HKN ECE 310 Final Review Session

Whomst: Arjun, Eric, Bex

Logistics + Shameless Plug

- Slides and Recording will be uploaded here: Student Services (illinois.edu)
  hkn.illinois.edu/services

SLIDES

Services:

YouTube channel (NEW!): HKN has a YouTube channel with videos for difficult or cool concepts from undergraduate courses! Subscribe and leave a like to keep up to date on our content. Link to channel **here**.

One-on-one tutoring: HKN offers one-on-one tutoring for all core ECE classes. A list of tutors with the classes they specialize in can be found here: **Tutors List**

Office Hours: A newer service HKN offers is Office Hours! We have Office Hours for all the core ECE classes, and more. Our Office Hours hosts have a variety of specializations from Quantum Computing to Destiny 2. The Office Hours Schedule can be found here: **Office Hours Schedule**

Pre-exam Panic Pacifiers: HKN will have students available to answer your questions right before (~an hour before) your exam for all core 100, 200, and 300-level ECE classes for both EE and CompE! Come with any last-minute questions or get some last-minute tips and tricks. The schedule, along with the schedule for all review sessions this semester, can be found here: **Review Session/Pre-exam Panic Pacifier Schedule**

Topics

- Circular Convolution
- Upsampling & Downsampling
- Windowing
- FFT
- Filter Design
- Generalized/linear phase
- Practical A/D and D/A conversion

Circular Convolution Exercise 1

Compute the linear AND circular convolution $y[n] = x[n] \circledast_4 h[n]$ where

$\{x[n]\}^3_{n=0} = \{ 2 , 4, 6, 8 \}$ & $h[n] = \{ -1, -1, 1 \}$

Table Method: h[n] rows and x[n] columns → y[n] = {-4, 2, -8, -10}

ℓ =	**0**	**1**	**2**	**3**	**y[n]**
$x[\ell]$ =	**2**	**4**	**6**	**8**	
$h(\langle 0 - \ell \rangle_4)$	-1	0	1	-1	(-1)*2+6-8 = -4
$h(\langle 1 - \ell \rangle_4)$	-1	-1	0	1	-2-4+8 = 2
$h(\langle 2 - \ell \rangle_4)$	1	-1	-1	0	2-4-6 = -8
$h(\langle 3 - \ell \rangle_4)$	0	1	-1	-1	4-6-8 = -10

Linear result: y[n] = {2, 2, 0, -2, -14, -8}

Upsampling

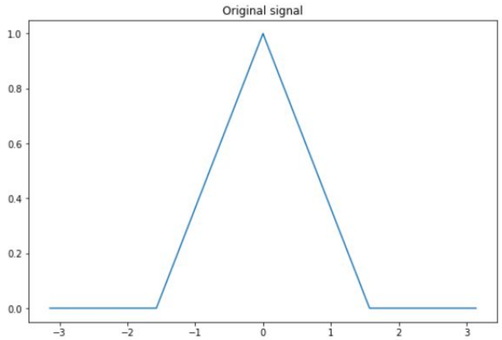
- In the DFT, we are adding zeros between every value we already have
  - If we upsample by U, we want there to be U times as many signal entries, so we add U-1 0s in between
- Example
  - Signal: [0, 1, 2, 3]
  - Upsampled by 3: [0, 0, 0, 1, 0, 0, 2, 0, 0, 3, 0, 0]
- DTFT Effects:

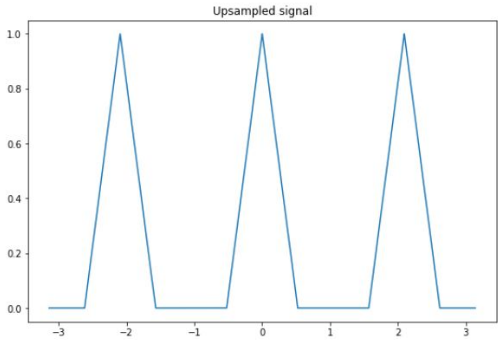
$$y[n] = \begin{cases} x\left[\frac{n}{U}\right], & n = \pm U, 2U, \ldots, kU \\ 0, & \text{else} \end{cases}$$

$$Y_d(\omega) = \sum_{n=-\infty}^{\infty} y[n]e^{-j\omega n}$$

$$= \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n U}$$

$$Y_d(\omega) = X_d(U\omega).$$

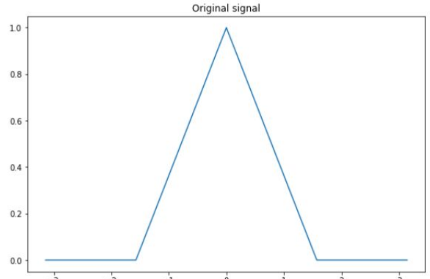


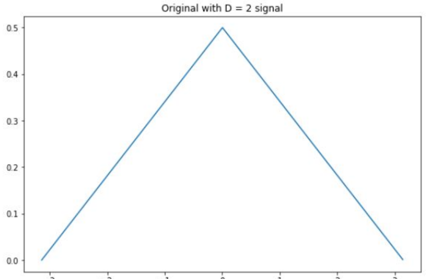


Downsampling

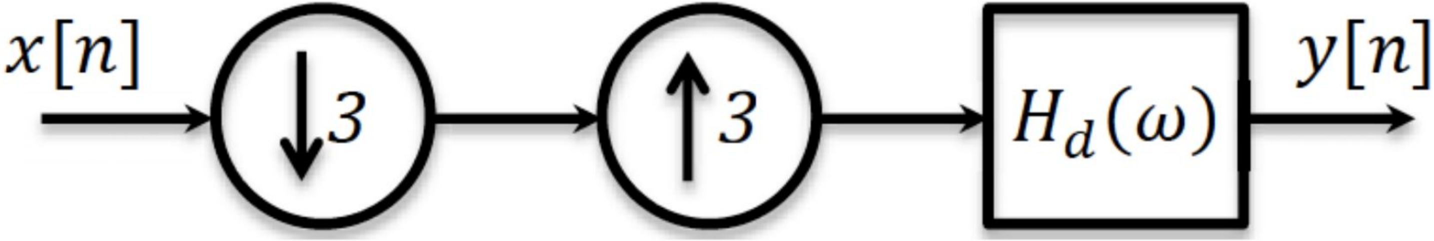
- The opposite of upsampling (wild)
  - $x_d[n] = x[Dn]$with downsampling factor D
  - Effectively, keep only every D values from input signal
- Example
  - Signal: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
  - Downsampled by 3: [0, 3, 6, 9]
- DTFT Effects:
  - Expands DTFT (susceptible to aliasing !!!)
    - Solution to aliasing: low pass filter first

$$Y_d(\omega) = \frac{1}{D}\sum_{k=0}^{D-1} X_d\left(\frac{\omega - 2\pi k}{D}\right).$$





Upsampling and Downsampling in DFT Exercise

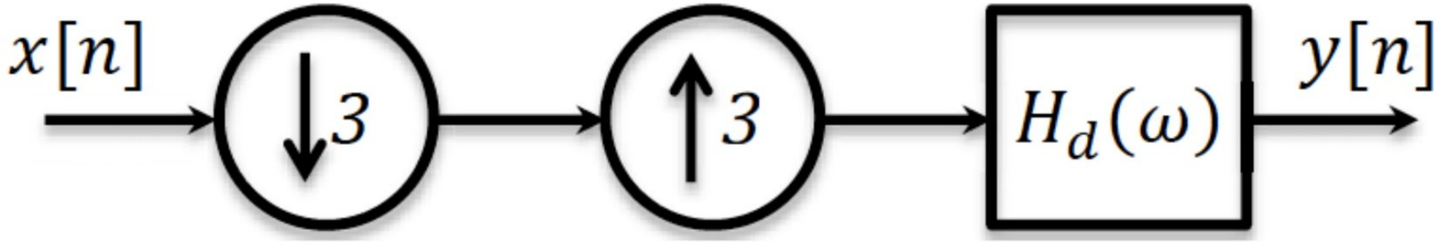


$H_d(w)$ is ideal LPF with cutoff $2\pi/5$

A. $x[n] = cos(0.2\pi n)$ y[n] = ?

B. $x[n] = cos(0.6\pi n)$ y[n] = ?

Upsampling and Downsampling in DFT Exercise



A. $y[n] = \frac{1}{3}\cos(0.2\pi n) = \frac{1}{3}x[n]$

B. $y[n] = \frac{1}{3}\cos(\frac{\pi}{15}n)$

Upsampling and Downsampling in DTFT Exercise

$$X_d(\omega) = \begin{cases} 1 - \frac{3|\omega|}{\pi}, & |\omega| \leq \frac{\pi}{3} \\ 0, & \frac{\pi}{3} < |\omega| \leq \pi \end{cases}$$

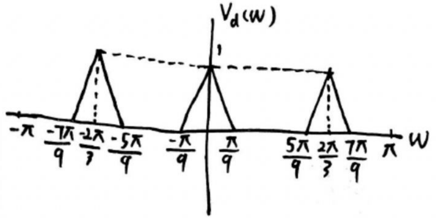
Given $V_d(w)$ is $X_d(w)$ upsampled by L, sketch $V_d(w)$ if L=3

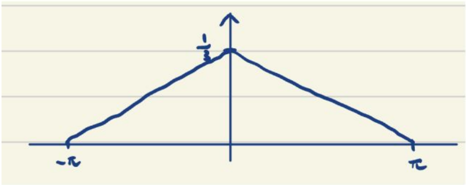
Given $W_d(w)$ is $X_d(w)$ downsampled by D, sketch $V_d(w)$ if D=3

Upsampling and Downsampling in DTFT Exercise

$$X_d(\omega) = \begin{cases} 1 - \frac{3|\omega|}{\pi}, & |\omega| \le \frac{\pi}{3} \\ 0, & \frac{\pi}{3} < |\omega| \le \pi \end{cases}$$

Given $V_d(w)$ is $X_d(w)$ upsampled by L, sketch $V_d(w)$ if L=3





Given $W_d(w)$ is $X_d(w)$ downsampled by D, sketch $V_d(w)$ if D=3

Windowing

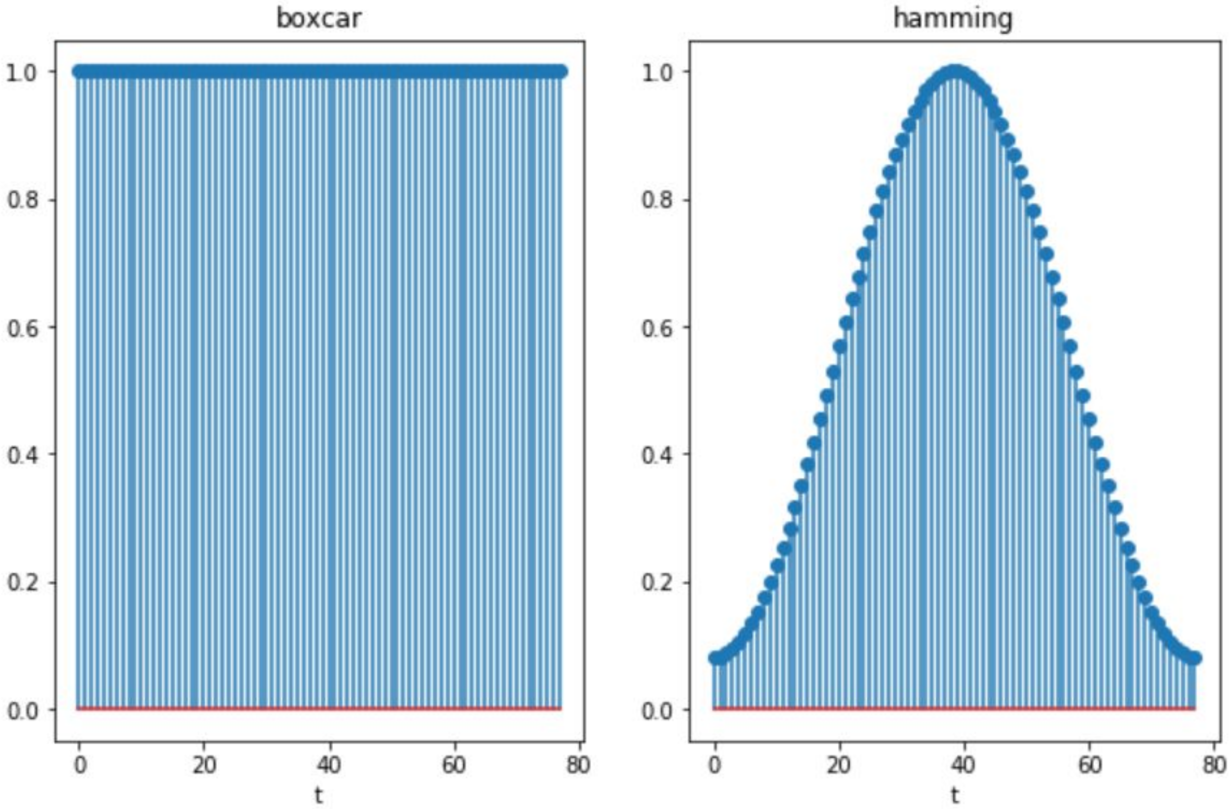
- Recall that the DFT implies infinite periodic extension of our signal.
- This extension can lead to artifacts known as "spectral leakage"
- Window functions help with these artifacts
  - Rectangular window
  - Hamming window
  - Hanning window
  - Kaiser window
- Windowing is just multiplication in the time domain

$$x_w = x[n]w[n]$$

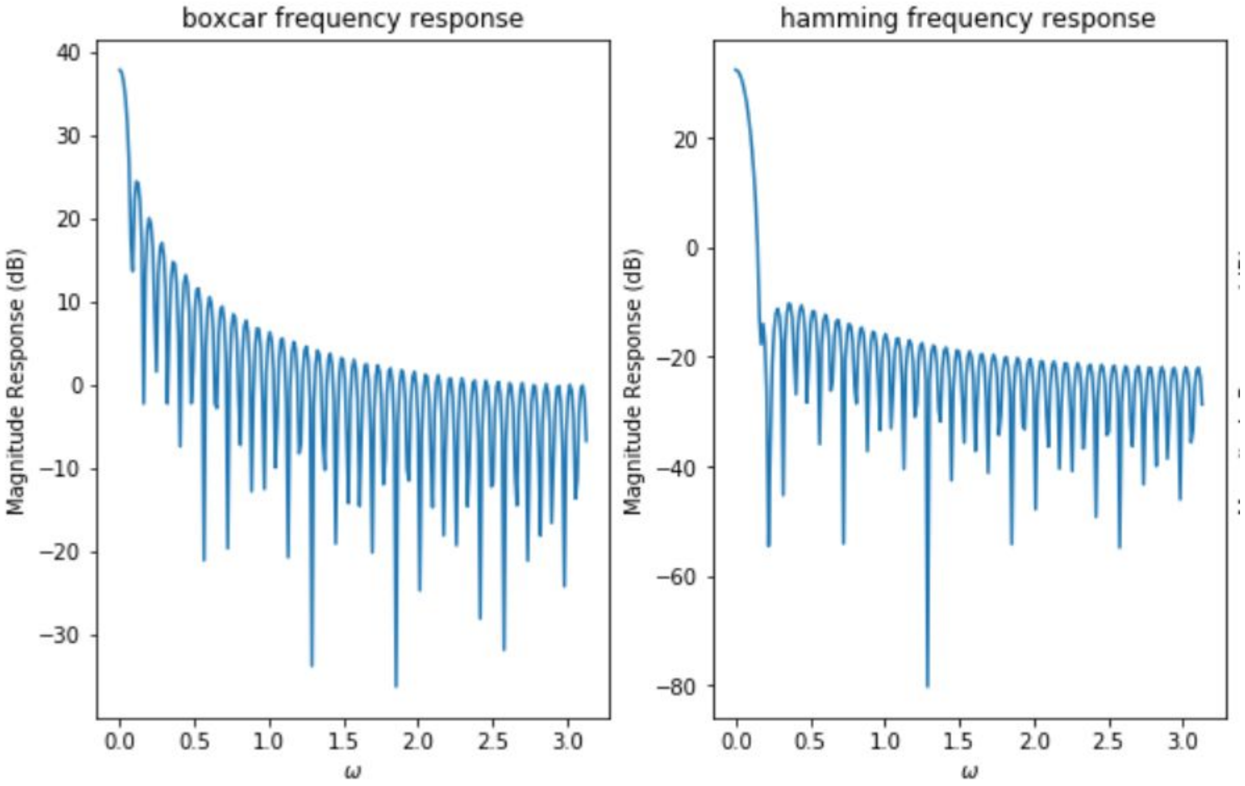
- We care about the main lobe width and side lobe attenuation of these windows.
  - In particular, know the tradeoffs between the rectangular and Hamming windows

    Hamming has more side lobe attenuation but wider main lobe

Windowing



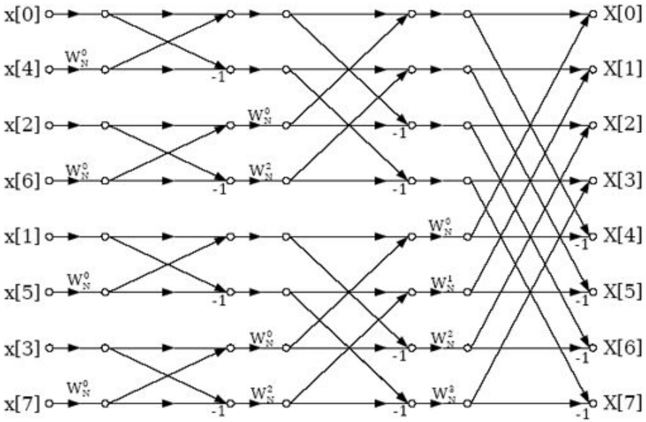
Windowing



boxcar frequency response
Magnitude Response (dB)
40
30
20
10
0
−10
−20
−30
0.0
0.5
1.0
1.5
2.0
2.5
3.0
ω
hamming frequency response
Magnitude Response (dB)
20
0
−20
−40
−60
−80
0.0
0.5
1.0
1.5
2.0
2.5
3.0
ω

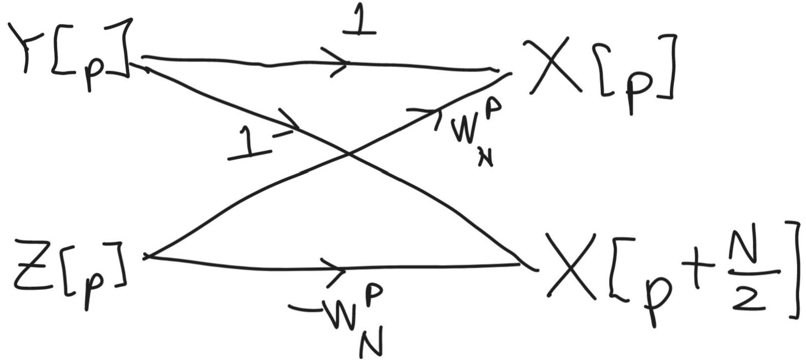
Fast Fourier Transform

- Class of “divide-and-conquer” algorithms to compute the DFT efficiently.
- Naïve DFT takes $O(n^2)$ operations, FFT takes $O(n \log n)$.
- Decimation in Time vs. Decimation in Frequency
- Butterfly diagrams

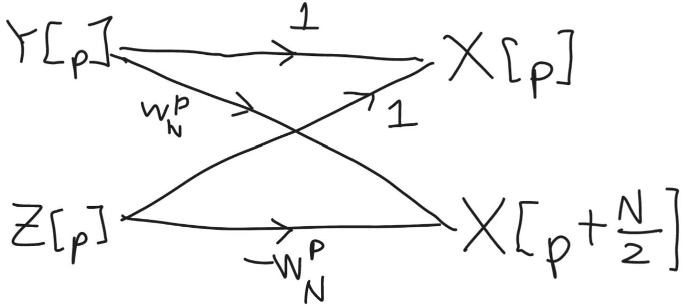


FFT Butterflies

$$W_N = e^{-j\frac{2\pi}{N}}$$



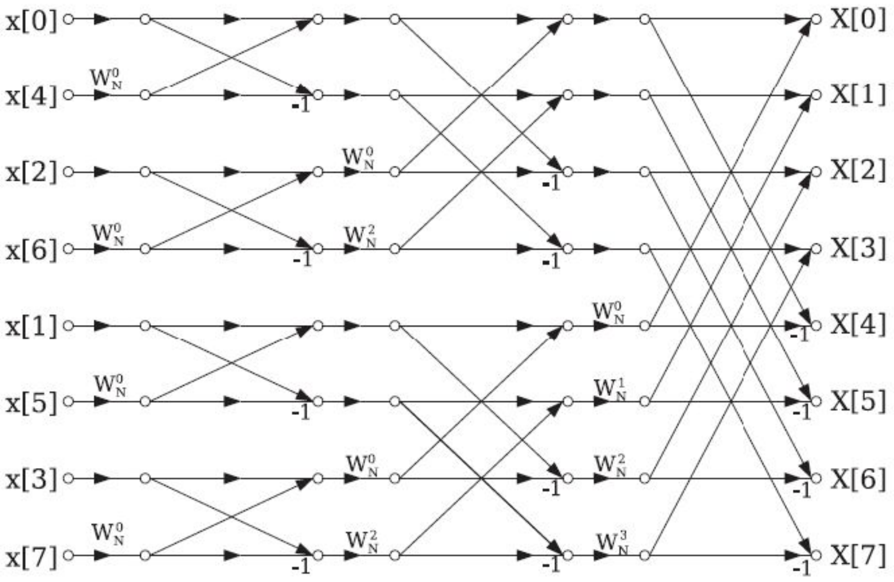
Decimation in Time



Decimation in Frequency

Slightly different!

FFT Butterflies



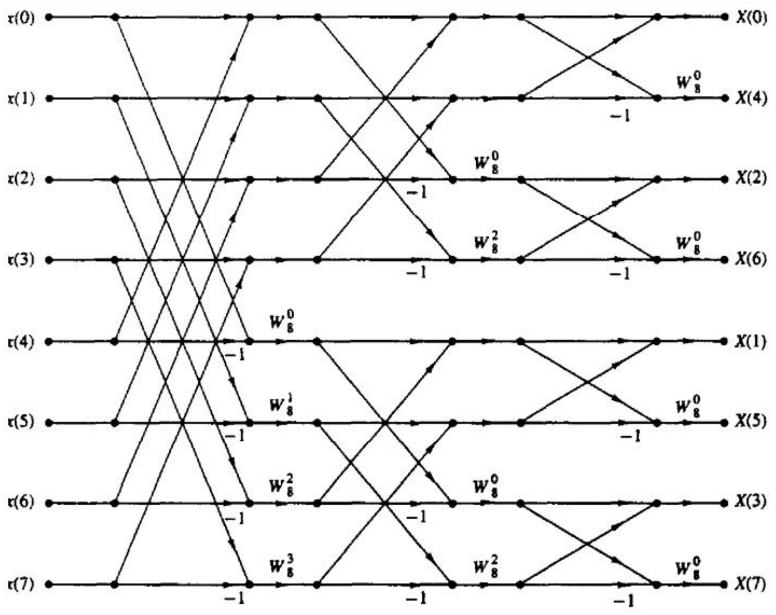


Figure 6.11 $N = 8$-point decimation-in-frequency FFT algorithmn.

Decimation in Time
The time points are “decimated” to do the first FFTs.

Decimation in Frequency
The frequency points are “decimated” at the end.

Fast Linear Convolution via FFT

- Convolution in the time domain requires $O(n^2)$ operations.
- By convolution theorem, perhaps we can do better in the frequency domain?
- Don't forget multiplication in DFT domain is *circular* convolution in time.
- To avoid aliasing, we adopt the following procedure
- Given signal x and filter h of lengths N and L, respectively:

1. Zero-pad x and h to length $N + L - 1$
2. Take their FFTs
3. Multiply in frequency domain
4. Take the inverse FFT

This procedure takes $O(n \log n)$ operations.

FFT Practice 1

Suppose you want to convolve two real-number signals of length 7000 and length 1100.

How many real-number multiplications does this take using classic convolution?

How many real-number multiplications does this take using using FFT?

Classic Convolution

Partial overlap at the beginning: (1 + 2 + ... + 1099) = (1099)(1100)/2

Partial overlap at the end: (1099 + 1098 + ... + 2 + 1) = (1099)(1100)/2

Full overlap in the middle: (1100)(7000 - 1100 + 1)

Total: 7,700,000 multiplications.

FFT

Zero-pad both signals to length 8192. (0)

2x FFT.

- Multiplication by 1 is free.
- Each left-side of the butterfly is multiplied by one complex number.
- This is done log_2 times, one for each butterfly phase.
- Total: (**2** * 8192 * log_2(8192) * 4)

Multiply the signals together (8192 * 4)

1x IFFT: (1 * 8192 * log_2(8192) * 4)

Total: 1,310,720 multiplications.

Some Hard Numbers...

Convolving length 7000 with length 1100:

- Classic convolution: 7,700,000 multiply operations

- FFT: 1,310,720 multiply operations

That’s 83% faster.

So what :|

- A modern-day computer can do around 1e9 operations per second.

- 7,700,000 operations = 7.7 milliseconds.
- 1,310,720 operations = 1.3 milliseconds.

- Ok that doesn’t really do anything.

Fine. Some Hard(er) Numbers...

The average MP3 file seems to be about 4MB.

Convolving length 4e6 with length 4e6 (apparently to add reverb or filter or something):

- Classic convolution: 1.6e13 multiply operations
  - About 4.5 hours.
- FFT: 1.12e9 multiply operations
  - About a second.

Ok that's pretty good.

FFT: The Most Important Algorithm of the Modern World

- Fast large-integer and polynomial multiplication
- Solve partial differential equations
- Filtering algorithms
- Digital recording, sampling, and pitch correction
- JPEG and MP3
- Used in 5G, LTE, Wi-Fi, and other communication systems

These are ALL made viable by FFT. Without it, these would be so slow...

Trivia Time!

Who was the first to come up with the FFT?

Answer: Gauss. Yes, that Gauss. And in 1805.

Yes. This predates Fourier.

And he didn't even publish it :(

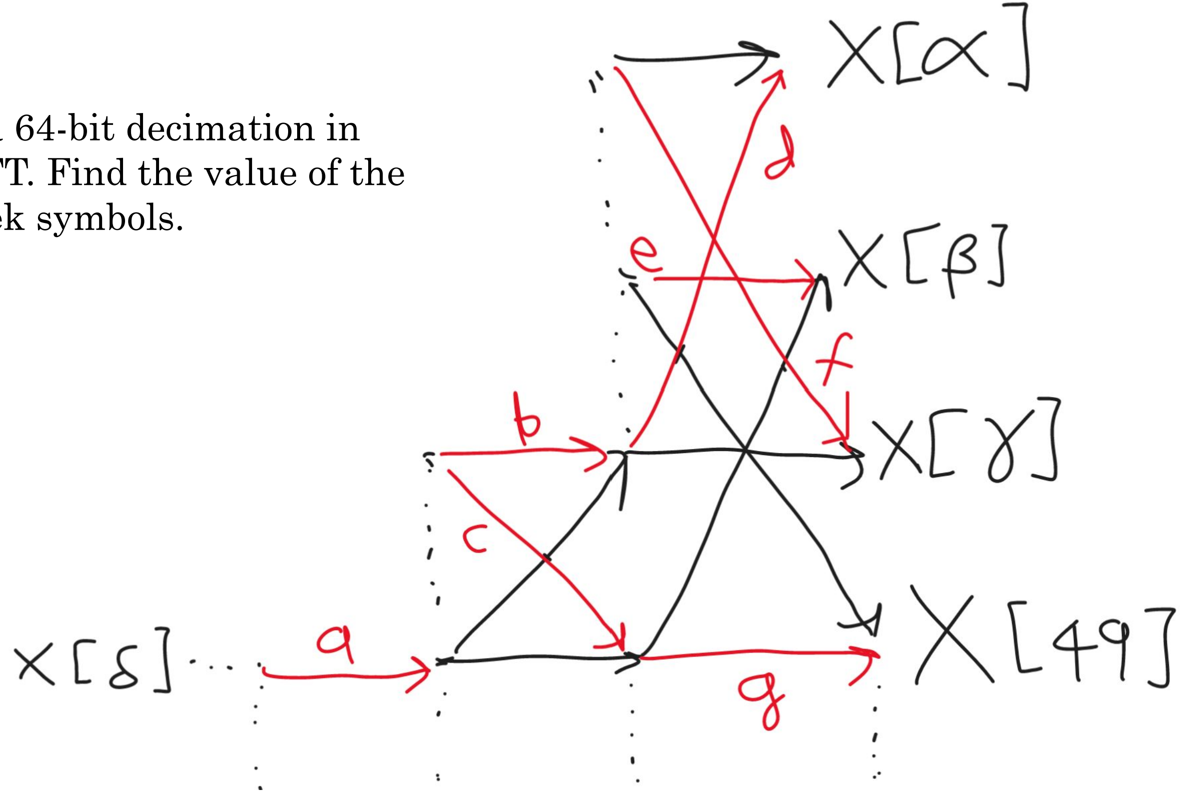
FFT Practice

What is the first step when drawing the entire FFT diagram of the following signal?

$$x[n] = \{1, 2, 3\}$$

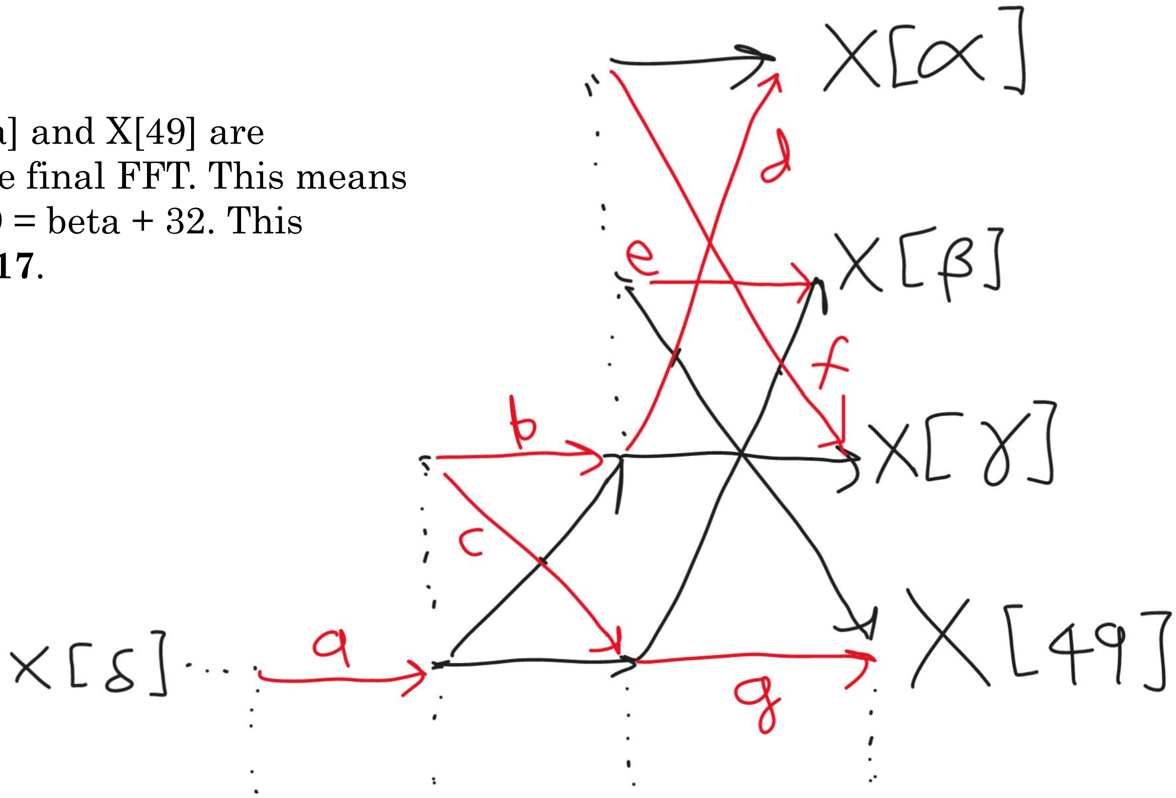
FFT Practice

This is part of a 64-bit decimation in time radix-2 FFT. Find the value of the letters and greek symbols.



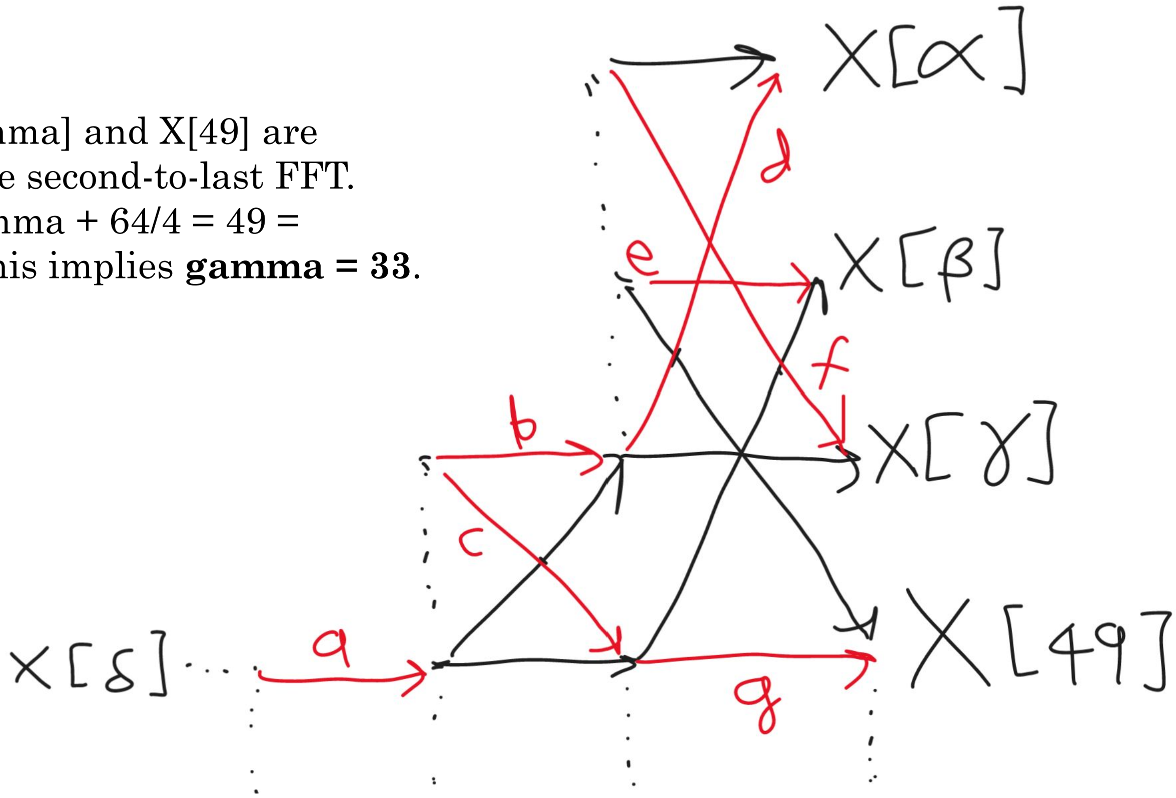
FFT Practice

We know X[beta] and X[49] are connected by the final FFT. This means beta + 64/2 = 49 = beta + 32. This implies **beta = 17**.



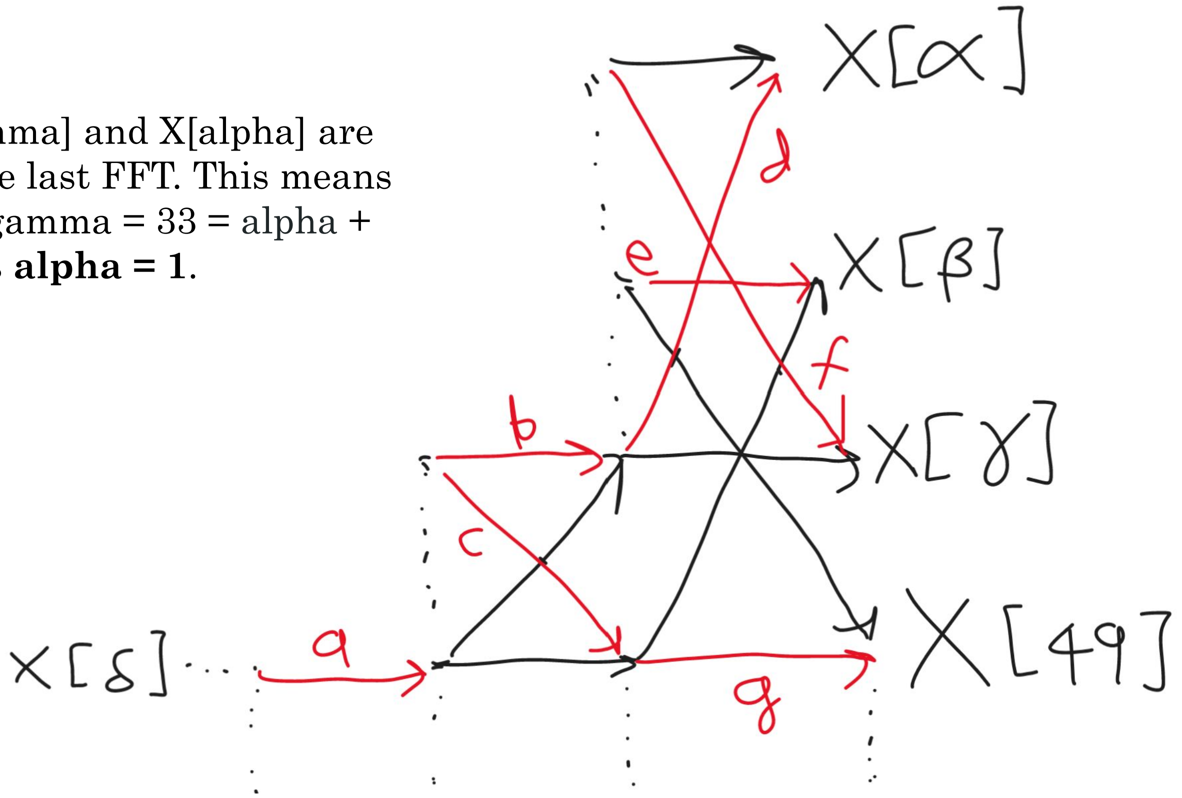
FFT Practice

We know X[gamma] and X[49] are connected by the second-to-last FFT. This means gamma + 64/4 = 49 = gamma + 16. This implies **gamma = 33**.



FFT Practice

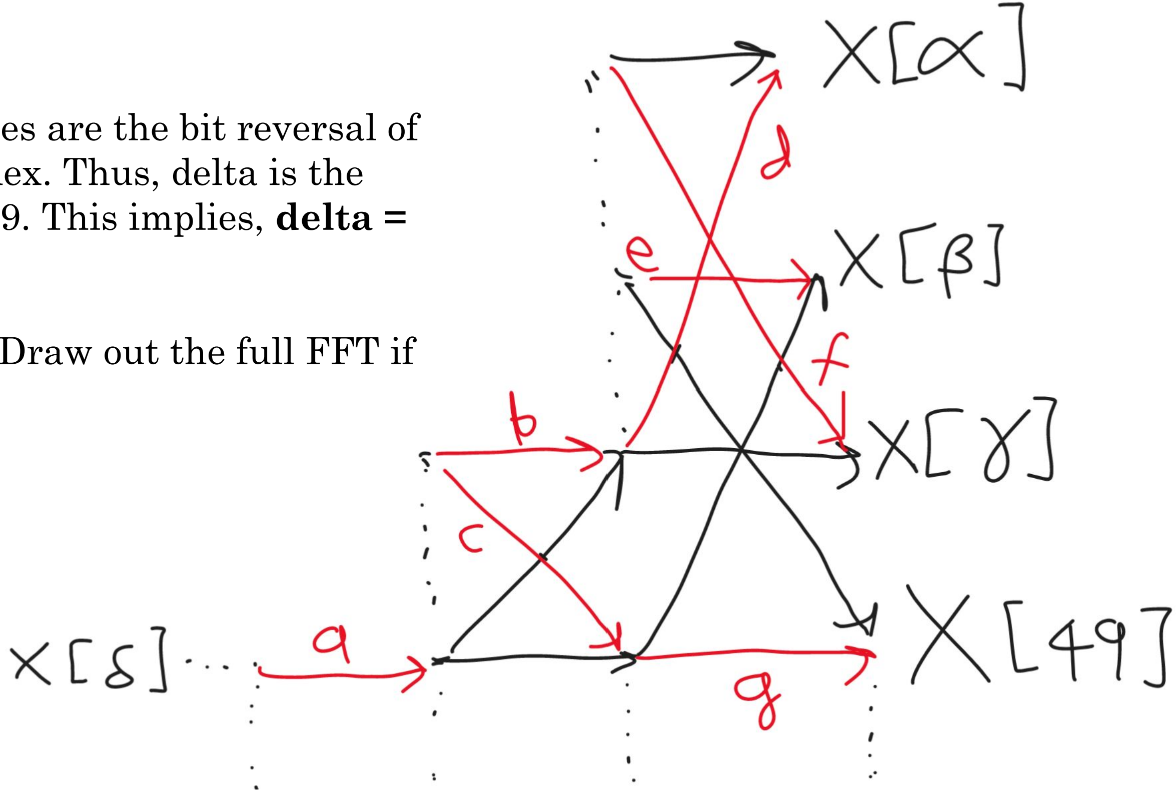
We know X[gamma] and X[alpha] are connected by the last FFT. This means alpha + 64/2 = gamma = 33 = alpha + 32. This implies **alpha = 1**.

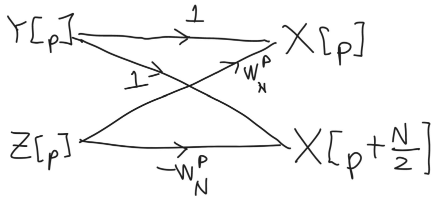


FFT Practice

FFT input indices are the bit reversal of their output index. Thus, delta is the bit-reversal of 49. This implies, **delta = 35**.

Not convinced? Draw out the full FFT if you wish.

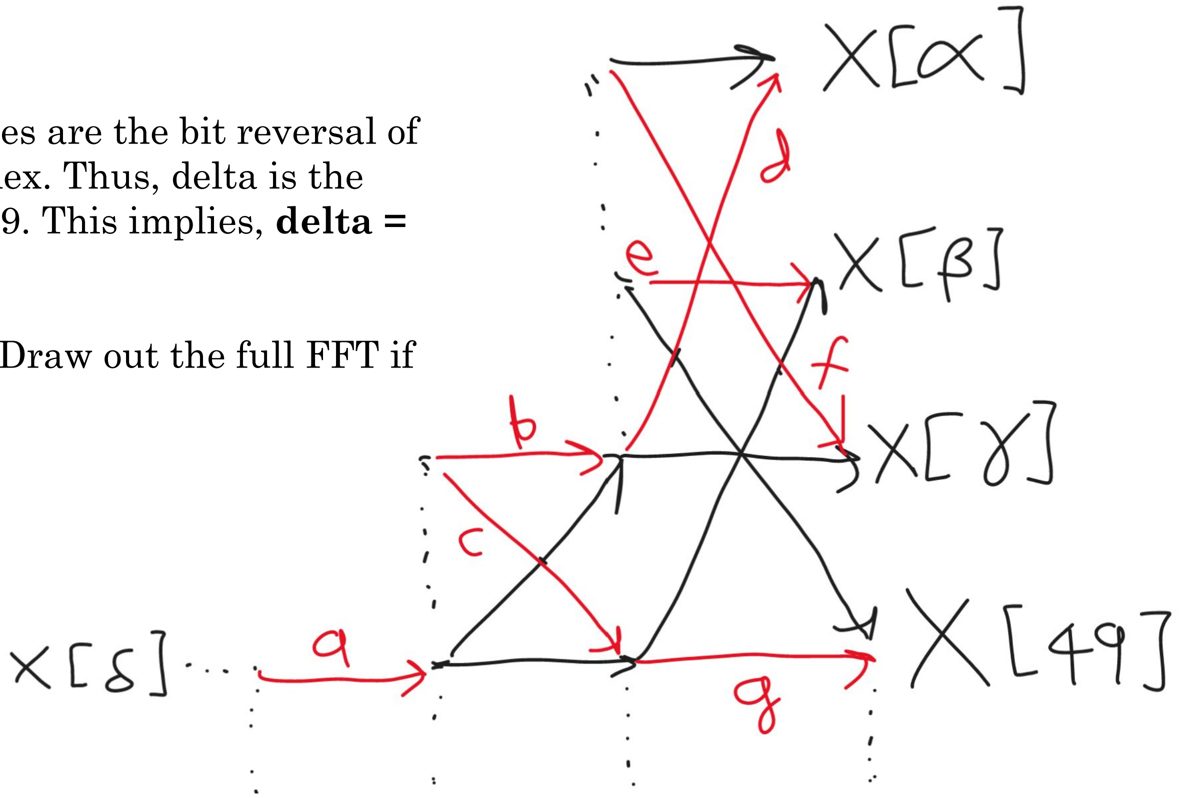




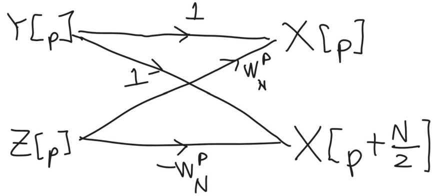
FFT Practice

FFT input indices are the bit reversal of their output index. Thus, delta is the bit-reversal of 49. This implies, **delta = 35**.

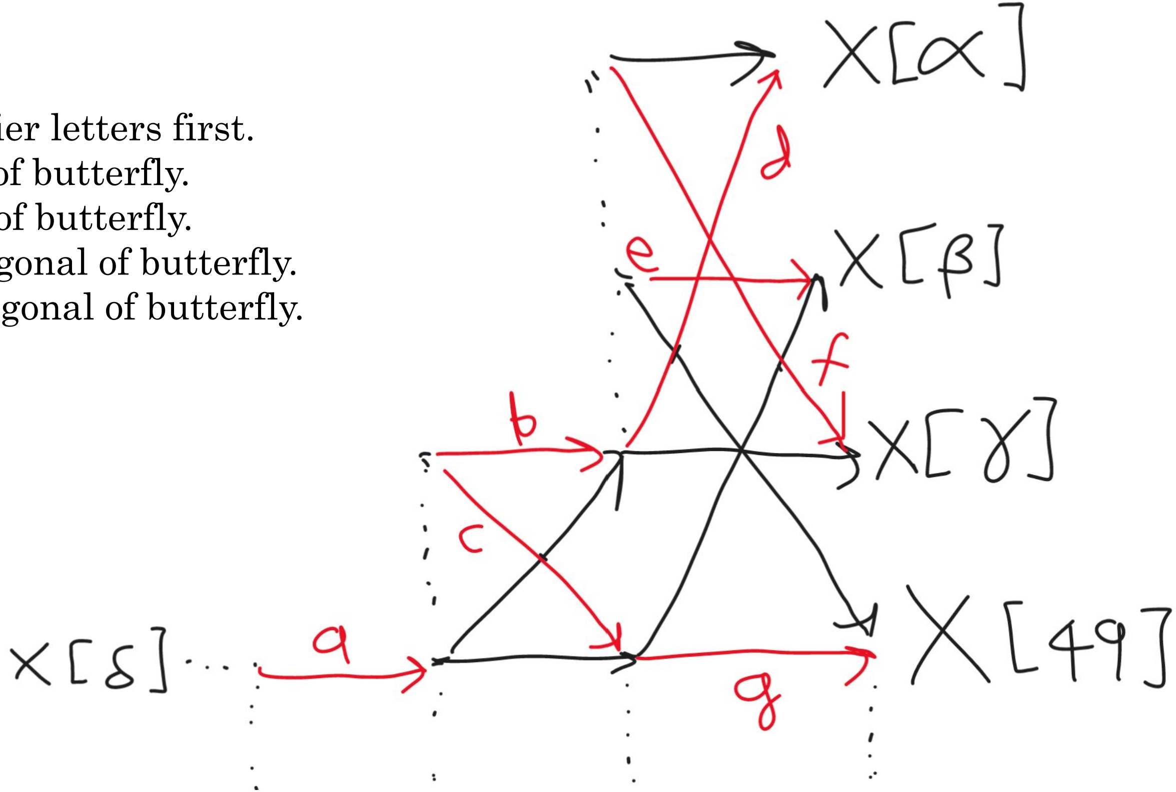
Not convinced? Draw out the full FFT if you wish.

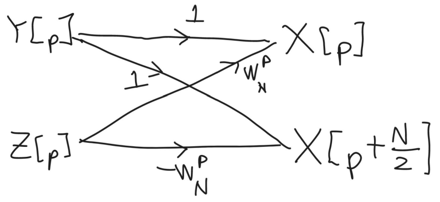


FFT Practice



Let’s do the easier letters first.
e = 1: Top part of butterfly.
b = 1: Top part of butterfly.
f = 1: Down-diagonal of butterfly.
c = 1: Down-diagonal of butterfly.



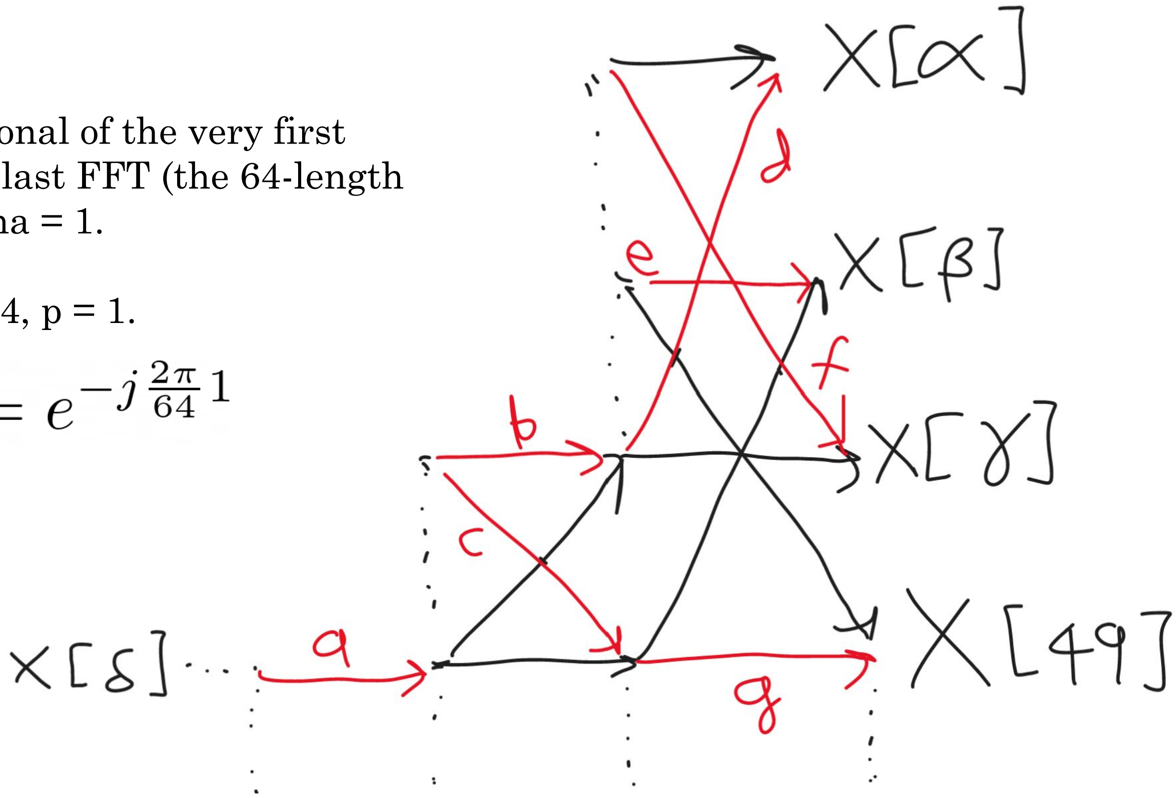


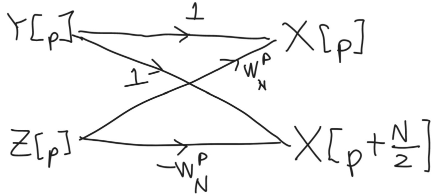
FFT Practice

d is the up-diagonal of the very first butterfly in the last FFT (the 64-length FFT), since alpha = 1.

Thus, use N = 64, p = 1.

$$d = W_{64}^{1} = e^{-j\frac{2\pi}{64}1}$$



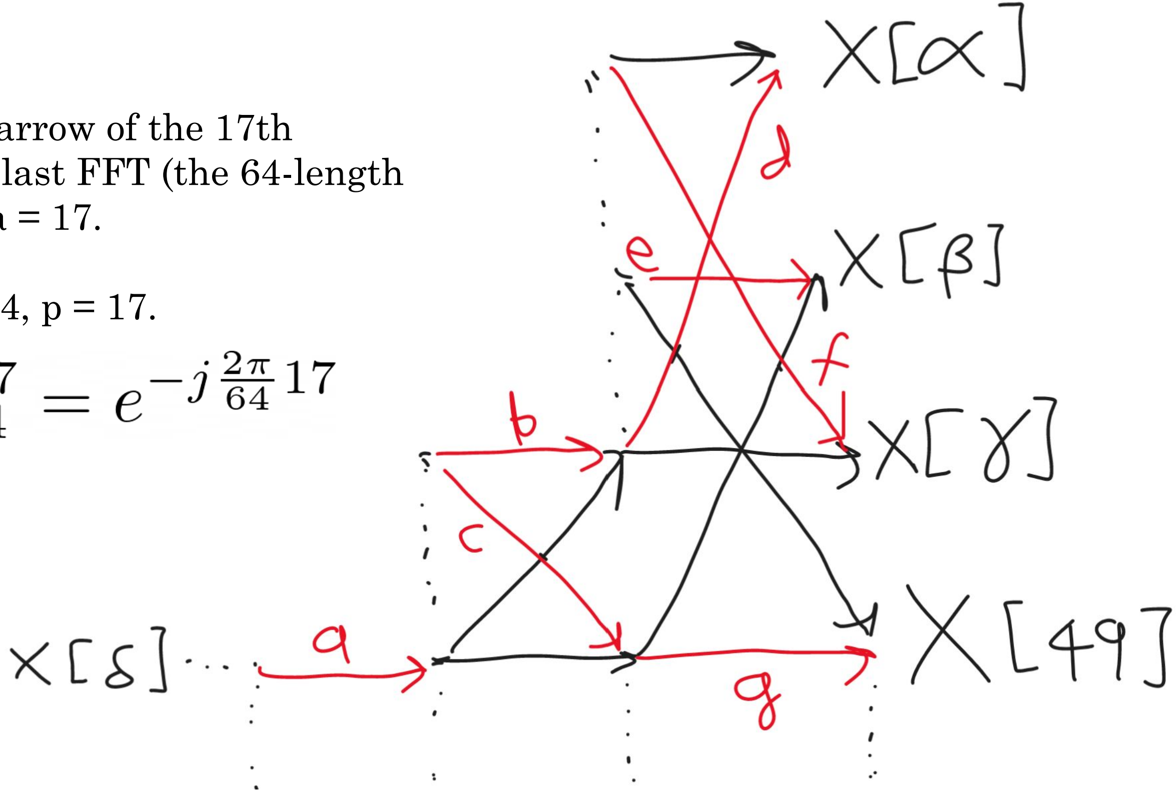


FFT Practice

g is the bottom arrow of the 17th butterfly in the last FFT (the 64-length FFT), since beta = 17.

Thus, use N = 64, p = 17.

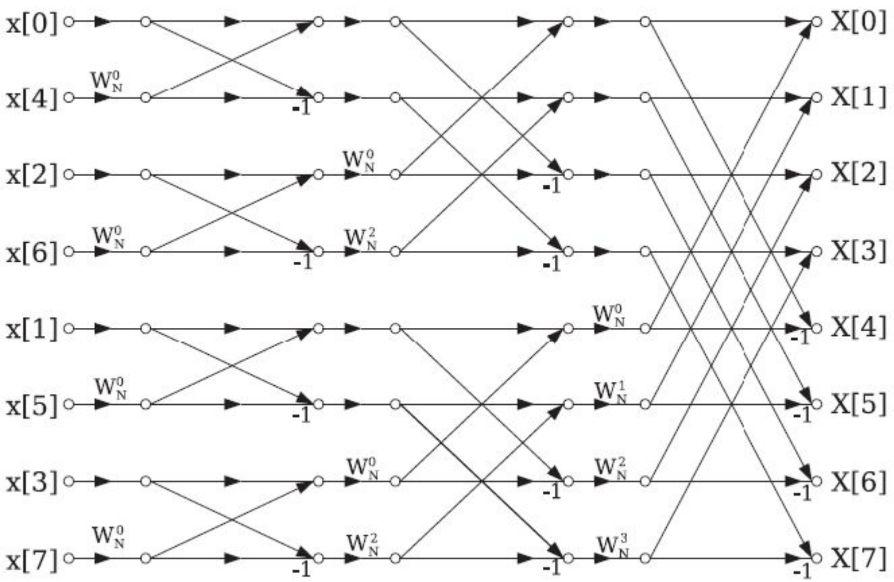
$$g = -W_{64}^{17} = e^{-j\frac{2\pi}{64}17}$$



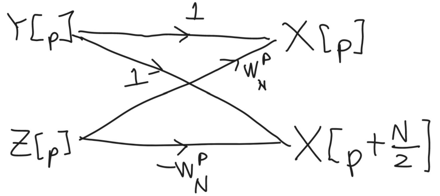
A reminder...

In the last FFT, we're doing 8-length butterflies.
In the second-to-last FFT, we're doing 4-length butterflies.
In the third-to-last FFT, we're doing 2-length butterflies.

"Final elements" 0 and 1 are grouped into one butterfly. 2 and 3 are grouped. 4 and 5 are grouped. 6 and 7 are grouped.



Decimation in time for length-8 signal

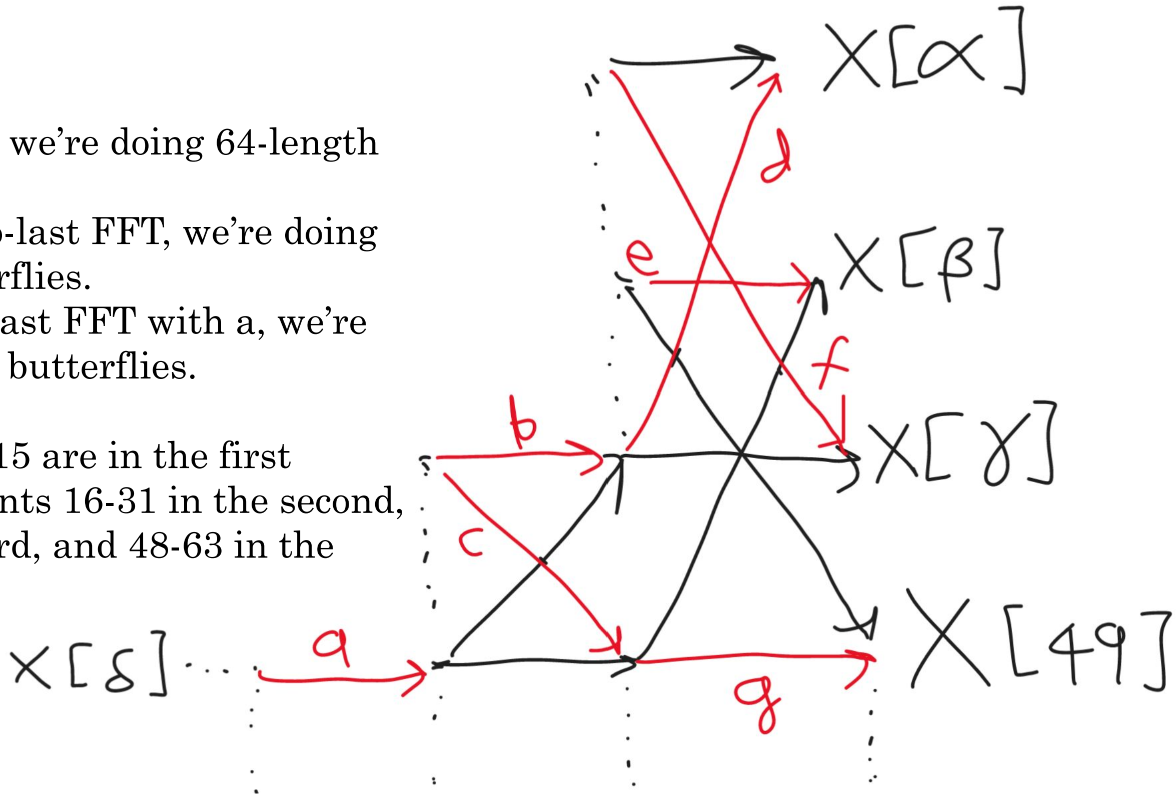


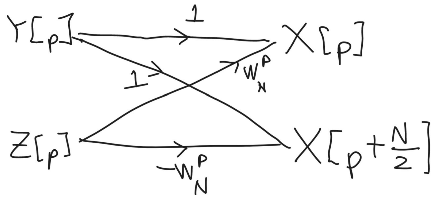
FFT Practice

Analogously,

In the last FFT, we're doing 64-length butterflies.
In the second-to-last FFT, we're doing 32-length butterflies.
In the third-to-last FFT with a, we're doing 16-length butterflies.

So, elements 0-15 are in the first butterfly, elements 16-31 in the second, 32-47 in the third, and 48-63 in the fourth.





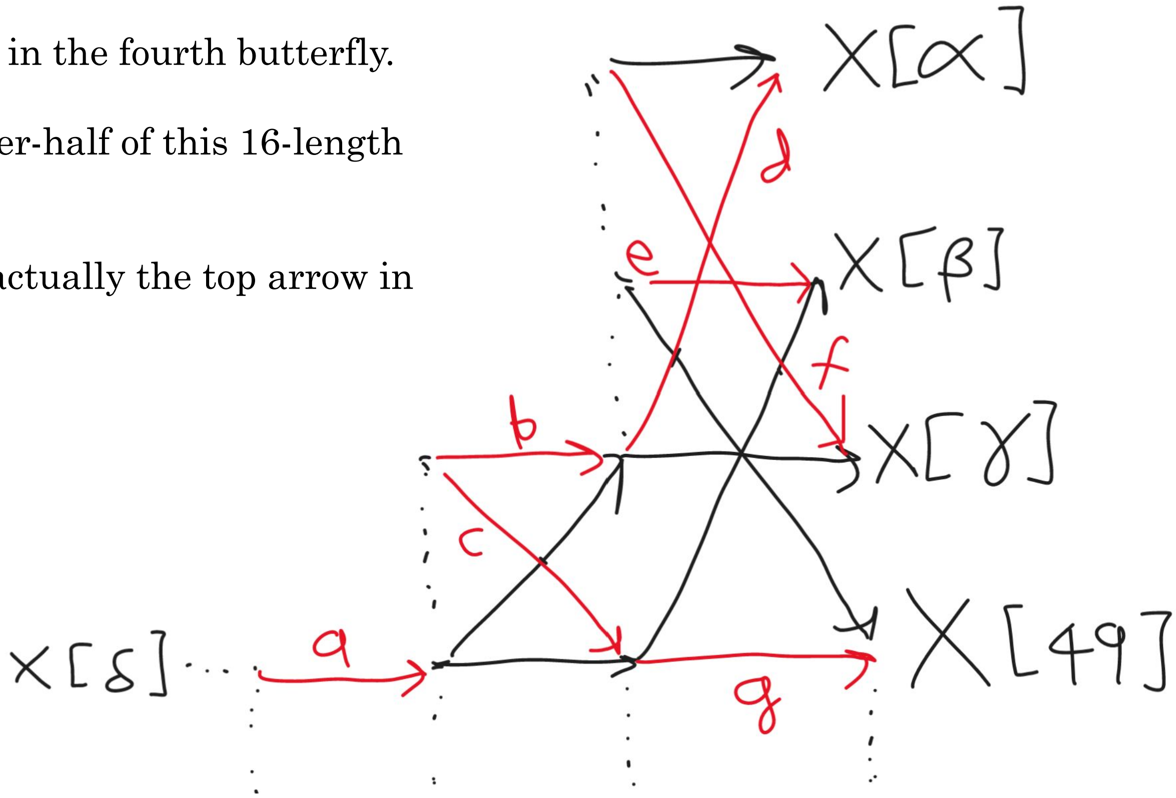
FFT Practice

Elements 48-63 in the fourth butterfly.

49 is in the upper-half of this 16-length butterfly.

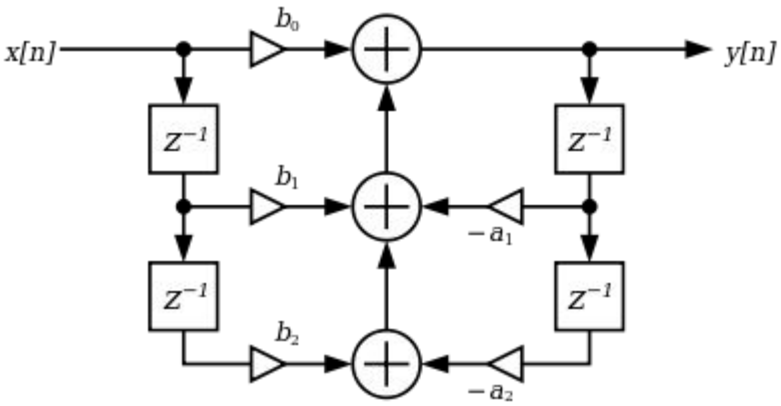
Therefore, a is actually the top arrow in the butterfly.

a = 1.

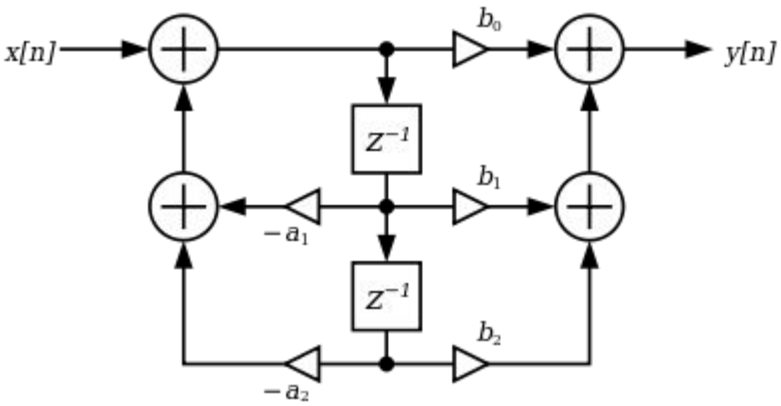


Filter Design - IIR

Direct Form 1



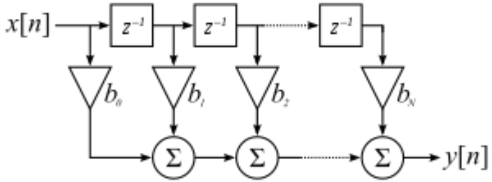
Direct form 2



$$y[n] + a_1 y[n-1] + a_2 y[n-2] = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2]$$

$$y[n] = -a_1 y[n-1] - a_2 y[n-2] + b_0 x[n] + b_1 x[n-1] + b_2 x[n-2]$$

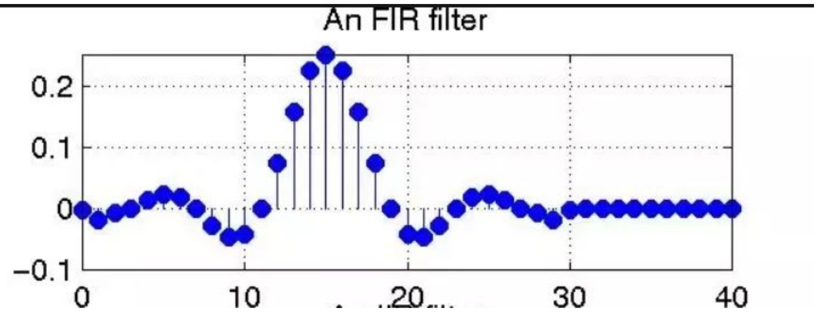
Filter Design - FIR



Filter type	Symmetry	Length	$H_d(0)$	$H_d(\pi)$	Possible canonical filter types
Type-I	Even	Odd	May be non-zero	May be non-zero	LP, HP, BP, BS
Type-II	Even	Even	May be non-zero	Always zero	LP, BP
Type-III	Odd	Odd	Always zero	Always zero	BP
Type-IV	Odd	Even	Always zero	May be non-zero	BP, HP

$$H(\omega) = A(\omega)e^{j\Psi(\omega)}. \longrightarrow H(\omega) = |A(\omega)|e^{j(\Psi(\omega)+\angle A(\omega))}.$$

$$|H(\omega)| = |A(\omega)|$$
$$\angle H(\omega) = \Psi(\omega) + \angle A(\omega).$$



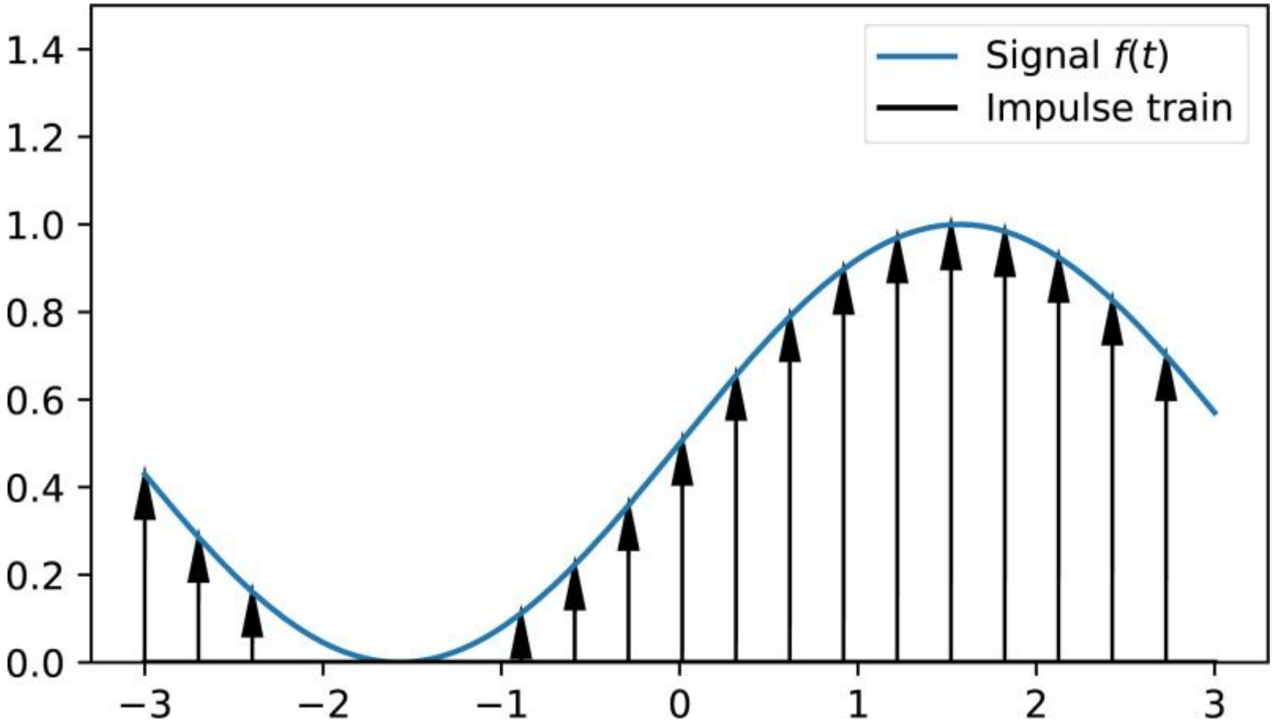
Go over Corey's notes, they're very good

GLP vs LP

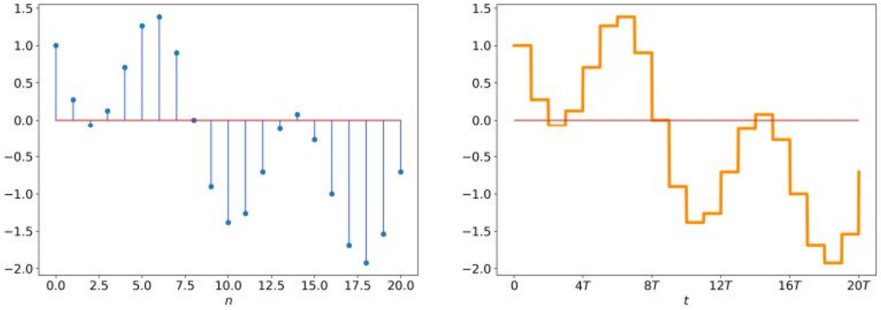
$$\text{Linear phase} : \angle H(\omega) = -\alpha\omega$$
$$\text{Generalized linear phase} : \angle H(\omega) = -\alpha\omega + \beta(\omega).$$

Practical sampling

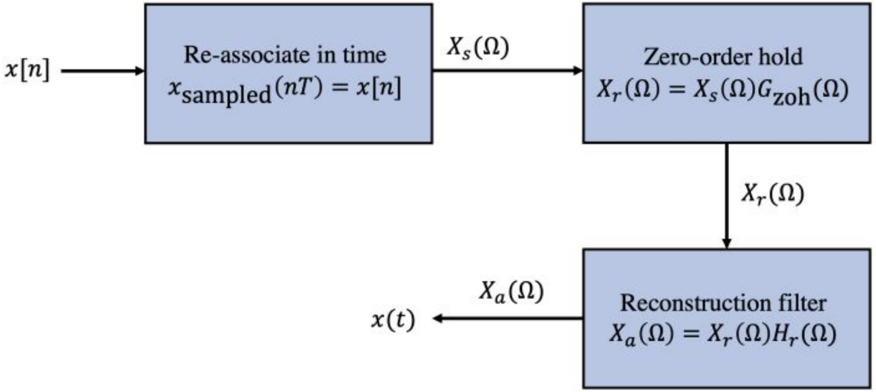
$$x_{\text{sampled}}(t) = x(t) \sum_{n=-\infty}^{\infty} \delta(t - nT).$$

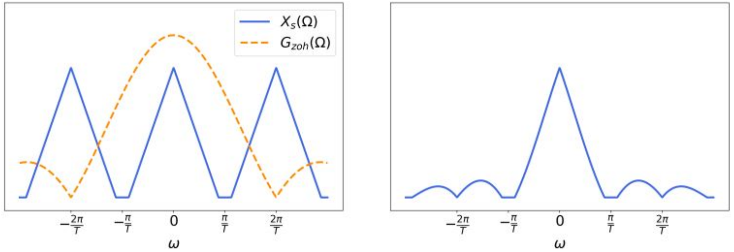


Practical reconstruction (yes I stole this from course notes)

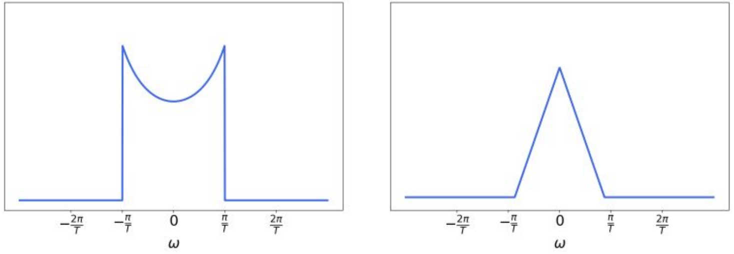


(a) $x[n]$ (b) $x_r(t) = \sum_{n=-\infty}^{\infty} x[n] g_{\text{zoh}}(t - nT)$





(a) $|X_s(\Omega)|$ and $|G_{\text{zoh}}(\Omega)|$ (b) $|X_r(\Omega)|$



(c) $|H_r(\Omega)|$ (d) $|X_a(\Omega)|$

The End

- Thanks for attending!
- Good luck studying!