EE482: Digital Signal Processing Applications

Fast Fourier Transform

http://www.ee.unlv.edu/~b1morris/ee482/
Outline

- Fast Fourier Transform
- Butterfly Structure
- Implementation Issues
DFT Algorithm

◆ The Fourier transform of an analogue signal $x(t)$ is given by:

$$X(\omega) = \int_{-\infty}^{+\infty} x(t)e^{-j\omega t} \, dt$$

◆ The Discrete Fourier Transform (DFT) of a discrete-time signal $x(nT)$ is given by:

$$X(k) = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{N}nk}$$

◆ Where:

$k = 0, 1, \ldots, N - 1$

$x(nT) = x[n]$
DFT Algorithm

- If we let: \( e^{-j\frac{2\pi}{N}} = W_N \) then: \( X(k) = \sum_{n=0}^{N-1} x[n]W_N^{nk} \)
DFT Algorithm

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

\( x[n] = \) input

\( X[k] = \) frequency bins

\( W = \) twiddle factors

\[
\begin{align*}
X(0) &= x[0] W_N^0 + x[1] W_N^{0*1} + \ldots + x[N-1] W_N^{0*(N-1)} \\
X(1) &= x[0] W_N^0 + x[1] W_N^{1*1} + \ldots + x[N-1] W_N^{1*(N-1)} \\
& \vdots \\
X(k) &= x[0] W_N^0 + x[1] W_N^{k*1} + \ldots + x[N-1] W_N^{k*(N-1)} \\
& \vdots \\
X(N-1) &= x[0] W_N^0 + x[1] W_N^{(N-1)*1} + \ldots + x[N-1] W_N^{(N-1)(N-1)}
\end{align*}
\]

Note: For \( N \) samples of \( x \) we have \( N \) frequencies representing the signal.
Performance of the DFT Algorithm

- The DFT requires $N^2$ (NxN) complex multiplications:
  - Each $X(k)$ requires $N$ complex multiplications.
  - Therefore to evaluate all the values of the DFT ($X(0)$ to $X(N-1)$) $N^2$ multiplications are required.

- The DFT also requires $(N-1)*N$ complex additions:
  - Each $X(k)$ requires $N-1$ additions.
  - Therefore to evaluate all the values of the DFT $(N-1)*N$ additions are required.
Can the number of computations required be reduced?
A large amount of work has been devoted to reducing the computation time of a DFT.

This has led to efficient algorithms which are known as the Fast Fourier Transform (FFT) algorithms.
DFT → FFT

\[ X(k) = \sum_{n=0}^{N-1} x[n] W_N^{nk} ; \quad 0 \leq k \leq N - 1 \]  \hspace{1cm} [1]

\( x[n] = x[0], x[1], \ldots, x[N-1] \)

- **Lets divide the sequence** \( x[n] \) **into even and odd sequences:**
  - \( x[2n] = x[0], x[2], \ldots, x[N-2] \)
  - \( x[2n+1] = x[1], x[3], \ldots, x[N-1] \)
DFT → FFT

- **Equation 1 can be rewritten as:**

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

- **Since:**

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

\[ W_N^{(2n+1)k} = W_N^{k} \cdot W_N^{nk} \]

- **Then:**

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

\[ = Y(k) + W_N^{k} Z(k) \]
DFT $\rightarrow$ FFT

- The result is that an N-point DFT can be divided into two N/2 point DFT's:

\[
X(k) = \sum_{n=0}^{N-1} x[n]W_N^{nk}; \quad 0 \leq k \leq N - 1
\]

N-point DFT

- Where $Y(k)$ and $Z(k)$ are the two N/2 point DFTs operating on even and odd samples respectively:

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

\[
= Y(k) + W_N^kZ(k)
\]

Two N/2-point DFTs
DFT → FFT

**Periodicity and symmetry** of $W$ can be exploited to simplify the DFT further:

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

\[
X\left(k + \frac{N}{2}\right) = \sum_{n=0}^{N-1} x_1[n] W_n^{\left(k + \frac{N}{2}\right)} + W_N^k \sum_{n=0}^{N-1} x_2[n] W_n^{\left(k + \frac{N}{2}\right)}
\]

**Or:**

\[
W_N^{k+N/2} = e^{-j \frac{2\pi}{N} k} e^{-j \frac{2\pi N}{2}} = e^{-j \frac{2\pi}{N} k} e^{-j\pi} = -e^{-j \frac{2\pi}{N} k} = -W_N^k
\]

**And:**

\[
W_N^{k+N/2} = e^{-j \frac{2\pi}{N/2} k} e^{-j \frac{2\pi N}{2/2}} = e^{-j \frac{2\pi}{N/2} k} = W_N^k
\]

: Symmetry

: Periodicity
DFT $\Rightarrow$ FFT

- Symmetry and periodicity:

\[ W_8^0 = W_8^8 \]
\[ W_8^1 = W_8^9 \]

\[ W_N^{k+N/2} = -W_N^k \]
\[ W_{N/2}^{k+N/2} = W_{N/2}^k \]
\[ W_8^{k+4} = -W_8^k \]
\[ W_8^{k+8} = W_8^k \]
Finally by exploiting the symmetry and periodicity, Equation 3 can be written as:

\[
X\left( k + \frac{N}{2} \right) = \sum_{n=0}^{N-1/2} x_1[n] W_N^{nk} - W_N^k \sum_{n=0}^{N-1} x_2[n] W_N^{nk}
\]

\[
= Y(k) - W_N^k Z(k)
\]
DFT $\rightarrow$ FFT

\[
X(k) = Y(k) + W_N^k Z(k); \quad k = 0, \ldots \left(\frac{N}{2} - 1\right)
\]
\[
X\left(k + \frac{N}{2}\right) = Y(k) - W_N^k Z(k); \quad k = 0, \ldots \left(\frac{N}{2} - 1\right)
\]

- $Y(k)$ and $W_N^k Z(k)$ only need to be calculated once and used for both equations.

- Note: the calculation is reduced from 0 to $N-1$ to 0 to $(N/2 - 1)$.
Y(k) and Z(k) can also be divided into $N/4$ point DFTs using the same process shown above:

\[
X(k) = Y(k) + W_N^k Z(k), \quad k = 0, \ldots, \left(\frac{N}{2} - 1\right)
\]

\[
X\left(k + \frac{N}{2}\right) = Y(k) - W_N^k Z(k), \quad k = 0, \ldots, \left(\frac{N}{2} - 1\right)
\]

\[
Y(k) = U(k) + W_N^k V(k), \quad Z(k) = P(k) + W_N^k Q(k)
\]

\[
Y\left(k + \frac{N}{4}\right) = U(k) - W_N^k V(k), \quad Z\left(k + \frac{N}{4}\right) = P(k) - W_N^k Q(k)
\]

The process continues until we reach 2 point DFTs.
Illustration of the first decimation in time FFT.
FFT Implementation

- To efficiently implement the FFT algorithm a few observations are made:
  - Each stage has the same number of butterflies (number of butterflies = N/2, N is number of points).
  - The number of DFT groups per stage is equal to \( \frac{N}{2^{\text{stage}}} \).
  - The difference between the upper and lower leg is equal to \( 2^{\text{stage}-1} \).
  - The number of butterflies in the group is equal to \( 2^{\text{stage}-1} \).
FFT Implementation

Decimation in time FFT:
- Number of stages = \( \log_2 N \)
- Number of blocks/stage = \( \frac{N}{2^{\text{stage}}} \)
- Number of butterflies/block = \( 2^{\text{stage}-1} \)

Example: 8 point FFT
Decimation in time FFT:

- **Number of stages** = $\log_2 N$
- **Number of blocks/stage** = $N/2^{\text{stage}}$
- **Number of butterflies/block** = $2^{\text{stage}-1}$

Example: 8 point FFT

(1) Number of stages:
Decimation in time FFT:

- Number of stages = \( \log_2 N \)
- Number of blocks/stage = \( N/2^{\text{stage}} \)
- Number of butterflies/block = \( 2^{\text{stage}-1} \)

Example: 8 point FFT

1. Number of stages:
   - \( N_{\text{stages}} = 1 \)
Example: 8 point FFT

(1) Number of stages:
   - \( N_{\text{stages}} = 2 \)

◆ Decimation in time FFT:
   - Number of stages = \( \log_2 N \)
   - Number of blocks/stage = \( N/2^{\text{stage}} \)
   - Number of butterflies/block = \( 2^{\text{stage-1}} \)
Example: 8 point FFT

(1) Number of stages:
- \( N_{\text{stages}} = 3 \)

- Decimation in time FFT:
  - Number of stages = \( \log_2 N \)
  - Number of blocks/stage = \( N / 2^{\text{stage}} \)
  - Number of butterflies/block = \( 2^{\text{stage}-1} \)
Example: 8 point FFT

1. Number of stages:
   - \( N_{\text{stages}} = 3 \)

2. Blocks/stage:
   - **Stage 1:**

**Decimation in time FFT:**

- Number of stages = \( \log_2 N \)
- Number of blocks/stage = \( N/2^{\text{stage}} \)
- Number of butterflies/block = \( 2^{\text{stage}-1} \)
Example: 8 point FFT

1. Number of stages:
   - \( N_{\text{stages}} = 3 \)

2. Blocks/stage:
   - Stage 1: \( N_{\text{blocks}} = 1 \)

- **Decimation in time FFT:**
  - Number of stages = \( \log_2 N \)
  - Number of blocks/stage = \( \frac{N}{2^{\text{stage}}} \)
  - Number of butterflies/block = \( 2^{\text{stage}-1} \)
Example: 8 point FFT

(1) Number of stages:
   - $N_{\text{stages}} = 3$

(2) Blocks/stage:
   - Stage 1: $N_{\text{blocks}} = 2$

Decimation in time FFT:

- Number of stages = $\log_2 N$
- Number of blocks/stage = $N/2^{\text{stage}}$
- Number of butterflies/block = $2^{\text{stage}-1}$
Example: 8 point FFT

1. Number of stages:
   - \( N_{\text{stages}} = 3 \)

2. Blocks/stage:
   - Stage 1: \( N_{\text{blocks}} = 3 \)

Decimation in time FFT:

- Number of stages = \( \log_2 N \)
- Number of blocks/stage = \( N/2^{\text{stage}} \)
- Number of butterflies/block = \( 2^{\text{stage}-1} \)
Example: 8 point FFT

1. Number of stages:
   - $N_{\text{stages}} = 3$

2. Blocks/stage:
   - **Stage 1**: $N_{\text{blocks}} = 4$

Decimation in time FFT:

- Number of stages = $\log_2 N$
- Number of blocks/stage = $N/2^{\text{stage}}$
- Number of butterflies/block = $2^{\text{stage}-1}$
Example: 8 point FFT

(1) Number of stages:
- \( N_{\text{stages}} = 3 \)

(2) Blocks/stage:
- Stage 1: \( N_{\text{blocks}} = 4 \)
- Stage 2: \( N_{\text{blocks}} = 1 \)

**Decimation in time FFT:**
- Number of stages = \( \log_2 N \)
- Number of blocks/stage = \( N/2^{\text{stage}} \)
- Number of butterflies/block = \( 2^{\text{stage}-1} \)
Example: 8 point FFT

1. Number of stages:
   - \( N_{\text{stages}} = 3 \)

2. Blocks/stage:
   - Stage 1: \( N_{\text{blocks}} = 4 \)
   - Stage 2: \( N_{\text{blocks}} = 2 \)

◆ Decimation in time FFT:
   1. Number of stages = \( \log_2 N \)
   2. Number of blocks/stage = \( N/2^{\text{stage}} \)
   3. Number of butterflies/block = \( 2^{\text{stage}-1} \)
Example: 8 point FFT

(1) Number of stages:
- \( N_{\text{stages}} = 3 \)

(2) Blocks/stage:
- Stage 1: \( N_{\text{blocks}} = 4 \)
- Stage 2: \( N_{\text{blocks}} = 2 \)
- Stage 3: \( N_{\text{blocks}} = 1 \)

Decimation in time FFT:
- Number of stages = \( \log_2 N \)
- Number of blocks/stage = \( N/2^{\text{stage}} \)
- Number of butterflies/block = \( 2^{\text{stage}-1} \)
Example: 8 point FFT

1) Number of stages:
   - $N_{\text{stages}} = 3$

2) Blocks/stage:
   - Stage 1: $N_{\text{blocks}} = 4$
   - Stage 2: $N_{\text{blocks}} = 2$
   - Stage 3: $N_{\text{blocks}} = 1$

3) B’flies/block:
   - Stage 1:

Decimation in time FFT:

- Number of stages = $\log_2 N$
- Number of blocks/stage = $\frac{N}{2^{\text{stage}}}$
- Number of butterflies/block = $2^{\text{stage}-1}$
Example: 8 point FFT

(1) Number of stages:
   - \( N_{\text{stages}} = 3 \)

(2) Blocks/stage:
   - Stage 1: \( N_{\text{blocks}} = 4 \)
   - Stage 2: \( N_{\text{blocks}} = 2 \)
   - Stage 3: \( N_{\text{blocks}} = 1 \)

(3) B’flies/block:
   - Stage 1: \( N_{\text{btf}} = 1 \)

**Decimation in time FFT:**

- Number of stages = \( \log_2 N \)
- Number of blocks/stage = \( N/2^{\text{stage}} \)
- Number of butterflies/block = \( 2^{\text{stage}-1} \)
Example: 8 point FFT

(1) Number of stages:
- $N_{\text{stages}} = 3$

(2) Blocks/stage:
- Stage 1: $N_{\text{blocks}} = 4$
- Stage 2: $N_{\text{blocks}} = 2$
- Stage 3: $N_{\text{blocks}} = 1$

(3) B’flies/block:
- Stage 1: $N_{\text{btf}} = 1$
- Stage 2: $N_{\text{btf}} = 1$

Decimation in time FFT:
- Number of stages = $\log_2 N$
- Number of blocks/stage = $N/2^{\text{stage}}$
- Number of butterflies/block = $2^{\text{stage}-1}$
Example: 8 point FFT

(1) Number of stages:
   - \( N_{\text{stages}} = 3 \)

(2) Blocks/stage:
   - Stage 1: \( N_{\text{blocks}} = 4 \)
   - Stage 2: \( N_{\text{blocks}} = 2 \)
   - Stage 3: \( N_{\text{blocks}} = 1 \)

(3) B’flies/block:
   - Stage 1: \( N_{\text{btf}} = 1 \)
   - Stage 2: \( N_{\text{btf}} = 2 \)

◆ Decimation in time FFT:
   - Number of stages = \( \log_2 N \)
   - Number of blocks/stage = \( N/2^{\text{stage}} \)
   - Number of butterflies/block = \( 2^{\text{stage}-1} \)
Example: 8 point FFT

(1) Number of stages:
- \( N_{\text{stages}} = 3 \)

(2) Blocks/stage:
- Stage 1: \( N_{\text{blocks}} = 4 \)
- Stage 2: \( N_{\text{blocks}} = 2 \)
- Stage 3: \( N_{\text{blocks}} = 1 \)

(3) B’flies/block:
- Stage 1: \( N_{\text{btf}} = 1 \)
- Stage 2: \( N_{\text{btf}} = 2 \)
- Stage 3: \( N_{\text{btf}} = 1 \)

 deceived in time FFT:
- Number of stages = \( \log_2 N \)
- Number of blocks/stage = \( N/2^{\text{stage}} \)
- Number of butterflies/block = \( 2^{\text{stage}-1} \)
Example: 8 point FFT

1. Number of stages:
   - \( N_{\text{stages}} = 3 \)

2. Blocks/stage:
   - Stage 1: \( N_{\text{blocks}} = 4 \)
   - Stage 2: \( N_{\text{blocks}} = 2 \)
   - Stage 3: \( N_{\text{blocks}} = 1 \)

3. B’flies/block:
   - Stage 1: \( N_{\text{btf}} = 1 \)
   - Stage 2: \( N_{\text{btf}} = 2 \)
   - Stage 3: \( N_{\text{btf}} = 2 \)

Decimation in time FFT:

- Number of stages = \( \log_2 N \)
- Number of blocks/stage = \( N/2^{\text{stage}} \)
- Number of butterflies/block = \( 2^{\text{stage}-1} \)
Example: 8 point FFT

(1) Number of stages:
   - $N_{\text{stages}} = 3$

(2) Blocks/stage:
   - Stage 1: $N_{\text{blocks}} = 4$
   - Stage 2: $N_{\text{blocks}} = 2$
   - Stage 3: $N_{\text{blocks}} = 1$

(3) B’flies/block:
   - Stage 1: $N_{\text{btf}} = 1$
   - Stage 2: $N_{\text{btf}} = 2$
   - Stage 3: $N_{\text{btf}} = 3$

◆ Decimation in time FFT:
   - Number of stages = $\log_2 N$
   - Number of blocks/stage = $N/2^{\text{stage}}$
   - Number of butterflies/block = $2^{\text{stage}-1}$
Example: 8 point FFT

1. Number of stages:
   - $N_{\text{stages}} = 3$

2. Blocks/stage:
   - Stage 1: $N_{\text{blocks}} = 4$
   - Stage 2: $N_{\text{blocks}} = 2$
   - Stage 3: $N_{\text{blocks}} = 1$

3. B’flies/block:
   - Stage 1: $N_{\text{btf}} = 1$
   - Stage 2: $N_{\text{btf}} = 2$
   - Stage 3: $N_{\text{btf}} = 4$

Decimation in time FFT:

- Number of stages = $\log_2 N$
- Number of blocks/stage = $N/2^{\text{stage}}$
- Number of butterflies/block = $2^{\text{stage}-1}$
FFT Implementation

Stage 1

Stage 2

Stage 3

Start Index 0 0 0
Input Index 1 2 4
Twiddle Factor Index N/2 = 4
FFT Implementation

Start Index
Input Index
Twiddle Factor Index

Stage 1
Stage 2
Stage 3

W_0 -1
W_0 -1
W_0 -1

W_0 -1
W_2 -1

W_0 -1
W_2 -1
W_4 -1

0
0
0

1
2
4

N/2 = 4
4/2 = 2
FFT Implementation

Start Index: 0, 0, 0
Input Index: 1, 2, 4
Twiddle Factor Index: N/2 = 4, 4/2 = 2, 2/2 = 1
FFT Implementation

<table>
<thead>
<tr>
<th>Stage 1</th>
<th>Stage 2</th>
<th>Stage 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Diagram" /></td>
<td><img src="image2" alt="Diagram" /></td>
<td><img src="image3" alt="Diagram" /></td>
</tr>
</tbody>
</table>

- **Start Index**: 0, 0, 0
- **Input Index**: 1, 2, 4
- **Twiddle Factor Index**: \(N/2 = 4\), \(4/2 = 2\), \(2/2 = 1\)
- **Indices Used**: \(W_0\), \(W_0\), \(W_0\), \(W_2\), \(W_1\), \(W_2\), \(W_3\)
FFT Decimation in Frequency

• Similar divide and conquer strategy
  ▫ Decimate in frequency domain
• \( X(2k) = \sum_{n=0}^{N-1} x(n)W_N^{2nk} \)
• \( X(2k) = \sum_{n=0}^{N/2-1} x(n)W_{N/2}^{nk} + \sum_{n=N/2}^{N-1} x(n)W_{N/2}^{nk} \)
  ▫ Divide into first half and second half of sequence
• \( X(2k) = \sum_{n=0}^{N/2-1} x(n)W_{N/2}^{nk} + \sum_{n=0}^{N/2-1} x(n) + \)
FFT Decimation in Frequency Structure

- Stage structure

- Full structure

- Bit reversal happens at output instead of input
Inverse FFT

- \( x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) W_N^{-kn} \)
- Notice this is the DFT with a scale factor and change in twiddle sign
- Can compute using the FFT with minor modifications
  - \( x^*(n) = \frac{1}{N} \sum_{k=0}^{N-1} X^*(k) W_N^{kn} \)
    - Conjugate coefficients, compute FFT with scale factor, conjugate result
    - For real signals, no final conjugate needed
  - Can complex conjugate twiddle factors and use in butterfly structure
FFT Example

- Example 5.10
- Sine wave with \( f = 50 \) Hz
  - \( x(n) = \sin\left(\frac{2\pi fn}{f_s}\right) \)
    - \( n = 0, 1, \ldots, 128 \)
    - \( f_s = 256 \) Hz

- Frequency resolution of DFT?
  - \( \Delta = f_s/N = \frac{256}{128} = 2 \) Hz

- Location of peak
  - \( 50 = k\Delta \rightarrow k = \frac{50}{2} = 25 \)
Spectral Leakage and Resolution

- Notice that a DFT is like windowing a signal to finite length
  - Longer window lengths (more samples) the closer DFT $X(k)$ approximates DTFT $X(\omega)$
- Convolution relationship
  - $x_N(n) = w(n)x(n)$
  - $X_N(k) = W(k) \ast X(k)$
- Corruption of spectrum due to window properties (mainlobe/sidelobe)
  - Sidelobes result in spurious peaks in computed spectrum known as spectral leakage
    - Obviously, want to use smoother windows to minimize these effects
  - Spectral smearing is the loss in sharpness due to convolution which depends on mainlobe width

Example 5.15
- Two close sinusoids smeared together

To avoid smearing:
- Frequency separation should be greater than freq resolution
- $N > \frac{2\pi}{\Delta \omega}, \quad N > f_s/\Delta f$
Power Spectral Density

- Parseval’s theorem
  \[ E = \sum_{n=0}^{N-1} |x(n)|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X(k)|^2 \]
  - \( |X(k)|^2 \) - power spectrum or periodogram

- Power spectral density (PSD, or power density spectrum or power spectrum) is used to measure average power over frequencies

- Computed for time-varying signal by using a sliding window technique
  - Short-time Fourier transform
  - Grab \( N \) samples and compute FFT
    - Must have overlap and use windows

- Spectrogram
  - Each short FFT is arranged as a column in a matrix to give the time-varying properties of the signal
  - Viewed as an image

"She had your dark suit in greasy wash water all year"
Fast FFT Convolution

- Linear convolution is multiplication in frequency domain
  - Must take FFT of signal and filter, multiply, and iFFT
  - Operations in frequency domain can be much faster for large filters
  - Requires zero-padding because of circular convolution
- Typically, will do block processing
  - Segment a signal and process each segment individually before recombining
Ex: FFT Effect of N

- Take FFT of cosine using different N values

\[
n = \{0:29\};
\]
\[
x = \cos(2*\pi*n/10);
\]
\[
N1 = 64;
\]
\[
N2 = 128;
\]
\[
N3 = 256;
\]
\[
X1 = \text{abs}(\text{fft}(x,N1));
\]
\[
X2 = \text{abs}(\text{fft}(x,N2));
\]
\[
X3 = \text{abs}(\text{fft}(x,N3));
\]
\[
F1 = \{0 : N1 - 1\}/N1;
\]
\[
F2 = \{0 : N2 - 1\}/N2;
\]
\[
F3 = \{0 : N3 - 1\}/N3;
\]

Subplotting the results:

- Transforms all have the same shape
- Difference is the number of samples used to approximate the shape

- Notice the sinusoid frequency is not always well represented
  - Depends on frequency resolution
Ex: FFT Effect of Number of Samples

- Select a large value of \( N \) and vary the number of samples of the signal

\[
\begin{align*}
n &= [0:29]; \\
x1 &= \cos(2\pi n/10); \ % \ 3 \ periods \\
x2 &= [x1 \ x1]; \ % \ 6 \ periods \\
x3 &= [x1 \ x1 \ x1]; \ % \ 9 \ periods \\
N &= 2048; \\
X1 &= \text{abs}(\text{fft}(x1,N)); \\
X2 &= \text{abs}(\text{fft}(x2,N)); \\
X3 &= \text{abs}(\text{fft}(x3,N)); \\
F &= [0:N-1]/N; \\
\text{subplot}(3,1,1) \\
\text{plot}(F,X1),\text{title('3 periods')},\text{axis([0 \ 1 \ 0 \ 50])} \\
\text{subplot}(3,1,2) \\
\text{plot}(F,X2),\text{title('6 periods')},\text{axis([0 \ 1 \ 0 \ 50])} \\
\text{subplot}(3,1,3) \\
\text{plot}(F,X3),\text{title('9 periods')},\text{axis([0 \ 1 \ 0 \ 50])}
\end{align*}
\]

- Transforms all have the same shape
  - Looks like sinc functions
- More samples makes the sinc look more impulse-like
- FFT with large \( N \) but fewer samples does zero-padding
  - E.g. taking length \( N \) signal and windowing with box
  - Multiplication in time is convolution in frequency
Spectrum Analysis with FFT and Matlab

- FFT does not directly give spectrum
  - Dependent on the number of signal samples
  - Dependent on the number of points in the FFT
- FFT contains info between $[0, f_s]$
  - Spectrum must be below $f_s/2$
- Symmetric across $f = 0$ axis
  - $\left[ -\frac{f_s}{2}, \frac{f_s}{2} \right]$
  - Use `fftshift.m` in Matlab

```matlab
n = [0:149];
x1 = cos(2*pi*n/10);
N = 2048;
X = abs(fft(x1,N));
X = fftshift(X);
F = [-N/2:N/2-1]/N;
plot(F,X),
xlabel('frequency / f s')
```