A Study of Integer Transform

Y.S. Zhang

Abstract

The DCT and the DFT are used widely in signal processing due to the invention of the FFT. Although the FFT is fast enough for real-time application, the cost will be expansive if we use floating-point processor. When we use fixed-point processor to compute the FFT, the result may not be good enough because of the existence of rounding error. A question now arises: If we use fixed-point processor for the FFT[1], how to find a method to make the result good enough. The integer transform[2] is one category of the solutions. The purpose of this paper is to summarize the integer transform categories proposed by the researchers up to now.

1 Background

Why are the DFT and the DCT so useful? Gilbert Strang[10] provided a comprehensive insight for these questions. We explain again the idea of Gilbert Strang.

1.1 An insight for DFT and DCT

Given an $N \times N$ matrix $\mathbf{A}_N$,

$$\mathbf{A}_n = \begin{bmatrix} 2 & -1 & 0 & \dots & -1 \\ -1 & 2 & -1 & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & 0 & -1 & 2 & -1 \\ -1 & 0 & \dots & -1 & 2 \end{bmatrix}. \tag{1}$$

How surprising, the columns of an $N \times N$ DFT matrix are eigenvectors of $\mathbf{A}_N$! Why? Let us look carefully at the structure of $\mathbf{A}_N$. The interior rows are the same in every matrix. The jth entry of $\mathbf{A}_N u$ is $-u_{j-1} + 2u_j - u_{j+1}$, which is discrete approximation for $-u''$. At $j = 0$ and $j = N-1$, the second difference contains u_{-1} and u_N. $u_{-1} = u_{N-1}$ and $u_N = u_0$ due to periodicity. The solution of $-u''(x) = \lambda u(x)$ is $u(x) = C_1 e^{j\sqrt{\lambda}x} + C_2 e^{-j\sqrt{\lambda}x}$. Using periodic property: $u(0) = u(N)$, then $\sqrt{\lambda} = \frac{2\pi k}{N} \quad \forall k \in \mathbb{Z}$. Discretize $x \in \mathbb{R}$ to $n \in \mathbb{Z}$, we see that $e^{\frac{2\pi kn}{N}}$ is an $N \times N$ DFT matrix!

Can we using the same method to derive all type of DCT? The answer is certainly. The key point is to change the boundary condition of $-u''(x) = \lambda u(x)$. The boundary condition that produces cosine is $u'(0) = 0$, Another boundary can be Neumann boundary condition or Dirichlet boundary condition:

[1] More generally, the FFT can be replace by any finite dimensional linear transform

[2] The entries of forward matrix $\mathbf{A}$ and inverse matrix $\mathbf{A}^{-1}$ are linear combination of $2^k \forall k \in \mathbb{Z}$

Neumann: $u'(N) = 0$ gives eigenfuntions $u_k(x) = \cos(\frac{\pi kx}{N})$.

Dirichlet: $u(N) = 0$ gives eigenfuntions $u_k(x) = \cos(\frac{\pi(k+\frac{1}{2})x}{N})$.

It's time to discretize the boundary condition. Notice that every boundary condition has two fundamental approximations. At each boundary, the condition on u can be impose at a meshpoint[3] or at a midpoint[4]. It is, in fact, the key to get all type of DCT.

How to derive all type of DCT is a question which I want to keep beyond the scope of this present discussion. For detail, reference Gilbert Strang's paper[10].

To conclude this section. I give a citation from G. Strang:

> Each continuous problem (differential equation) has many discrete approximations (difference equations). The discrete case has a new level of variety and complexity, often appearing in the boundary conditions.

The purpose of this quotation is to show that it is a mistake to think that discrete case is just a special case of continuous counterpart.

2 Introduction

Redisplay the question: If we use fixed-point processor for FFT, how to find a method to make the result good enough. Good enough..., What's mean? S.C. Pei and J.J. Ding[6] give a good criterion:

> (**Goal I, binary entries**): Both the forward transform $\mathbf{B}$ and inverse transform $\mathbf{B}_1$ are binary-valued matrices.
>
> (**Goal II, reversibility**): The integer transform is perfectly recoverable, i.e., $\mathbf{B}_1 \cdot \mathbf{B} = \mathbf{I}(\mathbf{B}_1 = \mathbf{B}^{-1})$.
>
> (**Goal III, easy to design**).
>
> (**Goal IV, high accuracy**): If $\mathbf{y}$ and $\mathbf{z}$ are the transform result of the original non-integer transform and the integer transform, respectively, then $\mathbf{z} \approx \sigma \cdot \mathbf{y}$ where σ is some constant.
>
> (**Goal V, not increase the complexity for implementation**): It includes the hardware cost and the number of time cycles.

Although There are many methods, they may be classified into three main categories: prototype method, lifting scheme, and triangular scheme. In 1989, W.-K. Cham[3] proposed a method that used an integer matrix to approximate the DCT, This integer matrix satisfied **Goal I** and **Goal II**. This method belongs to the prototype method. The feature of prototype method is to create a prototype matrix according to the characteristics of the original matrix, then tuning the parameters of the prototype matrix to what we want. So this method is hard to generalize to general matrix.

Subsequently, some researchers[2, 7, 8] propose a method applied to power-of-two FFT-like matrices[5]. This method belongs to lifting scheme. It meet **Goal I**, **Goal II**, and **Goal III**, but it limited by power-of two FFT-like matrices.

[3] e.g. $u'(0) = 0 \Rightarrow u_{-1} = u_1$

[4] e.g. $u'(0) = 0 \Rightarrow u_{-1} = u_0$

[5] Implementation of the matrix consists of 2×2 matrices.

In 2001, P. Hao, Q. Shi[9] give a new method to generalize lifting scheme. This method belongs to triangular method,which solves the limitation above. Then S.C. Pei and J.J. Ding[6] provide a method to improve triangular method.

In the following section, we will illustrate how these methods works, what are the advantages and the disadvantages of these methods.

3 Prototype Matrix Method

3.1 How to Convert

Consider an N-point DCT kernel $\mathbf{A}$ where

$$\mathbf{A}_{k,n} = \cos\left[\frac{\pi}{N}(n+\frac{1}{2})k\right] \tag{2}$$

How to convert it into the integer transform? The following gives steps for $N = 8$ to answer this question.

Step1: The 8-point DCT kernel $\mathbf{A}$ is

$$\mathbf{A} \approx \begin{pmatrix} 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 \\ 0.4904 & 0.4157 & 0.2778 & 0.0975 & -0.0975 & -0.2778 & -0.4157 & -0.4904 \\ 0.4619 & 0.1913 & -0.1913 & -0.4619 & -0.4619 & -0.1913 & 0.1913 & 0.4619 \\ 0.4157 & -0.0975 & -0.4904 & -0.2778 & 0.2778 & 0.4904 & 0.0975 & -0.4157 \\ 0.3536 & -0.3536 & -0.3536 & 0.3536 & 0.3536 & -0.3536 & -0.3536 & 0.3536 \\ 0.2778 & -0.4904 & 0.0975 & 0.4157 & -0.4157 & -0.0975 & 0.4904 & -0.2778 \\ 0.1913 & -0.4619 & 0.4619 & -0.1913 & -0.1913 & 0.4619 & -0.4619 & 0.1913 \\ 0.0975 & -0.2778 & 0.4157 & -0.4904 & 0.4904 & -0.4157 & 0.2778 & -0.0975 \end{pmatrix} \tag{3}$$

To express the order-8 DCT kernel $\mathbf{A}$ in the form of a matrix of variables:

$$\mathbf{A} = [k_0 J_0, k_1 J_1, k_2 J_2, \ldots, k_7 J_7]^T \tag{4}$$

where J_i is integer vector, and k_i is a constant scalar such that $|k_i \cdot J_i| = 1$. From(3), we know J

$$J = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ a & b & c & d & -d & -c & -b & -a \\ e & f & -f & -e & -e & -f & f & e \\ b & -d & -a & -c & c & a & d & -b \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ c & -a & d & b & -b & -d & a & -c \\ f & -e & e & -f & -f & e & -e & f \\ d & -c & b & -a & a & -b & c & -d \end{pmatrix} \tag{5}$$

Step2: To find the conditions under which J_i and J_j are orthogonal: To proceed we need some idea about dyadic symmetry.

Definition 3.1. *A vector of 2^m elements $[a_0, a_1, \ldots, a_{2^m-1}]$ is said to have the ith dyadic symmetry iff $a_j = s \cdot a_{j \oplus i}$, where $\oplus$ is the 'exclusive or' operation, j lies in the range $[0, 2^m - 1]$ and i in the range $[1, 2^m - 1]$, $s = 1$ when the symmetry is even, and $s = -1$ when the symmetry is odd.*

Table 1: Sth dyadic symmetry type in basis vector J_i

Dyadic symmetry S in $\mathbf{J}_i$							
i	1	2	3	4	5	6	7
0	E	E	E	E	E	E	E
1	-	-	-	-	-	-	O
2	-	-	O	-	-	-	E
3	-	-	-	-	-	-	O
4	O	O	E	E	O	O	E
5	-	-	-	-	-	-	O
6	-	-	O	-	-	-	E
7	-	-	-	-	-	-	O

Table 2: Conditions under which the ith basis vector and the jth basis vector are orthogonal

i							
1	2	3	4	5	6	7	j
*3	*2	*3	*2	*3	*2	*3	0
	*3	*1	*3	*1	*3	*4	1
		*3	*2	*3	*4	*3	2
			*3	*4	*3	*1	3
				*3	*2	*3	4
					*3	*1	5
						*3	6

*1 if $b = a \cdot c + b \cdot d + c \cdot d$
*2 must be orthogonal due to the 3rd dyadic symmetry
*3 must be orthogonal due to the 7th dyadic symmetry
*4 must be orthogonal as their dot product equals zero

Theorem 3.1 (Theorem of orthogonality)**.** *Two vectors* $\mathbf{u}$ *and* $\mathbf{v}$ *are orthogonal if* $\mathbf{u}$ *and* $\mathbf{v}$ *have the same type of dyadic symmetry and one is even and the other is odd.*

Table 1 lists the type of dyadic symmetry present in each basis vector.We then examine the condition under which J_i and J_j are orthogonal for all i, j. Table 2 gives that the only condition that the constants a, b, c, d, and e must satisfy to ensure that the transform $\mathbf{A}$ be orthogonal is

$$a \cdot b = a \cdot c + b \cdot d + c \cdot d \tag{6}$$

Step3: To set up boundary conditions and generate new transforms: For example, we can let

$$a \geq b \geq c \geq d, \quad e \geq f \tag{7}$$

Then use computer to find all solutions.

3.2 Advantage and Disadvantage

The advantage is that we can find a lot of integer transforms. The probability of high performance is vary large because there are mass solutions. But this advantage is also a disadvantage because of much time spent by searching these matrices. When a, b, c, d, e, f are represented by 8-bits, then we need to spend about half hour to find the solutions. Another disadvantage is that different matrix may have different features, we can just specify the features of the matrix case by case. No general rules can follow. It is too inconvenient to be used in practice. In the next section, we will introduce lifting scheme to overcome these problem.

4 Lifting Scheme

4.1 How Does Lifting Work

The basic idea of the lifting scheme is that if a matrix has the following form

$$\mathbf{B} = \begin{bmatrix} 1 & b \\ 0 & 1 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} 1 & 0 \\ b & 1 \end{bmatrix} \tag{8}$$

then

$$\mathbf{B}^{-1} = \begin{bmatrix} 1 & -b \\ 0 & 1 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} 1 & 0 \\ -b & 1 \end{bmatrix} \tag{9}$$

These matrices are called the lifting step. The lifting scheme is the matrix which can be decomposed by lifting step. If b is rounded by n-bit, the condition $\mathbf{B} \cdot \mathbf{B}^{-1} = \mathbf{I}$ is still satisfied. An example is to use this method to approximate order-8 fast floating-point DCT.

Notice Figure 1, the only place need to approximate is rotation angle$\{\frac{\pi}{4}, \frac{3\pi}{8}, \frac{\pi}{4}, \frac{7\pi}{16}, \frac{3\pi}{16}\}$. For any angle θ can be represented by a matrix T_θ, which can be decomposited to three lift step.

$$T_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} 1 & \frac{\cos\theta - 1}{\sin\theta} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \sin\theta & 1 \end{bmatrix} \begin{bmatrix} 1 & \frac{\cos\theta - 1}{\sin\theta} \\ 0 & 1 \end{bmatrix} \tag{10}$$

For example:

$$\begin{aligned} T_{\frac{\pi}{4}} &= \begin{bmatrix} 1 & \frac{\cos(\frac{\pi}{4}) - 1}{\sin(\frac{\pi}{4})} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \sin(\frac{\pi}{4}) & 1 \end{bmatrix} \begin{bmatrix} 1 & \frac{\cos(\frac{\pi}{4}) - 1}{\sin(\frac{\pi}{4})} \\ 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & -0.4142 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0.7071 & 1 \end{bmatrix} \begin{bmatrix} 1 & -0.4142 \\ 0 & 1 \end{bmatrix} \\ &\approx \begin{bmatrix} 1 & -\frac{7}{16} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \frac{3}{4} & 1 \end{bmatrix} \begin{bmatrix} 1 & -\frac{7}{16} \\ 0 & 1 \end{bmatrix} \end{aligned} \tag{11}$$

Then the transformation can be reconstructed perfectly.

4.2 Advantage and Disadvantage

For any power-of-two FFT-like matrix, we can design an integer transform easily. This scheme satisfies **Goal I**, **Goal II**, and **Goal III**. One may notice that if we use $\mathbf{T}_\theta$ to

Figure 1: The forward fast floating-point DCT

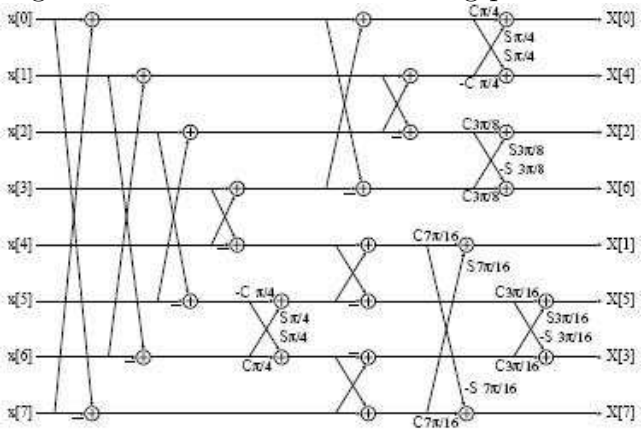


Figure 2: Butterfly structure for implementing a complex multiplication

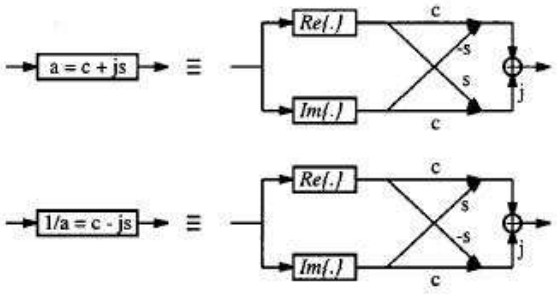
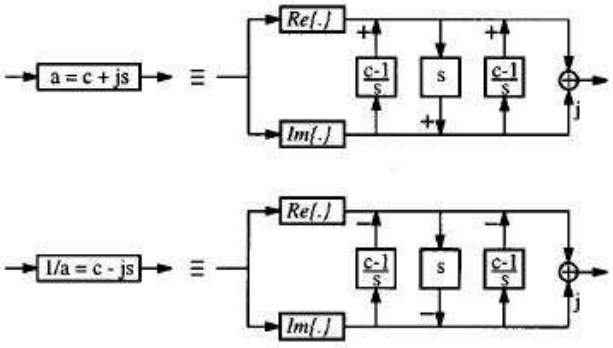


Figure 3: Lifting structure for implementing a complex multiplication and it's inverse.



input $\mathbf{x}$ directly

$$\mathbf{T}_\theta \mathbf{x} = \begin{bmatrix} \cos\theta \times x_1 - \sin\theta \times x_2 \\ \sin\theta \times x_1 + \cos\theta \times x_2 \end{bmatrix} \tag{12}$$

The number of multiplications and additions are four and two, respectively. If we use three lift steps to input $\mathbf{x}$,

$$\begin{bmatrix} 1 & \frac{\cos\theta - 1}{\sin\theta} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \sin\theta & 1 \end{bmatrix} \begin{bmatrix} 1 & \frac{\cos\theta - 1}{\sin\theta} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{13}$$

The number of multiplications and additions are three and three, respectively. The same result can observed from Figure 2 and Figure 3. This is another advantage of lifting scheme.

But this scheme is still not general engough, it can just apply on power-of-two FFT-like matrix. More noteworthy is that the bits which represent the input will increase when lift step applys on it. To take a simple example:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & \frac{1}{4} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{14}$$

$$y_1 = x_1 + \frac{1}{4} x_2 \tag{15}$$

The bits need to represent y_1 is increased by two. So we will induce trucation error if the storage is too small. This implies lifting scheme cannot meet **Goal IV**. In next section, we exhibit a new method to solve this problem.

5 Triangular Matrix Scheme

Before we introduce triangular matrix scheme, we explain what **Goal IV** means. For a linear transformation $\mathbf{y} = \mathbf{A}\mathbf{x}$, if it directly maps integers to integers by rounding operation (denoted by $\mathcal{Q}$), $\hat{\mathbf{y}} = \mathcal{Q}(\mathbf{A}\mathbf{x})$, rounding errors may be inevitable:
$\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}} = \mathbf{y} - \mathcal{Q}(\mathbf{A}\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathcal{Q}(\mathbf{A}\mathbf{x})$. How can we minimize $\mathbf{e}$? S.C. Pei and J.J. Ding based on P. Hao and Q. Shi's idea to provide a method to decrease $||\mathbf{e}||$. First of all, we have to introduce what P. Hao and Q. Shi mean. For **Goal I,II**, P. Hao, Q. Shi[9] give a rigorous view. The following is the view of they:

> For the uncertain rounding error vector, $\mathbf{c}$, $||\mathbf{c}|| \leq u$, where u is the unit roundoff, which is defined as the largest error that can occur in a rounding operation. Then, we have $\mathcal{Q}(\mathbf{c}) = \mathbf{0}$. Wait! The rounding operation has infinite choice. How can we choose an appropriate one here? From M.D. Adams[5], we have the following definition:
>
> **Definition 5.1.** *The rounding operator $\mathcal{Q}$ satisfies*
>
> $$\mathcal{Q}(x + \alpha) = x + \mathcal{Q}(\alpha), \text{ for all } x \in \mathbb{Z}, \text{and all } \alpha \in \mathbb{R} \tag{16}$$
>
> *is called integer-shift invariant operator.*
>
> The floor, biased floor, ceiling, or biased ceiling operator are this type. For simplifying, we choose $\mathcal{Q}$ which is integer-shift invariant operator. So, if the

transform matrix $\mathbf{A}$ is nonsingular, the perfectly reversible condition will be $\mathbf{x} = \mathcal{Q}(\mathbf{A^{-1}}\hat{\mathbf{y}}) = \mathcal{Q}(\mathbf{A^{-1}}(\mathcal{Q}(\mathbf{Ax}))) = \mathcal{Q}(\mathbf{A^{-1}}(\mathbf{Ax}-\mathbf{e})) = \mathcal{Q}(\mathbf{x}-\mathbf{A^{-1}e}) = \mathbf{x} + \mathcal{Q}(-\mathbf{A^{-1}e})$. Thus, its necessary and sufficient condition is that $||-\mathbf{A^{-1}e}||_\infty \leq u$. Since $||-\mathbf{A^{-1}c}||_\infty = ||\mathbf{A^{-1}c}||_\infty \leq ||\mathbf{A^{-1}}||_\infty \cdot ||\mathbf{c}||_\infty \leq ||\mathbf{A^{-1}}||_\infty \cdot u$, a sufficient condition for $||-\mathbf{A^{-1}e}||_\infty \leq u$ is $||\mathbf{A^{-1}}||_\infty \leq 1$. Since perfect reversible is two-sided, $||\mathbf{A}||_\infty \leq 1$. In generally, a matrix $\mathbf{A}$ cannot satisfy two condition simultaneously except elementary reversible matrices because $||\mathbf{A}||_\infty||\mathbf{A}^{-1}||_\infty \geq ||\mathbf{A}\mathbf{A}^{-1}||_\infty = 1$. Therefore, we concentrate on

(i) whether a transform matrix can be factorized into a series of elementary integer reversible transform matrices;

(ii) how to find this kind of factorization;

(iii) what its optimal form is.

There are some important theorems in P. Hao, Q. Shi[9]

Definition 5.2 (integer factor)**.** *An integer factor is defined as a multiplier of integers that does not change their magnitude*

For real numbers, an integer factor can be 1 or -1; for complex numbers, integer factors can be 1, -1, j, or $-j$.

Definition 5.3 (TERM)**.** *An upper or lower triangular transform matrix whose diagonal elements are integer factors is called a triangular elementary reversible matrix(TERM)*

Definition 5.4 (SERM)**.** *A single-row elementary reversible matrix(SERM) is a matrix with integer factors on the diagonal and only onw row of off-diagonal elements that are not all zeros.*

Theorem 5.1. *Matrix* $\mathbf{A}$ *has a TERM factorization of* $\mathbf{A} = \mathbf{P}\mathbf{V}_1\mathbf{V}_2\dots\mathbf{V}_M\mathbf{D}_R$ *iff* $|det\mathbf{A}| = 1$*, were* M *is finite,* $\mathbf{V}_k(k = 1, 2, \dots, M)$ *are unit TERMs,* P *is a permutation matrix, and* $\mathbf{D}_R$ *is a rotator for only one complex number.*

Theorem 5.2. *Matrix* $\mathbf{A}$ *has a unit triangular factorization of* $\mathbf{A} = \mathbf{LU}$ *iff the minors of the leading principal submatrices of* $\mathbf{A}$ *are all* 1*s.*

Theorem 5.3. *Given a nonsingular diagonal matrix* $\mathbf{D}_R = diag(1, 1, \dots, 1, e^{i\theta})$*, matrix* $\mathbf{A}$ *has a factorization of* $\mathbf{A} = \mathbf{PLD}_R\mathbf{US}_0$ *iff* $det\mathbf{P}^T\mathbf{A} = det\mathbf{D}_R \neq 0$.

Theorem 5.4. *Matrix* $\mathbf{A}$ *has a unit SERM factorization of* $\mathbf{A} = \mathbf{S}_N\mathbf{S}_{N-1}\dots\mathbf{S}_1$ *iff the minors of the leading principal submatrices of* $\mathbf{A}$ *are all* 1*s, where* $\mathbf{S}_m(m = 1, 2, \dots, N)$ *are unit SERMs.*

The above theorems just give the existence of TERM and SERM. For detail, please reference[9]. Following, we will provide two approaches to construct TERM and one approach to SERM

5.1 TERM based on P. Hao and Q. Shi(revised)

For a reversible matrix $\mathbf{A}$

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \dots & a_{NN} \end{bmatrix} \tag{17}$$

If $a_{1N}^{(1)} \neq 0$, then there must be a number s_1 such that $a_{11}^{(1)} - s_1 \cdot a_{1N}^{(1)} = 1$. So, we can get $s_1 = \frac{a_{11}^{(1)} - 1}{a_{NN}^{(1)}}$ and obtain a product of

$$\begin{aligned} \mathbf{AS}_{01} &= \mathbf{A} \begin{bmatrix} 1 & & \\ & \mathbf{I} & \\ -s_1 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & a_{12}^{(1)} & \dots & a_{1N}^{(1)} \\ a_{21}^{(1)} - s_1 a_{2N}^{(1)} & a_{22}^{(1)} & \dots & a_{2N}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1}^{(1)} - s_1 a_{NN}^{(1)} & a_{N2}^{(1)} & \dots & a_{NN}^{(1)} \end{bmatrix} \end{aligned} \tag{18}$$

Then, the forward elimination of the first column can be achieved by multiplying an elementary Gauss matrix $\mathbf{L}_1$

$$\begin{aligned} \mathbf{L}_1\mathbf{AS}_{01} &= \begin{bmatrix} 1 & & & \\ s_1 a_{2N}^{(1)} - a_{21}^{(1)} & 1 & & \\ \vdots & & \mathbf{I} & \\ s_1 a_{NN}^{(1)} - a_{N1}^{(1)} & & & 1 \end{bmatrix} \mathbf{AS}_1 \\ &= \begin{bmatrix} 1 & a_{12}^{(1)} & \dots & a_{1N}^{(1)} \\ 0 & a_{22}^{(1)} & \dots & a_{2N}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{N2}^{(1)} & \dots & a_{NN}^{(1)} \end{bmatrix} \end{aligned} \tag{19}$$

Continuing in this way, we get

$$\begin{aligned} \mathbf{L}_{N-1} \dots \mathbf{L}_2\mathbf{L}_1\mathbf{AS}_{01}\mathbf{S}_{02} \dots \mathbf{S}_{0(N-1)} &= \begin{bmatrix} 1 & a_{12}^{(N-1)} & \dots & a_{1N}^{(N-1)} \\ 0 & 1 & \dots & a_{2N}^{(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_{NN}^{(N-1)} \end{bmatrix} \\ &= \mathbf{D}_R\mathbf{U} \end{aligned} \tag{20}$$

where $a_{NN}^{(N-1)} = e^{i\theta}$, $\mathbf{D}_R = diag(1, 1, \dots, 1, e^{i\theta})$, and

$$\mathbf{U} = \begin{bmatrix} 1 & a_{12}^{(N-1)} & \dots & a_{1N}^{(N-1)} \\ 0 & 1 & \dots & a_{2N}^{(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_{NN}^{(N-1)} \end{bmatrix} \tag{21}$$

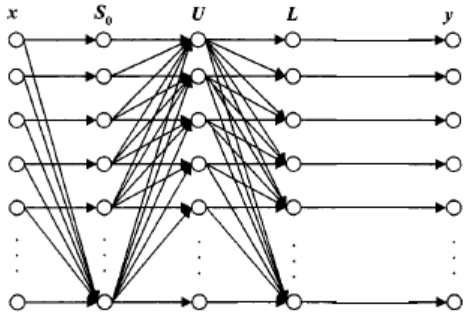


Figure 4: Flowchart structure of the linear transform implemented by three TERMs.

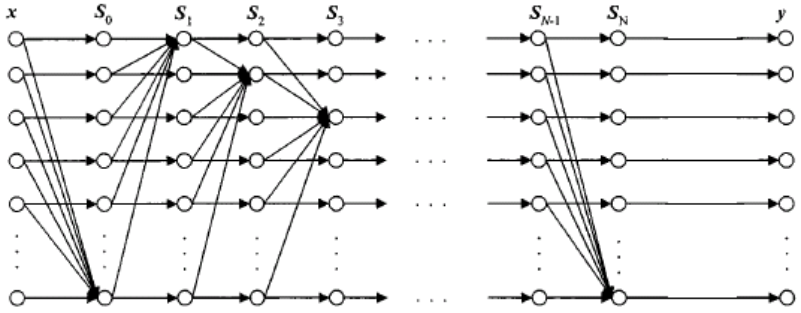


Figure 5: Flowchart structure of the linear transform implemented by SERMs.

The left and right sides of $\mathbf{A}$ is

$$\mathbf{L}_{N-1}\ldots\mathbf{L}_2\mathbf{L}_1 = \mathbf{L}^{-1} \tag{22}$$

$$\mathbf{S}_{01}\mathbf{S}_{02}\ldots\mathbf{S}_{0(N-1)} = \begin{bmatrix} 1 & & & \\ & \mathbf{I} & & \\ & & 1 & \\ -s_1 & \ldots & -s_{N-1} & 1 \end{bmatrix} = \mathbf{S}_0^{-1} \tag{23}$$

Hence, we obtain $\mathbf{L}^{-1}\mathbf{A}\mathbf{S}_0^{-1} = \mathbf{D}_R\mathbf{U}$ or $\mathbf{A} = \mathbf{L}\mathbf{D}_R\mathbf{U}\mathbf{S}_0$. Notice that for a complex matrix $\mathbf{A}$, if $det\mathbf{A} = \pm 1$, then $e^{i\theta}$ is 1 or -1, and the TERM factor matrices can also be real.

Having gotten a TERM factorization, we can easily obtain the corresponding SERM factorization by the following way

$$\begin{aligned} \mathbf{A} &= \mathbf{L}\mathbf{U} = (\mathbf{L}\mathbf{U}\mathbf{S}_1^{-1})\mathbf{S}_1 = (\mathbf{L}\mathbf{U}\mathbf{S}_1^{-1}\mathbf{S}_2^{-1})\mathbf{S}_2\mathbf{S}_1 = \ldots \\ &= (\mathbf{L}\mathbf{U}\mathbf{S}_1^{-1}\mathbf{S}_2^{-1}\ldots\mathbf{S}_{N-1}^{-1})\mathbf{S}_{N-1}\ldots\mathbf{S}_2\mathbf{S}_1 = \mathbf{S}_N\ldots\mathbf{S}_2\mathbf{S}_1 \end{aligned} \tag{24}$$

where the mth row of $\mathbf{S}_m$ is equal to the mth row of $\mathbf{L}\mathbf{U}\mathbf{S}_1^{-1}\mathbf{S}_2^{-1}\ldots\mathbf{S}_{m-1}^{-1}$, $m = 1, 2, 3, \ldots, N$. We can visualize how TERM and SERM work by Figure 4 and Figure 5.

5.2 TERM based on S.C Pei and J.J. Ding

(**A**) For an nonsingular $N \times N$ matrix $\mathbf{A}$, we can scale it by a constant σ such that

$$det(\sigma\mathbf{A}) = det(\mathbf{G}) = \pm 1, \tag{25}$$

where $\sigma = |det(\mathbf{A})|^{-1/N}$. Notice that, for most of the applications (such as filter design, spectrum analysis, etc), scaling does not affect the performance.

(**B**) Do permutation and sign-changing operations for $\mathbf{G}$

$$\mathbf{R} = \mathbf{D}_1\mathbf{P}\mathbf{G}\mathbf{Q}\mathbf{D}_2, \tag{26}$$

where $\mathbf{P}$, $\mathbf{Q}$ are any permutation matrix, $\mathbf{D}_1$, $\mathbf{D}_2$ are diagonal matrices which diagonal term are ± 1.

(**C**) Then we do triangular matrix decomposition. First, we find $\mathbf{L}_1$ such that $\mathbf{H} = \mathbf{R}\mathbf{L}_1$ has the following form if $\mathbf{L}_1$ exist

$$\begin{aligned}
&\begin{bmatrix}
h_{11} & h_{12} & h_{13} & \dots & h_{1(N-1)} & h_{1N} \\
h_{21} & 1 & 0 & \dots & 0 & 0 \\
h_{31} & h_{32} & 1 & \dots & 0 & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
h_{(N-1)1} & h_{(N-1)2} & h_{(N-1)3} & \dots & 1 & 0 \\
h_{N1} & h_{N2} & h_{N3} & \dots & h_{N(N-1)} & 1
\end{bmatrix} \\
&= \mathbf{R}\begin{bmatrix}
1 & u_{12} & u_{13} & \dots & u_{1(N-1)} & u_{1N} \\
0 & 1 & u_{23} & \dots & u_{2(N-1)} & u_{2N} \\
0 & 0 & 1 & \dots & u_{3(N-1)} & u_{3N} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & \dots & 1 & u_{(N-1)N} \\
0 & 0 & 0 & \dots & 0 & 1
\end{bmatrix}
\end{aligned} \tag{27}$$

$$\begin{bmatrix} u_{1n} \\ u_{2n} \\ \vdots \\ u_{(n-2)n} \\ u_{(n-1)n} \end{bmatrix} = \begin{bmatrix} r_{21} & r_{22} & \dots & r_{2(n-1)} \\ r_{31} & r_{32} & \dots & r_{3(n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ r_{n1} & r_{n2} & \dots & r_{n(n-1)} \end{bmatrix}^{-1} \left(\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} - \begin{bmatrix} r_{2n} \\ r_{3n} \\ \vdots \\ r_{(n-1)n} \\ r_{nn} \end{bmatrix} \right) \tag{28}$$

(**D**) Let $\mathbf{T}_1 = \mathbf{L}_1^{-1}$. Note that $\mathbf{T}_1$ is also an upper-triangular matrix.

(**E**) Decompose $\mathbf{T}_2$ and $\mathbf{T}_3$ such that $\mathbf{H} = \mathbf{T}_3\mathbf{T}_2$. where

$$\mathbf{T}_2 = \begin{bmatrix}
1 & 0 & 0 & \dots & 0 & 0 \\
h_{21} & 1 & 0 & \dots & 0 & 0 \\
h_{31} & h_{32} & 1 & \dots & 0 & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
h_{(N-1)1} & h_{(N-1)2} & h_{(N-1)3} & \dots & 1 & 0 \\
h_{N1} & h_{N2} & h_{N3} & \dots & h_{N(N-1)} & 1
\end{bmatrix} \tag{29}$$

$$\mathbf{T}_3 = \begin{bmatrix}
\rho_1 & \rho_2 & \rho_3 & \dots & \rho_{N-1} & \rho_N \\
0 & 1 & 0 & \dots & 0 & 0 \\
0 & 0 & 1 & \dots & 0 & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & \dots & 1 & 0 \\
0 & 0 & 0 & \dots & 0 & 1
\end{bmatrix} \tag{30}$$

From **(C),(D),(E)**, we have decomposed $\mathbf{G}$ into three triangular matrices $\mathbf{T}_1$, $\mathbf{T}_2$, $\mathbf{T}_3$.

$$\mathbf{G} = \mathbf{P}^T\mathbf{D}_1\mathbf{T}_3\mathbf{T}_2\mathbf{T}_1\mathbf{D}_2\mathbf{Q}^T \tag{31}$$

5.3 How triangular matrix work

If $\mathbf{A} = \{a_{mn}\}$ is an upper TERM, the computational ordering of linear transform $\mathbf{y} = \mathbf{Ax}$ can be arranged to be top-down:

$$y_m = \begin{cases} j_m x_m + \mathcal{Q}(\sum_{n=m+1}^{N} a_{mn} x_n), & \text{if } m = 1, 2, \ldots, N-1 \\ j_N x_N, & \text{if } m = N \end{cases} \tag{32}$$

where $j_m = \{1, -1, i, -i\}$. It's inverse ordering is reversed

$$x_m = \begin{cases} \frac{y_N}{j_N}, & \text{if } m = N \\ \frac{1}{j_m}\left(y_m - \mathcal{Q}(\sum_{n=m+1}^{N} a_{mn} x_n)\right), & \text{if } m = N-1, N-2, \ldots, 1 \end{cases} \tag{33}$$

If $\mathbf{A}$ is a lower TERM, the computational ordering of linear transform $\mathbf{y} = \mathbf{AX}$ can be arranged to be bottom-up and it's inversion to be top-down

$$y_m = \begin{cases} j_m x_m + \mathcal{Q}(\sum_{n=1}^{m-1} a_{mn} x_n), & m = N, N-1, \ldots, 2 \\ j_1 x_1, & m = N \end{cases} \tag{34}$$

$$x_m = \begin{cases} \frac{y_1}{j_1}, & m = 1 \\ \frac{1}{j_m}\left(y_m - \mathcal{Q}(\sum_{n=1}^{m-1} a_{mn} x_n)\right), & m = 2, 3, \ldots, N \end{cases} \tag{35}$$

5.4 Minimizing Error

If we just find a TERM, the error $\mathbf{e}$ may too large to meet our **Goal IV**. Based on S.C. Pei and J.J. Ding[4], we can find mass TERM then choose min[$\mathbf{e}$] by following way

(A) Choose different permutation $\mathbf{P}$ and $\mathbf{Q}$. there are $2 \cdot N!$ choices.

(B) Choose different sign changed matrix $\mathbf{D}_1$ and $\mathbf{D}_2$. there are 2^{N+1} choices.

(C) Decompose $\mathbf{A}^T, \mathbf{A}^{-1}, (\mathbf{A}^T)^{-1}$ instead of $\mathbf{A}$.

(D) The order of upper and lower triangular matrices can be changed.

(E) Choose different integer-shift invariant operator.

(F) The diagonal entries of $\mathbf{D}_1$ and $\mathbf{D}_2$ can be $\pm 2^k$ such that $|det(\mathbf{D}_1)| = |det(\mathbf{D}_2)| = 1$.

Although there are many choices, this way has the same problem with prototype matrix. So S.C. Pei and J.J. Ding provide a way to transform big set to smaller set.

5.5 Reducing Searching Time

S.C. Pei and J.J. Ding provided a method to spend less time for searching solution. We know $\mathbf{G} = \mathbf{P}^T\mathbf{D}_1\mathbf{T}_3\mathbf{T}_2\mathbf{T}_1\mathbf{D}_2\mathbf{Q}^T$. Now let $\mathbf{y} = \mathbf{G}x$ $\hat{\mathbf{T}}_1, \hat{\mathbf{T}}_2, \hat{\mathbf{T}}_3$ are binary approximation of $\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3$, respectively. According to section 5.3, $\hat{\mathbf{y}} = \mathbf{P}^T\mathbf{D}_1\{\hat{\mathbf{T}}_3[\hat{\mathbf{T}}_2(\hat{\mathbf{T}}_1\mathbf{D}_2\mathbf{Q}^T x + \mathbf{\Delta}_1) + \mathbf{\Delta}_2] + \mathbf{\Delta}_3\}$ where $\mathbf{\Delta}_1, \mathbf{\Delta}_2, \mathbf{\Delta}_3$ are $N \times 1$ vectors. The difference between $\hat{\mathbf{y}}$ and $\mathbf{z}$ is:

$$\begin{aligned} \hat{\mathbf{y}} - \mathbf{y} \;\approx\; & \mathbf{P}^T\mathbf{D}_1\mathbf{T}_3\mathbf{T}_2\mathbf{\Delta}_1 + \mathbf{P}^T\mathbf{D}_1\mathbf{T}_3\mathbf{\Delta}_2 + \mathbf{P}^T\mathbf{D}_1\mathbf{\Delta}_3 \\ + \;& \mathbf{P}^T\mathbf{D}_1\mathbf{T}_3\mathbf{T}_2(\hat{\mathbf{T}}_1 - \mathbf{T}_1)\mathbf{Qx} + \mathbf{P}^T\mathbf{D}_1\mathbf{T}_3(\hat{\mathbf{T}}_2 - \mathbf{T}_2)\mathbf{T}_1\mathbf{Qx} \\ + \;& \mathbf{P}^T\mathbf{D}_1(\hat{\mathbf{T}}_3 - \mathbf{T}_3)\mathbf{T}_2\mathbf{T}_1\mathbf{Qx} \end{aligned} \tag{36}$$

When we let $\hat{\mathbf{T}}_i$ and $\mathbf{T}_i$ closed enough, then $\hat{\mathbf{y}} - \mathbf{y} \approx \mathbf{P}^T\mathbf{D}_1\mathbf{T}_3\mathbf{T}_2\mathbf{\Delta}_1 + \mathbf{P}^T\mathbf{D}_1\mathbf{T}_3\mathbf{\Delta}_2 + \mathbf{P}^T\mathbf{D}_1\mathbf{\Delta}_3$. If we let $\mathbf{T}_2$ and $\mathbf{T}_3$ small enough, this solution is sub-optimal.

How to minimize $\mathbf{T}_2$ and $\mathbf{T}_3$? The following is an example

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \tag{37}$$

Find a column permutation $\mathbf{Q}_1$ such that the summation of column $\mathbf{c}_i$ are $\sum_{j=1}^{3} |a_{ji}|$, choose the minimum of c_i, assume is c_2, then

$$\mathbf{A}^{(1)} = \begin{pmatrix} a_{12} & a_{11} & a_{13} \\ a_{22} & a_{21} & a_{23} \\ a_{32} & a_{31} & a_{33} \end{pmatrix} \tag{38}$$

Let

$$\mathbf{L}_1 = \begin{pmatrix} 1 & l_{12} & l_{13} \\ 0 & 1 & l_{23} \\ 0 & 0 & 1 \end{pmatrix} \tag{39}$$

$\mathbf{A}^{(2)} = \mathbf{A}^{(1)}\mathbf{L}_1$.Fix column 1, test all row and column permutation(except column 1) and sign changed term($\mathbf{D}_{1(22)}$ and $\mathbf{D}_{2(22)}$), find a permutation $\mathbf{P}_2$ and $\mathbf{Q}_2$ such that $\mathbf{c}_2^{(2)} = \sum_{j=1}^{3} |\mathbf{A}_{j2}^{(2)}|$ is minimum. Assume when a_{33} is at $\mathbf{A}_{22}^{(1)}$, $\mathbf{c}_2^{(2)}$ is minimum. Then

$$\mathbf{A}^{(2)} = \begin{pmatrix} a_{12} & a_{13} & a_{11} \\ a_{32} & a_{33} & a_{31} \\ a_{22} & a_{23} & a_{21} \end{pmatrix} \tag{40}$$

Fix column 1 and column 2, by the same way, find a permutation satisfying the condition. Finally, we get $\mathbf{P}$, $\mathbf{Q}$, $\mathbf{D}_1$, $\mathbf{D}_2$, and use these parameters to perform the TERM.

5.6 Reducing the Number of Time Cycles

What is the meaning of the number of time cycle? Take an example:

$$y = x_1 + 2x_2 + 3x_3 \tag{41}$$

where $x_i, \quad 1 \leq i \leq 3$ is input, y is output. The number of time cycle of this case is 2. Another example:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{42}$$

Because y_1 and y_2 is computed parallel, the number of time cycle is 1.

For $N \times N$ matrix, the time cycle is $N-1$ if we implement it directly. If we implement integer transform of the form (31) directly, the time cycle will be $3N - 3 + 2 = 3N - 1$[6]. This is three times than direct implementation. If we use another implementation, we can reduce the time cycle greatly.

[6]The time cycle of permutation is 1, the time cycle of sign change is 0

The key point is treat T_2 and T_3 simultaneously, then use binary tree to deal with T_3, For $N = 4$, let $\mathbf{y} = \mathbf{T}_3\mathbf{x}$, $\mathbf{z} = \mathbf{T}_2\mathbf{y}$, $\mathbf{w} = \mathbf{T}_1\mathbf{z}$.

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ - \\ z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} = \begin{bmatrix} * & 1 & a & b & c \\ * & 0 & 1 & d & e \\ * & 0 & 0 & 1 & f \\ * & 0 & 0 & 0 & 1 \\ - & - & - & - & - \\ 1 & 0 & 0 & 0 & * \\ g & 1 & 0 & 0 & * \\ h & i & 1 & 0 & * \\ j & k & l & 1 & * \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ - \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \tag{43}$$

Cycle 0: (Initialization)
$y_1 = cx_4$, $y_2 = ex_4$, $y_3 = fx_4$, $y_4 = x_4$. Delete x_4, set $z_4 = y_4$ in the position of x_4.
Cycle 1:
$y_1 += bx_3$, $y_2 += dx_3$, $y_3 += x_3$. Delete x_3, set $z_3 = y_3$ in the position of x_3.
Cycle 2:
$y_1 += ax_2$, $y_2 += x_2$, $z_4 += ly_3$. Delete x_2, set $z_2 = y_2$ in the position of x_2.
Cycle 3:
$y_1 += x_1$, $z_4 += ky_2$, $z_3 += iy_2$. Delete x_1, set $z_1 = y_1$ in the position of x_1.
Cycle 4:
$z_4 += jy_1$, $z_3 += hy_1$, $z_2 += gy_1$.

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} = \begin{bmatrix} 1 & m & n & p \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} \tag{44}$$

Cycle 0: (Initialization)
$w_1 = pz_4$, $w_2 = z_2$, $w_3 = z_3$, $w_4 = z_4$. Delete z_4.
Cycle 1:
$w_1 += nz_3$, $\sigma_1 = z_1 + mz_2$. Delete z_1, z_2, z_3.
Cycle 2:
$w_1 += \sigma_1$, Delete σ_1.

For $N \times N$ matrix, the time cycle is $N + 2 + \log_2 N$.

6 Conclusions

We have given an overview for the effort of contemporary researchers. We have two directions to choose, one is to find a new method to meet **Goal I V**, and the way is nice than before; another way is to improve S.C. Pei and J.J. Ding's searching algorithm.

References

[1] Anil K. Jain, "A Sinusoidal Family of Unitary Transforms," *IEEE trans. pattern anal. mach. intell*, vol. PAMI-1, pp. 356-365, Oct. 1979.

[2] S. C. Chan and P. M. Yiu, "An Efficient Multiplierless Approximation of the Fast Fourier Transform Using Sum-of-Powers-of-Two(SOPOT) Coefficients," *IEEE signal processing letters*, vol. 9, no. 10, pp. 322-325, 2002.

[3] WK Cham, PhD,"Development of Integer Cosine Transforms by the Principle of Dyadic Symmetry,"*Proc. Inst. Elect. Eng.*,pt. 1, vol. 136, no. 4, pp. 276-282, Aug. 1989.

[4] S. C. Pei, J. J. Ding,"General Reversible Integer Transform Conversion Using the Triangular Matrix Scheme."

[5] M. D. Adams, F. Koosentini, R. K. Ward,"Generalized S Transform," *IEEE Trans. Signal Processing*, vol. 50, no. 11, pp. 2831-2842, Nov. 2002.

[6] S. C. Pei, J. J. Ding,"Improved Reversible Integer Transform," *Circuits and Systems, 2006. ISCAS 2006. Proceedings.*, 2006 IEEE International Symposium on 21-24 May 2006 Page(s):4 pp. 1091-1094.

[7] Trac D. Tran,"Fast Multiplierless Approximation of the DCT," *Proc. 33rd Annu. Conf. Inform. Sci. Syst.*, pp. 933-938, Mar. 1999.

[8] S. Oraintara, Y. J. Chen, T. Q. Nguyen,"Integer Fast Fourier Transform," *IEEE Trans. Signal Processing*, pp. 607-618, Mar. 2002.

[9] P. Hao, Q. Shi,"Matrix Factorizations for Reversible Integer Mapping," *IEEE Trans. Signal Processing*, vol. 49, pp. 2314-2324, Oct. 2001.

[10] G. Strang,"The Discrete Cosine Transform," *SIAM Review*, vol. 41 no. 1, pp. 135-147, Mar. 1999.