Note Set #26

• FFT Algorithm: Divide & Conquer Viewpoint
• Reading: Sect. 8.1.2 & 8.1.3 of Proakis & Manolakis
**Divide & Conquer Approach**

The previous note set’s FFT development was somewhat ad hoc. Here we develop a more formalized & generalized approach that can be used to develop other FFT approaches.

To illustrate the basic ideas consider the case of an $N$-pt DFT where $N$ is not prime can be factored into two integer factors: $N = LM$.

Can always zero-pad out to appropriate number.

We now can use either of two mappings from 2-D indices $l,m$ to the actual time index $n$:

- $n = Ml + m$ ("Row-Wise Input Mapping")
- $n = l + mL$ ("Column-Wise Input Mapping")

We use one or the other of these mappings to convert the input to a matrix and then take DFTs along either the rows or columns.

We now can use either of two mappings from 2-D indices $p,q$ to the actual DFT result index $k$:

- $k = Mp + q$ ("Row-Wise Output Mapping")
- $k = p + qL$ ("Column-Wise Output Mapping")
\[ n = Ml + m \]

**“Row-Wise Input Mapping”**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>( \ldots )</th>
<th>( M - 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( x(0) )</td>
<td>( x(1) )</td>
<td>( x(2) )</td>
<td>( \ldots )</td>
<td>( x(M - 1) )</td>
</tr>
<tr>
<td>1</td>
<td>( x(M) )</td>
<td>( x(M + 1) )</td>
<td>( x(M + 2) )</td>
<td>( \ldots )</td>
<td>( x(2M - 1) )</td>
</tr>
<tr>
<td>2</td>
<td>( x(2M) )</td>
<td>( x(2M + 1) )</td>
<td>( x(2M + 2) )</td>
<td>( \ldots )</td>
<td>( x(3M - 1) )</td>
</tr>
<tr>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \ldots )</td>
<td>( \vdots )</td>
</tr>
<tr>
<td>( L - 1 )</td>
<td>( x((L - 1)M) )</td>
<td>( x((L - 1)M + 1) )</td>
<td>( x((L - 1)M + 2) )</td>
<td>( \ldots )</td>
<td>( x(LM - 1) )</td>
</tr>
</tbody>
</table>

**DFTs Down Columns**

\[ n = l + mL \]

**“Column-Wise Input Mapping”**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>( \ldots )</th>
<th>( M - 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( x(0) )</td>
<td>( x(L) )</td>
<td>( x(2L) )</td>
<td>( \ldots )</td>
<td>( x((M - 1)L) )</td>
</tr>
<tr>
<td>1</td>
<td>( x(1) )</td>
<td>( x(L + 1) )</td>
<td>( x(2L + 1) )</td>
<td>( \ldots )</td>
<td>( x((M - 1)L + 1) )</td>
</tr>
<tr>
<td>2</td>
<td>( x(2) )</td>
<td>( x(L + 2) )</td>
<td>( x(2L + 2) )</td>
<td>( \ldots )</td>
<td>( x((M - 1)L + 2) )</td>
</tr>
<tr>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \ldots )</td>
<td>( \vdots )</td>
</tr>
<tr>
<td>( L - 1 )</td>
<td>( x(L - 1) )</td>
<td>( x(2L - 1) )</td>
<td>( x(3L - 1) )</td>
<td>( \ldots )</td>
<td>( x(LM - 1) )</td>
</tr>
</tbody>
</table>

**DFTs Across Rows**
Now to illustrate how to use this machinery… Use
- Column-wise input mapping  \( n = l + mL \)
- Row-wise output mapping  \( k = Mp + q \)

\[
X[k] = \sum_{n=0}^{N-1} x[n]W_N^{kn}
\]

\[
X[p, q] = \sum_{m=0}^{M} \sum_{l=0}^{L-1} x[l, m]W_N^{(Mp+q)(l+mL)}
\]

\[
= W_N^{MLmp} W_N^{mLq} W_N^{Mpl} W_N^{lq}
\]

\[
= W_N^{Nmp} = 1 W_N^{mq} W_N^{N/l} = W_N^{pl}
\]

\[
X[p, q] = \sum_{l=0}^{L} W_N^{lq} \left[ \sum_{m=0}^{M-1} x[l, m]W_N^{mq} \right] W_N^{pl}
\]

\[
= F[l, q] \triangleq G[l, q]
\]

\[
= X[p, q]
\]

Compute M-pt DFTs of Rows
Apply Twiddle Factors
Compute L-pt DFTs of Colns
Illustration of this Approach

- Compute $M$-pt DFTs of Rows
- Apply Twiddle Factors
- Compute $L$-pt DFTs of Colns

Figure 8.1.3  Computation of $N = 15$-point DFT by means of 3-point and 5-point DFTs.

Although this LOOKS more complicated… it is actually more efficient!
**Application to Develop Dec-In-Time Radix-2 FFT**  \( \text{Let } N = 2^\nu \)

We apply the divide-and-conquer approach with \( M = N/2 \) & \( L = 2 \)

\[ n = l + mL \]

“Column-Wise Input Mapping”

DFTs Across Rows

Now repeat this for each \( N/2\)-pt DFT… Etc., Etc., Etc.
Application to Develop Dec-In-Frequency Radix-2 FFT \( N = 2^v \)

We apply the divide-and-conquer approach with \( M = 2 \) & \( L = N/2 \)

\[ n = l + mL \]

“Column-Wise Input Mapping”

Compute \( N/2 \)-pt DFTs of Columns

Apply Twiddle Factors

Compute 2-pt DFTs of Rows

Now repeat this for each \( N/2 \)-pt DFT... Etc., Etc., Etc.
$N = 8$ First Stage of Dec-in-Freq FFT
Complete Dec-in-Freq FFT for $N = 8$

DFT Values are in Bit Reversed Order!
Butterfly Structure: DiT vs DiF

Butterfly Structure: Dec-in-Time

Butterfly Structure: Dec-in-Freq
3 Different Configurations of D-in-Time FFT

*N=8 D-in-Time FFT w/ BR Inputs*

Can be done “in-place”

*N=8 D-in-Time FFT w/ BR Outputs*

Can be done “in-place”

*N=8 D-in-Time FFT w/ both sides “Normal Order”*

Can NOT be done “in-place”


Writes over Data Needed Later!!
3 Different Configurations of D-in-Freq FFT

**N=8 D-in-Freq FFT w/ BR Inputs**  
Can be done “in-place”

**N=8 D-in-Freq FFT w/ BR Outputs**  
Can be done “in-place”

Writes over Data Needed Later!!

**N=8 D-in-Freq FFT w/ both sides “Normal Order”**  
Can NOT be done “in-place”

Implementation Issues

- We’ve looked at two radix-two methods.
  - Other radices: 4 & 8
  - Split radix (2 and 4)

- In-Place computation requires only $2N$ memory locations
  - But complicates the indexing & control operations
  - Doubling the memory to $4N$ locations can be advantageous
    - Reduces complexity of indexing & control
    - Allows natural ordering for both input and output

- In general, many factors come into play when determining best method
  - Parallelism, HW vs SW, fixed-point vs floating-point, etc.
- Also… no need to develop distinct IFFT algorithm

$$
x[n] = \frac{1}{N} \sum_{n=0}^{N-1} X[k] e^{j2\pi kn/N} = \frac{1}{N} \left[ \sum_{n=0}^{N-1} X^*[k] e^{-j2\pi kn/N} \right]^* 
$$

$$
IDFT \{X[k]\} = \frac{1}{N} \text{conj} \left\{ DFT \{X^*[k]\} \right\}
$$

<table>
<thead>
<tr>
<th>$N$</th>
<th>Radix 2</th>
<th>Radix 4</th>
<th>Radix 8</th>
<th>Split Radix</th>
<th>Radix 2</th>
<th>Radix 4</th>
<th>Radix 8</th>
<th>Split Radix</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>24</td>
<td>20</td>
<td>20</td>
<td>152</td>
<td>148</td>
<td>148</td>
<td>148</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>88</td>
<td>68</td>
<td>408</td>
<td></td>
<td>388</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>264</td>
<td>208</td>
<td>204</td>
<td>1,032</td>
<td>976</td>
<td>972</td>
<td>964</td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>712</td>
<td>516</td>
<td>2,504</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>1,800</td>
<td>1,392</td>
<td>1,284</td>
<td>5,896</td>
<td>5,488</td>
<td>5,380</td>
<td></td>
<td></td>
</tr>
<tr>
<td>512</td>
<td>4,360</td>
<td>3,204</td>
<td>3,076</td>
<td>13,566</td>
<td>12,420</td>
<td>12,292</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,024</td>
<td>10,248</td>
<td>7,856</td>
<td>7,172</td>
<td>30,728</td>
<td>28,336</td>
<td>27,652</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Source: Extracted from Duhamel (1986).
Two Tricks for Real-Valued Signals

1. Efficient DFT of two Real-Valued Signals

Let \( x_1[n] \) and \( x_2[n] \) be real-valued signals, each length \( N \)

Form: \( x[n] \triangleq x_1[n] + jx_2[n] \)

Then we have

\[
x_1[n] = \frac{x[n] + x^*[n]}{2} \quad \text{and} \quad x_2[n] = \frac{x[n] - x^*[n]}{2j}
\]

Thus

\[
X_1[k] = \frac{\text{DFT} \{ x[n] \} + \text{DFT} \{ x^*[n] \}}{2} \quad \text{and} \quad X_2[k] = \frac{\text{DFT} \{ x[n] \} - \text{DFT} \{ x^*[n] \}}{2j}
\]

But… \( \text{DFT} \{ x^*[n] \} = X^*[N - k] \)

\[
X_1[k] = \frac{1}{2} \left[ X[k] + X^*[N - k] \right] \quad \text{and} \quad X_2[k] = \frac{1}{2j} \left[ X[k] - X^*[N - k] \right]
\]
2. Efficient DFT of 2N-pt Real-Valued Signal

Let \( g[n] \) be a real-valued signal of length \( 2N \)

Then define: \( x_1[n] \equiv g[2n] \quad \& \quad x_2[n] \equiv g[2n+1] \)

And: \( x[n] = x_1[n] + jx_2[n] \)

From Trick #1 we have

\[
X_1[k] = \frac{1}{2} \left[ X[k] + X^*[N - k] \right] \quad \& \quad X_2[k] = \frac{1}{2j} \left[ X[k] - X^*[N - k] \right]
\]

But using ideas from Dec-in-Time FFT we know

\[
G[k] = \sum_{n=0}^{N-1} g[2n]W_{2N}^{2nk} + \sum_{n=0}^{N-1} g[2n+1]W_{2N}^{(2n+1)k}
\]

\[
= \sum_{n=0}^{N-1} x_1[n]W_{2N}^{2nk} + \sum_{n=0}^{N-1} x_2[n]W_{2N}^{(2n+1)k}
\]

So then we get

\[
G[k] = X_1[k] + W_{2N}^k X_2[k], \quad k = 0, 1, \ldots, N - 1
\]

\[
G[k + N] = X_1[k] - W_{2N}^k X_2[k], \quad k = 0, 1, \ldots, N - 1
\]

So computing one \( N \)-pt DFT of \( x[n] \) gets us the \( 2N \)-pt DFT of \( g[n] \)!