3F3 – Digital Signal Processing (DSP)

Simon Godsill

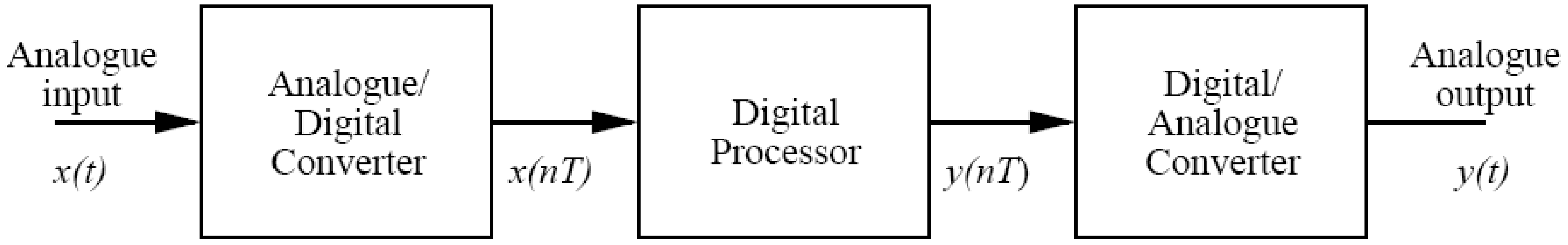
www-sigproc.eng.cam.ac.uk/~sjg/teaching/3F3

Course Overview

- 11 Lectures
- Topics:
  - Digital Signal Processing
  - DFT, FFT
  - Digital Filters
  - Filter Design
  - Filter Implementation
  - Random signals
  - Optimal Filtering
  - Signal Modelling
- Books:
  - J.G. Proakis and D.G. Manolakis, Digital Signal Processing 3rd edition, Prentice-Hall.
  - Statistical digital signal processing and modeling -Monson H. Hayes –Wiley
- Some material adapted from courses by Dr. Malcolm Macleod, Prof. Peter Rayner and Dr. Arnaud Doucet

Digital Signal Processing - Introduction

- Digital signal processing (DSP) is the generic term for techniques such as filtering or spectrum analysis applied to digitally sampled signals.
- Recall from 1B Signal and Data Analysis that the procedure is as shown below:



- T is the sampling period
- $f_0 = 1/T$ is the sampling frequency
- Recall also that low-pass anti-aliasing filters must be applied before A/D and D/A conversion in order to remove distortion from frequency components higher than $f_0/2$ Hz (see later for revision of this).

- Digital signals are signals which are sampled in time (“discrete time”) and quantised.

- Mathematical analysis of inherently digital signals (e.g. sunspot data, tide data) was developed by Gauss (1800), Schuster (1896) and many others since.

- Electronic digital signal processing (DSP) was first extensively applied in geophysics (for oil-exploration) then military applications, and is now fundamental to communications, mobile devices, broadcasting, and most applications of signal and image processing.

There are many advantages in carrying out digital rather than analogue processing; among these are flexibility and repeatability.

The flexibility stems from the fact that system parameters are simply numbers stored in the processor. Thus for example, it is a trivial matter to change the cut-off frequency of a digital filter whereas a lumped element analogue filter would require a different set of passive components. Indeed the ease with which system parameters can be changed has led to many adaptive techniques whereby the system parameters are modified in real time according to some algorithm. Examples of this are adaptive equalisation of transmission systems, adaptive antenna arrays which automatically steer the nulls in the polar diagram onto interfering signals. Digital signal processing enables very Digital signal processing enables very complex linear and non-linear processes to be implemented which would not be feasible with analogue processing. For example it is difficult to envisage an analogue system which could be used to perform spatial filtering of an image to improve the signal to noise ratio.

DSP has been an active research area since the late 1960s but applications tended to be only in large and expensive systems or in non real-time where a general purpose computer could be used. However, the advent of d.s.p chips enable real-time processing to be performed at very low cost and already this technology is commonplace in domestic products.

Sampling Theorem (revision from 1B)

For a continuous time signal $g(t)$ with Fourier transform $G(\omega)$, the Fourier transform of the corresponding sampled signal $g_s(t)$, sampled at a rate $\omega_0 = 2\pi/T$ rad.s^{-1} is given by:

$$\boxed{G_s(\omega) = \frac{1}{T}\sum_{n=-\infty}^{\infty} G(\omega - n\omega_0)} \quad (1)$$

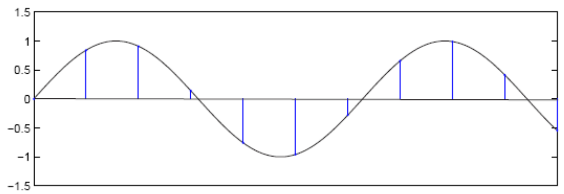
i.e. **The Fourier transform of the sampled signal is simply a summation of shifted versions of the original Fourier transform of the continuous signal $G(\omega)$**

Implications:

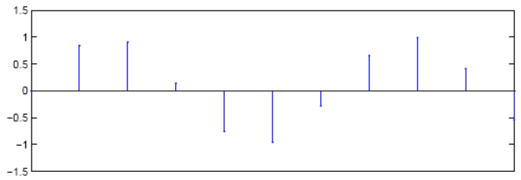
- Spectrum of sampled signal is always *periodic*.
- Can reconstruct an analogue signal *perfectly* from its digital samples provided its bandwidth is $< \omega_0/2$.
- For signals with bandwidth $\geq \omega_0/2$ we must pre-filter with an ideal lowpass filter having cut-off frequency $\omega_0/2$ to prevent *aliasing*
- Practical considerations - design of anti-aliasing filters with finite transition bandwidth, ...

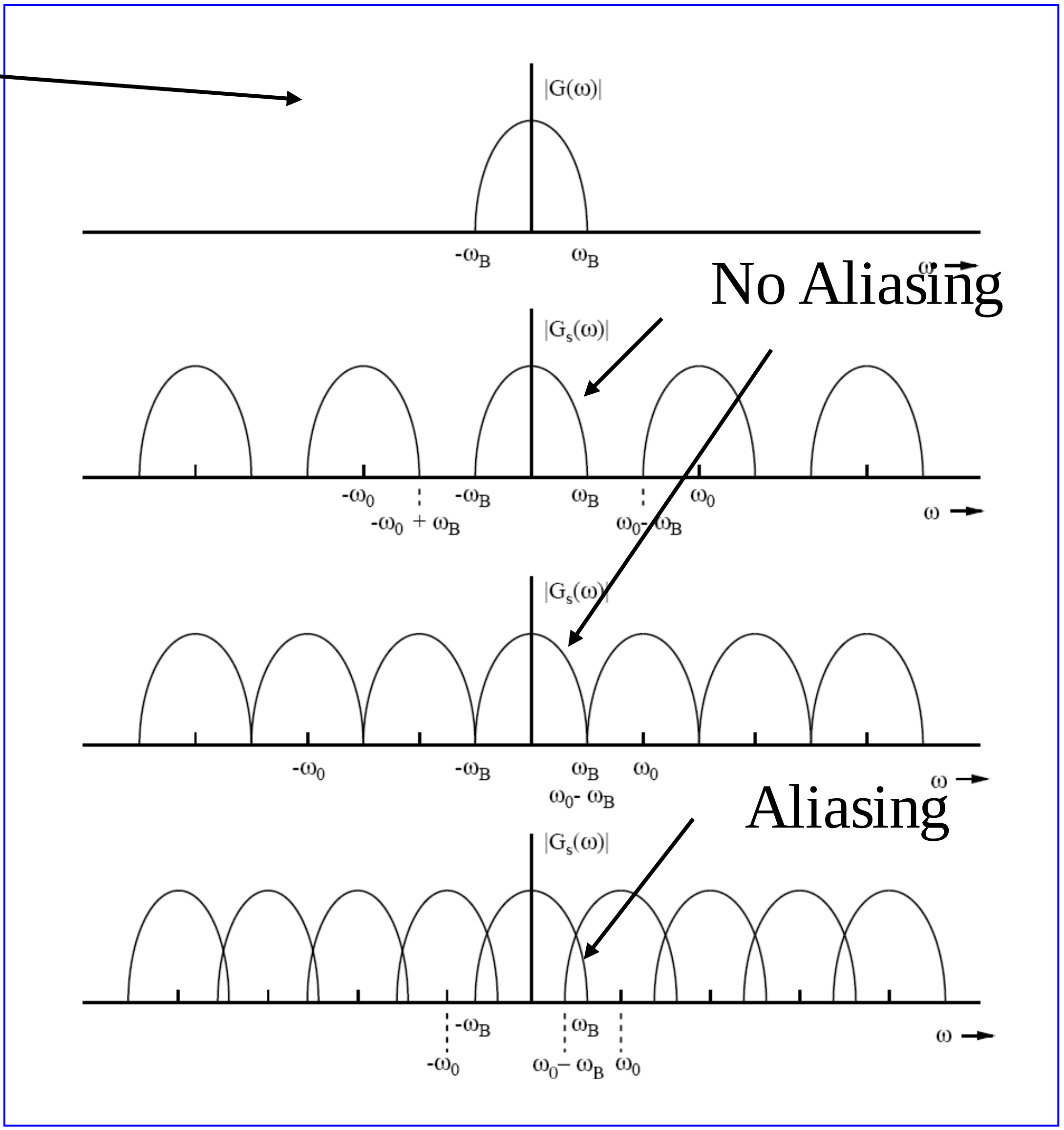
Sampled Signal Spectra:

Continuous signal g(t)



Sampled signal $g_s(t)$
(various values of ω_0)





Sketch of proof: [revision, see IB]

Define a mathematical representation of the sampled signal $g_s(t)$ using a train of δ-functions:

$$\begin{aligned} g_s(t) &= g(t) \sum_{n=-\infty}^{\infty} \delta(t - nT) \\ &= g(t)\delta_p(t) \end{aligned}$$

Fourier series representation of periodic function $\delta_p(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT)$:

$$\delta_p(t) = \frac{1}{T} \sum_{n=-\infty}^{\infty} \mathrm{e}^{jn\omega_0 t}$$

where $\omega_0 = 2\pi/T$.

Hence:

$$\begin{aligned} g_s(t) &= g(t) \frac{1}{T} \sum_{n=-\infty}^{\infty} \mathrm{e}^{jn\omega_0 t} \\ &= \frac{1}{T} \sum_{n=-\infty}^{\infty} g(t)\mathrm{e}^{jn\omega_0 t} \end{aligned}$$

Take Fourier Transform of $g_s(t)$:

$$G_s(\omega) = \frac{1}{T} \sum_{n=-\infty}^{\infty} G(\omega - n\omega_0)$$

Sampling Theorem: Summary

- Theorem shows us that we may represent a signal perfectly in the digital domain, provided the sampling rate is at least twice the maximum frequency component (`bandwidth') of the signal

- Denote the sampled values of a signal/function using the shorthand:

$$x_n = x(nT)$$

T is the sampling period
$f_0 = \frac{1}{T}\text{Hz}$ is the sampling frequency, or sampling 'rate'
$\omega_0 = 2\pi f_0 \text{rad/s}$ is the sampling frequency in radians

The DFT and the FFT

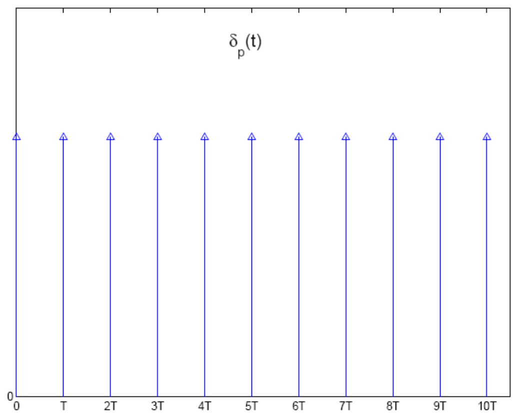
- The Discrete Fourier Transform is the standard way to transform a block of sampled data into the frequency domain (see IB)
- The Fast Fourier Transform (FFT) is a fast algorithm for implementation of the DFT
- The FFT revolutionised Digital Signal Processing. It is an elegant and highly effective algorithm that is still the building block used in many state-of-the-art algorithms in speech processing, communications, frequency estimation, …

The Discrete Time Fourier Transform (DTFT)

The DTFT is defined as the Fourier transform of the sampled signal. Define the sampled signal in the usual way:

$$g_s(t) = g(t) \underbrace{\sum_{p=-\infty}^{\infty} \delta(t - pT)}_{\delta_p(t)}$$

$\delta_p(t)$



Take Fourier transform directly

$$\begin{aligned} G_s(\omega) &= \int_{-\infty}^{\infty} g_s(t)\, e^{-j\omega t}\, dt \\ &= \int_{-\infty}^{\infty} g(t) \sum_{p=-\infty}^{\infty} \delta(t - pT)\, e^{-j\omega t}\, dt \\ &= \sum_{p=-\infty}^{\infty} g(pT) e^{-j\omega pT} = \sum_{p=-\infty}^{\infty} g_p e^{-j\omega pT} \end{aligned}$$

using the 'sifting' property of the δ-function to reach the last line.

Note that this expression, known as the DTFT, is a periodic function of frequency, usually written as:

$$\boxed{G(e^{j\omega T}) = \sum_{p=-\infty}^{\infty} g_p \, e^{-j\omega p T}} \quad (*)$$

The signal sample values may be expressed in terms of the DTFT by noting that the equation above has the form of a Fourier series *(as a function of ω)* and hence the sampled signal can be obtained directly as:

$$\boxed{g_n = \frac{1}{2\pi} \int_0^{2\pi} G(e^{j\omega T}) \, e^{+jn\omega T} \, d\omega T}$$

[You can show this for yourself by first noting that (*) is a complex Fourier series with coefficients g_p]

The Discrete Fourier Transform (DFT)

The DFT is the fundamental building block of many modern signal processing systems. We will later derive a fast algorithm for DFT computation, known as the *Fast Fourier Transform* (FFT). The FFT is possibly the most widely used digital signal processing algorithm ever invented.

The Discrete Time Fourier Transform (DTFT):

$$G(e^{j\omega T}) = \sum_{p=-\infty}^{\infty} g_p \, e^{-j\omega pT}$$

expresses the spectrum of a sampled signal in terms of the signal samples but is not computable on a digital computer for two reasons:

1. The frequency variable ω is continuous.
2. The summation involves an infinite number of samples

These problems can be overcome by:

1. evaluating the DTFT at a finite collection of discrete frequencies
2. Performing the summation over a finite number of data points.

Step 1. has no undesirable consequences since we can always include any frequency of interest in the collection .

Step 2. does have consequences, since signals are generally not of finite duration. These consequences can be rigorously analysed by windowing analysis.

The discrete set of frequencies chosen is arbitrary. However, since the DTFT is *periodic* we generally choose a uniformly spaced grid of N frequencies covering the range $\omega T = 0 \rightarrow 2\pi$. If the summation is then truncated to just N data points, we get the DFT:

$$G_p = G(e^{j\frac{2\pi}{N}p}) = \sum_{n=0}^{N-1} g_n \, e^{-j\frac{2\pi}{N}np} \tag{1}$$

The inverse DFT can be used to obtain the sampled signal values from the DFT:

Multiply each side of equation (1) by $e^{j\frac{2\pi}{N}pq}$ and sum over $p = 0$ to $N - 1$:

$$\sum_{p=0}^{N-1} G_p \, e^{j\frac{2\pi}{N}pq} = \sum_{p=0}^{N-1}\sum_{n=0}^{N-1} g_n \, e^{-j\frac{2\pi}{N}np} \, e^{j\frac{2\pi}{N}pq}$$

$$= \sum_{n=0}^{N-1} g_n \sum_{p=0}^{N-1} e^{j\frac{2\pi}{N}(q-n)p}$$

Note the orthogonality property of the complex exponentials:

$$\sum_{p=0}^{N-1} e^{j\,\frac{2\pi}{N}\,(q-n)p} = \begin{cases} N & n = q \\ 0 & n \neq q \end{cases}$$

Hence:

$$g_q = \frac{1}{N} \sum_{p=0}^{N-1} G_p \, e^{j\,\frac{2\pi}{N}\,pq}$$

The above equations for G_p and g_n are the Discrete Fourier Transform pair, summarised as:

$$G_p = \sum_{n=0}^{N-1} g_n \, e^{-j\,\frac{2\pi}{N}\,np}$$

$$g_n = \frac{1}{N} \sum_{p=0}^{N-1} G_p \, e^{j\,\frac{2\pi}{N}\,pn}$$

Properties of the DFT

- G_p is periodic, i.e.

$$G_{p+N} = G_p, \quad \text{for each } p$$

- g_n is periodic, i.e.

$$g_{n+N} = g_n, \quad \text{for each } n$$

  [This may seem strange since we defined the DFT by truncating the signal. However, the inverse DFT formula implies that the data are periodic if we calculate values of x_{p+N} where $p + N$ lies outside the usual range $\{0, 1, ..., N-1\}$].

- For *real* data g_n we have conjugate symmetry, i.e.

$$G_p = G^*_{-p} = G^*_{N-p}$$

[You should check that you can show these results from first principles]

Rewriting in terms of a sequence of sampled values x_n and transformed values X_p, the Discrete Fourier Transform (DFT) of a sequence of data $\{x_n\}$ is given by:

$$X_p = \sum_{n=0}^{N-1} x_n \, e^{-j\frac{2\pi}{N}np}, \quad p \in \{0, 1, ..., N-1\} \tag{1}$$

$$x_n = \frac{1}{N} \sum_{p=0}^{N-1} X_p e^{j\frac{2\pi}{N}np}, \quad n \in \{0, 1, 2, ..., N-1\} \tag{2}$$

Can think of this as a vector operation:

- Take a vector of samples as input:

$$\mathbf{x} = [x_0,\ x_1,\ ...,x_{N-1}]^T$$

- Get a vector of frequency values as output:

$$\mathbf{X} = [X_0,\ X_1,\ ...,X_{N-1}]^T$$

Can write this as:

$$\mathbf{X} = \mathbf{M}\mathbf{x}$$

where $\mathbf{M}$ is the appropriate (NxN) matrix

The Fast Fourier Transform (FFT)

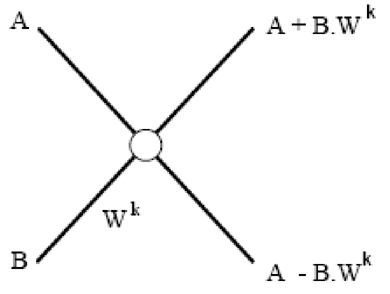
- The discovery of the FFT was 'first' announced by Cooley and Tukey in 1965 - their paper is the most cited mathematical paper ever written
- They were actually over 150 years late - the principle of the FFT was later discovered in an obscure section of Gauss' (as in Gaussian) own notebooks in 1806
- Computation of X_p for $p = 0, 1, \ldots, N-1$ requires on the order of N^2 complex multiplications and additions [assuming that the complex exponentials have been pre-computed and stored].
- The Fast Fourier Transform reduces the required number of arithmetic operations to the order of $\frac{N}{2}\log_2(N)$ when N is a power of 2.
- This speed-up is dramatic for large N - for $N = 1024$ the speed-up is a factor of 205, for $N = 8192$ the speed-up is 1260, ...
- There are many different types of FFT algorithm, and many different derivations possible, including some which operate for N not a power of 2.
- Here we consider the most basic 'radix-2' algorithm which requires N to be a power of 2.

Derivation

- The FFT derivation relies on redundancy in the calculation of the basic DFT
- A recursive algorithm is derived that repeatedly rearranges the problem into two simpler problems of half the size
- Hence the basic algorithm operates on signals of length a power of 2, i.e.

$$N = 2^M \qquad \text{(for some integer M)}$$

- At the bottom of the tree, we have the classic FFT `butterfly' structure (details later):



First, take the basic DFT equation:

$$X_p = \sum_{n=0}^{N-1} x_n \, e^{-j\frac{2\pi}{N}np}$$

Now, split the summation into two parts: one for even n and one for odd n:

$$\begin{aligned}
X_p &= \sum_{n=0}^{\frac{N}{2}-1} x_{2n} \, e^{-j\frac{2\pi}{N}(2n)p} + \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} e^{-j\frac{2\pi}{N}(2n+1)p} \quad (*) \\
&= \sum_{n=0}^{\frac{N}{2}-1} x_{2n} e^{-j\frac{2\pi}{(N/2)}np} + e^{-j\frac{2\pi}{N}p} \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} e^{-j\frac{2\pi}{(N/2)}np} \\
&= A_p + W^p B_p
\end{aligned}$$

where

$$\begin{aligned}
A_p &= \sum_{n=0}^{\frac{N}{2}-1} x_{2n} e^{-j\frac{2\pi}{(N/2)}np} \\
B_p &= \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} e^{-j\frac{2\pi}{(N/2)}np} \\
W &= e^{-j\frac{2\pi}{N}}
\end{aligned}$$

- Notice now that A_p and B_p are themselves DFTs each of length $N/2$:
  - A_p is the DFT of a sequence $\{x_{2n}\} = \{x_0,\, x_2,\, ...x_{N-4},\, x_{N-2}\}$
  - B_p is the DFT of a sequence $\{x_{2n+1}\} = \{x_1,\, x_3,\, ...x_{N-3},\, x_{N-1}\}$

- We know, however that the DFT is periodic in the frequency domain (in this case with period N/2). This leads to further simplifications, as follows.

To see how this simplifies, look at the original DFT in (*) above, but evaluated at frequencies $p + N/2$:

$$X_{p+N/2} = \sum_{n=0}^{\frac{N}{2}-1} x_{2n}\, e^{-j\frac{2\pi}{(N/2)}n(p+N/2)} + e^{-j\frac{2\pi}{N}(p+N/2)} \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1}\, e^{-j\frac{2\pi}{(N/2)}n(p+\frac{N}{2})}$$

Now, simplify terms as follows:

$$e^{-j\frac{2\pi}{(N/2)}n(p+N/2)} = e^{-j\frac{2\pi}{(N/2)}np}, \quad e^{-j\frac{2\pi}{N}(p+N/2)} = -e^{-j\frac{2\pi}{N}p}$$

Hence,

$$X_{p+N/2} = \sum_{n=0}^{\frac{N}{2}-1} x_{2n}\, e^{-j\frac{2\pi}{(N/2)}np} - e^{-j\frac{2\pi}{N}p} \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1}\, e^{-j\frac{2\pi}{(N/2)}np}$$

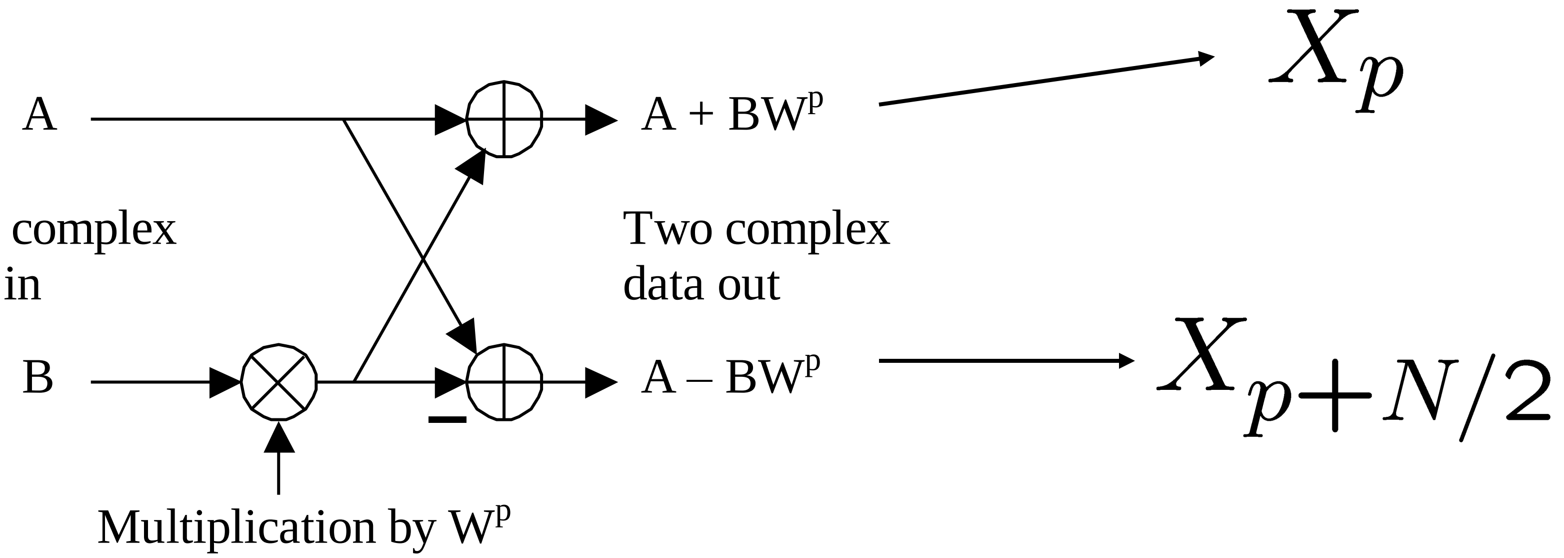
$$= A_p - W^p B_p$$

with A_p W^p and B_p defined as before

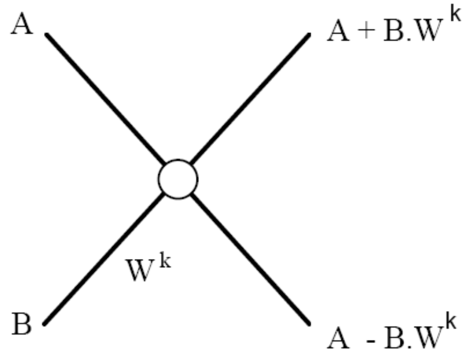
Now, compare the equations for $X_{p+N/2}$ with that for X_p:

$$X_p = A_p + W^p B_p, \quad X_{p+N/2} = A_p - W^p B_p$$

This defines the FFT butterfly structure:



Or, in more compact form:



('Butterfly')

Computational load:

Look at the two required terms (W^p assumed precomputed and stored):

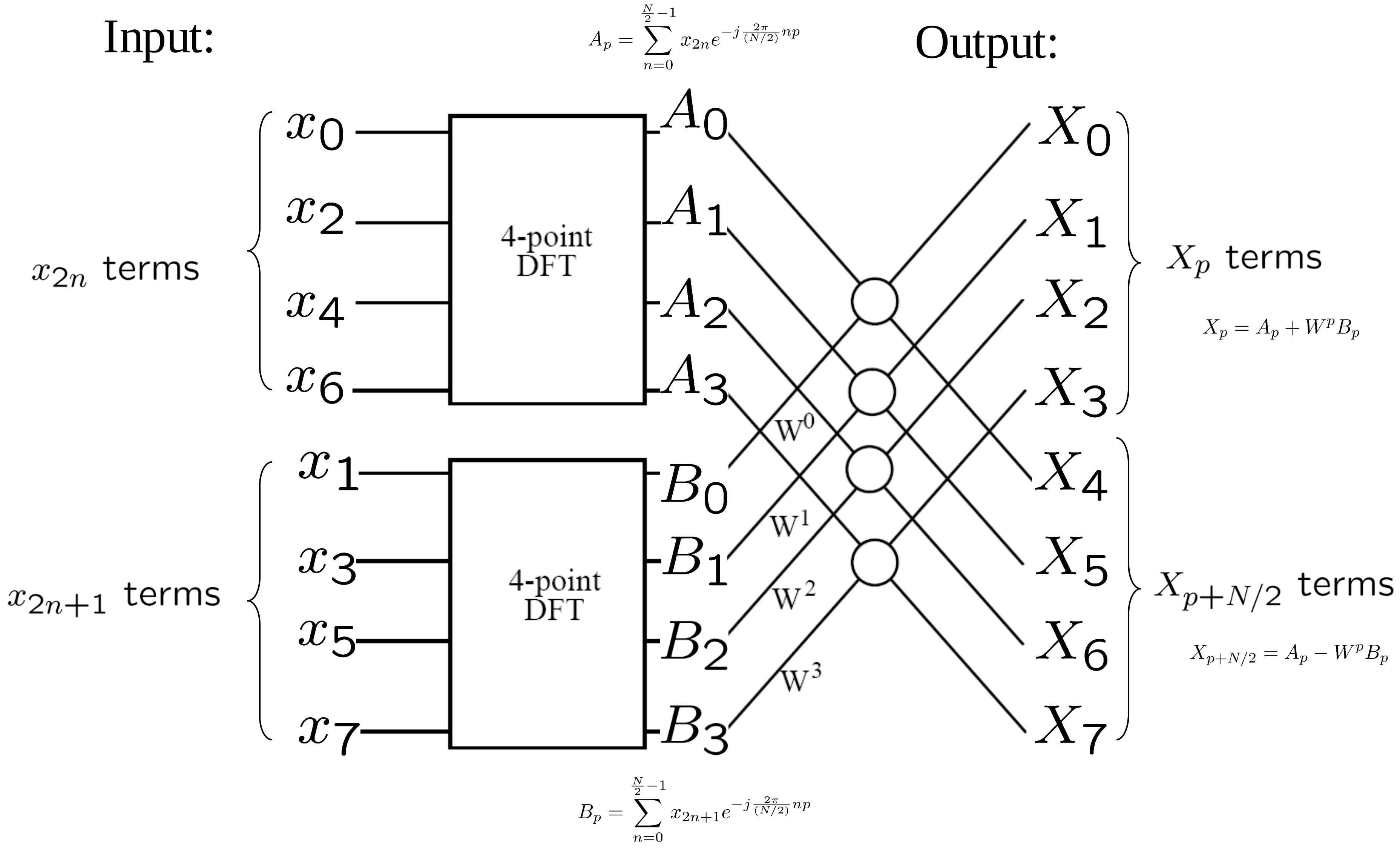
$$A_p = \sum_{n=0}^{\frac{N}{2}-1} x_{2n} e^{-j\frac{2\pi}{(N/2)}np}, \quad B_p = \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} e^{-j\frac{2\pi}{(N/2)}np},$$

- The terms A_p and B_p need only be computed for $p = 0, 1, ..., N/2 - 1$, since $X_{p+N/2}$ has been expressed in terms of A_p and B_p - hence we have uncovered redundancy in the DFT computation.

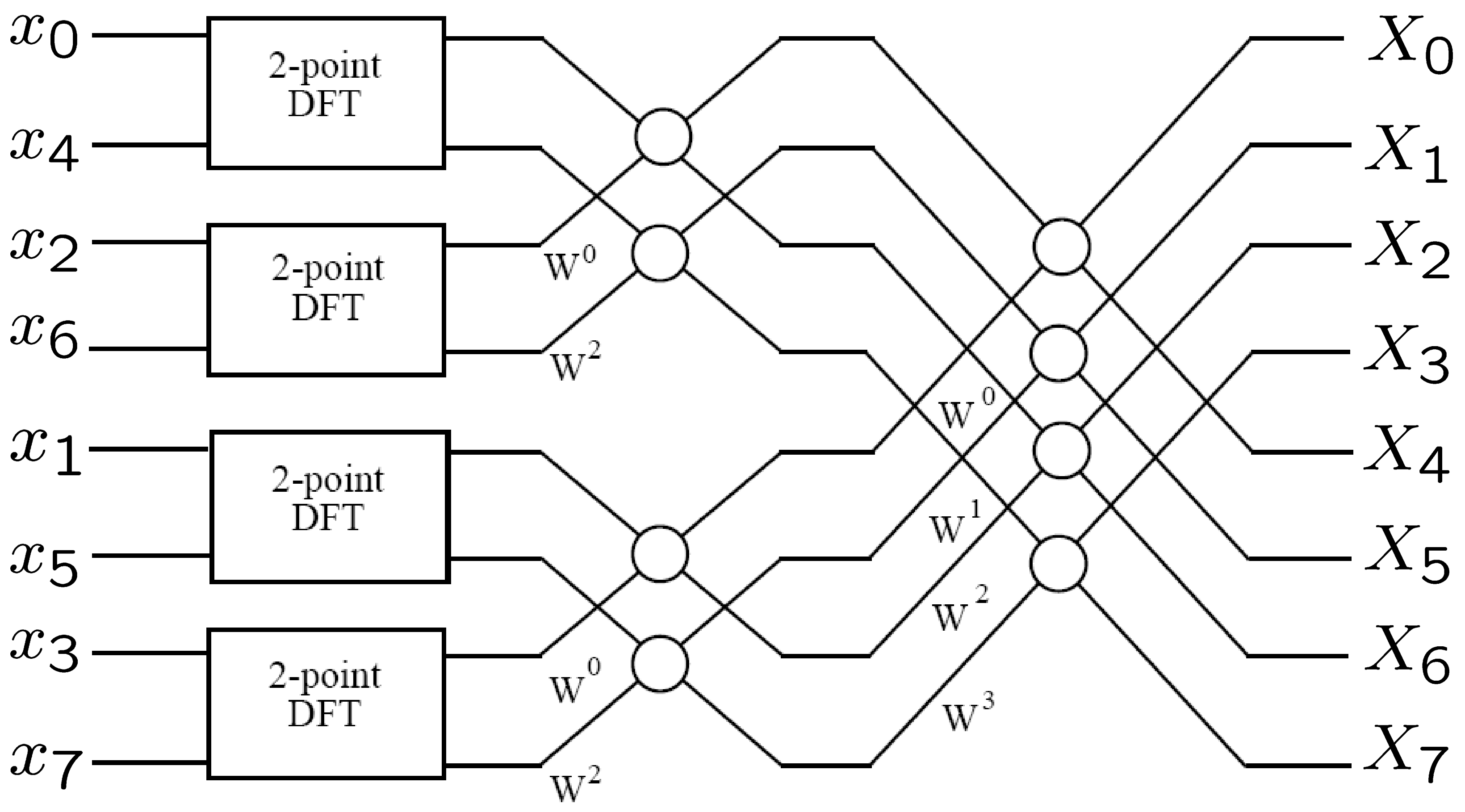
- Thus calculate the A_p and B_p for $p = 0, 1, ..., N/2 - 1$ and use them for calculation of both X_p and $X_{p+N/2}$

- The number of complex multiplies and additions is:
  - A_p requires $N/2$ complex multiplies and additions; so does B_p. The total for all $p = 0, 1, ..., N/2 - 1$ is then $2(N/2)^2$ multiplies and additions for the calculation of all the A_p and B_p terms.
  - $N/2$ multiplies for the calculation of $W^p B_p$ for all $p = 0, 1, 2, ..., N/2 - 1$
  - $N = N/2 + N/2$ additions for calculation of $A_p + W^p B_p$ and $A_p - W^p B_p$
- Thus total number of complex multiplies and additions is approximately $N^2/2$ for large N
- The computation is approximately halved compared to the direct DFT evaluation

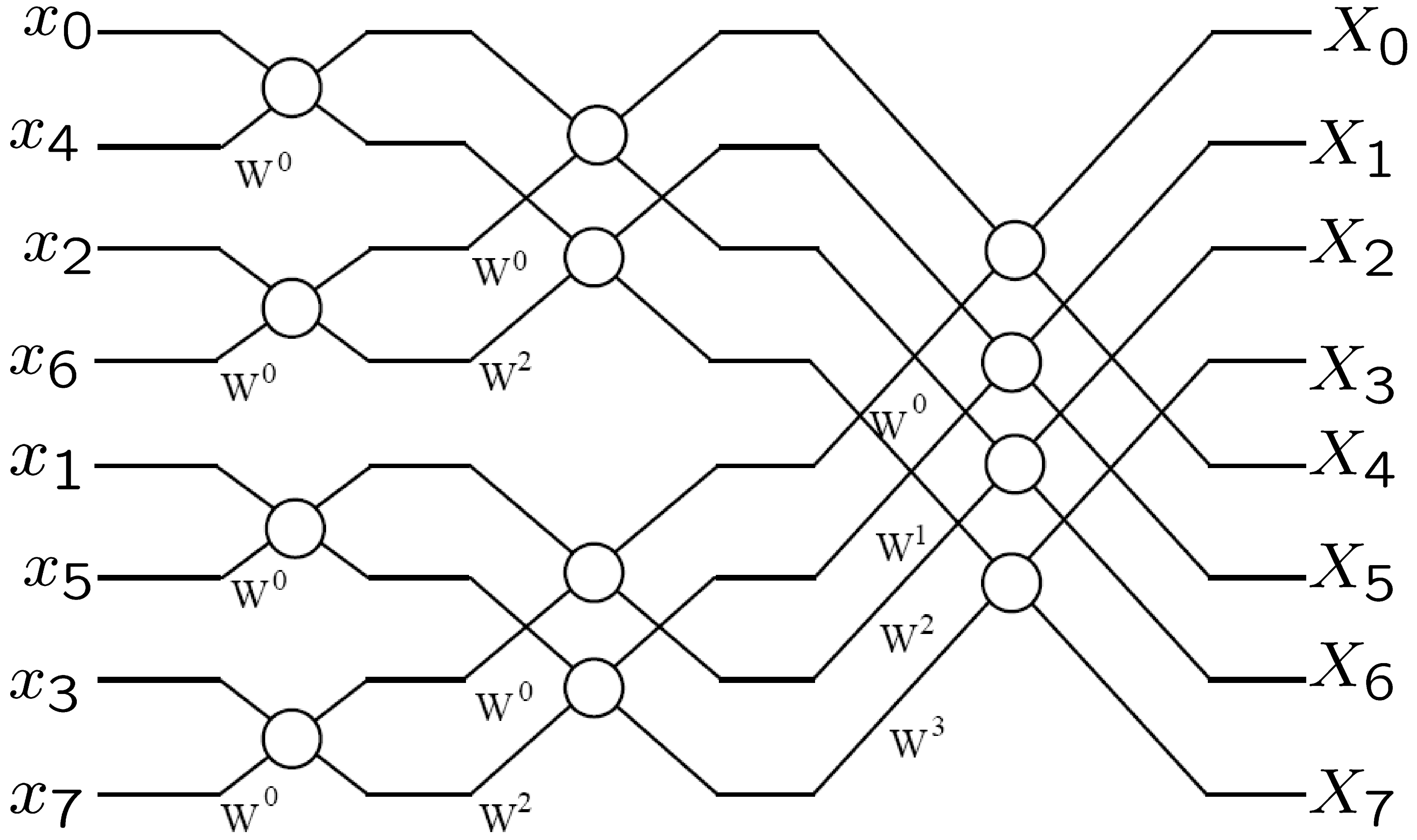
A flow diagram for a N=8 DFT is shown below:



Assuming that $(\frac{N}{2})$ is even, the same process can be carried out on each of the $(\frac{N}{2})$ point DFTs to further reduce the computation. The flow diagram for incorporating this extra stage of decomposition into the computation of the $N = 8$ point DFT is shown below.



It can be seen that if $N = 2^M$ then the process can be repeated M times to reduce the computation to that of evaluating N single point DFTs. Thus the flow chart for computing the $N = 8$ point DFT is as shown below.



Examination of the final chart shows that it is necessary to shuffle the order of the input data. This data shuffle is usually termed *bit-reversal* for reasons that are clear if the indices of the shuffled data are written in binary.

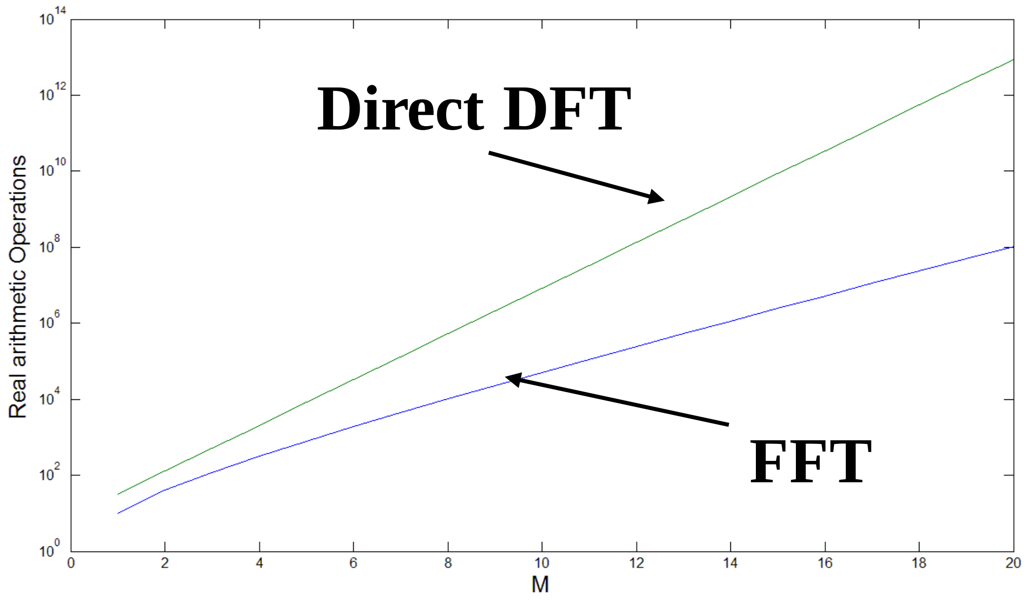
Binary	Bit Reverse	Decimal
000	000	0
001	100	4
010	010	2
011	110	6
100	001	1
101	101	5
110	011	3
111	111	7

Computational Load of full FFT algorithm:

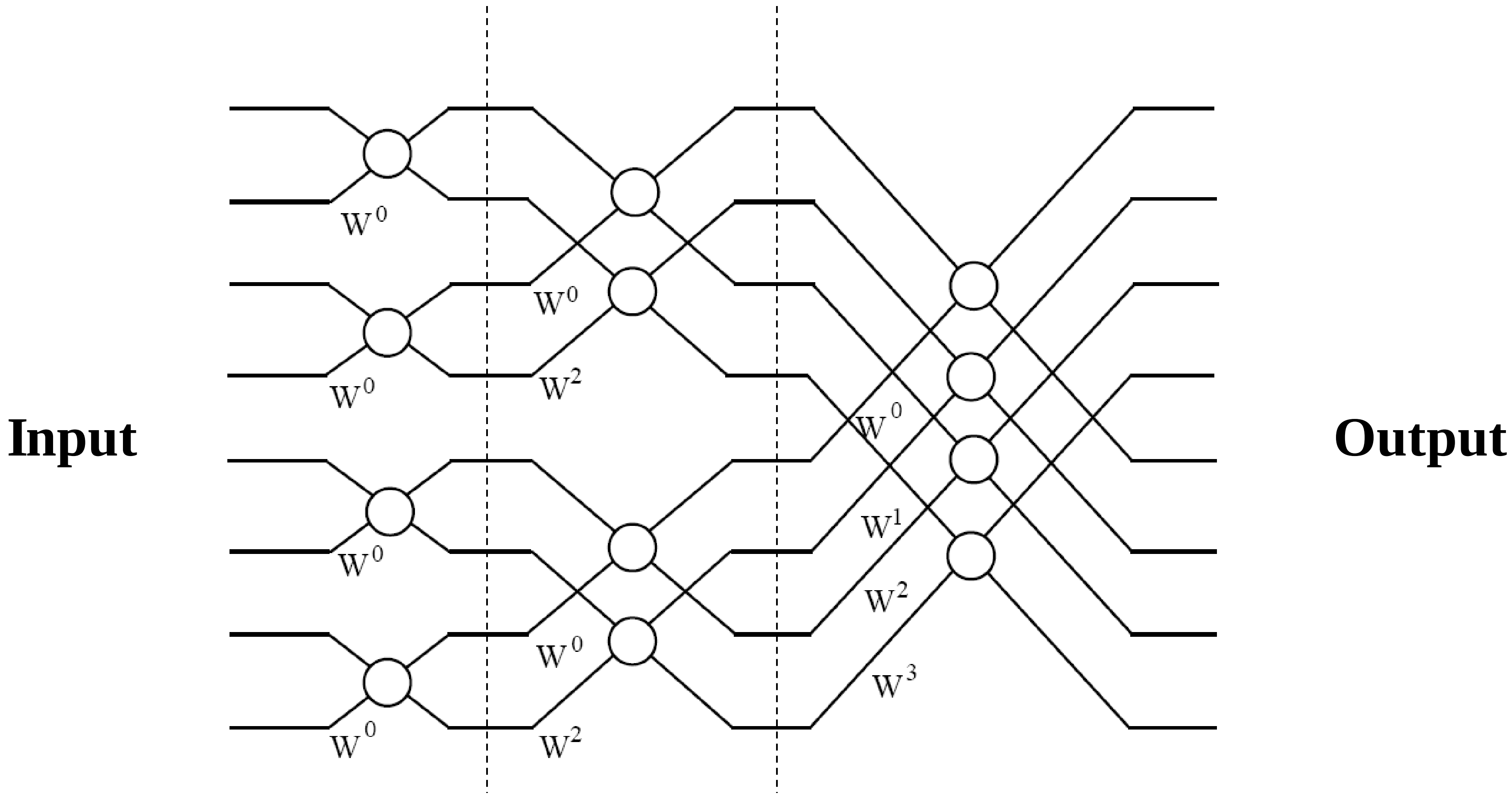
The type of FFT we have considered, where $N = 2^M$, is called a radix-2 FFT. It has $M = \log_2 N$ stages, each using $N / 2$ butterflies

Since a complex multiplication requires 4 real multiplications and 2 real additions, and a complex addition/subtraction requires 2 real additions, a butterfly requires 10 real operations. Hence the radix-2 N-point FFT requires $10(N / 2)\log_2 N$ real operations compared to about $8N^2$ real operations for the DFT.

This is a huge speed-up in typical applications, where N is 128 – 4096:



The FFT algorithm has a further significant advantage over direct evaluation of the DFT expression in that computation can be performed *in-place*. This is best illustrated in the final flow chart where it can be seen that after two data values have been processed by the *butterfly* structure, those data are not required again in the computation and they may be replaced, in the computer or in the chip memory, with the values at the output of the *butterfly* structure.



The Inverse FFT (IFFT)

Apart from the scale factor 1 / N, the Inverse DFT has the same form as the DFT, except that the conjugate W^* replaces W. Hence the computation algorithm is the same, with a final scaling by $1 / N$.

Other types of FFT

There are many FFT variants. The form of FFT we have described is called “decimation in time”; there is a form called “decimation in frequency” (but it has no advantages).

The "radix 2" FFT must have length N a power of 2. Slightly more efficient is the "radix 4" FFT, in which 2-input 2-output butterflies are replaced by 4-input 4-output units. The transform length must then be a power of 4 (more restrictive).

A completely different type of algorithm, the Winograd Fourier Transform Algorithm (WFTA), can be used for FFT lengths equal to the product of a number of mutually prime factors (e.g. 9*7*5 = 315 or 5*16 = 80). The WFTA uses fewer multipliers, but more adders, than a similar-length FFT.

Efficient algorithms exist for FFTing **real (not complex) data at about 60% the effort of the same-sized complex-data FFT.**

The Discrete Cosine and Sine Transforms (**DCT and DST) are similar real-signal algorithms used in image coding.**

Applications of the FFT

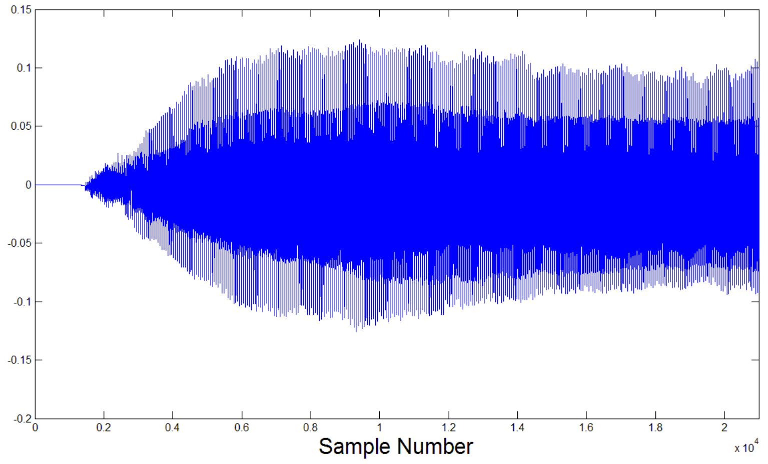
There FFT is surely the most widely used signal processing algorithm of all

It is the basic building block for a large percentage of algorithms in current usage

Specific examples include:

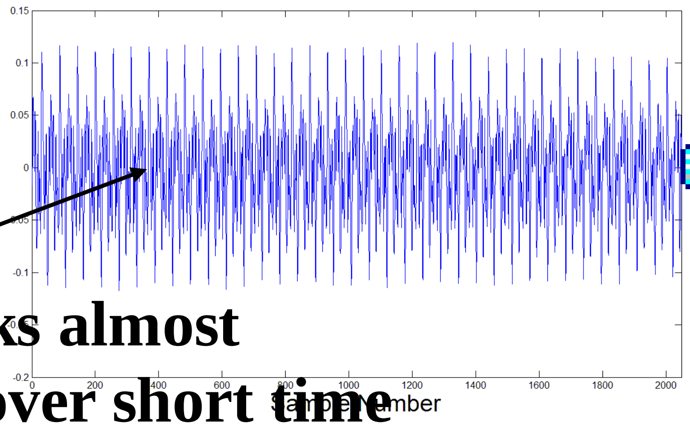
• Spectrum analysis – used for analysing and detecting signals
• Coding – audio and speech signals are often coded in the frequency domain using FFT variants (MP3, …)
• Another recent application is in a modulation scheme called OFDM, which is used for digital TV broadcasting (DVB) and digital radio (audio) broadcasting (DAB).
• Background noise reduction for mobile telephony, speech and audio signals is often implemented in the frequency domain using FFTs
….

Case Study: Spectral analysis of a Musical Signal



Sample rate is 10.025 kHz (T=1/10,025 s)

Extract a short segment:



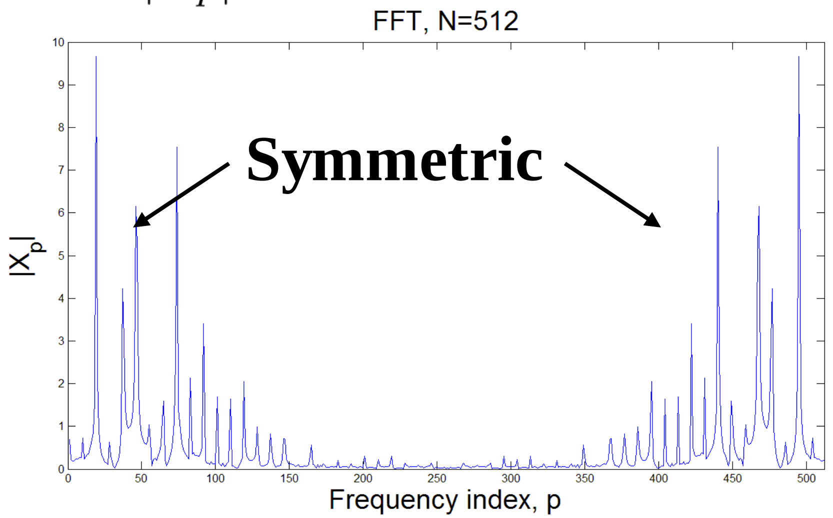
Load this into Matlab as a vector x

Take an FFT, N=512:

X=fft(x(1:512));

Note: looks almost Periodic over short time interval

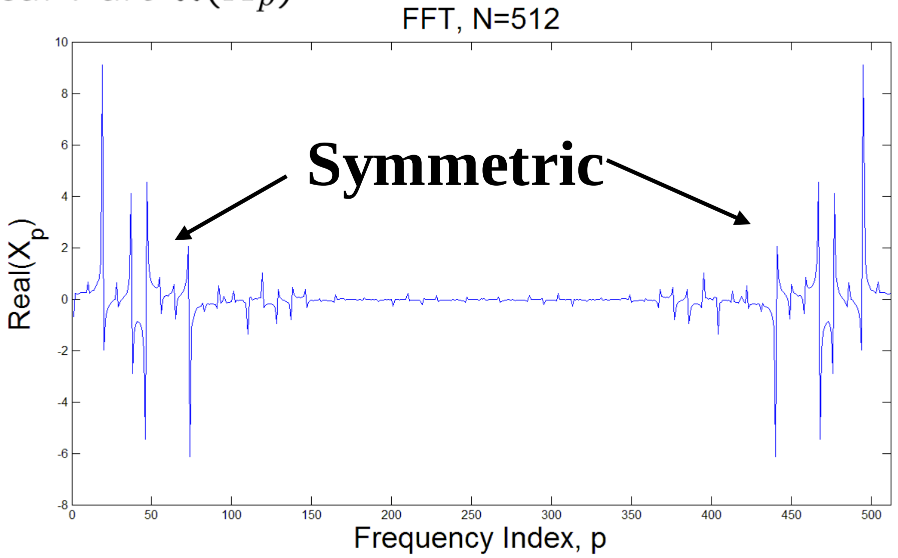
Modulus $|X_p|$



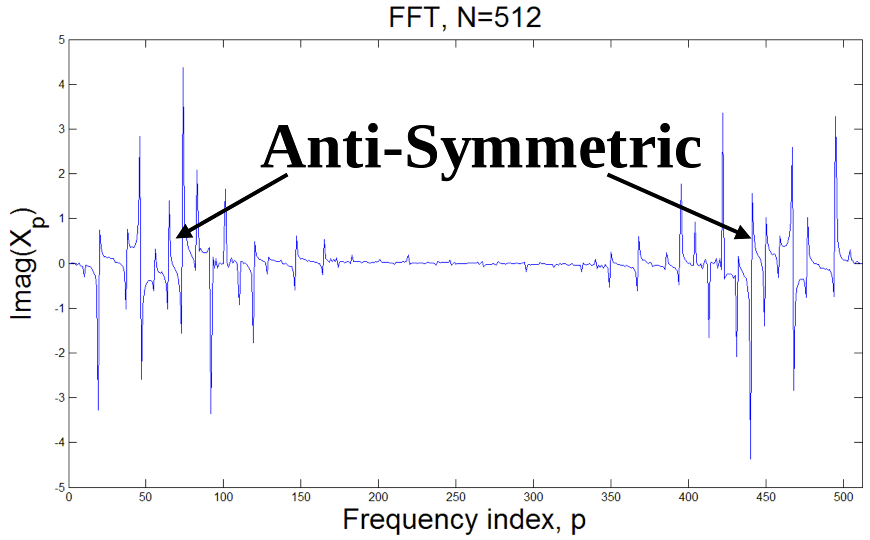
Note Conjugate symmetry as data are real:

$$X_p = X^*_{N-p}$$

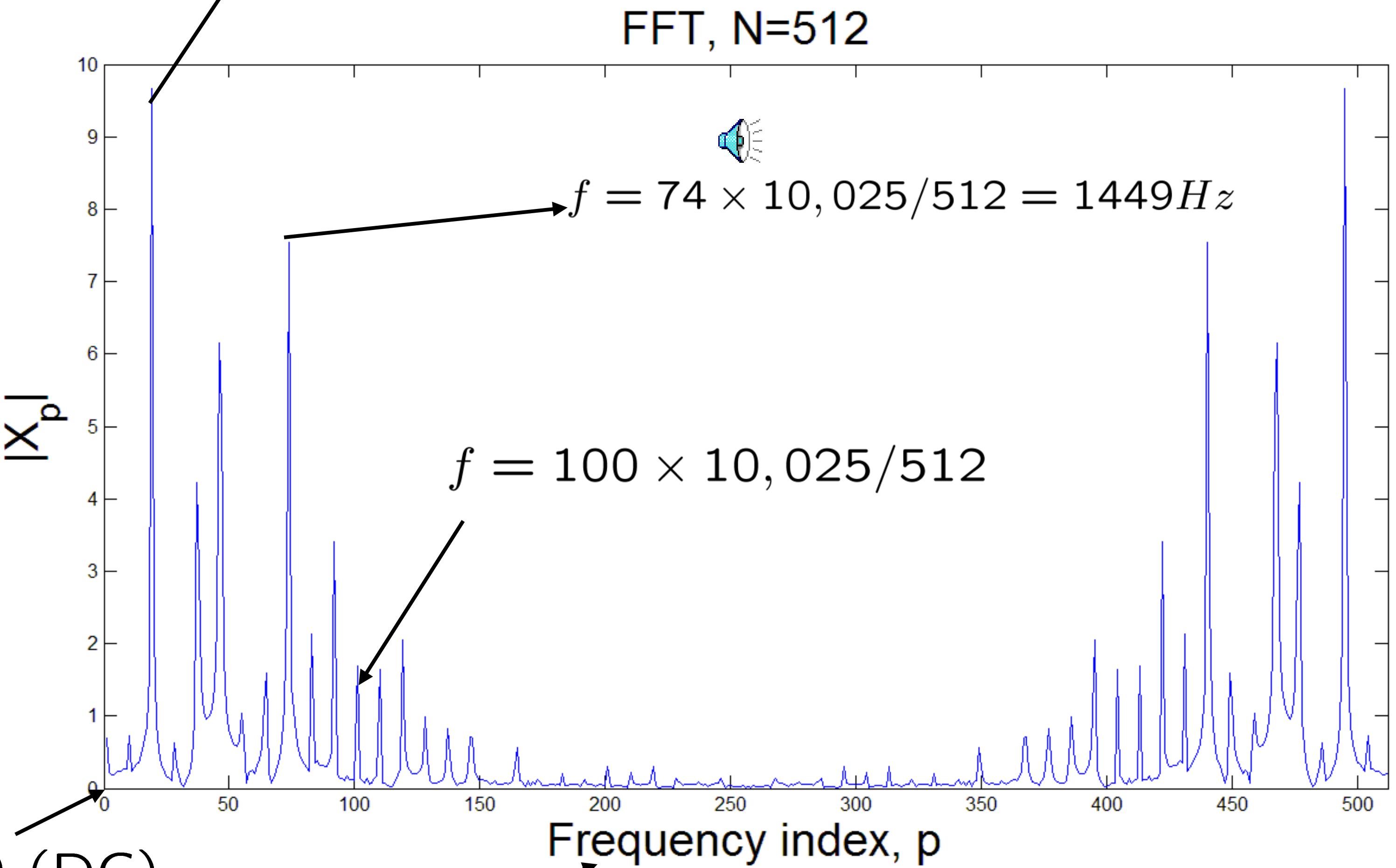
Real Part $\Re(X_p)$



Imaginary Part $\Im(X_p)$



$f = 19 \times 10,025/512 = 372Hz$



$f = 74 \times 10,025/512 = 1449Hz$

$f = 100 \times 10,025/512$

$f = 0$ (DC)

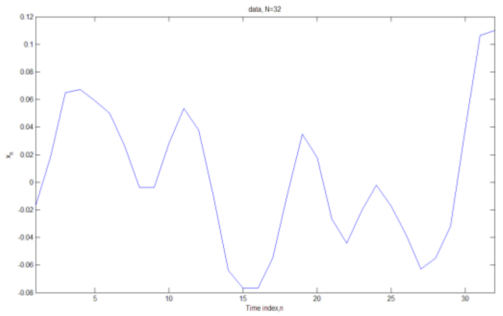
Frequency index corresponds to true frequency

$f = \frac{p f_0}{N} = \frac{p}{NT}$

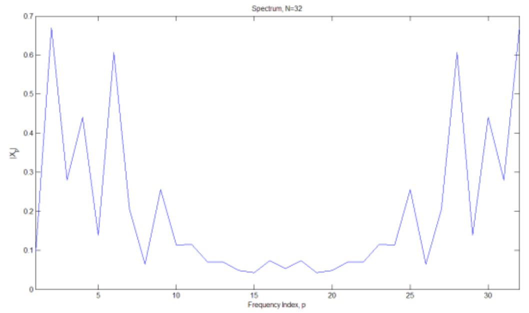
3F3 – Digital Signal Processing

The Effect of data length, N

N=32

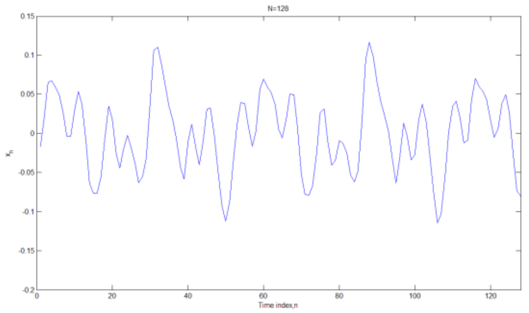


FFT

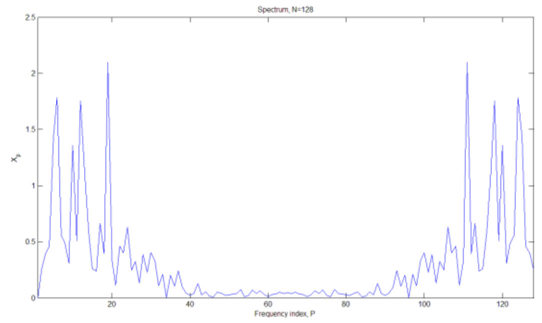


Low resolution

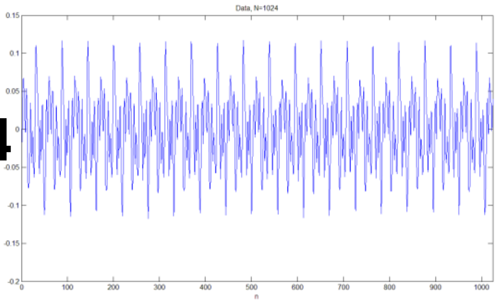
N=128



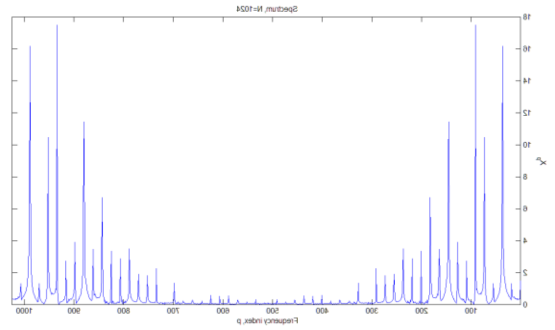
FFT



N=1024



FFT



High resolution

The DFT approximation to the DTFT

DTFT at frequency $\omega = \dfrac{2\pi p}{NT}$:

$$X(e^{j\frac{2\pi}{N}p}) = \sum_{n=-\infty}^{\infty} x_n \, e^{-j\frac{2\pi}{N}np}$$

DFT:

$$X_p = \sum_{n=0}^{N-1} x_n \, e^{-j\frac{2\pi}{N}np}$$

- Ideally the DFT should be a `good' approximation to the DTFT
- Intuitively the approximation gets better as the number of data points N increases
- This is illustrated in the previous slide – resolution gets better as N increases (more, narrower, peaks in spectrum).
- How to evaluate this analytically?
  - View the truncation in the summation as a multiplication by a rectangle window function
  - Then, in frequency domain, multiplication becomes *convolution*

Analysis:

Consider DTFT:

$$X(e^{j\omega T}) = \sum_{n=-\infty}^{\infty} x_n \, e^{-jn\omega T}$$

Truncate the summation, as for the DFT:

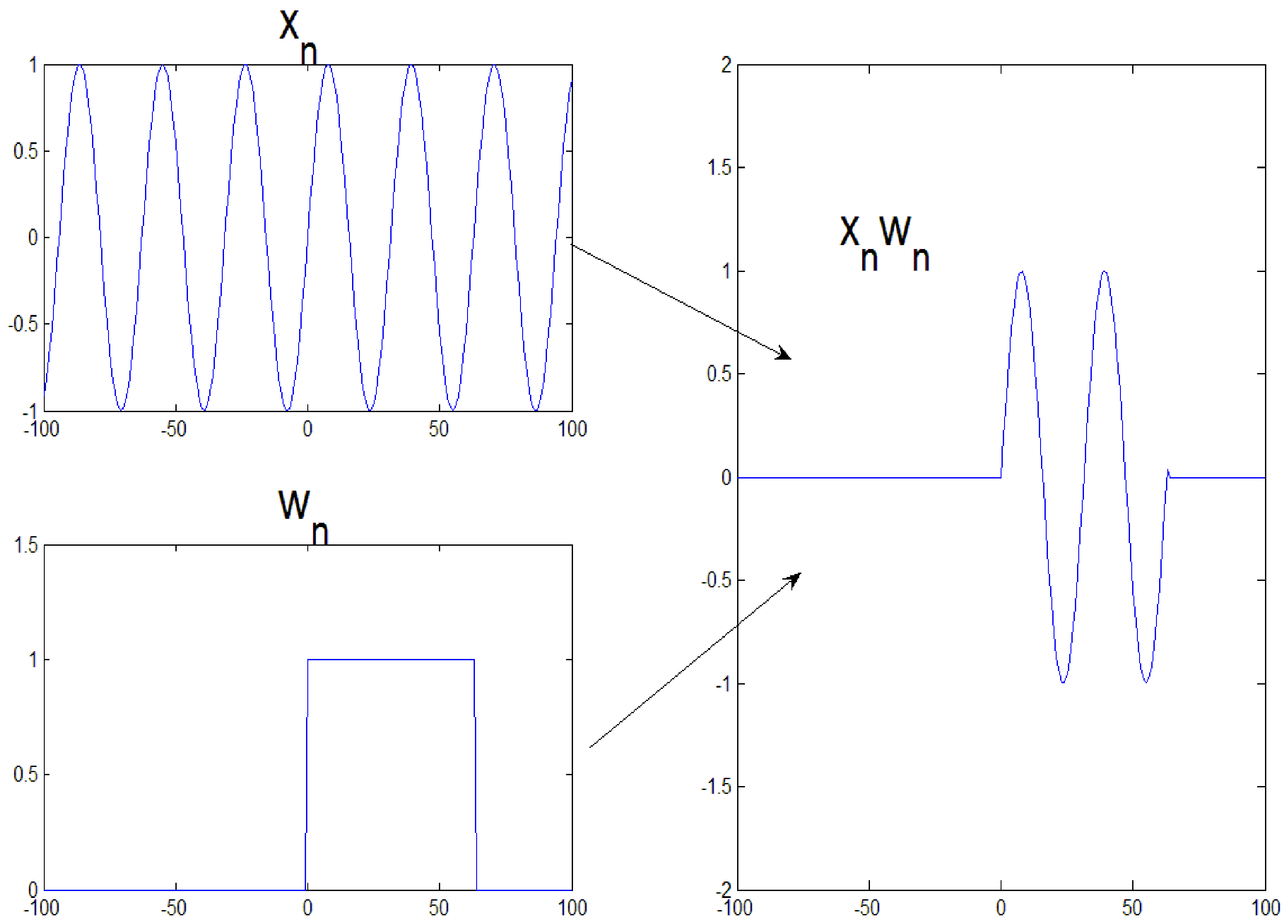
$$X_w(e^{j\omega T}) = \sum_{n=0}^{N-1} x_n \, e^{-jn\omega T}$$

Now, note that this is equivalent to an infinite summation, but with x_n pre-multiplied by a rectangle window function:

$$X_w(e^{j\omega T}) = \sum_{n=-\infty}^{\infty} w_n x_n \, e^{-jn\omega T}$$

with

$$w_n = \begin{cases} 1, & n = 0,\ 1,\ 2,\ ...,\ N-1 \\ 0, & \text{otherwise} \end{cases}$$


X_n
W_n
$X_n W_n$
1
0.5
0
-0.5
-1
1.5
2
-1.5
-2
-100
-50
0
50
100

DTFT of w_n is $W(e^{j\theta})$

Now, take the DTFT of the windowed signal $x_w = w_n x_n$ directly:

Inverse DTFT of $W(e^{j\theta})$

$$\begin{aligned}
X_w(e^{j\omega T}) &= \sum_{n=-\infty}^{\infty} \{x_n\, w_n\}\, e^{-jn\omega T} \\
&= \sum_{n=-\infty}^{\infty} x_n \left\{ \frac{1}{2\pi} \int_0^{2\pi} W(e^{j\theta}) e^{jn\theta} d\theta \right\} e^{-jn\omega T} \\
&= \frac{1}{2\pi} \int_0^{2\pi} W(e^{j\theta}) \sum_{n=-\infty}^{\infty} x_n\, e^{-jn(\omega T-\theta)}\, d\theta \\
X_w(e^{j\omega T}) &= \frac{1}{2\pi} \int_0^{2\pi} W(e^{j\theta})\, X(e^{j(\omega T-\theta)})\, d\theta
\end{aligned}$$

We see that the spectrum of the windowed signal is the convolution of the infinite duration signal spectrum and the window spectrum.

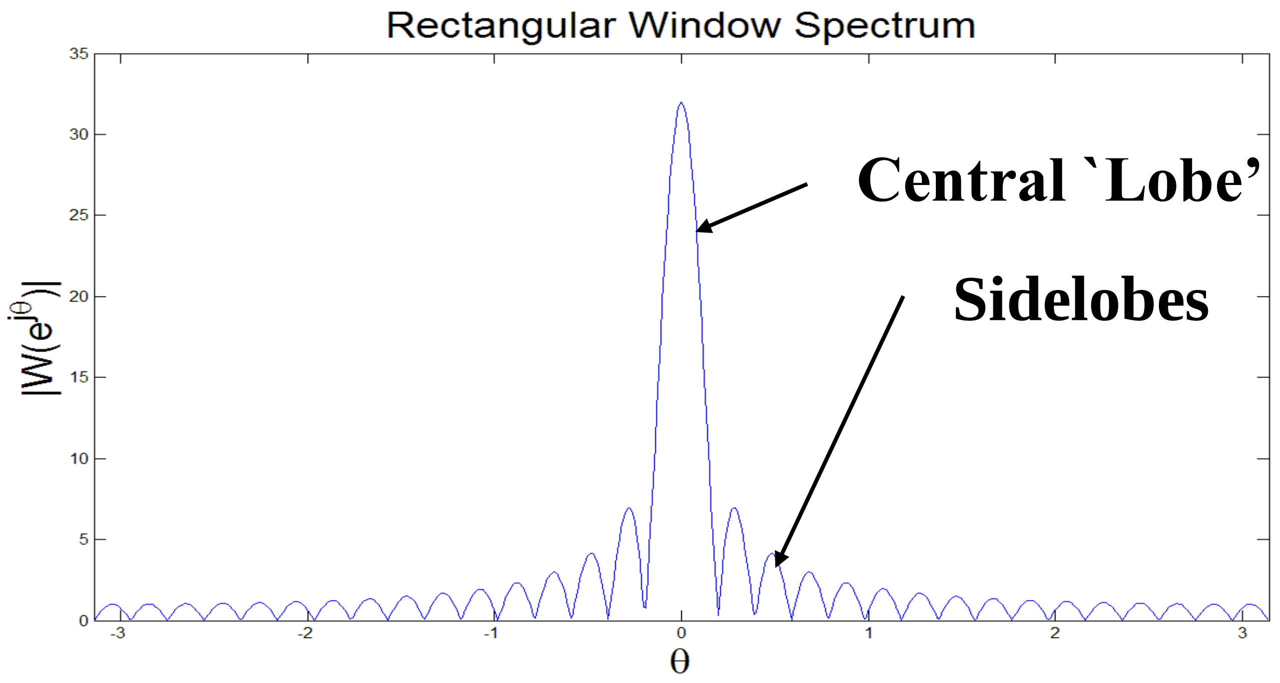
What is the DTFT of the window w_n?

[Subst. $\theta = \omega T$ - makes no difference to form of results]:

$$\begin{aligned} W(e^{j\theta}) &= \sum_{n=0}^{N-1} 1.e^{-jn\theta} \\ &= e^{-j(N-1)\theta/2} \frac{\sin(N\theta/2)}{\sin(\theta/2)} \end{aligned}$$

[Check you can get this result yourself as an extra examples question]

N=32

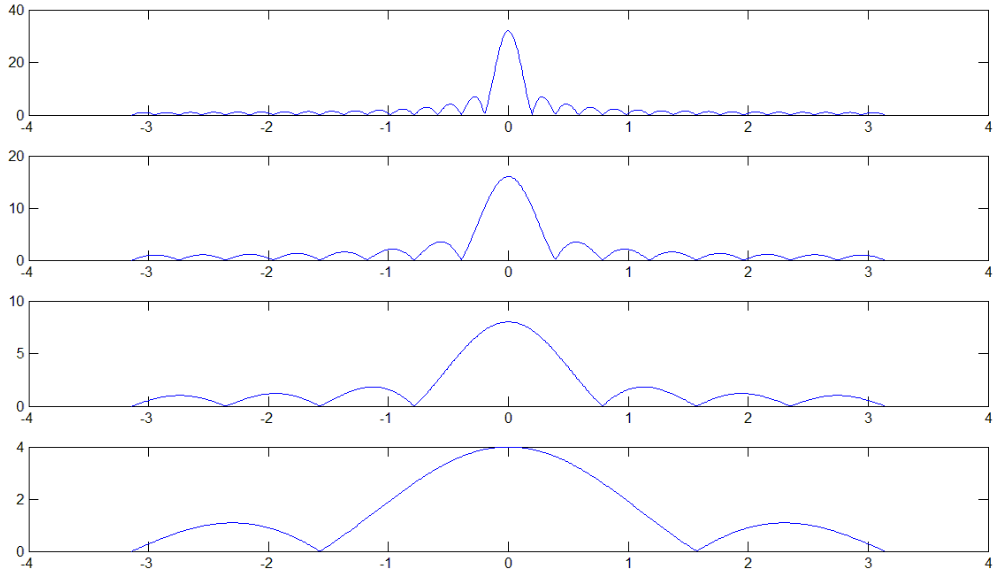


N=32

N=16

N=8

N=4



Lobe width inversely proportional to N

Now, imagine what happens when the sum of two frequency components is DFT-ed:

$$x_n = \exp(j\omega_1 nT) + \exp(j\omega_2 nT)$$

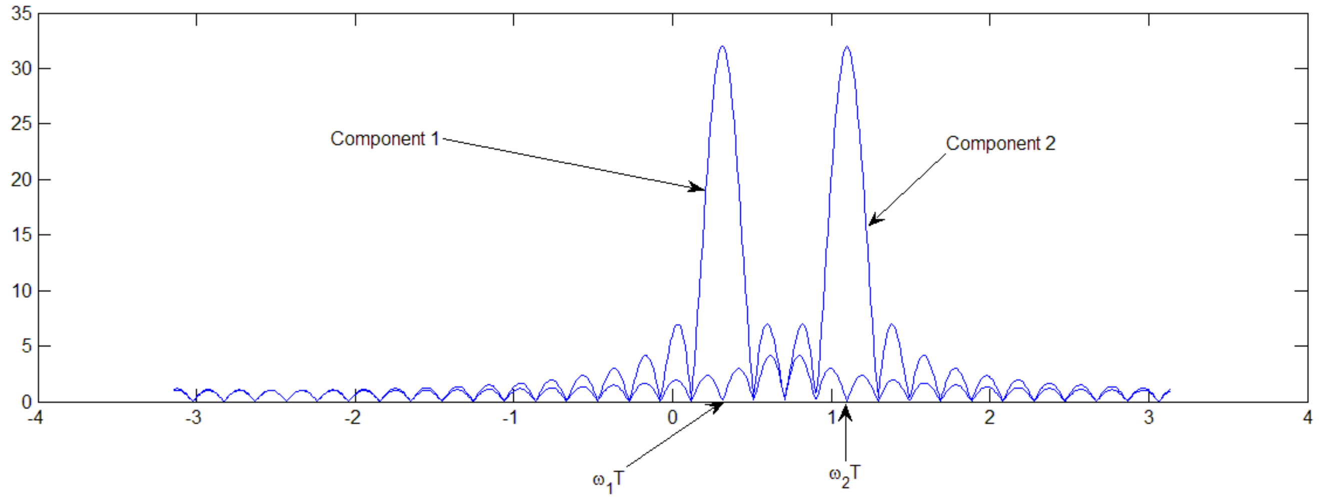
The DTFT is given by a train of delta functions:

$$X(e^{j\omega T}) = 2\pi \sum_{n=-\infty}^{+\infty} \delta(\omega T + 2n\pi - \omega_1 T) + \delta(\omega T + 2n\pi - \omega_2 T)$$

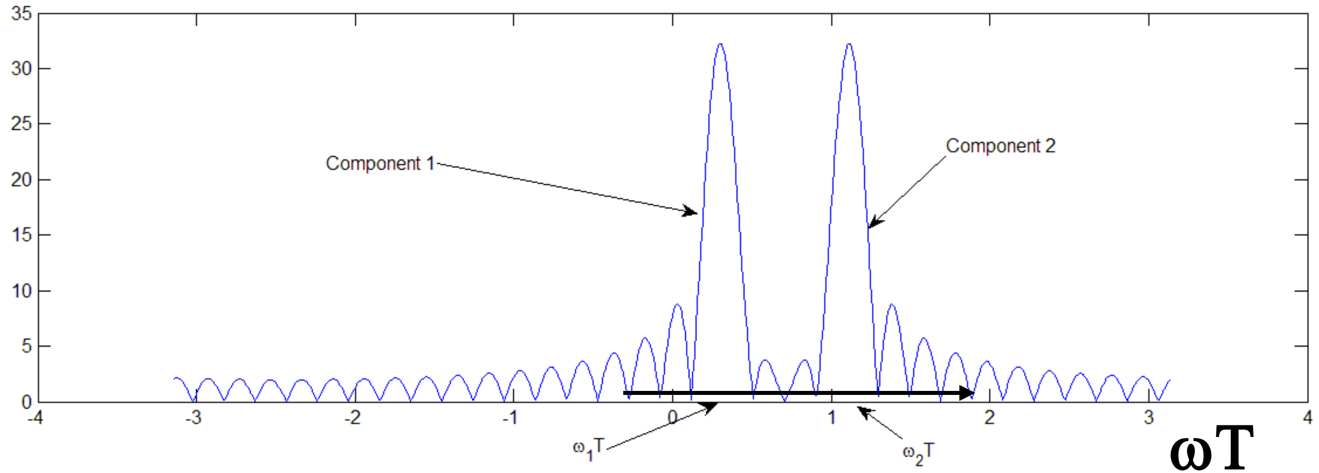
Hence the windowed spectrum is just the convolution of the window spectrum with the delta functions:

Now consider the DFT for the data:

Both components separately



Both components Together



ωT

Summary

- The rectangular window introduces broadening of any frequency components (`smearing') and sidelobes that may overlap with other frequency components (`leakage').
- The effect improves as N increases
- However, the rectangle window has poor properties and better choices of w_n can lead to better spectral properties (less leakage, in particular) – i.e. instead of just truncating the summation, we can pre-multiply by a suitable window function w_n that has better frequency domain properties.
- More on window design in the filter design section of the course – see later