9. IIR filters

1. Recursive filters
2. Chebyshev filters

[Smith, ch. 19, 20]

1. Recursive Filters

1.1 The recursion equation

Information that is available to **calculate *y*[*n*]** :

- The *input signal* samples: $x_n, x_{n-1}, x_{n-2}, \ldots$.

AND

- the *previously* calculated output signal values:

$y_n, y_{n-1}, y_{n-2}, \ldots,$.

The recursion equation:

$$y_n = a_0 \cdot x_n + a_1 \cdot x_{n-1} + a_2 \cdot x_{n-2} + \cdots + a_p \cdot x_{n-p} +$$
$$+ b_1 \cdot y_{n-1} + b_2 \cdot y_{n-2} + \cdots + b_q \cdot y_{n-q}$$

where

$x[n]$ is the input signal,
$y[n]$ is the output signal, and
the a's and b's are coefficients.

The filter that use a **recursion equation** are called **recursive filters**.

What happens when a **delta function** is passed through a recursive filter?

- The output is the filter's *impulse response*, and will typically be a **sinusoidal** oscillation that **exponentially decays**.
- Since this impulse response is **infinitely long**, recursive filters are often called ***infinite impulse response*** (IIR) filters.

Recursive filters ***convolve*** the input signal with a **very long** filter kernel (implicit kernel), although only **few coefficients** are involved (explicit coefficients).

1.2 Single pole recursive filter

The relationship between the recursion **coefficients** and the filter's **response** is given by the **z-transform**.

A **single pole** low-pass filter uses only two coefficients: a_0 , b_1 :

$$y_n = a_0 \cdot x_n + b_1 \cdot y_{n-1}$$

Example 9.1

An example of a **single pole** low-pass filter that uses $a_0 = 0.15$, $b_1 = 0.85$. (fig. 1).

A step function is transformed by this low-pass filter to a smooth, asymptotic rise to a final 100% step value.

Remark: for the digital filter, the step response starts to rise at sample 10, i.e. y_{10}= 0.15, and then it slowly, asymptotically grows to 1.

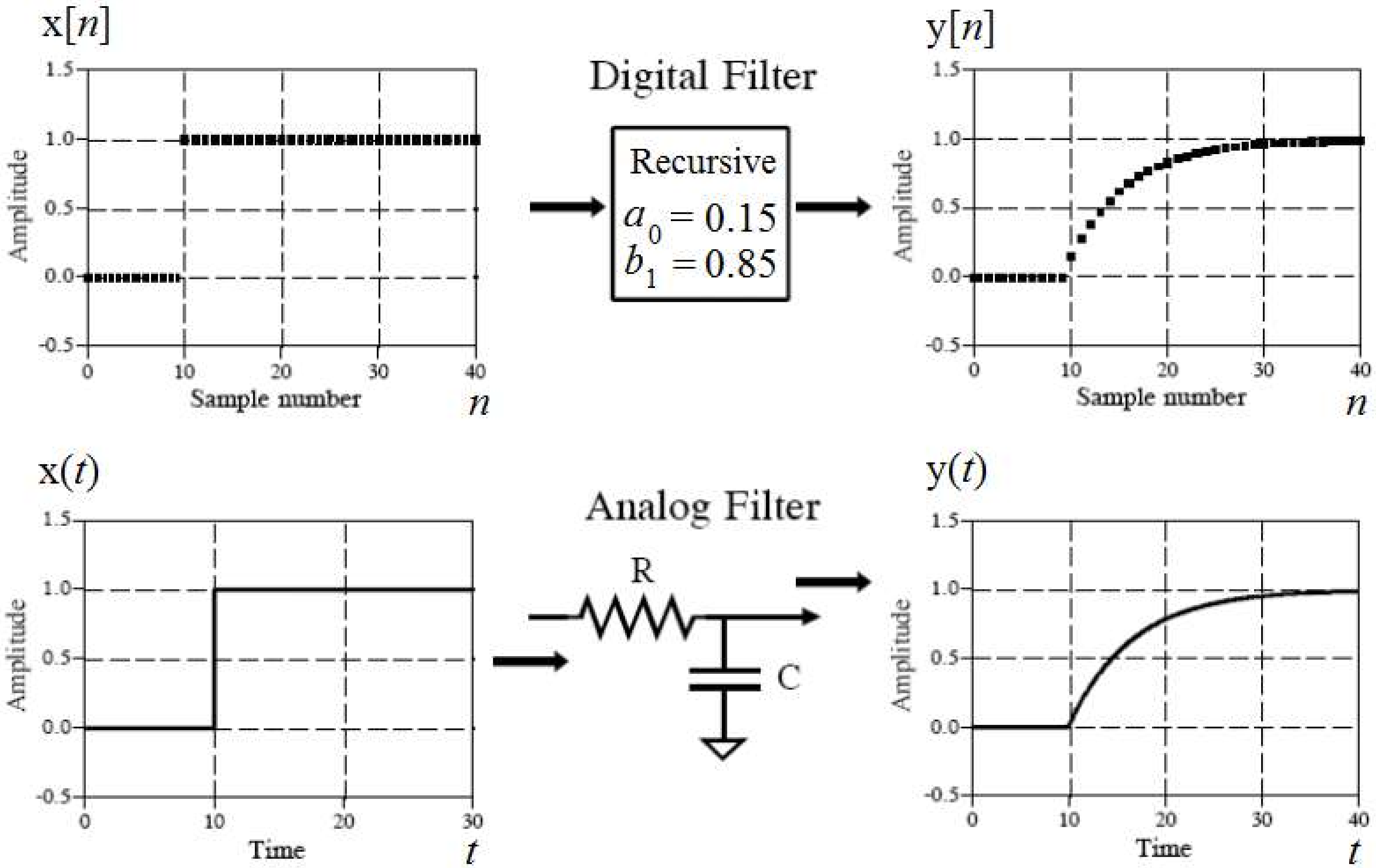


Fig. 1 Single pole low-pass filter. A low-pass recursive filter smoothes the edge of a step input, just as an electronic RC filter.

A **single pole** high-pass filter uses three coefficients: a_0 , a_1, b_1 :

$$y_n = a_0 \cdot x_n + a_1 \cdot x_{n-1} + b_1 \cdot y_{n-1}$$

Example 9.2.

An example of a **single pole** high-pass filter (fig. 2): a filter with three coefficients: $a_0 = 0.93$, $a_1 = -0.93$, $b_1 = 0.86$.

This digital filter simulates an electronic RC high-pass filter.

Remark: for the digital filter, the step response starts with the maximum value at sample 10, i.e. $y_{10} = 0.93$, and next it slowly, asymptotically decays to zero.

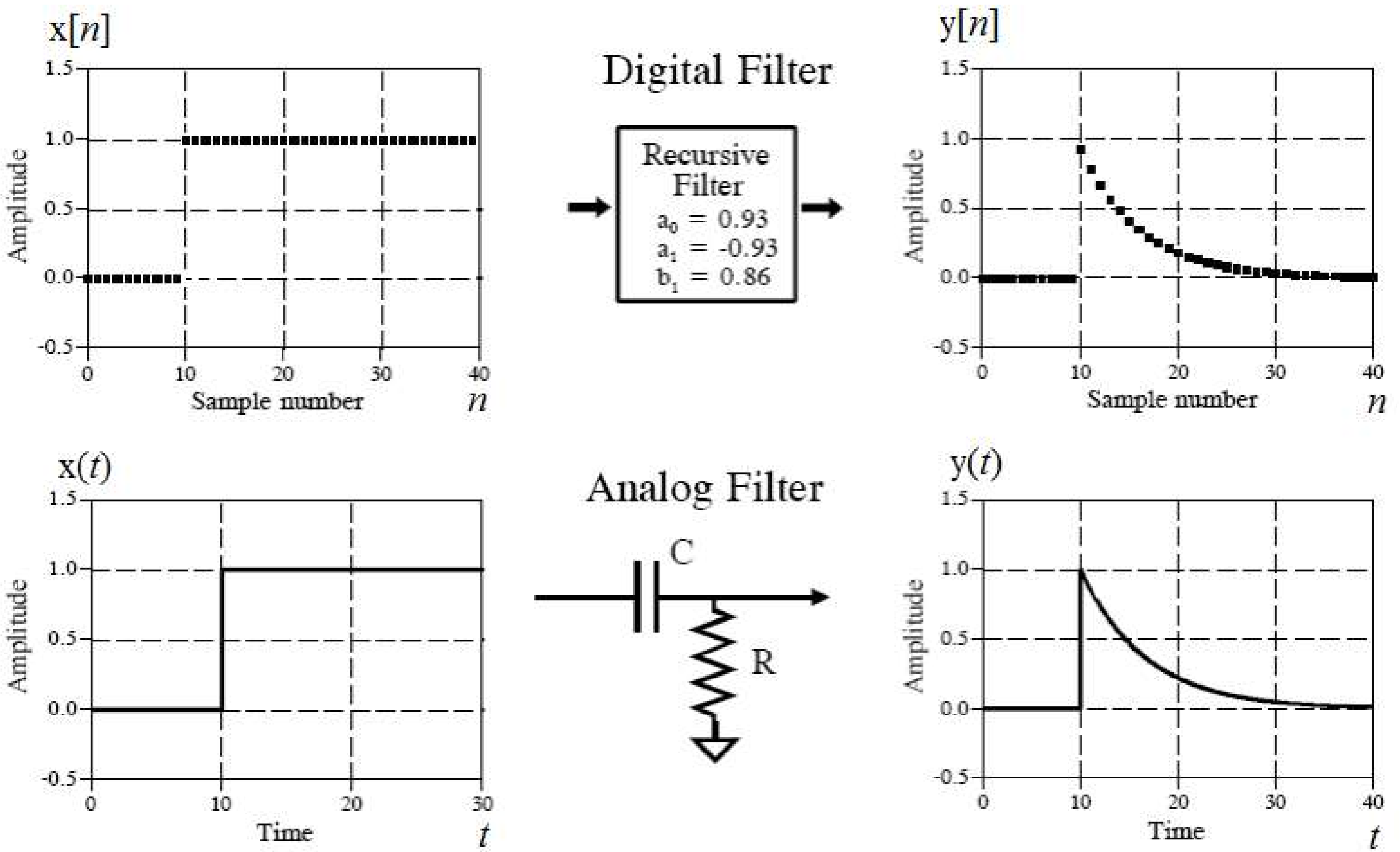


Fig. 2. Single pole high-pass filter. Proper coefficient selection can also make the recursive filter mimic an electronic RC high-pass filter.

Decay parameter ***k***

- In the single pole low-pass filter the filter's response is controlled by the **parameter**, ***k***, a value between zero and one:

$$a_0 = 1 - k;$$
$$b_1 = k;$$

- Similarly, in the single pole high-pass filter:

$$a_0 = \frac{1+k}{2}; \quad a_1 = -\frac{1+k}{2};$$
$$b_1 = k$$

- ***k*** is the ***decay ratio*** between adjacent pairs of step response samples: k is the ratio between (y_{n+1} - y_n) and (y_n - y_{n-1}):

$$k = \frac{y_{n+1} - y_n}{y_n - y_{n-1}}.$$

- The filter becomes ***unstable*** if ($k > 1$). The value for k can be directly specified, or found from the **desired *time constant*** of the RC filter.

Time constant of single pole filters

In a decreasing step response (high-pass filter), the filter's **time constant**, *d*, is the time required for the filter's step response to decay in value to 36.8 % ($\approx \frac{1}{e}$) of the initial maximum value.

In an increasing step response (low-pas filter), the filter's **time constant**, *d*, is the time required for the filter's step response to reach 63.2 % ($\approx 1 - \frac{1}{e}$) of its final (asymptotic) step increase value (i.e. maximum - initial value).

In digital filters

Time constant d is related to the decay k, by:

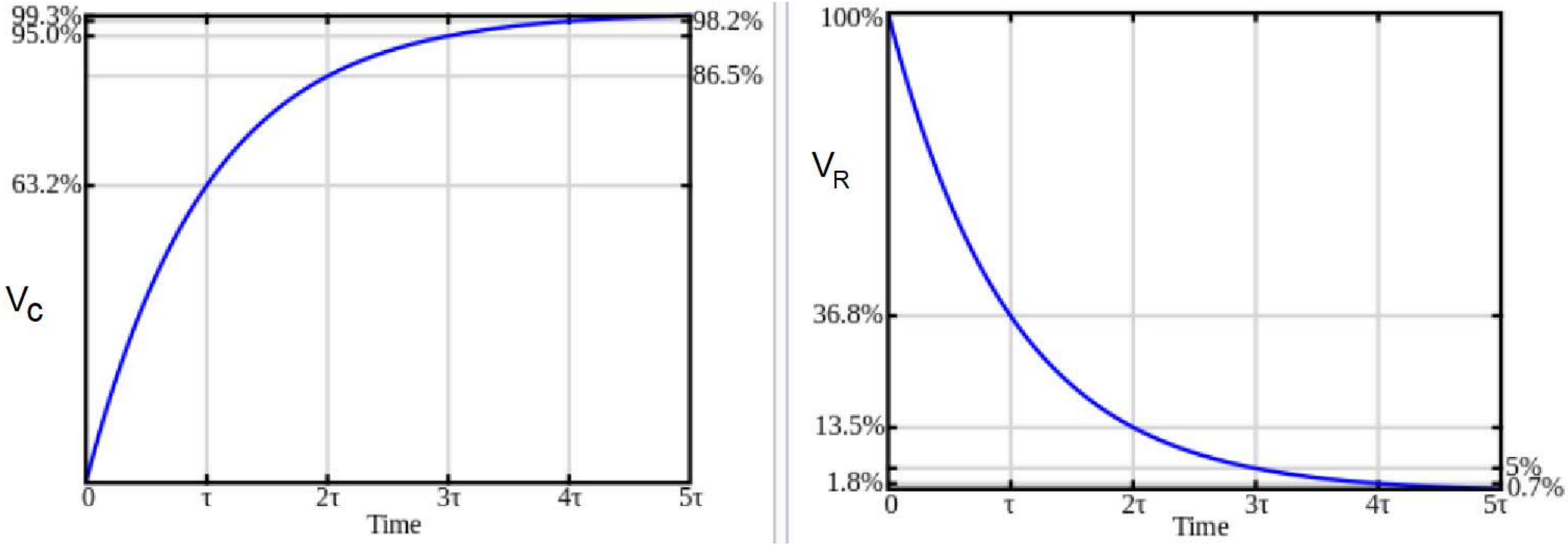
$$k = e^{-1/d}$$

$$d = -\frac{1}{\ln k}$$

For instance, a decay of $k = 0.86$,
corresponds to a time constant of $d = 6.63$ samples.

In analog RC filters

In an RC circuit composed of a single resistor and capacitor, the time constant τ (in seconds) is: $\tau = RC$,
where R is the resistance (in ohms) and C is the capacitance (in farads).



Cutoff frequency of single pole filters -

There is a fixed relationship between k and the -3dB *cutoff frequency*, f_C , a value between 0 and 0.5, of the digital filter:

$$k = e^{-2\pi f_C}$$

$$f_c = -\frac{\ln k}{2\pi}$$

Hence, k may be set in accordance to

1. the time constant,
2. the cutoff frequency, or
3. just directly.

Exercises 9

Task 9-1

Let a **single pole** low-pass IIR filter be given with recursive coefficient, a_0= 0.2.

a) Specify the remaining recursive coefficients of this filter.
b) Specify its step response (by 10 samples) – assume the step function, x[n] = 1, for n >=1,
c) Estimate the parameters of this filter: time constant and cutoff frequency.

Task 9-2

Let a **single pole** high-pass IIR filter be given with coefficient: a_0 = 0.90.

a) Specify the remaining recursive coefficients of this filter.
b) Estimate the parameters of this filter: time constant and cutoff frequency.
c) Specify its step response (by 10 samples) - assume the step function x[n] = 1 for n>=1.

Impulse response of single-pole IIR filters

In principle, the impulse response is **infinitely long**; however, it decays below the single precision round-off noise after about 15 to 20 time constants.

For example, when the time constant of the filter is $d = 6.63$ samples, the impulse response can be limited to about 128 samples.

Frequency response of single-pole IIR filters

The frequency response of single-pole recursive filters is not always what you expect. For example, in Fig. 3(c), the $f_C = 0.25$ curve is quite useless. Main reasons are:

- aliasing,
- round-off noise, and
- the nonlinear phase response.

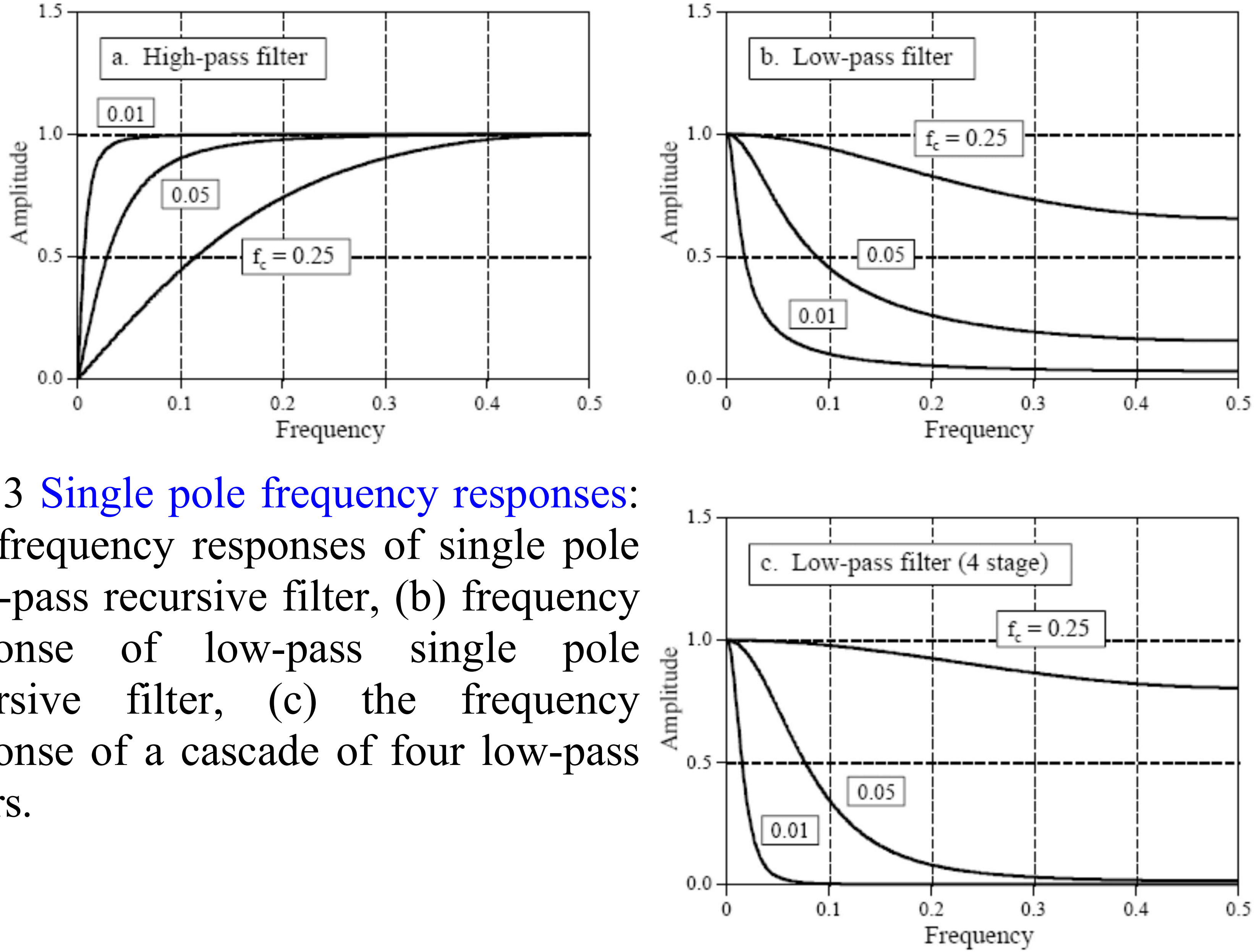


Fig. 3 Single pole frequency responses: (a), frequency responses of single pole high-pass recursive filter, (b) frequency response of low-pass single pole recursive filter, (c) the frequency response of a cascade of four low-pass filters.

Cascade of single pole filters

Single pole recursive filters have **little ability** to separate one band of frequencies from another:

- They perform **well** in the time domain, and **poorly** in the frequency domain.
- The frequency response can be improved slightly by **cascading** several stages:
  - the signal can be passed through the filter several times.
  - the z-transform can be used to find the recursion coefficients that combine the cascade into a single stage.

The **four stage** low-pass filter is comparable to the Blackman and Gaussian filters, but with a much faster execution speed.

The coefficients of a four stage single pole low-pass filter:

$$y_n = a_0 \cdot x_n + b_1 \cdot y_{n-1} + b_2 \cdot y_{n-2} + b_3 \cdot y_{n-3} + b_4 \cdot y_{n-4}$$

Where

$a_0 = (1 - k)^4$

$b_1 = 4k$

$b_2 = - 6k^2$

$b_3 = 4k^3$

$b_4 = - k^4$

k is the decay parameter.

1.3 Narrow-band filters

Two types of band-related frequency responses are available:

- the *band-pass* and
- the *band-reject* (also called a **notch filter**).

The band-pass filter has relatively large *tails* extending from the main peak. This can be improved by cascading several stages.

The band-reject (notch) filter is useful for removing narrow-band (e.g. 60 Hz) interference from time domain encoded waveforms. The narrowest bandwidth in a band-reject filter that can be obtained with single precision data is about 0.0003 of the sampling frequency.

The step response of the band-reject filter shows that the overshoot and ringing amplitudes are quite small introducing only a minor distortion to the time domain waveform.

Recursion coefficients for narrow-band filters

Band-pass filter

The coefficients of a band-pass filter:

$$y_n = a_0 \cdot x_n + a_1 \cdot x_{n-1} + a_2 \cdot x_{n-2} + b_1 \cdot y_{n-1} + b_2 \cdot y_{n-2}$$

Where

$a_0 = (1 - K)$

$a_1 = 2(K - R)\cos(2\pi f)$

$a_2 = R^2 - K$

$b_1 = 2R\cos(2\pi f)$

$b_2 = -R^2$

with

$$K = \frac{1 - 2R\cos(2\pi f) + R^2}{2 - 2\cos(2\pi f)}$$

$$R = 1 - 3BW$$

Two parameters must be chosen before using these equations:

- f, the **center frequency** of the frequency band, and
- BW, the **bandwidth** (measured at an amplitude of 0.707).

They are expressed as a fraction of the sampling rate, in the range of 0 to 0.5.
Next, calculate R, then K, and then the recursion coefficients.

Band-reject filter (a **notch** filter)

The coefficients of a band-pass filter:

$$y_n = a_0 \cdot x_n + a_1 \cdot x_{n-1} + a_2 \cdot x_{n-2} + b_1 \cdot y_{n-1} + b_2 \cdot y_{n-2}$$

Where

$$a_0 = K$$
$$a_1 = -2K \cos(2\pi f)$$
$$a_2 = K$$
$$b_1 = 2R \cos(2\pi f)$$
$$b_2 = -R^2$$

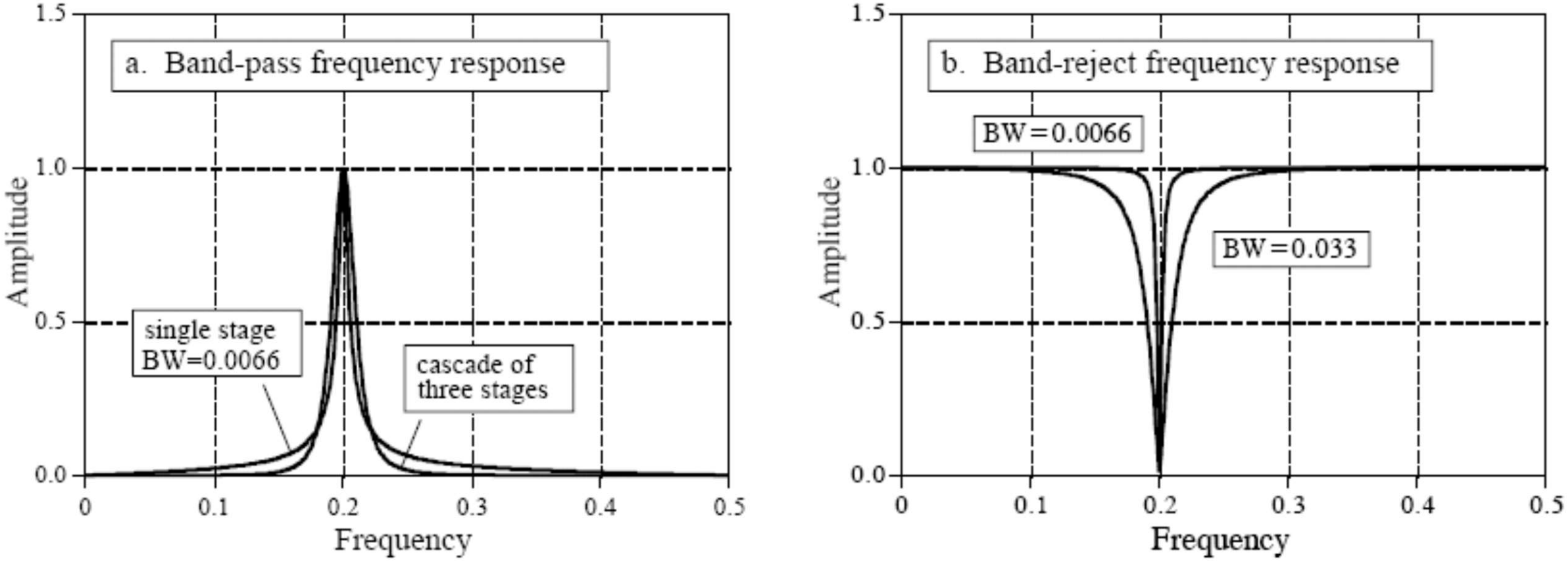
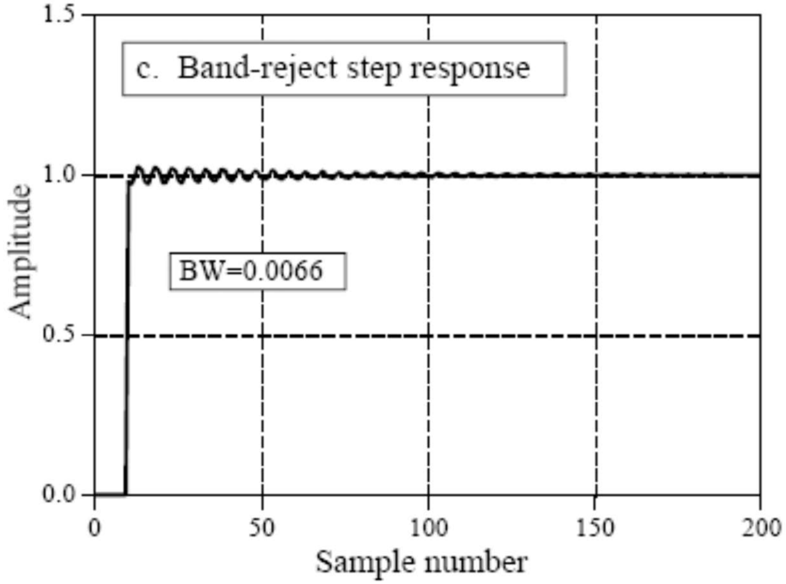


Fig. 4 Characteristics of narrow-band filters. (a) and (b) shows the frequency responses of various band-pass and band-reject filters.

The step response of the band-reject filter is shown in (c).



1.4 Phase Response. Pulse Response

There are three types of ***phase response*** that a filter can have:

zero phase, **linear phase**, and **nonlinear phase**.

The ***zero phase*** filter is characterized by an impulse response that is **symmetrical around sample zero**. It requires the use of negative indexes, which can be inconvenient to work with.

The ***linear phase*** means that the impulse response is **symmetrical between the left and right**; however, the location of symmetry has been shifted from zero. This shift results in the phase being a *straight line*. The slope of this straight line is directly proportional to the amount of the shift.

An impulse response that is ***not*** **symmetrical between the left and right** leads to a phase, that is *not* a straight line, i.e. it has a ***nonlinear phase***.

Why does anyone care if the **phase is linear or not**?

- These are the **pulse responses** of each of the three filters.

The pulse response is a positive going step response followed by a negative going step response:

- It shows what happens to both the rising and falling edges in a signal.
- Zero and linear phase filters have left and right edges that look the ***same***, while nonlinear phase filters have left and right edges that look ***different***.

The **pulse response** of a **recursive filter is *not* symmetrical** between the left and right, and therefore has a *nonlinear* phase.

Example (fig. 5)

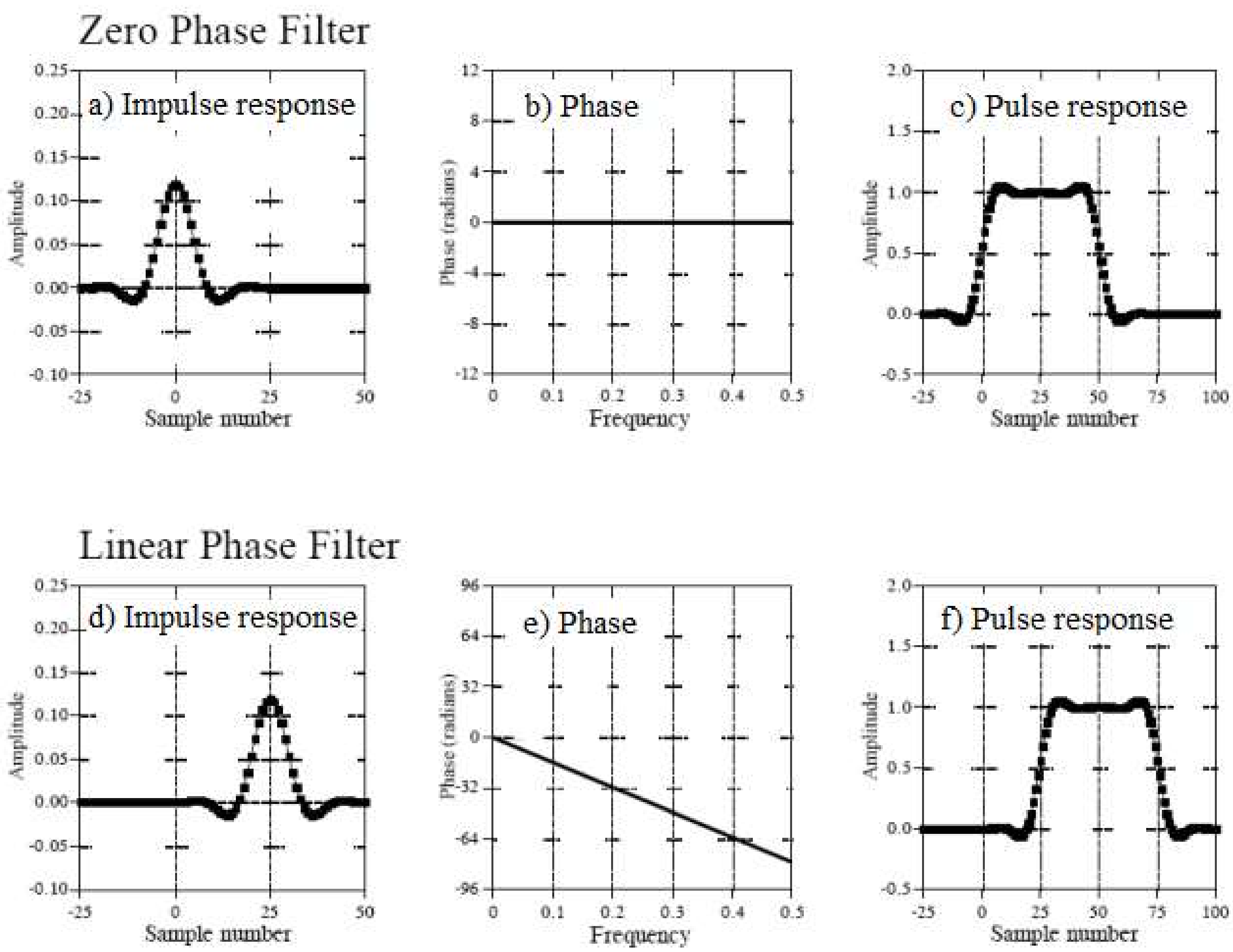


Fig. 5 Zero and linear phase filters.

(a), (b), (c) A *zero phase* filter - its step responses are symmetrical between the top and bottom, making a symmetric pulse response.

(d), (e), (f) A *linear phase* filter has also a symmetric pulse response.

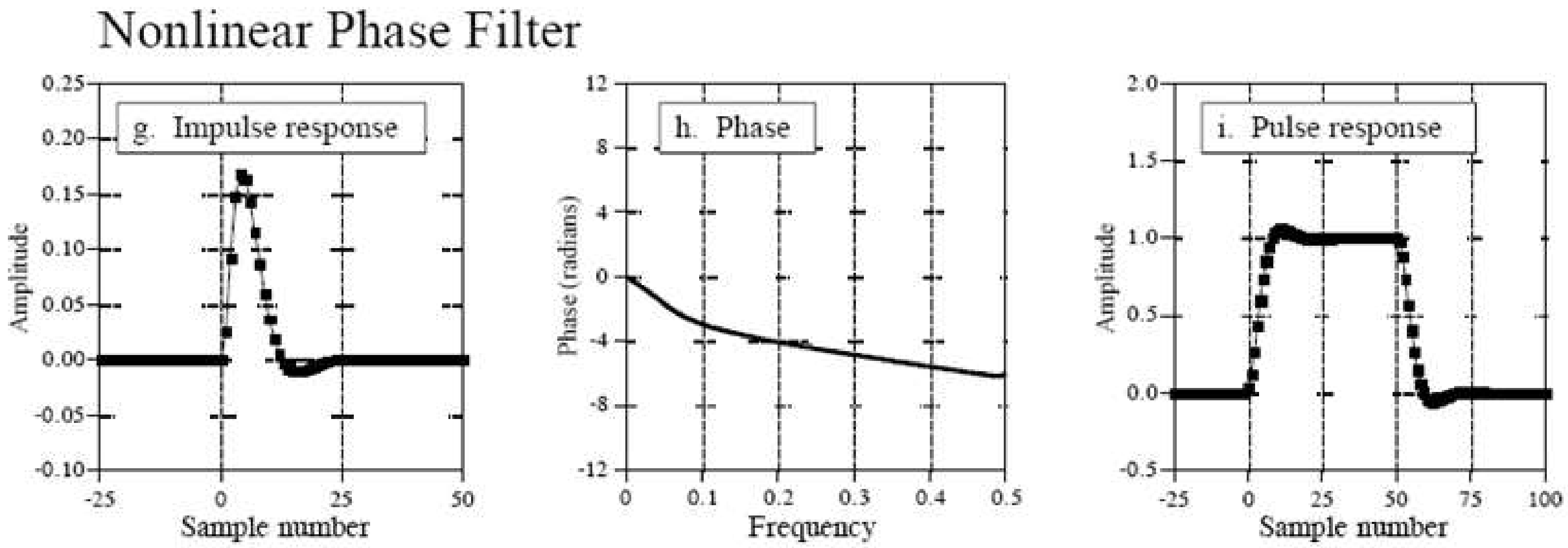


Fig. 5 (cont.) Nonlinear phase filter. (g) The impulse responses of *nonlinear phase* filters are not symmetrical, and the left and right edges of the pulse response are not the same (i).

- **Analog electronic circuits** have the same problem with the phase response. The **Bessel** filter is designed to have as linear phase as possible; however, it is far below the performance of digital filters.
- The ability to provide an *exact* linear phase is a clear advantage of **digital filters**.

1.5 Modifying recursive filters to obtain a *zero phase*

The ***reverse* recursion** equation

The signal is filtered from left-to-right, instead of right-to-left.

$$y_n = a_0 \cdot x_n + a_1 \cdot x_{n+1} + a_2 \cdot x_{n+2} + \cdots + a_p \cdot x_{n+p} +$$
$$+ b_1 \cdot y_{n+1} + b_2 \cdot y_{n+2} + \cdots + b_q \cdot y_{n+q}$$

Filtering in the reverse direction does not produce any benefit in itself; the filtered signal still has left and right edges that do not look alike.

But when forward and reverse filtering are *combined* :

- this produces a *zero phase* recursive filter.

Any recursive filter can be converted to zero phase with this **bidirectional filtering** technique.

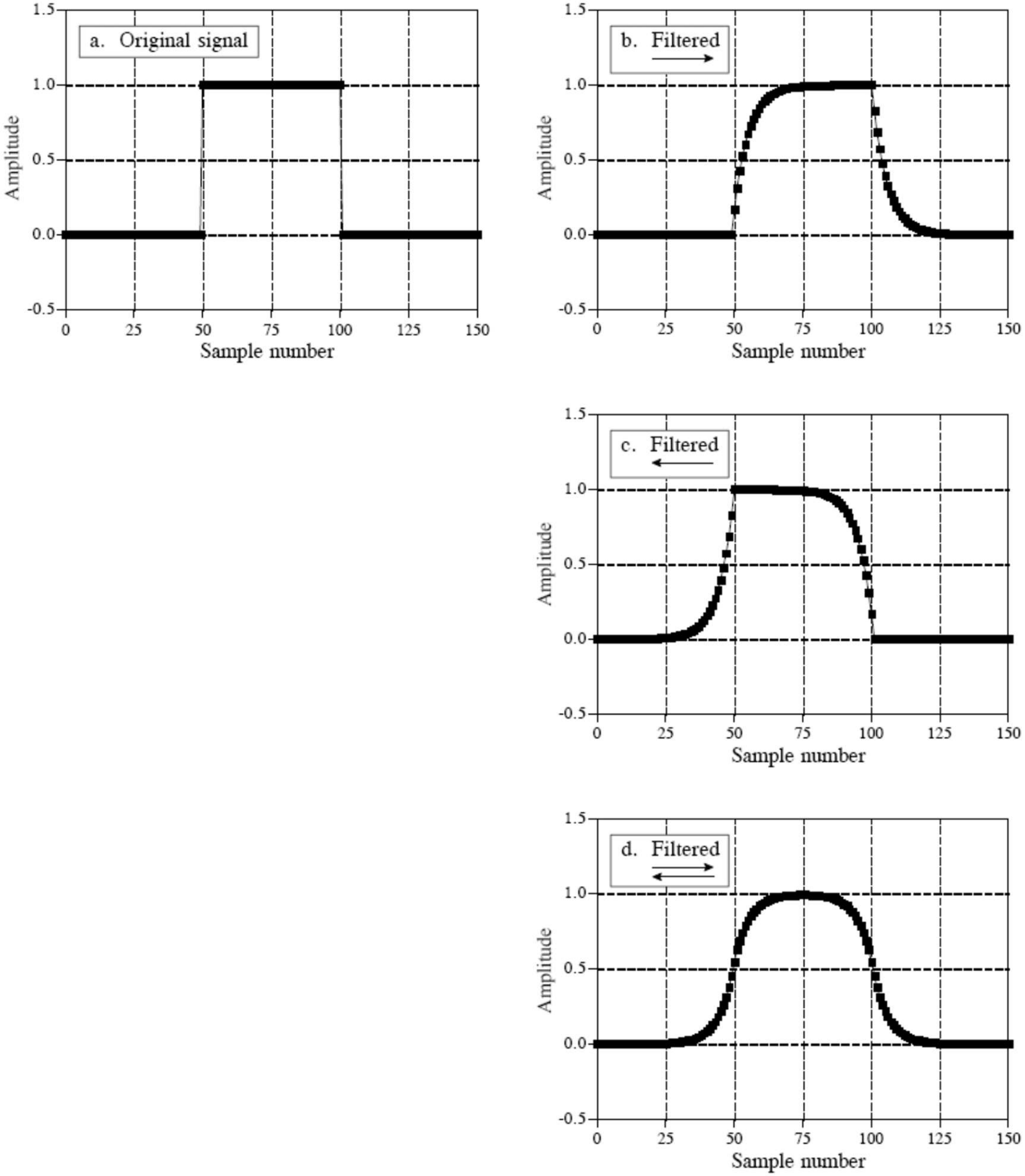


Fig. 6 Bidirectional recursive filtering applied to a rectangular pulse signal (a).
(b) is the signal after being filtered with a single pole recursive low-pass filter, passing from *left-to-right*.
(c) the signal has been processed in the same manner, but with the filter moving *right-to-left*.
(d) is the signal after being filtered both *left-to-right* and then *right-to-left*.

The *impulse* and *frequency responses* of the **bidirectional** filter

In the time domain, the impulse response of the bidirectional filter corresponds to:

- convolving the original impulse response with a left-for-right flipped version of itself.

The magnitude of the frequency response is the same for each direction, while the phases are opposite in sign. When the two directions are combined,

- the magnitude becomes *squared*, while
- the phase cancels to *zero*.

Example.

The impulse response of a single pole low-pass filter is a one-sided exponential.

The impulse response of the corresponding bidirectional filter is a one-sided exponential that decays to the right, convolved with a one-sided exponential that decays to the left.

→ This turns out to be a double-sided exponential that decays both to the left and right, with the same decay constant as the original filter.

2. Chebyshev Filters

- **Chebyshev** filters are used to **separate** one band of **frequencies** from another.
- Although they cannot match the performance of the windowed-sinc filter, the primary attribute of Chebyshev filters is their **speed,** typically more than an order of magnitude faster than the windowed-sinc.
- The design of these filters is based on the *z-transform*.

2.1 The Chebyshev and Butterworth Responses

The **Chebyshev response** is a strategy (based on *Chebyshev polynomials*) for achieving a **faster** *rolloff* by allowing a ***ripple*** in the frequency response. Analog and digital filters that use this approach are called ***Chebyshev filters*.** When the ripple is set to 0%, it is called a *maximally flat* or ***Butterworth* filter**.

Example 9.3 (figure 7)

It compares frequency responses of low-pass Chebyshev filters with pass-band ripples of: 0%, 0.5% and 20%. Consider using a ripple of 0.5% - the roll-off is much faster than the Butterworth.

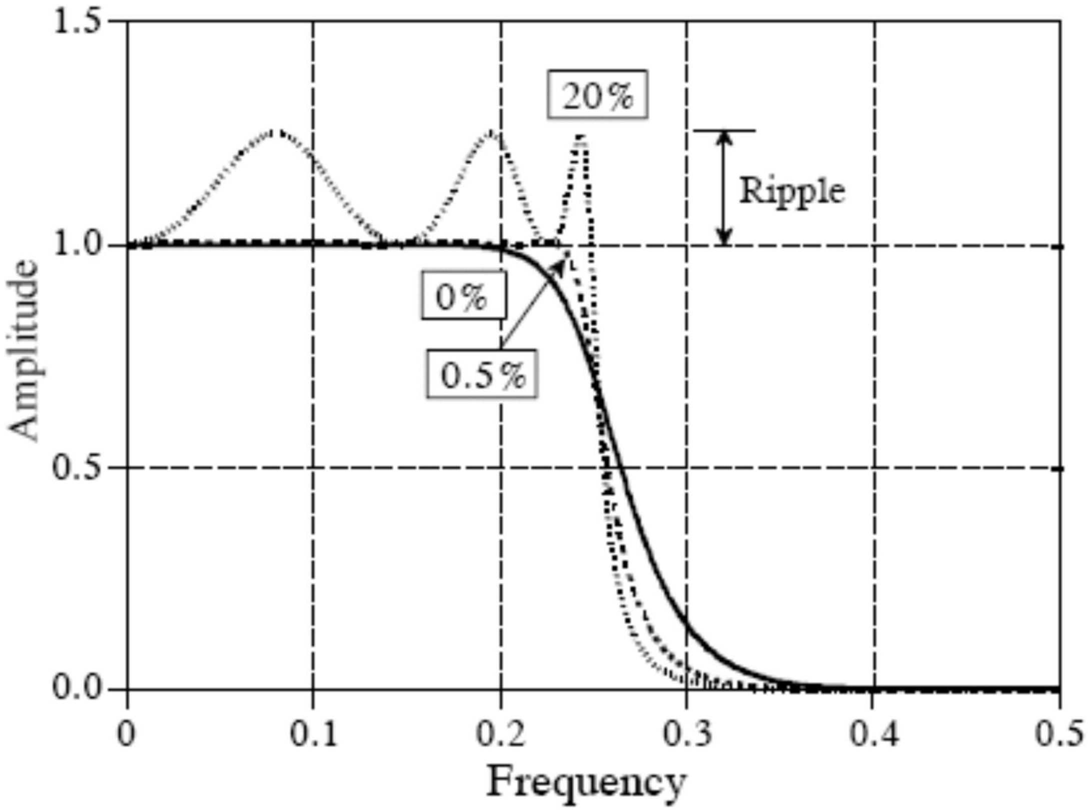


Fig. 7 The Chebyshev response. Chebyshev filters achieve a faster roll-off by allowing ripple in the passband.

Conclusion:

The Chebyshev response is an **optimal tradeoff** between ripple amplitude and roll-off speed.

As the ripple increases (bad), the roll-off becomes sharper (good):

- When the ripple is set to 0%, the filter is called a **maximally flat** or **Butterworth filter**.
- A ripple of 0.5% is usually a good choice for digital filters. This matches the typical precision and accuracy of the analog electronics.

Types of Chebyshev filter

Type 1 Chebyshev filter: the ripple is only allowed in the *passband*.
Type 2 Chebyshev filter have ripple only in the *stopband* (seldom used).
The elliptic filter has ripple in *both* the passband and the stopband

Elliptic filters provide the fastest roll-off for a given number of poles.

Step response overshoot

Chebyshev filters have an overshoot of 5 to 30% in their step responses: The overshoot in the step response results from the Chebyshev filter being optimized for the *frequency domain* at the expense of the *time domain*.

2.2 Designing the Chebyshev filter

Four parameters influence the design of a Chebyshev filter:
(1) a high-pass or low-pass response,
(2) the cutoff frequency,
(3) the **percent ripple** in the pass-band, and
(4) the number of **poles**.

What is **a *pole***?

The Laplace transform and z-transform allow to transform an impulse response by using sinusoids and decaying exponentials and to represent the system's characteristics as one complex polynomial divided by another complex polynomial.

The **roots** of the **numerator** are called ***zeros***, while the **roots** of the **denominator** are called ***poles*.**

Recursive filters are designed by first selecting the location of the poles and zeros, and then finding the appropriate recursion coefficients.

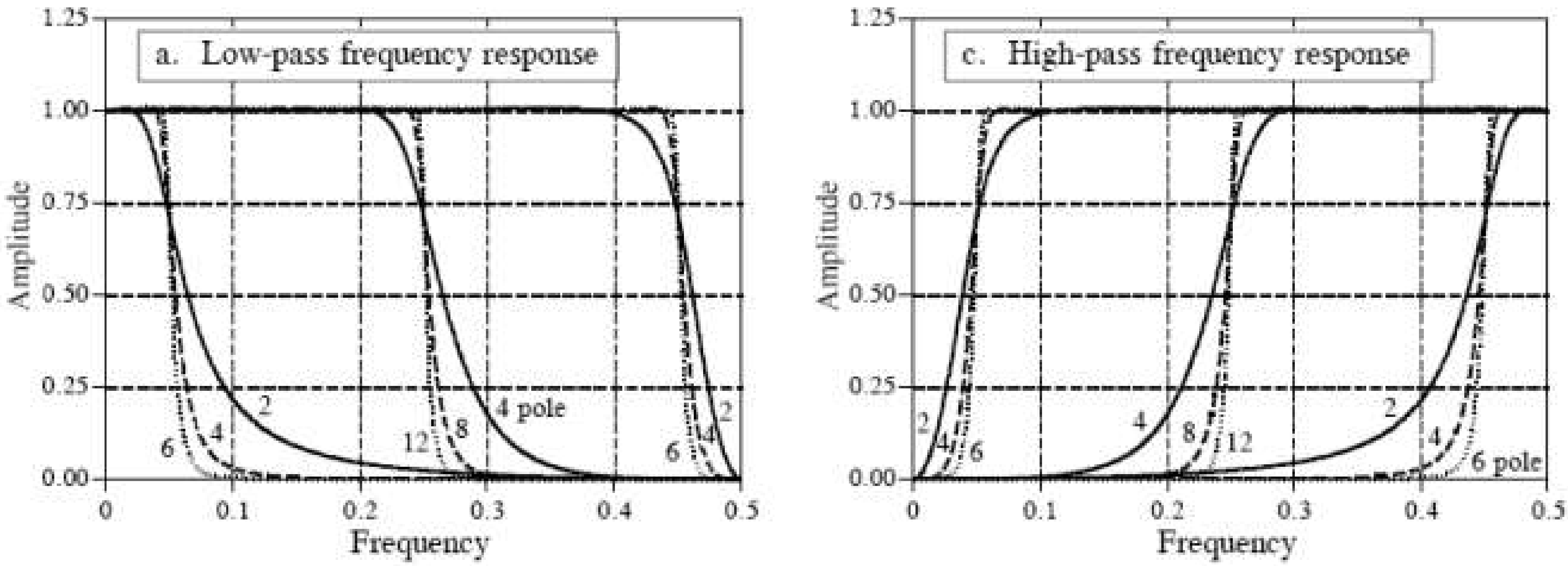
Poles have magic power - the **more** poles in a filter, the **better** the filter works.

Example 9.4 (fig. 8)

Compare the frequency response of several Chebyshev filters with 0.5% ripple. For the method used here the number of poles must be ***even***. The cutoff frequency is measured where the amplitude crosses 0.707 (-3dB).

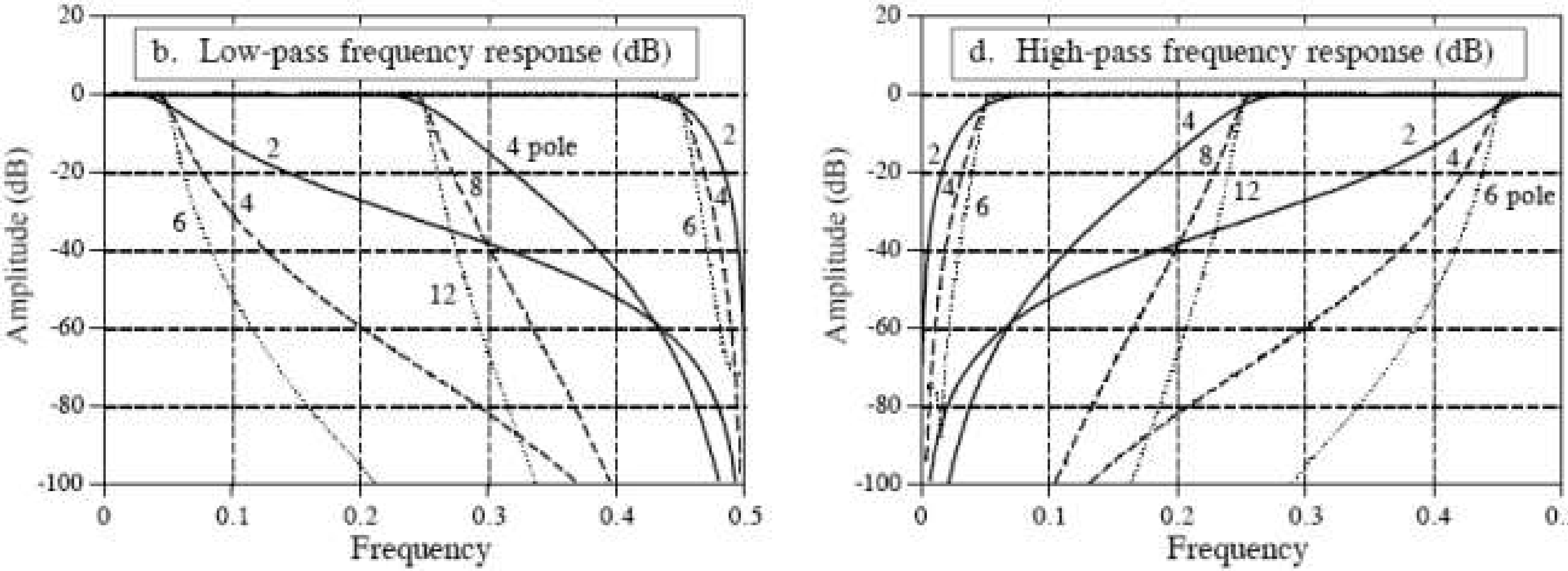
Conclusion:

- Filters with a cutoff frequency near 0 or 0.5 have a sharper roll-off than filters in the center of the frequency range. For example, a two pole filter at 0.05 f_C has about the same roll-off as a four pole filter at 0.25 f_C
- **Fewer poles** can be used near 0 and 0.5, thus better avoiding round-off noise.



(Linear amplitude scale)

Fig. 8 Chebyshev frequency responses: (a) and (b) show the frequency responses of low-pass Chebyshev filters with 0.5% ripple, while (c) and (d) show the corresponding high-pass filter responses.



(Decibel scale)

Fig. 8 (cont.) Chebyshev frequency responses. (a) and (b) show the frequency responses of low-pass Chebyshev filters with 0.5% ripple, while (c) and (d) show the corresponding high-pass filter responses.

2.3 Step Response Overshoot

Chebyshev filters have an overshoot of 5 to 30% in their step responses:

1. It is becoming larger as the number of poles is increased.
2. The amount of overshoot depends slightly on the cutoff frequency.

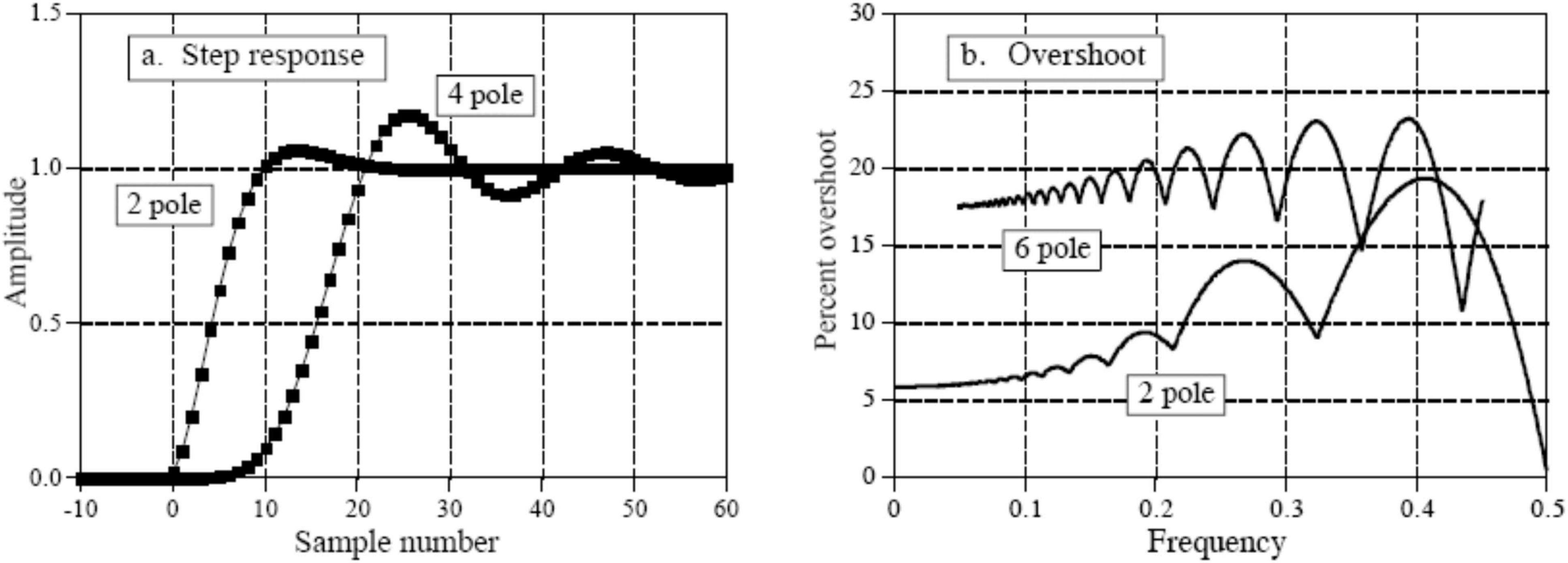


Fig. 9 The overshoot in the Chebyshev filter's step response (5% - 30%): (a) depends on the number of poles (for cutoff frequency of 0.05), and (b) on the cutoff frequency.

2.4 Stability

- The main limitation of digital filters carried out by **convolution** is *execution time*, but it is possible to achieve nearly any filter response.

- Recursive filters are just the opposite: they run very fast, however, they are limited in performance.

Example 9.5

Consider a 6 pole, 0.5% ripple, low-pass filter with a 0.01 cutoff frequency. The recursion coefficients for this filter are:

a_0= 1.391351 E-10	
a_1= 8.348109 E-10	b_1= 5.883343 E+00
a_2= 2.087027 E-09	b_2= -1.442798 E+01
a_3= 2.782703 E-09	b_3= 1.887786 E+01
a_4= 2.087027 E-09	b_4= -1.389914 E+01
a_5= 8.348109 E-10	b_5= 5.459909 E+00
a_6= 1.391351 E-10	b_6= -8.939932 E-01

- The **b** coefficients have an absolute value of about *ten*.
- Using single precision, the relative round-off noise is about 10^{-6}.
- The **a** coefficients have values of about 10^{-9}.
  The contribution from the input signal (via the "a" coefficients) will be 1000 times smaller than the *noise* from the previously calculated output signal (via the "b" coefficients).
- This filter **will not** work!

Conclusion:

Round-off noise limits the number of poles that can be used in a filter.

The maximum number of poles for single precision:

Cutoff frequency	0.02	0.05	0.10	0.25	0.40	0.45	0.48
Maximum poles	4	6	10	20	10	6	4

Extending the maximum number of poles:

1. Using double precision.

2. To implement the filter in *stages*.
   For example, a six pole filter is a cascade of three stages of two poles each.