CESE4030
Embedded Systems Laboratory

Introduction to Digital Filtering

Guohao Lan

26/02/2024

Reference

Steven W. Smith, "*The scientist and engineer's guide to digital signal processing*." 1997.

Sanjeev R. Kulkarni, "*Lecture Notes for ELE201 Introduction to Electrical Signals and Systems*", Princeton University, 2002.

Outline

- Introduction
- Z Transform
- Butterworth Filters
- Complementary Filter
- Fixed-point Implementation

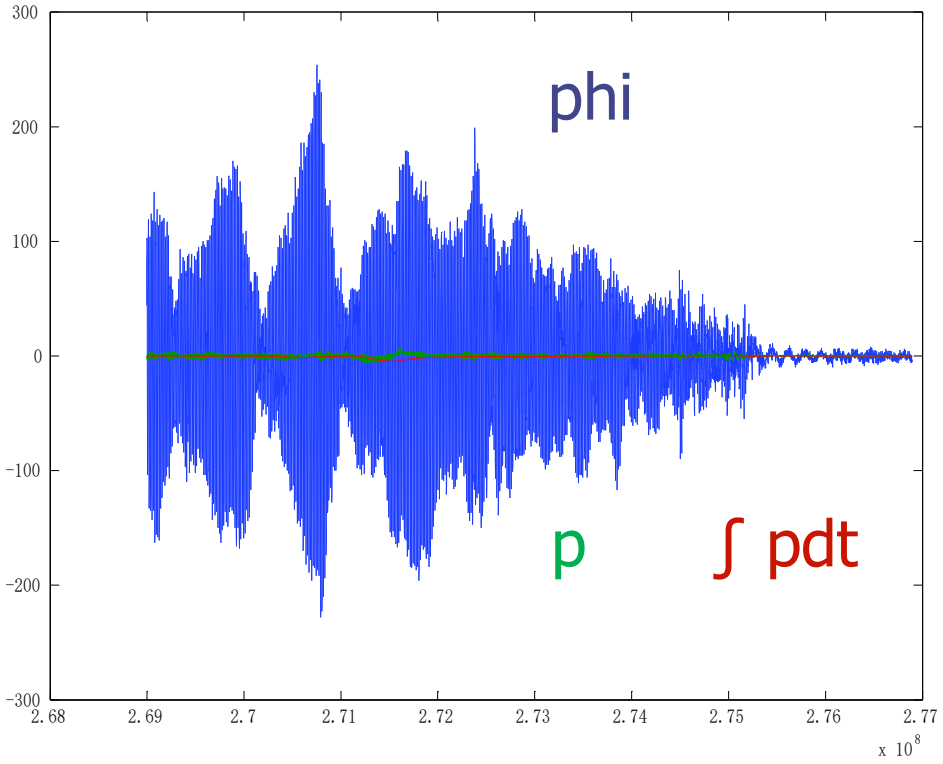
Why Signal Processing?

- Improve/restore media content
  - Compression/Decompression
  - Audio filtering (bass, treble, equalization)
  - Video filtering (enhancement, contours, ..)
  - Noise filtering (accel, gyro data)
  - Data fusion (mixing accel + gyro data)

DSP is Everywhere

- Smartphone: Sensing + Localization
- Cars: Autonomous vehicles
- Video Streaming
- Model Quad Rotors ...

Example: QR Sensor Signals phi, p



Signal from the previous version of QR:

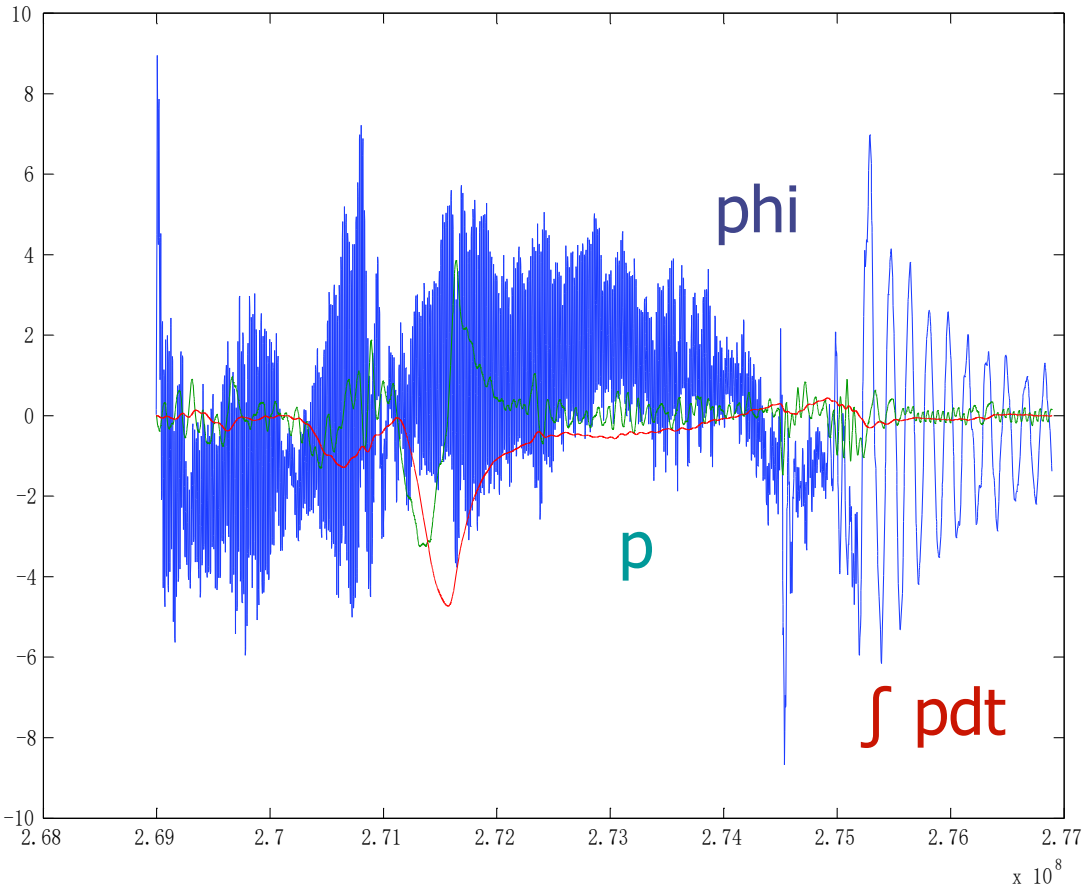
- phi is the roll angle (from acc)
- p is the roll rate (from gyro)

- red is the
  - Integration of roll rate = roll angle

Ideally, they should match.

Issues?
- Scaling
- Noise

After some low-pass filtering



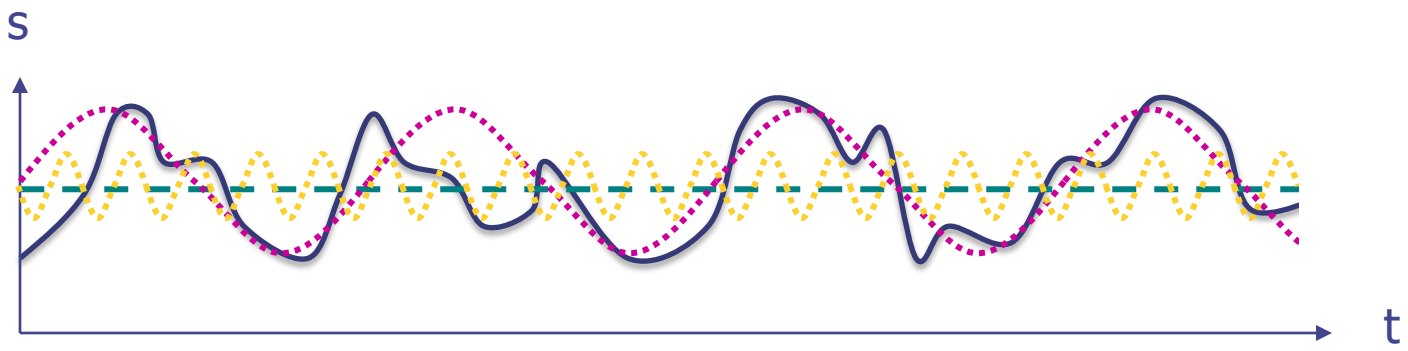
Objectives of this Lecture

- Understand the benefits of Digital Filtering
- Understand *some* of the basic principles

So that:

Implement your own filters for the QR

Signals and Frequency Synthesis

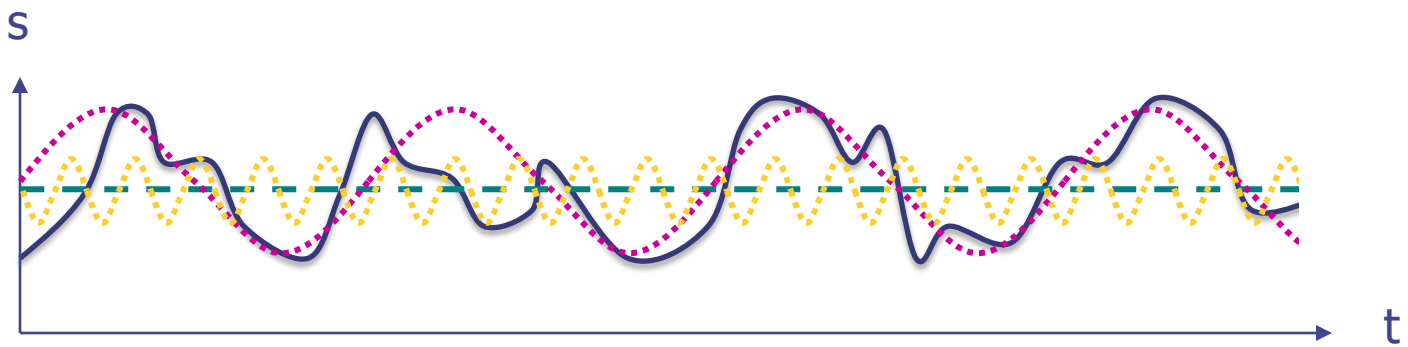


Usually signals (such as s) are composed of signals with many frequencies.

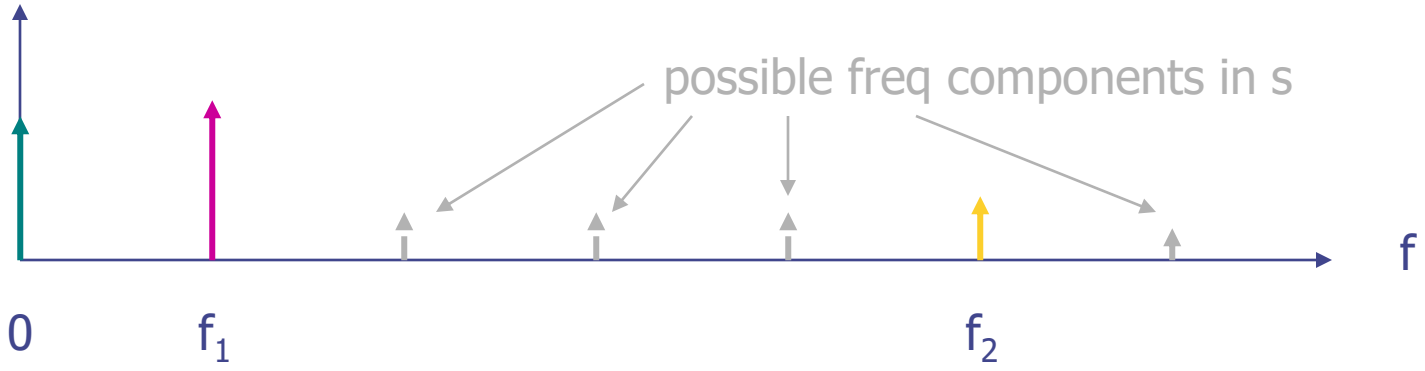
For instance, s contains
- 0 Hz component (green dashed line) ---- DC term
- lowest freq component (purple dashed line)
- higher freq component (yellow dashed line)
- and others

Fourier: Any *periodic* signal with base frequency f_b
can be constructed from sine waves with frequency f_b, $2f_b$, $3f_b$, …

Frequency Spectrum

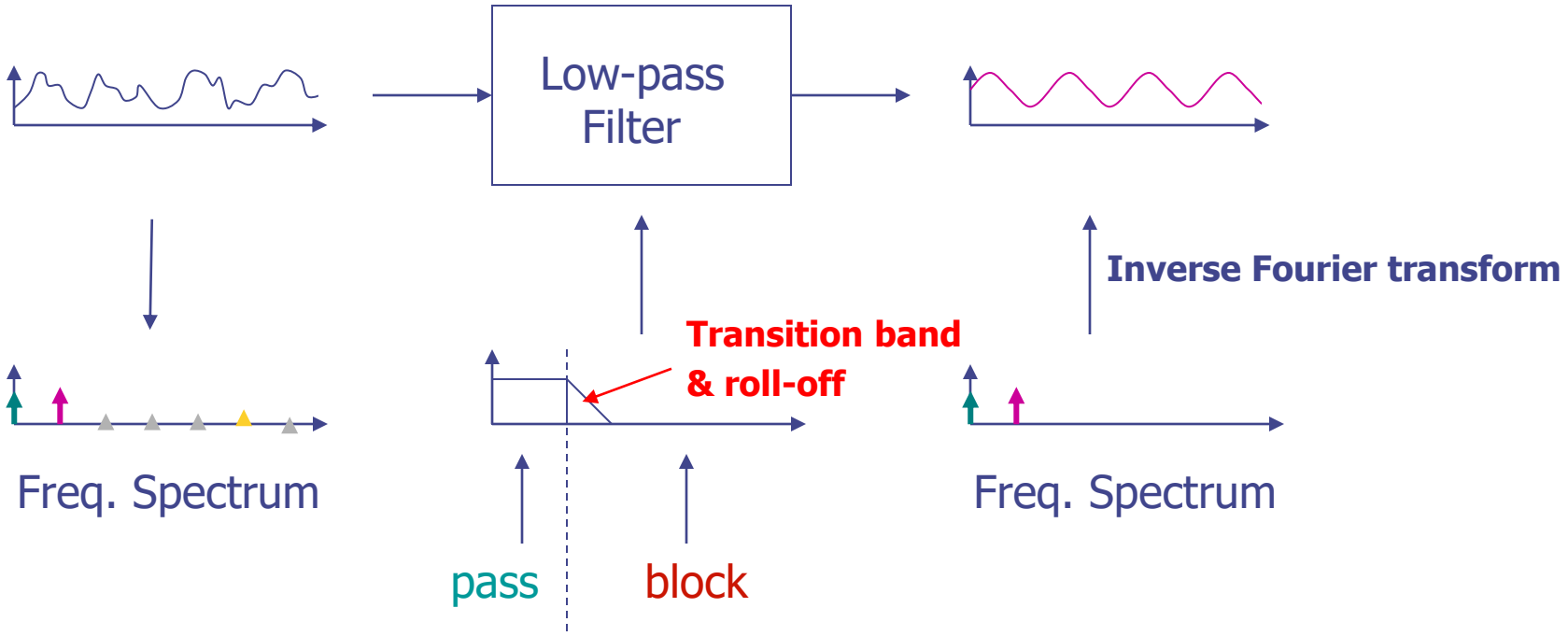


The frequency spectrum of s is:



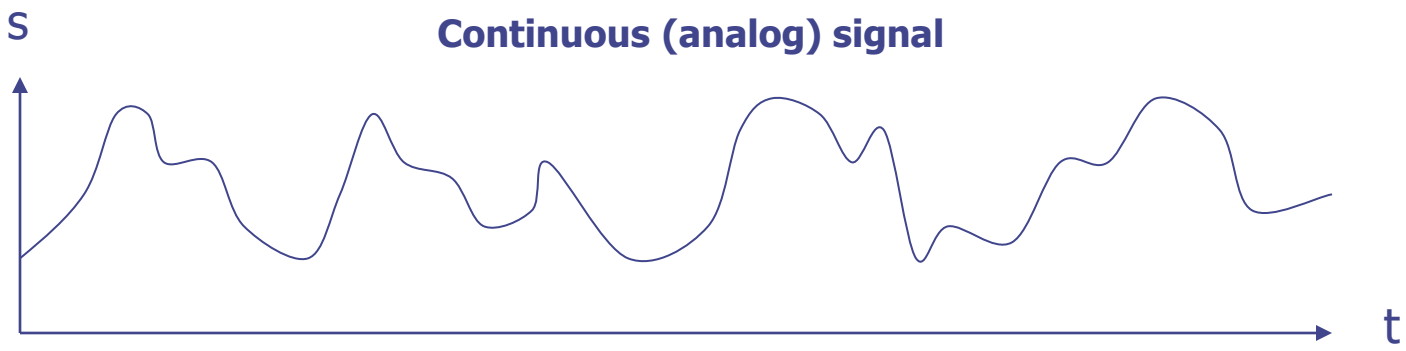
Filter: Frequency Response

Often filters are designed to filter frequency components in a signal

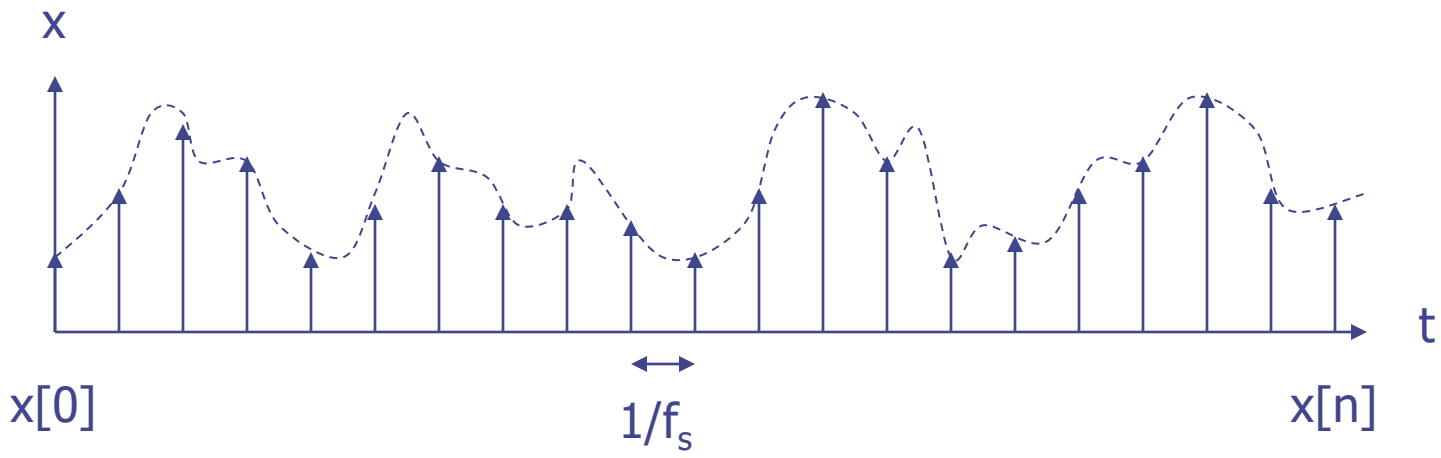


Filter's *Frequency Response*

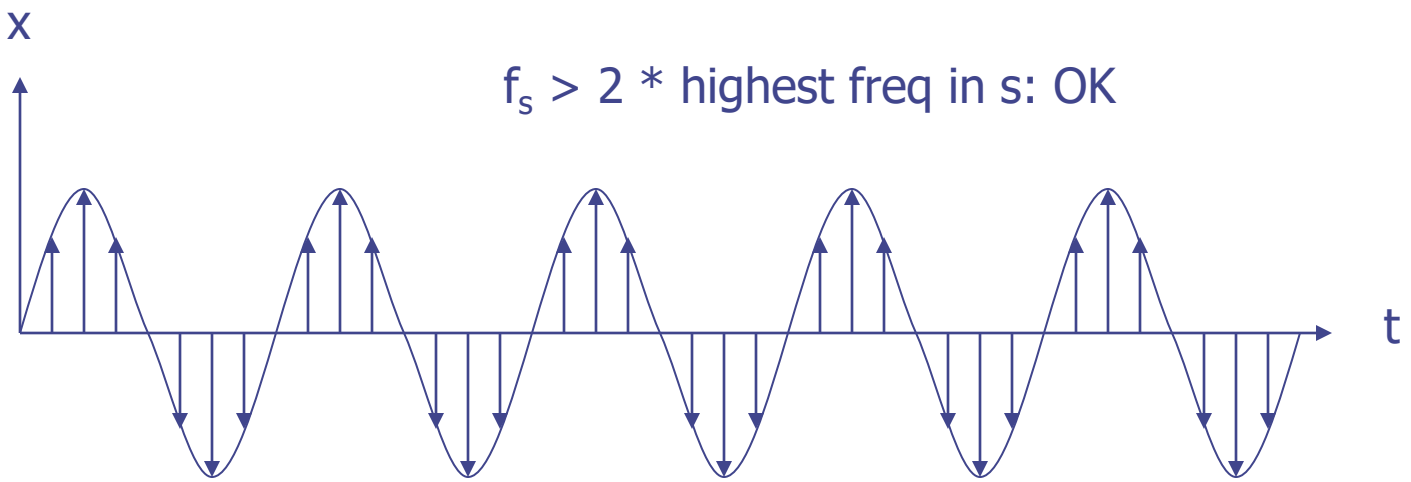
Sampling A Signal Properly

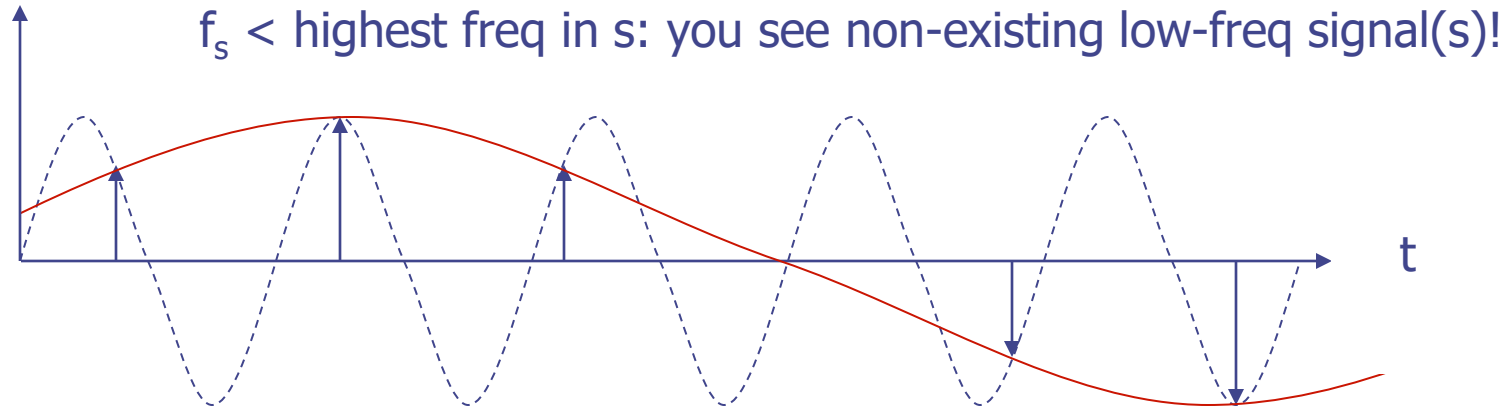


s sampled at *discrete* time intervals (sample frequency f_s): x[n]

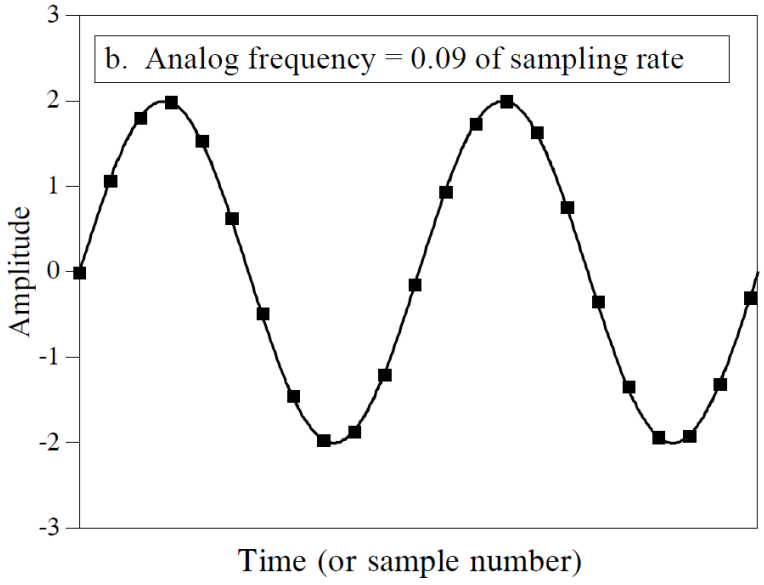


Sampling: Avoid Aliasing

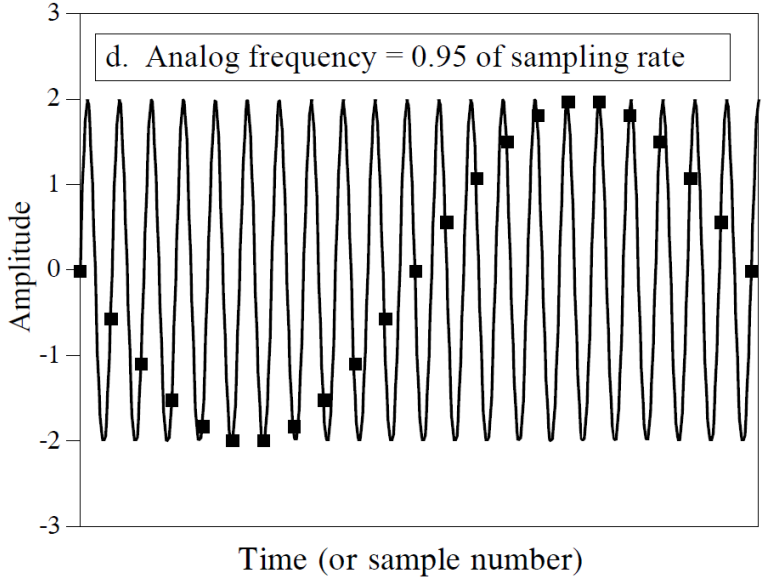




Sampling: Avoid Aliasing (cont.)

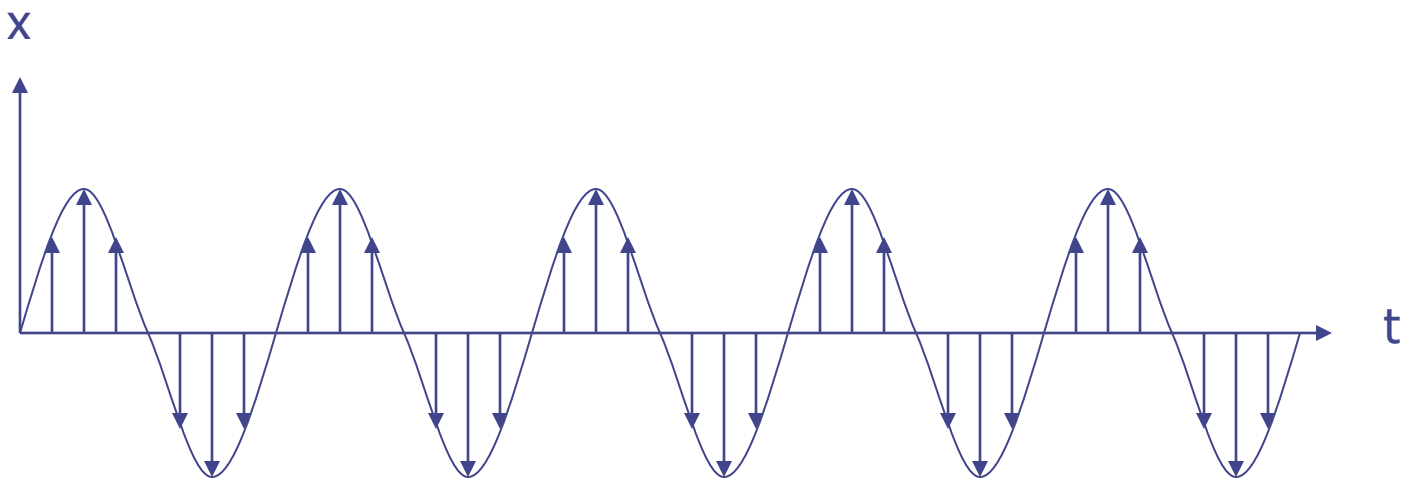


fs = 10 f



fs = f

Sampling: Avoid Aliasing (cont.)



Shannon Sampling Theorem: a bandlimited signal with maximum frequency s can be perfectly reconstructed from samples if the sampling frequency satisfies:

$$f_s > 2 * \text{highest freq in } s$$

Example Filter: Moving Average

y[n] = 1/3 x[n] + 1/3 x[n-1] + 1/3 x[n-2]

$$y[i] = \frac{1}{M} \sum_{j=0}^{M-1} x[i+j]$$


```
x[0] = get_sample();
y[0] = (x[0]+x[1]+x[2])/3;
put_sample(y[0]);
x[2] = x[1]; x[1] = x[0];
```

MA filter filters (removes) signals of certain frequency:

Frequency Behavior MA (n=3)

High sampling frequency fs = 12 f: amplitude y = 0.77

x = 0.00, 0.33, 0.66, 1.00, 0.66, 0.33, 0.00, -0.33, -0.66, -1.00, -0.66, -0.33, 0.00

y = 0.00, 0.11, 0.33, 0.66, 0.77, 0.66, 0.33, 0.00, -0.33, -0.66, -0.77, -0.66, -0.33

Lower sampling frequency fs = 4 f: amplitude y = 0.33

x = 0.00, 1.00, 0.00, -1.00, 0.00, 1.00, 0.00, -1.00, 0.00, 1.00, 0.00, -1.00, 0.00

y = 0.00, 0.33, 0.33, 0.00, -0.33, 0.00, 0.33, 0.00, -0.33, 0.00, 0.33, 0.00, -0.33

transient — steady-state

Question: Do you see any major difference between x and y?

Frequency Behavior MA (n=3)

High sampling frequency fs = 12 f: amplitude y = 0.77

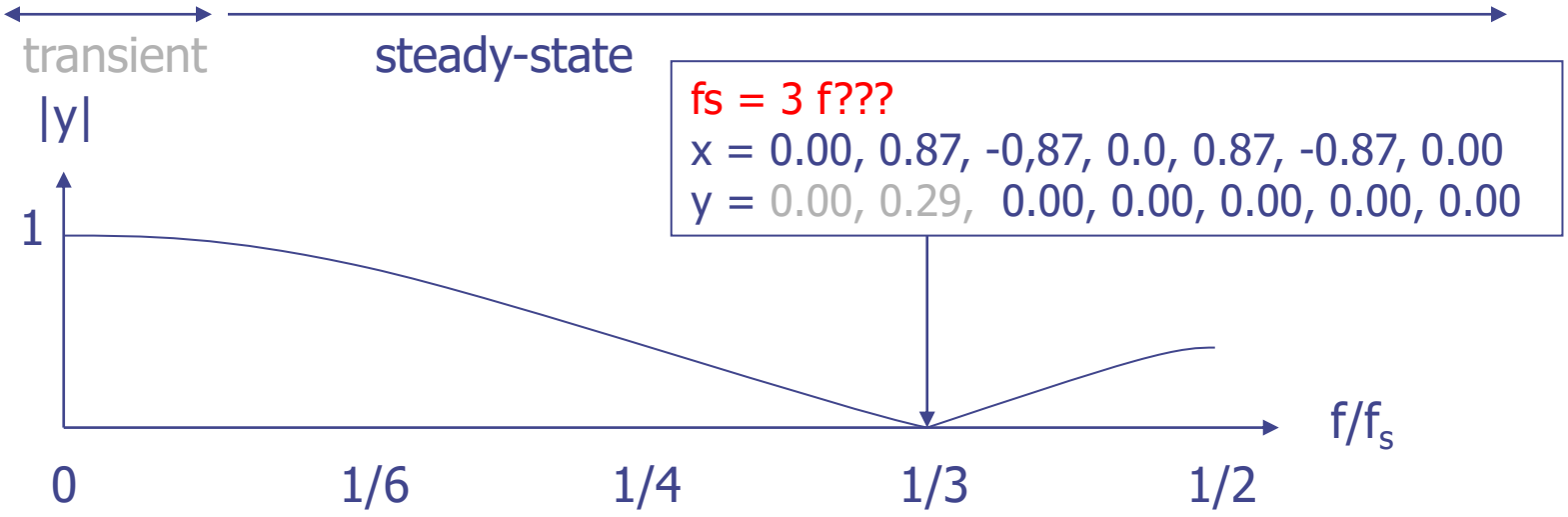
x = 0.00, 0.33, 0.66, 1.00, 0.66, 0.33, 0.00, -0.33, -0.66, -1.00, -0.66, -0.33, 0.00

y = 0.00, 0.11, 0.33, 0.66, 0.77, 0.66, 0.33, 0.00, -0.33, -0.66, -0.77, -0.66, -0.33

Lower sampling frequency fs = 4 f: amplitude y = 0.33

x = 0.00, 1.00, 0.00, -1.00, 0.00, 1.00, 0.00, -1.00, 0.00, 1.00, 0.00, -1.00, 0.00

y = 0.00, 0.33, 0.33, 0.00, -0.33, 0.00, 0.33, 0.00, -0.33, 0.00, 0.33, 0.00, -0.33



fs = 3 f???

x = 0.00, 0.87, -0,87, 0.0, 0.87, -0.87, 0.00

y = 0.00, 0.29, 0.00, 0.00, 0.00, 0.00, 0.00

Frequency response of MA with n=3

Outline

- Introduction
- Z Transform
- Butterworth Filters
- Complementary Filter
- Fixed-point Implementation

Analysis: Z Transform

Why do we need this?

- We can numerically evaluate frequency behavior (with the actual signal)
- Can we analyze frequency behavior through *analytic* ways?
- For this we introduce Z transformation

- Let x[n] be a signal in the time domain (n).
- The Z transform of x[n] is given by

$$X(z) = \Sigma_n \, x[n] \, z^{-n}$$

where z is a complex variable.

- Example:

  x = 0.00, 0.33, 0.66, 1.00, 0.66, ..

  $X = 0 + 0.33z^{-1} + 0.66z^{-2} + z^{-3} + 0.66z^{-4} + \ldots$

Z Transform: Properties

- Z transforms make life easy
- Properties of the Z transform, Shifting:

- Let y[n] = x[n-1] (i.e., signal delayed by 1 sample)

$$Y(z) = z^{-1} X(z)$$

- Example:

$$x = 0.00, 0.33, 0.66, 1.00, 0.66, ..$$
$$X = 0 + 0.33z^{-1} + 0.66z^{-2} + z^{-3} + 0.66z^{-4} + \ldots$$

$$y = 0.00, 0.00, 0.33, 0.66, 1.00, ..$$
$$Y = 0 + 0z^{-1} + 0.33z^{-2} + 0.66z^{-3} + z^{-4} + \ldots$$
$$= z^{-1} X$$

Z Transform: Properties (cont.)

- Other properties of the Z transform:

- Z transform of K a[n] = K A(z)
- Z transform of a[n] + b[n] = A(z) + B(z)

- Example:

$x = 0.00, 0.33, 0.66, 1.00, 0.66, ..$

$X = 0 + 0.33z^{-1} + 0.66z^{-2} + z^{-3} + 0.66z^{-4} + \ldots$

$y = 0.00, 0.66, 1.32, 2.00, 1.32, ..$

$Y = 0 + 0.66z^{-1} + 1.32z^{-2} + 2.00z^{-3} + 1.32z^{-4} + \ldots$

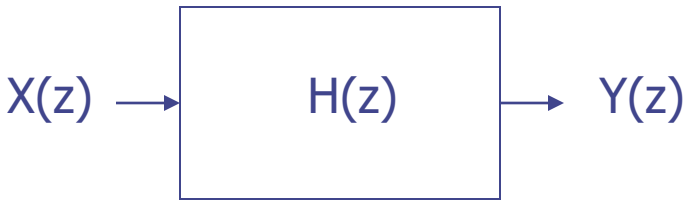
$= 2X$

Apply Z transform to MA Filter

$y[n] = 1/3\ x[n] + 1/3\ x[n-1] + 1/3\ x[n-2]$

In terms of the Z transform of y we have:

$$Y(z) = 1/3\ X(z) + 1/3\ z^{-1}\ X(z) + 1/3\ z^{-2}\ X(z)$$
$$= (1/3 + 1/3\ z^{-1} + 1/3\ z^{-2})\ X(z)$$
$$= H(z)\ X(z)$$



- It holds Y(z) = H(z) X(z), where H(z) is filter's (system's) transfer function

H(z) reveals frequency response of the filter: As Y(z) = H(z) X(z), |H(z)| determines *amplification* of X(z)

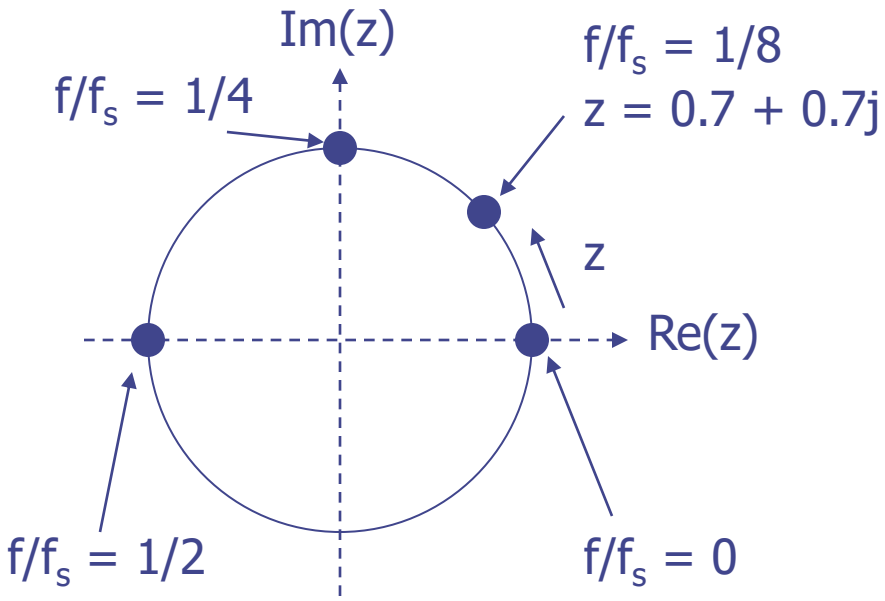
Frequency Response H(z)

H(z) reveals frequency response of the filter ($H(f)=H(z)|\ z=e^{j2\pi f}$):
As $Y(z) = H(z)\ X(z)$, $|H(z)|$ determines *amplification* of $X(z)$

The variable z is a complex variable and encodes frequency $F= f/f_s$ according to

$$z = e^{j2\pi F}$$
$$= \cos(2\pi F) + j \sin(2\pi F)$$

This corresponds to traversing the unit circle in the complex z plane:



Frequency Response MA Filter

The transfer function of the MA filter is given by:

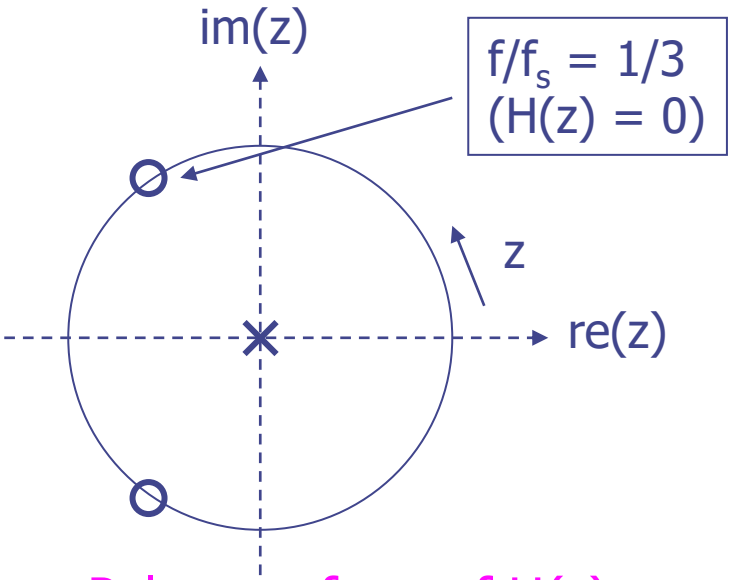
$H(z) = (1/3 + 1/3\ z^{-1} + 1/3\ z^{-2})$
$= (1/3\ z^2 + 1/3\ z + 1/3) / z^2$ (normalized)

Determine poles and zeros of H(z):

zero (= root of numerator):
$z_1 = -½+½\sqrt{3}j$, $z_2 = -½-½\sqrt{3}j$
$(H(z_{1,2}) = 0)$

pole (= root of denominator):
H(z) becomes infinite:
$z_3, z_4 = 0$
$(H(z_{3,4}) = \infty)$

<u>Simply inspect distance z to poles/zeros.</u>



Pole-zero form of H(z):

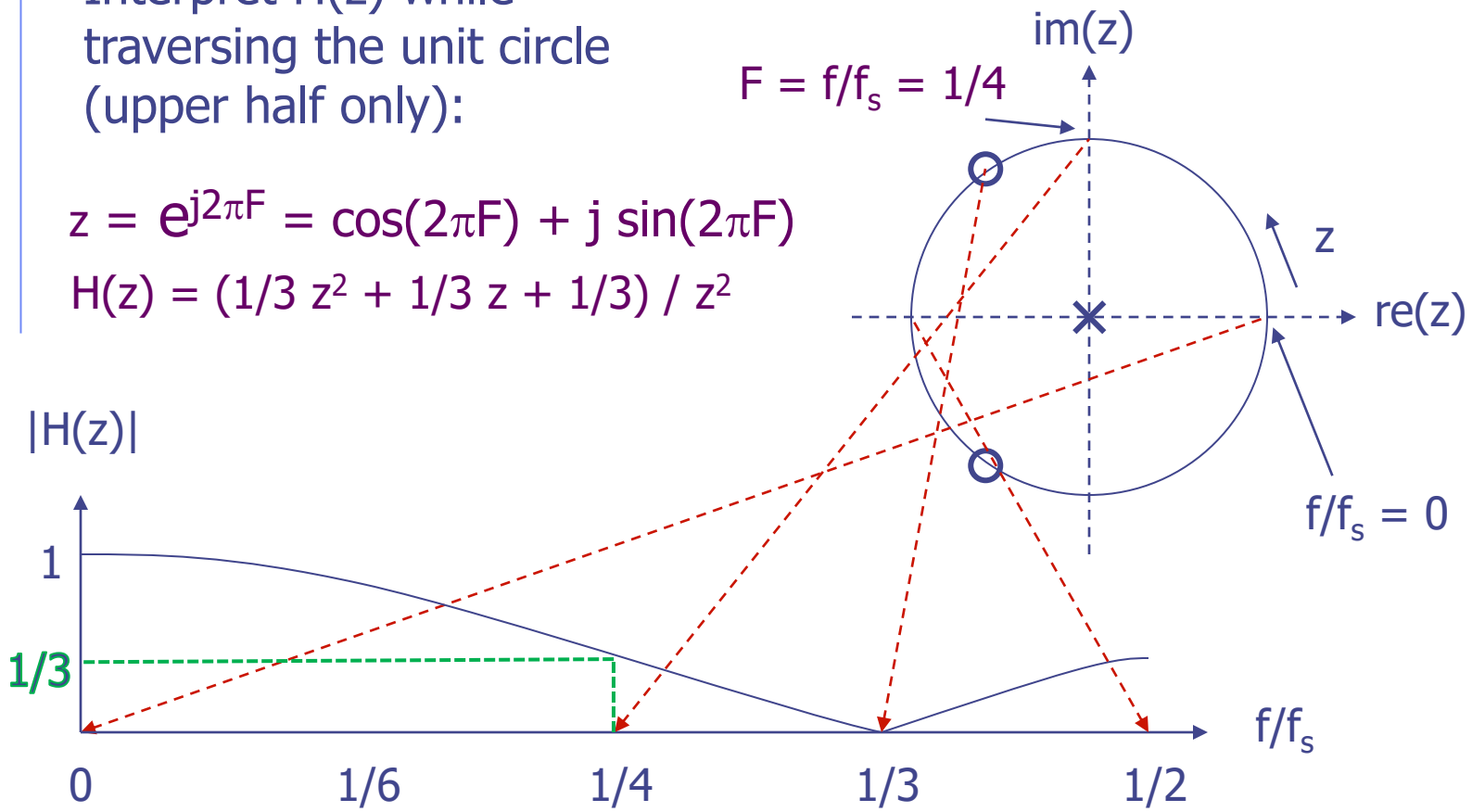
$$H[z] = \frac{(z-z_1)(z-z_2)(z-z_3)\cdots}{(z-p_1)(z-p_2)(z-p_3)\cdots}$$

Frequency Response MA Filter (cont.)

Interpret H(z) while traversing the unit circle (upper half only):

$z = e^{j2\pi F} = \cos(2\pi F) + j \sin(2\pi F)$

$H(z) = (1/3\ z^2 + 1/3\ z + 1/3) / z^2$



|H(z)| is smaller when it is closer to the zeros.

Outline

- Introduction
- Z Transform
- Butterworth Filters
- Complementary Filter
- Fixed-point Implementation

Enhancing the MA Filter

Suppose we want to extend MA filter to N terms:

$$y[n] = 1/N\ x[n] + 1/N\ x[n-1] + \ldots\ 1/N\ x[n-N-1]$$

Suppose we don't want to implement an N-cell FIFO + 2N ops and experiment with the following "short cut" (recursion):

$$y[n] = ((N-1)/N) * y[n-1] + 1/N * x[n]$$

(1st term approximates contents of FIFO after x[n-N-1] has been shifted out, 2nd term is newest sample shifted in)

Let's analyze the frequency response of this filter (recursive filter)

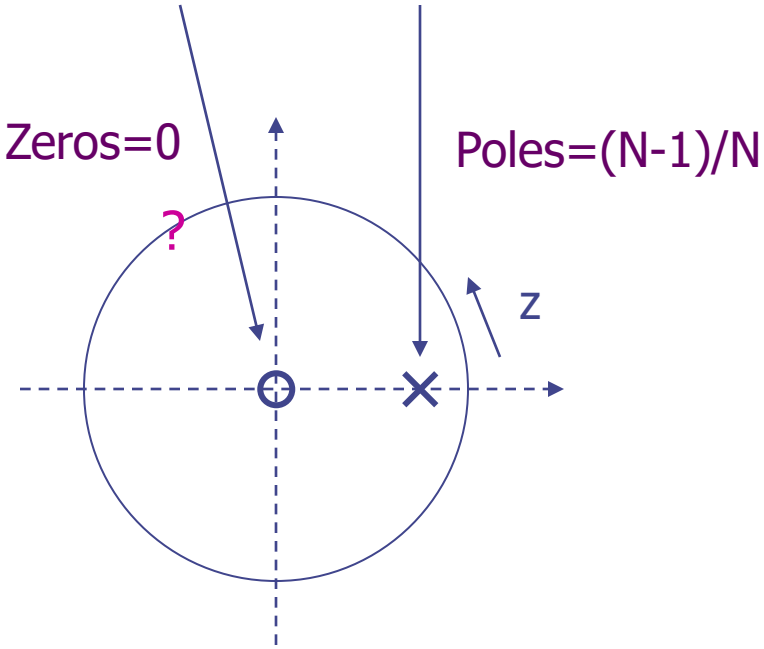
Frequency Response

$y[n] = (N-1)/N\ y[n-1] + 1/N\ x[n]$

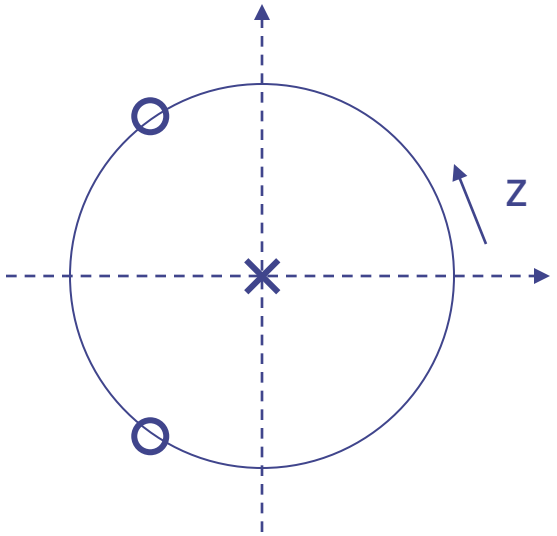
$Y(z) = (N-1)/N\ z^{-1}\ Y(z) + 1/N\ X(z)$ ⟲ Z-transform

$H(z) = (1/N) / (1 - (N-1)/N\ z^{-1})$

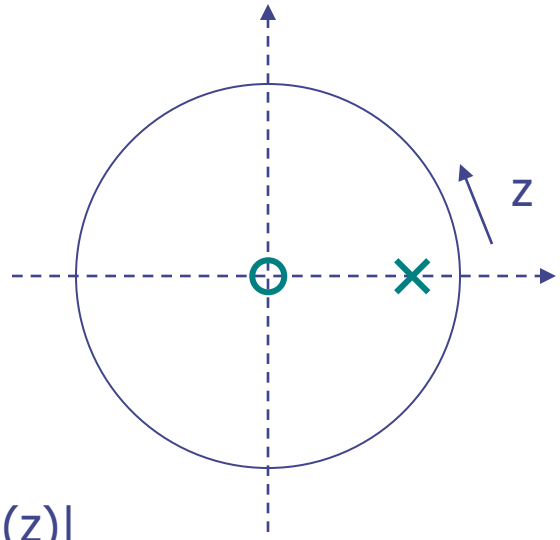
$= (z/N) / (z - (N-1)/N)$ ⟲ $H(z) = Y(z)/X(z)$

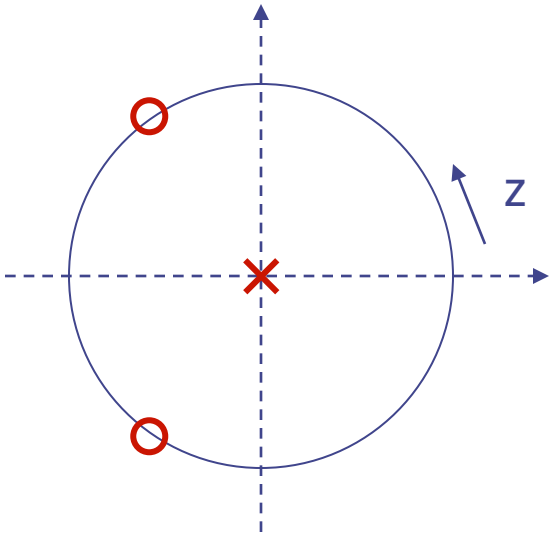


cf. MA filter:

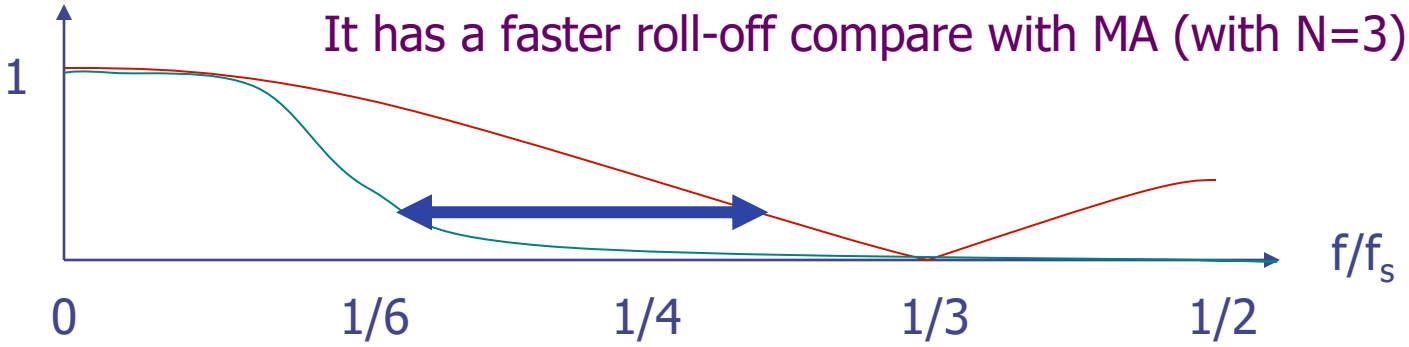


Frequency Response Comparison





|H(z)|



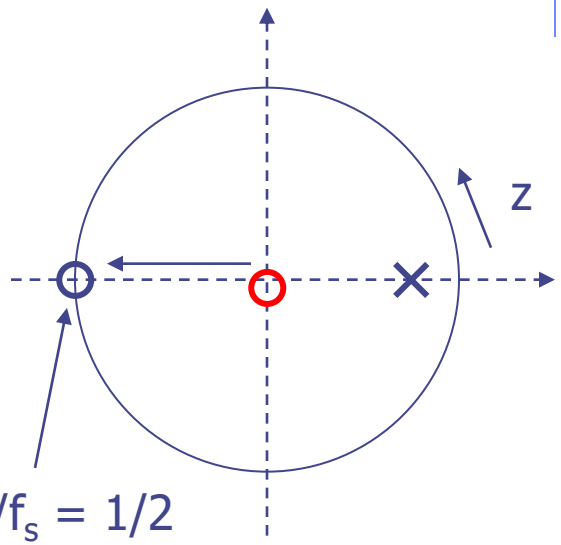
It has a faster roll-off compare with MA (with N=3)

The closer the pole is to unit circle, the sooner is the cut-off (in terms of frequency f)

Butterworth Filters

Looking at the pole-zero plot, the previous filter can be improved by moving zero to left:

now |H(z)| even becomes zero for $f = f_s/2$
so sharper cut-off.



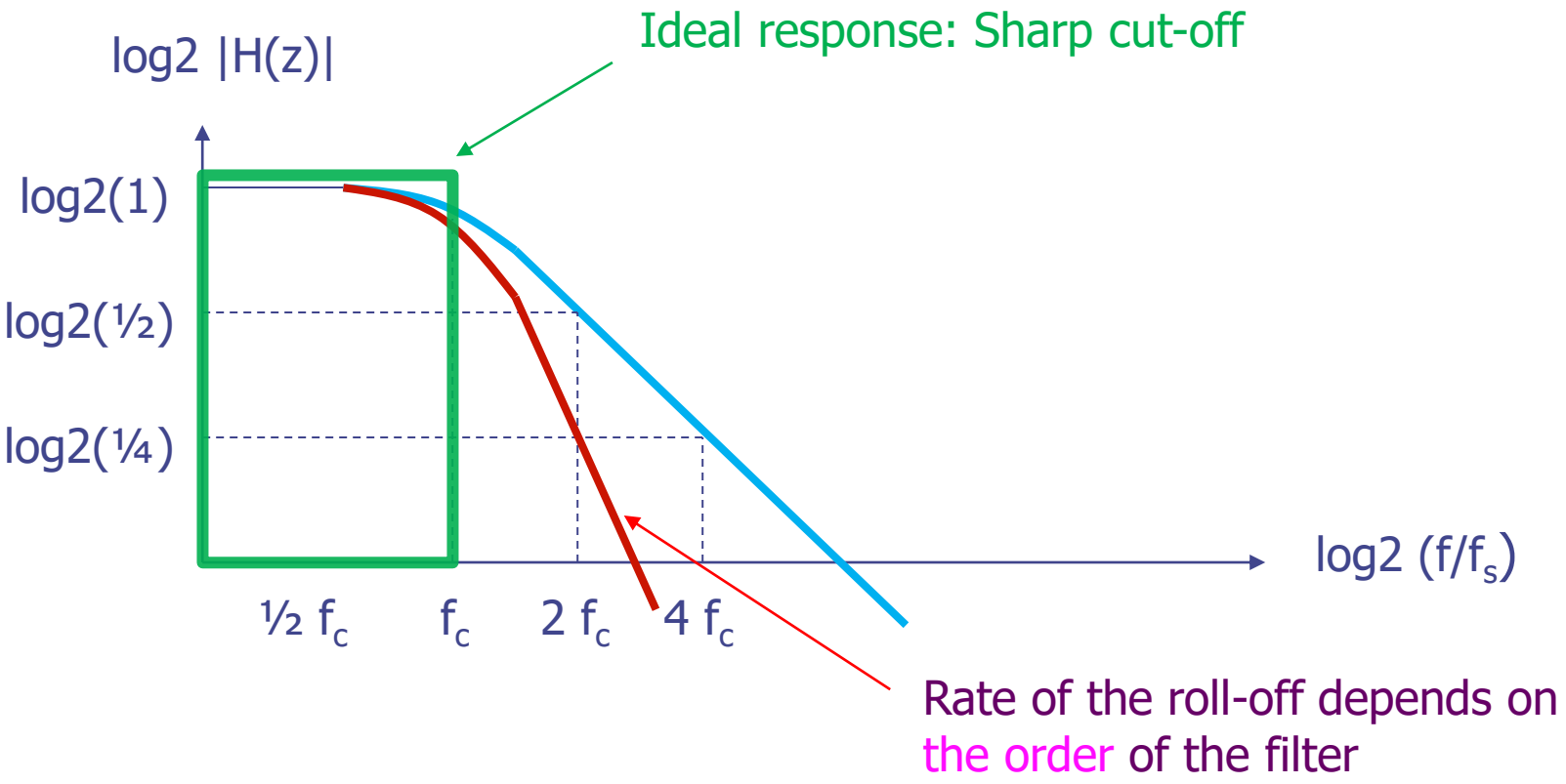
This plot corresponds to the
well-known class of **Butterworth** filters
(our case: 1st-order Butterworth):

The zero is created by adding x[n-1]:
Previously: y[n] = (N-1)/N y[n-1] + 1/N x[n]
Now: y[n] = (N-1)/N y[n-1] + 1/2N x[n] + 1/2N x[n-1]

y[n] – (N-1)/N y[n-1] = 1/2N x[n] + 1/2N x[n-1]
H(z) = ((z+1)/2N) / (z-(N-1)/N)

Butterworth Filters (cont.)

Frequency response 1st-order Butterworth:



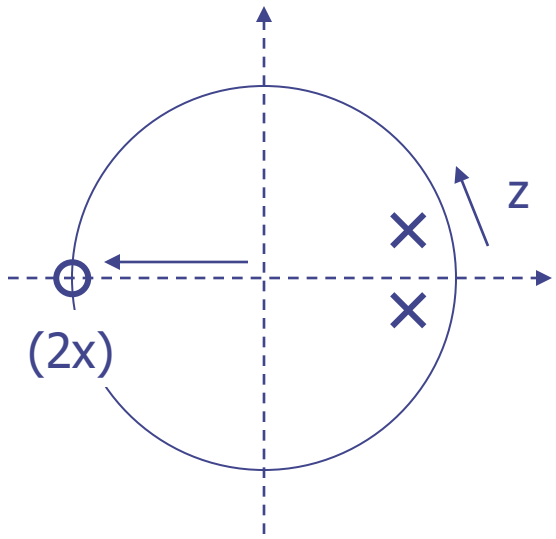
Second-order Butterworth

Looking at the pole-zero plot, the filter can be further improved by introducing more poles & zeros:

$b_0\, y[n] + b_1\, y[n-1] + b_2\, y[n-2] = a_0\, x[n] + a_1\, x[n-1] + a_2\, x[n-2]$

now |H(z)| has same cut-off freq f_c but sharper slope!

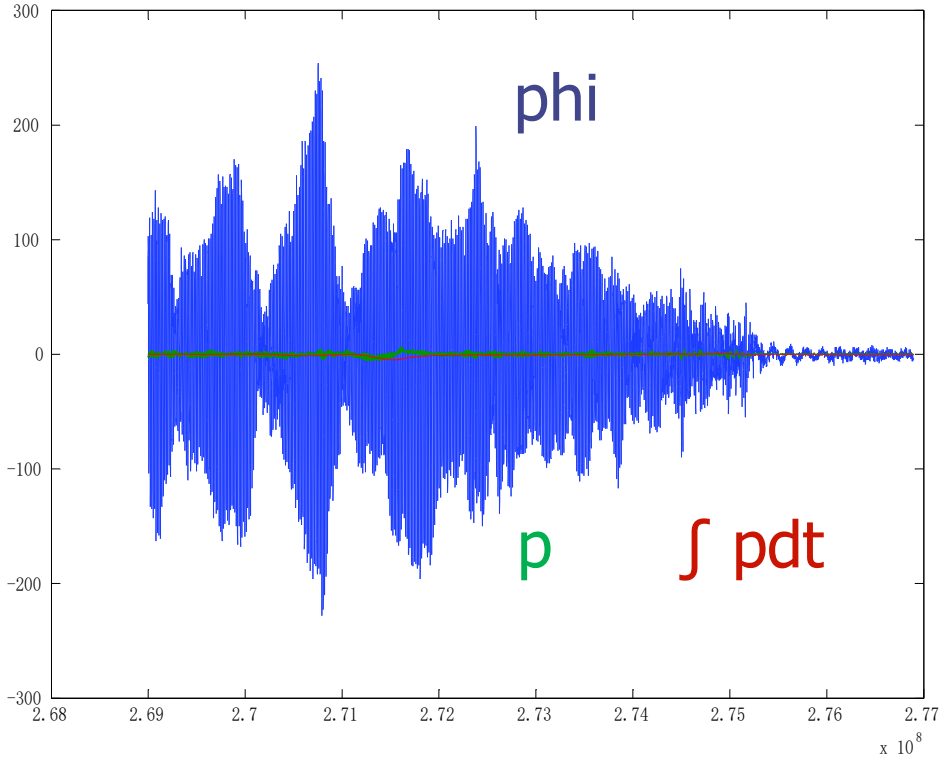
Computing h[n] (the a_i and b_i) is difficult, so use a tool to compute coefficients, given f_s and f_c (Matlab or Google)



Outline

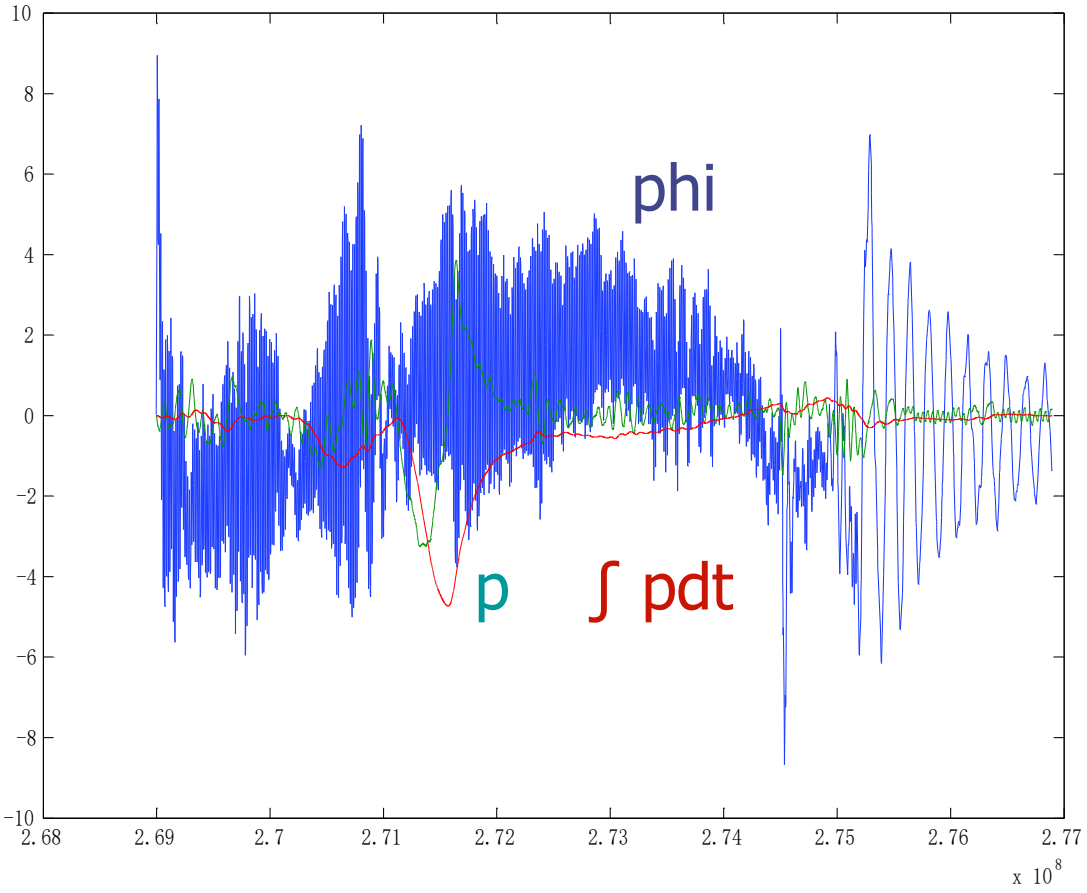
- Introduction
- Z Transform
- Butterworth Filters
- Complementary Filter
- Fixed-point Implementation

Recall: QR Sensor Signals phi, p

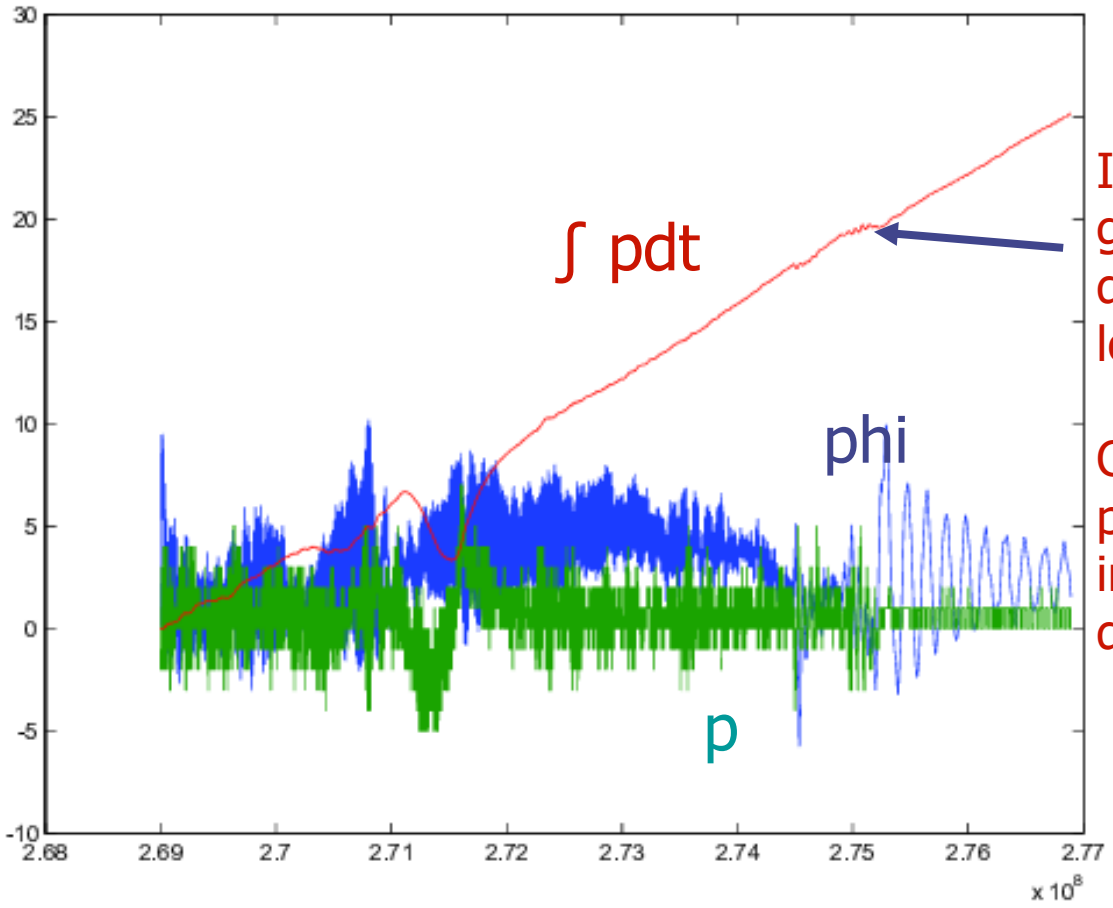


- phi is the plot of Acc
  - Roll angle
- p is the roll rate
- red is the
  - Integral of roll rate = roll angle

After 2nd-order Low-pass (10Hz)



Bias in p: Integration drift in phi



Integration of the readings from the gyro is much clean but will lead to drift in the angle estimation over long-term.

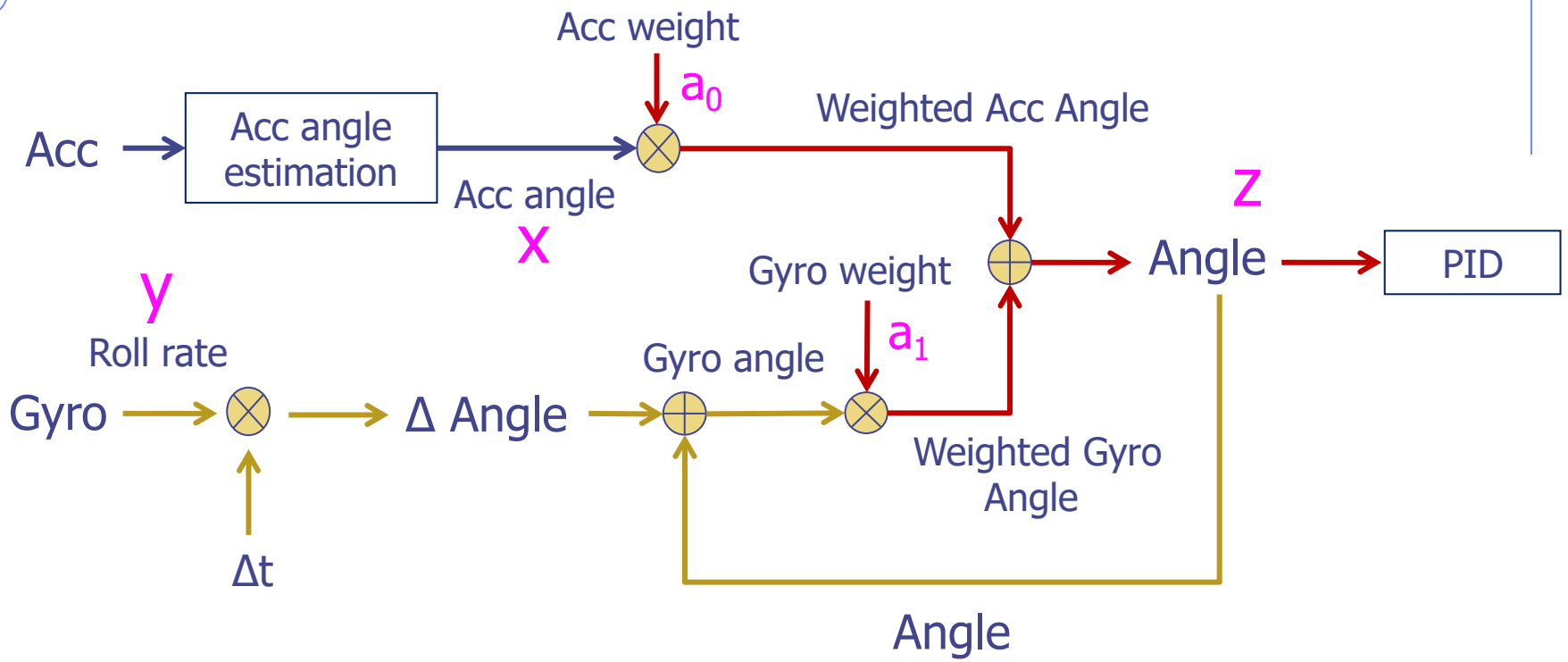
Calibrations may not solve the problem all the time, as the bias instability in gyro can lead to different drifts (e.g., temperature)

Problem Analysis

- Noise is still considerable (in acc)
- Still little correlation between (filtered) phi and p
- More aggressive filtering -> more phase delay
- Acc: negligible drift, too high noise
- Gyro: low noise, potential drift

- Combine the best of both worlds:
  - Using fast and less noisy sensor in the short term
  - Combine with noisy but more reliable sensor in the long run

Complementary Filter



$$z[n] = a_0 * x[n] + a_1 * (z[n-1] + y[n] * \Delta t)$$