Applications of Algebraic Numbers to Computation of Convolutions and DFTs with Few Multiplications

A. N. VENETSANOPOULOS

Department of Electrical Engineering, University of Toronto, Toronto, Canada

E. DUBOIS

INRS-Telecommunications, University of Quebec, Nun's Island, Canada

1. Introduction

In many special purpose digital signal processing applications, the cost of performing multiplications is significantly greater than that of additions. In such situations, algorithms which reduce the number of multiplications, perhaps with some increase in additions, may be attractive. Winograd[1] has shown that the number of multiplications required to perform certain polynomial multiplications related to computation of DFTs and convolutions depends on the underlying field of constants. Thus, by working in certain algebraic number fields, algorithms with a reduced number of multiplications may be devised, although a certain overhead may be incurred when transforming data into the new representation. In this paper, we describe the application of several rings and fields containing a cube root of unity to computation of DFTs and convolutions. In particular, we consider the applications to a radix-3 FFT which has no multiplications in the 3-point DFTs, and to number theoretic transforms.

2. Arithmetic with a cube root of unity

Let R be a commutative ring with identity and consider the extension ring

$$R(\theta)=\{a+b\theta : a,\ b\in R,\ \theta^2+\theta+1=0\}$$

θ is a cube root of unity, since $\theta^2=-\theta-1$ implies $\theta^3=-\theta^2-\theta=1$.

Operations in $R(\theta)$ are as follows:

$$(a+b\theta)+(c+d\theta)=(a+c)+(b+d)\theta$$

$$(a+b\theta)(c+d\theta)=(ac-bd)+[ad+b(c-d)]\theta$$

Multiplication requires three additions and four multiplications in R. It is possible to trade additions for multiplications by using the formula

$$(a+b\theta)(c+d\theta)=(ac-bd)+[(a+b)(c+d)-ac-2bd]\theta$$

which requires six additions and three multiplications in R. This may be useful if a multiplication takes longer than three additions. Also the addition $bd+bd=2bd$ my be implemented as a binary shift if desired.

3. Radix-3 FFT

The DFT of N points is given by

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

where $W=\exp(-j2\pi/N)$ and $X(k)$ and $x(n)$ are sequences of complex numbers. Assume that $N=3^M$. A decimation-in-time implementation of equation (1) for $N=27$ is shown in Fig. 1, using the notation described in ref. 2. The 3-point DFTs of Fig. 1 can be computed using four real multiplications and twelve real additions.[3]

If equation (1) is implemented in $\mathbb{R}(\theta)$, where $\mathbb{R}$ denotes the field of real numbers and

$$\theta=\exp(-j2\pi/3)=-\frac{1}{2}-\frac{\sqrt{3}}{2}j$$

then

$$W=\left[\cos\left(\frac{2\pi}{N}\right)+\sin\left(\frac{2\pi}{N}\right)/\sqrt{3}\right]+\left[2\ sin\left(\frac{2\pi}{N}\right)/\sqrt{3}\right]\theta$$

The 3-point transforms of Fig. 1 for this number system are shown in detail in Fig. 2. These require no multiplications, as shown explicitly by the following equations:

$$\begin{aligned} X(0)&=[x_1(0)+x_1(1)+x_1(2)]+[x_2(0)+x_2(1)+x_2(2)]\theta \\ &=[x_1(0)-x_2(1)x_1(2)+x_2(2)] \\ &\quad+[x_2(0)+x_1(1)-x_2(1)-x_1(2)]\theta \\ X(2)&=[x_1(0)-x_1(1)+x_2(1)-x_2(2)] \\ &\quad+[x_2(0)-x_1(1)+x_1(2)-x_2(2)]\theta \end{aligned} \tag{2}$$

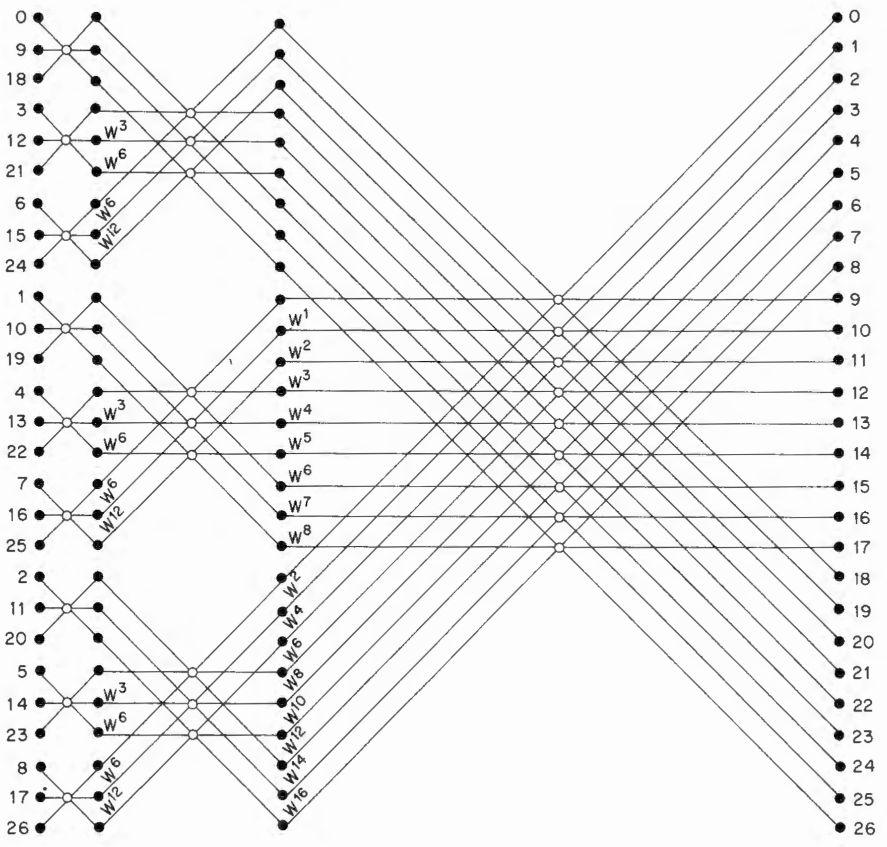

Fig. 1. A radix-3 in-place decimation-in-time FFT algorithm.

where $x(n)=x_1(n)+x_2(n)\theta$. Equation (2) can be evaluated with fourteen real additions. The only multiplications in the transform are those by the "twiddle factors" W^i which are shown in Fig. 1.

To obtain the DFT of a sequence of complex numbers using the above technique, the following equations must be used to transform input and output data between $\mathbb{C}$ and $\mathbb{R}(\theta)$.

$$a+bj \rightarrow \left(a-\frac{b}{\sqrt{3}}\right)-\frac{2b}{\sqrt{3}}\theta, \qquad a+b\theta \rightarrow \left(a-\frac{b}{2}\right)-\frac{\sqrt{3}b}{2}j$$

If either input or output are known to be real, the appropriate transformation can be waived. Although we have described the algorithm with

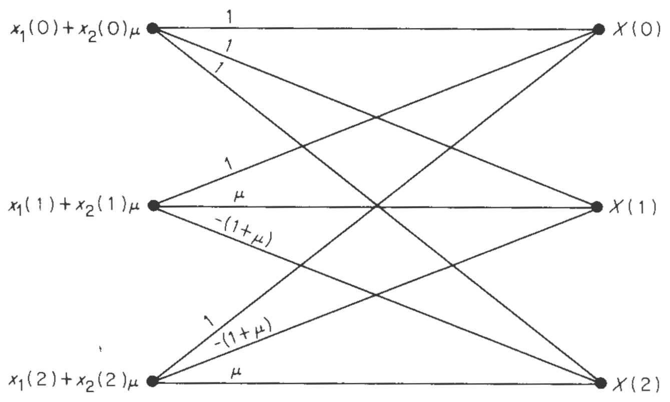

Fig. 2. Three-point DFT in $\mathbb{R}(\theta)$.

reference to the decimation-in-time implementation of Fig. 1, the technique applies to all the usual configurations a radix-3 FFT can take.

If the data are real, the above algorithm can be used to compute the DFT of sequences of length 2×3^M. This is accomplished by computing the transform of the length 3^M sequence $y(n)=x(2n)+x(2n+1)\theta$, and manipulating it in a fashion analogous to that for the complex-valued DFT.[4] If the transform is to be used to convolve real sequences, no data transformations are required. To convolve $x(n)$ and $h(n)$, the transforms $X(k)$ and $H(k)$ are computed in $\mathbb{R}(\theta)$, multiplied pointwise in $\mathbb{R}(\theta)$, and then inverse transformed. A technique to convolve length 2×3^M real sequences is given in ref. 5.

In an $N=3^M$ point radix-3 FFT, there are $(\frac{2}{3}M-1)N+1$ non-trivial complex multiplications by twiddle factors and $MN/3$ three point DFTs. Use of the above algorithm eliminates the $4MN/3$ real multiplications associated with these 3-point DFTs, while leaving the number of real multiplications associated with the twiddle factors fixed. In the limit for large N, the number of real multiplications is reduced by a factor of $33\frac{1}{3}\%$, if multiplication algorithms in $\mathbb{C}$ and $\mathbb{R}(\theta)$ with four real multiplications are used, and by a factor of 40%, if multiplication algorithms in $\mathbb{C}$ and $\mathbb{R}(\theta)$ with three real multiplications are used. However the number of additions is increased and the total computation savings depend on the relative costs of addition and multiplication on the processor being used. If the cost of a multiplication is r times that of an addition, then for large N the relative cost of the arithmetic operations in the new algorithm to those in the standard algorithm is given by

$$\frac{2r+5}{3r+4}$$

if multiplication algorithms in $\mathbb{C}$ and $\mathbb{R}(\theta)$ with four real multiplications are used. This ratio is shown in Fig. 3 as a function of r; it is unity for $r=1$ and decreases monotonically to $\frac{2}{3}$. If the multiplication algorithms in $\mathbb{C}$ and $\mathbb{R}(\theta)$ with three real multiplications are used, the corresponding ratio is

$$\frac{3r+13}{5r+11}$$

which is unity for $r=1$ and decreases monotonically to $\frac{3}{5}$.

Similar comparisons can be made with radix-2 and radix-4 FFTs. Restricting the situation to the case where multiplication in $\mathbb{C}$ and $\mathbb{R}(\theta)$ requires four real multiplications, for large N the relative cost of the arithmetic operations in the new algorithm and in a radix-2 algorithm is given by

$$\log_3 2\frac{8r+20}{6r+9}=\frac{5{\cdot}05r+12{\cdot}62}{6r+9}$$

This ratio is 1·18 for $r=1$, becomes equal to unity for $r=3{\cdot}81$ and decreases monotonically to 0·84. For radix-4, the ratio is

$$\log_3 4\frac{16r+40}{18r+33}=\frac{20{\cdot}19r+50{\cdot}47}{18r+33}$$

which is 1·39 at $r=1$ and decreases monotonically to 1·12. These two ratios are also shown in Fig. 3. The new algorithm can thus be more efficient than radix-2 but is less efficient than radix-4. It must be understood that these ratios represent relative efficiencies, as a radix-2 and radix-3 algorithm cannot both exist for the same N.

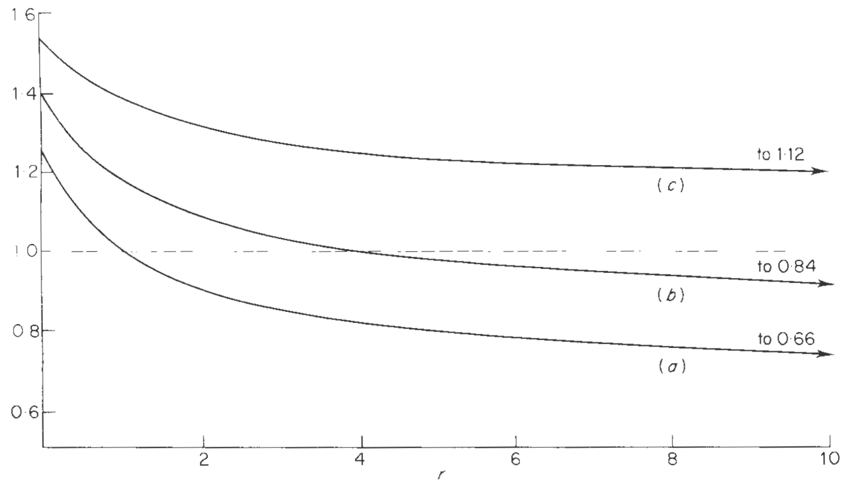

Fig. 3. Relative cost of computation between new radix-3 algorithm and (*a*) standard radix-3, (*b*) radix-2, (*c*) radix-4 FFT algorithms.

To extend this technique to higher radices, new bases $\{(\theta_1, \theta_2)\}$ are needed, for the complex field. The properties desired are: (a) that arithmetic in these bases has simplicity comparable to standard complex arithmetic and (b) that a p-point DFT has a reduced number of multiplications for some p. To date the authors have not discovered any further examples satisfying these two properties. In particular, the basis $(1, \psi)$ where ψ is a primitive pth root of unity does not satisfy them for $p>3$.

4. Application to number theoretic transforms

A number theorètic transform is a transform defined in a ring R of residues of algebraic integers having the DFT structure, which maps cyclic convolution isomorphically into pointwise product:[6,7]

$$U\colon X(k)=\sum_{n=0}^{N-1} x(n)\alpha^{kn}, \qquad k=0, \ldots, N-1$$

and

$$U(x*y)=Ux.Uy$$

where $x*y$ denotes cyclic convolution. A necessary condition for these properties to hold is that α be a primitive Nth root of unity in R.

Generally, the ring R is chosen to be a ring of integers, or complex integers, modulo an integer M. By appropriate choice of M and α, transforms which require no multiplications can be constructed. However, these suffer from the disadvantage that the wordlength required is proportional to the sequence length. Thus techniques which can give greater sequence lengths for given wordlength without introducing multiplications are of interest.

The most well known moduli used with number theoretic transforms are Fermat numbers: $F_t=2^{2^t}+1$. Using $\alpha=2$, transform lengths of 2^{t+1} can be achieved and with $\alpha=\sqrt{2}\triangleq 2^{t-2}(2^{t-1}-1)$ lengths 2^{t+2} are obtained with transforms in Z_{F_t}. By working in the ring $Z_{F_t}(\theta)$, these transform lengths can be increased by a factor of 3,[7] using $\alpha=2\theta$ or $\alpha=\sqrt{2\theta}$ (other values may be more advantageous). Then, for real data, the sequence lengths which can be convolved can be further doubled using techniques described in ref. 8. Arithmetic is carried out as indicated in Section 2.

For illustration, consider $M=2^8+1$ and $N=3\times 2^4$, with $\alpha=2^{11}\theta$. (2^{11} has order 16, and has the property $(2^{11})^3=2^{33}=2$.) The transform can be decomposed as three 16-point transforms with $\alpha=(2^{11}\theta)^3=2$, followed by sixteen 3-point transforms with $\alpha=(2^{11}\theta)^{16}=\theta$. Since 3 and 16 are relatively prime, the prime factor algorithm can be used, obviating the need for intermediate multiplications by twiddle factors.

The 3-point transforms can be implemented exactly as in Fig. 2, requiring fourteen additions, as given by equation (2). The flowgraph for the entire transform is shown in Fig. 4, where the 16-point transforms can be implemented using any standard radix-2 algorithm.

Useful results can also be obtained by taking R to be the ring of Gaussian integers modulo M, $Z_M[i]$. In this case, transforms in $Z_M[i](\theta)$ can be used to convolve sequences of complex numbers, giving an increase in sequence length by a factor of six over comparable algorithms in $Z_M[i]$. Thus for example, if a transform of length N having no multiplications exists in $Z_M[i]$ with $\alpha = 1+i$, then one of length $3N$ exists in $Z_M[i](\theta)$ with $\alpha =$

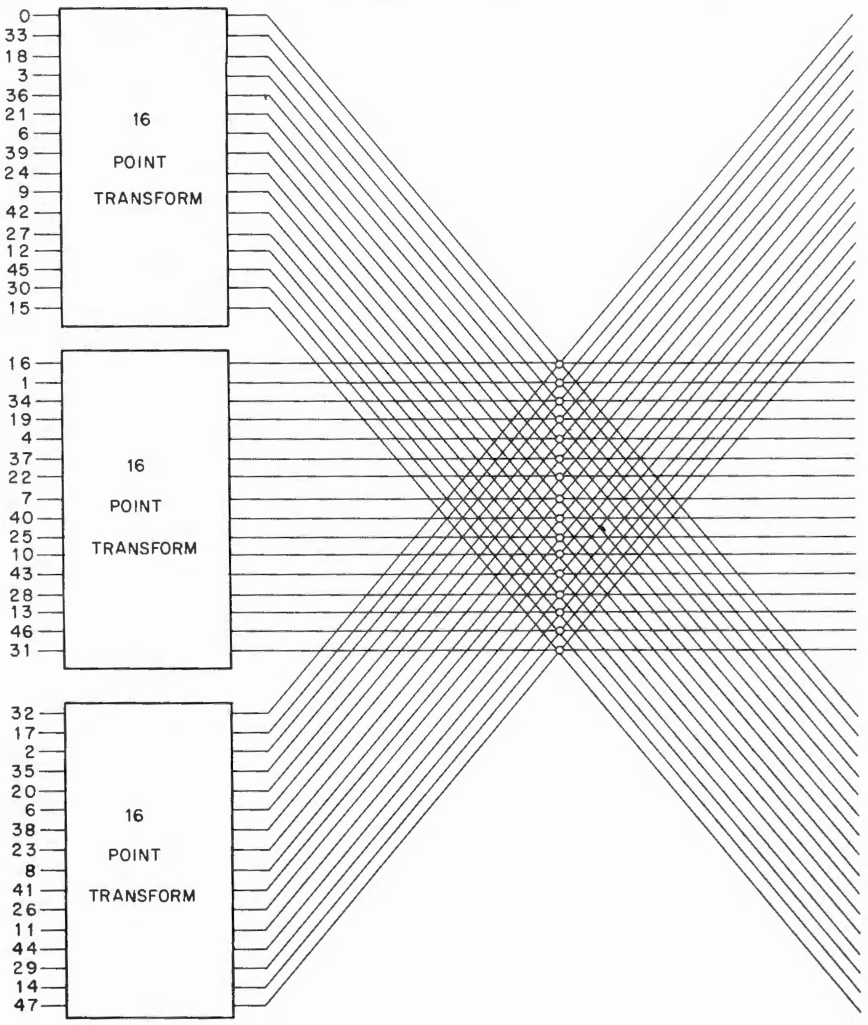

Fig. 4. Forty-eight-point DFT in $Z[\theta](F_t)$.

$(1+i)\theta$, and the techniques of ref. 8 can be used to convolve sequences of length $6N$. These transforms are most useful with the pseudo-Mersenne and pseudo-Fermat transforms.[9,10]

5. Conclusion

Techniques for computation of convolutions and DFTs using arithmetic in rings and fields containing a cube root of unity have been described. An algorithm for computing the radix-3 FFT using arithmetic in $\mathbb{R}(\theta)$ uses $33\frac{1}{3}\%$ or 40% fewer multiplications than the conventional radix-3 algorithm. Also, number theoretic transforms exhibiting increased sequence lengths can be obtained using rings of modular integers containing a cube root of unity. These results show that use of number systems other than the real and complex fields and their integers can lead to computational savings in digital signal processing applications.

Acknowledgment

This work was supported in part by the National Research Council of Canada under Grant No. A-7397.

References

1. S. Winograd, "The effect of the field of constants on the number of multiplications", *Proc. 16th Annual Symposium on Foundations of Computer Science*, pp. 1–2 (1975).
2. L. R. Rabiner and B. Gold, "Theory and Application of Digital Signal Processing", pp. 435–437. Englewood Cliffs, N.J.: Prentice-Hall, 1975.
3. S. Winograd, "On computing the discrete Fourier transform", *Proc. of the National Academy of Sciences, USA*, **73**, pp. 1005–1006 (April 1976).
4. J. W. Cooley, P. A. W. Lewis, and P. D. Welch, "The fast Fourier transform algorithm: Programming considerations in the calculation of sine, cosine and Laplace transforms", *J. Sound Vib.*, **12**, pp. 315–337 (July 1970).
5. E. Dubois and A. N. Venetsanopoulos, "A new algorithm for the radix-3 FFT", *IEEE Trans. on Acoustics, Speech and Signal Processing*, **ASSP-26** (June 1978).
6. R. C. Agarwal and C. S. Burrus, "Number theoretic transforms to implement fast digital convolution", *Proc. IEEE*, **63**, pp. 550–560 (April 1975).
7. E. Dubois and A. N. Venetsanopoulos, "The discrete Fourier transform over finite rings with application to fast convolution", *IEEE Trans. on Computers*, **C-27** (July 1978).
8. E. Dubois and A. N. Venetsanopoulos, "Convolution using a conjugate symmetry property for the generalized discrete Fourier transform", *IEEE Trans. on Acoustics, Speech, and Signal Processing*, **ASSP-26**, pp. 165–170 (April 1978).
9. H. J. Nussbaumer, "Digital filtering using complex Mersenne transforms", *IBM J. Res. Dev.*, **20**, pp. 498–504 (Sept. 1976).
10. H. J. Nussbaumer, "Digital filtering using pseudo Fermat number transforms", *IEEE Trans. on Acoustics, Speech, and Signal Processing*, **ASSP-26**, pp. 79–83 (Feb. 1977).