

Integer DCTs and Fast Algorithms

(IEEE Tran. SP – Nov. 2001)

I . Lifting matrix, Lifting Step, and properties

Definition 1:

A Lifting matrix is a matrix whose diagonal elements are 1s, and only one nondiagonal element is nonzero.

If the order of a lifting matrix is N , we use the notation $L_{i,j}(s)$ ($i \neq j$) to denote the lifting matrix whose only nonzero element is at the i -th row and the j -th column ($i, j = 0, 1, \dots, N-1$) and whose nondiagonal nonzero element is s .

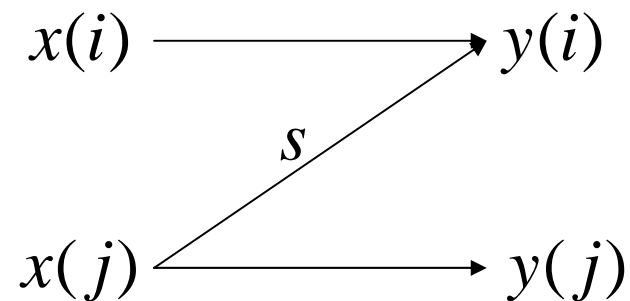
Example,

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 2.3 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \rightarrow L_{2,1}(2.3)$$

To multiply a lifting matrix with a vector, say, $y = L_{i,j}(s) x$, where x and y are column vectors of length N , we have

$$y(i) = x(i) + sx(j)$$

$$y(k) = x(k), k \neq i$$



Flowchart of a lifting step

Definition 2:

A lifting step is multiplying a lifting matrix with a vector.

A distinguished feature of lifting matrix is that its inverse is still a lifting matrix with the same shape.

In fact, we have

$$L_{i,j}^{-1}(s) = L_{i,j}(-s)$$

Therefore,

If a matrix can be factored into products of lifting matrices, its inverse is also products of lifting matrices.

Float-point multiplication is needed if s is an irrational number or even a rational number with unlimited digits.

→

Approximate s by another number. For easy realization, the number is desired to be of the form $\beta/2^\lambda$, where β and λ are integers.

Definition 3:

The notation $RB(s)$ is used to denote a number that is of the form $\beta/2^\lambda$ (dyadic rational number) and approximates to the real number s .

$RB(s)$ is not uniquely dependent on s . It depends on the required accuracy of the approximation.

For example,

$$2.3 \rightarrow 2, 5/2, 9/4, \dots$$

When s is approximated by $RB(s)$, the matrix $L_{i,j}(s)$ is then approximated by $L_{i,j}(RB(s))$, which is still invertible and whose inverse is $L_{i,j}(RB(-s))$.

One can also approximate the lifting step by using a nonlinear transform. For example, y can be approximated by \hat{y} , which is defined as

$$\hat{y}(i) = x(i) + \lfloor sx(j) \rfloor, \quad \hat{y}(k) = x(k), \quad k \neq i$$

This transform is nonlinear! In addition, it maps integer into integer! The nonlinear transform is invertible, and its inverse is:

$$x(i) = \hat{y}(i) + \lfloor s\hat{y}(j) \rfloor, \quad x(k) = \hat{y}(k), \quad k \neq i$$

Remark:

1. When a transform is factored into lifting steps, it is easy to approximate it by another transform, which needs no floating-point multiplications or even by an integer-to-integer nonlinear transform.

Furthermore, the resultant transform is invertible, and its inverse needs no floating-point multiplications as well.

→ That is why we want to factor DCT into lifting steps.

- 2. Theoretically, any matrix with determinant 1 can be factored into products of some lifting matrices. However, generally, $O(N^2)$ lifting matrices are needed in the factorization for a matrix of order N , which is not acceptable for signal processing.**

For special matrices, such as DCT- II s, FFTs, we want to factor them into products of at most **$O(N\log_2N)$** lifting matrices.

DCT- II with transform length $N=2^t$.

Lemma1:

Assume D is a diagonal matrix with determinant 1, say, $D=\text{diag}(b_0, b_1, \dots, b_{L-1})$ where $b_0, b_1, \dots, b_{L-1} = 1$. Let $\alpha_0 = b_0$, $\alpha_k = \alpha_{k-1} b_k$, $k=1, 2, \dots, L-1$.

Then

$D=B_1 B_2 = B_2 B_1$, where

$$B_1 = \text{diag}\left(\alpha_0, \frac{1}{\alpha_0}, \alpha_2, \frac{1}{\alpha_2}, \dots, \alpha_{L-2}, \frac{1}{\alpha_{L-2}}\right)$$

$$B_2 = \text{diag}\left(1, \alpha_1, \frac{1}{\alpha_1}, \alpha_3, \frac{1}{\alpha_3}, \dots, 1\right)$$

if L is even, and

$$B_1 = \text{diag}(\alpha_0, \frac{1}{\alpha_0}, \alpha_2, \frac{1}{\alpha_2}, \dots, 1)$$

$$B_2 = \text{diag}(1, \alpha_1, \frac{1}{\alpha_1}, \alpha_3, \frac{1}{\alpha_3}, \dots, \alpha_{L-2}, \frac{1}{\alpha_{L-2}})$$

if L is odd.

The matrices B_1 and B_2 can be factored as products of Lifting matrices, respectively.

pf:

Since B1 and B2 can be expressed as block diagonal matrices where each block is of the form $\begin{bmatrix} c & 0 \\ 0 & 1/c \end{bmatrix}$

The matrix of order 2 can be expressed as product of lifting matrices:

$$\begin{bmatrix} c & 0 \\ 0 & 1/c \end{bmatrix} = \begin{bmatrix} 1 & c-1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & \frac{1}{c}-1 \\ 0 & c \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -c & 1 \end{bmatrix}$$

Therefore, B1 and B2 (thus D) can also be factored into products

II . Integer DCT- II , DCT-III , and Fast Algorithms

Let $x(n)$ ($n=0,1,\dots,N-1$) be a real input sequence, $N=2^t$, $t>0$. The scaled DCT- II of $x(n)$ is defined as:

$$X(k) = \sum_{n=0}^{N-1} x(n) \cos \frac{\pi(2n+1)k}{2N} \quad , k=0,1,\dots,N-1$$

Let C_N^{II} be the matrix of the DCT- II , that is

$$C_N^{\text{II}} = \left(\cos \frac{\pi k(2n+1)}{2N} \right) \quad , k=0,1,\dots,N-1$$

Lemma 2:

$$C_N^{\Pi} = P_N \begin{bmatrix} I_{N/2} & 0 \\ 0 & U_{N/2} \end{bmatrix} \begin{bmatrix} C_{N/2}^{\Pi} & 0 \\ 0 & C_{N/2}^{\Pi} \end{bmatrix} \begin{bmatrix} I_{N/2} & 0 \\ 0 & D_{N/2} \end{bmatrix} \begin{bmatrix} I_{N/2} & \hat{I}_{N/2} \\ I_{N/2} & -\hat{I}_{N/2} \end{bmatrix}$$

where $I_{N/2}$ is the identity matrix of order $N/2$,
 $\hat{I}_{N/2}$ is the matrix by reversing the rows
of $I_{N/2}$,

$D_{N/2}$ is a diagonal matrix defined by

$$D_{N/2} = \text{diag}\left(2 \cos \frac{\pi}{2N}, 2 \cos \frac{3\pi}{2N}, \dots, 2 \cos \frac{(N-1)\pi}{2N}\right)$$

$$U_{N/2} = \begin{bmatrix} \frac{1}{2} & 0 & 0 & \dots & 0 & 0 \\ -\frac{1}{2} & 1 & 0 & \dots & 0 & 0 \\ \frac{1}{2} & -1 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 & 0 \\ \frac{1}{2} & -1 & 1 & \dots & 1 & 0 \\ -\frac{1}{2} & 1 & -1 & \dots & -1 & 1 \end{bmatrix}$$

$$P_N = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & 1 & \cdots & 0 \\ & & \cdots & & & & & 0 \\ 0 & 0 & \cdots & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

Floating-point multiplications are needed for the algorithm when the matrix $D_{N/2}$ is multiplied by a factor.

In order to avoid float multiplications, we want to turn $D_{N/2}$ into products of lifting matrices and then approximate the elements of the lifting matrices by numbers that are of the form $\beta/2^\lambda$, where β and λ are integers.

$D_{N/2} = E_{N/2} F_{N/2}$ where

$$E_{N/2} = \text{diag}(\sqrt{2}, 1, \dots, 1)$$

$$F_{N/2} = \text{diag}\left(\sqrt{2} \cos \frac{\pi}{2N}, 2 \cos \frac{3\pi}{2N}, \dots, 2 \cos \frac{(N-1)\pi}{2N}\right)$$

Lemma 3:

$$\det(F_{N/2}) = 1$$

pf: by mathematical induction.

Based on Lemma 1,

$$F_{N/2} = \left\{ \prod_{k=0}^{N/4-1} [L_{2k,2k+1}(\alpha_{2k} - 1)L_{2k+1,2k} \times L_{2k,2k+1}(\frac{1}{\alpha_{2k}} - 1)L_{2k+1,2k}(-\alpha_{2k})] \right\} \bullet$$

$$\left\{ \prod_{k=0}^{N/4-1} [L_{2k-1,2k}(\alpha_{2k-1} - 1)L_{2k,2k-1} \times L_{2k-1,2k}(\frac{1}{\alpha_{2k-1}} - 1)L_{2k,2k-1}(-\alpha_{2k-1})] \right\}$$

where

$$\alpha_0 = \sqrt{2} \cos \frac{\pi}{2N}$$

$$\alpha_k = \alpha_{k-1} \cdot 2 \cos \frac{(2k+1)\pi}{2N}, \quad k=1, 2, \dots, \frac{N}{2} - 1.$$

In order to avoid float multiplications, we now approximate the nonzero element of the lifting matrices by numbers that are of the form $\beta/2^\lambda$, where β and λ are integers, that is, we replace every nonzero element s by $RB(s)$, That is,

$$E_{N/2} \rightarrow \bar{E}_{N/2} = \text{diag}(RB(\sqrt{2}), 1, \dots, 1)$$

$$F_{N/2} = \bar{F}_{N/2}$$

where

$$\bar{F}_{N/2} = \left\{ \prod_{k=0}^{N/4-1} [L_{2k,2k+1}(RB(\alpha_{2k}) - 1)L_{2k+1,2k} \times L_{2k,2k+1}(RB(\frac{1}{\alpha_{2k}}) - 1)L_{2k+1,2k}(RB(-\alpha_{2k}))] \right\} \cdot$$

$$\left\{ \prod_{k=0}^{N/4-1} [L_{2k-1,2k}(RB(\alpha_{2k-1}) - 1)L_{2k,2k-1} \times L_{2k-1,2k}(RB(\frac{1}{\alpha_{2k-1}}) - 1)L_{2k,2k-1}(RB(-\alpha_{2k-1}))] \right\}$$

Then

$$C_N^{\Pi} \rightarrow \bar{C}_N^{\Pi} = P_N \begin{bmatrix} I_{N/2} & 0 \\ 0 & U_{N/2} \end{bmatrix} \begin{bmatrix} \bar{C}_{N/2}^{\Pi} & 0 \\ 0 & \bar{C}_{N/2}^{\Pi} \end{bmatrix} \begin{bmatrix} I_{N/2} & 0 \\ 0 & \bar{D}_{N/2} \end{bmatrix} \begin{bmatrix} I_{N/2} & \hat{I}_{N/2} \\ I_{N/2} & -\hat{I}_{N/2} \end{bmatrix}$$

where $\bar{D}_N = \bar{E}_{N/2} \bar{F}_{N/2}$

**The matrix $\bar{C}_{N/2}^{\text{II}}$ defines a new transform that does not need float multiplications.
: type- II Integer discrete cosine transform (IntDCT- II)**

Definition 4: $N=2^t$, the transform matrix of an IntDCT- II \bar{C}_N^{II} is defined recursively by $\bar{C}_1^{\text{II}} = (1)$, and

$$\bar{C}_{2^j}^{\text{II}} = P_N \begin{bmatrix} I_{2^{j-1}} & 0 \\ 0 & U_{2^{j-1}} \end{bmatrix} \begin{bmatrix} \bar{C}_{2^{j-1}}^{\text{II}} & 0 \\ 0 & \bar{C}_{2^{j-1}}^{\text{II}} \end{bmatrix} \begin{bmatrix} I_{2^{j-1}} & 0 \\ 0 & \bar{D}_{2^{j-1}} \end{bmatrix} \begin{bmatrix} I_{2^{j-1}} & \hat{I}_{2^{j-1}} \\ I_{2^{j-1}} & -\hat{I}_{2^{j-1}} \end{bmatrix}$$

$j=1, 2, \dots, t.$

The transform is not unique. Actually, any choice of function RB determines a transform.

Algorithm I - Fast Algorithm for IntDCT-II

step 1) Compute

$$g(n) = x(n) + x(N-1-n)$$

$$h(n) = x(n) - x(N-1-n), \quad n=0, 1, \dots, \frac{N}{2} - 1.$$

step 2) Compute $\hat{h} = \bar{E}_{N/2} \bar{F}_{N/2} h$, where

$$\hat{h} = (\hat{h}(0), \hat{h}(1), \dots, \hat{h}(\frac{N}{2} - 1))^t$$

$$h = (h(0), h(1), \dots, h(\frac{N}{2} - 1))^t$$

step 3) Compute the IntDCT- II with length $N/2$ of sequences $g(n)$ and let the outputs be $G(k)$ and $H(k)$, respectively. The steps 1 and 2 can be used recursively for the computations.

step 4) Compute

$$X(2k)=G(k), k=0,1,\dots, \frac{N}{2} -1$$

$$X(1)=H(0)/2, X(2k+1)=H(k)-X(2k-1)$$

$$k=0,1,\dots, \frac{N}{2} -1$$

Lemma 4: The computation of step 2 in Algorithm I needs $3\left(\frac{N}{2}-1\right)$ lifting steps, $\frac{N}{2}-1$ additions, and one integer multiplication.

Let $LC^{\text{II}}(N)$ and $AC^{\text{II}}(N)$ represent the number of lifting steps and additions for computing an IntDCT- II with length N , then

$$LC^{\text{II}}(N) = 2LC^{\text{II}}\left(\frac{N}{2}\right) + 3\left(\frac{N}{2} - 1\right)$$

$$AC^{\text{II}}(N) = 2AC^{\text{II}}\left(\frac{N}{2}\right) + 2(N - 1)$$

⇒

$$LC^{\Pi}(N) = \frac{3}{2} N \log_2 N - 3N + 3$$

$$AC^{\Pi}(N) = 2N \log_2 N - 2N + 2$$

\bar{C}_N^{Π} is invertible!!

$$(\bar{C}_N^{\Pi})^{-1} = \begin{bmatrix} \frac{1}{2} I_{\frac{N}{2}} & \frac{1}{2} I_{\frac{N}{2}} \\ \frac{1}{2} \hat{I}_{\frac{N}{2}} & -\frac{1}{2} \hat{I}_{\frac{N}{2}} \end{bmatrix} \begin{bmatrix} I_{\frac{N}{2}} & 0 \\ 0 & \bar{F}_{\frac{N}{2}}^{-1} \bar{E}_{\frac{N}{2}}^{-1} \end{bmatrix} (\bar{C}_{\frac{N}{2}}^{\Pi})^{-1} \begin{bmatrix} 0 & 0 \\ (\bar{C}_{\frac{N}{2}}^{\Pi})^{-1} & 0 \end{bmatrix} \begin{bmatrix} I_{\frac{N}{2}} & 0 \\ 0 & U_{\frac{N}{2}}^{-1} \end{bmatrix} P_N^{-1}$$

where

$$U_{\frac{N}{2}}^{-1} = \begin{bmatrix} 2 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 1 \end{bmatrix}$$

$$\bar{F}_{\frac{N}{2}}^{-1} = \left\{ \prod_{k=1}^{\frac{N}{4}-1} \left[L_{2k,2k-1}(\text{RB}(\alpha_{2k-1})) L_{2k-1,2k} \left(-\text{RB} \left(\frac{1}{\alpha_{2k-1}} \right) + 1 \right) \cdot L_{2k,2k-1}(-1) \cdot L_{2k-1,2k}(-\text{RB}(\alpha_{2k}) + 1) \right] \right\} \times$$

$$\left\{ \prod_{k=0}^{\frac{N}{4}-1} \left[L_{2k+1,2k}(-\text{RB}(\alpha_{2k-1})) L_{2k,2k+1} \left(-\text{RB} \left(\frac{1}{\alpha_{2k}} \right) + 1 \right) \cdot L_{2k+1,2k}(-1) \cdot L_{2k,2k+1}(-\text{RB}(-\alpha_{2k-1}) + 1) \right] \right\}$$

$$\bar{E}_{N/2}^{-1} = \text{diag}\left(\frac{1}{RB(\sqrt{2})}, 1, \dots, 1\right)$$

Since the inverse of DCT- II is DCT- III.
Therefore, the Integer DCT- III is defined
by $(\bar{C}_N^{\text{II}})^{-1}$

Algorithm 2 - Fast Algorithm for **IntDCT- III**

step 1) Compute

$$G(k) = X(2k) , k=0, 1, \dots, \frac{N}{2} - 1.$$
$$H(0) = 2X(1) , H(k) = X(2k+1) + X(2k-1)$$
$$k=1, 2, \dots, \frac{N}{2} - 1.$$

step 2) Compute the inverse IntDCT- II of sequences $G(K)$, and $H(K)$, and let the outputs be $g(n)$ and $h(n)$, respectively. Steps 1, 3, and 4 can be used recursively for the computations.

step 3) Compute $P = \bar{F}_{N/2}^{-1} \bar{E}_{N/2}^{-1} h$, where

$$p = (p(0), p(1), \dots, p(\frac{N}{2} - 1))^t$$

$$h = (h(0), h(1), \dots, h(\frac{N}{2} - 1))^t$$

step 4) Compute

$$x(n) = (g(n) + p(n))/2$$

$$x(N - 1 - n) = (g(n) - p(n))/2, \quad n=0, 1, \dots, \frac{N}{2} - 1.$$

The number of operations for reconstruction is the same as that for the forward transform.

Notice that, although $(\bar{C}_N^{\text{II}})^{-1}$ is not much different from $(\bar{C}_N^{\text{II}})^t$, $(\bar{C}_N^{\text{II}})^{-1}$ may very different from $(\bar{C}_N^{\text{II}})^t$. In other words, the matrix \bar{C}_N^{II} is not orthogonal in general. Therefore, we can not use $(\bar{C}_N^{\text{II}})^t$ for reconstruction!!