Course: Digital Signal and Data Processing

# Custom Filters

Lecture 4

**Associate professor Naila Allakhverdiyeva**

# Arbitrary Frequency Response

In the windowed-sinc filter, the frequency response and the filter kernel are both represented by ***equations***, and the conversion between them is made by evaluating the ***mathematics*** of the Fourier transform.

In the method presented here, both signals are represented by ***arrays of numbers***, with a ***computer program*** (the FFT) being used to find one from the other.

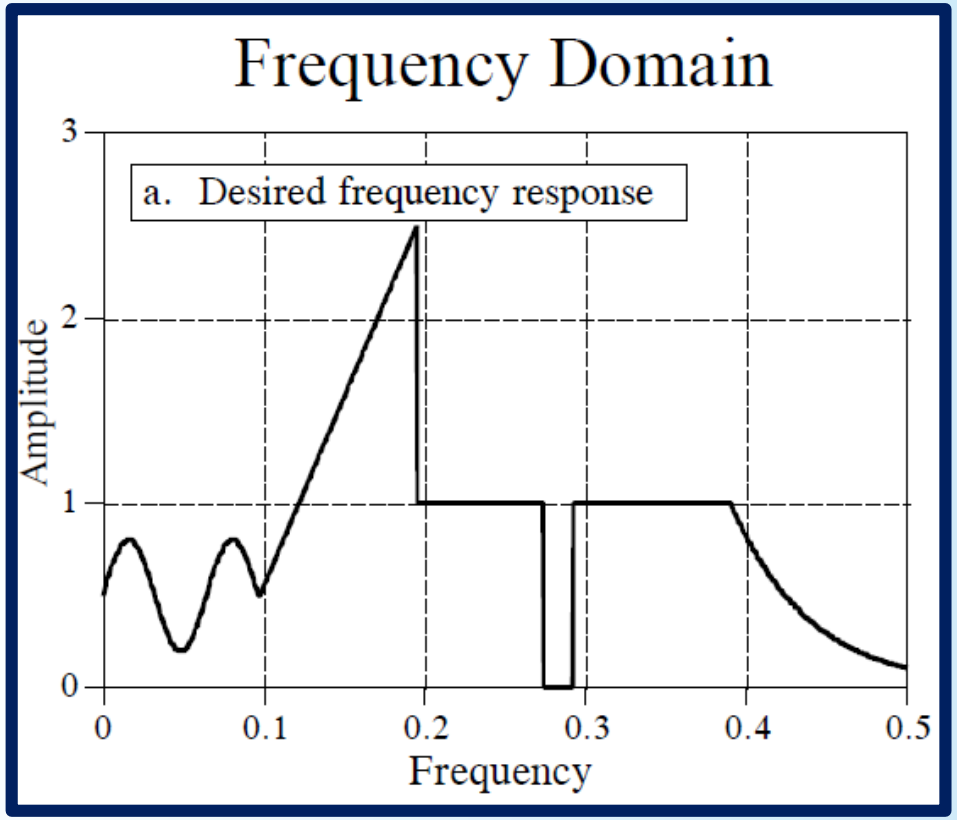
**How the desired response is moved from the frequency domain into the time domain?**

# Arbitrary Frequency Response



The frequency response we want the filter to produce is shown in Fig(a). It is very irregular and would be **virtually impossible to obtain with analog electronics**. This ideal frequency response is ***defined*** by an **array of numbers** that have been selected, not some mathematical equation.

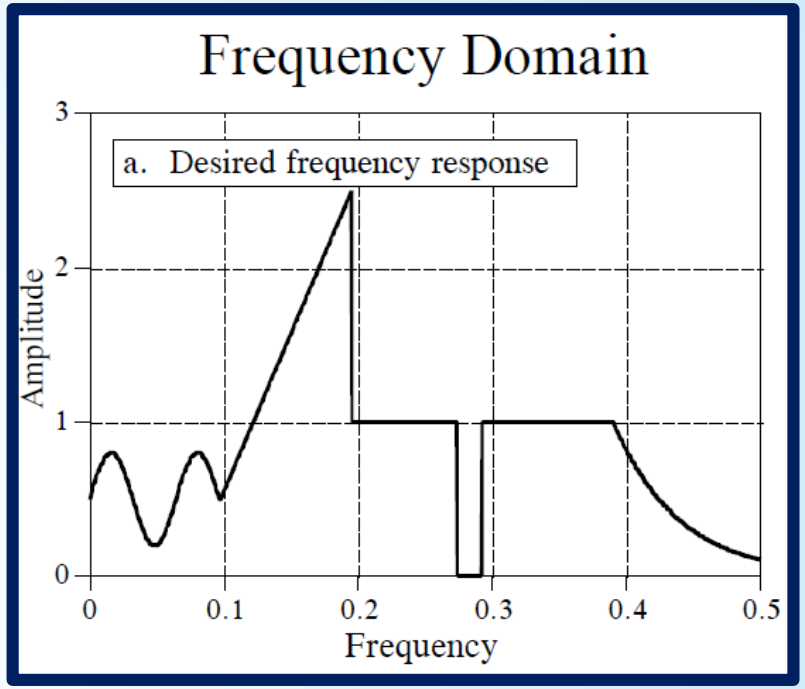
In this example, there are 513 samples spread between 0 and 0.5 of the sampling rate. More points could be used to better represent the desired frequency response, while a smaller number may be needed to reduce the computation time during the filter design.



# Arbitrary Frequency Response



- Besides the **desired *magnitude*** array shown in (a), there must be a corresponding ***phase*** array of the same length. In this example, the phase of the desired frequency response is **entirely *zero.***
- The first and last samples (i.e., 0 and 512) of the phase array must have a value of ***zero*** (or a multiple of $2\pi$).
- The frequency response can also be specified in rectangular form by defining the array entries for the ***real*** and ***imaginary parts***, instead of using the magnitude and phase.

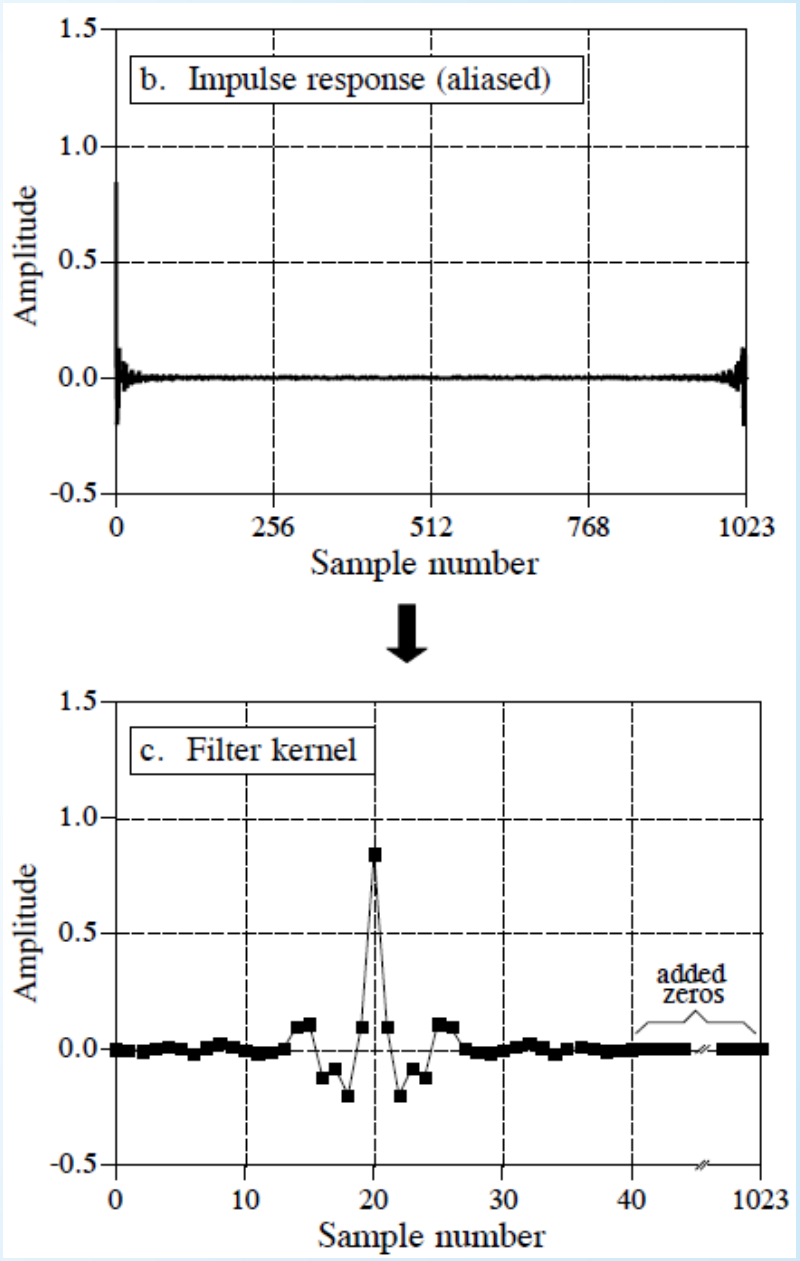


# Arbitrary Frequency Response



The next step is to take the **Inverse DFT** to move the filter into the time domain. The quickest way to do this is to convert the frequency domain to rectangular form, and then use the Inverse FFT. This results in a 1024 sample signal running from 0 to 1023, as shown in (b). This **impulse response** corresponds to the frequency response we want. However, *it is* ***not suitable*** *for use as a filter kernel*. it needs to be ***shifted***, ***truncated***, and ***windowed***. In this example, we will design the filter kernel with $M$=40, i.e., 41 points running from sample 0 to sample 40. Signal in (b) has been converted into the filter kernel shown in (c).

The points near the ends of the filter kernel are so **small** that they appear to be zero when plotted. **Don't make the mistake of thinking they can be deleted!**

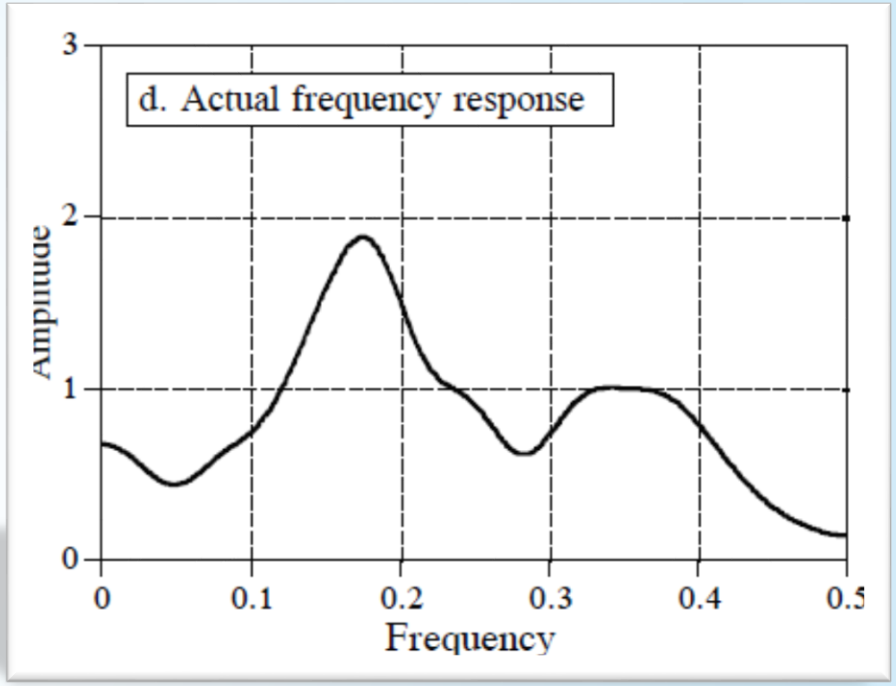


# Arbitrary Frequency Response



The last step is to ***test* the filter kernel**. This is done by taking the DFT (using the FFT) to find the **actual frequency response**, as shown in (d). To obtain better resolution in the frequency domain, pad the filter kernel with zeros before the FFT.

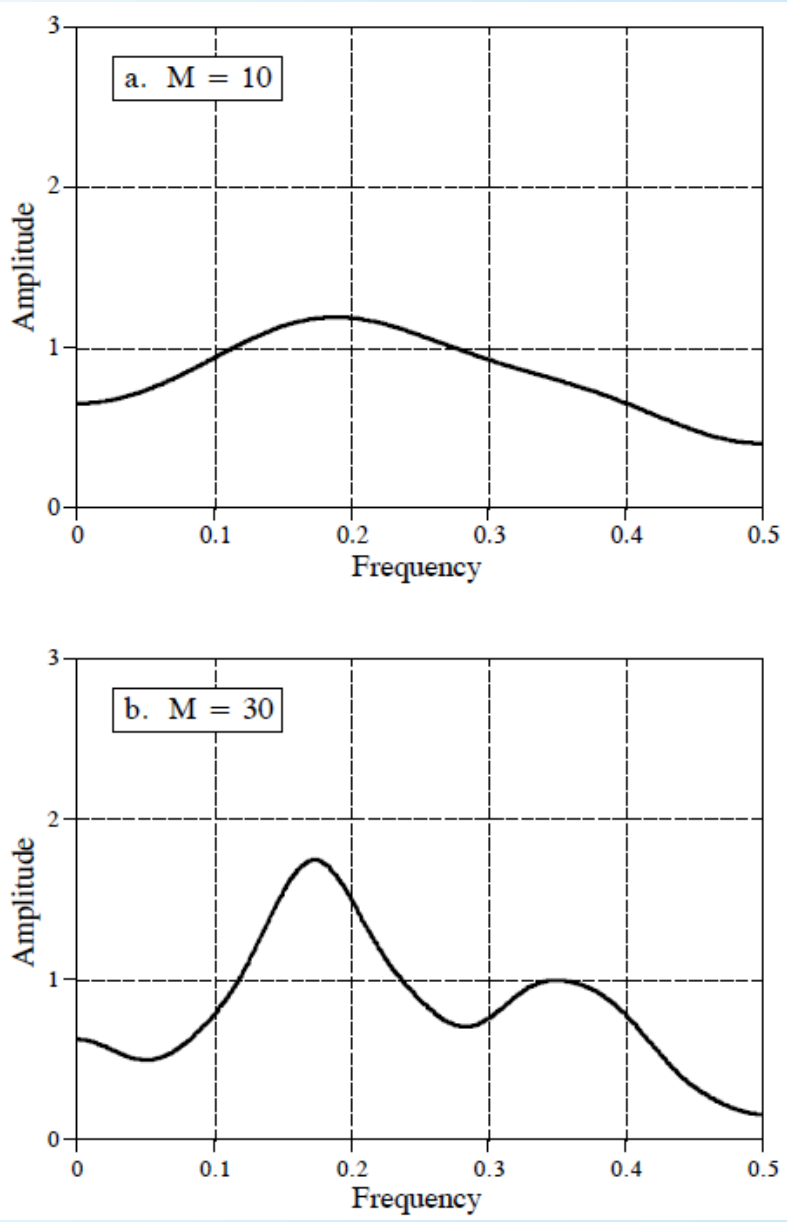
For instance, using 1024 total samples (41 in the filter kernel, plus 983 zeros), results in 513 samples between 0 and 0.5.



# Arbitrary Frequency Response

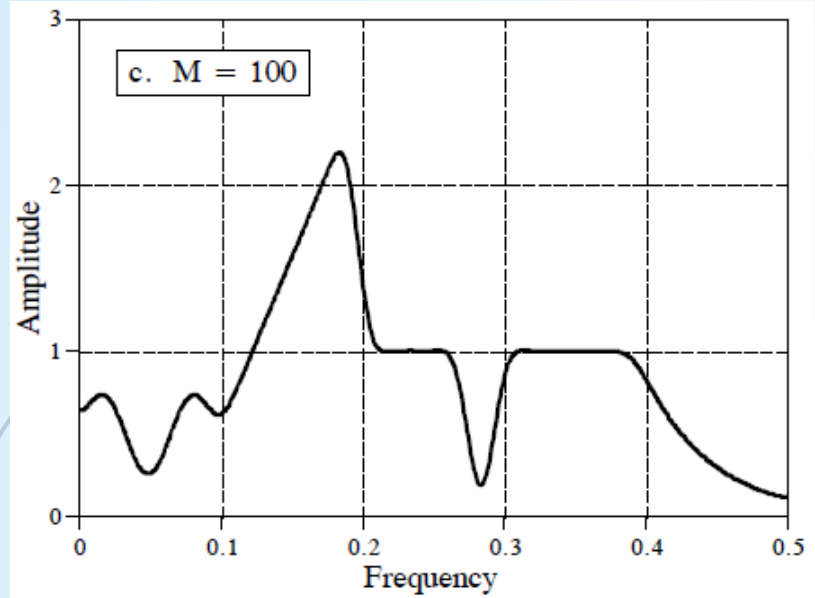
The **length** of the filter kernel determines how well the ***actual* frequency response matches the *desired* frequency response**.

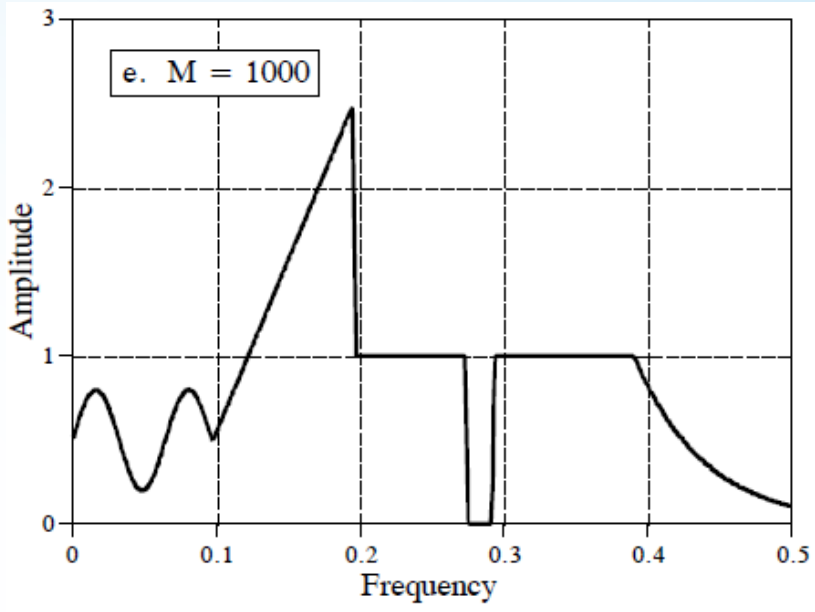
Virtually any frequency response can be obtained if a long enough filter kernel is used.

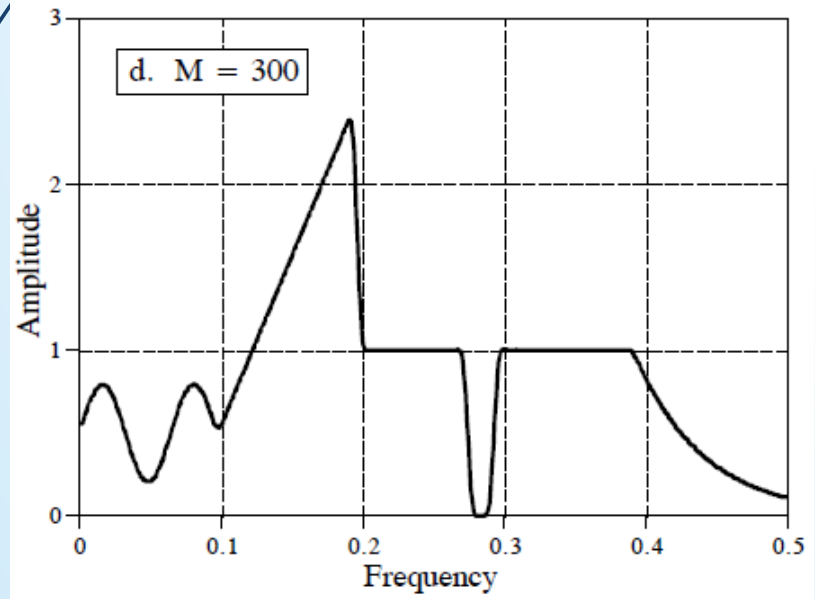


# Arbitrary Frequency Response





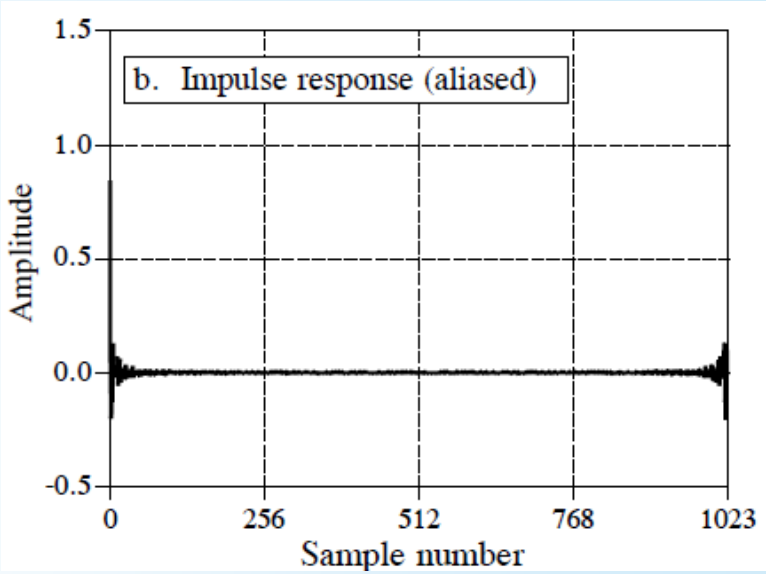




The number of points in each filter kernel is equal to *M*+1, running from 0 to *M*. **As more points are used in the filter kernel, the resulting frequency response more closely matches the desired frequency response.**

# Arbitrary Frequency Response

Why **isn't it possible** to directly use the **original impulse** response shown in (b) as the filter kernel?

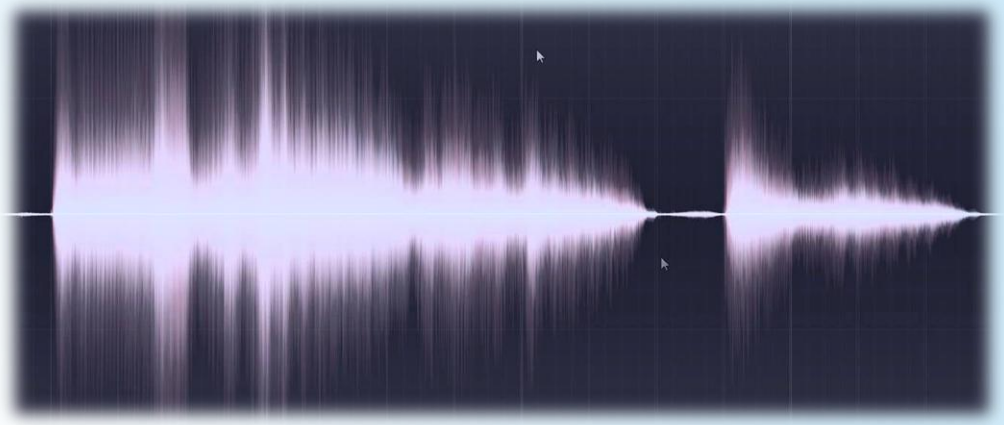
- When designing a custom filter, the desired frequency response is defined by the values in an array. Now consider this: what does the frequency response do ***between* the specified points**? For simplicity, two cases can be imagined, one "good" and one "bad." In the "good" case, the frequency response is a smooth curve between the defined samples. In the "bad" case, there are wild fluctuations between.
- The impulse response in (b) corresponds to the "bad" frequency response. This can be shown by padding it with a large number of zeros, and then taking the DFT.



# Arbitrary Frequency Response

- Imagine that we force the frequency response to be what we want by defining it at an infinite number of points between 0 and 0.5. That is, we create a **continuous curve**. The inverse DTFT is then used to find the impulse response, which will be ***infinite* in length**. In other words, the "good" frequency response corresponds to something that **cannot be represented in a computer**, an infinitely long impulse response.
- When we represent the frequency spectrum with $N/2+1$ samples, only $N$ points are provided in the time domain. The result is that **the infinitely long impulse response wraps up (aliases) into the *N* points**.
- Windowing the $N$ point impulse response greatly **reduces this aliasing**, providing a smooth curve between the frequency domain samples.

# Deconvolution



**Unwanted convolution** is a problem in transferring analog information. For instance, all of the following can be modeled as a convolution: image blurring in a shaky camera, echoes in long distance telephone calls, the finite bandwidth of analog sensors and electronics, etc.

**Deconvolution** is the process of filtering a signal to compensate for an undesired convolution. The goal of deconvolution is to ***recreate the signal as it existed before*** the convolution took place. This usually requires the characteristics of the convolution (i.e., the impulse or frequency response) to be known.

**Blind deconvolution** - the characteristics of the parasitic convolution are ***not* known**.

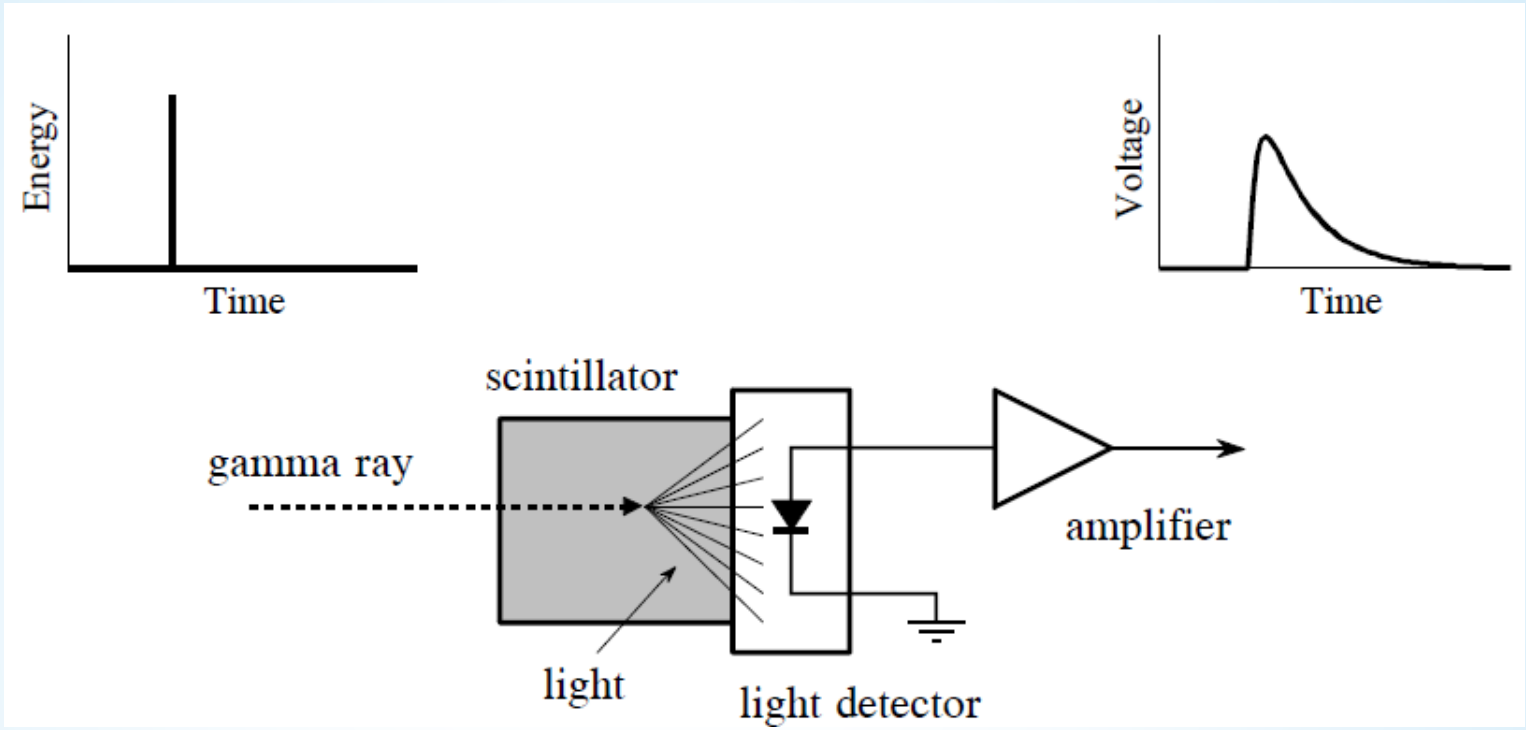
# Deconvolution

Deconvolution is nearly impossible to understand in the ***time domain***, but quite simple in the ***frequency domain***. Each sinusoid that composes the original signal can be changed in amplitude and/or phase as it passes through the undesired convolution. To extract the original signal, the **deconvolution filter must *undo* these amplitude and phase changes**.

For example, if the convolution changes a sinusoid's amplitude by 0.5 with a 30 degree phase shift, the deconvolution filter must amplify the sinusoid by 2.0 with a -30 degree phase change.

# Example -Gamma ray detector

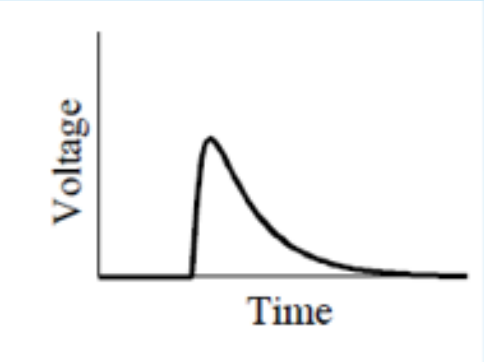





***Gamma ray detector*** can be formed by mounting a ***scintillator*** on a ***light detector***. When a gamma ray strikes the scintillator, its energy is converted into a pulse of light. This pulse of light is then converted into an electronic signal by the light detector. The gamma ray is an ***impulse***, while the output of the detector (i.e., the ***impulse response***) resembles a **one-sided exponential**.

# Example -Gamma ray detector

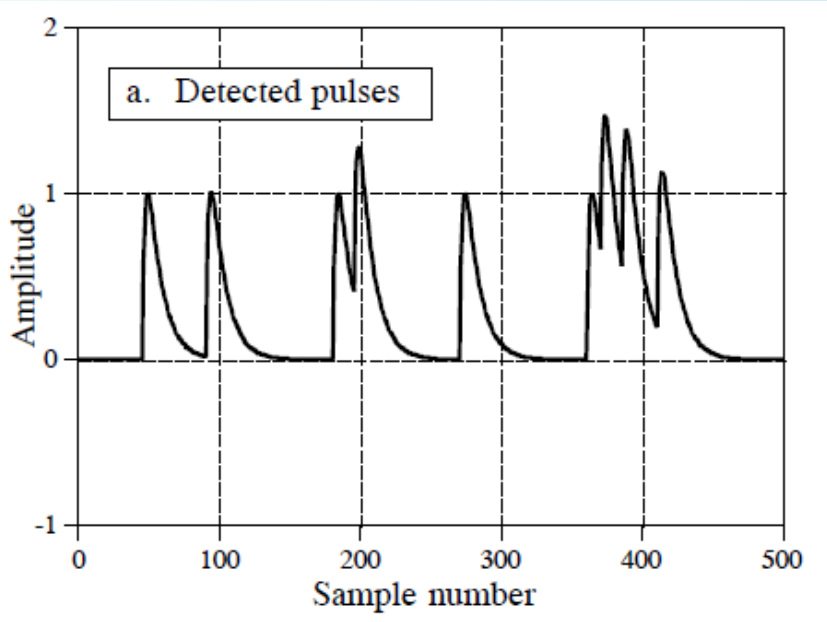
This shape is determined by the characteristics of the scintillator. When a gamma ray settles its energy into the scintillator, nearby atoms are excited to a higher energy level. These atoms randomly *deexcite*, each producing a single photon of visible light. The result is a *light pulse* whose amplitude decays over a few hundred nanoseconds.




Since the arrival of each gamma ray is an ***impulse***, the output pulse from the detector (i.e., the one-sided exponential) is the ***impulse response*** of the system.

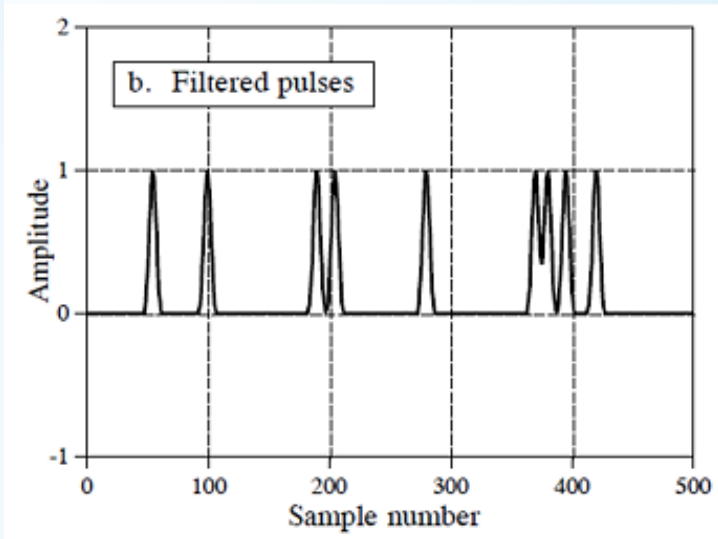
# Example -Gamma ray detector

Figure (a) shows pulses generated by the detector in response to randomly arriving gamma rays. The information we would like to extract from this output signal is the **amplitude** of each pulse, which is proportional to the *energy* of the gamma ray that generated it. This is **useful information** because the energy can tell interesting things about where the gamma ray has been. For example, it may provide medical information on a patient, tell the age of a distant galaxy, detect a bomb in airline luggage, etc.



# Example -Gamma ray detector

Everything would be fine if only an occasional gamma ray were detected, but this is usually not the case. As shown in (a), two or more pulses may overlap, shifting the measured amplitude. One answer to this problem is to ***deconvolve*** the detector's output signal, making the pulses narrower so that less overlap occurs.




Ideally, we would like each pulse look likes original impulse. This isn't possible and **we must settle for a pulse that is finite in length, but significantly shorter than the detected pulse**. This goal is illustrated in Figure (b).

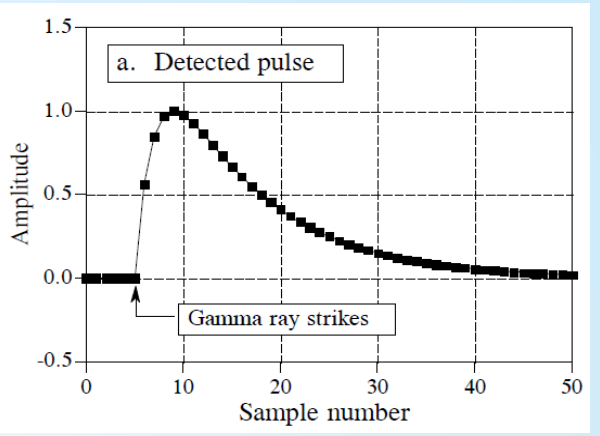
# Example -Gamma ray detector

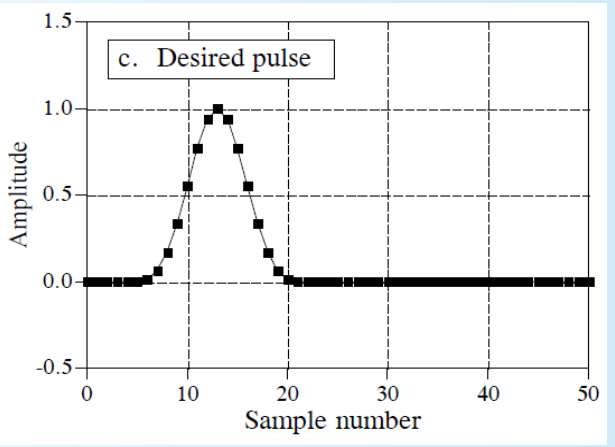


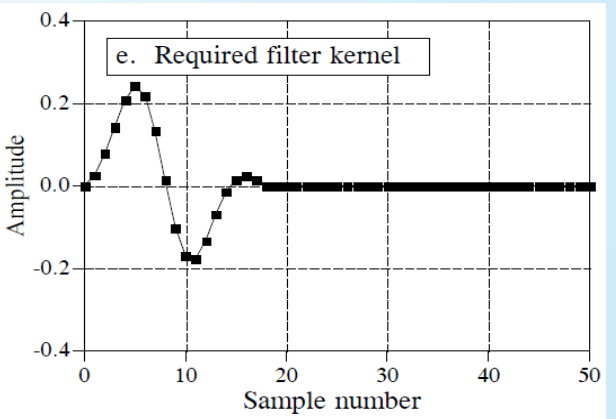
Even though the detector signal has its information encoded in the ***time domain***, much of our analysis must be done in the ***frequency domain***, where the problem is easier to understand.

Figure (a) is the signal produced by the detector (something we know). Figure (c) is the signal we wish to have (also something we know). This desired pulse was arbitrarily selected to be the same shape as a Blackman window, with a length about one-third that of the original pulse. **Our goal is to find a filter kernel, (e), that when convolved with the signal in (a), produces the signal in (c)**. In equation form:

$$if\ a * e = c\ ,\ a \text{ and } c \text{ are given. } e - ?$$



# Deconvolution

- If these signals were combined by addition or multiplication instead of convolution, the solution would be easy: *subtraction* is used to "de-add" and *division* is used to "de-multiply.“
- Convolution is different; **there is not a simple inverse operation that can be called "deconvolution."**
- Convolution is too messy to be undone by directly manipulating the time domain signals.

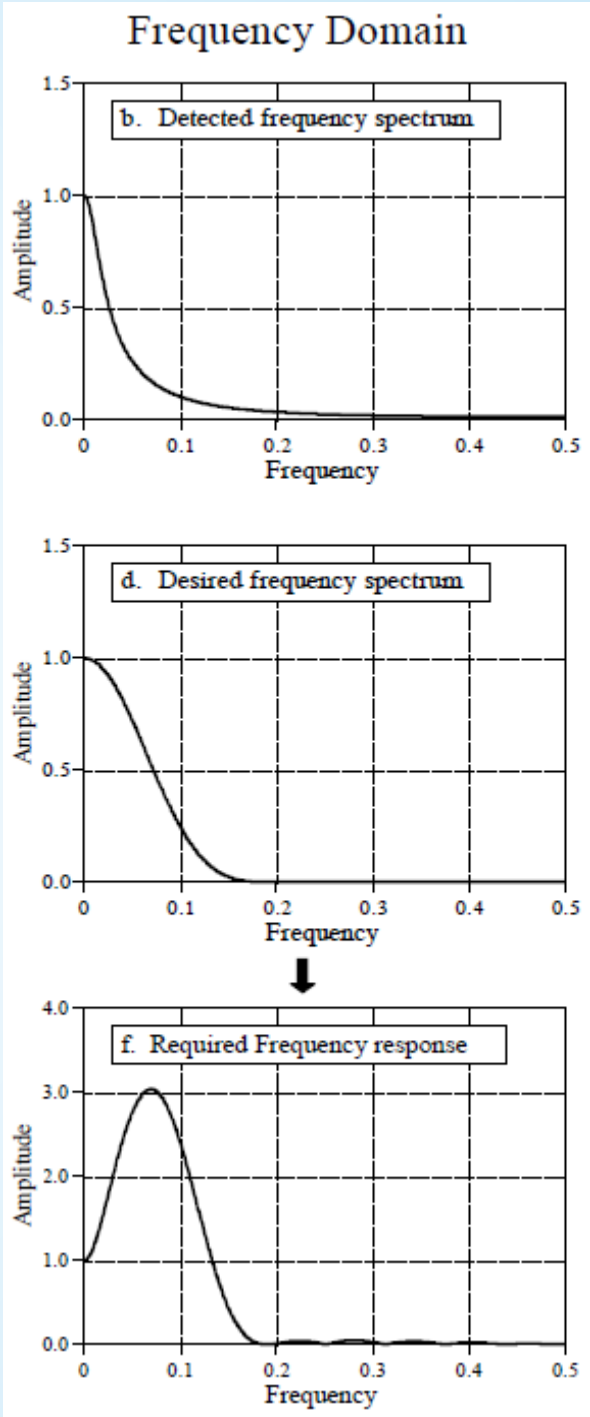
# Deconvolution

This problem is simpler in the **frequency domain**. ***Convolution*** in one domain corresponds ***multiplication*** in the other domain.

**if** $\boldsymbol{b} \cdot \boldsymbol{f} = \boldsymbol{d}$, and given $\boldsymbol{b}$ and $\boldsymbol{d}$, find $\boldsymbol{f}$. This is an easy problem to solve: the frequency response of the filter, ($f$), is the frequency spectrum of the desired pulse, (d), *divided* by the frequency spectrum of the detected pulse, (b). Since the detected pulse is asymmetrical, it will have a ***nonzero* phase.** This means that a *complex* division must be used.

$$MagH[f] = MagY[f] / MagX[f],\ PhaseH[f] = PhaseY[f] - PhaseX[f].$$

The required filter kernel, (e), is then found from the frequency response by the custom filter method (IDFT, shift, truncate, & multiply by a window).

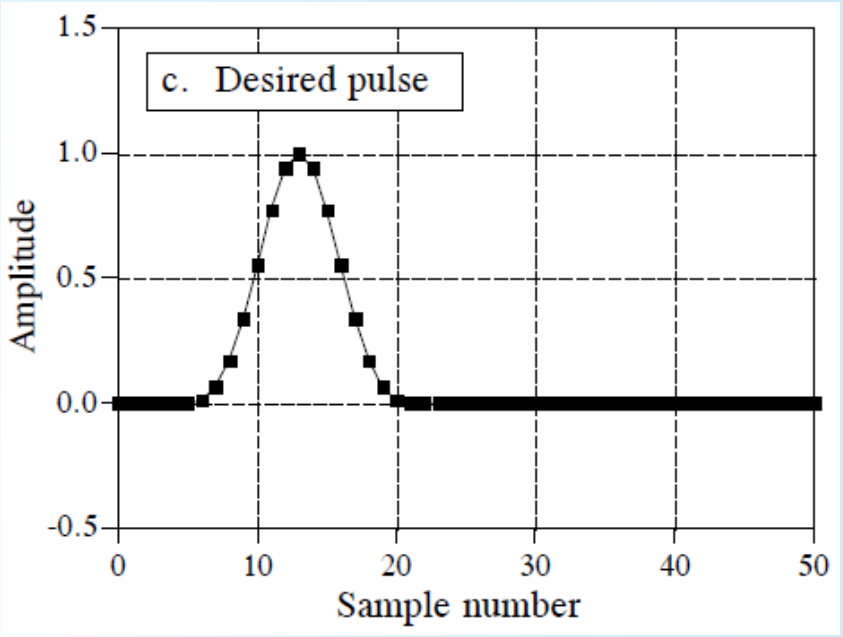


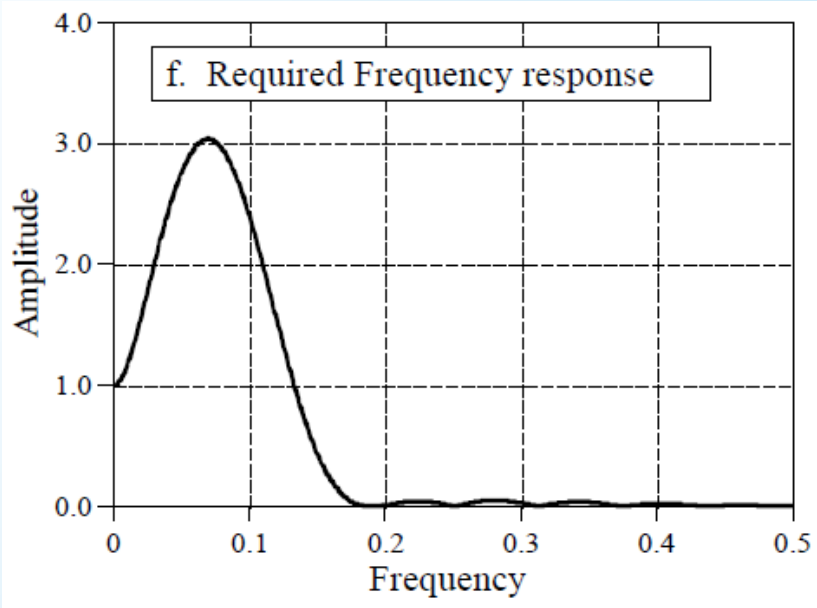
# Deconvolution

There are **limits** to the improvement that deconvolution can provide.

Let's look at what happens if we desired pulse is made narrower. Its frequency spectrum must contain more high frequency components. Since these high frequency components are at a very low amplitude in the detected pulse, the filter must have a very high gain at these frequencies.

For instance, (f) shows that some frequencies must be multiplied by a factor of ***three*** to achieve the desired pulse in (c). **If the desired pulse is made narrower, the gain of the deconvolution filter will be even greater at high frequencies.**



# Deconvolution

Problems of deconvolution:

- Small errors. For instance, if some frequency is amplified by 30, when only 28 is required, the deconvolved signal will probably be a mess.
- There are always unknowns in real world applications, caused by: electronic noise, temperature drift, variation between devices, etc. These unknowns set a limit on how well deconvolution will work.

# Deconvolution

- Factor that limits the performance of deconvolution is ***noise***.
- Most unwanted convolutions take the form of a low-pass filter, reducing the amplitude of the high frequency components in the signal. Deconvolution corrects this by amplifying these frequencies. However, if the amplitude of these components falls below the noise of the system, the information contained in these frequencies is lost. No amount of signal processing can retrieve it.
- As an extreme case, the amplitude of some frequencies may be completely **reduced to *zero***. This not only delete the information, it will try to make the deconvolution filter have ***infinite* gain** at these frequencies.

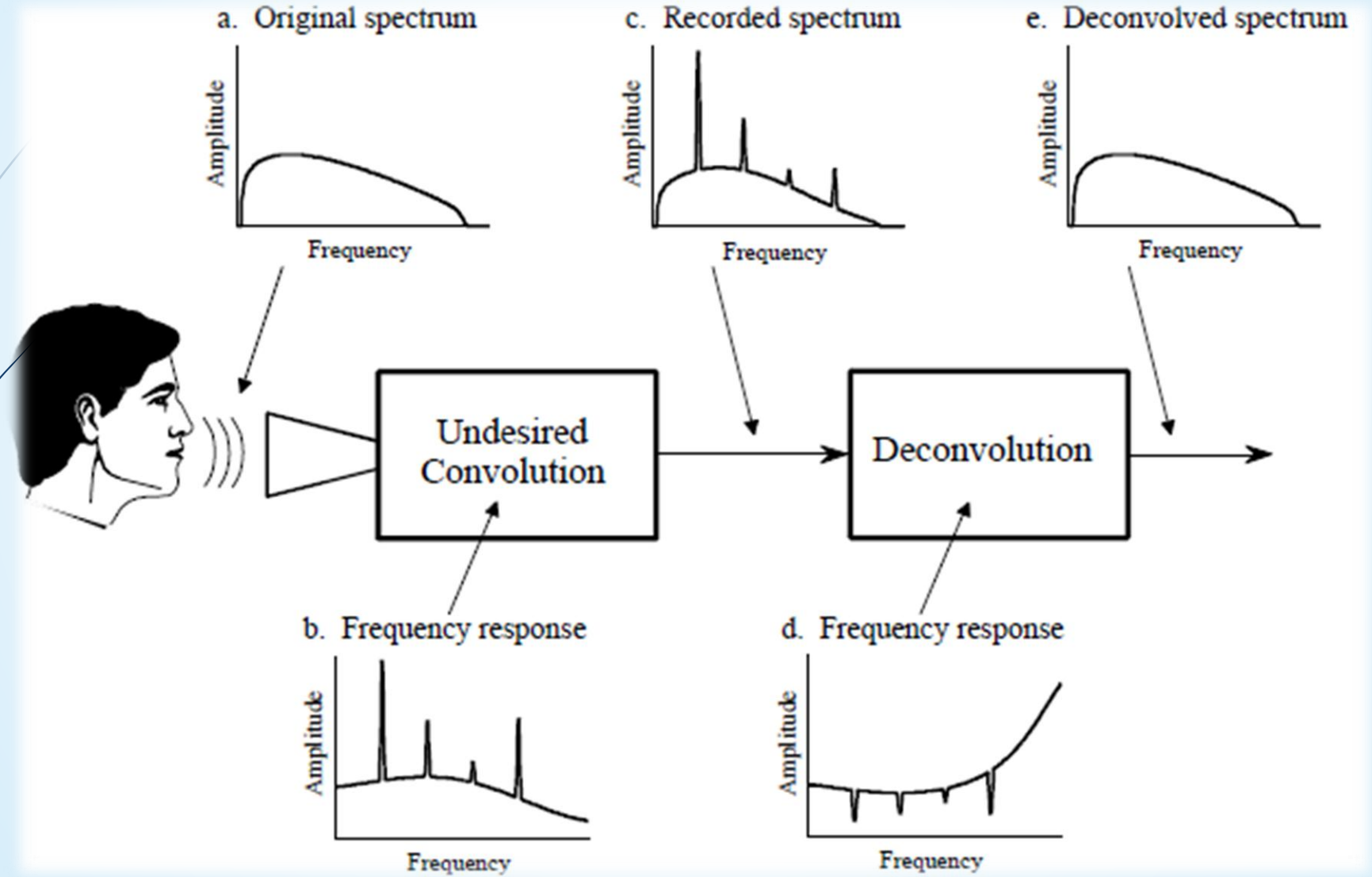
The solution: **design a less aggressive deconvolution filter and/or place limits on how much gain is allowed at any of the frequencies.**

# Blind deconvolution

Deconvolution can also be applied to ***frequency domain*** encoded signals.

A classic example is the **restoration of old recordings** of the famous opera singer, Enrico Caruso (1873-1921). These recordings were made with very primitive equipment by modern standards. The most significant problem is the ***resonances*** of the long tubular recording horn used to gather the sound. Whenever the singer happens to hit one of these resonance frequencies, the loudness of the recording abruptly increases. Digital deconvolution has improved the quality of these recordings by **reducing the loud spots** in the music.

# General approach of deconvolution



# Blind deconvolution problem

Previous Figure shows the general approach. The frequency spectrum of the original audio signal is illustrated in (a). Figure (b) shows the frequency response of the recording equipment, a relatively smooth curve except for several sharp resonance peaks. The spectrum of the recorded signal, shown in (c), is equal to the true spectrum, (a), multiplied by the uneven frequency response, (b).

The goal of the deconvolution is to ***neutralize*** **the undesired convolution**. In other words, the frequency response of the deconvolution filter, (d), must be the ***inverse*** of (b). That is, each peak in (b) is cancelled by a corresponding dip in (d). If this filter were perfectly designed, the resulting signal would have a spectrum, (e), identical to that of the original.

Here's the catch: the original recording equipment has long been discarded, and its frequency response, (b), is a mystery. In other words, this is a ***blind deconvolution*** problem; given only (c), how can we determine (d)?

# Blind deconvolution



Blind deconvolution problems are usually attacked by making an estimate or assumption about the unknown parameters. To deal with this example, the ***average spectrum*** of the original music is assumed to match the ***average spectrum*** of the same music performed by a present day singer using modern equipment.

The ***average spectrum*** is found by the following techniques:

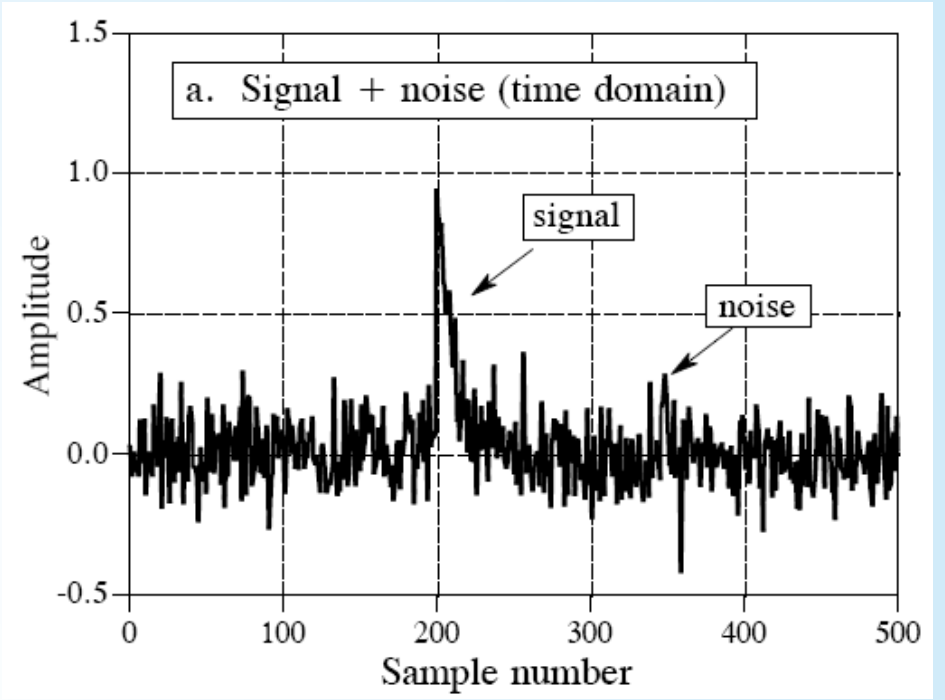
- break the signal into a large number of segments;
- take the DFT of each segment;
- convert into polar form;
- average the magnitudes together.

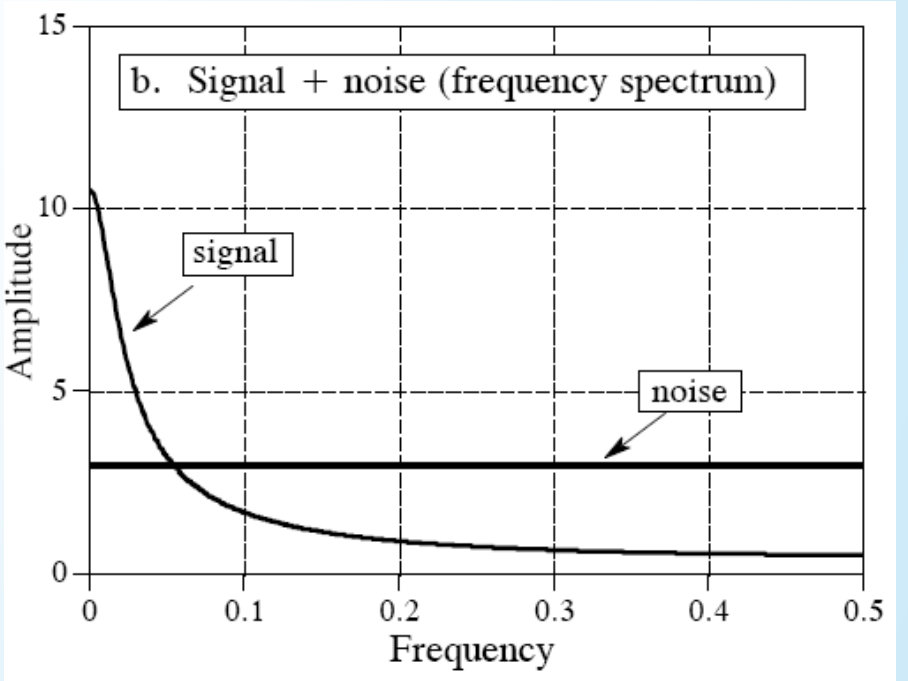
In the simplest case, the unknown frequency response is taken as the average spectrum of the old recording, divided by the average spectrum of the modern recording. The method is called ***homomorphic* processing -** provide a better estimate of the characteristics of the recording system.

# Optimal Filters

Figure (a) illustrates a common filtering problem: trying to extract a waveform (in this example, an exponential pulse) buried in random noise. As shown in (b), this problem is no easier in the frequency domain.

The signal has a spectrum composed mainly of **low frequency components**. In comparison, the spectrum of the noise is ***white*** (the same amplitude at all frequencies). Since **the spectra of the signal and noise *overlap***, it is not clear how the two can best be separated. We will look at three filters, each of which is "best" (**optimal**) in a different way.



# Optimal Filters: Moving Average Filter

In **moving average filter**, each output point is the average of a certain number of points from the input signal. This makes the filter kernel a rectangular pulse with an amplitude equal to the reverse of the number of points in the average.

The **moving average filter is optimal in the sense that it provides the fastest step response for a given noise reduction**.

# Optimal Filters: Matched Filter

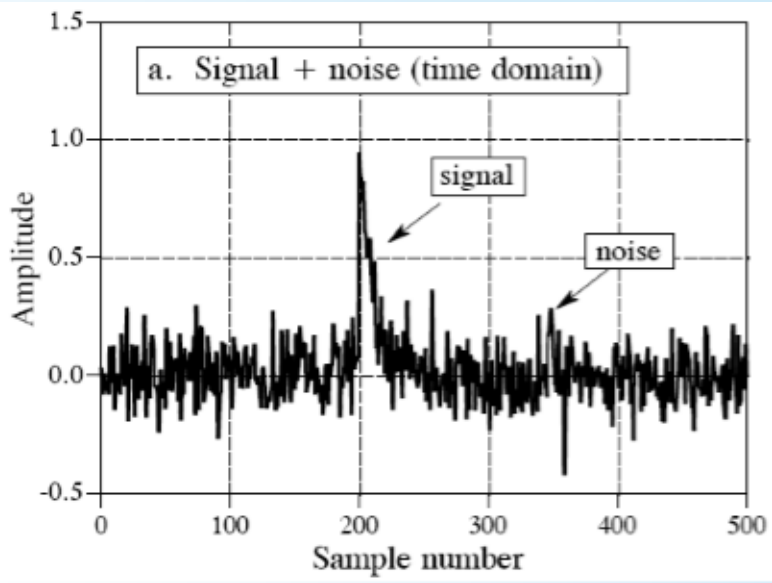
The **matched filter** kernel is the same as the target signal being detected, except it has been flipped left-for-right. The idea behind the matched filter is *correlation*, and this flip is required to perform ***correlation using convolution***. The amplitude of each point in the output signal is a measure of how well the filter kernel *matches* the corresponding section of the input signal. The output of a matched filter does not necessarily look like the signal being detected. This doesn't really matter; if a matched filter is used, the **shape of the target signal must already be known**.

The matched filter is optimal in the sense that the top of the peak is farther above the noise than can be achieved with any other linear filter.

# Optimal Filters: Wiener Filter

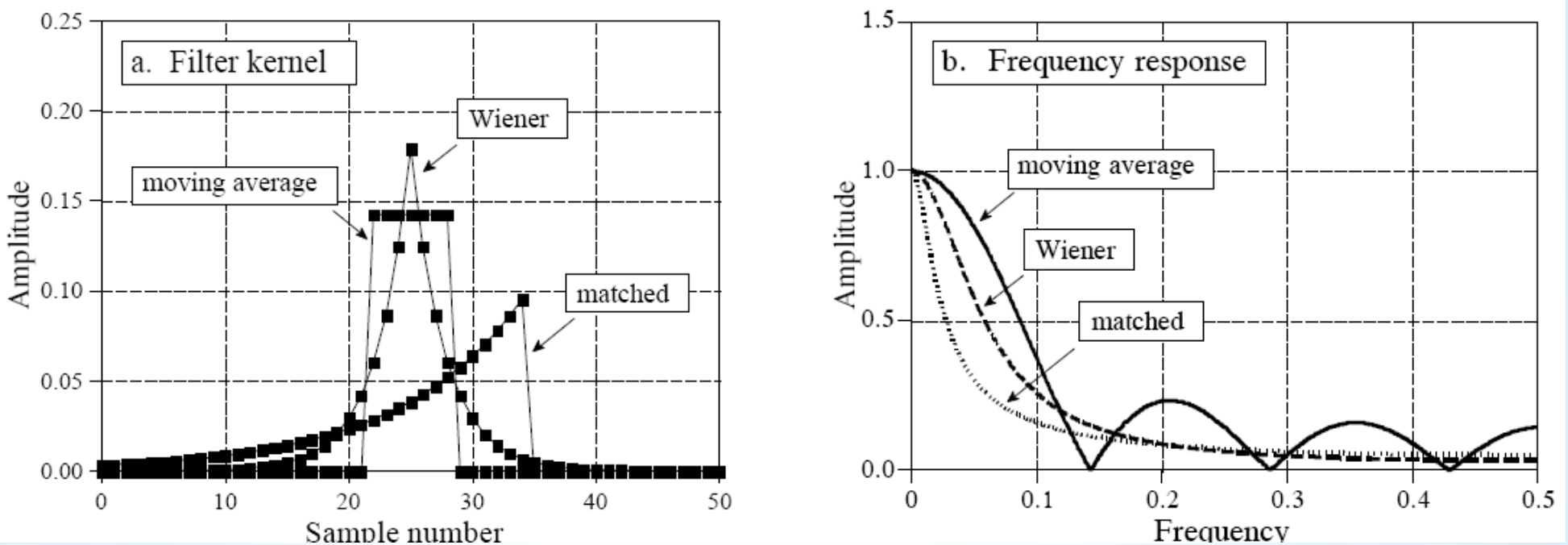
The **Wiener filter** (named after the optimal estimation theory of Norbert Wiener) separates signals based on their frequency spectra. As shown in Figure, at some frequencies there is mostly signal, while at others there is mostly noise. It seems logical that the "mostly signal" frequencies should be passed through the filter, while the "mostly noise" frequencies should be blocked. The Wiener filter takes this idea a step further; *the gain of the filter at each frequency is determined by the relative amount of signal and noise at that frequency*.

The Wiener filter is optimal in the sense that it maximizes the ratio of the signal power to the noise power (over the length of the signal, not at each individual point).



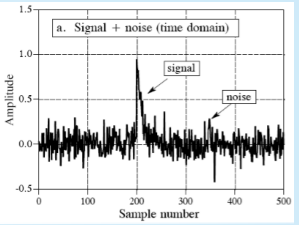

$$H[f] = \frac{S[f]^2}{S[f]^2 + N[f]^2}$$

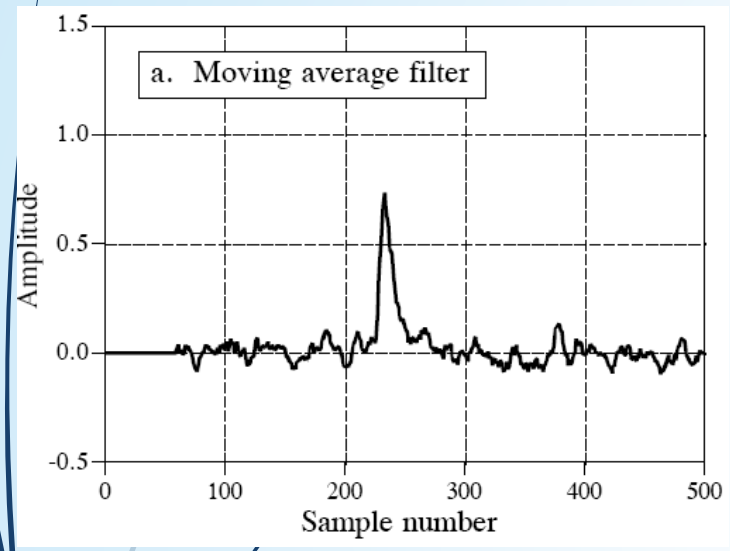
# Example of optimal Filters

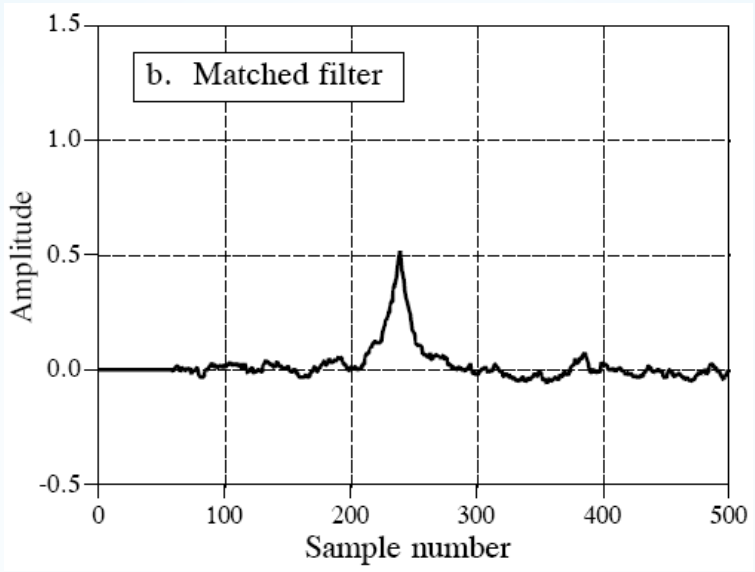


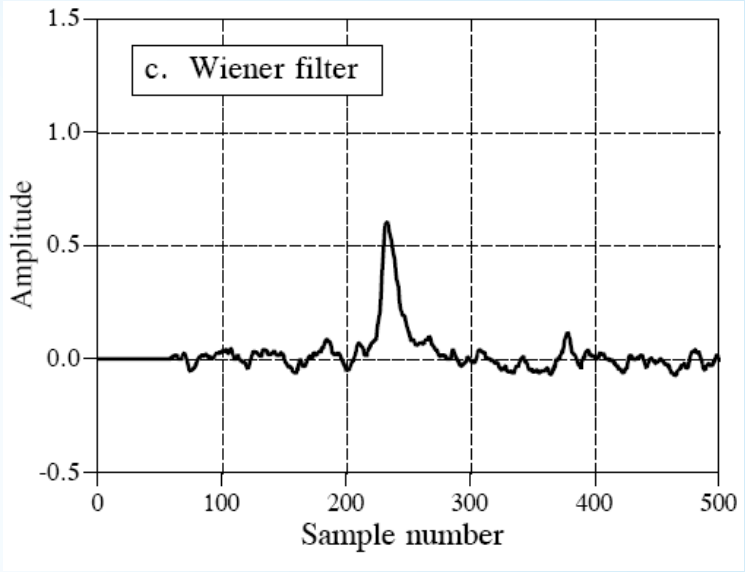

In (a), three filter kernels are shown, each of which is optimal in some sense. The corresponding frequency responses are shown in (b). The moving average filter is designed to have a rectangular pulse for a filter kernel. In comparison, the filter kernel of the matched filter looks like the signal being detected. The Wiener filter is designed in the frequency domain, based on the relative amounts of signal and noise present at each frequency.

# Example of using Optimal Filters




These signals result from filtering the waveform with the optimal filters. Each of these three filters is optimal ***in some sense***. In (a), the moving average filter results in the sharpest edge response for a given level of random noise reduction. In (b), the matched filter produces a peak that is farther above the residue noise than provided by any other filter. In (c), the Wiener filter optimizes the signal-to-noise ratio.