TA-2-2

Block Fast Projection Algorithm with Independent Block Sizes

Tanaka Masashi, Shoji Makino, Junji Kojima
NTT Human Interface Laboratories

1. Introduction

Block processing is an effective approach for reducing the computational complexity of adaptive filtering algorithms although it delays the adaptive filter output and degrades the convergence rate in some implementations.

Recently, Benesty[1] proposed a solution to the problems. He introduced the idea of 'exact' block processing which produces the filter output exactly the same as that of the corresponding sample-by-sample algorithm and has short delay by facilitating the fast FIR filtering method.

Block processing can be applied to two parts of the adaptive filtering algorithms, i.e. computing the filter output and updating the filter. Conventional 'exact' block algorithms have been using the identical block size for the two parts.

This short paper presents the 'exact' block projection algorithm [2] having two independent block sizes, which is listed in List 1. We see, by showing the relation between the filter length and the output delay for a given computation power, that the independent block sizes extend the availability of the 'exact' block fast projection algorithm toward use with longer delay.

2. Filter length and delay

In practice, we are interested in the filter lengths and the output delays under a given computation power. To represent the filter length and the delay in terms of computational complexity, which we define by the number of additions and multiply-and-add operations, we estimate the computational complexity of the 'exact' block fast projection algorithm.

• computational complexity

The 'exact' block algorithm consists of three phases, i.e. computing filter output, correcting the filter output, and updating the adaptive filter, whose computational complexity are estimated as follows.

For output: (proc. in List 1)

$$T_o = \mathrm{MVP}\,(N_1)\,(L/N_1) + N_1 \log_2 N_1 \text{ per } N_1 \text{ samples} \quad (1)$$

For correction: (procs. 2 to 7 in List 1)

$$T_c = N_1\,(2.5N_2 + 20p) \text{ per } N_1 \text{ samples} \quad (2)$$

For updating: (proc. 8 in List 1)

$$T_u = \mathrm{MVP}\,(N_2)\,(L/N_2) + N_2 \log_2 N_2 \text{ per } N_2 \text{ samples} \quad (3)$$

Here, N_1 and N_2 denote the block sizes for computing the filter output and updating the filter, respectively and are assumed to be powers of two here. p denotes the projection order. MVP(N) means the computational complexity necessary for a matrix and vector product (MVP) of dimension N. MVPs are done using either the fast FIR filtering (FFF) method as done in [1] or the fast Fourier transform (FFT) and estimated as

$$\mathrm{MVP}\,(N) = \begin{cases} N\,(3\,(3/2)^{\log_2 N} - 1) & \text{for FFF} \\ 8N\log_2 N & \text{for FFT} \end{cases} . \quad (4)$$

FFT is more efficient with the block size $N \geq 512$.

• filter length

Computational complexity per sample instance is given by summing up T_o, T_c, and T_u as

$$\begin{aligned} T &= T_o/N_1 + T_c/N_2 + T_u/N_2 \\ &\approx \mathrm{MVP}\,(N_1)\,(L/N_1^2) + (2.5N_2 + 20p) \\ &\quad + \mathrm{MVP}\,(N_2)\,(L/N_2^2) \end{aligned} . \quad (5)$$

Solving this equation with respect to the filter length L, we get

$$L = \frac{T - 2.5N_2 - 20p}{\mathrm{MVP}\,(N_1)/N_1^2 + \mathrm{MVP}\,(N_2)/N_2^2} . \quad (6)$$

As noted in [1], for $N_1 = N_2$, there exists a block size N_θ that gives the longest filter length. By differentiating $\ln L$ with respect to N_1 and N_2, and then setting it to be zero, we obtain the block size N_θ as

$$N_\theta = 0.12\,(T - 20p) . \quad (7)$$

• delay

Fig.1 illustrates the maximum filter output delay for two cases, i.e. $N_1 > N_2$ and $N_1 \leq N_2$. We see that the filter output is delayed by

$$D = \begin{cases} N_1 + T_o/T & , N_1 > N_2 \\ (2T_o + T_c + T_u)/T & , N_1 \leq N_2 \end{cases} . \quad (8)$$

The delay corresponding to N_θ is given approximately from (1)-(3), (6) and (7) as

$$D_\theta = 0.12\,(T - 20p) , \quad (9)$$

which is coincidentally the same value as N_θ in (7).

3. Numerical example

Substituting various pairs of the block sizes N_1 and N_2 into (6) and (8), we can draw the relationship between the delay and the filter length under a fixed computational complexity T. Fig. 2 shows an example of the relationship. The conditions are: the computational complexity per one sample instance $T = 2000$ and the projection order $p = 2$. The solid line connects the equal-block-size conditions. According to (9), we have $D_\theta = 235$, which is indicated by the vertical dotted line.

For delay longer than D_θ, pairs of $N_1 > N_2$ can give longer filter length than $N_1 = N_2$. For example, for the delay near 512 samples, the (N_1, N_2) =(512, 512) has the filter length of about 1500, while the (N_1, N_2) =(256, 128) achieves the filter length about 2900, which is almost twice as longer than the equal-block-size case.

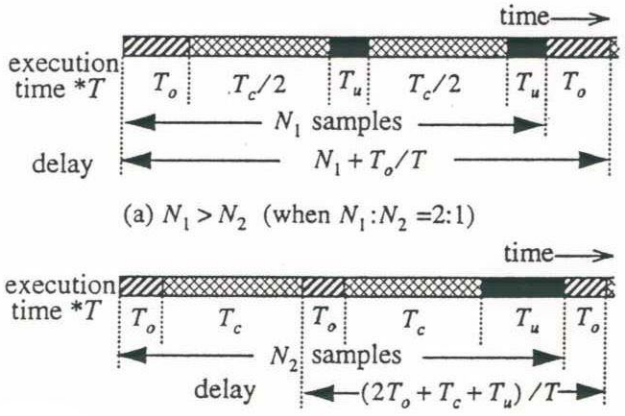
4. Conclusions

Employing independent block sizes in the 'exact' block fast projection algorithm allows options of longer delay but more reduction in computational complexity than the equal-block-size version. Long delay in adaptive filter may be permissible in applications where delays due to factors such as (de)coding and transmitting are unavoidable and dominant.

References

[1] J. Benesty and P. Duhamel: "A fast exact least mean square adaptive algorithm," IEEE Trans. on Signal Processing, vol. 40, no.12, pp. 2904-2920, (1992-12)

[2] M. Tanaka, et al.: "Projection algorithm using fast FIR filtering techniques," IEICE fall conf., p. A-79, (1995-9)



(a) $N_1 > N_2$ (when $N_1 : N_2$ =2:1)

(b) $N_1 \le N_2$ (when $N_1 : N_2$ =1:2)

Fig. 1 Filter output delay

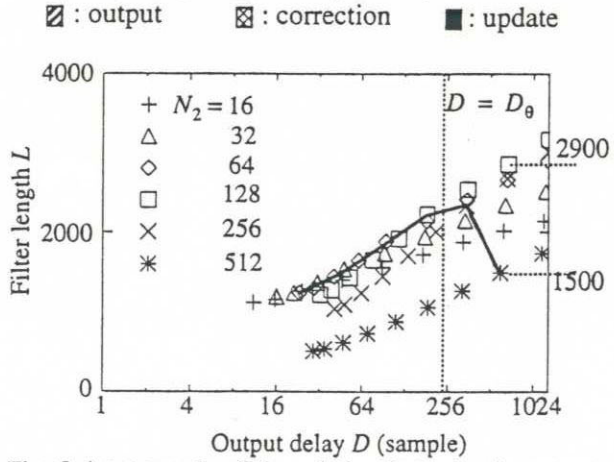


Fig. 2 An example of the relation between the output delay and the filter length.

T=2000, p=2, N_1=4, 8, 16, 32, 64, 128, 256, 512,1024 from the left for each N_2.

List 1: projection algorithm with exact block processing

Notations:

k : time index, $x(k)$: input signal, $y(k)$: desired signal

$e(k)$: error signal, L : filter length, p : projection order

N_1, N_2 : block sizes

$\hat{\mathbf{h}}_L(k)$: adaptive filter coefficient vector

(Subscripts denote the dimension of vectors)

$\mathbf{x}_L(k)$: $= [x(k), \ldots, x(k-L+1)]^T$

$\mathbf{z}_L(k)$: intermediate vector updated in stead of $\hat{\mathbf{h}}_L(k)$ and defined as

$$\mathbf{z}_L(k) = \hat{\mathbf{h}}_L(k) - \sum_{i=1}^{p-1} \mathbf{s}_p(k-1)_i \mathbf{x}_L(k-i)$$

$\mathbf{s}_p$:See proc. 6. $\mathbf{s}_p(k-1)_i$ is the i-th element of $\mathbf{s}_p(k-1)$

$\mathbf{e}_p(k)$: $= [e(k), \ldots, (1-\mu)^{p-1} e(k-p+1)]^T$

μ : step size $(0 \le \mu \le 2)$

$\mathbf{g}_p(k)$: the solution of $\mathbf{R}_p(k)\,\mathbf{g}_p(k) = \mathbf{e}_p(k)$

$\mathbf{R}_p(k)$: covariance matrix of dimension p defined as

$$\mathbf{R}_p(k) = \sum_{i=0}^{L-1} \mathbf{x}_p(k-i)\,\mathbf{x}_p(k-i)^T$$

$\mathbf{r}_{p+N_2-1}(k)$: autocorrelation of $x(k)$, from 1-st to $p+N_2-1$-th order.

0. initialization

$k_1 = k_2 = \max(N_1, N_2)$, $n_2 = 0$

$\mathbf{r}_{p+N_2-1}(k_1-1)_i = \mathbf{x}_L^H(k_1-1-i)\,\mathbf{x}_L(k_1-1)$

$(i = 1, \ldots, p+N_2-2)$ H : conjugate transposition

$\mathbf{s}_p(k_1-1) = \mathbf{e}_p(k_1-1) = \mathbf{g}_p(k_1-1) = 0$

Begin at k_1 and repeat 1 to 8.

1. If $k_1 \bmod N_1 = 0$, then compute

$y'(k_1+i) = \mathbf{z}_L^H(k_2)\,\mathbf{x}_L(k_1)$ $(i = 0, \ldots, N_1-1)$.

2. $\mathbf{r}_{p+N_2-2}(k_1) = \mathbf{r}_{p+N_2-2}(k_1-1)$
$+ x(k_1)\,\bar{\mathbf{x}}_{p+N_2-2}(k_1-1) - x(k_1-L)\,\bar{\mathbf{x}}_{p+N_2-2}(k_1-L-1)$
$-$: complex conjugate

3. $e(k_1) = y(k_1) - \hat{y}'(k_1) - \mathbf{s}_{p+n_2-1}^H(k_1-1)\,\mathbf{r}_{p+n_2-1}(k_1)$

4. $\begin{bmatrix} \mathbf{e}_p(k_1) \\ * \end{bmatrix} = \begin{bmatrix} e(k_1) \\ (1-\mu)\,\mathbf{e}_p(k_1-1) \end{bmatrix}$ *: don't care

5. Solve $\mathbf{R}_p(k_1)\,\mathbf{e}_p(k_1) = \mathbf{g}(k_1)$.

6. $\mathbf{s}_{p+n_2}(k_1) = \begin{bmatrix} 0 \\ \mathbf{s}_{p+n_2-1}(k_1-1) \end{bmatrix} + \begin{bmatrix} \mu \mathbf{g}_p(k_1) \\ \mathbf{0}_{n_2} \end{bmatrix}$

7. Increase indexes $k_1 = k_1 + 1$, $n_2 = n_2 + 1$.

8. If $k_1 \bmod N_2 = 0$ then update $\mathbf{z}$.

$\mathbf{z}_L(k_2+N_2) = \mathbf{z}_L(k_2) + [\mathbf{x}_L(k_2+N_2-p), \ldots, \mathbf{x}_L(k_2-p+1)]$
$\mathbf{s}_{N_2+p-1}(k_2+N_2-1)_{p-N_2+p-1}$
$(p$-th to N_2+p-1-th elements)

$k_2 = k_2 + N_2$, $n_2 = 0$