpolation is needed when the desired length of the delay line is not an integral multiple of unit samples, in other words, when it is needed to know the value of a discrete-time signal at a point $D = \lfloor D \rfloor + d$.

In principle it is straightforward to exploit FIR filters in a time-varying situation. This is because they do not have transient problems when the filter coefficients are modified. This is true, of course, only when the FIR filter has been implemented so that the input samples that are needed for computing the output of the filter after the change in the delay value are available. Special care is needed with FIR filters when the order of the filter is changed or whole unit delays are added to or subtracted from the system. An advantage of nonrecursive filters is that they are stable in all (time-invariant and time-varying) situations.

Figure 3.25 illustrates the use of an FIR interpolator for controlling the length of a delay line. The coefficient vector of the FIR interpolator is denoted by **h** and is defined as

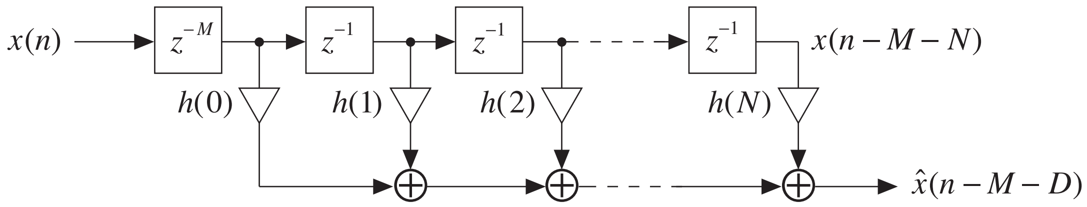

Fig. 3.25 A variable-length digital delay line implemented using FIR interpolation.

$$\mathbf{h} = [h(0) \quad h(1) \quad h(2) \quad \cdots \quad h(N)]^T \tag{3.138}$$

For simplicity of notation, the dependency of the coefficients $h(n)$ on the delay D is not explicitly shown. It should be remembered that we should use $h(n, D)$ in all the equations. Various techniques for determining the interpolation coefficients $h(n)$ have been discussed earlier in this chapter.

An estimate for the ideal interpolated signal value $x(n - M - D)$ is computed as (see Fig. 3.25)

$$\hat{x}(n - M - D) = \sum_{k=0}^{N} h(k)x(n - M - k) = \mathbf{h}^T \mathbf{x}_n \tag{3.139a}$$

where M is the number of unit delays in the delay line before the FIR interpolator [as defined by Eq. (3.37)] and the vector of signal samples to be used in interpolation is

$$\mathbf{x}_n = [x(n - M) \quad x(n - M - 1) \quad x(n - M - 2) \quad \cdots \quad x(n - M - N)]^T \tag{3.139b}$$

The length of a delay line can now be changed continuously by computing new values for the FIR filter coefficients $h(n)$ at every sampling interval.

In the following we concentrate on the case where the delay parameter is changed at time index n_c. The delay parameter is thus a function of time so that

$$D(n) = \begin{cases} D_1, & n < n_c \\ D_2, & n \ge n_c \end{cases} \tag{3.140}$$

If the next delay value D_2 does not fulfill the requirement $(N - 1)/2 \le D_2 < (N + 1)/2$, the filter taps of the FIR FD filter have to be moved to obtain the best approximation (for a given approximation method and given N). In other words, the taps of the FIR filter have to be moved with respect to the delay line if d passes the value 0 when N is odd or d passes the value 0.5 when N is even. This also implies that an integer must be added to or subtracted from D_2 to have a delay that is within the optimal range.

Special attention must be paid to correctly implement the delay change where $D_2 > D_1$ and the filter taps need to be moved: the transient signal will be completely eliminated if the delayed values of the input signal are available, i.e., the delay line is long enough. One solution is to postpone the change of delay parameter for the time that it takes to store the input signal values into the buffer memory (i.e., to wait until the delay line is filled with samples needed by the FIR filter) and change the delay only then. This will, however, cause an extra processing delay which depends on the value of the delay parameter and which may not be acceptable.

A practical solution is to define a maximum delay $D_{max} \in \mathbb{R}$ which is the assumed largest delay value (in samples) that will be needed in a particular application. The length of the buffer that stores the delayed values of the input signal is then set long enough for approximating D_{max}. In practice this means that the delay line length M_{tot} must be

$$M_{tot} = D_{max} + \lfloor N / 2 \rfloor \tag{3.141}$$

Here the $\lfloor N / 2 \rfloor$ extra samples are required because the fractional point D should be located near the central tap of the FIR interpolator. This issue has been discussed in Section 3.2.1 for the LS FIR filter design but it applies to other FD FIR filters as well.

It is also necessary to consider the case $D_2 < D_1$ when the filter taps should be moved. Since the filter taps should be located in such a way that $(N - 1)/2 \le D_2 < (N + 1)/2$, a delay $D_2 < (N - 1)/2$ cannot be approximated with good accuracy. (This discussion is irrelevant for the case $N = 1$ since for the first-order filter the smallest delay in the optimal range is zero.) There are two possible solutions to the problem of a small fractional delay:

1) to approximate the delay outside the optimal range or

2) to reduce the order N of the FIR FD filter.

Both methods imply reduced accuracy in the FD approximation. The choice depends on the application. In digital waveguide modeling, when Lagrange interpolation is employed, it is better to use a lower order filter (choice 2) since the Lagrange interpolator is not a passive filter outside its optimal range. From the discussion in Section 3.3.6 we can also deduce that this solution will actually give a smaller total error than using a higher-order filter outside its optimal range. Notice that the above discussion is mainly relevant to real-time implementation of an FIR delay filter†.

The use of an FIR interpolator is favorable in the sense that *no disturbing transients* will be generated when the delay D is changed. However, a *discontinuity* will occur at the output. If the change in the coefficient values is small enough the discontinuity will be unnoticeable, but if the change is large a serious disturbance may occur for certain signals. For computational savings it may be necessary to update the interpolating coefficients more seldom than for every sampling interval, say every 10 or 100 sampling cycles. It is recommended to experimentally verify that updating is frequent enough, so that the discontinuities do not cause trouble.

3.5.3 Transients in Time-Varying Recursive FD Filters

Due to the recursive nature of IIR filters, abrupt changes in their coefficients cause disturbances in the future values of the internal state variables and transients in the output values of the filter. These disturbances depend on the filter's impulse response so that they are in principle infinitely long. In the case of stable filters, however, the distur-

† In off-line processing where the entire input signal is stored in the computer memory and filtering is executed afterwards, the main concern is to take into account the edge effects due to the finite data size (see, e.g., Hamming, 1983): near the ends of the signal buffer some of the FIR filter's input signal values are missing (assumed to be zero). This would, in general, give rise to an error (transient) in the output signal. In most cases this transient at the beginning and at the end of the output signal can be removed by truncating the output signal.

bances decay exponentially and can be treated as having a finite length. In practice, these transients are often serious enough to cause problems, such as clicks in audio applications, and they are typically the most serious problem in the implementation of tunable or time-varying recursive filters.

The waveform of the transient signal depends on the structure of the filter. For example, recursive filters implemented with direct form (DF) I and DF II structures will yield a different kind of transient signal after the change of filter coefficients when their initial and target coefficients and the input signal are identical.

If the coefficients of a recursive filter are changed at constant time intervals, the disturbance caused by the transient bursts can yield a quasiperiodic signal. In audio applications, this is heard as a tonal sound.

3.5.4 Transient Elimination Methods for Time-Varying Recursive Filters

Despite the importance of the problem, there exists only a handful of research reports on strategies for elimination of transients in time-varying recursive filters. To the knowledge of the author, there are altogether five different techniques for this task:

1) a cross-fading method,

2) gradual variation of coefficients using interpolation (Mourjopoulos *et al.*, 1990),

3) intermediate coefficient matrix (Rabenstein, 1988),

4) an input-switching method (Verhelst and Nilens, 1986), and

5) updating of the state vector (Zetterberg and Zhang, 1988).

These methods are briefly described in this section.

Cross-Fading Method

The cross-fading method is the most intuitive of all transient elimination techniques. In this method, typically two copies, $H_1(z)$ and $H_2(z)$, of the time-varying filter are used. The input signal is connected to both of these filters.

Let us assume that first filter $H_1(z)$ is used for filtering the input signal. When the coefficients need to be changed, the filter $H_2(z)$ is given the target filter coefficients. During a *transition time* Δn, the outputs of the filters are cross-faded (e.g., using linear interpolation) so that afterwards only the output of filter $H_2(z)$ is connected to the overall output.

This method has been used, for example, in audio applications such as auralization, where the incident angle of the binaurally processed audio signal is changed using digital filters designed based on measurements of head-related transfer functions (Jot *et al.*, 1995). In this application it may be necessary to cross-fade between three or four filters, if the elevation angle—in addition to the horizontal angle—is being taken into account. The overall output signal is then a linear combination of the filter outputs.

Gradual Variation Technique

In gradual variation of coefficients one also allows a *transition time* Δn (in samples) during which the coefficients monotonically change from their initial values to the target values (Mourjopoulos *et al.*, 1990; Zölzer *et al.*, 199). Mourjopoulos *et al.* (1990) defined a *filter variation parameter* η as

$$\eta = \frac{M+1}{\Delta n T} \tag{3.142}$$

where M is the number of intermediate coefficient sets and T is the sampling interval. This parameter describes the number of intermediate coefficients per unit time. Note that it is assumed that each intermediate coefficient set is running for several sampling cycles.

Mourjopoulos *et al.* (1990) studied the useful range of values for parameter η by analyzing the transients in an audio filtering application using a computational model of human auditory properties. Their results were that when η is small (i.e., a small number of intermediate filter coefficients is used) the transition is perceivable because there are discontinuities in the filter parameters; at least 18 to 35 states per second—depending on parameter settings—should be used to ensure smooth coefficient transition. Abrupt changes generate audible transients. Mourjopoulos *et al.* (1990) conclude that these distortions are most easily heard when the input signal is a sine wave. Speech and music sounds mask the majority of the transient effects.

Nikolaidis *et al.* (1993) have considered a variation of the gradual variation technique where filter parameters (instead of coefficients) are interpolated. In this paper they also propose a generalization where two filters, $H_1(z)$ and $H_2(z)$, are used in parallel. Both filters receive the input signal all the time but the output signal of only one of them is connected to the output of the overall system. This goes on for n_u sampling cycles, which is determined as

$$n_\mathrm{u} = \frac{\Delta n}{M+1} \tag{3.143}$$

At the same moment when the output of the filter $H_1(z)$ is connected to the overall output, the coefficients of $H_2(z)$ are changed to correspond the next intermediate set and its output is disconnected. After n_u samples, the output is switched again and the coefficients of $H_1(z)$ are changed. Nikolaidis *et al.* (1993) also propose a two-speed implementation where the off-time filter runs at a higher sampling rate. With this approach this filter in effect has a longer time (a multiple of n_u) to settle before its output is switched on.

Intermediate Coefficient Matrix

Another technique to reduce the transients is the use of an intermediate coefficient matrix (Rabenstein, 1988). There are altogether three coefficient matrices involved in one change of filter characteristics: the initial, the intermediate, and the target matrices. The intermediate matrix is used for one sampling cycle to compensate for the transient. This approach involves assumptions on the nature of the input signal. The drawback of this technique is that the filter has to be implemented in a state-variable form which is far more expensive in terms of operations than a direct-form realization. For this reason we feel that this technique is not suitable for real-time DSP applications.

Input-Switching Method

The input-switching method was introduced by Verhelst and Nilens (1986). Their application was speech synthesis using a cascade formant filter bank where the coefficients of the recursive filters are abruptly changed every M samples.

In this technique several copies of each filter are used. When the coefficients need to be changed, the input is switched to the next filter and its coefficients are changed and the state vector cleared. The outputs of the filters are summed so that the decay of other filters and the starting transient of the "newest one" are all present in the output signal. Verhelst and Nilens (1986) called this 'the modified-superposition strategy' but the term 'input-switching method' suggested by Jot† appears to be more intuitive.

Verhelst and Nilens (1986) point out that the responses of the filters gradually attenuate after their input is disconnected. For this reason it is not needed to increase the number of parallel filters all the time but to run only a couple of filters so that the "oldest" filter is neglected at the time of coefficient change. In speech synthesis, a free-running filter can be disconnected after some 20 ms (Verhelst and Nilens, 1986). Naturally, the removed filter can be recycled so that its state is cleared and its coefficients are given the values of the next target filter. In practice, some 2 to 5 filters per time-varying filter need to be implemented.

Verhelst and Nilens report that the input-switching method is immune to formant-labeling errors that can cause large perturbations in the output signal of a conventional cascade speech synthesizer.

Updating of the State Vector

The most general approach to transient suppression was presented by Zetterberg and Zhang (1988) by means of a state-space formulation of the recursive filter. Their main result was that, assuming a stationary input signal, every change in filter coefficients should be accompanied by an appropriate change in the internal state variables. This guarantees that the filter switches directly from one state to another without any transient response. The Zetterberg–Zhang (ZZ) method can *completely eliminate the transients* but it does require that all the past input samples are known. This makes the approach impractical as such but provides a good starting point for more efficient approximate algorithms. In the next section we build on the ZZ method. First we review this technique in detail.

Let us consider a recursive filter structure that can be expressed in the state-variable form as

$$\mathbf{v}(n+1) = \mathbf{F}\mathbf{v}(n) + \mathbf{q}x(n) \tag{3.144a}$$

$$y(n) = \mathbf{g}^T\mathbf{v}(n) + g_0 x(n) \tag{3.144b}$$

where $x(n)$ and $y(n)$ are the input and output signal of the filter, respectively. The column vector $\mathbf{v}(n)$ contains the state variables, and the matrix $\mathbf{F}$, vectors $\mathbf{q}$ and $\mathbf{g}$, and coefficient g_0 contain the filter coefficients. These matrices and vectors depend on the implementation structure of the filter.

The state-variable vector can be expressed as a function of the input signal $x(n)$ and coefficient matrices when the coefficients have been changed at time index n_c, that is (Zetterberg and Zhang, 1988)

† J.–M. Jot, personal e-mail communication, Sept. 29, 1995.

$$\mathbf{v}(n) = \begin{cases} \mathbf{F}_1^n \mathbf{v}(0) + \sum_{k=0}^{n-1} \mathbf{F}_1^{n-1-k} \mathbf{q} x(k), & 0 < n \le n_c \\ \mathbf{F}_2^{n-n_c} \mathbf{v}(n_c) + \sum_{k=n_c}^{n-1} \mathbf{F}_2^{n-1-k} \mathbf{q} x(k), & n > n_c \end{cases} \tag{3.145}$$

where $\mathbf{v}(0)$ is the initial state of the filter, and $\mathbf{F}_1$ and $\mathbf{F}_2$ are the coefficient matrices before and after sample index n_c, respectively. After the coefficients have been changed, the state vector can be expressed as [substituting the upper form of Eq. (3.145) into the lower one and elaborating]

$$\mathbf{v}(n) = \mathbf{F}_1^{n-n_c} \mathbf{F}_2^{n_c} \mathbf{v}(0) + \sum_{k=0}^{n-1} \mathbf{F}_2^{n-k-1} \mathbf{q} x(k) + \mathbf{F}_2^{n-n_c} \Delta\mathbf{v}(n_c) \tag{3.146a}$$

with

$$\Delta\mathbf{v}(n_c) = \sum_{k=0}^{n_c-1} \left[\left(\mathbf{F}_1^{n_c-k-1} - \mathbf{F}_2^{n_c-k-1} \right) \mathbf{q} \right] x(k) \tag{3.146b}$$

The first term in (3.146a) is the contribution of the initial conditions to the state of the filter. We assume that the initial transient has died out so that this term can be neglected. The second term is the response of the filter to the input after the parameters have changed, and the last term represents the transient in the state vector due to coefficient change. Note that $\Delta\mathbf{v}(n_c)$ depends on the difference of the coefficient matrices $\mathbf{F}_1$ and $\mathbf{F}_2$ which implies that a small change in the coefficients causes a transient response with a smaller amplitude than a large change.

A way to completely eliminate the transient caused by the change of coefficients is to subtract the term $\Delta\mathbf{v}(n_c)$ from the state vector at time n_c (Zetterberg and Zhang, 1988). Equivalently, the state vector can be replaced with the following vector (Välimäki *et al.*, 1995b, 1995d)

$$\mathbf{v}_2(n_c) = \sum_{k=0}^{n_c-1} \mathbf{F}_2^{n_c-k-1} \mathbf{q} x(k) \tag{3.147}$$

A more pragmatic way to view the ZZ method of transient elimination is to consider that a copy of a recursive filter is running for each coefficient set that is ever needed in the application. All these filters are connected to the input of the system, but only one of them (the one with the appropriate coefficient set) is connected to the output at a time. The change of coefficients is then executed by selecting the filter with the appropriate coefficient set as illustrated in Fig. 3.26 for two filters. It is easy to understand that there will not be transients because of change of filter characteristics (assuming that the starting transient has died out) since all the filters are in the steady state all the time.

A serious drawback of this technique is that one filter is needed for each coefficient set. In many applications, such as in fractional delay filtering, practically a continuum of parameter values is desired, and this implies a very large number of filters running in parallel.

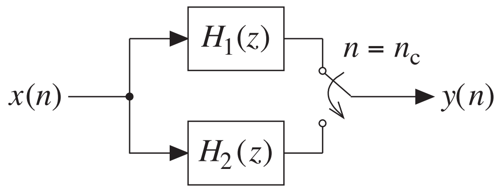

Fig. 3.26 A transientless implementation of a time-varying recursive filter according to Zetterberg and Zhang (1988). A single change of filter characteristics [from $H_1(z)$ to $H_2(z)$] at time $n = n_c$ is considered.

3.5.5 A Novel and Efficient Method for Elimination of Transients

The motivation for the development of a new transient elimination method was to find a practical way to update the state variables of a recursive filter when the filter coefficients are changed abruptly. It is clearly impractical to compute the state vector for a given coefficient set starting from time 0. We present a solution for transient suppression that gives an acceptable performance at a reasonable implementation complexity. This method has been initially proposed by Välimäki *et al.* (1995b, 1995d).

The new approach is applicable to any recursive filter structure but it is especially well suited for the *direct form (DF) II structure*. The DF II structure is described by the following state-variable presentation:

$$\mathbf{v}(n) = \begin{bmatrix} v_1(n) & v_2(n) & \cdots & v_N(n) \end{bmatrix}^T \tag{3.148a}$$

$$\mathbf{F} = \begin{bmatrix} -a_1 & -a_2 & \cdots & -a_{N-1} & -a_N \\ 1 & 0 & & 0 & 0 \\ 0 & 1 & & & \vdots \\ \vdots & & \ddots & 0 & 0 \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \tag{3.148b}$$

$$\mathbf{q} = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}^T \tag{3.148c}$$

$$\mathbf{g} = \begin{bmatrix} b_1 - b_0 a_1 & b_2 - b_0 a_2 & \cdots & b_N - b_0 a_N \end{bmatrix}^T \tag{3.148d}$$

$$g_0 = b_0 \tag{3.148e}$$

where a_k and b_k are the denominator and numerator coefficients of the recursive filter, respectively ($k = 0, 1, 2, ..., N$). The coefficient a_0 is equal to 1. The next-state and output equations of the state-variable representation are given by Eq. (3.144a) and (3.144b), respectively. The transfer function of the recursive filter is

$$H(z) = \frac{B(z)}{A(z)} = \frac{b_0 + b_1 z^{-1} + \ldots + b_N z^{-N}}{1 + a_1 z^{-1} + \ldots + a_N z^{-N}} \tag{3.149}$$

A key observation is that the state vector of a DF II structure is affected by the past input values only in proportion to the magnitude of the impulse response $h_r(n)$ of its recursive part, that is

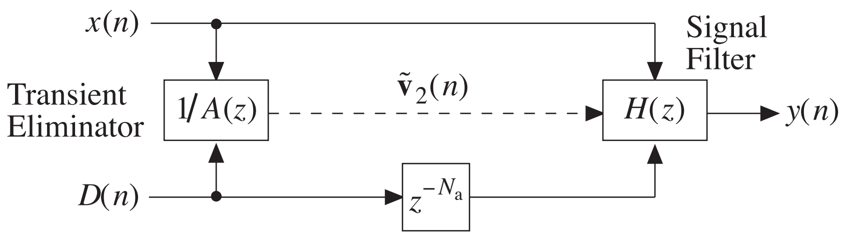

Fig. 3.27 The implementation of the novel transient elimination technique.

$$h_{\mathrm{r}}(n) = \begin{cases} 0, & n < 0 \\ 1, & n = 0 \\ \mathbf{q}^T \mathbf{F}^{n-1} \mathbf{q}, & n > 0 \end{cases} \tag{3.150}$$

This is because vector $\mathbf{g}$, that contains the feedforward coefficients, does not affect the state vector in (3.144a). The elements of the updated state vector (at time $n = n_{\mathrm{c}}$)

$$\mathbf{v}_2(n_{\mathrm{c}}) = \begin{bmatrix} v_{2,1}(n_{\mathrm{c}}) & v_{2,2}(n_{\mathrm{c}}) & \cdots & v_{2,N}(n_{\mathrm{c}}) \end{bmatrix}^T \tag{3.151}$$

can be rewritten by means of the IR $h_{2,\mathrm{r}}(n)$ of the recursive part of the filter $H_2(z)$:

$$v_{2,m}(n_{\mathrm{c}}) = \sum_{k=0}^{n_{\mathrm{c}}-1} h_{2,\mathrm{r}}(k) x(n_{\mathrm{c}} - m + 1 - k) = y_{\mathrm{r}}(n_{\mathrm{c}} - m + 1), \quad m = 1,2,\ldots,N \tag{3.152}$$

where $y_{\mathrm{r}}(n)$ is the output signal of the recursive part of the IIR filter.

Thus, the knowledge of the *effective length* of the impulse response $h_{\mathrm{r}}(n)$ (i.e., how many values of this impulse response are observably nonzero for the application) helps to estimate how many past input samples need to be taken into account in updating the state vector. The effective length of an infinite impulse response can be determined based on its time constant or by specifying an energy-based criterion. These approaches are discussed in Appendix B of this thesis. Denoting the effective length of the filter's impulse response (in samples) by N_{a} we can replace expression (3.147) by (Välimäki *et al.*, 1995b, 1995d)

$$\tilde{\mathbf{v}}_2(n_{\mathrm{c}}) = \sum_{k=n_{\mathrm{c}}-N_{\mathrm{a}}}^{n_{\mathrm{c}}-1} \mathbf{F}_2^{n_{\mathrm{c}}-k-1} \mathbf{q} x(k) = \sum_{k=1}^{N_{\mathrm{a}}} \mathbf{F}_2^{k-1} \mathbf{q} x(n_{\mathrm{c}} - k) \tag{3.153}$$

Let us consider implementation of the transient elimination technique when the time-varying filter $H(z) = B(z)/A(z)$ is implemented using the DF II structure. The transient elimination method then works as shown in Fig. 3.27: the input signal $x(n)$ is fed into two systems, the actual IIR filter $H(z)$ (hereafter called the *signal filter*) and the *transient eliminator* that consists of its recursive part $1/A(z)$. When the filter coefficients are changed, the coefficients of the eliminator are updated immediately. The coefficients of the signal filter are updated N_{a} samples after that, and at the same time the state vector is copied from the transient eliminator to the signal filter. The parameter N_{a} is called the *advance time* (in samples) of the transient eliminator. The larger the value of this parameter, the more the transient signal is being suppressed.

The state variables of the transient eliminator are updated according to the following

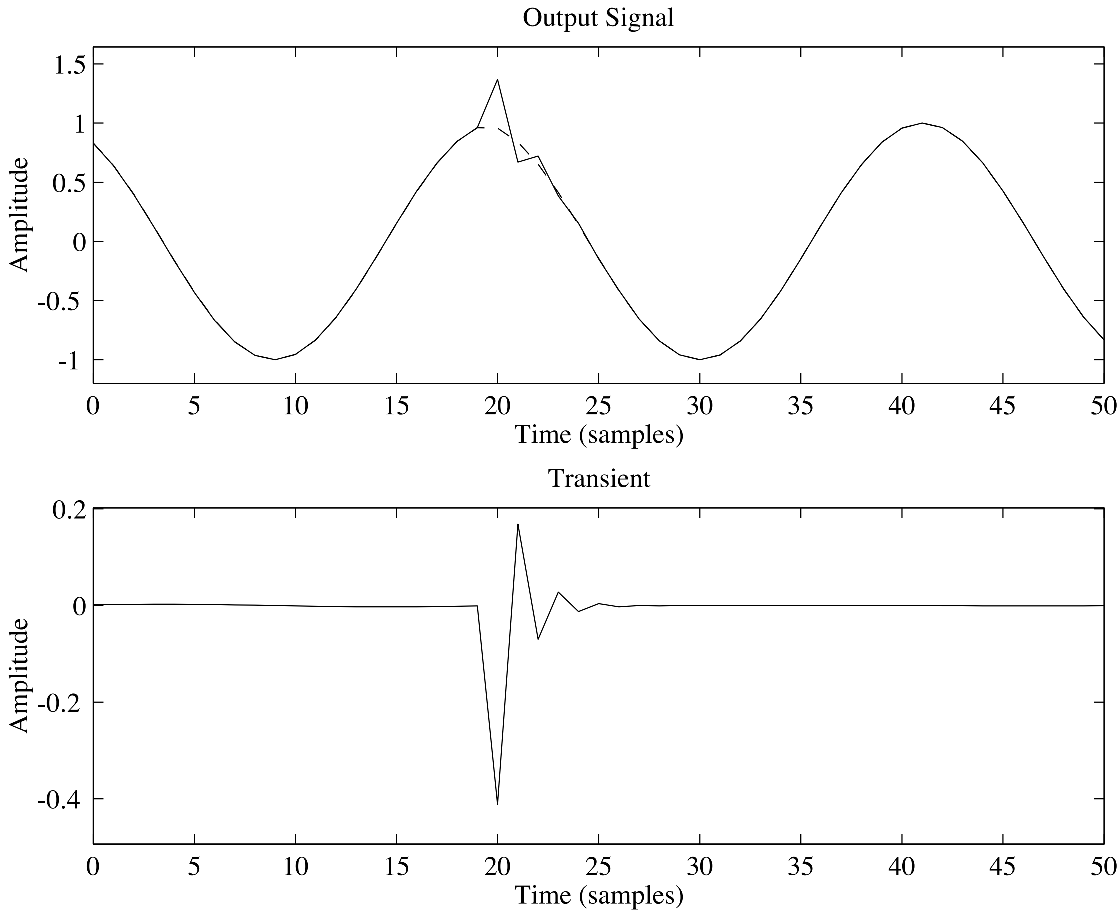

Fig. 3.28 The output signal of a first-order Thiran allpass filter when the delay parameter is changed from 1.42 to 0.42 at time index 20. The upper part shows the output signal of the time-varying filter without transient elimination (solid line) and the ideal output signal (dashed line). The lower part is the transient signal which is obtained as the difference of the two upper signals.

equation

$$v_{2,k}(n) = v_{2,k-1}(n-1) \quad \text{for } k = 2,3,\ldots,N \tag{3.154a}$$

$$v_{2,1}(n_{\mathrm{c}}) = x(n) + \sum_{k=1}^{N}(-a_k)v_{2,k}(n-1) \tag{3.154b}$$

It is seen that Eqs. (3.154a) and (1.154b) give a state-variable description of an *allpole filter* without output. The output signal of the transient eliminator is not needed since the purpose of this filter is to provide the state vector for the signal filter.

This implementation technique requires that the coefficient update is deferred by N_{a} sample cycles and during that time the eliminator is executed in parallel with the signal filter as illustrated in Fig. 3.27. If only one eliminator is used, it is then possible to change the coefficients every N_{a}th sample at the most. This lag can be compensated for if the next coefficient values are known beforehand at least N_{a} samples prior to the change. Then it is not necessary to defer the coefficient update but instead the eliminator can be started N_{a} samples before the actual time of change.

The above method of transient elimination can be used with practically all kinds of

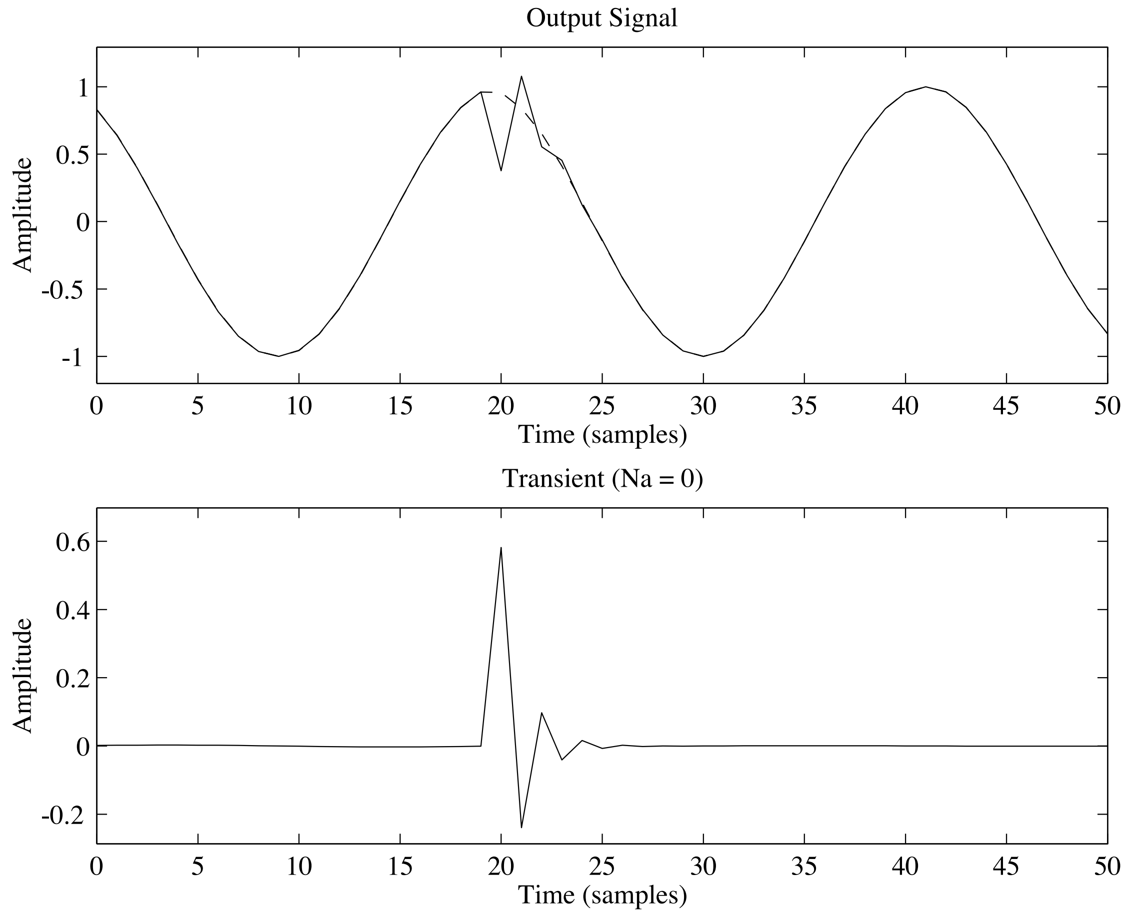

Fig. 3.29 The same example as in Fig. 3.28 but now the signal with the solid line is obtained using the new transient elimination technique with advance time N_a = 0. This corresponds to clearing the state of the allpass filter at time index n_c.

recursive filters. The technique is most efficiently implemented when the feedforward and feedback part of the filter have their own state variables or when only the recursive part affects the state of the filter, such as in the case of the DF II structure. In these cases the transient eliminator can be an allpole filter. On the other hand, if the state variables depend on both the recursive and the nonrecursive parts, the transient eliminator must be a pole-zero filter.

Two-Speed Implementation of the New Method

The new transient elimination technique can be modified to enable *instantaneous coefficient update* as often as needed—even every sampling interval (Välimäki *et al.*, 1995b, 1995d). This can be realized by keeping the last N_a input samples in a buffer memory and executing the necessary computations in a single sample interval by running the eliminator at N_a times the sampling rate of the signal filter. This two-speed method, of course, requires sufficiently fast hardware.

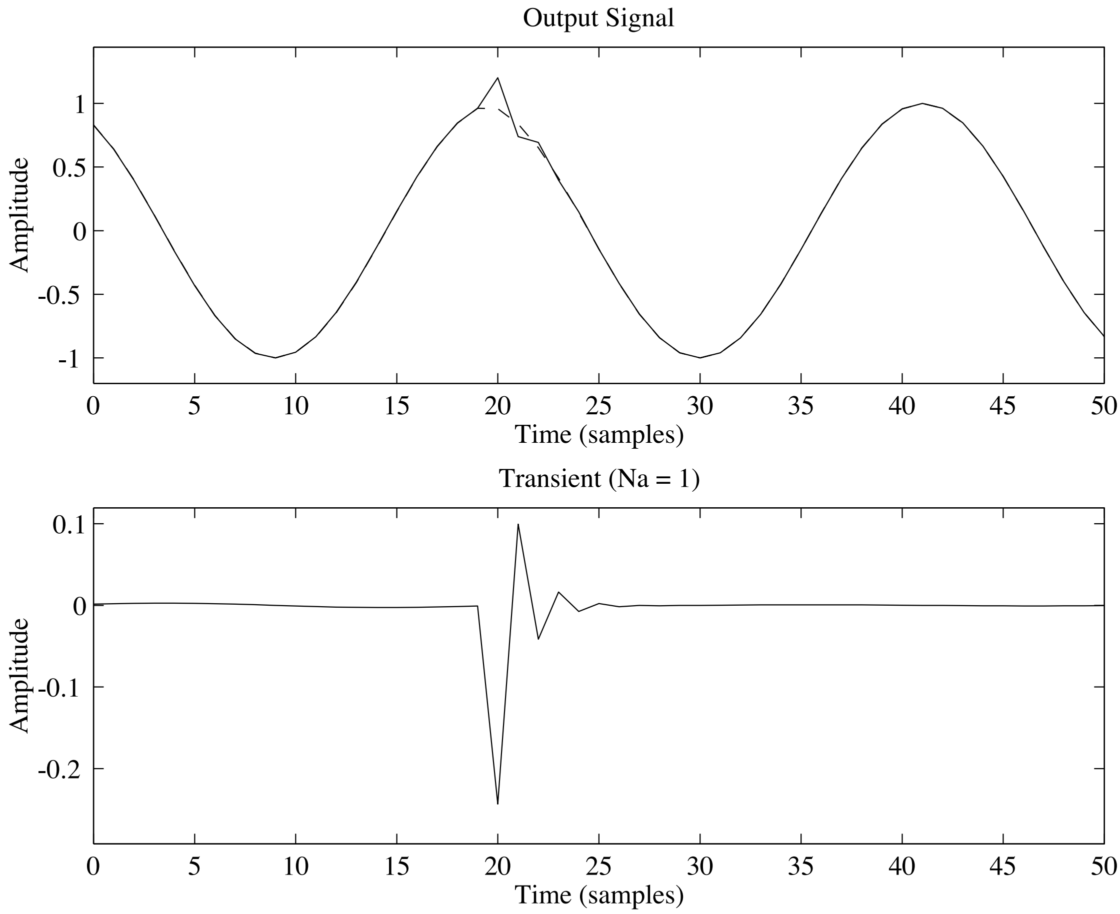

Fig. 3.30 The same example as in Fig. 3.28. The signal with the solid line is obtained using the new transient elimination technique with advance time $N_a = 1$.

3.5.6 Examples with a Time-Varying Thiran Allpass Filter

We have experimented with the new transient elimination method described in Section 3.5.5 to verify its quality. In the following examples, the input signal is a low-frequency sine signal. The delay parameter D of a first-order Thiran allpass filter is changed from 1.418 to 0.418 at time index $n_c = 20$. The corresponding pole radii are 0.173 and 0.410, respectively. Note that the decay rate of the transient is primarily determined by the pole of the target filter (i.e., Thiran allpass filter with $D = 0.418$).

The result without transient elimination is given in Fig. 3.28. The upper part shows the output signal of the filter with a solid line and the output signal of an 'ideal' time-varying FD filter with a dashed line. The ideal result is obtained by concatenating two shifted sine signals (i.e., output signals of two ideal interpolators). The lower part of Fig. 3.28 gives the difference of the two output signals. This signal can be interpreted as the *transient*. An exponentially decaying transient with peak value –0.41 is produced.

In Fig. 3.29, the state of the filter is cleared at the time of coefficient change. This case is obtained with the transient eliminator when the advance time parameter N_a is set to zero. The amplitude of the transient is now 0.58—substantially larger than without using the eliminator. This trick is definitely not worth using in practice.

Figure 3.30 presents the result obtained with advance time $N_a = 1$. The amplitude of the transient is 0.24. In effect, the first sample of the transient in Fig. 3.29 has now been truncated. It appears that N_a should be larger to suppress the transient.

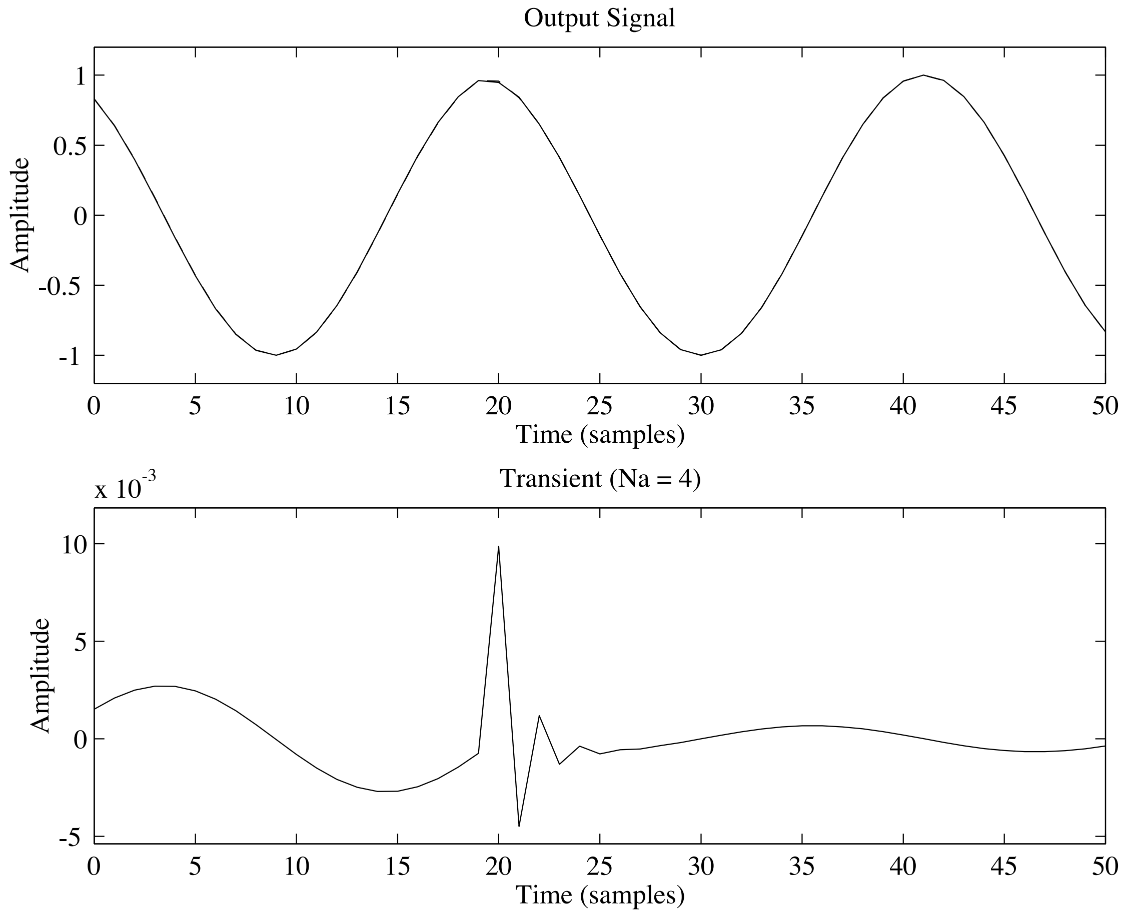

Fig. 3.31 The same example as in Fig. 3.28. The signal with the solid line is obtained using the new transient elimination technique with advance time $N_a = 4$. The slow oscillation in the lower figure is caused by the phase error of the Thiran allpass filter.

Finally, Fig. 3.31 shows the result with $N_a = 4$. Now the maximum amplitude of the difference signal is 0.0099. The low-frequency oscillation due to the phase approximation error of the Thiran allpass filter is clearly visible in Fig. 3.31.

3.6 Conclusions

Methods for the design of FIR and allpass fractional delay digital filters were studied. The maximally flat design of both the FIR and allpass interpolating filters has turned out to be useful in the context of waveguide modeling and they are also the only FD filter designs that have closed-form formulas for the coefficients. These techniques were thus discussed in detail. A novel implementation technique for Lagrange interpolation was introduced in Section 3.3.7. A new kind of parametrization was proposed for the Thiran allpass filter that was originally introduced by Laakso *et al.* (1992). The approximation errors were investigated and an optimal range for the delay parameter of Thiran allpass filters of different orders was determined.

Finally, this chapter discussed time-varying fractional delay filters. The FIR filters do not suffer from transients when their coefficients or locations of filter taps are changed—unless the filter has been implemented in an incorrect way. The output signal of an FIR filter will, however, be discontinuous at the time of the change. Recursive filters suffer from both problems: transients and discontinuities. Several transient elimination methods from recent DSP literature were reviewed.

A new technique to suppress the transients in recursive filters was proposed and its quality was analyzed. This method can suppress the transient signal as much as needed by increasing the advance time parameter. The amount of computation is directly proportional to this parameter. The implementation costs of this technique also depend on the structure of the recursive filter. The direct-form II realization results in an efficient method where effectively 1.5 filters need to be implemented and executed in parallel: the signal filter itself and a copy of its recursive part. This solution is computationally more efficient than any of the previously known transient elimination techniques.

In the next chapter, interpolated waveguide filter structures that employ fractional delay filters are developed.