論文 / 著書情報
Article / Book Information

Title	Generalized and Partial FFT
Authors	Todor Cooklev, AKINORI NISHIHARA
Citation	IEICE Trans. Fundamentals, Vol. E77, No. 9, pp. 1466-1474
Pub. date	1994, 9
URL	http://search.ieice.org/
Copyright	(c) 1994 Institute of Electronics, Information and Communication Engineers

IEICE TRANS. FUNDAMENTALS, VOL. E77–A, NO. 9 SEPTEMBER 1994

PAPER Special Section of Papers Selected from the 8th Digital Signal Processing Symposium

Generalized and Partial FFT*

Todor COOKLEV† and Akinori NISHIHARA†, Members

SUMMARY The relation between computing part of the FFT spectrum and the so-called generalized FFT (GFFT) is clarified, leading to a new algorithm for performing partial FFTs. The method can be applied when only part of the output is required or when the input data sequence contains many zeros. Such cases arize for example in decimation and interpolation and also in computing linear convolutions. The technique consists of decomposing the DFT into several generalized DFTs. Efficient algorithms for these generalized DFTs exist. The computational complexity of the new approach is roughly equal to the complexity of previous techniques, but the structure is superior, because only one type of butterfly is used and a few lines of code are sufficient. The theoretical properties of the GDFT are given. The case of multidimensional signals, defined on arbitrary sampling lattices is also considered.

key words: *fast Fourier transforms, orthogonal transforms, digital signal processing*

1. Introduction

The DFT and its inverse are defined as

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk}, \tag{1}$$

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) W_N^{-nk}, \tag{2}$$

where the indices are from 0 to $N-1$ and $W_N^{nk} = \exp[-j2\pi nk/N]$. The fast algorithms to calculate the DFT are called collectively the FFT. In some practical applications it is necessary to compute only part of the DFT spectrum, or the input sequence may be sparse. For the sake of brevity we can call the former case a partial output transform and the latter-partial input transform, or partial DFTs. When interpolation is computed zero-fill is also performed. In spectral analysis sometimes a finer spectral resolution is required, achieved again by zero-fill. Yet another example is when the frequency response of FIR digital filters is evaluated from the impulse response. Then the time-domain sequence is appended with zeros prior to the FFT. It is important to eliminate the operations on zeros in such cases.

In general there are two types of algorithms for the purpose. The first type of algorithms compute the DFT approximately [4], [5], [8]. The techniques [4], [5] are based on performing digital filtering on the sequence, followed by FFT.

We shall focus on the second type of methods that yield exact results if the finite-word-length effects are not considered. One solution is pruning, which can be applied to the radix-2 decimation-in-frequency (DIF) [6], radix-2 decimation-in-time (DIT) [7] and the split-radix DIF and DIT algorithms [11]. The idea of pruning is very simple. Pruning recognizes that there are three types of butterflies: butterflies with zero inputs are eliminated, butterflies with zero and non-zero inputs are pruned and the butterflies with non-zero inputs are left unchanged. This approach is not associated only with the radix-2 algorithms. Recently it was applied to the split-radix algorithm [11], which is the most efficient algorithm when the length of the transform is a power of 2 [14]. When real-data is processed pruning can also be applied [12].

The pruning scheme, however, has some inherent disadvantages, which motivated the present investigation. When pruning is applied the number of butterfly types increases, the complexity of the program and the occupied memory also increase. Pruning the split-radix algorithm [11], or the real-valued algorithms [12] requires many `if` statements. The disadvantages of pruning may even offset the gain in the computational complexity. The basic conclusion is that simply the number of multiplications and additions is not the only one figure of merit.

Sorensen et al. proposed an algorithm that is an alternative to pruning [9]. They claim that their algorithm is superior to the pruning approach, which in terms of computational complexity alone is not true (the authors of Ref. [9] compare radix-2 with split-radix implementations, which is unfair). The algorithm of Sorensen and Burrus consists of computing several smaller DFTs and combining the results with some additional multiplications [9]. (These additional multiplications can be performed using the Goertzel algorithm to reduce somewhat the computational cost.) This algorithm indeed has a better structure than pruning, but in terms of computational complexity alone pruning the split-radix algorithm [11] is better, because these additional multiplications are avoided. Another recently proposed algorithm [10] avoids these multiplications by developing another generalized FFT (but different from

Manuscript received January 31, 1994.
Manuscript revised June 20, 1994.

†The authors are with the Faculty of Engineering, Tokyo Institute of Technology, Tokyo, 152 Japan.

*This paper was presented at the 8th DSP Symposium.

the one here) and using a chirp z-transform (CZT) algorithm for the computation.

In this paper we present another solution, that is efficient under the framework of a broad class of FFT algorithms. The basic idea is to decompose the original partial DFT to several complete, but *generalized* DFTs. These GDFTs can be evaluated using fast algorithms. The complexity of our approach is the same as pruning for implementations with the same type of algorithm (radix-2 and split-radix algorithms will be discussed; other algorithms should also be of interest).

2. Partial Output and Input Transformations

2.1 Computing Only a Subset of the Output Points

We are interested in the DFT spectrum only for $k = 0, 1, \ldots, M-1$, where naturally $M \leqq N$. We also impose the restriction $MQ = N$. There are cases where such problem arizes and it can be solved as follows:

Substituting the index mapping

$$n = Qn_1 + n_2, \tag{3}$$

where

$$n_1 = 0, \ldots M-1 \tag{4}$$

$$n_2 = 0, \ldots Q-1 \tag{5}$$

in (1) we get

$$X(k) = \sum_{n_2=0}^{Q-1} \sum_{n_1=0}^{M-1} x(n_1Q + n_2) W_N^{(n_1Q+n_2)k}, \tag{6}$$

where $k = 0, 1, \ldots, M-1$. Now we use the restriction that N is a multiple of M

$$W_N^{(n_1Q+n_2)k} = W_{MQ}^{(n_1Q+n_2)k} \tag{7}$$

$$= W_M^{(n_1+\frac{n_2}{Q})k} \tag{8}$$

to obtain

$$X(k) = \sum_{n_2=0}^{Q-1} \left[\sum_{n_1=0}^{M-1} x(n_1Q + n_2) W_M^{(n_1+\frac{n_2}{Q})k} \right] \tag{9}$$

$$= \sum_{n_2=0}^{Q-1} X_{n_2}(k) \qquad k = 0, 1, \ldots, M-1 \tag{10}$$

Interestingly enough, the expression inside the brackets in (9) is a generalized DFT (GDFT), as suggested in Ref. [13], without considering partial transforms however. We think that the term parametric DFT is more suitable, because it indicates the direction of the generalization. For consistency, however, we shall keep calling it generalized. If $n_2/Q = \alpha < 1$ the GDFT can be written as

$$X_\alpha(k) = \sum_{n_1=0}^{M-1} x(n_1Q + n_2) W_M^{(n_1+\alpha)k}, \tag{11}$$

where the indices run from 0 to $M-1$. The GDFT becomes the standard DFT for $\alpha = 0$.

Our idea is that Q generalized DFTs (GDFTs) can be used for the computation of one partial DFT, as demonstrated by (10). Later we shall demonstrate that fast algorithms for the generalized DFT exist as long as M (not N) is an appropriate integer. Therefore this approach has somewhat broader significance than pruning, similarly as Ref. [9]. At this point let us make a brief comparison with Ref. [9]. The algorithm [9] requires Q M-point FFTs plus $(Q-1)(M-1)$ complex multiplications and $(M-1)(Q-1)$ complex additions (if no filtering is used). Our approach consists only of Q M-point GFFTs plus $(M-1)(Q-1)$ additions. However the total complexity is roughly the same, because the FFT requires slightly less multiplications than the GFFT. Later we shall make a detailed comparison.

2.2 FFT of a Sparse Sequence

Consider the FFT of a sequence, containing many trailing zeroes. Such is case when we compute linear convolutions. Since the FFT-based approach computes circular convolution, the input sequences are appended with zeroes prior to the FFT. Finally the circular convolution would be equal to the linear convolution in one period [2].

Suppose we have an N-point sequence, in which only the first M points are different from zero and we want to compute the complete spectrum. Then (1) becomes

$$X(k) = \sum_{n=0}^{M-1} x(n) W_N^{nk}, \quad k = 0, 1, \ldots, N-1 \tag{12}$$

The above expression is not a complete DFT. From it, however, we can obtain several DFTs. Once again we impose the constraint $MQ = N$. Now if we use the index mapping

$$k = k_1Q + k_2, \tag{13}$$

where

$$k_1 = 0, \ldots, M-1 \tag{14}$$

$$k_2 = 0, \ldots, Q-1 \tag{15}$$

we shall obtain

$$X(k_1Q + k_2) = \sum_{n=0}^{M-1} x(n) W_N^{n(k_1Q+k_2)} \tag{16}$$

$$= \sum_{n=0}^{M-1} x(n) W_M^{n(k_1+\frac{k_2}{Q})} \tag{17}$$

$$= \sum_{n=0}^{M-1} x(n) W_M^{n(k_1+\beta)}, \tag{18}$$

for $\beta = \frac{n_2}{Q}$. The above expression is Q GDFTs for each value of k_2, corresponding to $k_2 = 0, \ldots, Q-1$. As

long as M (not necessarily N) is a power of two FFT algorithms exist.

3. FFT for Sequences with Lengths Multiples-of-Power-of-Two

There are cases where the length of the sequence is not a perfect power-of-two. A widely performed technique is zero-fill. So the first question that must be answered is why not zero fill. The zero-fill procedure introduces sharp discontinuities in the signal. Because of this several authors are not quite satisfied with the zero-fill technique [17], [18], [20]†.

So we shall look for other techniques. If the mapping (3) and

$$k = iM + m \tag{19}$$

are used in the DFT (1) we obtain

$$X(iM+m) = \sum_{n_2=0}^{Q-1} W_N^{(iM+m)n_2} \sum_{n_1=0}^{M-1} x(n_1Q+n_2) W_M^{(m+iM)n_1}. \tag{20}$$

Therefore the algorithm for computing the DFT of such sequences consists of Q M-point FFTs, followed by $N(Q-1)$ additions and $N(Q-1)$ multiplications. This is what the authors of Ref. [20] have in mind.

A dual algorithm is also possible. If the mapping (13) and

$$n = iM + m \tag{21}$$

is used after some simple manipulation we obtain

$$X(k_1Q+k_2) = \sum_{i=0}^{Q-1} W_Q^{(Qk_1+k_2)i} \sum_{m=0}^{M-1} x(iM+m) W_M^{(k_1+k_2/Q)m} \tag{22}$$

(22) consists of Q M-point generalized FFTs (instead of ordinary FFTs, as in (20), followed by $N(Q-1)$ multiplications and $N(Q-1)$ additions.

4. GDFT for Time-Domain Interpolation

For time-domain interpolation zeros are inserted in the middle of the frequency-domain sequence and some additional care must be exercised. Suppose we have a signal $x(n)$, defined for $n = 0, \ldots, N-1$. We are looking for another signal $\hat{x}(m)$ of duration lN, which interpolates $x(n)$, that is $\hat{x}(ln) = x(n)$ for $n = 0, 1, \ldots, N-1$. Or, $l-1$ samples are inserted between every two samples and $l-1$ samples are added after the last sample. The algorithm for this is known [15]; our point is that the GDFT can also be used. First the DFT of $x(n)$ is computed. Then the DFT of $\hat{x}(m)$ is

$$\hat{X}(k) = \begin{cases} lX(k), & 0 \leqq k < \frac{N}{2} \\ \frac{l}{2}X(k), & k = \frac{N}{2} \\ 0, & \frac{N}{2} < k < lN - \frac{N}{2} \\ \frac{l}{2}X(k-(l-1)N), & k = lN - \frac{N}{2} \\ lX(k-(l-1)N), & lN - \frac{N}{2} < k < lN. \end{cases} \tag{23}$$

The Nyquist component is weighted by $1/2$, to preserve the Hermitian symmetry. The IDFT of $\hat{X}(k)$ gives the desired signal $\hat{x}(m)$.

$$\hat{x}(m) = \frac{1}{lN} \sum_{k=0}^{lN-1} \hat{X}(k) W_{lN}^{-mk} \tag{24}$$

$$= \frac{1}{lN} \sum_{k=0}^{N/2-1} \hat{X}(k) W_{lN}^{-mk} + \frac{1}{lN} \hat{X}(N/2) e^{j\pi m/l} + \frac{1}{lN} \sum_{k=N/2+(l-1)N}^{lN-1} \hat{X}(k) W_{lN}^{-mk}. \tag{25}$$

We use the widely accepted definitions of the DFT and IDFT, according to which most FFT software operates. Other definitions can also be used [10]. Since $\hat{X}(k)$ is clearly a sparse sequence then it is highly desireable to reduce the complexity of the lN-point IFFT. N-point transformation is sufficient and N is required to be multiple-of-power-of-two.

The second summation can be changed

$$\hat{x}(m) = \frac{1}{lN} \sum_{k=0}^{N/2-1} \hat{X}(k) W_{lN}^{-mk} + \frac{1}{lN} \hat{X}(N/2) e^{j\pi m/l} + \frac{1}{lN} \sum_{k=N/2}^{N-1} \hat{X}[k+(l-1)N] W_{lN}^{-m[k+(l-1)N]} \tag{26}$$

$$= \frac{1}{lN} \sum_{k=0}^{N/2-1} \hat{X}(k) W_N^{-\frac{m}{l}k} + \frac{1}{lN} \hat{X}(N/2) e^{j\pi m/l} + \frac{1}{lN} \sum_{k=N/2}^{N-1} \hat{X}[k+(l-1)N] W_N^{-\frac{m}{l}[k+(l-1)N]} \tag{27}$$

Now we use first the index mapping

$$m = m_1 l + m_2 \tag{28}$$

$$m_1 = 0, \ldots, N-1 \tag{29}$$

$$m_2 = 0, 1, \ldots l-1 \tag{30}$$

and second

†Our study was done completely independently of Ref. [20]. The authors of Ref. [20] do not consider partial transforms while we present a more detailed and complete study.

$$W_N^{-(m_1+\frac{m_2}{l})[(l-1)N]} = \exp\left[j2\pi\frac{m_2}{l}(l-1)\right] \tag{31}$$

to obtain

$$\hat{x}(m_1 l + m_2) = \frac{1}{lN}\sum_{k=0}^{N/2-1}\hat{X}(k)W_N^{-(m_1+\gamma)k} + \frac{1}{lN}\hat{X}(N/2)e^{j\pi m/l} + \frac{1}{lN}\sum_{k=N/2}^{N-1}\left[\hat{X}[k+(l-1)N]e^{j2\pi\gamma(l-1)}\right] W_N^{-(m_1+\gamma)k}, \tag{32}$$

where obviously $\gamma = m_2/l$. Our final step is to define

$$\tilde{X}(k) = \begin{cases} \frac{1}{l}\hat{X}(k), & 0 \leqq k < \frac{N}{2} \\ \frac{1}{l}\hat{X}(k)\left[1+e^{j2\pi\gamma(l-1)}\right], & k = \frac{N}{2} \\ \frac{1}{l}\hat{X}[k+(l-1)N]e^{j2\pi\gamma(l-1)}, & \frac{N}{2} < k < N. \end{cases} \tag{33}$$

Therefore instead of lN-point IFFT of the sequence $\hat{X}(k)$, the final sequence can be obtained by $l-1$ N-point GIFFTs:

$$\hat{x}(m_1 l + m_2) = \frac{1}{N}\sum_{k=0}^{N-1}\tilde{X}(k)W_N^{-(m_1+\gamma)k} \tag{34}$$

The above formula is l IGFFTs, but one of them (corresponding to $\gamma = 0$) need not be computed, because the result is the original sequence. Using a technique, outlined in Sect. 4 it is possible to relax the condition N to be power-of-two. This is important especially in computing interpolation sequences. If N is not a power-of-two using zero-fill technique to compute the forward FFT in step 1 means that we include the trailing zeroes in the interpolation. If $N = QM$ then

$$\hat{x}[(q_1+pM)l+m_2] = \frac{1}{N}\sum_{k_2=0}^{Q-1}W_N^{(q_1+pM)k_2} \sum_{k_1=0}^{M-1}\tilde{X}(k_1 Q+k_2)W_M^{-(q_1+pM+\gamma)k_1} \tag{35}$$

$$= \frac{1}{N}\sum_{k_2=0}^{Q-1}W_N^{(q_1+pM)k_2} \sum_{k_1=0}^{M-1}\tilde{X}(k_1 Q+k_2)W_M^{-(q_1+\gamma)k_1}, \tag{36}$$

since the GDFT kernel $W_M^{-(q_1+\gamma)k_1}$ is periodic with respect to q_1.

5. Properties of the GDFT

The generalized DFT was introduced only as a means of calculating the partial FFT. However the GDFT per se has some interesting mathematical properties, studied in this section. At the end of the previous section we used that the GDFT kernel $\exp[-j2\pi(n+\alpha)k]$ is periodic with respect to n

$$e^{-j2\pi(n\pm sN+\alpha)k/N} = e^{-j2\pi(n+\alpha)k/N} \tag{37}$$

and parametrically periodic with respect to k

$$e^{-j2\pi(n+\alpha)(k\pm sN)/N} = e^{\mp 2\pi(n+\alpha)s} e^{-j2\pi(n+\alpha)k/N} \tag{38}$$

Let us define the k-th row and n-th column of the GDFT matrix as

$$\boldsymbol{F}_\alpha = \frac{1}{\sqrt{M}}W_M^{(n+\alpha)k} \tag{39}$$

It is better to study the properties of the normalized matrix. Obviously the standard DFT matrix corresponds to $\alpha = 0$. $\boldsymbol{F}_\alpha$ is not symmetric, $\boldsymbol{F}_\alpha^T \neq \boldsymbol{F}_\alpha$. The orthogonality properties remain valid

$$\sum_{n=0}^{M-1}W_M^{(n+\alpha)k}(W_M^{(n+\alpha)j})^* = \begin{cases} M & k=j \\ 0 & k\neq j \end{cases} \tag{40}$$

$$\sum_{k=0}^{M-1}W_M^{(n+\alpha)k}(W_M^{(j+\alpha)k})^* = \begin{cases} M & n=j \\ 0 & n\neq j \end{cases} \tag{41}$$

It is verified that the matrix $\boldsymbol{F}_\alpha$ satisfies the unitary property

$$(\boldsymbol{F}_\alpha^T)^* = \boldsymbol{F}_\alpha^{-1} \tag{42}$$

Equivalently

$$\boldsymbol{F}_\alpha(\boldsymbol{F}_\alpha^T)^* = (\boldsymbol{F}_\alpha^T)^*\boldsymbol{F}_\alpha = \boldsymbol{I}. \tag{43}$$

Another interesting properties are

$$\boldsymbol{F}_\alpha\boldsymbol{F}_\alpha^T = \boldsymbol{\Gamma}_\alpha \tag{44}$$

$$\boldsymbol{\Gamma}_\alpha^2(\boldsymbol{\Gamma}_\alpha^2)^* = \boldsymbol{I}, \tag{45}$$

where $\boldsymbol{\Gamma}_\alpha$ is a circular matrix:

$$\boldsymbol{\Gamma}_\alpha = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & \dots & 0 & e^{(-j2\pi\alpha)} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & e^{(-j2\pi\alpha)} & \dots & 0 \end{pmatrix} \tag{46}$$

$$\boldsymbol{\Gamma}_\alpha^2 = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & e^{(-j4\pi\alpha)} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & e^{(-j4\pi\alpha)} \end{pmatrix} \tag{47}$$

The eigenvalues of the DFT matrix (corresponding to $\alpha = 0$) are $\pm 1, \pm j$ with appropriate multiplicities. The eigenvalues of $\boldsymbol{F}_\alpha$ are different, but all are on the unit circle.

Finally we point that the fractional Fourier transform, introduced in Ref. [10] is based on integer powers nk of fractional roots of unity $\exp(-j2\pi/\delta)$ and only the CZT can be used, but not the radix-2 and split-radix algorithms. On the contrary the GDFT is based on fractional powers e.g. $(n+\alpha)k$ of integral roots of unity $\exp(-j2\pi/N)$.

6. Generalized FFT

This section is important, because if there are no FFT algorithms for the generalized DFTs then the approach is not of practical importance. DIT algorithms exist for (18) ,(22), and (34), and DIF algorithms for (11).

6.1 Radix-2 DIF GFFT

Transforms of the type

$$X(k) = \sum_{n=0}^{M-1} x(n) W_M^{(n+\alpha)k} \tag{48}$$

can be evaluated by means of a DIF algorithm:

$$X(2k) = \sum_{n=0}^{M/2-1} [x(n) + x(n+M/2)] \; W_{M/2}^{(n+\alpha)k} \tag{49}$$

$$X(2k+1) = \sum_{n=0}^{M/2-1} [x(n) - x(n+M/2)] \; W_{M/2}^{(n+\alpha)k} W_M^{n+\alpha} \tag{50}$$

We have decomposed the original M-point GDFT to two $M/2$ GDFT of the same type. Continuing in the same manner the complete signal-flow-graph would be obtained. The difference with the ordinary DIF algorithm is the twiddle factor. Instead of $W_{M/2}^{n}$ the new twiddle factors are $W_{M/2}^{n+\alpha}$. Therefore the graph of the GFFT is exactly the same, only the exponents in the twiddle factors are different.

6.2 Radix-2 DIT GFFT

Transforms of the type

$$X(k) = \sum_{n=0}^{M-1} x(n) W_M^{n(k+\beta)} \tag{51}$$

can be evaluated by means of a radix-2 DIT algorithm.

$$X(k) = \sum_{n=0}^{M/2-1} x(2n) W_{M/2}^{n(k+\beta)} + W_M^{k+\beta} \sum_{n=0}^{M/2-1} x(2n+1) W_{M/2}^{n(k+\beta)} \tag{52}$$

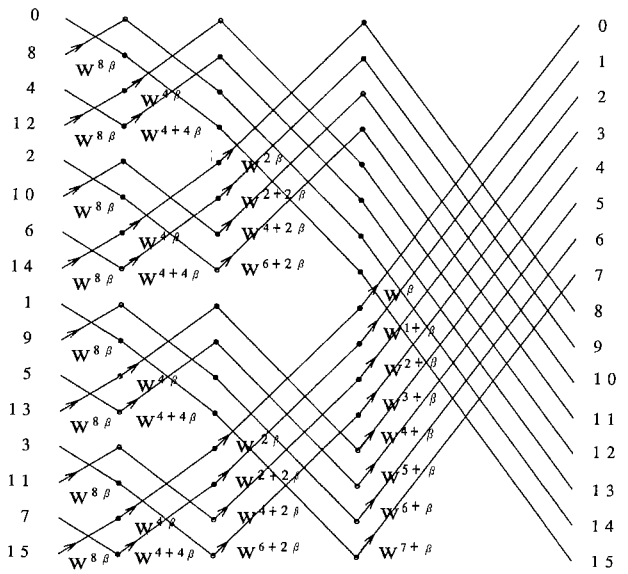


Fig. 1 Signal flow graph of the radix-2 DIT GFFT algorithm.

$$X(k+M/2) = \sum_{n=0}^{M/2-1} x(2n) W_{M/2}^{n(k+\beta)} - W_M^{k+\beta} \sum_{n=0}^{M/2-1} x(2n+1) W_{M/2}^{n(k+\beta)} \tag{53}$$

This is the ordinary DIT radix-2 algorithm, the only difference is the exponent of the twiddle factor. Each butterfly requires 1 complex multiplication and 2 complex additions. The complexity of these GFFTs is exactly

$$\tfrac{N}{2} \log_2 N \text{complex multiplications} \tag{54}$$

$$N \log_2 N \text{complex additions} \tag{55}$$

It is reminded that one complex multiplication may be implemented with 4 real multiplications and 2 additions or with 3 real multiplications and 3 real additions.

The SFG in Fig. 1 can be programmed very simply.

6.3 Split-Radix GFFT

The split-radix DIF and DIT algorithms for the GDFT can be derived similarly. The basic idea is to decompose the original GDFT into three GDFTs-one on the even-indexed samples and two on the odd-indexed samples [14]. The DIT algorithm will be

$$X(k) = \sum_{n=0}^{M/2-1} x(2n) W_{M/2}^{n(k+\beta)} + W_M^{k+\beta} \sum_{n=0}^{M/4-1} x(4n+1) W_{M/4}^{n(k+\beta)} + W_M^{3(k+\beta)} \sum_{n=0}^{M/4-1} x(4n+3) W_{M/4}^{n(k+\beta)} \tag{56}$$

$$X(k+\frac{M}{4}) = \sum_{n=0}^{M/2-1} x(2n)(-1)^n W_{M/2}^{n(k+\beta)} - jW_M^{k+\beta} \sum_{n=0}^{M/4-1} x(4n+1)W_{M/4}^{n(k+\beta)} + jW_M^{3(k+\beta)} \sum_{n=0}^{M/4-1} x(4n+3)W_{M/4}^{n(k+\beta)} \quad (57)$$

$$X(k+\frac{M}{2}) = \sum_{n=0}^{M/2-1} x(2n)W_{M/2}^{n(k+\beta)} - W_M^{k+\beta} \sum_{n=0}^{M/4-1} x(4n+1)W_{M/4}^{n(k+\beta)} - W_M^{3(k+\beta)} \sum_{n=0}^{M/4-1} x(4n+3)W_{M/4}^{n(k+\beta)} \quad (58)$$

$$X(k+\frac{3M}{4}) = \sum_{n=0}^{M/2-1} x(2n)(-1)^n W_{M/2}^{n(k+\beta)} + jW_M^{k+\beta} \sum_{n=0}^{M/4-1} x(4n+1)W_{M/4}^{n(k+\beta)} - jW_M^{3(k+\beta)} \sum_{n=0}^{M/4-1} x(4n+3)W_{M/4}^{n(k+\beta)} \quad (59)$$

Once again we see that only the exponents of the twiddle factors are different from the ordinary split-radix algorithm.

The computational complexity of these algorithms is roughly the same as the complexities of the standard FFTs. Note that in the standard FFT some twiddle factors are trivial and can be programmed separately. This step would decrease the computations at the price of increasing the number of types of butterflies, and the length and complexity of the program. In the generalized DFT we have no longer trivial twiddle factors and such technique is not possible. We remind that frequently repeated estimates for the complexity of the FFT are correct if all trivial twiddle factors are not counted.

Other algorithms seem also interesting. For example the multiplications by the twiddle factors correspond to vector rotations and can be implemented using the CORDIC algorithm.

7. Comparison

Now we have introduced the GFFT algorithms and let us make a comparison with other algorithms for calculating part of the FFT spectrum. Since the partial input and output FFTs are dual problems they have the same complexity. Radix-2-and split-radix-based implementations must be compared separately in order to avoid any confusion. (For this reason we are not satisfied with the comparisons in Ref. [9].) Three approaches will be compared: the efficient pruning scheme of Skinner [7], the transform-based approach [9] and the new approach. As a quantitative measure we shall use the number of complex multiplications and additions and as a qualitative measure-the number of butterfly types.

First let us compare the approaches when radix-2 algorithms are used. The 1-butterfly radix-2 FFT is of the same complexity as the GFFT:

$$N_{MUL} = n2^n - 1 \quad (60)$$

$$N_{ADD} = n2^n \quad (61)$$

If $N-1$ of the butterflies are programmed separately the complexity can be reduced. The 2-butterfly radix-2 FFT requires

$$N_{MUL} = (n-2)2^{n-1} + 1 \quad (62)$$

$$N_{ADD} = n2^n. \quad (63)$$

There are also 3-butterfly implementations, which introduce another type of butterfly to eliminate all redundancies. This does change our conclusions and they shall not be used in our comparison.

Our problem is given the $N = 2^n$-point DFT compare the complexity of evaluating only $M = 2^m$ spectral components. Introducing 1 pruned butterfly into the 1-butterfly radix-2 algorithm (and making the total number of butterfly types 2) as is the Skinner's algorithm the computational complexity is reduced to

$$N_{MUL} = m2^{n-1} \quad (64)$$

$$N_{ADD} = m2^n + 2^n - 2^m \quad (65)$$

The 3-butterfly Skinner pruning will require the same number of complex additions as (65), but the multiplications will be reduced to

$$N_{MUL} = m2^{n-1} - 2^m + 1 \quad (66)$$

The transform-based approach [9] consists of $Q = N/M$ FFTs, followed by $(M-1)(Q-1)$ multiplications and $M(Q-1)$ additions. If the FFTs are performed according to the 1-butterfly radix-2 algorithm, then the total complexity becomes

$$N_{MUL} = m2^{n-1} + 2^n - 2^{n-m} - 2^m + 1 \quad (67)$$

$$N_{ADD} = m2^n + 2^n - 2^m. \quad (68)$$

Increasing the number of butterfly types to 2 will leave the number of additions unchanged, but will decrease the number of multiplications to

$$N_{MUL} = (m-2)2^{n-1} + 2^n - 2^m + 1. \quad (69)$$

Our approach requires

$$N_{MUL} = m2^{n-1} \quad (70)$$

$$N_{ADD} = m2^n. \quad (71)$$

It is to be noted that the transform-based approach requires a post-processing block, which increases the complexity of the implementation. Therefore we are careful to measure the number of arithmetic operations, considering also the number of types of butterflies and the simplicity of the program. Assuming equal number of butterfly types the new approach is the most efficient. It is noted that the pruning approach is the most restrictive, because it requires both N and M to be powers of two, while the transform-based approach [9] and the new approach relax the constraint on N.

If the split-radix algorithm is used the conclusions are the same, namely if only one type of butterfly is used the new approach is the most efficient.

The comparison demonstrates that the three approaches achieve approximately equal performance by three different ways.

If we have a sequence of N samples the complexity of the FFT is $O(N \log_2 N)$. The expression $O(N \log_2 N)$ reflects the fact that we have $\log_2 N$ stages and each stage is of complexity $O(N)$. If only M samples are non-zero the complexity can be reduced.

The pruning approach decreases the number of stages from $\log_2 N$ to $\log_2 M$ and each stage is of complexity again $O(N)$. The method [9] consists of N/M FFTs each of complexity $O(M \log_2 M)$ plus additional stage of complexity $O(N)$. The new technique consists of performing N/M FFTs each with complexity $O(M \log_2 M)$. In all cases the overall complexity is $O(N \log_2 M)$.

We compare the execution time for these three approaches in Table 1 when a subset of the output points are computed.

When very small number of output points are computed ($M = 4$ in Table 1) we see that the method of Sorensen *et. al.* is the fastest, because in this case the majority of butterflies are trivial, and it pays off to remove the trivial multiplications. However, for all of the other cases the the described here technique is the fastest, offering improvement of between 10% and 25%. This confirms that not only the number of arithmetic operations is important, but the structure and simplicity of the program. Pruning requires many `if` statements, which is slow, and as we have demonstrated-inefficient.

Table 1 Comparison of the execution time (in ms) on a typical computer. It is assumed that radix-2 algorithms are used.

N	M	Pruning (Skinner)	Transform-based (Sorensen et al.)	New
256	4	4.55	3.433	3.517
	8	4.95	4.250	3.983
	16	5.483	5.116	4.400
	32	6.066	5.933	4.900
	64	6.75	6.700	5.633
	128	7.45	7.333	6.733
512	4	9.816	6.866	7.083
	8	10.466	8. 516	7.950
	16	11.350	10.250	8.700
	32	12.350	11.850	9.600
	64	13.583	13.416	10.666
	128	14.899	14.849	12.166
	256	16.266	16.199	14.666

8. Partial Input and Output Multidimensional Transforms

In this section we present corresponding algorithms for the multidimensional (N–D) case. Such techniques are more important in the N–D case than in the 1–D case, since the amount of computation increases significantly and any savings become all the more important. In what follows we shall pursue a general approach. Suppose $x(\boldsymbol{n})$ is a signal defined on arbitrary sampling lattice with a periodicity matrix $\boldsymbol{N}$ [16]. Then the DFT of $x(\boldsymbol{n})$ is defined as

$$X(\boldsymbol{k}) = \sum_{\boldsymbol{n} \in \boldsymbol{I}_N} x(\boldsymbol{n}) e^{-j2\pi \boldsymbol{k}^T \boldsymbol{N}^{-1} \boldsymbol{n}}, \quad \boldsymbol{k} \in \boldsymbol{J}_N \tag{72}$$

where $\boldsymbol{I}_N$ and $\boldsymbol{J}_N$ are sets of $|det \boldsymbol{N}|$ samples containing one period of the sequences $x(\boldsymbol{n})$ and $X(\boldsymbol{k})$.

Let us assume that we are interested in the FFT spectrum for only part of the possible values of $\boldsymbol{k}$. This problem arizes when performing decimation of discrete-time sequences. So we want to compute (72) for $\boldsymbol{k} \in \boldsymbol{J}_M$, where we assume that $\boldsymbol{N} = \boldsymbol{QM}$. Similarly to the 1–D case we use the multidimensional index mapping

$$\boldsymbol{n} = \boldsymbol{n}_2 + \boldsymbol{Q}\boldsymbol{n}_1, \quad \boldsymbol{n}_1 \in \boldsymbol{I}_M \text{ and } \boldsymbol{n}_2 \in \boldsymbol{I}_Q \tag{73}$$

to obtain

$$\begin{aligned} X(\boldsymbol{k}) &= \sum_{\boldsymbol{n}_2 \in \boldsymbol{I}_Q} \sum_{\boldsymbol{n}_1 \in \boldsymbol{I}_M} x(\boldsymbol{n}_2 + \boldsymbol{Q}\boldsymbol{n}_1) \\ &\quad e^{-j2\pi \boldsymbol{k}^T \boldsymbol{N}^{-1} (\boldsymbol{n}_2 + \boldsymbol{Q}\boldsymbol{n}_1)} \qquad (74) \\ &= \sum_{\boldsymbol{n}_2 \in \boldsymbol{I}_Q} \sum_{\boldsymbol{n}_1 \in \boldsymbol{I}_M} x(\boldsymbol{n}_2 + \boldsymbol{Q}\boldsymbol{n}_1) \\ &\quad e^{-j2\pi \boldsymbol{k}^T \boldsymbol{M}^{-1} (\boldsymbol{Q}^{-1}\boldsymbol{n}_2 + \boldsymbol{n}_1)} \qquad (75) \\ &= \sum_{\boldsymbol{n}_2 \in \boldsymbol{I}_Q} \sum_{\boldsymbol{n}_1 \in \boldsymbol{I}_M} x(\boldsymbol{n}_2 + \boldsymbol{Q}\boldsymbol{n}_1) \\ &\quad e^{-j2\pi \boldsymbol{k}^T \boldsymbol{M}^{-1} (\boldsymbol{\alpha} + \boldsymbol{n}_1)}, \qquad (76) \end{aligned}$$

where $\boldsymbol{k} \in \boldsymbol{J}_M$ and obviously $\boldsymbol{\alpha} = \boldsymbol{Q}^{-1}\boldsymbol{n}_2$. The original DFT is reduced to several smaller generalized DFTs.

Now let us investigate the dual problem of computing the complete spectrum of sequence, that has many trailing zeroes. So, let us assume that $x(\boldsymbol{n}) \neq 0$ only for $\boldsymbol{n} \in \boldsymbol{I}_M$ and

$$\boldsymbol{N} = \boldsymbol{M}\boldsymbol{Q}^T. \tag{77}$$

Using the multidimensional mapping

$$\boldsymbol{k} = \boldsymbol{k}_2 + \boldsymbol{Q}\boldsymbol{k}_1, \boldsymbol{k}_1 \in \boldsymbol{J}_M, \boldsymbol{k}_2 \in \boldsymbol{J}_Q, \tag{78}$$

we obtain

$$X(\boldsymbol{k}_2 + \boldsymbol{Q}\boldsymbol{k}_1) = \sum_{\boldsymbol{n} \in \boldsymbol{I}_M} x(\boldsymbol{n}) e^{-j2\pi(\boldsymbol{k}_2 + \boldsymbol{Q}\boldsymbol{k}_1)^T (\boldsymbol{M}\boldsymbol{Q}^T)^{-1}\boldsymbol{n}} \tag{79}$$

$$= \sum_{\boldsymbol{n} \in \boldsymbol{I}_M} x(\boldsymbol{n}) e^{-j2\pi(\boldsymbol{k}_2^T \boldsymbol{Q}^{-T} + \boldsymbol{k}_1^T)\boldsymbol{M}^{-1}\boldsymbol{n}} \tag{80}$$

$$= \sum_{\boldsymbol{n} \in \boldsymbol{I}_M} x(\boldsymbol{n}) e^{-j2\pi(\boldsymbol{\beta} + \boldsymbol{k}_1)^T \boldsymbol{M}^{-1}\boldsymbol{n}}, \tag{81}$$

where $\boldsymbol{\beta} = \boldsymbol{Q}^{-1}\boldsymbol{k}_2$. Again this is a GDFT, that can be evaluated using fast algorithms. These fast algorithms require the matrix $\boldsymbol{M}$ to be factorable.

9. Conclusions

In the paper we considered partial input or partial output FFT transformation. The need to perform such computations arizes in several cases, such as interpolation, decimation and linear convolution. Since the FFT has extremely wide application perhaps these generalized transforms can be used in other cases as well. The properties of the GDFT were studied and efficient algorithms were derived. Comparison was made with previous techniques. The approach presented here achieves the shortest execution time among the other known techniques.

Acknowledgment

The authors would like to thank Mr. Komori, a graduate student at Tokyo Institute of Technology for his help in conducting the comparison of the methods.

References

[1] Bellanger, M., *Digital processing of signals,* John Wiley, New York, 1989.

[2] Oppenheim, A.V. and Schafer, R., *Digital signal processing,* Prentice-Hall, Englewood Cliffs, NJ, 1975.

[3] Schafer, R.W. and Rabiner, L.R., "A digital signal processing approach to interpolation," *Proc. IEEE,* vol.61, no.6, pp.692–702, Jun. 1973.

[4] Liu, B. and Mintzer, F., "Calculation of narrow-band spectra by direct decimation," *IEEE Trans. ASSP,* vol.26, no.6, pp.529–534, Dec. 1978.

[5] Cooley, J.W. and Winograd, S., "A limited range discrete Fourier transform algorithm," in *Proc. IEEE Int'l Conf. Acoust, Speech and Signal Process.,* pp.213–217, 1980.

[6] Markel, J.D., "FFT pruning," *IEEE Trans. Audio Electr.,* vol.AU-19, pp.305–311, 1971.

[7] Skinner, D., "Pruning the decimation-in-time FFT algorithm," *IEEE Trans. Acoust. Speech and Signal Process.,* vol.24, no.2, pp.193–194, Apr. 1976.

[8] Lim, Y.-C., "An interpolation technique for computing the DFT of a sparse sequence," *IEEE Trans. Acoust, Speech and Signal Process.,* vol.33, no.11, pp.1456–1460, Nov. 1985.

[9] Sorensen, H.V. and Burrus, C.S., "Efficient computation of the DFT with only a subset of input or output points," *IEEE Trans. Signal Process.,* vol.41, no.3, pp.1184–1200, Mar. 1993.

[10] Bailey, D.H. and Swarztrauber, P.N., "The fractional Fourier transform and applications," *SIAM Review,* vol.33, no.3, pp.389–404, Sep. 1991.

[11] Roche, C., "A split-radix partial input/output fast Fourier transform algorithm," *IEEE Trans. Signal Process.,* vol.40, no.5, pp.1273–1276, May 1992.

[12] Cooklev, T. and Nishihara, A., "Pruning the real-valued FFT algorithms for efficient interpolation and linear convolution," *IEICE Tech. Rep.,* May 1993.

[13] Bongiovanni, G., Corsini, P. and Frosini, G., "One-dimensional and two-dimensional generalized discrete Fourier transform," *IEEE Trans. Acoust. Speech and Signal Process.,* vol.24, no.1, pp.97–99, Feb. 1976.

[14] Duhamel, P. and Hollman, H., "Split-radix FFT algorithm," *Electron. Lett.,* vol.20, no.1, pp.14–16, Jan. 1984.

[15] Adams, J.W., "A subsequence approach to interpolation using the FFT," *IEEE Trans. Circuits Syst.,* vol.34, no.5, pp.568–570, May 1987.

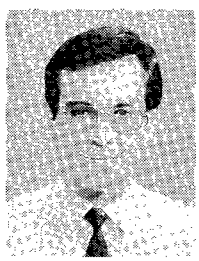
[16] Dudgeon, D. and Mersereau, R., *Multidimensional digital signal processing,* Prentice-Hall, Englewood Cliffs, NJ, 1984.

[17] Mitchell, R.L., "Prefolding and zero-fill in FFT processing," *IEEE Trans. Aerospace and Elec. Systems,* vol.25, no.4, pp.580–581, Jul. 1989.

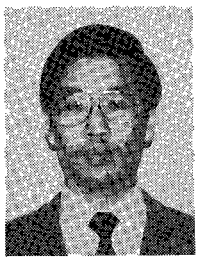
[18] Hamming, R.W., *Digital filters,* Prentice-Hall, Englewood Cliffs, NJ, 1977.

[19] Cooklev, T. and Nishihara, A., "Partial FFT," *Proc. 8th DSP Simposium,* Sendai, pp.35–42, 1993.

[20] Lanari, R. and Hirosawa, H., "Synthetic aperture rader data processing using non-standard FFT algorithm: JERS-1, a case study," *IEICE Trans. Communications,* vol.E76-B, no.10, pp.1271–1278, Oct. 1993.



Todor Cooklev was born in Plovdiv, Bulgaria on August 21, 1966. He received the Diploma degree in electrical engineering from the Technical University of Sofia in 1988. He has been on the faculty at the same University for one academic year. Currently he is with Tokyo Institute of Technology, where he is studying towards the Ph.D. degree as a Monbusho Scholar. His research interests are in 1D and multi-D digital filter banks and wavelets, filter design and implementation, fast algorithms, and the application of algebra in digital signal processing. He has contributed to more than 30 publications on these topics. Mr. Cooklev is a student member of IEEE.



Akinori Nishihara was born in Fukuoka, Japan on February 26, 1951. He received the B.E., M.E. and Dr. Eng. degrees in electronics from Tokyo Institute of Technology in 1973, 1975 and 1978, respectively. Since 1978 he has been with Tokyo Institute of Technology, where he is now Associate Professor of Department of Physical Electronics, Faculty of Engineering. His main research interests are in filter design, 1D and multi-D signal processing, and modern applications of classical circuit theory. From 1990 to 1994 he served as an Associate Editor of the IEICE Trans. Fundamentals. Dr. Nishihara is a member of IEEE, EURASIP and ECS.