DTFT & FFTs

http://elec3004.com

ELEC 3004: **Systems**: Signals & Controls

Tim Sherry & Surya Singh

Lecture 19

elec3004@itee.uq.edu.au

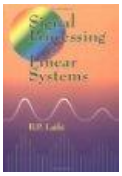
http://robotics.itee.uq.edu.au/~elec3004/

May 10, 2019

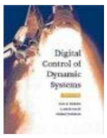
Lecture Schedule:

Week	Date	Lecture Title
1	27-Feb	Introduction
	1-Mar	Systems Overview
2	6-Mar	Systems as Maps & Signals as Vectors
	8-Mar	Systems: Linear Differential Systems
3	13-Mar	Sampling Theory & Data Acquisition
	15-Mar	Aliasing & Antialiasing
4	20-Mar	Discrete Time Analysis & Z-Transform
	22-Mar	Second Order LTID (& Convolution Review)
5	27-Mar	Frequency Response
	29-Mar	Filter Analysis
6	3-Apr	Digital Filters (IIR) & Filter Analysis
	5-Apr	PS 1: Q & A
7	10-Apr	Digital Windows
	12-Apr	Digital Filter (FIR)
8	17-Apr	Active Filters & Estimation
	19-Apr	Holiday
	24-Apr	
	26-Apr	
9	1-May	Introduction to Feedback Control
	3-May	Servoregulation & PID Control
10	8-May	State-Space Control
	10-May	**Guest Lecture: FFT**
11	15-May	Advanced PID
	17-May	State Space Control System Design
12	22-May	Digital Control Design
	24-May	Shaping the Dynamic Response
13	29-May	System Identification & Information Theory & Information Space
	31-May	Summary and Course Review

Follow Along Reading:



B. P. Lathi
Signal processing and linear systems
1998
TK5102.9.L38 1998



G. Franklin,
J. Powell,
M. Workman
Digital Control of Dynamic Systems
1990

TJ216.F72 1990
[Available as UQ Ebook]

Today

- Chapter 10 (**Discrete-Time System Analysis Using the *z*-Transform**)
  - § 10.3 Properties of DTFT
  - § 10.5 Discrete-Time Linear System analysis by DTFT
  - § 10.7 Generalization of DTFT to the $\mathcal{Z}$–Transform

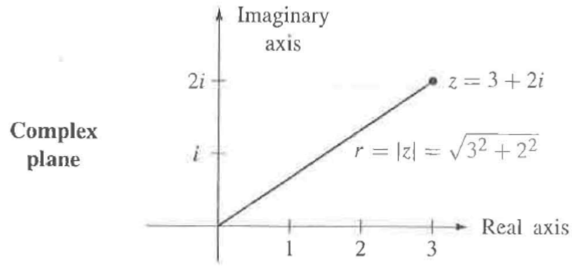
- FPW
  - Chapter 2: Linear, Discrete, Dynamic-Systems Analysis

Next Time

ELEC3004 is

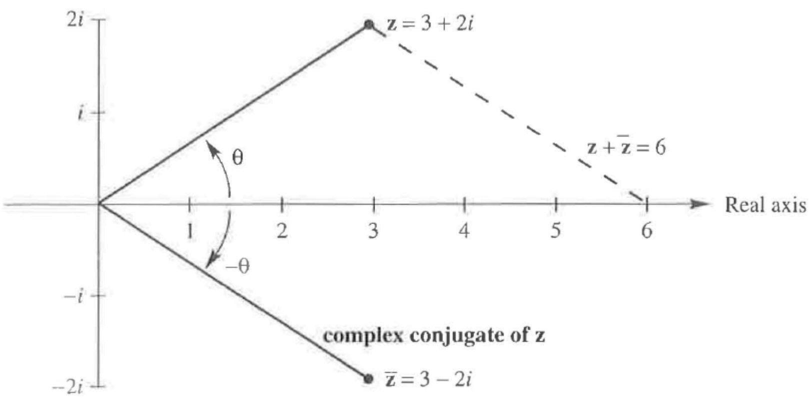
$$-e^{\pi i}$$

The Complex Plane Properties



- z=(a + bi)
- $z + \bar{z} = 2a$
- $z\bar{z} = (a + bi)(a - bi) = a^2 + b^2$

The Complex Plane Properties



- z=(a + bi) is also
- $z = r\cos\theta + ir\sin\theta$

The nth power of $z = r(\cos\theta + i\sin\theta)$ *is* $z^n = r^n(\cos n\theta + i\sin n\theta)$.

The Complex Plane Properties

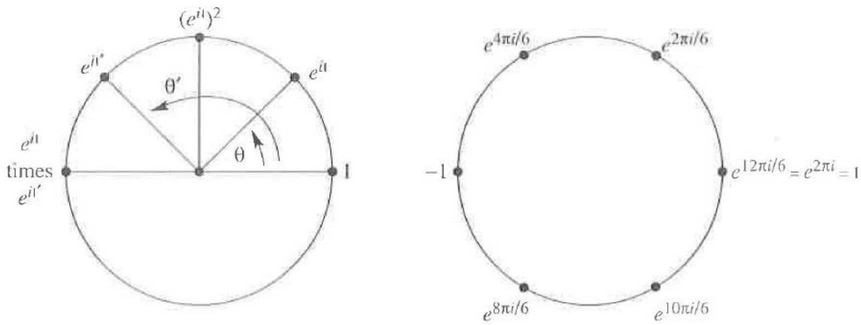
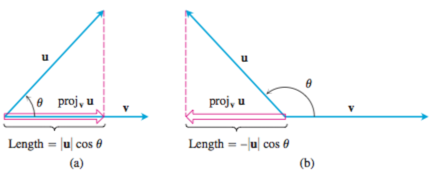


Figure 10.3 (a) Multiplying $e^{i\theta}$ times $e^{i\theta'}$. (b) The nth power of $e^{2\pi i/n}$ is $e^{2\pi i} = 1$.

The Inner product as a projection



$$Proj_{\{u \to v\}} = \frac{u \cdot v}{||v||^2} v$$

- What happens if v is unit length?
- What can be said about the error (pink dotted trace) wrt. v?
- What about multiple projections?
  - what condition between v_1, v_2, v_3 ... would we like?

DTFT

The Fourier Transform

- The continuous-time Fourier Transform

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

- What happens if we sample $x(t)|_{t=n\Delta t} = x_c(t)$?
- Represent $x_c(t)$ as sum of weighted impulses

$$x_c(t) = \sum_{n=-\infty}^{\infty} x(n\Delta t)\delta(t - n\Delta t)$$

$$X_c(\omega) = \int_{-\infty}^{\infty}\left[\sum_{n=-\infty}^{\infty} x(n\Delta t)\delta(t - n\Delta t)\right]\exp(-j\omega t)dt$$

Discrete-time Fourier Transform

- Changing order of integration & summation
  - and the simplifying (multiplication by impulse) gives

$$X_c(\omega) = \sum_{n=-\infty}^{\infty} x(n\Delta t)\left[\int_{-\infty}^{\infty} \delta(t - n\Delta t)\exp(-j\omega t)dt\right]$$

$$= \sum_{n=-\infty}^{\infty} x(n\Delta t)\exp(-j\omega n\Delta t)$$

- This is known as the DTFT
  - Requires an infinite number of samples $x(n\Delta t)$
  - discrete in time
  - continuous and periodic in frequency

DTFT of Finite Data Samples

- Assume only N samples of $x(n\Delta t)$
  - from $n = \{0, N - 1\}$
- Therefore, can only approximate $X_c(w)$

$$\hat{X}_c(\omega) = \sum_{n=0}^{N-1} x(n\Delta t)\exp(-j\omega n\Delta t)$$

- How good an estimate is this?
  - Finite samples are same as infinite sequence multiplied by a rectangular time domain 'window'

$$\hat{x}(n\Delta t) = x(n\Delta t)\cdot\Pi\left(\frac{t}{T}\right), \qquad \text{where } T = N\Delta t$$

Where rect(t) = $\Pi(t) = u\left(t + \frac{T}{2}\right) - u\left(t - \frac{T}{2}\right)$

DTFT and the DFT

- Fourier transform, $\hat{X}_c(w)$, of sampled data is
  - continuous in frequency, range $\{0, w_s\}$
  - and periodic (w_s)
  - known as DTFT
- If calculating on digital computer
  - then only calculate $\hat{X}_c(w)$ at discrete frequencies
  - normally equally spaced over $\{0, w_s\}$
  - normally N samples, i.e., same as in time domain
  - i.e, samples Δw apart

Can reduce Δw by increasing N

$$\Delta\omega = \frac{2\pi}{N\Delta t}$$

DFT

The DFT

- Discrete Fourier Transform (DFT)
  - samples of DTFT, $X^{\wedge}{}_c(w)|_{w = k\,\Delta w}$

$$\hat{X}(k\Delta\omega) = X[k] = \sum_{n=0}^{N-1} x[n] \exp\left(\frac{-jnk2\pi}{N}\right)$$

where $0 \le n, k \le N-1$

- Interpretation:
  - N equally spaced samples of $x(t)|_{t = n\,\Delta t}$
  - Calculates N equally spaced samples of $X(w)|_{w = k\Delta w}$
  - k often referred to a frequency 'bin': $X[k] = X(w_k)$

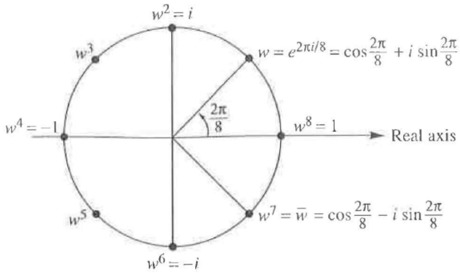
The DFT

$$X[k] = \sum_{n=0}^{N-1} x[n] \exp\left(\frac{-j2\pi nk}{N}\right)$$

- Sample number n where $0 \le n < N - 1$
  - time 0 to $N\Delta t$
- Frequency sample (bin) number k where $0 \le k < N - 1$
  - frequency 0 to ω_s $\left(\omega_s = \frac{2\pi}{\Delta t}\right)$
- Discrete in both time $x[n]$ and frequency $X[k]$
- Periodic in both time and frequency (due to sampling)
- Remember: $H(w) = H(z)|_z = \exp(j\omega t)$

  i.e., DFT samples around unit circle in the z-plane

Fourier Matrix



$$F_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & w & w^2 & w^3 \\ 1 & w^2 & w^4 & w^6 \\ 1 & w^3 & w^6 & w^9 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & i^2 & i^3 \\ 1 & i^2 & i^4 & i^6 \\ 1 & i^3 & i^6 & i^9 \end{bmatrix}.$$

$$Fc = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & w & w^2 & w^3 \\ 1 & w^2 & w^4 & w^6 \\ 1 & w^3 & w^6 & w^9 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} c_0 + c_1 + c_2 + c_3 \\ c_0 + c_1 w + c_2 w^2 + c_3 w^3 \\ c_0 + c_1 w^2 + c_2 w^4 + c_3 w^6 \\ c_0 + c_1 w^3 + c_2 w^6 + c_3 w^9 \end{bmatrix}$$

Naïve DFT in Matlab

```
%% function X : MyNiaveDFT(x)
% ELEC3004 - Lecture 13
function X  = MyNiaveDFT(x)
% Niave/direct implementation of the Discrete Fourier Transform (DFT)

% Calculate N samples of the DTFT, i.e., same number of samples
N=length(x);

% Initialize (complex) X to zero
X=[complex(zeros(size(x)),zeros(size(x)))];

for n = 0:N-1
    for k=0:N-1
        % Calculate each sample of DFT uSIng each sample of input.
        % Note: Matlab indexes vectors from 1 to N,
        % whilst DFT is defined from from 0 to (N-1)
        X(k+1) = X(k+1) + x(n+1)*exp(-i*n*k*2*pi/N);
    end
end
```

Computational Complexity

- Each frequency sample X[k]
  - Requires N complex multiply accumulate (MAC) operations
- $\therefore$ for N frequency samples
  - There are N^2 complex MAC
- Example:
  - 8-point DFT requires 64 MAC
  - 64-point DFT requires 4,096 MAC
  - 256-point DFT requires 65,536 MAC
  - 1024-point DFT requires 1,048,576 MAC
  - i.e., number of MACs gets very large, very quickly!

DFT Notation

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

Where: $$W_N^{nk} = \exp\left(\frac{-j2\pi nk}{N}\right)$$

W_N^{nk} are called "N^{th} roots of unity"

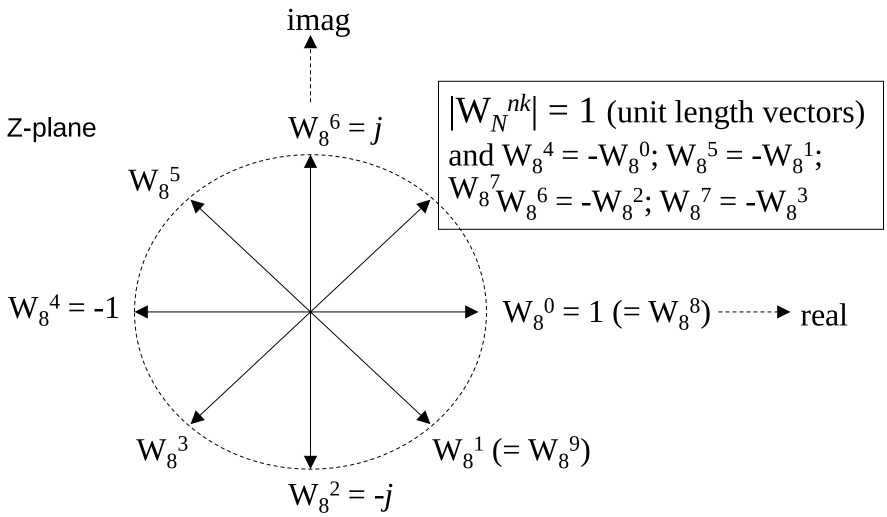
e.g., $N = 8$:

$W_8^0 = \exp(0) = 1$;

$W_8^1 = \exp(-j\pi/4) = \cos(\pi/4) - j\sin(\pi/4) = 0.7 - j0.7$;

$W_8^2 = -j$; $W_8^3 = -0.7 - j0.7$; $W_8^4 = -1$; etc

N^{th} Roots of Unity



All $2\pi/N$ apart

DFT Expansion

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

$$\begin{array}{lcccccccc}
X(k=0) & = & x(0) & + & x(1) & + & \ldots & + & x(N-1) \\
X(k=1) & = & x(0) & + & x(1)W_N^1 & + & \cdots & + & x(N-1)W_N^{N-1} \\
X(k=2) & = & x(0) & + & x(1)W_N^2 & + & \cdots & + & x(N-1)W_N^{N-2} \\
\vdots & & \vdots & & \vdots & & & & \vdots \\
X(k=N-1) & = & x(0) & + & x(1)W_N^{N-1} & + & \cdots & + & x(N-1)W_N^1
\end{array}$$

Remember $W_N^0 = 1$

DFT Matrix Formulation

DFT expansion can also be written as a matrix operation:

$$\underbrace{\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ . \\ X(N-1) \end{bmatrix}}_{X[k]} = \underbrace{\begin{bmatrix} 1 & 1 & 1 & . & 1 \\ 1 & W_N^1 & W_N^2 & . & W_N^{N-1} \\ 1 & W_N^2 & W_N^4 & . & W_N^{N-2} \\ . & . & . & . & . \\ 1 & W_N^{N-1} & W_N^{N-2} & . & W_N^1 \end{bmatrix}}_{\text{DFT Matrix}} \cdot \underbrace{\begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ . \\ x(N-1) \end{bmatrix}}_{x[n]}$$

Example: 8-point DFT Matrix

no rotation DC row | 1 rotation | 2 rotations | etc

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ X(4) \\ X(5) \\ X(6) \\ X(7) \end{bmatrix} = \begin{bmatrix} W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 \\ W_8^0 & W_8^1 & W_8^2 & W_8^3 & W_8^4 & W_8^5 & W_8^6 & W_8^7 \\ W_8^0 & W_8^2 & W_8^4 & W_8^6 & W_8^0 & W_8^2 & W_8^4 & W_8^6 \\ W_8^0 & W_8^3 & W_8^6 & W_8^1 & W_8^4 & W_8^7 & W_8^2 & W_8^5 \\ W_8^0 & W_8^4 & W_8^0 & W_8^4 & W_8^0 & W_8^4 & W_8^0 & W_8^4 \\ W_8^0 & W_8^5 & W_8^2 & W_8^7 & W_8^4 & W_8^1 & W_8^6 & W_8^3 \\ W_8^0 & W_8^6 & W_8^4 & W_8^2 & W_8^0 & W_8^6 & W_8^4 & W_8^2 \\ W_8^0 & W_8^7 & W_8^6 & W_8^5 & W_8^4 & W_8^3 & W_8^2 & W_8^1 \end{bmatrix} \cdot \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \\ x(4) \\ x(5) \\ x(6) \\ x(7) \end{bmatrix}$$

Increasing rotational frequency down the rows of the DFT matrix

Example: 8-point DFT Matrix

$$
\begin{array}{c}
\\
\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ X(4) \\ X(5) \\ X(6) \\ X(7) \end{bmatrix}
\end{array}
=
\begin{array}{c}
\begin{array}{cccccccc} x(0) & & x(2) & & x(4) & & x(6) & \end{array} \\
\begin{bmatrix}
W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 \\
W_8^0 & W_8^1 & W_8^2 & W_8^3 & W_8^4 & W_8^5 & W_8^6 & W_8^7 \\
W_8^0 & W_8^2 & W_8^4 & W_8^6 & W_8^0 & W_8^2 & W_8^4 & W_8^6 \\
W_8^0 & W_8^3 & W_8^6 & W_8^1 & W_8^4 & W_8^7 & W_8^2 & W_8^5 \\
W_8^0 & W_8^4 & W_8^0 & W_8^4 & W_8^0 & W_8^4 & W_8^0 & W_8^4 \\
W_8^0 & W_8^5 & W_8^2 & W_8^7 & W_8^4 & W_8^1 & W_8^6 & W_8^3 \\
W_8^0 & W_8^6 & W_8^4 & W_8^2 & W_8^0 & W_8^6 & W_8^4 & W_8^2 \\
W_8^0 & W_8^7 & W_8^6 & W_8^5 & W_8^4 & W_8^3 & W_8^2 & W_8^1
\end{bmatrix}
\end{array}
\cdot
\begin{array}{c}
\text{Even samples} \\
\begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \\ x(4) \\ x(5) \\ x(6) \\ x(7) \end{bmatrix}
\end{array}
$$

Repeated complex multiplications in EVEN rows

Properties of the DTFT and DFT

Window Effects

- Multiplication in time with rectangular window
- Equivalent to convolution in frequency
  - with 'sinc' function

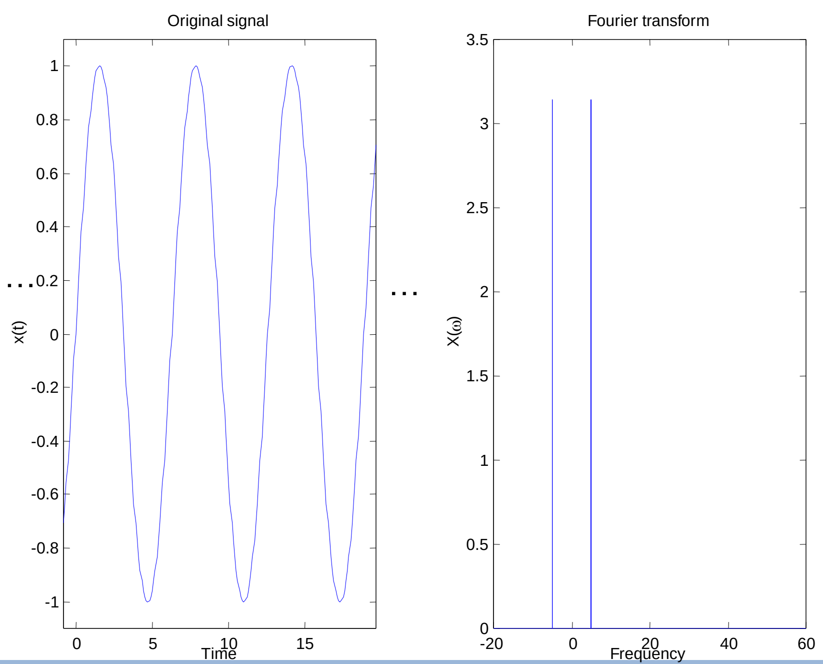
$$\hat{X}_c(\omega) = \frac{1}{2\pi} X_c(\omega) * T \,\mathrm{sinc}\left(\frac{T\omega}{2\pi}\right)$$

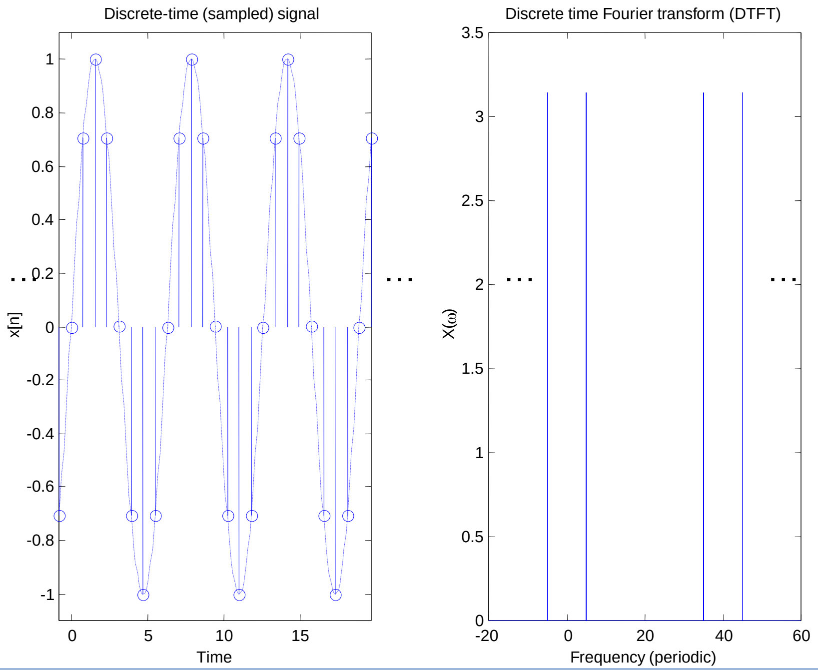
- In general, with arbitrary window function

$$\hat{x}_c(t) = x_c(t) \cdot w_T(t)$$

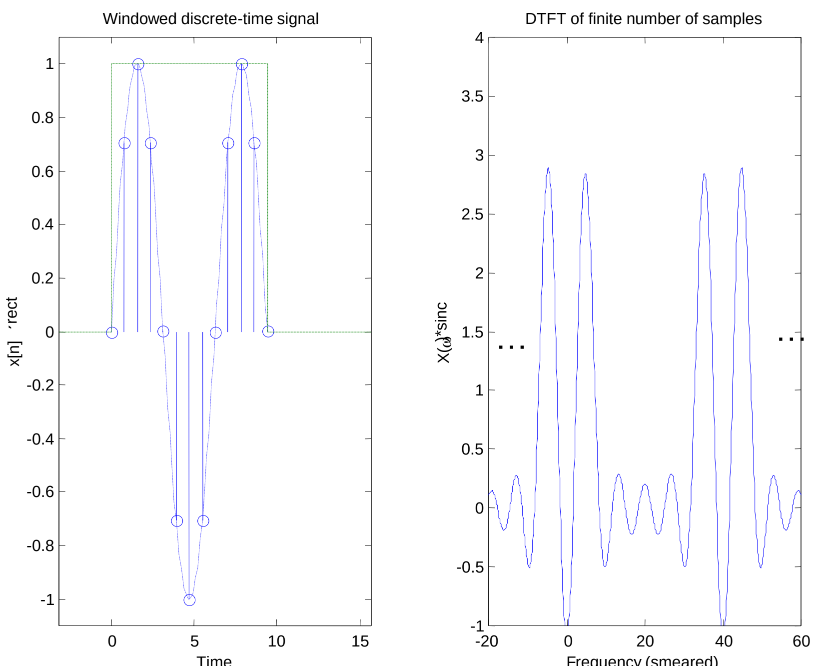
$$\hat{X}_c(\omega) = \frac{1}{2\pi} X_c(\omega) * W_T(\omega)$$

This is exactly same effect we saw in FIR filter design





ELEC 3004: **Systems** 10 May 2019 29

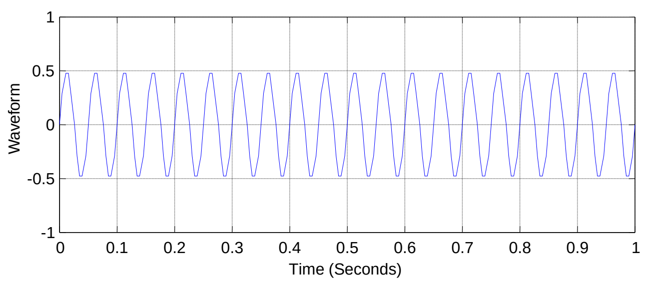


ELEC 3004: **Systems** 10 May 2019 30

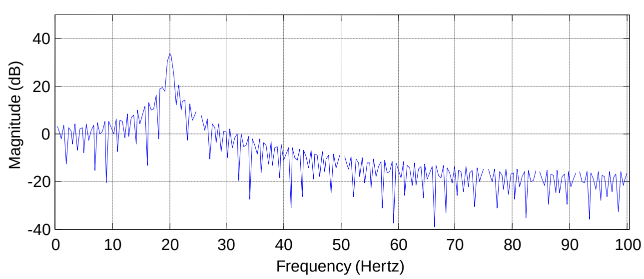
Reducing Window Effects

- We cannot avoid using a window function
  - as we must use a finite length of data

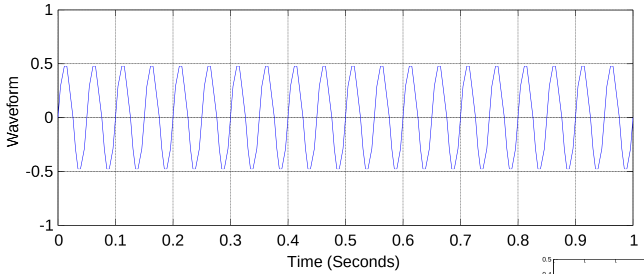
- Aim: to reduce window effect
  1. By choosing suitable window function
     - Hanning, Hamming, Blackman, Kaiser etc
  2. Increase number of samples (N)
     - reduces window effect (larger window)
     - increases resolution (No. samples)
     - Assumes signal is 'stationary' within sample window
       - Not true for most non-deterministic signals
       - e.g., speech, images etc



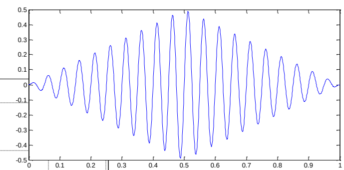
sinewave
frequency 20Hz
rectangular window
(1 second long)

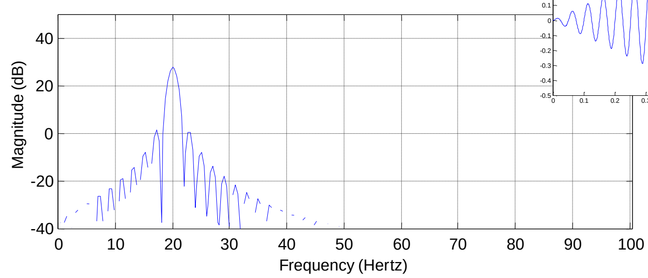


Note difficulty in detecting the single sinewave frequency present in time domain due to 'smearing.'

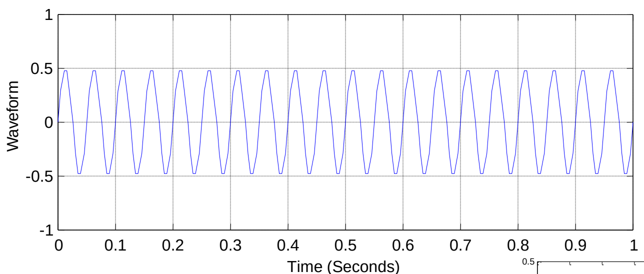


sinewave
frequency 20Hz
Triangular window
(AKA Bartlett)

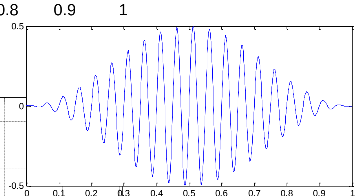


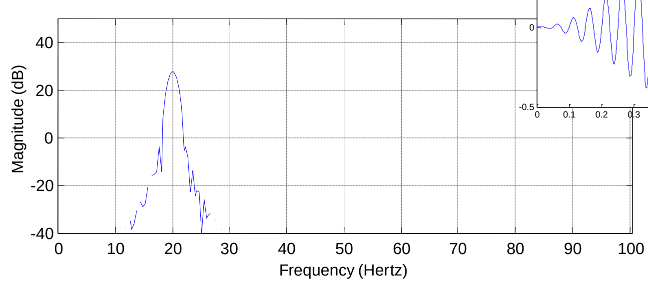


Wider main lobe
Increased side lobe
attenuation

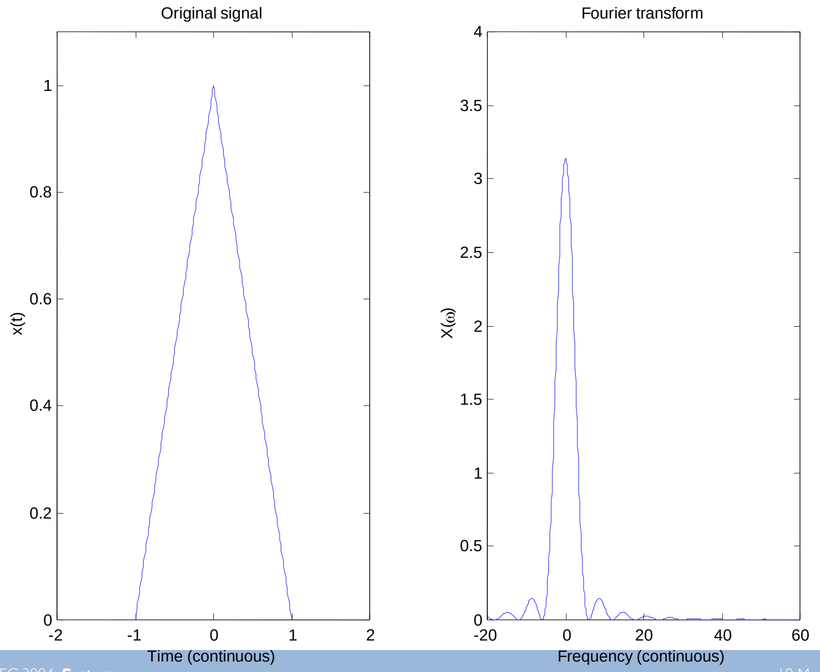


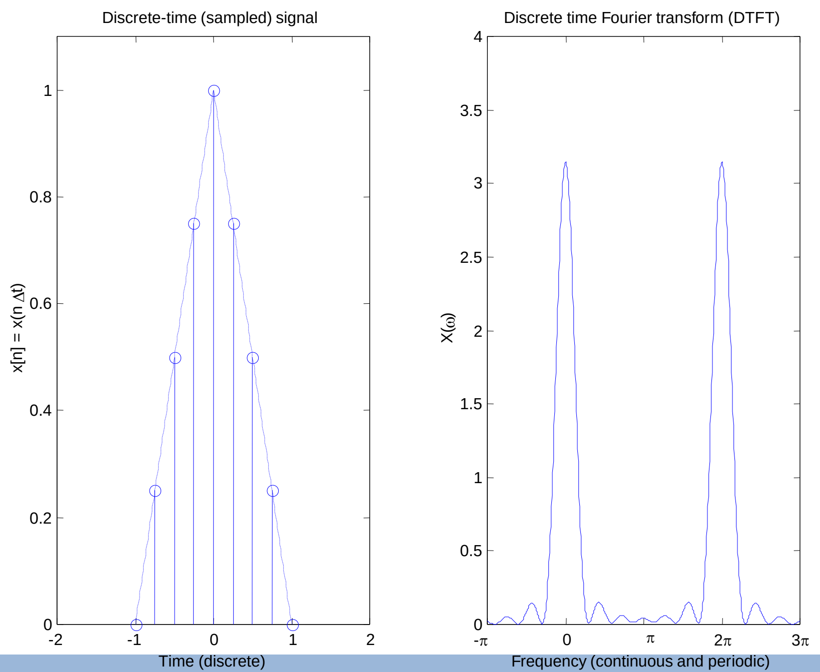
sinewave
frequency 20Hz
Hanning window

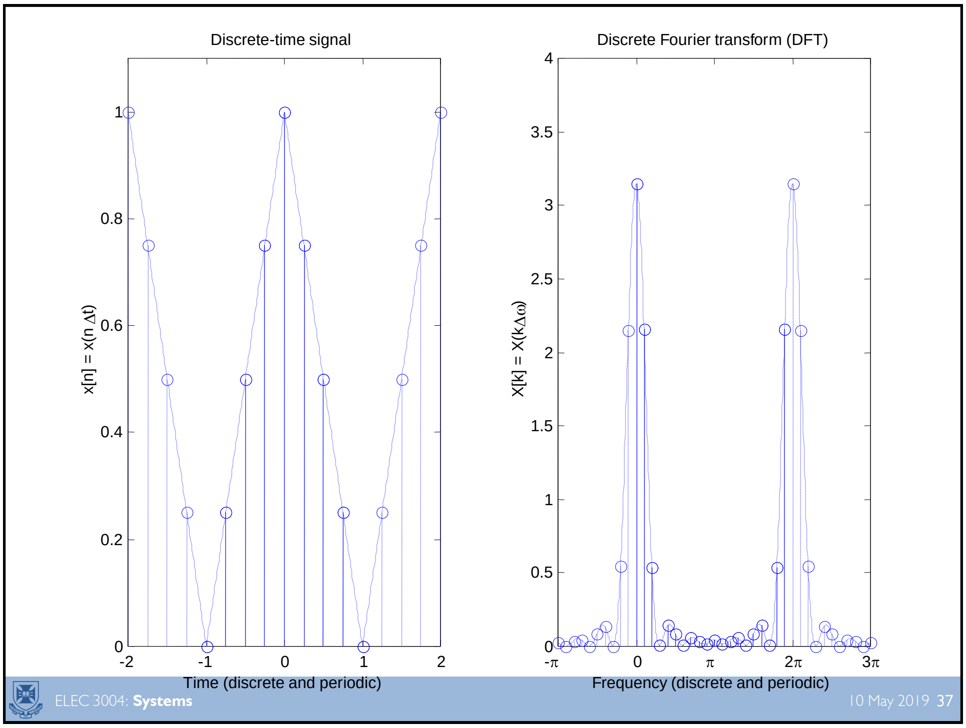


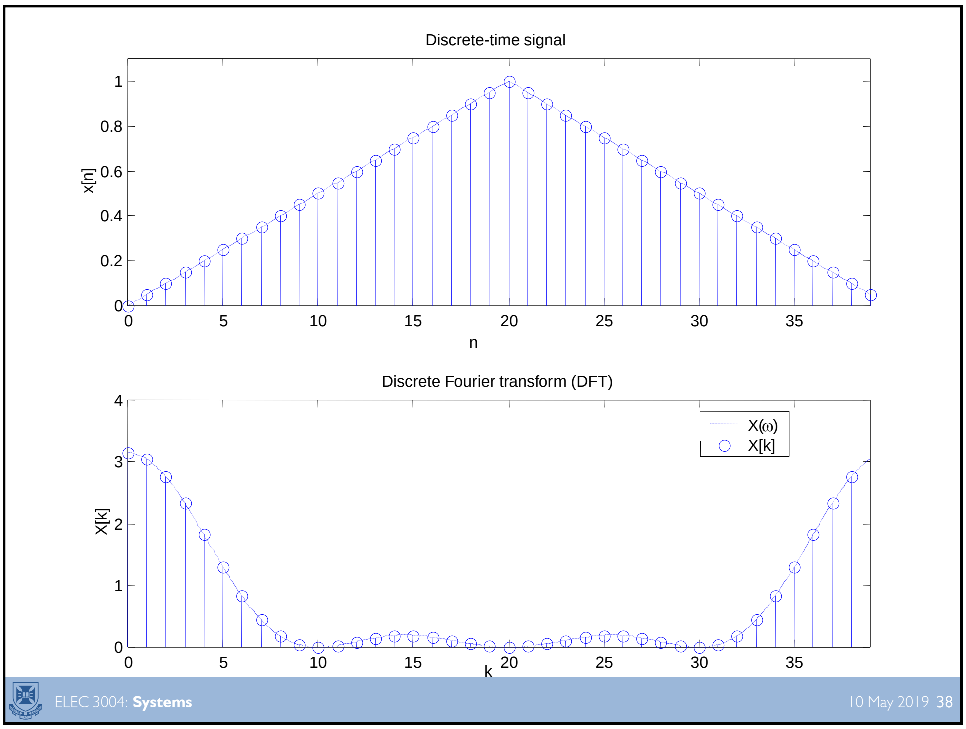


single sine wave
easier to detect


Original signal
x(t)
Time (continuous)
Fourier transform
X(ω)
Frequency (continuous)
0
0.2
0.4
0.6
0.8
1
-2
-1
0
1
2
0
0.5
1
1.5
2
2.5
3
3.5
4
-20
0
20
40
60


Discrete-time (sampled) signal
x[n] = x(n Δt)
Time (discrete)
Discrete time Fourier transform (DTFT)
X(ω)
Frequency (continuous and periodic)
0
0.2
0.4
0.6
0.8
1
-2
-1
0
1
2
0
0.5
1
1.5
2
2.5
3
3.5
4
-π
0
π
2π
3π


Discrete-time signal
x[n] = x(n Δt)
Time (discrete and periodic)
Discrete Fourier transform (DFT)
X[k] = X(kΔω)
Frequency (discrete and periodic)
ELEC 3004: Systems
10 May 2019 37


Discrete-time signal
x[n]
n
Discrete Fourier transform (DFT)
X[k]
k
X(ω)
X[k]
ELEC 3004: Systems
10 May 2019 38

Inverse DFT

- Relates frequency domain samples to
  - time domain samples

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]\exp\left(\frac{jnk2\pi}{N}\right)$$

- Note, differences to forward DFT
  - 1/N scaling and sign change on exponential
  - DFT & IDFT implemented with same algorithm
    - i.e., Fast Fourier Transform (FFT)
- Require both DFT and IDFT to implement (fast)
  - convolution as multiplication in frequency domain

Note, 1/N scaling can be on DFT only OR as 1/sqrt(N) on both DFT and IDFT

Fourier Transforms

Transform	**Time Domain**	**Frequency Domain**
Fourier Series (FS)	Continuous & Periodic	Discrete
Fourier Transform (FT)	Continuous	Continuous
Discrete-time Fourier Transform (DTFT)	Discrete	Continuous & Periodic
Discrete Fourier Transform (DFT)	Discrete & Periodic	Discrete & Periodic

Properties of the DFT

if…

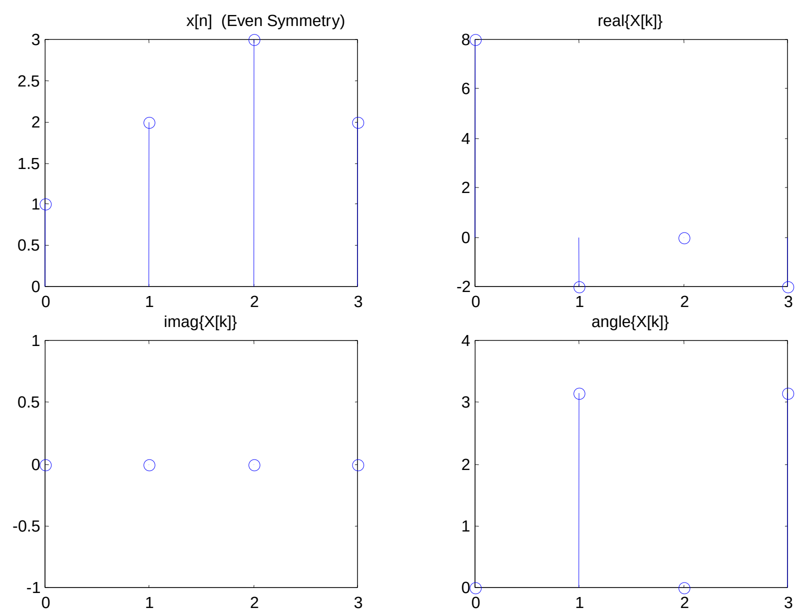
- $x[n]$ is real

- $x[n]$ is real and even

- $x[n]$ is real and odd

Then…

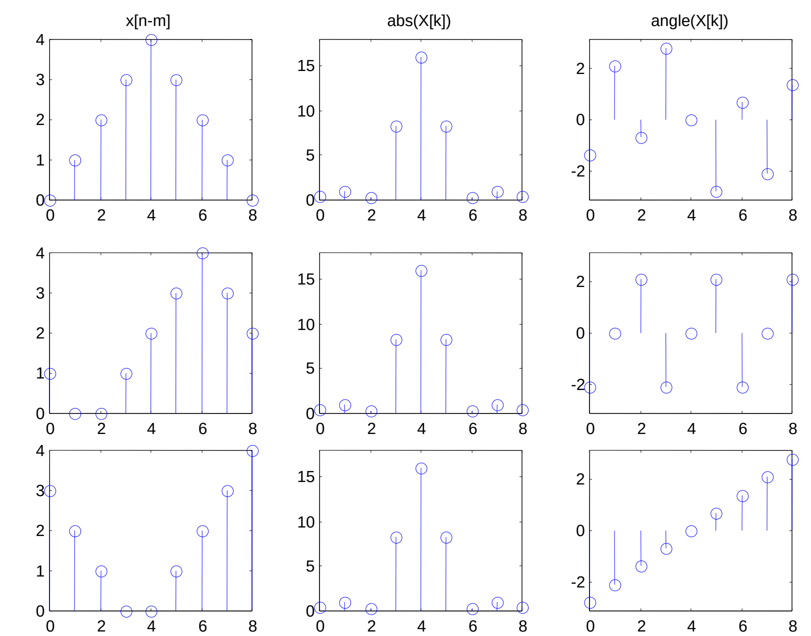
- $X[-k] = X[k]^*$
  - $\Re\{X[k]\}$ is even
  - $\Im\{X[k]\}$ is odd
  - $|X[k]|$ is even
  - $\angle X[k]$ is odd
- $X[k]$ is real and even
  - i.e., zero phase
- $X[k]$ is imaginary and odd



Note: DC at k = 0 & x[n] = x[n + N] & X[k] = X[k +N]

Properties of the DFT

- Periodic in frequency
  - period w_s i.e., the sampling frequency, or
  - period 2π (in normalised frequency)
- Repeats after N samples
  - $x[N + k] = X[k]$
- Mirror image (even) symmetry at $w_s/2$, i.e., π
  - $x[N - r] = X^*[r]$, where $r < N/2$
- Shift property
  - $x[n - m] = \exp(-jkm2\pi/N)\, X[k]$
  - i.e., $|X[k]|$ stays the same as input is shifted
  - only (phase) $\angle X[k]$ changes



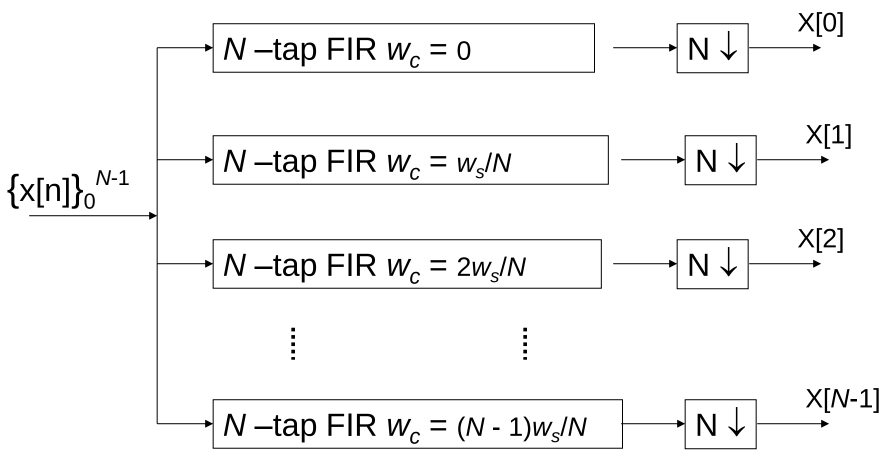
Note: all spectrums shifted to show $-w_s/2$ to $+ w_s/2$, i.e., DC at $k = 4$

Analogies for the DFT

- Analogy for DFT is a **Filterbank**
  - Set of N FIR bandpass filters
  - with centre frequencies kw_s/N
    - k in range $\{0, N-1\}$
  - often called 'frequency bins'

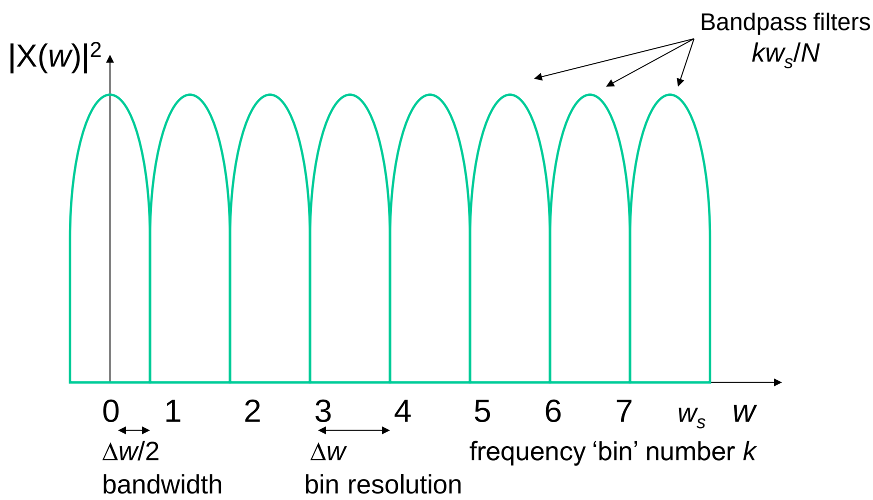
- e.g., 8 point DFT
  - 8 bandpass filters (bins), spaced $\Delta w = w_s/8$ apart
  - Bandwidth of each filter $\Delta w/2$ therefore
    - output can be down-sampled by factor of 8
    - i.e., one sample, $x[k]$, per filter output (frequency bin)

Filterbank Analogy of DFT



Down sample factor N

Filterbank Analogy of DFT

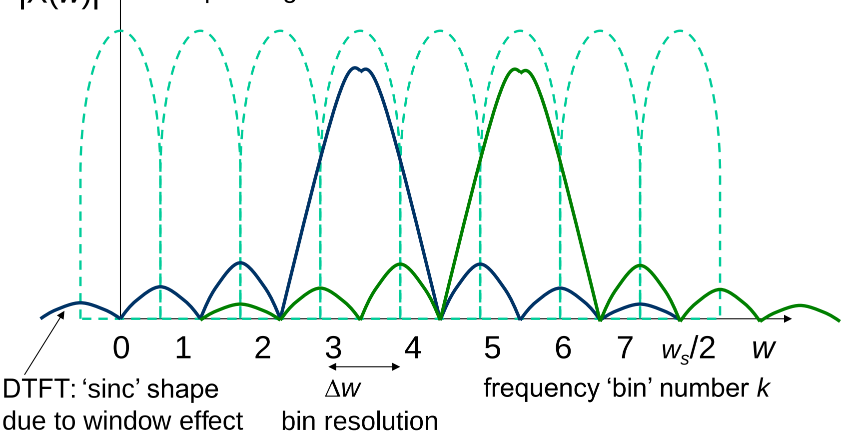


DFT Resolution

- Resolution is ability to distinguish
  - 2 (or more) closely spaced sinusoids
- <u>Minimum</u> resolution of DFT given by
  - $\Delta w = w_s/N = 2\pi/N\Delta t$
    - defined by sampling frequency, w_s
    - and number of samples, N
- Minimum resolution occurs when
  - integer number of complete cycles of input signal
  - in the N samples analysed
  - This is a 'best case' scenario
    - 'sinc' smearing always zero in adjacent frequency bins

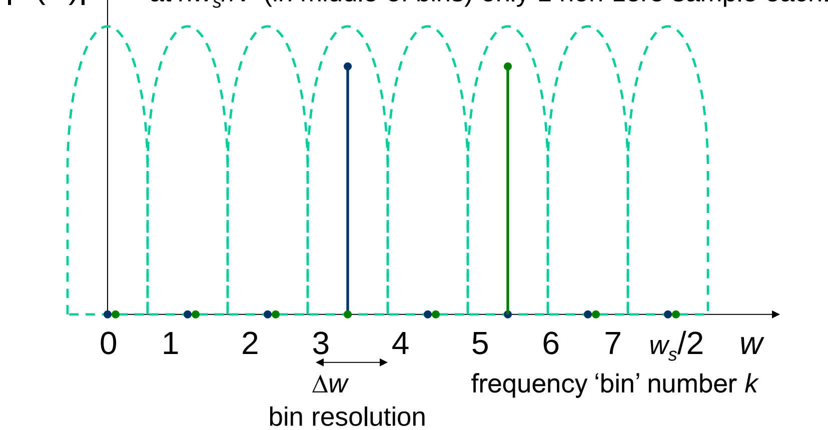
DFT Resolution: Example

Consider two sinusoids: frequencies $3w_s/16$ and $5w_s/16$
sample length $T = 16\Delta t$



DFT Resolution: Example

16 point DFT: results in samples of DTFT. As sinusoids are at nw_s/N (in middle of bins) only 1 non-zero sample each.

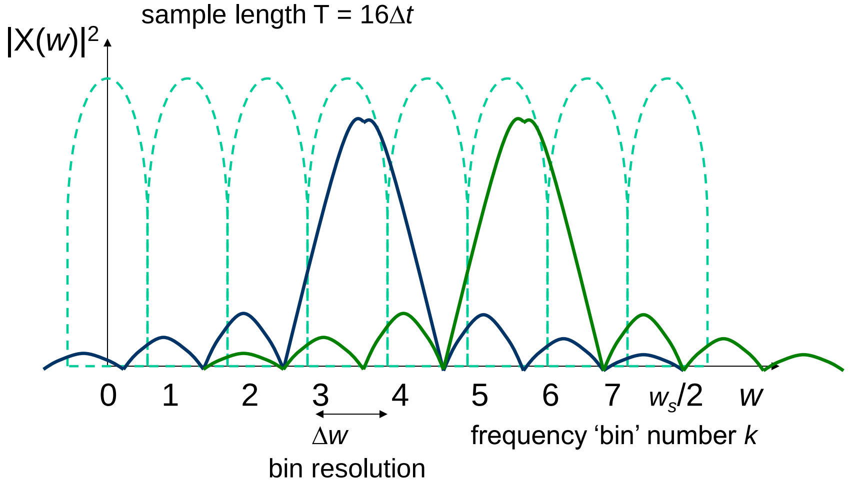


Leakage Effects

- In general, we can not capture
  - integer number of cycles of input
    - i.e., input will not be at bin frequencies nw_s/N
  - therefore, actual DFT resolution < Δw
- This is due to energy 'leakage'
  - between adjacent frequency bins
- Leakage due to finite data length
  - i.e., the 'window' effect
  - which 'smears' $X(w)$ -> $X[k]$
  - aim: to minimise window effect
    - using other than rectangular window

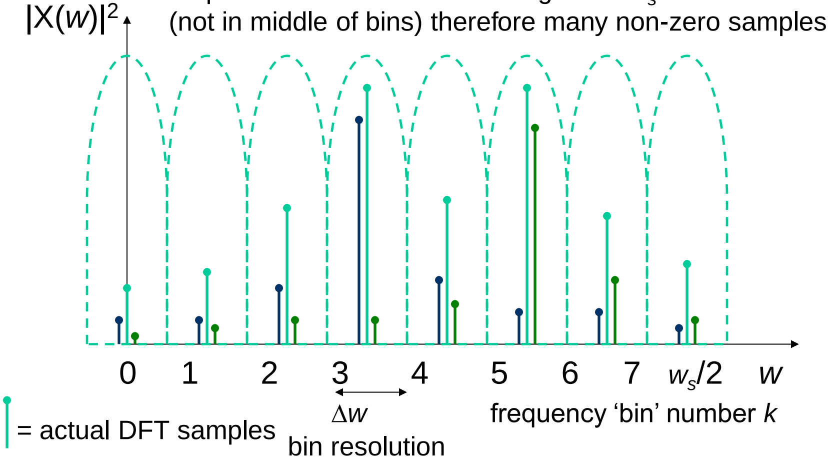
DFT Resolution: Example

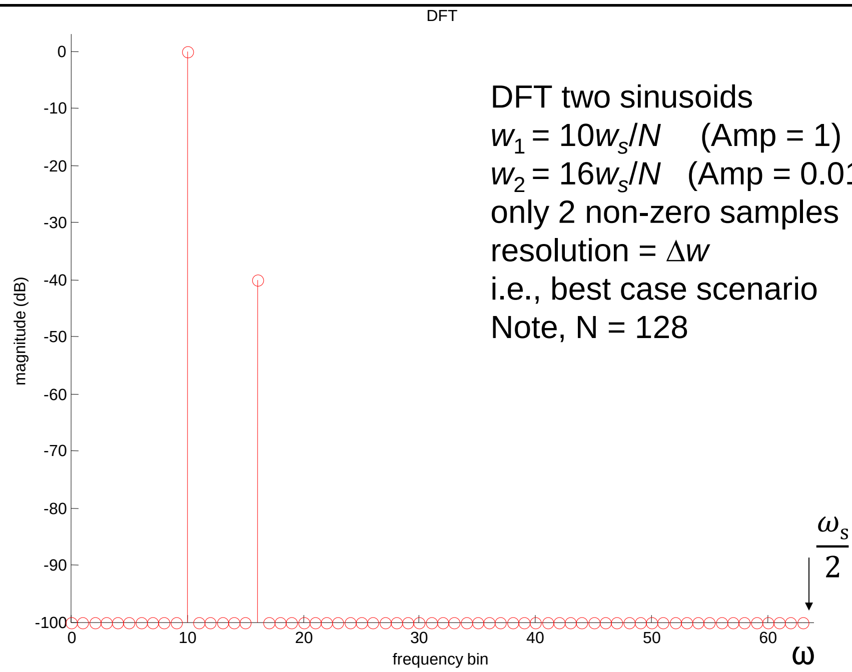
Consider two sinusoids: frequencies $3.1w_s/16$ and $5.1w_s/16$
sample length T = $16\Delta t$



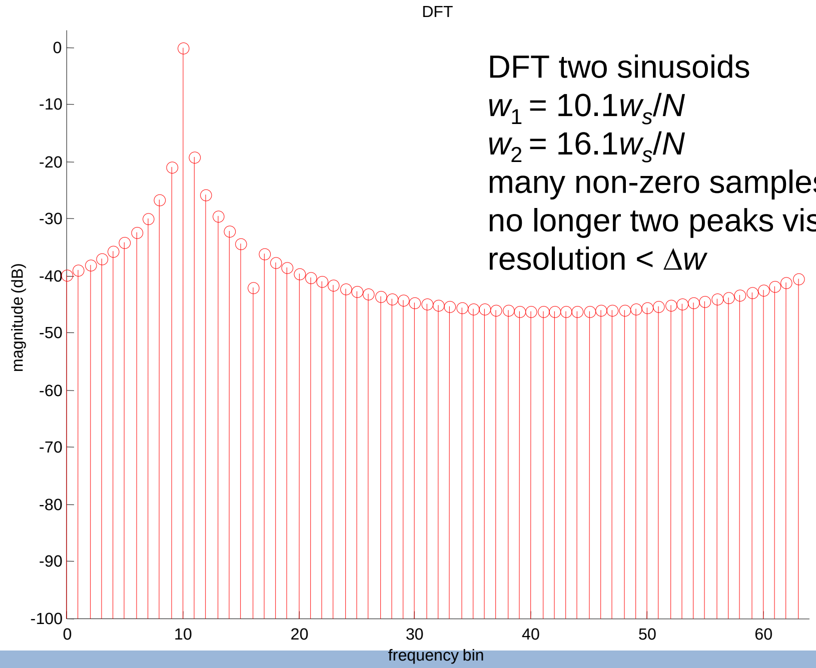
DFT Resolution: Example

16 point DFT: Sinusoids no longer at nw_s/N
(not in middle of bins) therefore many non-zero samples.

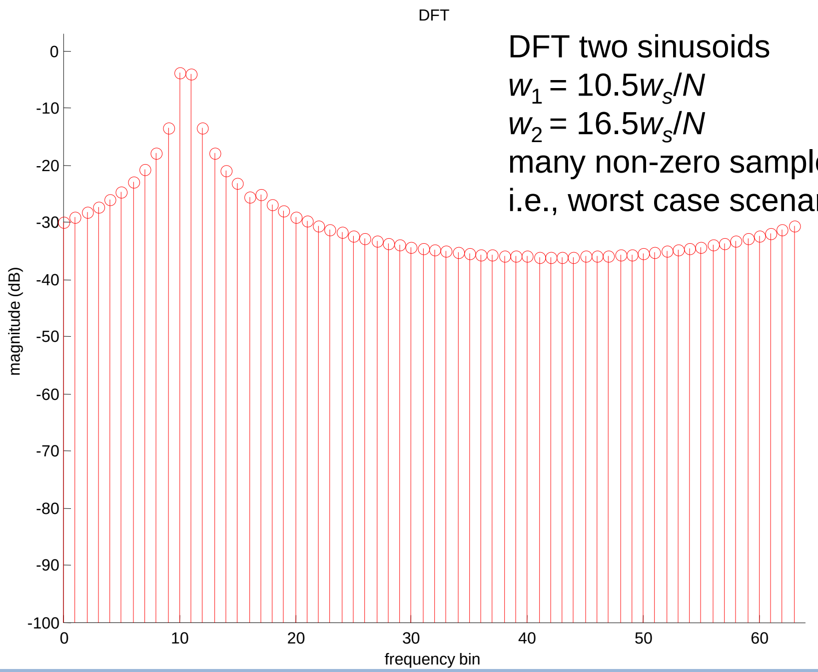




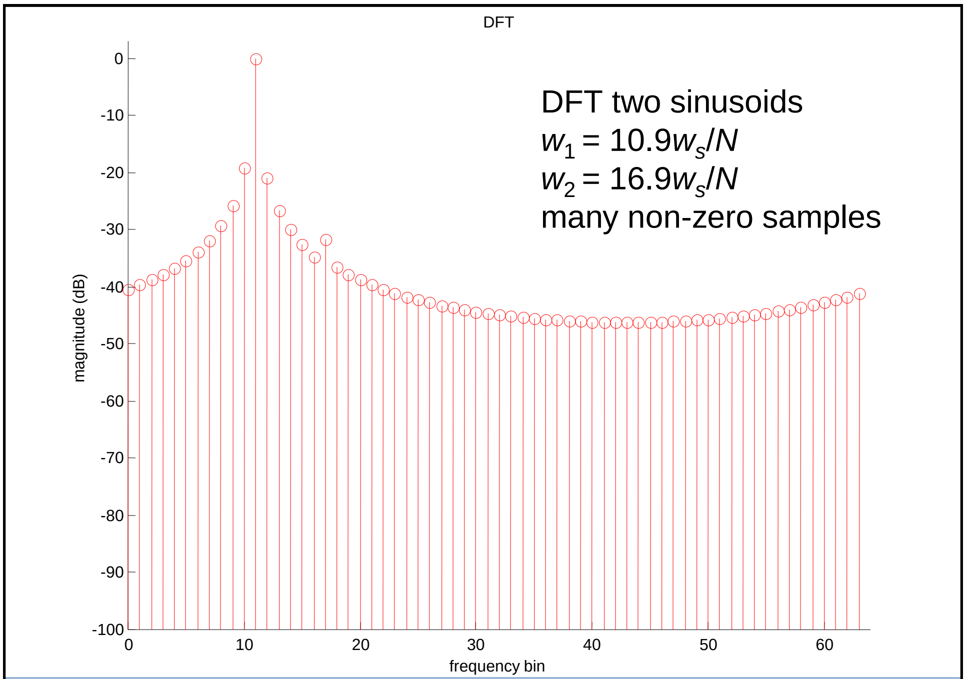
DFT two sinusoids
$w_1 = 10w_s/N$ (Amp = 1)
$w_2 = 16w_s/N$ (Amp = 0.01)
only 2 non-zero samples
resolution = Δw
i.e., best case scenario
Note, N = 128

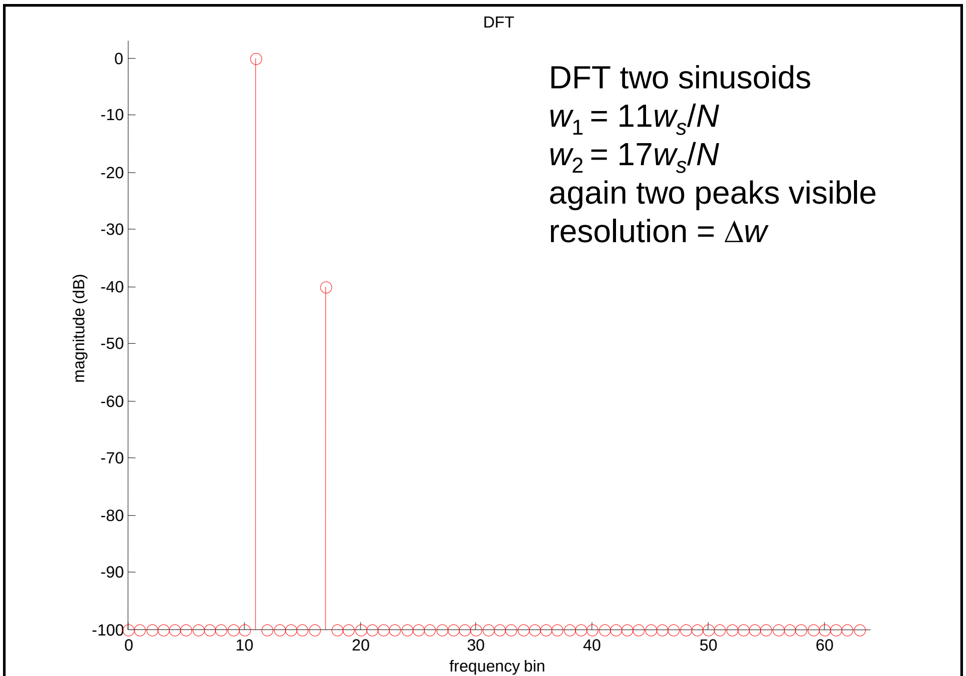


DFT two sinusoids
$w_1 = 10.1 w_s / N$
$w_2 = 16.1 w_s / N$
many non-zero samples
no longer two peaks visible
resolution < Δw



DFT two sinusoids
$w_1 = 10.5 w_s / N$
$w_2 = 16.5 w_s / N$
many non-zero samples
i.e., worst case scenario


DFT
DFT two sinusoids
$w_1 = 10.9 w_s/N$
$w_2 = 16.9 w_s/N$
many non-zero samples
magnitude (dB)
frequency bin

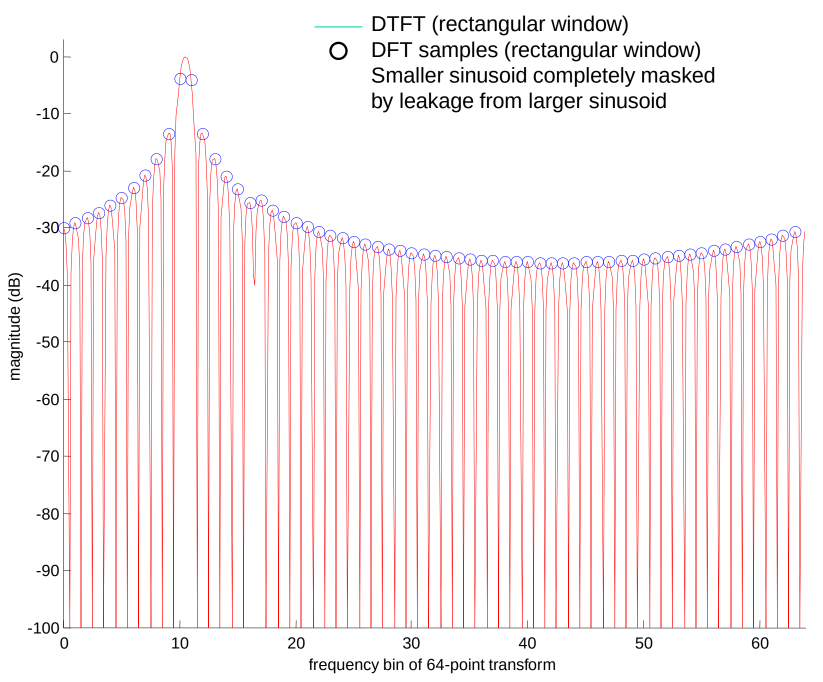

DFT
DFT two sinusoids
$w_1 = 11 w_s/N$
$w_2 = 17 w_s/N$
again two peaks visible
resolution = Δw
magnitude (dB)
frequency bin

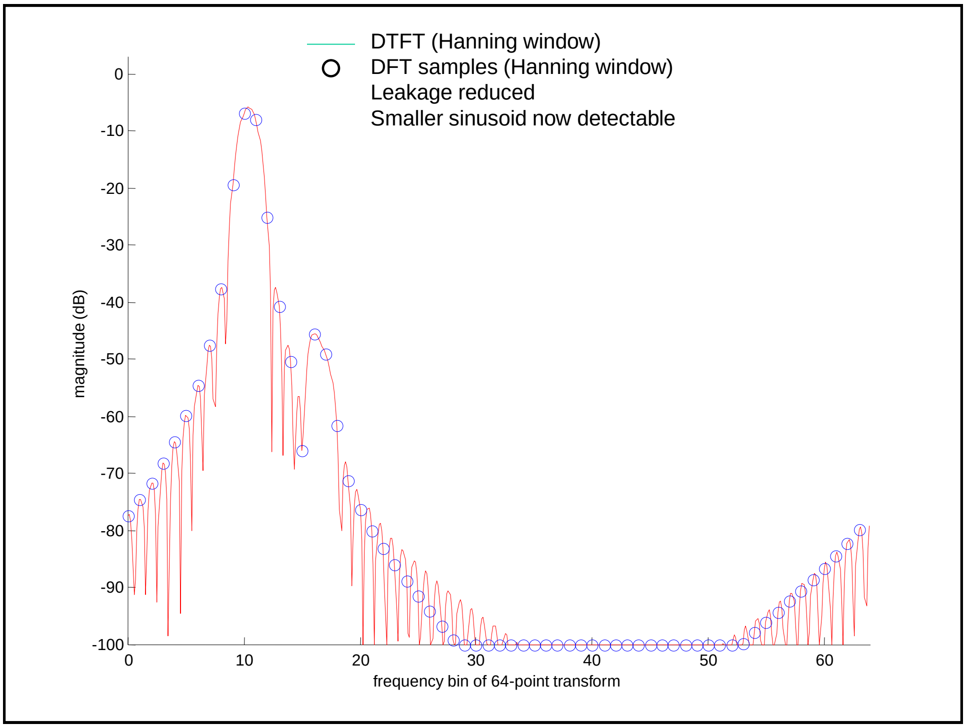
Reducing Leakage with Window Functions: Example

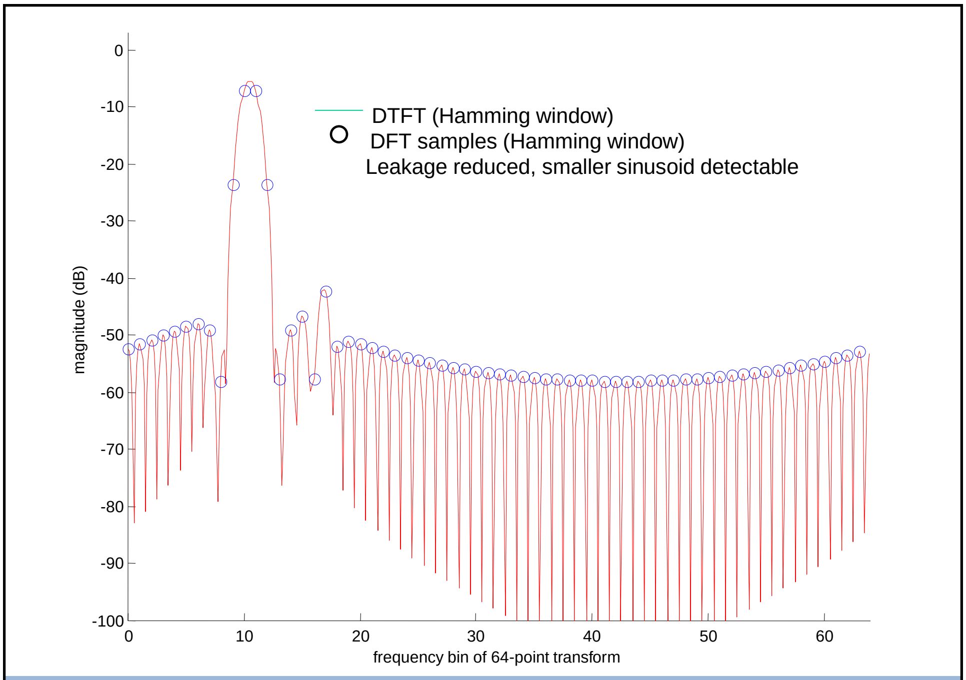
- Consider, two sinusoids,
  1. $\sin(10.5 w_s / N)$: amplitude 1
  2. $0.01 \sin(16.5 w_s / N)$: amplitude 0.01
     - i.e., significantly smaller (-40dB)

- This produces worst case leakage as
  – both sinusoids fall at edge of frequency bins
  – leakage due to large sinusoid > amplitude of smaller sinusoid (will be ‘masked’)

- Leakage can be reduced by using
  – non-rectangular window (Hanning/Hamming)
  – as used in FIR filter design




DTFT (Hanning window)
DFT samples (Hanning window)
Leakage reduced
Smaller sinusoid now detectable
magnitude (dB)
frequency bin of 64-point transform


DTFT (Hamming window)
DFT samples (Hamming window)
Leakage reduced, smaller sinusoid detectable
magnitude (dB)
frequency bin of 64-point transform

Window Functions

Window	-3dB bandwidth	Loss (dB)	Peak sidelobe (dB)	Sidelobe roll off (dB/octave)
Rectangular	$0.89/N\Delta t$	0	-13	-6
Hanning	$1.4/N\Delta t$	4	-32	-18
Hamming	$1.3/N\Delta t$	2.7	-43	-6
Dolph-Chebyshev	$1.44/N\Delta t$	3.2	-60	0

Note, trade-off between increased sidelobe attenuation
And increased 3dB (peak) bandwidth

Inverse and Interpolation

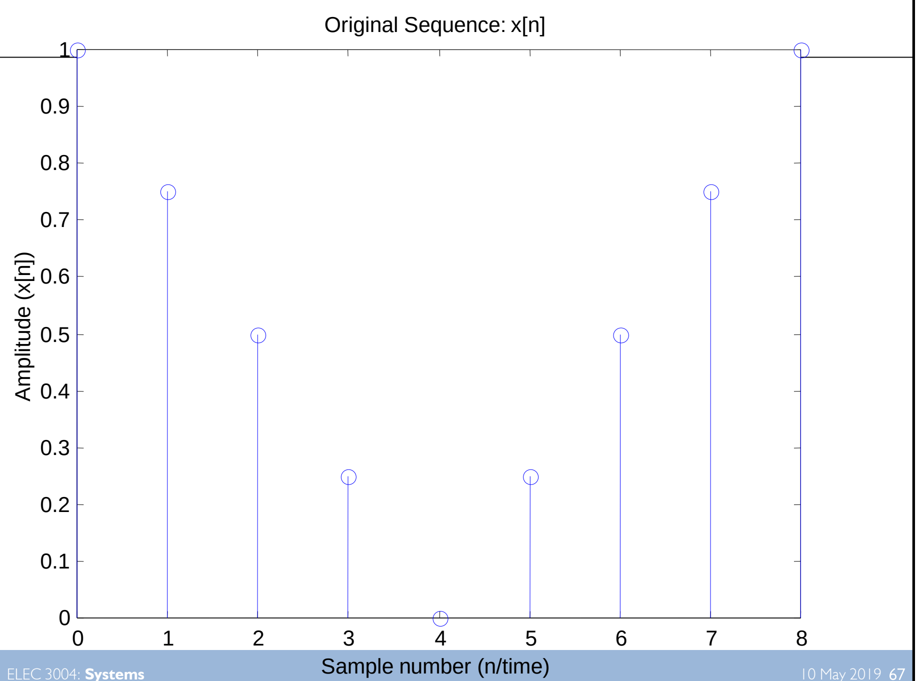
(Mini-section)

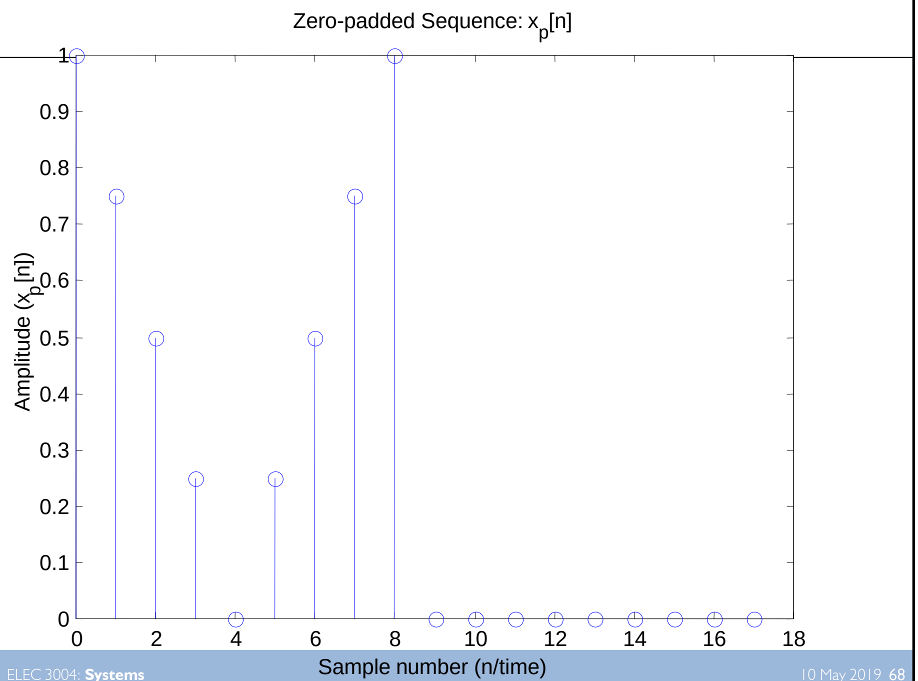
Inverse DFT

- One powerful property is the inverse fourier transform is simply the fourier transform with time-reversed basis functions
- So, IDFT and DFT obtained by
  - changing sign of $\omega_{N_{nk}}$
  - scaling by $\frac{1}{N}$

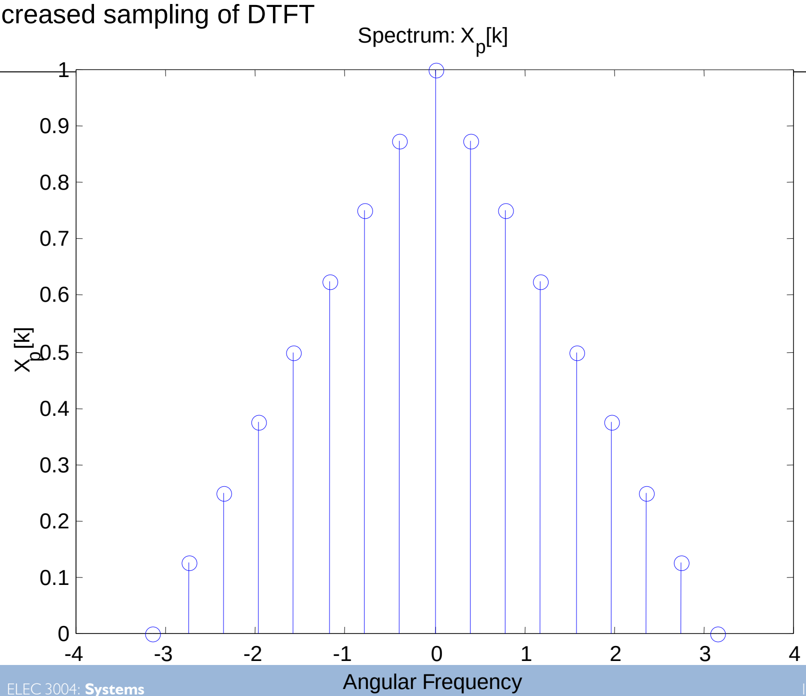
Interpolation using the DFT

- DFT samples the DTFT
  - Normally N samples in both time & Frequency
  - But we can increase the (DFT) sample density!
  - By zero padding
- Zero Pad in time domain
  - Calculates additional samples of DTFT
- Zero Pad in frequency domain
  - Adds additional high frequency components (zero)
  - **DFT zero padding ≡ sinc interpolation**
    - Windowed by length, N, of DFT (**not ideal sinc**)

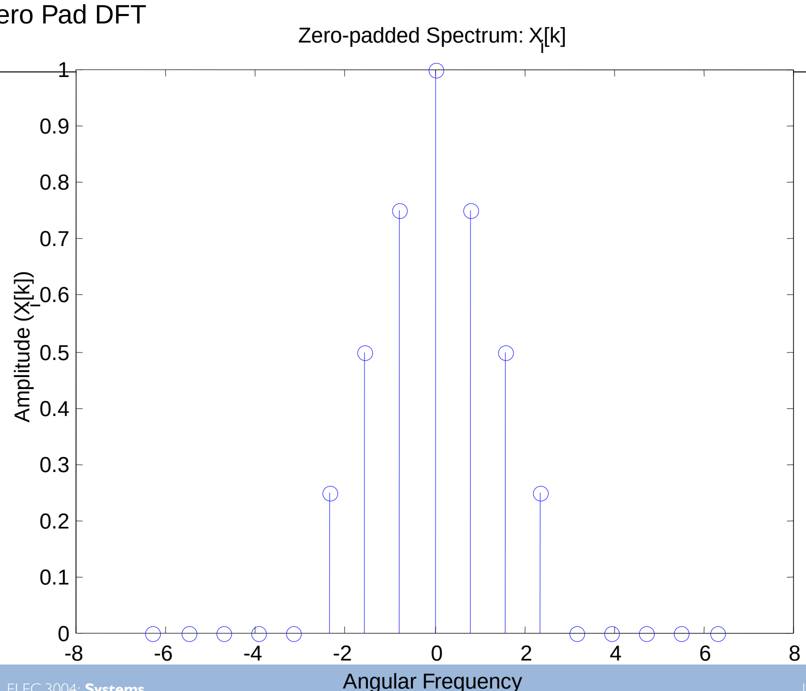

Original Sequence: x[n]
Amplitude (x[n])
Sample number (n/time)


Zero-padded Sequence: $x_p[n]$
Amplitude ($x_p[n]$)
Sample number (n/time)

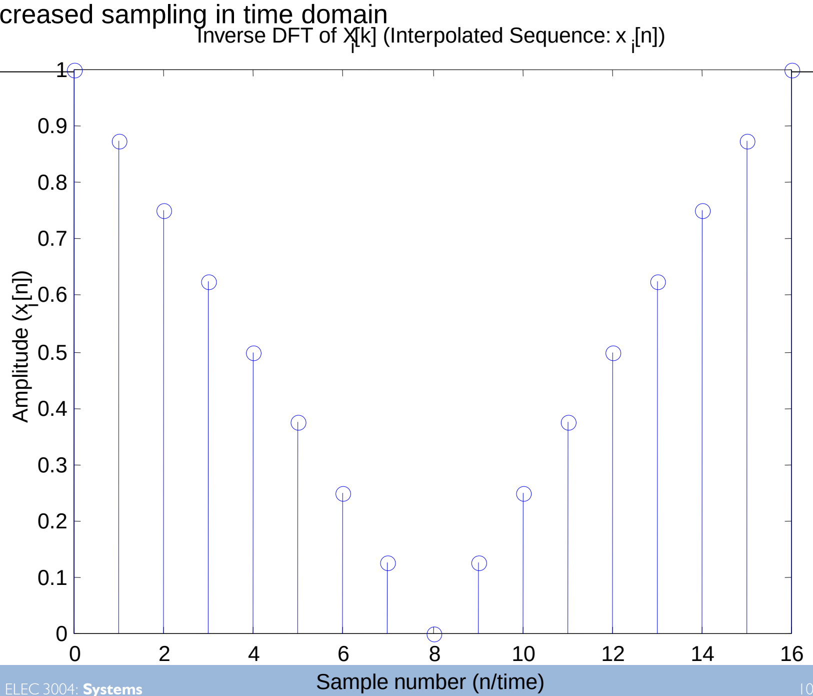
Increased sampling of DTFT



Zero Pad DFT



Increased sampling in time domain



Interpolation via DFT (FFT)

- Interpolation of X[k]
  - zero pad sequence x[n]
    - either start or end of x[n] (or both)
  - increased sampling of DTFT spectrum, X(w)
- Interpolation of x[n]
  - zero pad discrete spectrum X[k]
    - evenly, both at start or end of the sequence
    - to ensure xu[n] remains real
    - i.e., pad to preserve symmetry of X[k]

FFT

Example: 8-point DFT Matrix

$$
\begin{array}{c}
\\
\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ X(4) \\ X(5) \\ X(6) \\ X(7) \end{bmatrix}
\end{array}
=
\begin{array}{c}
\begin{array}{cccccccc} x(0) & & x(2) & & x(4) & & x(6) & \end{array} \\
\begin{bmatrix}
W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 \\
W_8^0 & W_8^1 & W_8^2 & W_8^3 & W_8^4 & W_8^5 & W_8^6 & W_8^7 \\
W_8^0 & W_8^2 & W_8^4 & W_8^6 & W_8^0 & W_8^2 & W_8^4 & W_8^6 \\
W_8^0 & W_8^3 & W_8^6 & W_8^1 & W_8^4 & W_8^7 & W_8^2 & W_8^5 \\
W_8^0 & W_8^4 & W_8^0 & W_8^4 & W_8^0 & W_8^4 & W_8^0 & W_8^4 \\
W_8^0 & W_8^5 & W_8^2 & W_8^7 & W_8^4 & W_8^1 & W_8^6 & W_8^3 \\
W_8^0 & W_8^6 & W_8^4 & W_8^2 & W_8^0 & W_8^6 & W_8^4 & W_8^2 \\
W_8^0 & W_8^7 & W_8^6 & W_8^5 & W_8^4 & W_8^3 & W_8^2 & W_8^1
\end{bmatrix}
\end{array}
\cdot
\begin{array}{c}
\text{Even samples} \\
\begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \\ x(4) \\ x(5) \\ x(6) \\ x(7) \end{bmatrix}
\end{array}
$$

Repeated complex multiplications in EVEN rows

Re-ordered DFT Matrix

Separate even and odd row operations (and re-order input vector)

$$
\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ X(4) \\ X(5) \\ X(6) \\ X(7) \end{bmatrix}
=
\begin{bmatrix}
W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 \\
W_8^0 & W_8^4 & W_8^2 & W_8^6 & W_8^1 & W_8^5 & W_8^3 & W_8^7 \\
W_8^0 & W_8^0 & W_8^4 & W_8^4 & W_8^2 & W_8^2 & W_8^6 & W_8^6 \\
W_8^0 & W_8^4 & W_8^6 & W_8^2 & W_8^3 & W_8^7 & W_8^1 & W_8^5 \\
W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^4 & W_8^4 & W_8^4 & W_8^4 \\
W_8^0 & W_8^4 & W_8^2 & W_8^6 & W_8^5 & W_8^1 & W_8^7 & W_8^3 \\
W_8^0 & W_8^0 & W_8^4 & W_8^4 & W_8^6 & W_8^6 & W_8^2 & W_8^2 \\
W_8^0 & W_8^4 & W_8^6 & W_8^2 & W_8^7 & W_8^3 & W_8^5 & W_8^1
\end{bmatrix}
\cdot
\begin{bmatrix} x(0) \\ x(4) \\ x(2) \\ x(6) \\ x(1) \\ x(5) \\ x(3) \\ x(7) \end{bmatrix}
$$

Even samples Odd samples

Phasor Rotational Symmetry

To highlight repeated computations on odd samples
as $W_8^4 = -W_8^0$, $W_8^5 = -W_8^1$, $W_8^6 = -W_8^2$, $W_8^7 = -W_8^3$

$$
\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ X(4) \\ X(5) \\ X(6) \\ X(7) \end{bmatrix}
=
\begin{bmatrix}
W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0 \\
W_8^0 & -W_8^0 & W_8^2 & -W_8^2 & W_8^1 & -W_8^1 & W_8^3 & -W_8^3 \\
W_8^0 & W_8^0 & -W_8^0 & -W_8^0 & W_8^2 & W_8^2 & -W_8^2 & -W_8^2 \\
W_8^0 & -W_8^0 & -W_8^2 & W_8^2 & W_8^3 & -W_8^3 & W_8^1 & -W_8^1 \\
W_8^0 & W_8^0 & W_8^0 & W_8^0 & -W_8^0 & -W_8^0 & -W_8^0 & -W_8^0 \\
W_8^0 & -W_8^0 & W_8^2 & -W_8^2 & -W_8^1 & W_8^1 & -W_8^3 & W_8^3 \\
W_8^0 & W_8^0 & -W_8^0 & -W_8^0 & -W_8^2 & -W_8^2 & W_8^2 & W_8^2 \\
W_8^0 & -W_8^0 & -W_8^2 & W_8^2 & -W_8^3 & W_8^3 & -W_8^1 & W_8^1
\end{bmatrix}
\cdot
\begin{bmatrix} x(0) \\ x(4) \\ x(2) \\ x(6) \\ x(1) \\ x(5) \\ x(3) \\ x(7) \end{bmatrix}
$$

Upper & lower left-hand quarters are identical
Right hand quarters identical except sign difference!

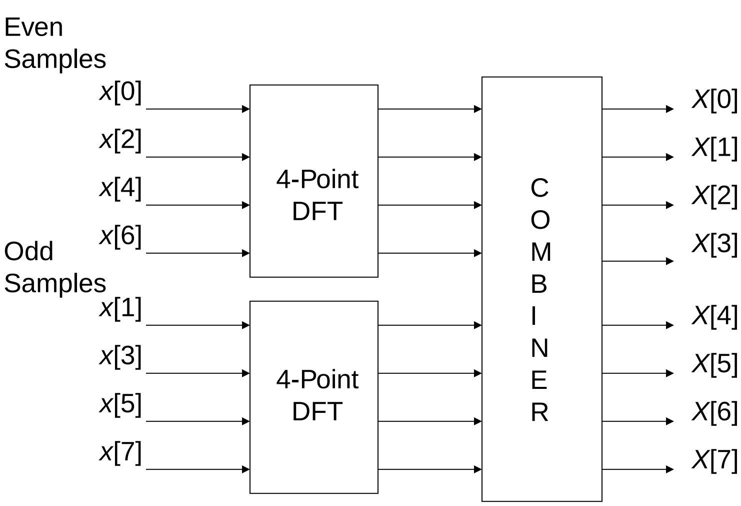
Adding “Twiddle Factors”

$$\begin{bmatrix}
W_8^0 & W_8^0 & W_8^0 & W_8^0 & W_8^0\times W_8^0 & W_8^0\times W_8^0 & W_8^0\times W_8^0 & W_8^0\times W_8^0\\
W_8^0 & -W_8^0 & W_8^2 & -W_8^2 & W_8^1\times W_8^0 & W_8^1\times -W_8^0 & W_8^1\times W_8^2 & W_8^1\times -W_8^2\\
W_8^0 & W_8^0 & -W_8^0 & -W_8^0 & W_8^2\times W_8^0 & W_8^2\times W_8^0 & W_8^2\times -W_8^0 & W_8^2\times -W_8^0\\
W_8^0 & -W_8^0 & -W_8^2 & W_8^2 & W_8^3\times W_8^0 & W_8^3\times -W_8^0 & W_8^3\times -W_8^2 & W_8^3\times W_8^2\\
W_8^0 & W_8^0 & W_8^0 & W_8^0 & -W_8^0\times W_8^0 & -W_8^0\times W_8^0 & -W_8^0\times W_8^0 & -W_8^0\times W_8^0\\
W_8^0 & -W_8^0 & W_8^2 & -W_8^2 & -W_8^1\times W_8^0 & -W_8^1\times -W_8^0 & -W_8^1\times W_8^2 & -W_8^1\times -W_8^2\\
W_8^0 & W_8^0 & -W_8^0 & -W_8^0 & -W_8^2\times W_8^0 & -W_8^2\times W_8^0 & -W_8^2\times -W_8^0 & -W_8^2\times -W_8^0\\
W_8^0 & -W_8^0 & -W_8^2 & W_8^2 & -W_8^3\times W_8^0 & -W_8^3\times -W_8^0 & -W_8^3\times -W_8^2 & -W_8^3\times W_8^2
\end{bmatrix}$$

i.e., 8-point DFT reduced to two 4-point DFT’s only need calculate upper left and right quarters

Twiddle Factors make the left and right hand quarters identical

8-Point DFT as Two 4-Point DFTs



Combiner adds twiddle factors to data

Radix-2 FFT

Each 4-point DFT can be reduced to two 2-point DFT's

$$\begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & -W^0 & W^2 & -W^2 \\ W^0 & W^0 & -W^0 & -W^0 \\ W^0 & -W^0 & -W^2 & W^2 \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 \times W^0 & W^0 \times W^0 \\ W^0 & -W^0 & W^2 \times W^0 & W^2 \times -W^0 \\ W^0 & W^0 & -W^0 \times W^0 & -W^0 \times W^0 \\ W^0 & -W^0 & -W^2 \times W^0 & -W^2 \times -W^0 \end{bmatrix}$$

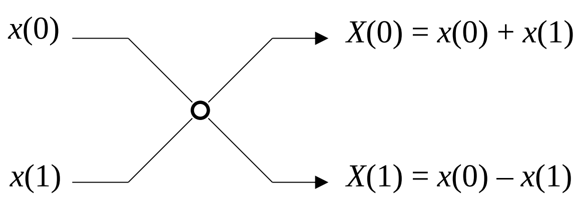
2x2 Quadrants are identical (with twiddle factors)

Two-point "Butterfly" operation

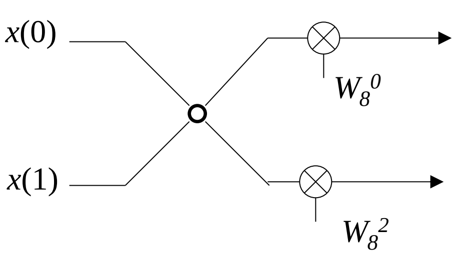
$$\begin{bmatrix} X(0) \\ X(1) \end{bmatrix} = \begin{bmatrix} W^0 & W^0 \\ W^0 & -W^0 \end{bmatrix} \cdot \begin{bmatrix} x(0) \\ x(1) \end{bmatrix}$$

$$\begin{bmatrix} X(0) \\ X(1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} x(0) \\ x(1) \end{bmatrix}$$

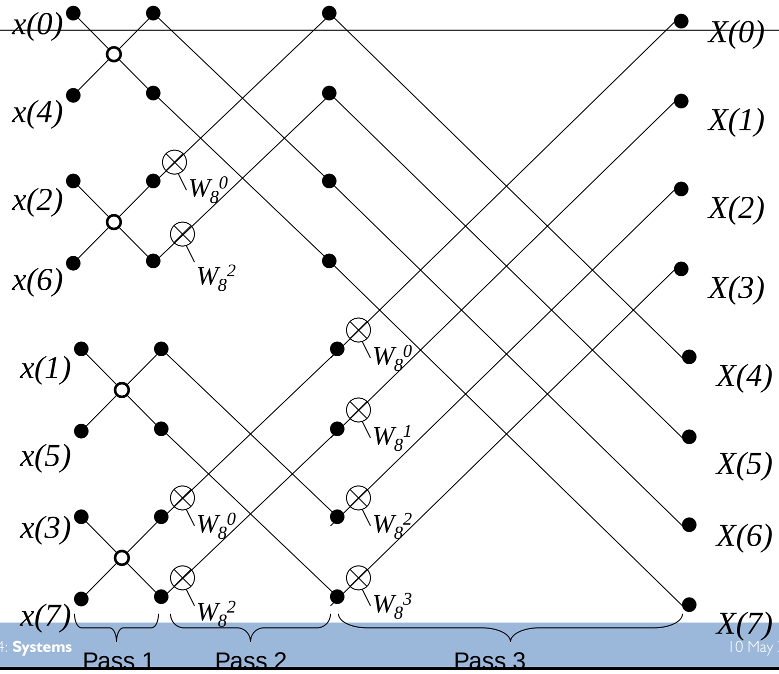
Two Point Butterfly



With twiddle factors:

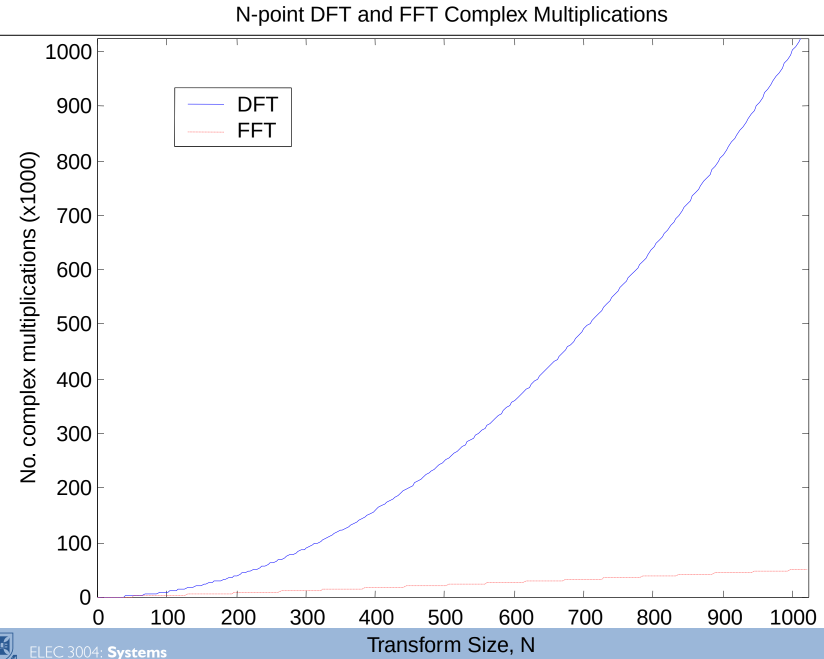


8-point radix-2 DIT FFT flowgraph:



Features of the FFT

- Reduce complex multiplications from N^2 to:
  - $\left(\frac{N}{2}\right)\log_2(N)$
  - As there are $\log_2(N)$ passes
  - Each pass requires $\frac{N}{2}$ complex multiplications
- Disadvantages
  - More complex memory addressing
    - To get appropriate samples pairs for each butterfly
  - FFT can be slower (than DFT) for small N (< 16)
- What about the IFFT? We can use same FFT algorithm
  - change sign of twiddle factors
  - and scale output to get $x[n]$



What does it let us do

- For high performance applications, the FFT can be the difference between feasible and infeasible
- EG – 4K video:
  - There are ~8M pixels
  - $N^2 = 6.4 \cdot 10^{13}$ $N \cdot Log_2(N) = 1.6 \cdot 10^8$
  - You need to do this 30x per second
  - So the flops is on the order of $2 \cdot 10^{15}$ = 2 PFlops using the DFT vs $4 \cdot 10^9$=4 Gflops using the FFT
- An Nvidia V100 maxes out at 120TFlops, a standard CPU is about 100GFlops

Alternative FFT Algorithms

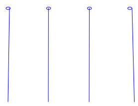
- Only case covered so far is
  - (one case of) radix-2 decimation in time (DIT) FFT
  - requires sequence length, N, to be a power of 2
  - achieved by ‘zero padding’ sequence to desired, N
- Decimation in Frequency
  - similar to DIT, twiddle factors on outputs
- Alternatives to radix-2 decomposition
  - Radix 3: for sequence length, N = power of 3
  - Radix 4: twice as fast as radix 2 FFT
    - half number of passes, log4(N)
  - Split radix: mixtures of the above

Applications of the FFT

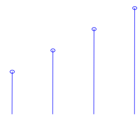
- Fast (circular) Convolution
  - Convolution requires N^2 MAC operations ☹
  - more efficient alternative via the FFT ☺
    - Take FFT of both sequences
    - Multiply them together (point-wise)
    - Take IFFT to get the result
    ["Hello FFT-W! *Bonjour cuFFT!*"]
  - Zeropad and you have linear convolution
- Spectral Analysis
  - Estimate (power) spectrum with less computations
  - i.e., what frequencies in our signal are carrying power (i.e., carrying information) ?
- Fast Cross-correlation
  - E.g., correlation detector in digital comm’s

(Linear) Convolution

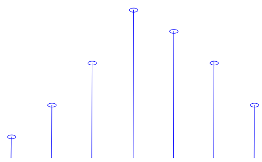
$h[n] = \{1\ 1\ 1\ 1\}$



$x[n] = \{0.5\ 0.75\ 1.0\ 1.25\}$



$y[n] = x[n] * h[n] = \{0.5\ 1.25\ \ 2.25\ \ 3.5\ \ 3.0\ \ 2.25\ \ 1.25\}$



In general: length($y[n]$) = length($x[n]$) + length($h[n]$) - 1

Circular Convolution

Given $X[k] = \mathrm{DFT}\{x[n]\}$ and $H[k] = \mathrm{DFT}\{h[n]\}$

from convolution theorem we know
$\mathrm{IDFT}\{X[k] \cdot H[k]\} \equiv x[n] * h[n]$

$\mathrm{IDFT}\{X[k] \cdot H[k]\} = \{3.5\ 3.5\ 3.5\ 3.5\}$ $\leftarrow$ Wrong Length!

Solution: zero pad both sequences to required length

$h_p[n] = \{1\ 1\ 1\ 1\ 0\ 0\ 0\}$ $x_p[n] = \{0.5\ 0.75\ 1.0\ 1.25\ 0\ 0\ 0\}$

$\mathrm{IDFT}\{X_p[k] \cdot H_p[k]\} = [0.5\ 1.25\ \ 2.25\ \ 3.5\ \ 3.0\ \ 2.25\ \ 1.25]$

i.e., $x[n]$ and $h[n]$
are periodic in time

Spectral Analysis

- Power Spectral Density (PSD) defined as
  – Fourier Transform of Autocorrelation function

$$S_{xx}(w) = \sum_{m=-\infty}^{\infty} \varphi_{xx}(m) \exp(-jwm\Delta t)$$

  - In practice, we estimate $S_{xx}(w)$ from $\{x[n]\}_0^{N-1}$
    - i.e., a finite length of sampled data
  - This can be done using N - point DFT
    - and implemented using the FFT algorithm

Spectral Analysis

- Estimate of PSD is given by

$$\widehat{S}_{xx}[k] = \frac{1}{N}\left|\sum_{n=0}^{N-1} x[n] \exp\left(\frac{-jnk2\pi}{N}\right)\right|^2$$

- This is known as a **periodogram**
  – DFT effectively implements narrow-band filter bank
  – calculate power (i.e., square) at each frequency k

- Again, window functions often required
  – to improve PSD estimate
  – e.g., Hanning, ~~Hamming~~, Bartlet etc

Spectral Analysis

- Alternatively, we can estimate PSD as
  - DFT (FFT) of the estimate of the autocorrelation

$$\hat{S}_{xx}[k] = \sum_{m=-M}^{M} \hat{\varphi}_{xx}[m] \exp\left(\frac{-jmk2\pi}{2M+1}\right)$$

Where: $$\hat{\varphi}_{xx}[m] = \frac{1}{N}\sum_{n=0}^{N-1} x[n]x[n+m]$$

- Assuming $x[n]$ is ergodic (at least stationary)

- Normally restricted range of PSD
  - e.g., $0 < M < \frac{N}{10}$

Spectral Analysis

- When finding PSD as DFT of $\phi^{xx[m]}$:
  - $\phi^{xx[m]}$ has an odd length! $(2M + 1)$

- Therefore, to use the radix-2 FFT we need to
  - zero pad $\phi^{xx[m]}$ to length = power of 2

- e.g., for $M = 2$, $\phi^{xx[m]}$ is of length 5
  - we need to zero pad to length 8, i.e.,
  - $\{\phi^{xx[-2]}\ \phi^{xx[-1]}\ \phi^{xx[0]}\ \phi^{xx[1]}\ \phi^{xx[2]}\ 0\ 0\ 0\}$
  - Note, sequence made causal (no change to PSD)

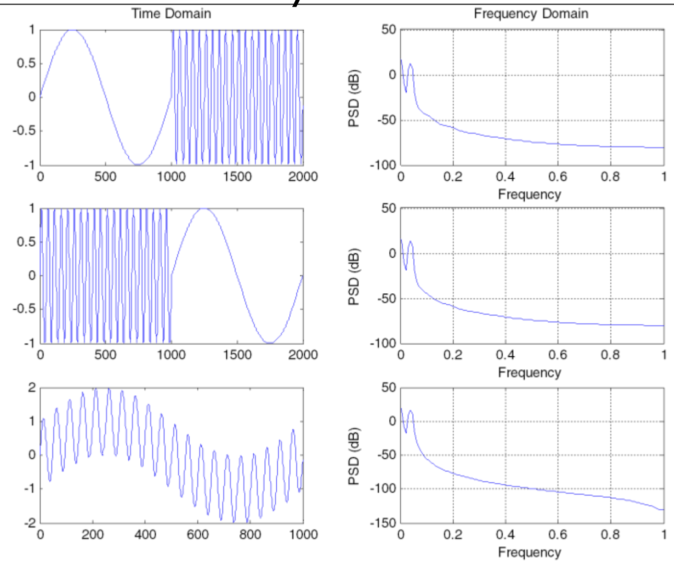
- This estimate of PSD is known as correlogram
  - Note, periodogram is most common estimate of PSD

Limitations of Fourier Analysis

Low – high

High – low

Mixed



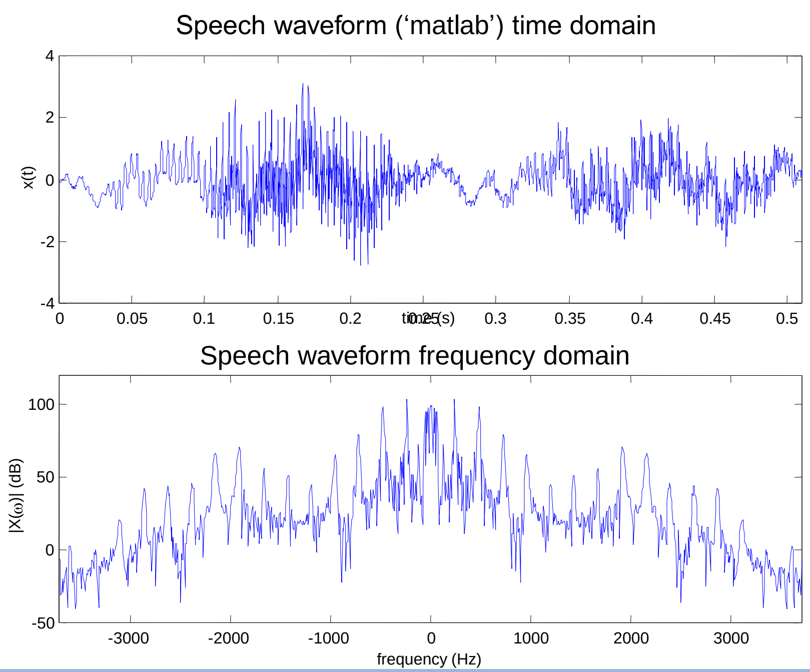
Note: These signals differ in Phase. PSD is zero phase as $F\{\phi_{xx}(k)\}$ real & even

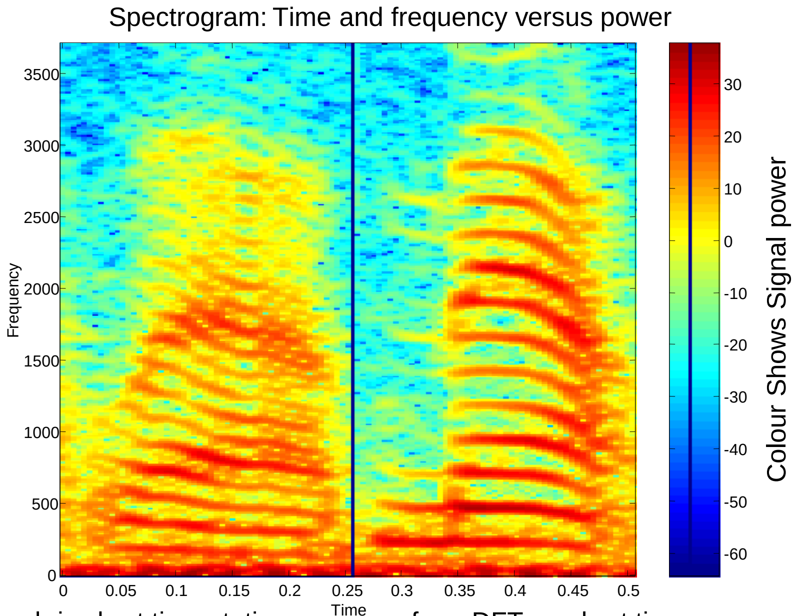
Time Frequency

(Mini-section)

Spectrum Analysis of Non-Stationary Signals

- Spectrum of non-deterministic Signal $X(w)$
  - is only valid if $x(t)$ is stationary
  - i.e., statistics of $x(t)$ do not change over time
- Real-world signals often only stationary over a short time period of time
  - e.g., speech: assumed stationary over t < 60ms
- Therefore, take 'short-time' DFT of signal
  - i.e., take multiple DFT's over stationary periods
  - plot how frequency components change over time
  - for speech the plot of time V frequency V power
    - is called a **Spectrogram**





Speech is short time stationary, so perform DFT on short time sequences

Next Time…

- **Estimation! (Kalman Filters!)**
- **Digital Control!**
- Review:
  - Chapter 12 of Lathi
  - FPE Chapter 1 and 2
- Ponder? $y[k] = f[k] * h[k]$ $\quad$ $Y(\Omega) = F(\Omega)H(\Omega)$

  where $F(\Omega)$, $Y(\Omega)$, and $H(\Omega)$ are DTFTs of $f[k]$, $y[k]$, and $h[k]$, respectively; that is,

  $$f[k] \Longleftrightarrow F(\Omega), \quad y[k] \Longleftrightarrow Y(\Omega), \quad \text{and} \quad h[k] \Longleftrightarrow H(\Omega)$$

Summary

- FT of sampled data is known as
  - discrete-time Fourier transform (DTFT)
  - discrete in time
  - continuous & periodic in frequency
- DFT is sampled version of DTFT
  - discrete in both time and frequency
  - periodic in both time and frequency
    - due to sampling in both time and frequency
- DFT is implemented using the FFT
- Leakage reduced (dynamic range increased)
  - with non-rectangular window functions

Summary

- FFT exploits symmetries in the DFT
  - Successively splits DFT in half
    - odd and even samples
  - Reduction to elementary butterfly operation
    - with 'twiddle factors'
  - Reduce computations from N^2 to $\left(\frac{N}{2}\right)\log_2(N)$ ☺
- FFT can be used to implement DFT for
  - PSD estimates (periodogram and correlogram)
  - Circular (fast) convolution (and correlation)
    - Requires zero padding to obtain "correct" answer