EFFICIENT REALIZATION OF THE BLOCK FREQUENCY DOMAIN ADAPTIVE FILTER

Daniël W.E. Schobben, Gerard P.M. Egelmeers, Piet C.W. Sommen

Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
D.W.E.Schobben@ele.tue.nl

ABSTRACT

In many frequency domain adaptive filters Fast Fourier Fourier Transforms (FFTs) are used to transform signals which are augmented with zeros. The overall computational complexity of these adaptive filters is mainly determined by these windowed FFTs, rather than by the filtering operation itself. This contribution presents a new way of calculating these windowed FFTs efficiently. In addition, a method is deduced for implementing non-windowed FFTs of overlapping input data using the previously mentioned efficient windowed FFTs. Also, a method is presented for implementing the windowed FFTs in the update part even more efficient.

1. INTRODUCTION

Large adaptive filters can be implemented with low complexity using Block Frequency Domain approaches. The resulting computational complexity of such a Block Frequency Domain Adaptive Filter (BFDAF) is mostly determined by the five (I)FFTs involved [1, 2, 3]. Of these (I)FFTs, four (I)FFTs are windowed and one is not. Windowed FFTs are transforms of which only a block of Q input samples are nonzero. Windowed IFFTs are inverse transforms of which only a block of Q outputs is needed.

This contribution is concerned with the efficient implementation of these transforms and is outlined as follows. First, an overview is given of the BFDAF. A new method for efficiently calculating windowed (I)FFTs is presented in Section 4. In Section 5 it is shown that an FFT of overlapping data can be calculated using a windowed FFT. An efficient method for implementing the FFT and IFFT in the update algorithm is given in Section 6.

The number of real floating point operations (flops), i.e. the sum of the real multiplications and additions, will be considered as a measure of computational complexity throughout. Also, it is assumed that 4 real multiplications and 2 real additions are used for 1 complex multiplication.

2. NOTATION

Throughout, time and frequency signals will be denoted by lower case and upper case characters respectively. A character which denotes a vector will be underlined. Superscripts denote the vector or matrix dimensions, a matrix with one superscript is square. Also, A^*, A^T and A^{-1} denote complex conjugate, matrix transpose and matrix inverse respectively. Elementwise multiplication is denoted by $\otimes$. The expectation operator will be denoted by $E\{.\}$. The $N \times N$ matrix identity and the $K \times L$ zero matrix will be denoted by I^N and $0^{K,L}$ respectively. In figures, FFTs will be depicted as boxes with a double line at one side. The most recent time sample and the highest frequency bin are at the side of this double line, so that splitting of the input and output busses is depicted in a unique way.

3. BLOCK FREQUENCY DOMAIN ADAPTIVE FILTER

Large adaptive filters can be implemented efficiently in frequency domain using FFTs [1]. The BFDAF is such a frequency domain adaptive filter and is depicted in Figure 1. First the input signal $x[\kappa]$ is segmented into blocks of length B by a serial to parallel converter. Next, the segments are overlapped

$$\underline{x}^M[kB] = (x[kB - M + 1] \dots x[kB])^T . \qquad (1)$$

The overlapping data is transformed to the frequency domain using

$$\underline{X}^M[kB] = \mathcal{F}^M \underline{x}^M[kB] , \qquad (2)$$

with $(\mathcal{F}^M)_{a,b} = \mathrm{e}^{-\mathrm{j}2\pi\frac{ab}{M}}$ for $0 \leq a < M$ and $0 \leq b < M$. The filtering comprises the elementwise multiplication of the transformed input data $\underline{X}^M[kB]$ and

the transformed weights $\underline{W}^M[kB]$. The data is inverse transformed by $(\mathcal{F}^M)^{-1}$ and a part of the result is thrown away (windowed) because of the cyclic nature of the FFTs. This yields the adaptive filter output

$$\underline{\hat{e}}^B[kB] = \left(0^{B,M-B}\mathrm{I}^B\right)(\mathcal{F}^M)^{-1}(\underline{W}^M[kB] \otimes \underline{X}^M[kB]), \tag{3}$$

The residual signal vector $\underline{r}^B[kB]$ is the difference between the adaptive filter output $\underline{\hat{e}}^B[kB]$ and the reference signal $\underline{\tilde{e}}^B[kB]$. This residual signal vector is transformed using a windowed FFT, i.e. only B input samples are non zero which yields

$$\underline{R}^M[kB] = \mathcal{F}^M \left(0^{B,M-B}\mathrm{I}^B\right)^T \underline{r}^B[kB]\,. \tag{4}$$

Next an estimate of the gradient vector $\underline{C}^M[kB]$ is calculated which is needed for the update

$$\underline{C}^M[kB] = 2\alpha(\underline{\hat{P}}_x^M[kB])^{-1} \otimes \underline{R}^M[kB] \otimes \underline{X}^*[kB]\,, \tag{5}$$

where $(\underline{\hat{P}}_x^M[kB])^{-1}$ is the elementwise inverse of the estimate of the signal power which is defined as

$$\underline{P}_x^M[kB] = \tfrac{1}{M} E\{\underline{X}^M[kB] \otimes (\underline{X}^M[kB])^*\}\,. \tag{6}$$

The update part is depicted in Figure 2 and the update equation is discussed in Section 6. From the above it follows that all Fourier transforms are windowed, either at the input or at the output, except for the FFT in (2) that transforms the overlapping input data. First a new method is presented in the next section which efficiently performs the windowed transforms.

4. EFFICIENT COMPUTATION OF WINDOWED FFTS

A new efficient method for calculating windowed FFTs of real data and windowed IFFTs that yield real data is presented. Both the FFT length M and the window length Q are assumed to be powers of 2. The number of flops needed to calculate such an FFT and IFFT is denoted as $\Psi_{\mathrm{FFT}}\{M,Q\}$ and $\Psi_{\mathrm{IFFT}}\{M,Q\}$. A windowed length M FFT of real input data is defined as

$$\underline{X}^M = \mathcal{F}^M \begin{pmatrix} \underline{x}^Q \\ \underline{0}^{M-Q} \end{pmatrix}, \tag{7}$$

with $\underline{x}^Q$ defined by (1) with $B = Q$. Time indices are suppressed to simplify notation. A radix-2 length M Decimation In Time (DIT) FFT decomposition is defined for $0 \le l < M$ as [4]

$$(\underline{X}^M)_l = \sum_{k=0}^{M} \begin{pmatrix} \underline{x}^Q \\ \underline{0}^{M-Q} \end{pmatrix}_k \mathrm{e}^{-\mathrm{j}2\pi\frac{kl}{M}},$$

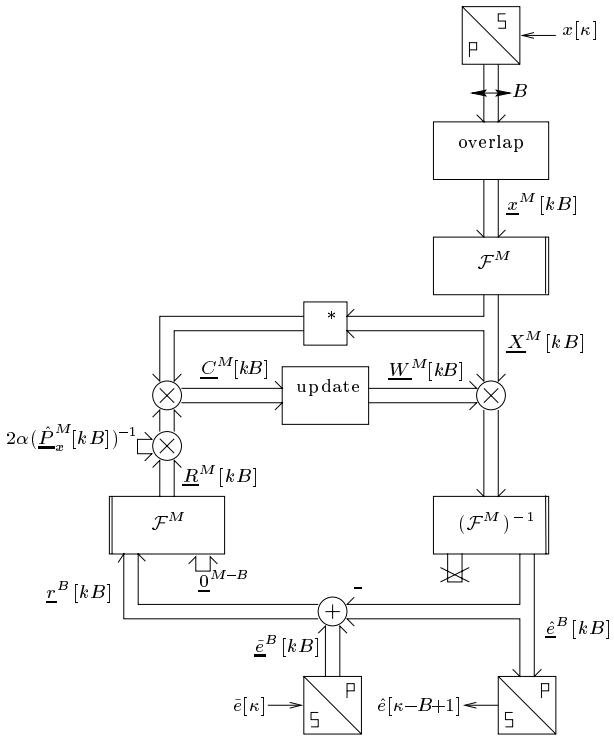

Figure 1: Block Frequency Domain Adaptive Filter

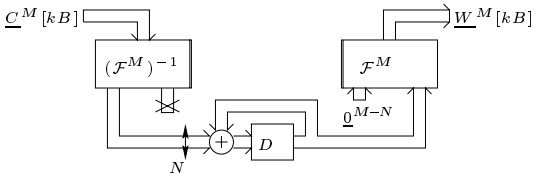

Figure 2: BFDAF update part

$$= \sum_{k=0}^{\frac{M}{2}-1} \begin{pmatrix} \underline{x}^Q \\ \underline{0}^{M-Q} \end{pmatrix}_{2k} \mathrm{e}^{-\mathrm{j}2\pi\frac{kl}{M/2}} + \mathrm{e}^{-\mathrm{j}2\pi\frac{l}{M}} \sum_{k=0}^{\frac{M}{2}-1} \begin{pmatrix} \underline{x}^Q \\ \underline{0}^{M-Q} \end{pmatrix}_{2k+1} \mathrm{e}^{-\mathrm{j}2\pi\frac{kl}{M/2}}, \tag{8}$$

where the subscript denotes the element that is taken from the corresponding vector. The number of flops needed to combine these two FFTs is denoted as $\Psi_C\{M\}$. The total number of flops equals for $M > 4$ and $Q > 2$

$$\Psi_{\mathrm{FFT}}\{M,Q\} = 2\,\Psi_{\mathrm{FFT}}\{\tfrac{M}{2},\tfrac{Q}{2}\} + \Psi_C\{M\}\,. \tag{9}$$

Stated in words this means that a windowed FFT is split in two length $\frac{M}{2}$ FFTs with window length $\frac{Q}{2}$.

The combining is achieved by multiplying the result of one FFT elementwise with elements of the Fourier matrix $\mathcal{F}^M$ and adding this result to the result of the other FFT. This takes

$$\Psi_C\{M\} = \tfrac{5}{2}M - 8\,, \tag{10}$$

flops. The FFTs are split until FFTs of length $\frac{2M}{Q}$ remain with windows of length 2. The number of flops needed to calculate a length $\frac{2M}{Q}$ FFT is

$$\Psi_{\mathrm{FFT}}\{\tfrac{2M}{Q}, 2\} = \tfrac{3M}{2Q} - 1\,, \tag{11}$$

for $Q < M$. The overall complexity can now be calculated by inserting (11) and (10) in (9) which yields for $Q < M$

$$\begin{aligned}\Psi_{\mathrm{FFT}}\{M, Q\} &= \tfrac{Q}{2}\Psi_{\mathrm{FFT}}\{\tfrac{2M}{Q}, 2\} + \sum_{b=1}^{\log_2(\frac{Q}{2})} 2^{b-1}\Psi_{\mathrm{C}}\{\tfrac{M}{2^{b-1}}\} \\ &= \tfrac{5}{2}M\log_2(Q) - \tfrac{7}{4}M - \tfrac{9}{2}Q + 8\,. \end{aligned} \tag{12}$$

When calculating IFFTs, a similar procedure can be followed which yields

$$\Psi_{\mathrm{IFFT}}\{M, Q\} = \tfrac{9}{4}M\log_2(Q) - \tfrac{3}{4}M - \tfrac{3}{2}Q + 6\,, \tag{13}$$

for $Q < \frac{M}{2}$. Note that the FFT can be implemented with even lower complexity for $Q = M$ than that indicated by (12) by starting with lengths 4 FFT which requires $\Psi_{\mathrm{FFT}}\{4, 4\} = 6$ flops. Also, a more efficient implementation of the IFFT is achieved for $Q = M$ and $Q = \frac{M}{2}$ by starting with length 4 IFFTs which requires $\Psi_{\mathrm{IFFT}}\{4, 4\} = 8$ and $\Psi_{\mathrm{IFFT}}\{4, 2\} = 6$ flops respectively.

Four (I)FFTs in the BFDAF can be calculated using the windowed procedure described above. The FFT which transforms the overlapping input data does not contain a window and will therefore be the largest computational burden. An efficient method for implementing this FFT is presented in the next section.

5. EFFICIENT RECURSIVE COMPUTATION OF FFTS OF OVERLAPPING DATA

In this section a method is presented which performs the FFT of the overlapping input data using a windowed FFT with window length B. The overlapping input data at time $(k+1)B$ can be written as

$$\underline{x}^M[(k+1)B] = D_B^M\,(\underline{x}^M[kB] + \underline{y}_B^M[kB])\,, \tag{14}$$

with the windowed data vector

$$\underline{y}_B^M[kB] = \begin{pmatrix} \underline{x}^B[(k+1)B] - \underline{x}^B[kB - M + B] \\ \underline{0}^{M-B} \end{pmatrix}\,, \tag{15}$$

and D_B^M a time domain rotation defined as

$$\mathrm{D}_B^M = \begin{pmatrix} 0^{M-B,B} & \mathrm{I}^{M-B} \\ \mathrm{I}^B & 0^{B,M-B} \end{pmatrix}\,. \tag{16}$$

Now the transform of the overlapping input data at time $(k+1)B$ can be written as

$$\begin{aligned}\underline{X}^M[(k+1)B] &= \mathcal{F}^M D_B^M\,(\underline{x}^M[kB] + \underline{y}_B^M[kB]) \\ &= \mathcal{F}^M D_B^M\,(\mathcal{F}^M)^{-1}(\underline{X}^M[kB] + \mathcal{F}^M \underline{y}_B^M[kB])\,. \end{aligned} \tag{17}$$

Thus, the transform of the overlapping input data at time $(k+1)B$ can be calculated from the sum of previously transformed data and the result of a windowed FFT with window length B. Also, a time domain rotation $\mathcal{F}^M D_B^M (\mathcal{F}^M)^{-1}$ must be performed. So, the resulting computational complexity will equal $\Psi_{\mathrm{FFT}}\{M, B\} + \Psi_{\mathrm{ROT}}\{M, B\}$, with $\Psi_{\mathrm{ROT}}\{M, B\}$ the number of flops needed for the time domain rotation. Using the circular structure of D_B^M yields $\mathrm{D}_B^M = (\mathrm{D}_1^M)^B$ [5]. This time domain rotation can be computed efficiently in the frequency domain as

$$\mathcal{F}^M \mathrm{D}_1^M (\mathcal{F}^M)^{-1} = \mathrm{diag}\{\underline{v}^M\}\,, \tag{18}$$

with

$$\underline{v}^M = \mathcal{F}^M \begin{pmatrix} \underline{0}^{M-1} \\ 1 \end{pmatrix}\,. \tag{19}$$

As $\mathcal{F}^M \mathrm{D}_B^M (\mathcal{F}^M)^{-1} = (\mathcal{F}^M \mathrm{D}_1^M (\mathcal{F}^M)^{-1})^B$, the k'th element of the main diagonal of $\mathcal{F}^M \mathrm{D}_B^M (\mathcal{F}^M)^{-1}$ equals $((\underline{v}^M)_k)^B$. For the diagonal matrix $\mathcal{F}^M \mathrm{D}_B^M (\mathcal{F}^M)^{-1}$ then the next theorem can be deduced

$$\mathcal{F}^M \mathrm{D}_B^M (\mathcal{F}^M)^{-1} = \mathrm{diag}\{\begin{pmatrix} \underline{v}_{\frac{B}{B}}^{\frac{M}{B}} \\ \vdots \\ \underline{v}_{\frac{M}{B}}^{\frac{M}{B}} \end{pmatrix}\}. \tag{20}$$

Thus, the rotation reduces to an elementwise multiplication in (17). Using that the rotation contains B length M/B Fourier vectors and that both the left and right operand of the elementwise multiplication are Fourier transforms of real valued vectors the total rotation requires $\Psi_{\mathrm{ROT}}\{M, B\}$ real operations

$$\Psi_{\mathrm{ROT}}\{M, B\} = \begin{cases} 0 & \text{for } \frac{M}{B} < 8 \\ 3M - 16B & \text{for } \frac{M}{B} \geq 8 \end{cases}\,. \tag{21}$$

So, all five FFTs can be implemented with a computational complexity which is in the order of $M\log_2(Q)$, with Q the window length of the transform.

The size of the Fourier transform M is equal to or larger than $N + B - 1$, with N the number of adaptive weights. The algorithm delay of the adaptive filter

equals $B-1$. For large adaptive filters in which only a small delay can be tolerated, such as the adaptive echo canceler, B will be small compared to the transform size M. In this case, the filter length N will approximately equal the transform size M. Therefore, the FFT and IFFT in the update part take a dominant amount of flops because the window almost equals the transform size. Furthermore, the efficient implementation of windowed FFTs in Section 4 requires that both the transform size and the window size are powers of 2 which is not the case. Therefore, 2 regular FFTs are required for the update part. The next section presents a method to reduce the computational complexity of the (I)FFT in the update part.

6. EFFICIENT COMPUTATION OF THE UPDATE PART

The BFDAF update part is shown in Figure 2, with D a delay of one block of data. Next, it is shown that the computational complexity of the two (I)FFTs of the update part can be reduced even further by using complementary windowed (I)FFTs. The update equation is

$$\underline{W}^M[(k+1)B] = \mathcal{F}^M \begin{pmatrix} \mathrm{I}^N \\ 0^{N,M-N} \end{pmatrix} \cdot \left[\underline{w}^N[kB] + \left(\mathrm{I}^N 0^{M-N,N}\right) (\mathcal{F}^M)^{-1} \underline{C}^M[kB]\right] . \quad (22)$$

Next, (22) is written as

$$\underline{W}^M[(k+1)B] = \underline{W}^M[kB] + \underline{W}_\Delta^M[kB] , \quad (23)$$

where $\underline{W}_\Delta^M[kB]$ is equal to

$$\mathcal{F}^M \begin{pmatrix} \mathrm{I}^N & 0^{M-N,N} \\ 0^{M-N,N} & 0^{M-N} \end{pmatrix} (\mathcal{F}^M)^{-1} \underline{C}^M[kB] . \quad (24)$$

This equation incorporates a time domain window of length N. A more efficient method would be implementing the complementary time domain window. This is achieved by rewriting (24)

$$\underline{W}_\Delta^M[kB] = \underline{C}^M[kB] - \mathcal{F}^M \begin{pmatrix} 0^N & 0^{N,M-N} \\ 0^{M-N,N} & \mathrm{I}^{M-N} \end{pmatrix} (\mathcal{F}^M)^{-1} \underline{C}^M[kB] . \quad (25)$$

The transform size M will now be chosen to be equal to $N+B$, so that the time domain window in (25) is of length B and the (I)FFT can be implemented using the method described in Section 4. Consequently M complex additions are required in stead of N real additions. In comparison to (24) a reduction is achieved of $\Psi_{\mathrm{FFT}}\{M,M\} - \Psi_{\mathrm{FFT}}\{M,B\} + N - 2M$ flops. The corresponding modified update is depicted in Figure 3.

When using the complementary window and choosing $M = N+B$ it is possible to shift all windows so that they correspond with (7) without affecting the scheme. Defining a windowed (I)FFT with the window at the other side would be a less efficient solution.

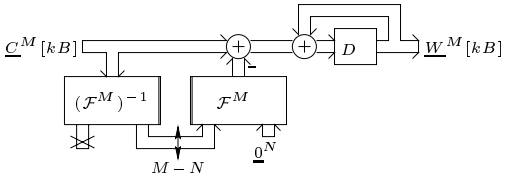

Figure 3: Equivalent of BFDAF update part

7. CONCLUSIONS

A frequency domain adaptive filter contains five (I)FFTs of which four are windowed. A new method is presented to implement the windowed (I)FFTs efficiently. Also, it is shown that the non-windowed FFT of overlapping input data can be implemented as a windowed FFT and a time domain rotation. In addition, the update part is implemented using a complementary time domain window. The overall complexity of the BFDAF is now determined by the five transforms which all have windows of length B. The overall complexity of the proposed implementation of the BFDAF is therefore in the order of $M \log_2(B)$, with M the Fourier transform size. When non-windowed FFTs where used, this complexity would be in the order of $M \log_2(M)$ flops.

8. REFERENCES

[1] P.C.W. Sommen. *Adaptive Filtering Methods.* Eindhoven (Netherlands): Eindhoven Technical University of Technology, June 1992. Ph. D. Thesis.

[2] D. Mansour and A.H. Gray jr. Unconstrained frequency domain adaptive filter. *IEEE Trans. Ac. Speech and Signal Proc.*, 30(5):726–734, Oct. 1982.

[3] G.P.M. Egelmeers. *Real Time Realization of Large Adaptive Filters.* Eindhoven (Netherlands): Eindhoven Technical University of Technology, Nov. 1995. Ph. D. Thesis.

[4] A.V. Oppenheim and R.W. Shafer. *Digital Signal Processing.* Englewood Cliffs, New York (USA): Prentice-Hall, 1975.

[5] P.J. Davis. *Circulant Matrices.* New York (USA): Wiley, 1979.