BLOCK-RECURSIVE FILTERS AND FILTER-BANKS[1]

Gábor Péceli *Annamária R. Várkonyi-Kóczy*

Department of Measurement and Instrument Engineering,
Technical University of Budapest, H-1521 Budapest, Hungary
email: peceli@mmt.bme.hu koczy@mmt.bme.hu

ABSTRACT

Block-oriented signal processing techniques have exceptional role due to the availability of fast algorithms. However, if larger data segments are to be evaluated in real-time, the delay caused by the block-oriented approach is not always tolerable especially if the response time of our evaluating system is also specified. This can be exceptionally critical if the signal processing is related to feedback loops. In this paper block-oriented signal processing methods are combined with recursive ones. This combination reduces the delay problem caused by the block-oriented fast algorithms and at the same time keeps the computational complexity on relatively low level. Possibly the most original component of the suggested solution is the extension of given size signal transformer-bank channels (e.g. DFT channels) toward larger blocks simply via recursive averaging.

1. INTRODUCTION

In this paper the concept of block-recursive filters and filter-banks is introduced. These filters perform signal processing in two major consecutive steps. In the first step typically a conventional block-oriented operation like signal transformation or FIR filtering is executed while in the second one the previous results are block-recursively updated. This mechanism is in complete correspondence with the usual recursive schemes only the correcting term is replaced by the result of a block-oriented operation. If this latter enables decimation then the updating can be performed at lower rate.

The motivation to introduce such schemes is on one hand the possible applicability of fast algorithms while on the other the availability of partial results or estimates in case of long or possibly infinite input data sequences.

As a simple example consider the case of discrete Fourier transformation (DFT). For its evaluation fast algorithms (FFTs) are available. If we apply the FFT e.g. for two consecutive blocks of 1K input data then, by taking the average of the corresponding values, the DFT channels will represent every second channel of a 2K DFT. By recursive averaging this can be extended arbitrarily, obviously the number of the DFT channels will equal the initial block size. With proper frequency transposition technique we can operate FFTs parallelly, and have more DFT channels. By two parallel 1K FFTs we can calculate in the first step a signal transformation similar to a 2K DFT with smaller accuracy (that of the 1K DFT) while after processing the second block and averaging the complete 2K DFT will be available. With such and similar techniques the further processing of the data using their approximate frequency domain representation can start earlier and the not always tolerable side-effects of processing delay can be reduced.

The paper is organized as follows: In Section 2 first the block-recursive linear and exponential averagers are introduced. It is shown that there is complete formal correspondence with the standard averaging algorithms. This is followed by the derivation of the sliding block-averager. As a next step these averager schemes are generalized toward signal transformations and filter-banks. In Section 3 illustrative examples show the application of the suggested methods.

2. BLOCK-RECURSIVE AVERAGERS

In this Section the standard algorithms for recursive averaging are extended for data-blocks as single elements.

To illustrate the key steps first the block-recursive linear averaging will be introduced. For an input sequence $x(n)$, n =1,2, ..., the recursive linear averaging can be expressed as

$$y(n) = \frac{n-1}{n}\, y(n-1) + \frac{1}{n}\, x(n-1) \ . \qquad (1)$$

For $n \geq N$ the "block-oriented" linear averaging has the form of

$$X(n-N) = \frac{1}{N} \sum_{k=1}^{N} x(n-k) \ , \qquad (2)$$

while the block-recursive average can be written as

$$y(n) = \frac{n-N}{n}\, y(n-N) + \frac{N}{n}\, X(n-N) \ . \qquad (3)$$

If (3) is evaluated only in every Nth step, i.e. it is maximally decimated, then we can replace (3) with $n = mN$, m =1,2, ..., by

$$y(mN) = \frac{m-1}{m}\, y[(m-1)N] + \frac{1}{m}\, X[(m-1)N] \ , \qquad (4)$$

or simply

$$y(m) = \frac{m-1}{m}\, y(m-1) + \frac{1}{m}\, X(m-1) \qquad (5)$$

where m stands as block identifier. Note the formal correspondence with (1).

If the block identifier m in equation (5) is replaced by a constant $Q > 1$ then an exponential averaging effect is

[1]This work was supported by the Hungarian Fund for Scientific Research (OTKA) under contract T 017448.

achieved. This change makes the above block-oriented filter time-invariant and thus a frequency-domain characterization is also possible. In many practical applications exponential averaging provides the best compromise if both the noise reduction and the signal tracking capabilities are important. This is valid in our case, as well, however, in this paper only the linear and the sliding averagers are investigated because they can be used directly to extend the size of certain signal transformation channels.

A similar development can be provided for the sliding-window averagers. The recursive form of this algorithm is given for a block size of N by

$$y(n) = y(n-1) + \frac{1}{N}\left[x(n-1) - x(n-N-1)\right] \quad . \qquad (6)$$

If in (6) the input samples are replaced by preprocessed data, e.g. as in (2), then a block-recursive form is also possible:

$$y(n) = y(n-N) + [X(n-N) - X(n-2N)] \qquad (7)$$

which, however, has no practical meaning, since it gives back (2). But if the window size is integer multiple of N, e.g. MN, then the form

$$y(n) = y(n-N) + \frac{1}{M}\left[X(n-N) - X(n-(M+1)N)\right] \quad (8)$$

has real importance. If (8) is evaluated only in every Nth step, i.e. it is maximally decimated, then we can replace (8) with $n = mN$, m =1,2, ..., by

$$\begin{aligned} y(mN) &= y[(m-1)N] + \\ &+ \tfrac{1}{M}\left[X((m-1)N) - X((m-M-1)N)\right] \ , \end{aligned} \qquad (9)$$

or simply

$$y(m) = y(m-1) + \frac{1}{M}\left[X(m-1) - X(m-M-1)\right] \ , \ (10)$$

where m stands as block identifier. Note the formal correspondence with (6).

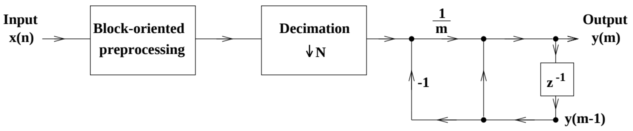

Figure 1. Block-recursive linear averaging signal processing scheme, $n = mN$

The generalization of these averaging schemes to signal transformations and/or filter-banks is straightforward. Only (2) should be replaced by the corresponding "block-oriented" operation. Figure 1 shows the block diagram of the linear averaging scheme. This is valid also for the exponential averaging except m must be replaced by Q. On Figure 2 the sliding-window averager is presented. These frameworks can incorporate a variety of possible transformations and corresponding filter-banks which permit decimation by the block-size. Standard references, e.g. [1] provide the necessary theoretical and practical background. The idea of transform-domain signal processing proved to be very efficient especially in adaptive filtering (see e.g. [2]). The contribution of this paper is directly applicable for the majority of these intensively cited algorithms. The most important practical advantage here compared to other methods is the early availability of rough estimates which can orientate in making decisions concerning further processing. The multiple-block sliding-window technique can be mentioned as a very characteristic algorithm of the proposed family. For this the computational complexity figures are also advantageous since using conventional methods to evaluate in "block-sliding-window" mode the transform of a block of MN samples would require M times an $(MN)*(MN)$ transformation, while the block-recursive solution calculates only for the last input block of N samples, i.e. M times an $(MN)*(N)$ "transformation".

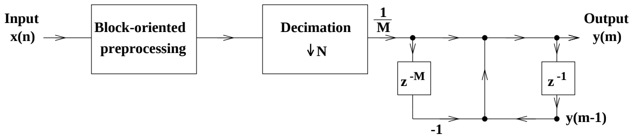

Figure 2. Block-recursive sliding-window averager scheme, $n = mN$, window-size: MN

As block-oriented preprocessing the DFT is the most widely used transformation for its fast algorithms (FFTs) and relatively easy interpretation. The above schemes can be operated for every "channel" of the DFT and after averaging this will correspond to the channel of a larger scale DFT. If linear averager is applied this scale equals mN while for sliding averager this figure is MN. The number of channels obviously remains N unless further parallel DFTs are applied. These additional DFTs have to locate their channel to the positions not covered by the existing channels. For the case where $M = 2$, i.e. only one additional parallel DFT is needed, where this positioning can be solved with the so-called complementary DFT which is generated using the Nth roots of -1. This DFT locates its channels into the positions π/N, 3 π/N, etc. For $M > 2$ proper frequency transposition techniques must be applied. If e.g. $M = 4$ then the full DFT will be of size $4N$ and four N-point DFTs (working on complex data) are to be used. The first DFT is responsible for the channels in positions 0, 8 $\pi/4N$, etc. The second DFT should cover the 2 $\pi/4N$, 10 $\pi/4N$, etc., the third the 4 $\pi/4N$, 12 $\pi/4N$, etc, and finally the fourth the 6 $\pi/4N$, 14 $\pi/4N$, etc. positions, respectively. The first DFT does not need extra frequency transposition. The second and the fourth process complex input data coming from a complex modulator which multiplies the input samples by $e^{j2\pi n/4N}$ and $e^{j6\pi n/4N}$, respectively. The third DFT should be a complementary DFT.

It is obvious from the above development that if a full DFT is required the sliding-window DFT must be preferred otherwise the number of the parallel channels should grow with m.

The majority of the transform-domain signal processing methods prefer the DFT to other possible transformations. However, there are certain applications where other orthogonal transformations can also be utilized possibly with much better overall performance. A further aspect of practical interest can be the end-to-end delay of the block-oriented processing. The time-recursive transformation algorithms described e.g. in [3] and [4] are sliding-window transforma-

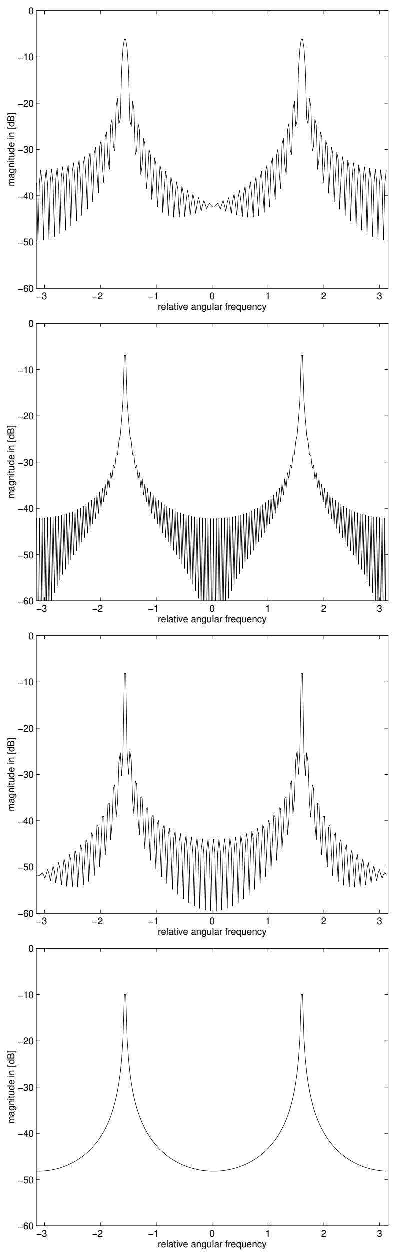

Figure 3. 256-channel DFT of a single sinusoid with $N = 64$, $m = 1, 2, 3, 4$, respectively

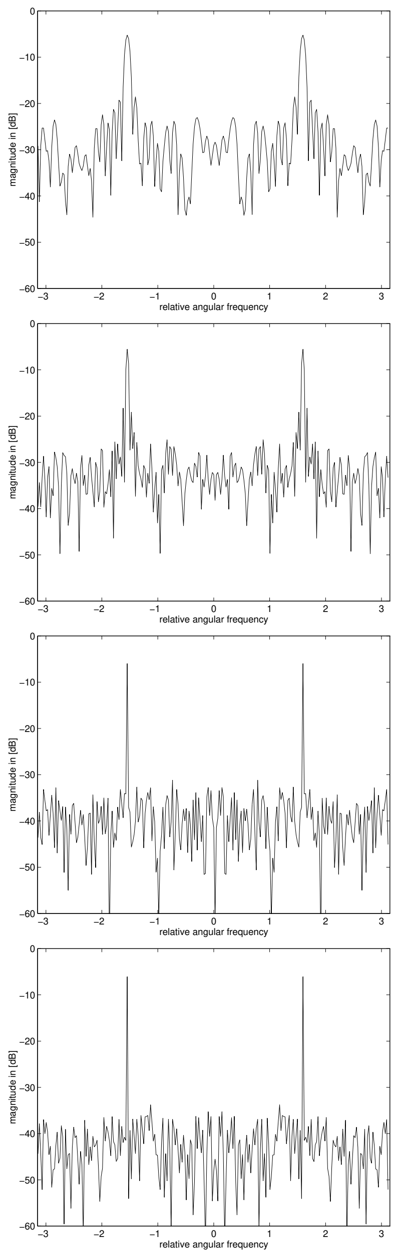

Figure 4. 256-channel DFT of a single sinusoid plus noise. $N = 64$, $m = 1, 2, 8, 16$, respectively

tions, i.e. filter-banks providing transform domain representation of the last input data block in every step. Decimation is not "inherent" as it is the case if the transformation is considered as a serial to parallel conversion, therefore the processing rate can be either the input rate, the maximally decimated one, or any other in between. These techniques are not fast algorithms, however, "produce" less delay as those block-oriented algorithms which start working only after the arrival of the complete input data block.

Very recently a fast polyphase filter-bank family has been reported [5] which if maximally decimated has the same computational complexity as the fast transforms and additionally provides a uniform processing load along time.

This approach seems to be advantageous if the end-to-end delay is to be minimized. The applicability of this approach to certain dedicated measurement problems has also been investigated [6].

3. ILLUSTRATIVE EXAMPLES

In the first example a 256-channel DFT is calculated recursively with $N = 64$ for $m = 1, 2, 3, 4$. The input sequence applied was

$$x(n) = \cos\left(\frac{\pi(N + 0.5)n}{2N}\right) . \tag{11}$$

This single sinusoid is just in the middle between two measuring channels. The MATLAB simulations after processing the first, second, etc. blocks are given on Figure 3.

In the second example a 256-channel DFT is calculated recursively with $N = 64$ for $m = 1, 2, 8, 16$. The input sequence was

$$x(n) = \cos\left(\frac{\pi n}{2}\right) + rand - 0.5 \;, \qquad (12)$$

where $rand$ stands for a random number generated by MATLAB between 0 and 1. The sinusoid is located exactly to a DFT channel position. The simulation results for $m = 1, 2, 8$ and 16 are given on Figure 4. The improvement in resolution and noise reduction is remarkable.

In the third example a 256-channel DFT is calculated recursively with $N = 64$ for $m = 1, 2, 4, 8$. The input sequence applied was

$$x(n) = \cos\left(\frac{\pi(N + 0.5)n}{2N}\right) + rand - 0.5 \;, \qquad (13)$$

i.e. the single sinusoid of the first example plus the random sequence. The simulation results for $m = 1, 2, 4, 8$ are given on Figure 5. On the first three diagrams the improvements can be detected. On the last diagram the sinusoid is missing, since for $m = 8$ an integer number of complete periods are considered.

4. CONCLUSIONS

In this paper the concept of block-recursive filters and filter-banks has been introduced. The combination of block-oriented and therefore typically fast algorithms with simple recursive averagers may improve on one hand the accuracy and/or resolution while on the other with the early availability of some rough estimates may reduce the side-effects of the delay caused by the block-oriented approach itself. The reduction of the delay is of real importance in applications where the response time is also specified. The idea of block-recursive filters and filter-banks can be extended toward higher-order averagers and other filtering effects, as well.

REFERENCES

[1] R.E. Crochiere, L.R. Rabiner, *Multirate Digital Signal Processing*, Prentice-Hall, Inc., Englewood Cliffs, N.J, 1983.

[2] J.J. Shynk, "Frequency-Domain and Multirate Adaptive Filtering", *IEEE Signal Processing Magazine*, pp. 15-37, Jan. 1992.

[3] G. Péceli, "A Common Structure for Recursive Discrete Transforms", *IEEE Trans. on Circuits and Systems*, Vol.33, pp. 1035-1036, Oct. 1986.

[4] M. Padmanabhan, K. Martin and G. Péceli, *Feedback-Based Orthogonal Filters*, Kluwer Academic Publishers, Boston/London/Dordrecht, 1996.

[5] A. R. Várkonyi-Kóczy, "A Recursive Fast Fourier Transformation Algorithm", *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing*, Vol. 42, No. 9, pp. 614-616, Sept. 1995.

[6] A. R. Várkonyi-Kóczy, "Multi-Sine Synthesis and Analysis Via Walsh-Hadamard Transformation", *Proc. of the 1996 IEEE International Symposium on Circuits and Systems, ISCAS'96*, May 12-15, Atlanta, Vol. 2, pp. 457-460.

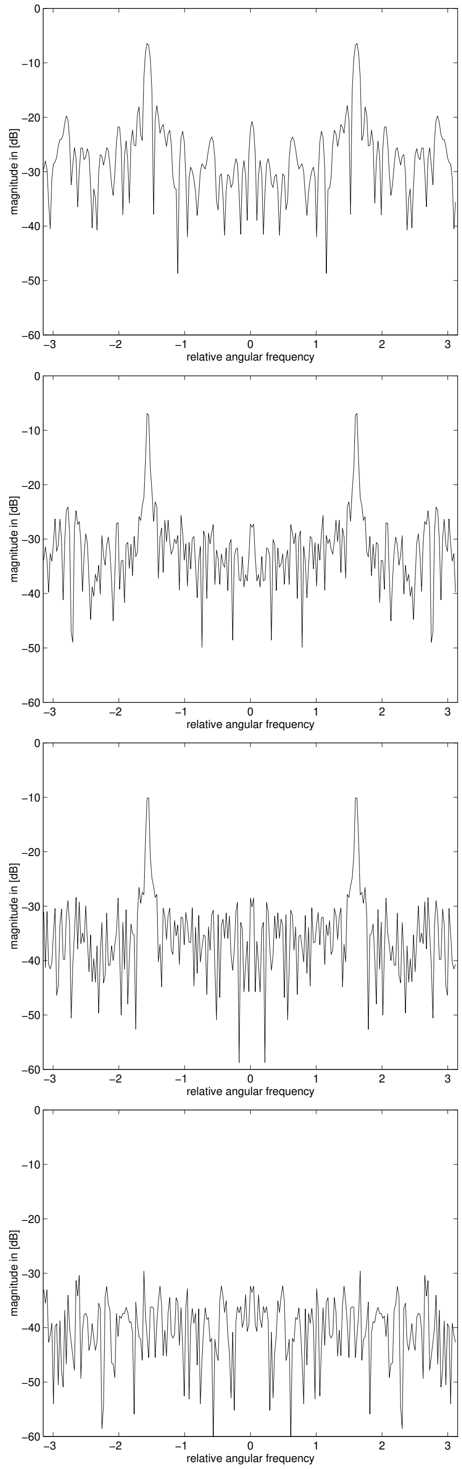

Figure 5. 256-channel DFT of a single sinusoid plus noise. $N = 64$, $m = 1, 2, 4, 8$, **respectively**