DSP-CIS

Part-II : Filter Design & Implementation

Chapter-6 : Filter Implementation

Marc Moonen
Dept. E.E./ESAT-STADIUS, KU Leuven
marc.moonen@kuleuven.be
www.esat.kuleuven.be/stadius/

Filter Design Process

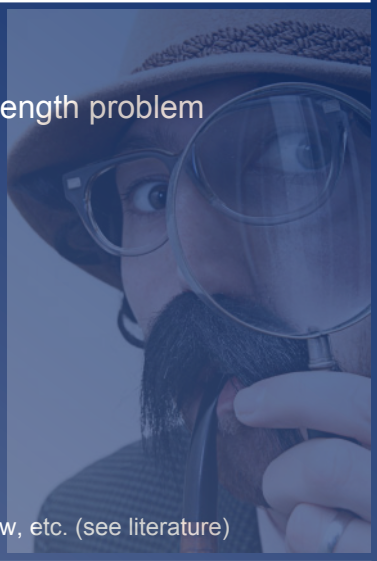
- **Step-1 : Define filter specs**
  Pass-band, stop-band, optimization criterion,…
- **Step-2 : Derive optimal transfer function**
  FIR or IIR design — **Chapter-4**
- **Step-3 : Filter realization** (block scheme/flow graph)
  Direct form realizations, lattice realizations,… — **Chapter-5**
- **Step-4 : Filter implementation** (software/hardware)
  Finite word-length issues, … — **Chapter-6**
  Question: implemented filter = designed filter ?

'You can't always get what you want' -Jagger/Richards (?)

Chapter-6 : Filter Implementation

- **Introduction**
  Filter implementation & finite wordlength problem
- **Coefficient Quantization**
- **Arithmetic Operations**
  Quantization noise
  Statistical Analysis
  Limit Cycles
  ~~Scaling~~

- **PS: Short version, does not include…**
  Fixed & floating point representations, overflow, etc. (see literature)



Introduction

Back to Chapter-5…

Q:Why bother

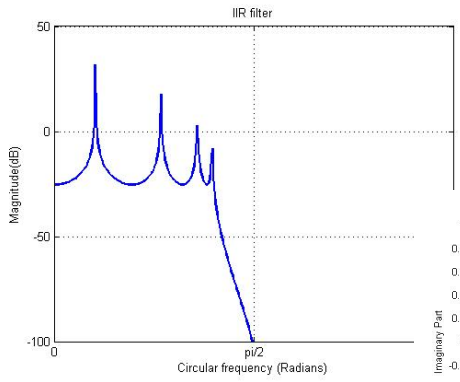
about many different realizations for one and the same filter?

Introduction

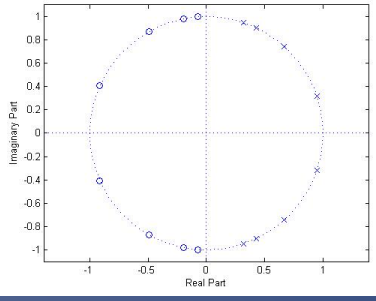
Filter implementation & finite word-length problem

- So far have assumed that signals/coefficients/arithmetic operations are represented/performed with infinite precision
- In practice, numbers are represented only to a finite precision, hence signals/coefficients/arithmetic operations are subject to quantization (truncation/rounding/...) errors
- Quantization effects relevant in fixed-point implementations with a `short' word-length (versus less of an issue when `sufficiently long' word-length is used (e.g. 24 bits), or with floating-point representations and arithmetic)
- Investigate impact of…
  - **Quantization of filter coefficients**
  - **Quantization in arithmetic operations**

Introduction: Example



Transfer function



Poles & zeros

```
% IIR Elliptic Lowpass filter designed using
% ELLIP function.
% All frequency values are in Hz.
Fs = 48000;     % Sampling Frequency
L    = 8;       % Order
Fpass = 9600;  % Passband Frequency
Apass = 60;     % Passband Ripple (dB)
Astop = 160;    % Stopband Attenuation (dB)
```

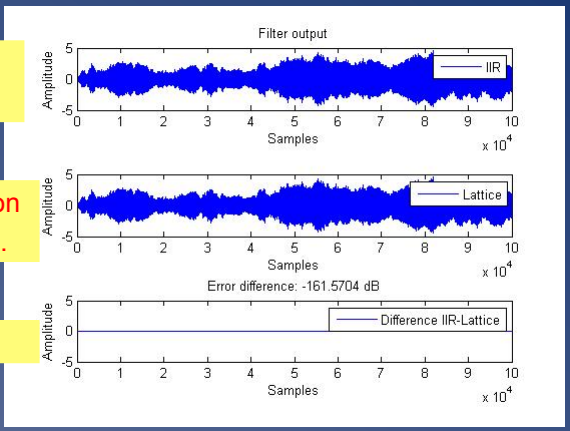
Introduction: Example

Filter outputs…

Direct form realization @ infinite precision…

Lattice-ladder realization @ infinite precision…

Difference…



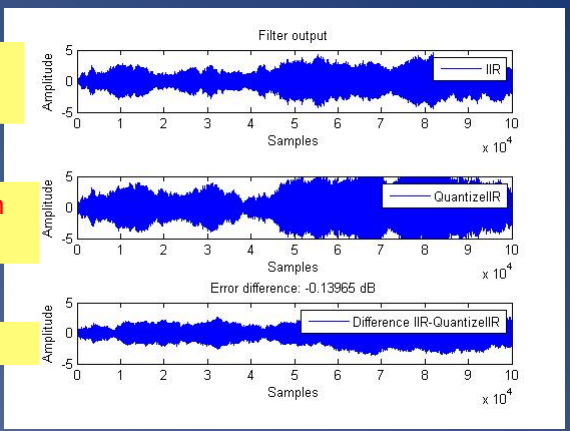
Introduction: Example

Filter outputs…

Direct form realization @ infinite precision…

Direct form realization @ 8-bit precision…

Difference…



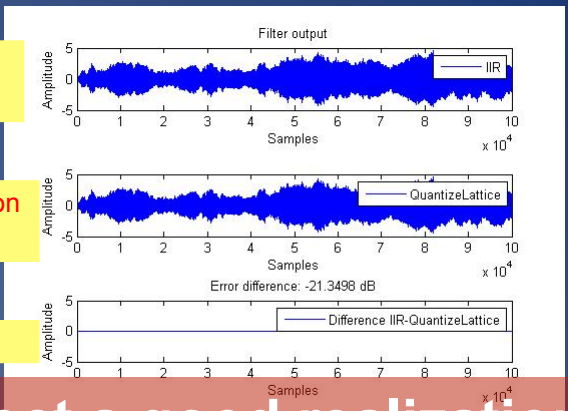
Introduction: Example

Filter outputs…

Direct form realization @ infinite precision…

Lattice-ladder realization @ 8-bit precision…

Difference…



Better select a good realization !

Coefficient Quantization

Coefficient quantization problem

- Filter design in Matlab (e.g.) provides filter coefficients to 15 decimal digits (such that filter meets specifications)
- For implementation, have to quantize coefficients to the word-length used for the implementation
- As a result, implemented filter may fail to meet specifications…

Example from

PEARSON *Discrete-Time Signal Processing*, Third Edition
Alan V. Oppenheim • Ronald W. Schafer

Figure 6.47 IIR coefficient quantization example. (a) Log magnitude for unquantized elliptic bandpass filter. (b) Magnitude in passband for unquantized (solid line) and 16-bit quantized cascade form (dashed line).

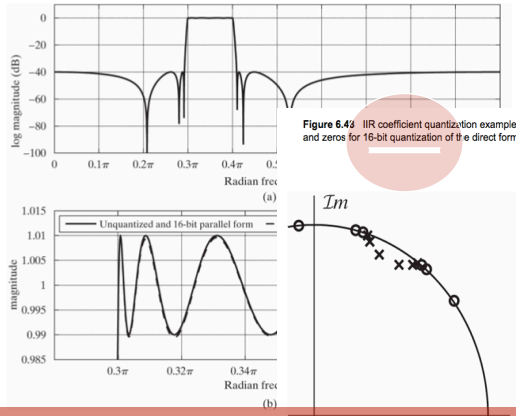
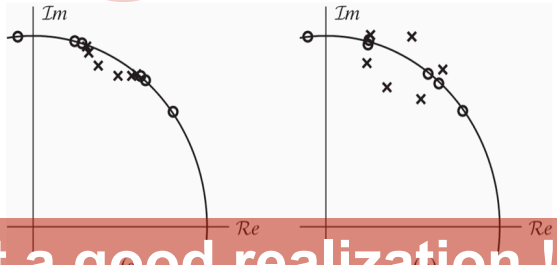


Figure 6.48 IIR coefficient quantization example. (a) Poles and zeros of $H(z)$ for unquantized coefficients. (b) Poles and zeros for 16-bit quantization of the direct form coefficients.



Better select a good realization !

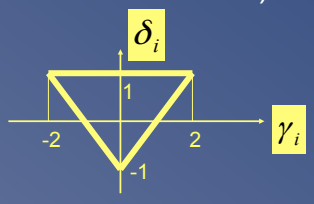
Coefficient Quantization

Coefficient quantization effect on pole locations

- Example : 2nd-order system (e.g. for cascade/direct form realization)

$$H_i(z) = \frac{1 + \alpha_i . z^{-1} + \beta_i . z^{-2}}{1 + \gamma_i . z^{-1} + \delta_i . z^{-2}}$$

`Triangle of stability' : denominator polynomial is stable (i.e. roots inside unit circle) iff coefficients lie inside triangle…



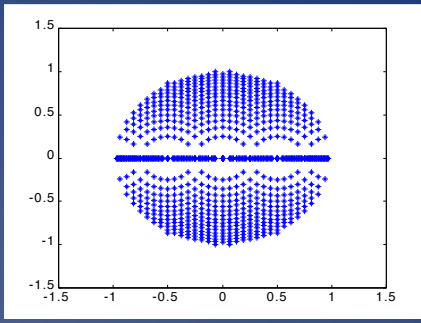
Proof: Apply Schur-Cohn stability test (see Chapter-5).

Coefficient Quantization

- Example (continued)

  With 5 bits per coefficient, all possible `quantized' pole positions are...

```
for γ_i = −2 : 0.1250 : 2
  for δ_i = −1 : 0.0625 : 1
    plot(poles)  (if stable)
  end
end
```

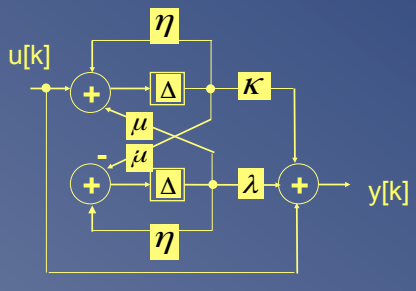


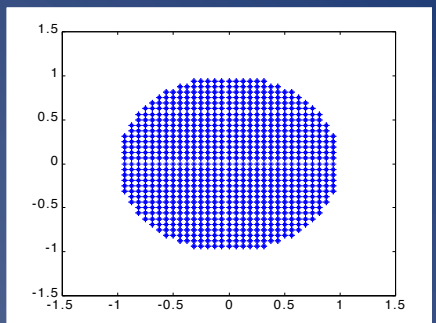
Low density of `quantized' pole locations at z=1, z=-1,
hence problem for narrow-band LP and HP filters in (transposed) direct form (see Chapter-4).

Coefficient Quantization

- Example (continued)

  Possible remedy: `coupled realization'
  Poles are $\eta \pm j.\mu$ where $-1 < \eta, \mu < 1$ are realized/quantized
  hence 'quantized' pole locations are (5 bits)





coefficient precision = pole precision

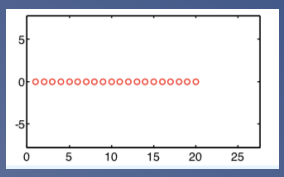
Coefficient Quantization

Coefficient quantization effect on pole locations

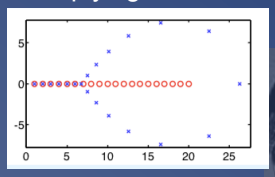
- Fact: For high-order polynomials, roots can be very sensitive to small changes in coefficient values
- Famous example: Wilkinson's polynomial

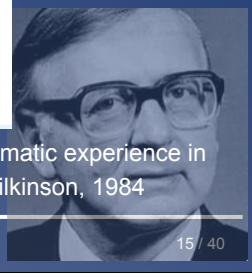
$$A(z) = \prod_{n=1}^{20}(z-n)$$

Roots



Roots after multiplying coefficient of z^{19} by 1.000001





"Speaking for myself I regard it as the most traumatic experience in my career as a numerical analyst" James H. Wilkinson, 1984

Coefficient Quantization

Coefficient quantization effect on pole locations

- Higher-order systems (first-order analysis)

$$\text{polynomial}: \ 1 + a_1.z^{-1} + a_2.z^{-2} + \ldots + a_L.z^{-L}$$
$$\text{roots are}: p_1, p_2, \ldots, p_L$$

$$\text{`quantized' polynomial}: \ 1 + \hat{a}_1.z^{-1} + \hat{a}_2.z^{-2} + \ldots + \hat{a}_L.z^{-L}$$
$$\text{`quantized' roots are}: \ \hat{p}_1, \hat{p}_2, \ldots, \hat{p}_L$$

$$\hat{p}_l - p_l \approx -\sum_{i=1}^{L} \frac{p_l^{L-i}}{\prod_{j \neq l}(p_l - p_j)}.(\hat{a}_i - a_i)$$

➔ Tightly spaced poles (e.g. for narrow band filters) imply high sensitivity of pole locations to coefficient quantization

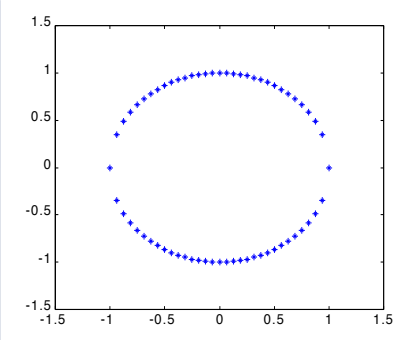
➔ Hence preference for low-order systems (e.g. in parallel/cascade)

Coefficient Quantization

Coefficient quantization effect on zero locations

- Analog filter design + bilinear transformation often lead to numerator polynomial of the form (e.g. 2nd-order cascade realization)
  $1-2\cos\theta_i.z^{-1}+z^{-2}$ hence with zeros always on the unit circle

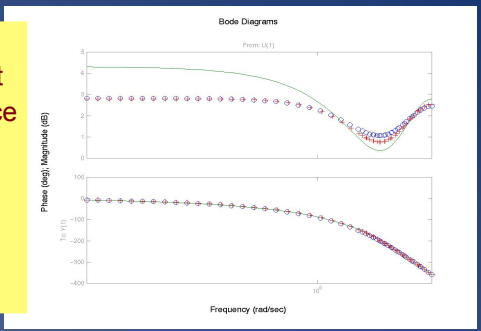
Quantization of the coefficient $2\cos\theta_i$ shifts zeros on the unit circle, which mostly has only minor effect on the filter characteristic.
Hence mostly ignored…



Coefficient Quantization

Coefficient quantization in lossless lattice realizations

o = original transfer function
+ = transfer function after 8-bit truncation of lossless lattice filter coefficients
- = transfer function after 8-bit truncation of direct-form coefficients (bi's)



In lossless lattice, all coefficients are sines and cosines, hence all values between –1 and +1…, i.e. `dynamic range' and coefficient quantization error well under control.

Arithmetic Operations

Quantization noise problem

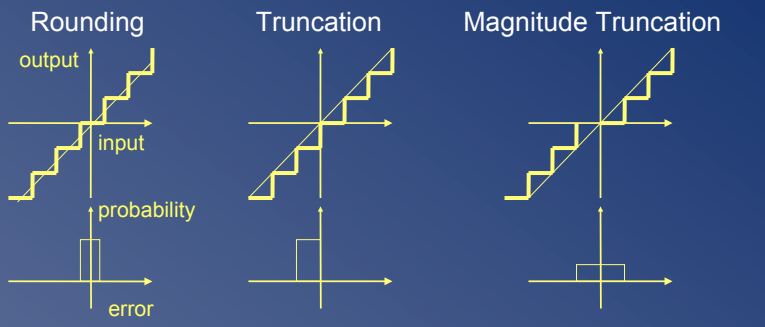
- If two B-bit numbers are added, the result is a B+1 bit number.
- If two B-bit numbers are multiplied, the result is a 2B-1 bit number.
- Typically (especially so in an IIR (feedback) filter), the result of an addition/multiplication has to be represented again as a B'-bit number (e.g. B'=B). Hence have to remove least significant bits (*).
- Rounding/truncation/… to B' bits introduces **quantization noise**.

- The effect of quantization noise is usually analyzed in a statistical manner (see p.20-25)
- Quantization, however, is a deterministic non-linear effect, which may give rise to **limit cycle oscillations** (see p.26-30)

(*) ..and/or most significant bits - not considered here

Quantization Noise / Statistical Analysis

Quantization mechanisms



Rounding	Truncation	Magnitude Truncation
mean=0	mean=(-0.5)LSB (biased!)	mean=0
variance=(1/12)LSB^2	variance=(1/12)LSB^2	variance=(1/6)LSB^2

PS: …assuming input to quantization is uniformly distributed (is it?)

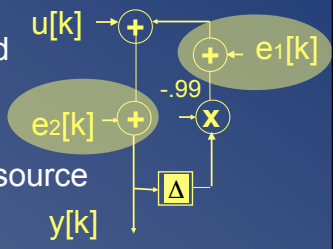
Quantization Noise / Statistical Analysis

Statistical analysis is based on the following **assumptions** :

- Each quantization error is random, i.e. uncorrelated/independent of the number that is quantized, and with uniform probability distribution function (see previous slide) (ps: model more suited for multipliers than for adders)
- Successive quantization errors at the output of a given multiplier/adder are uncorrelated/independent (=white noise assumption)
- Quantization errors at the outputs of different multipliers/adders are uncorrelated/independent (=independent sources assumption)

➔A noise source (representing quantization) is inserted after each (ideal, then) **multiplier/adder**

➔Since the filter is a **linear filter** the output noise generated by each noise source is added to the output signal



Quantization Noise / Statistical Analysis

Effect on the output signal of a noise generated at a particular point in the filter is computed as follows:

- Noise is e[k], assumed white (=flat PSD) with mean & variance μ_e, σ_e^2
- Transfer function from from e[k] to filter output is G(z),g[k] (= 'noise transfer function')
- Noise mean at the output is $\mu_e.('\mathrm{DC-gain}') = \mu_e.G(z)\big|_{z=1}$
- Noise variance at the output is

$$\sigma_e^2.(\text{`noise-gain'}) = \sigma_e^2.(\frac{1}{2\pi}\int_{-\pi}^{\pi}\left|G(e^{j\omega})\right|^2 d\omega)$$

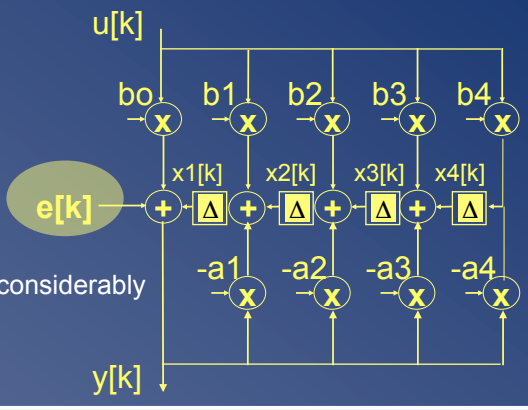
$$= \sigma_e^2.\sum_{k=0}^{\infty}\left|g[k]\right|^2 = \sigma_e^2.\left\|g\right\|_2^2$$

Repeat procedure for each noise source…

Quantization Noise / Statistical Analysis

PS: In a transposed direct form realization all noise transfer functions are equal (up to delay), hence all noise sources can be lumped into one equivalent noise source

…which simplifies analysis considerably



Quantization Noise / Statistical Analysis

PS: In a direct form realization all noise sources can be lumped into two equivalent noise sources

…which simplifies analysis considerably

Quantization Noise / Statistical Analysis

PS: Quantization noise of ***A/D-converters*** can be modeled/ analyzed in a similar fashion.

Noise transfer function is filter transfer function H(z)

PS: Quantization noise of ***D/A-converters*** can be modeled/ analyzed in a similar fashion.

Non-zero quantization noise if D/A converter wordlength is shorter than filter wordlength.

Noise transfer function = 1

Quantization Noise / Limit Cycles

Statistical analysis is simple/convenient, but quantization is truly a **non-linear** effect, and should be analyzed as a **deterministic** process

Though very difficult, such analysis may reveal odd behavior :

Example: y[k] = -0.625.y[k-1]+u[k]
4-bit rounding arithmetic
input u[k]=0, y[0]=3/8
output y[k] = 3/8, -1/4, 1/8, -1/8, 1/8, -1/8, 1/8, -1/8, 1/8,..

Oscillations in the absence of input (u[k]=0) are called
`zero-input limit cycle oscillations'

Quantization Noise / Limit Cycles

Example: y[k] = -0.625.y[k-1]+u[k]
4-bit truncation (instead of rounding)
input u[k]=0, y[0]=3/8
output y[k] = 3/8, -1/4, 1/8, 0, 0, 0,.. (no limit cycle!)

Example: y[k] = 0.625.y[k-1]+u[k]
4-bit rounding
input u[k]=0, y[0]=3/8
output y[k] = 3/8, 1/4, 1/8, 1/8, 1/8, 1/8,..

Example: y[k] = 0.625.y[k-1]+u[k]
4-bit truncation
input u[k]=0, y[0]=-3/8
output y[k] = -3/8, -1/4, -1/8, -1/8, -1/8, -1/8,..

Conclusion: weird, weird, weird,… !

Quantization Noise / Limit Cycles

- Limit cycle oscillations are clearly **unwanted** (e.g. may be audible in speech/audio applications)
- Limit cycle oscillations can only appear if the filter has feedback. Hence **FIR filters** cannot have limit cycle oscillations
- Mathematical analysis is very difficult ☹

Quantization Noise / Limit Cycles

- Truncation often helps to avoid limit cycles (e.g. **magnitude truncation**, where absolute value of quantizer output is never larger than absolute value of quantizer input (=`passive quantizer'))
- Some filter realizations can be made limit cycle free, e.g. coupled realization, orthogonal filters (details omitted)

Quantization Noise / Limit Cycles

Here's the good news:

For a..
- lossless lattice realization of a general IIR filter
- lattice-ladder realization of a general IIR filter

and when
- magnitude truncation (=`passive quantization') is used

the filter is **guaranteed** to be free of limit cycles !

(details omitted)

Intuition: magnitude truncation consumes energy/power, orthogonal filter operations do not generate power to feed limit cycle

Scaling

The scaling problem

- Finite word-length implementation implies maximum representable number. Whenever a signal (output or internal) exceeds this value, ***overflow*** occurs.
- Digital overflow may lead (e.g. in 2's-complement arithmetic) to polarity reversal (instead of saturation such as in analog circuits), hence may be very harmful.
- Avoid overflow through proper signal ***scaling***, implemented by bit shift-operations applied to signals, or by scaling of filter coefficients, or..
- Scale transfer function may be c.H(z) instead of H(z) (hence need proper tracking of scaling factors)

Skip this slide

Scaling

Time domain ('deterministic') scaling:

- Assume input signal is bounded in magnitude $|u[k]| \le u_{\max}$

  i.e. u-max is the largest number that can be represented in the 'words' reserved for the input signal

- Then output signal is bounded by

$$|y[k]| = \left| \sum_{\bar{k}=0}^{\infty} h[\bar{k}]\, u[k-\bar{k}] \right| \le \sum_{\bar{k}=0}^{\infty} |h[\bar{k}]| . |u[k-\bar{k}]| \le u_{\max} . \sum_{\bar{k}=0}^{\infty} |h[\bar{k}]| = u_{\max} . \|h\|_1$$

ps : stability of the filter h implies that its 1-norm is finite

Skip this slide

Scaling

Time domain ('deterministic') scaling: (continued)

- Assume input signal is bounded in magnitude

$$|u[k]| \le u_{\max}$$

- Then output signal is bounded by

$$|y[k]| = u_{\max} \cdot \|h\|_1$$

- To satisfy

$$|y[k]| \le y_{\max}$$

i.e. $y_{\max}$ is the largest number that can be represented in the words reserved for the output signal

we have to scale H(z) to c.H(z), with

$$c = \frac{y_{\max}}{u_{\max} \cdot \|h\|_1}$$

Scaling

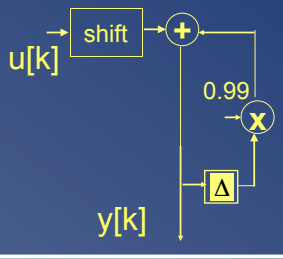
- Example:

$$H(z) = \frac{1}{1 - 0.99 z^{-1}}$$

$$\|h\|_1 = \ldots = \frac{1}{1 - 0.99} = 100$$

- assume u[k] produced by 12-bit A/D-converter
- assume we use 16-bit arithmetic for u[k] & multiplier

$$c = \frac{2^{16}}{2^{16} \cdot \|h\|_1} = 0.16 > \frac{1}{2^3}$$



- hence input u[k] has to be shifted by 3 bits to the right before entering the filter (=remove 3 LSB's)

Scaling

Time domain ('deterministic') scaling:

$$c = \frac{y_{\max}}{U_{\max} \cdot \|h\|_1}$$

Frequency domain ('deterministic') scaling:

- Frequency-domain analysis leads to alternative scaling factors, e.g...

$$c = \frac{y_{\max}}{U_{\max} \cdot \|H\|_1}, \qquad U_{\max} = \max_{\omega} |U(e^{j\omega})|, \qquad \|H\|_1 = \frac{1}{2\pi} \cdot \int_{-\pi}^{\pi} |H(e^{j\omega})| d\omega$$

...or...

$$c = \frac{y_{\max}}{H_{\max} \cdot \|U\|_1}, \qquad H_{\max} = \max_{\omega} |H(e^{j\omega})|, \qquad \|U\|_1 = \frac{1}{2\pi} \cdot \int_{-\pi}^{\pi} |U(e^{j\omega})| d\omega$$

…which may (or may not) be less conservative

-> proper choice of scaling factor in general is a difficult problem

Skip this slide

Scaling

L2-scaling: (`scaling in L2 sense', `probabilistic scaling')

- Time-domain scaling is simple & guarantees that overflow will never occur, but often over-conservative (=too small c)
- Define L2-norm :

$$\|h\|_2 = \sqrt{\sum_{k=0}^{\infty} |h[k]|^2} = \sqrt{\frac{1}{2\pi} \int_{-\pi}^{\pi} |H(e^{j\omega})|^2 d\omega}$$

- If input signal u[k] is (`wide sense') stationary signal with power spectral density (`PSD', i.e. Fourier transform of its covariance sequence) $P_u(\omega)$
  then **variance** of output signal y[k] is bounded:

$$\sigma_y^2 \le \|h\|_2^2 . \max_{-\pi \le \omega \le \pi} P_u(\omega)$$

- Leads to scaling factor

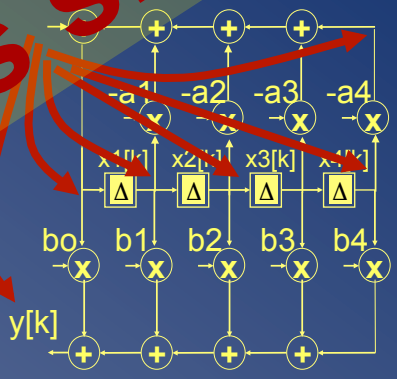
$$c = \frac{\alpha . y_{\max}}{\sqrt{\max_{\omega} P_u(\omega)} . \|h\|_2}$$

where alpha defines overflow probability

Skip this slide

Scaling

- So far considered scaling of H(z), i.e. transfer function from u[k] to y[k].
- In practice, have to consider ***overflow and scaling of each internal signal***, i.e. scaling of transfer function from u[k] to each and every internal signal !
  May require quite some thinking..
  (but doable)



Skip this slide

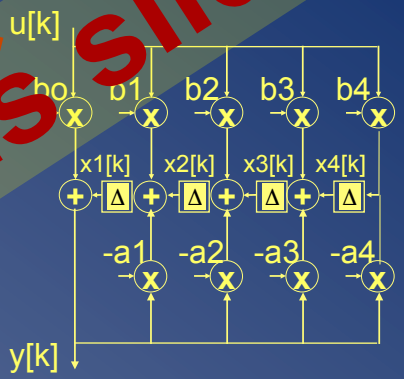
Scaling

- **Something that may help:** If 2's-complement arithmetic is used, and if the sum of K numbers (K>2) is guaranteed not to overflow, then overflows in partial sums cancel out and do not affect the final result (similar to 'modulo arithmetic')
- **Example:**
  if x1+x2+x3+x4 is guaranteed not to overflow, then if in (((x1+x2)+x3)+x4) the sum (x1+x2) overflows, this overflow can be ignored without affecting the final result
- As a result, in a **direct form** realization only 2 signals have to be considered in view of scaling :

Skip this slide

Scaling

- As a result (2), in a **transposed direct form** realization, only 1 signal has to be considered in view of scaling.



Skip this slide

Scaling

- As a result (3), in a **state space** realization only output signal + internal states have to be considered in view of scaling:

$$x[k+1] = A.x[k] + B.u[k]$$
$$y[k] = C.x[k] + D.u[k]$$

-Matrix $\begin{bmatrix} 0 & B & AB & A^2B & A^3B & \dots \end{bmatrix}$

defines transfer from input to internal states
(i-th row has impulse response from input u[k] to i-th internal state)

-Internal states can be re-scaled by means of diagonal transformation T, such that

$$\begin{bmatrix} 0 & \tilde{B} & \tilde{A}\tilde{B} & \tilde{A}^2\tilde{B} & \tilde{A}^3\tilde{B} & \dots \end{bmatrix} = \dots = T^{-1}.\begin{bmatrix} 0 & B & AB & A^2B & A^3B & \dots \end{bmatrix}$$

-If T is such that all rows of this tilde-matrix have equal L2-norm, then overflow probability is the same in all states.
This is referred to as 'L2 scaled realization'

Skip this slide