Design of high-performance fixed-point transforms using the common factor method

Arianne T. Hinds*
Ricoh InfoPrint Solutions Company, 6300 Diagonal Highway, Boulder, CO, USA 80301

ABSTRACT

Fixed-point implementations of transforms such as the Discrete Cosine Transform (DCT) remain as fundamental building blocks of state-of-the-art video coding technologies. Recently, the 16x16 DCT has received focus as a transform suitable for the high efficiency video coding project currently underway in the Joint Collaboration Team – Video Coding. By its definition, the 16x16 DCT is inherently more complex than transforms of traditional sizes such as 4x4 or 8x8 DCTs. However, scaled architectures such as the one employed in the design of the 8x8 DCTs specified in ISO/IEC 23002-2 can also be utilized to mitigate the complexity of fixed-point approximations of higher-order transforms such as the 16x16 DCT. This paper demonstrates the application of the Common Factor method to design two scaled implementations of the 16x16 DCT. One implementation can be characterized by its exceptionally low complexity, while the other can be characterized by its relatively high precision. We review the Common Factor method as a method to arrive at fixed-point implementations that are optimized in terms of complexity and precision for such high performance transforms.

Keywords: Discrete Cosine Transform, fixed-point approximations, MPEG, JPEG, ISO/IEC 23002-2

1. INTRODUCTION

Transforms such as the Discrete Cosine Transform continue to serve as basic elements of video coding systems as demonstrated by the use of these transforms in proposals recently submitted in response to the call for proposals for the High Efficiency Video Coding project currently underway in the Joint Collaboration Team for Video Coding1,2. In particular, transforms of sizes larger than 4x4 or 8x8, especially 16x16 and 32x32 are being considered because of their increased applicability to the decorrelation of high resolution video signals. In practical implementations, these transforms are implemented in fixed-precision for which the most challenging task is that of finding efficient approximations of the underlying, typically irrational, constants defining the transforms, e.g. cosines and sines.

A common method of approximating these irrational constants is to approximate each constant separately as a dyadic rational. That is, each constant \( \theta_i \) is approximated as:

\[
\theta_i \approx \frac{P_i}{2^K}
\]

in which \( P_i \) and \( K \) are both in \( \mathbb{N} \). Implementations of these approximations can then be designed with multiplication operations and bit-wise shift operations such as

\[
x \ast \theta_i \approx (x \ast P_i) \gg K
\]

or optionally using a multiplierless approach in which the binary composition of \( P_i \) is used to define bit-wise shift and addition operations equivalent to the multiplication by the same integer \( P_i \) in (2). That is:

\[
x \ast \theta_i \approx ((x \ll a) + (x \ll b) + \cdots + (x \ll n)) \gg K
\]

or

\[
2^a + 2^b + \ldots + 2^n = P_i
\]

* arianne.hinds@infoprint.com; phone 1 720 663-3565; fax 1720 663-4507; infoprint.com
Considerations with regard to the complexity of these dyadic approximations include the size of the parameter \( K \), for which larger values can require more gates in hardware implementations and wider registers for software implementations; and the value of the parameter \( P_i \) for which larger values impact the growth of the data in the transform or the number of elementary operations required for multiplierless implementations. Yet another consideration in the fixed-point implementation of these transforms is the degree to which the transform approximation performs to the mathematical definition of the transform. That is, how well the implementation approximates the mathematical definition of the transform, which in turn is directly related to the selection of the fixed-point approximations chosen for each of the irrational constants. This latter point brings us to a fundamental cross-road for compromise in the design of practical implementations of the DCT that we will discuss later in this paper.

The remainder of this paper is organized as follows. Section 2 provides background material on the DCT and the development of Integer Cosine Transforms. Section 3 reviews the common factor method for finding integer approximations of irrational constants used in transforms, and adopted in the implementation of DCT in ISO/IEC 23002-2. Section 4 presents two new implementations of the 16x16 DCT based on the 16x16 factorization presented in using the common factor method to approximate the constants. Section 5 provides our conclusions.

2. DISCRETE AND INTEGER COSINE TRANSFORMS

The DCT is widely regarded as the best suboptimal orthogonal transform for the purposes of energy compaction, especially with regard to natural images. That is, its ability to concentrate the energy of natural images into a relatively small number of transform coefficients closely approximates the optimal Karhunen-Loeve transform (KLT). The DCT is also a unitary transform which means that it is comprised of orthonormal basis functions (the norm of each of its basis vectors is one) and its inverse is its transpose. Of the four types of DCTs described in, the Type-2 DCT is the most widely deployed in popular still image and video standards such as JPEG, MPEG 1, MPEG 2, H.263, and MPEG 4 and also serves as the definition for the fixed-point implementation that approximates it in MPEG C (ISO/IEC 23002-2). A two-dimensional Type-2 DCT of an M-by-N matrix \( f \) is defined as:

\[
F(u, v) = c(u)c(v) \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \cos \left( \frac{2x + 1}{2M} \right) \cos \left( \frac{2y + 1}{2N} \right), 0 \leq x \leq M - 1, 0 \leq y \leq N - 1
\]  
(5)

where \( c(u) = \begin{cases} \frac{1}{\sqrt{M}}, & u = 0 \\ \sqrt{\frac{2}{M}}, & 1 \leq u \leq M - 1 \end{cases} \) and \( c(v) = \begin{cases} \frac{1}{\sqrt{N}}, & v = 0 \\ \sqrt{\frac{2}{N}}, & 1 \leq v \leq N - 1 \end{cases} \)

\( f(x, y) \) is the input matrix of sample values, and \( F(u, v) \) is the resulting M-by-N matrix of transform coefficients.

The matrix \( f(x, y) \) can be perfectly reconstructed from \( F(u, v) \) by application of the corresponding inverse DCT (IDCT) which is defined as follows:

\[
f(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} c(u)c(v)F(u, v) \cos \left( \frac{2x + 1}{2M} \right) \cos \left( \frac{2y + 1}{2N} \right), 0 \leq x \leq M - 1, 0 \leq y \leq N - 1
\]  
(6)

The DCT is a separable transform. For example, by simply rearranging the terms in (5), we arrive at (7)

\[
F(u, v) = c(u) \sum_{x=0}^{M-1} f(x, y) \cos \left( \frac{2x + 1}{2M} \right) c(v) \sum_{y=0}^{N-1} f(x, y) \cos \left( \frac{2y + 1}{2N} \right)
\]  
(7)

which upon inspection illustrates that the 2-D transform can be separated into two single 1-D transforms that are performed first along one axis (such as the columns in a block of video samples), and then along the other axis (the rows of the output block resulting from the column transform). Expanding the cosine kernel for a single dimension reveals...
the dyadic symmetries of a DCT of order $2N$. For example, the cosine matrix for a 1-D 16x16 DCT is shown in Figure 1
where $c_n$ denotes the value $\cos \frac{\pi n}{32}$.

<table>
<thead>
<tr>
<th>C0</th>
<th>C0</th>
<th>C0</th>
<th>C0</th>
<th>C0</th>
<th>C0</th>
<th>C0</th>
<th>C0</th>
<th>C0</th>
<th>C0</th>
<th>C0</th>
<th>C0</th>
<th>C0</th>
<th>C0</th>
<th>C0</th>
<th>C0</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>C3</td>
<td>C5</td>
<td>C7</td>
<td>C9</td>
<td>C11</td>
<td>C13</td>
<td>C15</td>
<td>-C15</td>
<td>-C13</td>
<td>-C11</td>
<td>-C9</td>
<td>-C7</td>
<td>-C5</td>
<td>-C3</td>
<td>-C1</td>
</tr>
<tr>
<td>C2</td>
<td>C6</td>
<td>C10</td>
<td>-C14</td>
<td>-C10</td>
<td>-C6</td>
<td>-C2</td>
<td>-C2</td>
<td>-C6</td>
<td>-C10</td>
<td>-C14</td>
<td>C10</td>
<td>C6</td>
<td>C2</td>
<td>C10</td>
<td>C6</td>
</tr>
<tr>
<td>C3</td>
<td>C9</td>
<td>C15</td>
<td>-C11</td>
<td>-C5</td>
<td>-C1</td>
<td>-C7</td>
<td>-C13</td>
<td>C13</td>
<td>C7</td>
<td>C1</td>
<td>C5</td>
<td>C11</td>
<td>C15</td>
<td>-C9</td>
<td>-C3</td>
</tr>
<tr>
<td>C4</td>
<td>C12</td>
<td>-C12</td>
<td>-C4</td>
<td>-C4</td>
<td>-C12</td>
<td>C12</td>
<td>C4</td>
<td>C4</td>
<td>C12</td>
<td>-C12</td>
<td>-C4</td>
<td>-C4</td>
<td>C12</td>
<td>C12</td>
<td>C4</td>
</tr>
<tr>
<td>C5</td>
<td>C15</td>
<td>-C7</td>
<td>-C3</td>
<td>-C13</td>
<td>C9</td>
<td>C1</td>
<td>C11</td>
<td>-C11</td>
<td>-C1</td>
<td>-C9</td>
<td>C13</td>
<td>C3</td>
<td>C7</td>
<td>-C15</td>
<td>-C5</td>
</tr>
<tr>
<td>C6</td>
<td>-C14</td>
<td>-C2</td>
<td>-C10</td>
<td>C10</td>
<td>C2</td>
<td>C14</td>
<td>-C6</td>
<td>-C6</td>
<td>C14</td>
<td>C2</td>
<td>C10</td>
<td>-C10</td>
<td>-C2</td>
<td>-C14</td>
<td>C6</td>
</tr>
<tr>
<td>C7</td>
<td>-C11</td>
<td>-C3</td>
<td>C15</td>
<td>C1</td>
<td>C13</td>
<td>-C5</td>
<td>-C9</td>
<td>C9</td>
<td>C5</td>
<td>-C13</td>
<td>-C1</td>
<td>-C15</td>
<td>C3</td>
<td>C11</td>
<td>-C7</td>
</tr>
<tr>
<td>C8</td>
<td>-C8</td>
<td>-C8</td>
<td>C8</td>
<td>-C8</td>
<td>-C8</td>
<td>C8</td>
<td>C8</td>
<td>-C8</td>
<td>-C8</td>
<td>C8</td>
<td>C8</td>
<td>-C8</td>
<td>-C8</td>
<td>C8</td>
<td>C8</td>
</tr>
<tr>
<td>C9</td>
<td>-C5</td>
<td>-C13</td>
<td>C1</td>
<td>-C15</td>
<td>-C3</td>
<td>C11</td>
<td>C7</td>
<td>-C7</td>
<td>-C11</td>
<td>C3</td>
<td>C15</td>
<td>-C1</td>
<td>C13</td>
<td>C5</td>
<td>-C9</td>
</tr>
<tr>
<td>C10</td>
<td>-C2</td>
<td>C14</td>
<td>C6</td>
<td>-C6</td>
<td>-C14</td>
<td>C2</td>
<td>-C10</td>
<td>-C10</td>
<td>C2</td>
<td>-C14</td>
<td>C6</td>
<td>C6</td>
<td>C14</td>
<td>-C2</td>
<td>C10</td>
</tr>
<tr>
<td>C11</td>
<td>-C1</td>
<td>C9</td>
<td>C13</td>
<td>-C3</td>
<td>C7</td>
<td>C15</td>
<td>-C5</td>
<td>-C5</td>
<td>-C15</td>
<td>-C7</td>
<td>C3</td>
<td>-C13</td>
<td>-C9</td>
<td>C1</td>
<td>-C11</td>
</tr>
<tr>
<td>C12</td>
<td>-C4</td>
<td>C4</td>
<td>-C12</td>
<td>-C12</td>
<td>C4</td>
<td>-C4</td>
<td>C12</td>
<td>C12</td>
<td>-C4</td>
<td>C4</td>
<td>-C12</td>
<td>-C12</td>
<td>C4</td>
<td>-C4</td>
<td>C12</td>
</tr>
<tr>
<td>C13</td>
<td>-C7</td>
<td>C1</td>
<td>-C5</td>
<td>C11</td>
<td>C15</td>
<td>-C9</td>
<td>C3</td>
<td>C3</td>
<td>C9</td>
<td>C15</td>
<td>-C11</td>
<td>C5</td>
<td>-C1</td>
<td>C7</td>
<td>-C13</td>
</tr>
<tr>
<td>C14</td>
<td>-C10</td>
<td>C6</td>
<td>-C2</td>
<td>C2</td>
<td>-C6</td>
<td>C10</td>
<td>-C14</td>
<td>-C14</td>
<td>C10</td>
<td>-C6</td>
<td>C2</td>
<td>-C2</td>
<td>C6</td>
<td>-C10</td>
<td>C14</td>
</tr>
<tr>
<td>C15</td>
<td>-C13</td>
<td>C11</td>
<td>-C9</td>
<td>C7</td>
<td>-C5</td>
<td>C3</td>
<td>-C1</td>
<td>C1</td>
<td>-C3</td>
<td>C5</td>
<td>C7</td>
<td>C9</td>
<td>C11</td>
<td>C13</td>
<td>C15</td>
</tr>
</tbody>
</table>

Figure 1. Cosine matrix for 16x16 DCT where $c_n = \cos \frac{\pi n}{32}$.

Fundamentally, the above cosine values are difficult to approximate in fixed precision rendering fixed point implementations subject to rounding error in practical applications. Rounding errors, in turn, can introduce enough error into the computations such that the transform is no longer orthogonal. That is, the original matrix $\mathbf{f}$ which serves as input into (5) cannot be perfectly reconstructed by (6) with the computations in one or both of the equations no longer mathematically precise. Early efforts to address the problem for how to implement the DCT, in particular, the 8x8 DCT (which suffers the same difficulties with irrational constants)$^{12}$, led to two approaches: first, to approximate it within some known tolerance of precision such as that specified by ANSI/IEEE Standard 1180$^{13}$ – now withdrawn and replaced for use in MPEG video technologies by MPEG-C Pt. 1 (ISO/IEC 23002-1)$^{14}$ and then second, to preserve the relative magnitudes, relationships, and symmetries of the cosines in the cosine matrix to create a reversible integer cosine matrix$^{15-18}$.

The first approach permitted implementers, in particular those designing video coding systems for MPEG 1 Pt. 2, MPEG 2 Pt.2, and MPEG 4 Pt.2, the freedom to create designs optimized for various hardware or software platforms. That is, so long as a given implementation met the precision criteria specified in IEEE 1180 (and now in ISO/IEC 23002-1), an implementation was considered to be conformant to the relevant video specification. Experience ultimately proved that this approach led to the phenomenon known as “drift”$^{19-21}$, which was manifested by mismatched implementations of the IDCT in encoders and decoders$^{22}$. For example, the precision specification permits the reconstructed pixels values to have a maximum error of $\pm 1$ relative to output of an ideal reference IDCT. While one conformant IDCT implementation could meet this tolerance by reconstructing errors with errors biased toward negative values, another conformant IDCT could similarly meet the tolerance specification with errors biased toward positive values. The net effect of these two IDCTs used within the context of a single encoder-decoder system would introduce a net error with a magnitude of two. Experiments conducted in conjunction with the development of ISO/IEC 23002-2 showed that this
error could dramatically accumulate across inter-predicted frames and would be especially exacerbated in static regions of consecutive video frames\textsuperscript{23-26.}

The premise of the second and latter approach was to specify an integer based transform that could be exactly specified and (relatively) easily implemented, in the decoder and hence, implemented to produce the exact mathematical equivalent in corresponding encoders. Moreover, these transforms were designed to ensure that the forward and inverse transform are fully reversible, and hence completely orthogonal. However, orthogonality is achieved at the expense of no longer utilizing an exact cosine kernel such as in the original DCT. With an ICT, the kernel values are implemented entirely with integers or dyadic rationals with which there is no error associated with rounding to integers. In essence, a compromise is made to exchange the precision of the cosine basis functions of the DCT for basis functions that are similar, but not numerically equivalent, to the original cosine basis functions in order to preserve the property of perfect reconstruction.

3. COMMON FACTOR METHOD

The common factor method\textsuperscript{27-30} introduces the use of another constant, a common factor $\xi$ in $\mathbb{R}$, to aid in the approximation of sets of irrational constants that occur in linear transforms such as in the DCT. That is, rather than directly approximating the set of constants $\{\theta_1, ..., \theta_m\}$ with independent dyadic rationals as in (1), we instead approximate them collectively as a set using the common factor $\xi$:

$$\theta_i \approx \xi p_i, \ldots, \theta_m \approx \xi p_m$$

where as in (1) $P_1, ..., P_m$ are in $\mathbb{N}$.

Assume, for example that the set $\{\theta_0, ..., \theta_{m-1}\}$ is the set of $m$ constants used in the basis function defined in the second row of the cosine matrix in Figure 1 to compute $F(1)$, where $m = 8$, as shown in (9) and $Cn$ denotes the value $\cos \frac{\pi n}{32}$.

$$\{\theta_0 = C1, \theta_1 = C3, \theta_2 = C5, \theta_3 = C7, \theta_4 = C9, \theta_5 = C11, \theta_6 = C13, \theta_7 = C15\}$$

Then $F(1)$ in the 1-D computation for the 16x16 DCT can be expressed as

$$F(1) = C1(x_0 - x_{15}) + C3(x_1 - x_{14}) + C5(x_2 - x_{13}) + C7(x_3 - x_{12}) + C9(x_4 - x_{11}) + C11(x_5 - x_{10}) + C13(x_6 - x_9) + C15(x_7 - x_8)$$

where $(x_0, x_1, ..., x_{15})$ is the input vector of samples from $f(x, y)$.

Assume now that we wish to approximate each of the cosines in (10) with integers $P_1, ..., P_m$ such that each integer can be characterized by some metric relative to the complexity of its implementation either in terms of 1) the bit-depth required for the approximation or 2) the number of addition operations required for a multiplierless factorization as described in (3) and (4). Then, by substitution of the cosines in (10) with the expressions in (8) and (9), equation (10) can be recast into the form shown in (11):

$$F(1) = \xi P_0(x_0 - x_{15}) + \xi P_1(x_1 - x_{14}) + \xi P_2(x_2 - x_{13}) + \xi P_3(x_3 - x_{12}) + \xi P_4(x_4 - x_{11}) + \xi P_5(x_5 - x_{10}) + \xi P_6(x_6 - x_9) + \xi P_7(x_7 - x_8)$$

In the computation of the 16x16 DCT, the parameter $\xi$ is subsequently absorbed into a scaling stage which will illustrated later in this section.

3.1 Common factor method applied to each of the basis functions of the 16x16 DCT

Note that the same process of identifying a unique parameter $\xi$ can be applied to each of the rows in Figure 1, but in practical implementations such as in the 16x16 described in \textsuperscript{32}, sets of constants that are used concurrently, such as those enumerated in (9) for the computation of $F(1)$ can optionally utilize the same approximations and factor $\xi$. For example, the set of constants used for $F(1)$ is the same set of constants used for each of the odd basis functions shown in Figure 1. That is, the equations for $F(1)$, $F(3)$, $F(5)$, $F(7)$, $F(9)$, $F(11)$, $F(13)$, and $F(15)$ in the 16x16 DCT each use the same set of constants as in (9) and can therefore be recast into the same form as shown in (11) and computed using the same parameter $\xi$ and corresponding integers $P_0, ..., P_7$. 
Similarly, a second set of constants, and correspondingly a second parameter \( \xi \), can be identified for the basis functions required to compute \( F(2) \), \( F(6) \), \( F(10) \), and \( F(14) \) in the 16x16 DCT shown in Figure 1. For clarity, let us denote the parameter \( \xi \) and corresponding set of integers \( P_0, \ldots, P_7 \) used in (11) for the odd basis functions in Figure 1 as \( \xi_0 \) and the set \( P_{odd} = \{ P_0, \ldots, P_7 \} \) as the set of integers used to approximate the cosines for these odd basis functions. Then, a second set of cosines can be identified for the basis functions required to compute \( F(2) \), \( F(6) \), \( F(10) \), and \( F(14) \) in the 16x16 DCT shown in Figure 1 as the set of four cosine values consisting of \{\( C2, C6, C10, C14 \)\}. Let us assume that we will approximate this second set of cosines with the parameter \( \xi_1 \) and a second set of integers \( P_{Even} = \{ P_0, \ldots, P_3 \} \).

Then the computation of \( F(2) \) can be cast as shown in (12):

\[
F(2) = \xi_1 P_0 (x_0 - x_7 + x_0 - x_{15}) + \xi_1 P_1 (x_1 - x_6 - x_9 + x_{14}) + \xi_1 P_2 (x_2 - x_5 - x_{10} + x_{12}) + \xi_1 P_3 (x_3 - x_4 - x_{11} + x_{12})
\]

where

\[
\xi_1 P_0 \approx C2, \quad \xi_1 P_1 \approx C6, \quad \xi_1 P_2 \approx C10, \text{ and } \xi_1 P_3 \approx C14
\]

Likewise, the computations for \( F(6) \), \( F(10) \), and \( F(14) \) can be recast as shown in (12) using the same approximations \( \xi_1 \) and \( P_{Even} = \{ P_0, \ldots, P_3 \} \) in (13).

The third set of irrational constants for the 16x16 DCT can also be approximated using the common factor parameters \( \xi_2 \) and integers \( P_{Even} = \{ P_0, P_1 \} \) for \( F(4) \) and \( F(12) \) as follows:

\[
\xi_2 P_0 \approx C4, \quad \xi_2 P_1 \approx C12
\]

Thus the computation of equation \( F(4) \) can be recast as:

\[
F(4) = \xi_2 P_0 (x_0 - x_3 - x_4 + x_5 + x_6 - x_{11} - x_{12} + x_{15}) + \xi_2 P_1 (x_1 - x_2 - x_5 + x_6 + x_9 - x_{10} + x_{13} + x_{14})
\]

Similarly, the computation of \( F(12) \) can be recast using the same approximations in (14).

### 3.2 Selection of approximations for common factor method applied to 16x16 DCT

The selection of each parameter \( \xi \) and corresponding set of integers \( P_i, \ldots, P_m \) for each set of constants can be cast into a multi-criteria optimization problem whereby in one dimension, we wish to find integers meeting some criteria relative to complexity (i.e., the number of addition operations required to implement each of the approximations \( P_i, \ldots, P_m \)), and in another dimension, we wish to minimize the error introduced into the approximations for the final calculation of each \( F(n) \). In this paper, we shall assume that we wish to optimize the number of addition operations required to implement each set of integers \( P_i, \ldots, P_m \) in order to achieve low complexity multiplierless implementations of the 16x16 DCT. Each candidate solution for each of the sets of the approximations can then be characterized by the following tuple:

\[
\{ \xi_n, \text{operation count, maximum absolute error}, P_i, \ldots, P_m \}
\]

where for ease of notation we shall denote the operation count (in terms of addition operations) as the primary figure of merit (FOM) and maximum error occurring in the set of approximations as \( \max \delta \).

Using a branch and bound search method, we can arrive at a list of candidate solutions for each of the three sets of irrational constants used in the 16x16 DCT. Two candidate tuples for the parameter \( \xi_0 \), integers \( P_{odd} = \{ P_0, \ldots, P_7 \} \), and constants \{\( C1, C3, C5, C7, C9, C11, C13, C15 \)\} used for the odd basis functions in the 16x16 DCT are shown below in (17) and (18):

\[
\{ \xi_0 = 0.0318256099610648, \text{FOM} = 8, \max \delta = 9.19E - 3, P_0 = 31, P_1 = 30, P_2 = 28, P_3 = 24, P_4 = 20, P_5 = 15, P_6 = 9, P_7 = 3 \}
\]

\[
\{ \xi_0 = 0.0034505624833603, \text{FOM} = 12, \max \delta = 1.42E - 3, P_0 = 288, P_1 = 277, P_2 = 256, P_3 = 224, P_4 = 184, P_5 = 137, P_6 = 84, P_7 = 28 \}
\]
Given that these solutions were derived using a multi-criteria approach implemented with a branch and bound search algorithm, the solutions in (17) and (18) can be described as Pareto-optimal in the sense that there is no solution with a given cost metric (i.e. the number of additions required for the approximations) with a lower maximum absolute error. The flexibility of this approach is that the solutions in either (17) or (18) can be used to approximate the cosines in the odd basis functions. Whereas the approximations in (17) can be characterized by the relatively low complexity in terms of the addition operations required to implement them (only 8 operations are needed for the entire set), the error incurred for these low complexity solutions is approximately one order of magnitude larger than for the set of approximations described in (18). Nevertheless, the approximations in (18) require a total of 12 addition operations; 33% more addition operations than needed for the approximations in (17).

Candidate solutions for the remaining two sets of constants can be derived for the 16x16 DCT as follows:

For the set of constants used in the approximations: \( \xi_1 P_0 \approx C2, \quad \xi_1 P_1 \approx C6, \quad \xi_1 P_2 \approx C10, \) and \( \xi_1 P_3 \approx C14 \) required for the basis functions in F(2), F(6), F(10), and F(14), the candidate solutions in (19) or (20) can be used:

\[
\begin{align*}
\{\xi_1 & = 0.069169153058235, FOM = 3, \max_\delta = 1.241E - 2, P_0 = 14, P_1 = 12, P_2 = 8, P_3 = 3\} \\
\{\xi_1 & = 0.001435151521912351, FOM = 5, \max_\delta = 1.435E - 3, P_0 = 160, P_1 = 136, P_2 = 96, P_3 = 32\}
\end{align*}
\]

As in the approximations for the odd basis functions, the solutions above are Pareto-optimal. Where (19) utilizes only 3 addition operations to approximate the set of four cosines, it incurs an error that is an order of magnitude larger than the error occurred in (20). However, the operation count for the approximations in (20) is 5 addition operations, nearly twice the number of operations than those needed in (19).

Finally, for the set of constants in the approximations: \( \xi_2 P_0 \approx C4 \) and \( \xi_2 P_1 \approx C12 \) required for the basis functions in F(4) and F(12), the candidate solutions in (21) and (22) can be used:

\[
\begin{align*}
\{\xi_2 & = (0.186651852073 * \sqrt{2}), FOM = 1, \max_\delta = 9.379E - 3, P_0 = 2, P_1 = 5\} \\
\{\xi_2 & = (0.0115625041107168 * \sqrt{2}), FOM = 2, \max_\delta = 1.12E - 3, P_0 = 80, P_1 = 33\}
\end{align*}
\]

### 3.3 Common factors implemented in scaled architectures

It is well known that the DCT and other linear transforms can be implemented in scaled architectures as demonstrated by methods to mitigate the complexity of the final implementation of the complete 2-D transform. Both the DCT and the IDCT specified in ISO/IEC 23002-2 utilize such scaled architectures. Scaled architectures can further be exploited to absorb the factors \( \xi_n \) derived in the approximations for the common factor method.

Let \( T \) denote the matrix operator corresponding to a one-dimensional DCT of order 16, and \( f \) denote the 16x16 block of samples that we wish to decorrelate to produce the 16x16 matrix \( F \) consisting of the corresponding DCT coefficients. Then

\[
F = T^\top f T^T
\]

where \( T^\top \) denotes the transpose operator.

As demonstrated in the above examples for the common factor method, the matrix \( T \) can be factorized into a matrix of approximations \( A \) and a diagonal matrix \( D \) consisting of the factors \( \xi_n \) as follows:

\[
D = diag\{1.0, \xi_0, \xi_1, \xi_0, \xi_2, \xi_0, \xi_1, \xi_0, 1.0, \xi_0, \xi_1, \xi_0, \xi_2, \xi_0, \xi_1, \xi_0\}
\]

\[
T = DA
\]
\[ T^T = A^T D \]  

If we express \( D \) as a 16x16 matrix \( S \) then equations (25) and (26) can be rewritten as (27) and (28) respectively:

\[ T = S \otimes A \]  

and

\[ T^T = A^T \otimes S^T \]

where \( \otimes \) denotes the pointwise multiplication between two matrices of the same size. Then (23) can similarly be rewritten as

\[ F = S \otimes A f A^T \otimes S^T \]  

By application of the mixed-product property, the terms in (29) can be reorganized to (30):

\[ F = (A f A^T) \otimes (S \otimes S^T) \]  

which implies that the equation (30) can be separated into two major processes: 1) the matrix multiplications performed using the approximations in \( A \), followed by 2) a scaling stage in which the outputs of \( A f A^T \) are scaled by a pointwise multiplication operation with \( S \otimes S^T \).

4. APPROXIMATIONS OF ORDER-16 DCT USING COMMON FACTOR METHOD

We use the approximations derived above to create two implementations of the 16x16 DCT. Both implementations employ the 16x16 factorization described in \(^{32}\text{ (LLM)}\) with some of the multiplications by \( \sqrt{2} \) moved directly into the scaling matrix \( S \). The 16x16 DCT as presented by LLM is shown in Figure 2 utilizing rotation operations in stages 2, 3, 4, and 5. For the rotations performed in Stage 1, the following constants are approximated with the factor \( \xi_0 \) where

\[ Cn = \cos \frac{n\pi}{32} \text{ and } Sn = \sin \frac{n\pi}{32} \]

\[ \{C3, S3, C7, S7, C11, S11, C15, S15\} \]

with the observation that \( S3 = C13, S7 = C9, S11 = C5, \text{ and } S15 = C1 \). Hence, the same set of approximations derived for \( P_{\text{odd}} = \{P_o, ..., P_7\} \) can be employed. Moreover, the LLM factorization computes the outputs \( F(3), F(4), F(5), F(11), F(12), \) and \( F(13) \) using rotations by the angle \( \frac{12n\pi}{32} \), that is by the constants comprising the set \( \{C12, S12 = C2\} \). Hence, the approximations for \( P_{\text{Even}_2} = \{P_o, P_1\} \) can also be employed for \( F(3), F(4), F(5), F(11), F(12) \) and \( F(13) \). Finally, the outputs \( F(2), F(6), F(10), \) and \( F(14) \) are computed using rotations by the angles \( \frac{10n\pi}{32} \) and \( \frac{14n\pi}{32} \). These outputs can likewise be computed using the approximations for \( P_{\text{Even}_1} = \{P_o, ..., P_3\} \) noting that the constants \( \{C10, S10, C14, S14\} \) are used in this set where \( S10 = C6 \) and \( S14 = C2 \).

For both implementations, the scaling matrix \( S \) is computed from the diagonal matrix \( D \) as defined below:

\[ D = \text{diag}\{1.0, \xi_0, (\xi_1\sqrt{2}), (\xi_0\xi_2\sqrt{2}), (\xi_1\sqrt{2}), (\xi_0\xi_2\sqrt{2}), (\xi_0\xi_2\sqrt{2}), (\xi_0\xi_2\sqrt{2}), (\xi_0\xi_2\sqrt{2}), (\xi_0\xi_2\sqrt{2})\} \]
4.1 Low Complexity Implementation

The first implementation is described as a “low complexity” implementation because it uses each of the approximations derived above such that the smaller number of additions is used in the approximation matrix $A$ in (28). Table 1-3 summarizes the approximations for the each of the three sets of constants in the LLM factorization:

Table 1. Approximations used for Stage 1 rotations in low complexity implementation of 16x16 DCT

<table>
<thead>
<tr>
<th>Irrational Constants used in Stage 2 rotations</th>
<th>Integer Approximation</th>
<th>number of addition operations for integer approximation</th>
<th>$\xi_0$ to be used in scaling matrix S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\cos(7\pi/32)$</td>
<td>24&gt;&gt;4</td>
<td>1</td>
<td>(0.0318256099610648 * 16.0)</td>
</tr>
<tr>
<td>$\sin(7\pi/32)$</td>
<td>20&gt;&gt;4</td>
<td>1</td>
<td>(0.0318256099610648 * 16.0)</td>
</tr>
<tr>
<td>$\cos(11\pi/32)$</td>
<td>15&gt;&gt;4</td>
<td>1</td>
<td>(0.0318256099610648 * 16.0)</td>
</tr>
<tr>
<td>$\sin(11\pi/32)$</td>
<td>28&gt;&gt;4</td>
<td>1</td>
<td>(0.0318256099610648 * 16.0)</td>
</tr>
<tr>
<td>$\cos(3\pi/32)$</td>
<td>30&gt;&gt;4</td>
<td>1</td>
<td>(0.0318256099610648 * 16.0)</td>
</tr>
<tr>
<td>$\sin(3\pi/32)$</td>
<td>9&gt;&gt;4</td>
<td>1</td>
<td>(0.0318256099610648 * 16.0)</td>
</tr>
<tr>
<td>$\cos(15\pi/32)$</td>
<td>3&gt;&gt;4</td>
<td>1</td>
<td>(0.0318256099610648 * 16.0)</td>
</tr>
<tr>
<td>$\sin(15\pi/32)$</td>
<td>31&gt;&gt;4</td>
<td>1</td>
<td>(0.0318256099610648 * 16.0)</td>
</tr>
</tbody>
</table>
In terms of complexity, at most one addition operation is required for each of the multiplication operations for a total of 82 additions (or less if using the rotation implementation as described in LLM) required for all five stages in the low-complexity implementation. The scaling matrix S used to complete the scaling process is implemented with 10 bits of precision and rounded to the nearest integers as shown in Figure 3:

![Table 2. Approximations used for Stage 3 rotations in low complexity implementation of 16x16 DCT](image)

![Table 3. Approximations used for Stage 4 and Stage 5 rotations in low complexity implementation of 16x16 DCT](image)

With respect to the precision of the low complexity implementation, the resulting transform is nearly orthogonal as demonstrated by the computation of

\[ F = (AA^T) \otimes (S \otimes S^T) \div 2^{14} \]  

(33)
which by the mathematical definition of the DCT should be $I$. Figure 4 illustrates the result of the computation of (32) for the low-complexity implementation described above.

![Table 4. Approximations used for Stage 1 rotations in high-precision implementation of 16x16 DCT.](image)

**4.2 High precision implementation of 16x16 DCT**

We demonstrate the application of the high-precision approximations described in Section 3.1 to create another implementation of the 16x16 DCT based on LLM. As in the low-complexity implementation, the diagonal matrix $D$ used to scale the matrix $A$ is as defined in (31), but with the approximations for the high-precision implementation defined in Tables 4-6.

![Table 5. Approximations used for Stage 3 rotations in high-precision implementation of 16x16 DCT.](image)
Table 6. Approximations used for Stage 4 and Stage 5 rotations in low complexity implementation of 16x16 DCT

<table>
<thead>
<tr>
<th>Irrational Constants used in Stage 4 and Stage 5 rotations</th>
<th>Integer Approximation</th>
<th>number of addition operations for integer approximation</th>
<th>$\xi_2$ to be used in scaling matrix S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\cos(12 \pi/32)$</td>
<td>33&gt;&gt;4</td>
<td>1</td>
<td>(0.0115625041107168 * 16.0)</td>
</tr>
<tr>
<td>$\sin(12 \pi/32)$</td>
<td>80&gt;&gt;4</td>
<td>1</td>
<td>(0.0115625041107168 * 16.0)</td>
</tr>
</tbody>
</table>

At most 94 addition operations are required for the high-precision implementation. The scaling matrix S used to complete the scaling process is implemented with 11 bits of precision and rounded to the nearest integers as shown in Figure 5:

2048, 452, 1135, 118, 536, 118, 802, 640, 2048, 640, 802, 118, 536, 118, 1135, 452
452, 100, 251, 26, 118, 26, 177, 141, 452, 141, 177, 26, 118, 26, 251, 100
1135, 251, 629, 66, 297, 66, 444, 354, 1135, 354, 444, 66, 297, 66, 629, 251
118, 26, 66, 7, 31, 7, 46, 37, 118, 37, 46, 7, 31, 7, 66, 26
118, 26, 66, 7, 31, 7, 46, 37, 118, 37, 46, 7, 31, 7, 66, 26
802, 177, 444, 46, 210, 46, 314, 251, 802, 251, 314, 46, 210, 46, 444, 177
640, 141, 354, 37, 167, 37, 251, 200, 640, 200, 251, 37, 167, 37, 354, 141
2048, 452, 1135, 118, 536, 118, 802, 640, 2048, 640, 802, 118, 536, 118, 1135, 452
640, 141, 354, 37, 167, 37, 251, 200, 640, 200, 251, 37, 167, 37, 354, 141
802, 177, 444, 46, 210, 46, 314, 251, 802, 251, 314, 46, 210, 46, 444, 177
118, 26, 66, 7, 31, 7, 46, 37, 118, 37, 46, 7, 31, 7, 66, 26
118, 26, 66, 7, 31, 7, 46, 37, 118, 37, 46, 7, 31, 7, 66, 26
1135, 251, 629, 66, 297, 66, 444, 354, 1135, 354, 444, 66, 297, 66, 629, 251
452, 100, 251, 26, 118, 26, 177, 141, 452, 141, 177, 26, 118, 26, 251, 100

Figure 5. Table of scale factors used in matrix $S \otimes S^T$ for high-precision implementation of 16x16 DCT.

To analyze the precision of the high complexity implementation, we compute the approximation of $I$ that results from equation (33). Note that in this case, 15 bits are shifted out of the final result to accommodate the 11 (rather than 10) bits that were used to compute the factors of the scaling matrix $S \otimes S^T$.

$$F = (AA^T) \otimes (S \otimes S^T) \div 2^{15}$$

The precision analysis performed for the high-complexity implementation with respect to (34) is shown in Figure 6:
The common factor method employs the use of the common factor method to design low-complexity, fixed-point approximations for sinusoidal transforms such as the DCT. Specifically, this paper provides two implementations of a 16x16 DCT, both of which are nearly orthogonal, with one implementation requiring no more than one addition operation to perform the equivalent of each multiplication operation required in the main transform. This paper further demonstrates how scaled architectures, in general, can mitigate the complexity in a transform implementation by moving the scaling of the approximations used in the main (simplified) transform matrix into a single scaling stage. This single scaling stage can be performed using point-wise (rather than traditional) matrix multiplication. The common factor method employs such scaling to absorb the common factors used to create the approximations used in the simplified transform matrix. The simplified transform matrix can therefore be implemented entirely with integers or dyadic rationals without overly compromising the precision of approximating the exact numeric values for the cosines (or sines) of a sinusoidal transform.

5. CONCLUSION

We have demonstrated the use of the common factor method to design low-complexity, fixed-point approximations for sinusoidal transforms such as the DCT. Specifically, this paper provides two implementations of a 16x16 DCT, both of which are nearly orthogonal, with one implementation requiring no more than one addition operation to perform the equivalent of each multiplication operation required in the main transform. This paper further demonstrates how scaled architectures, in general, can mitigate the complexity in a transform implementation by moving the scaling of the approximations used in the main (simplified) transform matrix into a single scaling stage. This single scaling stage can be performed using point-wise (rather than traditional) matrix multiplication. The common factor method employs such scaling to absorb the common factors used to create the approximations used in the simplified transform matrix. The simplified transform matrix can therefore be implemented entirely with integers or dyadic rationals without overly compromising the precision of approximating the exact numeric values for the cosines (or sines) of a sinusoidal transform.

REFERENCES


