Running FIR and IIR Filtering Using Multirate Filter Banks

MARTIN VETTERLI, MEMBER, IEEE

Abstract—**The computational complexity of running FIR and IIR filtering using multirate filter banks is considered. No restrictions are put on signal, filter, or block lengths. It is shown how to map long running convolutions into smaller ones by using filter banks based on aperiodic convolution algorithms and short-time Fourier transforms. With the proposed approach, good tradeoffs among computational complexity, system architecture, and input–output delay can be achieved.**

I. Introduction

COMPUTATION of FIR and IIR filters is a central and computationally intensive task in digital signal processing. As it is well known, if the output of a filter is computed for a block of samples at once, savings in the number of operations are possible. The price to be paid, however, is an additional input–output delay on the order of the block size. In real-time applications, such a delay might not be tolerable or has to meet certain bounds. Most proposed schemes for speeding up convolutions assume a block length that is larger than the filter size. Besides a large delay, this also leads to complex architectures that might be more difficult to realize in hardware than straight convolution would be.

Our point of view will thus be to explore the speeding up of convolution in the space of the following three parameters:

- computational complexity
- architectural complexity
- system delay.

While the computational complexity will be minimum for large block sizes, both the architectural complexity and the system delay are increasing functions of the block size, thus showing that depending on the application, the optimal block size might not be the one that leads to the minimum number of operations. As an example, note that small convolutions can be efficiently computed in hardware (using lookup tables or distributed arithmetic). Thus, rather than computing a long convolution by means of a Fourier transform of greater size (resulting in a huge delay), one might choose to reduce the problem to several smaller convolutions that can then be computed in parallel with special-purpose hardware. Thus, both the delay and the structural complexity are reduced.

When processing data blocks that step N samples forward after each computation, one gets a periodically time-varying system with period N. A good model of such a system is a multirate filter bank analysis/synthesis system with a sample rate change of N [18], [19]. The resulting system, when working properly, should be time invariant when seen from the outside. However, any nonidealities (such as roundoff errors, for example) will cause time variance to reappear, resulting in aliased versions of the input signal appearing in the output. Therefore, a complete multirate model is useful when studying such systems.

Our emphasis will be on simple algorithms since we consider the structural complexity as one of the parameters to be minimized. Most of the presented algorithms do not reduce the number of arithmetic operations when compared to optimal schemes. However, they seem much more suited for hardware or VLSI implementation.

Historically, the usefulness of block processing for computing convolutions was recognized soon after the introduction of the fast Fourier transform [17]. The scheme known as overlap add or save (since overlapping portions of the input or output signal are involved) has been widely described [13], [4]. A multirate filter bank interpretation of that scheme was given by Portnoff [16], [6]. The application of blockwise convolution to recursive filtering was first suggested by Gold and Jordan [9] and Burrus gave a general approach [5]. A number of publications has extended these basic ideas [11], [2], [12], [1], showing reductions in computational complexity and reduced sensitivity to roundoff noise.

Our presentation will go as follows. Section II introduces briefly the concept of a multirate system that will be used in later sections. Section III shows how aperiodic convolution algorithms can be used to derive multirate filter banks that will compute running convolutions more efficiently. Section IV presents multirate filter banks based on the short-time Fourier transform. This leads to overlap add/save schemes with no restrictions on block lengths. Section V considers recursive filtering and presents a simple algorithm that is efficient for most recursive filters of interest. Finally, Section VII points to aliasing effects that can appear in block processing algorithms.

Note that in the following, linear and running convo-

Manuscript received February 13, 1987; revised June 19, 1987. This work was supported in part by the American National Science Foundation under Grant CDR-84-21402.

The author is with the Department of Electrical Engineering, Columbia University, New York, NY 10027.

IEEE Log Number 8719424.

lution are used synonymously to indicate (noncircular) convolution of an infinite signal with a finite or infinite impulse response filter.

II. Multirate Systems

A digital signal processing system that works on blocks of data points, but advances by N samples between each successive processing step, performs essentially a subsampling of N on its internal variables. When the system is correctly designed, this subsampling has no effect on the output of the system (a typical example is the overlap method of fast convolution). However, in case of nonidealities, the system will show a periodically time-varying behavior (with period N), and aliased versions of the input signal will appear at the output [15], [18], [19].

A general multirate analysis/synthesis filter bank with M filters, subsampling by N, and channel modification is shown in Fig. 1. This is quite a general, yet simple, model for block processing algorithms. Below, we will derive systems of this type that will efficiently implement fast convolution algorithms for infinite input signals.

Recall that when a signal $x(n)$ with z transform $X(z)$ is subsampled by N to yield a signal $y(n)$ with z transform $Y(z)$, then [6]

$$Y(z) = \frac{1}{N} \cdot \sum_{n=0}^{N-1} X(W_N^n z^{1/N}), \qquad W_N = e^{-j2\pi/N}. \quad (1)$$

Conversely, if $y(n)$ is obtained by upsampling $x(n)$ by N, then

$$Y(z) = X(z^N). \quad (2)$$

Using relations (1) and (2), one can verify that the output of the block processing system depicted in Fig. 1 can be written as (see [18], [19] for details and note that boldface is used to denote vectors and matrices)

$$Y(z) = \frac{1}{N} \cdot [\mathbf{g}(z)]^T \cdot \mathbf{C}(z^N) \cdot \mathbf{H}_m(z) \cdot \mathbf{x}_m(z) \quad (3)$$

where

$$\mathbf{g}(z) = [G_0(z)\, G_1(z) \cdots G_{M-1}(z)]^T \quad (3a)$$

is the vector of output filters,

$$\mathbf{C}(z^N) = \text{diag}\,[C_0(z^N)\, C_1(z^N) \cdots C_{M-1}(z^N)]^T \quad (3b)$$

is a diagonal matrix containing the channel filters [note that these filters have Nth powers of z in (3)],

$$\mathbf{H}_m(z) = [\mathbf{h}(z)\, \mathbf{h}(W_N z) \cdots \mathbf{h}(W_N^{N-1} z)] \quad (3c)$$

is the modulated filter matrix containing the input filters and their modulated versions [following (1)],

$$\mathbf{h}(z) = [H_0(z)\, H_1(z) \cdots H_{M-1}(z)]^T \quad (3d)$$

is the vector of input filters, and

$$\mathbf{x}_m(z) = [X(z)\, X(W_N z) \cdots X(W_N^{N-1} z)]^T \quad (3e)$$

is the vector of modulated input signals [following (1)]. While (3) might seem complex at first sight, it represents a complete characterization of the block processing system of Fig. 1. Systems of this type have been analyzed in [18], [19] from where we state the following result without proof.

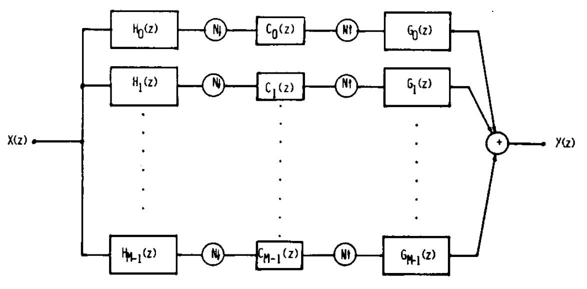


Fig. 1. M channel analysis/synthesis filter bank with subsampling by N and filtering of the channel signals. The subsampling by N is equivalent to moving the input by N samples between successive computations of the output.

Minimum Delay Property: An analysis/synthesis system with down/upsampling by N produces a delay of at least $N - 1$ samples if all filters involved are causal.

This property gives a lower bound on the input-output delay that will be achieved when using block processing algorithms. Note that the processing time is not taken into account, and would thus be added to this minimum delay.

III. Filter Banks Based on Aperiodic Convolution Algorithms

Efficient aperiodic convolution algorithms are based on good ways to evaluate polynomial products [13], [4]. It will be shown here that the polynomials can be z transforms of infinite signals, thus yielding good running convolution algorithms. Let us start with a simple yet useful example. Assume a three-channel filter bank with subsampling by 2 and with the following filters:

$$\mathbf{h}(z) = [z^{-1}, 1 + z^{-1}, 1]^T \quad (4a)$$

$$\mathbf{C}(z^2) = \text{diag}\,[H_0(z^2), H_0(z^2) + H_1(z^2), H_1(z^2)] \quad (4b)$$

$$\mathbf{g}(z) = [1 - z^{-1}, z^{-1}, z^{-2} - z^{-1}]^T. \quad (4c)$$

From (3), one can verify that $Y(z)$ equals

$$Y(z) = [z^{-1} \cdot H_0(z^2) + z^{-2} \cdot H_1(z^2)] \cdot X(z). \quad (5)$$

Note that the aliased version of the input [the term with z transform $X(-z)$] has disappeared in (5). Now, given a desired filter with z transform $H(z)$, one chooses $H_0(z^2)$ and $H_1(z^2)$ in the following way:

$$H_0(z^2) = \tfrac{1}{2} \cdot [H(z) + H(-z)] \quad (6a)$$

$$H_1(z^2) = \tfrac{1}{2} \cdot z \cdot [H(z) - H(-z)]. \quad (6b)$$

When $H(z)$ is a length-$2K$ FIR filter, then the two filters in (6a), (6b) are FIR filters of length K. Using (6) in

(5) leads to

$$Y(z) = z^{-1} \cdot H(z) \cdot X(z). \tag{7}$$

Thus, at the cost of one delay of z^{-1}, the filtering by $H(z)$ has been replaced by three filterings of half length and at half speed, that is, a gain of 25 percent in the number of multiplies per output sample. The algorithm is schematically shown in Fig. 2. Note that the input filter bank uses one addition per two input samples (because of the subsequent subsampling by two), while the output filter bank takes three additions per two output samples (because of the previous upsampling by two). Obviously, one can apply the algorithm again to the three subfilters, assuming that the delay is still tolerable. Table I shows the number of operations for computing the convolution of an infinite signal with a 32-tap FIR filter and using various breakups into smaller filters. The additional input and output additions might lead to the need for more bits of precision, and thus the gains would be diminished accordingly.

Note that when the original filter is symmetric or antisymmetric (an important case in practice), then the simple technique of pairing input samples corresponding to equal filter coefficients will reduce the multiplicative complexity by 50 percent. In this case, the reduced computation is not a convolution any more, making additional savings impossible. The technique proposed in (4) does not take full advantage of the filter symmetry (only one out of the three reduced filters is symmetric). Nevertheless, as shown in Table I, it will eventually outperform the pairing technique when iterated sufficiently. Note also that specific techniques can be developed for symmetric/antisymmetric filters where the reduced filters are still symmetric/antisymmetric (see [4, Sect. 9.4]).

The tree structure that appears when applying the algorithm recursively can lead to a complex architecture. One can instead expand the tree into parallel filters [8]. The filters appearing in the middle are the same as in the corresponding tree. The filters of the analysis and synthesis banks are obtained by passing filters on the other side of sampling rate changes (see [6] for details). Take, for example, a two-step radix-2 tree. An equivalent parallel bank will have nine filters, a subsampling by four, and channel filters of length one-fourth of the original filter. The filters of the analysis bank are obtained by

$$\boldsymbol{h}'(z) = \boldsymbol{h}(z) \otimes \boldsymbol{h}(z^2) \tag{8}$$

where $\otimes$ denotes the Kronecker product [4] and $\boldsymbol{h}(z)$ is as in (4a). A similar relation holds for the vector $\boldsymbol{g}(z)$ in the synthesis bank.

Note that a radix-4 tree could be built directly from an optimal four-point aperiodic convolution algorithm. This would lead to only seven filters of length one-fourth (instead of nine as discussed above), but the number of additions is tripled. Actually, an optimal aperiodic convolution algorithm for two sequences of length M and N requires $M + N - 1$ general multiplications [13]. This means that even with small lengths filters, one can lower

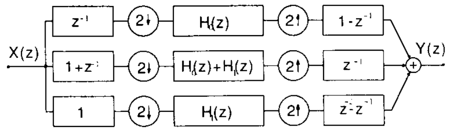


Fig. 2. Fast running convolution based on the two-point aperiodic convolution algorithm. The length-$2K$ running convolution has been replaced by three length-K running convolutions at half speed.

TABLE I
COMPUTATION OF THE RUNNING CONVOLUTION WHEN THE FILTER IS A 32-POINT FIR FILTER. COMPARISON OF THE VARIOUS BREAKUPS INTO SMALLER FILTERS

Method	Subsampling	Delay	Mult./ Point	Add./ Point
1 32-pt. FIR filter	1	0	32	31
3 16-pt. FIR filters	2	1	24	24.5
9 8-pt. FIR filters	4	3	18	17.5
27 4-pt. FIR filters	8	7	13.5	19.6
81 2-pt. FIR filters	16	15	10.125	21.3
243 1-pt. multiplic.	32	31	7.59	26.4

the computational complexity by evaluating more outputs at a time (for example, if three outputs of a length-2 filter are computed simultaneously, only four multiplies are used instead of three for two outputs). However, these algorithms become cumbersome in terms of the number of additions and their architecture is more involved.

Below, we show how to construct in general a multirate running filtering algorithm from an aperiodic convolution algorithm. The following property holds.

Running Filtering Property of Multirate Filter Banks Based on Aperiodic Convolution Algorithms: Given is an aperiodic convolution algorithm for length-N sequences that requires M multiplications ($M \geq 2N - 1$). In order to process an infinite signal with a FIR filter of length KN, one can use a filter bank of size M, subsampled by N, and having length-K channel filters. That is, the length-KN convolution is replaced by M length-K convolutions and at rate $1/N$.

Proof: We decompose all signals into their polyphase components of order N [6], [18], [19], that is,

$$X(z) = \sum_{n=0}^{N-1} z^{-n} \cdot X_n(z^N) \tag{9a}$$

$$H(z) = \sum_{n=0}^{N-1} z^{-n} \cdot H_n(z^N) \tag{9b}$$

$$Y(z) = \sum_{n=0}^{N-1} z^{-n} \cdot Y_n(z^N). \tag{9c}$$

Then, we evaluate the product $X(z) \cdot H(z)$ by using the fast aperiodic convolution algorithm, that is, we obtain the following product terms:

$$P_l(z^N) = \sum_{n=0}^{l} X_n(z^N) \cdot H_{l-n}(z^N) \qquad l = 0 \cdots 2N - 2 \tag{10}$$

where $X_n(z^N)$ and $H_n(z^N)$ are zero when n does not belong to $0 \cdots N-1$. Now, the final output is obtained from

$$Y_l(z^N) = P_l(z^N) + P_{l+N}(z^N), \qquad l = 0 \cdots N-1 \tag{11}$$

where $P_l(z)$ is zero when l is greater than $2N-2$. Because of (11), $N-1$ additions are needed for each set of N new outputs. Now, the evaluation of the terms of $P(z)$ involves only products of terms in z^N (besides the phase factors). Thus, these products can be subsampled by N. Then, because of the fast algorithm, there are only M such products to be evaluated, and thus the proof is completed. Fig. 3 shows schematically how the running convolution is computed at N times lower rate and using M reduced filters only.

The approach just described works when using an $n \times n$ aperiodic convolution algorithm. If one starts with an algorithm suited for sequences of length $m \times n$ (m different from n), one cannot use the filter bank approach because the subsampling of the signal and of the filter are not equal. One can still use matrix factorizations [4], but the intuitive interpretation of mapping a long running convolution into several smaller and subsampled convolutions is lost.

IV. Filter Banks Based on Short-Time Fourier Transforms

The short-time Fourier transform (STFT) is well known for its convolution property when used in an overlap save or add scheme [16], [6]. It is usually tacitly assumed that the length of the transform (denoted by M and equal to the number of channels of the multirate filter bank) is greater or equal to $L + N - 1$ where L is the length of the FIR filter and N is the subsampling factor of the multirate system. It is shown below that this condition is not necessary. In fact, a short-time Fourier transform can be used to map a filter of arbitrary length into several smaller and subsampled filters.

Consider an overlap add scheme based on a size M Fourier transform. In terms of a multirate filter bank implementation, the scheme uses the following filters:

$$H(z) = z^{N-1} + z^{N-2} + \cdots + z + 1 \tag{12a}$$

$$H_i(z) = z^{-N+1} \cdot H(W_M^i z) \tag{12b}$$

$$G(z) = 1 + z^{-1} + \cdots + z^{-M+1} \tag{12c}$$

$$G_i(z) = G(W_M^i z). \tag{12d}$$

The outputs of $H_i(z)$ are subsampled by N, and since these are length-N filters, this means that a complete set of N new samples is used in the analysis filter bank. The inputs to $G_i(z)$ are upsampled by N, which means that because these are length-M filters, a new set of outputs from the channels will overlap with $M - N$ current outputs of the synthesis filter bank. The multiplicative channel modification is given by

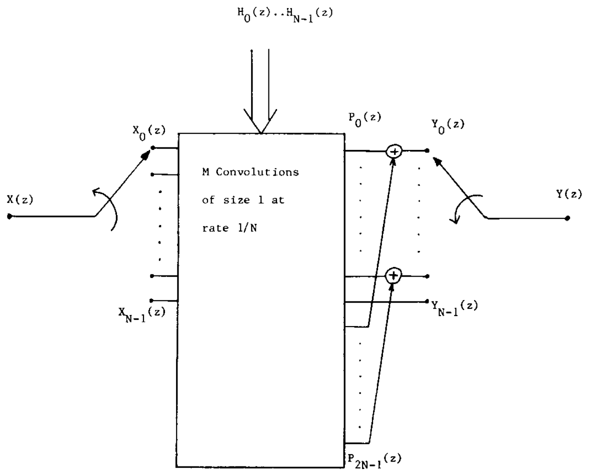


Fig. 3. General implementation of running convolution using shorter convolutions at slower speed. The number of small convolutions depends on the aperiodic convolution algorithm that is used, and the computation rate is reduced by a factor N.

$$C_i = \frac{1}{M} \cdot \sum_{l=0}^{L-1} W_M^{il} \cdot f_l \tag{13}$$

where f_l is a filter impulse response. An overlap save scheme is simply obtained by interchanging $H_i(z)$ with $G_i(z)$ in (12). In order to verify that the multirate filter bank defined by (12) and (13) indeed performs a running convolution when $M \geq L + N - 1$, we consider an input at time $-j$ ($j = 0 \cdots N-1$). This produces the following output in the ith channel after subsampling by N:

$$x_i = W_M^{i(N-1-j)}. \tag{14}$$

In the channel, this is multiplied by C_i (13). At time k, the output of the ith synthesis filter $G_i(z)$ is equal to

$$y_i(k) = W_M^{-ik} \cdot C_i \cdot x_i = \sum_{l=0}^{L-1} W_M^{i(N-1-j+l-k)} \cdot f_l. \tag{15}$$

At time k, the total output is obtained by summing (15) over all channels:

$$y(k) = \frac{1}{M} \sum_{l=0}^{L-1} \sum_{i=0}^{M-1} W_M^{i(N-1-j+l-k)} \cdot f_l. \tag{16}$$

When the exponent of W_M is not equal to zero, the second sum in (16) is zero because of the orthogonality of the roots of unity. Thus, the only term that does not disappear corresponds to

$$l = k + j - N + 1. \tag{17}$$

This means that an impulse at time $-j$ reproduces the filter impulse response delayed by $N - 1$ samples. Now, if the condition $M \geq L + N - 1$ is not respected, for example, if $L = M - N + 1 + d$, then for $j = -d + 1 \cdots 0$, there will be a wraparound. That is, inputs at different times will have different impulse responses, resulting in a periodically time-varying system [15]. A similar analysis can be made for the overlap save method as well. After this rapid analysis of the conventional overlap schemes, we can state the following property of multirate filter banks.

Running Filtering Property of Multirate Filter Banks Based on the Short-Time Fourier Transform: Assume a multirate filter bank of size M and subsampled by N with analysis and synthesis filters implementing the overlap add or save method of convolution [see (12)]. Given are M FIR filters of length K with z transforms

$$C_i(z) = C_{0,i} + C_{1,i}z^{-1} + \cdots + C_{K-1,i}z^{-K+1}. \quad (18)$$

Now, if the $C_{k,i}$ are of the form

$$C_{k,i} = \frac{1}{M} \cdot \sum_{l=0}^{L-1} W_M^{il} \cdot c_{kl} \quad (19)$$

and $L \leq M - N + 1$, then the output of the multirate filter bank will be a time-invariant running convolution of the input with a filter having the following z transform:

$$H(z) = z^{-N+1} \cdot \left(\sum_{k=0}^{K-1} z^{-kN} \sum_{l=0}^{L-1} c_{kl} \cdot z^{-l} \right). \quad (20)$$

That is, the impulse response has a basic delay of $N - 1$ samples, and is then made of the superposition of K impulse responses of length L, each shifted by kN samples ($k = 0 \cdots K - 1$). Fig. 4 gives a pictorial interpretation of (20).

The proof of (20) is relatively straightforward by using the superposition principle. Each of the sets $C_{k,i}$ ($i = 0 \cdots M - 1$) will produce a running filtering when applied as a multiplicative modification in the multirate filter bank (see (12)-(17) above). The corresponding z transform from input to output is

$$T_k(z) = z^{-N+1} \cdot \sum_{l=0}^{L-1} c_{kl} \cdot z^{-l}. \quad (21)$$

Now, the set $C_{k,i}$ is delayed by k samples in the channel [see (18)], that is, by Nk samples from input to output. Thus, the total transmission from input to output is obtained by superposing delayed versions of $T_k(z)$:

$$T(z) = z^{-N+1} \cdot \sum_{k=0}^{K-1} \cdot z^{-kN} T_k(z) \quad (22)$$

which is equal to (20). Note that only linearity was used, and thus the result holds as well if the system is time varying. Note that the condition $M \geq L + N - 1$ is also necessary because, otherwise, the system will become time varying.

Assume now that one wants to filter an infinite signal with a filter given by its z transform $H(z)$. We want to use a size M filter bank with subsampling by N. We require further that L in (19) is greater or equal to N or, equivalently, that $N \leq M/2$ (M even) or $N \leq (M + 1)/2$ (M odd) since the condition $M \geq L + N - 1$ has to be met. Typically, M is even and both N and L equal $M/2$.

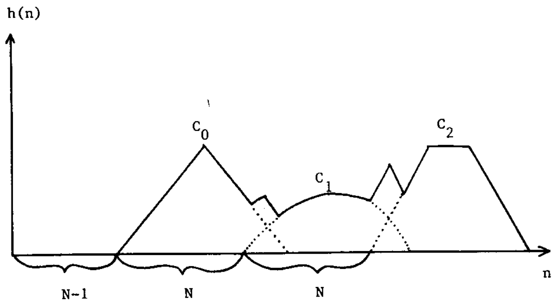


Fig. 4. Total impulse response as the superposition of K shifted impulse responses. In the overlap regions, the impulse responses add up.

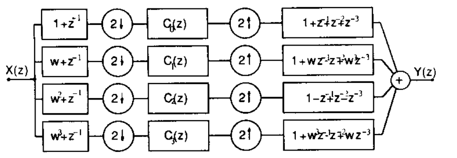


Fig. 5. Running convolution using a four-point short-time Fourier transform analysis/synthesis filter bank. The filters $C_i(z)$ are derived from the desired filter $H(z)$ following (24) and W stands for the fourth root of unity.

In order to use the short-time Fourier transform-based filter bank, one can decompose $H(z)$ as follows:

$$H(z) = \sum_{k=0}^{K-1} (h_{kN}z^{-kN} + h_{kN+1}z^{-kN-1} + \cdots + h_{(k+1)N-1}z^{-(k+1)N+1}) \quad (23)$$

where we assumed that the filter length was a multiple of N. Now, we use the following channel filters [see (18)-(19)]:

$$C_{k,i} = \frac{1}{M} \cdot \sum_{l=0}^{N-1} W_M^{il} \cdot h_{kN+l} \quad (24)$$

and thus, following (20)-(22), the output of the multirate filter bank is

$$Y(z) = z^{-N+1} \cdot H(z) \cdot X(z). \quad (25)$$

In short, a length-KN convolution was mapped into M length-K convolution subsampled by N, and this by using size-M analysis and synthesis filter banks. Fig. 5 shows a simple example using a size-4 short-time Fourier transform (note that this example is for illustration purposes only since it does not actually improve the computational complexity). The process can be reiterated on the subfilters as well. From an architectural point of view, this means that a single filter bank device can be multiplexed to map an arbitrary long running convolution into multiple arbitrary short convolutions (at the limit, pointwise multiplications).

Again, there is a possible tradeoff among the number of operations, the architectural complexity, and the overall delay. We will show this by an example. For simplicity, all signals and filters are supposed to be complex, and we consider only transforms with lengths that are powers

TABLE II
LINEAR CONVOLUTION OF AN INFINITE SIGNAL WITH A 32-POINT FIR FILTER (BOTH ARE COMPLEX VALUED). EXAMPLES OF POSSIBLE ARCHITECTURES

Method	Delay	Mult./ Point	Architecture
a) Direct	0	96	Simple
b) With a 128-point STFT subsampled by 97	96	15	Complex (128-point FFT's)
c) With a 16-point STFT subsampled by 8	7	29	Medium complexity (16-point FFT's)
d) Same as c) but with a fast algorithm for the four-point channel filters	31	18.5	Medium complexity, same as c) plus a simple fast algorithm for the channel filters

of 2. Assume that we want to filter a complex signal with a length-32 complex FIR filter. The minimum number of multiplies is achieved by taking a length-128 circular convolution in an overlap add or save scheme [7]. This leads to about 15 multiplies per output point. The system delay is 96 samples, and the architecture is complex (length-128 FFT's are required). Minimum delay is achieved by direct computation, which requires, however, 96 multiplies per output (using the three real multiplications algorithm for the complex product). Using the multirate filter bank approach, one can devise architectures that will achieve any desired tradeoff between these two extreme solutions. Take, for example, a size-16 short-time Fourier transform subsampled by 8. This leads to 16 length-4 channel filters. For eight outputs, one has to evaluate two length-16 FFT's (20 multiplications each) and one output of each channel filter (16 times four times three multiplications). This leads to 29 multiplies per output, but with a delay of only seven samples. Now, the length-4 channel filters can be evaluated with the algorithm of the previous section. Using the radix-2 scheme, this leads to 2.25 instead of four complex multiplications. The total is then 18.5 multiplications for each new output of the complete filter bank. The delay is increased by three samples in the channel (due to the algorithm for a four-point filter), which results in a total delay of 31 samples from input to output. Table II compares the various schemes that where just described.

Of course, there are many more ways to compute the 32-point running convolution (using various sizes of STFT, aperiodic convolution algorithms, and combinations thereof). The purpose of this section was mainly to show that the traditional way to compute overlap add or save fast convolution (by using an FFT of about twice the filter length) is by far not the only way to speed up the running convolution. Actually, other schemes might be more attractive in terms of architectural simplicity and system delay.

V. Recursive Filtering

If the desired filter has both poles and zeros, it still can be implemented with a multirate filter bank by using the following expansion [3], [6], [18], [19]:

$$H(z) = \frac{N(z)}{D(z)} = \frac{N(z)\,D'(z)}{D(z^N)} \tag{26a}$$

where $D'(z)$ is an FIR filter given by

$$D'(z) = \frac{D(z^N)}{D(z)}. \tag{26b}$$

Since the channels are subsampled by N, the channel filters will be of the form

$$C_i(z) = \frac{C_{N,i}(z)}{D(z)} \tag{27}$$

where the numerator $C_{N,i}(z)$ is obtained from $N(z)\,D'(z)$ with the methods developed so far for FIR filters. From a computational point of view, the transformation (26)–(27) does not reduce the amount of computations (actually, it is increased since we now have M channels at rate $1/N$, $M > N$). However, the parallelism that is gained might be of interest.

Methods for reducing the number of operations in IIR filters usually assume a block size larger than the size of the numerator or denominator filter. We will instead look at arbitrary block sizes. Assume an all-pole filter (the pole/zero case can be treated as the cascade of a FIR and an all-pole filter) whose output is given by the following recursive equation [14]:

$$y(n) = x(n) - a_1 y(n-1) - \cdots - a_N y(n-N). \tag{28}$$

Start with $N = 2$, and compute two outputs at a time. This can be written as

$$\begin{pmatrix} y(n+1) \\ y(n) \end{pmatrix} = \begin{pmatrix} x(n+1) \\ x(n) \end{pmatrix} - \begin{pmatrix} y(n) & y(n-1) \\ y(n-1) & y(n-2) \end{pmatrix} \cdot \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}. \tag{29}$$

The matrix–vector product in (29) is the usual two-point running convolution. It can be factored so as to use only three multiplications, but only if the recursive equation is still solvable. Let us consider the factorization

$$\begin{pmatrix} y(n+1) \\ y(n) \end{pmatrix} = \begin{pmatrix} x(n+1) \\ x(n) \end{pmatrix} - \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} y(n) - y(n-1) & 0 & 0 \\ 0 & y(n-1) & 0 \\ 0 & 0 & y(n-1) - y(n-2) \end{pmatrix} \cdot \begin{pmatrix} a_1 \\ a_1 + a_2 \\ a_2 \end{pmatrix}. \tag{30}$$

Now, it is clear that by evaluating (30) first for $y(n)$ and second for $y(n+1)$, then the right side is always defined. Since $a_1 + a_2$ is supposed to be precomputed, (30) takes three multiplications and six additions per two output values [instead of four multiplications and additions for a direct evaluation of (28)]. Now, if the block size is larger, one can consider the elements of (29) as block vectors and block matrices. Then, we get the following set of equations [from (30)]:

$$\begin{pmatrix} \mathbf{y}_1 \\ \mathbf{y}_0 \end{pmatrix} = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_0 \end{pmatrix} - \begin{pmatrix} \mathbf{I} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & -\mathbf{I} \end{pmatrix} \cdot \begin{pmatrix} (\mathbf{Y}_0 - \mathbf{Y}_1)\mathbf{a}_1 \\ \mathbf{Y}_1(\mathbf{a}_1 + \mathbf{a}_2) \\ (\mathbf{Y}_1 - \mathbf{Y}_2)\mathbf{a}_2 \end{pmatrix} \tag{31}$$

where

$$\mathbf{y}_0 = [y(n + N/2 - 1) \cdots y(n)]^T \tag{31a}$$
$$\mathbf{y}_1 = [y(n + N - 1) \cdots y(n + N/2)]^T \tag{31b}$$
$$\mathbf{x}_0 = [x(n + N/2 - 1) \cdots x(n)]^T \tag{31c}$$
$$\mathbf{x}_1 = [x(n + N - 1) \cdots x(n + N/2)]^T \tag{31d}$$
$$\mathbf{a}_1 = [a_1 \cdots a_{N/2}]^T \tag{31e}$$
$$\mathbf{a}_2 = [a_{N/2+1} \cdots a_N]^T \tag{31f}$$

and $\mathbf{Y}_i$ are Hankel matrices [10] defined by

$$\mathbf{Y}_i = \begin{pmatrix} y(n + N - iN/2 - 2) & \cdots & y(n + N/2 - iN/2 - 1) \\ \vdots & \iddots & \vdots \\ y(n + N/2 - iN/2 - 1) & \cdots & y(n - iN/2) \end{pmatrix}. \tag{31g}$$

Since the three matrices are Hankel as well (sum and difference of Hankel matrices are Hankel matrices), one can apply the algorithm recursively. It can be verified that when solving (31), $y(n+i)$ depends only on $y(n+i-k)$, $k = 1 \cdots i$, and thus the $y(n)$'s can be found iteratively. Note that while this is true for the above algorithm (which is based on aperiodic convolution), it would not hold for FFT-based schemes (which use cyclic convolution where each output depends on all inputs). Note that the above algorithm can also be used to break long feedback filters into smaller ones, thus reducing the system delay.

TABLE III
COMPUTATION OF AN ALL-POLE FILTER USING THE FFT OR THE APERIODIC CONVOLUTION ALGORITHM

Filter Length	Number of Mult. per Point Using Two Circular Convolutions	Number of Mult. per Point Using the Aperiodic Convolution Algorithm
2	5	1.5
4	7.16	2.25
8	9.58	3.38
16	12.13	5.06
32	14.73	7.59
64	17.15	11.39
128	19.44	17.09
256	21.71	25.62

Transform techniques have been used for implementing IIR filters [5]. However, they become efficient for large block sizes and for relatively high-order filters (which is a seldom case in applications). The simple method just described can thus be attractive for low-order recursive filtering, and we will show this by comparing the two schemes. Assume an L pole filter and a block size M for the FFT. Take M greater than $2L$ and equal to a power of 2. The FFT scheme will thus use two cyclic convolutions of length M in order to yield $M - L$ correct output values (see [5] for details). The complexity for the cyclic convolution of real sequences is taken from [7]. The number of multiplications per output sample of this FFT scheme is compared in Table III to the number required by the above described algorithm.

As can be seen, the direct approach remains competitive up to relatively high filter orders ($L = 128$). This is mainly due to the fact that the FFT scheme requires two cyclic convolutions because the filter is all pole (in an FIR case, only one such convolution is required).

VI. ROUNDOFF ERRORS EFFECTS

Roundoff errors in block processing algorithms will actually lead to aliasing effects because the system is time varying. This is true for all block processing schemes (like traditional overlap add or save schemes) and not only for the algorithms presented in this paper. Obviously, this basic problem might lead to tighter precision requirements, that is, to an increase in the number of bits used in all calculations. This means that part of the savings in terms of number of arithmetic operations can be lost since the fewer operations become more complex.

Below, we show in the simple example of the algorithm given in (4) how aliasing appears in the output of a block processing algorithm. Assume that the roundoff errors can be modeled as additional noise on the channels of the system. Then, from (4b), we have

$$\begin{aligned} C(z) = \operatorname{diag}\,\big[&H_0(z) + N_0(z),\\ &H_0(z) + H_1(z) + N_1(z),\\ &H_1(z) + N_2(z)\big] \end{aligned} \tag{32}$$

where $H_0(z)$ and $H_1(z)$ are chosen as in (6) and $N_0(z)$, $N_1(z)$, and $N_2(z)$ are z transforms of the noise components produced by the roundoff process. Then, the output of the filter bank equals

$$\begin{aligned} Y(z) = z^{-1} \cdot H(z) \cdot X(z) + \tfrac{1}{2} \cdot \big[&(z^{-1} - z^{-2}) \cdot (N_0(z^2)\\ &- N_2(z^2)) + (z^{-1} + z^{-2}) \cdot N_1(z^2)\big] \cdot X(z)\\ + \tfrac{1}{2} \cdot \big[&(z^{-2} - z^{-1}) \cdot (N_0(z^2)\\ &- N_1(z^2)) + N_2(z^2)\big] \cdot X(-z). \end{aligned} \tag{33}$$

That is, an aliased component $X(-z)$ of the input $X(z)$ will appear at the output. Similar relations can be derived for any block processing algorithm, and thus aliased components ($N - 1$ of them where N is the subsampling factor) will always appear. The amplitude of these components (assuming similar noise in the various channels) is on the order of the noise amplitude times a factor M/N (where M is the number of channels and N is the subsampling factor).

The purpose of this section was to indicate a potential problem linked to block processing algorithms. It is our belief that a simulation of the finite precision effects is necessary in order to assign a sufficient number of bits to the variables and arithmetic units in such algorithms.

VII. Conclusion

The use of multirate filter banks for realizing running convolution has been investigated. It was shown how to map long convolutions into smaller, subsampled ones, and this without any restriction on filter, signal, or block length. Because of this absence of restriction on block length, it is possible to achieve any tradeoff among computational complexity, architectural complexity, and system delay (which is a function of the block length).

For the FIR case, filter banks based on aperiodic convolution algorithms as well as on short-time Fourier transforms have been derived. In the IIR case, a simple scheme (based on aperiodic convolution) was shown to be attractive for filters having up to 100 poles. Because of the internal subsampling inherent to block processing algorithms, the resulting systems are periodically time varying in general. It was thus shown how roundoff errors in the processing can lead to aliasing in the output.

Note that the FIR running convolution scheme has been generalized to the two-dimensional case as well [20] and is more efficient than FFT-based schemes for filter sizes up to 8×8.

As a conclusion, it was shown that there are numerous alternative algorithms for computing running convolution besides the classical overlap add or save schemes (that use block sizes larger than the filter length). The mapping of long convolutions into small, subsampled convolutions is attractive in hardware (VLSI), software (signal processors), and multiprocessor implementations since the basic building blocks remain convolutions which can be computed efficiently once small enough.

Acknowledgment

The author would like to thank the anonymous reviewers for their constructive criticism, J. S. White for his kind help, Y.-W. Bai for his careful reading of the manuscript, and P. Duhamel of CNET-Paris for fruitful discussions. It seems that results similar to what is reported above have been found independently by Duhamel's group as well.

References

[1] M. R. Azimi-Sadjadi and R. A. King, "Two-dimensional block processors—Structures and implementations," *IEEE Trans. Circuits Syst.*, vol. CAS-33, pp. 42-50, Jan. 1986.
[2] C. W. Barnes and S. Shinnaka, "Block-shift invariance and block implementation of discrete-time filters," *IEEE Trans. Circuits Syst.*, vol. CAS-27, pp. 667-672, Aug. 1980.
[3] M. G. Bellanger and J. L. Daguet, "TDM-FDM transmultiplexer: Digital polyphase and FFT," *IEEE Trans. Commun.*, vol. COM-22, pp. 1199-1204, Sept. 1974.
[4] R. E. Blahut, *Fast Algorithms for Digital Signal Processing.* Reading, MA: Addison-Wesley, 1984.
[5] C. S. Burrus, "Block realization of digital filters," *IEEE Trans. Audio Electroacoust.*, vol. AU-20, pp. 230-235, Oct. 1972.
[6] R. E. Crochiere and L. R. Rabiner, *Multirate Digital Signal Processing.* Englewood Cliffs, NJ: Prentice-Hall, 1983.
[7] P. Duhamel and M. Vetterli, "Improved Fourier and Hartley transform algorithms. Application to cyclic convolution of real data," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, pp. 818-824, June 1987.
[8] C. R. Galand and H. J. Nussbaumer, "New quadrature mirror filter structures," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, no. 3, pp. 522-531, June 1984.
[9] B. Gold and K. L. Jordan, "A note on digital filter synthesis," *Proc. IEEE*, vol. 56, pp. 1717-1718, Oct. 1968.
[10] T. Kailath, *Linear Systems.* Englewood Cliffs, NJ: Prentice-Hall, 1980.
[11] S. K. Mitra and R. Gnanasekaran, "Block implementation of recursive digital filters—New structures and properties," *IEEE Trans. Circuits Syst.*, vol. CAS-25, pp. 200-207, Apr. 1973.
[12] C. L. Nikias, "Fast block data processing via a new IIR digital filter structure," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp. 770-779, Aug. 1984.

[13] H. J. Nussbaumer, *Fast Fourier Transform and Convolution Algorithms.* Berlin, Germany: Springer, 1982.

[14] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing.* Englewood Cliffs, NJ: Prentice-Hall, 1975.

[15] L. Pelkowitz, "Frequency domain analysis of wraparound error in fast convolution algorithms," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-29, pp. 413-422, June 1981.

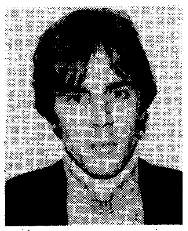
[16] M. R. Portnoff, "Time-frequency representation of digital signals and systems based on short-time Fourier analysis," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, pp. 356-372, Mar. 1987.

[17] T. G. Stockham, "High speed convolution and correlation," in *Proc. 1966 Spring Joint Comput. Conf., AFIPS*, vol. 28, 1966, pp. 229-233.

[18] M. Vetterli, "Analysis, synthesis and computational complexity of digital filter banks," Ph.D. dissertation, Ecole Polytechnique Fédérale de Lausanne, Switzerland, Apr. 1986.

[19] ——, "A theory of multirate filter banks," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, pp. 356-372, Mar. 1987.

[20] ——, "Running FIR and IIR filtering using multirate filter banks," Cen. for Telecommun. Res., Columbia Univ., New York, NY, Tech. Rep. 034, Feb. 1987.



Martin Vetterli (S'86-M'86) was born in Switzerland in 1957. He received the Dipl.El.-Ing. degree from the Eidgenösssische Technische Hochschule Zürich, Switzerland, in 1981, the Master of Science degree from Stanford University, Stanford, CA, in 1982, and the Doctorat ès Science degree from Ecole Polytechnique Fédérale de Lausanne, Switzerland, in 1986.

In 1982 he was a Research Assistant with the Department of Computer Science, Stanford University, and from 1983 to 1986 he was a Researcher at the Ecole Polytechnique. He has worked for Siemens, Switzerland, and AT&T Bell Laboratories, Holmdel, NJ. Since 1986 he has been at Columbia University, New York, NY, first with the Center for Telecommunications Research and now with the Department of Electrical Engineering where he is currently an Assistant Professor. His research interests include multirate signal processing, computational complexity, algorithm design for VLSI, and signal processing for telecommunications.

Dr. Vetterli is a member of the Editorial Board of *Signal Processing*. He was recipient of the Best Paper Award of EURASIP in 1984 and of the Research Prize of the Brown Bovery Corporation (Switzerland) in 1986.