Digital Signal Processing

Lecture 8
FFT
Spectral Analysis

based on slides by J.M. Kahn
Announcements

- Last time:
  - Started FFT
- Today
  - Finish FFT
  - Start Frequency Analysis with DFT
- Read Ch. 10.1-10.2

- Who started playing with the SDR?
Most FFT algorithms exploit the following properties of $W_N^{kn}$:

- Conjugate Symmetry
  \[ W_N^{k(N-n)} = W_N^{-kn} = (W_N^{kn})^* \]

- Periodicity in $n$ and $k$:
  \[ W_N^{kn} = W_N^{k(n+N)} = W_N^{(k+N)n} \]

- Power:
  \[ W_N^2 = W_{N/2} \]
Most FFT algorithms decompose the computation of a DFT into successively smaller DFT computations.

- *Decimation-in-time* algorithms decompose $x[n]$ into successively smaller subsequences.
- *Decimation-in-frequency* algorithms decompose $X[k]$ into successively smaller subsequences.

We mostly discuss decimation-in-time algorithms here.

Assume length of $x[n]$ is power of 2 ($N = 2^\nu$). If smaller zero-pad to closest power.
**Decimation-in-Time Fast Fourier Transform**

- We start with the DFT

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

- Separate the sum into even and odd terms:

\[ X[k] = \sum_{n \text{ even}} x[n] W_N^{kn} + \sum_{n \text{ odd}} x[n] W_N^{kn} \]

These are two DFT’s, each with half of the samples.
Let $n = 2r$ (n even) and $n = 2r + 1$ (n odd):

$$X[k] = \sum_{r=0}^{(N/2) - 1} x[2r]W_N^{2rk} + \sum_{r=0}^{(N/2) - 1} x[2r + 1]W_N^{(2r+1)k}$$

$$= \sum_{r=0}^{(N/2) - 1} x[2r]W_N^{2rk} + W_N^k \sum_{r=0}^{(N/2) - 1} x[2r + 1]W_N^{2rk}$$

Note that:

$$W_N^{2rk} = e^{-j\left(\frac{2\pi}{N}\right)(2rk)} = e^{-j\left(\frac{2\pi}{N/2}\right)rk} = W_N^{rk}$$

Remember this trick, it will turn up often.
Decimation-in-Time Fast Fourier Transform

Hence:

\[
X[k] = \sum_{r=0}^{(N/2)-1} x[2r] W_N^{rk} + W_N^k \sum_{r=0}^{(N/2)-1} x[2r + 1] W_N^{rk}
\]

\[\triangleq G[k] + W_N^k H[k], \quad k = 0, \ldots, N - 1\]

where we have defined:

\[
G[k] \triangleq \sum_{r=0}^{(N/2)-1} x[2r] W_N^{rk} \quad \Rightarrow \text{DFT of even idx}
\]

\[
H[k] \triangleq \sum_{r=0}^{(N/2)-1} x[2r + 1] W_N^{rk} \quad \Rightarrow \text{DFT of odd idx}
\]
Decimation-in-Time Fast Fourier Transform

An 8 sample DFT can then be diagrammed as
Decimation-in-Time Fast Fourier Transform

- Both $G[k]$ and $H[k]$ are periodic, with period $N/2$. For example

\[
G[k + N/2] = \sum_{r=0}^{(N/2)-1} x[2r] W^{r(k+N/2)}_{N/2}
\]

\[
\begin{align*}
&= \sum_{r=0}^{(N/2)-1} x[2r] W^{rk}_{N/2} W^{(N/2)}_{N/2} \\
&= \sum_{r=0}^{(N/2)-1} x[2r] W^{rk}_{N/2} \\
&= G[k]
\end{align*}
\]

So

\[
G[k + (N/2)] = G[k]
\]

\[
H[k + (N/2)] = H[k]
\]
The periodicity of $G[k]$ and $H[k]$ allows us to further simplify.

For the first $N/2$ points we calculate $G[k]$ and $W_N^k H[k]$, and then compute the sum

$$X[k] = G[k] + W_N^k H[k] \quad \forall\{k : 0 \leq k < \frac{N}{2}\}.$$

How does periodicity help for $\frac{N}{2} \leq k < N$?
Decimation-in-Time Fast Fourier Transform

\[ X[k] = G[k] + W_N^k H[k] \quad \forall \{k : 0 \leq k < \frac{N}{2}\}. \]

- for \( \frac{N}{2} \leq k < N: \)

\[ W_N^{k+(N/2)} = ? \]

\[ X[k + (N/2)] = ? \]
\[ X[k + (N/2)] = G[k] - W_N^k H[k] \]

We previously calculated \( G[k] \) and \( W_N^k H[k] \).

Now we only have to compute their difference to obtain the second half of the spectrum. No additional multiplies are required.
Decimation-in-Time Fast Fourier Transform

- The $N$-point DFT has been reduced two $N/2$-point DFTs, plus $N/2$ complex multiplications. The 8 sample DFT is then:

\[
x[0] \quad x[1] \quad x[2] \quad x[3] \\
N/2 - Point \ DFT \\
W_N^0 \\
G[k] \\
X[0] \quad X[1] \quad X[2] \quad X[3]
\]

\[
N/2 - Point \ DFT \\
W_N^1 \quad W_N^2 \quad W_N^3 \\
H[k] \quad W_N^k \\
\]

Based on Course Notes by J.M Kahn
Note that the inputs have been reordered so that the outputs come out in their proper sequence.

We can define a *butterfly operation*, e.g., the computation of $X[0]$ and $X[4]$ from $G[0]$ and $H[0]$:

$$G[0]X[0] = G[0] + W_N^0 H[0]$$

$$H[0]W_N^0 X[4] = G[0] - W_N^0 H[0]$$

This is an important operation in DSP.
Decimation-in-Time Fast Fourier Transform

- Still $O(N^2)$ operations..... What shall we do?

![Diagram showing the process of the Decimation-in-Time Fast Fourier Transform](image)

- Even Samples
- Odd Samples

Diagram labels:
- $x[0]$
- $x[2]$
- $x[4]$
- $x[6]$
- $x[1]$
- $x[3]$
- $x[5]$
- $x[7]$
- $N/2$ - Point DFT
- $G[k]$
- $H[k]$
- $W_N^0$
- $W_N^1$
- $W_N^2$
- $W_N^3$
- $X[0]$
- $X[1]$
- $X[2]$
- $X[3]$
- $X[4]$
- $X[5]$
- $X[6]$
- $X[7]$

Miki Lustig UCB. Based on Course Notes by J.M Kahn SP 2014 EE123 Digital Signal Processing
We can use the same approach for each of the $N/2$ point DFT’s. For the $N = 8$ case, the $N/2$ DFTs look like

*Note that the inputs have been reordered again.
At this point for the 8 sample DFT, we can replace the $N/4 = 2$ sample DFT’s with a single butterfly. The coefficient is

$$W_{N/4} = W_{8/4} = W_2 = e^{-j\pi} = -1$$

The diagram of this stage is then

$$\begin{align*}
x[0] & \quad \rightarrow \quad x[0] + x[4] \\
x[4] & \quad \rightarrow \quad x[0] - x[4]
\end{align*}$$
Combining all these stages, the diagram for the 8 sample DFT is:

This is the decimation-in-time FFT algorithm.
In general, there are $\log_2 N$ stages of decimation-in-time.

Each stage requires $N/2$ complex multiplications, some of which are trivial.

The total number of complex multiplications is $(N/2) \log_2 N$.

The order of the input to the decimation-in-time FFT algorithm must be permuted.

- First stage: split into odd and even. Zero low-order bit first
- Next stage repeats with next zero-lower bit first.
- Net effect is reversing the bit order of indexes
Decimation-in-Time Fast Fourier Transform

This is illustrated in the following table for $N = 8$.

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
<th>Bit-Reversed Binary</th>
<th>Bit-Reversed Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>100</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
<td>010</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>011</td>
<td>110</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>101</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>011</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
<td>111</td>
<td>7</td>
</tr>
</tbody>
</table>
The DFT is

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

If we only look at the even samples of \( X[k] \), we can write \( k = 2r \),

\[ X[2r] = \sum_{n=0}^{N-1} x[n] W_N^{n(2r)} \]

We split this into two sums, one over the first \( N/2 \) samples, and the second of the last \( N/2 \) samples.

\[ X[2r] = \sum_{n=0}^{(N/2)-1} x[n] W_N^{2rn} + \sum_{n=0}^{(N/2)-1} x[n + N/2] W_N^{2r(n+N/2)} \]
Decimation-in-Frequency Fast Fourier Transform

But $W_N^{2r(n+N/2)} = W_N^{2rn}W_N^N = W_N^{2rn} = W_{N/2}^{rn}$. We can then write

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

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

$$= \sum_{n=0}^{(N/2)-1} (x[n] + x[n + N/2]) W_{N/2}^{rn}$$

This is the $N/2$-length DFT of first and second half of $x[n]$ summed.
Decimation-in-Frequency Fast Fourier Transform

\[ X[2r] = \text{DFT}_{N/2} \{ (x[n] + x[n + N/2]) \} \]
\[ X[2r + 1] = \text{DFT}_{N/2} \{ (x[n] - x[n + N/2]) W_N^n \} \]

(By a similar argument that gives the odd samples)

Continue the same approach is applied for the \( N/2 \) DFTs, and the \( N/4 \) DFT’s until we reach simple butterflies.
The diagram for an 8-point decimation-in-frequency DFT is as follows:

This is just the decimation-in-time algorithm reversed! The inputs are in normal order, and the outputs are bit reversed.
Non-Power-of-2 FFT’s

A similar argument applies for any length DFT, where the length $N$ is a composite number. For example, if $N = 6$, a decimation-in-time FFT could compute three 2-point DFT’s followed by two 3-point DFT’s.
Non-Power-of-2 FFT’s

Good component DFT’s are available for lengths up to 20 or so. Many of these exploit the structure for that specific length. For example, a factor of

\[ W_N^{N/4} = e^{-j\frac{2\pi}{N} (N/4)} = e^{-j\frac{\pi}{2}} = -j \]

Why?

just swaps the real and imaginary components of a complex number, and doesn’t actually require any multiplies. Hence a DFT of length 4 doesn’t require any complex multiplies. Half of the multiplies of an 8-point DFT also don’t require multiplication.

Composite length FFT’s can be very efficient for any length that factors into terms of this order.
For example $N = 693$ factors into

$$N = (7)(9)(11)$$

each of which can be implemented efficiently. We would perform

- $9 \times 11$ DFT’s of length 7
- $7 \times 11$ DFT’s of length 9, and
- $7 \times 9$ DFT’s of length 11
Historically, the power-of-two FFTs were much faster (better written and implemented).

For non-power-of-two length, it was faster to zero pad to power of two.

Recently this has changed. The free FFTW package implements very efficient algorithms for almost any filter length. **Matlab has used FFTW since version 6**
FFT computation time (Matlab FFTW) on MacBookPro
FFT as Matrix Operation

\[
\begin{pmatrix}
X[0] \\
\vdots \\
X[k] \\
\vdots \\
X[N - 1]
\end{pmatrix}
= 
\begin{pmatrix}
W_N^{00} & \cdots & W_N^{0n} & \cdots & W_N^{0(N-1)} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
W_N^{k0} & \cdots & W_N^{k*n} & \cdots & W_N^{k(N-1)} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
W_N^{(N-1)0} & \cdots & W_N^{(N-1)n} & \cdots & W_N^{(N-1)(N-1)}
\end{pmatrix}
\begin{pmatrix}
x[0] \\
\vdots \\
x[n] \\
\vdots \\
x[N - 1]
\end{pmatrix}
\]

- \(W_N\) is fully populated \(\Rightarrow N^2\) entries.
FFT as Matrix Operation

\[
\begin{pmatrix}
X[0] \\
\vdots \\
X[k] \\
\vdots \\
X[N-1]
\end{pmatrix} = 
\begin{pmatrix}
W_N^{00} & \cdots & W_N^{0n} & \cdots & W_N^{0(N-1)} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
W_N^{k0} & \cdots & W_N^{kn} & \cdots & W_N^{k(N-1)} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
W_N^{(N-1)0} & \cdots & W_N^{(N-1)n} & \cdots & W_N^{(N-1)(N-1)}
\end{pmatrix}
\begin{pmatrix}
x[0] \\
\vdots \\
x[n] \\
\vdots \\
x[N-1]
\end{pmatrix}
\]

- \(W_N\) is fully populated \(\Rightarrow N^2\) entries.
- FFT is a decomposition of \(W_N\) into a more sparse form:

\[
F_N = \begin{bmatrix}
I_{N/2} & D_{N/2} \\
I_{N/2} & -D_{N/2}
\end{bmatrix}
\begin{bmatrix}
W_{N/2} & 0 \\
0 & W_{N/2}
\end{bmatrix}
\begin{bmatrix}
\text{Even-Odd Perm.} \\
\text{Matrix}
\end{bmatrix}
\]

- \(I_{N/2}\) is an identity matrix. \(D_{N/2}\) is a diagonal with entries 1, \(W_N\), \(\cdots\), \(W_N^{N/2-1}\)
Example: $N = 4$

$$F_4 = \begin{bmatrix}
1 & 0 & 1 & 0 \\
0 & 1 & 0 & W_4 \\
1 & 0 & -1 & 0 \\
0 & 1 & 0 & -W_4 \\
\end{bmatrix} \begin{bmatrix}
1 & 1 & 0 & 0 \\
1 & -1 & 0 & 0 \\
0 & 0 & 1 & 1 \\
0 & 0 & 1 & -1 \\
\end{bmatrix} \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}$$
Beyond $\text{NlogN}$

- What if the signal $x[n]$ has a $k$ sparse frequency
  - A. Gilbert et. al, “Near-optimal sparse Fourier representations via sampling
  - H. Hassanieh et. al, “Nearly Optimal Sparse Fourier Transform”
  - Others......
- $O(K \log N)$ instead of $O(N \log N)$

From: http://groups.csail.mit.edu/netmit/sFFT/paper.html
Spectral Analysis with the DFT

The DFT can be used to analyze the spectrum of a signal.

It would seem that this should be simple, take a block of the signal and compute the spectrum with the DFT.

However, there are many important issues and tradeoffs:

- Signal duration vs spectral resolution
- Signal sampling rate vs spectral range
- Spectral sampling rate
- Spectral artifacts
Consider these steps of processing continuous-time signals:
Spectral Analysis with the DFT

Two important tools:

- Applying a window to the input signal – reduces spectral artifacts
- Padding input signal with zeros – increases the spectral sampling

Key Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sampling interval</td>
<td>$T$</td>
<td>s</td>
</tr>
<tr>
<td>Sampling frequency</td>
<td>$\Omega_s = \frac{2\pi}{T}$</td>
<td>rad/s</td>
</tr>
<tr>
<td>Window length</td>
<td>$L$</td>
<td>unitless</td>
</tr>
<tr>
<td>Window duration</td>
<td>$L \cdot T$</td>
<td>s</td>
</tr>
<tr>
<td>DFT length</td>
<td>$N \geq L$</td>
<td>unitless</td>
</tr>
<tr>
<td>DFT duration</td>
<td>$N \cdot T$</td>
<td>s</td>
</tr>
<tr>
<td>Spectral resolution</td>
<td>$\frac{\Omega_s}{L} = \frac{2\pi}{L \cdot T}$</td>
<td>rad/s</td>
</tr>
<tr>
<td>Spectral sampling interval</td>
<td>$\frac{\Omega_s}{N} = \frac{2\pi}{N \cdot T}$</td>
<td>rad/s</td>
</tr>
</tbody>
</table>
We consider an example:

\[ x_c(t) = A_1 \cos \omega_1 t + A_2 \cos \omega_2 t \]

\[ X_c(j\Omega) = A_1 \pi [\delta(\Omega - \omega_1) + \delta(\Omega + \omega_1)] + A_2 \pi [\delta(\Omega - \omega_2) + \delta(\Omega + \omega_2)] \]
Sampled Signal
If we sampled the signal over an infinite time duration, we would have:

\[ x[n] = x_c(t)\big|_{t=nT}, \quad -\infty < n < \infty \]

described by the discrete-time Fourier transform:

\[ X(e^{j\Omega T}) = \frac{1}{T} \sum_{r=-\infty}^{\infty} X_c \left( j \left( \Omega - r \frac{2\pi}{T} \right) \right), \quad -\infty < \Omega < \infty \]

Recall \( X(e^{j\omega}) = X(e^{j\Omega T}) \), where \( \omega = \Omega T \) ... more in ch 4.
In the examples shown here, the sampling rate is $\Omega_s/2\pi = 1/T = 20$ Hz, sufficiently high that aliasing does not occur.
**Block of $L$ Signal Samples**

In any real system, we sample only over a finite block of $L$ samples:

$$\mathbf{x}[n] = \mathbf{x}_c(t)|_{t=nT}, \quad 0 \leq n \leq L - 1$$

This simply corresponds to a rectangular window of duration $L$.

Recall: in Homework 1 we explored the effect of rectangular and triangular windowing
Windowed Block of $L$ Signal Samples
We take the block of signal samples and multiply by a window of duration $L$, obtaining:

$$v[n] = x[n] \cdot w[n], \quad 0 \leq n \leq L - 1$$

Suppose the window $w[n]$ has DTFT $W(e^{j\omega})$.

Then the windowed block of signal samples has a DTFT given by the periodic convolution between $X(e^{j\omega})$ and $W(e^{j\omega})$:

$$V(e^{j\omega}) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\theta}) W(e^{j(\omega-\theta)}) d\theta$$
Convolution with \( W(e^{j\omega}) \) has two effects in the spectrum:

1. It limits the spectral resolution. – Main lobes of the DTFT of the window
2. The window can produce \textit{spectral leakage}. – Side lobes of the DTFT of the window

* These two are always a tradeoff - time-frequency uncertainty principle
## Windows (as defined in MATLAB)

<table>
<thead>
<tr>
<th>Name(s)</th>
<th>Definition</th>
<th>MATLAB Command</th>
<th>Graph ($M = 8$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rectangular Boxcar Fourier</td>
<td>$w[n] = \begin{cases} 1 &amp;</td>
<td>n</td>
<td>\leq M/2 \ 0 &amp;</td>
</tr>
<tr>
<td>Triangular</td>
<td>$w[n] = \begin{cases} 1 - \frac{</td>
<td>n</td>
<td>}{M/2 + 1} &amp;</td>
</tr>
<tr>
<td>Bartlett</td>
<td>$w[n] = \begin{cases} 1 - \frac{</td>
<td>n</td>
<td>}{M/2} &amp;</td>
</tr>
</tbody>
</table>
## Windows (as defined in MATLAB)

<table>
<thead>
<tr>
<th>Name(s)</th>
<th>Definition</th>
<th>MATLAB Command</th>
<th>Graph (M = 8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hann</td>
<td>[ w[n] = \begin{cases} \frac{1}{2} \left[ 1 + \cos\left( \frac{\pi n}{M/2} \right) \right] &amp; \text{if }</td>
<td>n</td>
<td>\leq M/2 \ 0 &amp; \text{if }</td>
</tr>
<tr>
<td>Hanning</td>
<td>[ w[n] = \begin{cases} \frac{1}{2} \left[ 1 + \cos\left( \frac{\pi n}{M/2 + 1} \right) \right] &amp; \text{if }</td>
<td>n</td>
<td>\leq M/2 \ 0 &amp; \text{if }</td>
</tr>
<tr>
<td>Hamming</td>
<td>[ w[n] = \begin{cases} 0.54 + 0.46 \cos\left( \frac{\pi n}{M/2} \right) &amp; \text{if }</td>
<td>n</td>
<td>\leq M/2 \ 0 &amp; \text{if }</td>
</tr>
</tbody>
</table>
Windows

- All of the window functions $w[n]$ are real and even.
- All of the discrete-time Fourier transforms

$$W(e^{j\omega}) = \sum_{n=-\frac{M}{2}}^{\frac{M}{2}} w[n]e^{-jn\omega}$$

are real, even, and periodic in $\omega$ with period $2\pi$.

- In the following plots, we have normalized the windows to unit d.c. gain:

$$W(e^{j0}) = \sum_{n=-\frac{M}{2}}^{\frac{M}{2}} w[n] = 1$$

This makes it easier to compare windows.
Miki Lustig UCB. Based on Course Notes by J.M Kahn

Spring 2014, EE123 Digital Signal Processing