

# Finite Word Length Effects in Digital Filters: A Tutorial Review

Peter Koch, Assoc. Professor, [pk@es.aau.dk](mailto:pk@es.aau.dk)

Dept. Electronic Systems, Aalborg University, Denmark

International Symposium on Wireless Personal  
Multimedia Communication, 5G Way Forward to 6G  
Aarhus University, Herning, Denmark  
October 30 – November 2, 2022.



AALBORG  
UNIVERSITY



# Overview of the presentation

- Why fixed-point is (still) an extremely useful implementation paradigm
- 2's complement number representation
- Truncation and Rounding – a linear noise model of the variables in the filter
- The noise model applied in Linear Time Invariant filters, and output SNR
- Filter coefficients in fixed point number representation – both for IIR and for FIR filters
- Scaling in recursive filters – the compromise between overflow and SNR
- Noise optimal filter structures – Direct Canonic Form and the State Space Form

So, for some of you it will be a **walk down memory lane** – for others, it may be a **valuable extension** to what you have already studied in terms of digital filters and how they are efficiently prepared for real-time implementation.

# Floating-point is easily available these days, so why bother about fixed-point...?

Digital hardware is becoming the **primary means** by which control systems and **signal processing filters are implemented**. Digital hardware can be classified as either **off-the-shelf hardware** (for example, microcontrollers, microprocessors, general-purpose processors, and digital signal processors) or **custom hardware**. Within these two types of hardware, there are **many architecture designs**. These **designs range from** systems with a single instruction, single data stream processing unit to systems with multiple instruction, multiple data stream processing units.

Within digital hardware, numbers are represented as either **fixed-point or floating-point data types**. For both these data types, **word sizes are fixed** at a set number of bits. However, the **dynamic range** of fixed-point values is much less than floating-point values with equivalent word sizes. Therefore, in order to avoid overflow or unreasonable quantization errors, fixed-point values must be scaled. **Since floating-point hardware can greatly simplify the real-time implementation of a control or signal processing systems, and floating-point numbers can effectively approximate real-world numbers, then why should we bother about processors or custom hardware solutions with fixed-point hardware support?**

# Floating-point is easily available these days, so why bother about fixed-point...?

**Size and Power Consumption** — The logic circuits of fixed-point hardware are much less complicated than those of floating-point hardware. This means that the **fixed-point chip size is significantly smaller and with less power consumption** when compared with floating-point hardware. One of the major design goals in portable devices is to achieve an extended battery life-time which is therefore best accommodated using fixed-point hardware.

**Memory Usage** — In general, fixed-point calculations require **less memory** because in floating-point numbers more bits are needed to hold information on both mantissa and exponent.

**Speed** — The significantly larger complexity of floating-point circuitry, as compared to similar fixed-point counterparts, enables a **lower propagation delay** and thus a shorter execution time.

AI can solve so many things by today – however, the **computational complexity is still so large** (even with efficient pruning of the network) that real-time execution on portable battery-powered platforms is not really possible.

A prominent example of a challenge application domain is **Hearing Assistive Devices**.

# Floating-point is easily available these days, so why bother about fixed-point...?

**Cost** — Fixed-point hardware is more effective when cost is an important consideration – there are simply fewer transistors in the circuit, thus leading to a smaller die size which is normally equivalent to lower cost.

There is one **important downside** though, which relates to the fact that **designing fixed-point algorithms** is a significantly more complicated task as compared to similar floating-point based algorithms. This fact essentially has two major consequences;

- 1) The designer must have an **extended mathematical knowledge** about the numerical characteristics of the algorithms, and
- 2) The **development time is in some cases longer** than for equivalent floating-point systems.

**THEREFORE;** Today we will survey some of the most important and useful topics related to the design and analysis of fixed-point recursive digital filters.

# A few words on 2's complement numbers

2's complement is **by far the most applied** fixed-point number representation in real-time signal processing.

$x = X_m(-b_0 + \sum_{i=1}^{\infty} b_i \cdot 2^{-i})$  where  $X_m$  is a scaling factor determining the dynamic range, and  $b_j$  are the individual bits in the number  $x$ , (normally also referred to as the word). The bit  $b_0$  is the **sign bit**.

With an upper limit equal to  $\infty$ , the number  $x$  is be represented with with infinite accuracy.

This however, is not practically feasible, and thus we **restrict the upper limit to  $B - 1$  bits**, i.e.;

$$\hat{x} = Q_B[x] = X_m(-b_0 + \sum_{i=1}^{B-1} b_i \cdot 2^{-i})$$



Fixed decimal point (can be shifter right, but for now we assume this position).

Sign bit;  $x$  is positive for  $b_0 = 0$  and negative for  $b_0 = 1$ .

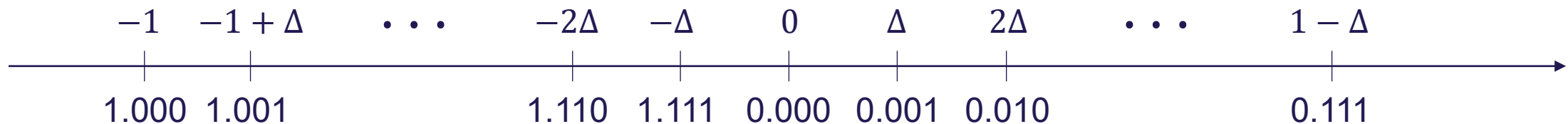
# A few words on 2's complement numbers



The word consists of  $B$  bits, i.e.,  $2^B$  different representations. The numerical distance between two adjacent numbers is known as the "quantization step";  $\Delta = X_m \cdot 2^{-(B-1)}$ .

In many practical situations we are interested in  $X_m = 1$ , i.e., the dynamic range is  $[-1; 1[$ .

Here the dynamic range  $[-1; 1[$  is exemplified for a 4 bit 2's complement number.



# 2's complement numbers in digital filters

Real-time implementation of digital filters is (in most cases) done in terms of difference equations;

$$y[n] = \sum_{i=1}^N a_i \cdot y[n - i] + \sum_{j=0}^M b_j \cdot x[n - j]$$

The overall computation here is "sum of products", i.e., we need to conduct multiplication and addition.

In a binary scenario, and thus also for 2's complement numbers, the following characteristics holds;

## Multiplication:

- Multiplying a  $B_1$  bit number with a  $B_2$  bit number leads to a  $B_1 + B_2$  bit product.
- For a dynamic range  $[-1; 1[$ , the product does not exceed this interval.

## Addition:

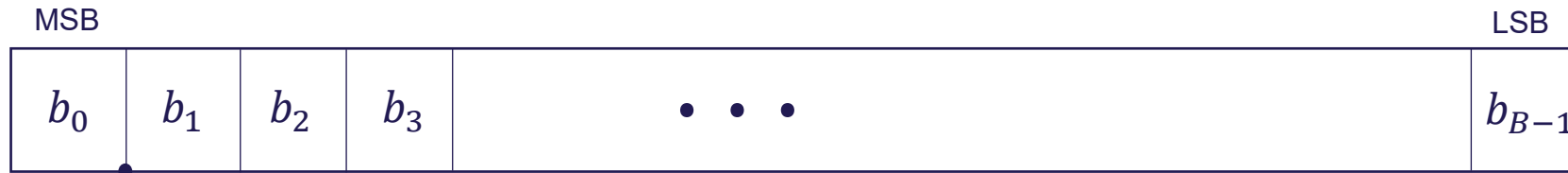
- Adding a  $B_1$  bit number to a  $B_2$  bit number leads to a  $\max\{B_1, B_2\}$  bit sum.
- The sum may exceed the dynamic range, thus potentially leading to overflow.
- Overflow can occur only if the two operands have identical sign.

We now want to discuss how to handle 1) word-length extension, and later on 2) overflow.



# Word-length reduction of binary numbers – Truncation and Rounding

**Truncation** is a quantization process where a  $B_1$  bit number is reduced in **word length** to a  $B$  bit number.



**The error  $e$  introduced** (for 2's complement) is:  $-(2^{-(B-1)} - 2^{-(B_1-1)}) \leq e \leq 0$

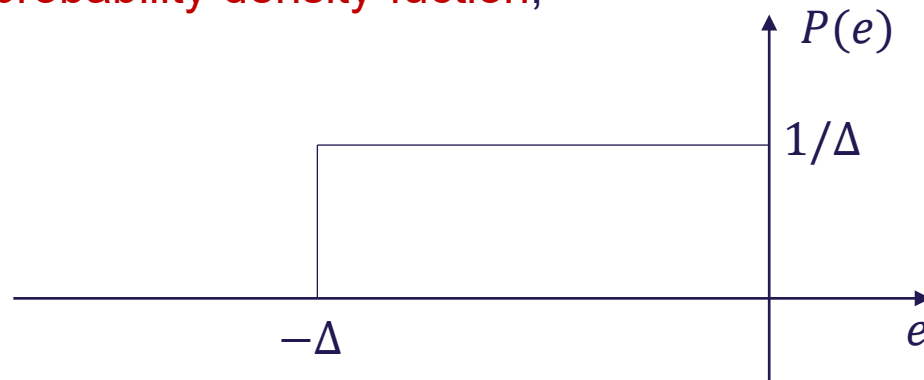
So, the error is always non-positive no matter the the sign of  $x$ .

Similarly, we can state that  $|Q[x]| \geq |x|$  for  $x < 0$ . This observation indicates that **truncation does not lead to "magnitude truncation"** for negative 2's complement numbers.

# Truncation, contd.

- If we assume that  $x$  is a uniform distributed stochastic variable, and
- If  $B$  and  $B_1 - B$  are "sufficiently" large, then

The quantization error  $e$  can be modelled as a stochastic variable with constant, i.e., uniform probability density-fuction;



$$-\Delta \leq e \leq 0 \quad \text{quantization error}$$

$$\mu_e = -\Delta/2 \quad \text{mean}$$

$$\sigma_e^2 = \Delta^2/12 \quad \text{variance}$$

In a digital filter, where an internal variable is truncated (using truncation arithmetic), the quantization noise  $e$  can be observed directly on the output of the filter, and therefore we conclude;

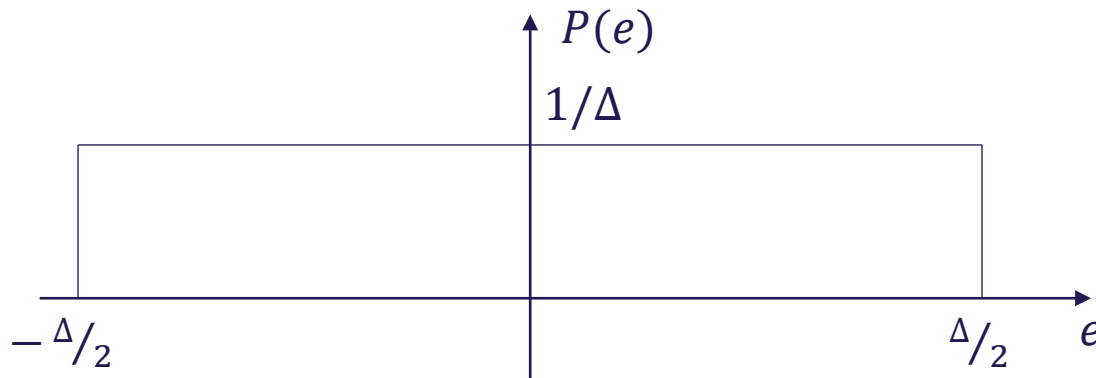
At the filter output, the mean of the quantization noise is "non-zero", and therefore **truncation leads to a bias of the output signal.**

# Rounding

**Rounding** is a quantization mechanism where the word-length reduction from  $B_1$  to  $B$  bits is implemented by choosing the most "near by"  $B$  bit number.



The **quantization error  $e$**  is thus;  $-\Delta/2 \leq e \leq \Delta/2$  where the sign of  $e$  is un-correlated with  $x$ .

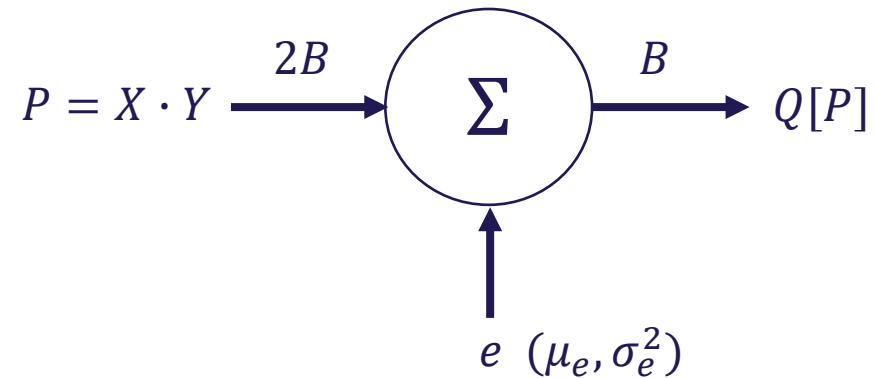
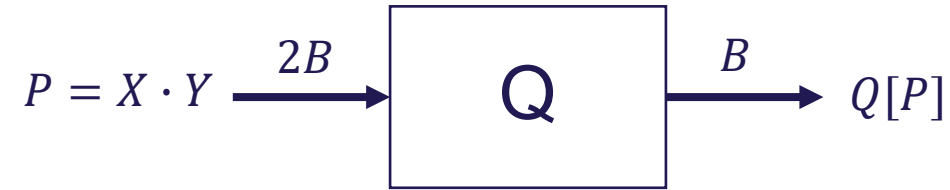


$$\mu_e = 0 \quad \text{mean}$$

$$\sigma_e^2 = \Delta^2/12 \quad \text{variance}$$

Here it is worth noting that  $|Q[x]|$  not necessarily is less than  $|x|$  - rounding may increase the numerical value.

# Linear noise model



Onwards, we will assume that the **quantizer  $Q$**  is implemented using **rounding arithmetic**.



# So, multiplication is a noisy operation

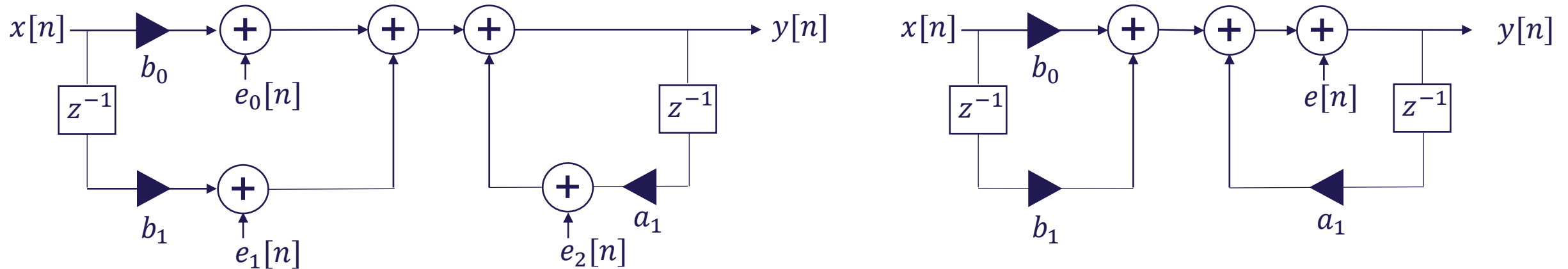
As related to the difference equation,  $y[n] = \sum_{i=1}^N a_i \cdot y[n - i] + \sum_{j=0}^M b_j \cdot x[n - j]$ , the sum-of-products operations can be done as either "single precision" or "double precision".

- **Single precision:** The quantization to  $B$  bit is conducted immediately **after each multiplication**, and thus the addition is performed in  $B$  bit.
- **Double precision:** The quantization to  $B$  bit is conducted **only after the addition**, which can then be performed in  $2B$  bit.
- Choosing Single vs. Double precision is a **trade-off** between accuracy, execution time, power consumption, and physical size of the circuit.

# An example

## 1st order IIR filter, $B$ bit single precision

$$y[n] = \sum_{i=1}^1 a_i \cdot y[n-i] + \sum_{j=0}^1 b_j \cdot x[n-j]$$



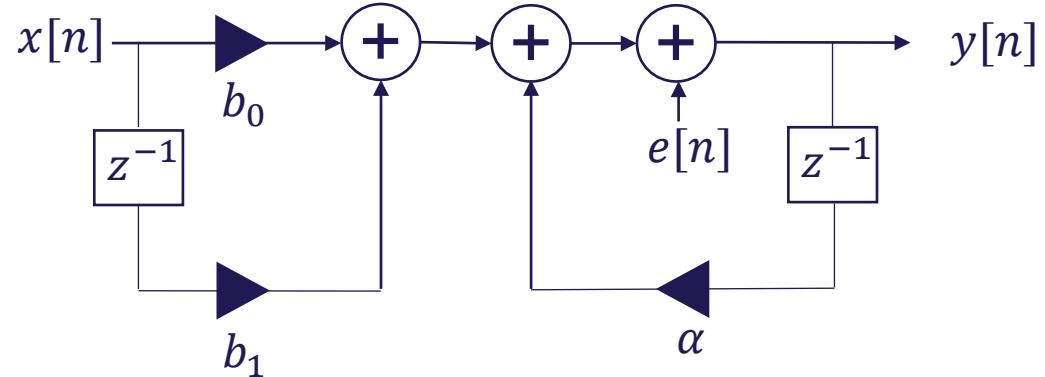
Since the system is LTI, the three noise sources  $e_k[n]$  can propagate through the network and act as a single noise source  $e[n] = \sum_{k=1}^3 e_k[n]$  directly at the output.

From this we observe that the **Signal-to-Noise-Ratio (SNR), at the output is not infinite.**

# Assumption concerning the noise $e[n]$

- $e[n]$  is a **Wide-Sense Stationary (WSS) random process**, i.e., mean and autocovariance of  $e[n]$  are independent of  $n$ .
- $e[n]$  is **"white"**, i.e., the autocorrelation  $r_{ee}[m] = E\{e[n] \cdot e[n + m]\} = \sigma_e^2 \delta[n] + \mu_e^2$ .
- $e[n]$  is **un-correlated** with other signals in the filter, i.e., input, output, and internal variables.
- $e[n]$  is **uniform distributed** over one quantization step  $\Delta$ .

# The filter also processes the noise



The signal  $e[n]$  passes through the filter, similarly as the signal  $x[n]$  does. In the 1<sup>st</sup> order IIR example, the noise attack at the output, and thus it is only the recursive part of the filter, which modify the noise.

This can be analyzed in terms of the noise transfer function;

$$\frac{Y(z)}{E(z)} = \frac{1}{1 - \alpha z^{-1}} \quad \text{given that } x[n] = 0.$$



# The noise transfer function

$$\frac{Y(z)}{E(z)} = G(z) = \frac{1}{1 - \alpha z^{-1}}$$

Conducting an inverse z-transform, we derive the expression;

$g[n] = \alpha^n \cdot u[n]$  which is the **impulse response** from the point where the noise attacks and to the output.

Knowing the impulse response, we can now specify **the noise contribution at the output**;

$$y_e[n] = \sum_{m=0}^{\infty} g[m] \cdot e[n - m]$$

This signal is also a WWS process, and is characterized by a **mean** and a **variance**;

$$\mu_{y_e} = \mu_e \cdot \sum_{n=0}^{\infty} g[n] \quad \text{and} \quad \sigma_{y_e}^2 = \sigma_e^2 \cdot \sum_{n=0}^{\infty} g^2[n] \quad (\text{see next page}).$$

Note that  $y_e[n]$  is not a "white" signal, but rather a signal which is colored through the filter.

# The noise variance on the output

$$y_e[n] = \sum_{m=0}^{\infty} g[m] \cdot e[n - m]$$

Using this expression for the noise at the output, we can calculate the corresponding variance;

$$\begin{aligned}\sigma_{y_e}^2 &= E\{(\sum_{m=0}^{\infty} g[m] \cdot e[n - m]) \cdot (\sum_{l=0}^{\infty} g[l] \cdot e[n - l])\} \\ &= \sum_{m=0}^{\infty} \sum_{l=0}^{\infty} g[m] \cdot g[l] \cdot E\{e[n - m] \cdot e[n - l]\} \\ &= \sum_{m=0}^{\infty} \sum_{l=0}^{\infty} g[m] \cdot g[l] \cdot \sigma_e^2 \cdot \delta[l - m] \quad (\text{because } e[n] \text{ is a white signal with variance } \sigma_e^2) \\ &= \sigma_e^2 \cdot \sum_{m=0}^{\infty} g^2[m]\end{aligned}$$

So, basically what this result tells us is, that the **noise variance at the output** is equal to the **noise variance multiplied with the squared and summed impulse response** from the point where the noise attacks and to the output.

This is a general result which describes **how a signal couples through an LTI system**.

Since we know the impulse response, we can now calculate the noise variance.

# Noise variance at the output

$$\sigma_{y_e}^2 = \sigma_e^2 \cdot \sum_{n=0}^{\infty} g^2[n]$$

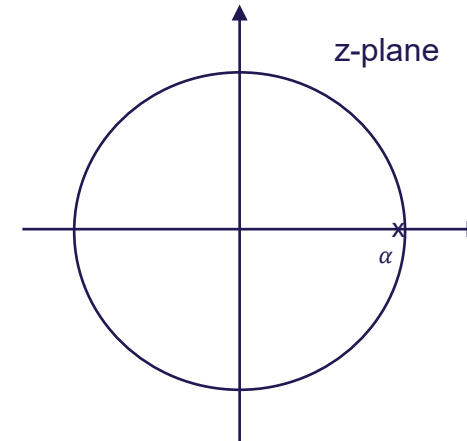
$$\sigma_{y_e}^2 = \sigma_e^2 \cdot \sum_{n=0}^{\infty} (\alpha^n \cdot u[n])^2 \quad \text{Assuming that the filter is stable, then } |\alpha| < 1 \text{ and thus we can write;}$$

$$\sigma_{y_e}^2 = \sigma_e^2 \cdot \sum_{n=0}^{\infty} (\alpha^2)^n \quad \text{which is a } \text{geometric series}, \text{ converging towards}$$

$$\sigma_{y_e}^2 = \sigma_e^2 \cdot \frac{1}{1-\alpha^2}$$

So, in conclusion we see that the **noise variance on the filter output is a function of the pole location.**

# An example



Given a narrow-band LP-filter with  $\alpha = 0.997$ .

$$\sigma_{y_e}^2 = \sigma_e^2 \cdot \frac{1}{1-\alpha^2} = 167 \cdot \sigma_e^2$$

Assuming a noise variance  $\sigma_e^2 = 1$ , the corresponding variance at the output (in  $dB$ ) is;

$$\sigma_{y_e}^2 = 10 \cdot \log(167) = 22 \text{ dB noise gain...!!}$$

We realize that poles located close to the unit circle are "more noisy", as compared to poles located more towards origo.

This is an extremely important realisation.



# Signal to Noise Ratio

The noise variance, **as a stand-alone metric**, is not very useful. Rather, we would like to measure the filter **noise performance in terms of the Signal to Noise Ratio, SNR**.

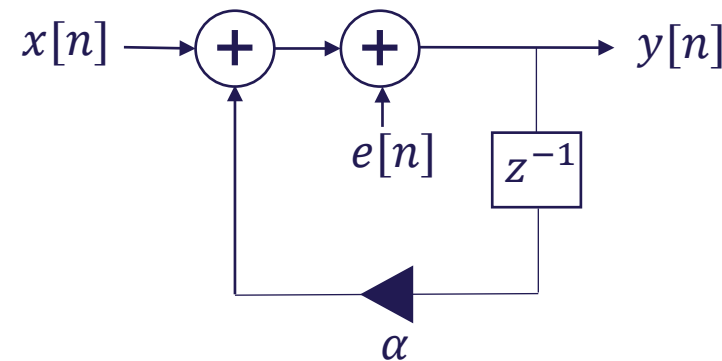
$$SNR \triangleq \frac{\text{Variance of the signal at the output}}{\text{Variance of the noise at the output}}$$

So, let's assume a filter  $H(z) = \frac{Y(z)}{X(z)} = \frac{1}{1-\alpha z^{-1}}$ , i.e., a filter with one pole in  $z = \alpha$  and one zero in  $z = 0$ .

The signal variance at the output is;

$$\sigma_y^2 = \sigma_x^2 \cdot \frac{1}{1-\alpha^2}$$

and thus the SNR is;  $SNR = \frac{\sigma_y^2}{\sigma_{y_e}^2} = \frac{\sigma_x^2 \cdot \frac{1}{1-\alpha^2}}{\sigma_e^2 \cdot \frac{1}{1-\alpha^2}} = \frac{\sigma_x^2}{\sigma_e^2}$



This is an interesting result, since it **indicates** that the SNR at the filter output is **independent** of the pole location  $\alpha$ . This however, is a wrong interpretation – let's argue why...

# Output SNR depends on the pole location

- Let's assume the filter is implemented using fixed-point arithmetic with a **dynamic range**  $[-1; 1[$ .
- Ideally, we want the output signal to **utilize the complete dynamic range**.
- At the same time, we want to **avoid that the output signal variable overflows**.
- These are **conflicting requirements**, and therefore we scale the input signal;  $x'[n] = s \cdot x[n]$  to obtain the best possible compromise between these two demands.
- The numerical value of **the scaling factor  $s$  however, depends on how much the signal  $x$  is being amplified throughout the filter** – and this actually depends on the pole location, i.e., the value  $|\alpha|$ .

Therefore, if  $|\alpha| \rightarrow 1$ , then the **value of  $\frac{1}{1-\alpha^2}$  increases**, which calls for  $s \rightarrow 0$  to reduce the probability for overflow, and thus  **$\sigma_{x'}^2$  decreases**.

Consequently,  **$SNR' = \frac{\sigma_{x'}^2}{\sigma_e^2} \rightarrow 0$  if  $|\alpha| \rightarrow 1$** . We will address in more detail "Scaling" later in the presentation (p.44).

# Considerations on Coefficient Quantization

Given a filter with transfer function  $H(z) = \frac{B(z)}{A(z)} = \frac{\sum_{j=0}^M b_j z^{-j}}{1 + \sum_{i=1}^N a_i z^{-i}}$  where all **filter coefficients** have to be represented using a  $B$  bit 2's complement notation.

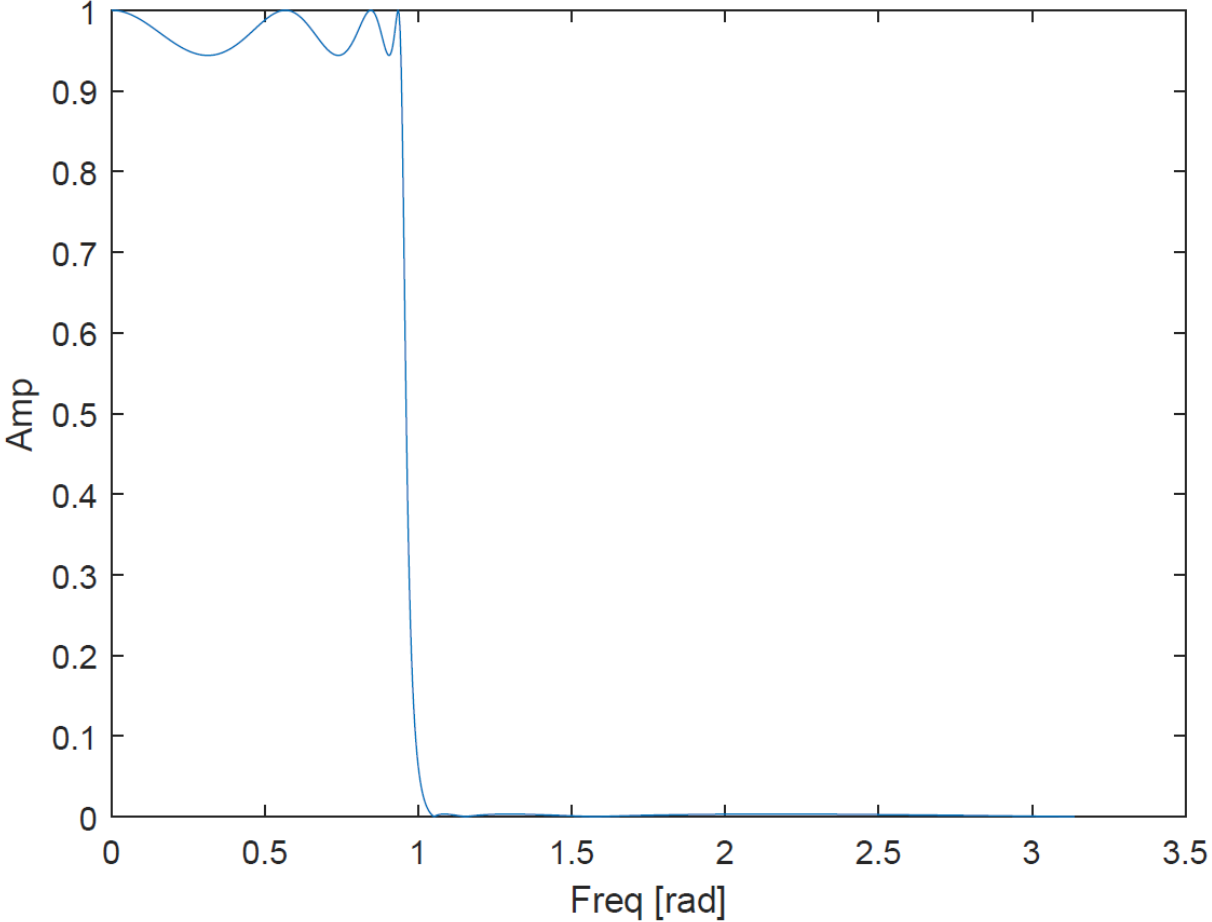
Therefore;  $\hat{b}_j = b_j + \varepsilon_j$  and  $\hat{a}_i = a_i + \varepsilon_i$  where  $\varepsilon$  denotes quantization from floating point to  $B$  bit fixed-point.

Now, the transfer function to be used for real-time implementation thus is  $\hat{H}(z) = \frac{\hat{B}(z)}{\hat{A}(z)} = \frac{\sum_{j=0}^M \hat{b}_j z^{-j}}{1 + \sum_{i=1}^N \hat{a}_i z^{-i}}$

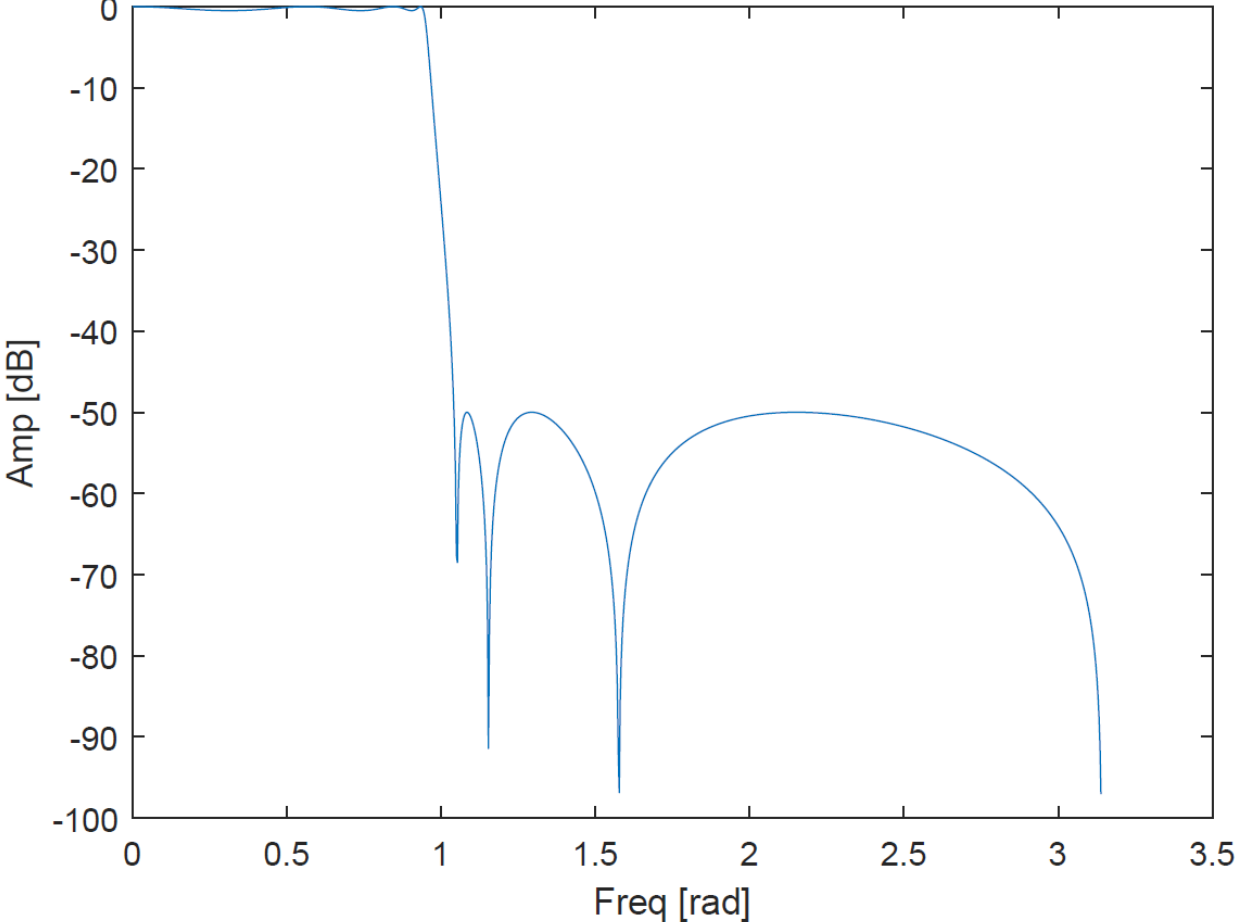
We will briefly illustrate what might be the effect of quantizing the filter coefficients...

**An example –  
preparation of a 7th order elliptic LP-filter with  $\omega_c = 0.3\pi$ ,  
bandpass ripple  $0.5\text{dB}$ , and stopband attenuation  $50\text{dB}$ .**

**7th order Elliptic LP-filter  
 $f_c = 0.3\pi$ , ripple=0.5dB, att.=50dB**



**7th order Elliptic LP-filter  
 $f_c = 0.3\pi$ , ripple=0.5dB, att.=50dB**



# The filter coefficients – with 15 decimals

$$H(z) = \frac{B(z)}{A(z)} = \frac{\sum_{j=0}^M b_j z^{-j}}{1 + \sum_{i=1}^N a_i z^{-i}}$$

b\_unquantized =

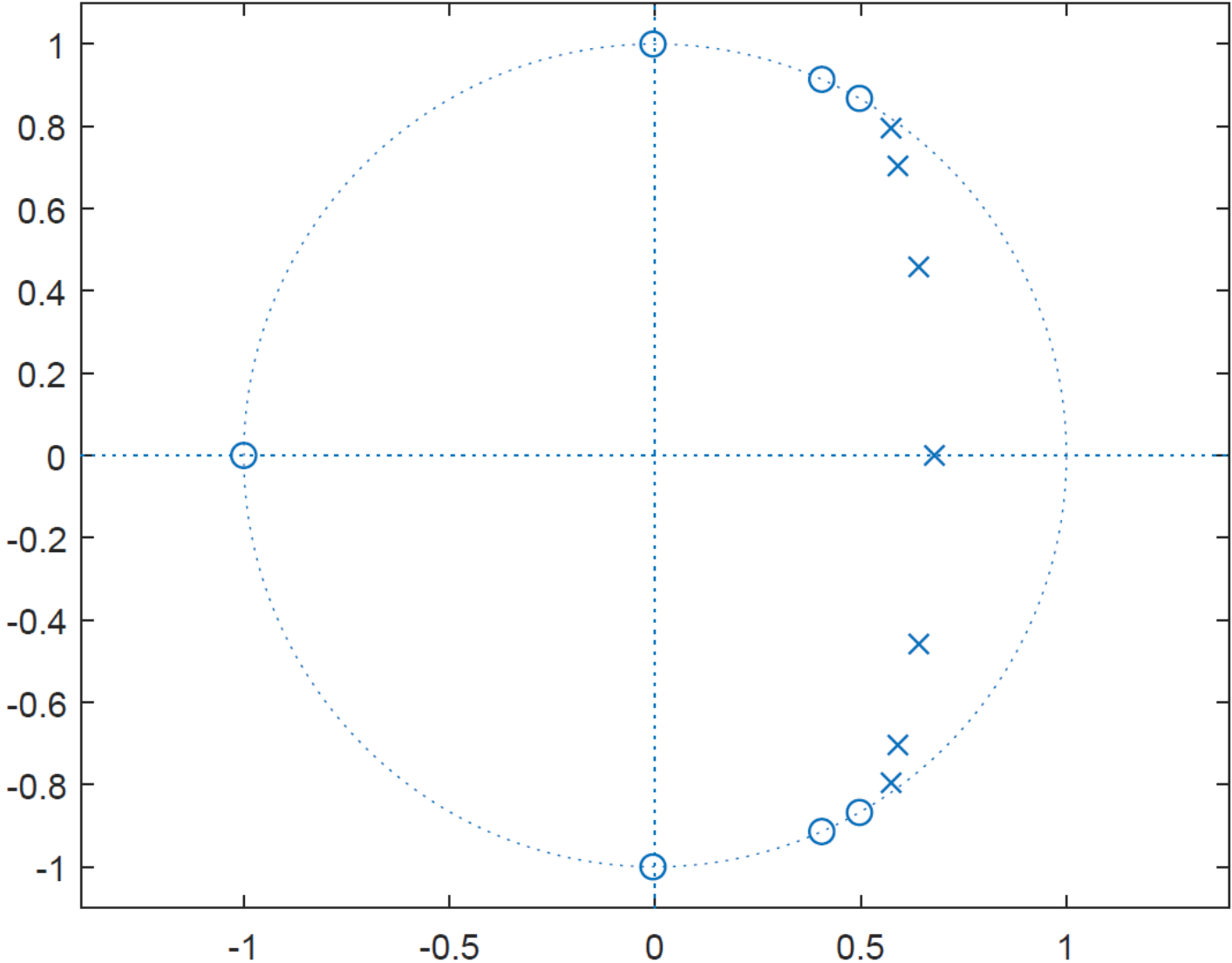
0.012218357882143  
-0.009700754662078  
0.024350450826845  
0.002532504848041  
0.002532504848041  
0.024350450826845  
-0.009700754662078  
0.012218357882143

a\_unquantized =

1.000000000000000  
-4.288900601525730  
9.216957436091187  
-12.195350561406688  
10.633166152311439  
-6.062798190498842  
2.098067018562065  
-0.342340135743531

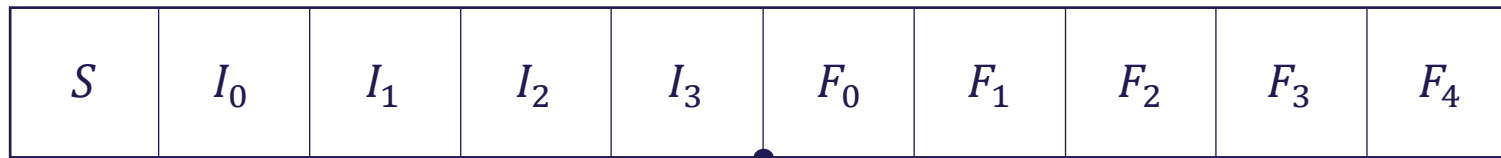
- Note the symmetry in the  $B$  coefficients
- Note "how small" the  $B$  coefficients actually are
- Note "how large" the  $A$  coefficients actually are

# Pole-Zero locations, un-quantized filter



# Map the filter onto a 10 bit 2's complement architecture

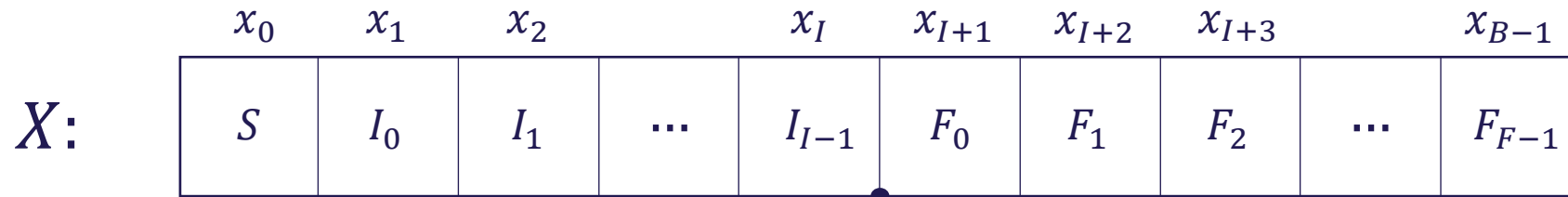
- The dynamic range exceeds  $[-1; 1[$
- The numerically largest coefficient is  $a_3 = -12.195350561406688$
- We need to interpret the 2's complement number a little different.



- $S$  is the sign bit
- $I_i$  is the integer part;  $2^4 = 16 > | - 12.195 \dots |$
- $F_i$  is the fractional part

This is known as **Q5.5 format** (some literatur denote it Q4.5)

# General interpretation of $Q(I + 1).F$



- $B$  bit word
- $I$  integer bits
- $F$  fractional bits

$$X = -2^I \cdot x_0 + \sum_{j=1}^I x_j \cdot 2^{(I-j)} + \sum_{j=I+1}^{B-1} x_j \cdot 2^{-(j-I)}$$

Now – back to our example, where we use **Q5.5 for all coefficients...**



# All coefficients quantized to Q5.5

$$\hat{b}_k = \frac{\text{round}(b_k \cdot 2^5)}{2^5}$$

b\_quantized\_Q55=

0.0000000000000000  
0.0000000000000000  
0.0312500000000000  
0.0000000000000000  
0.0000000000000000  
0.0312500000000000  
0.0000000000000000  
0.0000000000000000

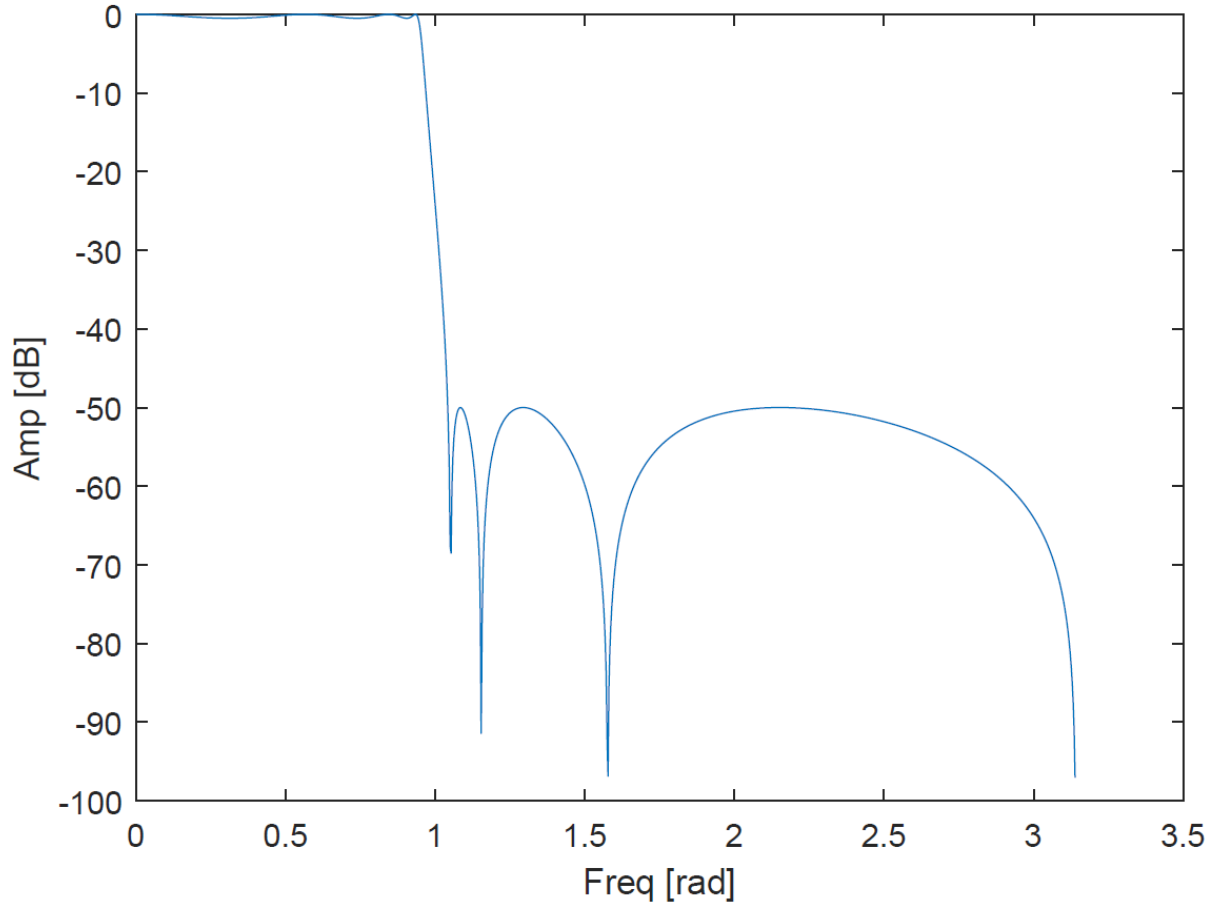
a\_quantized\_Q55=

1.0000000000000000  
-4.2812500000000000  
9.2187500000000000  
-12.1875000000000000  
10.6250000000000000  
-6.0625000000000000  
2.0937500000000000  
-0.3437500000000000

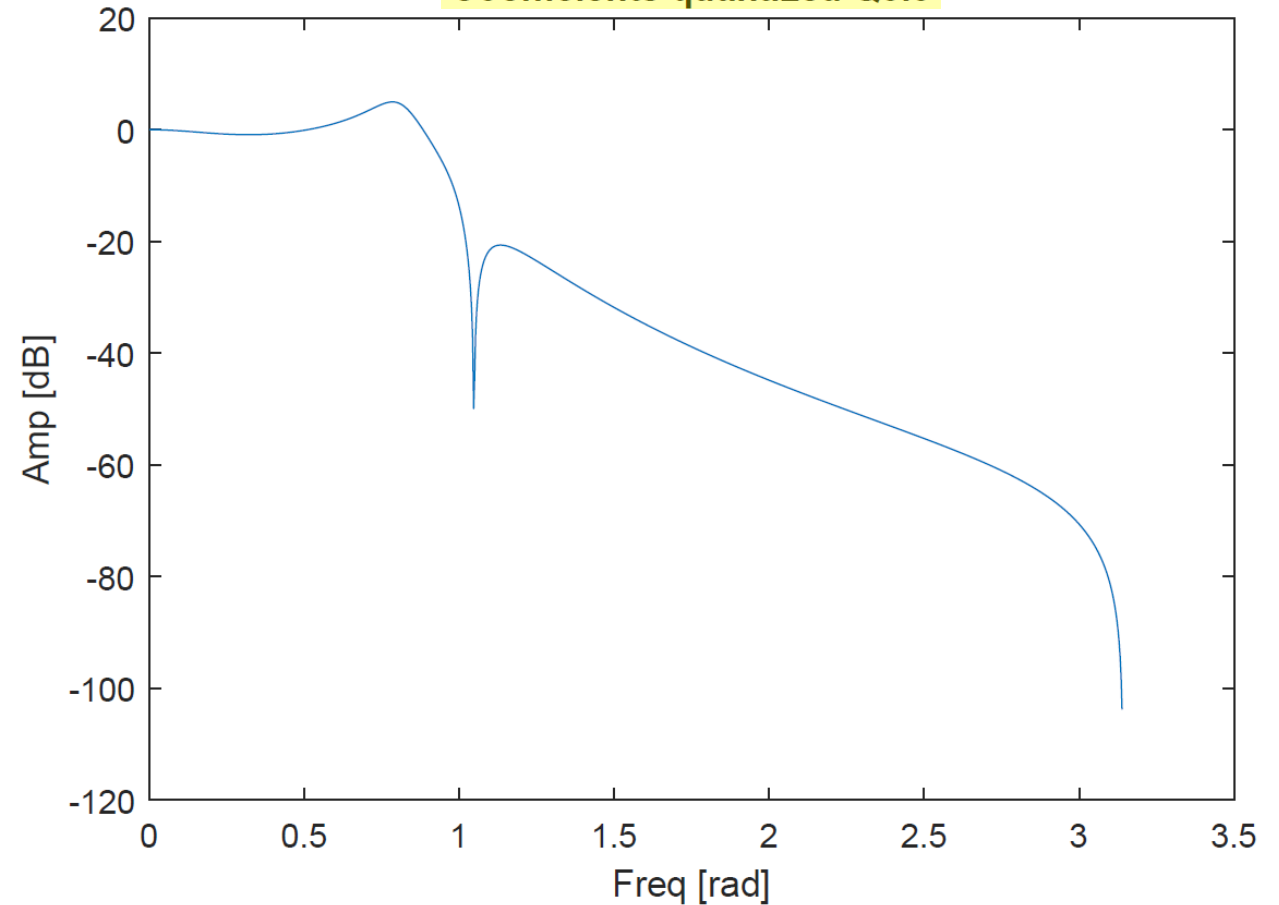
- There might be a challenge here... If  $b_k \cdot 2^5 < 0.5$  then  $\text{round}(b_k \cdot 2^5) = 0$ . True for most  $b_k$  coefficients.
- Note that with only 5 fractional bits, there are similarly **only 5 significant decimals**.
- Also note that  $B(z)$  has been reduced to a 5<sup>th</sup> order polynomial.

# Amplitude response for the Q5.5 filter

7th order Elliptic LP-filter  
 $f_c = 0.3\pi$ , ripple=0.5dB, att.=50dB

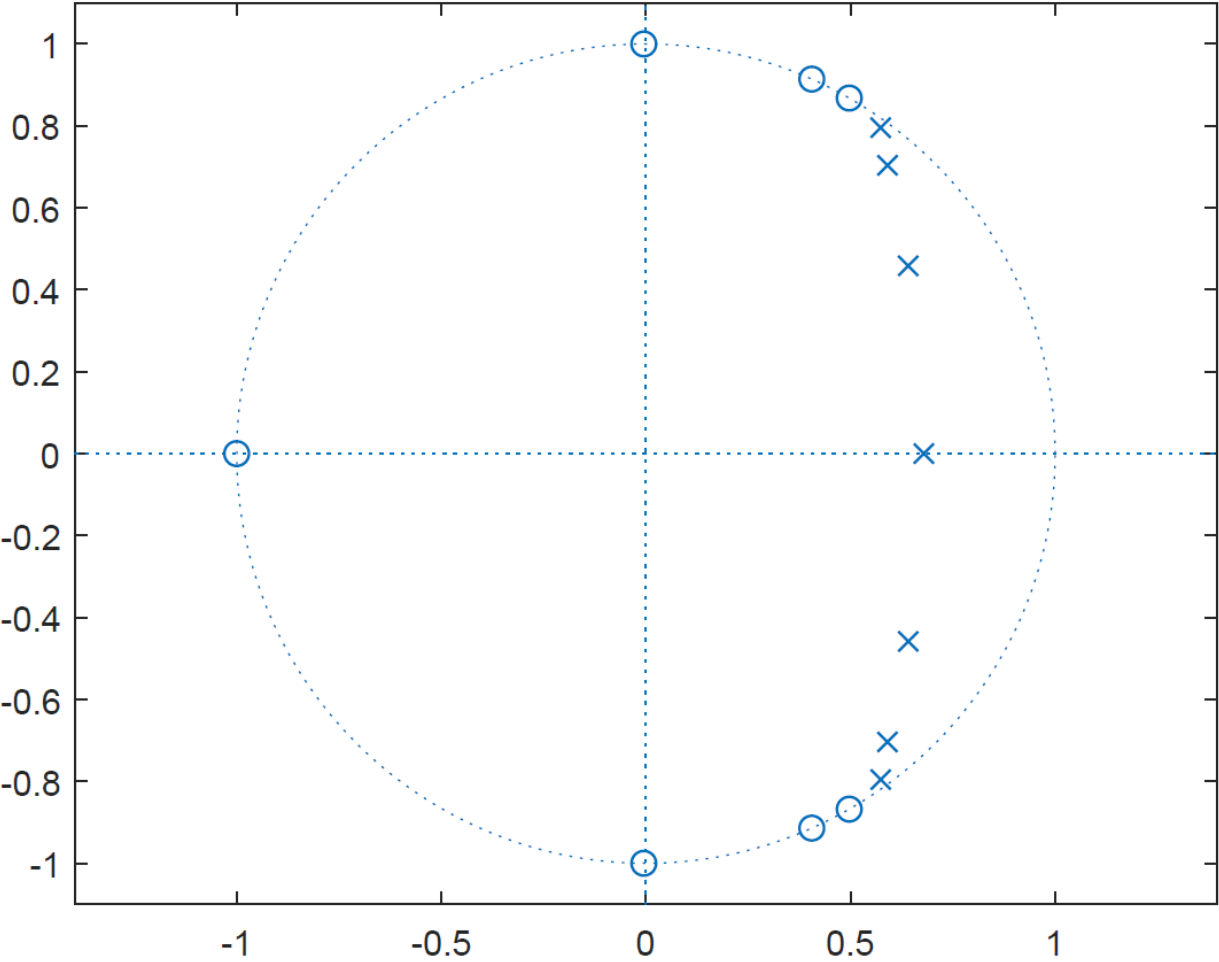


7th order Elliptic LP-filter  
 $f_c = 0.3\pi$ , ripple=0.5dB, att.=50dB  
Coefficients quantized Q5.5

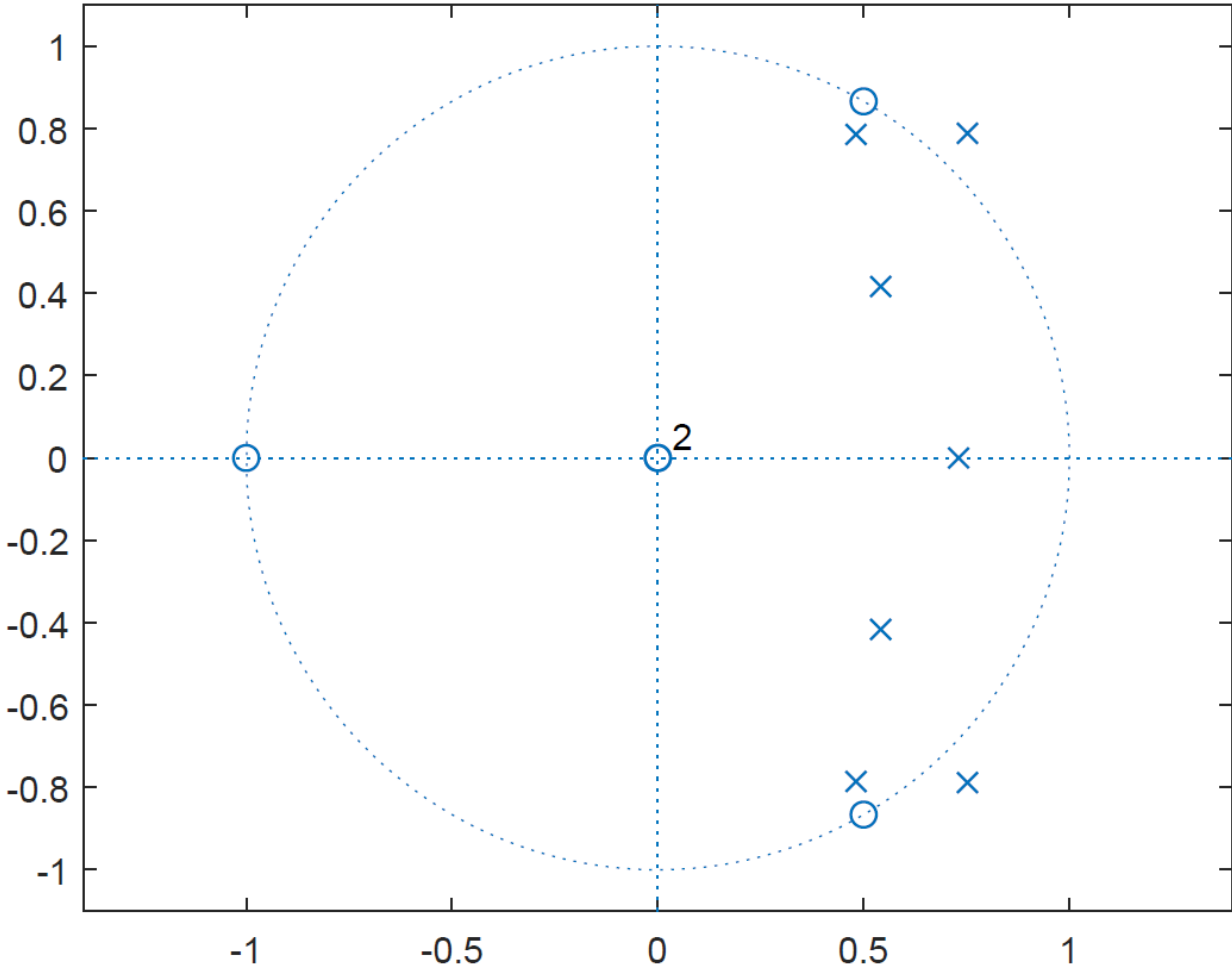


# Pole-Zero locations for the $Q5.5$ filter

Before quantization



After quantization



What is the cure...?

# Better utilization of the dynamic range

b\_unquantized =

0.012218357882143  
-0.009700754662078  
0.024350450826845  
0.002532504848041  
0.002532504848041  
0.024350450826845  
-0.009700754662078  
0.012218357882143

All un-quantized  $b_k$  coefficients are numerically less than 1.

For a 10 bit architecture, we therefore could suggest to use  $Q1.9$  for the  $B$  coefficients, i.e.,

$$\hat{b}_k = \frac{\text{round}(b_k \cdot 2^9)}{2^9}$$

Now, since all  $B$  coefficients are less than  $2^{-5} = 0.03125$ , then we could scale all coefficients with  $2^5$ , i.e.,  $|b'_k| = |b_k \cdot 2^5| < 1$ .

$$\hat{b}'_k = \frac{\text{round}(b'_k \cdot 2^9)}{2^9} \quad \text{thus utilizing the full dynamic range, however...}$$

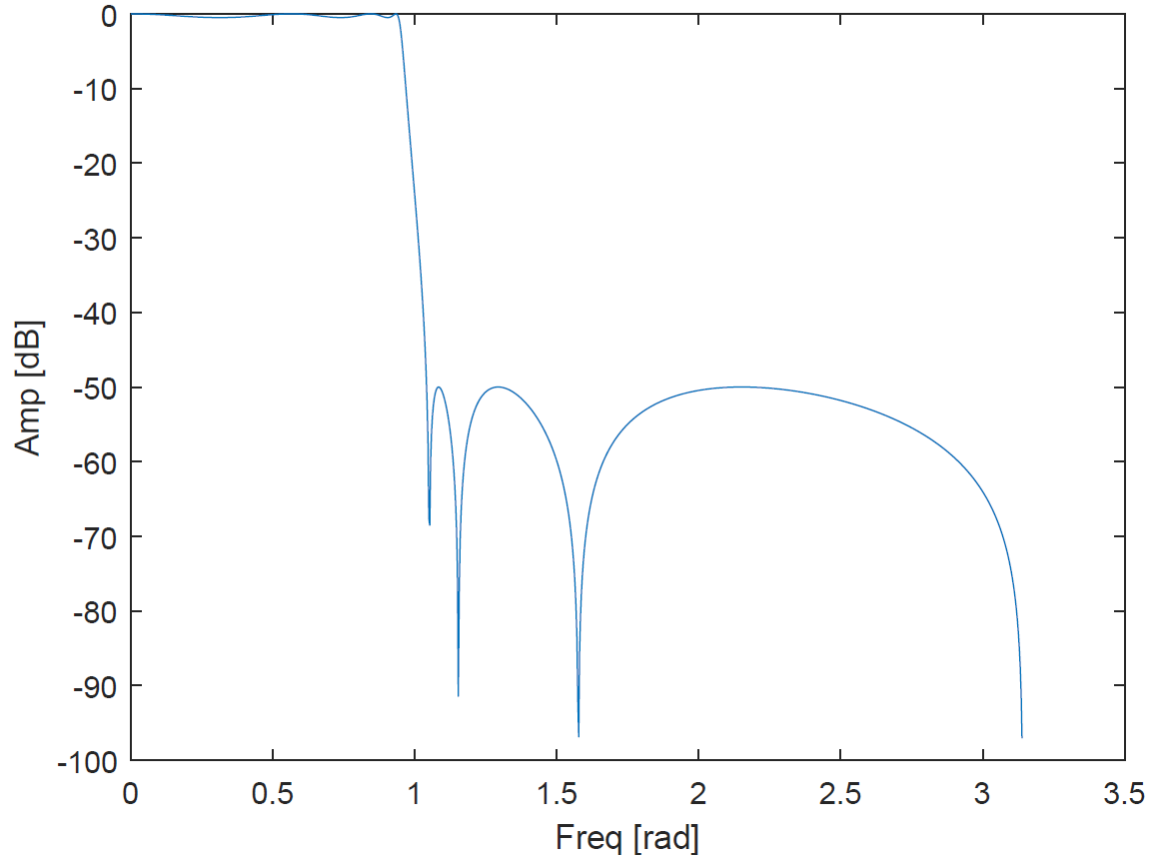
$$H(z) = \frac{B(z)}{A(z)} \begin{array}{l} \leftarrow \text{Using } Q1.9 \\ \leftarrow \text{Using } Q5.5 \end{array}$$

This is a problem – which can be alleviated by re-scaling the scaled and rounded coefficients.

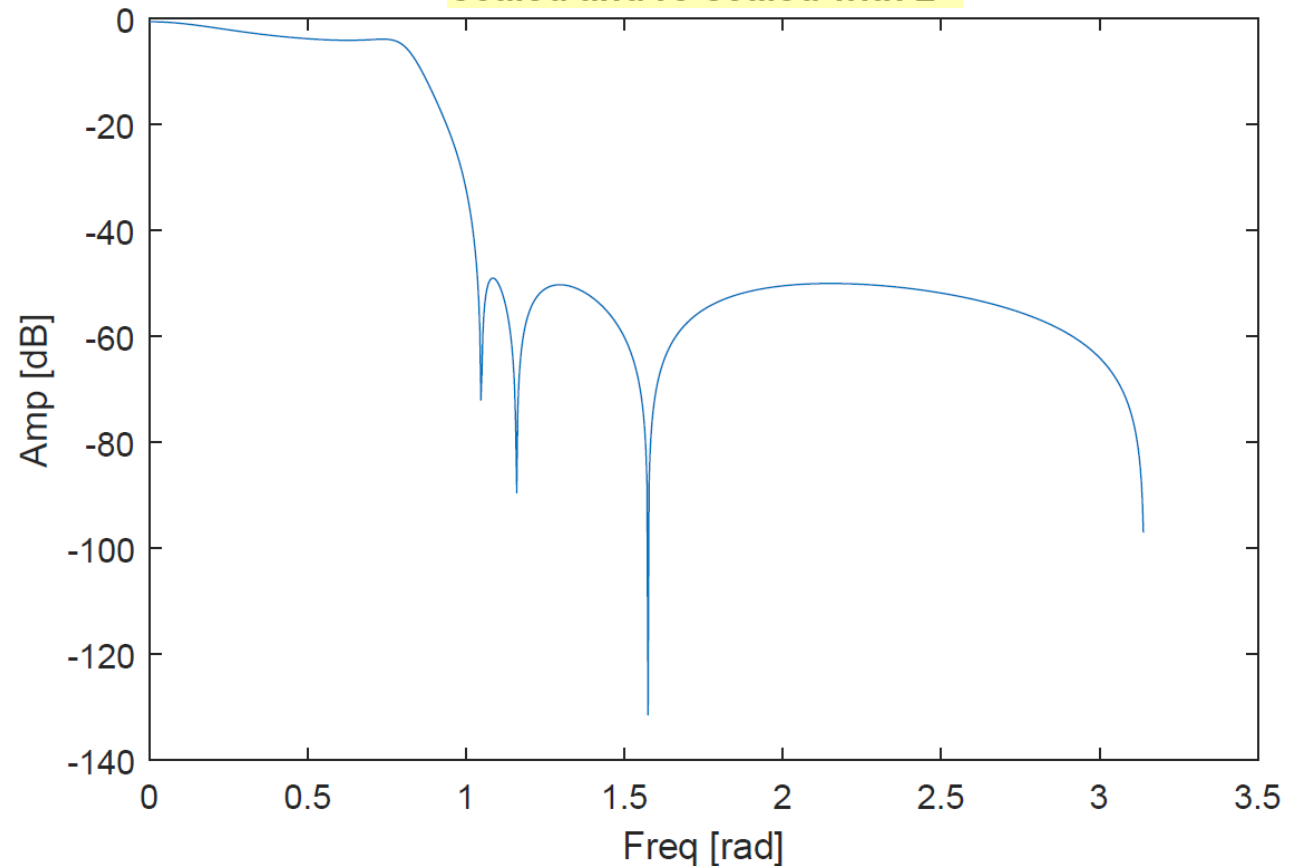
# Re-scaling the scaled and rounded coef.

$$\hat{b}'_k = \frac{\text{round}(b'_k \cdot 2^9)}{2^9} \quad \text{and thus} \quad \hat{b}_k = 2^{-5} \cdot \hat{b}'_k = 2^{-5} \frac{\text{round}(b'_k \cdot 2^9)}{2^9} = \frac{\text{round}(b'_k \cdot 2^9)}{2^{9+5}}$$

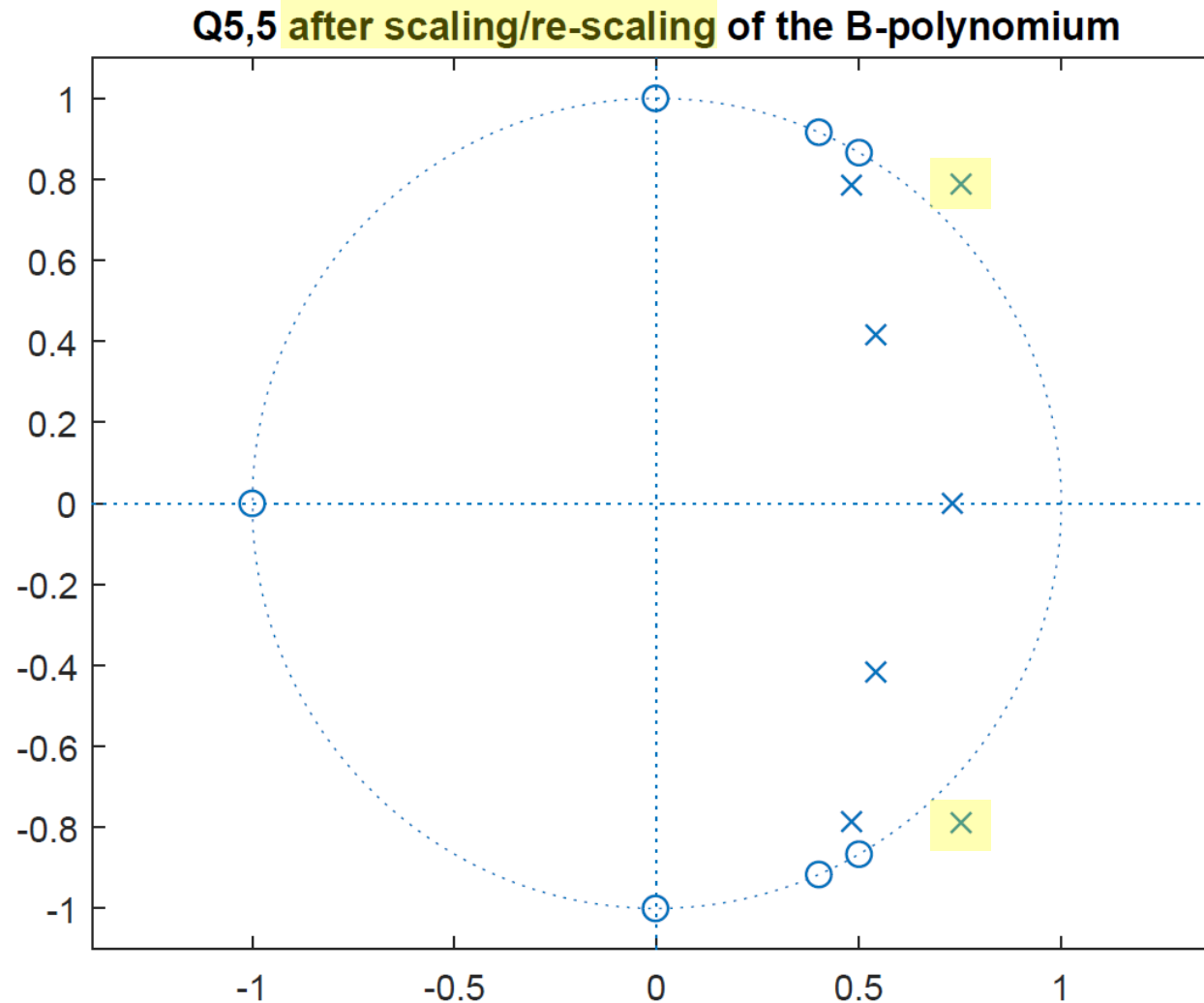
7th order Elliptic LP-filter  
 $f_c = 0.3 \cdot \pi$ , ripple=0.5dB, att.=50dB



7th order Elliptic LP-filter  
 $f_c = 0.3 \cdot \pi$ , ripple=0.5dB, att.=50dB  
Quantized Q5,5 after b-coef. are  
scaled and re-scaled with  $2^5$



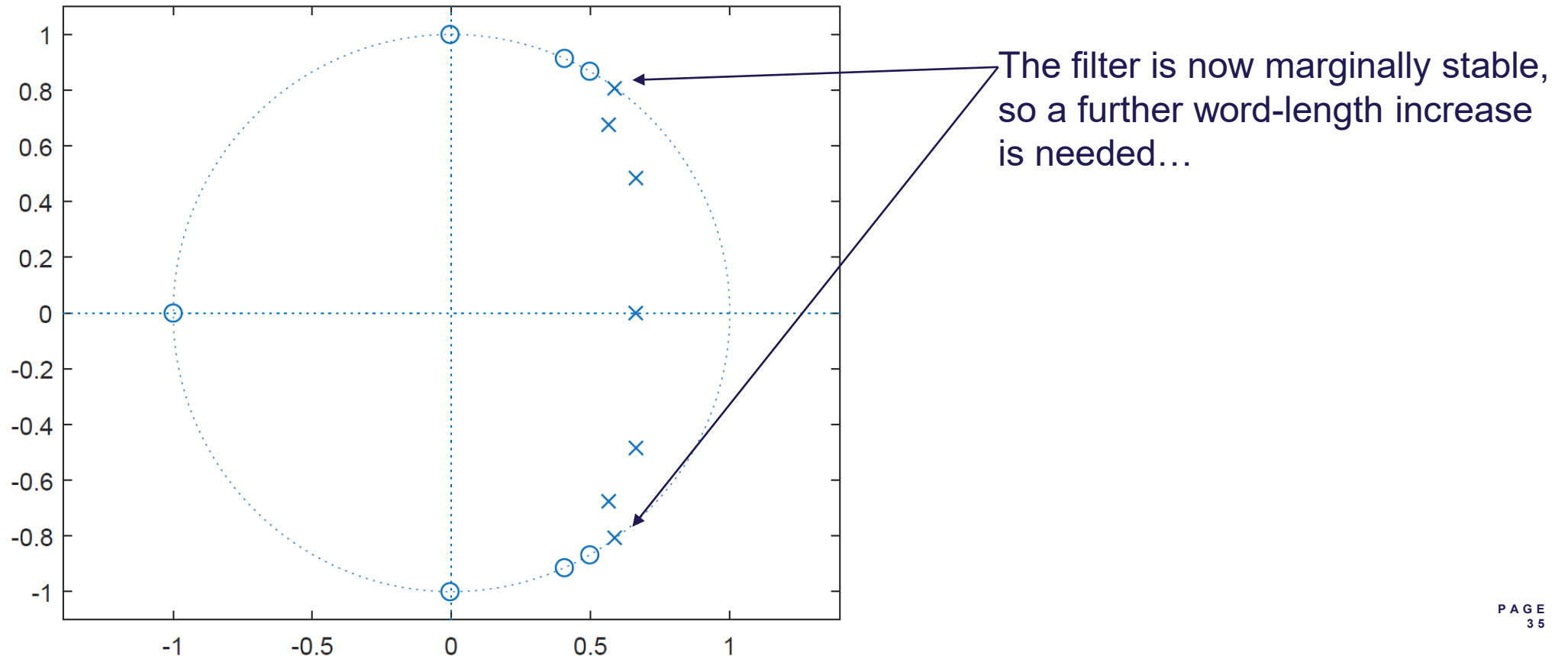
# How to solve the pole problem..?



# How to solve the pole problem..?

Since the  $A$  coefficients represent the **recursive part** of the filter, there are **no other options** than **experimenting with increased word-length** – both for the  $A$  and the  $B$  polynomial.

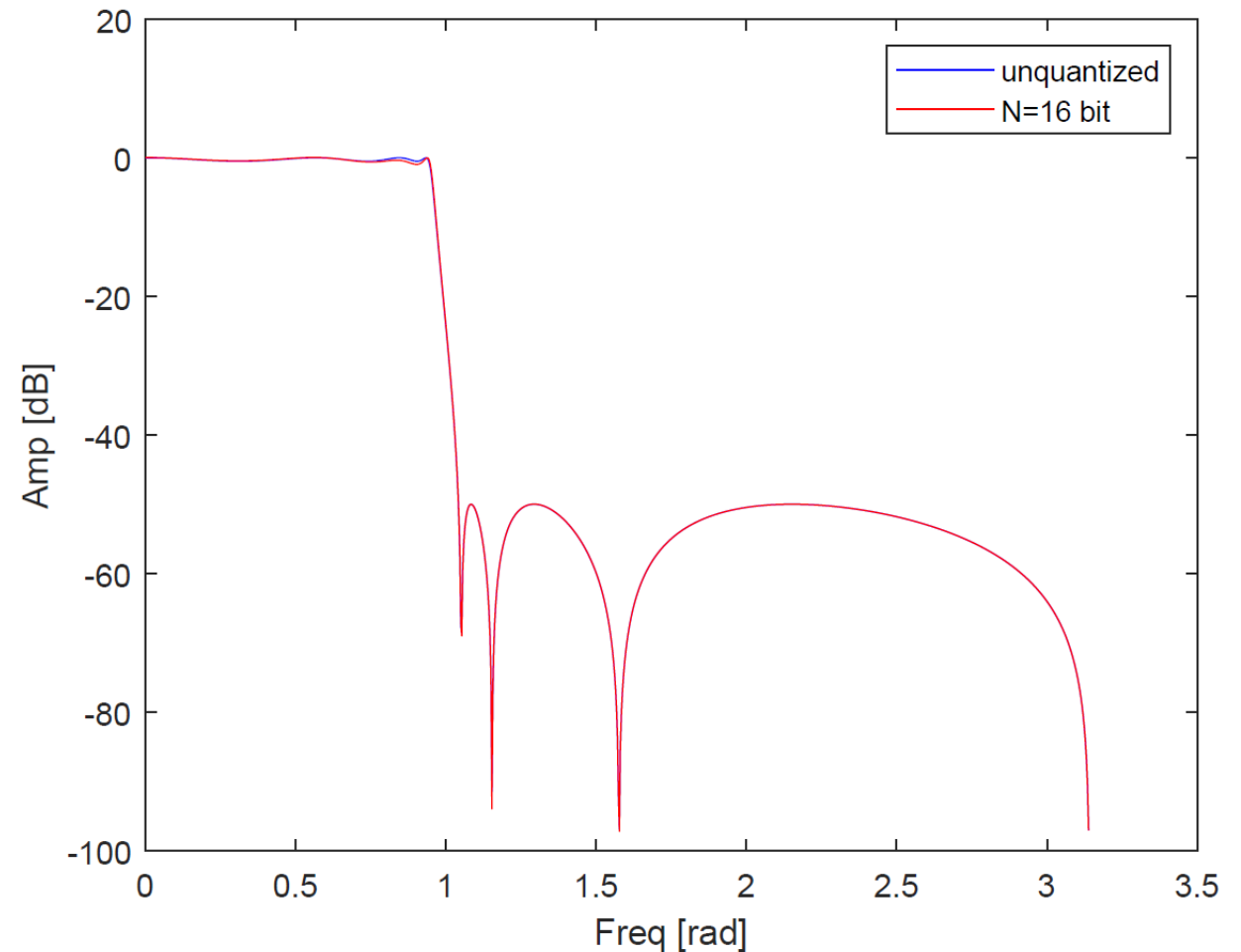
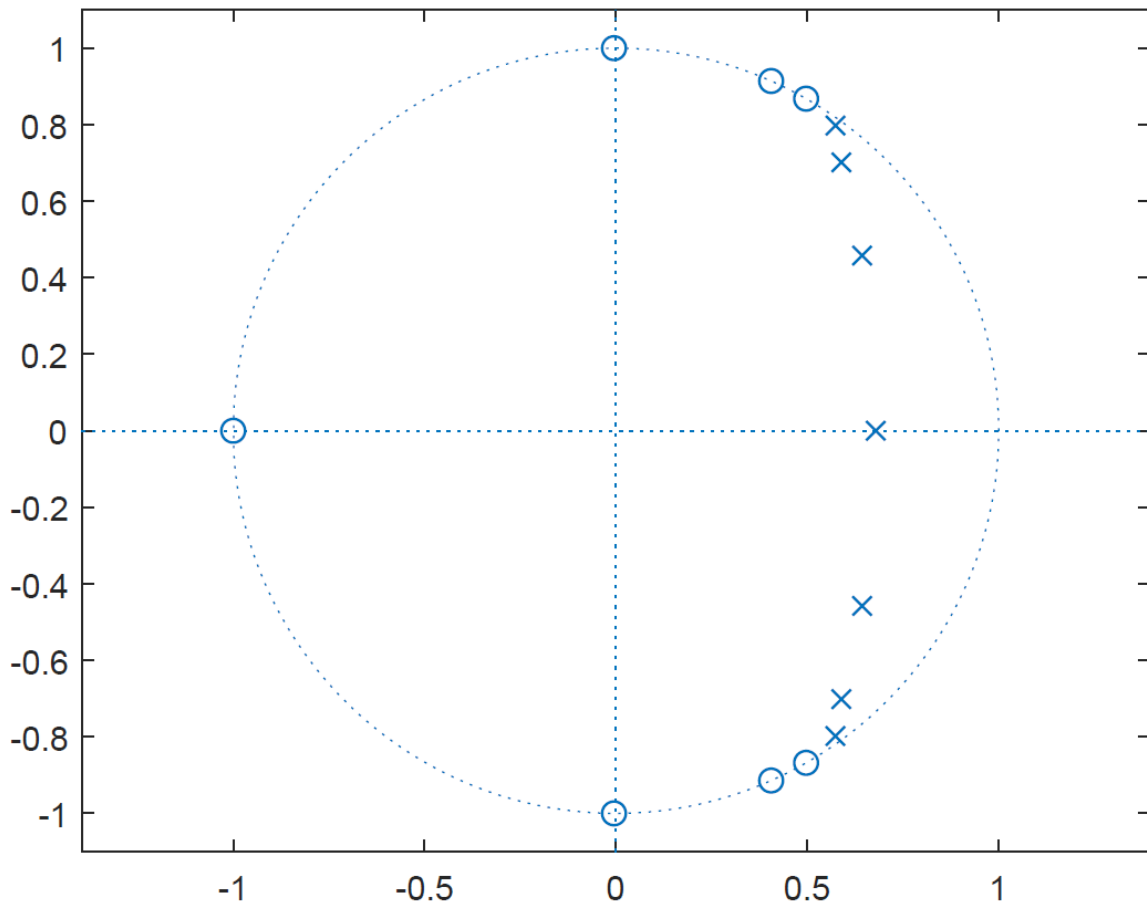
We therefore increase the word-length with 2 bit, from 10 to 12, maintaining 4 bit for the integer part, i.e.,  $Q5.7$



# How to solve the pole problem..?

At a 16 bit  $Q5.11$  representation, the filter is stable.

In conclusion, there are many possibilities for representing/adjusting the coefficients – with pros and cons...





# High-order filters

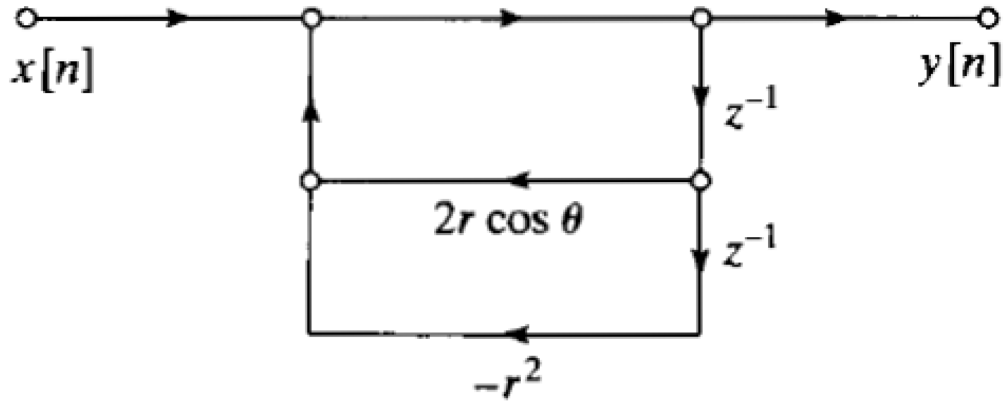
- Always consider a **factorization of  $H(z)$**  in order to derive a cascaded version consisting of **2nd order section**, i.e.,  $H(z) = \prod_{k=1}^{N/2} H_k(z)$ , where  $H_k(z) = \frac{B_k(z)}{A_k(z)}$  is a rational 2nd order function.
- Alternatively, using **partial fraction expansion**,  $H(z)$  can be sub-divided into a parallel version also consisting of **2nd order section**, i.e.,  $H(z) = \sum_{k=1}^{N/2} \frac{C_k}{A_k(z)}$  where  $A_k(z)$  is a 2nd order polynomial and  $C_k$  is a constant.
- Also however, be **careful with 2nd order sections...**!

Given  $H(z) = \frac{Y(z)}{X(z)} = \frac{1}{A(z)} = \frac{1}{1+a_1z^{-1}+a_2z^{-2}} = \frac{1}{1-2r \cdot \cos(\theta)z^{-1}+r^2z^{-2}}$  where  $r$  and  $\theta$  represent the complex conjugated roots of  $A(z)$ ,  $z = r \cdot e^{\pm j\theta}$ .

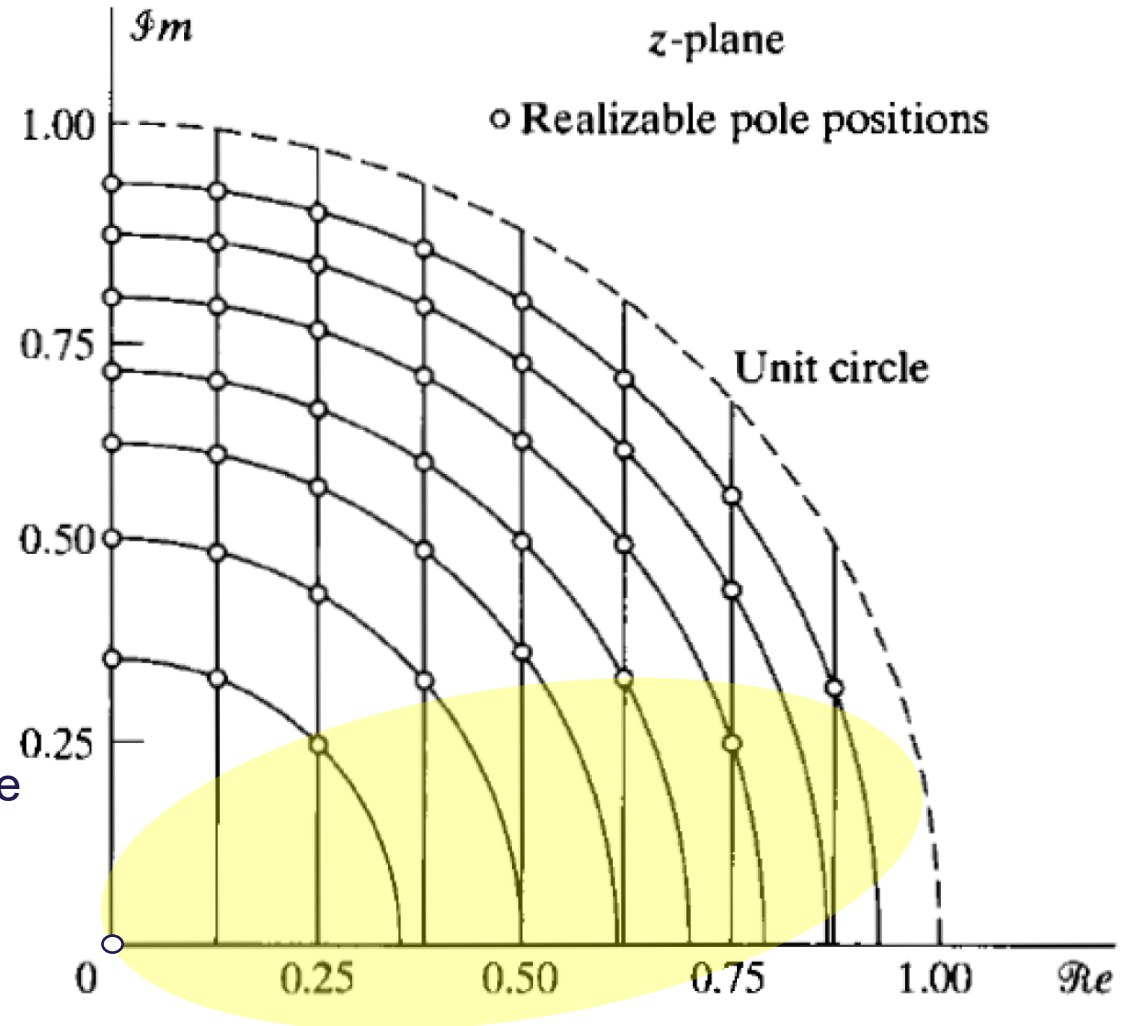
Quantizing the coefficient  $r \cdot \cos(\theta)$ , which is the real part of the poles, leads to linear spacing. However, quantizing linearly the coefficient  $r^2$  leads to a square root spaced distances between possible radii – so **the larger the  $r$ -value, the more closely spaced are the possible pole locations**. The combined pattern is shown on the next page...

# Possible pole locations in 2<sup>nd</sup> order sections

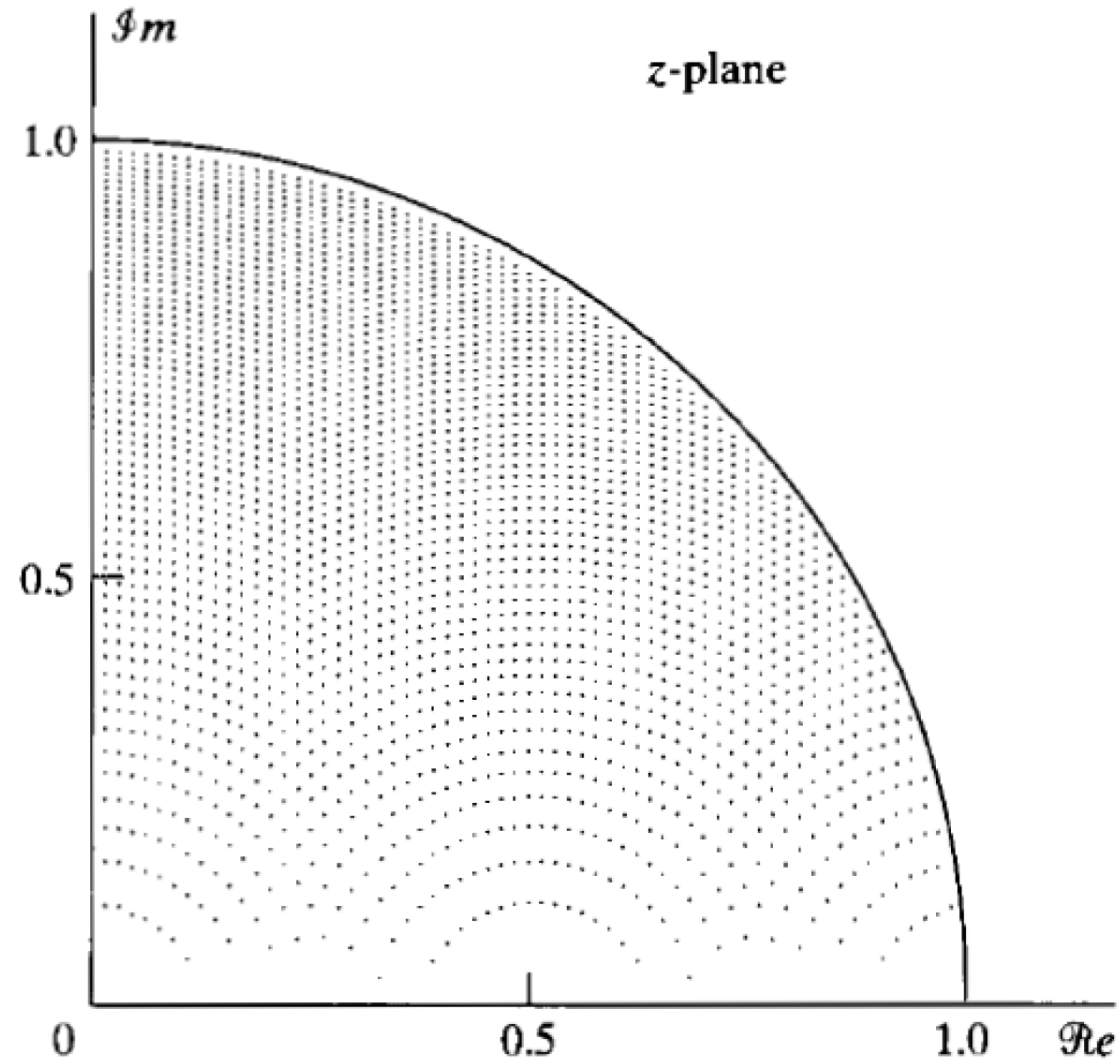
Here shown for  $B = 4$ .



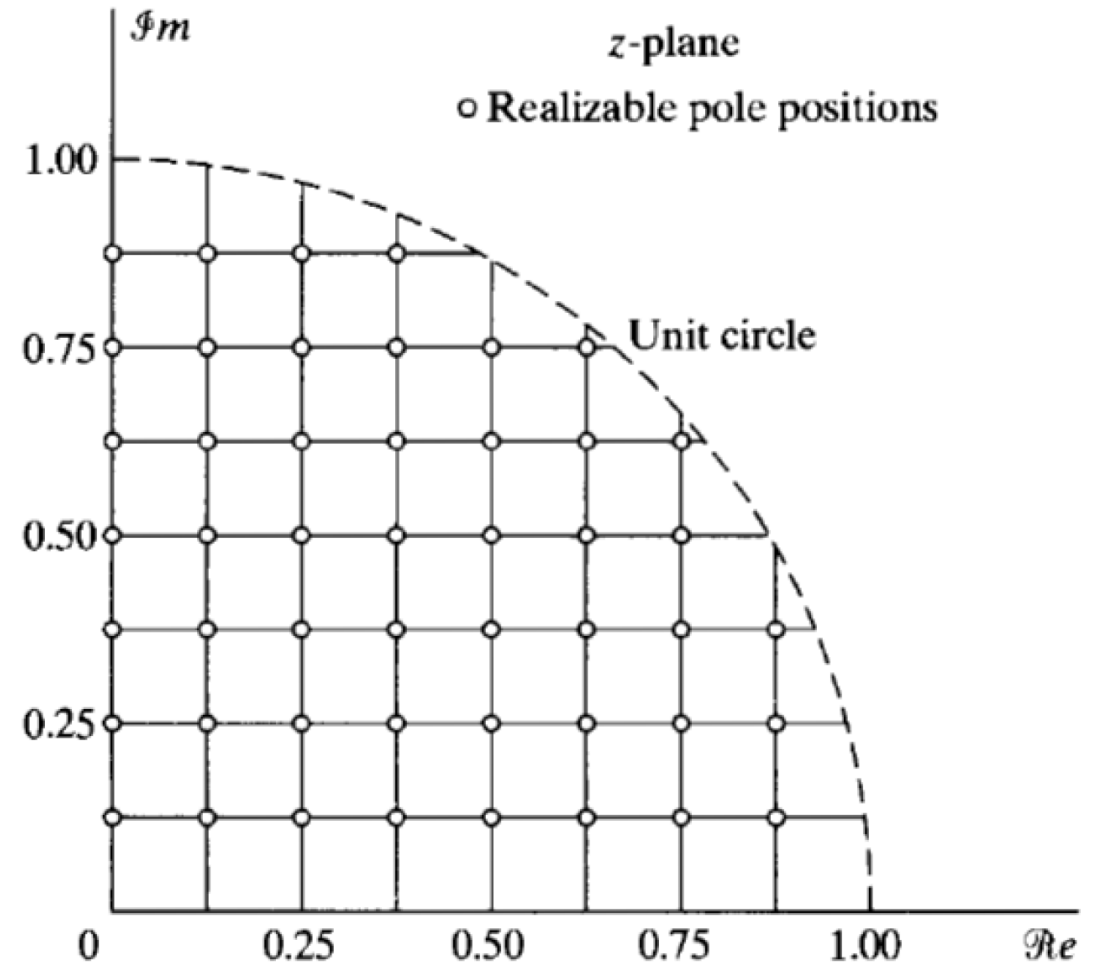
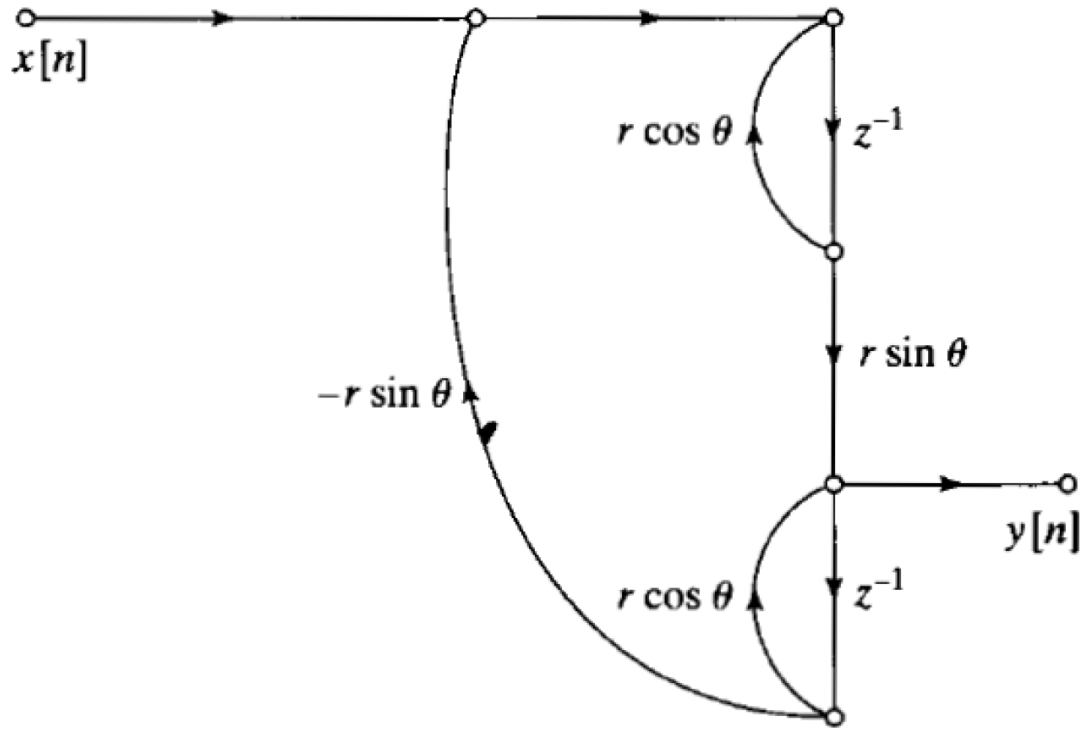
Poles located in this region of the z-plane are prone to major dis-alignment in a fixed-point implementation scenario.



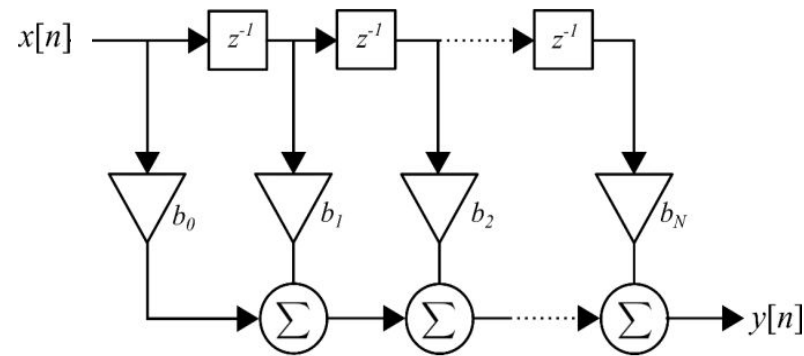
...and here shown for  $B = 7$



# One possible solution is the Coupled Form



# A few words on Coefficient quantization in FIR filters



The general FIR filter is characterized by the transfer function  $H(z) = \frac{B(z)}{1} = \sum_{k=0}^M b_k z^{-k}$ .

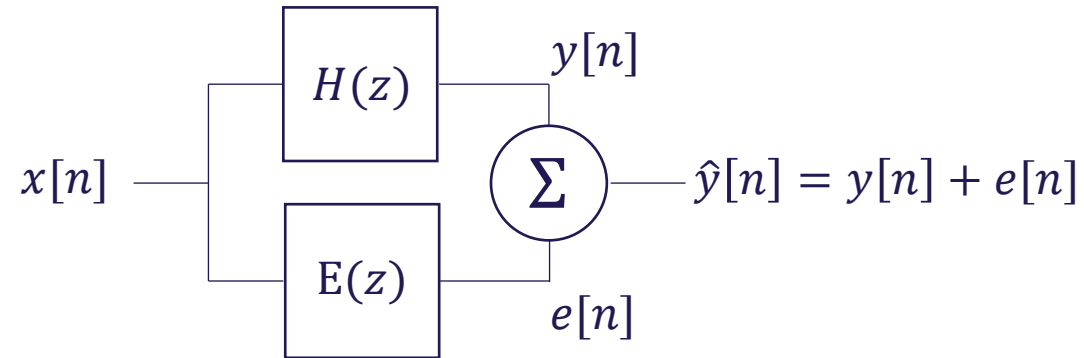
Quantizing the coefficients, we obtain

$$\hat{H}(z) = \sum_{k=0}^M \hat{b}_k z^{-k} = \sum_{k=0}^M (b_k + \varepsilon_k) z^{-k} = \sum_{k=0}^M b_k z^{-k} + \sum_{k=0}^M \varepsilon_k z^{-k} = H(z) + E(z)$$

# Coefficient quantization in FIR filters

$$\hat{H}(z) = H(z) + E(z)$$

So, the quantized FIR filter can be characterized as **two causal FIR filters coupled in parallel**.



Let's denote  $E(z)$  the "error filter". The **frequency response** of this filter is;

$$E(e^{j\omega}) = \sum_{k=0}^M \varepsilon_k \cdot e^{-j\omega k}$$

# Coefficient quantization in FIR filters

$$E(e^{j\omega}) = \sum_{k=0}^M \varepsilon_k \cdot e^{-j\omega k}$$

Assume now that the **coefficient quantization is conducted by rounding**, then  $-\Delta/2 \leq \varepsilon_k \leq \Delta/2$ , and thus;

$$\begin{aligned} |E(e^{j\omega})| &= \left| \sum_{k=0}^M \varepsilon_k \cdot e^{-j\omega k} \right| \\ &\leq \sum_{k=0}^M |\varepsilon_k \cdot e^{-j\omega k}| \leq \sum_{k=0}^M |\varepsilon_{max}| = \sum_{k=0}^M \Delta/2 = (M+1) \cdot \Delta/2 \end{aligned}$$

This value represents an **upper limit** on the amplitude response of the error filter, and thus we conclude;

$$|H(e^{j\omega})| - \frac{(M+1) \cdot \Delta}{2} \leq |\hat{H}(e^{j\omega})| \leq |H(e^{j\omega})| + \frac{(M+1) \cdot \Delta}{2}$$

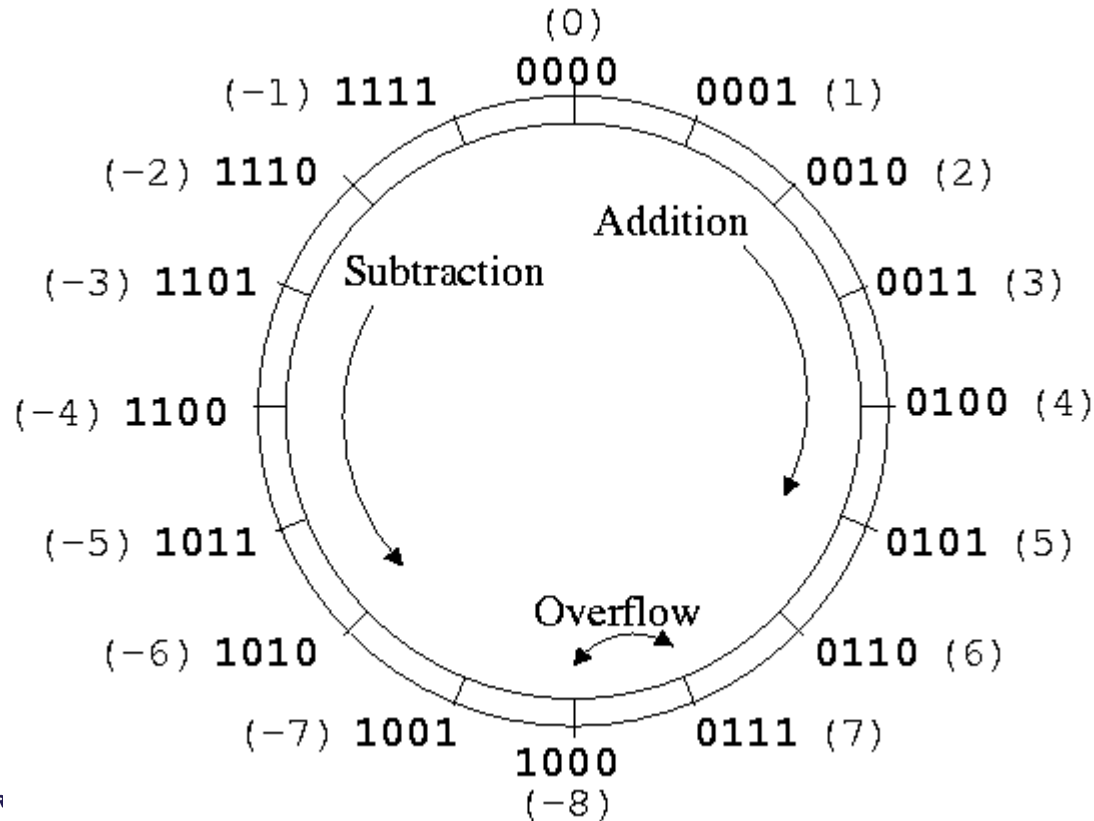
It can **be shown** that a more general expression for  $|E(e^{j\omega})|$  (for linear phase filters with odd order  $M$ ) is;

$$|E(e^{j\omega})| \leq \frac{\Delta}{2} \left\{ 1 + 2 \sum_{k=1}^{(M-1)/2} |\cos(k\omega)| \right\}$$

# Scaling in recursive filters

As we discussed earlier, we are very much interested in finding a **compromise** between best possible utilization of the **dynamic range**, and the **probability of overflow**.

Before we search for this compromise, we need to realize a neat feature of 2's complement numbers.



Assume that we add three numbers;  $a + b + c$ .

If  $|a + b| > 1$ , then this partial sum overflows.

If  $|a + b + c| < 1$ , then it doesn't matter that the  $(a + b)$  partial sum overflows – the total sum is within the legal dynamic range.

This feature is utilized in fixed-point implementation of digital filters...



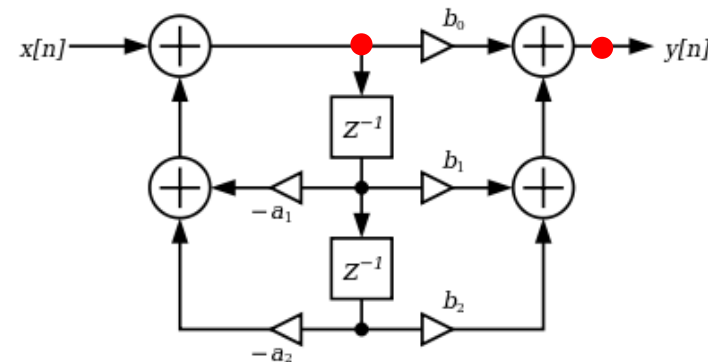
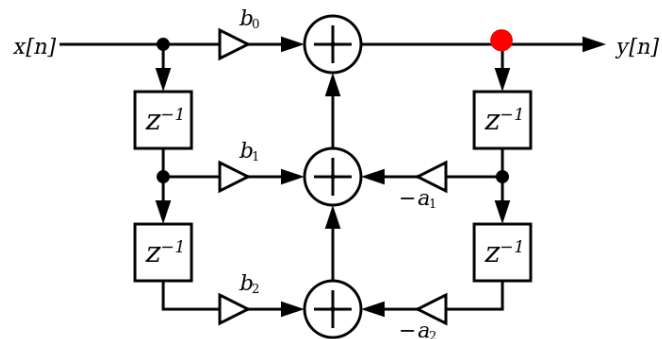
# Scaling in recursive filters

Implementing a digital filter means that we have to conduct a sum-of-product. If **the resulting sum** is within the **legal dynamic range**, i.e., without overflow, then we **allow the partial sums to overflow**;

$$|y[n]| = \left| \sum_{i=1}^N a_i \cdot y[n-i] + \sum_{j=0}^M b_j \cdot x[n-j] \right| < 1$$

Overflow is allowed in these partial sums

Note that the actual implementation structure (Direct Form I, Direct Form II, etc.) dictates which internal variables which may overflow, and **which may not...!!!**

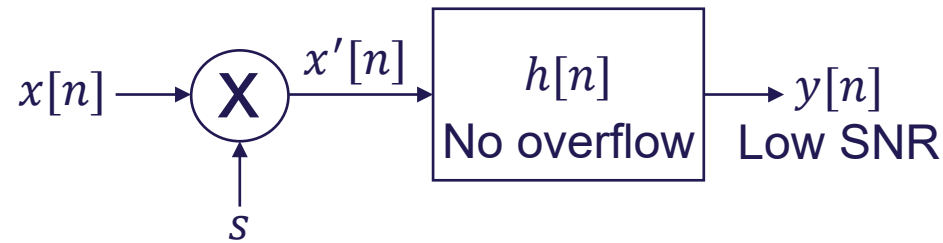


# Scaling in recursive filters

Now, we can search for a **compromise** between best possible utilization of the dynamic range, and the minimal probability of overflow – **in those variable where overflow is not allowed**.

The idea is to **determine a scaling constant  $s$**  which is multiplied onto the input variable;  $x'[n] = s \cdot x[n]$

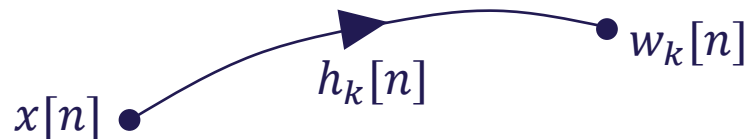
There exists **several methodologies** for finding  $s$  – the most immediate one is denoted "Abs-value scaling" or  **$L_1$  scaling**. The idea being that  $s$  is calculated in such a way that **overflow cannot occur** in those variable where **overflow is not allowed**. At a first glance, a very attractive idea – but unfortunately also leading to a very conservative result in the sense that  **$s$  is often so small** that the SNR suffers significantly.



Better though is to allow overflow in a **tiny amount of the overall processing time**, thus accepting unwanted numerical behavior in the implementation, but at the same time improving significantly on the SNR – i.e., introducing a larger input signal, and thus obtain a **better utilization of the dynamic range**.

# Scaling in recursive filters – Variance / Energy / $L_2$ scaling

The idea here is to require, that **the energy in all those variables  $w_k$** , where overflow is not allowed, **is less than the energy in the input signal  $x$** .



$h_k[n]$  represents the impulse response from the input to the variable  $w_k$ .

$$\sigma_{w_k}^2 < \sigma_x^2 \Rightarrow \frac{\sigma_{w_k}^2}{\sigma_x^2} < 1$$

Using **Parseval's identity**, we now introduce an expression for the energy in the internal variable  $w_k$ .

$$\sigma_{w_k}^2 = \sum_{n=-\infty}^{\infty} |w_k[n]|^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} |W_k(e^{j\omega})|^2 d\omega$$

Since  $w_k[n] = h_k[n] * x[n]$ , the Fourier transform is  $W_k(e^{j\omega}) = H_k(e^{j\omega}) \cdot X(e^{j\omega})$

# Scaling in recursive filters

$$\sigma_{w_k}^2 = \sum_{n=-\infty}^{\infty} |w_k[n]|^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} |W_k(e^{j\omega})|^2 d\omega \quad \text{and} \quad W_k(e^{j\omega}) = H_k(e^{j\omega}) \cdot X(e^{j\omega})$$

$$\sigma_{w_k}^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} |H_k(e^{j\omega}) \cdot X(e^{j\omega})|^2 d\omega$$

Now, using Cauchy–Schwarz inequality and Parseval (in reverse) we rewrite the expression to;

$$\sigma_{w_k}^2 \leq \sum_{n=-\infty}^{\infty} |x[n]|^2 \cdot \frac{1}{2\pi} \int_{-\pi}^{\pi} |H_k(e^{j\omega})|^2 d\omega$$

We can **always fulfill** this inequality by multiplying  $x[n]$  by a proper constant  $s$ .

# Scaling in recursive filters

$$\sigma_{w_k}^2 \leq s^2 \sum_{n=-\infty}^{\infty} |x[n]|^2 \cdot \frac{1}{2\pi} \int_{-\pi}^{\pi} |H_k(e^{j\omega})|^2 d\omega = s^2 \cdot \sigma_x^2 \cdot \frac{1}{2\pi} \int_{-\pi}^{\pi} |H_k(e^{j\omega})|^2 d\omega$$

$$\frac{\sigma_{w_k}^2}{\sigma_x^2} \leq s^2 \cdot \frac{1}{2\pi} \int_{-\pi}^{\pi} |H_k(e^{j\omega})|^2 d\omega < 1$$

$$s^2 < \frac{1}{\frac{1}{2\pi} \int_{-\pi}^{\pi} |H_k(e^{j\omega})|^2 d\omega} = \frac{1}{\sum_{n=-\infty}^{\infty} |h_k[n]|^2}$$

$$s < \frac{1}{\sqrt{\sum_{n=-\infty}^{\infty} |h_k[n]|^2}}$$

$$s = \frac{1}{\max_k \{\sqrt{\sum_{n=-\infty}^{\infty} |h_k[n]|^2}\}} \quad k \text{ represents variables in the filter structure where overflow is not allowed.}$$

# Probability for overflow

Assume that the signal  $x[n]$  is a **WSS stochastic process**.

The **Power Density Spectrum** for  $x[n]$  is defined as;

$$S_{xx}(e^{j\omega}) = \sum_{m=-\infty}^{\infty} r_{xx}(m) \cdot e^{-j\omega m} \quad \text{where} \quad r_{xx}(m) = E\{x[n] \cdot x[n+m]\} \quad \text{is the Autocorrelation sequence for } x[n].$$

Given the impulse response  $h_k[n]$  between the input and a variable  $w_k$ : 

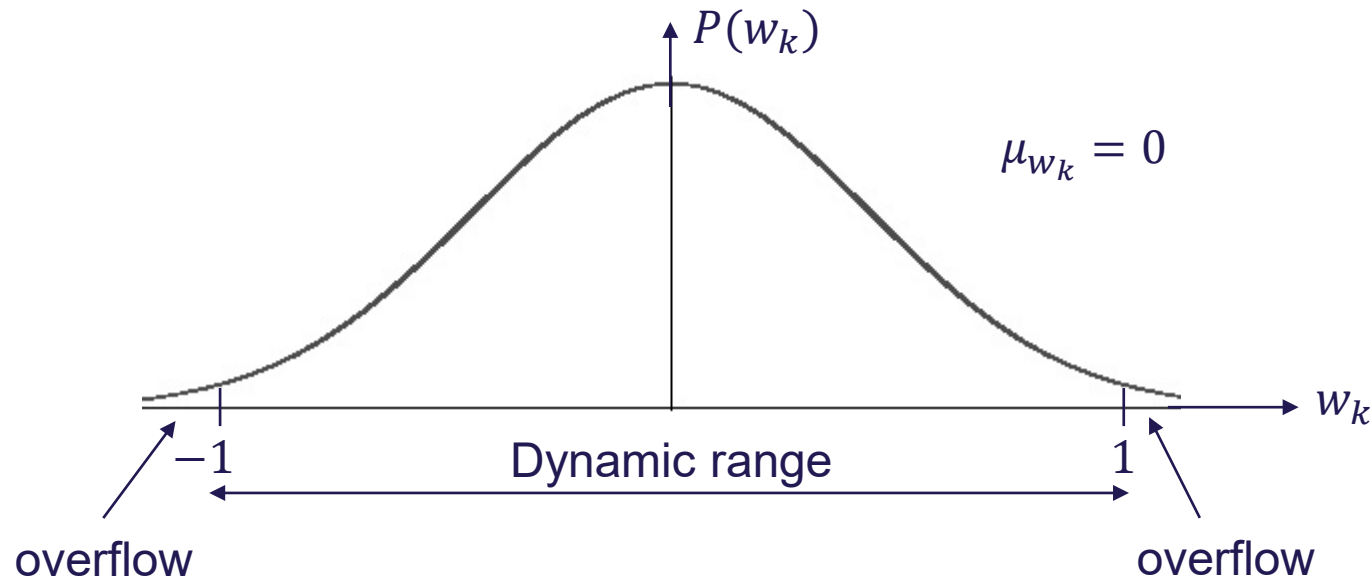
The **energy (variance) in variable  $w_k$**  can be determined from the relation;

$$\sigma_{w_k}^2 = \frac{1}{2\pi} \int_0^{2\pi} |H_k(e^{j\omega})|^2 \cdot S_{xx}(e^{j\omega}) d\omega \leq \|H_k^2\|_2 \cdot \|S_{xx}\|_2 \quad \text{where } \|\cdot\|_2 \text{ is the } L_2 \text{ norm.}$$

Due to the  **$\leq$  relation**, this equation represents an **upper limit on the energy** of the signal  $w_k[n]$ .

# Probability for overflow

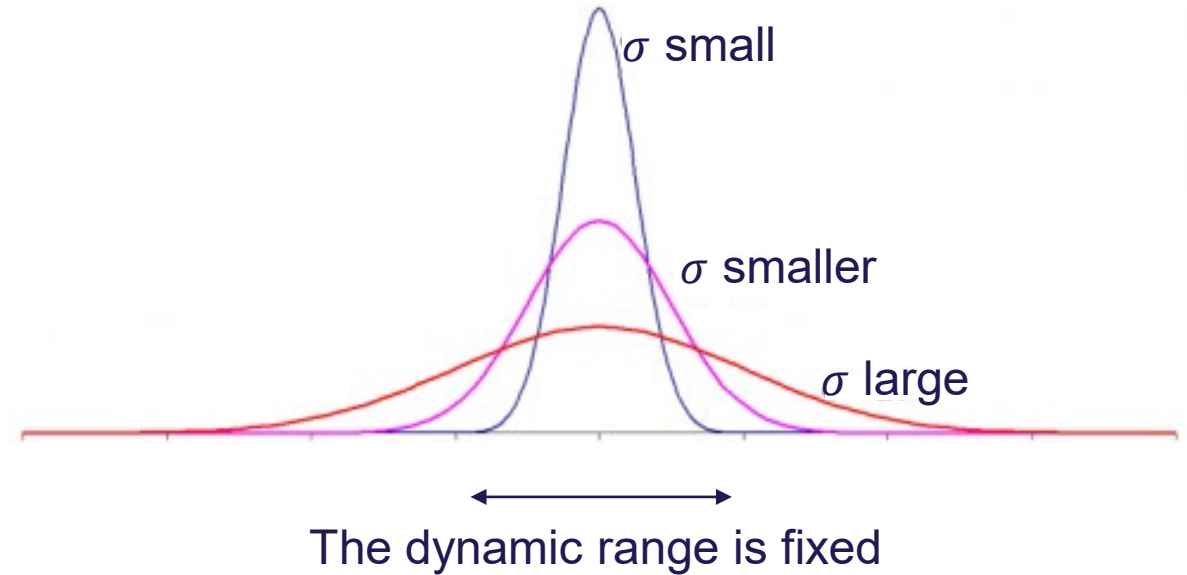
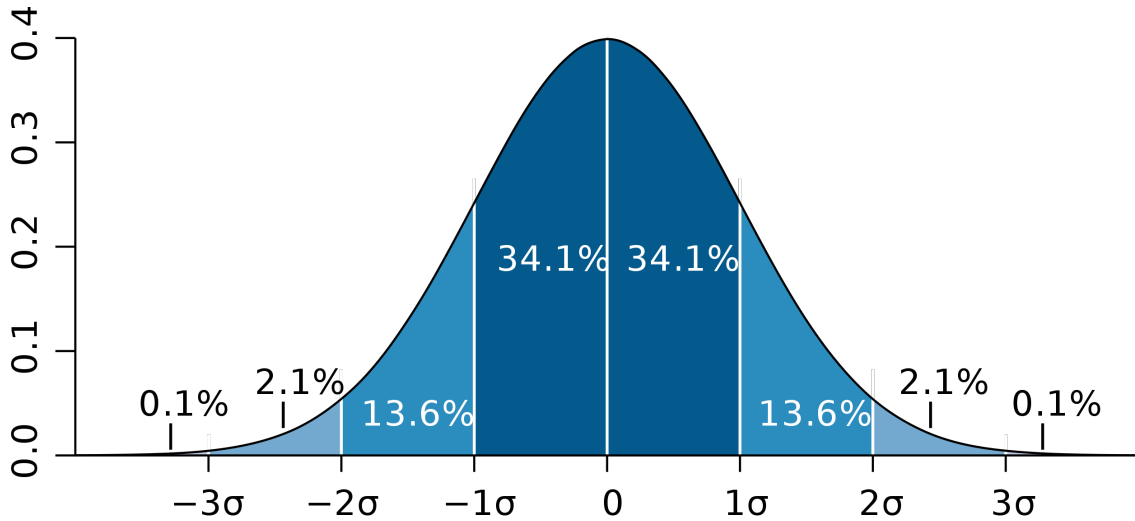
A realistic assumption is that the signal  $w_k[n]$  represents a sequence which (in amplitude) is a normal distribution.



Given the upper limit on  $\sigma_{w_k}^2$ , we similarly have an **upper limit on the standard deviation  $\sigma_{w_k}$** .

Knowing the upper limit on  $\sigma_{w_k}$ , we can also tell accurately the **upper limit on the probability for overflow**.

# Probability for overflow



Therefore, when  $\sigma$  is continuously reduced, the distribution becomes more narrow, and thus a continuously larger part of the distribution becomes centered around the mean, leading to a continuously smaller area outside the dynamic range, i.e., continuously reduced probability for overflow...



# Probability for overflow

Upper limit on $\sigma_{w_k}$	Upper limit on the overflow probability
1.0	0.318
0.8	0.212
0.6	0.096
0.4	0.012
0.333	0.0028
0.3	0.00092
0.25	6.4e-5
0.20	6.0e-7

If we choose  $\sigma_{w_k} < 1/3$  then the probability for overflow is less than 0.003, i.e., in such a case, **statistically there is overflow in 3 0/00 of the overall execution time** – this is normally seen as an acceptable rate.

One way to accomplish this is to introduce a **safety factor  $\delta = 3$**  on the input after  $L_2$  scaling, i.e., the input signal  $x[n]$  is multiplied by  $1/\delta$  (assuming that  $\sigma_x^2 = 1$ ).

# Noise-optimal filter structures

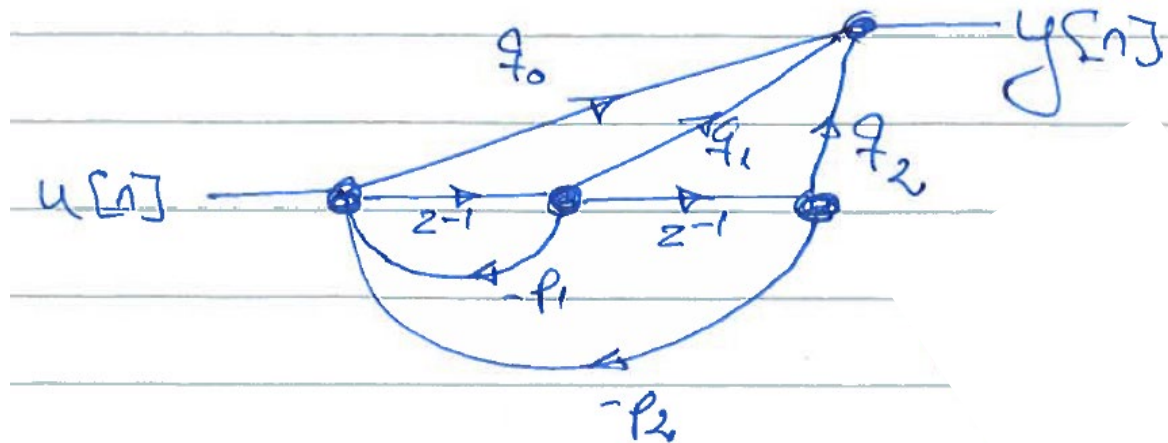
We now turn into a brief discussion on **realization structures for recursive filters** which can be transformed into **noise-optimale topologies**.

Given the transfer function  $H(z)$ ;

$$H(z) = \frac{Y(z)}{U(z)} = \frac{Q(z)}{P(z)} = \frac{\sum_{k=0}^M q_k z^{-k}}{1 + \sum_{k=1}^N p_k z^{-k}}$$

For  $N = M = 2$ , the inverse z-transform leads to the difference equation;

$y[n] = q_0 u[n] + q_1 u[n - 1] + q_2 u[n - 2] - p_1 y[n - 1] - p_2 y[n - 2]$ , here illustrated as a **Signal Flow Graph**



This is known as the **Canonic Structure** due to the minimal number of delays.  
(Direct Form II)

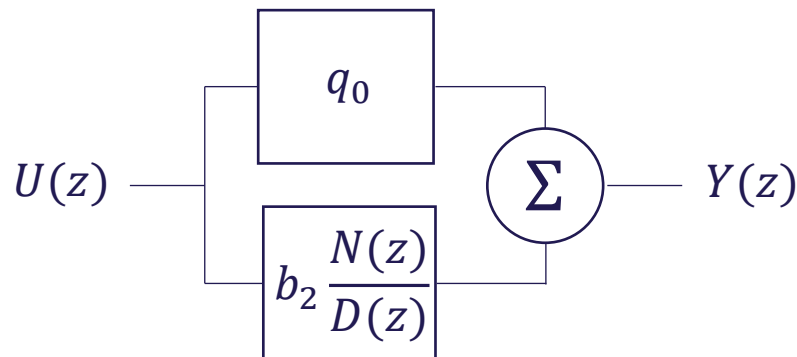
# Noise-optimal filter structures – the Direct Canonic Form

Through a series of manipulations, the general transfer function  $H(z)$  can be re-structured into;

$$H(z) = \frac{Y(z)}{U(z)} = q_0 + b_N \cdot \frac{\sum_{k=1}^N \left( \frac{q_k + q_0 \cdot p_k}{b_N} \right) z^{-k}}{1 + \sum_{k=1}^N p_k z^{-k}} \quad \text{which for } N = 2 \text{ reduces to}$$

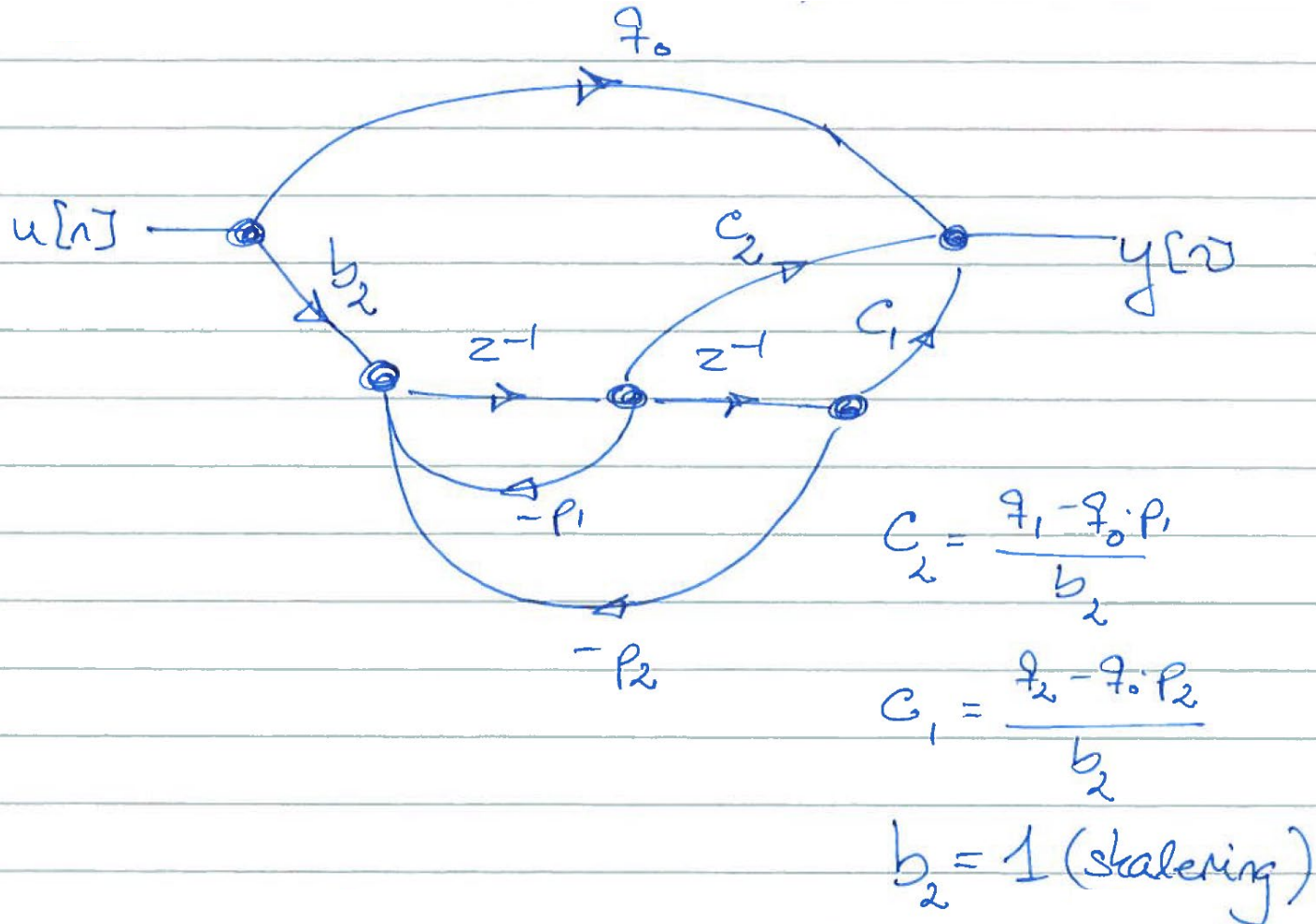
$\swarrow$   $c_{N+1-k}$

$$Y(z) = q_0 \cdot U(z) + b_2 \cdot \frac{c_2 z^{-1} + c_1 z^{-2}}{1 + p_1 z^{-1} + p_2 z^{-2}} \cdot U(z)$$



This is a very interesting version of the general 2<sup>nd</sup> order transfer function – because there is a **direct coupling from input to output**.

# Noise-optimal filter structures – the Direct Canonic Form



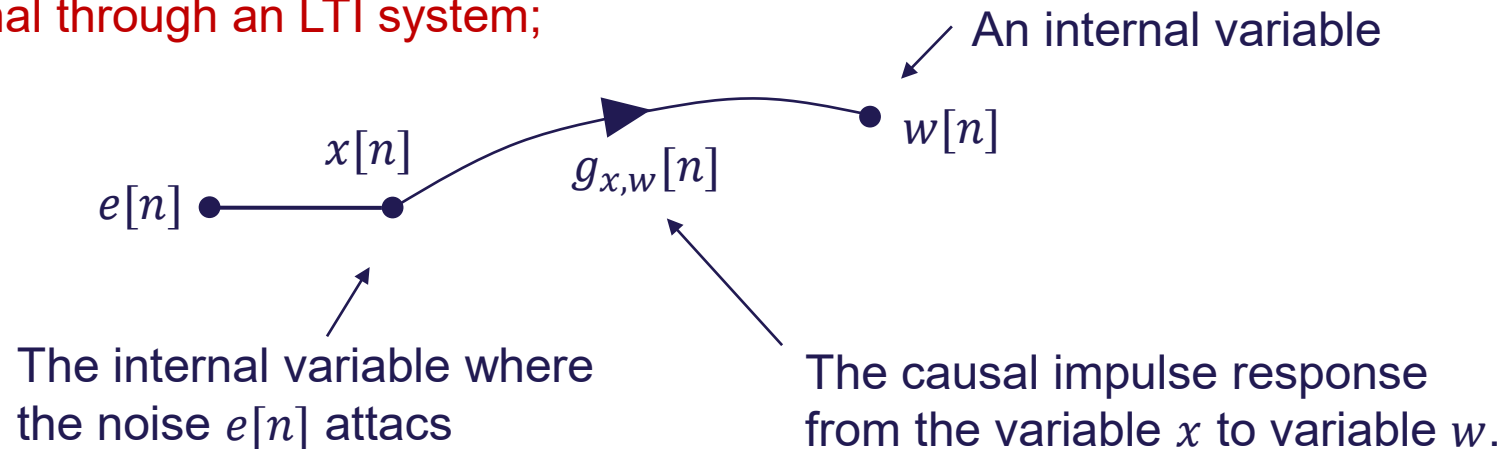
The **advantages of this structure** – here shown as a signal flow graph...

- The direct coupling from input to output.
- The Direct Canonic Form is a **special case** of the State Space structure.
- The **SS structure** has the interesting feature that it **can be transformed**, with respect to the internal variable, such that it becomes **optimal with regard to quantization noise**...!

# Scaling the Direct Canonic Form

Remember that we previously made the assumption that **quantization noise** (rounding from  $2B$  to  $B$  bit) is a "white noise" source  $e[n]$ .

Coupling of a white signal through an LTI system;



$$\sigma_w^2 = \sigma_e^2 \sum_{n=0}^{\infty} g_{x,w}^2[n]$$

This is the **noise variance** in the internal variable  $w$ .

# Scaling the Direct Canonic Form

Realistically assuming that the individual noise noises,  $e_i[n]$  are independent (un-correlated), then the total noise variance at the output of the filter can be calculated as a sum of the individual noise contributions (here single precision);

$$\sigma_{out,total}^2 = \sum_{i=1}^{\# \text{ mult}} \left( \frac{\Delta^2}{12} \right) \cdot \sum_{n=0}^{\infty} g_{i,out}^2 [n]$$

Here we assume identical word-length in all variables.

Similarly, if the calculation is conducted in double precision, the total noise variance at the output is;

$$\sigma_{out,total}^2 = \sum_{i=1}^{\# \text{ product sums}} \left( \frac{\Delta^2}{12} \right) \cdot \sum_{n=0}^{\infty} g_{i,out}^2 [n]$$

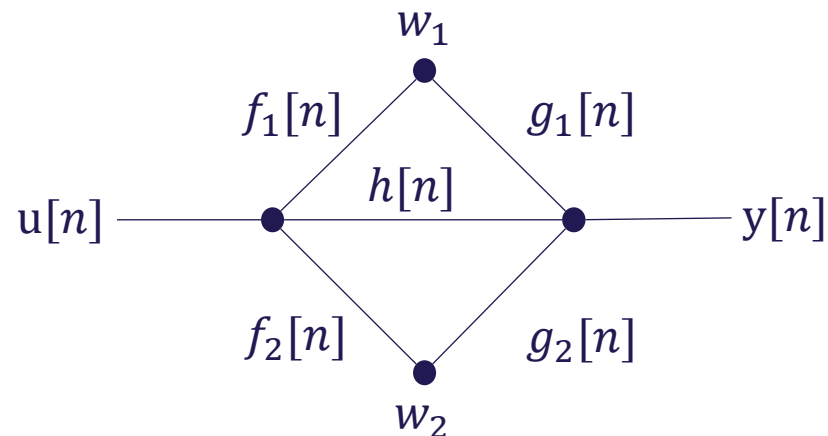
# Scaling the Direct Canonic Form

Again, we want to **avoid overflow**... Essentially, there are two different approaches; 1) apply **Saturation Arithmetic**, or 2) conduct **proper scaling** of the filter structure, which means;

1. enable **full utilization of the dynamic range** in all variables where overflow is not allowed, and
2. **adjust the input amplitude level**, i.e., tune the standard deviation on the input signal using a safety factor.

Ad 1) The  $L_2$  scaling paradigm:  $\sigma_{out}^2 = \sigma_{intern}^2 = \sigma_{input}^2$

We will now use the **symbolic SFG** to explain the procedure;



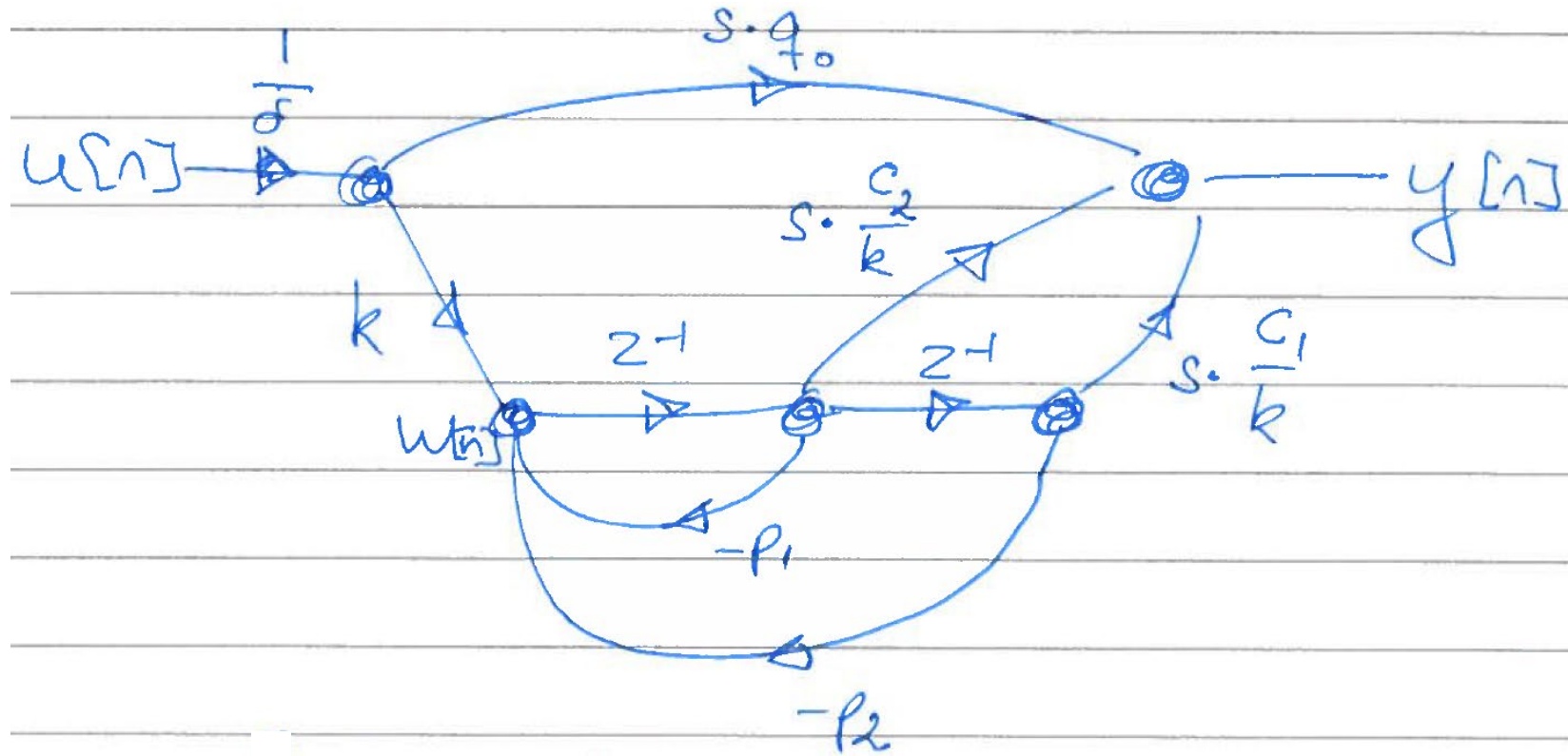
The impulse response from input to output is  $h[n]$ .

Internally in the structure, there are two variables where **overflow is not allowed**,  $w_1$  and  $w_2$ .





# Scaling the Direct Canonic Form – the totally scaled structure

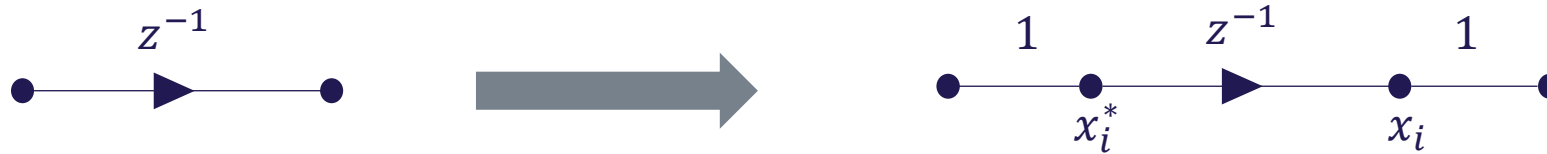


Note that there is only one internal variable  $w[n]$ .

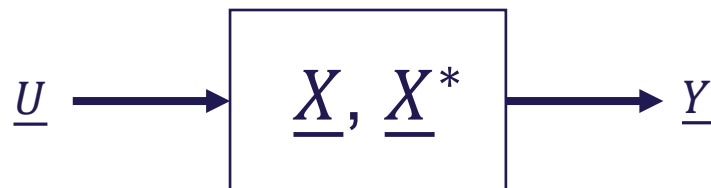
# The State Space notation

We now extend to the **general case**, where the **filter structure** is described using a SS notation. A State Space filter can **modified into a noise-optimal structure**.

The idea is to **transform the signal flow graph** by introducing "State Variables",  $x_i^*$  and  $x_i$ .



In any transformed SFG, there are "a number" of such state variable, and thus it makes sense to introduce a **vector notation** based on the **state vectors  $\underline{X}$  and  $\underline{X}^*$** .



$$\underline{X}^* = \underline{A}\underline{X} + \underline{B}\underline{U}$$

$$\underline{Y} = \underline{C}\underline{X} + \underline{D}\underline{U}$$

$$\underline{X}^* = z\underline{X}$$

# The State Space notation

$$\underline{X}^* = \underline{A}\underline{X} + \underline{B}\underline{U}$$

$$\underline{Y} = \underline{C}\underline{X} + \underline{D}\underline{U}$$

$$\underline{X}^* = z\underline{X}$$

For an  $N'$ th order filter we have 1 input  $u[n]$ , 1 output  $y[n]$ , and  $N$  state variables  $\underline{X}$ , and in the time domain the state description is;

Computation of next state

$$\underline{X}^*[n] = \underline{X}[n + 1] = \underline{A}\underline{X}[n] + \underline{b}u[n]$$

$$y[n] = \underline{c}\underline{X}[n] + du[n]$$

Computation of output

# The State Space notation for $N = 2$

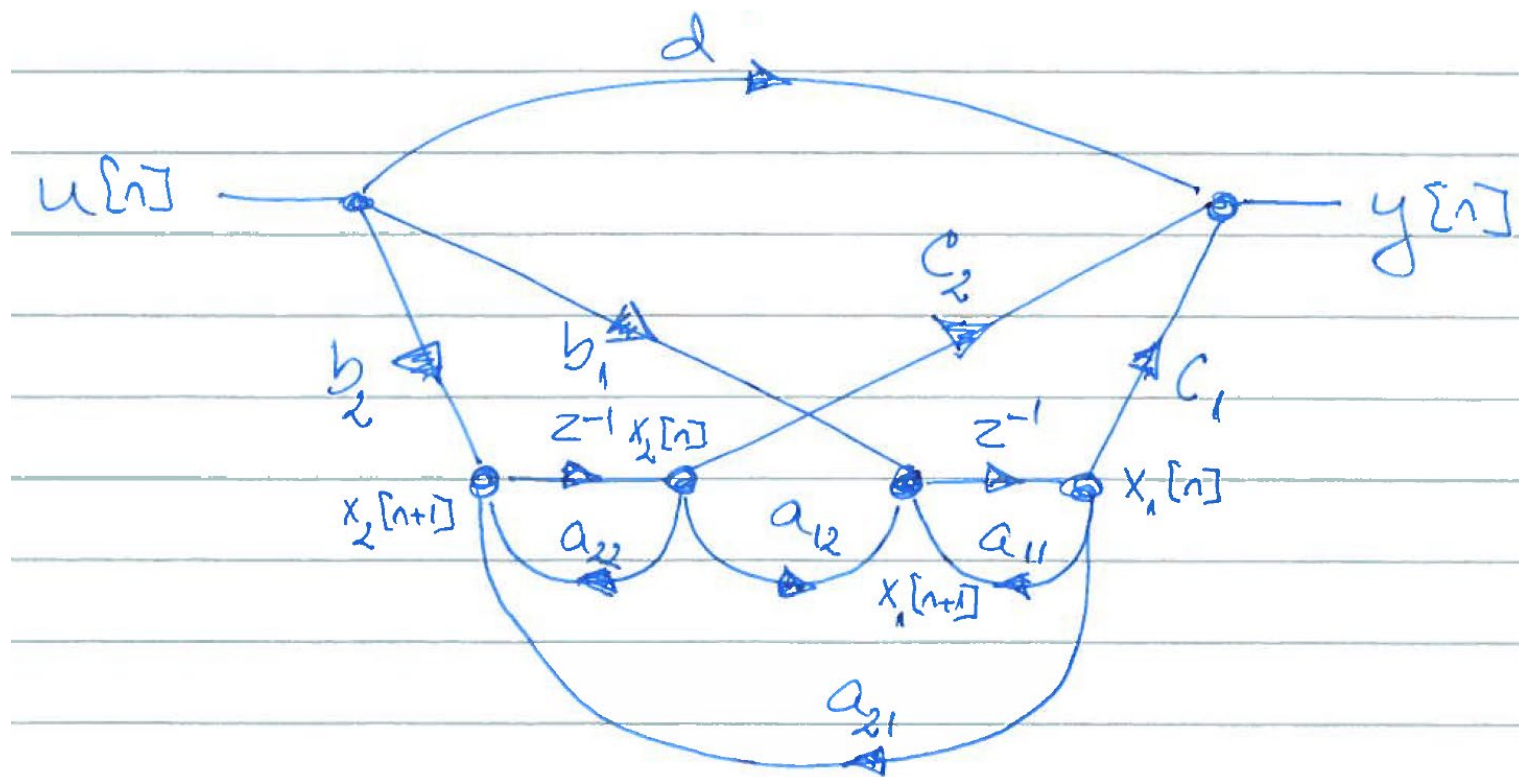
$$\underline{X}^*[n] = \underline{X}[n + 1] = \underline{A}\underline{X}[n] + \underline{b}u[n]$$

$$y[n] = \underline{c}\underline{X}[n] + du[n]$$

$$y[n] = c_1x_1[n] + c_2x_2[n] + du[n]$$

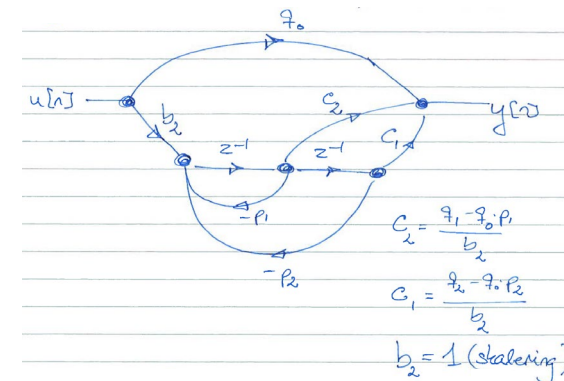
$$x_1[n + 1] = a_{11}x_1[n] + a_{12}x_2[n] + b_1u[n]$$

$$x_2[n + 1] = a_{21}x_1[n] + a_{22}x_2[n] + b_2u[n]$$



For the **Direct Canonic Form**, we see that

$$\underline{A} = \begin{Bmatrix} 0 & 1 \\ -p_2 & -p_1 \end{Bmatrix} \quad \underline{b} = \begin{Bmatrix} 0 \\ b_2 \end{Bmatrix} \quad \underline{c} = \begin{Bmatrix} q_2 - q_1 p_2 \\ b_2 \\ q_1 - q_0 p_1 \\ b_2 \end{Bmatrix} \quad d = q_0$$



# State Space – Transfer function and Transformation

In order to establish a transfer function, we  **$z$  transform the SS equations**;  $\underline{X}[n + 1] = \underline{A}\underline{X}[n] + \underline{b}u[n]$   
 $y[n] = \underline{c}\underline{X}[n] + du[n]$  


$$\underline{zX}(z) = \underline{A}\underline{X}(z) + \underline{b}U(z)$$
$$Y(z) = \underline{c}\underline{X}(z) + dU(z)$$

which can be manipulated into;  $H(z) = \frac{Y(z)}{U(z)} = d + \underline{c}(z\mathbf{I} - \underline{A})^{-1}\underline{b}$

**Without changing the transfer function  $H(z)$** , we now search for a **transformation** which can modify the **internal variable** in the filter.

The idea is to derive **new internal variables**, using a **Transform Matrix  $T$** ;

$$\underline{X}_{new}[n] = \underline{X}'[n] = \underline{T}^{-1}\underline{X}[n] \text{ where } \underline{T} \text{ is a non-singular matrix.}$$

We now apply  $\underline{X}[n] = \underline{T}\underline{X}'[n]$  into  which leads to;

# State Space – Variable Transformation

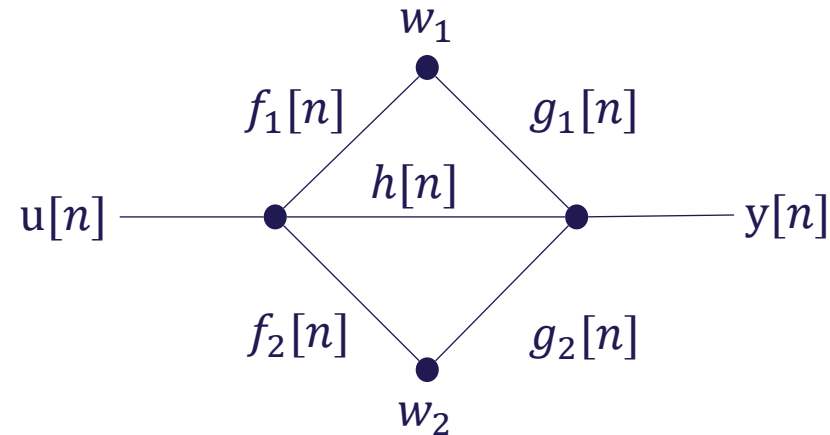
$$\begin{aligned} \underline{X}'[n+1] &= \underline{A}'\underline{X}'[n] + \underline{b}'u[n] \\ y'[n] &= \underline{c}'\underline{X}'[n] + d'u[n] \end{aligned} \quad \text{where} \quad \begin{aligned} \underline{A}' &= \underline{T}^{-1}\underline{A}\underline{T} \\ \underline{b}' &= \underline{T}^{-1}\underline{b} \\ \underline{c}' &= \underline{c}\underline{T} \\ d' &= d \end{aligned}$$

Inserting this expression for **the transformed SS equations** into the expression for the transfer function  $H(z)$ , p.65, it can be shown that  $T$  is **neutral to  $H(z)$** .

Question is; **Can we find any helpful  $T$  which can now be used to modify the State Space structure..??**

# State Space – Systems Matrices $K$ and $W$

Definition of two new matrices,  $K$  and  $W$ .



$$K_{ij} = \sum_{n=0}^{\infty} f_i[n] \cdot f_j[n]$$

$$W_{ij} = \sum_{n=0}^{\infty} g_i[n] \cdot g_j[n]$$

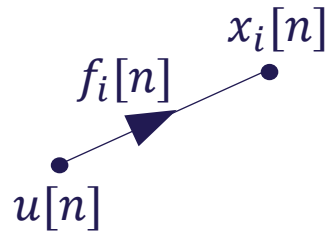
We normally can find the impulse responses  $f$  and  $g$ , and since we look only at stable filters then (from a numerical point of view), **both  $f$  and  $g$  are finite length, and thus we can determine  $K_{ij}$  and  $W_{ij}$ .**

**The two matrices  $K$  and  $W$  can now be used for  $L_2$  scaling.**

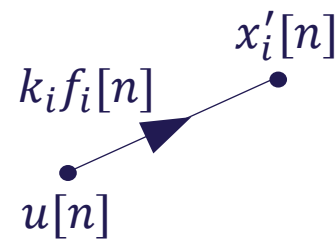
# State Space – internal variable scaling, X

$$\sigma_{x_i}^2 = \sigma_u^2 \cdot \sum_{n=0}^{\infty} (k_i \cdot f_i[n])^2 \Rightarrow k_i = \frac{1}{\sqrt{\sum_{n=0}^{\infty} f_i^2[n]}} = \frac{1}{\sqrt{K_{ii}}}$$

Un-scaled filter



Scaled filter



Defining the **scaling Matrix**;

$$\mathbf{T}_s = \begin{bmatrix} \sqrt{K_{11}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sqrt{K_{NN}} \end{bmatrix} = \begin{bmatrix} 1/k_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1/k_N \end{bmatrix}$$

Since  $\mathbf{T}_s$  is a diagonal matrix, **the inverse matrix**,  $\mathbf{T}_s^{-1}$ , is easily found;

$$\mathbf{T}_s^{-1} = \begin{bmatrix} 1/\sqrt{K_{11}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1/\sqrt{K_{NN}} \end{bmatrix} = \begin{bmatrix} k_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & k_N \end{bmatrix}$$



# State Space – internal variable scaling, $\underline{X}$

The **transformation of the internal variables** is conducted as;  $\underline{X}'[n] = \mathbf{T}_s^{-1}\underline{X}[n]$  c.f., p.65, and thus;

$$\begin{aligned} \mathbf{A}' &= \mathbf{T}_s^{-1}\mathbf{A}\mathbf{T}_s \\ \underline{b}' &= \mathbf{T}_s^{-1}\underline{b} \\ \underline{c}' &= \underline{c}\mathbf{T}_s \\ d' &= d \end{aligned}$$

which represents the **internally scaled** filter.

Next, we need to conduct **external scaling**, i.e., scaling from input  $u$  to output  $y$ .

External scaling **must not** interfere with the internal scaling just performed, and thus **it can impact only** **The vector  $c$  and the scalar  $d$ .**

$y''[n] = s \cdot \underline{c}'\underline{x}'[n] + s \cdot d'u[n]$  where the scaling factor  $s$  is found via the impulse response from input to output, i.e.,  $s = \frac{1}{\sqrt{\sum_{n=0}^{\infty} h^2[n]}}$  (see p. 49).

# The totally scaled State Space filter

$$\begin{aligned} \underline{A}'' &= \underline{T}_s^{-1} \underline{A} \underline{T}_s \\ \underline{b}'' &= \underline{T}_s^{-1} \underline{b} \\ \underline{c}'' &= s \cdot \underline{c} \underline{T}_s \\ \underline{d}'' &= s \cdot d \end{aligned}$$

Note that despite we use '' on all coefficients (on the left hand side), representing the totally scaled filter,  $\underline{A}''$  and  $\underline{b}''$  still denote the internally scaled filter.

Using the definitions of the system matrices  $\underline{K}$  and  $\underline{W}$ , we can find expressions for their scaled versions;

## Internal scaling

$$\underline{K}' = \underline{T}_s^{-1} \underline{K} (\underline{T}_s^T)^{-1}$$

$$\underline{W}' = \underline{T}_s^T \underline{W} \underline{T}_s$$

## External scaling

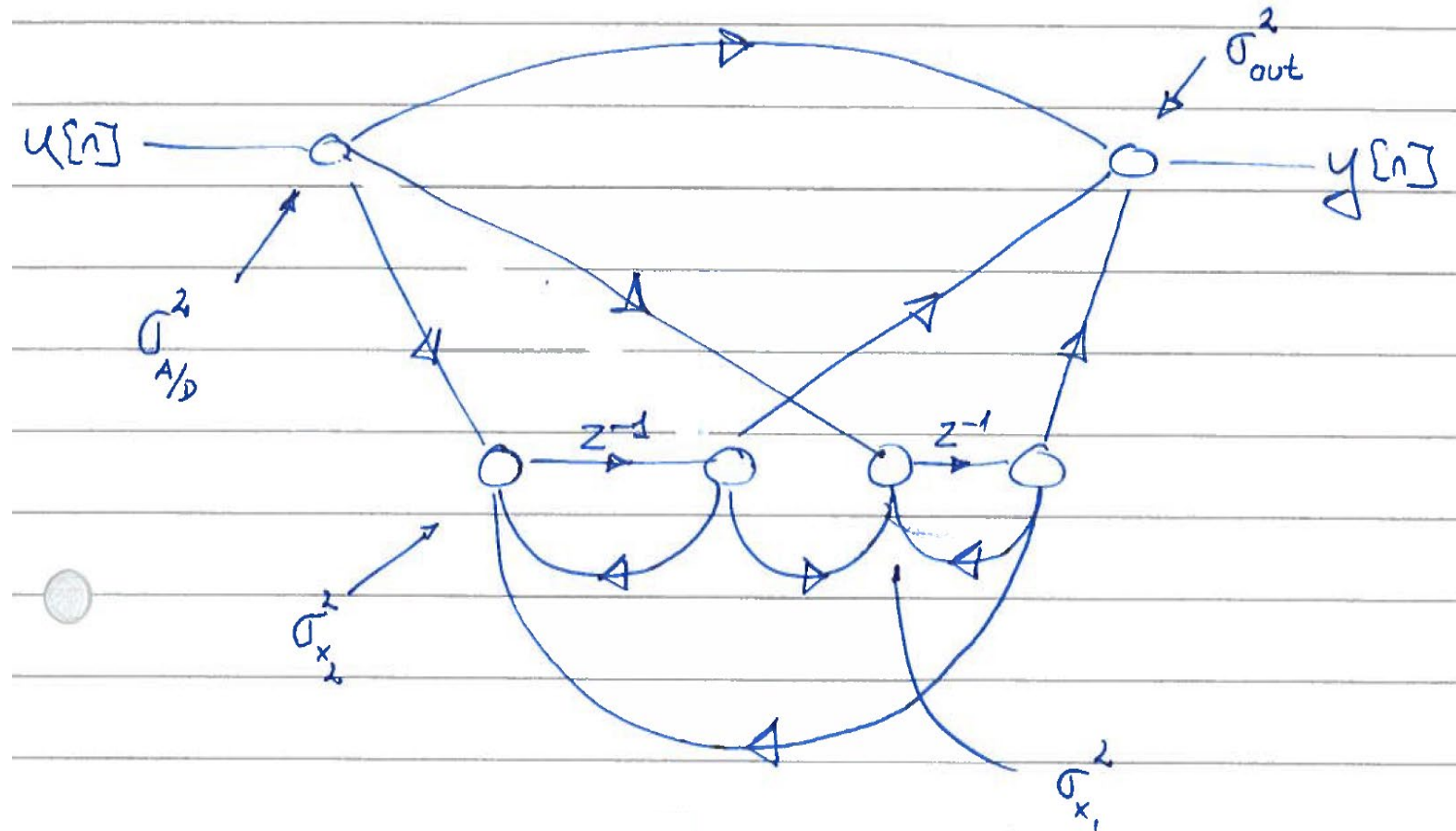
$$\underline{K}'' = \underline{K}'$$

$$\underline{W}'' = s^2 \underline{W}'$$

Again, note that external scaling does not impact matrix  $\underline{K}$ .

# Totally scaled State Space filter – noise variance

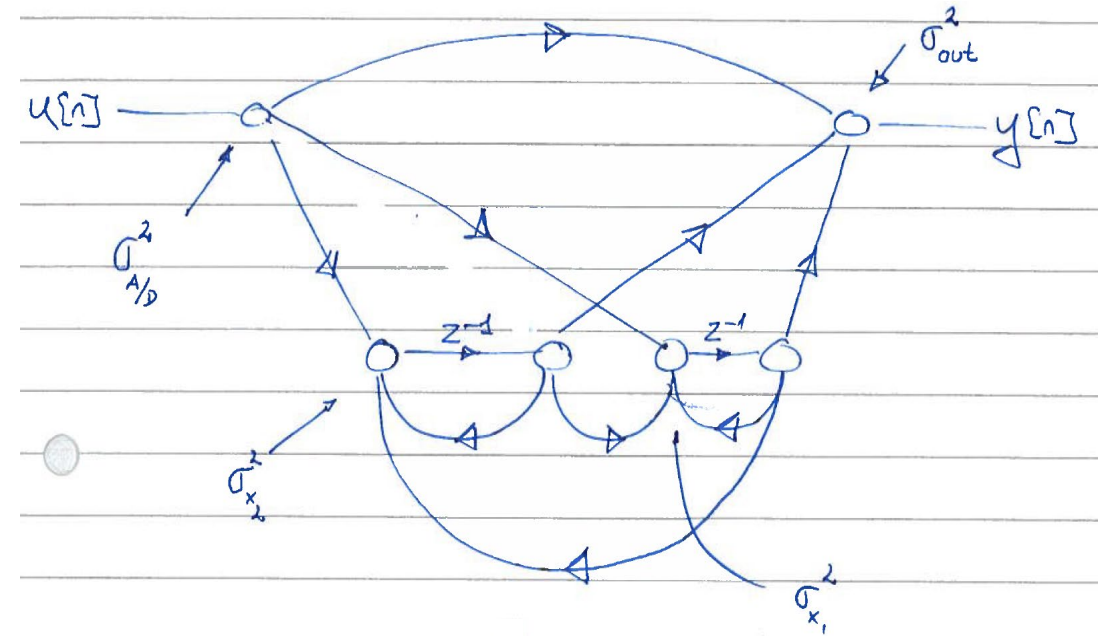
For a 2<sup>nd</sup> order State Space filter, the **quantization noise can attack in four variables**; 1) at the input due to quantization in the ADC, 2) in the two internal variables, and finally 3) at the output.



# Noise variance at State Space filter output – general $N'$ th order case

We assume **identical word-length** throughout all variable in the filter, and **product summation in double precision**.

Thus the **total noise variance** at the output of an  $N'$ th order filter can be expressed as;



$$\sigma_{out,total}^2 = \sigma_{A/D}^2 \sum_{n=0}^{\infty} h^2[n] + \sigma_{out}^2 + \sum_{i=1}^N \sigma_{x_i}^2 \cdot \sum_{n=0}^{\infty} (g_i''[n])^2$$

$$\sigma_{out,total}^2 = \frac{\Delta^2}{12} \left( 2 + \sum_{i=1}^N w_{ii}'' \right)$$

# Noise-optimal State Space structure

From p. 70 we have that the **System Matrices** for the **internally scaled filter** are  $K' = T_s^{-1}K(T_s^T)^{-1}$   
 $W' = T_s^TWT_s$

The **individual entries in  $K'$**  are therefore;  $K'_{ii} = (T_s^{-1}K(T_s^T)^{-1})_{ii}$

From p.68 we have;  $T_s^{-1} = \begin{bmatrix} k_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & k_N \end{bmatrix}$  and since  $T_s$  is diagonal, we also have  $T_s^T = T_s$  which leads to;

$K'_{ii} = (T_s^{-1}K(T_s^T)^{-1})_{ii} = k_i^2 \cdot K_{ii} = 1$  because  $K_{ii} = \frac{1}{k_i^2}$  (also derived at p.68).

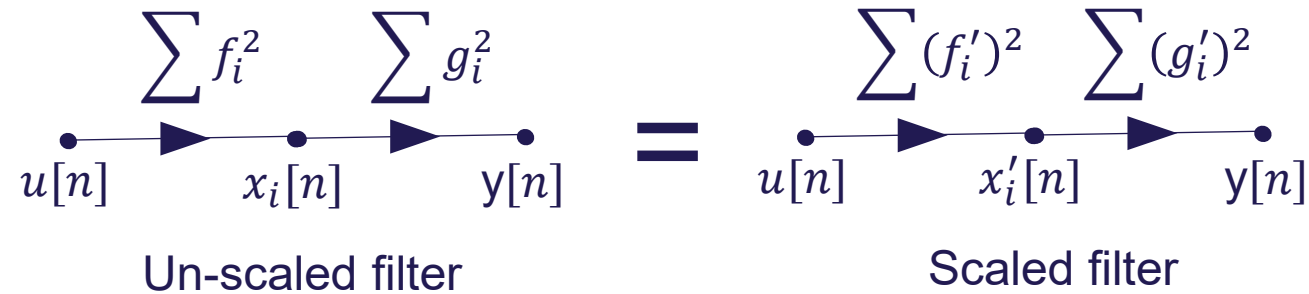
$$K' = \begin{bmatrix} 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \end{bmatrix}$$

The **interpretation** of this result is very important...! It tells us that the square-sum of the impulse responses **from input to ALL internal variables equals 1**, and thus  $\sigma_{w_k}^2 = \sigma_{input}^2$  for  $k = 1..N$ , and thus there is a **full utilization of the dynamic range in all internal variables**, thus improving the SNR.

This is different as compared to  $L_2$  scaling of a Direct Form structure, where this is true for only one internal variable (see p. 49).

# Final remarks on the State Space filter

It can be shown that  $W'_{ii} \cdot K'_{ii} = W_{ii} \cdot K_{ii}$ , which we will denote; **transformation in-variant products**.



It tells us that **internal scaling does not change the output variance**.

So, for the State Space filter, we can modify the internal variable, independently from the output.

# Final remarks on the State Space filter

Further, it can be shown that there exists a noise-optimal transformation matrix,  $T_{opt}$ , which enables the least possible noise variance at the output. The overall transformation scheme is the same as introduced on p. 70;

$$K' = T_{opt}^{-1} K (T_{opt}^T)^{-1}$$

$$W' = T_{opt}^T W T_{opt}$$

Without further arguments, we claim that  $T_{opt}$  is the matrix, which transform  $K$  and  $W$  into diagonal matrices.

Why is that...??

Let's give an intuitive explanation.

# Final remarks on the State Space filter

$$K_{ij} = \sum_{n=0}^{\infty} f_i[n] \cdot f_j[n]$$

Remember (p. 67) that

$$W_{ij} = \sum_{n=0}^{\infty} g_i[n] \cdot g_j[n]$$

These are **cross-correlations** between the impulse responses 1) from input to the internal variables, and 2) from the internal variables to the output.

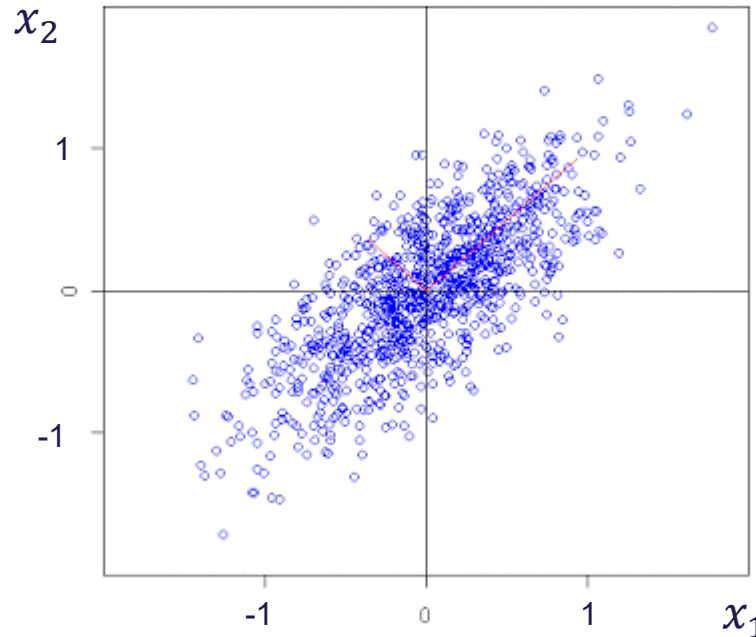
Therefore, if we can obtain  $K = \begin{bmatrix} K_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & K_{NN} \end{bmatrix}$  and  $W = \begin{bmatrix} W_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & W_{NN} \end{bmatrix}$  then **all the cross-correlations are equal to 0** (the diagonal represents the auto-correlations).

If the  $K$  cross-correlations are all 0, then for a broad-band ("white") input signal, the internal variables  $x_i[n]$  and  $x_j[n]$  are un-correlated. Similarly, if the  $W$  cross-correlations are all 0, then all the sequences being added at the output are also independent – and thus, the inherent noise sequences (which are also "white") will cancel each other, leading to the noise-optimal filter. We cannot design a fixed-point filter which has a better SNR on the output.



# Final remarks on the State Space filter

If the State Space filter is " $L_2$  scaled" using the  $T_s$  matrix (p. 68), then we typically will see **correlation between the internal variables  $x_1$  and  $x_2$  which is elliptic shaped.**



If, on the other hand, we scale the filter using the  $T_{opt}$  matrix, then the **correlation ellipse becomes a circle.**

**...and there is much more**

However, it is enough for today, so..

Thanks a lot for your attention – and now please questions.

# Essential book reference

- A.V. Oppenheim and R.W. Schaffer, "Discrete-Time Signal Processing", 3rd ed., Pearson, 2014
- L.R. Rabiner and B. Gold, "Theory and Application of Digital Signal Processing", Prentice-Hall, 1975
- R.A. Roberts and C.T. Mullis, "Digital Signal Processing", Addison-Wesley, 1987
- L.B. Jackson, "Digital Filters and Signal Processing", 3rd ed., Kluwer, 1996
- E.C. Ifeachor and B.W. Jervis, "Digital Signal Processing – A Practical Approach", Addison-Wesley, 1993