ELSEVIER

Signal Processing 79 (1999) 251–259

SIGNAL PROCESSING

www.elsevier.nl/locate/sigpro

Real-time implementation of the moving FFT algorithm

Pei-Chen Lo*, Yu-Yun Lee

Department of Electrical and Control Engineering, National Chiao Tung University, 1001 Ta-Hsueh Road, Hsinchu, Taiwan, ROC

Received 11 February 1999; received in revised form 29 June 1999

Abstract

An integrated algorithm for analyzing the real-time, running Fourier spectra is presented. When applying to the real-time analysis, the computational efficiency and synchronization ability are highly improved by integrating the recursive procedure with the real-time implementation strategy. Given the FFT (frame) size $N = 2^r$ and moving size $M = 2^p$, the algorithm implements the recursive procedure for updating the succeeding decomposed (into size N/M) DFTs at the $(r - p)$th stage, and thereafter constructs the sr-FFT (split-radix fast-Fourier-transform) butterfly modules in a real-time way. The recursive procedure highly reduces the number of complex arithmetic operations when the moving size M is small. The real-time implementation scheme enables the running spectral analysis to better synchronize with the data acquisition process. The computational complexity of the integrated algorithm is analyzed in detail, which shows the dependence of the number of complex arithmetic operations on the ratio N/M. The experimental result tested on Pentium 66 agrees with the analysis. © 1999 Elsevier Science B.V. All rights reserved.

Zusammenfassung

Es wird ein integrierter Algorithmus zur Echtzeit-Analyse eines laufenden Fourier-Spektrums präsentiert. Bei der Anwendung auf die Echtzeit-Analyse werden die rechnerische Effizienz und die Synchronisationsfähigkeit durch Integration der rekursiven Prozedur mit der Echtzeit-Implementierungsstrategie stark verbessert. Für eine gegebene FFT- (Rahmen-) Länge $N = 2^r$ und Schrittweite $M = 2^p$ implementiert der Algorithmus die rekursive Prozedur zur Aktualisierung der nachfolgenden zerlegten DFTs der Länge N/M in der $(r - p)$-ten Stufe und konstruiert danach die sr-FFt (split-radix fast Fourier transform) Butterfly-Module in echtzeitkompatibler Weise. Bei kleiner Schrittweite M bewirkt die rekursive Prozedur eine starke Reduktion der Anzahl komplexer arithmetischer Operationen. Das Echtzeit-Implementierungsverfahren erlaubt eine bessere Synchronisation der laufenden Spektralanalyse mit dem Datenerfassungsprozeß. Eine detaillierte Analyse der rechnerischen Komplexität des integrierten Algorithmus zeigt, daß die Anzahl der komplexen arithmetischen Operationen vom Verhältnis N/M abhängt. Das mit einem Pentium 66 erhaltene experimentielle Ergebnis bestätigt die Analyse. © 1999 Elsevier Science B.V. All rights reserved.

Résumé

Nous présentons un algorithme intégré pour l'analyse en temps réel de spectres de Fourier glissants. Lors de l'application de l'analyse en temps réel, l'efficacité du calcul et la capacité de synchronisation sont grandement améliorées en intégrant la procédure récursive à la stratégie d'implantation en temps réel. Pour une taille $N = 2^r$ de FFT (trame)

*Corresponding author. Tel.: 886-3-573-1667; fax: 886-3-571-5998.
E-mail address: pclo@cc.nctu.edu.tw (P.-C. Lo)

et un glissement $M = 2^p$, l'algorithme implante la procédure récursive de mise à jour des DFT suivantes décomposées (de taille N/M) à la $(r-p)$ième étape, et construit ensuite les modules tampons des papillons de la sr-FFT (transformation de Fourier rapide de type split-radix) en temps réel. Cette procédure récursive réduit grandement le nombre d'opérations arithmétiques complexes lorsque la taille M du glissement est petite. Cette approche d'implantation en temps réel permet que l'analyse spectrale glissante soit mieux synchronisée avec le processus d'acquisition des données. La complexité en calcul de l'algorithme intégré est analysée en détail, et montre la dépendance du nombre d'opérations arithmétiques complexes vis-à-vis du rapport N/M. Les résultats expérimentaux obtenus sur un Pentium 66 sont en accord avec cette analyse. © 1999 Elsevier Science B.V. All rights reserved.

Keywords: Running spectral analysis; Recursive procedure for updating DFT; Decimation-in-time split-radix fast-Fourier-transform (DIT sr-FFT); Real-time implementation of sr-FFT; Local butterfly modules

1. Introduction

Running spectral analysis based on short-time Fourier transform (STFT) is frequently used in the long-term monitoring of the multi-channel EEG (electroencephalograph) signals. The procedure involves the evaluation of the DFTs (discrete Fourier transforms) on successive overlapping segments. Consider a 1-hour EEG recording with a sampling rate of 200 Hz. To analyze the time-varying Fourier spectra with a DFT (frame) size of 512 samples and a moving size of 64 samples, it requires the evaluation of a total of 11,243 512-point DFTs! In fact, the circularly periodic property of the DFT enables 'inheritance' of the results evaluated on the preceding frame.

Not long after the key contribution to digital spectral analysis, the development of the FFT (fast Fourier transform) by Cooley and Tukey [5], Halberstein [9] proposed the idea of evaluating the successive DFTs using the recursive computation. Bongiovanni et al. presented the formulated procedures which allowed a moving size M to be any integer power of 2; nonetheless, M is no greater than the frame size [3]. Several algorithms for efficient computation of the moving-frame DFTs have been developed thereafter [1,2,4,8,15,17]. Most of them discussed the methods for updating the successive DFTs at a rate of one sample point each time, that is, the moving size $M = 1$. Many practical applications do not require such a small temporal resolution when computing the time-varying Fourier spectra. Furthermore, these FFT algorithms were implemented in the manner that the algorithms started building the butterfly structure for the FFT computation only after all the framed data points had been collected. The authors previously presented a new algorithm (called the 'real-time FFT algorithm') which simultaneously constructed the local butterfly structure while the data acquisition proceeded [10]. The feasibility of the real-time FFT algorithm is endorsed by the modular property of the FFT butterfly structure. As the development of the recursive moving FFT algorithm aims at on-line real-time evaluation of the running Fourier spectra, the authors propose a new algorithm of which the recursive procedure is implemented in the real-time manner.

2. Recursive procedure for updating the successive DFT

2.1. Recursive procedure for M = 1

Let $X_i[k]$ denote the ith N-point DFT of non-stationary long-term sequence $x[n]$ where $N = 2^r$. For generalization, $x[n]$ may be complex. Since the running Fourier spectrum is updated at a rate of one sample point each time, the $X_i[k]$ is given by

$$X_i[k] = \sum_{n=0}^{N-1} x[i+n]W_N^{nk}, \quad 0 \leqslant k \leqslant N-1, \tag{1}$$

where $W_N = \mathrm{e}^{-\mathrm{j}2\pi/N}$ and $i \geqslant 0$. The succeeding DFT becomes

$$X_{i+1}[k] = \sum_{n=0}^{N-1} x[i+1+n]W_N^{nk}$$
$$= \sum_{n=1}^{N} x[i+n]W_N^{(n-1)k}$$

$$= W_N^{-k} \sum_{n=1}^{N} x[i+n] W_N^{nk}$$

$$= W_N^{-k}\{X_i[k] - x[i] + x[i+N]\}. \quad (2)$$

The above equation relates the DFT of each current frame to that of the preceding frame via a simple recursive formula. Note that the $(-x[i] + x[i+N])$ only needs to be evaluated once for computing the $X_{i+1}[k], 0 \leqslant k \leqslant N-1$. Hence, the number of arithmetic operations is reduced to $N+1$ complex additions and N complex multiplications for updating the DFT. The computational complexity will be analyzed in detail in Section 5 and in Appendix A.

2.2. *Recursive procedure for $M = 2^p, 0 < p \leqslant r$*

Consider that the frame moves M sample points each time where $1 < M \leqslant N$ and $M = 2^p$, an integer power of 2. Alternatively speaking, when computing the FFT of the succeeding frame, the first M samples of the current frame leave the FFT butterfly structure, and M new samples enter the butterfly structure. To illustrate the 'moving' effect on the butterfly structure, Fig. 1 plots the arrangement of the input data sequence given $N = 16$ and $M = 4$. The solid (dashed) frame encloses those sample points taking part in evaluating $X_i[k]$ ($X_{i+1}[k]$). At the first stage, each butterfly substructure (module) performs a 2-point DFT computation. The data points in the solid and dashed frames are all different. When proceeding to the second stage, the current and succeeding DFT computations share three data points. Apparently, in each of the four blocks performing the 4-point DFT computation, the updating mechanism is the same as the case of $M = 1$ where the frame moves one sample each time. The above concept is formulated in the following derivation.

Considering a moving size of $M = 2^p$ $(0 < p \leqslant r)$, the ith N-point DFT $(i \geqslant 0)$ is

$$X_i[k] = \sum_{n=0}^{N-1} x[iM+n] W_N^{nk}, \quad 0 \leqslant k \leqslant N-1. \quad (3)$$

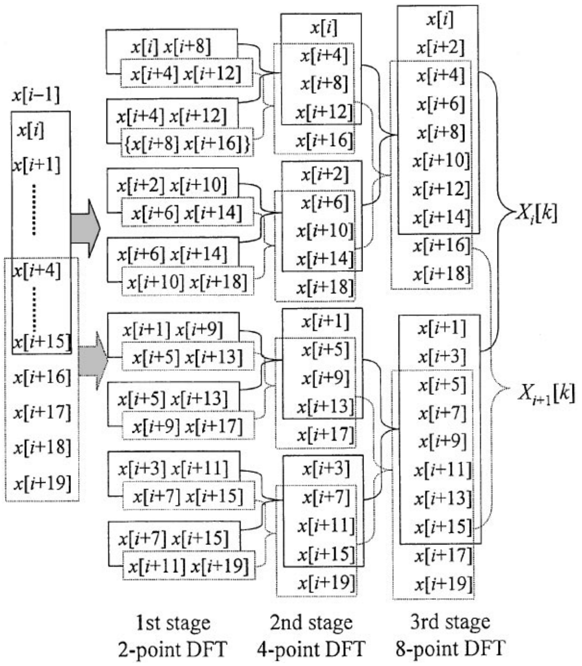

Fig. 1. Effect of moving frame on the arrangement of the input data sequence in the FFT structure ($N = 16, M = 4$).

Let n equal $mM + 1$, Eq. (3) becomes

$$X_i[k] = \sum_{l=0}^{M-1} \sum_{m=0}^{N/M-1} x[iM + mM + l] W_N^{(mM+l)k}$$

$$= \sum_{l=0}^{M-1} W_N^{lk} \left[\sum_{m=0}^{N/M-1} x[M(i+m)+l] W_{N/M}^{mk} \right]$$

$$= \sum_{l=0}^{M-1} W_N^{lk} G_{i,l}[k], \quad (4)$$

where the $G_{i,l}[k]$ is an (N/M)-point DFT. Note that the (N/M)-point DFT is performed at the $(r-p)$th stage of the butterfly structure where M blocks are arranged according to the index l, from top downwards, in the bit-reversed order. Observing the example in Fig. 1, the four blocks at the second stage are $G_{i,0}[k]$, $G_{i,2}[k]$, $G_{i,l}[k]$ and $G_{i,3}[k]$ from the top. Similarly, the succeeding DFT can be expressed as

$$X_{i+1}[k] = \sum_{n=0}^{N-1} x[iM + M + n] W_N^{nk}$$

$$
\begin{aligned}
&= \sum_{l=0}^{M-1} \sum_{m=0}^{N/M-1} x[iM + M + mM + l\,]W_N^{(mM+l)k} \\
&= \sum_{l=0}^{M-1} W_N^{lk}\left[\sum_{m=0}^{N/M-1} x[M(i+1+m)+l]W_{N/M}^{mk}\right] \\
&= \sum_{l=0}^{M-1} W_N^{lk} G_{i+1,l}[k]. \qquad (5)
\end{aligned}
$$

In comparison with $G_{i,l}[k]$ in Eq. (4), the $G_{i+1,l}[k]$ is the DFT of the data set from which $x[Mi + l\,]$ moves out and $x[Mi + N + l]$ moves in. Therefore the recursive procedure shown in Eq. (2) is implemented in each (N/M)-point local butterfly module at the $(r - p)$th stage as follows:

$$
\begin{aligned}
G_{i+1,l}[k] &= \sum_{m=0}^{N/M-1} x[M(i+1+m)+l]W_{N/M}^{mk} \\
&= \sum_{m=1}^{N/M} x[M(i+m)+l\,]W_{N/M}^{(m-1)k} \\
&= W_{N/M}^{-k}\left\{\sum_{m=0}^{(N/M)-1} x[M(i+m)+l]W_{N/M}^{mk} \right. \\
&\qquad \left. - x(Mi + l\,] + x[Mi + N + l\,]\right\} \\
&= W_{N/M}^{-k}\{G_{i,l}[k] - x[Mi + l\,] \\
&\qquad + x[Mi + N + l]\}, \\
&0 \leqslant k \leqslant N/M - 1. \qquad (6)
\end{aligned}
$$

3. Real-time implementation of the split-radix FFT

This section briefly describes the real-time implementation of the sr-FFT algorithm [10]. The sr-FFT algorithm derived by Duhamel and Hollmann [6,7] has a simple structure and an explicit theoretical basis. It is a mixture of the radix-2 and radix-4 FFTs. Hence, the algorithm has the advantage of better computational efficiency of the radix-4 FFT, but yet it does not require the FFT size to be an integer power of 4. Considering the N-point DIT sr-FFT with $N = 2^r$, the basic idea is to decompose any local L-point ($L \leqslant N$) DFT block $G[k]$ into one $L/2$-point DFT $G_{2n}[k]$ and two $L/4$-point DFTs $G_{4n+1}[k]$ and $G_{4n+3}[k]$ with the twiddle factors. The implementing procedures were discussed in detail in [6,7,10]. In this part of paper,

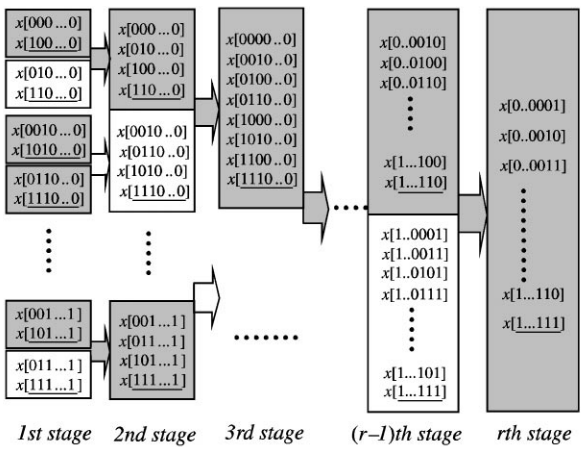

Fig. 2. The block diagram of an N-point DIT sr-FFT structure ($N = 2^r$).

the authors emphasize the real-time implementation procedure firstly presented in [10]. Development of the 'real-time FFT' algorithm was originally inspired by the input/output pruning scheme for FFT computation [11–14,16]. The algorithm is particularly important when dealing with the on-line running spectral analysis for the multi-channel EEG signals which are sampled at a low rate (200 Hz). Ability in synchronizing with the data acquisition process is highly improved with the real-time implementation algorithm. The main idea is to finish building as many butterfly modules as possible with the data points on hand. To on-line construct the local butterfly modules, the algorithm must determine how much work it can accomplish upon receipt of a data point $x[n]$. Fig. 2 displays the block diagram of an N-point DIT-sr-FFT structure ($N = 2^r$). The shaded block are the 'DFT blocks' which actually perform decomposed (into smaller size) DFT computation. The blank blocks are the 'non-DFT blocks' which perform the multiplication by twiddle factors. In [10], the authors proposed the strategies of identifying those local butterfly modules ready to be constructed. Consider an N-point DFT where $N = 2^r$. The array indexes are expressed in binary form: $x[0 \ldots 0] \sim x[1 \ldots 1]$, with the first bit representing the least significant bit. Let $n\text{bit}_1$ denote the number of consecutive bit '1' counted from the leftmost (most significant) bit position. If $n\text{bit}_1$ equals 0 (i.e., the most significant bit is '0'), the algorithm still slugs at the first stage since the first butterfly module can be built only when $x[100\dots0]$ ($x[N/2]$) is collected. For a nonzero $n\text{bit}_1$, the algorithm is able to build the butterfly modules from the first stage up to the ($n\text{bit}_1$)th stage. In addition, the sequence of local butterfly modules being built is $<$ non-DFT $\rightarrow$ DFT $\rightarrow \cdots \rightarrow$ DFT block $>$ for an even $n\text{bit}_1$, which is $<$ DFT $\rightarrow$ non-DFT $\rightarrow \cdots \rightarrow$ DFT block $>$ for an odd $n\text{bit}_1$. Alternatively speaking, the algorithm starts building a DFT (non-DFT) block at the first stage if the $n\text{bit}_1$ is odd (even). Then alternating between the non-DFT and DFT block, the construction of butterfly module always ends at a DFT block when reaching the ($n\text{bit}_1$)th stage. As an illustration, consider that $x[1110..0]$ ($n\text{bit}_1 = 3$) is received. The algorithm is capable of building the local butterfly modules up to the third ($n\text{bit}_3$) stage. The sequence of butterfly modules being built is $<$ DFT $\rightarrow$ non-DFT $\rightarrow$ DFT $>$ from the first stage to the third stage.

4. Real-time implementation of the recursively moving FFT

Integrated with the recursive procedures for moving FFT computation, the real-time implementation procedure is carried out from the ($r - p + 1$)th stage to the last stage given that the moving size $M = 2^p$ and frame size $N = 2^r$. At the beginning ($r - p$) stages, the algorithm need not perform any butterfly structural construction, which is totally replaced by the recursive procedure for updating the (N/M)-point DFTs (Fig. 3). At the ($r - p + 1$)th stage, the algorithm prepares for transition to the sr-FFT structure. If a block at the ($r - p + 1$)th stage belongs to a non-DFT block, its descending DFT blocks at the ($r - p$)th stage are turned into non-DFT blocks by the corresponding twiddle factors (Fig. 4(a)); whereas the DFT blocks at the ($r - p + 1$)th stage are constructed using the conventionals radix-2 FFT algorithm (Fig. 4(b)).

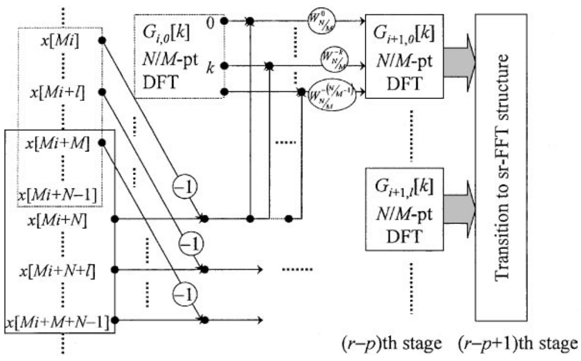

Fig. 3. The recursive procedure implemented at the ($r - p$)th stage.

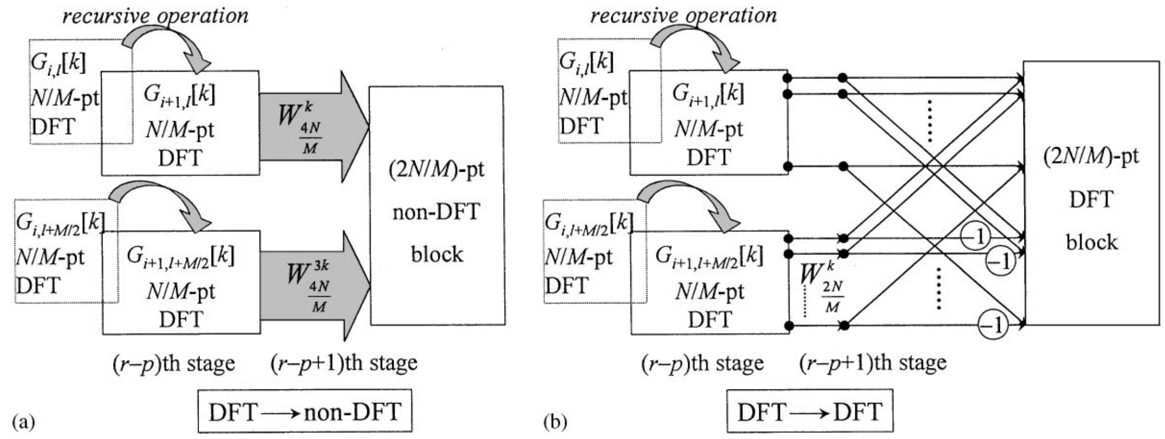

Fig. 4. From the results of recursive operation, the algorithm prepares for transition to the sr-FFT structure by transforming two N/M-point DFT blocks at the ($r - p$)th stage into (a) one $2N/M$-point non-DFT block, and (b) one $2N/M$-point DFT block at the ($r - p + 1$)th stage.

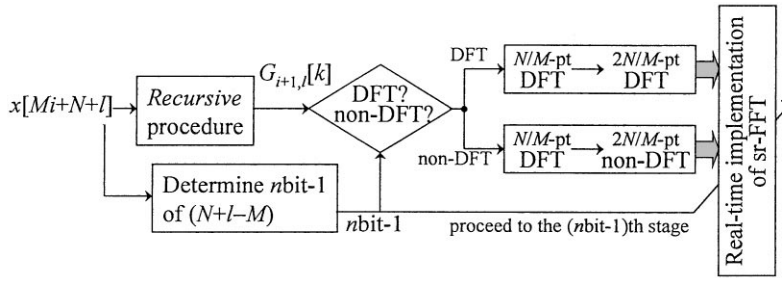

Fig. 5. The block diagram of the integrated algorithm.

Fig. 5 illustrates the block diagram that integrates the recursive procedure with the real-time implementation of sr-FFT. As $x[Mi+N+l]$, $l=0,\ldots,M$, is collected, the recursive procedure automatically updates the (N/M)-point DFT $G_{i+1,l}[k]$ by using Eq. (6). From the array index of current frame, $x_{i+1}[n]=x[i(M+1)+n]$, $n=0,\ldots,N-1$, the algorithm counts the $n\mathrm{bit}_1$ of $n=N+l-M$. The value of $n\mathrm{bit}_1$ directs the construction of butterfly modules up to the ($n\mathrm{bit}_1$)th stage. In addition, $n\mathrm{bit}_1$ enables the algorithm to determine the block attribute (DFT or non-DFT) when preparing for a transition to the sr-FFT structure (Fig. 5). Then the algorithm will construct as many butterfly modules as possible in the real-time manner. The algorithm fully utilizes the time span of the long sampling period and accordingly provides a tool to better synchronize with on-line data acquisition process.

5. Evaluation of computational complexity

To analyze the computational complexity of the real-time moving FFT algorithm, the authors compute the numbers of complex arithmetic operations required by the recursive procedure (up to the $(r-p)$th stage) and the real-time sr-FFT implementation (from the $(r-p+1)$th to the rth stage). Let $N_{\mathrm{recurs}}(\mathrm{MUL_c})$ and $N_{\mathrm{recurs}}(\mathrm{ADD_c})$ denote the number of complex multiplications and complex additions, respectively, required by the recursive operation. At the $(r-p)$th stage, M blocks of (N/M)-point DFTs are evaluated by recursive procedure. Note that the $(-x(Mi+l)+x[Mi+N+l])$ in Eq. (6) only needs to be evaluated once when computing the $G_{i+1,l}[k]$, $0\leqslant k\leqslant N/M-1$. According to Eq. (6), each block requires $1+N/M$ complex additions and N/M complex multiplications. Thus $N_{\mathrm{recurs}}(\mathrm{MUL_c})=N$ and $N_{\mathrm{recurs}}(\mathrm{ADD_c})=M+N$. At the $(r-p+1)$th stage, the algorithm actually prepares for a transition to the sr-FFT structure (Fig. 4). The number of complex additions and complex multiplications are denoted by $N_{r-p+1}(\mathrm{ADD_c})$ and $N_{r-p+1}(\mathrm{MUL_c})$, respectively. The real-time sr-FFT implementation (from the $(r-p+2)$th to the rth stage) requires $N_{\text{sr-FFT}}(\mathrm{MUL_c})$ complex multiplications and $N_{\text{sr-FFT}}(\mathrm{ADD_c})$ complex additions. Detailed derivation of these numbers is presented in Appendix A. To summarize, the total number of complex multiplications $N_{\mathrm{total}}(\mathrm{MUL_c})$ is (Eq. (A.7))

$$N_{\mathrm{total}}(\mathrm{MUL_c})=\left(\frac{p}{3}+\frac{13}{9}\right)N+\frac{5N}{9M}(-1)^p, \tag{7}$$

and the total number of complex additions $N_{\mathrm{total}}(\mathrm{ADD_c})$ is (Eq. (A.8))

$$N_{\mathrm{total}}(\mathrm{ADD_c})=M+(p+1)N. \tag{8}$$

Table 1 lists the $N_{\mathrm{total}}(\mathrm{ADD_c})$ and $N_{\mathrm{total}}(\mathrm{MUL_c})$ for different moving sizes and frame sizes. The shaded rows displays the numbers of complex multiplications and additions without implementing the recursive procedure. In that case, the algorithm simply constructs the sr-FFT structure using the real-time implementation method. Note that the real-time FFT algorithm [10] has about the same order of computational complexity as the normal

Table 1
Comparison of the number of complex arithmetic operations for different moving sizes (M) and frame sizes (N)

		@Not recursive	$M = 4$ $p = 2$	$M = 8$ $p = 3$	$M = 16$ $p = 4$	$M = 32$ $p = 5$	$M = 64$ $p = 6$	$M = 128$ $p = 7$	$M = 256$ $p = 8$
$N = 256$	$N_{\text{total}}(\text{ADD}_c)$	2134	772	1032	1296	1568	1856	2176	
	$N_{\text{total}}(\text{MUL}_c)$	626	576	608	720	792	884	966	
$N = 512$	$N_{\text{total}}(\text{ADD}_c)$	4778	1540	2056	2576	3104	3648	4224	4864
	$N_{\text{total}}(\text{MUL}_c)$	1422	1152	1216	1440	1584	1768	1932	2106
$N = 1024$	$N_{\text{total}}(\text{ADD}_c)$	10582	3076	4104	5136	6176	7232	8320	9472
	$N_{\text{total}}(\text{MUL}_c)$	3186	2304	2432	2880	3168	3536	3864	4212

@: The algorithm simply constructs the sr-FFT structure using the real-time implementation method [10] without implementing the recursive procedure.

Table 2
The t_{last} is the computer time spent on constructing the residuary butterfly modules upon receipt of the last data point $x[511]$. The t_{total} is the total time required by the integrated algorithm to compute one 512-point FFT for different moving sizes ($M = 2^p$, $p = 0,1,\dots,9$). The algorithm is run on Pentium 66

	$p = 9$ $M = 512$	$p = 8$ $M = 256$	$p = 7$ $M = 128$	$p = 6$ $M = 64$	$p = 5$ $M = 32$	$p = 4$ $M = 16$	$p = 3$ $M = 8$	$p = 2$ $M = 4$	$p = 1$ $M = 2$	$p = 0$ $M = 1$
t_{last} (ms)	0.45	0.45	0.45	0.47	0.46	0.46	0.48	0.47	0.49	0.49
t_{total} (ms)	3.30	2.80	2.42	2.08	1.92	1.54	1.37	1.04	0.88	0.49

sr-FFt algorithm. However, it provides better synchronization ability with the real data acquisition process. In comparison with the numbers listed in the shaded rows for each given N, the integrated approach does not advantage the computational efficiency as the moving size M increases (the right side of the bold-faced lines, Table 1). Considering the example of $N = 512$ and $M \geqslant 64$, one would rather apply the real-time FFT algorithm without implementing the recursive procedure. The experimental results listed in Table 2 corroborates the above analysis. Note that the real-time FFT algorithm without implementing the recursive procedure spends 2.01 ms on computing the 512-point sr-FFT on Pentium 66. While integrated with the recursive procedure, it results in poorer computational efficiency ($t_{\text{total}} \geqslant 2.08$ ms) when $M \geqslant 64$. Table 2 also shows the synchronization ability of real-time implementation. When $M = 128$, the integrated algorithm takes 0.45 ms (t_{last}) to build the residuary butterfly modules upon receipt of the last data point. Yet the conventional whole-block algorithm takes 2.42 ms after all the 512 data points have been collected. The efficacy of real-time implementation is more evident when applying to the multi-channel running spectral analyisis. According to Table 2, the total time required to analyze the running spectrum of a long-time EEG record (off-line) is simply $t_{\text{total}} \times$(number of frames). As addressed in the beginning, a total of 11,243 512-point DFTs need to be evaluated in a one-hour EEG record ($M = 64$). The conventional sr-FFT algorithm takes 27.21 s to analyze the whole record, whereas the real-time, moving FFT algorithm proposed takes 23.39 s. When $M = 8$ is used for a better temporal resolution, the computational time is 217.65 s (conventional) and 123.22 s (proposed), respectively. A significant improvement on computational efficiency is achieved for a small moving size.

6. Conclusion

This paper presents a new approach for computing the real-time, running Fourier spectra. By integrating the recursive procedure with the real-time implementation strategy presented in [10], we are able to achieve better computational efficiency and synchronization ability. The real-time FFT algorithm without implementing the recursive procedure takes effect when the moving size is large ($N/M < 16$). On the other hand ($N/M \geqslant 16$), the recursive procedure plays a key role in reducing the number of complex arithmetic operations.

Appendix A

We follow the approach developed in [10] for computing the number of complex multiplications and complex additions required to construct the butterfly modules from the $(r-p+1)$th to the rth stage. The total numbers of complex multiplications and complex additions at the $(r-j)$th stage are

$$N_{r-j}(\mathrm{MUL_c}) = 2^{r-1}\cdot\left[\frac{2}{3}+\frac{1}{3}\left(-\frac{1}{2}\right)^{j}\right],\quad 0\leqslant j\leqslant p-1, \tag{A.1}$$

$$N_{r-j}(\mathrm{ADD_c}) = (3\cdot 2^{r-1})\cdot\left[\frac{2}{3}+\frac{1}{3}\left(-\frac{1}{2}\right)^{j}\right],\quad 0\leqslant j\leqslant p-1. \tag{A.2}$$

Note that, at the $(r-p+1)$th stage, the algorithm actually prepares for a transition to the sr-FFT structure (Fig. 4). According to Fig. 4(a), it requires $2N/M = 2^{r-p+1}$ complex multiplications to transform two (N/M)-point DFT blocks at the $(r-p)$th stage into one $(2N/M)$-point non-DFT block. While constructing the $(2N/M)$-point DFT block from two DFT blocks at the preceding stage, $2N/M = 2^{r-p+1}$ complex additions and $N/M = 2^{r-p}$ complex multiplications are performed. The sr-FFT structure at the $(r-p+1)$th stage consists of $2^{p-1}[\frac{2}{3}+\frac{1}{3}(-\frac{1}{2})^{p-1}]$ DFT blocks and $2^{p-1}[\frac{1}{3}-\frac{1}{3}(-\frac{1}{2})^{p-1}]$ non-DFT blocks [10]. Hence the number of complex arithmetic operations performed at the $(r-p+1)$th stage is

$$\begin{aligned} N_{r-p+1}(\mathrm{MUL_c}) &= 2^{r-p}2^{p-1}[\tfrac{2}{3}+\tfrac{1}{3}(-\tfrac{1}{2})^{p-1}] \\ &\quad + 2^{r-p+1}2^{p-1}[\tfrac{1}{3}-\tfrac{1}{3}(-\tfrac{1}{2})^{p-1}] \\ &= \frac{2}{3}N + \frac{N}{3M}(-1)^{p},\quad 1\leqslant p\leqslant r-1, \end{aligned} \tag{A.3}$$

$$\begin{aligned} N_{r-p+1}(\mathrm{ADD_c}) &= 2^{r-p+1}2^{p-1}[\tfrac{2}{3}+\tfrac{1}{3}(-\tfrac{1}{2})^{p-1}] \\ &= \frac{2}{3}N - \frac{2N}{3M}(-1)^{p},\quad 1\leqslant p\leqslant r-1, \end{aligned} \tag{A.4}$$

To construct the butterfly modules from the $(r-p+2)$th to the rth stage, the number of complex multiplications $N_{\text{sr-FFT}}(\mathrm{MUL_c})$ is

$$\begin{aligned} N_{\text{sr-FFT}}(\mathrm{MUL_c}) &= \sum_{j=0}^{p-2} N_{r-j}(\mathrm{MUL_c}) \\ &= 2^{r-1}\sum_{j=0}^{p-2}\left[\frac{2}{3}+\frac{1}{3}\left(-\frac{1}{2}\right)^{j}\right] \\ &= \frac{p}{3}N - \frac{2}{9}N + \frac{2N}{9M}(-1)^{p}, \end{aligned} \tag{A.5}$$

and the number of complex additions $N_{\text{sr-FFT}}(\mathrm{ADD_c})$ is

$$\begin{aligned} N_{\text{sr-FFT}}(\mathrm{ADD_c}) &= \sum_{j=0}^{p-2} N_{r-j}(\mathrm{ADD_c}) \\ &= (3\times 2^{r-1})\sum_{j=0}^{p-2}\left[\frac{2}{3}+\frac{1}{3}\left(-\frac{1}{2}\right)^{j}\right] \\ &= pN - \frac{2}{3}N + \frac{2N}{3M}(-1)^{p}. \end{aligned} \tag{A.6}$$

Finally, the total number of complex multiplications $N_{\text{total}}(\mathrm{MUL_c})$ is

$$\begin{aligned} N_{\text{total}}(\mathrm{MUL_c}) &= N_{\text{recurs}}(\mathrm{MUL_c}) + N_{r-p+1}(\mathrm{MUL_c}) \\ &\quad + N_{\text{sr-FFT}}(\mathrm{MUL_c}) \\ &= \left(\frac{p}{3}+\frac{13}{9}\right)N + \frac{5N}{9M}(-1)^{p}, \end{aligned} \tag{A.7}$$

and the total number of complex additions $N_{\text{total}}(\mathrm{ADD_c})$ is

$$\begin{aligned} N_{\text{total}}(\text{ADD}_c) &= N_{\text{recurs}}(\text{ADD}_c) + N_{r-p+1}(\text{ADD}_c) \\ &\quad + N_{\text{sr-FFT}}(\text{ADD}_c) \\ &= M + (p+1)N. \end{aligned} \tag{A.8}$$

References

[1] M.G. Amin, A new approach to recursive Fourier transform, Proc. IEEE 75 (11) (November 1987) 1537–1538.

[2] J.L. Aravena, Recursive moving window DFT algorithm, IEEE Trans. Comput. 39 (1) (January 1990) 145–148.

[3] G. Bongiovanni, P. Corsini, G. Frosini, Procedures for computing the discrete Fourier transform on staggered blocks, IEEE Trans. Acoust. Speech Signal Process. ASSP-24 (2) (April 1976) 132–137.

[4] W. Chen, N. Kehtamavaz, T. W. Spencer, An efficient recursive algorithm for time-varying Fourier transform, IEEE Trans. Signal Process. 41 (7) (July 1993) 2488–2490.

[5] J.W. Cooley, J.W. Tukey, An algorithm for machine computation of complex Fourier series, Math. Comput. 19 (April 1965) 297–301.

[6] P. Duhamel, Implementation of 'split-radix' FFT algorithms for complex real and real-symmetric data, IEEE Trans. Acoust. Speech Signal Process. ASSP-34 (April 1986) 285–295.

[7] P. Duhamel, H. Hollmann, Split radix FFT algorithm, Electron. Lett. 20 (1) (January 1984) 14–16.

[8] B.G. Goldberg, A continuous recursive DFT analyzer – The discrete coherent memory filter, IEEE Trans. Acoust. Speech Signal Process. ASSP-28 (6) (December 1980) 760–762.

[9] J.H. Halberstein, Recursive, complex Fourier analysis for real-time applications, Proc. IEEE (Lett.) 54 (June 1966) 903.

[10] P.-C. Lo, Y.-Y. Lee, Real-time implementation of the split-radix FFT – An algorithm to efficiently construct local butterfly modules, Signal Processing 71 (1998) 291–299.

[11] J.D. Markel, FFT pruning, IEEE Trans. Audio Electroacoust. AU-19 (4) (December 1971) 305–311.

[12] A.V. Oppenheim, R.W. Schafer, Discrete-Time Signal Processing, Prentice-Hall, Englewood Cliffs, NJ, 1989, Chapter 9.

[13] D.E. Paneras, R. Mani, S.H. Nawab, STFT computation using pruned FFT algorithms, IEEE Signal Process. Lett. 1 (4) (April 1994) 61–63.

[14] C. Roche, A split-radix partial input/output fast Fourier transform algorithm, IEEE Trans. Signal Process. 40 (5) (May 1992) 1273–1276.

[15] B.G. Sherlock, D.M. Manro, Moving discrete Fourier transform, IEE Proc.-F 139 (4) (August 1992) 279–282.

[16] H.V. Sorensen, C.S. Burrus, Efficient computation of the DFT with only a subset of input or output points, IEEE Trans. Signal Process. 41 (3) (March 1993) 1184–1200.

[17] M. Unser, Recursion in short-time signal analysis, Signal Processing 5 (1983) 229–240.