ELEG–305: Digital Signal Processing
Lecture 17: The Fast Fourier Transform; Radix–2 and Radix–4 Algorithms

Kenneth E. Barner
Department of Electrical and Computer Engineering
University of Delaware
Fall 2008

Outline

1. Review of Previous Lecture

2. Lecture Objectives

3. Efficient Computation of the DFT: FFT Algorithms
   - Radix–2 FFT (Decimation–in–Frequency)
   - Radix–4 FFT (Decimation–in–Time)
   - Radix–4 FFT (Decimation–in–Frequency)
   - Computation Complexity Analysis
   - FFT–Based Filtering
Review of Previous Lecture

- Direct DFT calculation – complexity $O(N^2)$
- FFT calculation – complexity $O(N \log_2 N)$
- Radix–2 Decimation–in–Time (FFT) algorithm – Decompose signal $\log_2 N$ times; Basic computation is the butterfly

Sample Ordering Affects – Radix–2 decimation–in–time algorithm uses bit reversed order inputs (shuffled samples) and produces natural order outputs

Lecture Objectives

Objective
Derive the radix–2 decimation–in–frequency and radix–4 Fast Fourier Transform (FFT) algorithms; Analyze the FFT computational cost; Develop FFT–based filtering methods

Reading
Chapters 8 (8.1); Next lecture, applications of FFT algorithms & linear filtering DFT computation (Chapter 8.2–8.3); Implementation of Discrete–Time Systems (Chapter 9)
Recall: Decimating the time-domain signal (e.g., radix–2 splits input into even/odd sequences) yields the decimation–in–time FFT.

Radix–2 Decimation in Frequency FFT

Objective: Derive an alternate FFT algorithm by decimating in frequency.

Approach: Split the DFT into 2 summations – first half and second half.

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

\[
= \sum_{n=0}^{N/2-1} x(n) W_N^{kn} + \sum_{n=N/2}^{N-1} x(n) W_N^{kn}
\]

\[
= \sum_{n=0}^{N/2-1} x(n) W_N^{kn} + W_N^{kN/2} \sum_{n=0}^{N/2-1} x\left(n + \frac{N}{2}\right) W_N^{kn}
\]

Note that \( W_N^{kN/2} = (-1)^k \).
\[ X(k) = \sum_{n=0}^{N/2-1} x(n)W_N^{kn} + W_N^{kN/2} \sum_{n=0}^{N/2-1} x\left(n + \frac{N}{2}\right)W_N^{kn} \]

\[ = \sum_{n=0}^{N/2-1} \left[ x(n) + (-1)^k x\left(n + \frac{N}{2}\right) \right] W_N^{kn} \]

Decimate \( X(k) \) into even and odd samples, noting that \( W_N^{2kn} = W_{N/2}^{kn} \)

[even samples] \( X(2k) = \sum_{n=0}^{N/2-1} \left[ x(n) + x\left(n + \frac{N}{2}\right) \right] W_N^{kn} \)

[odd samples] \( X(2k + 1) = \sum_{n=0}^{N/2-1} \left\{ \left[ x(n) - x\left(n + \frac{N}{2}\right) \right] W_N^n \right\} W_N^{kn} \)

where for both cases \( k = 0, 1, \ldots, \frac{N}{2} - 1 \) (denote above as \( (*) \))

To simplify the notation, define

\[ g_1(n) = x(n) + x\left(n + \frac{N}{2}\right) \quad k = 0, 1, \ldots, \frac{N}{2} - 1 \]

\[ g_2(n) = \left[ x(n) - x\left(n + \frac{N}{2}\right) \right] W_N^n \quad k = 0, 1, \ldots, \frac{N}{2} - 1 \]

These computations define a new butterfly graph

Note: \( W_N^n \) is a post–weighting term in the frequency decimation case
Then from (*),

\[
X(2k) = \sum_{n=0}^{N/2-1} \left[ x(n) + x\left(n + \frac{N}{2}\right) \right] W_{N/2}^{kn}
\]

[even samples]

\[
X(2k) = \sum_{n=0}^{N/2-1} \left[ x(n) + x\left(n + \frac{N}{2}\right) \right] W_{N}^{n} W_{N/2}^{kn}
\]

[odd samples]

by substituting \(g_1(n)\) and \(g_2(n)\) we get

\[
X(2k) = \sum_{n=0}^{N/2-1} g_1(n) W_{N/2}^{kn}
\]

\[
X(2k+1) = \sum_{n=0}^{N/2-1} g_2(n) W_{N/2}^{kn}
\]

Observation: \(X(2k)\) and \(X(2k+1)\) are the DFTs of \(N/2\) point sequences – computational load has been reduced

Remember the butterfly operations,

\[
g_1(n) = x(n) + x\left(n + \frac{N}{2}\right)
\]

\[
g_2(n) = \left[ x(n) - x\left(n + \frac{N}{2}\right) \right] W_{N}^{n}
\]

Example: Let \(N = 8\). The first stage of the decimation–in–frequency FFT is depicted on the right

Observations: The input is in natural order while the output and shuffled, or in decimation order

Procedure: Repeat the decimation process \(\log_2 N\) times
Efficient Computation of the DFT: FFT Algorithms

Radix–2 FFT (Decimation–in–Frequency)

8–point decimation–in–frequency FFT algorithm

Note: The decimation–in–frequency algorithm utilizes natural order input terms but yields shuffled, decimation order, outputs (DFT coefficients); Also note the weighting pattern, which holds for all \( N \)

Suppose \( N = 4^\nu \) ⇒ advantages to a radix–4 decomposition

Approach: Decimate the signal, i.e., break sequence into fourths

\[
\begin{align*}
  f_1(n) &= x(n) \quad n = 0, 1, \ldots, \frac{N}{4} - 1 \\
  f_2(n) &= x(n + \frac{N}{4}) \quad n = 0, 1, \ldots, \frac{N}{4} - 1 \\
  f_3(n) &= x(n + \frac{N}{2}) \quad n = 0, 1, \ldots, \frac{N}{4} - 1 \\
  f_4(n) &= x(n + \frac{3N}{4}) \quad n = 0, 1, \ldots, \frac{N}{4} - 1
\end{align*}
\]

Break DFT summation into four summations

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

\[
= \sum_{n=0}^{N/4-1} x(n)W_N^{kn} + \sum_{n=N/4}^{N/2-1} x(n)W_N^{kn} + \sum_{n=N/2}^{3N/4-1} x(n)W_N^{kn} + \sum_{n=3N/4}^{N-1} x(n)W_N^{kn}
\]
Shift all summations to $0 \leq n < N/4$

$$X(k) = \sum_{n=0}^{N/4-1} x(n) W_N^{kn} + W_N^{kN/4} \sum_{n=0}^{N/4-1} x \left( n + \frac{N}{4} \right) W_N^{kn}$$

$$+ W_N^{kN/2} \sum_{n=0}^{N/4-1} x \left( n + \frac{N}{2} \right) W_N^{kn} + W_N^{3kN/4} \sum_{n=0}^{N/4-1} x \left( n + 3\frac{N}{4} \right) W_N^{kn}$$

Note: $W_N^{kN/4} = (-j)^k$, $W_N^{kN/2} = (-1)^k$, $W_N^{3kN/4} = (j)^k$

**Approach:** (1) Substitute in $f_i(n)$ terms; (2) write as a single summation; (3) factor out $W_N^{kn}$ terms

$$X(k) = \sum_{n=0}^{N/4-1} \left[ f_1(n) + (-j)^k f_2(n) + (-1)^k f_3(n) + (j)^k f_4(n) \right] W_N^{kn}$$

**Question:** Is this the sum of 4 DFTs?

No. Need $W$ term to match sequence length, i.e., $W_N^{kn_{N/4}}$

**Solution:** Change summation index, $k = \frac{N}{4} p + q$, $q = 0, 1, \ldots, \frac{N}{4} - 1$ and $p = 0, 1, 2, 3$, i.e,

$$p = 0 \Rightarrow \left( \frac{N}{4} p + q \right) = 0, 1, \ldots, \frac{N}{4} - 1$$

$$p = 1 \Rightarrow \left( \frac{N}{4} p + q \right) = \frac{N}{4}, \frac{N}{4} + 1, \ldots, \frac{N}{2} - 1$$

$$p = 2 \Rightarrow \left( \frac{N}{4} p + q \right) = \frac{N}{2}, \frac{N}{2} + 1, \ldots, \frac{3N}{4} - 1$$

$$p = 3 \Rightarrow \left( \frac{N}{4} p + q \right) = \frac{3N}{4}, \frac{3N}{4} + 1, \ldots, N - 1$$

Note that $W_N^{(\frac{N}{4} p + q)n} = W_N^{Np} W_N^{qn} = W_N^{Np} W_N^{qn}$. Also note that

$$(-j)^{\frac{N}{4} p + q} = (-j)^{\frac{N}{4} p} (-j)^q = \left( \frac{1}{(-j)^4} \right)^N (-j)^q = (-j)^q$$

and similarly, $(-1)^{\frac{N}{4} p + q} = (-1)^q$ and $(j)^{\frac{N}{4} p + q} = (j)^q$
Efficient Computation of the DFT: FFT Algorithms

Radix–4 FFT (Decimation–in–Time)

\[ X(p, q) \equiv X \left( \frac{N}{4} p + q \right) = \sum_{n=0}^{N/4-1} \left[ f_1(n) + (-j)^{N/4} p + q f_2(n) \right. \\
+ \left. (-1)^{N/4} p + q f_3(n) + (j)^{N/4} p + q f_4(n) \right] W_{4n}^{pn} W_N^{nq} \]

\[ = \sum_{n=0}^{N/4-1} \left[ f_1(n) + (-j)^q f_2(n) + (-1)^q f_3(n) + (j)^q f_4(n) \right] W_{4n}^{pn} W_N^{nq} \]

\[ = \sum_{n=0}^{N/4-1} W_{4n}^{pn} F(n, q) W_N^{nq} \tag{*} \]

where \( F(n, q) = \left[ f_1(n) + (-j)^q f_2(n) + (-1)^q f_3(n) + (j)^q f_4(n) \right] \)

Noting \( W_{4n}^{pn} = (-j)^{pn} = \pm 1, \pm j \), we express (*) in matrix form (\( N = 4 \)),

\[
\begin{bmatrix}
X(0, q) \\
X(1, q) \\
X(2, q) \\
X(3, q)
\end{bmatrix} =
\begin{bmatrix}
1 & 1 & 1 & 1 \\
1 & -j & -1 & j \\
1 & -1 & 1 & -1 \\
1 & j & -1 & -j
\end{bmatrix}
\begin{bmatrix}
W_0^N F(0, q) \\
W_q^N F(1, q) \\
W_2^q F(2, q) \\
W_3^q F(3, q)
\end{bmatrix}
\]

Radix–4 FFT butterfly diagram
Note: The decomposition can be repeated $\nu = \log_4(N)$ times

Example: Radix–4 decimation in time FFT algorithm

Observations: Natural order input, shuffled (base 4 digit reverse) order output; graph multipliers represent $W_{16}$ exponent

Note: Decimation can also applied in the frequency domain

Example: Radix–4 decimation in frequency FFT algorithm

Observations: Samples are post weighted in each butterfly; graph multipliers represent $W_{16}$ exponent
Observations:
- If \(N = r^\nu\), then a radix \(r\) decomposition is most efficient
- Utilizing a radix-2 (split signal into two sequences of length \(4^\nu\)) followed by two radix-4 algorithms has advantages – split radix method

### Comparison of FFT Algorithm Complexities

<table>
<thead>
<tr>
<th>(N)</th>
<th>Real Multiplications</th>
<th>Real Additions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Radix 2</td>
<td>Radix 4</td>
</tr>
<tr>
<td>16</td>
<td>24</td>
<td>20</td>
</tr>
<tr>
<td>32</td>
<td>88</td>
<td>68</td>
</tr>
<tr>
<td>64</td>
<td>264</td>
<td>208</td>
</tr>
<tr>
<td>128</td>
<td>712</td>
<td>516</td>
</tr>
<tr>
<td>256</td>
<td>1,800</td>
<td>1,392</td>
</tr>
<tr>
<td>512</td>
<td>4,360</td>
<td>3,204</td>
</tr>
<tr>
<td>1,024</td>
<td>10,248</td>
<td>7,856</td>
</tr>
</tbody>
</table>

*Source: Extracted from Duhamel (1986).*

### FFT–Based Filtering

**Observations:** The IDFT differs from DFT only by scale term \(\frac{1}{N}\) and the exponent sign of \(W_N^{kn}\)

\[\Rightarrow\] Same decimation techniques hold for the IFFT \((N \log_2(N)\) complexity)

**Result:** Efficient FFT based linear filtering

1. Compute \(N\)-point FFT of \(h(n)\) and \(x(n)\), where \(N \geq M + L - 1\) (and a power of 2)
2. Multiply \(H(k)X(k)\)
3. Complete IFFT to get \(h(n) \ast x(n)\)

**Note:** For long data sequences
- Break \(x(n)\) into length \(L\) blocks
- Use overlap and save or overlap and add method
Lecture Summary

- FFT and IFFT calculation – complexity $O(N \log_2 N)$; radix–2, radix–4, and split–radix methods
- Radix–2 Decimation–in–Freq. (FFT) algorithm – Decompose signal $\log_2 N$ times; Basic computation is the butterfly
- Radix–4 Decimation–in–Time (FFT) algorithm – Decompose signal $\log_4 N$ times; Basic computation is the butterfly
- FFT–Based Filtering – Compute $N$–point FFT of $h(n)$ and $x(n)$, $N \geq M + L – 1$ (and a power of 2); multiply $H(k)X(k)$; complete IFFT to get $h(n) \ast x(n)$; process long signals block–wise
- Next lecture – Applications of FFT algorithms & linear filtering DFT computation (Chapter 8.2–8.3); Implementation of Discrete–Time