



INTRODUCTION TO DIGITAL FILTERS

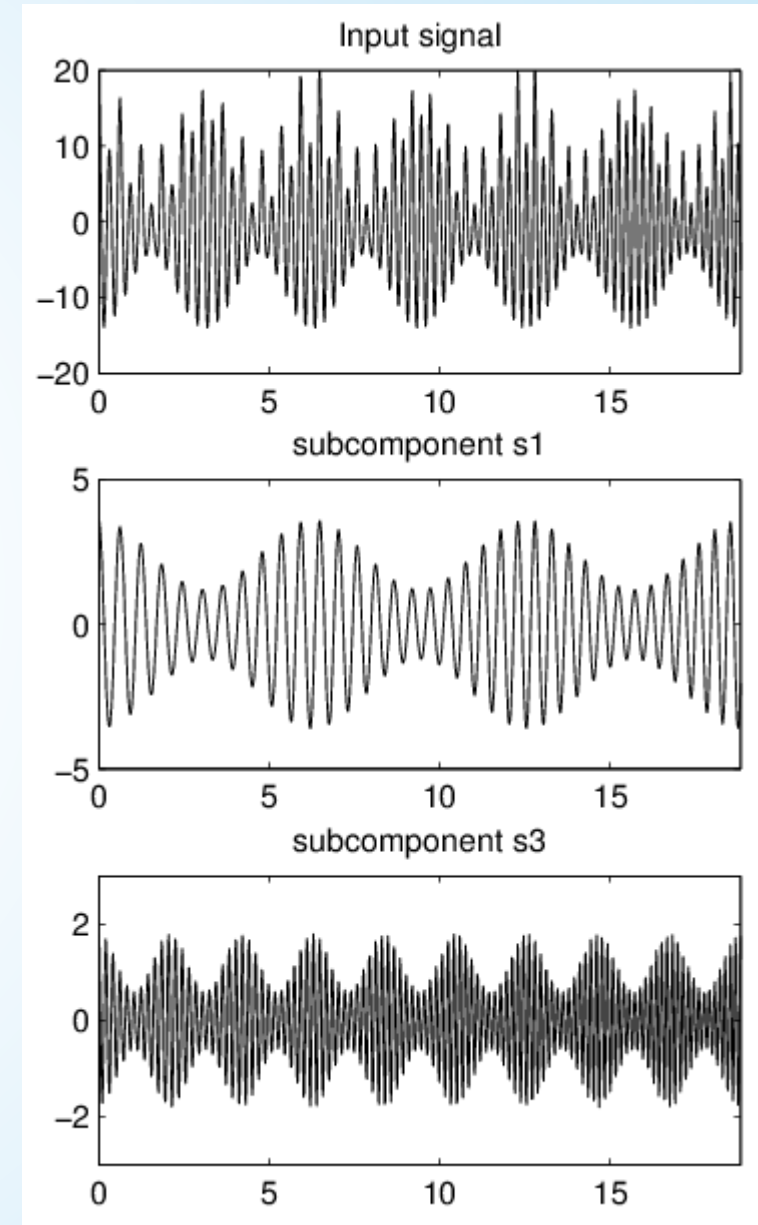
Lecture 1

Associate professor Naila Allakhverdiyeva

Filter Basics

Digital filters are a very important part of DSP. Filters have two uses: **signal separation** and **signal restoration**.

- **Signal separation** is needed when a signal has been contaminated with interference, noise, or other signals. For example, imagine a device for measuring the electrical activity of a baby's heart (EKG) while still in the womb. The raw signal will likely be corrupted by the breathing and heartbeat of the mother. A filter might be used to separate these signals so that they can be individually analyzed.



Filter Basics

- **Signal restoration** is used when a signal has been distorted in some way. For example, an audio recording made with poor equipment may be filtered to better represent the sound as it actually occurred. Another example is the deblurring of an image acquired with an improperly focused lens, or a shaky camera.



Analog or digital filters?

- Analog filters are cheap, fast, and have a large dynamic range in both amplitude and frequency.
- Digital filters, in comparison have **high level of performance** that can be achieved.
- Digital filters can achieve **thousands of times** better performance than analog filters.
- Analog filters have limitations of the electronics, such as the accuracy and stability of the resistors and capacitors. In comparison, digital filters are so good that the performance of the filter is frequently ignored. They have limitations of the *signals*, and the *theoretical* issues regarding their processing.

Time or Space Domain?

- Filter's input and output signals are in the **time domain**. This is because signals are usually created by sampling at regular intervals of **time**. But this is not the only way sampling can take place.
- The second most common way of sampling is at equal intervals in **space**.
- For example, imagine taking simultaneous readings from an array of strain sensors mounted at one centimeter increments along the length of an aircraft wing.
- Many other domains are possible; however, time and space are by far the most common.
- When you see the term *time domain* in DSP, remember that it may actually refer to samples taken over time, or it may be a general reference to any domain that the samples are taken in.

Impulse response. Step response.

Frequency response


- Every linear filter has an **impulse response**, a **step response** and a **frequency response**.
- Each of these responses contains complete information about the filter, but in a different form.
- If one of the three is specified, the other two are fixed and can be directly calculated.
- All three of these representations are important, because they describe how the filter will react under different circumstances.

Impulse response. Step response.

7

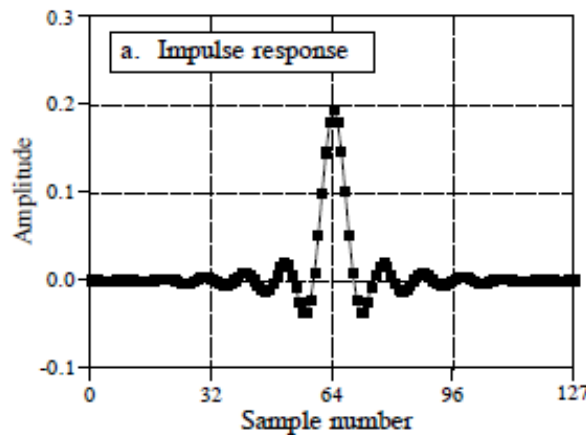
Frequency response

- The most simple way to implement a digital filter is by **convolving** the **input signal** with the digital filter's **impulse response**. All possible linear filters can be made in this manner.
- When the *impulse response* is used in this way, filter designers give it a special name: the **filter kernel**.

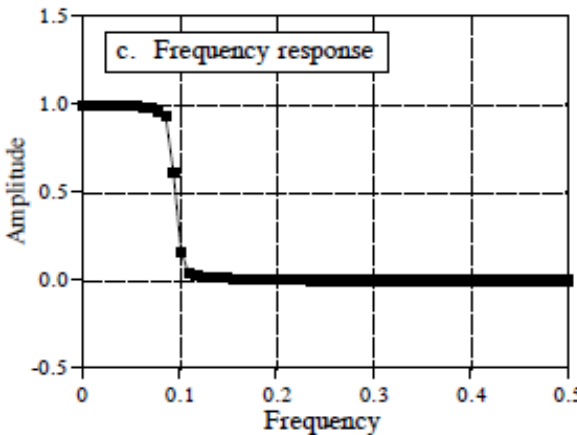
$$y[n] = \sum_{k=0}^{N-1} x[k]h[n - k]$$


Impulse response. Step response. Frequency response

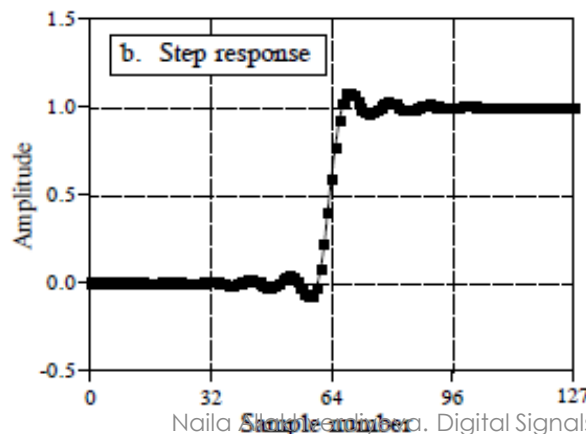
8



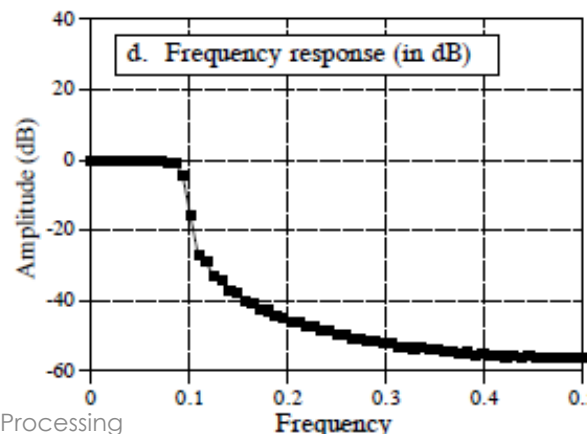
FFT
→



Integrate
↓



20 Log()
↓



Every linear filter has an impulse response, a step response, and a frequency response.

The **step response**, (b), can be found by discrete integration of the **impulse response**, (a).

The **frequency response** can be found from the impulse response by using the Fast Fourier Transform (FFT), and can be displayed either on a linear scale, (c), or in decibels, (d).

Impulse response. Step response.

9

Frequency response

Ways to make digital filters: **convolution** and **recursion**.

- When a filter is implemented **by convolution**, each sample in the output is calculated by *weighting* the samples in the input, and adding them together.
- **Recursive filters** uses previously calculated values from the *output*, besides points from the *input*. Instead of using a filter kernel, recursive filters are defined by a set of **recursion coefficients**.
- All linear filters have an impulse response, even if you don't use it to implement the filter.
- To find the impulse response of a recursive filter, simply feed in an impulse, and see what comes out. The impulse responses of recursive filters are composed of sinusoids that exponentially decay in amplitude. In principle, this makes their impulse responses ***infinitely long***.
- Recursive filters are also called **Infinite Impulse Response** or **IIR** filters.
- Filters carried out by convolution are called **Finite Impulse Response** or **FIR** filters.

Impulse response. Step response.

10

Frequency response

- The **impulse response** is the output of a system when the input is an **impulse**.
 $\delta[n] \rightarrow h[n]$
- The **step response** is the output when the input is a **step** (also called an edge, and an edge response). $u[n] \rightarrow s[n], u[n] = \begin{cases} 0, n < 0 \\ 1, n \geq 0 \end{cases}$ ←
- Since the step is the integral of the impulse, the step response is the integral (or running sum) of the impulse response. $u[n] = \sum_{k=0}^{\infty} \delta[n - k]$

Two ways to find the step response:

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

$$s[n] = \sum_{k=0}^{\infty} h[n - k]$$

- The **frequency response** can be found by taking the DFT of the impulse response.

Frequency Response

11

➡ The frequency response can be plotted on a **linear vertical axis**, or on a **logarithmic scale (decibels)**. The linear scale is best at showing the passband ripple and roll-off, while the decibel scale is needed to show the stopband attenuation.

A **bel** (in honor of Alexander Graham Bell) means that the power is changed by a **factor of ten**.

➡ For example, an electronic circuit that has 3 bels of amplification produces an output signal with $10 \times 10 \times 10 = 1000$ times the power of the input.

A **decibel (dB)** is one-tenth of a bel. Therefore, the decibel values of: -20dB, -10dB, 0dB, 10dB & 20dB, mean the power ratios: 0.01, 0.1, 1, 10, & 100, respectively.

➡ In other words, every **ten decibels** mean that the power has changed by a factor of ten.

Decibels

- Decibels are a way of expressing a *ratio* between two signals. Ratios of power (P_1 & P_2) use a different equation from ratios of amplitude (A_1 & A_2).

Conversions between decibels and *amplitude ratios*:

Desibels are ideal for describing the **gain of a system**, i.e., the ratio between the output and the input signal.

$$\text{dB} = 10 \log_{10} \frac{P_2}{P_1}$$

$$\text{dB} = 20 \log_{10} \frac{A_2}{A_1}$$

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

How Information is Represented in Signals

There are only two ways for information to be represented in naturally occurring signals:

- information represented in the **time domain**;
- 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.
- A single sample, in itself, contains **no information** about the periodic motion. The information is contained in the *relationship* between many points in the signal.

How Information is Represented in Signals

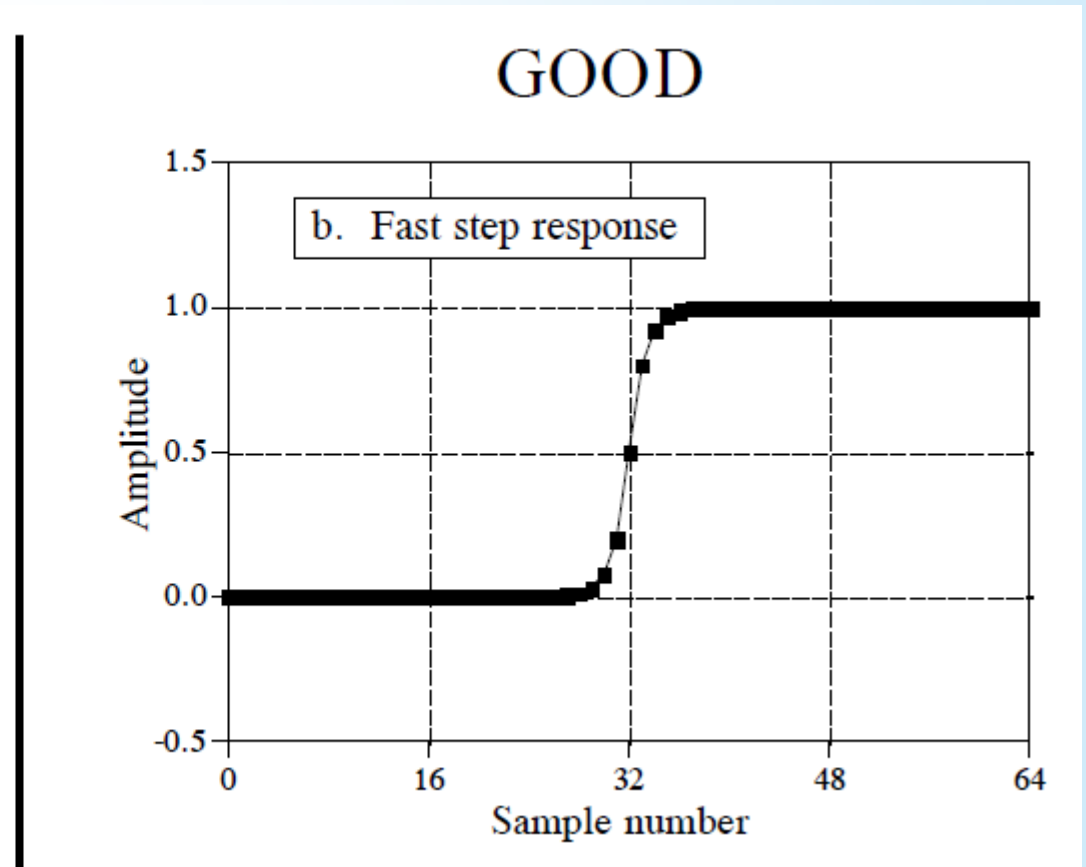
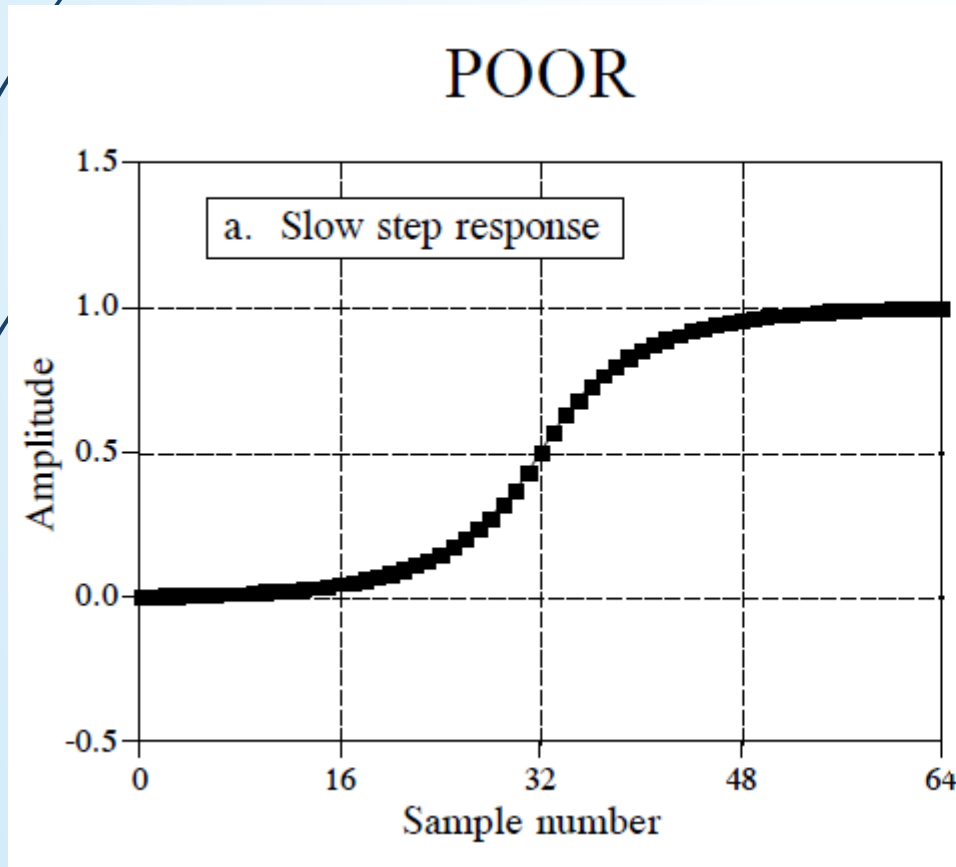
- The *step response* describes how information represented in the *time domain* is being modified by the system.
- The *frequency response* shows how information represented in the *frequency domain* is being changed.
- This distinction is absolutely critical in filter design because it is not possible to optimize a filter for both applications. Good performance in the time domain results in poor performance in the frequency domain, and vice versa.
- If you are 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. If your task is to design 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.

Time Domain Parameters

- The step function is the purest way of representing a division between two dissimilar regions. It can mark when an event starts, or when an event ends. It tells you that whatever is on the *left* is somehow different from whatever is on the *right*.
- This is how the human mind views time domain information: a group of step functions dividing the information into regions of similar characteristics. The step response, in turn, is important because it describes how the dividing lines are being modified by the filter.

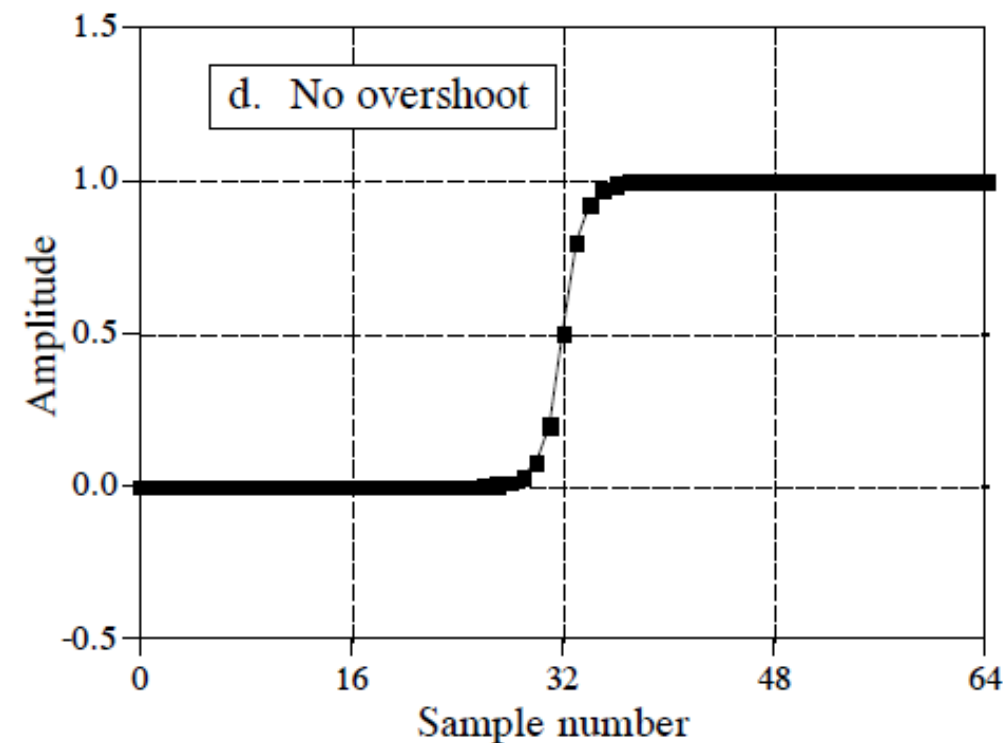
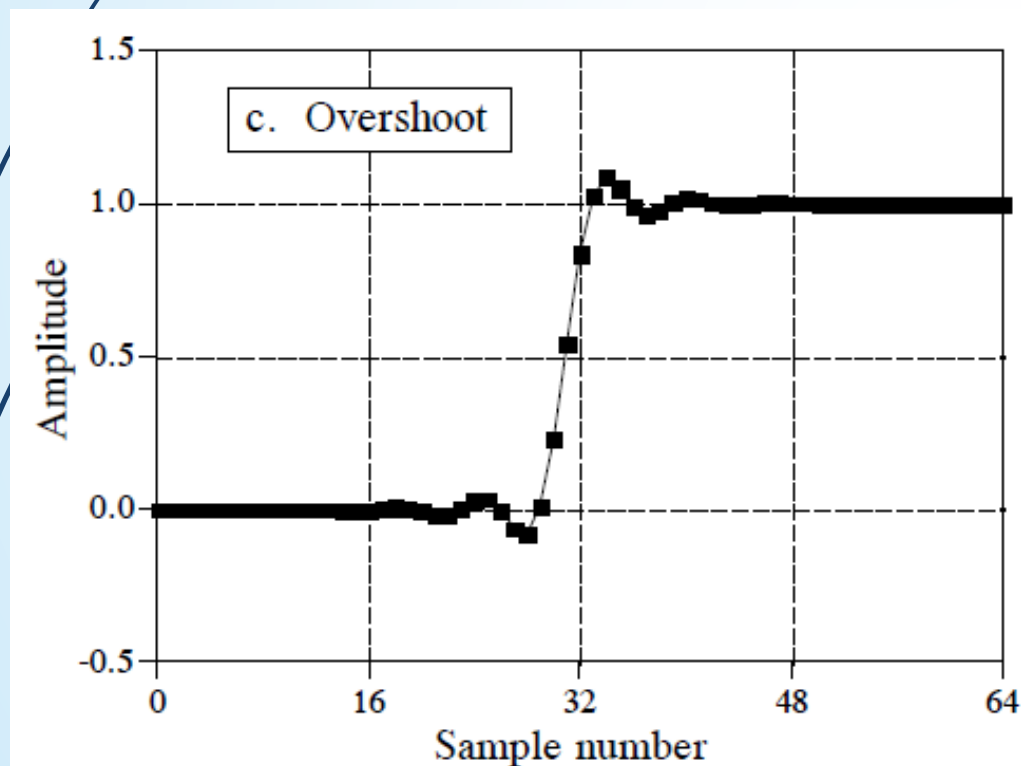
Step Response parameters

- ➔ **Risetime** - number of samples between the 10% and 90% amplitude levels. The step response should be as **fast** as possible.



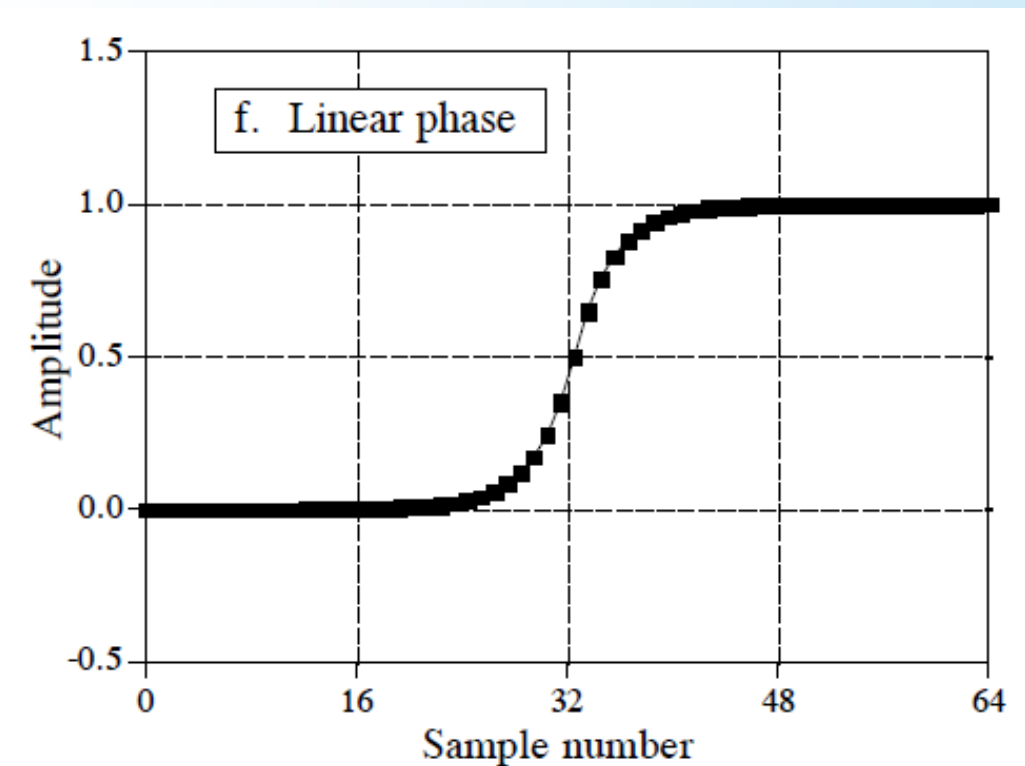
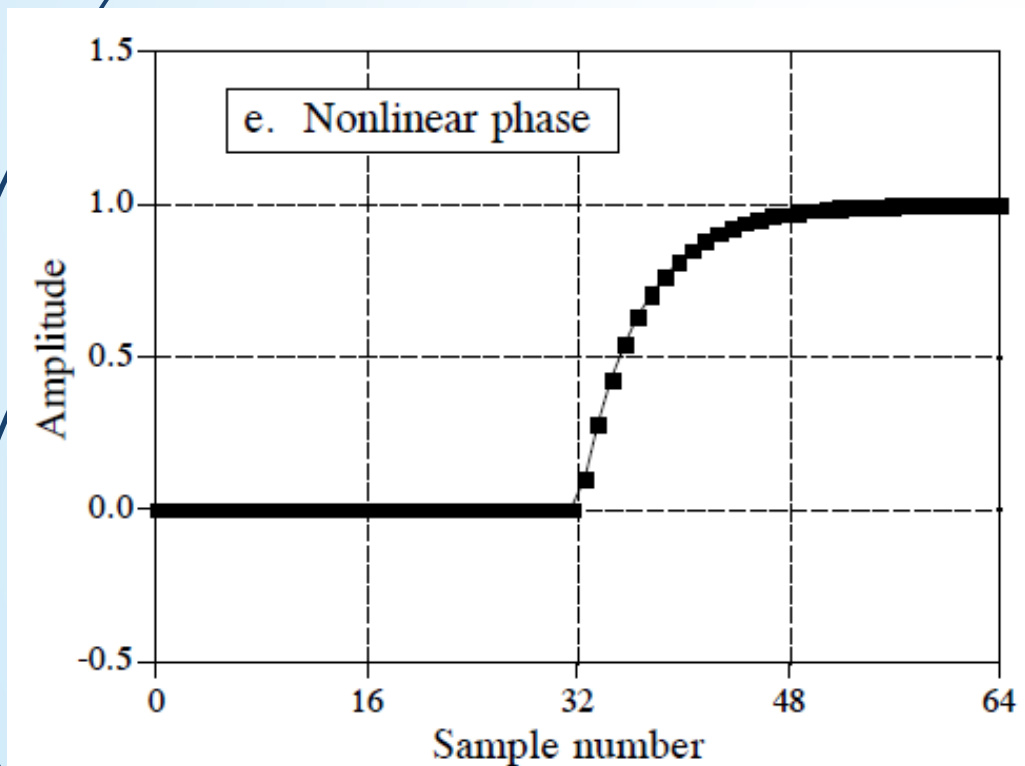
Step Response parameters

- **Overshoot** – it 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.



Step Response parameters

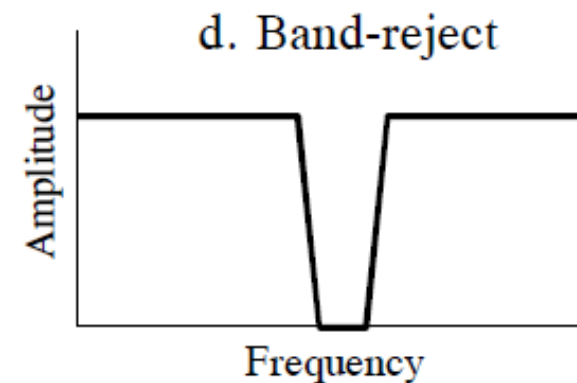
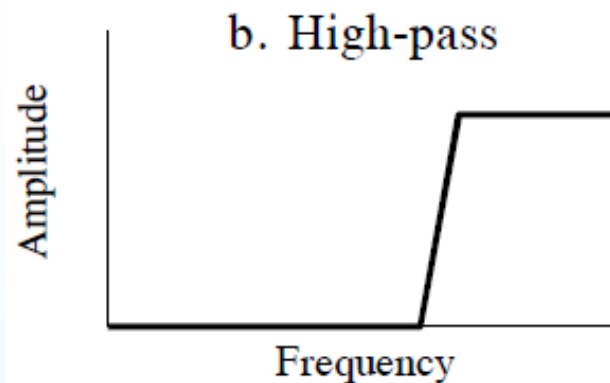
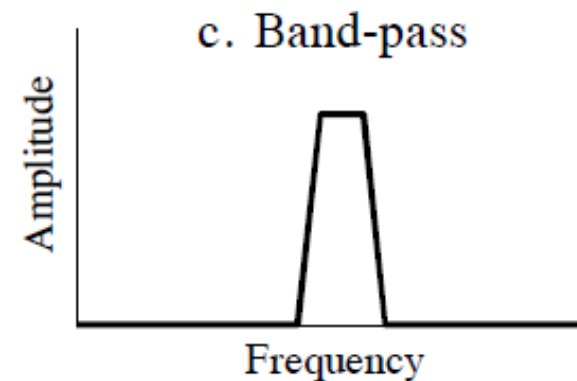
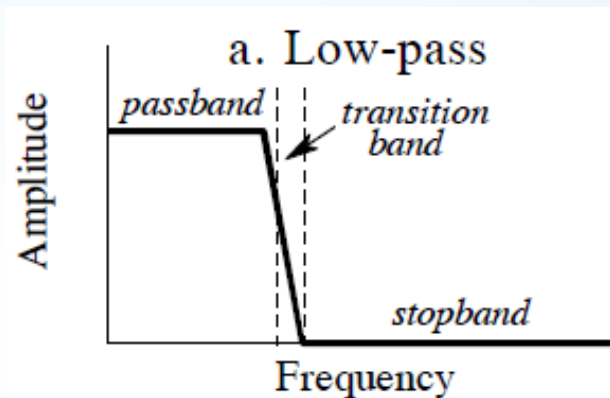
- **Linear phase** - upper half of the step response is symmetrical with the lower half. *Rising edges* look the same as the *falling edges*. Frequency response has a phase that is a **straight line**



Four Common Frequency responses

- ☐ Low-pass
- ☐ High-pass
- ☐ Band-pass
- ☐ Band-reject

Frequency domain filters are generally used to pass certain frequencies (the *passband*), while blocking others (the *stopband*).

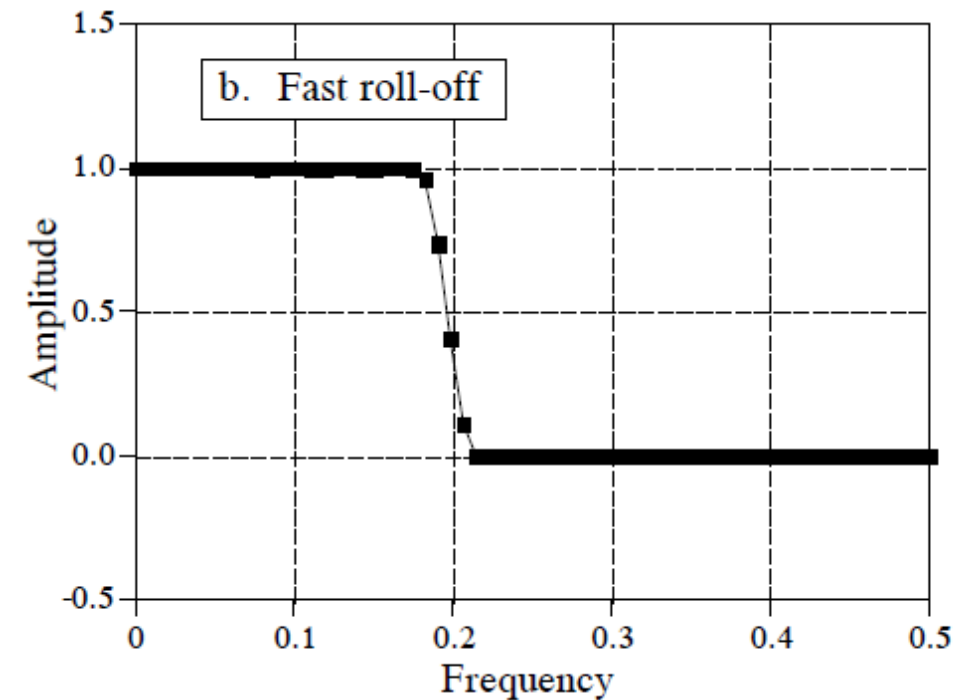
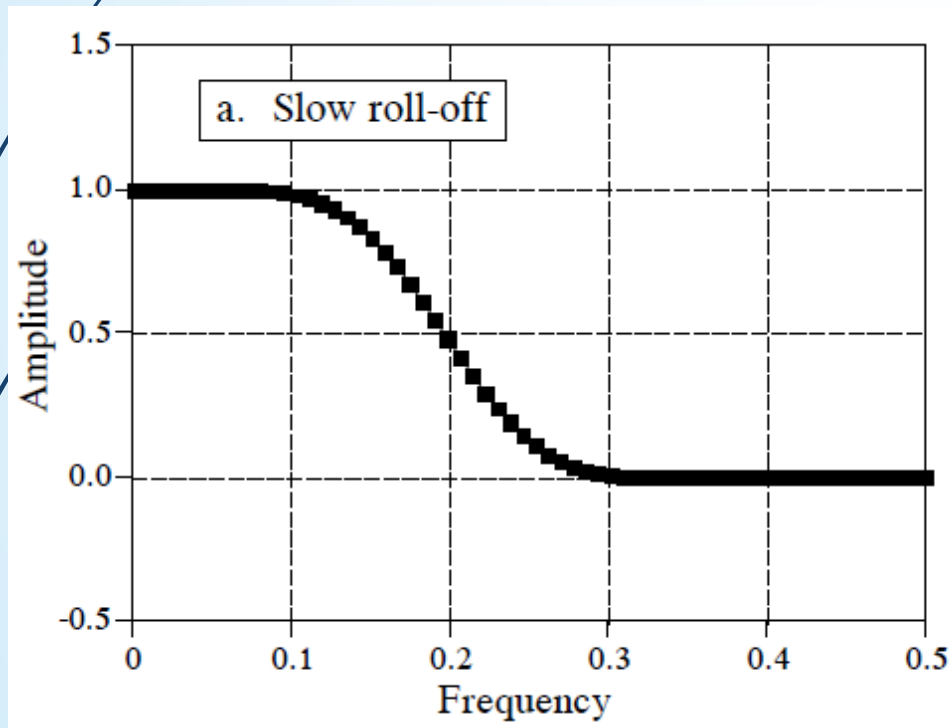


Filter Parameters

- **Passband** – frequencies that are passed
- **Stopband** – frequencies that are blocked
- **Transition band** – between passband and stopband. Fast roll-off – transition band is very narrow
- **Cutoff frequency** - division between the passband and transition band. For analog filter design - the amplitude is reduced to 0.707 (i.e., -3dB). For digital filters – no standards.

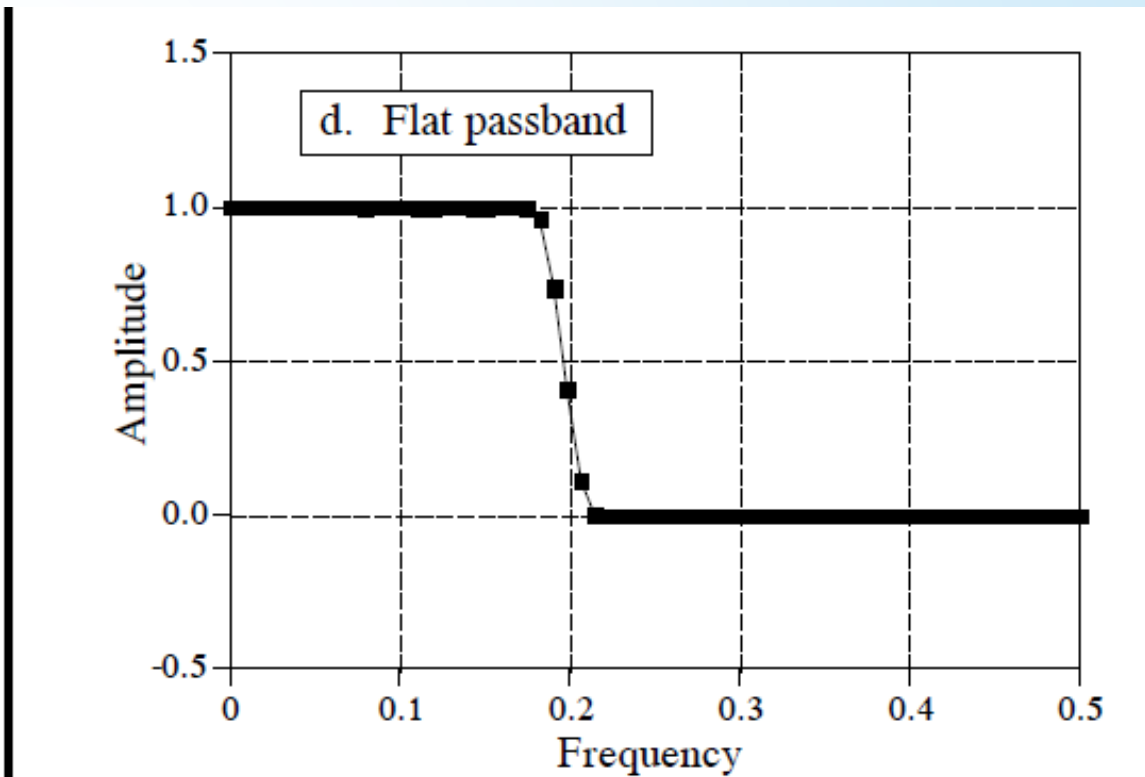
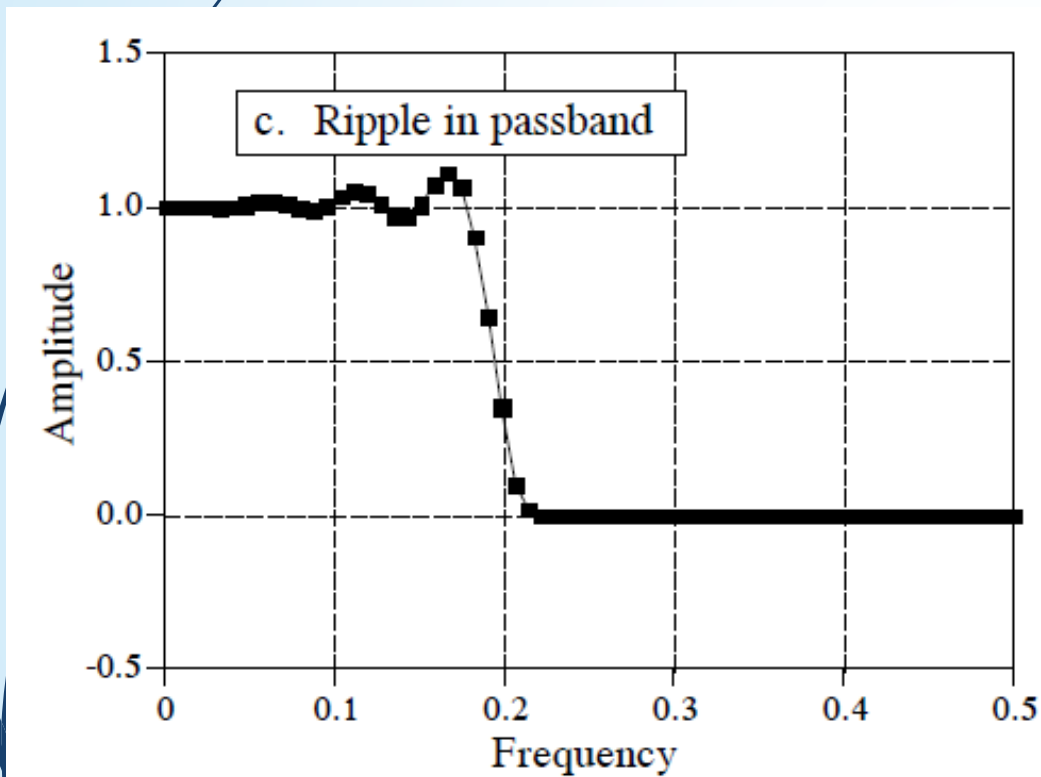
Frequency Domain Parameters

- ➔ **Roll-off** - to separate closely spaced frequencies filter must be **fast roll-off**



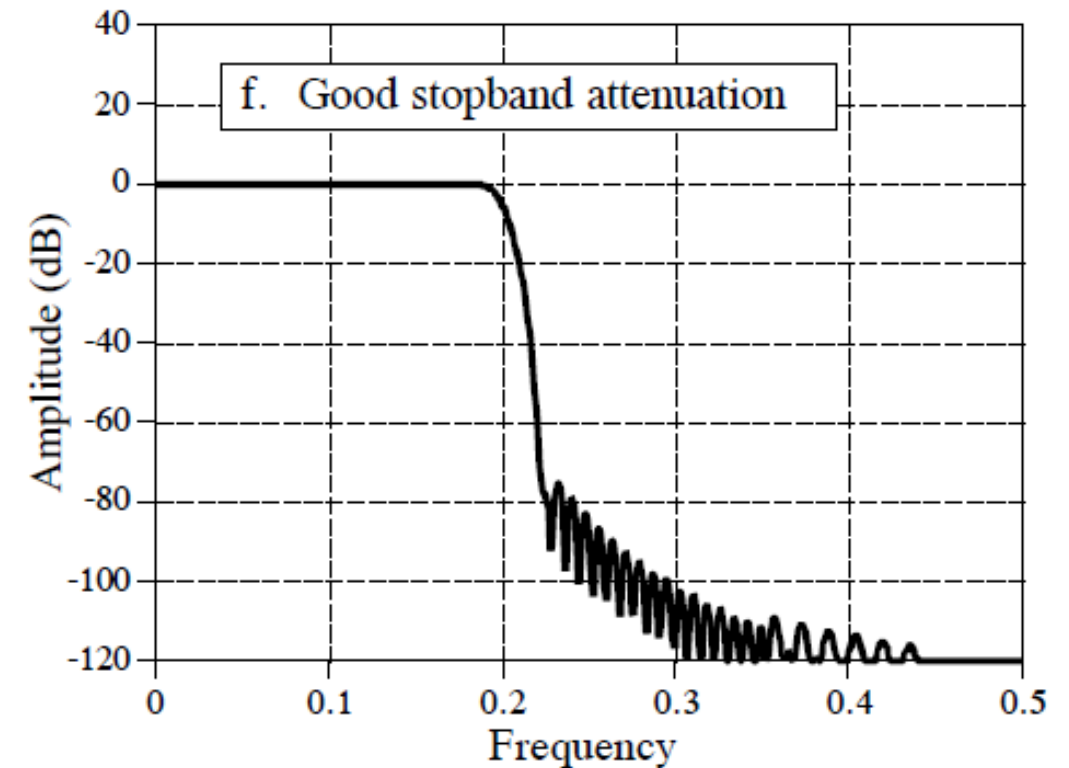
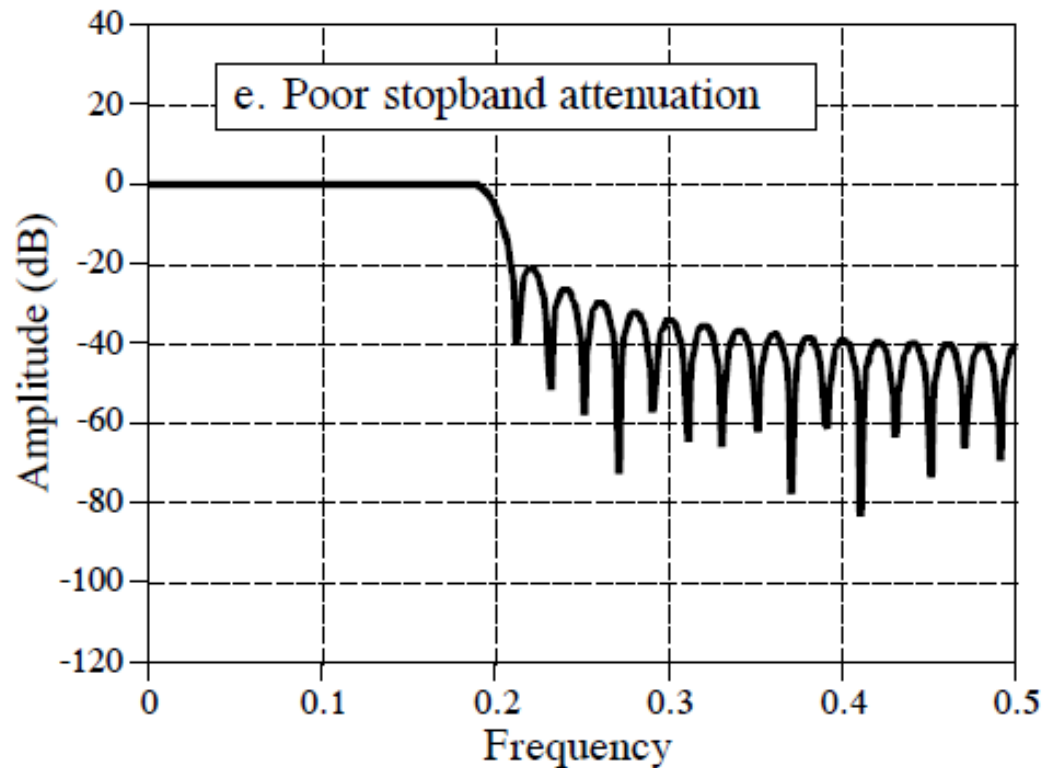
Frequency Domain Parameters

- **Passband ripple** - for the passband frequencies to move through the filter unaltered, there must be no **passband ripple**



Frequency Domain Parameters

- ➔ **Stopband attenuation** - to adequately block the stopband frequencies, it is necessary to have good **stopband attenuation**



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

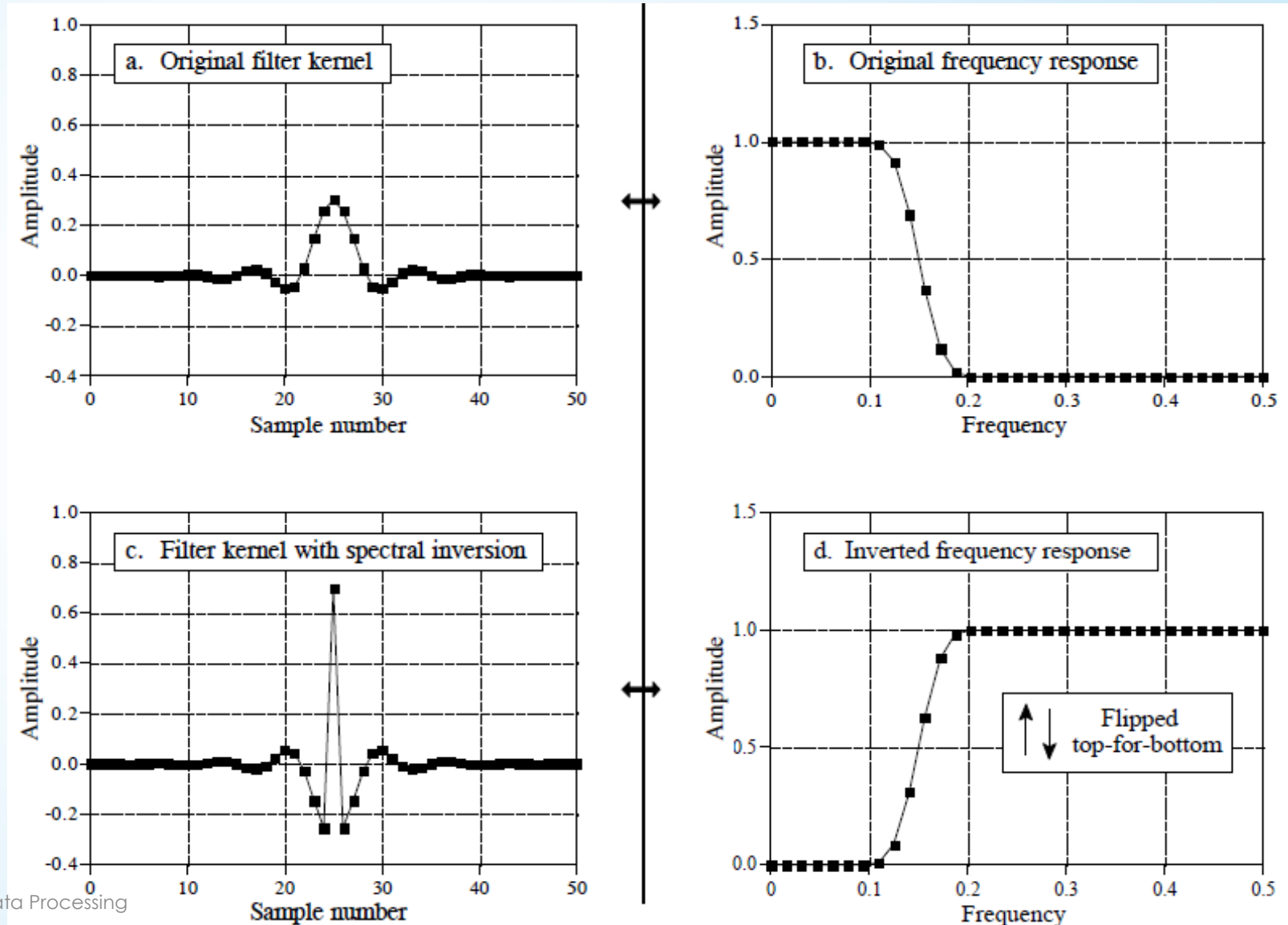
- High-pass, band-pass and band-reject filters are designed by starting with a **low-pass filter**, and then converting it into the desired response.



Spectral inversion method

Fig.(a) shows a lowpass filter kernel called a windowed-sinc. The corresponding frequency response is shown in (b), found by taking a 64 point FFT. Two things must be done to change the low-pass filter kernel into a high-pass filter kernel.

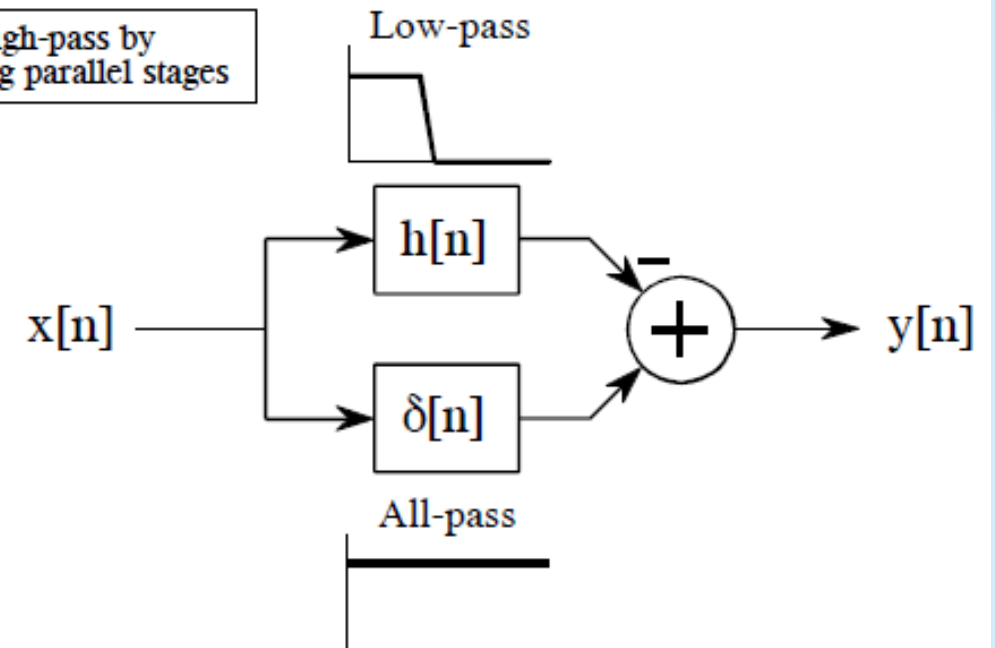
1. change the sign of each sample in the filter kernel.
 2. add one to the sample at the center of symmetry.
- This results in the high-pass filter kernel shown in (c), with the frequency response shown in (d).



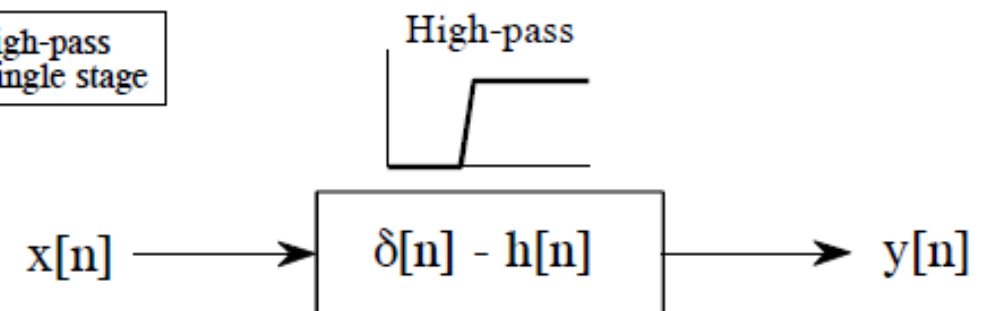
Spectral inversion method

- **Spectral inversion** *flips* the frequency response *top-for-bottom*, changing the passbands into stopbands, and the stopbands into passbands. In other words, it 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.

a. High-pass by adding parallel stages



b. High-pass in a single stage



Spectral inversion method

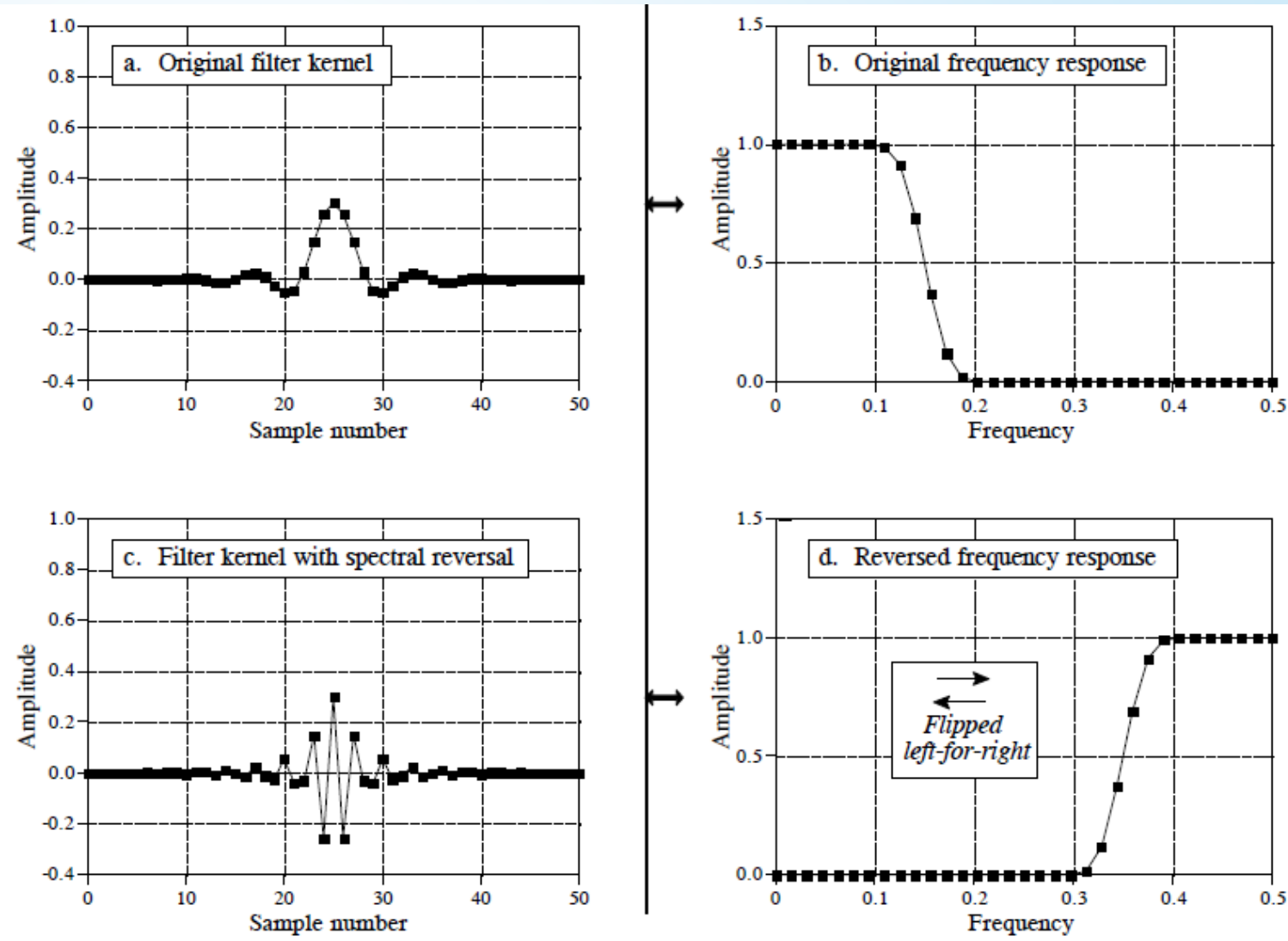
- For this technique to work, the **low-frequency components exiting the low-pass filter must have the same phase as the low-frequency components exiting the all-pass system.** Otherwise a complete subtraction cannot take place.

This places **two restrictions** on the method:

1. the original filter kernel must have left-right symmetry (i.e., a zero or linear phase)
2. the impulse must be added at the center of symmetry.

Spectral Reversal method

The low-pass filter kernel in (a) corresponds to the frequency response in (b). The high-pass filter kernel, (c), is formed by **changing the sign of every other sample** in (a). As shown in (d), this flips the frequency domain *left-for-right*: 0 becomes 0.5 and 0.5 becomes 0. The cutoff frequency of the example low-pass filter is 0.15, resulting in the cutoff frequency of the high-pass filter being 0.35.

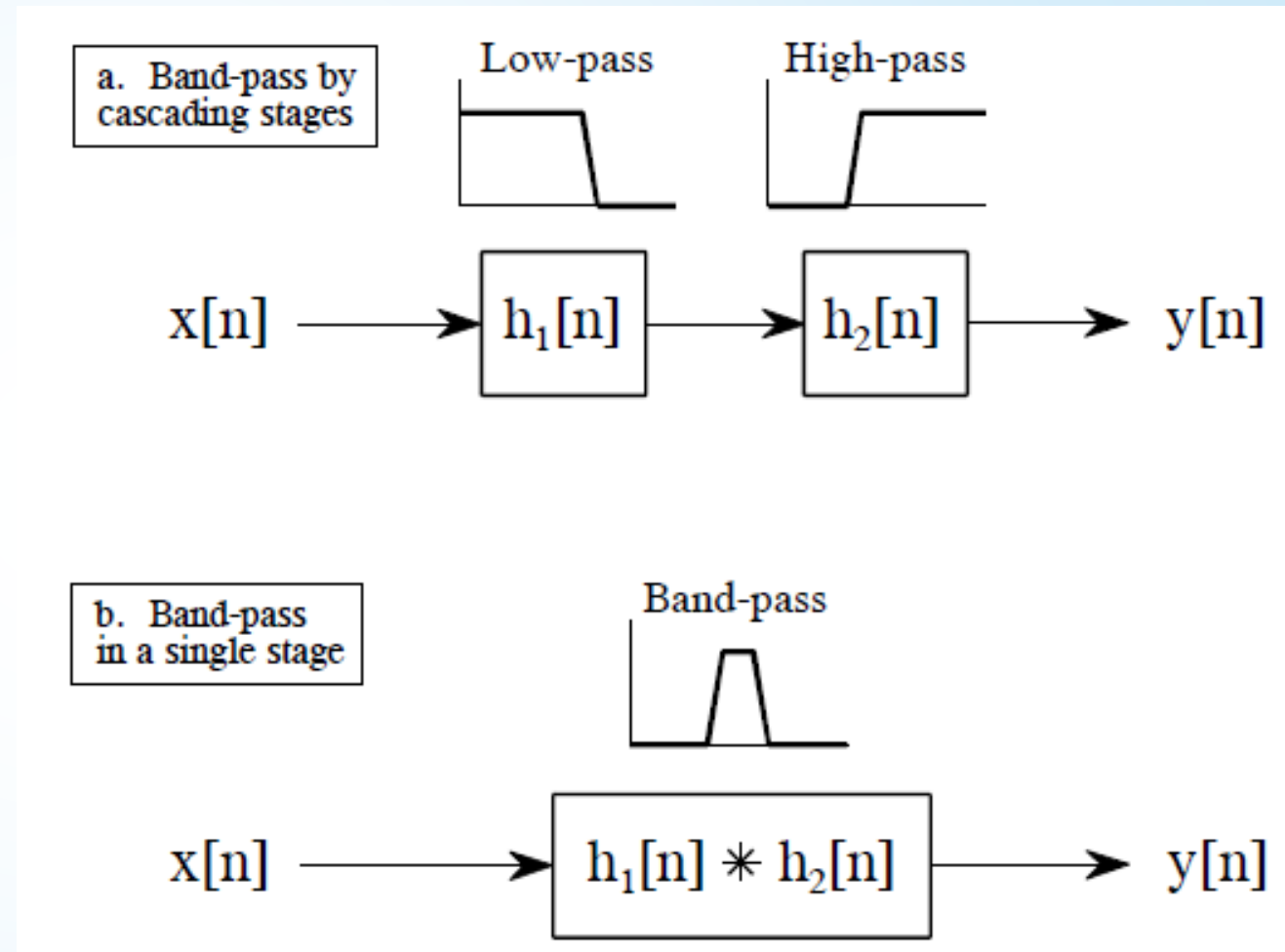


Spectral Reversal method

- 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.
- Look at (b) and imagine the negative frequencies between -0.5 and 0 that are of mirror image of the frequencies between 0 and 0.5. The frequencies that appear in (d) are the negative frequencies from (b) shifted by 0.5.

Band-pass filter design

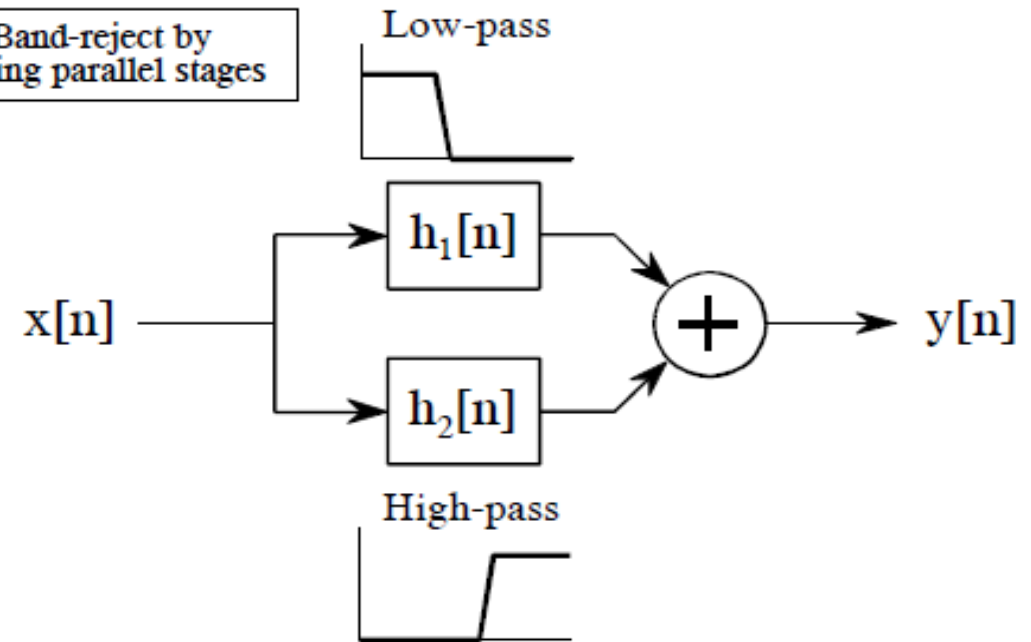
- ❑ As shown in (a), a band-pass filter can be formed by **cascading** a low-pass filter and a high-pass filter.
- ❑ This can be reduced to a single stage, shown in (b). The filter kernel of the single stage is equal to the **convolution** of the low-pass and highpass filter kernels.



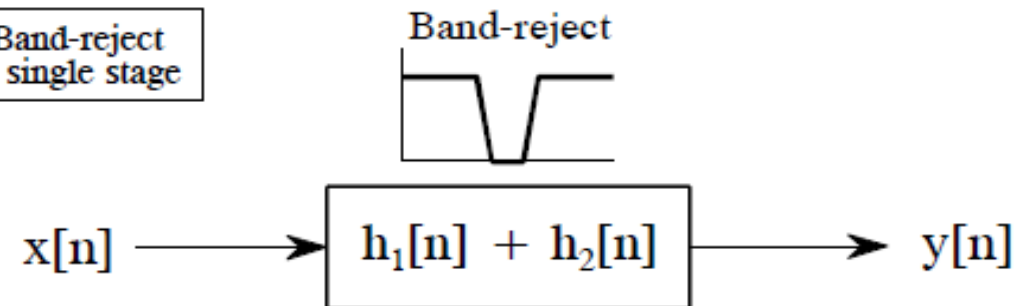
Band-reject Filter design

- ❑ As shown in (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.
- ❑ Figure (b) shows this reduced to a single stage, with the filter kernel found by *adding* the low-pass and high-pass filter kernels.

a. Band-reject by adding parallel stages

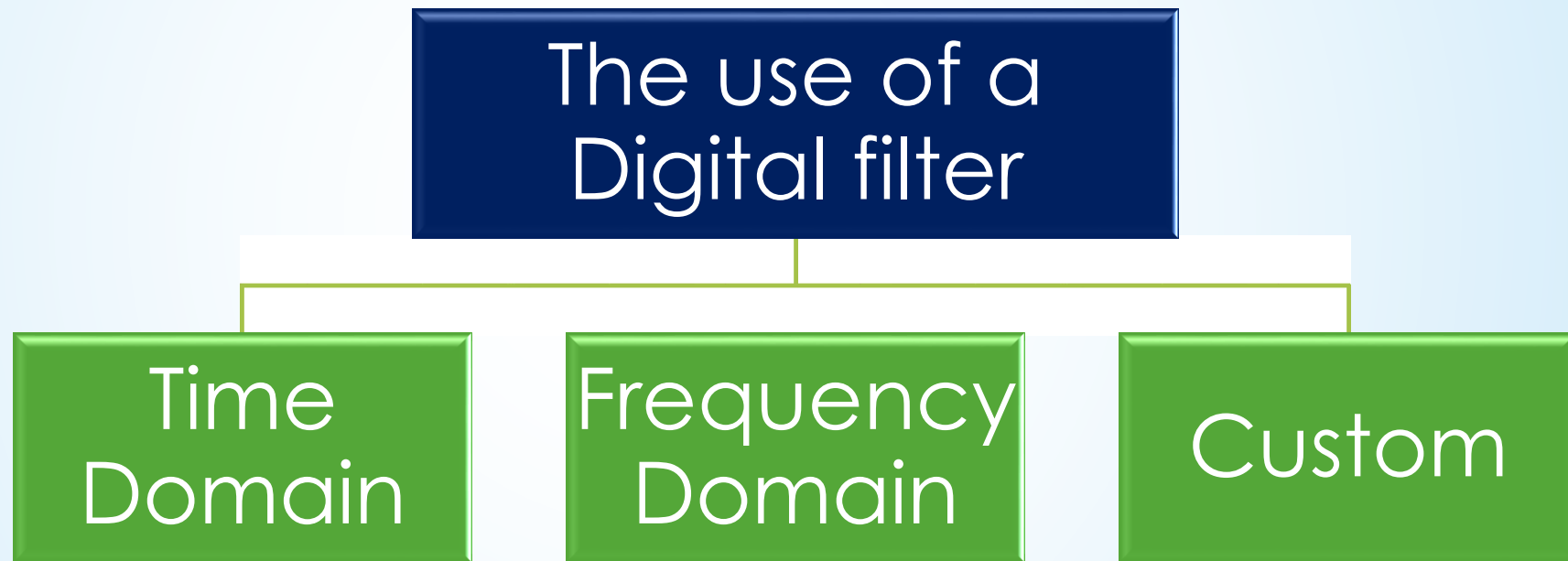


b. Band-reject in a single stage



Filter Classification

- ➡ Digital filters are classified by their **use** and by their **implementation**.



Filter Classification

- **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 elaborate than the four basic responses (high-pass, low-pass, band-pass and band-reject).

Filter Classification

Digital filters can be implemented in two ways, by **convolution** (also called **finite impulse response** or **FIR**) and by **recursion** (also called **infinite impulse response** or **IIR**). Filters carried out by convolution can have far better performance than filters using recursion, but execute much more slowly.

The implementation
of a Digital filter

```
graph TD; A[The implementation of a Digital filter] --> B[Convolution]; A --> C[Recursion];
```

Convolution

Recursion