A Sequency-Ordered Fast Walsh Transform

JOSEPH W. MANZ, Student Member, IEEE

Abstract

A method is described that yields the fast Walsh transform (FWT) in sequency order. The advantages of this method over others are: 1) it is based on the Cooley–Tukey-type fast Hadamard transform (FHT) algorithm, 2) the computational effort is identical to the conventional FHT, and 3) the transform remains its own inverse.

Certain properties of a Hadamard matrix of order $N=2^M$ make it a convenient vehicle for the computation of the fast Walsh transform (FWT) [1], [2]. The rows or columns of the matrix are Walsh functions. Since each element can only be ± 1 and each row or column is orthogonal to every other one, the inverse transform is accomplished by repeated forward transforms due to the symmetry properties of the matrix, and certain special Hadamard matrices of order $N=2^M$ possess the additional property of having their rows or columns defined by a simple recursive formula. This last property lends itself to the computation of a "fast" transform using fast matrix factorization techniques, i.e., Cooley–Tukey factorization. In fact, by replacing the trigonometric multipliers in the standard fast Fourier transform (FFT) routine by a ± 1, the FHT can be generated [2], [3].

The only drawback with the fast Hadamard transform (FHT) is that those matrices that possess a simple recursive formula and, therefore, a fast algorithm, are not capable of directly producing the output coefficients ordered by increasing sequency [4], [5]. Sequency, as defined by Harmuth [6, p. 50], is one-half the average number of zero crossings per unit time interval. The ordering of the output coefficients of a typical FHT is called dyadic or Paley ordering [5].

In order to convert from dyadic to sequency ordering, the output coefficient ordering must be decoded by using a Gray code-to-binary decoder [5], [7]. This, of course, slows down the fast nature of the transform and results in additional computational costs. Fig. 1 shows Walsh functions in both sequency and dyadic ordering.

Fast Algorithm for Sequency Ordering

By suitably modifying the FHT approach, a sequency-ordered FWT can be computed that shares all of the good properties of the FHT but eliminates the Gray code decoding. The modification is best illustrated for the case when $M=3$ and $N=2^M=8$.

Manuscript received January 17, 1972. Support for the use of the computer facilities at the University of Connecticut was provided by National Science Foundation Grant GJ-9.

The author is with the Bio-Engineering Laboratory, University of Connecticut, Storrs, Conn. 06268.

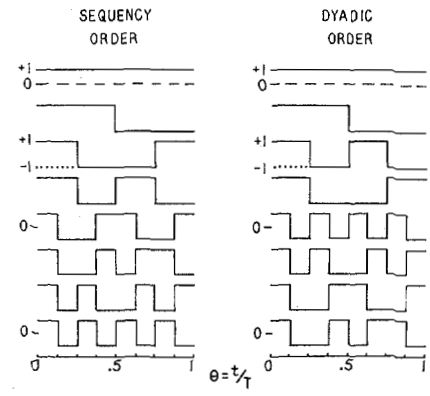

Fig. 1. Walsh functions 0 through 7 ordered dyadically (right) and by increasing sequency (left). $\theta=t/T$ is normalized time.

A property of all FHT techniques based on fast matrix factorization is that coefficients at the output of the transform are in bit-reversed order [3], [4]. A necessary adjustment involves the rereversing of the bits and reordering the coefficients in ascending index order. Fig. 2 shows a general flow graph for the Cooley–Tukey-type FHT.

The first step in the modification scheme is to bit reverse the input data and reorder it in ascending index order. In a standard FHT, one may transform and then bit reverse or bit reverse and then transform, achieving the same end result. In the modified FHT, the input *must* be bit reversed prior to the actual transformation.

The second step is to define a reversal. A reversal involves altering or reversing the action represented by the solid and dotted lines arising from the nodes in Fig. 2. The solid line indicates the transfer of the quantity from the node where the line originates to the node where the line terminates and the multiplication of this quantity by plus one. The dotted line represents a similar transfer, but the multiplying factor is minus one. Fig. 3 illustrates a reversal. In the reversal case, the action may be pictured in two ways: the roles of the solid and dotted lines are interchanged (a solid line transformed into a dotted line and vice versa), or the quantity stored at the node from which the dotted and solid lines arise is multiplied by a minus one (-1) and the action of the lines left unchanged. A reversal can occur only at those nodes which give rise to both a solid and a dotted line.

The third step is to determine which nodes are to be reversed and which are to remain unchanged. The following rules describe the selection procedure (all letters refer to Fig. 2).

Rule 1: Lines originating from the input nodes IN are *never* reversed.

Rule 2: The next set of nodes to the right, column A_1, has $2^0=1$ "blocks" of reversal. A block is used to denote that group of computations (represented by the lines) which are disconnected from its neighbors above or below. For example, in Fig. 2 there is only one block between the input nodes and nodes A_1. There are two blocks between nodes A_1 and A_2, etc. Starting with the $2^0=1$ blocks between nodes IN and A_1, the number of blocks increases to the right as 2^1, 2^2, and so on up until the set of blocks between nodes A_{M-2} and A_{M-1} which number $2^{M-1}=N/2$. In Fig. 2, the blocks are separated by the heavy broken lines.

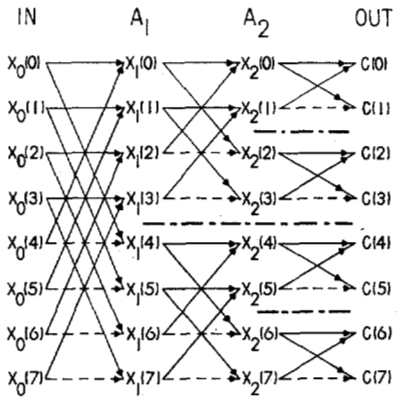

Fig. 2. Signal flow graph of discrete FHT. Multipliers are +1 and −1 as indicated by the solid and dashed lines, respectively. The heavy broken lines separate blocks.

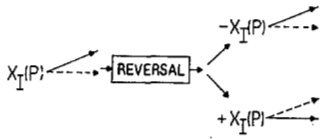

Fig. 3. Illustration of the effect of a reversal on the computational flow of element $X_I(P)$. The solid line indicates multiplication by +1 and the dotted line by −1.

Rule 3: Reversals occur in blocks. At each column of nodes A_I after the input nodes, the bottommost block or the one containing the largest indexed term has a reversal. The number of reversals per block equals the number of nodes per block divided by two. Working up a string of nodes A_I, from $X_I(N-1)$ toward $X_I(0)$ every other block has a reversal. The topmost block or the one containing the term indexed (0), *never* has a reversal.

Fig. 4 shows the resultant flow graph with just the blocks indicated and demonstrating the reversal pattern that is generated. Fig. 5 shows the final flow diagram for $N=2^3=8$ of the FWT that generates the output coefficients in sequency order.

Other Sequency Ordering Techniques

There are basically three other general schemes for calculation of a sequency-ordered Walsh transform. The process of decoding the output of a standard FHT is slower and more costly than the plain FHT, and slower than the modified FHT previously described.

A method described by Harmuth [6, pp. 45–48] necessitates writing a new algorithm since it differs from the familiar FFT algorithm which may already be available to the investigator. Harmuth's method also requires a programming change to compute the inverse transform and defining the function of interest on the interval $[-\frac{1}{2}, \frac{1}{2})$ instead of on the more usual [0, 1) interval.

A third method involves arranging the rows of a Hadamard matrix in the proper manner to insure output sequency ordering and then trying to construct a fast computational scheme for this structure [4]. Since this matrix does not have a very simple recursive formula [7], this method has not been notably successful as yet.

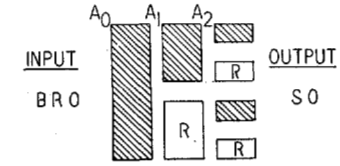

Fig. 4. Illustrates the positions of blocks in the flow graph of a sequency-ordered FWT of order $N=2^3=8$. *R* indicates a block that has a reversal. BRO means bit-reversed order. SO means sequency ordered.

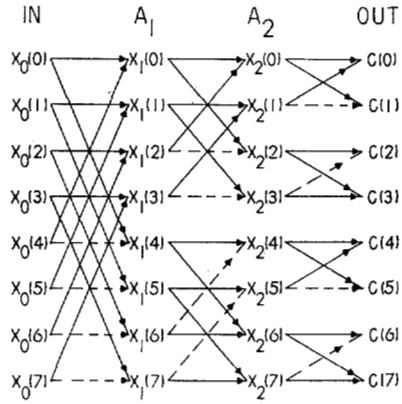

Fig. 5. Signal flow graph for the sequency-ordered FWT. The input data is assumed to be in bit-reversed order. The output coefficients C(*i*) are in sequency order. The solid line indicates multiplication by +1 and the dotted line by −1.

Conclusions

A program for computing the FHT was made by modifying an FFT program and was run on an IBM 360-65 computer in Fortran IV. The FHT program was then modified as described above to yield sequency ordering and run again. A comparison was then made between execution times for the FHT and the modified FHT or FWT for N up to $2^9=512$. It was found that the increase in computation time for the modified FHT over the standard FHT was negligible.

Therefore, the method for generating a sequency-ordered FWT or FHT as described by this paper has several advantages over other sequency ordering methods. The modified FHT is convenient since it can be generated from well-known FHT or FFT techniques, is computationally no more costly than the standard FHT, and retains the attribute of being its own inverse transform.

References

[1] D. A. Bell, "Walsh functions and Hadamard matrices," *Electron. Lett.*, vol. 2, pp. 340–341, Sept. 3, 1966.
[2] J. L. Shanks, "Computation of the fast Walsh–Fourier transform," *IEEE Trans. Comput.* (Short Notes), vol. EC-18, pp. 457–459, May 1969.
[3] N. Ahmed, K. R. Rao, and R. B. Schultz, "A generalized discrete transform," *Proc. IEEE* (Lett.), vol. 59, pp. 1360–1362, Sept. 1971.
[4] S. J. Campanella and G. S. Robinson, "Digital sequency decomposition of voice signals," in *Proc. Walsh Function Symp.*, 1970, pp. 230–237.
[5] C. Yuen, "Walsh Functions and Gray Code," in *Proc. Walsh Function Symposium*, 1970, pp. 68–73.
[6] H. F. Harmuth, *Transmission of Information by Orthogonal Functions.* New York: Springer, 1969, pp. 45–48, p. 50.
[7] K. W. Henderson, "Some notes on the Walsh functions," *IEEE Trans. Electron. Comput.* (Corresp.), vol. EC-13, pp. 50–52 Feb. 1964.