ALGORITHM 749
Fast Discrete Cosine Transform

BARRY G. SHERLOCK
Parks College of Saint Louis University
and
DONALD M. MONRO
University of Bath

An in-place algorithm for the fast direct computation of the forward and inverse Discrete Cosine Transform is presented and evaluated. The transform length may be an arbitrary power of two.

Categories and Subject Descriptors: F.2.1 [Theory of Computation]: Numerical Algorithms and Problems; G.4 [Mathematics of Computing]: Mathematical Software; I.4.2 [Image Processing]: Compression (Coding); E.4 [Data]: Data Compaction and Compression.
General Terms: Algorithms; Transforms.
Additional Key Words and Phrases: Discrete cosine transform; fast transform; data compression.

1. INTRODUCTION

The purpose of this algorithm is the efficient evaluation of the discrete cosine transform (DCT) of a sequence $x = \{x_0, x_1, \ldots, x_{N-1}\}$ of length $N=2^p$, defined as

$$X_n = \frac{2}{N} e(n) \sum_{k=0}^{N-1} x_k \cos\frac{(2k+1)n\pi}{2N} \quad , \quad n=0,1,...,N-1$$

where $X = \{X_0, X_1, \ldots, X_{N-1}\}$ is the transformed sequence, and $e(n) = \begin{cases} \frac{1}{\sqrt{2}} & \mathit{if}\ n=0 \\ 1 & \mathit{otherwise} \end{cases}$

The inverse transform is

$$x_k = \sum_{n=0}^{N-1} e(n)\ X_n \cos\frac{(2k+1)n\pi}{2N} \quad , \quad k=0,1,...,N-1$$

Since its introduction in 1974 [1], the DCT has found widespread application in image and signal processing in general, and in data compression, filtering and feature extraction in particular. The primary reason for its success is its close approximation to the optimal Karhunen-Loeve transform for first-order Markov stationary random data.

Fast transforms for the DCT may be indirect (via the Fast Fourier or other fast transforms) or direct (via direct factorisation or recursive computation). Indirect algorithms include those of Narasimha and Peterson [11], Monro [10], Makhoul [8], Wang [14], Malvar [9] and Chan and Ho [2]. Although various fast direct algorithms have been proposed [3-7,15], published implementations have been for transforms of fixed length, typically N=8 or 16 [12]. The implementation presented here is based upon the direct factorisation algorithm of Cvetkovic and Popovic [4], is in-place and allows the transform length to be an arbitrary power of 2. In terms of numbers of multiplications and additions

Authors' addresses: B.G. Sherlock, Electrical Engineering Department, Parks College of Saint Louis University, Cahokia, IL 62206; D.M. Monro, School of Electronic and Electrical Engineering, University of Bath, Claverton Down, Bath BA7 2AY, U.K.

required, this algorithm equals the most efficient algorithms known [13].

The software is available in single and double precision forms, and has been run on a VAXstation II/GPX under VMS with VAX Fortran version 4.0, a SPARCstation IPX under SunOS release 4.1.2 with Sun Fortran, and a 486-based IBM-compatible personal computer under MS-DOS 6.0 with Microsoft Fortran version 5.00.

2. DESCRIPTION OF THE SOFTWARE

The forward and inverse transforms are implemented in the subroutines FCT and IFCT respectively. Figure 1 shows the flow diagram for the forward transform of length 8. By reference to figure 1 we see that the input data must first be re-ordered with even-indexed points (in natural order) in the first half of the array, and odd-indexed points (in reverse order) in the second half. The data is then transformed by butterfly stages, rearranged into bit-reversed sequence, and finally operated upon by summation stages. The butterfly stages make use of a table of cosine values which are initialised during the first call to FCT or IFCT for the current value of N. The private labelled COMMON variable LENGTH, initialised to zero, is used to hold the current value of N. Whenever FCT or IFCT is called with a value of N not equal to LENGTH (or N <= 1), subroutine INIFCT is called. If INIFCT determines that the new N is a valid transform length, it sets the value of LENGTH to N and initialises the cosine array C to contain values of the form $\left(2\,C_{2N}^{2^{m-1}(4k+1)}\right)^{-1}$

where $\begin{cases} m=1,2,...,p \\ k=0,1,...,\dfrac{N}{2^m}-1 \end{cases}$ and $C_j^i = \cos\dfrac{i\pi}{j}$. They are stored into array C in decreasing order of m as major index and natural order of k as minor index, since this corresponds to the order in which the butterfly stages of subroutines FCT and IFCT will process them.

Because of the orthogonality of the DCT, the inverse transform is obtained simply by reversing the direction of all arrows on the flow diagram of the forward transform.

3. CALLING SEQUENCES

The subroutine FCT has the following calling sequence:

```
        CALL FCT(F,C,N,IFAULT)
```

where the parameters are defined as follows

F	An array of dimension N containing the data sequence to be transformed. On exit, it contains the transformed sequence.
C	An array of dimension N containing cosine coefficients as previously calculated by a call to INIFCT. When this is the first call to FCT or IFCT for the current value of N, this array will be initialized by a call to INIFCT.
N	The length of the discrete cosine transform desired. Must be a power of two satisfying $2 <= N <= 2^{MAXPOW}$. Parameter MAXPOW is explained under "Restrictions" below.

IFAULT On exit:

= 0,	if there are no faults.
= 1,	if N <= 1.
= 2,	if $N > 2^{MAXPOW}$.
= 3,	if $2 <= N <= 2^{MAXPOW}$ but N is not a power of two.

The subroutine IFCT has the following calling sequence:

```
        CALL IFCT(F,C,N,IFAULT)
```

where the parameters are defined as follows

F	An array of dimension N containing the sequence to be inverse transformed. On exit, it contains the inverse transformed sequence.
C	
N	as defined for subroutine FCT
IFAULT	

Subroutine INIFCT generates the array of cosine coefficients required by subroutines FCT and IFCT. It is called internally by FCT and IFCT and need never be called explicitly by the user.

4.RESTRICTIONS

The transform length given by N must be a power of two and greater than 1. The maximum transform length is determined by the parameter MAXPOW which we have arbitrarily set at 20, thereby allowing transforms of length up to 2^{20}. Should longer transforms be desired, MAXPOW can be increased, but 2^{MAXPOW} must not exceed half the maximum positive integer value representable by the version of FORTRAN used. Each of the subroutines checks the length and returns with the fault parameter set and arrays untouched if an error is detected.

5. TIMINGS

Timings of transforms on the VAXstation II/GPS using the VAX FORTRAN compiler version 4.0 and single-precision floating-point numbers are tabulated below:

Transform length N	Forward transform time (s)	Inverse transform time (s)	Cosine array setup time (s)
2	0.000320	0.000321	0.000281
4	0.000663	0.000651	0.000588
8	0.00115	0.00115	0.00110
16	0.00227	0.00226	0.00214
32	0.00485	0.00491	0.00428
64	0.00999	0.00990	0.00840
128	0.0215	0.0215	0.0168
256	0.0471	0.0470	0.0338
512	0.109	0.108	0.0789
1024	0.239	0.235	0.137
2048	0.521	0.513	0.277
4196	1.19	1.19	0.560
8192	2.75	2.63	1.13
16384	5.76	6.07	2.50
32768	11.8	11.8	4.70
65536	25.7	25.2	9.46
131072	54.0	53.0	18.8

6. ACCURACY

The accuracy of the transforms was tested by evaluating forward transforms followed by inverse transforms of various sequences for a wide range of lengths with the results compared point by point with the original. For pseudorandom data scaled between 0 and 1, the following mean square errors were obtained for various transform lengths, using the VAX FORTRAN compiler V4.0 on the VAXstation II/GPX and single-precision floating-point numbers (each error is the average over 200 trials):

Transform length (N)	Mean Square Error
2	8.08 e −16
4	6.02 e −15
8	6.37 e −15
16	2.46 e −14
32	4.92 e −14
64	7.02 e −14
128	2.22 e −13
256	4.33 e −13
512	8.60 e −13
1024	1.67 e −12
2048	3.09 e −12
4096	6.60 e −12
8192	1.24 e −11
16384	2.01 e −11
32768	4.35 e −11
65536	9.17 e −11
131072	3.38 e −10

REFERENCES

[1] AHMED, N., NATARAJAN, T., and RAO, K.R. (1974). Discrete cosine transform. *IEEE Trans. Comput.*, **C-23**, 90-93.

[2] CHAN, S-C. and HO, K-L. (1992). Fast algorithms for computing the discrete cosine transform. *IEEE Trans. Circuits Systems II*, **39**, 185-190.

[3] CHEN, W-H., SMITH, C.H., and FRALICK, S.C. (1977). A fast computational algorithm for the discrete cosine transform. *IEEE Trans. Comms.* **COM-25**, 1004-1009.

[4] CVETKOVIC, Z., and POPOVIC, M.V. (1992). New Fast Recursive Algorithms for the Computation of Discrete Cosine and Sine Transforms, *IEEE Trans. Signal Process.* **SP-40**, 8 , 2083-2086.

[5] HOU, H.S. (1987). A fast recursive algorithm for computing the discrete cosine transform. *IEEE Trans. Acoust. Speech Signal Process.* **ASSP-35**, 1455–1461.

[6] LEE, B.G. (1984). A new algorithm to compute the discrete cosine transform, *IEEE Trans. Acoust. Speech Signal Process.* **ASSP-32**, 1243–1245.

[7] LI, W. (1991). A new algorithm to compute the DCT and its inverse. *IEEE Trans. Signal Process.* **39**, 1305–1313.

[8] MAKHOUL, J. (1980). A fast cosine transform in one and two dimensions. *IEEE Trans. Acoust. Speech Signal Process.* **ASSP-28**, 27–34.

[9] MALVAR, H.S. (1986). Fast computation of the discrete cosine transform through the fast Hartley transform. *Electronics Letters*, **22**, 352–353.

[10] MONRO, D.M. (1979). Interpolation by fast Fourier and Chebyshev transforms. *Int. J. Numerical Methods in Engineering*, **14**, 1679–1692.

[11] NARASIMHA, M.J. and PETERSON, A.M. (1978). On the computation of the discrete cosine transform. *IEEE Trans. Comms.* **COM-26**, 934–936.

[12] RAO, K.R. and YIP, P. (1990). Discrete cosine transform: algorithms, advantages, applications. Academic Press, Inc., 1990.

[13] SKODRAS, A.N. and CHRISTOPOULOS, C.A. (1993). Split-radix fast cosine transform algorithm. *Int. J. Electronics*, **74**, 513–522.

[14] WANG, Z. (1984). Fast algorithms for the discrete W transform and for the discrete Fourier transform. *IEEE Trans. Acoust. Speech Signal Process.*, **ASSP-32**, 803–816.

[15] YIP, P. and RAO, K.R. (1988). The decimation-in-frequency algorithms for a family of discrete sine and cosine transforms. *Circuits Systems Signal Process.*, **7**, 3–19.