Carnegie Mellon DSP-I Electrical & Computer ENGINEERING
Digital Signal Processing I (18-791)
Fall Semester, 2005

DECIMATION IN TIME FOR VALUES OF *N* THAT ARE NOT A POWER OF 2

Issued: 10/27/05

Note: This handout is a copy of a section of the original text by Oppenheim and Schafer (1975) which describes the development of decimation-in-time algorithms for values of N that are not an integer power of 2.

The section opens with the following text:

6.4 FFT Algorithms for *N* a Composite Number

Our previous discussion has illustrated the basic principles of decimation-in-time and decimation-in-frequency for the important special case of N power of 2; *i.e.* $N=2^{\nu}$. More generally, the efficient computation of the discrete Fouier transform is tied to the representation of N as a product of factors; *i.e.,* suppose that

$$N = p_1 p_2 \ldots p_{\nu}$$

As we have seen in the case of N a power of 2 (where all the factors can be taken to be equal to 2), such as decomposition leads to a highly efficient ….

computational algorithm. Furthermore, all the required computations are butterfly computations that correspond essentially to two-point DFTs. For this reason the power-of-2 algorithms are particularly simple to implement, and often in applications it is advantageous to always deal with sequences whose length is a power of 2. This can be done in many cases by simply augmenting a finite-length sequence with zero samples if necessary. However, in some cases it may not be possible to choose N to be a power of 2, thus making it necessary to consider the more general situation of Eq. (6.25). Let us therefore consider the application of the decimation-in-time principle in the case where N is a product of factors that are not all necessarily equal to 2.

Let us define

$$q_1 = p_2 p_3 \cdots p_\nu$$

so that

$$N = p_1 \cdot q_1$$

If N is a power of 2, we could choose $p_1 = 2$ and $q_1 = N/2$. Using decimation-in-time we would then decompose the sequence $x(n)$ into two sequences, each $(N/2)$ samples in length, consisting of the even- and odd-numbered samples, respectively. When $N = p_1 \cdot q_1$, we can divide the input sequence into p_1 sequences of q_1 samples each by associating every p_1th sample with a given subsequence. For example if $p_1 = 3$ and $q_1 = 4$, so that $N = 12$, then we can decompose $x(n)$ into three sequences of length 4, with the first sequence consisting of the samples $x(0)$, $x(3)$, $x(6)$, $x(9)$; the second sequence consisting of $x(1)$, $x(4)$, $x(7)$, $x(10)$; and the third sequence consisting of $x(2)$,

$x(5)$, $x(8)$, and $x(11)$. In general we can write $X(k)$ as

$$\begin{aligned} X(k) &= \sum_{n=0}^{N-1} x(n)W_N^{kn} \\ &= \sum_{r=0}^{q_1-1} x(p_1r)W_N^{p_1rk} + \sum_{r=0}^{q_1-1} x(p_1r+1)W_N^{k}W_N^{p_1rk} + \cdots \\ &\quad + \sum_{r=0}^{q_1-1} x(p_1r+p_1-1)W_N^{(p_1-1)k}W_N^{p_1rk} \end{aligned}$$

or

$$X(k) = \sum_{l=0}^{p_1-1} W_N^{lk} \sum_{r=0}^{q_1-1} x(p_1r+l)W_N^{p_1rk} \tag{6.26}$$

The inner sums can be expressed as the q_1-point DFTs

$$G_l(k) = \sum_{r=0}^{q_1-1} x(p_1r+l)W_{q_1}^{rk} \tag{6.27}$$

since, as is easily verified,

$$W_N^{p_1rk} = W_{q_1}^{rk} \qquad \text{for} \qquad N = p_1 \cdot q_1 \tag{6.28}$$

Thus Eq. (6.26) expresses $X(k)$ in terms of p_1 discrete Fourier transforms of sequences of length q_1 samples. To determine the number of complex multiplications and additions required in implementing the DFT according to Eq. (6.26), let us consider, as we did in the original discussion of decimation-in-time, that the q_1-point DFTs are implemented by means of the direct computation. From Eq. (6.26) we observe that the number of q_1-point DFTs to be evaluated is p_1. Thus a total of $p_1 \cdot q_1^2$ complex multiplications and additions are required. The outer sum in Eq. (6.26) is implemented by multiplying the q_1-point DFTs by the factor W_N^{lk} and adding the results together. Since the double summation in Eq. (6.26) is to be implemented for N values of k, a total of $N(p_1 - 1)$ complex multiplications and additions are required to combine the $p_1 q_1$-point DFTs.† Therefore, the total number of complex multiplications and additions required to compute the discrete Fourier transform in the form of Eq. (6.26) is $N(p_1 - 1) + p_1q_1^2$. Now the q_1-point DFTs can be decomposed in a similar manner. In particular, if we now represent q_1 as

$$q_1 = p_2 \cdot q_2$$

† Summing p_1 terms requires p_1-1 additions and we do not need to multiply by W_N^{kl} when $l = 0$. We remind the reader that throughout this chapter we have generally counted multiplication by W_N^{kl} even when W_N^{kl} is unity or j. In interpreting Eq. (6.26), however, it is convenient to recognize W_N^{kl} as unity for $l = 0$ in order that the result which we obtain be consistent with the discussion in Section 6.2.

then the q_1-point sequences in the inner sum of Eq. (6.26) can be broken into p_2 subsequences, each being q_2-points long so that the inner sum in Eq. (6.26) can be replaced by a double summation in the same way that we began. When this is done, the number of operations required in computing the q_1-point DFTs in Eq. (6.26) is, instead of q_1^2,

$$q_1(p_2 - 1) + p_2 q_2^2 \tag{6.29}$$

Consequently, the factor q_1^2 in the expression $N(p_1 - 1) + p_1 q_1^2$ is replaced by Eq. (6.29), and thus the total number of complex multiplications and additions required is

$$N(p_1 - 1) + N(p_2 - 1) + p_1 p_2 q_2^2 \tag{6.30}$$

If we continue this procedure by further decomposing the q_2-point DFTs, then when the original sequence has been decomposed as much as possible the number of complex multiplications and additions will be

$$N(p_1 + p_2 + \ldots + p_\nu - \nu) \tag{6.31}$$

For example, when $p_1 = p_2 = \ldots = p_\nu = p$, the number of complex multiplications and additions is $N(p - 1)\nu$. When $p = 2$, this number is $N \cdot \nu$, as discussed before.† In general, it can be seen from Eq. (6.31) that it is preferable to carry out the decomposition on the basis of as many factors as possible for a given N. Formally, there is no advantage to choosing anything but prime factors since if $p_i = r_i \cdot s_i$, with r_i and $s_i > 1$, then $p_i > r_i + s_i$, except when $r_i = s_i = 2$, in which case $p_i = r_i + s_i$. However, there are examples (notably $p_i = 4$ or 8) where additional economies result which are not accounted for by Eq. (6.31).

To illustrate the decimation-in-time procedure for N not a power of 2, let us consider computation of an 18-point DFT; i.e., $N = 3 \cdot 3 \cdot 2$. Letting $p_1 = 3$ and $q_1 = 6$, and following the preceding discussion, we first divide the original sequence into three sequences, each six points long:

Sequence 1:	$x(0)$	$x(3)$	$x(6)$	$x(9)$	$x(12)$	$x(15)$
Sequence 2:	$x(1)$	$x(4)$	$x(7)$	$x(10)$	$x(13)$	$x(16)$
Sequence 3:	$x(2)$	$x(5)$	$x(8)$	$x(11)$	$x(14)$	$x(17)$

By dividing the original sequence into these three subsequences, we can express $X(k)$ as

$$\begin{aligned} X(k) &= \sum_{l=0}^{2} W_{18}^{lk} \sum_{r=0}^{5} x(3r + l) W_{18}^{3rk} \qquad (6.32) \\ &= G_0(k) + W_{18}^{k} G_1(k) + W_{18}^{2k} G_2(k) \end{aligned}$$

† Recall that the number of multiplications can be further reduced by exploiting symmetry.

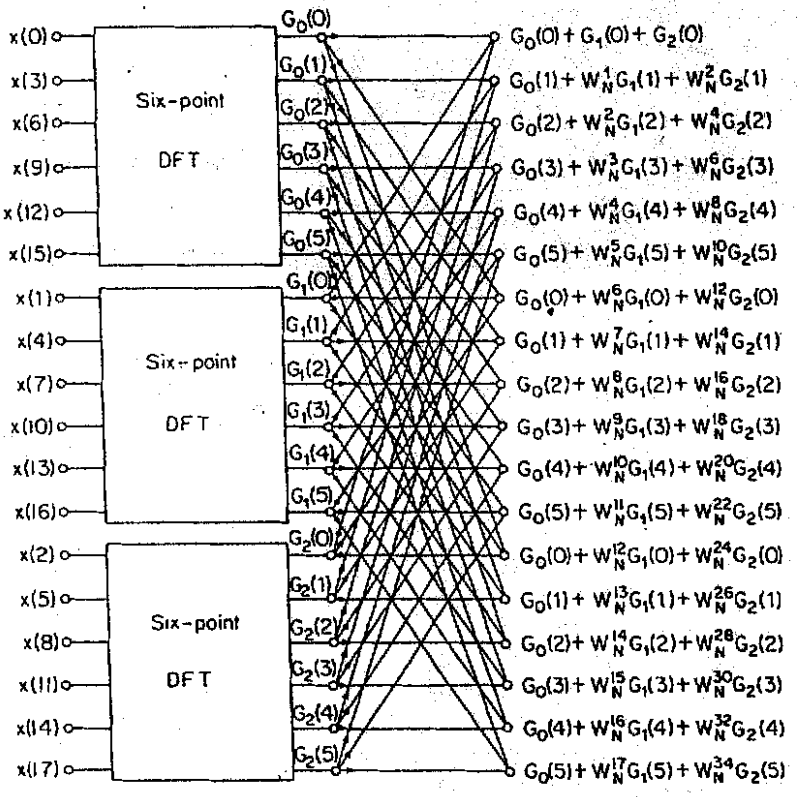

Fig. 6.24 Flow graph of first stage of decomposition of an 18-point DFT.

The inner sum is a six-point DFT with $l = 0$ corresponding to sequence 1, $l = 1$ corresponding to sequence 2, and $l = 2$ corresponding to sequence 3. In this case the six-point DFTs $G_l(k)$ are periodic with period 6. The computation of Eq. (6.32) is illustrated in Fig. 6.24.†

The six-point DFTs corresponding to the inner sum in Eq. (6.32) can be further decomposed by breaking the sequences $x(3r + l)$ into three sequences, each two points long or, alternatively, two sequences, each three points long. Choosing the former, i.e., breaking each of the six-point subsequences into three sequences two points long, we replace the inner sum by

$$G_l(k) = \sum_{r=0}^{5} x(3r + l)W_6^{rk} = \sum_{s=0}^{2} W_6^{sk} \sum_{p=0}^{1} x(9p + 3s + l)W_6^{3pk}$$

† In this figure, the transmittances on each of the branches are to be inferred from the algebraic expression associated with each output node.

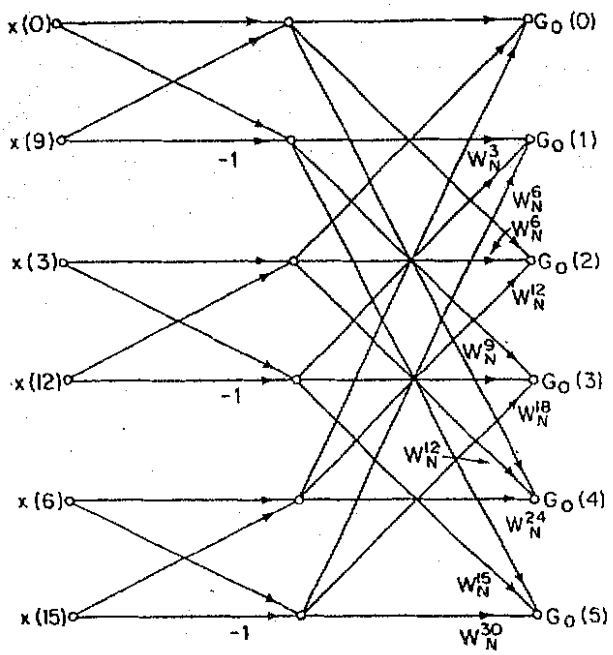

Fig. 6.25 Flow graph of further decomposition of one of the six-point DFTs in Fig. 6.24.

so that the overall computation of $X(k)$ becomes

$$X(k) = \sum_{l=0}^{2} W_{18}^{lk} \sum_{s=0}^{2} W_6^{sk} \sum_{p=0}^{1} x(9p + 3s + l) W_2^{pk} \tag{6.33}$$

One of the six-point DFTs ($G_0(k)$) is shown in detail in Fig. 6.25. The other two have identical form. When Fig. 6.25 and the corresponding flow graphs for $G_1(k)$ and $G_2(k)$ are placed in appropriate positions in Fig. 6.24, the input sequence is in the order: $x(0)$, $x(9)$, $x(3)$, $x(12)$, $x(6)$, $x(15)$, $x(1)$, $x(10)$, $x(4)$, $x(13)$, $x(7)$, $x(16)$, $x(2)$, $x(11)$, $x(5)$, $x(14)$, $x(8)$, and $x(17)$. We observe from Figs. 6.24 and 6.25 that for this ordering of the inputs, the computation can be done in place. The two-point transforms are shown as the familiar butterflies in the first stage of Fig. 6.25; the basic three-point DFT operation is somewhat more complicated but still obviously an in-place computation. Instead of bit reversal, the ordering of the input is somewhat more complicated. Specifically, if we denote $X_0(\,)$ as the input array, then it can be shown that

$$X_0(6l + 2s + p) = x(9p + 3s + l)$$

where $p = 0, 1$; $s = 0, 1, 2$; and $l = 0, 1, 2$. That is, the input must be stored in a generalized "digit-reversed" order in order to carry out the computation in place. As is evident from Fig. 6.24, the resulting output is in normal order. We note from Fig. 6.24 that the basic computation in the last stage (as for factors of 3 in general) is as depicted in Fig. 6.26 for $N = 3 \cdot q_1$.

We recall that in the case of factors of 2, we were able to reduce the number of multiplications by a factor of 2 by exploiting symmetry. In the case of factors

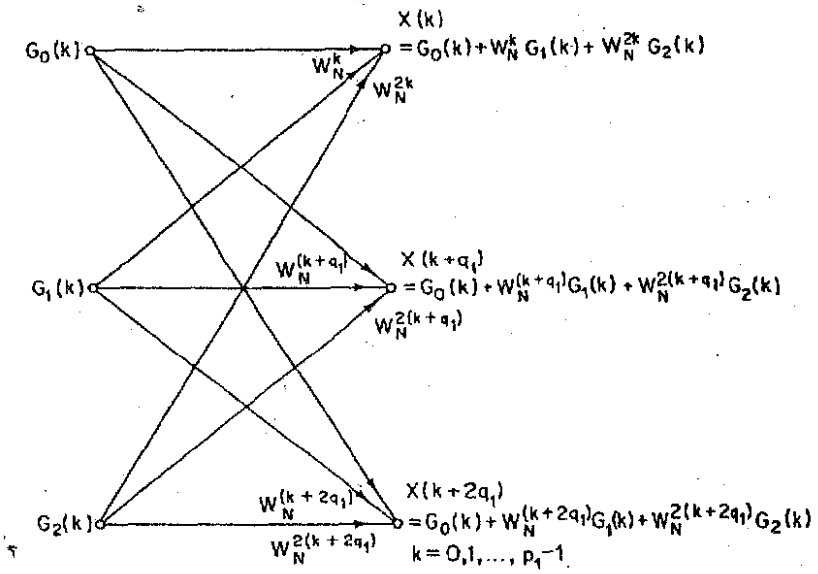

Fig. 6.26 Flow graph of the basic computation for factors of 3.

of 3, in Fig. 6.26, the comparable manipulation of the flow graph yields Fig. 6.27. Since $N = 3q_1$, the basic complex multiplier $W_N^{q_1}$ is

$$W_N^{q_1} = e^{-j(2\pi/3)}$$

Therefore, $W_N^{q_1}$ and all powers thereof are complex coefficients that require multiplications. Thus Fig. 6.27 is no more efficient than Fig. 6.26.

In contrast to factors of 3, it can be shown (see Problem 7 of this chapter) that the basic DFT computation for a factor of 4 (i.e., $N = 4q_1$) is as shown in Fig. 6.28. In this case, the flow graph of Fig. 6.28 can be redrawn as in Fig. 6.29, with a concomitant saving of at least 9 complex multiplications out of the 12 shown in Fig. 6.28. Similar savings result for factors of 8, 16, etc. [11]. Thus, even if $N = 2^\nu$, it is sometimes advantageous to base the computation on factors of 4, using one stage based on a factor of 2 if ν is odd.

Our discussion in this section, although paralleling the earlier discussion, has been far from complete in the sense that we have only attempted to indicate some of the advantages and disadvantages of using values of N with factors other than 2. The basic advantages are increased flexibility and speed in some cases; the basic disadvantage is greatly increased complexity of the computational algorithm. Although we have only discussed decimation-in-time, a similar discussion clearly holds for decimation-in-frequency as well. For a more detailed discussion of FFT algorithms for N a general composite number, see Gentleman and Sande [9] and Singleton [10].

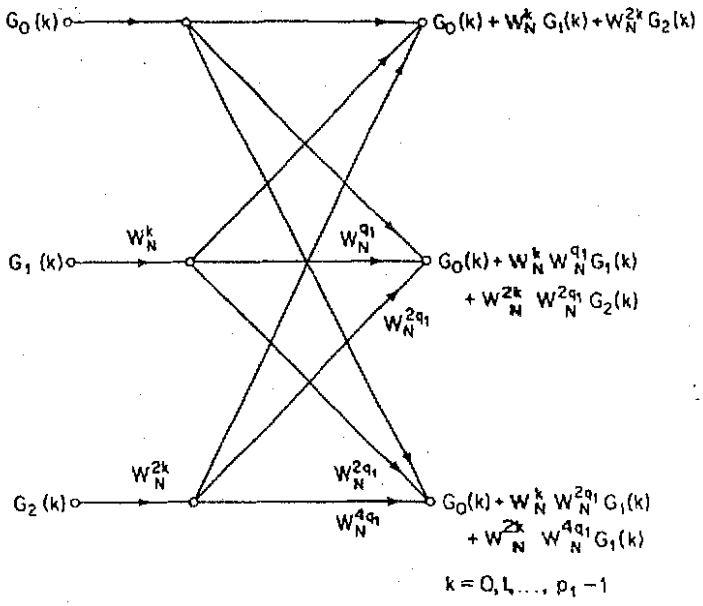

Fig. 6.27 Alternative arrangement of Fig. 6.26.

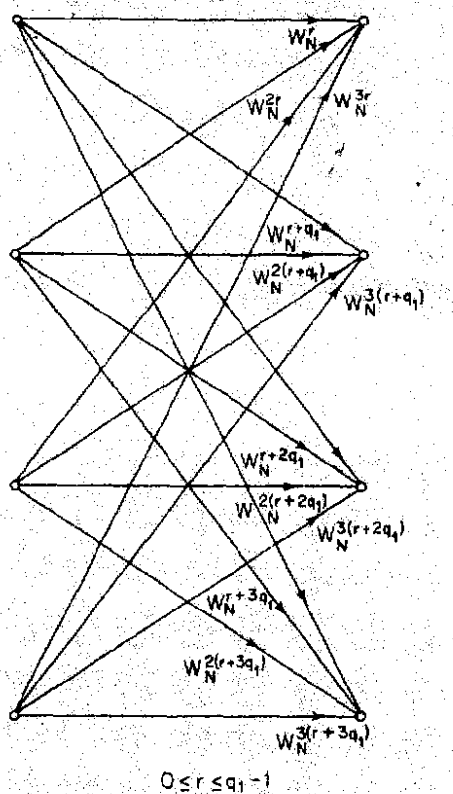

Fig. 6.28 Flow graph of the basic computation for factors of 4.

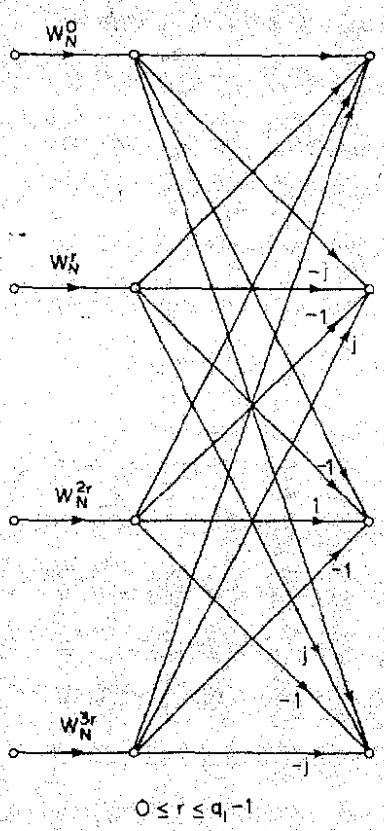

Fig. 6.29 Alternative arrangement of Fig. 6.28 resulting in savings of multiplications.