4 Integer Fast Fourier Transform

4.1 Introduction

Integer fast Fourier transform (IntFFT) is an integer approximation of the DFT [I-4]. The transform can be implemented by using only bit shifts and additions but no multiplications. Unlike the fixed-point FFT (FxpFFT), IntFFT is power adaptable and reversible. IntFFT has the same accuracy as the FxpFFT when the transform coefficients are quantized to a certain number of bits. Complexity of IntFFT is much lower than that of FxpFFT, as the former requires only integer arithmetic.

Since the DFT has the orthogonality property, the DFT is invertible. The inverse is just the complex conjugate transpose. Fixed-point arithmetic is often used to implement the DFT in hardware. Direct quantization of the coefficients destroys the invertibility of the transform. The IntFFT keeps the invertibility property of the DFT while the coefficients can be quantized to finite-length binary numbers.

Lifting factorization can replace the $2 \times 2$ orthogonal matrices appearing in fast structures to compute the DFT of input with length of $N = 2^n$ for $n$ an integer such as split-radix, radix-2 and radix-4. The resulting transforms or IntFFTs are invertible, even though the lifting coefficients are quantized and power-adaptable, that is, different quantization step sizes can be used to quantize the lifting coefficients.
The lifting scheme is used to construct wavelets and perfect reconstruction (PR) filter banks [1-1,2,3,4,6]. Biorthogonal filter banks having integer coefficients can be easily implemented and can be used as integer-to-integer transform.

The two-channel system in Fig. 4.1 shows the lifting scheme. The first branch is operated by $A_0$ and called dual lifting whereas the second branch is operated by $A_1$ and is called lifting. We can see that the system is PR for any choices of $A_0$ and $A_1$. It should be noted that $A_0$ and $A_1$ can be nonlinear operations like rounding or flooring operations, etc.

The butterfly structure for implementing a complex multiplication where

$$s = \sin \theta \quad \text{and} \quad c = \cos \theta \quad [I-4].$$
4.3 Algorithms

Integer fast Fourier transform algorithm approximates the twiddle factor multiplication [I-4,5]. Let \( x = x_r + jx_i \) be a complex number. The multiplication of \( x \) with a twiddle factor \( a = e^{j\theta} = c + js \), is the complex number, \( y = ax \) and can be represented as

\[
y = (1 + j) \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x_r \\ x_i \end{bmatrix} = (1 + j) \mathbf{R}_\theta \begin{bmatrix} x_r \\ x_i \end{bmatrix}
\]

where

\[
\mathbf{R}_\theta = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}
\]

The main difficulty in constructing a multiplier-less or integer transform by using the sum-of-powers-of-two (SOPOT) representation of \( \mathbf{R}_\theta \) is that the coefficients of the inverse matrix of \( \mathbf{R}_\theta \) cannot in general be represented in terms of SOPOT coefficients. If \( \cos \theta \) and \( \sin \theta \) in (1) are quantized and represented as \( \alpha \) and \( \beta \) in terms of SOPOT coefficients, then an approximation of \( \mathbf{R}_\theta \) is in (4.3).

\[
\tilde{\mathbf{R}}_\theta = \begin{bmatrix} \alpha & -\beta \\ \beta & \alpha \end{bmatrix}
\]

and its inverse is in (4.4).
As \( \alpha \) and \( \beta \) are SOPOT coefficients, the term \( \sqrt{\alpha^2 + \beta^2} \) cannot in general be represented as SOPOT coefficients. The basic idea of the integer or multiplier-less transform is to decompose \( R_\theta \) into three lifting steps.

If \( \det([A]) = 1 \) and \( c \neq 0 \) \([1-6]\),

\[
[A] = \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} 1 & (a-1)/c \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ c & 1 \end{bmatrix} \begin{bmatrix} 1 & (d-1)/c \\ 0 & 1 \end{bmatrix}
\]

(4.5)

From (4.5), we can decompose \( R_\theta \) as

\[
R_\theta = \begin{bmatrix} 1 & \cos \theta - 1 \\ 0 & \sin \theta \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \sin \theta & 1 \end{bmatrix} = R_1 R_2 R_3
\]

(4.6)

\[
R_\theta^{-1} = R_3^{-1} R_2^{-1} R_1^{-1} = \begin{bmatrix} 1 & -\cos \theta - 1 \\ 0 & \sin \theta \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\sin \theta & 1 \end{bmatrix} = R_3^{-1} R_2^{-1} R_1^{-1}
\]

(4.7)

The coefficients in the factorization of (4.6) can be quantized to SOPOT coefficients to form

\[
R_\theta \approx \tilde{R}_\theta = \begin{bmatrix} 1 & \alpha_\theta \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \beta_\theta & 1 \end{bmatrix} \begin{bmatrix} 1 & \alpha_\theta \\ 0 & 1 \end{bmatrix}
\]

(4.8)

where \( \alpha_\theta \) and \( \beta_\theta \) are respectively SOPOT approximations to \((\cos \theta - 1)/\sin \theta\) and \(\sin \theta\) having the form in (4.9).

\[
\alpha_\theta = \sum_{i=1}^{t} a_i 2^b_i
\]

(4.9)

where \( a_i \in \{-1, 1\} \), \( b_i \in \{ -r, \ldots, -1, 0, 1, \ldots, r \} \), \( r \) is the range of the coefficients and \( t \) is the number of terms used in each coefficients. The variable \( t \) is usually limited so that the multiplication can be implemented with limited number of addition and shift operations. The integer FFT converges to the DFT when \( t \) increases.

The lifting structure has two advantages over butterfly structure. First, the number of real multiplications is reduced from four to three, although the number of additions is increased from two to three (see Fig. 4.2 and Fig 4.3). Second, the structure allows for
quantization of the lifting coefficients and the quantization does not destroy the PR property. To be specific, instead of quantizing the elements of \( [R_\theta] \) in (4.2) directly, the lifting coefficients, \( s \) and \((s-1)/c\) are quantized and therefore, the inversion also consists of three lifting steps with the same lifting coefficients but with opposite signs.

Example 4.1

In case of the twiddle factor, \( W_8^{-1} \), \( \theta = -\pi/4 \), \((\cos \theta - 1)/\sin \theta = \sqrt{2} - 1\) and \(\sin \theta = -1/\sqrt{2}\). If we round these numbers respectively to the right-hand one digit of the decimal point, then \( \alpha_\theta = 0.4 \) and \( \beta_\theta = 0.7 \).

\[
\Sigma_\theta = \begin{bmatrix}
1 & 0.4 \\
0.1 & -0.7 \\
0 & 1
\end{bmatrix}
\]

\[
\Sigma_\theta^{-1} = \begin{bmatrix}
1 & -0.4 \\
0.1 & 0.7 \\
0 & 1
\end{bmatrix}
\]

For any real numbers of \( \alpha_\theta \) and \( \beta_\theta \) in the lattice structure, \( \Sigma_\theta^{-1} \Sigma_\theta = I \).

In summery, in implementing a complex number multiplication a twiddle factor in matrix form has a butterfly structure and if we round coefficients, its inverse is computationally complex, but if we decompose the twiddle factor into a lifting structure, the twiddle factor has a perfect inverse even if we round coefficients. Once the coefficients are rounded in the lifting structure, the twiddle factor may have either structure for perfect inverse but the lifting structure has one less multiplication.

An eight-point integer FFT based on the split-radix structure is constructed in [I-4]. Figure 4.5 shows the lattice structure of the integer FFT, where the twiddle factors \( W_8^{-1} \) and \( W_8^{-1} \) are implemented using the factorization. Another integer FFT based on the radix-2 decimation-in-frequency is covered in [I-5]. At the expense of precision, we can develop computationally effective integer FFT algorithms.

If \( \theta \in (-\pi, -\pi/2) \cup (\pi/2, \pi) \), then \(|(\cos \theta - 1)/\sin \theta| > 1\). Thus the absolute values of the lifting coefficients need to be controlled to be less than or equal to one by replacing \( R_\theta \) by \(-R_{\theta+\pi}\) as follows:

\[
R_\theta = -R_{\theta+\pi} = \begin{bmatrix}
-cos \theta & sin \theta \\
-sin \theta & -cos \theta
\end{bmatrix}
= \begin{bmatrix}
1 & (c+1)/s \\
0 & 1
\end{bmatrix}
\begin{bmatrix}
0 & 1 \\
1 & -s
\end{bmatrix}
\begin{bmatrix}
1 & 0 \\
1 & 0
\end{bmatrix}
\]

(4.10)
If we assume \( \theta = \phi \pm (\pi/2) \), then we can have another choice of lifting factorization as follows:

\[
\begin{align*}
R_\theta &= \begin{bmatrix} 
\cos \theta & -\sin \theta \\
\sin \theta & \cos \theta 
\end{bmatrix} = \begin{bmatrix} 
-\cos \phi & -\sin \phi \\
\sin \phi & -\cos \phi 
\end{bmatrix} \\
&= \begin{bmatrix} 
1 & -\frac{1}{c} - 1 \\
0 & 1
\end{bmatrix} \begin{bmatrix} 
1 & 1 \\
s & 1
\end{bmatrix}, \\
R_\theta &= \begin{bmatrix} 
\cos \theta & -\sin \theta \\
\sin \theta & \cos \theta 
\end{bmatrix} = \begin{bmatrix} 
\cos \phi & \sin \phi \\
-\sin \phi & \cos \phi 
\end{bmatrix} \\
&= \begin{bmatrix} 
1 & -\frac{1}{c} - 1 \\
0 & 1
\end{bmatrix} \begin{bmatrix} 
1 & 1 \\
s & 1
\end{bmatrix}.
\end{align*}
\] (4.11)

However, if \( \theta \in (-\pi, 0) \) and \( \sin \theta < 0 \), \( (\sin \theta - 1)/\cos \theta \) will be larger than one. Thus \( R_\theta \) should be replaced by \(-R_{\theta+x}\) as follows:

\[
\begin{align*}
R_\theta &= \begin{bmatrix} 
-\cos \theta & \sin \theta \\
-\sin \theta & -\cos \theta 
\end{bmatrix} = \begin{bmatrix} 
\cos \phi & \sin \phi \\
-\sin \phi & -\cos \phi 
\end{bmatrix} \\
&= \begin{bmatrix} 
1 & -\frac{1}{c} - 1 \\
0 & 1
\end{bmatrix} \begin{bmatrix} 
1 & 1 \\
s & 1
\end{bmatrix}.
\end{align*}
\] (4.13)

For example, suppose we are given the twiddle factor, \( W_8^3 = e^{-j6\pi/8} \). Then \( \theta = -3\pi/4 \). Then we have the two options, (4.10) and (4.14). We select (4.10) here. Then, by using (4.1)

\[
R_\theta = -R_{\theta+x} = \begin{bmatrix} 
\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\
\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}
\end{bmatrix} \\
= \begin{bmatrix} 
1 & 0 \\
0 & 1
\end{bmatrix} \begin{bmatrix} 
1 & -\sqrt{2} \\
1 & 1 - \sqrt{2}
\end{bmatrix} \begin{bmatrix} 
1 & 1 - \sqrt{2} \\
0 & 1
\end{bmatrix}. 
\] (4.15)

The equivalent block diagram for (4.15) is shown in Fig. 4.5.
Table 4.1 Possible lifting factorizations for each value of $\theta$ with all lifting coefficients between $-1$ and $1$ [I-4].

<table>
<thead>
<tr>
<th>Range of $\theta$</th>
<th>Lifting factorization</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(0, \pi/2)$</td>
<td>(4.6), (4.11), and (4.12)</td>
</tr>
<tr>
<td>$(\pi/2, \pi)$</td>
<td>(4.10), (4.11), and (4.12)</td>
</tr>
<tr>
<td>$(-\pi, -\pi/2)$</td>
<td>(4.10), (4.13), and (4.14)</td>
</tr>
<tr>
<td>$(-\pi/2, 0)$</td>
<td>(4.6), (4.13), and (4.14)</td>
</tr>
</tbody>
</table>

4.4 Performances and Complexities

While for two and higher dimensions, the row-column method, the vector-radix FFT and the polynomial transform FFT algorithms are commonly used fast algorithms for computing multidimensional discrete Fourier transform (M-D DFT).

Figure 4.5 Lattice structure of eight-point integer FFT using split-radix structure [I-4].

85
Table 4.2  Computational complexities of the split-radix FFT and the integer versions (FxpFFT and IntFFT) when the coefficients are quantized to $N_c = 10$ bits.

<table>
<thead>
<tr>
<th>$N$</th>
<th>Split-radix FFT</th>
<th>FxpFFT</th>
<th>IntFFT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Multiplies</td>
<td>Adds</td>
<td>Adds</td>
</tr>
<tr>
<td>16</td>
<td>20</td>
<td>148</td>
<td>262</td>
</tr>
<tr>
<td>32</td>
<td>68</td>
<td>388</td>
<td>746</td>
</tr>
<tr>
<td>64</td>
<td>196</td>
<td>964</td>
<td>1910</td>
</tr>
<tr>
<td>128</td>
<td>516</td>
<td>2308</td>
<td>4674</td>
</tr>
<tr>
<td>256</td>
<td>1284</td>
<td>5380</td>
<td>10990</td>
</tr>
<tr>
<td>512</td>
<td>3076</td>
<td>12292</td>
<td>25346</td>
</tr>
<tr>
<td>1024</td>
<td>7172</td>
<td>27652</td>
<td>57398</td>
</tr>
</tbody>
</table>
4.5 Integer Discrete Fourier Transform [I-5]

Integer Fourier transform approximates the DFT for the fixed-point multiplications. The fixed-point multiplications can be implemented by the addition and binary shifting operations. For example

\[ 7 \times a = a \ll 2 + a \ll 1 + a \]

where \( a \) is an integer and \( \ll \) is a binary left-shift operator.

Two types of integer transforms are presented in this section. Forward and inverse transform matrices can be the same and different. They are referred to as near-complete and complete integer DFTs.

Near-Complete Integer DFT

Let \([F]\) be the DFT matrix, and let \([B]^T\) be an integer DFT. Then for integer DFT to be orthogonal and, hence, be reversible, it is required that

\[
[B]^T[B]^T = \text{diag}(r_0,r_1,r_2,\ldots,r_r) = [C]
\]

(4.16)

where \( r_t = 2^m \) and \( m \) is an integer. Since \([C]\) is a diagonal matrix, it follows that

\[
\]

(4.17)

To approximate the DFT, integer DFT \([B]^T\) keeps all the signs of entries of \([F]\) as follows.

\[
[B] = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & -j & -1 & j & 1 & -j & -1 & j \\
1 & j & -j & -1 & 1 & -j & -1 & j \\
1 & j & j & -1 & 1 & -j & -1 & j \\
1 & -j & j & -1 & 1 & -j & -1 & j \\
1 & j & -j & -1 & 1 & -j & -1 & j \\
1 & -j & j & -1 & 1 & -j & -1 & j \\
1 & j & -j & -1 & 1 & -j & -1 & j
\end{bmatrix}
\]

(4.18)

In order for (4.16) to be satisfied, the complex inner products of the following column pairs of \([B]^T\) should be zero.

\[ \langle \text{Column 1, Column 5} \rangle = 0 \quad \langle \text{Column 3, Column 7} \rangle = 0 \]

(4.19)

Here a complex inner product is defined by

\[ \langle z, w \rangle = w^H z \]

for complex vectors \( z, w \). \( w^H \) is the transpose of \( w^* \). From (4.19)
\[ a, b_1 = 2a, c_2 \]  \hspace{1cm} (4.20)

From (4.16)

\[ r_6 = r_2 = r_4 = r_6 = N \]
\[ r_1 = r_7 = (N/2)a_i^2 + Na_x^2 \]
\[ r_5 = r_3 = (N/2)c_i^2 + Nc_x^2 \]

Some possible choices of the parameters of 8-point integer DFT are listed in Table 4.3.

<table>
<thead>
<tr>
<th>( a_1 )</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>8</th>
<th>10</th>
<th>17</th>
<th>99</th>
<th>500</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a_2 )</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>12</td>
<td>70</td>
<td>353</td>
</tr>
<tr>
<td>( b_1 )</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>7</td>
<td>24</td>
<td>140</td>
<td>353</td>
</tr>
<tr>
<td>( b_2 )</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>17</td>
<td>99</td>
<td>500</td>
</tr>
</tbody>
</table>

**Complete Integer DFT**

Let \([F]\) be the DFT matrix, and let \([B]^T\) and \([D]^T\) be the forward and inverse integer DFT. Then for integer DFT to be orthogonal and, hence, be reversible, it is required that

\[ [D]^T[B]^T = \text{diag}(r_0, r_1, r_2, \ldots, r_7) = [C] \]  \hspace{1cm} (4.21)

where \( r_j = 2^{-m} \) and \( m \) is an integer. Since \([C]\) is a diagonal matrix, it follows that

\[ [C]^{-1}[D]^T[B]^T = [I] \]  \hspace{1cm} (4.22)

To approximate the DFT, integer DFT \([B]^T\) keeps all the signs of entries of \([F]\).
\[ [D] = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
- a_4 & - a_4 + ja_4 & ja_3 & ja_3 & a_4 & ja_4 & a_4 & ja_4 \\
1 & j & 1 & - j & j & 1 & - j & -1 \\
1 & -1 & 1 & -1 & -1 & -1 & -1 & j \\
1 & j & -1 & -j & 1 & j & -1 & -j \\
a_3 & a_4 + ja_4 & ja_3 & - a_4 + ja_4 & - a_3 & - a_4 & - ja_4 & - ja_3 & a_4 & ja_4 \\
\end{bmatrix} \]

Except that the complex inner products of the following pairs should be zero, (4.21) is satisfied.

\[
\langle \text{Column 1 of } [B]^T, \text{Column 5 of } [D]^T \rangle, \quad \langle \text{Column 5 of } [B]^T, \text{Column 1 of } [D]^T \rangle
\]

\[
\langle \text{Column 3 of } [B]^T, \text{Column 7 of } [D]^T \rangle, \quad \langle \text{Column 7 of } [B]^T, \text{Column 3 of } [D]^T \rangle
\]

\[ a_2 b_3 = 2a_2 b_4 \quad \quad a_2 b_1 = 2a_2 b_2 \]

Since the inner product of the corresponding columns of \([D]^T\) and \([B]^T\) should be the power of two from (4.21),

\[ a_i a_3 + 2a_2 a_4 = 2^k \quad \quad c_i c_3 + 2c_2 c_4 = 2^k \]

From (4.21)

\[ r_5 = r_2 = r_4 = r_6 = N \]

\[ r_1 = r_7 = (N/2)c_1 a_3 + Nc_2 a_4 \]

\[ r_3 = r_5 = (N/2)c_1 c_3 + Nc_2 c_4 \]

Some possible choices of the parameters of 8-point integer DFT are listed in Table 4.4.

<table>
<thead>
<tr>
<th>( a_i )</th>
<th>2</th>
<th>7</th>
<th>3</th>
<th>4</th>
<th>4</th>
<th>5</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a_1 )</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>( b_1 )</td>
<td>2</td>
<td>13</td>
<td>17</td>
<td>12</td>
<td>44</td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td>( b_2 )</td>
<td>1</td>
<td>9</td>
<td>10</td>
<td>7</td>
<td>31</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>( a_3 )</td>
<td>1</td>
<td>18</td>
<td>34</td>
<td>7</td>
<td>31</td>
<td>24</td>
<td>13</td>
</tr>
</tbody>
</table>
Example
Figure 4.6 shows the near-complete and complete integer DFTs of two random input vectors $x_1$, $x_2$ compared to the DFTs of those.

$$x_1^T = (2, 3, 4, 5, 4, 5, 2, 3)$$
$$x_2^T = (2.8, 4.3 - j0.6, 3.7 + j0.9, 3.1 - j0.6, 4.6, 3.1 + j0.6, 3.70 - j0.9, 4.3 + j0.6)$$

A parameter set chosen for the near-complete integer DFT is:
$$\{a_1 = 2, a_2 = 1, c_1 = 1, c_2 = 1\}$$

A parameter set chosen for the complete integer DFT is:
$$\{a_1 = 7, a_2 = 5, c_1 = 13, c_2 = 9, a_3 = 18, a_4 = 13, c_3 = 10, c_4 = 7\}$$

Entries $B(k,n)$ of $[B]$ are normalized by the first column $B(k,0)$ as

$$\tilde{B}(k,n) = B(k,n) / B(k,0) \quad k, n = 0, 1, ..., N - 1$$

$[\tilde{B}]^T$ is normalized again using (4.17) to get the normalized integer DFT $[\tilde{B}]^\top$ as

$$[\tilde{B}]^\top = [C]^{1/2} [\tilde{B}]^T$$

1.
Figure 4.6 (a), (b): Input signals, $x_1$, $x_2$. (c), (d): Near complete integer DFTs of $x_1$, $x_2$. (e), (f): Complete integer DFTs of $x_1$, $x_2$. Dashed and solid lines represent the regular and integer DFTs of input signals.
4.6 Summary

This chapter has developed the integer FFT (IntFFT) based on the lifting scheme. Its advantages are enumerated. A specific algorithm (eight-point IntFFT) using split-radix structure is formulated. Extension of the 1-D DFT to the multi-D DFT (specifically 2-D DFT) is the focus of the next chapter. Besides the definitions and properties, filtering of 2-D signals such as images and variance distribution in the DFT domain are some relevant topics.
Problems

4.1 If $\det[A] = 1$ and $b \neq 0$, 

$$
[A] = \begin{bmatrix}
a & b \\
c & d
\end{bmatrix} = \begin{bmatrix}
1 & 0 \\
(d-1)/b & 1
\end{bmatrix} \begin{bmatrix}
1 & 0 \\
(a-1)/b & 1
\end{bmatrix}
$$

(4.16)

Assume $c \neq 0$ and derive (4.5) from (4.16).

4.2 Develop a flow-graph for implementing 8-point inverse integer FFT using split-radix structure (see Fig. 4.5).

4.3 Repeat Problem 4.2 for 16-point for forward and inverse integer FFTs.

4.4 Generate random data sequence $x(n)$, $n = 0, 1, \ldots, 15$. Apply forward integer FFT. What are $X^f(k)$, $k = 0, 1, \ldots, 15$? Apply inverse integer FFT to $X^f(k)$ and recover $x(n)$.

4.5 List 5 other parameter sets for the integer DFT than those in Table 4.3. What equation do you need?
Lifting Scheme


Integer FFT


DCT and DST


See also [B26].
integer DCT


See also [I-7, I-8, I-10].

integer MDCT


Discrete Fourier-Hartley Transform
