7A. Digital filters

1. Introduction to digital filters
2. Time-domain parameters
3. Frequency-domain parameters
4. Filter classification

[Smith, ch. 14]

1. Introduction to Digital Filters

1.1 Filter basics

Two **goals** of using digital filters:

(1) **separation** *of signals* that have been combined, and

(2) **restoration** *of signals* that have been distorted in some way.

A **linear filter** has:

1. an **impulse response** : the output of a system when the input is an *impulse*.
2. a ***step response*** : output when the input is a *step* (also called an *edge*, and an ***edge*** *response*).
3. a ***frequency response*** : can be found by taking the DFT (or FFT) of the impulse response

Each of these responses contains **complete information about the filter**.
If **one of the three** is specified, the **other two** can be directly calculated.

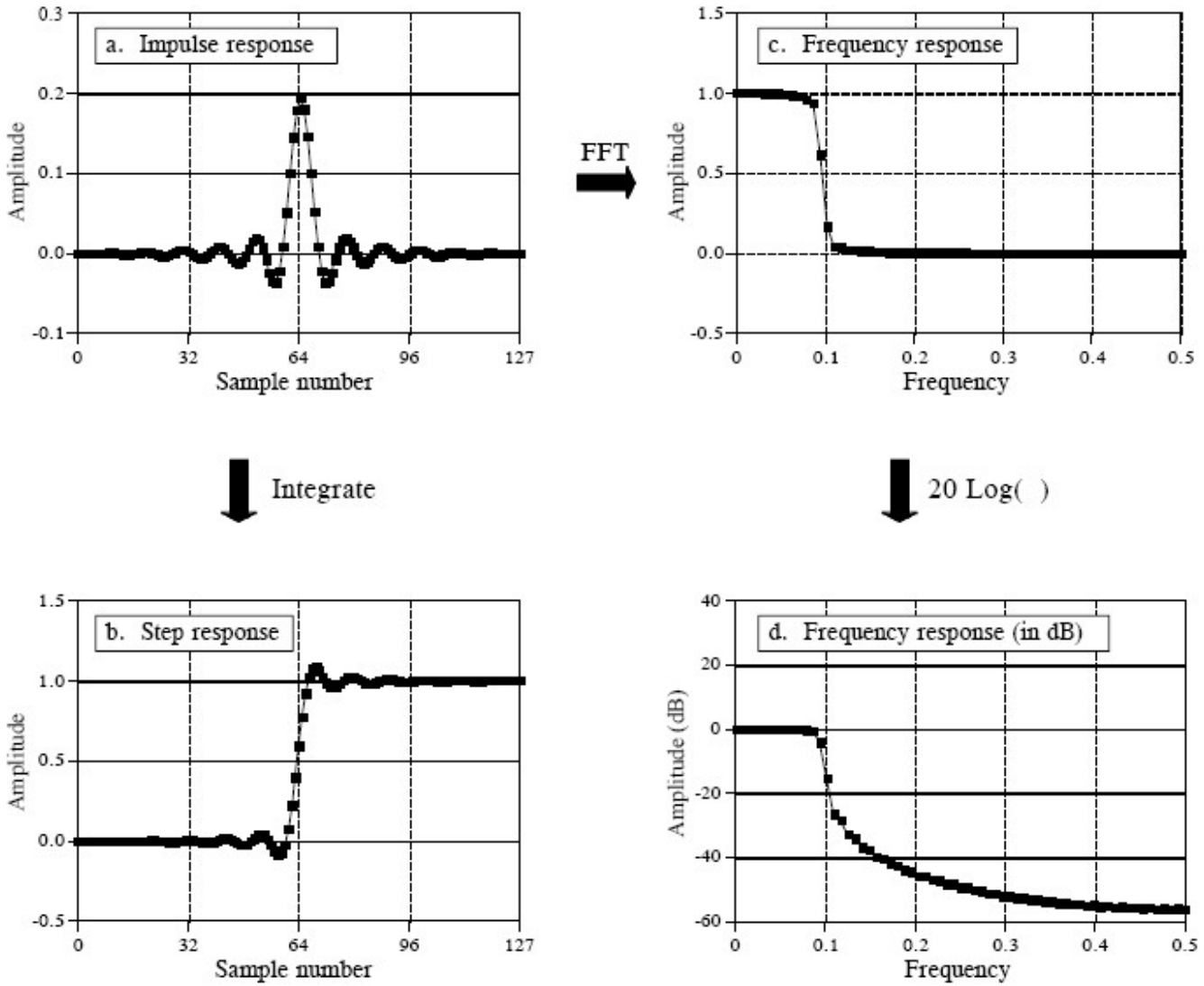

Fig.1.
Filter parameters.
(a) **impulse response**, (b) the **step response** can be found by discrete integration of the impulse response, (c) the **frequency response** can be found from the impulse response by using the FFT - displayed on a **linear scale**, (c), or (d) in **decibels**.

Step response

Two ways to find the step response:
(1) feed **a step** waveform into the filter and see what comes out, or
(2) **integrate** the impulse response (*integration* is used with continuous signals, while *discrete integration*, i.e. a **running sum**, is used with discrete signals).

Frequency response

The frequency response can be plotted on a linear vertical axis or on a logarithmic scale (**decibels**):

- the linear scale is best to show the **passband ripple** and **roll-off**, while
- the decibel scale is needed to show the **stopband attenuation**.

Decibel

- A **bel** means that the **power** is changed by a ***factor** of **ten***.
- A **decibel (dB)** is one-tenth of a bel; i.e. every *ten* decibels mean that the power has changed by a ***factor** of ten*.

Example. 3 bels <--> an output signal with 10×10×10 = 1000 times the power of the input.

Example. The decibel values of: -20dB, -10dB, 0dB, 10dB, 20dB, mean the power ratios: 0.01, 0.1, 1, 10 and 100, respectively.

We usually work with a signal's ***amplitude***, not its *power*.

- Amplitude is proportional to the **square-root of power**.

Example. 20 dB means a factor of 100 in power, but only a factor of 10 in amplitude.
Every *twenty* decibels mean that the amplitude has changed by a factor of ten.

$$dB \leftrightarrow 10\log_{10}\frac{P_2}{P_1}, \quad dB \leftrightarrow 20\log_{10}\frac{A_2}{A_1}$$

The above equations use the **base 10 logarithm**. Using the base *e* logarithm (the **natural log**, written $\log_e x$ *or* $\ln x$) we have:

$$dB = 4.342945 \log_e (P_2 / P_1) \quad , \text{ and}$$

$$dB = 8.685890 \log_e (A_2 / A_1).$$

Remarks

1. **-3 dB** means that the amplitude is reduced from 1 to 0.707 (and the power is reduced from 1 to 0.5).
2. There are following conversions between decibels and ***amplitude*** ratios:

60 dB = 1000 × ;
40dB = 100× ;
20dB = 10× ;
0dB = 1×
-20dB = 0.1×;
-40dB = 0.01×;
-60dB = 0.001×

1.2 Implementation of a digital filter

Filter kernel

The most straightforward way to implement a digital filter is by *convolving* the **input signal** with the **digital filter's *impulse response*** (when the *impulse response* is used in this way, it is given the name **filter kernel**).

Filters carried out by convolution are called **Finite Impulse Response** or **FIR** filters.

Recursive filter

Another way to make digital filters is called **recursion**.

- Recursive filters are using previously calculated values from the ***output***, besides points from the ***input***.
- Recursive filters are defined by a set of **recursion coefficients**.

Impulse response of recursive filters

The impulse responses of recursive filters are composed of **sinusoids** that **exponentially decay** in amplitude.

- This makes their impulse responses ***infinitely long***.
- After the amplitude drops below the round-off noise of the system, the remaining samples can be ignored.

Recursive filters are also called **Infinite Impulse Response** or **IIR** filters.

1.3 How Information is Represented in Signals

In naturally occurring signals we can have :

- **information** represented in **the time domain**, and
- **information** represented in **the frequency domain**.

Information represented in the time domain describes **when** something occurs and **what** the amplitude of the occurrence is.

Information represented in the frequency domain is more indirect. Many things in our universe show **periodic motion**. By measuring the frequency, phase, and amplitude of this periodic motion, information can often be obtained about the system producing the motion.

The information is contained in the ***relationship*** **between many points** in the signal.

The importance of the **step** and **frequency responses**:

1. The **step response** describes how information represented in the *time domain* is being **modified** by the system.
2. The **frequency response** shows how information represented in the *frequency domain* is being **changed**.

In filter design it is **not possible** to optimize a filter for **both applications** - good performance in the time domain results in poor performance in the frequency domain, and vice versa.

Example

- Designing a filter to remove noise from an EKG signal (information represented in the time domain), the step response is the important parameter, and the frequency response is of little concern.
- Designing a digital filter for a hearing aid (with the information in the frequency domain), the frequency response is all important, while the step response doesn't matter.

2. Time Domain Parameters

The **step function** represents a division between two **different** regions:

- it can mark when an **event starts**, or when an **event ends**.
- whatever is on the *left* is somehow different from whatever is on the *right*.

The **step response** is important because:

- it describes how the dividing lines **are being modified** by the filter.

Step response parameters:

1. **Risetime**: the duration of the step response must be shorter than the spacing of the events, i.e. the step response should be as *fast* as possible. The most common way to specify the **risetime** is to count the number of samples between the 10% and 90% amplitude levels.
2. **Overshoot** in the step response: overshoot must generally be eliminated because it changes the amplitude of samples in the signal; this is a basic distortion of the information contained in the time domain.
3. **Linear phase**: it is often desired that the upper half of the step response be symmetrical with the lower half - this symmetry is needed to make the *rising edges* look the same as the *falling edges*. This symmetry is called **linear phase**, because its frequency response has a phase that is a straight line.

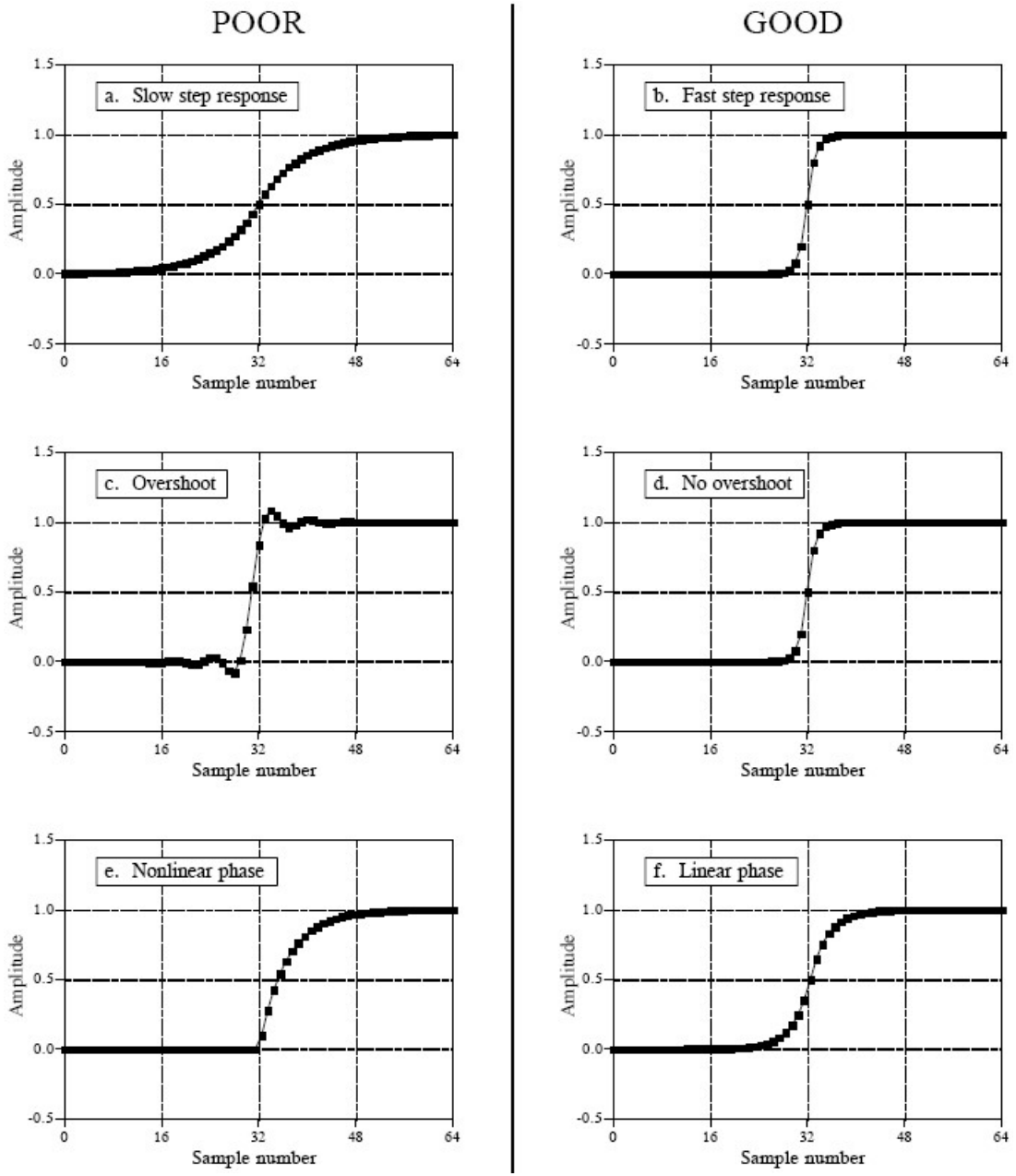

Fig. 2.
The **step response** is used to evaluate **time domain performance** of a filter.
Parameters:
(1) transition speed (risetime), in (a) and (b),
(2) overshoot, in (c) and (d), and
(3) phase linearity (symmetry between the top and bottom halves of the step), in (e) and (f).

3. Frequency Domain Parameters

1) Four basic frequency responses:

1. **low-pass**,
2. **high-pass**,
3. **band-pass**, and
4. **band-reject**.

The purpose of these filters is to allow some frequencies to **pass unaltered**, while completely **blocking** other frequencies:

Parameters:

1. The **passband** - those frequencies that are passed → **passband ripple**
2. The **stopband** - those frequencies that are blocked → **stopband attenuation**
3. The **transition band** is between *passband* and *stopband* → a **fast roll-off** means that the transition band is very narrow.
4. The border between the *passband* and *transition band* is the **cutoff frequency**.

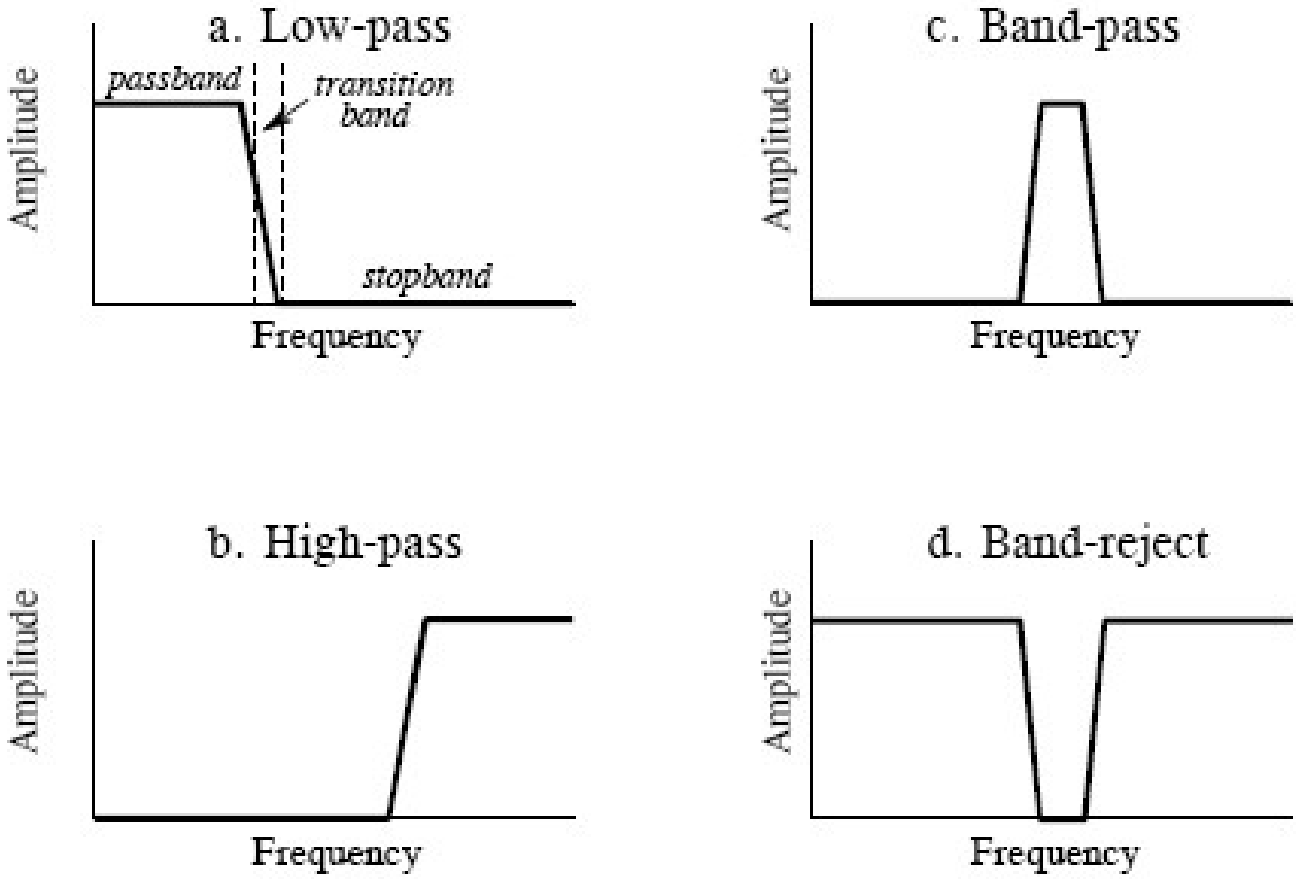

Fig. 3 The four **common frequency responses**.

Cutoff frequency

- In analog filters the cutoff frequency is usually defined to be where the amplitude is reduced to 0.707 (i.e., -3dB).
- Digital filters are less standardized, and it is common to see 99%, 90%, 70.7%, and 50% amplitude levels defined to be the cutoff frequency.

Performance evaluation

Three parameters measure the filter's performance in the frequency domain:

1. To separate closely spaced frequencies, the filter must have a **fast roll-off**.
2. For the passband frequencies to move through the filter unaltered, there must be no **passband ripple**.
3. To adequately block the stopband frequencies, it is necessary to have good **stopband attenuation**.

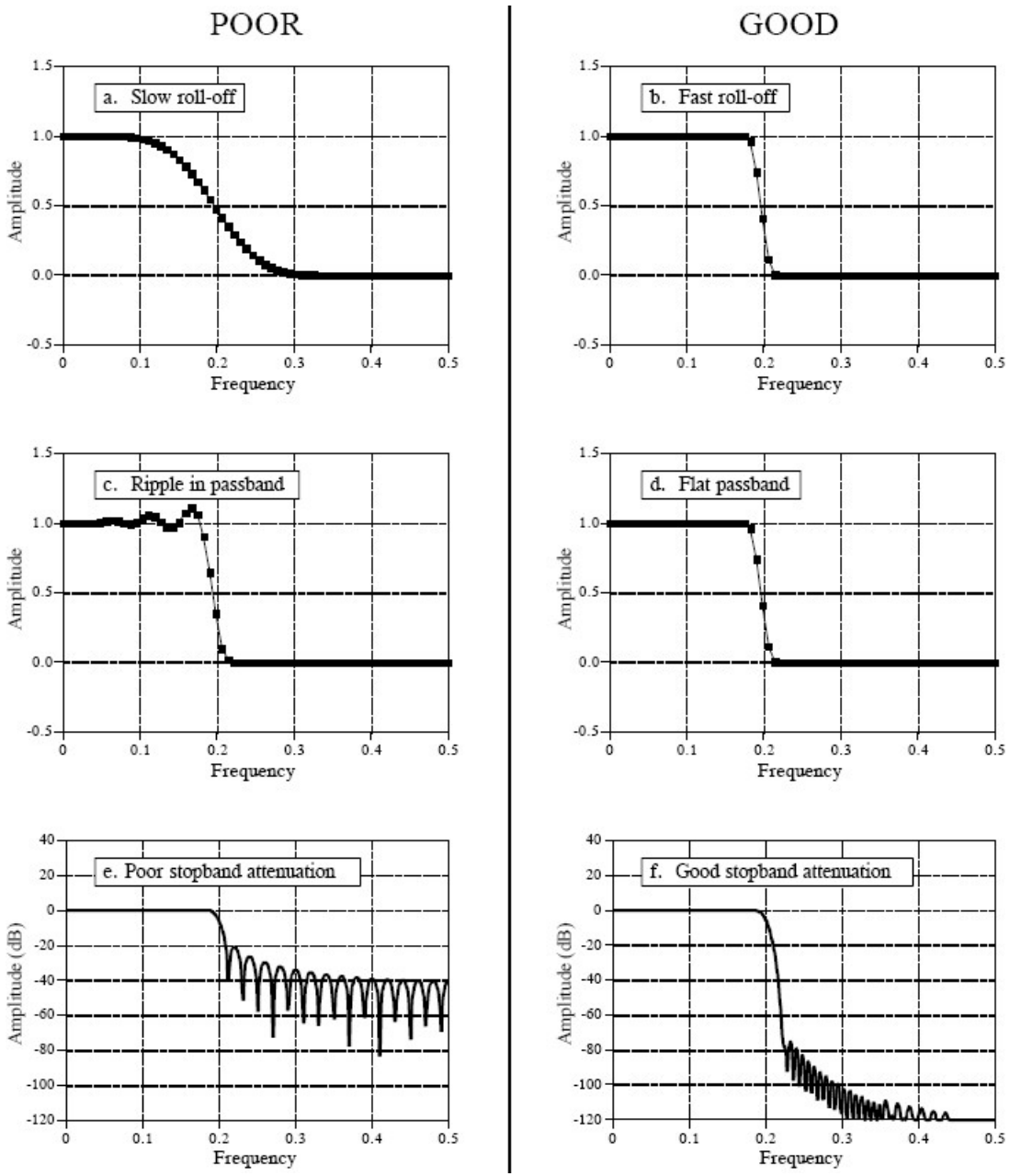

Fig. 4 Parameters for evaluating *frequency domain* performance. The frequency responses shown are for low-pass filters.

Three parameters are important:
(1) **roll-off** sharpness, (a) and (b),
(2) **passband ripple**, (c) and (d), and
(3) **stopband attenuation**, (e) and (f).

Why nothing about the *phase* ?

1. The phase isn't important in most frequency domain applications. For example, the phase of an audio signal is almost completely random.
2. Even if the phase is important, digital filters with a *perfect* phase response can be made, i.e., all frequencies pass with a zero phase shift.

The DFT converts a **system's impulse response** into its **frequency response:**

- The number of samples used to represent the impulse response can be arbitrarily **large**, as a ***padding with zeros*** does not change the impulse response.

- The frequency response of a filter is really a ***continuous* signal** between DC and one-half of the sampling rate. The output of the DFT is a ***sampling*** of this continuous line.

2) High-Pass, Band-Pass and Band-Reject Filters

- High-pass, band-pass and band-reject filters are designed by **starting with a low-pass filter,** and then converting it into the desired response.

  Hence, most discussions on filter design is on ***low-pass filters.***

There are two useful methods for the **low-pass to high-pass conversion**:

- **spectral inversion** and
- **spectral reversal**.

Spectral inversion (but it is performed in the time domain)

Change the **low-pass** filter kernel into a **high-pass** filter kernel:

1. First, **change the sign** of each sample in the filter kernel.
2. Second, **add *one*** to the sample at the **center of symmetry**.

Spectral inversion changes a filter from

- low-pass to high-pass,
- high-pass to low-pass,
- band-pass to band-reject, or
- band-reject to band-pass.

Restriction: the low-frequency components exiting the low-pass filter must have **the same** phase as the low-frequency components exiting the all-pass system. Hence:
(1) the original filter kernel must have **left-right symmetry** (i.e., a zero or linear phase), and
(2) the impulse must be added at the center of symmetry.

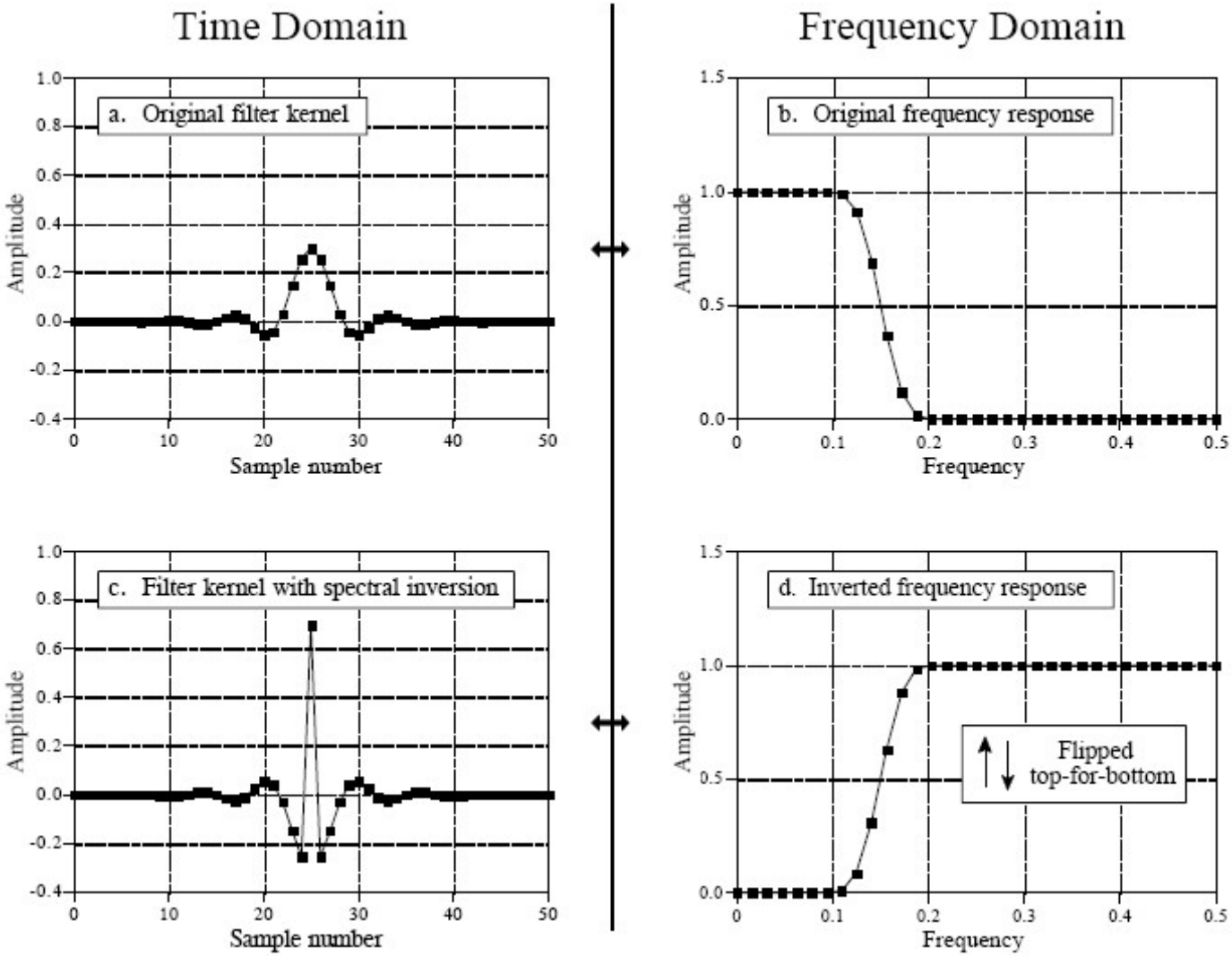

Fig. 5 Example of **spectral inversion**.
(a) The **low-pass filter** kernel has (b) the frequency response.
(c) A **high-pass filter** kernel is formed by changing the sign of each sample in (a), and adding one to the sample at the center of symmetry.
(d) This action in the time domain ***inverts*** the frequency spectrum (i.e., flips it top-for-bottom).

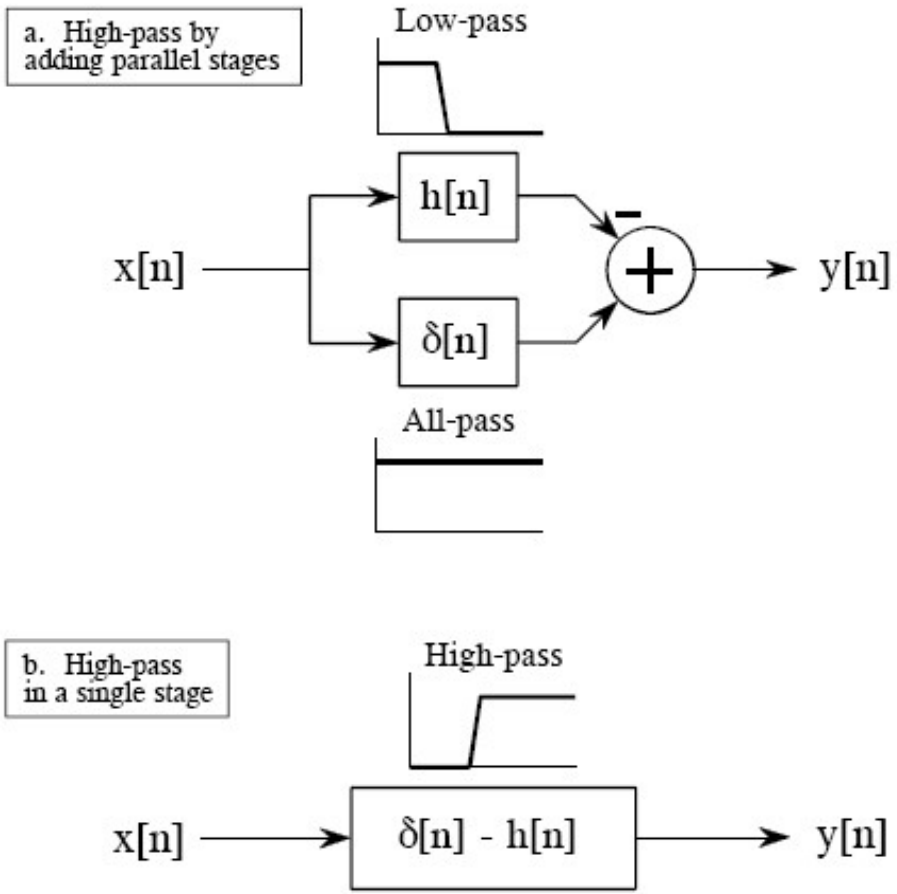

Fig. 6 Block diagram of **spectral inversion**. In (a), the input signal, $x[n]$, is applied to two systems in parallel, having impulse responses of $h[n]$ and $\delta[n]$. In (b), the combined system has an impulse response of $\delta[n]$ - $h[n]$. The frequency response of the combined system is the ***inversion*** of the frequency response of $h[n]$.

Spectral reversal (but it is performed in the time domain)

Spectral reversal - a second method for **low-pass to high-pass** conversion.

- The high-pass filter kernel is formed by *changing the sign of* **every other *sample***.

This flips the frequency domain *left-for-right*: 0 becomes 0.5 and 0.5 becomes 0.

- Changing the sign of every other sample is equivalent to multiplying the filter kernel by a **sinusoid** with a frequency of 0.5.
  This has the effect of **shifting** the frequency domain by 0.5.

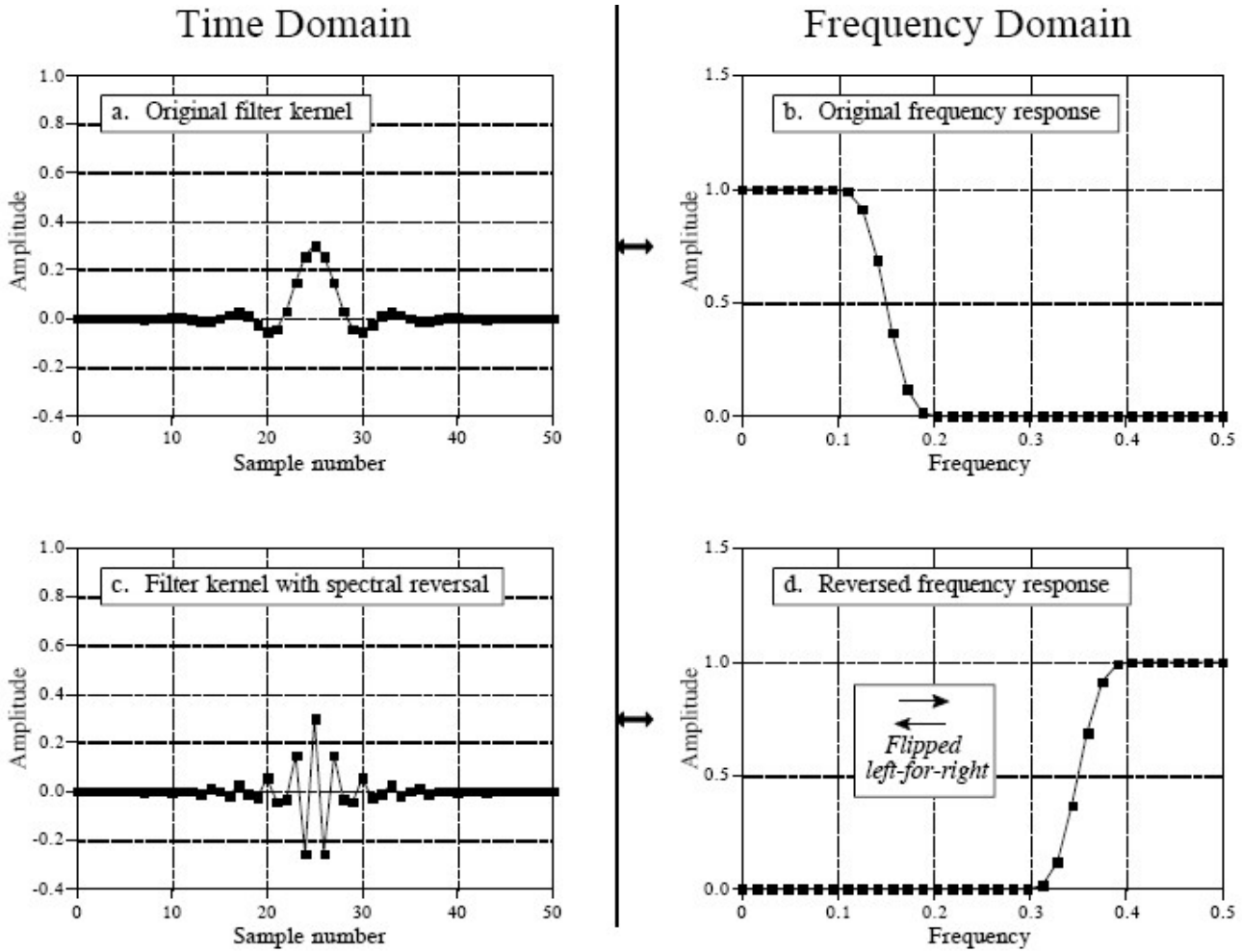

Fig. 7 Example of **spectral reversal**. (a) The low-pass filter kernel and its (b) the frequency response. (c) A high-pass filter kernel is formed by changing the sign of every other sample in (a). This results in (d) the frequency domain being flipped *left-for-right*, resulting in the high-pass frequency response.

Band-pass and band-reject filters

Low-pass and high-pass filter kernels can be **combined** to form **band-pass** and **band-reject** filters:

- ***adding*** the filter kernels produces a ***band-reject*** filter,
- ***convolving*** the filter kernels produces a ***band-pass*** filter.

These are based on the way how **parallel** (for addition) or **cascaded** (for convolution) **systems** are combined.

Multiple combination of these techniques can also be used. For example:

- a band-pass filter can also be designed by adding the two filter kernels and then using *spectral inversion* or *spectral reversal*.

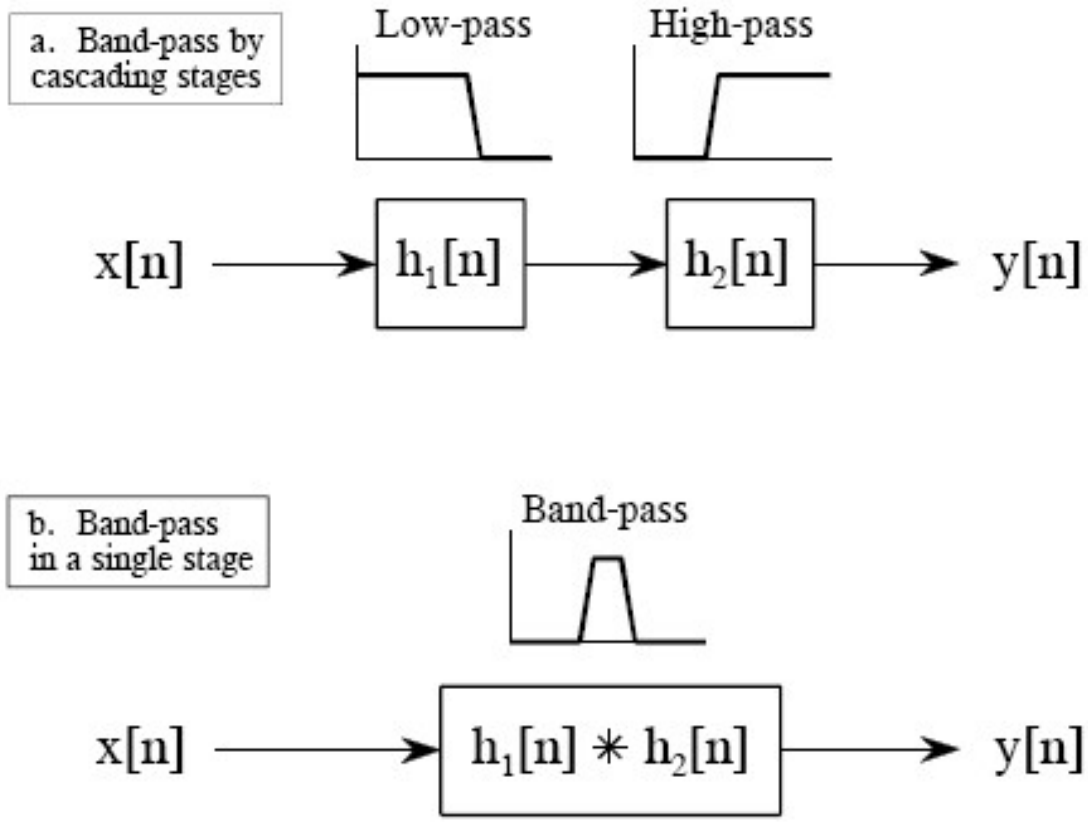

Fig. 8 Designing a **band-pass filter**: (a) a band-pass filter can be formed by cascading a low-pass filter and a high-pass filter; (b) this can be reduced to a single stage - the filter kernel of the single stage is equal to the *convolution* of the low-pass and high-pass filter kernels.

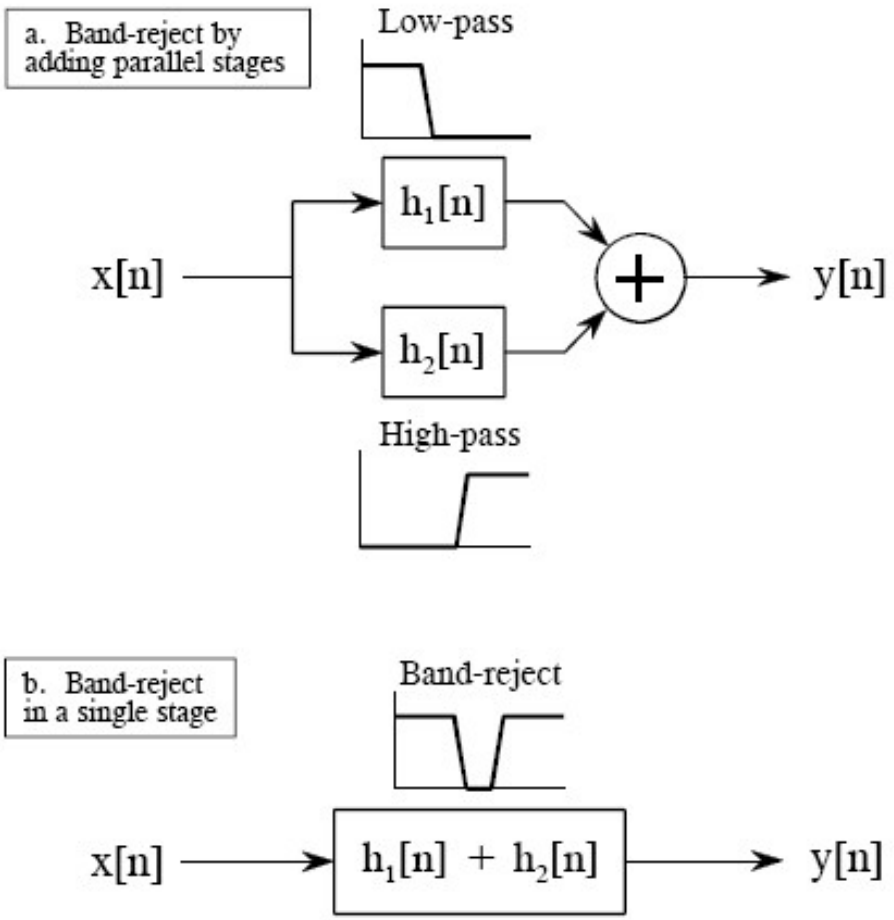

Fig. 9 Designing a **band-reject filter**: (a) a band-reject filter is formed by the parallel combination of a low-pass filter and a high-pass filter with their outputs added; (b) this is reduced to a single stage, with the filter kernel found by *adding* the low-pass and high-pass filter kernels.

4. Filter classification

Digital filters are classified

- by their ***use*** and
- by their ***implementation***.

The **use** of a digital filter can be broken into three categories:

- *time domain*, *frequency domain* and *custom*.

Digital filters can be **implemented** in two ways,

- by *convolution* (called *finite impulse response* or *FIR*) and
- by *recursion* (called *infinite impulse response* or *IIR*).

		FILTER IMPLEMENTATION	
		Convolution FIR	Recursion IIR
FILTER USED FOR:	Time domain (smoothing, DC removal)	Moving average	Single pole
	Frequency domain (separating frequencies)	Windowed-sinc	Chebyshev
	Custom (e.g. deconvolution)	FIR custom	Iterative

Table 1 Filter classification. Filters can be divided by their *use*, and how they are *implemented*.

Time domain filters are used when the information is encoded in the shape of the signal's waveform. Time domain filtering is used for such actions as:

- smoothing, DC removal, waveform shaping, etc.

Frequency domain filters are used when the information is contained in the amplitude, frequency, and phase of the component sinusoids. The goal of these filters is to separate one band of frequencies from another.

Custom filters are used when a special action is required by the filter, something more specific than the four basic responses (high-pass, low-pass, band-pass and band-reject). For instance, custom filters can be used for *deconvolution*.

Filters carried out by convolution can have far better performance than filters using recursion, but execute much more slowly.

Filters carried out by convolution:

- the *moving average* is used in the time domain,
- the *windowed-sinc* is used in the frequency domain, and
- *FIR custom* is used when something special is needed.

Recursive filters:

- the *single pole* recursive filter is used in the time domain,
- the *Chebyshev* filter is used in the frequency domain,
- a custom response filter is designed by *iterative techniques*.